



# WebMedia2023

## Minicursos | XXIX Simpósio Brasileiro de Sistemas Multimídia e Web

23 a 27 de outubro de 2023  
Ribeirão Preto | SP | Brasil

### ORGANIZADORES

Manoel Carvalho Marques Neto (IFBA), Maria da Graça Pimentel (ICMC/USP),  
Roberto Willrich (UFSC), Alessandra Alaniz Macedo (FFCLRP/USP) e  
Rudinei Goularte (ICMC/USP)

#### PATROCÍNIO



#### COOPERAÇÃO



#### ORGANIZAÇÃO



#### REALIZAÇÃO





# XXIX Simpósio Brasileiro de Sistemas Multimídia e Web

De 23 a 27 de outubro de 2023  
Ribeirão Preto, Brasil

## LIVRO DE MINICURSOS

### Organizadores

Manoel Carvalho Marques Neto (IFBA)  
Maria da Graça Pimentel (ICMC/USP)  
Roberto Willrich (UFSC)  
Alessandra Alaniz Macedo (FFCLRP-USP)  
Rudinei Goularte (ICMC-USP)

### Realização

Sociedade Brasileira de Computação – SBC

### Em cooperação com

ACM/SIGWEB e ACM/SIGMM

### Organização

Faculdade de Filosofia, Ciências e Letras de Ribeirão Preto – FFCLRP-USP  
Instituto de Ciências Matemáticas e de Computação – ICMC-USP

#### Dados Internacionais de Catalogação na Publicação

---

Simpósio Brasileiro de Sistemas Multimídia e Web (29. : 23-27 nov. 2023 : Ribeirão Preto – SP)

Minicursos do XXIX Simpósio Brasileiro de Sistemas Multimídia e Web (WebMedia) [recurso eletrônico] / organização Manoel Carvalho Marques Neto... [et al.]. – Ribeirão Preto, SP : Faculdade de Filosofia, Ciências e Letras de Ribeirão Preto: Sociedade Brasileira de Computação, 2024.

1 recurso eletrônico (171 p.): il. PDF; 8,6MB

ISBN 978-85-7669-582-0

Inclui bibliografias.

Modo de acesso: World Wide Web

Título extraído da tela de título (visualizado em 24 abr. 2024)

1. Sistemas multimídia - Congressos. 2. World Wide Web (Sistema de recuperação da informação). 3. Redes sociais on-line. 4. Multimídia interativa. 5. Hipermídia. I. Manoel Carvalho Marques Neto. II. Faculdade de Filosofia, Ciências e Letras de Ribeirão Preto. III. Sociedade Brasileira de Computação. IV. Título.

CDD: ed. 29 – 006.7

---

## Prefácio

Em 2023, como em anos anteriores, o Simpósio Brasileiro de Sistemas Multimídia e Web proporcionou aos participantes a oportunidade de explorar e aprender sobre temas relevantes da área na trilha *Minicursos e Tutoriais*. Nesta edição foi realizada uma consulta aos alunos de graduação e pós-graduação da região de Ribeirão Preto e São Carlos sobre temas e tecnologias que eles teriam interesse em aprender. Como resultado da consulta, a chamada para Minicursos e Tutoriais focou em temas relacionados ao desenvolvimento de aplicações Web. A chamada, divulgada nacionalmente, atraiu pesquisadores e professores com experiência nos temas. Como resultado da avaliação, foram aceitos para apresentação durante o evento dois minicursos de oito horas, um minicurso de seis horas e dois tutoriais de duas horas. Tradicionalmente, os ministrantes dos minicursos produzem um texto correspondente ao minicurso para publicação na forma de capítulo de livro. Este livro publica, a posteriori, o material correspondente a dois minicursos e aos dois tutoriais.

O Capítulo 1, intitulado *Como testar a acessibilidade em soluções mobile*, aborda o problema de que, no cenário atual, os aplicativos nativos para Android continuam a apresentar obstáculos significativos para os usuários com deficiência, evidenciando a necessidade urgente de aprimorar sua acessibilidade. Estudos revelam que os desenvolvedores reconhecem a importância desse aspecto, embora muitos enfrentem desafios na implementação devido à falta de orientação prática, especificações claras de requisitos e definições de responsabilidades. O capítulo aborda essas questões, capacitando profissionais de desenvolvimento, design e teste a lidar com as complexidades da acessibilidade em soluções móveis. São exploradas estratégias e ferramentas para testar e melhorar a acessibilidade em aplicativos nativos para Android.

O Capítulo 2, *Desenvolvendo Aplicativos Android usando Kotlin*, apresenta os conceitos essenciais e atividades práticas para criar software nativo para dispositivos Android. São abordados desde a criação de um aplicativo *Hello World* até tópicos avançados, como o gerenciamento do ciclo de vida de atividades e fragmentos. Ao final, os leitores estarão aptos a desenvolver aplicativos Android com uma arquitetura simplificada e interface de usuário eficiente.

O Capítulo 3, *Desenvolvimento full-stack com JavaScript: uma visão geral e prática*, explora elementos essenciais no desenvolvimento de uma aplicação Web *full-stack* com base em serviços REST. O objetivo é empregar a linguagem JavaScript em ambas as camadas, *front-end* e *back-end*, utilizando frameworks e bibliotecas modernas como o *Express*, para a construção de servidores web, e o *Sequelize*, para mapeamento objeto-relacional. Discutem-se princípios fundamentais, tecnologias e padrões arquitetônicos relevantes para embasar o uso dessas tecnologias na Web. Por fim, é apresentada uma demonstração prática de uma página Web que consome os serviços desenvolvidos.

O Capítulo 4, *Submissão de Projetos de Pesquisa Web e Multimídia a Comitês de Ética em Pesquisa*, tem como propósito aprimorar a capacidade e compreensão dos pesquisadores sobre a submissão de projetos de pesquisa envolvendo seres humanos a um Comitê de Ética em Pesquisa (CEP), visando melhorar a qualidade dos estudos acadêmicos e cien-



tíficos na área de Web e Multimídia, entre outras. Potenciais benefícios para os leitores incluem o desenvolvimento de sua noção moral, ética e legal sobre a submissão de projetos de pesquisa, o entendimento do processo de submissão, e aprender por meio de casos práticos na área de Web e Multimídia.

Agradecemos aos organizadores do evento pela oportunidade e aos revisores pelo trabalho na avaliação e na seleção das propostas aceitas. Temos certeza que este livro seja útil para todas as pessoas interessadas nos temas abordados.

Ribeirão Preto, abril de 2024.

*Manoel Carvalho Marques Neto (IFBA)*  
Coordenador de Minicursos e Tutorais  
Coordenador do Processo de Seleção

*Maria da Graça Pimentel (ICMC/USP)*  
Coordenadora de Minicursos e Tutorais

*Roberto Willrich (UFSC)*  
Editor

## **XXIX Simpósio Brasileiro de Sistemas Multimídia e Web**

*23 a 27 de outubro de 2023*

*Ribeirão Preto, Brasil*

### **Coordenação Geral**

Alessandra Alaniz Macedo (FFCLRP-USP) – *Coordenadora Geral*

Rudinei Goularte (ICMC-USP) – *Coordenadora Geral*

Celso Alberto Saibel Santos (UFES) – *Coordenador do Comitê de Programa*

Diego Roberto Colombo Dias (UFSJ) – *Coordenador do Comitê de Programa*

Leonardo Rocha (UFSJ) – *Coordenador do Comitê de Programa*

### **Coordenador do V Concurso de Teses e Dissertações (CTD)**

José Augusto Baranauskas (FFCLRP-USP)

Windson Viana de Carvalho (UFC)

### **Coordenadores do III Concurso de Trabalhos de Iniciação Científica (CTIC)**

Carlos André Guimarães Ferraz (UFPE)

Cléver Farias (FFCLRP-USP)

### **Coordenadores do XXII Workshop de Ferramentas e Aplicações e Prêmio LF**

Paulo Mazzoncini de Azevedo Marques (FMRP-USP)

Carlos de Salles Soares Neto (UFMA)

### **Coordenador de Publicação**

Roberto Willrich (UFSC)

### **Coordenação de Patrocínios e Contatos Institucionais**

Adriano César Machado Pereira (UFMG)

### **Coordenação de Palestras e Painéis**

Mirela M. Moro (UFMG)

Débora C. Muchalut Saade (UFF)

Cristina Godoy Bernardo de Oliveira (FDRP-USP)

Evandro Eduardo Seron Ruiz (FFCLRP-USP)

Kamila Rios da Hora Rodrigues (ICMC-USP)

Sílvia Amélia Bim (UTFPR)

## **Coordenação de Minicursos e Tutoriais**

Manoel Carvalho Marques Neto (IFBA)  
Maria da Graça Campos Pimentel (ICMC-USP)

## **Coordenação do Workshop Futuro da TV Digital Interativa**

Débora Christina Muchaluat Saade (UFF)  
Marcelo Ferreira Moreno (UFJF)

## **Coordenação de Competições**

Vinícius Fernandes Soares Mota (UFES)  
Flavio Vinicius Diniz de Figueiredo (UFMG)  
Humberto Torres Marques Neto (PUC Minas)  
Felipe Ferreira (Rede Globo)

## **Comissão Local**

Clever Ricardo Guareis de Farias (FFCLRP-USP)  
Evandro Eduardo Seron Ruiz (FFCLRP-USP)  
José Antonio Camacho-Guerrero (i-MedSys)  
José Augusto Baranauskas (FFCLRP-USP)  
Paulo Mazzoncini de Azevedo Marques (FMRP-USP)  
Renato Tinós (FFCLRP-USP)

## **Estudantes Voluntários**

Anastácia de Souza Freitas	Igor Ferronato Fonseca Pio
Bianca Rocha Nicoletti	João Gabriel de Oliveira Fernandes
Caio Uehara Martins	João Pedro Alves Januário
Caliu Lopes Pina	Kathley Lanna Rezende de Azevedo
Carlos Eugênio Costa de Lima	Matheus Carlos de Oliveira Carvalho
Filipe Pio Magalhães	Pedro Henrique da Silva Passos
Gabriel Rosa Arcangelo	Raúl Eduardo Sousa García
Gustavo Bender	Victor Hugo da Silva Lembor

## **Coordenação da Comissão Especial de Sistemas Multimídia e Web**

Joel dos Santos (Cefet/RJ) – *Coordenador*  
Leonardo Rocha (UFSJ) – *Vice-Coordenador*

## **Comitê Gestor**

Adriano César Machado Pereira (UFMG)	José Valdeni (UFRGS)
Alessandra Alaniz Macedo (FFCLRP-USP)	Jussara M. Almeida (UFMG)
Carlos de Salles Soares Neto (UFMA)	Leyza Baldo Dorini (UTFPR)
Celso Alberto Saibel Santos (UFES)	Maria da Graça C. Pimentel (ICMC-USP)
Cezar Teixeira (UFSCar)	Reinaldo Ferraz (NIC.Br/CEWEB.Br)
Diego Roberto Colombo Dias (UFSJ)	Roberto Willrich (UFSC)
Eduardo Barrère (UFJF)	Rudinei Goularte (ICMC-USP)
Humberto T. Marques-Neto (PUC Minas)	Thiago Henrique Silva (UTFPR)
José Andery Carneiro (USP)	Windson Viana de Carvalho (UFC)

## **Sociedade Brasileira de Computação (SBC)**

### **Presidência**

Thais Vasconcelos Batista (UFRN) – *Presidente*

Cristiano Maciel (UFMT) – *Vice-Presidente*

### **Diretoria**

Alírio Santos Sá (UFBA)

André Luís de Medeiros Santos (UFPE)

Carlos Eduardo Ferreira (USP)

Claudia Lage Rebello da Motta (UFRJ)

Denis Lima do Rosário (UFPA)

Eunice Pereira dos Santos Nunes (UFMT)

José Viterbo Filho (UFF)

Leila Ribeiro (UFRGS)

Lisandro Zambenedetti Granville (UFRGS)

Michelle Silva Wangham (UNIVALI)

Renata de Matos Galante (UFGRS)

Ronaldo Alves Ferreira (UFMS)

Tanara Lauschner (UFAM)

### **Contato**

Av. Bento Gonçalves, 9500

Setor 4 - Prédio 43.412 - Sala 219

Bairro Agronomia

91.509-900 – Porto Alegre RS

CNPJ: 29.532.264/0001-78

<http://www.sbc.org.br>



## Sumário

### **Capítulo 1. Como testar a acessibilidade em soluções mobile ..... 1**

Anderson C. Garcia (ICMC/USP), Juliana M. L. Eusébio (ICMC/USP),  
Kamila R. H. Rodrigues (ICMC/USP)

### **Capítulo 2. Desenvolvendo Aplicativos Android usando Kotlin .... 23**

Larissa Cardoso Zimmermann (ICMC/USP), Juliana Martins Leônico Eusébio  
(ICMC/USP), Maria da Graça Campos Pimentel (ICMC/USP)

### **Capítulo 3. Desenvolvimento full-stack com JavaScript: uma visão geral e prática ..... 89**

Bruna C. R. Cunha (ICMC/USP)

### **Capítulo 4. Submissão de Projeto de Pesquisa Web e Multimídia ao Comitê Nacional de Ética em Pesquisa ..... 125**

Maria da Graça Campos Pimentel (ICMC/USP), André Pimenta Freire (UFLA),  
Luiz Paulo Carvalho (UFRJ), Kamila Rios da Hora Rodrigues (ICMC/USP)

## Capítulo

# 1

## Como testar a acessibilidade em soluções *mobile*

Anderson C. Garcia, Juliana M. L. Eusébio e Kamila R. H. Rodrigues

<sup>1</sup>Instituto de Ciências Matemáticas e de Computação (ICMC), USP

{andersongarcia, julianaleoncio}@usp.br, kamila.rios@icmc.usp.br

### *Abstract*

*Currently, native Android apps still present interaction barriers for users with disabilities, emphasizing the ongoing need to enhance the accessibility of these applications. Research indicates that developers have shown some awareness of the importance of accessibility, but knowledge on the subject is superficial, practical guidance is lacking, accessibility requirements are not well-specified, responsibilities are not well-defined, and accessibility is not prioritized by stakeholders. This chapter aims to address these challenges and empower development, design, and testing professionals to tackle the complexities of accessibility in mobile solutions, exploring approaches and tools for testing and improving accessibility.*

### *Resumo*

*Atualmente, aplicativos Android nativos ainda apresentam barreiras que dificultam a interação dos usuários com deficiência, o que ressalta a necessidade contínua de melhorar a acessibilidade desses aplicativos. Pesquisas indicam que os desenvolvedores têm apresentado alguma consciência sobre a importância da acessibilidade, porém o conhecimento sobre o tema é superficial, faltam orientações práticas, requisitos de acessibilidade não são bem especificados, as responsabilidades não são bem definidas e a acessibilidade não é priorizada pelas partes interessadas. Este capítulo visa abordar esses desafios e capacitar profissionais de desenvolvimento, design e teste para enfrentar as complexidades da acessibilidade em soluções mobile, explorando abordagens e ferramentas para testar e melhorar a acessibilidade.*

### **1.1. Introdução**

A acessibilidade é um direito fundamental, defendido pela “Lei Brasileira de Inclusão” [1]. Acessibilidade significa eliminar barreiras para permitir o acesso de todas as pessoas, independentemente de deficiências ou outras limitações. Para cumprir essa missão, a acessibilidade é vital, especialmente no contexto das soluções computacionais.



Atualmente, aplicativos Android nativos ainda apresentam barreiras que dificultam a interação dos usuários, o que ressalta a necessidade contínua de melhorar a acessibilidade desses aplicativos. Existem instrumentos disponíveis, como guias, recomendações e ferramentas de testes, para auxiliar os desenvolvedores a eliminarem essas barreiras em seus aplicativos. No entanto, se observa que, apesar da disponibilidade dessas soluções, muitos aplicativos continuam apresentando problemas recorrentes [2, 3], o que indica que essas ferramentas podem não estar sendo adequadamente aplicadas.

Pesquisas têm buscado entender a razão de designers e desenvolvedores não implementarem produtos acessíveis [4, 5, 6, 7]. Embora considerem diferentes contextos de desenvolvimento, como *Web*, *mobile* e desenvolvimento ágil de software, essas pesquisas convergem em suas conclusões. Entre os argumentos, é possível citar: a) desenvolvedores têm apresentado alguma consciência sobre a importância da acessibilidade, porém o conhecimento sobre o tema é superficial, b) faltam orientações práticas, c) requisitos de acessibilidade não são bem especificados, d) as responsabilidades não são bem definidas e e) a acessibilidade não é priorizada pelas partes interessadas. Entre as possíveis soluções, apontadas nos estudos para esses problemas, destacam-se: a) oferecer cursos e treinamentos para ampliar o conhecimento dos desenvolvedores sobre como resolver os problemas de acessibilidade e b) utilizar suítes de testes apropriadas, que sejam simples e fáceis de serem usadas, por exemplo, incluindo testes automatizados.

Este capítulo visa discorrer sobre esses desafios e oferecer uma alternativa para capacitar profissionais de desenvolvimento, *design* e teste para enfrentar os desafios ainda existentes para implementar a acessibilidade em soluções *mobile*. Com foco em aplicativos Android nativos, serão explorados aqui abordagens e ferramentas para testar e melhorar a acessibilidade.

Conceitos teóricos sobre acessibilidade, com ênfase na acessibilidade em aplicativos móveis, serão explorados na Seção 3.2. Ainda nessa seção, os principais problemas de acessibilidade reportados na literatura serão discutidos, bem como os guias e recomendações disponíveis para auxiliar na eliminação das barreiras de acessibilidade e as diferentes abordagens de testes de acessibilidade. As Seções 3.3 e 3.4, por sua vez, apresentarão as principais ferramentas disponíveis, para aplicativos Android nativos, para auxiliar a realização de testes manuais, testes automatizados e análise de código. Em seguida, na Seção 3.5 este documento disponibiliza uma sugestão de atividade prática, em formato de tutorial, que visa auxiliar na identificação de problemas de acessibilidade em um aplicativo de exemplo – um aplicativo Contador — usando cada uma das ferramentas apresentadas previamente também neste documento.

## 1.2. Acessibilidade

Acessibilidade é o termo geral que qualifica um produto de modo a permitir que quaisquer pessoas tenham seu acesso, visando usufruir os benefícios da vida em sociedade. No contexto de desenvolvimento de software, a acessibilidade é um requisito não-funcional e uma sub característica de usabilidade [8].

A Norma ISO 9241-11 (2018) define acessibilidade como “o grau em que produtos, sistemas, serviços, ambientes e instalações podem ser usados por pessoas de uma população com a mais ampla gama de necessidades, características e capacidades de

usuários para atingir objetivos identificados em contextos de uso identificados”.

No contexto da Web, a acessibilidade visa que todas as pessoas, especialmente as pessoas com deficiência e idosos, possam acessar os conteúdos dos sites em uma variedade de contextos de uso, incluindo os mais usuais, e aqueles que contam com apoio de Tecnologia Assistiva (T.A.) [9]. Considerar que diferentes usuários possuem diferentes habilidades e podem utilizar diferentes tecnologias é imprescindível para que os conteúdos sejam de fato acessíveis.

### 1.2.1. Diretrizes de acessibilidade

Para criar aplicativos acessíveis a todos os usuários, é imperativo seguir os padrões internacionais de acessibilidade [10]. Para contribuir com uma Web acessível a um número cada vez maior de pessoas, a Iniciativa para a Acessibilidade na Web (do Inglês *Web Accessibility Initiative* (WAI)) foi lançada em 1997 pelo W3C [11]. Como resultado desta iniciativa, foi publicada em 1999 a primeira versão das Diretrizes para Acessibilidade do Conteúdo da Web - *do Inglês: Web Content Accessibility Guidelines* (WCAG), que definem um conjunto de recomendações sobre como tonar o conteúdo da Web acessível (W3C, 1999). Na versão 2.1 da WCAG [12]<sup>1</sup>, de 2018, as diretrizes buscaram melhorar as orientações de acessibilidade para grupos específicos de usuários, incluindo usuários com deficiências que utilizam dispositivos móveis.

Um exemplo da importância dessas diretrizes na acessibilidade ao conteúdo Web é a primeira delas, que discorre sobre textos alternativos para conteúdo não textual. Por exemplo, um usuário com deficiência visual pode não conseguir visualizar uma foto apresentada em tela, mas um leitor de telas poderá ler o texto alternativo que descreve essa foto, quando fornecido, sintetizando de modo sonoro o que é lido naquele texto alternativo, para o usuário.

Para corresponder às necessidades da variedade de interessados na WCAG, o documento é estruturado em camadas de orientação que incluem [12]:

- **Princípios.** Estabelecem a base necessária para acesso ao conteúdo da Web por qualquer pessoa. São quatro os princípios:
  1. **Perceptível.** Os usuários devem ser capazes de perceber a informação apresentada;
  2. **Operável.** Os usuários devem ser capazes de operar a interface;
  3. **Compreensível.** Os usuários devem ser capazes de entender a informação apresentada e a interação esperada;
  4. **Robusto.** Os usuários devem ser capazes de acessar o conteúdo de acordo com as tecnologias atuais e futuras disponíveis, incluindo as tecnologias de apoio.
- **Diretrizes.** Procuram auxiliar para que o conteúdo seja acessível ao maior número de pessoas e adaptável às diferentes habilidades físicas, sensoriais ou cognitivas. São 13 diretrizes, associadas aos 4 princípios estabelecidos;

---

<sup>1</sup>A versão 2.2 da WCAG se tornou estável e passou a ser a versão recomendada a partir de 5 outubro de 2023. Já existe uma versão 3.0 em andamento, disponível em <https://www.w3.org/TR/wcag-3.0/>, que versa sobre interfaces IoT, para vestíveis, dispositivos móveis, entre outros, mas ainda em fase de consolidação.

- **Critério de sucesso.** Sob cada diretriz, estão alocados os critérios de sucesso, os quais são afirmações testáveis para determinar se o conteúdo satisfaz o critério. No total, são 78 critérios de sucesso que se organizam nas 13 diretrizes, e se assemelham a requisitos. Os critérios são definidos em três níveis de conformidade: A (o mais básico), AA e AAA (o mais elevado);
- **Técnicas suficientes e sugeridas.** São técnicas suficientes aquelas que, uma vez implementadas, desde que exista suporte à acessibilidade para o usuário (por exemplo, T. A. disponível), atenderá ao critério de sucesso. Um exemplo de técnicas sugeridas são as que podem melhorar a acessibilidade ao conteúdo, mas podem não ser consideradas suficientes.

### 1.2.2. Acessibilidade em aplicativos móveis

As diretrizes da WCAG também são aplicáveis para aplicativos móveis. Em 2015 a W3C chegou a publicar um documento com dicas sobre como aplicar o conteúdo das diretrizes no contexto *mobile*<sup>2</sup>. Porém, a documentação é considerada extensa para desenvolvedores, e nem sempre é simples de adaptar as diretrizes a aplicativos nativos.

Além disso, existem outros conjuntos de guias e recomendações para desenvolvedores com foco em aplicativos nativos. Por exemplo, um trabalho desenvolvido por pesquisadores do Pernambuco e disponibilizado pelo instituto Samsung Instituto de Desenvolvimento para a Informática (SIDI) reúne 48 requisitos de acessibilidade com foco em deficiência visual [13]. O documento também apresenta recomendações para designers e boas práticas para desenvolvedores, incluindo exemplos de código e uma sucinta explanação sobre como planejar testes de acessibilidade.

Outro guia disponível é o *Mobile Accessibility Standards and Guidelines*, da BBC [14]. O guia reúne melhores práticas, recomendações e padrões para aplicativos móveis e apresentam, quando aplicável, instruções para a realização de testes de acessibilidade com ferramentas de Tecnologia Assistiva.

### 1.2.3. Problemas de acessibilidade para pessoas com deficiência visual

Pessoas com deficiência visual usam principalmente leitores de tela para interagir com aplicativos móveis. No Android, o leitor de tela integrado à plataforma é o *Talkback*.

A usabilidade dos aplicativos para esses usuários depende diretamente da compatibilidade com os recursos de T.A. Por exemplo, o leitor de tela precisa que os elementos de UI (Interface de Usuário) sejam devidamente rotulados e organizados de forma lógica para que possam ser corretamente interpretados e a informação ser transmitida ao usuário.

Estudos recentes analisaram aplicativos Android disponíveis na *Play Store*, e identificaram que a maioria deles ainda apresenta problemas de acessibilidade [2, 3, 15]. Os problemas mais recorrentes, especialmente os que afetam usuários com deficiência visual, são de detecção e correção relativamente simples com os recursos atuais da literatura e indústria.

Além disso, um estudo de 2019 [2] identificou que 5 widgets representam 92% dos elementos de UI e respondem por 89% das violações de acessibilidade encontradas.

---

<sup>2</sup>Disponível em: <https://www.w3.org/TR/mobile-accessibility-mapping/>

A Tabela 3.1 relaciona seis desses problemas de acessibilidade mais recorrentes nos aplicativos Android avaliados nesses estudos, associados às respectivas recomendações de acessibilidade da literatura (nomeadas de maR). Além disso, pensando em soluções mais pragmáticas, foram endereçados para cada recomendação os *widgets* Android, dentre os mais utilizados, e os respectivos atributos que devem ser verificados [16]. Por exemplo, para o problema de contraste de texto insuficiente, recomenda-se utilizar uma taxa de contraste de pelo menos 4.5:1, e esse requisito deve ser aplicado a *TextViews* e *Buttons*, por exemplo.

**Tabela 1.1. Problema de acessibilidade e recomendações para os *widgets* mais usados [16].**

Problema	Recomendação / Requisito	Widgets
Contraste de texto insuficiente	Deve-se utilizar uma taxa de contraste de pelo menos 4.5:1 (maR7)	<i>TextView</i> , <i>Button</i>
Tamanho do alvo de toque inadequado	Alvo de toque deve ter pelo menos 48x48dp (maR17)	<i>Button</i> , <i>ImageButton</i>
Falta de rótulo no componente	Para cada controle de formulário, deve existir um <i>TextView</i> com o atributo <i>labelFor</i> associado, ou deve ter o atributo <i>hint</i> fornecido (maR10)	<i>EditText</i> , <i>CheckBox</i> , <i>RadioButton</i> , <i>Switch</i>
Imagens e/ou ícones sem alternativa textual	Todo conteúdo não textual, como imagens e conteúdo de mídia, deve possuir descrição alternativa em texto (maR1)	<i>ImageButton</i> , <i>ImageView</i>
Interação e/ou navegação confusa	Aplicativo deve suportar navegação baseada em foco (maR13)	<i>Accessibility</i> <i>NodeInfo</i>
Falta de espaçamento suficiente entre os elementos	Componentes de interação devem ter um espaçamento mínimo de 8dp entre si e das bordas da tela (maR18)	

Para a realização das tarefas sugeridas na Seção 3.5, é importante compreender melhor três desses problemas.

### Contraste de texto insuficiente

Usuários com baixa visão têm mais dificuldade para ler informações em uma tela quando não há contraste suficiente entre as cores do primeiro e segundo planos (fundo e fonte, por exemplo). Proporções baixas de contraste podem deixar a tela embaçada para alguns usuários, enquanto proporções mais altas deixam a tela mais nítida. Situações diferentes de luz podem aumentar as dificuldades criadas por proporções baixas de contraste. O uso de um bom contraste ajuda todos os usuários, e não apenas aqueles com deficiência.

### Falta de rótulo no componente

Usuários cegos ou com baixa visão usam leitores de tela, como o *TalkBack*, para interagir com os dispositivos. O *TalkBack* anuncia o conteúdo da tela para os usuários, e esses podem interagir com ele. Quando um elemento não tem um texto associado (um *ImageButton*, por exemplo), o *TalkBack* não sabe como informar o propósito dele ao usuário. Em casos assim, o recurso pode anunciar títulos genéricos, como “Botão sem rótulo”, o que não é útil para o usuário. Quando o desenvolvedor fornece um rótulo descritivo adequado, o *TalkBack* pode anunciá-lo ao usuário.

## Tamanho do alvo de toque inadequado

Muitas pessoas têm dificuldade para focar em áreas de toque pequenas na tela. Isso pode acontecer por terem dedos grandes ou alguma condição médica que afeta as habilidades motoras delas. As áreas de toque pequenas também dificultam para os usuários de leitor de tela navegarem em aplicativos movendo um dedo na tela, como durante o uso do recurso “Explorar por toque” no *TalkBack*.

Em algumas circunstâncias, áreas de toque inadequadas tornam aplicativos menos acessíveis para todos os usuários, e não apenas aqueles com deficiência.

### 1.3. Testes automatizados no Android

A automação de testes traz benefícios como a redução do tempo de execução, uma menor propensão a erros, a liberação de recursos, a segurança para testes de regressão e o *feedback* rápido e frequente. Além disso, os testes automatizados oferecem benefícios adicionais, como impulsionar a codificação, servir como documentação e ter um bom retorno sobre o investimento [17]. Os testes automatizados aprimoram a eficiência, a qualidade e cobertura dos testes, permitindo que os desenvolvedores se concentrem em outras atividades importantes para as aplicações [18].

Os testes podem ser classificados de acordo com o tamanho ou grau de isolamento. Os testes de unidade ou testes pequenos verificam pequenas partes do código, como uma classe ou método. Para testes que precisam de mais de uma unidade de código ou usam recursos fora da unidade de código, é recomendada a realização de testes de integração ou testes médios. Já os testes de interface do usuário ou testes grandes são usados para testar fluxos de interação do usuário ou o funcionamento de partes maiores do aplicativo que precisam ser testadas [19].

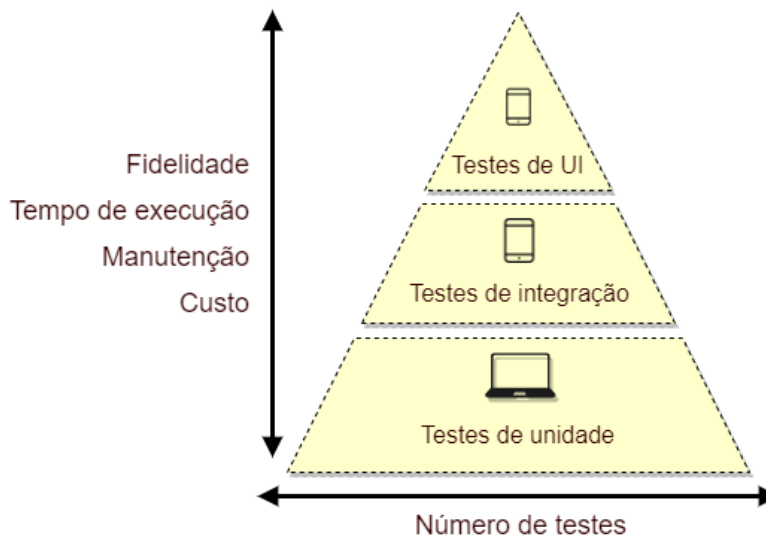
Uma abordagem recomendada para testes é ilustrada na Figura 3.1. Os testes pequenos são mais baratos e fornecem um *feedback* mais rápido aos desenvolvedores, sendo preferíveis. Já os testes grandes são mais caros e demorados, mas mais confiáveis para testar o fluxo de interação do usuário [17, 19, 20].

A plataforma Android também classifica os testes de acordo com o ambiente de execução. Os testes locais são aqueles que podem ser executados diretamente no ambiente de desenvolvimento. Os testes instrumentados são testes que requerem um dispositivo Android, seja físico ou emulado, e são necessários para os testes de interface do usuário [21]. Testes de unidade instrumentados são possíveis, porém, são significativamente mais lentos do que os testes de unidade locais [20, 21].

### 1.4. Testes de acessibilidade no Android

Quatro abordagens são sugeridas para testes de acessibilidade, e idealmente elas devem ser combinadas. São elas:

- **Testes manuais**, que exploram o aplicativo da forma como o usuário o utilizaria. Por exemplo, usando um leitor de tela para simular o uso por usuário com deficiência visual;
- **Testes com ferramentas de análise**, que fazem verificações estáticas no código;
- **Testes automatizados**, geralmente realizados com auxílio de *frameworks* de testes;



**Figura 1.1. Pirâmide de sugestão para distribuição de testes de acordo com os tipos. Fonte: Adaptada da Google [21].**

- **Testes de usuários**, os mais completos e ricos, pois envolvem o *feedback* de usuários reais.

Para auxiliar a criação de testes de acessibilidade, é disponibilizado no Android a biblioteca *Accessibility Test Framework*<sup>3</sup>, uma base para outras ferramentas da plataforma, no que se refere à acessibilidade. Ela contém uma série de verificações pré-definidas, que incluem presença de *labels*, tamanho da área de toque, contraste de cores, além de outras propriedades relacionadas com serviços de acessibilidade (aqueles usados por T.A.). Além disso, também faz parte da biblioteca a API *AccessibilityChecks*<sup>4</sup>, que permite, com poucas linhas de código, que essas verificações sejam feitas automaticamente ao realizar testes que interagem com a tela.

A Tabela 3.2 resume as principais ferramentas sugeridas na documentação para desenvolvedor Android [19] para verificações de acessibilidade. O Scanner de Acessibilidade, um aplicativo que analisa a tela de outros aplicativos, é o único que foi criado especificamente para a acessibilidade. *UI Automator* [22] e *Espresso* [23] são ferramentas para testes automatizados de UI, que se diferem pelo tipo de teste: o *UI Automator* realiza testes *blackbox*, já o *Espresso* realiza testes *whitebox*. Como o *Espresso* é possível acionar a API *AccessibilityChecks*. Todos esses recursos requerem um dispositivo para serem executados.

Uma opção para testes locais é o *Roboletric* [24]. Ele chegou a suportar testes com a *AccessibilityChecks*, porém essa funcionalidade foi descontinuada na versão 4.5, pois não era capaz de identificar problemas que eram identificados com o *Espresso*. O *Roboletric* suporta alguma interação com elementos da UI, por simulações, porém o suporte a gráficos nativos só foi adicionado na versão 4.10, de abril de 2023.

<sup>3</sup>Disponível em: <https://github.com/google/Accessibility-Test-Framework-for-Android>

<sup>4</sup>Disponível em: <https://developer.android.com/training/testing/espresso/accessibility-checking?hl=pt-br>

**Tabela 1.2. Ferramentas para testar acessibilidade no Android.**

Ferramenta	Testes	Execução	Esforço requerido
Scanner de Acessibilidade	Teste grandes	Instrumentado	Execução e verificação do <i>app</i> tela a tela
<i>UI Automator</i>	Testes grandes	Instrumentado	Automatização de testes de UI
Espresso	Testes médios Testes grandes	Instrumentado	Chamada da API <i>AccessibilityChecks</i> a partir de <i>View</i> raiz de cada tela
<i>Robolectric</i>	Testes pequenos	Local	Escrita de testes para cada verificação de acessibilidade
Lint	Estáticos	Local	Inspeção automática em conjunto pré-definido de verificações

Por último, o Android *Lint* [25] é uma ferramenta de análise estática de código integrada à IDE de desenvolvimento.

Além das ferramentas da própria plataforma Android, podem ser encontradas na literatura uma série de outras ferramentas que podem ser utilizadas por desenvolvedores para testar a acessibilidade de aplicativos Android, como **PUMA** [26], **forApp**<sup>5</sup>, **Axe Android**<sup>6</sup>, **IBM Mobile Accessibility Checker**<sup>7</sup>, **Latte** [27], **Bility** [28], **Mate** [29] e **AccessibiLint** [30].

Para as tarefas sugeridas neste capítulo e descritas na Seção 3.5, a seguir, são utilizados o **Scanner de Acessibilidade**, o **Espresso** e o **Automated Accessibility Tests Kit (AATK)** [16]. Esse último, um kit de testes locais de acessibilidade para aplicativos Android. Trata-se de uma biblioteca escrita em Java, com testes escritos para serem executados com o *Robolectric*, ou seja, não requerem uso de dispositivos.

### 1.5. Sugestão de atividade prática para testes

Nesta seção são apresentadas três alternativas de ferramentas disponíveis para auxiliar a realização de testes de acessibilidade em aplicativos Android nativos. O leitor é guiado a identificar problemas de acessibilidade em um aplicativo de exemplo – um app Contador – com cada uma das seguintes ferramentas:

- **Scanner de Acessibilidade (*Accessibility Scanner*)**: ferramenta do Google para verificar aplicativos móveis em busca de problemas de acessibilidade;
- **Espresso**: ferramenta de automação de testes de interface do usuário no Android, para testes instrumentados;
- **AATK**: biblioteca Android contendo testes de acessibilidade automatizados locais, ou seja, que não requerem uso de dispositivo físico ou emulado [16].

Para cada ferramenta são fornecidas as instruções passo-a-passo para preparação, configuração, escrita dos testes (quando aplicável), execução e visualização dos resultados. Ao final de cada uma delas, também são oferecidas sugestões para solucionar os problemas encontrados e re-executar os testes.

Os recursos necessários para a execução das tarefas são:

<sup>5</sup><https://www.forapp.org/>

<sup>6</sup><https://www.deque.com/android-accessibility/>

<sup>7</sup><https://www-03.ibm.com/able/mobile-accessibility-checker.html>



- Estação de trabalho com Android Studio<sup>8</sup> instalado (versão 2021.1.1 ou superior);
- Dispositivo:
  - smartphone Android com cabo USB e modo de desenvolvedor habilitado; ou
  - emulador com Play Store baixado e configurado no Android Studio (sugestão: Pixel 2 API 30).

Não é necessário nenhum conhecimento prévio sobre acessibilidade ou testes automatizados para realizar essas tarefas. No entanto, se assume que o interessado seja capaz de baixar um projeto do GitHub e abrir no Android Studio.

A realização deste treinamento permite:

- Compreender os conceitos básicos de acessibilidade em sistemas computacionais;
- Aprender a utilizar ferramentas para testar a acessibilidade em aplicativos Android nativos;
- Identificar, solucionar e prevenir problemas de acessibilidade em um aplicativo de exemplo por meio de testes de acessibilidade automatizados.

### 1.5.1. O aplicativo Contador

Neste treinamento é utilizado um aplicativo existente, o Contador, derivado do Google Codelabs.<sup>9</sup> Esse aplicativo permite aos usuários rastrear, incrementar e decrementar uma contagem numérica. Embora o aplicativo seja simples, existem nele alguns problemas de acessibilidade que podem dificultar que usuários com deficiência interajam com sua interface.

O código-fonte da versão inicial do aplicativo pode ser obtido no link <https://github.com/AALT-Framework/poor-accessibility-apps>.

Para executar testes de acessibilidade e identificar os problemas com cada uma das três ferramentas, para simplificação deste tutorial, devem ser criadas três cópias da pasta do projeto localmente.

1. Abra o terminal ou prompt de comando no diretório onde deseja clonar o projeto;
2. Clone o repositório disponível em <https://github.com/AALT-Framework/poor-accessibility-apps/>;
3. Crie uma cópia da pasta “Contador”, que está dentro do repositório clonado, para cada projeto. Nomeie cada cópia de acordo com a ferramenta que se deseja testar, da seguinte forma:
  - Contador-AccessibilityScanner
  - Contador-Espresso
  - Contador-AATK
4. Para cada ferramenta, abra a respectiva pasta do projeto no Android Studio e siga as instruções do tutorial.

---

<sup>8</sup><https://developer.android.com/studio>

<sup>9</sup><https://codelabs.developers.google.com/>

### 1.5.2. Testes de acessibilidade no aplicativo Contador com o Scanner de Acessibilidade

O Scanner de acessibilidade (*Accessibility Scanner*) é uma ferramenta criada pelo Google para sugerir melhorias de acessibilidade em aplicativos Android, como aumento de áreas de toque pequenas, aumento de contraste e descrições de conteúdo. As melhorias possibilitam que pessoas com deficiência possam usar os aplicativos com mais facilidade.

#### Preparação

1. No Android Studio, abra o projeto do aplicativo Contador, da pasta *Contador-AccessibilityScanner*, criada no 2º passo deste treinamento;
2. Execute o projeto no dispositivo Android ou emulador. O aplicativo Contador será aberto;
3. Siga estas etapas para fazer o *download* e configurar o Scanner de Acessibilidade:
  - (a) Abra o aplicativo Play Store no dispositivo Android;
  - (b) Faça o download do Scanner de Acessibilidade na Google Play Store;
  - (c) Após instalar o Scanner de Acessibilidade, navegue até Configurações > Acessibilidade no dispositivo. Localize e ative o Scanner de acessibilidade (toque em "Permitir" ou "OK" e, em seguida, em "Iniciar autorização" para concluir o fluxo de configuração);
  - (d) Retorne ao aplicativo Contador. A tela deverá estar como a ilustrada na Figura 3.2.

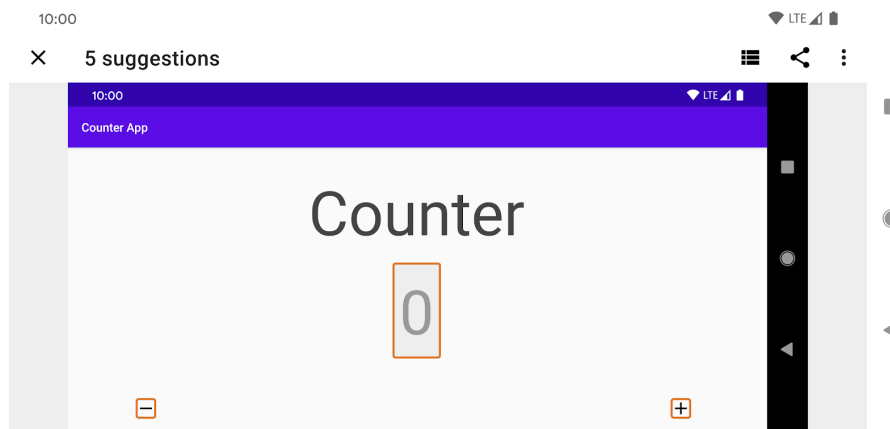


Figura 1.2. Tela do aplicativo Contador após habilitar Scanner de Acessibilidade.

O Scanner de acessibilidade cria um botão de ação flutuante azul (FAB, na sigla em Inglês), que fica sobreposto ao conteúdo na tela.

1. Toque no FAB para iniciar a verificação. Ao fazer isso, o Scanner de acessibilidade examina a interface de usuário da tela, realiza uma auditoria rápida de acessibilidade e prepara as sugestões de melhoria;
2. Toque em "Verificar" para ver as sugestões de melhoria. O Scanner de acessibilidade exibe uma lista de sugestões de melhoria de acessibilidade, classificadas por prioridade. As sugestões de melhoria são baseadas nas diretrizes de acessibilidade do Google;

3. Toque em uma sugestão de melhoria para ver mais detalhes. Por exemplo, toque em "Aumentar o contraste" para ver mais detalhes sobre essa sugestão.

O Scanner de acessibilidade tem cinco sugestões para melhorar a acessibilidade do Contador. Nos passos a seguir são sugeridas alternativas para realizar essas melhorias.

### 1.5.3. Como garantir um contraste de cor adequado

No Contador, o contraste de cor é simples de melhorar. A *TextView*, que mostra a contagem, usa um plano de fundo cinza-claro com texto cinza:

```
<TextView
    ...
    android:background="@color/lightGrey"
    android:textColor="@color/grey"
    ...
/>
```

O desenvolvedor pode remover o plano de fundo, escolher outro mais claro ou deixar o texto mais escuro. Nesta atividade, é sugerido escolher uma cor mais escura para o texto. Segue abaixo exemplos de cores que foram definidas em **colors.xml**:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    ...
    <color name="lightGrey">#EEEEEE</color>
    <color name="grey">#999999</color>
    <color name="darkGrey">#666666</color>
</resources>
```

Abra o arquivo **res/layout/activity\_main.xml** e mude *android:textColor="@color/grey"* para *android:textColor="@color/darkGrey"*:

```
<TextView
    ...
    android:background="@color/lightGrey"
    android:textColor="@color/darkGrey"
    ...
/>
```

Agora, execute o aplicativo e veja o contraste melhorado. A proporção de contraste agora é de 4.94:1, consideravelmente melhor que 2.45:1, que é o valor da taxa de contraste anterior.

Diretrizes de acessibilidade de conteúdo da Web [12] recomendam uma proporção de contraste mínima de 4.5:1 para todo o texto. A proporção de 3.0:1 é considerada aceitável para textos grandes ou em negrito.

Pressione o FAB para iniciar outra verificação no Scanner de acessibilidade. É possível ver que o aplicativo não tem mais sugestões relacionadas ao contraste de cor.

### 1.5.4. Como adicionar rótulos ausentes

No aplicativo Contador, as ações de decrementar e incrementar são representadas por dois *ImageButton*, (-) e (+) respectivamente. Por serem imagens sem rótulos, um leitor de tela como o *TalkBack* não consegue comunicar adequadamente a semântica das visualizações para o usuário, anunciando simplesmente “botão sem rótulo”, quando um desses botões é focado.

Para corrigir esse problema, atribua uma *android:contentDescription* para cada botão:

```
<ImageButton
    android:id="@+id/subtract_button"
    ...
    android:contentDescription="@string/decrement" />

<ImageButton
    android:id="@+id/add_button"
    ...
    android:contentDescription="@string/increment" />
```

Use *strings* localizadas nas descrições de conteúdo. Assim, elas poderão ser adequadamente traduzidas. Para esta atividade, as *strings* já foram definidas em **res/values/strings.xml**.

Agora, um leitor de tela pode anunciar o valor da *contentDescription* fornecida (adequadamente traduzida para o idioma local) quando o usuário foca nos botões.

Execute o Scanner de acessibilidade novamente. Não deverá mais ter sugestões relacionadas a rótulos ausentes.

### 1.5.5. Como aumentar áreas de toque

O Scanner de acessibilidade continua sugerindo que os botões (-) e (+) precisam ter uma área de toque maior. a sugestão de ajuste é a seguinte.

Os dois botões no Contador são pequenos (24dp x 24dp). No geral, o tamanho adequado para a área de toque de itens focáveis precisa ser, pelo menos, de 48dp x 48dp. Se for possível criar uma área ainda maior, melhor. Ao aumentar a área de toque de 24dp x 24dp para 48dp x 48dp, ela é expandida por um fator de 4.

O desenvolvedor/designer tem várias opções para aumentar a área de toque dos botões. Por exemplo, escolher uma das opções abaixo:

- Adicionar *padding* em volta dos ícones;
- Adicionar um *minWidth* e/ou *minHeight* (os ícones ficarão maiores);
- Registrar um *TouchDelegate*.

Adicione um pouco de *padding* a cada visualização:

```
<ImageButton
    ...
    android:padding="@dimen/icon_padding"
```

```
... />  
  
<ImageButton  
...  
    android:padding="@dimen/icon_padding"  
... />
```

O valor de `@dimen/icon_padding` está definido como 12dp (veja `res/dimens.xml`). Quando o padding é aplicado, a área de toque do controle se torna 48dp x 48dp (24dp + 12dp em cada direção).

Execute o aplicativo novamente para confirmar os novos limites de layout. Agora, a área de toque dos botões é maior.

Execute o Scanner de acessibilidade novamente. Desta vez, a análise será concluída sem sugestões.

## Como desativar o Scanner de Acessibilidade

Após concluídos os testes com o Scanner de Acessibilidade, caso se deseje desativá-lo, navegue para **Configurações, Acessibilidade** e defina o Scanner de acessibilidade como **Desativado**.

### 1.5.6. Testes de Acessibilidade no Aplicativo Contador com o Espresso

O Espresso é um *framework* de teste de interface do usuário para aplicativos Android, permitindo que os desenvolvedores criem testes automatizados para interagir com os elementos da interface do usuário do aplicativo. É integrado com o Android Studio e pode ser executado em dispositivo físico ou emulado.

Para testar acessibilidade com o Espresso, é possível usar a API *Accessibility-Checks* do ATF.

## Preparação

No Android Studio, abra o projeto do aplicativo Contador, da pasta Contador-Espresso criada no 2º passo deste treinamento.

O desenvolvedor precisará de uma nova dependência para o pacote *androidTestImplementation*. Confira se a linha seguinte já foi adicionada no arquivo **app/build.gradle**.

1. Edite o arquivo *build.gradle*<sup>10</sup> da raiz, adicionando a seguinte linha na lista de dependências;
2. Depois de fazer essas alterações, sincronize seu projeto para garantir que elas entrem em vigor.

---

<sup>10</sup>**Onde encontro isso?** Ao abrir o projeto no Android Studio, na visualização Android (painel esquerdo), há uma seção *Gradle Scripts*. Dentro, há um arquivo chamado **build.gradle (Módulo: Contador.app)**, ou algo parecido.

```
dependencies {
    ...
    androidTestImplementation
        'androidx.test.espresso:espresso-accessibility:3.3.0-alpha05'
    ...
}
```

### Crie uma classe de teste instrumentado para a tela principal

1. No Android Studio, abra o painel *Project* e encontre esta pasta:
  - **com.example.contador (androidTest)**;
2. Clique com o botão direito na pasta contador e selecione *New - Java Class*;
3. Nomeie como *MainActivityInstrumentedTest*. Assim é possível saber que esta classe de teste instrumentado se refere à *MainActivity*.

Com a classe *MainActivityInstrumentedTest* gerada e aberta, crie seu primeiro teste. Para o propósito deste treinamento, será escrito apenas um único teste, que verifica se o código para incrementar a contagem funciona corretamente (por questões de brevidade, o teste para decrementar a contagem foi omitido). Sua classe deverá ficar assim:

```
public class MainActivityInstrumentedTest {
    @Rule
    public ActivityScenarioRule<MainActivity>
        mActivityTestRule =
            new ActivityScenarioRule<>(MainActivity.class);

    @Test
    public void testIncrement(){
        Espresso.onView(withId(R.id.add_button))
            .perform(ViewActions.click());
        Espresso.onView(withId(R.id.countTV))
            .check(matches(withText("1")));
    }
}
```

Agora, verifique se o seu computador está conectado a um dispositivo com a depuração USB ativada, e execute os testes clicando no botão de seta verde imediatamente à esquerda de *@Test public void testIncrement()*. Se um dispositivo físico conectado via USB estiver sendo usado, se certifique de que o dispositivo esteja desbloqueado e com a tela ligada. Observe que pressionar **Ctrl+Shift+F10** (**Control+Shift+R** em um Mac) executa os testes no arquivo atualmente aberto.

O teste deve ser executado até o final e deve passar, confirmando que o incremento da contagem funciona como esperado.

A seguir, é preciso modificar o teste para verificar também a acessibilidade.

## Habilite as checagens de acessibilidade com o Espresso

Com o Espresso, é possível habilitar verificações de acessibilidade chamando *AccessibilityChecks.enable()* de um método de configuração. Adicionar essa única linha de código permite que seja feito o teste da interface do usuário para acessibilidade, tornando fácil integrar a verificação de acessibilidade em seu conjunto de testes.

Para configurar a classe **MainActivityInstrumentedTest** para checagens de acessibilidade, adicione o seguinte método de configuração antes do teste.

```
@BeforeClass
public static void beforeClass ()
{
    AccessibilityChecks . enable ();
}
```

Agora, execute o teste novamente. Desta vez, será possível perceber que o teste falha. No painel *Run*, clique duas vezes em *testIncrement* para ver os resultados. Será possível notar a mensagem de erro apontando dois problemas de acessibilidade:

- O *ImageButton* de adição (+) contém uma imagem, mas não tem um rótulo;
- O *ImageButton* de adição (+) precisa de um alvo de toque maior.

Essas verificações de acessibilidade estão associadas ao botão “adicionar”, que é a visualização na qual foi executada a ação do teste de incremento. Para permitir que a API examine outras visualizações na hierarquia, sem precisar executar ações adicionais em outras visualizações, é necessário chamar o método *setRunChecksFromRootView*, do objeto *AccessibilityValidator* na habilitação da *AccessibilityChecks*.

Modifique seu método da seguinte forma:

```
@BeforeClass
public static void beforeClass ()
{
    AccessibilityChecks . enable ()
        . setRunChecksFromRootView ( true );
}
```

Execute os testes novamente. Desta vez, os problemas de acessibilidade encontrados pelo Espresso, com o *AccessibilityChecks*, no aplicativo Contador, são os mesmos identificados na Seção 3.5.2, com o Scanner de Acessibilidade, incluindo as falhas encontradas no botão de decremento (-) e no *TextView* de visualização da contagem. Portanto, para corrigi-los, devem ser realizadas as mesmas tarefas indicadas nas seções 3.5.3, 3.5.4 e 3.5.5.

Após todas as correções, execute o teste novamente. O teste deve executar com sucesso até o final.

### 1.5.7. Testes de acessibilidade no aplicativo Contador com o AATK

O Kit de Testes de Acessibilidade Automatizados (do inglês *Automated Accessibility Tests Kit* (AATK)) para Aplicativos Android consiste em uma coleção de testes de acessibili-



dade automatizados projetados para serem executados com o *Robolectric*. Isso permite que sejam executados como testes locais, sem a necessidade de um dispositivo físico ou emulado.

Este kit foi desenvolvido com foco nos problemas de acessibilidade mais comuns, para pessoas com deficiência visual, e nos *widgets* mais usados, em que muitos problemas de acessibilidade tendem a ocorrer.

## Preparação

No Android Studio, abra o projeto do aplicativo Contador, da pasta Contador-AATK, criada no 2º passo deste treinamento.

Siga os seguintes passos para preparar o projeto para adicionar testes de acessibilidade automatizados:

1. Edite o arquivo **settings.gradle** da raiz<sup>11</sup>, adicionando *maven url 'https://jitpack.io'* na lista de *repositories*:

```
dependencyResolutionManagement {
    repositoriesMode
        .set(RepositoriesMode.FAIL_ON_PROJECT_REPOS)
    repositories {
        google()
        mavenCentral()
        maven { url 'https://jitpack.io' }
    }
}
```

2. Configure seu arquivo **build.gradle** no nível do aplicativo<sup>12</sup> para habilitar a execução de testes com o *Robolectric* e o AATK, atualizando os *testOptions* e adicionando as dependências necessárias.

Primeiro, adicione a diretiva *testOptions* com as seguintes linhas, dentro da diretiva *android*, assim:

```
android {
    ...
    testOptions {
        // Usado para testar elementos dependentes do Android
        // na pasta de teste
        unitTests.includeAndroidResources = true
        unitTests.returnDefaultValues = true
    }
}
```

Em seguida, adicione estas duas dependências como *testImplementation*:

```
android {
    ...
```

<sup>11</sup>**Onde encontro isso?** Ao abrir o projeto no Android Studio, na visualização do Android (painel esquerdo), há uma seção *Gradle Scripts*. Dentro, há um arquivo chamado **settings.gradle (Project: Settings)**.

<sup>12</sup>**Onde encontro isso?** Ao abrir o projeto no Android Studio, na visualização do Android (painel esquerdo), há uma seção *Gradle Scripts*. Dentro, há um arquivo chamado **build.gradle (Módulo: Contador.app)**, ou algo parecido.

```
testOptions {
    dependencies {
        ...
        testImplementation 'org.robolectric:robolectric:4.9'
        testImplementation
            'com.github.AALT-Framework:
            .....android-accessibility-test-kit:v1.0.0-alpha'
        ...
    }
}
```

3. Depois de fazer essas alterações, sincronize seu projeto para garantir que elas entrem em vigor.

### Crie uma classe de teste local para a tela principal

1. No Android Studio, abra o painel *Project* e encontre esta pasta:
  - **com.example.contador (test)**
2. Clique com o botão direito na pasta contador e selecione *New - Java Class*;
3. Nomeie como *MainActivityTest*. Assim será possível saber que essa classe de teste se refere à *MainActivity*.

Com a classe **MainActivityTest** gerada e aberta, comece a configurá-la para executar os testes AATK.

Sua classe deverá ficar assim:

```
@RunWith(RobolectricTestRunner.class)
public class MainActivityTest {
    private View rootView;
    private AccessibilityTestRunner runner;

    @Rule
    public ErrorCollector collector = new ErrorCollector();

    @Before
    public void setUp() {
        // Crie uma instancia da atividade
        MainActivity activity =
            Robolectric
                .buildActivity(MainActivity.class)
                .create()
                .get();

        // Obtenha a view raiz da hierarquia de exibicao
        rootView = activity
            .getWindow()
            .getDecorView()
            .getRootView();
        runner = new AccessibilityTestRunner(collector);
    }
}
```

```
}
```

O que foi feito:

1. Adicionado o escopo da classe para executar com *RoboletricTestRunner*;
2. Declarado um atributo privado para manter o *rootView* e o *AccessibilityTestRunner*;
3. Declarada uma propriedade pública para o *ErrorCollector*;
4. Adicionado um método *setUp* que habilita a execução do kit em qualquer novo teste criado.

### Escreva seu primeiro teste com o AATK

Adicione um método de teste para cada teste de acessibilidade que deseja executar. Será iniciado com a verificação da taxa de contraste de cores.

O desenvolvedor pode utilizar o teste de taxa de contraste do AATK (*TestAdequateContrastRatio*) da seguinte forma:

1. Adicione um método de teste. Procure seguir boas convenções para nomenclatura de testes. Por exemplo: **deve\_UsarTaxaDeContrasteAdequada**;
2. Chame o método *runAccessibilityTest* do executor do kit, passando como parâmetro a *view* raiz e uma nova instância do teste desejado:

```
@Test
public void deve_UsarTaxaDeContrasteAdequada () {
    runner.runAccessibilityTest (rootView ,
        new TestAdequateContrastRatio ());
}
```

3. Execute seu teste. Clique com o botão direito do mouse sobre o nome do método e selecione *Run MainActivityTest.deve\_UsarTaxaDeContrasteAdequada*;
4. No painel *Run*, clique duas vezes em *deve\_UsarTaxaDeContrasteAdequada* para ver os resultados. Será possível notar a mensagem de erro, a identificação da visualização, a taxa esperada e a taxa atual;
5. Abra o arquivo **res/layout/activity\_main.xml**, encontre o *TextView* e altere *android:textColor="@color/grey"* para *android:textColor="@color/darkGrey"*;
6. Volte ao item 3 para refazer o teste e ver se ele passou.

#### 1.5.7.1. Escreva novos testes

Agora que já criou seu primeiro teste, é possível adicionar outros. A seguir, é sugerido mais dois exemplos. Consulte a documentação do AATK, disponível em <https://github.com/AALT-Framework/android-accessibility-test-kit>, para ter acesso a todos os testes disponíveis.

Para testar conteúdos não textuais sem descrição alternativa, será preciso utilizar o teste de texto alternativo do AATK (*TestMustHaveAlternativeText*), assim como foi feito para o teste de contraste.

1. Adicione um método de teste. Por exemplo: *deve\_ConterAlternativaTextual*;

2. Chame o método *runAccessibilityTest* do executor do kit, passando como parâmetro a *view* raiz e uma nova instância do teste desejado:

```
@Test
public void deve_ConterAlternativaTextual () {
    runner.runAccessibilityTest (rootView ,
        new TestMustHaveAlternativeText ());
}
```

3. Execute seu teste. Verifique os resultados. Realize as correções, conforme visto na Seção 3.5.4. Reexecute o teste.

Para verificar o tamanho dos alvos de toque, será preciso utilizar o teste de texto alternativo do AATK (*TestTouchTargetSize*), assim como foi feito para os testes anteriores.

1. Adicione um método de teste. Por exemplo: *deve\_AlvoDeToquePossuirTamanhoMinimo*;
2. Chame o método *runAccessibilityTest* do executor do kit, passando como parâmetro a *view* raiz e uma nova instância do teste desejado:

```
@Test
public void deve_AlvoDeToquePossuirTamanhoMinimo () {
    runner.runAccessibilityTest (rootView ,
        new TestTouchTargetSize ());
}
```

3. Execute seu teste. Verifique os resultados. Realize as correções, conforme visto na Seção 3.5.5. Re-execute o teste.

## 1.6. Considerações Finais

A acessibilidade em soluções computacionais, além de ser um imperativo ético, é uma condição legal no contexto brasileiro. A empatia com essa temática deve ser complementada por uma abordagem profissional e legalmente respaldada, exigindo que os profissionais envolvidos no desenvolvimento de tecnologias compreendam e implementem de forma séria os requisitos de acessibilidade, desde a concepção do projeto e não como um requisito adicional.

Apesar dos recursos disponíveis e da crescente conscientização sobre a importância da acessibilidade, muitos aplicativos Android ainda apresentam barreiras de acessibilidade recorrentes. Essas barreiras, contudo, possuem soluções relativamente simples com os recursos atuais disponíveis, sugerindo que a implementação de práticas que conduzam à acessibilidade pode ser mais disseminada e efetiva.

É fundamental oferecer alternativas tangíveis para o desenvolvimento de aplicativos acessíveis, proporcionando aos profissionais da área ferramentas e conhecimentos que possibilitem a criação de soluções inclusivas. Neste contexto, o presente documento desempenha um papel relevante ao apresentar, de maneira prática, a utilização dos recursos existentes para solucionar os problemas mais comuns de acessibilidade em aplicativos Android. A partir do estabelecimento de prioridades, com base na literatura, são oferecidas tarefas passo-a-passo com o objetivo de capacitar os desenvolvedores na promoção de aplicativos Android nativos, com especial foco nos usuários com deficiência visual.

Este tutorial também contribui para a disseminação e fomento da acessibilidade

na comunidade de desenvolvedores. Ao oferecer uma abordagem prática e aplicável, que busca influenciar positivamente a mentalidade e as práticas dos profissionais de tecnologia.

## Referências

- [1] BRASIL. Lei nº 13.146, de 6 de julho de 2015 (lei brasileira de inclusão da pessoa com deficiência / estatuto da pessoa com deficiência), 7 2015. URL <http://www2.camara.leg.br/legin/fed/lei/2015/lei-13146-6-julho-2015-781174-normaatualizada-pl.pdf>.
- [2] S. Yan and P. G. Ramachandran. The current status of accessibility in mobile apps. *ACM Transactions on Accessible Computing*, 12(1), 2 2019. ISSN 19367228. doi: 10.1145/3300176. URL <https://doi.org/10.1145/3300176>.
- [3] Alberto Dumont Alves Oliveira, Paulo Sérgio Henrique Dos Santos, Wilson Estécio Marcílio Júnior, Wajdi M Aljedaani, Danilo Medeiros Eler, and Marcelo Medeiros Eler. Analyzing accessibility reviews associated with visual disabilities or eye conditions. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, CHI '23, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9781450394215. doi: 10.1145/3544548.3581315. URL <https://doi.org/10.1145/3544548.3581315>.
- [4] Yavuz Inal, Kerem Rızvanoğlu, and Yeliz Yesilada. Web accessibility in Turkey: awareness, understanding and practices of user experience professionals. *Universal Access in the Information Society*, 18(2):387–398, June 2019. ISSN 1615-5297. doi: 10.1007/s10209-017-0603-3. URL <https://doi.org/10.1007/s10209-017-0603-3>.
- [5] M. V. Rodrigues Leite, L. P. Scatalon, A. P. Freire, and M. M. Eler. Accessibility in the mobile development industry in brazil: Awareness, knowledge, adoption, motivations and barriers. *Journal of Systems and Software*, 177:110942, 7 2021. ISSN 0164-1212. doi: 10.1016/J.JSS.2021.110942.
- [6] T. Bi, X. Xia, D. Lo, J. Grundy, T. Zimmermann, and D. Ford. Accessibility in software practice: A practitioner’s perspective. *Trans. on Software Engineering and Methodology*, 31:1–26, 10 2022. ISSN 1049-331X. doi: 10.1145/3503508. URL <https://dl.acm.org/doi/10.1145/3503508>.
- [7] Darliane Miranda and João Araujo. Studying industry practices of accessibility requirements in agile development. In *Proc of the 37th ACM/SIGAPP Symposium on Applied Computing*, volume 10, page 1309–1317. Association for Computing Machinery, 4 2022. ISBN 9781450387132. doi: 10.1145/3477314.3507041. URL <https://doi.org/10.1145/3477314.3507041>.
- [8] ISO 9241-11. Ergonomics of human-system interaction – part 11: Usability: Definitions and concepts. Technical report, ISO - International Organization for Standardization, 2018.

- [9] Helen Petrie, Andreas Savva, and Christopher Power. Towards a unified definition of web accessibility. In *Proc of the 12th {Web} for {All} {Conference}*, pages 35:1–35:13. ACM, 2015. ISBN 978-1-4503-3342-9. doi: 10.1145/2745555.2746653. URL <http://doi.acm.org/10.1145/2745555.2746653>.
- [10] Amina Bouraoui and Imen Gharbi. Model driven engineering of accessible and multi-platform graphical user interfaces by parameterized model transformations. *Science of Computer Programming*, 172:63–101, 2019. ISSN 01676423. doi: 10.1016/j.scico.2018.11.002. URL <https://linkinghub.elsevier.com/retrieve/pii/S0167642318304301>.
- [11] Daniel Dardailler. Wai early days, 2009. URL <https://www.w3.org/WAI/history>.
- [12] W3C. Web content accessibility guidelines (wcag) 2.1, 2018. URL <https://www.w3.org/TR/WCAG21/#identify-input-purpose>.
- [13] Claurton Siebra, Tatiana B Gouveia, Jefte Macedo, Fabio Q B Da Silva, Andre L M Santos, Walter Correia, Marcelo Penha, Fabiana Florentin, and Marcelo Anjos. Toward accessibility with usability: Understanding the requirements of impaired uses in the mobile context. In *Proc of the 11th International Conference on Ubiquitous Information Management and Communication, IMCOM 2017*, pages 6:1–6:8, New York, NY, USA, 2017. ACM. ISBN 9781450348881. doi: 10.1145/3022227.3022233. URL <http://doi.acm.org/10.1145/3022227.3022233>.
- [14] BBC. Accessibility - bbc, 2022. URL <https://www.bbc.co.uk/accessibility/>.
- [15] A. Alshayban, I. Ahmed, and S. Malek. Accessibility issues in android apps: State of affairs, sentiments, and ways forward. In *Proc of the 42nd Intl Conf on Software Engineering, ICSE '20*, page 1323–1334, New York, NY, USA, 2020. ACM. ISBN 9781450371216. doi: 10.1145/3377811.3380392. URL <https://doi.org/10.1145/3377811.3380392>.
- [16] Anderson Canale Garcia, Silvana Maria Affonso de Lara, Lianna Mara Castro Duarte, Renata Pontin de Mattos Fortes, and Kamila Rios Da Hora Rodrigues. Early accessibility testing – an automated kit for android developers. In *Proceedings of the 29th Brazilian Symposium on Multimedia and the Web, WebMedia '23*, page 11–15, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400709081. doi: 10.1145/3617023.3617028. URL <https://doi.org/10.1145/3617023.3617028>.
- [17] Lisa Crispin and Janet Gregory. *Agile testing: A practical guide for testers and agile teams*. Pearson Education, Boston, MA, 2009. ISBN 978-0-321-53446-0.
- [18] Márcio Eduardo Delamaro, José Carlos Maldonado, and Mário Jino. *Introdução ao teste de software*. Elsevier, Rio de Janeiro, 2016. ISBN 978-85-352-8352-5.
- [19] Google. Developer guides, 2023. URL <https://developer.android.com/guide/>.

- [20] M Aniche. *Real World Test-Driven Development*. Casa do Código, São Paulo, 2014. ISBN 9788566250572.
- [21] Google. Fundamentals of testing android apps, 2023. URL <https://developer.android.com/training/testing/fundamentals>.
- [22] Google. Write automated tests with ui automator, 2023. URL <https://developer.android.com/training/testing/ui-testing/uiautomator-testing>.
- [23] Google. Espresso, 2023. URL <https://developer.android.com/training/testing/espresso>.
- [24] Google. Robolectric. <https://github.com/robolectric/robolectric>, 2023.
- [25] Google. Improve your code with lint checks, 2023. URL <https://developer.android.com/studio/write/lint>.
- [26] Shuai Hao, Bin Liu, Suman Nath, William G.J. Halfond, and Ramesh Govindan. Puma: Programmable ui-automation for large-scale dynamic analysis of mobile apps. In *Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services, MobiSys '14*, page 204–217, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450327930. doi: 10.1145/2594368.2594390. URL <https://doi.org/10.1145/2594368.2594390>.
- [27] Navid Salehnamadi, Abdulaziz Alshayban, Jun-Wei Lin, Iftekhar Ahmed, Stacy Branham, and Sam Malek. Latte: Use-case and assistive-service driven automated accessibility testing framework for android. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems, CHI '21*, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450380966. doi: 10.1145/3411764.3445455. URL <https://doi.org/10.1145/3411764.3445455>.
- [28] Aaron R. Vontell. *Bility : Automated Accessibility Testing for Mobile Applications*. PhD thesis, Massachusetts Institute of Technology, 2019. URL <https://dspace.mit.edu/handle/1721.1/121685>.
- [29] M. M. Eler, J. M. Rojas, Yan Ge, and G. Fraser. Automated accessibility testing of mobile apps. In *2018 IEEE 11th Intl Conf on Software Testing, Verification and Validation (ICST)*, pages 116–126, Västerås, Sweden, 5 2018. IEEE Inc. ISBN 9781538650127. doi: 10.1109/ICST.2018.00021.
- [30] Arthur Floriano Barbosa Andrade de Oliveira and Lucia Vilela Leite Filgueiras. Accessilint: A tool for early accessibility verification for android native applications. In *Procngs of the 18th Brazilian Symposium on Human Factors in Computing Systems*. Association for Computing Machinery, 2019. ISBN 9781450369718. doi: 10.1145/3357155.3360474. URL <https://doi.org/10.1145/3357155.3360474>.



## Capítulo

# 2

## Desenvolvendo Aplicativos Android usando Kotlin

Larissa Cardoso Zimmermann<sup>1</sup>, Juliana Martins Leônico Eusébio<sup>1</sup>,  
Maria da Graça Campos Pimentel<sup>1</sup>

<sup>1</sup> Universidade de São Paulo. Instituto de Ciências Matemáticas e de Computação.

larissa.zimmermann@usp.br, julianaleoncio@usp.br,  
mcp@icmc.usp.br

### *Abstrat*

*In this mini-course, the development of applications for Android platform using the Kotlin programming language will be introduced. Essential concepts for native software development for Android platform will be presented concurrently with practical activities. Participants of the mini-course should have access to computers with Android Studio installed and prior knowledge of some programming language. Firstly, the creation of a "Hello World" application will be introduced to familiarize participants with Android Studio. Secondly, the Android Studio Layout Editor will be presented, briefly teaching how to create interfaces with user inputs and buttons. Following that, the concept of the lifecycle of Activities and Fragments will be introduced, aiming to learn how to manage lifecycle events with more organized and maintainable code. Finally, with the goal of managing data in the face of configuration changes of applications for Android, the architecture components ViewModel and LiveData will be taught. These components will be linked to data in order to simplify layout views and eliminate the need for click handlers in the interface controllers. Therefore, by the end of this mini-course, participants will be capable of creating applications for the Android platform with a simplified architecture, configuring interface elements, and maintaining data persistence during application configuration changes.*

## **Resumo**

*Neste minicurso de tratamento prático, será introduzido o desenvolvimento de aplicativos para a plataforma Android utilizando a linguagem de programação Kotlin. Serão apresentados conceitos essenciais para o desenvolvimento de software nativo para a plataforma Android concomitante com atividades práticas. Para isso, os participantes do minicurso deverão ter acesso a computadores com a plataforma Android Studio instalada e conhecimento prévio de alguma linguagem de programação. Em primeiro lugar, será introduzido como criar um aplicativo “Hello World”, com o objetivo de familiarizar os participantes com a plataforma Android Studio. Em segundo lugar, apresentaremos o Editor de Layouts do Android Studio, com o objetivo de ensinar brevemente como criar interfaces, com entradas dos usuários e botões. Em sequência, o conceito de ciclo de vida de Atividades e de Fragmentos será apresentado, com a finalidade de aprender a gerenciar eventos do ciclo de vida com códigos mais organizados e fáceis de manter. Por fim, com o objetivo de gerenciar os dados mediante as mudanças de configuração de aplicativos para plataforma Android, serão ensinados os componentes de arquitetura ViewModel e LiveData. Estes componentes serão vinculados aos dados para que as visualizações do layout sejam simplificadas e elimine a necessidade de gerenciadores de cliques nos controladores da interface. Portanto, ao final deste minicurso, os participantes estarão aptos a criar aplicativos para a plataforma Android, com arquitetura simplificada, configurando elementos de interface e mantendo a persistência dos dados durante mudanças de configurações do aplicativo.*

### **2.1. Introdução**

A introdução ao minicurso de Android com Kotlin representa a porta de entrada para o desenvolvimento de aplicativos móveis e para o mundo da computação ubíqua. Em um cenário tecnológico em constante evolução, a criação de aplicativos para a plataforma Android se destaca como uma habilidade essencial. Este curso propõe-se a oferecer uma compreensão prática e abrangente, capacitando os participantes a dominar as nuances do desenvolvimento de *software* para dispositivos Android utilizando a linguagem de programação Kotlin.

A escolha do Android como plataforma é motivada pela sua presença ubíqua em dispositivos móveis, abrangendo desde *smartphones* até *tablets*, *smartwatches* e TVs. A capacidade de criar aplicativos para essa plataforma oferece não apenas uma oportunidade de inovação, mas também a chance de impactar a vida cotidiana dos usuários em escala global.

O minicurso inicia-se com a configuração do ambiente de desenvolvimento no Android Studio, a ferramenta oficial do Google para criação de aplicativos Android. Os participantes serão guiados passo a passo, desde a instalação até a compreensão das principais funcionalidades, preparando o terreno para uma jornada prática e imersiva.

O marco inicial, a criação do aplicativo "*Hello World*", não é apenas uma formalidade; é um convite para mergulhar no código, compreender a estrutura básica e, mais importante, sentir a empolgação de dar vida a um aplicativo funcional.

A exploração do Editor de *Layouts* do Android Studio oferece a oportunidade de aprimorar a estética e a usabilidade dos aplicativos, garantindo que os desenvolvedores possam criar interfaces atraentes e intuitivas.

À medida que avançamos, mergulharemos nas complexidades do ciclo de vida das atividades e fragmentos, destacando a importância de gerenciar eventos de maneira eficaz. Esse conhecimento não apenas melhora a estabilidade dos aplicativos, mas também aprimora a experiência do usuário.

A implementação da arquitetura *ViewModel* e *LiveData*, conceitos fundamentais do desenvolvimento Android moderno, proporcionará aos participantes ferramentas poderosas para simplificar o código, gerenciar dados de maneira eficiente e criar aplicativos robustos e escaláveis.

Dessa forma, este minicurso não é apenas uma oportunidade de aprendizado; é um convite para desbravar o excitante campo do desenvolvimento de aplicativos Android com Kotlin, abrindo portas para a inovação e a excelência técnica.

## 2.2. Motivação

A motivação do curso de Desenvolvimento de Aplicativos Android usando Kotlin é alimentada por uma série de fatores que convergem para criar uma experiência de aprendizado enriquecedora e prática. Este curso busca inspirar e capacitar os participantes a se destacarem no desenvolvimento de aplicativos para a plataforma Android, utilizando a linguagem de programação Kotlin.

- Relevância do Android:

O sistema operacional Android é onipresente em dispositivos móveis em todo o mundo. A capacidade de desenvolver aplicativos para essa plataforma oferece oportunidades significativas, pois os aplicativos Android desempenham um papel vital na vida cotidiana, abrangendo desde comunicação até entretenimento e produtividade.

- Kotlin como linguagem de programação para desenvolvimento de aplicativos Android:

A escolha da linguagem de programação Kotlin destaca-se por sua modernidade, expressividade e preferência no ecossistema Android. A transição para Kotlin é uma resposta à demanda crescente por uma linguagem mais concisa e segura, proporcionando aos participantes uma habilidade altamente valorizada no mercado de desenvolvimento de software. Kotlin é a linguagem de programação oficial do Google para o desenvolvimento de aplicativos Android.

- Abordagem Prática:

O curso adota uma abordagem prática desde o início, com a criação do aplicativo "*Hello World*". Essa metodologia permite que os participantes não apenas absorvam conceitos teóricos, mas também coloquem imediatamente em prática o que aprendem, proporcionando uma experiência de aprendizado mais envolvente e eficaz.

- Domínio do Ambiente de Desenvolvimento:

A configuração inicial no Android Studio é uma peça fundamental do curso. Os participantes não apenas aprendem a criar aplicativos, mas também se tornam proficientes no uso da principal ferramenta de desenvolvimento para Android, preparando-os para desafios do mundo real.

- Exploração de Tecnologias Avançadas:

O curso não se limita ao básico; ele avança para conceitos avançados, como o ciclo de vida de atividades e fragmentos, bem como a implementação da arquitetura *ViewModel* e *LiveData*, para persistência e consistência dos dados. Essas tecnologias representam as melhores práticas do desenvolvimento moderno de aplicativos Android.

- Relevância Profissional:

A demanda por desenvolvedores de aplicativos Android qualificados e versados em Kotlin é alta. Este curso visa preparar os participantes não apenas para criar aplicativos, mas também para se destacarem no mercado de trabalho, proporcionando-lhes habilidades práticas e aplicáveis. Portanto, trata-se de um curso extracurricular importante para quem quer se destacar na área de desenvolvimento de aplicativos e computação ubíqua.

Assim, a motivação intrínseca do curso reside na oportunidade de capacitar os participantes a se tornarem a terem uma visão geral do desenvolvimento de aplicativos Android, com conhecimentos práticos e uma base sólida para se aprofundarem na área.

### 2.3. Público-alvo

O público-alvo deste curso de Android com Kotlin é diversificado, abrangendo desde iniciantes entusiastas da programação até desenvolvedores mais experientes que desejam expandir suas habilidades no desenvolvimento de aplicativos móveis. O curso é projetado para ser inclusivo e oferecer benefícios significativos a diferentes perfis de participantes.

- Iniciantes em Programação:

Este curso é adequado para indivíduos que estão dando os primeiros passos na programação. A abordagem prática, começando com a criação do "*Hello World*", permite uma entrada suave para o mundo da programação e do desenvolvimento de aplicativos.

- Discentes de Ciência da Computação e Engenharia de Software:

Discentes que buscam complementar seus estudos com uma compreensão prática do desenvolvimento de aplicativos para dispositivos móveis. O curso fornece uma base sólida para projetos futuros e experiência relevante para carreiras na área.

- Desenvolvedores Iniciantes em Android:

Profissionais que têm alguma experiência em desenvolvimento, mas desejam migrar ou expandir para o desenvolvimento Android com Kotlin. O curso oferece uma transição

suave, cobrindo conceitos fundamentais e avançados específicos para o ecossistema Android.

- **Desenvolvedores Experientes:**

Desenvolvedores que já têm experiência em Android, mas desejam atualizar suas habilidades para incluir Kotlin e adotar práticas modernas de arquitetura. O curso oferece uma oportunidade de aprimoramento profissional, explorando conceitos avançados e técnicas atualizadas.

- **Pesquisadores e Criadores Independentes:**

Pessoas que desejam desenvolver suas próprias ideias de aplicativos ou aprimorar suas habilidades para participar ativamente no desenvolvimento de seus projetos de pesquisa. O curso proporciona conhecimentos práticos para transformar conceitos em aplicativos funcionais.

Em resumo, o curso é projetado para atender a uma ampla gama de participantes, independentemente do nível de experiência em programação ou desenvolvimento Android. A abordagem prática e os conceitos explorados proporcionam uma base sólida para o crescimento profissional e a inovação na área de desenvolvimento de aplicativos móveis.

## 2.4. Pré-requisitos

Para participar do curso de Desenvolvimento de Aplicativos Android usando Kotlin, os participantes devem atender a alguns pré-requisitos fundamentais para garantir uma experiência de aprendizado eficaz e bem-sucedida. Estes pré-requisitos visam proporcionar uma base mínima de conhecimento e habilidades necessárias para acompanhar o conteúdo do curso:

- **Conhecimento Básico em Programação:**

Os participantes devem ter conhecimentos básicos em lógica de programação e compreender conceitos fundamentais, como variáveis, estruturas de controle de fluxo (if, else, loops) e estruturas de dados simples.

- **Familiaridade com Alguma Linguagem de Programação:**

Embora não seja obrigatório ter experiência prévia em Kotlin, é recomendável que os participantes tenham alguma familiaridade com pelo menos uma linguagem de programação, como Java, C++, Python ou JavaScript.

- **Ambiente de Desenvolvimento Android Studio Instalado:**

Os participantes devem ter o Android Studio instalado em seus computadores antes do início do curso. Instruções sobre como realizar essa instalação podem ser fornecidas antecipadamente.

- **Compreensão Básica de Desenvolvimento de Software:**

É útil que os participantes tenham uma compreensão básica do ciclo de vida do desenvolvimento de software, incluindo a criação, teste e depuração de aplicativos.

- Disponibilidade de um Dispositivo Android para Testes:

Para obter uma experiência prática completa, os participantes devem ter acesso a um dispositivo Android para testar os aplicativos desenvolvidos durante o curso. Em alternativa, podem utilizar emuladores fornecidos pelo Android Studio.

Estes pré-requisitos são projetados para garantir que os participantes tenham uma base mínima de conhecimentos em programação e estejam prontos para se envolverem nas atividades práticas propostas pelo curso. Aqueles que atendem a esses pré-requisitos estarão mais bem preparados para absorver os conceitos apresentados e aplicá-los no desenvolvimento de aplicativos Android com Kotlin.

## **2.5. Estrutura em tópicos**

O minicurso é organizado em quatro tópicos principais.

### **1. INTRODUÇÃO AO ANDROID STUDIO E CRIAÇÃO DO PRIMEIRO APP**

O objetivo desta atividade é ensinar a configurar o Android Studio para usar o Kotlin e a criar um app. Você começará com o “Hello World”, evoluindo até um app que usa arquivos de imagem e um gerenciador de cliques. Você verá como os projetos do Android são estruturados, como usar e modificar visualizações no seu app Kotlin para Android e como garantir que seus apps sejam compatíveis com versões anteriores. Você também aprenderá sobre os níveis de API e as bibliotecas do Android Jetpack.

### **2. ENTENDENDO O EDITOR DE LAYOUT DO ANDROID STUDIO**

Nesta etapa, os participantes aprenderão a usar o Editor de Layout do Android Studio para criar layouts lineares e restritos. Os participantes criarão apps que recebem e exibem entradas do usuário, respondem a toques e mudam a visibilidade e a cor das visualizações. Esta lição também ensina a usar a vinculação de dados a elementos de interface.

### **3. CICLO DE VIDA DE ATIVIDADES E FRAGMENTOS**

Nesta fase do minicurso, serão introduzidos os ciclos de vida de atividades e fragmentos. Será ensinado também como gerenciar situações complexas de ciclos de vida. Os participantes trabalharão com um aplicativo com diversos bugs relacionados ao ciclo de vida do Android. Os participantes entenderão como funciona o ciclo de vida de atividades e fragmentos, além de aprender sobre a biblioteca Lifecycle do Android Jetpack, que pode ajudar a gerenciar eventos do ciclo de vida com códigos mais organizados e fáceis de manter.

#### 4. COMPONENTES DE ARQUITETURA

Por fim, será ensinado a usar os objetos `ViewModel` e `LiveData`. Os participantes utilizarão objetos `ViewModel` para autorizar que os dados sobrevivam a mudanças de configuração, como a rotação da tela. Será ensinado como converter os dados da interface de um aplicativo em um `LiveData` encapsulado e adicionará métodos do observador que são notificados quando o valor do `LiveData` muda. Será possível também integrar o `LiveData` e o `ViewModel` à vinculação de dados para que as visualizações no seu layout se comuniquem diretamente com objetos `ViewModel`. Essa técnica simplifica o código e elimina a necessidade de gerenciadores de cliques nos controladores da IU.

Este é um minicurso teórico e prático. É seguida a metodologia do Google para a apresentação teórica e o desenvolvimento de aplicativos Android com a ferramenta Android Studio. Os códigos empregados nas atividades foram baseados na documentação oficial de Kotlin para Android fornecido pelo Google.

#### 2.6. Configuração inicial do laboratório

Este minicurso deve ser realizado em laboratório com acesso à Internet e estações de trabalho com o Android Studio instalado. As estações de trabalho devem atender aos requisitos técnicos recomendados pelo distribuidor do Android Studio. A utilização de um dispositivo Android e um cabo de carregamento é opcional, pois estes podem ser substituídos por emuladores de dispositivos Android integrados ao Android Studio.

Na indisponibilidade da alocação de um laboratório, há a possibilidade de uma estrutura alternativa. Os participantes podem usar seus notebooks pessoais para realização das atividades. Neste caso, a sala deve oferecer acesso à internet por sem fio e mesas de trabalho para acomodar os participantes de forma confortável. Vale ressaltar que, dada a duração do treinamento, recomendamos que a sala possua tomadas distribuídas próximo as mesas de trabalho. É recomendado também o participante instalar previamente o Android Studio, pois trata-se de uma aplicação cujo processo de instalação é demorado e não há tempo hábil durante o minicurso.

#### 2.7. Conteúdo do minicurso

O minicurso de “Desenvolvendo Aplicativos *Android* usando Kotlin” proporciona aos participantes uma introdução prática e abrangente ao desenvolvimento de aplicativos para a plataforma *Android*. Durante o curso, os tópicos essenciais são cobertos, incluindo a configuração inicial do ambiente de desenvolvimento no Android Studio, a criação do primeiro aplicativo "*Hello World*", a exploração do Editor de *Layouts*, o entendimento do ciclo de vida de atividades e fragmentos, e a implementação da arquitetura *ViewModel* e *LiveData*.

Os participantes aprendem a configurar corretamente o ambiente de desenvolvimento, tornando-se familiares com o Android Studio e Kotlin. O desenvolvimento prático começa com a criação de um aplicativo simples para estabelecer uma base sólida. Em seguida, o curso explora o Editor de *Layouts*, capacitando os participantes a criar interfaces de usuário eficientes.

A compreensão do ciclo de vida de atividades e fragmentos é destacada, proporcionando aos desenvolvedores as habilidades necessárias para gerenciar eventos de forma eficaz. Além disso, a introdução e implementação dos componentes de arquitetura *ViewModel* e *LiveData* simplificam o desenvolvimento, promovendo código organizado e a gestão eficiente de dados, mesmo em situações de mudanças de configuração do aplicativo.

Ao final do minicurso, os participantes devem estar aptos a desenvolver aplicativos *Android* usando Kotlin, com uma compreensão sólida dos conceitos fundamentais, da criação de interfaces à gestão avançada de dados.

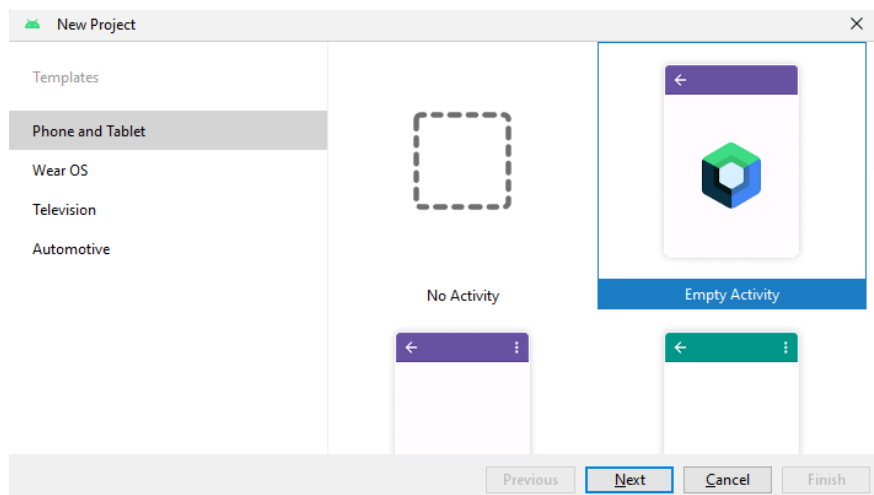
Em seguida, será apresentado o conteúdo do minicurso.

## 1. Introdução ao Android Studio e criação do primeiro APP

### 1.1. Hello World!

Primeiro vamos criar um projeto utilizando um modelo pronto do Android Studio:

- Com o Android Studio instalado e aberto, clique em ***New Project***;
- A janela de modelos será aberta. Confira se a guia ***Phone and Tablet*** está selecionada e então escolha o modelo ***Empty Activity***;



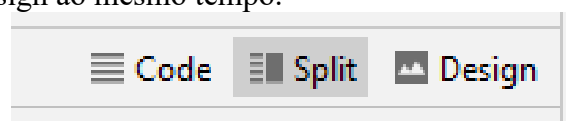
Esse modelo é um projeto simples, que tem uma única tela e mostra o texto "Hello Android!".

- Clique em ***Next***. Você verá alguns campos para configurar o projeto. Eles devem ser preenchidos da seguinte maneira:
  - ***Name*** - é usado para inserir o nome do projeto. Você pode usar "Greeting Card" ou "Cartão de Apresentação" para esse projeto.
  - ***Package name*** - é assim que seus arquivos vão ser organizados na estrutura do arquivo. Deixe esse campo como ele está.

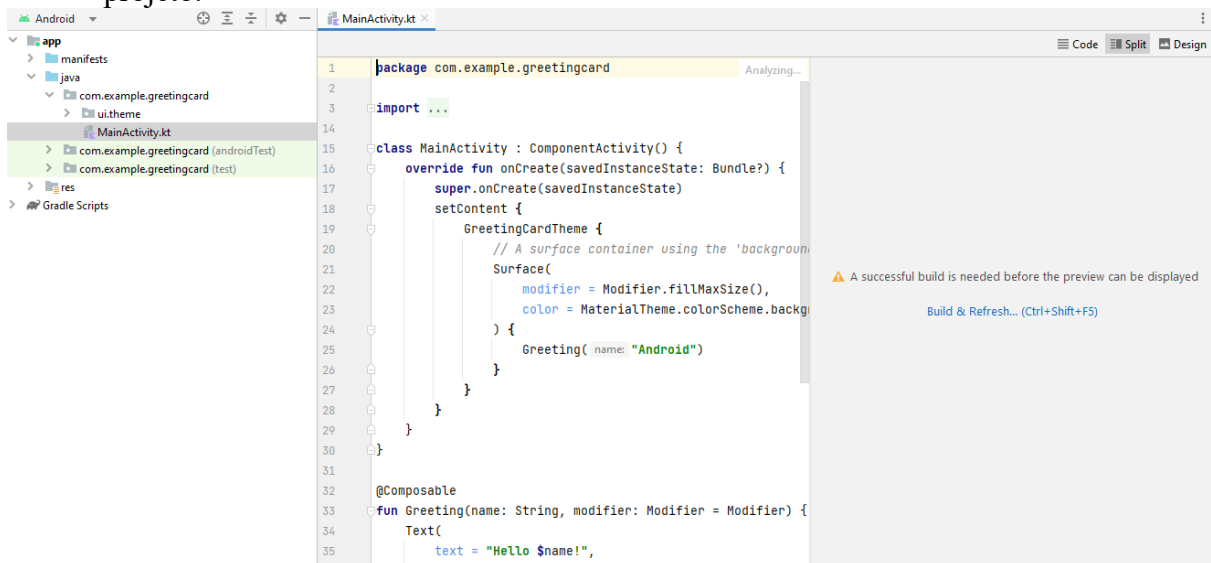


- **Save location** - é o local em que todos os arquivos relacionados ao projeto são salvos. Deixe esse campo como ele está.
- **Language** - é a linguagem de programação utilizada no seu projeto. A linguagem Kotlin já deve estar selecionada.
- **Minimum SDK** - indica a versão mínima do Android em que o seu aplicativo poderá ser executado. Selecione **API 21: Android 5.0 (Lollipop)**.

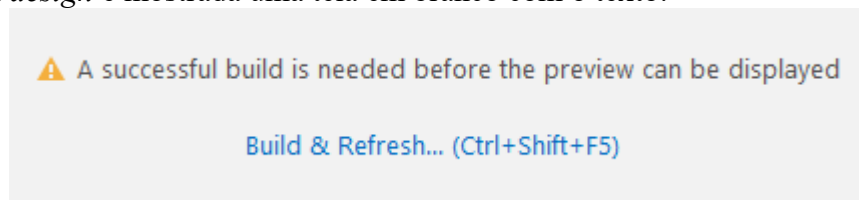
- Selecione **Finish**. Isso talvez demore um pouco. Espere todos os arquivos serem carregados.
- Na parte superior direita, clique em **Split**. Dessa maneira você conseguirá ver o código e o design ao mesmo tempo.



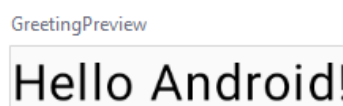
- Perceba que além do código e da aba design, você deve estar vendo uma aba chamada *project* (aba mais a esquerda), que mostra os arquivos e pastas do projeto.



- Na aba *design* é mostrada uma tela em branco com o texto:

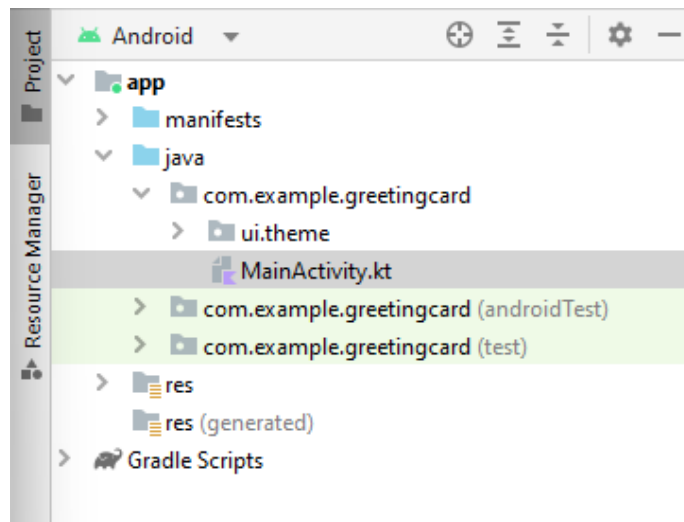


- Clique no botão **Build & Refresh** (a criação pode demorar um pouco), em seguida uma caixa de texto com as palavras “Hello Android!” deve aparecer.

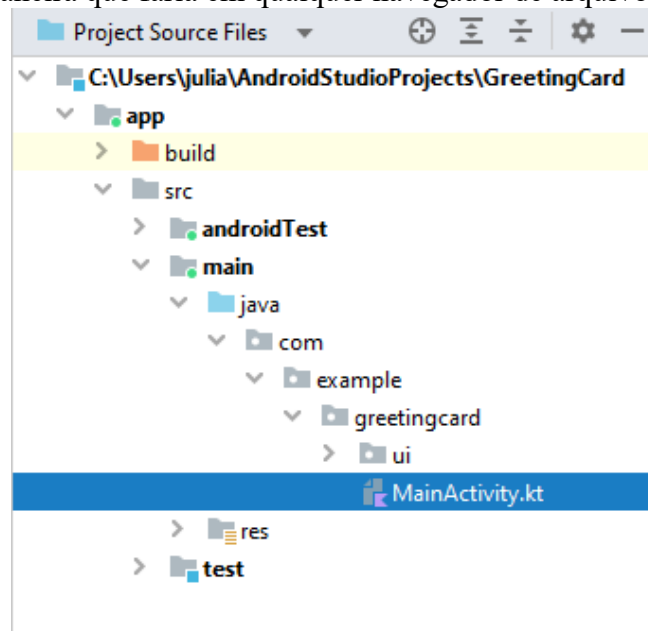


## 1.2. Conhecendo o Android Studio

No Android Studio, veja a guia *project*. Ela mostra os arquivos e as pastas do seu projeto.



Durante a configuração do projeto, escolhemos o nome do pacote como **com.example.greetingcard**. Um pacote é uma pasta em que o código está localizado. Selecione **Project Source Files** no menu suspenso. Dessa maneira você pode procurar arquivos da mesma maneira que faria em qualquer navegador de arquivos.



Selecione **Android** para retornar à visualização anterior.

## 1.3. Personalizando sua Carta de Apresentação

Prestando atenção no código do arquivo **MainActivity.kt**, percebemos algumas funções geradas automaticamente pelo modelo.

```

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView {
            GreetingCardTheme {
                // A surface container using the 'background' color from the theme
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colorScheme.background
                ) {
                    Greeting("Android")
                }
            }
        }
    }
}

```

A função **onCreate()** atua como o ponto de entrada do aplicativo e convoca outras funções para criar a interface do usuário. Em programas escritos em Kotlin, a função **main()** é o local específico no código onde o compilador Kotlin é inicializado. Nos aplicativos Android, a função **onCreate()** desempenha esse papel.

Dentro da função **onCreate()**, a função **setContent()** é utilizada para definir o layout usando funções de composição. Todas as funções que são marcadas com a anotação **@Composable** podem ser chamadas dentro da função **setContent()** ou em outras funções de composição. A anotação informa ao compilador Kotlin que essa função é utilizada pelo *Jetpack Compose* para gerar a interface do usuário.

Uma função de composição recebe uma entrada e gera o que vai ser mostrado na tela.

Observando a função **Greeting()** vemos que ela é uma função de composição e por isso a anotação **@Composable** aparece acima dela.

```

@Composable
fun Greeting(name: String, modifier: Modifier = Modifier) {
    Text(
        text = "Hello $name!",
        modifier = modifier
    )
}

```

Alguns detalhes:

- Nomes de funções **@Composable** usam letras maiúsculas.
- A anotação **@Composable** é adicionada antes da função.

- As funções **@Composable** não podem retornar nada.

No momento, a função **Greeting()** recebe um nome e mostra “*Hello*” para essa pessoa. Para personalizar sua carta de apresentação:

Atualize a função **Greeting()** para apresentar você ao invés de dizer “*Hello*”:

```
@Composable
fun Greeting(name: String, modifier: Modifier = Modifier) {
    Text(
        text = "Hi, my name is $name!",
        modifier = modifier
    )
}
```

Pressione o botão de “atualizar” no canto superior esquerdo do painel de design para que seu novo texto apareça:

GreetingPreview

Hi, my name is Android!

Agora precisamos personalizar o app para mostrar seu nome, e não “Android”. A função **GreetingPreview()** permite ver como o app vai ficar sem que você precise terminar de criá-lo. Para isso, é necessário usar a anotação **@Preview** antes da função.

No nosso caso, a anotação **@Preview** está recebendo o parâmetro “*showBackground*”. Se ele for definido como **true**, um plano de fundo será adicionado à visualização do app. Para atualizar seu nome, basta atualizar a função **GreetingPreview()**. Depois disso, atualize e confira seu Cartão.

```
@Preview(showBackground = true)
@Composable
fun GreetingPreview() {
    GreetingCardTheme {
        Greeting("Juliana")
    }
}
```

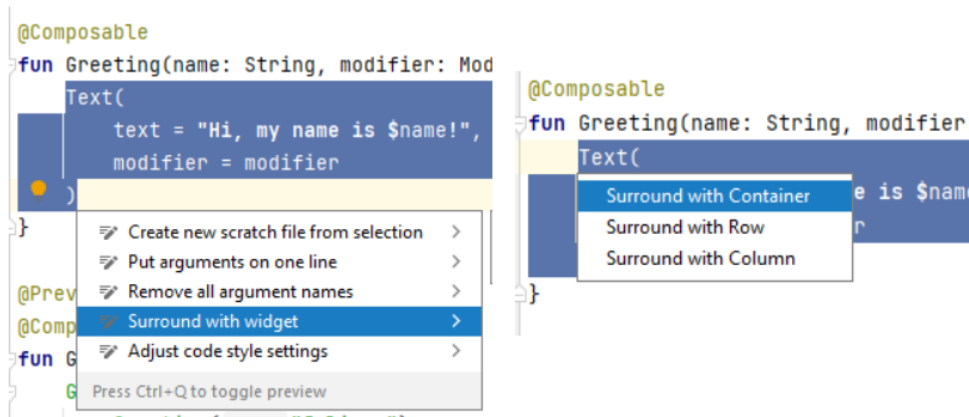
GreetingPreview

Hi, my name is Juliana!

#### 1.4. Mudando a cor do plano de fundo

Para configurar uma cor de fundo diferente no cartão de apresentação, é necessário envolver o texto com uma **Surface**, que é um contêiner que representa uma seção da interface em que você pode mudar a aparência do plano de fundo.

- Para fazer isso, destaque a linha do texto, pressione **Alt + Enter** e selecione **Surround with widget** e então **Surround with Container**.



O contêiner padrão é do tipo **Box**, mas você pode mudá-lo para o tipo que precisar usar:

- Exclua **Box** e digite **Surface()**

```
@Composable
fun Greeting(name: String, modifier: Modifier = Modifier) {
    Surface() {
        Text(
            text = "Hi, my name is $name!",
            modifier = modifier
        )
    }
}
```

- O contêiner **Surface** tem um parâmetro **color**, definido como **Color**.

```
@Composable
fun Greeting(name: String, modifier: Modifier = Modifier) {
    Surface(color = Color) {
        Text(
            text = "Hi, my name is $name!",
            modifier = modifier
        )
    }
}
```

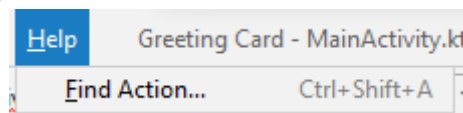
- Ao digitar “Color”, o parâmetro fica vermelho. Para resolver isso, basta ir ao começo do arquivo, abrir a aba de **import**.

```
import ...
```

- E adicionar a instrução abaixo no fim da lista.

```
import androidx.compose.ui.graphics.Color
```

- É recomendado listar as importações em ordem alfabética. Para fazer isso, pressione **Help** na barra de ferramentas, clique em **Find Action...** e digite **Optimize Imports**.



- Voltando ao parâmetro **Color**, percebemos que agora ele está apenas sublinhado em vermelho. Para corrigir esse problema, adicione um ponto depois da palavra **Color**. Uma lista de cores deve aparecer.

```
@Composable
fun Greeting(name: String, modifier: Modifier = Modifier) {
    Surface(color = Color.) {
        Text(
            text = "Hi",
            modifier =
        )
    }
}
```

- Escolha uma cor para a superfície, clique em **Build & Refresh** novamente e veremos o texto ser cercado pela cor escolhida.

GreetingPreview

Hi, my name is Juliana!

### 1.5. Adicionando *Padding*

Agora vamos adicionar um espaço (*padding*) ao redor do texto. Para isso podemos usar um **Modifier**, que é um modificador de elementos. O modificador que usaremos é o **padding**, que aplica espaço ao redor do elemento. Isso é feito usando a função **Modifier.padding()**.

- Adicione as seguintes importações à seção de instruções de importação (não esqueça de otimizar as importações):

```
import androidx.compose.ui.unit.dp
import androidx.compose.foundation.layout.padding
```

- Adicione um *padding* ao texto com tamanho de **24.dp** e clique em ***Build & Refresh***.

```
@Composable
fun Greeting(name: String, modifier: Modifier = Modifier) {
    Surface(color = Color.Magenta) {
        Text(
            text = "Hi, my name is $name!",
            modifier = modifier.padding(24.dp)
        )
    }
}
```

GreetingPreview

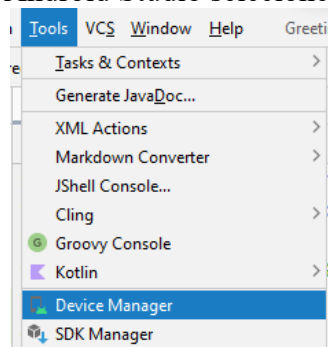


Hi, my name is Juliana!

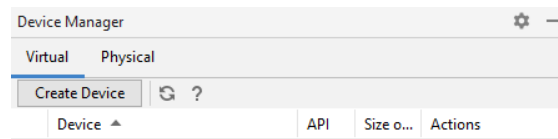
### 1.6. Executando o aplicativo no Android *Emulator*

Agora vamos criar um Dispositivo Virtual Android (AVD) para executar o nosso aplicativo. Um AVD é um emulador de um dispositivo móvel, que é executado no computador. Pode ser qualquer smartphone, tablet, TV, relógio ou dispositivo Android Auto.

- Para criar um AVD, no Android Studio selecione ***Tools > Device Manager***.



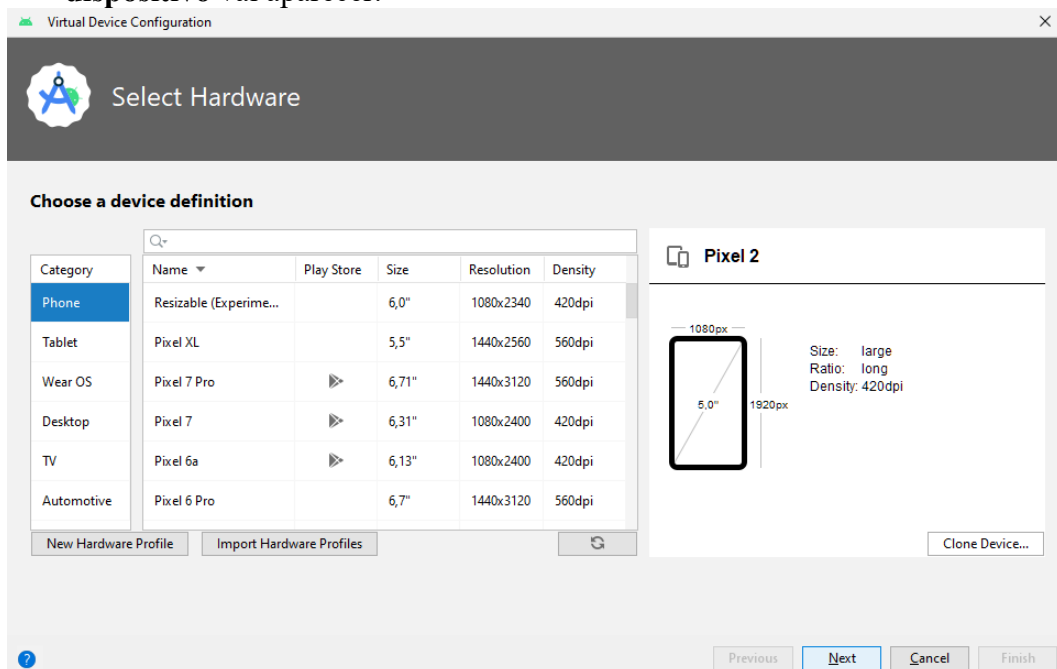
- A caixa de diálogo ***Device Manager*** será aberta.



No virtual devices added. Create a virtual device to test applications without owning a physical device.

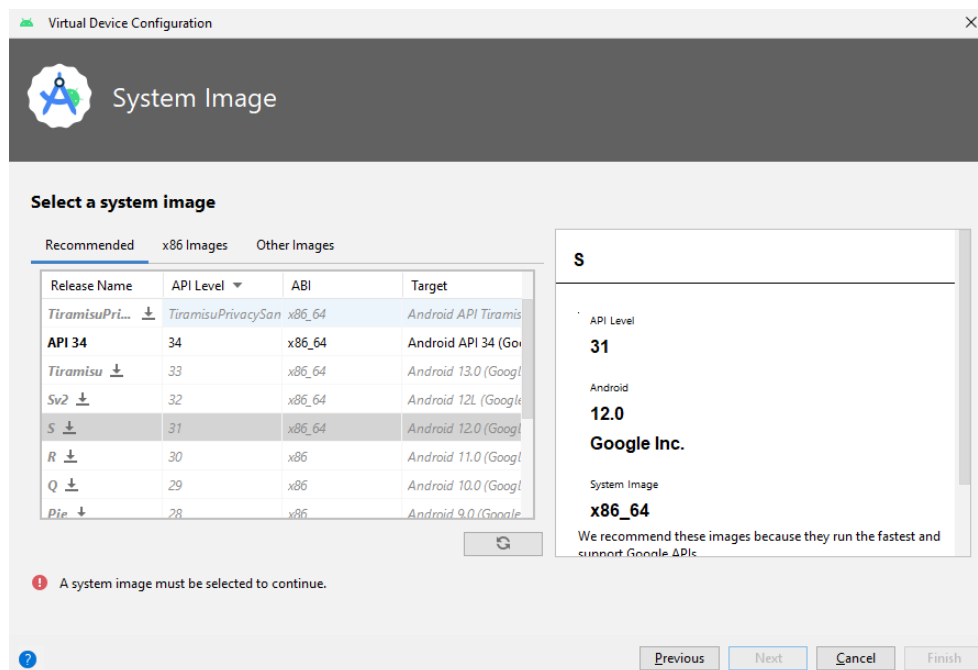
[Create virtual device](#)

- Clique em **Create virtual device**. A caixa de diálogo do **Configurador do dispositivo** vai aparecer.

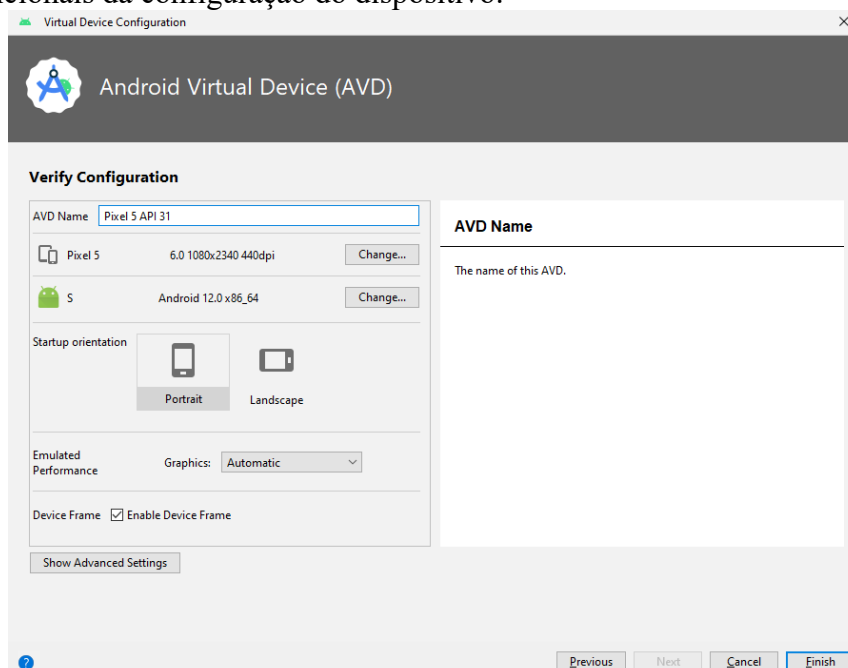


- Selecione **Phone** como a categoria e então selecione um smartphone, como o Pixel 5, e clique em **Next**.
- Agora vamos escolher a versão do Android a ser executada no dispositivo virtual.

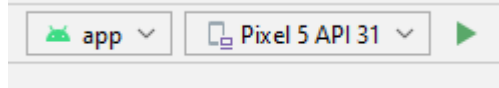




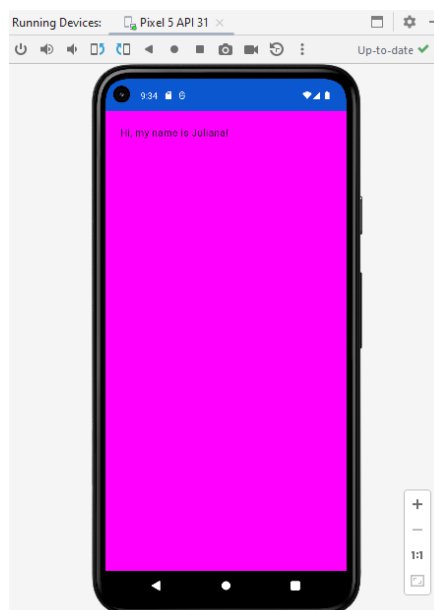
- Caso haja uma indicação de *download* ao lado do S (como mostrado na foto acima), clique para que o download seja efetuado. Isso significa que a imagem não está instalada no seu computador. Aguarde a conclusão do *download*. Isso pode demorar um pouco.
- Após o *download*, selecione S como a versão do Android a ser executada, e clique em *next*. Isso abrirá uma nova tela, em que você pode escolher detalhes adicionais da configuração do dispositivo.



- No campo **AVD Name**, digite um nome para o seu AVD ou use o padrão. Não mude os outros campos. Em seguida, clique em **Finish**.
- Essa ação retorna ao painel do **Device Manager**. Você pode fechá-lo por enquanto.
- Por fim, no menu suspenso na parte superior do Android Studio, selecione o AVD que você criou. E então clique no ícone de **Play** ao lado dele.



O dispositivo virtual será inicializado como um dispositivo físico. Pode levar um tempo até que ele inicialize. O emulador será aberto ao lado do editor de código e ele deve exibir o seu aplicativo.

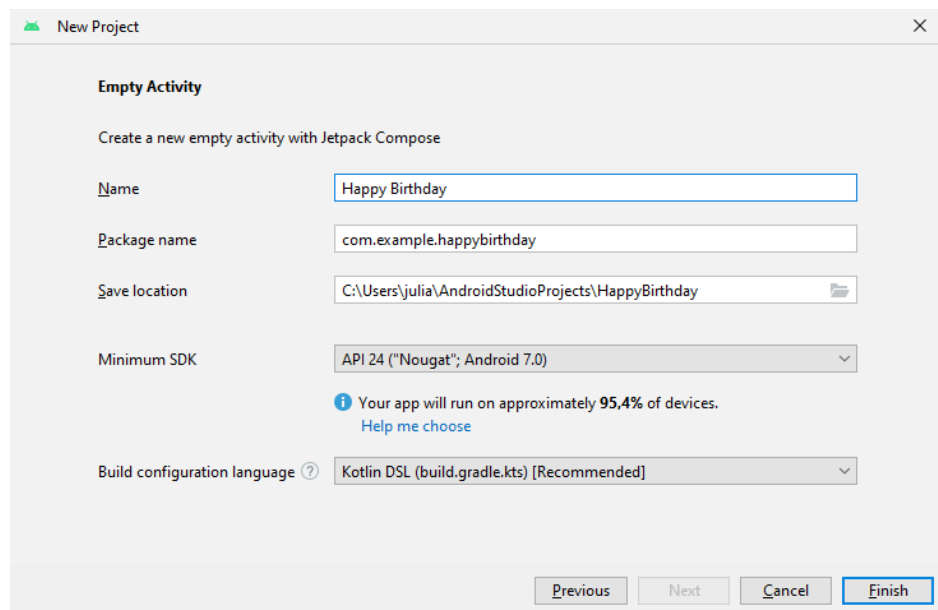



## 2. Entendendo o Editor de *Layout* do Android Studio

### 2.1. Criando o novo projeto

Agora vamos criar um novo projeto para fazer um cartão de aniversário.

- Na página inicial do Android Studio, selecione **New Project** -> **Empty Activity** -> **Next**.
- Dê o nome de **Happy Birthday** para o seu projeto, selecione o **Android 7.0 (Nougat)** no campo **Minimum SDK** e clique em **Finish**.



- Espere o Android Studio criar o projeto e em seguida clique em  **Run**.
- Você verá a mesma tela que vimos no início do capítulo anterior, inteiramente branca com “*Hello Android!*” escrito.

## 2.2. O que é uma interface de usuário?

A interface do usuário de um aplicativo é o que é mostrado na tela: texto, imagens, botões e muitos outros tipos de elementos, além de como eles são organizados na tela. É a forma como o *app* mostra as informações e como o usuário interage com ele.

Quase tudo o que você vê na tela do *app* é chamado de componente da interface. Esses componentes podem ser interativos, como um botão clicável ou um campo de entrada editável, ou podem ser imagens decorativas ou informativas.

Neste projeto vamos trabalhar com o elemento da interface que mostra texto, conhecido como elemento *Text*.

## 2.3. O que é o *Jetpack Compose*?

O *Jetpack Compose* é um kit de ferramentas moderno que simplifica e acelera o desenvolvimento da interface no Android com menos código, ferramentas poderosas e recursos Kotlin intuitivos. Com o Compose, podemos criar a interface definindo um conjunto de funções, conhecidas como **funções combináveis**, que recebem dados e descrevem elementos da interface. Essas funções combináveis não retornam nada, mas geram o que será mostrado na tela.

## Anotações

As anotações são meios de anexar informações extras ao código. Essas informações ajudam o compilador do *Jetpack Compose* e até outros desenvolvedores a entender o código do *app*.


Uma anotação é aplicada utilizando o caractere `@` como prefixo. Vários elementos de código, como propriedades, funções e classes, podem ser anotados. Abaixo temos um exemplo de função com anotação:

```
@Composable
fun Greeting(name: String, modifier: Modifier = Modifier) {...}
```

### Parâmetros

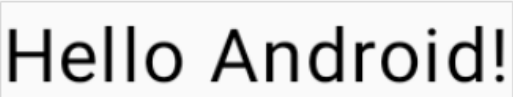
As anotações podem usar parâmetros, que fornecem mais informações para as ferramentas que fazem o processamento delas:

```
@Preview
@Composable
fun GreetingPreview() {
    HappyBirthdayTheme {
        Greeting(name: "Android")
    }
}
```




*Anotação sem parâmetros*

```
@Preview(showBackground = true)
@Composable
fun GreetingPreview() {
    HappyBirthdayTheme {
        Greeting(name: "Android")
    }
}
```



*Anotação que exibe o plano de fundo da prévia*

```
@Preview(name = "My preview")
@Composable
fun GreetingPreview() {
    HappyBirthdayTheme {
        Greeting(name: "Android")
    }
}
```



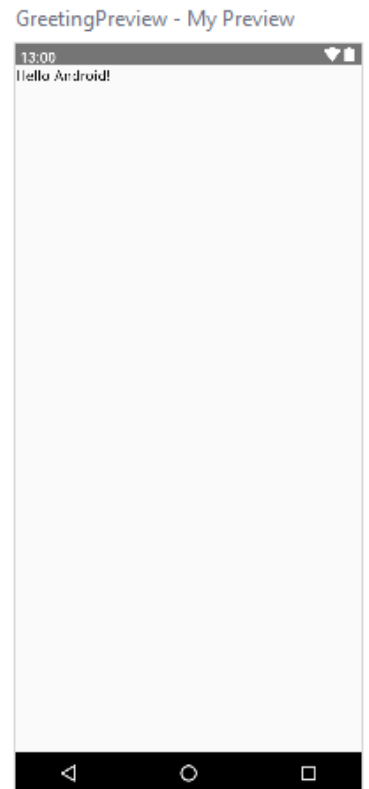
*Anotação com o título da prévia*

É possível transmitir vários parâmetros para a anotação:

```

@Preview(
    showBackground = true,
    showSystemUi = true,
    name = "My Preview"
)
@Composable
fun GreetingPreview() {
    HappyBirthdayTheme {
        Greeting( name: "Android")
    }
}

```



Anotação com título de visualização e interface do sistema (tela do smartphone).

### Funções Combináveis

A função combinável recebe a anotação **@Composable**. Todas as funções desse tipo precisam ter essa anotação. Ela informa ao compilador do Compose que essa função se destina a converter dados em interface.

```

@Composable
fun Greeting(name: String) {
    Text(
        text = "Hello $name!"
    )
}

```

As funções combináveis podem aceitar argumentos. Nesse caso, o elemento da interface aceita uma *String* para que a mensagem use o nome do usuário.

- **Nomes de funções combináveis**

A função de composição que não retorna nada e carrega a anotação **@Composable** **PRECISA** ser nomeada usando o padrão Pascal Case, que se refere a uma convenção de nomenclatura em que a **primeira letra de cada palavra** em uma palavra composta é maiúscula.

A função Compose:

- **PRECISA** ser um substantivo: DoneButton()
- **NÃO** pode ser um verbo ou frase verbal: **Draw**TextField()

- NÃO pode ser uma preposição nominal: `TextFieldWithLink()`
- NÃO pode ser um adjetivo: `Bright()`
- NÃO pode ser um advérbio: `Outside()`
- Substantivos PODEM ter adjetivos descritivos como prefixos: `RoundIcon()`

Além disso, como prática recomendada, as funções devem ser nomeadas para descrever a funcionalidade que elas têm. Portanto:

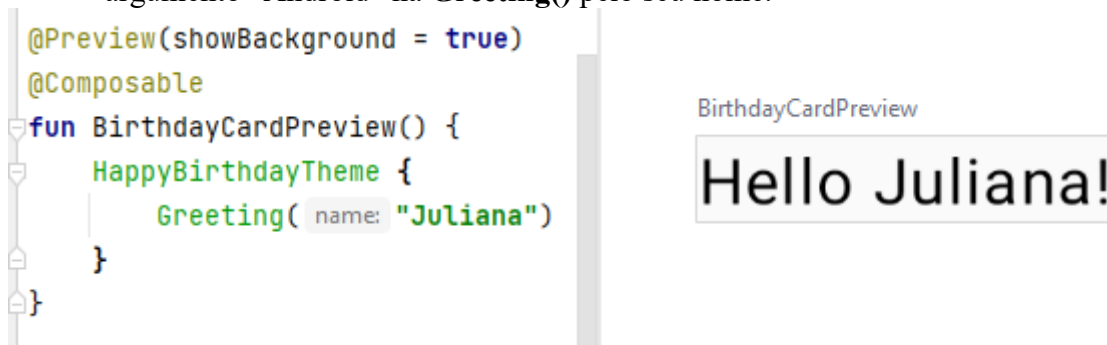
- Role até a função `GreetingPreview()` e mude o nome da função para `BirthdayCardPreview()`.

```
@Preview(showBackground = true)
@Composable
fun BirthdayCardPreview() {
    HappyBirthdayTheme {
        Greeting( name: "Android")
    }
}
```

#### 2.4. Painel *Design* do Android Studio

Anteriormente vimos que é possível visualizar uma prévia da aparência do app no painel *Design* do Android Studio. Para que a prévia possa ser mostrada, a função combinável precisa fornecer valores padrão para todos os parâmetros. Por isso, não é recomendado fazer uma prévia direta da função `Greeting()`. Aí entra a função `BirthdayCardPreview()` que chama a `Greeting()`.

- Vamos conferir uma prévia do seu cartão com seu nome. Para isso, substitua o argumento “Android” na `Greeting()` pelo seu nome.



#### 2.5. Adicionando um novo elemento de texto

Agora vamos remover a saudação `Hello $name!` e adicionar uma mensagem de aniversário no lugar.

- No arquivo `MainActivity.kt`, exclua a definição da função `Greeting()`:

```

@Composable
fun Greeting(name: String, modifier: Modifier = Modifier) {
    Text(
        text = "Hello $name!",
        modifier = modifier
    )
}

```

- Dentro da função **onCreate()**, observe que a chamada da função **Greeting()** agora está em vermelho. Essa cor indica um erro. Passe o cursor sobre essa chamada de função, e o Android Studio vai mostrar informações sobre o erro.

```

) {
    Greeting("Android")
}

```

Unresolved reference: Greeting

Create class 'Greeting' Alt+Shift+Enter    More actions... Alt+Enter

- Exclua as chamadas da função **Greeting()** tanto na função **onCreate()**, quanto na função **BirthdayCardPreview()**. Seu código deverá ficar parecido com este:

```

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView {
            HappyBirthdayTheme {
                // A surface container using the 'background' color from the theme
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colorScheme.background
                ) {
                }
            }
        }
    }
}

@Preview(showBackground = true)
@Composable
fun GreetingPreview() {
    HappyBirthdayTheme {
    }
}

```

- Antes da função **BirthdayCardPreview()**, adicione uma nova função chamada **GreetingText()**. Adicione a anotação **@Composable** antes da função, porque ela será uma função combinável.

```
@Composable
fun GreetingText() {
}
```

- É uma boa prática fazer com que a função combinável aceite um parâmetro **Modifier**. Isso será explicado mais para frente.

```
@Composable
fun GreetingText(modifier: Modifier = Modifier) {
}
```

- Adicione um parâmetro **message** do tipo **String**.

```
@Composable
fun GreetingText(message: String, modifier: Modifier = Modifier) {
}
```

- Dentro da função, adicione uma função combinável **Text** transmitindo a mensagem de texto como um argumento nomeado.

```
@Composable
fun GreetingText(message: String, modifier: Modifier = Modifier) {
    Text(
        text = message
    )
}
```

- Agora a função **GreetingText()** mostra texto na interface. Para visualizar essa função, temos que chamá-la dentro da **BirthdayCardPreview()**.

```
@Preview(showBackground = true)
@Composable
fun GreetingPreview() {
    HappyBirthdayTheme {
        GreetingText(message = "")
    }
}
```

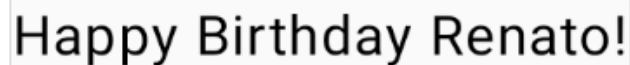
- Dentro do argumento **String** de **message**, você pode escrever a saudação de aniversário que preferir!

```
@Preview(showBackground = true)
```



```
@Composable
fun GreetingPreview() {
    HappyBirthdayTheme {
        GreetingText(message = "Happy Birthday Renato!")
    }
}
```

GreetingPreview



## 2.6. Mudando o tamanho da fonte

Nosso *app* ainda não está com a aparência final. Agora vamos aprender como mudar o tamanho da fonte e a cor do texto. A unidade de medida usada para o tamanho da fonte é de *pixels escalonáveis* (SP, na sigla em inglês).

- No elemento combinável `Text()` na função `GreetingText()`, transmita um argumento `fontSize` à função `Text()` como um segundo argumento nomeado e o defina como um valor de `100.sp`.

```
Text(
    text = message,
    fontSize = 100.sp
)
```

- Android Studio destaca o código `.sp` porque é preciso importar algumas classes ou propriedades para compilar o app. Para isso clique em cima do `.sp` e pressione `Alt+Enter`.
- Observe a prévia atualizada. O motivo dessa sobreposição é porque precisamos especificar também a altura da linha.

GreetingPreview



- Atualize o elemento `Text` para incluir a altura da linha.

```
Text(
    text = message,
    fontSize = 100.sp,
```

```
lineHeight = 116.sp
)
```



Você pode testar diferentes tamanhos de fonte!

## 2.7. Adicionando outro elemento de texto

Agora vamos assinar o cartão, com o nome do remetente.

- Adicione um parâmetro *from* do tipo *String* à função **GreetingText()**.

```
fun GreetingText(message: String, from: String, modifier: Modifier = Modifier)
```

- Após a mensagem de aniversário *Text*, adicione outro elemento *Text* que aceite um argumento *text* definido como o valor *from*.

```
@Composable
fun GreetingText(message: String, from: String, modifier: Modifier = Modifier) {
    Text(
        text = message,
        fontSize = 100.sp,
        lineHeight = 116.sp
    )
    Text(
        text = from
    )
}
```

- Adicione um argumento *fontSize* definido com o valor desejado.

```
Text(
    text = from,
    fontSize = 36.sp
)
```

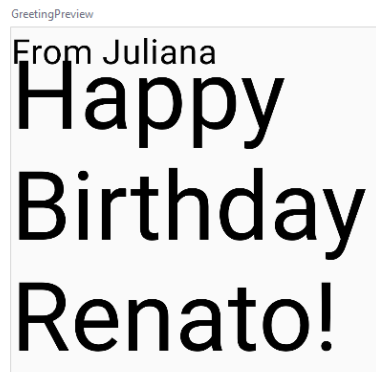
- Na função `BirthdayCardPreview()` adicione outro argumento *String* para assinar o cartão com seu nome.

```

@Preview(showBackground = true)
@Composable
fun GreetingPreview() {
    HappyBirthdayTheme {
        GreetingText(message = "Happy Birthday Renato!", from = "From Juliana")
    }
}

```

Como não especificamos a maneira que queremos que os elementos se organizem, o *Compose* os organizou da própria maneira, e eles acabaram se sobrepondo.



## 2.8. Organizando os elementos do texto

Um componente da interface pode conter um ou mais componentes, e às vezes os termos pai e filho são usados. O contexto é que os elementos pais da interface têm elementos filhos, que por sua vez podem conter outros elementos filhos.

Os três elementos básicos de layout padrão do Compose são *Column*, *Row* e *Box*. Eles são funções combináveis que usam conteúdo de composição como argumentos para que você possa colocar itens dentro desses elementos de layout.

Agora vamos organizar os elementos de texto do nosso app em uma linha, para evitar a sobreposição.

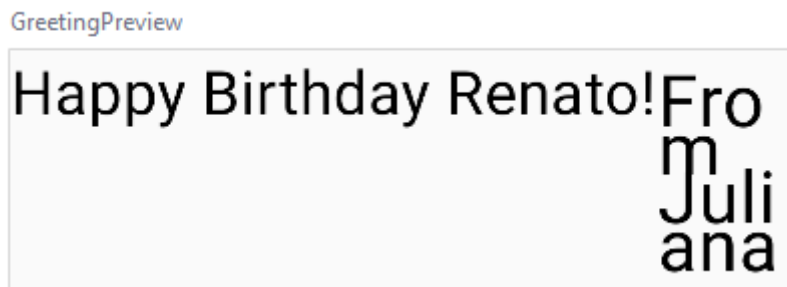
- Na função `GreetingText()`, adicione o elemento combinável *Row* ao redor dos elementos de texto. Para isso selecione os dois elementos *Text* e clique no ícone de lâmpada. Depois, selecione *Surround with widget > Surround with Row*. A função deverá ficar assim:

```

@Composable
fun GreetingText(message: String, from: String, modifier: Modifier = Modifier) {
    Row { this: RowScope
        Text(
            text = message,
            fontSize = 100.sp,
            lineHeight = 116.sp
        )
        Text(
            text = from,
            fontSize = 36.sp
        )
    }
}

```

- Fazendo isso, o Android Studio importa automaticamente a função *Row*. Mude temporariamente o tamanho da fonte da mensagem de aniversário para 30.sp e observe a prévia.



As duas frases não estão mais se sobrepondo, no entanto ainda não temos espaço suficiente para a assinatura. Para resolver isso, vamos organizar os elementos do texto em uma coluna.

- Para fazer isso, basta realizar o mesmo passo a passo anterior, mas utilizar *column* ao invés de *row*. Tente você mesmo! Seu código deverá ficar assim.

```

@Composable
fun GreetingText(message: String, from: String, modifier: Modifier = Modifier) {
    Column { this: ColumnScope
        Text(
            text = message,
            fontSize = 100.sp,
            lineHeight = 116.sp
        )
        Text(
            text = from,
            fontSize = 36.sp
        )
    }
}

```

Com isso, temos o resultado:



Obs: É recomendável transmitir os atributos de modificador junto ao modificador do elemento combinável pai.

### 2.9. Adicionando a saudação ao *APP*

Quando estiver contente com a prévia, é hora de adicionar o elemento combinável ao app no seu dispositivo ou emulador.

- Na função `onCreate()`, chame a função `GreetingText()` do bloco `Surface`, e transmita a mensagem de aniversário. Sua função deve ficar parecida com isso:

```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView {
            HappyBirthdayTheme {
                // A surface container using the 'background' color from the theme
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colorScheme.background
                ) {
                    GreetingText(message = "Happy Birthday Renato!", from = "From Juliana" )
                }
            }
        }
    }
}
```

- Execute o *APP* no emulador.



- Se você quiser, podemos centralizar a saudação. Para isso, adicione um parâmetro chamado **verticalArrangement** e defina-o como **Arrangement.Center**.

```
@Composable
fun GreetingText(message: String, from: String, modifier: Modifier = Modifier) {
    Column(
        verticalArrangement = Arrangement.Center
    ){
        //...
    }
}
```

- Em seguida, centralize o texto da saudação, utilizando **textAlign**.

```
Text(
    text = message,
    fontSize = 100.sp,
    lineHeight = 116.sp,
    textAlign = TextAlign.Center
)
```

GreetingPreview

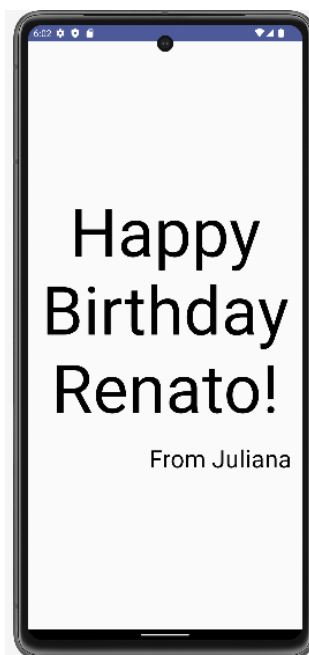
# Happy Birthday Renato!

From Juliana

Vemos que somente a mensagem de parabéns está centralizada e que a assinatura está muito próxima à borda. Agora vamos alinhar a assinatura à direita e adicionar um *padding* à ela, para que ela não fique tão grudada.

```
Text(  
  text = from,  
  fontSize = 36.sp,  
  modifier = Modifier  
    .padding(16.dp)  
    .align(alignment = Alignment.End)  
)
```

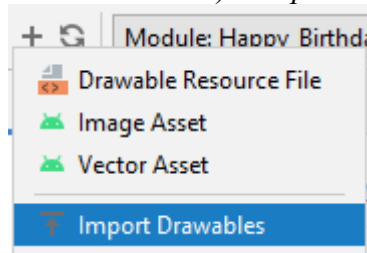
Lembre-se que é recomendável transmitir os atributos de modificador junto ao modificador do elemento combinável pai.



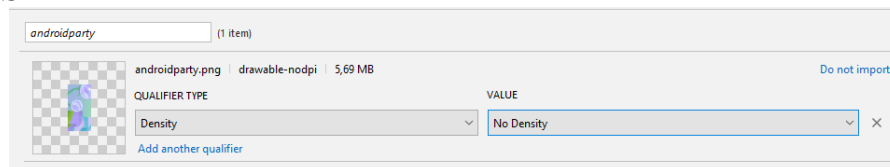
### 3. Adicionando imagens ao APP Android

#### 3.1. Configurando o app

- Baixe uma imagem (arquivo .png) para ser o fundo do seu cartão de aniversário.
- No Android Studio, clique em *View > Tool Windows > Resource Manager*.
- Clique em + (*Add resources to the module*) > *Import Drawables*.

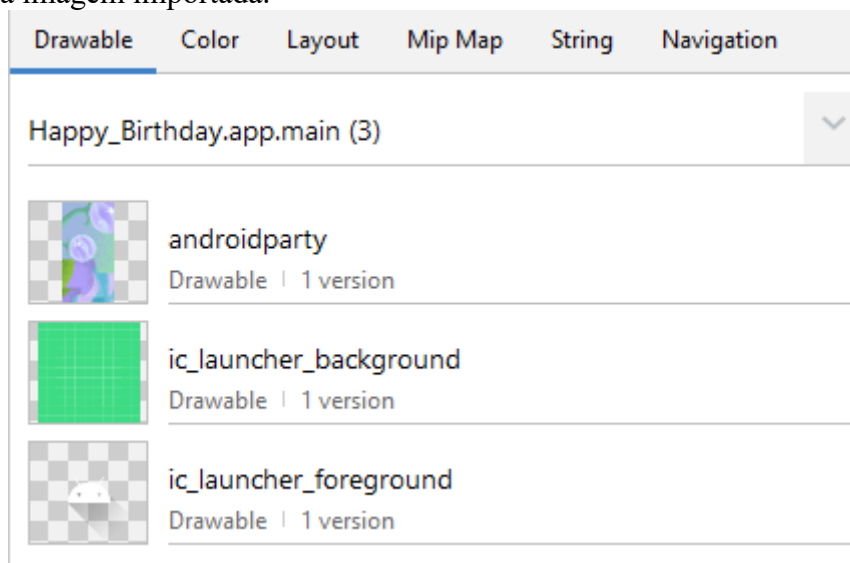


- No navegador de arquivos, selecione o arquivo de imagem que você transferiu por download e clique em Open. Essa ação abre a caixa de diálogo *Import Drawables*.
- Na lista **QUALIFIER TYPE** selecione **Density**, e na lista **VALUE** selecione **NO DENSITY**.



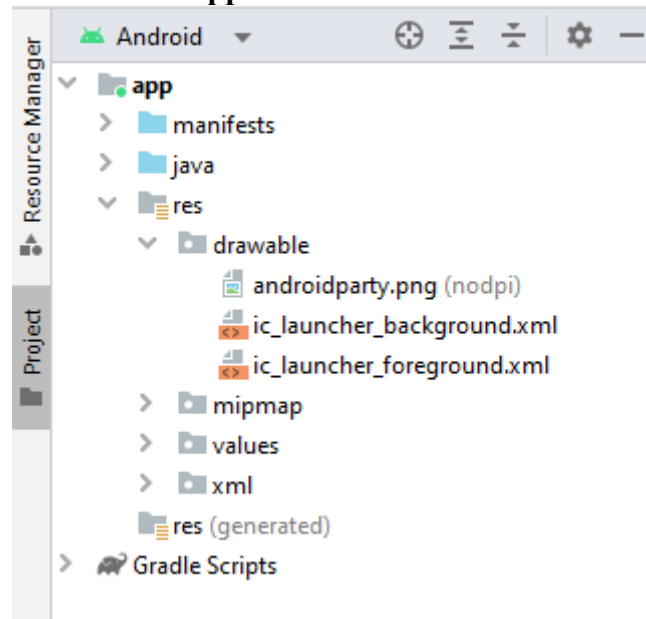
- Clique em Next e em seguida clique em Import.

Caso a imagem seja importada corretamente, ela será adicionada pelo Android Studio à lista na guia **Drawable**. Essa lista inclui todos os ícones e imagens do app, então você já pode usar a imagem importada.





- Na lateral esquerda do Android Studio, volte à aba **Project**, e confirme que a imagem está no caminho **app > res > drawable**.



### 3.2. Incluindo uma função combinável para adicionar uma imagem

Para que uma imagem seja mostrada no app, precisamos informar o local de exibição. Do mesmo modo que usamos um elemento combinável **Text** para mostrar texto, é possível usar um elemento **Image** para mostrar uma imagem.

- No arquivo **MainActivity.kt**, adicione uma função combinável **GreetingImage()** após a função **GreetingText()**.
- Transmita dois parâmetros **String** à ela: o **message** e o **from**. Adicione também o parâmetro **modifier**. Ele informa para um elemento da interface como esses elementos serão dispostos, exibidos ou se comportarão no layout **pai**.

```
@Composable
fun GreetingImage(message: String, from: String, modifier:
Modifier = Modifier){
}

```

- Na função **GreetingImage()**, declare uma propriedade **val** e a nomeie como **image**.
- Faça uma chamada para a função **painterResource()** transmitindo o recurso **androidparty**. Para acessar os recursos é necessário usar as IDs do projeto. No nosso caso, o recurso **androidparty** seria acessado pela ID **R.drawable.graphic**. **R** é a classe gerada automaticamente pelo Android que contém os IDs de todos os recursos do projeto. **Drawable** é o subdiretório dentro da pasta **res** em que a imagem está contida. Atribua o valor retornado à variável **image**.

```
@Composable
fun GreetingImage(message: String, from: String, modifier:
Modifier = Modifier){
    val image = painterResource(R.drawable.androidparty)
}

```

```
}

```

- Após a chamada para a função **painterResource()**, adicione um elemento combinável **Image** e, em seguida, transmita a image para o painter como um argumento nomeado.

```
Image(
    painter = image
)

```

- Adicione as importações necessárias utilizando **Alt + Enter** onde necessário.

Você pode adicionar elementos que melhorem a acessibilidade do seu aplicativo. Um deles é o **contentDescription**, que descreve um elemento para que o seu aplicativo seja melhor utilizado com o *TalkBack*. Como nossa imagem só foi adicionada para fins decorativos, uma descrição da imagem dificultaria o uso com o TalkBack neste caso específico. Então podemos definir o argumento **contentDescription** como **null**, para que o *TalkBack* ignore esse elemento.

```
Image(
    painter = image,
    contentDescription = null
)

```

Para visualizar nosso elemento Image, basta substituir a função **GreetingText()** pela função **GreetingImage()** dentro da **BirthdayCardPreview()**. Seu código deverá ficar parecido com esse:

```
@Preview(showBackground = true)
@Composable
fun BirthdayCardPreview() {
    HappyBirthdayTheme {
        GreetingImage("Happy Birthday Renato!", "From Juliana")
    }
}

```

Com isso, veremos a tela abaixo. Não é mais possível ver o texto, pois a nova função tem apenas um elemento **Image**, mas não tem nenhum **Text**.



### 3.3. Adicionando o layout de caixa

Com o layout **Box** podemos empilhar elementos uns sobre os outros.

- Na função **GreetingImage()**, adicione um elemento **Box** ao redor do elemento **Image**. O código ficará como abaixo:

```
@Composable
fun GreetingImage(message: String, from: String, modifier:
Modifier = Modifier){
    val image = painterResource(R.drawable.androidparty)
    Box{
        Image(
            painter = image,
            contentDescription = null
        )
    }
}
```

- Abaixo da função **Image**, chame a função **GreetingText()**, e transmita a ela a mensagem de aniversário, a assinatura e o modificador, conforme mostrado abaixo:

```
@Composable
fun GreetingImage(message: String, from: String, modifier:
Modifier = Modifier) {
    val image = painterResource(R.drawable.androidparty)
    //Step 3 create a box to overlap image and texts
    Box {
        Image(
            painter = image,
            contentDescription = null
        )
    }
}
```

```
)  
GreetingText(  
    message = message,  
    from = from,  
    modifier = Modifier  
        .fillMaxSize()  
        .padding(8.dp)  
)  
}  
}
```

Agora podemos ver as palavras no cartão:



Para que as mudanças acima sejam aplicadas no emulador ou em um dispositivo, na função `onCreate()`, substitua a chamada de função `GreetingText()` por `GreetingImage()`. A imagem tem a mesma largura da tela, mas está fixada na parte de cima. Há um espaço em branco na parte de baixo da tela, o que não fica muito interessante. Em seguida, vamos aprender a redimensionar imagens.



### 3.4. Dimensionando a imagem e mudando a opacidade

Para redimensionar a imagem, podemos utilizar o parâmetro **ContentScale**. Há vários tipos de **ContentScale** disponíveis, mas nós usaremos o **ContentScale.Crop**, que não muda a proporção da imagem.

- Adicione um argumento nomeado **ContentScale** à imagem e importe a propriedade necessária.

```
Image(  
  painter = image,  
  contentDescription = null,  
  contentScale = ContentScale.Crop  
)
```

Agora a imagem preenche toda a tela.



Para melhorar o contraste do app, vamos diminuir a opacidade do plano de fundo.

- Adicione o parâmetro **alpha** à imagem e defina-o como 0.5F

```
Image(  
  painter = image,  
  contentDescription = null,  
  contentScale = ContentScale.Crop,  
  alpha = 0.5F  
)
```

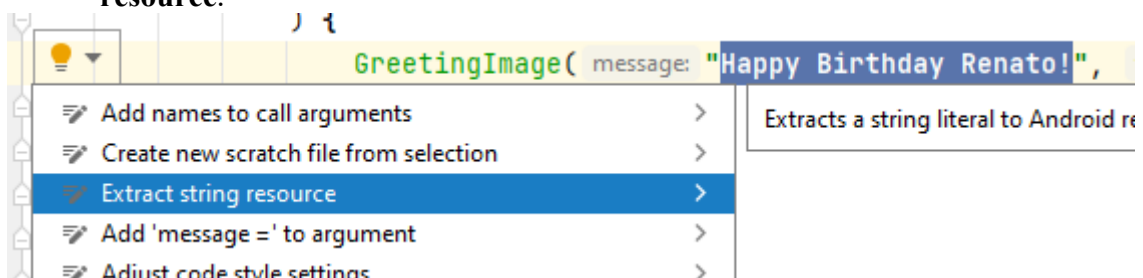
Perceba a diferença na opacidade.



### 3.5. Boas práticas de código

É importante lembrar que nossos apps podem ser traduzidos para outros idiomas em algum momento. Para facilitar essa tradução, podemos extrair as Strings que estão escritas diretamente no nosso código para um arquivo de recursos. Dessa maneira, em vez de fixar strings no código, colocaremos em um arquivo, nomearemos os recursos de string e utilizaremos os nomes sempre que quisermos usar as strings.

- No arquivo **MainActivity.kt**, role até a função **onCreate()**. Selecione a mensagem de aniversário, string **Happy Birthday Renato!** sem aspas.
- Clique na lâmpada no lado esquerdo da tela, e selecione **Extract string resource**.



- O Android Studio vai abrir a caixa de diálogo **Extract Resource**. Nela, mude o resource name para **happy\_birthday\_text**.
- Perceba as mudanças no código.

```
HappyBirthdayTheme {
    Surface (
        modifier = Modifier.fillMaxSize(),
        color = MaterialTheme.colorScheme.background
    ) {
        GreetingImage(stringResource(R.string.happy_birthday_text),
            "From Juliana")
    }
}
```

```

    }
}

```

Obs: algumas versões do Android Studio substituem a string fixada no código pela função `getString()`. Nesses casos, mude manualmente a função para `stringResource()`.

- Repita essas etapas para a String da assinatura, mas dessa vez utilize `signature_text` como Resource name
- Você pode ver o **arquivo de recursos** no caminho `app > res > values > strings.xml`

### 3.6. Boas práticas de código

Organize ou alinhe o elemento combinável de texto de assinatura para que ele fique centralizado na tela.

**Dica:** o Compose oferece o modificador `align` para posicionar um elemento combinável filho individualmente, desafiando as regras de layout de alinhamento aplicadas pelo layout pai. Encadeie o argumento `.align(alignment = Alignment.CenterHorizontally)` com o texto combinável Modifier.

Caso a imagem seja importada corretamente, ela será adicionada pelo Android Studio à lista na guia Drawable. Essa lista inclui todos os ícones e imagens do app, então você já pode usar a imagem importada.

## 4. Estágios do ciclo de vida da atividade

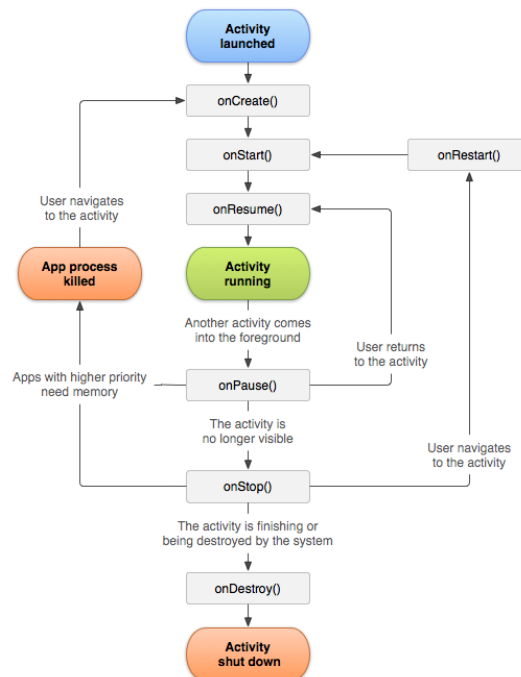
O ciclo de vida de uma atividade é o nome dado à transição de estados que uma atividade passa. Uma atividade pode mostrar várias telas e alternar entre elas, conforme necessário. O ciclo de vida da atividade se estende desde a criação até a destruição da atividade, quando o sistema recupera os recursos dela. À medida que o usuário navega para dentro e para fora de uma atividade, cada atividade transita entre diferentes estados no ciclo de vida.

Nessa etapa, vamos utilizar um aplicativo inicial chamado “Dessert Clicker”. Nele, sempre que o usuário tocar em uma sobremesa na tela, o app a “compra” para o usuário. O app atualiza valores no layout para o número de sobremesas que foram “compradas” e para o custo total das sobremesas “compradas”.

### 4.1. Examinar o método `onCreate()` e adicionar registros

- Baixe o código em: <https://github.com/google-developer-training/basic-android-kotlin-compose-training-dessert-clicker/tree/starter>
- No Android Studio, abra o projeto `basic-android-kotlin-compose-training-dessert-clicker`.
- Execute o app `DessertClicker` e toque várias vezes na foto de uma sobremesa. Observe como o valor de `Desserts sold` e o valor total em dólares mudam.

Para descobrir o que está acontecendo com o ciclo de vida do Android, é útil saber quando os vários métodos de ciclo de vida são chamados. Uma maneira simples de determinar essas informações é usar a funcionalidade de geração de registros do Android, que permite que você escreva mensagens curtas em um console enquanto o app está em execução.



- Adicione a constante a seguir no nível superior do **MainActivity.kt**, acima da declaração de classe **class MainActivity**.

```
private const val TAG = "MainActivity"
```

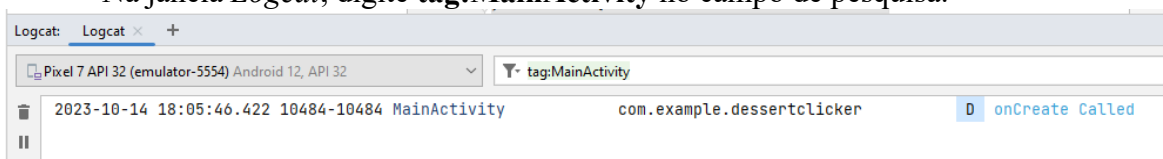
- No método **onCreate()**, logo após a chamada para **super.onCreate()**, adicione a seguinte linha, e importe a classe necessária.

```
Log.d(TAG, "onCreate Called")
```

A classe *Log* escreve mensagens no *Logcat*, que é o console usado para registrar mensagens do Android sobre seu app.

A mensagem de registro chamada msg (o segundo parâmetro) é uma *string* curta. Neste caso, "**onCreate Called**".

- Compile e execute o app *Dessert Clicker*. Você não vai notar diferenças de comportamento no app quando tocar na sobremesa. No Android Studio, na parte de baixo da tela, clique na guia *Logcat*.
- Na janela *Logcat*, digite **tag:MainActivity** no campo de pesquisa.





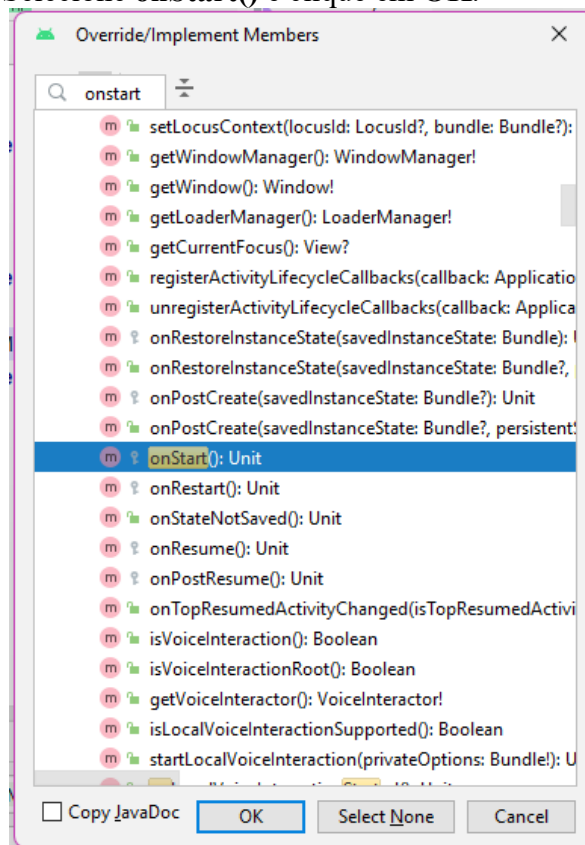
Com isso estamos filtrando as mensagens do **Logcat**. Como utilizamos a MainActivity como tag de registro no código, é possível usá-la para filtrar o registro. Como essa mensagem aparece no registro, você sabe que o método **onCreate()** foi executado.

#### 4.2. Implementar o método onStart()

O método **onStart()** é chamado logo depois de **onCreate()**. Depois que **onStart()** é executado, sua atividade fica visível na tela. Ao contrário de **onCreate()**, que é chamado apenas uma vez para inicializar a atividade, **onStart()** pode ser chamado pelo sistema várias vezes durante o ciclo de vida da atividade.

Todo **onStart()** é pareado com um método do ciclo de vida **onStop()** correspondente. Se o usuário inicia o app e retorna à tela inicial do dispositivo, a atividade é interrompida e não fica mais visível na tela.

- No Android Studio, com MainActivity.kt aberto e o cursor na classe MainActivity, selecione **Code > Override Methods....** Uma caixa de diálogo vai aparecer com uma longa lista de todos os métodos que você pode substituir nessa classe. Selecione **onStart()** e clique em **OK**.



- No código criado adicione a linha com a TAG para o método onStart(). Seu código deve ficar assim:

```
override fun onStart() {
    super.onStart()
    Log.d(TAG, "onStart Called")
}
```

}

- Compile e execute o app *Dessert Clicker* e abra o painel do *Logcat* novamente com a **tag:MainActivity** no campo de pesquisa.

Agora podemos ver que os métodos **onCreate()** e **onStart()** foram chamados um após o outro. Tente sair da tela do aplicativo e depois abri-la novamente. Você perceberá que o **onStart()** é registrado uma segunda vez no *Logcat*, enquanto o **onCreate()** não é chamado de novo.

Por fim, substitua os métodos restantes do ciclo de vida na **MainActivity** e adicione *log statements* para cada um, como mostrado neste código:

```

override fun onResume() {
    super.onResume()
    Log.d(TAG, "onResume Called")
}

override fun onRestart() {
    super.onRestart()
    Log.d(TAG, "onRestart Called")
}

override fun onPause() {
    super.onPause()
    Log.d(TAG, "onPause Called")
}

override fun onStop() {
    super.onStop()
    Log.d(TAG, "onStop Called")
}

override fun onDestroy() {
    super.onDestroy()
    Log.d(TAG, "onDestroy Called")
}

```

Agora que configuramos o *Dessert Clicker* para a geração de registros, já podemos começar a usar o app e explorar como os *callbacks* do ciclo de vida são acionados. Dessa maneira, podemos encontrar falhas no nosso aplicativo. Vamos começar com um caso de uso básico: fechar e abrir o aplicativo.

- Compile e execute o app *Dessert Clicker*, se ele ainda não estiver em execução. Os *callbacks* **onCreate()**, **onStart()** e **onResume()** são chamados quando a atividade é inicializada pela primeira vez.
- Toque no *cupcake* algumas vezes e em seguida volte para a tela inicial do celular.
- No *Logcat*, **onPause()** e **onStop()** são chamados nessa ordem.

2023-10-15 22:00:05.479	25841-25841 MainActivity	com.example.dessertclicker	D onCreate Called
2023-10-15 22:00:05.541	25841-25841 MainActivity	com.example.dessertclicker	D onStart Called
2023-10-15 22:00:05.545	25841-25841 MainActivity	com.example.dessertclicker	D onResume Called
2023-10-15 22:00:28.666	25841-25841 MainActivity	com.example.dessertclicker	D onPause Called
2023-10-15 22:00:29.094	25841-25841 MainActivity	com.example.dessertclicker	D onStop Called

Nesse caso, o uso do botão Voltar faz com que a atividade e o app sejam removidos da tela e movidos para a parte de trás da pilha de atividades. Isso coloca o aplicativo em **segundo plano**. Quando **onPause()** é chamado, o app não está mais em foco. Depois de **onStop()**, o app não fica mais visível na tela.

- Use a tela "Recentes" para retornar ao app. No Logcat, a atividade é reiniciada por **onRestart()** e **onStart()** e depois é retomada por **onResume()**.

```
MainActivity      com.example.dessertclicker      D  onPause Called
MainActivity      com.example.dessertclicker      D  onStop Called
MainActivity      com.example.dessertclicker      D  onRestart Called
MainActivity      com.example.dessertclicker      D  onStart Called
MainActivity      com.example.dessertclicker      D  onResume Called
```

Quando a atividade retorna para o primeiro plano, o método **onCreate()** não é chamado novamente. O objeto da atividade não foi destruído, por isso não precisa ser criado novamente.

A diferença entre foco e visibilidade é importante. É possível que uma atividade fique parcialmente visível na tela, mas não tenha o foco do usuário.

- Com o app *Dessert Clicker* em execução, clique no botão **Compartilhar** na parte superior direita da tela.



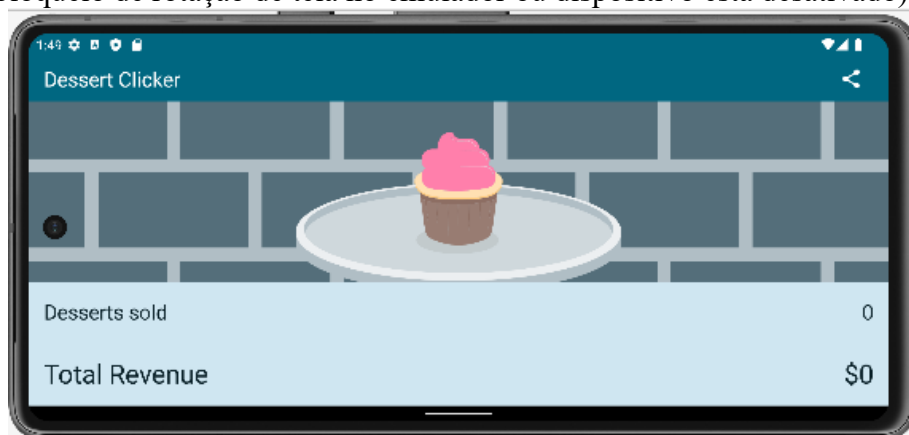
A atividade de compartilhamento vai aparecer na metade de baixo da tela, mas a atividade do app ainda vai estar visível na metade de cima. Analisando o *Logcat*, vemos que apenas o **onPause()** foi chamado. Neste caso de uso, **onStop()** não é chamado, porque a atividade ainda está parcialmente visível. Mas ela não tem o foco do usuário, e não é possível interagir com ela. A atividade de "compartilhamento" que está em primeiro plano tem o foco do usuário.

- Clique fora da caixa de diálogo de compartilhamento para retornar ao app. Observe que o **onResume()** é chamado.

### 4.3. Explorando mudanças de configuração

Uma mudança de configuração ocorre quando o estado do dispositivo muda de forma tão radical que a maneira mais fácil do sistema resolver a mudança é encerrar completamente e recriar a atividade.

- Com o aplicativo aberto e sendo executado, clique algumas vezes no doce.
- Em seguida, gire o dispositivo para o modo paisagem (certifique-se que o bloqueio de rotação de tela no emulador ou dispositivo está desativado).



Analise o *Logcat* e entenda que, quando a atividade é encerrada, ele chama **onPause()**, **onStop()** e **onDestroy()**, nessa ordem. Perceba que quando o dispositivo é girado e a atividade é encerrada e recriada, ela é reiniciada com valores padrão. Ou seja, a imagem da sobremesa, o número de sobremesas vendidas e a receita são redefinidos como zero, quando na realidade, não queríamos que isso acontecesse. Vamos ver como consertar esse problema.

A IU do app é criada inicialmente pela execução de funções de composição. Essas funções têm um **ciclo de vida próprio** que é independente do ciclo de vida da atividade. O ciclo de vida delas é composto pelos seguintes eventos: entrar na composição, recompor 0 ou mais vezes e, depois, sair da composição. Para que o *Compose* acompanhe e acione uma recomposição, ele precisa saber quando o estado mudou. Para indicar ao *Compose* que ele precisa rastrear o estado de um objeto, o objeto precisa ser do tipo **State** ou **MutableState**.

Assim, criamos a variável mutável **revenue**, declarando-a usando **mutableStateOf**. 0 é o valor padrão inicial (não é necessário adicionar esse código no seu projeto).

```
var revenue = mutableStateOf(0)
```

Esse código, porém, apenas aciona uma recomposição caso o valor da receita mude. Se o elemento combinável for executado novamente, ele vai reinicializar o valor da receita para o valor padrão inicial de 0. Para instruir o *Compose* a reter e reutilizar o valor durante as recomposições, é necessário declarar o valor com a API **remember** (esse código já está no seu projeto).

```
var revenue by remember {mutableStateOf(0)}
```

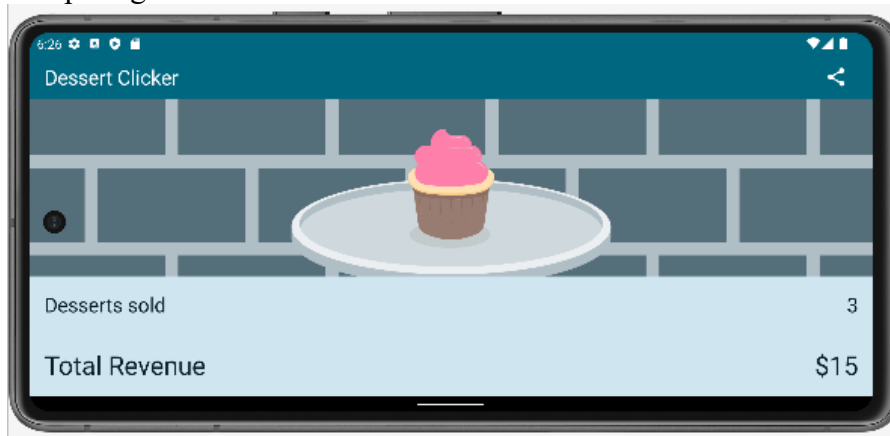
Dessa maneira, se o valor de revenue mudar, o Compose vai programar a recomposição de todas as funções de composição que leem esse valor.

Embora o Compose lembre o estado da receita durante as recomposições, ele ainda não retém esse estado durante uma mudança de configuração. Para que o Compose retenha o estado durante uma mudança de configuração, use **rememberSaveable**.

- No **MainActivity**, atualize o grupo de cinco variáveis que atualmente usam **remember** para **rememberSaveable**.

```
var revenue by rememberSaveable {mutableStateOf(0)}
```

- Compile e execute seu código.
- Clique no cupcake algumas vezes e observe que as sobremesas vendidas e a receita total não são zero. Em seguida, gire o dispositivo ou o emulador para o modo paisagem.



Observe que, depois que a atividade é destruída e recriada, a imagem da sobremesa, as sobremesas vendidas e a receita total são restauradas para os valores anteriores.

## 5. Componentes de Arquitetura

Agora vamos aprender a criar um app com eficiência e preservar os dados dele durante mudanças de configuração, usando as diretrizes da biblioteca Android Jetpack, do *ViewModel* e da arquitetura de apps Android.

A arquitetura de apps é um conjunto de regras de design para um app. Assim como a planta de uma casa, a arquitetura fornece a estrutura. Uma boa arquitetura pode deixar seu código otimizado, flexível, escalonável, testável e sustentável por anos.

Nesta etapa, vamos utilizar o aplicativo “*Unscramble*”, que é um jogo de palavras embaralhadas. O app exibe uma palavra embaralhada, e o jogador precisa adivinhá-la usando todas as letras mostradas. O jogador marca pontos se a palavra está correta. Caso contrário, o jogador pode tentar adivinhar a palavra quantas vezes quiser. O app também

tem a opção de pular a palavra atual. No canto superior direito, ele mostra a contagem de palavras, que é o número de palavras embaralhadas jogadas na sessão atual. Cada partida tem 10 palavras embaralhadas.

- Baixe o projeto no link abaixo, na ramificação *starter* <https://github.com/google-developer-training/basic-android-kotlin-compose-training-unscramble.git>
- Execute o projeto e toque nos botões do app para testá-lo.

Você vai notar bugs no app. A palavra embaralhada não é mostrada, mas está fixada no código como "*scrambleun*", e nada acontece quando você toca nos botões.

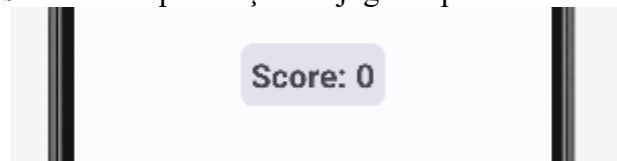
### 5.1. Código Inicial

Neste projeto, temos três arquivos **.kt** principais:

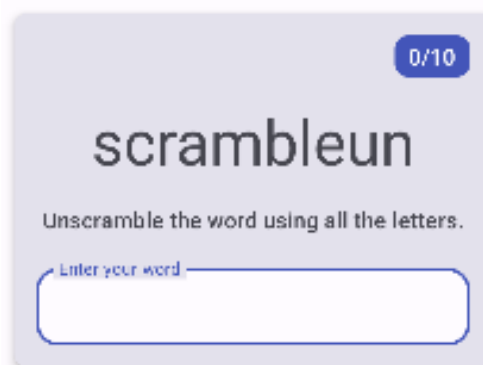
- **WordsData.kt** - contém uma lista das palavras usadas no jogo, constantes para o número máximo de palavras por partida e o número de pontos que o jogador marca para cada palavra correta.
- **MainActivity.kt** - contém principalmente o código gerado pelo modelo. O elemento combinável `GameScreen` é mostrado no bloco `setContent {}`.
- **GameScreen.kt** - define todos os elementos combináveis de interface

Algumas funções criadas no nosso projeto são:

- **GameStatus** - mostra a pontuação do jogo na parte de baixo da tela



- **GameLayout** - mostra a funcionalidade principal do jogo, incluindo a palavra embaralhada, as instruções do jogo e um campo de texto que aceita os palpites do usuário.



Por fim, os elementos combináveis que usamos são:

- **GameScreen** - contém as funções combináveis `GameStatus` e `GameLayout`, o título do jogo, a contagem de palavras e os elementos combináveis dos botões `Submit` (Enviar) e `Skip` (Pular).



- **FinalScoreDialog** - mostra uma caixa de diálogo, ou seja, uma janela pequena que dá ao usuário opções para *Play Again* (Jogar novamente) ou *Exit* (Sair) do jogo.



## 5.2. Um pouco mais sobre arquitetura de apps

Baseando-se em princípios de arquitetura comum, cada app precisa ter pelo menos duas camadas:

- **Camada de interface:** uma camada que exhibe os dados do app na tela, mas é independente dos dados, que é composta por:

- **Elementos da interface:** componentes que renderizam os dados na tela. Crie esses elementos usando o Jetpack Compose.
- **Detentores de estado:** componentes que armazenam os dados, os expõem à interface e processam a lógica do app. Um exemplo de detentor de estado é o **ViewModel**. Ao contrário da instância de atividade, os objetos **ViewModel** não são destruídos. O app retém automaticamente objetos **ViewModel** durante mudanças de configuração para que os dados retidos fiquem imediatamente disponíveis após a recomposição.
- **Camada de dados:** uma camada que armazena, recupera e expõe os dados do app.

### 5.3. Adicionando um ViewModel

Para resolver os problemas do código inicial que percebemos, vamos salvar os dados do jogo em um *ViewModel*.

- Abra **build.gradle.kts (Module :app)**, role até o bloco *dependencies* e adicione a seguinte dependência para *ViewModel*. Ela é usada para adicionar o modelo de visualização com reconhecimento de ciclo de vida ao app do *Compose*. Sincronize o projeto após esse passo.

```
implementation("androidx.lifecycle:lifecycle-viewmodel-  
compose:2.6.1")
```

- Na pasta *ui*, crie uma classe/arquivo Kotlin com o nome **GameViewModel**. Estenda-a da classe **ViewModel**.

```
import androidx.lifecycle.ViewModel  
  
class GameViewModel : ViewModel() {  
}
```

- Ainda na pasta *ui*, adicione uma classe de modelo para a interface do estado chamada **GameUiState**. Transforme-a em uma classe de dados e adicione uma variável para a palavra embaralhada atual.

```
data class GameUiState(  
    val currentScrambledWord: String = ""  
)
```

Um **StateFlow** (fluxo observável detentor de dados que emite as atualizações de estado novas e atuais) pode ser exposto no **GameUiState** para que os elementos combináveis possam detectar atualizações de estado da interface e fazer com que o estado da tela sobreviva às mudanças de configuração.

- Na classe **GameViewModel**, adicione a seguinte propriedade **\_uiState**:

```
import kotlinx.coroutines.flow.MutableStateFlow  
  
// Game UI state  
private val _uiState = MutableStateFlow(GameUiState())
```



Uma propriedade de apoio permite que você retorne algo de um *getter* diferente do objeto exato. Para os métodos *getter* e *setter*, é possível substituir um deles, ou ambos, e fornecer um comportamento personalizado próprio, protegendo assim, o dado original. Isso é útil pois queremos evitar atualizações de estado vindas de outras classes.

- No arquivo **GameViewModel.kt**, adicione a **uiState** uma propriedade de apoio com o nome **\_uiState**. Nomeie a propriedade como **uiState** e use o tipo **StateFlow<GameUiState>**.

```
import kotlinx.coroutines.flow.StateFlow

// Game UI state
private val _uiState = MutableStateFlow(GameUiState())
val uiState: StateFlow<GameUiState>
```

- Defina **uiState** como **\_uiState.asStateFlow()**.

```
import kotlinx.coroutines.flow.asStateFlow

// Game UI state
private val _uiState = MutableStateFlow(GameUiState())
val uiState: StateFlow<GameUiState> = _uiState.asStateFlow()
```

#### 5.4. Exibindo palavra embaralhada aleatória

Agora vamos adicionar métodos auxiliares para escolher uma palavra aleatória do **WordsData.kt** e criar a palavra embaralhada.

- No **GameViewModel**, adicione uma propriedade com o nome **currentWord** do tipo **String** para salvar a palavra embaralhada atual.

```
private lateinit var currentWord: String
```

- Chame o arquivo de **pickRandomWordAndShuffle()** sem parâmetros de entrada e faça com que ele retorne uma **String**.

```
private lateinit var currentWord: String
```

- O AndroidStudio sinaliza alguns erros. Adicione a seguinte propriedade após a **currentWord** para servir como um conjunto mutável no armazenamento de palavras usadas no jogo:

```
private var usedWords: MutableSet<String> = mutableSetOf()
```

- Adicione outro método auxiliar para embaralhar a palavra atual com o nome **shuffleCurrentWord()**, que usa uma **String** e retorna a **String** embaralhada.

```
private fun shuffleCurrentWord(word: String): String {
    val tempWord = word.toCharArray()
    // Scramble the word
```

```
tempWord.shuffle()
while (String(tempWord).equals(word)) {
    tempWord.shuffle()
}
return String(tempWord)
}
```

- Adicione uma função auxiliar para inicializar o jogo com o nome **resetGame()**. Vamos usar essa função mais tarde para iniciar e reiniciar o jogo. Nessa função, limpe todas as palavras do conjunto **usedWords** e inicie o **\_uiState**. Escolha uma nova palavra para **currentScrambledWord** usando **pickRandomWordAndShuffle()**.

```
fun resetGame() {
    usedWords.clear()
    _uiState.value = GameUiState(currentScrambledWord =
pickRandomWordAndShuffle())
}
```

- Por fim, adicione um bloco **init** ao **GameViewModel** e chame o **resetGame()** dele.

```
init {
    resetGame()
}
```

Se executarmos nosso app agora, ainda não vão aparecer mudanças na interface. Ainda não estamos transmitindo os dados do *ViewModel* aos elementos combináveis na **GameScreen**.

Como as funções de composição aceitam estados e expõem eventos, um padrão de fluxo de dados unidirecional é adequado para o Jetpack Compose. Esse tipo de fluxo de dados é um padrão onde os estados fluem para baixo, e os eventos para cima. Ao seguir o fluxo de dados unidirecional, você pode dissociar os elementos que exibem o estado na interface das partes do app que armazenam e mudam o estado.

- Na função **GameScreen**, transmita um segundo argumento do tipo **GameViewModel** com um valor padrão de **viewModel()**.

```
import
androidx.lifecycle.viewmodel.compose.viewModel

@Composable
fun GameScreen(
    gameViewModel: GameViewModel = viewModel()
) {
    // ...
}
```

- Na função **GameScreen()**, adicione uma nova variável com o nome **gameUiState**. Use o delegado **by** e chame **collectAsState()** no **uiState**. Essa abordagem garante que, sempre que haja uma mudança no valor do **uiState**, a recomposição ocorra para os elementos combináveis usando o valor **gameUiState**.

```
import androidx.compose.runtime.collectAsState
import androidx.compose.runtime.getValue

@Composable
fun GameScreen(
    // ...
) {
    val gameUiState by
    gameViewModel.uiState.collectAsState()
    // ...
}
```

- Transmita **gameUiState.currentScrambledWord** para o elemento combinável **GameLayout()**. Vamos adicionar o argumento em uma etapa posterior, então ignore o erro por enquanto.

```
GameLayout (
    currentScrambledWord =
    gameUiState.currentScrambledWord,
    modifier = Modifier
        .fillMaxWidth()
        .wrapContentHeight()
        .padding(mediumPadding)
)
```

- Adicione **currentScrambledWord** como outro parâmetro à função combinável **GameLayout()**.

```
@Composable
fun GameLayout(currentScrambledWord: String, modifier: Modifier =
Modifier)
{
    ...
}
```

- Atualize a função de composição **GameLayout()** para mostrar **currentScrambledWord**. Defina o parâmetro **text** do segundo campo de texto na coluna como **currentScrambledWord**.

```
@Composable
fun GameLayout(
    // ...
) {
    Column(
        //...
    ) {
        Text(
```

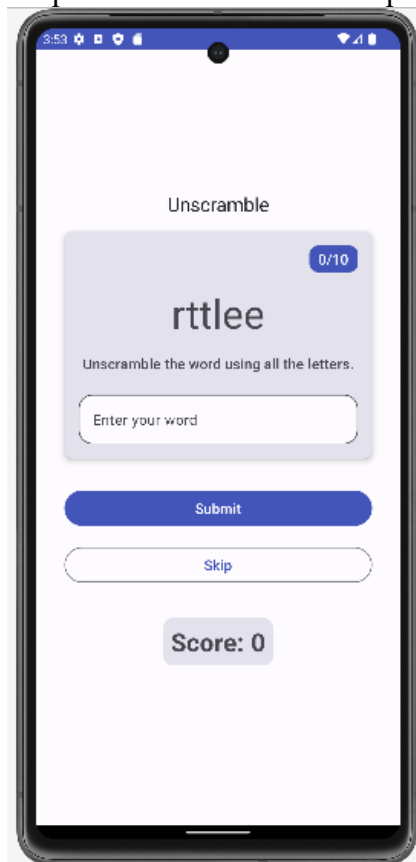
```

        text = currentScrambledWord,
        style =
            typography.displayMedium
    )

    //...
}

```

- Execute e crie o app. A palavra embaralhada vai aparecer.



- Para mostrar a palavra do palpite, no arquivo **GameScreen.kt**, no elemento combinável **GameLayout()**, defina **onValueChange** como **onUserGuessChanged** e **onKeyboardDone()** como ação do teclado **onDone**. Não ligue para os erros por enquanto.

```

OutlinedTextField(
    value = "",
    singleLine = true,
    shape = shapes.large,
    modifier = Modifier.fillMaxWidth(),
    colors = TextFieldDefaults.colors(
        focusedContainerColor = colorScheme.surface,
        unfocusedContainerColor = colorScheme.surface,
    )
)

```

```

        disabledContainerColor = colorScheme.surface,
    ),
    onChange = onUserGuessChanged,
    label = { Text(stringResource(R.string.enter_your_word))
},
    isError = false,
    keyboardOptions = KeyboardOptions.Default.copy(
        imeAction = ImeAction.Done
    ),
    keyboardActions = KeyboardActions(
        onDone = { onKeyboardDone() }
    )
)

```

- Na função combinável **GameLayout()**, adicione mais dois argumentos: o lambda **onUserGuessChanged** recebe um argumento **String** e não retorna nada, e **onKeyboardDone** não recebe nem retorna nada.

```

@Composable
fun GameLayout(
    onUserGuessChanged: (String) ->
Unit,
    onKeyboardDone: () -> Unit,
    currentScrambledWord: String,
    modifier: Modifier = Modifier
)

```

- Na chamada de função **GameLayout()**, adicione argumentos lambda para **onUserGuessChanged** e **onKeyboardDone**.

```

GameLayout(
    onUserGuessChanged = { gameViewModel.updateUserGuess(it)
},
    onKeyboardDone = { },
    currentScrambledWord = gameUiState.currentScrambledWord,
    modifier = Modifier
        .fillMaxWidth()
        .wrapContentHeight()
        .padding(mediumPadding)
)

```

Vamos definir o método **updateUserGuess** no **GameViewModel** em breve.

- No arquivo **GameViewModel.kt**, adicione um método com o nome **updateUserGuess()** que usa um argumento **String**, a palavra adivinhada pelo usuário. Dentro da função, atualize a **userGuess** usando a **guessedWord** transmitida.

```

fun updateUserGuess(guessedWord:
String){
    userGuess = guessedWord
}

```

- No arquivo **GameViewModel.kt**, adicione uma propriedade var chamada **userGuess**. Use **mutableStateOf()** para que o **Compose** observe esse valor e defina o valor inicial como " ".

```
import
androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.getValue
import androidx.compose.runtime.setValue

var userGuess by mutableStateOf("")
private set
```

- No arquivo **GameScreen.kt**, em **GameLayout()**, adicione outro parâmetro String para **userGuess**. Defina o parâmetro value do **OutlinedTextField** como **userGuess**.

```
fun GameLayout(
    currentScrambledWord: String,
    userGuess: String,
    onUserGuessChanged: (String) -> Unit,
    onKeyboardDone: () -> Unit,
    modifier: Modifier = Modifier
) {
    Column(
        verticalArrangement =
Arrangement.spacedBy(24.dp)
    ) {
        //...
        OutlinedTextField(
            value = userGuess,
            //..
        )
    }
}
```

- Na função **GameScreen**, atualize a chamada de função **GameLayout()** para incluir o parâmetro **userGuess**.

```
GameLayout(
    currentScrambledWord = gameUiState.currentScrambledWord,
    userGuess = gameViewModel.userGuess,
    onUserGuessChanged = { gameViewModel.updateUserGuess(it)
},
    onKeyboardDone = { },
    //...
)
```

- Compile e execute o app. O campo de texto agora mostra o palpite do usuário.



### 5.5. Verificando a palavra adivinhada e atualizando a pontuação

No arquivo `GameViewModel`, adicione outro método com o nome `checkUserGuess()`. Na função `checkUserGuess()`, adicione um bloco `if else` para verificar se o palpite do usuário é igual à `currentWord`. Redefinir `userGuess` para uma string vazia.

```
fun checkUserGuess() {
    if (userGuess.equals(currentWord, ignoreCase = true))
    {
    } else {
    }
    // Reset user guess
    updateUserGuess("")
}
```

Se o palpite do usuário estiver errado, defina `isGussedWordWrong` como `true`. `MutableStateFlow<T>.update()` atualiza o `MutableStateFlow.value` usando o valor especificado (links em inglês).

```
import kotlinx.coroutines.flow.update

if (userGuess.equals(currentWord, ignoreCase = true))
{
} else {
    // User's guess is wrong, show an error
    _uiState.update { currentState ->
        currentState.copy(isGussedWordWrong = true)
    }
}
```

Na classe `GameUiState`, adicione um `Boolean` chamado `isGussedWordWrong` e inicialize-o como `false`.

```

data class GameUiState(
    val currentScrambledWord: String = "",
    val isGuessedWordWrong: Boolean =
false,
)

```

No arquivo **GameScreen.kt**, no final da função combinável **GameScreen()**, chame **gameViewModel.checkUserGuess()** dentro da expressão lambda **onClick** do botão **Submit**. Esse passo serve para definir o erro no campo de texto.

```

Button(
    modifier = Modifier.fillMaxWidth(),
    onClick = { gameViewModel.checkUserGuess() }
) {
    Text(
        text = stringResource(R.string.submit),
        fontSize = 16.sp
    )
}

```

Na função combinável **GameScreen()**, atualize a chamada de função **GameLayout()** para transmitir **gameViewModel.checkUserGuess()** na expressão lambda **onKeyboardDone**.

```

GameLayout(
    onUserGuessChanged = { gameViewModel.updateUserGuess(it) },
    userGuess = gameViewModel.userGuess,
    onKeyboardDone = { gameViewModel.checkUserGuess() },
    currentScrambledWord = gameUiState.currentScrambledWord,
    modifier = Modifier
        .fillMaxWidth()
        .wrapContentHeight()
        .padding(mediumPadding)
)

```

Na função de composição **GameLayout()**, adicione um parâmetro de função para Boolean, **isGuessWrong**. Defina o parâmetro **isError** do **OutlinedTextField** como **isGuessWrong** para mostrar o erro no campo de texto quando o palpite do usuário estiver errado.

```

fun GameLayout(
    currentScrambledWord: String,
    isGuessWrong: Boolean,
    userGuess: String,
    onUserGuessChanged: (String) -> Unit,
    onKeyboardDone: () -> Unit,
    modifier: Modifier = Modifier
) {
    Column(
        // ,...
        OutlinedTextField(
            // ...
            isError = isGuessWrong,
            keyboardOptions =
KeyboardOptions.Default.copy(
imeAction = ImeAction.Done

```



```

    ),
    keyboardActions = KeyboardActions(
        onDone = { onKeyboardDone() }
    ),
)
}
}

```

Na função combinável **GameScreen()**, atualize a chamada de função **GameLayout()** para transmitir **isGuessWrong**.

```

GameLayout (
    onUserGuessChanged = { gameViewModel.updateUserGuess(it) },
    userGuess = gameViewModel.userGuess,
    onKeyboardDone = { gameViewModel.checkUserGuess() },
    currentScrambledWord = gameUiState.currentScrambledWord,
    isGuessWrong = gameUiState.isGuessedWordWrong,
    modifier = Modifier
        .fillMaxWidth()
        .wrapContentHeight()
        .padding(mediumPadding)
)

```

Compile e execute o app. Insira um palpite errado e clique em Submit. Observe que o campo de texto fica vermelho, indicando o erro.



Agora vamos atualizar a pontuação e a contagem de palavras enquanto o usuário joga.

No **GameUiState**, adicione uma variável **score** e a inicialize em zero.

```
data class GameUiState(
    val currentScrambledWord: String = "",
    val isGuessedWordWrong: Boolean = false,
    val score: Int = 0
)
```

Para atualizar o valor da pontuação, aumente o valor de score na função **checkUserGuess()**, em **GameViewModel**, dentro da condição if para quando o palpite do usuário estiver correto.

```
fun checkUserGuess() {
    if (userGuess.equals(currentWord, ignoreCase = true)) {
        val updatedScore =
        uiState.value.score.plus(SCORE_INCREASE)
    } else {
        ...
    }
}
```

No arquivo **GameViewModel**, adicione outro método com o nome **updateGameState** para atualizar a pontuação, incrementar a contagem de palavras atual e escolher uma nova palavra no arquivo **WordsData.kt**. Adicione uma **Int** chamada **updatedScore** como parâmetro. Atualize as variáveis da interface do estado do jogo desta forma:

```
private fun updateGameState(updatedScore: Int) {
    _uiState.update { currentState ->
        currentState.copy(
            isGuessedWordWrong = false,
            currentScrambledWord = pickRandomWordAndShuffle(),
            score = updatedScore
        )
    }
}
```

Na função **checkUserGuess()**, se o palpite do usuário estiver correto, faça uma chamada para **updateGameState** com a pontuação atualizada para preparar o jogo para a próxima rodada.

```
fun checkUserGuess() {
    if (userGuess.equals(currentWord, ignoreCase = true)) {
        val updatedScore =
        _uiState.value.score.plus(SCORE_INCREASE)
        updateGameState(updatedScore)
    } else {
        // . . .
    }
}
```

A **checkUserGuess()** concluída vai ficar assim:

```

fun checkUserGuess() {

    if (userGuess.equals(currentWord, ignoreCase = true)) {
        val updatedScore =
        _uiState.value.score.plus(SCORE_INCREASE)
        updateGameState(updatedScore)
    } else {
        _uiState.update { currentState ->
            currentState.copy(isGuessedWordWrong = true)
        }
    }
    updateUserGuess("")
}

```

Em seguida, assim como as atualizações da pontuação, vai ser necessário atualizar a contagem de palavras. Adicione outra variável para a contagem em **GameUiState**. Chame-a de **currentWordCount** e inicialize em 1.

```

data class GameUiState(
    val currentScrambledWord: String = "",
    val currentWordCount: Int = 1,
    val isGuessedWordWrong: Boolean = false,
    val score: Int = 0
)

```

No arquivo **GameViewModel.kt**, na função **updateGameState()**, aumente a contagem de palavras, como mostrado abaixo. A função **updateGameState()** é chamada para preparar o jogo para a próxima rodada.

```

private fun updateGameState(updatedScore: Int) {
    _uiState.update { currentState ->
        currentState.copy(
            isGuessedWordWrong = false,
            currentScrambledWord = pickRandomWordAndShuffle(),
            currentWordCount =
            currentState.currentWordCount.inc(),
            score = updatedScore
        )
    }
}

```

Agora precisamos transmitir dados de pontuação e contagem de palavras de **ViewModel** para **GameScreen**. No arquivo **GameScreen.kt** da função combinável **GameLayout()**, adicione a contagem de palavras como argumento e transmita os argumentos do formato **wordCount** para o elemento de texto.

```

fun GameLayout(
    onUserGuessChanged: (String) -> Unit,
    onKeyboardDone: () -> Unit,
    wordCount: Int,
    //...
) {
    //...
}

```

```

Card(
    //...
) {
    Column(
        // ...
    ) {
        Text(
            //..
            text = stringResource(R.string.word_count,
wordCount),
            style = typography.titleMedium,
            color = colorScheme.onPrimary
        )
        // ...
    }
}

```

Atualize a chamada de função **GameLayout()** para incluir a contagem de palavras.

```

GameLayout (
    onUserGuessChanged = { gameViewModel.updateUserGuess(it) },
    userGuess = gameViewModel.userGuess,
    //...
)

```

Na função combinável **GameScreen()**, atualize a chamada de função **GameStatus()** para incluir os parâmetros score. Transmita a pontuação do **gameUiState**.

```

GameStatus(score=gameUiState.score,
modifier=Modifier.padding(20.dp) )

```

Crie e execute o app. Insira um palpite para a palavra e clique em Submit. A pontuação e a contagem de palavras são atualizadas. Clique em Skip e observe que nada acontece.



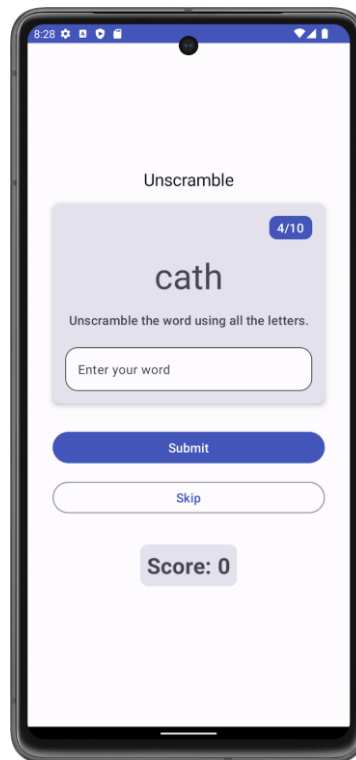
Para implementar o **Skip**, precisamos transmitir o callback do evento de pulo para o **GameViewModel**. Para isso, no arquivo **GameScreen.kt**, na função de composição **GameScreen()**, faça uma chamada para **gameViewModel.skipWord()** na expressão lambda **onClick**.

```
OutlinedButton(
    onClick = { gameViewModel.skipWord() },
    modifier = Modifier.fillMaxWidth()
)
```

Em seguida, em **GameViewModel.kt**, adicione o método **skipWord()**. Na função **skipWord()**, faça uma chamada para **updateGameState()**, transmitindo a pontuação e redefina o palpite do usuário.

```
OutlinedButton(
    onClick = { gameViewModel.skipWord() },
    modifier = Modifier.fillMaxWidth()
)
```

Agora execute o app e jogue uma partida, você conseguirá pular palavras! Ainda não limitamos o número de partidas, então você conseguirá jogar mais de 10.



### 5.6. Processando a última rodada do jogo

Para implementar a lógica de fim de jogo, primeiro é preciso verificar se o usuário atingiu o número máximo de palavras. No `GameViewModel`, adicione um bloco `if-else` e mova o corpo da função existente para o bloco `else`. Adicione uma condição `if` para verificar se o tamanho das `usedWords` é igual a `MAX_NO_OF_WORDS`.

No bloco `if`, adicione a sinalização Boolean `isGameOver` e a defina como `true` para indicar o fim do jogo. Atualize a `score` e redefina `isGuessedWordWrong` dentro do bloco `if`. O código abaixo mostra como a função vai ficar.

```
private fun updateGameState(updatedScore: Int) {
    if (usedWords.size == MAX_NO_OF_WORDS) {
        //Last round in the game, update isGameOver to true,
        don't pick a new word
        _uiState.update { currentState ->
            currentState.copy(
                isGuessedWordWrong = false,
                score = updatedScore,
                isGameOver = true
            )
        }
    } else {
        // Normal round in the game
        _uiState.update { currentState ->
            currentState.copy(
                isGuessedWordWrong = false,
                currentScrambledWord =
```

```

pickRandomWordAndShuffle(),
                currentWordCount =
currentState.currentWordCount.inc(),
                score = updatedScore
            )
        }
    }
}

```

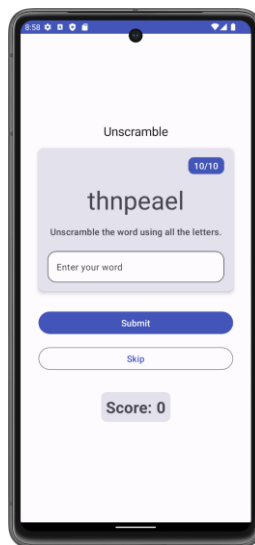
No **GameUiState**, adicione a variável Boolean **isGameOver** e a defina como **false**.

```

data class GameUiState(
    val currentScrambledWord: String = "",
    val currentWordCount: Int = 1,
    val isGuessedWordWrong: Boolean = false,
    val score: Int = 0,
    val isGameOver: Boolean = false
)

```

Execute o app e jogue uma partida. Não será possível jogar mais de 10 palavras.



Agora precisamos informar ao usuário que o jogo terminou e perguntar se ele quer jogar de novo. Para isso criaremos uma caixa de diálogo, que são pequenas janelas que levam o usuário a tomar uma decisão ou inserir mais informações.

O arquivo **GameScreen.kt** no código inicial já fornece uma função que mostra uma caixa de diálogo de alerta com opções para sair ou reiniciar o jogo, chamada **FinalScoreDialog**. Precisamos fazer ela ser chamada quando o jogo realmente é finalizado.

No arquivo **GameScreen.kt**, no final da função combinável **GameScreen()**, depois do bloco Column, adicione uma condição if para conferir **gameUiState.isGameOver**.

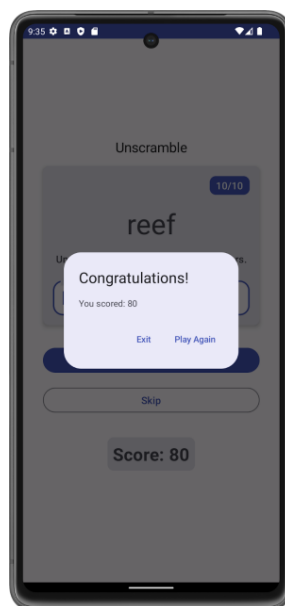
No bloco `if`, exiba a caixa de diálogo de alerta. Faça uma chamada para `FinalScoreDialog()` transmitindo `score` e `gameViewModel.resetGame()` para o callback de evento `onPlayAgain`.

```
Column(  
    //...  
)  
  
{  
    if (gameUiState.isGameOver) {  
        FinalScoreDialog(  
            score = gameUiState.score,  
            onPlayAgain = { gameViewModel.resetGame() }  
        )  
    }  
}
```

No arquivo `GameViewModel.kt`, chame a função `resetGame()`, inicialize `_uiState` e escolha uma nova palavra.

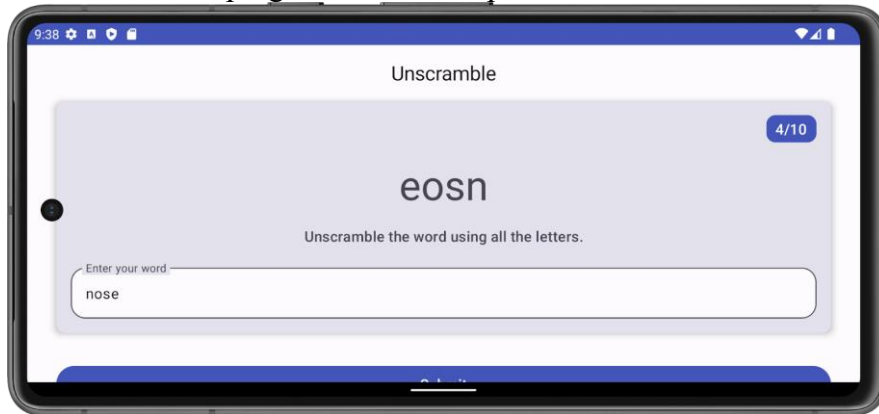
```
fun resetGame() {  
    usedWords.clear()  
    _uiState.value = GameUiState(currentScrambledWord =  
        pickRandomWordAndShuffle())  
}
```

Compile e execute o app. Jogue até o final e observe a caixa de diálogo de alerta com as opções Sair ou Jogar novamente. Teste as opções mostradas na caixa de diálogo de alerta.





O *ViewModel* armazena os dados relacionados ao app que não são destruídos quando o framework do Android destrói e recria a atividade. Os objetos *ViewModel* são retidos automaticamente e não são destruídos, como a instância de atividade, durante a mudança da configuração. Os dados retidos ficam imediatamente disponíveis após a recomposição. Por isso, se você mudar a orientação do dispositivo no meio de uma partida, os dados do seu progresso não serão perdidos.



## Referência

Documentação oficial de Kotlin para Android. Noções básicas do Android com o Compose. Google. Acesso em 29 de agosto de 2023. URL: [https://developer.android.com/courses/android-basics-compose/course?gclid=CjwKCAjwivemBhBhEiwAJxNWN25rWjfhk8Hx\\_7zs43wqKEgN1FoY9kkzSd4u6rlzSkx52TmZ9iuTxxoCcc0QAxD\\_BwE&gclsrc=aw.ds&hl=pt-br](https://developer.android.com/courses/android-basics-compose/course?gclid=CjwKCAjwivemBhBhEiwAJxNWN25rWjfhk8Hx_7zs43wqKEgN1FoY9kkzSd4u6rlzSkx52TmZ9iuTxxoCcc0QAxD_BwE&gclsrc=aw.ds&hl=pt-br)

Documentação oficial de Kotlin para Android. Conceitos básicos do Kotlin para Android. Google. Acesso em 29 de agosto de 2023. URL: <https://developer.android.com/courses/kotlin-android-fundamentals/overview?hl=pt-br>

Documentação oficial de Kotlin para Android. Android para desenvolvedores. Google. Acesso em 29 de agosto de 2023. URL: <https://developer.android.com/?hl=pt-br>

Curso de formação em desenvolvimento Android. Desenvolvido por Paulo Salvatore, Globalcode e Movable. Julho de 2022. URL: <https://developer.android.com/?hl=pt-br>

## Capítulo

# 3

## Desenvolvimento Full-Stack com Javascript: uma Visão Geral e Prática

Bruna C. R. Cunha  
Instituto de Ciências Matemáticas e de Computação (ICMC)  
Universidade de São Paulo (USP)  
São Carlos – SP – Brasil  
brunaru@icmc.usp.br

### *Abstract*

*This chapter presents basic principles for developing a service-based full-stack Web application using REST. Considering the current Web development scenario, the specific objectives are to use the JavaScript language for both front-end and back-end development, and to apply frameworks and libraries for developing modern Web applications, such as Express, a framework for web server constructions, and Sequelize, a library used for Object-Relational Mapping. We discuss basic principles, including technologies and the main architectural patterns currently used, that support the use of technologies for the Web. At the end, we present a Web page that uses JS to consume the developed services.*

### *Resumo*

*Este capítulo apresenta aspectos básicos do desenvolvimento de uma aplicação Web full-stack baseada em serviços utilizando REST. Considerando o atual cenário do desenvolvimento Web, os objetivos específicos são utilizar a linguagem JavaScript em ambas as camadas de desenvolvimento, front-end e back-end, e aplicar frameworks e bibliotecas para o desenvolvimento de aplicações Web modernas, como o Express, framework para construções de servidores web, e o Sequelize, biblioteca utilizada para o Mapeamento Objeto-Relacional. São discutidos princípios básicos, incluindo as tecnologias e os principais padrões arquitetônicos utilizados atualmente, de forma a fundamentar o uso de tecnologias para a Web. Ao final, é apresentada uma página Web que utiliza JS para consumir os serviços desenvolvidos.*

### 3.1. Introdução

Uma aplicação Web é um *software* distribuído e acessível por meio da Internet por clientes, mais especificamente navegadores Web (*i.e.*, *Web browser*, em inglês). Esses sistemas são organizados de forma que o usuário interage com um programa em execução no computador local (*i.e.*, um navegador) que, por sua vez, realiza requisições pela Internet (Sommerville, 2019). As requisições são atendidas por computadores remotos, ou seja, servidores, que hospedam o sistema e seus demais recursos. Um servidor Web fornece serviços, como acesso a páginas da Web, recursos multimídia, dados e aplicações, de acordo com as funções determinadas pela arquitetura adotada. Em outros termos, uma aplicação web trata-se de um *software*, hospedado em um servidor, que apresenta funcionalidades diversas.

Nesse domínio, é importante clarificar também a notação de servidor Web. Um servidor Web é referente a um *hardware* ou um *software* (ou o conjunto de ambos) que atende requisições de clientes (Mozilla Foundation, 2023). Enquanto o *software* é composto de diferentes componentes que permitem que clientes acessem diferentes recursos, o *hardware* trata-se de um computador que armazena o *software* e, possivelmente, outros elementos relacionados, como arquivos e bancos de dados. Esse último trata-se de um *software* dedicado a armazenar, manter e manipular coleções organizadas de informações estruturadas.

Bancos de dados podem ser relacionais, os quais organizam dados em tabelas relacionadas entre si por meio de chaves e utilizam a *Structured Query Language* (SQL) como linguagem padrão. Exemplos são MySQL, Oracle Database, PostgreSQL. Dado que o formato relacional foi dominante por muitos anos, bancos não relacionais são chamados NoSQL. O MongoDB e o Neo4j são duas alternativas não relacionais populares atualmente, sendo um baseado em documentos e o outro orientado a grafos, respectivamente.

Considerando esses elementos, uma aplicação Web pode apresentar um ou mais servidores, responsáveis por entregar uma interface, executável em navegadores Web, que se comunicam com servidor(es) lógico(s), responsáveis, minimamente, pelo processamento de requisições e acesso a banco(s) de dados.

Aplicações Web são apoiadas pelo modelo cliente-servidor, considerando que o cliente, um navegador, exibe e executa parte do código recebido do servidor, enquanto o servidor Web responde requisições e as processa. Considerando a organização de aplicações Web, é importante contextualizar os termos programação *front-end* e *back-end*. *Front-end* é o termo utilizado para definir a aplicação executada pelo cliente (navegador) e trata-se do código responsável, principalmente, pela interface com o usuário e pela lógica de processamento e comunicação com o servidor. Já a expressão *back-end* se refere ao(s) componente(s) executados em servidores (físicos), como regras de negócio e acesso ao(s) banco(s) de dados (Marinho e da Cruz, 2020). Por fim, o termo *full-stack* é utilizado para denominar a programação de uma aplicação Web por completo.

O desenvolvimento *front-end* é apoiado pelo uso de linguagens interpretadas pelo navegador e que permitem a exibição de conteúdo interativo. Atualmente, são três linguagens que lideram esse cenário: HTML, Cascading Style Sheets (CSS) e JavaScript

(JS). Enquanto o HTML é utilizado para estruturar o conteúdo e a semântica de uma página Web, o CSS é o recurso utilizado para formatar e estruturar as páginas e o JavaScript é a principal linguagem de programação utilizada para atribuir interatividade e permitir a comunicação com servidores Web. Essas tecnologias possuem organizações responsáveis por manter sua evolução e seus padrões de implementação, o HTML e o CSS são mantidas pelo World Wide Web Consortium (W3C), enquanto o JS segue a especificação de linguagem mantida pela ECMA Script.

Na outra ponta, o desenvolvimento *back-end* utiliza uma ampla variedade de linguagens. Para que uma determinada linguagem possa ser utilizada, é preciso, apenas, que ela seja executável pelo *hardware* do servidor e que, de alguma forma, disponibilize recursos de comunicação HTTP que permitam a implementação de aplicações Web. Entre as linguagens utilizadas pela programação *back-end* temos JavaScript, Java, Python, C#, PHP, entre outras.

Tanto o desenvolvimento *front-end* como o *back-end* são apoiados por bibliotecas e *frameworks* que auxiliam o trabalho de desenvolvedores. Bibliotecas são coleções de classes e métodos empacotados para reusabilidade. Ou seja, uma biblioteca implementa um conjunto conexo e específico de funcionalidades visando resolver um problema particular. Por outro lado, um *framework* pode ser definido como uma coleção de bibliotecas que incorporam uma abordagem específica de desenvolvimento que estruturam e agilizam o processo de criação de um *software*.

## 3.2. Tecnologias

Esta seção apresenta as tecnologias envolvidas no desenvolvimento de aplicações Web. Os conceitos aqui discutidos fundamentam explicações sobre o funcionamento de recursos aplicados na programação JS que serão trabalhados em seções futuras.

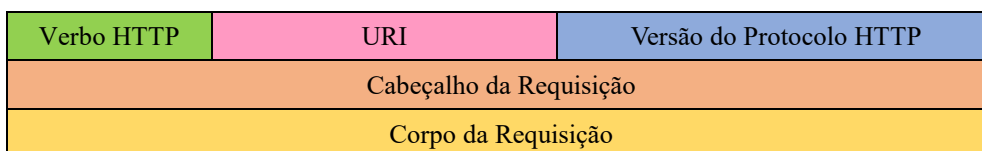
### 3.2.1. O Protocolo HTTP

A Internet apresenta uma arquitetura de protocolos organizada em cinco camadas: aplicação, transporte, rede, enlace e física. O *Hypertext Transfer Protocol* (HTTP), assim como sua variante segura, o *Hypertext Transfer Protocol Secure* (HTTPS), está localizado na camada de aplicação, sendo responsabilidade dos navegadores e servidores Web implementarem o seu padrão. Por estar na camada de aplicação, o HTTP dialoga com protocolos da camada de transporte, especialmente o *Transmission Control Protocol* (TCP).

O HTTP é um protocolo de comunicação baseado em mensagens de requisição e resposta, partindo de um cliente para um servidor. Uma característica importante dessa comunicação é que ela é sem estado (*i.e.*, *stateless*), ou seja, após receber uma requisição de um cliente, o servidor processa e responde, sem guardar informações de estado sobre a requisição realizada. Essa característica torna o modelo da Web bastante escalável. O entendimento do HTTP é essencial para o desenvolvimento consciente de aplicações e arquiteturas baseadas em seu funcionamento. Isso ficará mais evidente no decorrer deste capítulo.

Em termos de estrutura, uma mensagem de requisição é composta por um método HTTP (*e.g.*, verbos GET, POST, PUT, DELETE), pelo endereço do recurso (*i.e.*, seu *Uniform Resource Identifier*, ou URI) e pela versão do protocolo utilizada. A

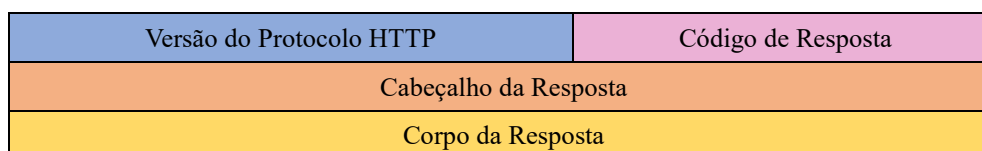
mensagem também pode, opcionalmente, conter um cabeçalho, com informações adicionais sobre a requisição, como *tokens* de autenticação, tipo de dados, tipo de *encoding*, *cookies* etc. Dependendo do método, a mensagem pode apresentar um corpo, que contém os dados da requisição. O corpo pode conter dados em XML e JSON ou até mesmo uma mídia, a depender do verbo utilizado. A Figura 3.1 ilustra a estrutura de uma mensagem de requisição (*request*).



**Figura 3.1. Estrutura de uma mensagem de requisição HTTP**

Uma requisição do tipo POST, por exemplo, é utilizada para enviar dados a um servidor e seu corpo irá conter informações, como dados de um formulário para cadastro. Já uma requisição do tipo GET não apresenta corpo e é utilizada, geralmente, para recuperar um recurso. A extensibilidade do protocolo permite que ele seja utilizado para a recuperação de diferentes tipos de recurso. A própria ação de acessar um website gera uma requisição do tipo GET para recuperar a página correspondente. Além da página, outras requisições são disparadas para recuperar os demais recursos associados, como arquivos CSS e JavaScript, imagens e demais arquivos multimídia.

Mensagens HTTP utilizadas na resposta seguem uma estrutura similar: uma primeira linha, contendo a versão do protocolo e um código de resposta, acompanhada por cabeçalho e corpo opcionais. A Figura 3.2 ilustra a estrutura de uma mensagem de resposta (*response*). O código da resposta (*status*) é um valor numérico, definido no servidor, e, quando corretamente implementado, segue uma lógica de acordo com intervalos definidos. Por exemplo, mensagens de *status* na casa dos 200 representam o sucesso da execução da requisição, valores na casa dos 400 são requisições malformadas (entre as quais consta o famoso erro 404, de recurso não encontrado) e entre os 500 estão os erros de servidor. O cabeçalho pode conter informações adicionais sobre a resposta, como o tipo de conteúdo, controle de cache e *encoding*. O corpo pode conter uma página, texto ou qualquer outro dado ou arquivo requisitado.



**Figura 3.2. Estrutura de uma mensagem de resposta HTTP**

### 3.2.2. A linguagem HTML

Proposta em 1990 por Berners-Lee, a linguagem HTML trata-se de uma “linguagem de marcação de hipertexto”, ou seja, como o nome indica, é uma linguagem que permite a definição de textos enriquecidos pela inserção de conexões para outros documentos e recursos. O HTML passou por diversas versões e atualmente está em sua versão 5, publicada como padrão em 2016.

Utiliza-se a "marcação" do HTML para definir os elementos e descrever a estrutura de uma página Web. A marcação é realizada por marcadores chamados "tags". Esses elementos indicam ao navegador como exibir o conteúdo. O HTML apresenta um conjunto extenso de marcadores, os quais permitem definir diversos tipos de conteúdo, como parágrafos (*p*), cabeçalhos (*h1*, *h2*, *h3*, etc), seções (*div*, *section*, *nav*, *footer*, etc), links (*a*), listas (*ul*, *ol*), tabelas (*table*), imagens (*img*), vídeos (*video*) entre outros. As *tags* também podem ser acompanhadas por atributos, que variam com o tipo da *tag*, que são utilizados para definir comportamentos do elemento. A marcação ocorre com o nome da *tag* entre os símbolos "<" e ">". Na maioria dos casos, a marcação necessita de ser repetida para indicar o fim daquele conteúdo, isso é realizado repetindo a marcação ao final do conteúdo, porém com um símbolo de "/" antes do nome da *tag*. Alguns exemplos de uso:

- `<html> </html>`: elemento raiz que indica o início e o fim de uma página HTML;
- `<div> </div>`: define seções (divisões) na página;
- `<form> </form>`: define um formulário com auxílio de outros elementos, como marcadores do tipo "input" para indicar diferentes tipos de entrada do usuário (e.g., `<input type="text">`);
- `<img src="">`: permite a inclusão de imagens, indicadas no atributo "src".

Basicamente, toda página HTML é um documento iniciado pelo seu elemento raiz (i.e., `<html>`) que contém as demais marcações. Esse documento pode ser estruturado em uma árvore hierárquica e acessado programaticamente por meio da interface *Document Object Model* (DOM). A manipulação da árvore DOM pela linguagem JavaScript permite a modificação dinâmica de páginas HTML, o que será demonstrado na Seção 3.2.4 e nos exemplos práticos.

Este capítulo pressupõe que o leitor possua conhecimentos básicos em HTML. Caso seja necessário, o tutorial sobre HTML<sup>1</sup> da MDN Web Docs, um projeto colaborativo da Mozilla (2023), é indicado para referência e estudo.

### 3.2.3. Folhas de Estilo em Cascata

*Cascading Style Sheets* (CSS), ou Folhas de Estilo em Cascata em português, é uma linguagem de estilo para definição da apresentação de elementos HTML. O nome "cascata" refere-se às regras de aplicação dos estilos, que consideram a ordem e a especificidade de seus seletores. A primeira versão do CSS foi proposta em 1996, enquanto sua atual versão, o CSS3, teve o desenvolvimento iniciado em 1999 e têm recebido atualizações e melhorias constantes ao longo dos anos.

O CSS permite estilizar textos, imagens, seções, tabelas, listas, entre outros componentes, considerando aspectos de sua aparência, como tamanho, cor, fonte, fundo, margens, borda e posicionamento na tela. Além disso, a linguagem permite descrever visualizações distintas de acordo com o meio de apresentação (e.g., tela ou impressão) ou a resolução dos dispositivos, flexibilizando a criação de designs adaptativos e responsivos.

---

<sup>1</sup> [developer.mozilla.org/en-US/docs/Web/HTML](https://developer.mozilla.org/en-US/docs/Web/HTML)

Até sua versão 4, o HTML permitia formatar seus elementos por meio de atributos, o que deixava o código poluído e de difícil manutenção, pois alterações visuais ficavam incorporadas às marcações. O HTML5 eliminou esse comportamento, de forma que folhas de estilos se tornaram obrigatórias para a configuração de sua apresentação.

A inclusão do CSS pode ser feita pela inclusão de um arquivo externo ao HTML (forma mais indicada, exemplo: `<link rel="stylesheet" href="/style.css">`) ou pela adição de definições diretamente no código, por meio do marcador “<style>” ou do atributo “*style*”.

A nomenclatura utilizada na construção do CSS segue o padrão *dashed-case*, ou seja, separado por hifens. Declarações são feitas em blocos para um ou mais seletores. Cada bloco declarativo é identificado por um seletor e delimitado por chaves, as quais abrigam propriedades com valores que definem seu estilo, conforme ilustra a Figura 3.3.



**Figura 3.3. Exemplo de um bloco declarativo em CSS**

O seletor indica qual será o alvo de uma declaração. No exemplo da Figura 3 o seletor é a *tag* HTML “*p*”, o que indica que a formatação declarada será aplicada a todos os parágrafos do documento. Seletores podem ser *tags* HTML (e.g., *p*, *h1*, *h2*, *a*, *body*, *div*), identificadores (*#id-do-elemento*) e classes (*.nome-da-classe*) criadas para fins de aplicação de estilo. O seletor também pode ser universal (\*), ou seja, aplicado em todos os elementos. Pode-se associar um único bloco a vários seletores ou realizar combinações entre eles.

A chave define a propriedade CSS que está sendo considerada, enquanto o seu valor é o comportamento aplicado. Cada propriedade aceita valores específicos. Por exemplo, a chave “color” aceita valores relativos à cor, codificados em inglês, hexadecimal ou modelo RGB.

O capítulo espera que o leitor possua conhecimentos básicos em CSS. Caso seja necessário, o tutorial sobre CSS<sup>2</sup> da MDN Web Docs é indicado para referência e estudo.

<sup>2</sup> [developer.mozilla.org/en-US/docs/Web/CSS](https://developer.mozilla.org/en-US/docs/Web/CSS)



### 3.2.4. A linguagem JavaScript

O terceiro pilar do desenvolvimento *front-end*, ou seja, da parte referente ao código executado pelo navegador, é o JavaScript (JS). Diferentemente de HTML e CSS, que são, respectivamente, linguagens de marcação e de estilo, o JavaScript é uma linguagem de programação. É o JS que permite programar o comportamento da interface, habilitando a criação de páginas Web interativas e dinâmicas, ou seja, o conteúdo apresentado na página irá mudar de acordo com a interação do usuário e dos dados armazenados. JavaScript é uma linguagem de *script*, ou seja, seu código roda em um ambiente de execução, sendo interpretado, e não é submetido a um processo de compilação. Os navegadores atuais implementam mecanismos para interpretar *scripts* em JS e, mais recentemente, a plataforma Node.js tornou possível que a linguagem seja utilizada em outros cenários, particularmente na programação de servidores Web.

Criado em 1995, o JavaScript foi bem aceito devido a sua simplicidade e por ser atualizado com frequência. Desde 1997 a linguagem é padronizada pela ECMAScript, a qual garante a interoperabilidade entre as páginas Web e navegadores. Porém, pode-se dizer que o ponto de virada do JS foi em 2005, quando o mecanismo *Asynchronous JavaScript and XML* (AJAX) foi proposto (Garrett, 2005). O objetivo do AJAX é permitir o tratamento de requisições assíncronas por parte do cliente, ou seja, ele possibilita requisitar um conteúdo e tratar seu resultado sem interrupções ou atualizações completas da página. Essa abordagem proporciona um diálogo interativo com o servidor, sem a necessidade de recarregar a página no navegador, o que amplia a experiência do usuário na Web, deixando-a similar a experiência “*desktop*” de um *software* nativo.

A sintaxe da linguagem JavaScript não difere muito de suas predecessoras, como C ou Java, de forma que suas estruturas de blocos, condicionais, de repetição e declaração de funções se assemelham. O Quadro 3.1 apresenta um exemplo de sua sintaxe.

**Quadro 3.1. Exemplo de código JS com estruturas de condição (*if*) e repetição (*for*)**

```
const valor = 'O valor é igual a '  
for (let i = 1; i < 6; i++) {  
  console.log(valor + i)  
  if (i == 5) {  
    console.log('fim...')  
  }  
}
```

A tipagem dinâmica, também referenciada como tipagem “*fraca*”, é um dos diferenciais da linguagem JS, a qual infere o tipo atribuído em tempo de execução. Em JS, a declaração de variáveis ocorre por meio das palavras-chave “*var*” e “*let*”, em escopo global e local respectivamente, e “*const*”, para a declaração de constantes. O Quadro 3.2 exemplifica a declaração de uma variável local que, inicialmente, recebe um valor numérico, imprime esse valor no *log* e, em seguida, seu valor é alterada para um texto, que também é impresso.

**Quadro 3.2. Demonstração da tipagem dinâmica em JS**

```
let a = 3
console.log(a)
a = 'Um texto!'
console.log(a)
```

A tipagem dinâmica exige uma maior cautela quando operações entre variáveis, principalmente advindas de formulários ou serviços, são realizadas. Considerando, por exemplo, que o operador de adição é aplicado tanto para concatenação de textos (*i.e.*, *strings*) como para a soma de números, a soma de um número representado em um formato textual não irá gerar o resultado esperado. O Quadro 3.3 ilustra essa situação. Uma variável com o valor textual “5” é declarada e, ao aplicar o operador de adição com o valor número 5, o resultado é uma concatenação de texto. Para obter um resultado numérico é necessário converter o valor do texto em um inteiro, por meio do método “*parseInt()*”.

**Quadro 3.3. Concatenação ou soma de valores de acordo com sua tipagem**

```
let num = '5'
console.log(num + 5) // o resultado é 55
console.log(parseInt(num) + 5) // o resultado é 10
```

Assim como em outras linguagens, em JS é possível organizar blocos de código em unidades invocáveis, ou seja, encapsular partes do algoritmo em funções. Funções em JS são definidas pela palavra-chave “*function*”, são delimitadas por chaves e podem (ou não) possuir parâmetros e retornar um valor (o retorno não é obrigatório). As funções também podem ser anônimas, o que implica que seu nome não será declarado e a mesma será atribuída a uma variável. A definição de funções é uma das características que demonstra a flexibilidade da linguagem. Há ainda o formato de expressão “*arrow*”, em que a definição de uma função é reduzida, não havendo a necessidade da palavra-chave “*function*”. O Quadro 3.4 exemplifica esses três tipos de função. Priorizando a didática, este documento irá utilizar apenas o formato clássico.

**Quadro 3.4. Três formas diferentes de definir funções em JS**

```
/* Formato tradicional. */
function quadrado_a(num) {
  return num * num
}

/* Função anônima. */
const quadrado_b = function (num) {
  return num * num
}

/* Função arrow. */
const quadrado_c = num => num * num

console.log(quadrado_a(2))
```

```
console.log(quadrado_b(3))
console.log(quadrado_c(4))
```

Outro conceito importante do JS são as funções do tipo “*callback*”. De forma simplificada, uma função “*callback*” é uma função que é passada como argumento para outra função, sendo invocada dentro de outro contexto. Ou seja, funções em JS aceitam funções como argumentos, além de valores e referências. Essas funções podem ser invocadas dentro do bloco de execução daquela que a recebeu. O Quadro 3.5 apresenta um exemplo simples em que uma função chamada “soma” recebe dois valores numéricos e uma função como argumentos, realizando a somatória dos valores e invocando a função recebida com o resultado da operação. Já o Quadro 3.6 ilustra a situação da função “*forEach*”, pertencente ao objeto do tipo “*array*” (lista), a qual recebe uma função como argumento e a invoca para cada item presente na lista.

#### Quadro 3.5. Definição de uma função que aceita uma *callback* como argumento

```
/* Função que escreve um texto na tela. */
function minhaCallback(texto) {
  console.log('Valor: ' + texto)
}

/* Função soma */
function soma(x, y, func) {
  const res = x + y
  func(res)
}

soma(3, 4, minhaCallback)
```

#### Quadro 3.6. Exemplo do *forEach*, o qual recebe uma função *callback* como argumento

```
/* Array (lista) de números. */
const numeros = [23, 44, 1, 8, 71, 92, 33, 47, 55, 60]

/* Função que escreve um texto na tela. */
function minhaCallback(texto) {
  console.log('Valor: ' + texto)
}

/* Percorre a lista chamando a callback. */
numeros.forEach(minhaCallback)

/* A função pode ser escrita diretamente como argumento. */
numeros.forEach(function(num) {
  console.log('Valor: ' + num)
})
```

Ainda no contexto de funções, é importante considerar que essas podem ser síncronas ou assíncronas. No contexto deste tutorial, algumas operações assíncronas são realizadas, como consultas a bancos de dados e a serviços. É imprescindível que tais tarefas sejam assíncronas pois envolvem requisições fora do escopo do programa que

podem ter um retorno demorado. Um dos recursos utilizados para tratar operações assíncronas em JS são as promessas (*Promisses*). Uma *Promise* é um objeto que representa uma conclusão ou uma falha de uma tarefa assíncrona. A conclusão é indicada pela função “*then*”, a qual recebe uma função *callback* como argumento, enquanto o erro é tratado pelo “*catch*”, que também recebe uma *callback*. O Quadro 3.7 demonstra uma chamada para uma função assíncrona “*findAll()*”, da classe “*Client*”, que recebe uma função para tratar os dados retornados e, ao final, trata um possível erro, caso sua execução não seja bem sucedida.

#### Quadro 3.7. Chamada de uma função assíncrona (*findAll*) tratada por meio de *Promise*

```
import Client from '../models/client.model.js'
function findAll(request, response) {
  Client.findAll().then(function(results) {
    /* aqui é realizado o tratamento dos resultados da promise */
  }).catch(function(err) {
    /* aqui é realizado o tratamento de erros */
  })
}
```

Além de suportar os estilos de programação funcional, imperativo e orientada a objetos, o JS permite manipular o DOM de uma página HTML. O DOM organiza os elementos da página HTML em uma árvore. Esses elementos podem ser recuperados pela manipulação da árvore ou, mais comumente, pelas suas características, como identificador (*getElementById*), classe (*getElementsByClass*), nome (*getElementsByName*) ou *tag* (*getElementsByTagName*). Além disso, eventos que mapeiam funções JS podem ser associados aos elementos (e.g., *onClick*, *onMouseOver*, *onLoad*). O Quadro 1.8 exemplifica um código que modifica um texto.

#### Quadro 3.8. Código HTML e JS: um botão que permite alterar o texto do cabeçalho

```
<!DOCTYPE html>
<html>
<body>
  <h1 id="titulo">Bem vindo!</h1>
  <button onclick="mudar()">Clique</button>
</body>
<script>
  const textoNovo = "Pague seus boletos!"
  function mudar() {
    let t = document.getElementById("titulo")
    t.innerText = textoNovo
  }
</script>
</html>
```

Nesse exemplo, a função “*mudar()*”, invocada ao pressionar o botão, recupera o elemento *h1* por meio de seu identificador (i.e., *id*) e altera o seu texto. Esse exemplo poderia conter uma consulta assíncrona a um serviço, o qual enviaria dados textuais (ou até mídias) que poderiam ser utilizados para alterar ou criar elementos na página. Ou

seja, ao identificar e tratar um evento, que pode ser uma ação do usuário ou da própria página, é possível processar esse evento de forma dinâmica e interativa, sem a necessidade de carregamentos de página.

Este capítulo não propõe ser um guia extensivo da linguagem JavaScript, mas utilizará a mesma para construção de um serviço Web com acesso ao banco e uma página que permite listar e cadastrar dados referentes a esse serviço. Para mais detalhes sobre a linguagem, recomenda-se o tutorial da MDN Web Docs<sup>3</sup> (2023).

### 3.3. Arquiteturas Web

A forma como uma aplicação Web é organizada afeta diretamente a escolha das tecnologias utilizadas. Em outras palavras, as soluções arquiteturais, propostas para contemplar demandas e melhorar a qualidade de um produto de *software*, influenciam diretamente na escolha das tecnologias e na proposição de bibliotecas e *frameworks* associados, os quais estruturam a aplicação e agilizam sua implementação.

Entre as arquiteturas aplicadas no desenvolvimento de sistemas Web, a Arquitetura Cliente-Servidor de Duas Camadas (*two-tier*, em inglês) trata-se de uma abordagem simples e clássica, de forma que o sistema é centralizado por motivos de segurança ou simplicidade (Sommerville, 2019). São possíveis diferentes configurações quanto à divisão de processamento (*i.e.*, regras de negócio) entre o cliente e o servidor, porém, o cliente é sempre responsável pela apresentação enquanto o servidor lógico é responsável pela manipulação de dados e acesso ao banco.

Além de apresentar uma baixa manutenibilidade (*i.e.*, qualidade relacionada à facilidade de manutenção), a organização em duas camadas é pouco escalável, dado que a existência de um único componente lógico impede que o trabalho seja distribuído em servidores físicos. Em casos de sistemas de complexidade superior ou que atendam muitas requisições, uma arquitetura multicamadas (*n-tier*) é mais adequada, de forma que a aplicação é dividida em camadas lógicas de acordo com suas funcionalidades.

Em termos de divisão do processamento, a aplicação pode priorizar uma abordagem baseada em um maior processamento pelo cliente ou pelo servidor. No modelo de cliente magro (*thin client*) a maior parte do processamento é gerenciado pelo servidor. Já na abordagem de cliente gordo (*thick client*), a situação é inversa, de forma que a maior parte ou todo o processamento é realizado no cliente com a intenção de reduzir a carga do servidor. Considerando as tecnologias atuais, JavaScript é uma linguagem que permite que o processamento de dados e ações seja realizado no navegador, tornando possível que o modelo cliente gordo seja adotado sem a necessidade de instalação de um *software* adicional.

Por muitos anos a programação de sistemas Web foi estruturada por uma arquitetura *Model-View-Controller* (MVC) com alto acoplamento entre cliente e servidor. As tecnologias utilizadas no *front* e *back-end* eram dependentes e as “páginas dinâmicas” eram geradas no servidor, não no navegador. Um bom exemplo disso é a tecnologia *JavaServer Pages* (JSP), utilizada com a linguagem de programação Java.

---

<sup>3</sup>[developer.mozilla.org/en-US/docs/Learn/Getting\\_started\\_with\\_the\\_web/JavaScript\\_basics](https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/JavaScript_basics)

Por mais que o código fosse inserido nas páginas HTML, o processamento em si era realizado no servidor, o qual era responsável por gerar a visão (*i.e.*, a *View*).

No entanto, com o crescimento da Web e o aumento do número de usuários, novos requisitos surgiram devido à necessidade de maior escalabilidade, componentização e comunicação entre sistemas de diferentes organizações. A Arquitetura Orientada a Serviços (*Service-Oriented Architecture*, SOA) foi proposta como alternativa para esses problemas. A proposta da SOA é o desenvolvimento de sistemas distribuídos em que os componentes são serviços independentes. As principais características dessa arquitetura são (Sommerville, 2019):

- **Acoplamento fraco:** uma mudança efetuada em um componente deve afetar o mínimo possível os outros componentes;
- **Interoperabilidade:** toda troca de mensagem é independente de plataforma;
- **Reuso:** serviços podem ser utilizados por diferentes aplicações clientes, de forma independente;
- **Capacidade de descoberta:** o ambiente de execução e o contrato de um serviço devem ser identificados e vinculados durante a execução.

Nesse contexto, os Serviços Web (*Web Services*, ou WS, em inglês, notação mais utilizada) são a estratégia mais utilizada para a implementação da Arquitetura Orientada a Serviços. Um serviço Web é um componente de *software* reutilizável e fracamente acoplado, que encapsula funcionalidades discretas e que pode ser distribuído e acessado programaticamente. (Mahmoud, 2005). Serviços Web são acessíveis de forma independente da plataforma e da linguagem de programação utilizadas, pois sua comunicação segue padrões para transmissão de dados, considerando protocolos da Internet e arquivos textuais (geralmente em XML ou JSON). Esses serviços podem ser implementados utilizando o *Simple Object Access Protocol* (SOAP) ou o *Representational State Transfer* (REST), conforme discutido a seguir.

O SOAP trata-se de uma especificação para troca de informações estruturadas baseada em mensagens XML. Apesar de robusto e seguro, o SOAP adiciona um *overhead* considerável em forma de metainformação às suas mensagens, de modo que a serialização, a desserialização e a troca de mensagens é dispendiosa. Por esse motivo, a tecnologia é chamada de “Big Web Services”.

Outra solução para implementação de serviços é o REST, um estilo arquitetural baseado na transferência de representações de recursos de um servidor para um cliente (Fielding, 2000). A proposta do REST é utilizar apenas o protocolo HTTP e os métodos já implementados pelo mesmo, mais especificamente os verbos GET, PUT, POST e DELETE, para definir as ações a serem aplicadas em um recurso, enquanto os recursos são definidos em mensagens textuais no formato JSON ou XML. Devido a essa proposta enxuta, serviços REST envolvem uma sobrecarga menor do que os definidos pela especificação SOAP. Os princípios do estilo arquitetural REST são (Fielding, 2000):

1. **Cliente-Servidor:** cliente e servidor são desacoplados e comunicam-se por uma interface uniforme, sendo o desenvolvimento entre as partes é independente;

2. **Sem Estado:** o servidor não guarda o estado da aplicação para nenhum cliente, cada requisição deve enviar toda informação necessária para seu entendimento;
3. **Cache:** resultados gerados podem ser armazenados temporariamente para atender requisições iguais;
4. **Sistema em camadas:** a arquitetura do sistema é composta por camadas hierárquicas;
5. **Código sob Demanda:** servidores podem retornar pedaços de código (*scripts*) para serem executados no cliente;
6. **Interface Uniforme:** característica central do padrão, definida por quatro restrições de interface:
  - a. **Identificação de recursos:** recursos devem ser acessados por meio de endereços únicos (URI);
  - b. **Representação de recursos:** cliente e servidor devem compreender o formato padrão (dados e metadados);
  - c. **Mensagens auto descritivas:** cada requisição do cliente e cada resposta do servidor devem ser uma mensagem padronizada que contém dados que a descrevem;
  - d. **Hipermídia:** links devem ser usados dentro de recursos para encontrar outros recursos.

O estilo REST também pode ser uma abordagem para criação de microsserviços. Microsserviço é uma abordagem arquitetônica para desenvolver uma única aplicação dividida em pequenos serviços, construídos através de pequenas responsabilidades e publicados de forma independente. Cada serviço é executado em um processo próprio, os quais se comunicam por meio de mecanismos leves (LEWIS; FOWLER, 2014).

Na arquitetura de microsserviços o *back-end* é fragmentado em várias partes com escopos bem definidos. Trata-se de uma arquitetura altamente distribuída, que pode incluir pacotes independentes e diversos bancos de dados, sendo que o sistema pode ser facilmente descentralizado em múltiplos servidores físicos. Essas diferentes partes podem ser implementadas em linguagens distintas, de acordo com as demandas advindas de seus requisitos. É importante destacar que nem todo serviço Web é um microsserviço: é possível que a aplicação seja constituída por apenas um (grande) serviço monolítico, centralizado em um único pacote e um único banco.

O tutorial proposto neste capítulo utiliza o REST para implementar serviços Web com a linguagem JavaScript na plataforma Node.js. Destaca-se que essa abordagem pode ser utilizada tanto para construção de um serviço único como para uma implementação descentralizada, em que várias instâncias são produzidas para gerar microsserviços completamente independentes.

### 3.4. Programação Back-end com JavaScript

A plataforma Node.js permite a execução de código JavaScript e o desenvolvimento de aplicações do lado servidor em diferentes sistemas operacionais. Nesta seção será explorada a criação de um serviço Web no estilo REST utilizando JS no ambiente Node.js.

### 3.4.1. Instalação do software necessário

Para seguir este tutorial é necessário baixar o Node.js pelo seu site oficial<sup>4</sup> em versão compatível com o seu Sistema Operacional (SO). Ao instalar o Node.js você também deverá instalar o *Node Package Manager* (NPM), incluído no pacote de instalação. Os códigos apresentados foram testados no Node.js versão 21.1.0 e NPM versão 10.2.0. Para verificar as versões instaladas, execute os comandos “*node -v*” e “*npm -v*”, respectivamente, no terminal do SO. O ambiente de desenvolvimento recomendado é o Visual Studio Code<sup>5</sup> (VSCoDe), mas é possível utilizar qualquer editor ou ambiente de sua preferência. O banco utilizado será o PostgreSQL em sua versão 16, caso se utilize outro banco relacional, é necessário instalar a biblioteca adequada.

A instalação e manejo de bibliotecas é uma atividade essencial em aplicações do lado servidor. O NPM é um gerenciador de pacotes que facilita a instalação e o gerenciamento de bibliotecas que podem ser usadas em aplicações executadas em Node.js. Sendo assim, ele será uma ferramenta essencial no projeto apresentado, sendo utilizado para a instalação das bibliotecas empregadas pela aplicação. O NPM não é a única opção de gerenciador de pacotes para Node.js, sua principal alternativa é o Yarn<sup>6</sup>.

### 3.4.2. Inicialização do projeto

Após a instalação das ferramentas, pode-se iniciar a criação do projeto. Para isso, crie um diretório para o projeto e execute o comando “*npm init*” no terminal, já dentro do diretório de destino. O comando irá gerar um processo interativo que solicita informações sobre o projeto, como nome, versão, autor, descrição e licenças. Ao finalizar, o NPM criará um arquivo “*package.json*” que contém as informações sobre o projeto. Esse arquivo é indispensável pois irá permitir o gerenciamento das bibliotecas instaladas pelo seu criador e colaboradores.

O próximo comando do NPM a ser utilizado será o “*npm install*” ou “*npm i*”. Esse é o comando utilizado para instalar pacotes no projeto, de forma que o NPM procura o(s) pacote(s) especificado(s) em seu repositório e o(s) baixa para o diretório “*node\_modules*” dentro projeto. O comando pode ser executado indicando o pacote desejado, por exemplo, o comando “*npm i express*” instala a versão mais recente e estável do *framework Express*, ou sem nenhum argumento. Nesse último caso, o NPM irá baixar as bibliotecas listadas no arquivo “*package.json*”. A cada instalação, o pacote baixado é adicionado à seção “*dependencies*” do arquivo “*package.json*”.

Para o projeto descrito será necessário instalar as seguintes dependências: *express*, *sequelize*, *pg* e *cors*. Isso pode ser realizado pelo comando “*npm i express sequelize pg cors*”. Cada um desses pacotes é explicado a seguir.

O Express<sup>7</sup> trata-se de um *framework* de aplicações Web que fornece uma maneira conveniente e flexível de criar e gerenciar aplicações. O Express fornece várias funcionalidades, como gerenciamento de rotas, tratamento de erros, integração com banco de dados e recursos de segurança, automatizando tarefas repetitivas de

---

<sup>4</sup> nodejs.org

<sup>5</sup> code.visualstudio.com

<sup>6</sup> yarnpkg.com

<sup>7</sup> https://expressjs.com/



desenvolvimento. Em particular, neste tutorial o *framework* irá auxiliar na criação de um serviço Web baseado em *Application Programming Interfaces* (APIs) do tipo REST.

O Sequelize<sup>8</sup> é uma ferramenta de Mapeamento Objeto-Relacional (ORM) (*Object Relational Mapper*, em inglês), técnica de programação que permite relacionar objetos com dados em um banco de dados. Ao utilizar uma ferramenta ORM, o desenvolvedor não precisa se preocupar com a escrita de código SQL, dado que a técnica realiza o mapeamento dos objetos em tabelas e dados no banco, automaticamente.

Por fim, as bibliotecas “*pg*”<sup>9</sup> e “*cors*”<sup>10</sup> serão utilizadas para conexão com o banco PostgreSQL e para configuração do *Cross-Origin Resource Sharing*, respectivamente.

Para finalizar a configuração do projeto, é necessário incluir a linha “*type*”: “*module*” no arquivo “*package.json*”. Essa configuração é essencial para se utilizar o novo sistema de importação definido pela ECMAScript 6 (ES6). O Quadro 3.9 mostra o arquivo de configuração resultante.

**Quadro 3.9. Arquivo *package.json* após instalação dos pacotes via *npm***

```
{
  "name": "pet-app-back-pg",
  "version": "1.0.0",
  "description": "back-end do pet-app",
  "main": "index.js",
  "type": "module",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "bruna",
  "license": "ISC",
  "dependencies": {
    "cors": "^2.8.5",
    "express": "^4.18.2",
    "pg": "^8.11.3",
    "sequelize": "^6.35.1"
  }
}
```

### 3.4.3. Início do Código

Após a configuração do projeto pelo NPM, é possível iniciar a implementação da aplicação. Primeiramente, é proposta a criação de um serviço de teste que retorna apenas um texto, indicando que a aplicação está sendo executada e está acessível. Para isso, crie um diretório “*src*” dentro do diretório do projeto: esse será o diretório que irá abrigar todo o código fonte em JS. Em seguida, crie um arquivo “*app.js*” com o código do

<sup>8</sup> <https://sequelize.org/>

<sup>9</sup> <https://www.npmjs.com/package/pg>

<sup>10</sup> <https://www.npmjs.com/package/cors>

Quadro 3.10. Para iniciar a aplicação basta executar o comando “`node ./src/app.js`” no terminal (ou utilizar uma extensão como o *Code Runner*<sup>11</sup> no VSCode). O resultado pode ser acessado pela URL: `http://localhost:3000/hello`.

Observe que o código do Quadro 3.10 importa o *express* (conforme o padrão ES6) e inicializa uma instância do mesmo em uma constante, nomeada “*app*”. Essa constante é chamada para definir um serviço, mapeado pelo método GET e pela URI “/hello”, e interceptado por uma função *callback* que retorna o texto “*Hello World!*”. Esse processo é chamado de roteamento, de forma que o *framework* irá mapear um método HTTP e uma URI a uma função desejada. Em resumo, o roteamento realizado pela instância do Express é feito por meio de um método HTTP (verbos GET, PUT, POST e DELETE, conforme discutido na Seção 3.3) que requer como argumentos, minimamente, um caminho (URI do recurso) e uma função que será executada. A função roteada receberá os atributos “*request*” e “*response*”, referentes à sua chamada. Ao final do código, é indicado que a aplicação deve ouvir as requisições na porta 3000 do servidor (essa é a porta padrão recomendada).

**Quadro 3.10. Código fonte do arquivo *app.js*: exemplo de um primeiro serviço do tipo REST.**

```
import express from 'express'

const app = express()
const port = 3000
app.get('/hello', function(request, response) {
  response.send('Hello World!')
})
app.listen(port, function() { console.log('Servidor na porta ' + port) })
```

O próximo passo do tutorial envolve a criação do Banco de Dados e dos modelos que irão gerar as tabelas no banco.

### 3.4.4. Modelagem de dados

O serviço que será implementado permitirá cadastrar clientes e seus animais de estimação, de forma que é preciso representar essas informações em duas tabelas, relacionadas entre si. Como a proposta do tutorial é utilizar uma ferramenta de ORM, no caso, a biblioteca Sequelize, é necessário apenas a criação de um usuário e um banco, os quais serão informados no momento da conexão. Como mencionado na etapa de instalação, o tutorial se baseia no PostgreSQL (versão 16.1.1). Considera-se que o banco foi instalado na porta 5432. Recomenda-se criar um usuário e um banco dedicados à aplicação, conforme mostra o Quadro 3.11. Caso prefira, essa criação pode ser realizada também pelo *software* gráfico *pgAdmin*, o qual acompanha a instalação.

**Quadro 3.11. Comandos para criação de usuário e banco via terminal (Unix). Usuário “userdev”, senha “123456” e banco “devpet”**

```
psql -d postgres
```

<sup>11</sup> <https://marketplace.visualstudio.com/items?itemName=formulahendry.code-runner>

```
CREATE ROLE userdev WITH LOGIN PASSWORD '123456';
ALTER ROLE userdev CREATEDB;
\q
psql -d postgres -U userdev
CREATE DATABASE devpet;
```

Retornando ao código JS, dentro do diretório “*src*”, crie um arquivo “*model.js*”. No ambiente Node.js cada arquivo é visto como um módulo JS e esse será o módulo que irá conter a conexão com o banco e criação dos modelos. O código é apresentado no Quadro 3.12. Inicialmente, são importadas as classes necessárias da biblioteca Sequelize. Em seguida, o primeiro passo cria uma conexão com o banco ao instanciar um objeto, nomeado “*database*”, do tipo Sequelize com os dados (nome do banco, usuário e senha) do banco recém-criado, além da indicação do dialeto (*postgres*) e domínio (*localhost*). Tanto na inicialização da conexão como na instanciação dos modelos é requerida cautela, pois, o uso de separadores, como colchetes e vírgulas, deve ser respeitado. Por exemplo, ao instanciar um novo objeto Sequelize, os primeiros argumentos são configurações textuais de banco, usuário e senha, enquanto o terceiro argumento trata-se de um conjunto de opções delimitado por chaves.

Ainda no Quadro 3.12, após a instanciação do objeto “*database*”, são inicializados os modelos de cliente e animal de estimação, as classes “*Client*” e “*Pet*”, respectivamente. Ambas as classes estendem a classe “*Model*” da biblioteca do Sequelize e em seguida utiliza o método herdado “*init*” para definição de suas tabelas. A definição de uma tabela se dá pela indicação de seus atributos (colunas) e, ao final, das configurações da tabela. Por exemplo, o modelo de “*Pet*” inclui os atributos “*id*, *name*, *type*, *breed* e *birth*”, sendo *id* a chave-primária (um inteiro auto incrementável), os demais são atributos textuais, sendo *name* e *type* duas colunas que não aceitam valores nulos. Novamente, é importante destacar que é necessário cautela na colocação dos separadores.

Ao final da chamada de “*init*”, nas opções do modelo, é indicado um objeto Sequelize, o qual é a instância de conexão, nesse caso, o objeto “*database*”, e demais opções. No exemplo, indica-se que *timestamps* não são desejadas. Outra opção comum é nomear a tabela, o que é feito pela propriedade “*tableName*”. Quando a tabela correspondente ao modelo não é nomeada, a biblioteca pluraliza o nome do modelo (e.g., *Pet* se torna a tabela *Pets*).

**Quadro 3.12. Arquivo *models.js***

```
import { Sequelize, Model, DataTypes } from 'sequelize'

const database = new Sequelize('devpet', 'userdev', '123456',
  { dialect: 'postgres', host: 'localhost' })

class Client extends Model {}
Client.init( {
  id: { type: DataTypes.INTEGER, autoIncrement: true, primaryKey: true },
  name: { type: DataTypes.STRING, allowNull: false },
  document: { type: DataTypes.STRING, allowNull: false }
```

```

}, { sequelize: database, timestamps: false })

class Pet extends Model {}
Pet.init({
  id: { type: DataTypes.INTEGER, autoIncrement: true, primaryKey: true },
  name: { type: DataTypes.STRING, allowNull: false },
  type: { type: DataTypes.STRING, allowNull: false },
  breed: DataTypes.STRING,
  birth: DataTypes.STRING
}, { sequelize: database, timestamps: false })

Pet.belongsTo(Client)
Client.hasMany(Pet)
export default { Client, Pet }

```

Após definidos os modelos, as relações entre eles podem ser adicionadas. O Sequelize suporta as associações clássicas de Um-Para-Um, Um-Para-Muitos e Muitos-Para-Muitos. Isso pode ser realizado por meio dos comandos: *hasOne*, *belongsTo*, *hasMany* e *belongsToMany*. No código do exemplo (Quadro 3.12), o cliente pode possuir vários animais de estimação (*pets*), enquanto um *pet* deve ser de responsabilidade de um cliente. Esse relacionamento foi indicado pelas operações *Pet.belongsTo(Client)* e *Client.hasMany(Pet)*. Como resultado, a tabela *Pets* no banco irá conter uma chave estrangeira para a tabela *Clients*, chamada *ClientId*. A chave estrangeira também pode ser renomeada, utilizando a opção *foreignKey* na definição da relação *belongsTo*. Note que a relação foi estabelecida em duas vias, o que permitirá a recuperação de informações do cliente pelo modelo *Pet*, e a recuperação de *pets* pelo modelo *Client*.

Por fim, para que os modelos criados sejam acessíveis em outros módulos do projeto, ambos são exportados pelo comando *export default*.

Para sincronizar os modelos implementados, foi criado um arquivo (*dbsync.js*) que os importa e invoca o método “*sync*” (Quadro 3.13). Esse método sincroniza as tabelas com o banco, criando-as caso não existam. Caso as tabelas existam, pode-se utilizar a opção “{ *alter: true* }” para forçar uma atualização do banco (não indicado para sistemas em produção). O código pode ser executado pelo comando “*node ./src/dbsync.js*”.

### Quadro 3.13. Sincronização dos modelos (arquivo *dbsync.js*)

```

import model from './model.js'
await model.Client.sync()
await model.Pet.sync()

```

A biblioteca do Sequelize permite outros comandos para criação dos modelos. Sua documentação pode ser consultada em seu site oficial<sup>12</sup>. Além disso, caso se utilize outro banco, como MySQL, é necessário instalar a biblioteca adequada (via NPM) e apontar o dialeto correspondente na instância do objeto Sequelize. A mudança do

<sup>12</sup> <https://sequelize.org/docs/v6/core-concepts/model-basics/>

número da porta também implica que essa deve ser indicada entre os parâmetros de configuração.

### 3.4.5. Controlador

Com os modelos prontos, pode-se realizar as ações de criação, leitura, alteração e remoção dos dados no banco. Como visto anteriormente, esse processo pode ser realizado ao mapear um método REST, disponibilizado pela API do Express, a uma função *callback*, técnica chamada de roteamento. No entanto, priorizando a organização do código, será criado um módulo controlador, o qual conterà as funções roteadas que farão uso do modelo. Nesse exemplo, foi criado apenas um modelo e apenas um controlador, porém, esses podem ser divididos em vários módulos, cada um dedicado a uma entidade.

O Quadro 3.14 apresenta parte do código do controlador (*controller.js*), referente às funções relacionadas ao modelo do cliente. Todas as funções do controlador já são exportadas por padrão no início do módulo. Observe que todas as funções recebem os argumentos *“request”* e *“response”*, pois, conforme explicado na Seção 3.4.2, elas serão roteadas pela aplicação. Esses atributos irão conter os dados da requisição e da resposta às mensagens HTTP (conforme discutido na Seção 3.2.1), respectivamente, de forma que é possível recuperar informações da requisição e determinar os dados de retorno.

Para realizar as operações no banco, são utilizados os métodos gerados para os modelos pela biblioteca do Sequelize. Por exemplo, o método *“findAll()”* recupera todos os registros no banco, enquanto o *“findByPk(id)”* retorna um registro específico. Portanto, a utilização desses métodos é feita de acordo com a documentação de sua biblioteca (a qual é acessível no próprio VSCode). Por envolverem acesso ao banco, todos esses métodos são executados de forma assíncrona, o que implicou em seu tratamento via promessas. Nesse caso, o resultado de cada promessa é referente a uma operação realizada no banco e que pode ser convertido e enviado em formato JSON no objeto *response*. Já os valores de entrada são recuperados pelo objeto *request*.

Como mencionado, uma mensagem de requisição HTTP pode ou não apresentar um corpo. Mensagens do tipo GET não apresentam um corpo, de forma que são utilizados parâmetros na URI para o envio dados na requisição. Isso pode ser observado na função *“findClientByPk”* no Quadro 3.14. Ao utilizar o método de busca de um registro pelo *id*, o valor de *id* é recebido via parâmetro, sendo recuperado ao acessar o *“request.params.id”*. Nas operações de alteração e remoção o parâmetro *id* é utilizado para formar uma cláusula de consulta, determinada pela condição *“where: { id: request.params.id }”*.

Por outro lado, quando o verbo POST é utilizado, o corpo da mensagem contém dados. No Quadro 3.14, a função *“createClient”* utiliza o método *“create”* do modelo e recupera informações do corpo da requisição para determinar os valores dos atributos *“name”* e *“document”*. Nesse caso, espera-se que os valores recebidos estejam em *“request.body.name”* e *“request.body.document”*.

Considerando a resposta, é comum que sejam retornados dados em formato JSON, de forma que na maioria dos casos foi utilizado o comando *“response.json(result)”*. Em algumas situações pode ser adequado um resultado vazio ou até mesmo uma mensagem de texto, o que pode ser obtido pelo método *“send()”*. Essas

chamadas podem ser encadeadas com o *status* da resposta. O *status* retornado deve estar de acordo com a convenção para mensagens HTTP. *Status* na casa de 200 indicam sucesso, enquanto na casa de 400 são requisições malformadas e na casa de 500 são relacionados a erros no servidor.

**Quadro 3.14. Código fonte do módulo *controller.js* (parte 1)**

```
import model from './model.js'
export default { findAllClients, findAllPets, findClientByPk, findPetByPk, findPetsOfClient,
createClient, createPet, updateClient, updatePet, deleteClientByPk, deletePetByPk }

function findAllClients(request, response) {
  model.Client.findAll().then(function (results) {
    response.json(results).status(200)
  }).catch(function (e) { console.log(e) })
}

function findClientByPk(request, response) {
  model.Client.findByPk(request.params.id).then(function (result) {
    response.json(result).status(200)
  }).catch(function (e) { console.log(e) })
}

function createClient(request, response) {
  model.Client.create(
    { name: request.body.name, document: request.body.document }
  ).then(function (result) {
    response.status(201).json(result)
  }).catch(function (e) { console.log(e) })
}

function updateClient(request, response) {
  model.Client.update(
    { name: request.body.name, document: request.body.document },
    { where: { id: request.params.id } }
  ).then(function (result) {
    response.status(200).send()
  }).catch(function (e) { console.log(e) })
}
```

```

function deleteClientByPk(request, response) {
  model.Client.destroy({ where: { id: request.params.id } })
    .then(function (result) {
      if (result == 1) {
        response.status(200).send()
      } else {
        response.status(404).send()
      }
    }).catch(function (e) { console.log(e) })
}

```

O Quadro 3.15 contém a continuação do código do controlador, referente às operações com o modelo *Pet*. Observe que as tarefas são similares, com exceção da função “*findPetsOfClient*”, a qual retorna os animais de estimação de um cliente específico. Nesse caso é utilizado o método “*findAll*” do modelo com uma cláusula de consulta que indica que deverão ser retornados apenas os registros cuja chave-estrangeira seja igual a chave do cliente indicado no parâmetro da requisição (*i.e.*, *where: { ClientId: request.params.id }*).

**Quadro 3.15. Código fonte do módulo *controller.js* (parte 2)**

```

function findAllPets(request, response) {
  model.Pet.findAll().then(function (results) {
    response.json(results).status(200)
  }).catch(function (e) { console.log(e) })
}

function findPetByPk(request, response) {
  model.Pet.findByPk(request.params.id, { include: model.Client })
    .then(function (result) {
      response.json(result).status(200)
    }).catch(function (e) { console.log(e) })
}

function findPetsOfClient(request, response) {
  model.Pet.findAll({ where: { ClientId: request.params.id } })
    .then(function (results) {
      response.json(results).status(200)
    }).catch(function (e) { console.log(e) })
}

function createPet(request, response) {
  model.Pet.create({
    name: request.body.name, type: request.body.type,
    breed: request.body.breed, birth: request.body.birth, ClientId: request.body.ClientId }

```

```

).then(function (result) {
  response.json(result).status(201)
}).catch(function (e) { console.log(e) })
}

function updatePet(request, response) {
  model.Pet.update({
    name: request.body.name, type: request.body.type,
    breed: request.body.breed, birth: request.body.birth, ClientId: request.body.client },
  { where: { id: request.params.id } })
  .then(function (result) {
    response.json(result).send()
  }).catch(function (e) { console.log(e) })
}

function deletePetByPk(request, response) {
  model.Pet.destroy({ where: { id: request.params.id } })
  .then(function (result) {
    if (result === 1) { response.status(200).send()
    } else { response.status(404).send()
    }
  }).catch(function (e) { console.log(e) })
}
}

```

### 3.4.6. Rotas

Com o controlador pronto, podem ser indicadas as rotas da API. Isso poderia ser implementado diretamente no arquivo “*src/app.js*”, no entanto, visando uma melhor organização, foi criado o arquivo individualizado “*src/routes.js*”. Nesse tutorial, há apenas um módulo de rotas, no entanto, podem ser criados vários, o que é indicado para diferenciar funcionalidades (*e.g.*, um módulo dedicado às rotas de autenticação e segurança).

O Quadro 3.16 demonstra a definição das rotas associadas às funções implementadas pelo controlador. O objeto “*routes*” é uma instância da classe “*Routes*” do Express e, por meio desse, são mapeados os métodos e as URIs às funções criadas. Ao final, o objeto “*routes*” é exportado. Observe que nenhuma função recebe argumentos, de forma que apenas seus nomes são indicados. Isso ocorre pois o *framework* irá invocar as funções com os argumentos *request* e *response*. Por exemplo, o cadastro de um cliente é roteado para a função “*createClient*” pelo método “*post*” no recurso “*/clients*”.

Outro ponto essencial do roteamento é a indicação do nome dos parâmetros na URI. No exemplo, as operações de busca, alteração e deleção de recursos fazem uso de um parâmetro *id*, indicado nas URIs por “*:id*” (o caractere de dois pontos é utilizado na definição de parâmetros nas rotas do Express). O nome do parâmetro definido pela rota



deverá ser respeitado tanto pela URI da requisição quanto pela função implementada no controlador.

**Quadro 3.16. Arquivo *routes.js***

```
import express from 'express'
import controller from './controller.js'

const routes = express.Router()

routes.get('/clients', controller.findAllClients)
routes.get('/clients/:id', controller.findClientByPk)
routes.post('/clients', controller.createClient)
routes.put('/clients/:id', controller.updateClient)
routes.delete('/clients/:id', controller.deleteClientByPk)

routes.get('/pets', controller.findAllPets)
routes.get('/pets/:id', controller.findPetByPk)
routes.post('/pets', controller.createPet)
routes.put('/pets/:id', controller.updatePet)
routes.delete('/pets/:id', controller.deletePetByPk)
routes.get('/clients/:id/pets', controller.findPetsOfClient)

export default routes
```

É importante destacar que a nomenclatura aplicada nas rotas do Quadro 3.16 respeita as convenções do estilo REST, ou seja, a pluralização dos nomes dos recursos, a identificação de itens específicos pelo caminho dos recursos e seu identificador (*i.e.*, */recursos/id*) e de recursos relacionados pela composição da URI (*i.e.*, */recursos/id/recursoRelacionado*). A rota definida para a recuperação dos animais de um cliente específico é um exemplo dessa última situação, pois sua URI é determinada pelo caminho desse cliente (*"/clients/:id/pets"*).

### 3.4.7. Finalização da configuração do serviço

Para finalizar a API é necessário incluir as rotas programadas na instância do Express. Além disso, devem ser adicionadas algumas configurações para que a API aceite o formato JSON e as requisições da página que será implementada.

O Quadro 3.17 apresenta a versão final do arquivo *"src/app.js"*. Nele foram importados a biblioteca do CORS, instalada em passos anteriores, e o recém-criado módulo de roteamento (*"src/routes.js"*). A linha *"app.use(cors())"* foi adicionada a fim de permitir requisições *"cross-origin"*, ou seja, basicamente, o CORS permite informar ao navegador que a aplicação terá seus recursos acessados por outra aplicação em uma origem distinta (em termos de domínio). Caso essa linha não seja incluída, será gerado um erro de CORS no navegador quando a página Web, que será executada em um servidor de porta diferente da aplicação Node.js, realizar uma requisição HTTP para a API. As duas linhas que seguem a habilitação do CORS são necessárias para que as requisições do tipo POST e PUT possam enviar dados em JSON no corpo da mensagem

HTTP. Após as configurações, o módulo de rotas é associado à instância do Express, pelo comando “`app.use(routes)`”.

**Quadro 3.17. Arquivo `app.js`**

```
import express from 'express'
import cors from 'cors'
import routes from './routes.js'
const app = express()
const port = 3000
/* Habilita requisições de diferentes origens */
app.use(cors())
/* Permite receber dados em json */
app.use(express.json())
/* Indica como o parser do json será realizado */
app.use(express.urlencoded({extended: true}))

app.use(routes)
app.listen(port, function() { console.log('Servidor rodando na porta ' + port) })
```

### 3.4.8. Testando a API por requisições

Antes de iniciar a criação da página Web que será o cliente da API implementada, é possível realizar testes para garantir que o código implementado não possui erros e está de acordo com os requisitos planejados. Existem ferramentas que simulam clientes REST ao executar requisições iguais àquelas que são enviadas pelos navegadores. Além de permitir apontar o verbo e a URI da requisição, essas ferramentas possibilitam personalizar as mensagens HTTP, incluindo, por exemplo, o corpo da mensagem, e exibir o resultado retornado pelo servidor. A ferramenta que foi utilizada para testar o código do exemplo foi a Thunder Client<sup>13</sup>, a qual pode ser instalada como extensão no VSCode.

A Figura 3.4 mostra a configuração da ferramenta para inclusão de um registro via API pelo método POST. A interface da ferramenta permite a seleção do verbo, nesse caso o POST, e a inclusão da URI do recurso (`localhost:3000/clients`). Na aba *body* é possível enviar dados em diferentes formatos no corpo da mensagem sendo que, no exemplo da figura, foi incluído um JSON com nome e documento do cliente a ser cadastrado. A parte direita da ferramenta mostra o resultado da requisição, conforme definido pela API implementada, que lista o cliente cadastrado com o seu *id* e o *status* 201 (criado). Os clientes cadastrados podem ser recuperados via método GET, utilizado a mesma URI de recurso. A Figura 3.5 mostra o resultado após o cadastro de três clientes.

<sup>13</sup> <https://www.thunderclient.com/>

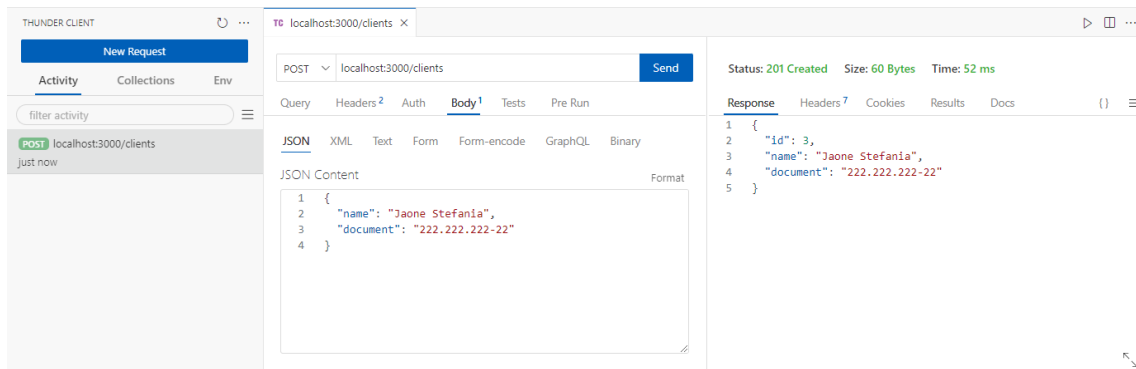


Figura 3.4. Cadastro de um cliente via método POST

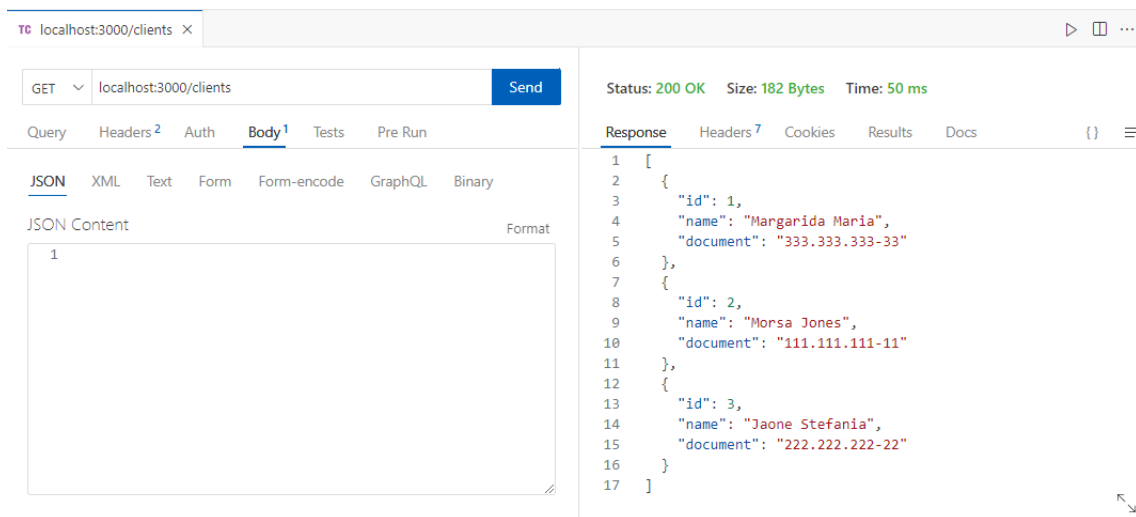


Figura 3.5. Consulta por todos os clientes cadastrados

Na Figura 3.6 é apresentado mais um exemplo em que é feito o cadastro de um animal de estimação para um cliente específico, cujo *id* é 3. O JSON correspondente contém os dados do modelo *Pet*, incluindo a chave estrangeira do cliente. Após essa inserção, é possível buscar pelos animais do cliente de *id* igual a 3, como demonstrado na Figura 3.7 (*localhost:3000/clients/3/pets*).

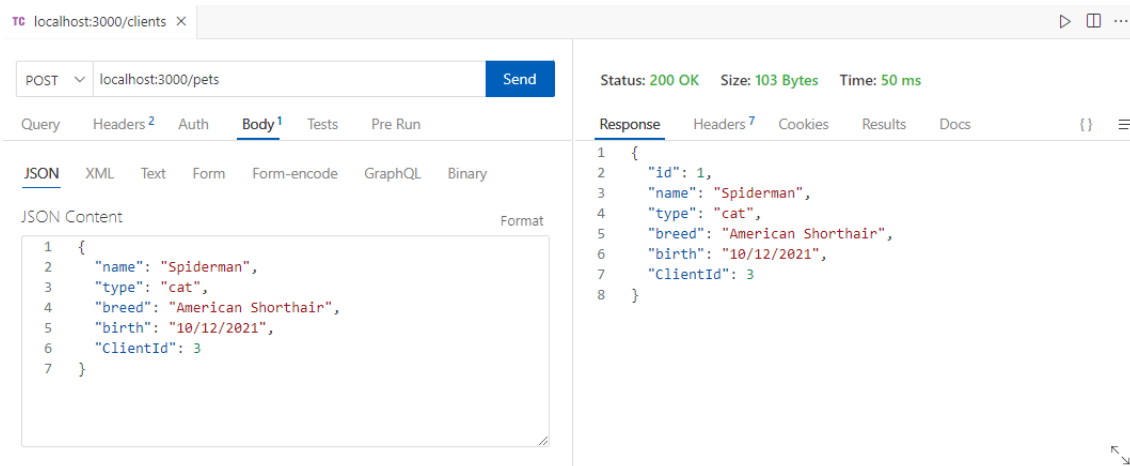


Figura 3.6. Cadastro de um pet via método POST

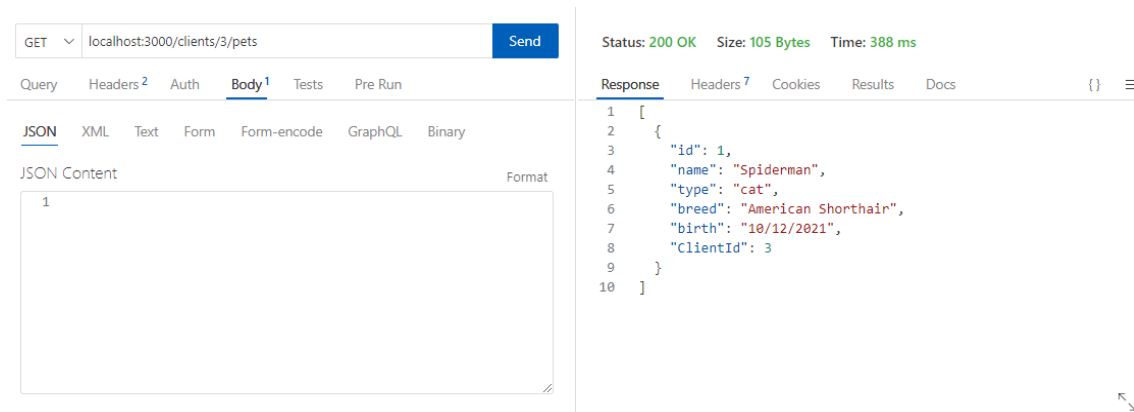
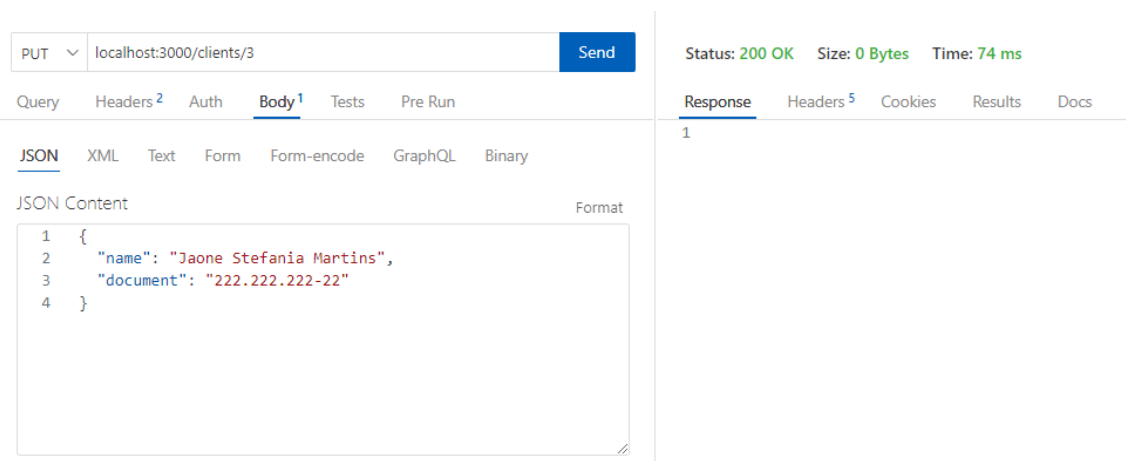


Figura 3.7. Listagem dos pets do cliente de id igual a 3

Operações de alteração e remoção demandam que a URI aponte para um recurso de *id* específico. A Figura 3.8 exemplifica a operação de PUT na URI do cliente de *id* 3 (*localhost:3000/clients*) para alteração de seu nome.



**Figura 3.8. Alteração do nome do cliente de id igual a 3**

Para garantir a corretude do código implementado é recomendando testar todas as funções de cadastro, consulta, alteração e remoção de registros.

### 3.5. Programação Front-end com JavaScript

Esta seção irá demonstrar como criar uma página Web que seja capaz de utilizar o pequeno serviço Web com API REST criado. A linguagem JavaScript apresenta diversos *frameworks* e bibliotecas para a programação *front-end*, no entanto, objetivando o foco na linguagem, esse tutorial utilizará o JS “puro”, ou seja, sem esse ferramental de apoio. O uso da linguagem JS sem o suporte de um *framework* ou biblioteca de grande porte é chamado de *Vanilla JavaScript* pela comunidade. Por ter como alvo de estudo o uso de JavaScript, este tutorial não entrará em detalhes sobre o código HTML e CSS apresentado. As Seções 3.2 e 3.3 apresentam conceitos básicos e indicam um material complementar para o estudo dessas tecnologias.

O Quadro 3.18 contém o código HTML da página (*index.html*). Observe que dois arquivos JS são incluídos em seu cabeçalho. O primeiro (<https://unpkg.com/axios/dist/axios.min.js>) trata-se da biblioteca Axios<sup>14</sup>, a qual será utilizada para realizar requisições HTTP para o serviço Web criado. O Axios simplifica essas chamadas e será utilizado a fim de deixar o código mais claro, porém o JS apresenta uma API própria, chamada *Fetch*, a qual também pode ser utilizada para requisições baseadas em promessas. O próximo script incluído trata-se do código JavaScript que será desenvolvido (*script.js*). Esse código é referenciado logo no início da página pelo evento “*onload*”, o qual invoca a função “*getClients()*”. No cabeçalho também é incluído um arquivo CSS (*style.css*) que é listado no Quadro 3.19.

O código HTML apresenta a estrutura básica de formulários para o cadastro de registros no banco. Inicialmente, o formulário para inclusão de animal do cliente não deverá estar visível na tela, devendo aparecer apenas quando um cliente for aberto. O controle de visibilidade será feito por meio de CSS, mais especificamente pelas classes “*.hide*” e “*.show*”. O Quadro 3.19 contém essas classes e as demais definições de estilo.

<sup>14</sup> <https://axios-http.com/>

Quadro 3.18. Código da página *index.html*

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <script src="https://unpkg.com/axios/dist/axios.min.js"></script>
  <script src="./script.js"></script>
  <link rel="stylesheet" href="./style.css">
  <title>Pet App</title>
</head>
<body onload="getClients()">
  <h1>Pet App</h1>
  <div id="div-clients">
    <h3>Add Client</h3>
    <form>
      <label for="name">Name:</label>
      <input type="text" name="name" id="name" required>
      <label for="document">Document:</label>
      <input type="text" name="document" id="document" required>
      <input type="button" value="Submit" onclick="postClient()">
    </form>
  </div>
  <div id="div-client-pets"></div>
  <div id="div-form-pet">
    <h3>Add New Pet</h3>
    <form>
      <label for="petname">Name:</label>
      <input type="text" name="petname" id="petname" required>
      <label for="type">Type:</label>
      <select name="type" id="type" required>
        <option value="Dog">Dog</option>
        <option value="Cat">Cat</option>
        <option value="Bird">Bird</option>
        <option value="Turtle">Turtle</option>
        <option value="Rabitt">Rabitt</option>
      </select>
      <label for="breed">Breed:</label>
      <input type="text" name="breed" id="breed" required>
      <label for="birth">Birth:</label>
      <input type="date" name="birth" id="birth" required>
      <input type="button" value="Submit" id="submit-pet">
    </form>
  </div>
</body>
</html>

```

Quadro 3.19. Código da folha de estilos *style.css*

```
body {
  font-family: 'Gill Sans', 'Gill Sans MT', Calibri, 'Trebuchet MS', sans-serif;
}
h1 {
  color: white;
  background-color: #4B99AD;
  margin: 0px;
  padding: 10px;
}
.client-list {
  background-color: gainsboro;
  padding-left: 10px;
  border-bottom: 1px solid gray;
}
.pets-list {
  background-color: aliceblue;
  padding-left: 10px;
  border-bottom: 1px solid cyan;
}
.client-list spam {
  cursor: pointer;
}
.client-list p {
  display: inline-block;
  margin-right: 10px;
}
.pets-list p {
  display: inline-block;
  margin-right: 10px;
}
form {
  margin: 10px;
}
button {
  margin-left: 5px;
}
.hide {
  display: none;
}
.show {
  display: block;
}
```

Com a estrutura da página pronta, o código JS correspondente pode ser integrado. O código será contínuo e fará parte do arquivo “*script.js*”, porém, para fins didáticos, ele será dividido em múltiplos quadros. O Quadro 3.20 apresenta a primeira parte, a qual é responsável por listar os clientes cadastrados na tela. A primeira linha do

código define uma constante com o endereço do serviço. Em seguida, vem a função “*getClientes()*”, a qual é chamada pelo evento “*onload*” do “*body*” da página. A função inicialmente garante que o formulário para cadastro de animais está oculto, ao recuperar seu container (*div*) e excluir e adicionar as classes adequadas. Em seguida, é realizada uma requisição do tipo GET pela biblioteca Axios na URI dos clientes, a qual retorna, por meio de promessa, uma resposta ou um erro. No caso de uma resposta, um JSON que contém a lista de clientes é recuperado do objeto de resposta pelo comando “*response.data*”. A lista de clientes é percorrida e, para cada um, é invocada a função “*addClient*”, a qual é apresentada no Quadro 3.21.

**Quadro 3.20. Arquivo *scripts.js* (parte 1): listagem dos clientes**

```
const api = 'http://localhost:3000/'
function getClientes() {
  document.getElementById('div-form-pet').classList.remove('show')
  document.getElementById('div-form-pet').classList.add('hide')
  axios.get(api + 'clients').then(function(response) {
    const clients = response.data
    for(let client of clients) {
      addClient(client)
    }
    console.log(clients)
  })
  .catch(
    function (error) {
      console.error(error)
    }
  )
}
```

O Quadro 3.21 apresenta a continuação do código com a função “*addClient*”, responsável por adicionar cada cliente na tela. Conforme explicado na Seção 3.4, esse comportamento pode ser obtido ao manipular o DOM da página. São criados quatro elementos: (1) um parágrafo, que contém o nome e o documento do cliente; (2) um botão para abrir o cliente; (3) um botão para deletar o cliente; e (4) um container, que agrupa os três elementos anteriores e é anexado a uma *div* já presente no documento. O container também guarda o *id* do cliente, dado que esse será indispensável para sua posterior identificação. Na criação dos botões, note que são atribuídas funções aos eventos de clique por meio da definição de atributo (*i.e.*, pelo método “*setAttribute*”). Essas funções são demonstradas no Quadro 3.22.

Conforme mostra o Quadro 3.21, as funções referenciadas pelos botões criados de forma dinâmica são a de visualização de animais de um cliente (“*getPets*”) e a de remoção do cliente (“*deleteClient*”). Em ambos os casos, o evento associado é o de clique (“*onclick*”) e são incluídos argumentos à chamada das funções. O Quadro 3.22 apresenta a implementação do código. A primeira função apresentada, “*deleteClient(id)*”, recebe o identificador do cliente a ser removido e realiza uma requisição do tipo DELETE, com o auxílio da biblioteca do Axios, na URI dos clientes concatenada com uma barra e um *id*. Na Seção 3.4.5, foi demonstrado que na construção das rotas o *id* é mapeado como um parâmetro na própria URI. Em caso de sucesso, o cliente removido do sistema deve, também, ser removido da interface, o que é executado



pelo comando “`document.getElementById(id).remove()`” (esse comando depende da definição do *id* do container, conforme feito no Quadro 3.21).

**Quadro 3.21. Arquivo *scripts.js* (parte 2): adição de cliente no documento HTML**

```
function addClient(client) {
  const clientItem = document.createElement('p')
  clientItem.innerHTML = client.name + ', ' + client.document

  const viewButton = document.createElement('button')
  viewButton.innerHTML = 'Open'
  viewButton.setAttribute('onclick','getPets(' + client.id + ', "' + client.name + '"')

  const trashButton = document.createElement('button')
  trashButton.innerHTML = 'Delete'
  trashButton.setAttribute('onclick','deleteClient(' + client.id + ')')

  const container = document.createElement('div')
  container.classList.add('client-list')
  container.id = client.id
  container.appendChild(clientItem)
  container.appendChild(viewButton)
  container.appendChild(trashButton)

  const divClients = document.getElementById('div-clients')
  divClients.appendChild(container)
}
```

As próximas funções do Quadro 3.22 tratam da listagem dos animais de estimação do cliente selecionado. Os primeiros comandos tratam da configuração do formulário para adição de um novo animal. Nesse caso, além de configurar a visibilidade do formulário, a função do botão de submissão é definida dado que é necessário que o *id* do cliente, responsável por aquele *pet*, seja enviado como argumento para a função. Em seguida, a biblioteca do Axios é utilizada para realizar uma requisição do tipo GET, nesse caso, na rota definida para o retorno dos animais de um cliente específico (`api + 'clients/' + clientId + '/pets'`). Em caso de sucesso, são listados os animais na tela, utilizando operações similares às utilizadas para a listagem de clientes.

**Quadro 3.22. HTML Arquivo *scripts.js* (parte 3): adição de cliente no documento HTML**

```
function deleteClient(id) {
  axios.delete(api + 'clients/' + id)
  .then(function (response) {
    document.getElementById(id).remove()
  }).catch(function (error) {
    console.error(error)
  })
}
```

```
function getPets(clientId, clientName) {
  document.getElementById('div-form-pet').classList.remove('hide')
  document.getElementById('div-form-pet').classList.add('show')
  document.getElementById('submit-pet').setAttribute('onclick','postPet(' + clientId + ')')

  const divPets = document.getElementById('div-client-pets')
  divPets.innerHTML = ""

  axios.get(api + 'clients/' + clientId + '/pets').then(function (response) {
    const h2 = document.createElement('h2')
    h2.innerText = 'Pets of ' + clientName
    divPets.appendChild(h2)
    const pets = response.data
    for (let pet of pets) {
      addPet(pet)
    }
  }).catch(function (error) {
    console.error(error)
  })
}

function addPet(pet) {
  const p = document.createElement('p')
  p.innerText = pet.name + ', ' + pet.type + ', ' + pet.breed + ', ' + pet.birth
  const container = document.createElement('div')
  container.classList.add('pets-list')
  container.id = pet.id
  container.appendChild(p)
  document.getElementById('div-client-pets').appendChild(container)
}
```

No código HTML apresentado no Quadro 3.18, o formulário dedicado à inclusão de clientes possui um botão que faz referência à função “*postClient()*”, enquanto o Quadro 3.22 possui uma referência à função “*postPet(clientId)*”. O Quadro 3.23 contém o código dessas duas funções, as quais possuem um funcionamento muito similar: ambas recuperam os dados dos formulários, realizam uma requisição do tipo POST passando esses valores no formato JSON e, em caso de sucesso, adicionam o registro no documento e limpam o formulário.

**Quadro 3.23 - Arquivo *scripts.js* (parte 4): funções responsáveis pelo cadastro de clientes e animais de estimação**

```
function postClient() {
  const name = document.getElementById('name').value
  const doc = document.getElementById('document').value
  axios.post(api + 'clients', {
    name: name, document: document
  })
  .then(function (response) {
    addClient(response.data)
    document.getElementsByTagName('form')[0].reset()
  })
  .catch(function (error) {
    console.log(error)
  })
}

function postPet(clientId) {
  const name = document.getElementById('petname').value
  const type = document.getElementById('type').value
  const breed = document.getElementById('breed').value
  const birth = document.getElementById('birth').value
  console.log(clientId)
  axios.post(api + 'pets', {
    name: name, type: type, breed: breed, birth: birth, ClientId: clientId })
  .then(function (response) {
    addPet(response.data)
    document.getElementsByTagName('form')[1].reset()
  })
  .catch(function (error) {
    console.log(error)
  })
}
```

A página resultante trata-se de uma *Single Page Application* (SPA) que permite listar, cadastrar e apagar clientes, assim como listar e cadastrar seus animais de estimação, utilizando o serviço Web desenvolvido para Node.js. A Figura 3.9 ilustra uma tela dessa aplicação. Sugere-se que a página seja estendida para incluir as demais funcionalidades disponíveis no serviço.

## Pet App

### Add Client

Name:  Document:

Margarida Maria, 333.333.333-33

Morsa Jones, 111.111.111-11

Ursula Lamentadora, 234.567.890-11

Jaone Stefania Martins, 222.222.222-22

### Pets of Jaone Stefania Martins

Spiderman, cat, American Shorthair, 10/12/2021

Navi, Dog, Border Collie, 2019-11-11

### Add New Pet

Name:  Type:  Breed:  Birth:

Figura 3.9. Tela da página Web resultante

## 3.6. Conclusão

A introdução e a evolução da linguagem JavaScript foi o que permitiu que páginas Web se tornassem verdadeiras aplicações, com funcionalidades interativas e um processo de comunicação pervasivo com servidores. Essa rápida progressão revolucionou a experiência dos usuários e permitiu que aplicações Web substituíssem diversas aplicações *desktop*. Em consonância, arquiteturas de *software* são propostas para resolver problemas relacionados a atributos de qualidade, os quais são resultados de tendências de desenvolvimento tecnológico, de forma geral. Ao passo que tecnologias associadas evoluíram, o desenvolvimento Web passou a ser orientado por arquiteturas altamente componentizadas e fracamente acopladas. É nesse contexto que se encaixam as arquiteturas orientadas a serviço. Considerando o estilo arquitetural REST e o uso da linguagem JavaScript, este capítulo se propôs a apresentar conceitos fundamentais para o desenvolvimento Web *back-end* e *front-end*. O uso da mesma linguagem nas duas frentes é uma vantagem em termos operacionais, dado que facilita a atuação de desenvolvedores em diferentes âmbitos. Nesse sentido, é importante apontar que o uso do JS para a implementação de um servidor Web depende do uso da plataforma Node.js, enquanto o lado cliente permite o uso da linguagem sem nenhum componente adicional. Todavia, o desenvolvimento *front-end* pode ser beneficiado em vários aspectos quando um *framework* ou biblioteca (e.g., Vue.js ou React) é incorporado ao projeto. A manutenibilidade é um desses principais aspectos, considerando que essas tecnologias aprimoram a componentização e o reuso de código.

### 3.7. Referências

- Berners-Lee, T. and Fischetti, M. (1999) “Weaving the Web: The original design and ultimate destiny of the World Wide Web by its inventor” Harper San Francisco.
- Fielding, R. T. (2000) “Architectural styles and the design of network-based software architectures”. University of California, Irvine.
- Garrett, J. J. et al. (2005) “Ajax: A new approach to web applications”.
- Marinho, A. L. and Da Cruz, J. L. (org.). (2020) “Desenvolvimento de aplicações para Internet”. 2º ed. São Paulo: Pearson.
- Mahmoud, Q. (2005) “Service-Oriented Architecture (SOA) and Web Services: The Road to Enterprise Application Integration (EAI)”. Disponível em: <<https://www.oracle.com/technical-resources/articles/javase/soa.html>>. Acesso realizado em: 30 de ago. de 2023.
- Mozilla Foundation. “MDN Web Docs.” Disponível em: <<https://developer.mozilla.org/en-US>>
- Sommerville, I. (2019) “Engenharia de Software”. 10º ed. São Paulo: Pearson.

## Capítulo

# 4

## Submissão de Projetos de Pesquisa Web e Multimídia a Comitês de Ética em Pesquisa

<sup>1</sup>Maria da Graça Campos Pimentel, <sup>2</sup>André Pimenta Freire,  
<sup>3</sup>Luiz Paulo Carvalho, <sup>1</sup>Kamila Rios da Hora Rodrigues

<sup>1</sup>Universidade de São Paulo (USP/ICMC)

<sup>2</sup>Universidade Federal de Lavras (UFLA)

<sup>2</sup>Universidade Federal do Rio de Janeiro (UFRJ)

mgp@icmc.usp.br, apfreire@dcc.ufla.br

luiz.paulo.carvalho@ppgi.ufrj.br, kamila.rios@icmc.usp.br

### **Abstract**

*This communication aims to advance the training or understanding of researchers about when, how, and why to submit a research project to a Human Research Ethics Committee (CEP), improving the quality of academic and scientific studies in the Web and Multimedia domain, also relevant to other interested parties. The following are potential capabilities for the person reading this communication: 1. Develop their moral and legal notion about submitting research projects to a CEP; 2. Submit appropriately, avoiding future regulatory problems for the research and its communication, e.g., dissemination of results, publication, or even retractions; 3. Develop your understanding of submitting research projects to a CEP and respective documentation and deadlines; 4. Improve the quality of filling out submission forms; 5. Develop effective communication with the CEP; 6. Understand practical cases in the area of Web and Multimedia.*

### **Resumo**

*Esta comunicação tem como objetivo avançar a capacitação ou compreensão de pessoas pesquisadoras sobre quando, como e porque submeter um projeto de pesquisa envolvendo seres humanos para um Comitê de Ética em Pesquisa, melhorando a qualidade dos estudos acadêmicos e científicos na área de Web e Multimídia, entre outras. São capacitações potenciais à pessoa leitora desta comunicação: 1. Desenvolver sua noção moral, ética e legal sobre submeter projetos de pesquisa para um CEP; 2. Submeter no momento apropriado, evitando futuros problemas normativos para a pesquisa e publicações resultantes; 3. Desenvolver sua compreensão sobre o processo de submissão de projetos de pesquisa para um CEP, respectivas documentações e prazos; 4. Melhorar a qualidade do preenchimento de formulários de submissão; 5. Desenvolver uma comunicação eficaz com o CEP; 6. Aprender casos práticos na área de Web e Multimídia.*

## 4.1. Introdução

Há mais de uma década, King [1] já alertava para a crescente responsabilidade de pesquisas em computação que envolvem seres humanos. Nesse contexto e na mesma época, Estes et al. [2] mostraram os resultados positivos obtidos em um curso de graduação em computação no qual alunos tinham que passar pelos processos de *treinamento em proteção a seres humanos* e de submissão de projeto a um Comitê de Ética em Pesquisa (CEP) para realizar seus projetos de interfaces de sistemas interativos. Em particular no contexto da Web, Phelps-Hillen [3] naquela época já advogava pela necessidade submeter a CEP as pesquisas envolvendo textos publicados por internautas em redes sociais, necessidade essa persiste até hoje como destacado no *survey* de Fiesler et al. [4]. Apesar de toda essa história, ainda hoje no Brasil é tímida a presença de pesquisas envolvendo participação humana que atendem o requisito de aprovação de projeto a CEPs, por exemplo nas áreas de Web e Multimídia [5], Interação Humano-Computador [6, 7], Qualidade de Software [8] e Sistemas de Informação [9].

A regulamentação de pesquisas envolvendo seres humanos no Brasil remonta a 1996 [10] e avançou em 2012 com a criação do Sistema CEP/Conep, uma estrutura nacional destinada a proteger os participantes de pesquisas no país [11]. Este documento tem como objetivo fornecer orientações para o processo de submissão de projetos de pesquisa que envolvam ou incluam a participação humana ao Sistema CEP/Conep.

Inicialmente, apresentamos os conceitos de ética e moral como discutidos por Feil and Paz [12], os quais abordam a ética como sinônimo de moral e como legislação, estatuto ou código. A seguir, discutimos a regulamentação e os mecanismos que determinam a adoção de procedimentos normativos rigorosos para proteger participantes e garantir a qualidade da pesquisa, para uma condução ética e responsável da pesquisa científica. Ao final, incluímos exemplos da área de Web e Multimídia. Este documento detalha:

- **Por quê:** A submissão é necessária para uma avaliação interdisciplinar externa sobre projetos de pesquisa. Evita ou mitiga, dentre outros, vieses das pessoas pesquisadoras ao analisar os riscos oferecidos às pessoas participantes de seus estudos. O CEP atua como um órgão independente das pessoas pesquisadoras, que compartilha a responsabilidade de garantir a segurança e bem-estar das pessoas participantes das pesquisas aprovadas;
- **Quem:** A submissão pode ser realizada por pessoas pós-graduandas e pesquisadoras. No caso de estudantes de graduação, a submissão é feita pelas respectivas pessoas orientadoras;
- **Quando:** O projeto de pesquisa deve ser submetido antes do envolvimento ou participação humana na prática da pesquisa.
- **O quê:** O projeto submetido ao CEP deve fornecer detalhes sobre como a pessoa pesquisadora pretende conduzir *a parte de sua pesquisa que envolve pessoas* de forma ética e segura. É necessário informar a equipe e as instituições envolvidas.
- **Onde e Como:** A submissão do projeto deve ser feita por meio da Plataforma Brasil (PB), cuja tela principal está ilustrada na Figura 4.1. Um *Manual do Pesquisador* é um dos documentos disponíveis a partir da página inicial da Plataforma Brasil (PB), selecionado em *Manuais da PB* (ver Figura 4.2). Em posse das informações e dos documentos necessários, a pessoa pesquisadora deve submeter o projeto seguindo

as instruções da PB.

- **Exemplos:** Selecionamos artigos das áreas de Web e Multimídia que registraram submissão de projetos a CEPs, e artigos nos quais a submissão não foi necessária.

## 4.2. Ética, Moral e o Sistema CEP/Conep

Considerando os termos *ética* e *moral*, Feil and Paz [12] discutem conceitos associados e esses termos, entre os quais a concepção da *ética como sinônimo de moral* e a visão da *ética como legislação, estatuto ou código*.

### 4.2.1. Ética como sinônimo de moral.

Na discussão do conceito *ética* como sinônimo de *moral*, Feil and Paz [12] citam a explicação de La Taille [13] sobre os dois termos terem origem em duas culturas — latim (*moral*) e grego (*ética*) — que os utilizavam para denotar um conjunto de regras consideradas como obrigatórias.

Feil and Paz [12] argumentam que *moral* consiste em princípios de comportamento que influenciam a forma de viver de um grupo de indivíduos na forma de regras de comportamento transmitidas de uma geração para outra, que o grupo entende ser bom — e não na forma de leis ou estatutos. Ainda para ilustrar esse o conceito de *ética* como *moral*, Feil and Paz [12] citam Barros Filho [14] que sumariza “a *moral* é um conjunto de princípios que seguimos livremente na nossa vida, aquilo que nos obrigamos a respeitar porque livremente decidimos assim; a *moral* é o que não faríamos de jeito nenhum, mesmo que não tivesse ninguém olhando, mesmo que fossemos invisíveis. [...] Quando você está sozinho com você mesmo, dialogando consigo mesmo para encontrar o melhor caminho e a melhor conduta, você se encontra no coração da *moral*.”

### 4.2.2. Ética como legislação, estatuto ou código.

Neste conceito, *ética* ainda se relaciona totalmente com *moral*, mas se distingue ao separar a esfera pública da privada. La Taille [13] exemplificam que a *moral* abrange normas privadas, como o comportamento esperado de pais, enquanto a *ética* diz respeito às normas públicas. É no escopo desse conceito de *ética* que estão os conselhos ou comitês de *ética* de instituições e os códigos de *ética* de profissões [15]. É também no escopo desse conceito de *ética* que se enquadram os comitês de *ética* em pesquisa e as resoluções que estabelecem diretrizes e normas aplicáveis a pesquisas envolvendo seres humanos, tratados no restante deste documento.

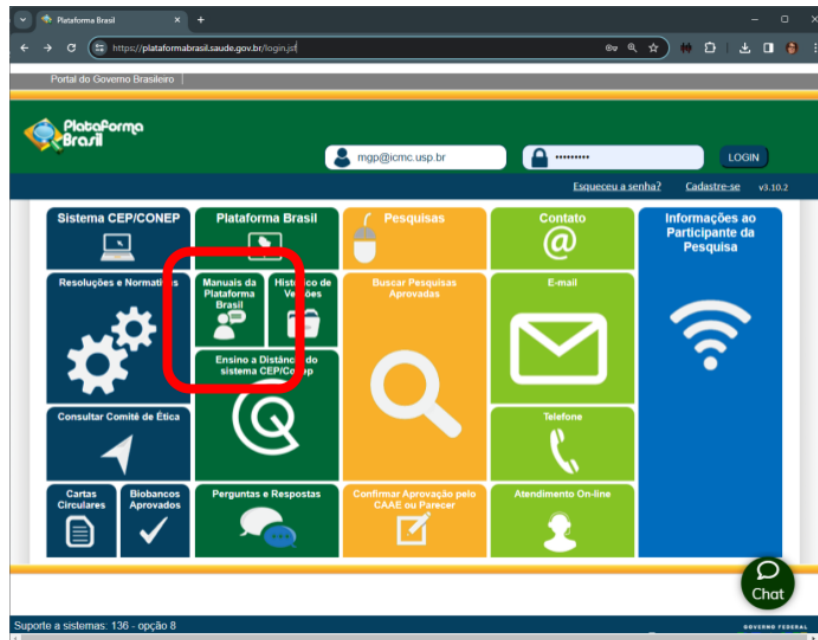
### 4.2.3. Sistema CEP/Conep.

O Sistema CEP/Conep é um mecanismo brasileiro que visa, especialmente, à proteção dos participantes de pesquisa do Brasil [11]<sup>1</sup> a partir da regulamentação efetuada pelo Conselho Nacional de Saúde (CNS). Esse sistema é composto por dois principais componentes: os Comitês de Ética em Pesquisa (CEPs) e a Comissão Nacional de Ética em Pesquisa (Conep) [11, 17].

---

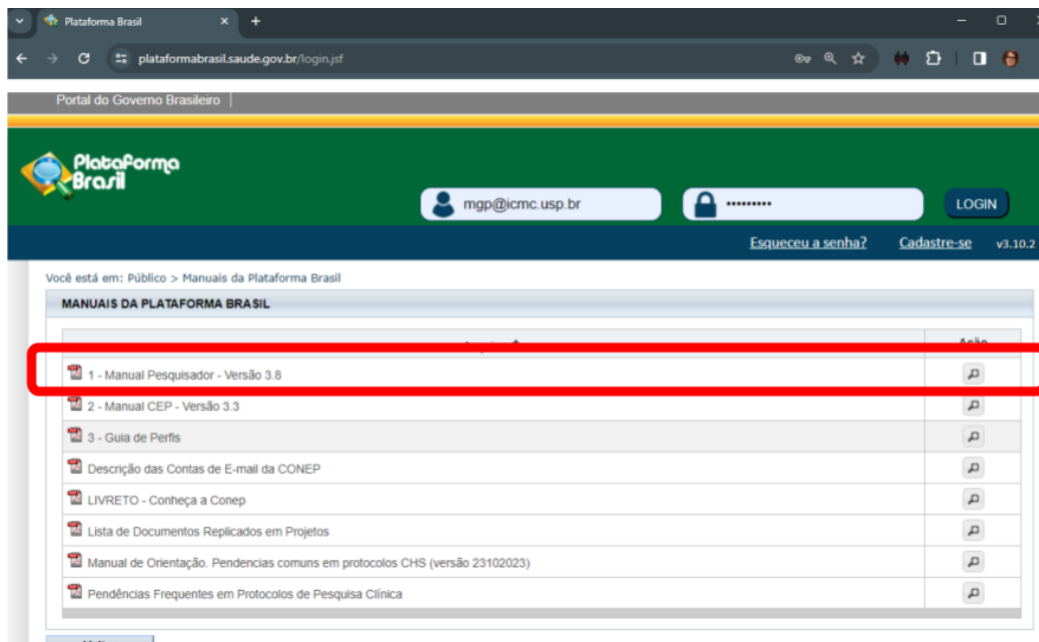
<sup>1</sup>Esta comunicação trata apenas de pesquisas com envolvimento ou participação humana. Envolver animais em pesquisas envolve normas, resoluções e procedimentos próprios, normatizados pelos Comitês de Ética de Uso de Animais (CEUA) [16].





46

Figura 4.1. Tela inicial da (PB): <https://plataformabrasil.saude.gov.br/> [acesso 28/03/2024]



47

Figura 4.2. Manuais de uso da PB.

CEPs são comitês designados em instituições de pesquisa em todo o Brasil, responsáveis pela avaliação ética de projetos de pesquisa a serem desenvolvidos nessas instituições ou outras, quando designadas pela Conep. Eles asseguram que os projetos estejam em conformidade com as normas éticas nacionais [11, 17].

A Conep é um órgão de coordenação e supervisão do sistema de revisão ética de pesquisas no Brasil, e tem o papel de analisar projetos de pesquisa que envolvem maior complexidade ética, como estudos multicêntricos, pesquisas com produtos novos ou não registrados no Brasil, estudos com populações vulneráveis, entre outros [11, 17].

O Sistema CEP/Conep funciona de maneira integrada para garantir a proteção de participantes da pesquisa, respeitando sua dignidade e direitos, e promovendo a ética na pesquisa científica. As submissões de projetos de pesquisa são feitas através da Plataforma Brasil (PB), um sistema eletrônico que unifica o processo de submissão, revisão e acompanhamento dos estudos em todo o país. Esse sistema facilita a comunicação entre as pessoas pesquisadoras, os CEPs e a Conep, e proporciona maior transparência e eficiência na gestão dos processos éticos em pesquisa [17].

### **4.3. Por quê?**

O Sistema CEP/Conep está alinhado, quanto a diretrizes e princípios, com sistemas semelhantes em outros países, baseado nos princípios éticos da autonomia das pessoas participantes da pesquisa, da beneficência, da não maleficência e da justiça. O sistema tem algumas especificidades que asseguram maior proteção das pessoas participantes de pesquisas, e que não devem ser comprometidas, como o reconhecimento da vulnerabilidade de tais pessoas, e a necessidade de proteção das mesmas [11].

#### **4.3.1. Como era antes?**

A falta de escrutínio ético pode resultar em abusos hediondos, violando os princípios fundamentais de respeito à dignidade humana e de proteção dos direitos individuais. Durante a Segunda Guerra Mundial, pessoas com formação em medicina e nazistas conduziram experimentos atroz em pessoas aprisionadas de campos de concentração, causando sofrimento e morte. Como observa Weindling [18], “Não se tratava de que a pesquisa nazista conduzida em pessoas não tinha ética, mas sim que as formulações éticas foram moldadas por prioridades nazistas” [tradução nossa]. Esses experimentos revelaram a necessidade crítica de estabelecer diretrizes éticas rigorosas para a pesquisa envolvendo pessoas. Um marco chave nesse contexto foi o julgamento dessas pessoas no Tribunal Militar de Nuremberg em 1946. A revelação das práticas levou à criação do Código de Nuremberg em 1948, que estabeleceu princípios éticos para pesquisas médicas, incluindo o consentimento voluntário de participantes. Essa história rememora a necessidade de avaliar e apreciar a ética da pesquisa, assegurando que o consentimento voluntário e a proteção de participantes sejam prioridades em investigações científicas.

Outras preocupações estão relacionadas a experimentos como os realizados pelo médico Edward Jenner, conhecido como “pai da imunologia” por seu trabalho que levou à descoberta das vacinas no século XVIII. Jenner utilizou métodos que não atenderiam aos padrões éticos e científicos modernos, nos quais as pessoas participantes são informadas sobre os riscos e benefícios da pesquisa e escolhem participar ou não, bem como deixar

de participar a qualquer momento, por exemplo. À época, a varíola, que é especialmente virulenta em crianças, era fatal para entre 20% e 50% dos infectados. Em seus experimentos, Jenner inoculou pessoas adultas e crianças, algumas com vínculo empregatício com o mesmo, com material retirado de lesões de pessoas com varíola bovina e com varíola humana, para testar a efetividade de proteção da primeira inoculação contra a doença que viria da segunda [19]. Suas descobertas fundamentaram o desenvolvimento das vacinas, embora seus experimentos levantem questões éticas sobre o consentimento informado e a exposição voluntária a riscos. Vacinas continuam sendo desenvolvidas: estudos relativos à sua eficácia e aos seus riscos devem atender aos procedimentos éticos normativos atuais.

Um exemplo complementar é o caso de cientistas laureados com o prêmio Nobel, Marie Curie, Pierre Curie e Henri Becquerel. Em suas pesquisas pioneiras sobre a radioatividade, no final do século XIX e início do século XX, os Curies e Becquerel expuseram-se voluntariamente a substâncias radioativas, sem que fosse possível ter pleno conhecimento dos riscos à saúde daquela exposição. Como resultado de sua dedicação à ciência, sofreram graves problemas de saúde, incluindo doenças por radiação [20]. Esses episódios ressaltam que até mesmo pessoas pesquisadoras experientes podem desconhecer, por ignorância, alguns ou todos os riscos associados a suas experiências.

Um caso mais recente das Ciências Humanas é famoso no contexto de ética em pesquisa: os Experimentos de Milgram realizados nas décadas de 1960 e 1970 [21]. Nesses experimentos, pessoas participantes foram instruídas a dar choques elétricos em outras pessoas sempre que elas respondessem incorretamente a perguntas em um teste de memória. Na realidade, os choques eram fictícios e as outras pessoas eram atores fingindo sentir dor. O objetivo era investigar até que ponto participantes obedeceriam a uma autoridade (neste caso, a pessoa pesquisadora), mesmo que isso envolvesse causar dor física intensa a outras pessoas. O estudo revelou que participantes estavam dispostos a continuar dando choques cada vez mais fortes, independentemente do nível de dor da vítima. A literatura apresenta inúmeras críticas aos procedimentos adotados por Milgram considerando, entre outros, que alguns participantes desenvolveram traumas psicológicos após participar do estudo [22, 23].

A avaliação de projetos de pesquisa por um CEP é uma resposta histórica e ética às atrocidades e aos erros do passado. O objetivo é garantir que os estudos sejam conduzidos de acordo com normas morais pautadas por princípios éticos, visando a proteção, segurança e o bem-estar de participantes da pesquisa [11].

#### **4.3.2. Providências da comunidade científica**

Mandal et al. [24] resumizam marcos históricos na ética da pesquisa médica que têm desempenhado um papel fundamental na garantia da proteção de participantes de pesquisa ao longo do tempo. Em 1946, o Tribunal Militar contra Médicos Alemães julgou experimentos cruéis realizados em prisioneiros de campos de concentração durante a Segunda Guerra Mundial, levando à condenação das pessoas médicas envolvidas. Em 1948, o Código de Nuremberg estabeleceu princípios éticos, enfatizando o consentimento voluntário como pedra angular da pesquisa médica. A Declaração de Helsinki, de 1964, continuou a orientar a pesquisa internacional, delineando diretrizes tanto para “pesquisa com cuidados clínicos” como “pesquisa não terapêutica.”

Mandal et al. [24] notam que, em 1979, o *Belmont Report* estabeleceu os princípios éticos fundamentais de respeito às pessoas, beneficências positiva e preventiva, e justiça como base regulatória para pesquisa envolvendo pessoas. A beneficência positiva enfatiza a promoção do benefício, enquanto a beneficência preventiva exige que se evite causar dano aos participantes e que se tomem medidas para minimizar qualquer risco potencial. Assim, o conceito de beneficência implica que a pesquisa deve almejar benefícios para participantes ao mesmo tempo que todo e qualquer risco seja evitado ou, quando não possível, minimizado. É nesse escopo que, no Brasil, a Resolução CNS 466/2012 [11] “incorpora, sob a ótica do indivíduo e das coletividades, referenciais da bioética, tais como, autonomia, não maleficência, beneficência, justiça e equidade, dentre outros, e visa a assegurar os direitos e deveres que dizem respeito aos participantes da pesquisa, à comunidade científica e ao Estado”(resoluções são discutidas na próxima seção).

Independente de área, as atuais boas práticas científicas e de pesquisa no Brasil devem seguir princípios éticos fundamentais, para assegurar que a pesquisa com pessoas seja conduzida com ética e responsabilidade, priorizando o bem-estar das pessoas envolvidas. Consequentemente, a ética da pesquisa em Computação, quando envolve pessoas, é regida por essas mesmas diretrizes que visam proteger participantes e garantir sua autonomia, através de pesquisas responsáveis. Como nas demais áreas, as questões éticas envolvem preocupações como privacidade e segurança de dados.

Logo, pesquisas em áreas como Sistemas Interativos Web e Multimídia e Interação Humano-Computador (IHC) devem priorizar o consentimento voluntário das pessoas participantes nas pesquisas, minimização de riscos e benefícios claros, dentre outros.

### 4.3.3. Resoluções do Conselho Nacional de Saúde

O Conselho Nacional de Saúde (CNS) é um órgão colegiado, deliberativo e permanente, vinculado ao Ministério da Saúde (MS).

A Resolução CNS 196/1996 de 10/outubro/1996 [10], a Resolução CNS 466/2012 de 12/ dezembro/2012 [11], e a Resolução CNS 510/2016 de 7/abril/2016 [25] estabelecem diretrizes e normas aplicáveis a pesquisas que envolvem seres humanos.

A Conep foi criada pelo CNS em 1996 em resposta à necessidade de um órgão regulador específico para lidar com questões éticas envolvendo pesquisas com pessoas, especialmente após a promulgação da Resolução CNS 196/1996.

Os CEPs, por sua vez, já existiam em diversas instituições. Após a Resolução CNS 196/1996, eles se tornaram parte de um sistema normatizado, estruturado e integrado: o Sistema CEP/Conep. A resolução formalizou e fortaleceu o papel dos CEPs, estabelecendo-os como instâncias locais para a avaliação moral de pesquisas<sup>2</sup> em suas

<sup>2</sup>Moral e ética estão aqui como construtos conceituais distintos [26]. Resoluções e normas estão cobertas pelo conjunto da moral, tratando da efetividade do escrutínio ético, do comportamento em si. A ética trata da reflexão e do escrutínio sobre os atos em um nível elevado e abstrato. Quando seguimos as resoluções e normas, estamos de acordo e em conformidade com uma moralidade objetiva. Os princípios que fundamentam estas resoluções ou normas (por exemplo, beneficência), por sua vez partem de premissas éticas, objetivados moralmente e coercitivamente nas determinações. Aplicar o TCLE de acordo com os requisitos e as regras presentes nas resoluções cabe à moral, refletir sobre o respeito e os valores no acordo entre participantes e pessoa pesquisadora cabe à ética. Ignorar as resoluções ou as normas, quando cabível e tendo conhecimento das mesmas, é tanto imoral quanto é antiético, pois se trata de uma escolha racional,

respectivas instituições.

#### 4.3.3.1. Resoluções 196/1996 de 10/outubro/1996 e 466/2012 de 12/dezembro/2012

A Resolução CNS 196/1996 [10], um ponto pivotal na ética de pesquisa no Brasil, estabeleceu diretrizes e normas regulamentadoras de pesquisas envolvendo seres humanos, incluindo consentimento informado, proteção de grupos vulneráveis, confidencialidade e indicou parâmetros éticos gerais para as pesquisas com pessoas no Brasil.

A Resolução CNS 466/2012 [11] atualizou a resolução anterior, refletindo avanços éticos e morais desde 1996. Foram incluídos esclarecimentos e novas diretrizes, como as exigências relativamente a aspectos éticos que devem ser observadas nas pesquisas, detalhamento sobre a inclusão de grupos vulneráveis, sobre o assentimento e o consentimento informados, e sobre riscos e benefícios das pesquisas, entre outros.

É importante observar que a Resolução CNS 466/2012 [11] estabeleceu diretrizes e normas regulamentadoras para a condução de pesquisas envolvendo pessoas *independente de área de pesquisa*, tendo sido elaborada em conformidade com princípios éticos internacionais, como a Declaração de Helsinque (em suas seis versões entre 1969 e 2000), o Pacto Internacional sobre os Direitos Econômicos, Sociais e Culturais, de 1966, e as Diretrizes Internacionais para Pesquisa Biomédica envolvendo Seres Humanos de 2004, da Organização Mundial de Saúde (OMS) [11].

Alguns dos principais aspectos da Resolução 466/2012 são:

- *Objetivo e âmbito de aplicação:* A resolução regulamenta a pesquisa envolvendo pessoas em todo o território nacional, independentemente da área de conhecimento ou do financiamento. Ela se aplica a todas as pessoas pesquisadoras, instituições e comitês de ética em pesquisa que realizam ou avaliam estudos com participação de pessoas;
- *Princípios Éticos Fundamentais:* A resolução estabelece princípios éticos fundamentais que devem orientar toda pesquisa envolvendo pessoas, incluindo o respeito à dignidade, autonomia, beneficência, não maleficência, justiça e equidade. Esses princípios garantem que participantes sejam tratados com respeito e protegidos de possíveis danos;
- *Comitê de Ética em Pesquisa (CEP):* A resolução exige a criação e a regulamentação de CEPs em instituições que conduzem pesquisas com pessoas. Esses comitês são responsáveis por revisar, aprovar e monitorar os projetos de pesquisa, garantindo que eles atendam aos critérios éticos estabelecidos;
- *Participante da pesquisa* é definido como “Indivíduo que, de forma esclarecida e voluntária, ou sob o esclarecimento e autorização de seu(s) responsável(is) legal(is), aceita ser pesquisado. A participação deve ser gratuita [...]” [11, p.2].
  - É crucial observar que os termos *esclarecida* e *voluntária* demandam completo esclarecimento e completa liberdade de escolha (ver, a seguir, TCLE), e que o termo *gratuita* implica que os participantes não podem ser remunerados, de qualquer forma, por sua participação.

---

consciente, livre e responsabilizável [9].

- *Termo de Consentimento Livre e Esclarecido (TCLE)*: O TCLE é um dos itens obrigatórios para protocolos de pesquisa [17]. A resolução estabelece que participantes devem receber informações sobre os objetivos, procedimentos, riscos e benefícios da pesquisa, antes de dar seu consentimento. O consentimento deve ser voluntário, sem qualquer forma de coerção [11];
- *Grupos Vulneráveis*: A resolução reconhece a necessidade de proteger grupos vulneráveis que requeiram cuidados especiais na análise ética, por exemplo, crianças, populações indígenas, e alguns casos que possam ter limitações no exercício do livre consentimento, como pessoas encarceradas, militares, populações indígenas, estudantes e pessoas com alguns tipos de deficiência, dentre outros;
- *Termo de Assentimento Livre e Esclarecido (TALE)*: O termo é um documento elaborado em linguagem acessível para os menores ou para os legalmente incapazes, por meio do qual, após os participantes da pesquisa serem devidamente esclarecidos, explicitarão sua anuência em participar da pesquisa, sem prejuízo do consentimento de seus responsáveis legais por meio do Termo de Consentimento Livre e Esclarecido (TCLE) [11, 25]
- *Riscos e Benefícios*: As pessoas pesquisadoras devem avaliar e minimizar os riscos potenciais para as pessoas participantes, garantindo que os benefícios da pesquisa superem esses riscos. Caso os riscos sejam considerados excessivos em relação aos benefícios, a pesquisa não deve ser realizada;
- *Sigilo e Privacidade*: A resolução enfatiza a importância de proteger o sigilo e a privacidade das pessoas participantes. As informações pessoais devem ser mantidas em sigilo e não podem ser divulgadas sem o consentimento prévio dos envolvidos;
- *Pesquisa Internacional*: Quando a pesquisa envolver colaboração internacional, as pessoas pesquisadoras brasileiras devem assegurar que ela seja conduzida de acordo com os princípios éticos estabelecidos pela resolução, bem como pelas leis e regulamentos do país onde a pesquisa ocorre;
- *Publicação e Divulgação dos Resultados*: Os resultados da pesquisa devem ser relatados de forma precisa e transparente. A resolução incentiva a publicação dos resultados em periódicos científicos e a divulgação para a comunidade científica e o público em geral;
- *Vigilância e Fiscalização*: Órgãos governamentais e instituições de pesquisa são responsáveis pela vigilância e fiscalização da conformidade com a resolução. Qualquer irregularidade ou violação ética deve ser investigada e tratada adequadamente.

Em relação a grupos vulneráveis, destaques são necessários:

- A resolução define *Assentimento Livre e Esclarecido* como a anuência da pessoa participante da pesquisa, criança, adolescente ou legalmente incapaz, livre de vícios (simulação, fraude ou erro), dependência, subordinação ou intimidação, e estabelece que “tais participantes devem ser esclarecidos sobre a natureza da pesquisa, seus objetivos, métodos, benefícios previstos, potenciais riscos e o incômodo que esta possa lhes acarretar, na medida de sua compreensão e respeitados em suas singularidades.” A Resolução 510/2016, discutida na Seção 4.3.3.3, acrescenta que a obtenção do assentimento não elimina a necessidade do consentimento das pessoas responsáveis;
- Em pesquisas nas quais participantes se enquadrem nas categorias de crianças, ado-

lescentes, pessoas com transtorno ou doença mental, ou em situação de substancial diminuição em sua capacidade de decisão, é necessário justificar claramente a escolha desses grupos. Nesses casos, a etapa de obtenção do consentimento livre e esclarecido deve ser realizada por meio dos representantes legais das pessoas participantes, respeitando o direito à informação das mesmas, até onde for possível, dada sua capacidade. No caso de crianças e adolescentes, o assentimento é recomendado quando são capazes de compreender o que está acontecendo na pesquisa, e deve ser formulado em uma linguagem e formato adequado a sua faixa etária, como nos exemplos de Miranda et al. [27] e Varejão et al. [28];

- É fundamental assegurar a liberdade do consentimento especialmente para participantes de pesquisa que, mesmo plenamente capazes, possam estar sujeitos a influências ou condicionamentos específicos que possam limitar sua autonomia. Isso inclui grupos como militares, empregados, presidiários, internos em centros de readaptação, em casas-abrigo, asilos, associações religiosas e situações semelhantes. Para esses participantes, é essencial garantir que tenham total liberdade para decidir se desejam ou não participar da pesquisa, sem enfrentar qualquer tipo de represália.

Observa-se que é comum que pesquisas em diversas áreas, incluindo Computação, apresentem estudantes enquanto participantes. Ocorre que estudantes estão sob influência e condicionamento de circunstâncias ambientais diretamente relacionadas às pessoas pesquisadoras em relações de docência ou de orientação, dentre outros. As influências dessas relações colocam em risco o fato de que a participação deve ser informada, voluntária, livre e esclarecida, e sem potenciais prejuízos ou danos em caso de negativa na participação. Além disso, a determinação de não-remuneração se estende a estudantes, inclusive para elementos tradicionais da cultura educacional. Portanto, docentes não devem “premiar” estudantes com bônus no grau, presença, facilidades ou demais benefícios, e estudantes não podem ser prejudicados por não concordarem em participar. Em síntese, as mesmas considerações para participantes externos ou desconhecidos são aplicadas a estudantes: estudantes não podem ser obrigados a participar, não podem ser prejudicados por não participar, e não podem ser premiados por participar. Cabe lembrar que estudantes menores de 18 (dezoito) anos são considerados menores de idade para fins desta resolução.

#### **4.3.3.2. Instância de Ciências Humanas e Sociais (ICHS) da Conep**

A Instância de Ciências Humanas e Sociais (ICHS) da Conep foi criada em 2006 com o objetivo de atender às demandas específicas relacionadas a pesquisas que usam métodos próprios das áreas de Ciências Humanas e Sociais no Brasil. A criação da ICHS se deu para reconhecer as peculiaridades das pesquisas em Ciências Humanas e Sociais, que frequentemente envolvem questões sensíveis, subjetivas e complexas relacionadas a aspectos sociais, culturais e psicológicos. Essas pesquisas muitas vezes envolvem a participação direta de pessoas e podem abordar tópicos como comportamento humano, relações interpessoais, questões éticas e culturais, entre outros. Diversas pesquisas realizadas em Computação, e que envolvem pessoas, se encontram nessa categoria. Apesar de localizadas na Computação, seus fenômenos dialogam com as Ciências Humanas e Sociais.

A ICHS desempenha um papel fundamental na revisão e avaliação ética desses

projetos de pesquisa, garantindo que os princípios éticos sejam aplicados de maneira adequada a cada contexto. Ela considera aspectos como consentimento informado, proteção da privacidade, confidencialidade e respeito às pessoas participantes, adaptando as diretrizes éticas cabíveis às especificidades das Ciências Humanas e Sociais.

#### 4.3.3.3. Resolução 510/2016 de 7 de abril de 2016

A Resolução CNS 510/2016 de 7 de abril de 2016<sup>3</sup> [25] dispõe sobre as normas aplicáveis a pesquisas em Ciências Humanas e Sociais cujos procedimentos metodológicos envolvam a utilização de dados diretamente obtidos com os participantes ou de informações identificáveis ou que possam acarretar riscos maiores do que os existentes na vida cotidiana. Essa resolução é relevante para a Computação e outras áreas que às vezes enfrentam questões éticas e metodológicas distintas das pesquisas na área médica.

A Resolução 510/2016 estabelece que suas diretrizes e normas se aplicam a todas as áreas do conhecimento, incluindo as Ciências Humanas e Sociais. Assim, a ICHS opera dentro do contexto e das normas estabelecidas pela Resolução 510, garantindo que os princípios éticos e legais sejam observados em projetos nessas áreas.

Os princípios de ética em pesquisa que norteiam Resolução CNS 510/2016 são:

- I. reconhecimento da liberdade e autonomia de todos os envolvidos no processo de pesquisa, inclusive da liberdade científica e acadêmica;
- II. defesa dos direitos humanos e recusa do arbítrio e do autoritarismo nas relações que envolvem os processos de pesquisa;
- III. respeito aos valores culturais, sociais, morais e religiosos, bem como aos hábitos e costumes, de participantes das pesquisas;
- IV. empenho na ampliação e consolidação da democracia por meio da socialização da produção de conhecimento resultante da pesquisa, inclusive em formato acessível ao grupo ou população que foi pesquisada;
- V. recusa de todas as formas de preconceito, incentivando o respeito à diversidade, à participação de indivíduos e grupos vulneráveis e discriminados e às diferenças dos processos de pesquisa;
- VI. garantia de assentimento ou consentimento de participantes das pesquisas, esclarecidos sobre seu sentido e implicações;
- VII. garantia da confidencialidade das informações, da privacidade de participantes e da proteção de sua identidade, inclusive do uso de sua imagem e voz;
- VIII. garantia da não utilização, por parte do pesquisador, das informações obtidas em pesquisa em prejuízo dos seus participantes;
- IX. compromisso de todos os envolvidos na pesquisa de não criar, manter ou ampliar as situações de risco ou vulnerabilidade para indivíduos e coletividades, nem acentuar o estigma, o preconceito ou a discriminação; e
- X. compromisso de propiciar assistência a eventuais danos materiais e imateriais, decorrentes da participação na pesquisa, conforme o caso sempre e enquanto necessário.

<sup>3</sup>[https://bvsmms.saude.gov.br/bvs/saudelegis/cns/2016/res0510\\_07\\_04\\_2016.html](https://bvsmms.saude.gov.br/bvs/saudelegis/cns/2016/res0510_07_04_2016.html)

[acesso 28/03/2024]



O Art.1o da Resolução 510/2016 lista oito tipos de pesquisa como isentos de aprovação prévia. As muitas dúvidas relativamente às isenções instituídas pela Resolução 510/2016 levaram à publicação do Ofício Circular N° 17/2022/Conep/SECNS/MS [29]. Esse ofício orienta pesquisadores e membros do Sistema CEP/Conep quanto aos projetos que utilizam metodologias características das Ciências Humanas e Sociais e que estão dispensados de submissão ao Sistema CEP/Conep. Sumarizamos esse conteúdo a seguir:

#### I. Pesquisa de Opinião Pública com Participantes Não Identificados

- *Definição*: “As pesquisas aqui enquadradas têm como único propósito descrever a valoração que o participante atribui ao objeto de consulta. Como exemplos, pode-se citar pesquisas eleitorais, de mercado e de monitoramento de um serviço, para fins de sua melhoria ou implementação, sem que haja qualquer possibilidade de identificação de participantes pelo/a pesquisador/a, desde o momento da coleta de dados”;
- *Dispensa de Avaliação*: Não necessita submissão ao Sistema CEP/Conep devido à ausência de identificação de participantes *desde o procedimento de recrutamento*:
  - É necessário cuidado ao utilizar ferramentas tais como questionários eletrônicos, como o *Google Forms*, uma vez que essas podem, por padrão, coletar o e-mail ou outras informações de respondentes. Tal coleta viola a condição de não identificação dos participantes exigida nesta isenção;
  - Pela definição dessa isenção, uma pesquisa de opinião realizada como parte de uma pesquisa científica pode consultar a opinião das pessoas participantes sobre o objeto da consulta. Assim, uma pesquisa de opinião não coleta dados demográficos que permitam identificar respondentes direta ou indiretamente. Também não se enquadram como “pesquisa de opinião” as consultas que procuram identificar experiências prévias de respondentes.

#### II. Pesquisa com Informações de Acesso Público

- *Definição*: Uso de dados disponíveis ao público em geral, incluindo informações de órgãos públicos não sigilosos;
- *Dispensa de Avaliação*: Não requer submissão ao Sistema CEP/Conep, já que as informações são publicamente acessíveis e não privadas.

#### III. Pesquisa com Informações de Domínio Público

- *Características*: Uso de informações sem restrições de direitos autorais ou propriedade intelectual;
- *Dispensa de Avaliação*: Essas informações podem ser utilizadas livremente, dispensando submissão ao Sistema CEP/Conep.

#### IV. Pesquisa Censitária

- *Definição*: A pesquisa censitária é aquela realizada pelo poder público, por exemplo, por meio do Instituto Brasileiro de Geografia e Estatística (IBGE), para quantificar populações e produzir dados quantitativos sobre diversos aspectos da vida;
- *Dispensa de Avaliação*: Geralmente, essas pesquisas são de domínio público e não necessitam submissão ao Sistema CEP/Conep.

#### V. Pesquisa com Bancos de Dados Agregados

- *Dados Agregados*: Informações sobre um conjunto de pessoas que não per-

mitem identificação individual;

- *Dispensa de Apreciação*: Não requer submissão ao Sistema CEP/Conep quando os dados são obtidos para o estudo de forma agregada, sem que seja possível a identificação de participantes.

VI. Pesquisa Exclusivamente com Textos Científicos

- *Foco*: Pesquisas voltadas para a revisão da literatura científica usando fontes acadêmico-científicas;
- *Dispensa de Apreciação*: Dispensada de submissão ao Sistema CEP/Conep, visto que utiliza apenas fontes bibliográficas científicas.

VII. Pesquisa de Aprofundamento Teórico de Práticas Profissionais

- *Contexto*: Pesquisas surgidas de práticas cotidianas profissionais, sem identificação de participantes;
- *Dispensa de Apreciação*: Não requer submissão ao Sistema CEP/Conep se não houver identificação de participantes.

VIII. Atividades de Educação, Ensino ou Treinamento

- *Natureza*: Atividades de formação de estudantes sem finalidade de pesquisa científica;
- *Dispensa de Apreciação*: Dispensada de submissão, exceto em casos em que há intenção de incorporar os resultados em um projeto de pesquisa.

#### 4.3.3.4. Carta Circular nº 1/2021-CONEP/SECNS/MS

A Carta Circular nº 1/2021-CONEP/SECNS/MS apresenta orientações detalhadas para a condução de pesquisas que envolvem *ambientes virtuais* em qualquer etapa do processo, abrangendo desde o contato inicial com pessoas participantes até a coleta e gestão de dados ao longo de toda a pesquisa.

As orientações são fundamentais para a área de Web e Multimídia, pois estabelecem padrões éticos e seguros para a interação em ambientes virtuais. Elas visam garantir a proteção de dados pessoais e sensíveis, reforçam a necessidade de consentimento informado em etapas não presenciais, e ajudam a manter a integridade e a confidencialidade das informações coletadas. Sumarizamos alguns destaques da carta circular.

Definições:

- *Meio ou ambiente virtual*: Refere-se ao uso da internet (e-mails, sites eletrônicos, formulários online disponibilizados por programas, etc.), telefone (chamadas de áudio ou vídeo, uso de aplicativos de chamadas, etc.), e outros programas e aplicativos que operam nesses meios;
- *Forma não presencial*: Contato realizado por meio ou ambiente virtual, incluindo telefônico, que não envolve a presença física do pesquisador e do participante de pesquisa.

Destaques no escopo de submissão ao Sistema CEP/CONEP:

- Descrição completa das etapas não presenciais, incluindo métodos e tecnologias utilizadas;
- Envio de modelos de formulários e termos de consentimento que serão utilizados;

- Detalhamento do processo de obtenção do consentimento, incluindo o registro do mesmo e discussão dos riscos específicos do ambiente virtual.

Destaques no tange procedimentos que envolvem contato por meio virtual ou telefônico:

- Proteção da identidade dos participantes, utilizando métodos como o envio de convites por BCC em e-mails;
- Apresentação do Termo de Consentimento Livre e Esclarecido antes do início da coleta de dados;
- Garantia de que os participantes podem se recusar a responder perguntas e se retirar da pesquisa a qualquer momento sem justificativa.

Destaques no escopo de segurança na transferência e armazenamento dos dados:

- Assegurar a proteção dos dados coletados contra acessos não autorizados;
- Dados devem ser transferidos para dispositivos seguros e removidos de plataformas online após a conclusão da coleta.

Destaques no escopo de conteúdo dos documentos relacionados ao consentimento:

- Documentos devem fornecer todas as informações necessárias de maneira clara e completa;
- Informações claras sobre como os participantes podem retirar seu consentimento a qualquer momento.

#### **4.3.3.5. Resolução 674/2022 de 6 de maio de 2022**

A Resolução Nº 674 de 6 de maio de 2022 [30], que complementa o quadro regulatório, define tipificações de pesquisas e estabelece diretrizes para a tramitação dos protocolos de pesquisa no sistema CEP/Conep. A resolução estabelece que:

- *Tipificação da pesquisa* é o “processo pelo qual se define o tipo da pesquisa, baseando-se no *delineamento do estudo* e nos *procedimentos da pesquisa*.”;
- *Fatores de modulação* são “características do processo de consentimento, da confidencialidade e/ou dos métodos da pesquisa, que possam modificar o tipo de tramitação do protocolo no Sistema CEP/Conep.”

Cada projeto de pesquisa que envolve pessoas deve ser submetido a um processo de avaliação ética, que pode passar por diferentes tipos de *tramitação* – Tramitação Expressa, Simplificada, Colegiada, e Colegiada Especial – dependendo dos chamados *fatores de modulação*, que estão associados do nível de risco, complexidade e natureza específica da pesquisa. Os quatro tipos de tramitação têm diferentes *prazos de análise*.

A seguir discutimos os conceitos de Delineamento do estudo, Procedimento de pesquisa, Tipificação, Fatores de modulação, Tramitação e Prazos de Análise.

*Delineamento do estudo.*

O *delineamento do estudo* é dividido em dois tipos, de acordo com seus objetivos (Art.3o):

- *Tipo I* - Estudos que visam descrever ou compreender fenômenos que aconteceram ou acontecem no cotidiano da pessoa participante de pesquisa;

- *Tipo II* - Estudos que visam verificar o efeito de produto ou técnica em investigação, deliberadamente aplicado na pessoa participante em virtude da pesquisa, de forma prospectiva, com grupo controle ou não.

#### *Procedimento de pesquisa.*

A tipificação segundo o *procedimento da pesquisa* também as divide em dois tipos (Art.4o):

- *Tipo I* - Estudos que envolvem intervenção no corpo humano;
- *Tipo II* - Estudos que não envolvem intervenção no corpo humano

Além disso, o procedimento da pesquisa que envolve intervenção no corpo humano pode ter, ou não, *caráter invasivo na dimensão física* (Art.5o).

#### *Tipificação.*

As pesquisas são classificadas, segundo o delineamento e o procedimento do estudo, em três tipos: *Tipo A*, *Tipo B* e *Tipo C* (Art 6o), apresentados no Quadro 4.3.1 criada a partir do diagrama de detalhamento incluído como Anexo I da resolução:

- As tipificações *Tipo A* e *Tipo B* envolvem estudos que visam descrever ou compreender fenômenos que aconteceram ou acontecem no cotidiano da pessoa participante da pesquisa, e se diferenciam por envolver ou não intervenção *direta* no corpo da pessoa participante:
  - *Tipo A*: Estudos que não envolvem intervenção direta no corpo humano. Exemplos: estudos que utilizam apenas dados pré-existentes, observações não participativas, entrevistas, questionários e outras formas de coleta de dados não invasiva;
  - *Tipo B*: Estudos que envolvem algum grau de intervenção no corpo humano, mas de forma minimamente invasiva. Exemplos: coleta de amostras biológicas não invasivas (como saliva ou cabelo), uso de dispositivos de monitoramento externo (como sensores de acelerômetro em pulseiras), e intervenções leves em estudos comportamentais.
- *Tipo C*: Estudos que visam verificar o efeito de produto ou técnica de investigação, da área da Saúde ou não, deliberadamente aplicado na pessoa participante em virtude da pesquisa, com grupo controle ou não. Exemplos na área da Saúde são estudos que incluem intervenções mais invasivas, como procedimentos clínicos ou cirúrgicos, ou que envolvem riscos mais significativos para participantes. Estudos clínicos com novos medicamentos ou terapias, procedimentos cirúrgicos experimentais e coleta de amostras biológicas invasivas se enquadram nessa categoria.

As três tipificações são subdivididas. O *Tipo A*, que não envolve intervenção direta no corpo humano, se divide na subcategorias *A1*, *A2* e *A3*, dependendo da forma como os dados do estudo são coletados. O *Tipo B*, que envolve intervenção direta no corpo humano, se subdivide em *B1* e *B2*, dependendo se a intervenção no corpo humano ocorre ou não na dimensão física. Finalmente, o *Tipo C*, engloba *C1* e *C2*, diferenciados pelo fato do estudo ser realizado com produtos ou dispositivos aplicáveis ou não na área da Saúde.

A tipificação das pesquisas *Tipo A* é aplicada a muitos estudos que envolvem pessoas, especialmente no que diz respeito à coleta e análise de dados. Por exemplo:

- *A1*: Estudos realizados exclusivamente a partir de acervos de dados pré-existentes

em meio físico ou eletrônico que não sejam de acesso público. Essa tipificação é particularmente relevante para a Computação, quando envolve a análise de grandes conjuntos de dados já existentes, como bancos de dados históricos;

- *A2*: Estudos realizados com observação ou observação participante. Essa categoria pode incluir estudos da área de Computação, que observam comportamentos de usuários em ambientes digitais ou interações com tecnologias, sem intervenção direta do pesquisador;
- *A3*: Estudos que fazem uso de entrevista, aplicação de questionários, grupo focal ou outras formas de coleta dirigida de dados (presencial ou não-presencial, virtual, eletrônica, telefônica e similares). Essa tipificação é aplicável a pesquisas em Computação que envolvem coleta de dados por meio de interações diretas com participantes, como entrevistas sobre uso de tecnologia, experiência do usuário ou estudos de usabilidade;
- *A4*: Estudos realizados com material biológico armazenado em biobanco ou biorrepositório, ou exclusivamente com culturas de células humanas já estabelecidas. Embora essa categoria seja mais aplicável em contextos biomédicos, pode ter relevância em pesquisas de Computação que utilizam dados biológicos para desenvolver ou testar algoritmos, como aqueles usados em bioinformática.

As pesquisas *Tipo B* envolvem intervenções minimamente invasivas em pessoas. Essas pesquisas são classificadas em duas subcategorias:

- *B1*: Pesquisas que envolvem a coleta de amostras biológicas de forma não-invasiva ou minimamente invasiva, como a coleta de saliva, cabelo, unhas, ou o uso de sensores externos para monitoramento de sinais vitais. No contexto da Computação, isso pode incluir estudos que coletam e analisam esses dados para desenvolver novos algoritmos ou sistemas de monitoramento de saúde;
- *B2*: Estudos que envolvem intervenções leves e controladas, como o uso de dispositivos de realidade virtual ou aumentada em testes de usabilidade. Essas pesquisas podem ser utilizadas para avaliar a interação do usuário com novas tecnologias e interfaces, sendo caracterizadas por um baixo risco e um impacto minimamente invasivo em participantes.

As pesquisas *Tipo C* são classificadas em C1 e C2 dependendo do objeto de investigação ser ou não um medicamento, fármaco, produto biológico ou dispositivo aplicável na área da Saúde. Trata-se de estudos de natureza *mais invasiva* e que apresentam *riscos potencialmente maiores*, exigindo uma revisão ética rigorosa. Na área de Computação, incluiriam estudos que utilizam tecnologias de Computação para testar ou avaliar a eficácia de novos dispositivos médicos ou softwares de diagnóstico (C2), ou softwares ou dispositivos que implementam ou avaliam intervenções, fora da área da Saúde, com riscos maiores (C1).

O Quadro 4.3.2 lista exemplos de Pesquisas Tipo A e Tipo B nas áreas de Computação, Web e Multimídia.

#### *Fatores de modulação (extraídos do Art.10o da Resolução 674/2022)*

Os fatores de modulação modificam a forma de tramitação do protocolo de pesquisa conforme detalhado no Anexo II da Resolução 674/2022. Esses fatores envolvem *características do processo de consentimento e confidencialidade* bem como *características dos métodos da pesquisa*, muitos dos quais se aplicam diretamente à área de Saúde.

DELINEAMENTO DO ESTUDO*		
<b>Estudos que visam descrever ou compreender fenômenos que ocorrem no cotidiano da pessoa participante da pesquisa.</b>		<b>Estudos que visam verificar o efeito de produto ou técnica em investigação, deliberadamente aplicado na pessoa participante em virtude da pesquisa, de forma prospectiva, com grupo-controle ou não.</b>
<i>Estudos sem intervenção no corpo humano (Tipo A)</i>	<i>Estudos com intervenção no corpo humano (Tipo B)</i>	<i>Efeito de produto ou técnica em investigação (Tipo C)</i>
A1: Acervo de dados pré-existentes, em meio físico ou eletrônico, que não sejam de acesso público.	B1: Sem procedimento invasivo na dimensão física.	C1: Objeto de investigação não é um medicamento, fármaco, produto biológico ou dispositivo na área da Saúde.
A2: Observação ou observação participante.	B2: Com procedimento invasivo na dimensão física.	C2: Objeto de investigação é um medicamento, fármaco, produto biológico ou dispositivo na área da Saúde.
A3: Entrevista, aplicação de questionários, grupos focais ou outras formas de coleta dirigida de dados.		
A4: Uso de material biológico coletado e armazenado em biobanco ou biorepositório.		
<i>*se o procedimento da pesquisa implicar mais de um subtipo, prevalece a tramitação de maior complexidade</i>		

Quadro 4.3.1: Tipificação das Pesquisas segundo o delineamento e o procedimento do estudo. Fonte: Resolução 674/2022 [30, p. 10].

EXEMPLOS DE ESTUDOS EM ÁREAS DE WEB E MULTIMÍDIA E TIPIFICAÇÃO	
<b>Tipo A</b>	
Estudos com Dados Pré-existentes	Análise de padrões de comportamento de usuários em redes sociais utilizando dados históricos anonimizados.
Pesquisas Observacionais	Estudo sobre como diferentes designs de interface de usuário afetam a navegação dos usuários em um site de notícias.
Coleta Dirigida de Dados Não-Invasiva	Pesquisa de opinião online para avaliar a usabilidade de um novo aplicativo móvel.
Estudos com Material Biológico Armazenado	Uso de dados genéticos armazenados para desenvolver algoritmos de bioinformática.
<b>Tipo B</b>	
Coleta de Amostras Biológicas	Estudo que coleta saliva para análise de hormônios relacionados ao estresse em um jogo de realidade virtual.
Uso de Dispositivos de Monitoramento Externo	Uso de pulseiras com sensores de acelerômetro para monitorar atividade física em uma aplicação de <i>fitness</i> .
Estudos Intervencionais Não-Clínicos	Teste dos efeitos de diferentes tipos de música em plataformas de <i>streaming</i> sobre a concentração de estudantes.
Pesquisas Observacionais com Intervenções Leves	Alteração da configuração de iluminação em um ambiente de trabalho para avaliar efeitos sobre a produtividade.

Quadro 4.3.2: Exemplos de Estudos e correspondente tipificação.

*Características do processo de consentimento e confidencialidade:*

- A pesquisa prevê a solicitação de dispensa de consentimento da pessoa participante para o uso do seu material biológico previamente armazenado em biobanco ou biorrepositório;
- A pesquisa prevê a solicitação de dispensa do consentimento para acesso a um acervo que tenha dados pessoais identificadores da pessoa participante;
- A confidencialidade dos dados da pessoa participante ou de terceiros não está assegurada pelas circunstâncias da pesquisa;
- Há inviabilidade para obtenção do Registro/TCLE ou TALE;
- Pesquisa encoberta ou em que o consentimento será obtido *a posteriori*. Por exemplo, relatos de experiência;
- A pesquisa envolve situações passíveis de limitação da autonomia da pessoa participante, geradas por relações hierárquicas, de autoridade ou de dependência;
- Pesquisa realizada em comunidades cuja cultura reconheça a autoridade do líder ou do coletivo sobre o indivíduo.

*Características dos métodos da pesquisa:*

- Pesquisas que preveem a anonimização irreversível dos dados;
- Pesquisas que envolvem manipulação genética de gametas ou uso de células-tronco embrionárias;
- Pesquisas que incluem o uso de pré-embriões, embriões ou fetos;
- Pesquisas que envolvem interação com organismos geneticamente modificados ou de alto risco coletivo;
- Pesquisas que preveem o encaminhamento de material biológico humano para o exterior;
- Pesquisas que avaliam fármacos, medicamentos, produtos biológicos, equipamentos ou dispositivos terapêuticos já registrados na Anvisa; estudos de bioequivalência; ou estudos relacionados a alimentos, nutrição, higiene pessoal, cosméticos e perfumes;
- Pesquisas que visam avaliar exclusivamente o processo de ensino-aprendizagem;
- Pesquisas-ação ou que envolvem protagonismo da pessoa participante e convite às pessoas participantes para análise dos dados.

*Tramitação*

A Resolução 674/2022 define o tipo de tramitação dos protocolos de pesquisa no Sistema CEP/Conep de acordo com a tipificação da pesquisa e fatores de modulação (Art.13o):

- *Tramitação Expressa*: Aplicável a pesquisas do tipo A1 e A2, caracterizadas pela ausência de intervenção no corpo humano e menor complexidade;
- *Tramitação Simplificada*: Destina-se a pesquisas dos tipos A3, A4 e B1, que podem envolver procedimentos não invasivos ou coleta de dados que não demandam revisão ética intensiva;
- *Tramitação Colegiada*: Utilizada para pesquisas dos tipos B2 e C1, que incluem intervenções no corpo humano ou estudo dos efeitos de técnicas ou produtos, mas que não são consideradas de alto risco;
- *Tramitação Colegiada Especial*: Reservada para pesquisas do tipo C2, que investigam medicamentos, fármacos, produtos biológicos ou dispositivos na área da Saúde

e apresentam maior risco ou complexidade.

Essas modalidades de tramitação levam em consideração os riscos envolvidos, a complexidade do estudo e as características específicas da pesquisa, garantindo uma revisão ética proporcional à natureza do estudo. Os fatores de modulação podem alterar a tramitação padrão, como detalhado no Anexo II da Resolução.

É vital salientar que pesquisas focadas exclusivamente na avaliação do processo de ensino-aprendizagem, bem como pesquisas-ação que envolvem o protagonismo de participantes e ou pesquisa que convidam participantes a realizar análise de dados, geralmente se qualificam para Tramitação Expressa. Isso se aplica especificamente às pesquisas classificadas como Tipo A3, caracterizadas pelo uso de entrevistas, aplicação de questionários, grupos focais ou outras formas de coleta dirigida de dados. Esse indicativo de tramitação mais rápida reconhece a natureza menos invasiva e o potencial risco reduzido associado a esses tipos de estudos.

As pesquisas em Web e Multimídia frequentemente estão sujeitas à tramitação expressa ou simplificada devido à sua natureza geralmente não-invasiva e ao uso comum de dados públicos ou anonimizados:

- *Baixo Risco e Impacto Direto em Participantes:* Muitas pesquisas são de baixo risco, como estudos de usabilidade ou análises de interação do usuário com interfaces Web. *Exemplo:* Avaliação da usabilidade de uma nova operação de navegação implementada em um site de e-commerce;
- *Natureza Não-Invasiva das Pesquisas:* Pesquisas nesses campos não envolvem procedimentos invasivos ou exposição a condições prejudiciais. *Exemplo:* Estudo sobre padrões de navegação em websites;
- *Uso de Dados Públicos ou Anonimizados:* Utilização de grandes conjuntos de dados já existentes, que são públicos ou anonimizados. *Exemplo:* Análises de padrões em redes sociais usando dados publicamente disponíveis ou anonimizados;
- *Pesquisas de Campo e Estudos de Caso:* Realização de estudos que envolvem interação direta com participantes, mas com riscos mínimos. *Exemplo:* Estudo observacional sobre o uso de dispositivos móveis em ambientes educacionais.

#### *Prazos de Análise*

Os prazos estabelecidos para a tramitação dos processos nos CEPs, conforme detalhados no Quadro 4.3.3, são planejados e adequados às necessidades da comunidade científica. Por exemplo, o tempo de análise da documentação, que é de até 7 dias, seguido por um período de até 21 ou 30 dias para tramitação expressa ou colegiada, respectivamente, mostra uma estrutura de tempo apropriada. É razoável esperar que esses prazos sejam flexibilizados durante os períodos de férias, acomodando as variações no funcionamento institucional nesses momentos. Essa configuração de prazos é pertinente para a maioria das pesquisas com pessoas na área de Computação, incluindo subáreas Web e Multimídia.

No entanto, há relatos de pesquisadores que experimentaram atrasos significativamente maiores do que os prazos estipulados para receber respostas dos CEPs. Esses atrasos podem ser frustrantes e impactar negativamente o cronograma de pesquisa, levando a possíveis atrasos na conclusão dos projetos e na publicação dos resultados. É crucial reconhecer que tais atrasos podem refletir problemas específicos no funcionamento dos CEPs



ou nas instituições às quais estão vinculados. Quando for esse o caso, é determinante que as pessoas pesquisadoras não apenas relatem sua insatisfação, mas também comuniquem formalmente esses atrasos à Conep. Exercer pressão sobre os CEPs e suas instituições pode ajudar a melhorar a eficiência do processo e garantir que os prazos sejam mais rigorosamente observados no futuro. Em março/2024, a Conep informou que tem realizado medidas de gestão do Sistema CEP/Conep que têm efetivamente reduzido atrasos, e que “os CEPs têm acompanhado a Conep no movimento de redução dos prazos, aumentando o número de reuniões e melhorando processos de trabalho.”<sup>4</sup>

É fundamental ressaltar, entretanto, que a existência de atrasos no Sistema *não deve ser utilizada como justificativa* para a não realização do processo ético de revisão ou para prosseguir com a pesquisa sem aguardar a aprovação necessária. A revisão ética é um componente crítico da pesquisa envolvendo pessoas e deve ser vista como parte integrante e indispensável da pesquisa responsável em qualquer campo.

Artigo	Descrição
Art.21	O prazo para checagem documental é de até <b>7</b> (sete) dias.
Art.22	O prazo para emissão do parecer, após a checagem documental, é de até <b>15</b> (quinze) dias para a <i>tramitação expressa</i> ; até <b>21</b> (vinte e um) dias para a <i>tramitação simplificada</i> ; até <b>30</b> (trinta) dias para a <i>tramitação colegiada</i> ; e até <b>45</b> (quarenta e cinco) dias para a <i>tramitação colegiada especial</i> . Caso haja alteração na forma de tramitação, por avaliação do CEP, a contagem do prazo será iniciada com a nova modalidade de tramitação.
Art.23	A pessoa pesquisadora tem prazo de até <b>30</b> (trinta) dias, prorrogáveis mediante justificativa, para responder a um parecer de <i>pendência</i> do CEP, na PB.
Art.24	A <i>primeira instância recursal</i> é o CEP, no qual se houver a não aprovação do protocolo de pesquisa, a Conep é a próxima e última instância de recurso no Sistema CEP/Conep. O prazo de <i>solicitação do recurso</i> é de até <b>30</b> (trinta) dias para cada instância.
Art.25	A submissão, pelo pesquisador, de resposta a um parecer de <i>pendência</i> ou de recurso a um parecer de não aprovação <i>reinicia a contagem dos prazos</i> de tramitação.

Quadro 4.3.3: Tempo de tramitação no Sistema CEP/Conep.

#### 4.3.4. Publicações científicas e instituições de pesquisa

Diversas editoras que publicam trabalhos científicos exigem a apresentação de comprovação de que houve preocupação e aprovação prévia de pesquisa envolvendo pessoas.

A *Association for Computing Machinery Association for Computing Machinery (ACM)*, ao publicar sua *ACM Publications Policy on Research Involving Human Participants and Subjects* (Política de Publicações para Pesquisas Envolvendo Participantes e Sujeitos Humanos da ACM), destaca a necessidade por política da ACM para pesquisas com participantes humanos:

*“À medida que o número de artigos publicados pela ACM que incluem pesquisas com participantes humanos aumenta, é uma imperativa moral que a ACM estabeleça uma política clara que defina padrões para a prática em relação às publicações da ACM. Em termos simples, é a coisa certa a se fazer e é importante para todos os envolvidos no processo de publicação, incluindo autores, revisores, editores, comitês de programa, leitores e terceiros que tornam as publicações da ACM acessíveis ao*

<sup>4</sup>[https://conselho.saude.gov.br/imagens/comissoes/conep/documentos/SEI\\_MS\\_-\\_0039318502\\_-\\_Comunicado\\_PL.pdf](https://conselho.saude.gov.br/imagens/comissoes/conep/documentos/SEI_MS_-_0039318502_-_Comunicado_PL.pdf) [acesso 28/03/2024]

*mundo, saber que qualquer pesquisa publicada em uma plataforma da ACM pode ser confiável e baseada em padrões éticos de pesquisa.”<sup>5</sup>*

Nesse contexto, a ACM determina que os autores devem garantir que suas práticas de pesquisa estejam em conformidade com o Código de Ética e Conduta Profissional da ACM [31], bem como com as normas internacionais e nacionais aplicáveis a tais pesquisas (listando, entre outros, a Declaração de Helsinque e o Belmont Report) incluindo, mas não se limitando a: minimização de possíveis danos, garantindo que quaisquer riscos sejam justificados pelos benefícios potenciais; adesão às regulamentações institucionais, locais, nacionais e internacionais relevantes; adesão ao princípio de consentimento informado; adesão ao princípio de justiça; e conformidade com todas as outras políticas da ACM.

Já o *Institute of Electrical and Electronics Engineers* Institute of Electrical and Electronics Engineers (IEEE) destaca a necessidade de aprovação por CEP:

*“Os autores de artigos que relatam pesquisas envolvendo pessoas ou animais, incluindo, mas não se limitando à pesquisa médica, devem incluir uma declaração no artigo de que a pesquisa foi realizada sob a supervisão de um comitê de revisão institucional ou órgão local/regional equivalente, incluindo o nome oficial do CEP, ou fornecer uma explicação sobre por que tal revisão não foi realizada. Para pesquisas envolvendo pessoas, os autores também devem relatar que obtiveram o consentimento dos sujeitos da pesquisa ou explicar porque o consentimento não foi obtido.”<sup>6</sup>*

Outras editoras que publicam periódicos da área de Computação apresentam exigências similares, como é o caso da Springer,<sup>7</sup> Taylor & Francis,<sup>8</sup> e SAGE.<sup>9</sup>

A Sociedade Brasileira de Computação (SBC) atualizou em 2024 seu *Código de Conduta para Autores em Publicações da SBC* [15] e incluiu uma recomendação para que autores observem as resoluções e as normas discutidas neste documento:

*“Pesquisas envolvendo a participação de seres humanos: a realização de pesquisas cujo método envolva (i) coleta de dados pré-existent de pessoas em meio físico ou eletrônico, que não sejam de acesso público; (ii) observação ou observação participante; (iii) entrevista, aplicação de questionários, grupo focal ou outras formas de coleta dirigida de dados, sejam presenciais ou não-presenciais (virtual/eletrônica/telefônica), devem observar as normas do Conselho Nacional de Saúde, atualmente expressas nas Resoluções 674/2022, 510/2016 e 466/2012 e nas que vierem a ser publicadas após a data de aprovação deste documento.”*

A prática das universidades de exigir dos orientadores de projetos de iniciação científica a comprovação de aprovação por um Comitê de Ética em Pesquisa (CEP) é uma medida comum, particularmente no contexto da concessão de bolsas de pesquisa. Similarmente, percebe-se uma expectativa implícita de que essa mesma exigência se aplique no âmbito das bolsas de pós-graduação, tanto pelas agências de fomento quanto pelas instituições de

<sup>5</sup><https://www.acm.org/publications/policies/research-involving-human-participants-and-subjects> [tradução nossa][acesso 28-03-2024]

<sup>6</sup><https://magazines.ieeeauthorcenter.ieee.org/get-started-with-ieee-magazines/publishing-ethics/guidelines-policies/submission-and-peer-review-policies/> [tradução nossa][acesso 28-03-2024]

<sup>7</sup><https://www.springer.com/gp/editorial-policies/research-involving-human-and-or-animal-participants> [acesso 28-03-2024]

<sup>8</sup><https://files.taylorandfrancis.com/madt-ethics-statement-guidance.pdf> [acesso 28-03-2024]

<sup>9</sup><https://journals.sagepub.com/author-instructions/rea> [acesso 28-03-2024]

ensino superior. Essa política visa assegurar que todos os projetos de pesquisa, independentemente do nível acadêmico, adiram rigorosamente aos padrões éticos estabelecidos, reforçando a importância da conduta ética e responsável em todas as etapas da pesquisa.

A ausência ou fraude na apresentação dessa comprovação podem acarretar implicações negativas para os autores e para a comunidade científica em geral, e.g:

- I. *Retração do artigo*: Se for descoberto que um artigo científico foi publicado sem a devida aprovação ética, a editora pode optar por retratar o artigo, retirando-o de sua publicação. Representa potencial risco à reputação dos autores e compromete a credibilidade de sua pesquisa;
- II. *Perda de credibilidade dos autores*: A falta de ética na pesquisa pode resultar em uma potencial perda de credibilidade dos autores, afetando negativamente suas carreiras acadêmicas e profissionais. Outras pessoas pesquisadoras podem ficar relutantes em colaborar com pares ou grupos envolvidos em práticas antiéticas;
- III. *Sanções institucionais*: As instituições de ensino e pesquisa também podem aplicar sanções às pessoas pesquisadoras envolvidos em práticas antiéticas, incluindo suspensão ou demissão de seus cargos;
- IV. *Implicações legais*: Em alguns casos graves, a ausência ou a fraude na aprovação ética pode resultar em implicações legais para os autores, incluindo processos judiciais e multas.

#### 4.4. Quem? Quando?

**Quem**: A submissão pode ser realizada por pessoas pesquisadoras, pós-graduandas ou orientadoras. No caso de estudantes de graduação, a submissão deve feita pelas pessoas orientadoras. A colaboração entre pessoas orientadoras e estudantes é essencial para a condução ética da pesquisa.

O projeto deve ter a concordância do representante legal da instituição sede da pesquisa (que assina a folha de rosto submetida junto com o projeto na PB) e, quando pertinente, dos responsáveis pelos locais onde o estudo com pessoas vai ser realizado. Por exemplo, no caso de coleta de dados realizada em uma escola, a direção da escola deve concordar com a coleta com o projeto proposto por meio de cartas de concordância.

**Quando**: O processo de submissão deve ocorrer após o planejamento minucioso do projeto de pesquisa, mas antes de qualquer envolvimento efetivo de pessoas no estudo. Isso garante que a pesquisa seja cuidadosamente avaliada quanto a ética e segurança.

Uma vez que um projeto de pesquisa seja idealizado, estruturado e explicitado, caso haja previsão de envolvimento ou participação humana na prática da pesquisa fora das isenções previstas, esta etapa só pode ocorrer *após* aprovação por um CEP. Podem ocorrer etapas anteriores, como revisão da literatura, antes do envolvimento ou participação humana. Entretanto, caso a interação entre pessoas pesquisadoras e pessoas participantes, ou quaisquer coletas de dados, ocorram antes da submissão, o projeto de pesquisa é automaticamente rejeitado pelo CEP. Caso ocorram durante a submissão, e enquanto ocorre a apreciação pelo CEP, configura infração das determinações normativas; caso o CEP identifique essa situação, o projeto de pesquisa será rejeitado.

Uma informação obrigatória é o *Cronograma de execução*, no qual a pessoa pes-

quisadora deve informar, no mínimo, uma etapa informando os campos solicitados (Identificação da Etapa, Início e Término). Uma sugestão é incluir no Cronograma a etapa de avaliação do projeto pelo CEP, que necessariamente antecederá as etapas de realização de experimentos envolvendo pessoas e prever o início das atividades envolvendo pessoas considerando os prazos dos CEPs.

#### 4.5. O quê?

O CEP avalia projetos de pesquisa que envolvem pessoas, exigindo documentação que comprove o apoio institucional às pesquisas. A submissão de um projeto ao CEP envolve a apresentação detalhada de como a pessoa pesquisadora planeja conduzir a pesquisa, com ênfase especial na ética e segurança no tratamento de participantes humanos. O foco da análise do CEP é especificamente a interação com pessoas, assegurando a adesão aos princípios éticos e às práticas moralmente adequadas pela resolução. Portanto, o projeto submetido ao CEP é diferente de propostas de pesquisa acadêmica com objetivos e escopos distintos ou maiores, como as de mestrado ou doutorado.

A submissão de projetos é realizada por meio do sistema web PB, conforme detalhado na Seção 4.7. Para contemplar as informações solicitadas na plataforma, *o projeto de pesquisa* deve incluir os campos listados no Quadro 4.5. Um projeto exemplo está abertamente disponível online e contém os tópicos esperados para submissão do texto do projeto (denominado *Brochura Investigador* na plataforma) na PB.<sup>10</sup>

Devem ser informados, no sistema, a área e o desenho do estudo, se há envolvimento de parceiros do Brasil ou do exterior, o orçamento e as etapas do estudo, em quais das etapas haverá a participação de pessoas e os grupos nos quais participantes serão divididos, o país de origem das pessoas participantes, e se há uso de fontes secundárias de dados. Ademais, são informados a instituição responsável, o tipo de financiamento e palavras-chave. Igualmente, são informados resumo, introdução, hipótese, objetivos, metodologia, critérios de inclusão e exclusão, riscos e benefícios, metodologia de análise de dados, desfechos e bibliografia. Não menos importante, a pessoa pesquisadora deve submeter documento de concordância das pessoas responsáveis pela instituição de pesquisa (Folha de Rosto) e projeto de pesquisa (Brochura Investigador).

No projeto, é essencial incluir, como apêndice, todos os documentos pertinentes para a interação com participantes. Isso inclui o TCLE para o qual apresentamos um exemplo no **Apêndice A**, ou a justificativa para sua dispensa, roteiros de entrevistas planejadas e quaisquer outros formulários que participantes venham a preencher. O TCLE, em particular, é também informado separadamente no sistema.

##### 4.5.1. Resumo

Para facilitar a análise pelo CEP, é sugerido que o resumo seja organizado em partes. *Introdução*: Descrever o contexto da pesquisa no que tange a participação de pessoas e o problema a ser resolvido. *Objetivo*: Descrever objetivo geral do trabalho e os objetivos específicos associados. *Metodologia*: Descrever de forma resumida a metodologia do

---

<sup>10</sup><https://www.overleaf.com/project/6536d2ae114c6690e452dc83>.

*Nota*: Destaca-se que este projeto é um *template*, elaborado em outubro de 2023, seguindo os tópicos solicitados na ocasião para preenchimento na PB.

Título da Pesquisa que Envolve Pessoas
1. Resumo
2. Introdução
3. Desenho
4. Hipótese (4000 caracteres)
5. Objetivos (primário e secundários) (4000 caracteres cada)
6. Metodologia Proposta (4000 caracteres)
7. Critérios de Inclusão e Exclusão (4000 caracteres cada)
8. Riscos (4000 caracteres)
9. Benefícios (4000 caracteres)
10. Metodologia de Análise de Dados (4000 caracteres)
11. Desfechos (primário e secundários) (4000 caracteres cada)
12. Cronograma de Execução (etapas, início, fim)
13. Orçamento Financeiro (item, valor)
14. Bibliografia
15. Anexos e/ou Apêndices

Quadro 4.5.1: Estrutura de projeto com campos exigidos na PB (limites aplicáveis).

trabalho, quais são as etapas previstas com destaque para aquelas que terão estudo com pessoas. *Palavras-chave*: Definir entre 2 e 6 palavras.

#### 4.5.2. Introdução

Como ocorre em outros textos científicos, essa seção é dedicada a descrever os conceitos fundamentais inerentes à pesquisa, além de discutir trabalhos relacionados, apoiando-se em citações de referências relevantes. Essa discussão deve conduzir ao problema de pesquisa que está sendo investigado, o qual é então delineado de forma clara na formulação da questão de pesquisa do projeto.

Conclui-se a seção informando sucintamente como o projeto pretende responder à questão de pesquisa e explicitando o envolvimento de pessoas, por exemplo:

*Nesse cenário, estendemos a plataforma <Nome> para permitir <Nova funcionalidade a ser testada> como detalhado em <citação para relatório ou artigo>. Assim, os objetivos desta pesquisa são realizar avaliações quantitativas e qualitativas da percepção dos usuários em relação às novas funcionalidades disponibilizadas, bem como da usabilidade do sistema, durante as sessões de teste.*

#### 4.5.3. Desenho

O desenho deve detalhar o tipo da pesquisa que envolve pessoas (ex.: observacional qualitativa, experimental quantitativa etc.). Deve ainda descrever a parte do ensaio que especifica os procedimentos que serão avaliados, as unidades experimentais, a variável em análise e o modo como procedimentos serão designados às unidades experimentais. Uma referência abrangente sobre métodos qualitativos e quantitativos, de modo geral, é o livro de Creswell and Creswell [32]. Já o livro da Goldenberg [33], professora da Universidade Federal do Rio de Janeiro, oferece uma base sólida para pesquisa qualitativa: escrito originariamente para apoiar pesquisadores de Ciências Sociais, o material se aplica a muitas das pesquisas com pessoas nas áreas de Web, Multimídia e IHC. Especificamente para a

área de IHC, recomenda-se o livro de Lazar et al. [34].

#### 4.5.4. Hipótese

A Hipótese é definida como uma resposta suposta, provável e provisória ao problema, pode ser testada como verdadeira ou falsa pelo experimento delineado, independentemente de valores de opinião:

- *Suposta*: a hipótese é uma suposição, pois o experimento ainda não produziu evidências para sua confirmação ou negação;
- *Provável*: toda hipótese deve ser passível de teste experimental para sua comprovação;
- *Provisória*: a hipótese será aceita até sua confirmação ou negação pelos resultados do experimento.

A hipótese é um instrumento de pesquisa que orienta a teoria e a metodologia adotadas. Ela é formulada com base em uma determinada estrutura teórica e em resposta a um problema a ser solucionado, implicando a necessidade de comprovação por meio da metodologia e da pesquisa.

É necessário observar que, no sentido metodológico mais restrito, a hipótese é uma formulação preliminar, destinada a dar início à pesquisa. Ao longo do processo de investigação, a hipótese pode ser confirmada ou refutada, sem que isso desqualifique o papel que desempenhou em impulsionar a pesquisa. De fato, de acordo com a concepção moderna de ciência, todas as ideias científicas estão em constante teste. Nesse contexto, pode-se afirmar que as ideias científicas, independentemente de serem classificadas como conjecturas, postulados (axiomas) ou leis, são e sempre serão hipóteses. Hipóteses e fatos são elementos intrínsecos e indispensáveis à construção teórica. Além disso, hipóteses são amplamente utilizadas nas várias áreas científicas, incluindo as Ciências Humanas.

Finalmente, é necessário destacar que, no caso de pesquisas de caráter qualitativo, em particular de métodos próprios das áreas de Ciências Humanas e Sociais tratadas pela Resolução CNS 510/2016, pode não haver definição prévia de hipótese. Nesses casos, deve-se informar que o estudo não tem hipótese prévia definida.

#### 4.5.5. Objetivos:

Deve-se definir os objetivos da pesquisa e há duas modalidades possíveis:

- *Objetivo Primário*: É o objetivo geral e amplo da pesquisa;
- *Objetivos Secundários*: Dizem respeito aos objetivos específicos (campo não obrigatório na PB).

Na maioria dos casos, os objetivos secundários estão diretamente relacionados ao objetivo primário e são estabelecidos para alcançar esse objetivo principal de maneira mais eficaz. Em algumas situações, pode haver objetivos secundários que são considerados benefícios adicionais ou colaterais do projeto: podem surgir como resultado das ações tomadas para alcançar o objetivo principal, mas não são o foco principal do projeto

#### 4.5.6. Metodologia Proposta

A seção de metodologia deve explicar como o estudo será realizado, dividindo-o em etapas e descrevendo o envolvimento ou não de participantes em cada uma delas. Deve

abordar o consentimento informado, detalhamento da participação, proteção de dados pessoais e procedimentos éticos, bem como a descrição da análise dos dados. A redação deve ser clara, completa e atender aos padrões éticos e científicos. A transparência e o detalhamento nessa seção permitem que o CEP avalie a viabilidade e a ética do projeto de pesquisa. Os principais aspectos são:

- *Descrição Geral da Metodologia*: Inicie com uma breve descrição geral da metodologia adotada no estudo e o método de pesquisa (e.g, estudo de caso, pesquisa qualitativa);
- *Etapas do Estudo*: Divida o estudo em etapas ou fases. Cada etapa deve ser claramente identificada e nomeada. Detalhe as etapas do estudo na ordem em que serão realizadas, incluindo as que não envolvem participantes;
- *Envolvimento de Participantes*: Para cada etapa, especifique se participantes estarão envolvidos. É necessário indicar claramente se haverá interação com participantes em cada fase do estudo. Especifique os momentos em que participantes serão selecionados, instruídos sobre sua participação, e em que receberão uma devolutiva de sua participação;
- *Detalhamento da Participação*: Quando participantes estiverem envolvidos em uma etapa específica, explique como essa participação ocorrerá. Isso pode incluir entrevistas, questionários, coleta automática de dados da interação com um sistema, observações etc;
- *Consentimento Informado*: Certifique-se de abordar a obtenção do consentimento informado de participantes. Descreva como o processo de consentimento será conduzido, incluindo a documentação e o esclarecimento de participantes sobre os objetivos, riscos e benefícios do estudo;
- *Proteção da Privacidade e Dados Pessoais*: Se o estudo envolver a coleta de informações pessoais ou confidenciais, explique as medidas adotadas para proteger a privacidade e a segurança dos dados de participantes;
- *Procedimentos Éticos*: Descreva qualquer procedimento ético específico que será seguido durante o estudo, como o registo e a anonimização dos dados etc;
- *Análise dos Resultados*: Sumarize como os dados serão analisados;
- *Referências*: Caso você utilize métodos ou técnicas específicas, cite as fontes ou literatura que embasam a sua escolha metodológica.

#### **4.5.7. Critérios de Inclusão e Exclusão**

Deve-se definir os critérios para a participação na pesquisa:

- *Critérios de Inclusão*: Quem serão as pessoas a serem incluídas no projeto, por quais motivos e como essas serão recrutadas;
- *Critérios de Exclusão*: Após o recrutamento das pessoas realizado de acordo com critérios de inclusão, explicitar quais critérios de exclusão serão adotados caso seja preciso excluir um participante *recrutado*.

#### **4.5.8. Riscos**

A resolução 466/2012 assume que toda pesquisa envolvendo pessoas apresenta algum tipo de risco, que deve ser explicitado no item 'Risco'. Nota-se que os riscos são para as pessoas participantes e não para a pesquisa. Exemplos de risco incluem: risco emocional,

risco de interrupção do serviço, risco de exigência de tempo, risco econômico, risco à confidencialidade dos dados, risco de exposição à radiação, entre outros. Os riscos devem estar claros também no TCLE.

Os riscos caracterizados como mínimos podem ser, por exemplo, responder um questionário que pode gerar fadiga. Nenhuma pesquisa envolvendo pessoas está isenta de riscos, e o CEP é um intermediário capaz de apontar alguns. No caso de a pesquisa relembrar experiências negativas que podem gerar mal-estar em participantes, se a pessoa pesquisadora perceber a interação como inócua, cabe ao CEP indicar possíveis riscos.

#### **4.5.9. Benefícios**

A resolução 466/2012 [11] estabelece que toda pesquisa envolvendo pessoas deve apresentar benefícios para quem participa e não para a pesquisa. Exemplos de benefícios incluem: relevância e aplicabilidade, aquisição de conhecimento e aprendizado, desenvolvimento de habilidades, reflexão sobre experiências e práticas, possibilidade de oferecer retorno ou resposta aos pesquisadores, entre outras.

#### **4.5.10. Metodologia de Análise de Dados**

Explicitar como serão trabalhados estatística ou qualitativamente os dados coletados.

#### **4.5.11. Desfechos**

O desfecho primário indica o “objetivo Primário” do estudo. É o resultado principal que é medido no final de um estudo para determinar se um tratamento específico foi eficaz e atingiu o objetivo geral da pesquisa. O desfecho primário é determinado antes do início do estudo e é geralmente a variável mais relevante e convincente em relação ao objetivo principal da pesquisa. É imperativo que cada estudo tenha apenas um desfecho primário, embora considerações de segurança e tolerabilidade também possam ser incluídas como desfechos primários em alguns casos. Outras variáveis, como medidas de qualidade de vida e economia em saúde, também podem ser consideradas desfechos primários. A seleção do desfecho primário deve seguir as normas e padrões estabelecidos no campo de pesquisa relevante e deve ser pré-especificada no protocolo do estudo, juntamente com uma justificativa para a sua escolha. É inadequado redefinir o desfecho primário após a análise dos resultados do estudo, pois isso pode gerar interferências difíceis de serem avaliadas.

Desfechos secundários estão relacionados aos “objetivos Secundários”, delineados na seção correspondente do projeto. A relação entre essas duas seções é de sinergia: os desfechos secundários devem ser escolhidos de forma a complementar e enriquecer a consecução dos objetivos específicos.

#### **4.5.12. Cronograma de Execução**

Define etapas previstas no projeto, que *devem* corresponder àquelas explicitadas na seção *Metodologia Proposta*. É crítico prever o início das atividades após a aprovação do CEP, considerando o tempo para aprovação. Na PB, cada etapa é explicitada com seu nome, data de início e final, de modo similar ao apresentado no Quadro 4.5.12.



<b>Etapas</b>	<b>Início</b>	<b>Fim</b>
Recrutamento de participantes	01/10/2023	31/10/2023
Detalhamento das tarefas (não envolve participantes)	01/11/2023	15/11/2023
Sessão de Teste e Coleta de Dados	16/11/2023	01/06/2024
Análise dos resultados (não envolve participantes)	02/06/2024	31/07/2024

Quadro 4.5.2: Exemplo de cronograma seguindo formato da PB.

#### 4.5.13. Orçamento Financeiro

É necessário detalhar os custos associados ao projeto, que podem incluir despesas como material gráfico, transporte e alimentação de participantes. Na PB, o campo referente aos custos é obrigatório, pois parte-se do princípio de que *toda pesquisa* implica algum custo, mesmo que mínimo. O campo de preenchimento do orçamento irá solicitar: Identificação do Orçamento, Tipo (que pode ser: custeio, capital, bolsas e outros), e Valor em Reais (R\$). O Quadro 4.5.3 ilustra um orçamento reduzido.

As resoluções determinam que *não é permitido* realizar pagamentos para participantes por sua participação em projetos de pesquisa, de qualquer natureza [11, 25], por exemplo, número de rifa para um prêmio específico. As pessoas pesquisadoras devem *explicitar* essa norma tanto no TCLE quanto no próprio projeto de pesquisa, por exemplo:

*Conforme as resoluções vigentes, este estudo não oferece compensação às pessoas participantes pela sua participação. Asseguramos que a participação é voluntária e não associada a qualquer forma de remuneração.*

Assim, só podem ser compensados gastos específicos para realização da pesquisa, como transporte ou uso de Internet por parte de participantes. Por outro lado, o orçamento *deve explicitar* o reconhecimento dos custos que participantes terão ao participar do estudo, como despesas de alimentação, traslado ou uso de Internet. Portanto, *deve-se* prever uma verba no orçamento para ressarcir esses custos. Em um projeto no qual a coleta de dados se dá por meio de uma sessão em que participantes participam remotamente, poderia ser incluída uma declaração como:

*Garantimos que você pode ser ressarcido pelas despesas relacionadas à sua participação no que tange custos adicionais de Internet, energia elétrica ou outra despesa decorrente do uso do computador e de conexão à Internet durante o período da pesquisa. O reembolso será limitado a R\$10.00 por sessão remota de avaliação dos recursos de colaboração. Para solicitar reembolso, envie e-mail com assunto “solicito reembolso” para a pessoa pesquisadora responsável após a realização da sessão remota, informando data e hora de participação, e o identificador da sessão.*

No caso de projeto conduzido em um laboratório de Computação típico, a descrição do orçamento poderia incluir:

*O projeto possui um orçamento financeiro reduzido, justificado por vários fatores. Primeiramente, serão empregados equipamentos e software de projetos anteriores < citar projetos >, o que elimina a necessidade de investimentos adicionais nesses recursos. Adicionalmente, a instituição de pesquisa < incluir nome da instituição > financiará o projeto no que tange à verba de custeio, cobrindo contas de água, luz, Internet, funcionários e outros custos operacionais.*

Conseqüentemente, o projeto deve detalhar o valor de todas as verbas necessárias, a fonte dos recursos (por exemplo, ‘Financiamento Próprio’) e o procedimento para que participantes solicitem o ressarcimento. Essas informações também devem ser claramente especificadas no TCLE, assegurando transparência e compreensão por parte de participantes – como é o caso do exemplo de TCLE do Apêndice A.

Orçamento	
Item	Valor
Material permanente	R\$0
Material de consumo	R\$0
Serviços de Terceiros	R\$0
Despesas com transporte	R\$0
Diárias	R\$0
Verba para ressarcimento de despesas, de participantes, relativas a custos adicionais decorrentes do uso do computador e Internet durante a Sessão de Teste e Coleta de Dados ( <b>Financiamento Próprio</b> )	R\$20.00
Custeio relativo à infraestrutura para Grupo de Pesquisa	<b>Institucional</b>

Quadro 4.5.3: Exemplo de orçamento.

#### 4.5.14. Bibliografia

Listagem das referências completas utilizadas na pesquisa e citadas ao longo do projeto.

#### 4.5.15. Anexo e/ou Apêndices

Os anexos do projeto de pesquisa incluem, se aplicável, documentos elaborados por terceiros, como cartas de autorização da instituição parceira ou carta de aceite de um local coparticipante.

Os apêndices do projeto de pesquisa incluem arquivos complementares e elaborados pelas pessoas pesquisadoras. São exemplos o TCLE, formulários de levantamento de perfil, termos de autorização de uso de imagem e nome da pessoa participante, e demais formulários que serão respondidos por participantes. O TCLE costuma ser obrigatório e, no caso de solicitação de dispensa do mesmo, há um campo na PB para inserção de justificativa.

#### 4.5.16. Outras Informações

Além dos itens apontados acima que devem estar descritos no projeto de pesquisa, ao preencher a PB com os dados do projeto, deve-se preencher o *número total de participantes* e as intervenções a serem realizadas com os mesmos. É sugerido que essas informações venham descritas na seção de metodologia proposta do projeto. As mesmas informações, no entanto, têm preenchimento obrigatório na plataforma. Após informar a quantidade total de participantes, será preciso informar cada intervenção a ser feita durante a pesquisa e a quantidade de pessoas relacionadas em cada uma.

#### 4.5.17. Folha de Rosto

Uma *Folha de Rosto* deve obrigatoriamente ser anexada aos arquivos submetidos na PB e é gerada pela mesma, na última etapa de preenchimento do projeto. Tal documento contém

informações como: o nome do pesquisador principal, a quantidade de participantes na pesquisa, a instituição sede, dados do representante legal da instituição sede.

Esse documento deve ser gerado, assinado pelo pesquisador principal e enviado ao setor que responde pelo representante legal da instituição sede da pesquisa para assinatura. Ao enviar a folha de rosto para assinatura de uma pessoa representante legal da instituição, a pessoa pesquisadora responsável deve enviar junto o projeto completo (referenciado na plataforma como ‘Brochura Investigador’) para que o mesmo seja apreciado pelo representante legal.

O representante legal da instituição, ao concordar com a submissão do projeto, insere sua assinatura na folha de rosto e a devolve para a pessoa pesquisadora. É essa folha que deve ser inserida na PB para submissão do projeto.

#### 4.6. Pendências Comuns

A *Instância de Ciências Humanas e Sociais da Comissão Nacional de Ética em Pesquisa (Conep) (ICHES)*, conforme disposto na Resolução CNS nº 510 de 2016, disponibilizou o “Manual de Orientação: Pendências Comuns em Protocolos de Ciências Humanas e Sociais no Sistema CEP/Conep.”<sup>11</sup> O documento também está disponibilizado na página de manuais da PB (ver Figura 4.2). A leitura desse manual mitiga e facilita a identificação erros evitáveis antes da submissão do projeto.

Alguns destaques:

- I. *Folha de Rosto*: Deve conter todos os campos preenchidos, datados e assinados com identificação dos signatários. As informações devem ser compatíveis com as do protocolo, e as assinaturas devem identificar claramente o nome completo e a função do signatário. A assinatura deve ser digital e certificada. É necessária a inclusão ou atualização da folha de rosto em casos de mudança de área temática. Deve-se evitar conflitos de interesse, como quando a pessoa pesquisadora é também a responsável institucional, exigindo a assinatura de outra responsável institucional sem conflitos de interesse;
- II. *Informações Básicas da Pesquisa e Área Temática*: As informações básicas devem ser claramente descritas e consistentes em todos os documentos;
- III. *Projeto Detalhado*: Deve-se evitar incoerências nas informações entre o projeto detalhado e outros documentos, como orçamento financeiro, cronograma, número de participantes, objetivos e metodologia;
- IV. *Consentimento e Assentimento Livre e Esclarecido*: Deve-se usar linguagem clara e de fácil entendimento, adequada ao público-alvo. O processo de assentimento deve adotar uma linguagem simples, compreensível e diferente da usada no consentimento destinado aos responsáveis legais. É necessário escrever o registro de consentimento/assentimento para adequá-lo à compreensão da pessoa participante, considerando os diferentes grupos do estudo. O processo de consentimento e assentimento deve ser contínuo, aberto ao diálogo e ao questionamento, e pode ser retirado a qualquer momento sem prejuízo à pessoa participante. Deve-se descrever o processo de comunicação do consentimento e assentimento, evitando formalida-

---

<sup>11</sup>Primeira edição: [https://conselho.saude.gov.br/images/Manual\\_de\\_Orientacao\\_Pendencias\\_comuns\\_em\\_protocolos\\_CHS\\_versao\\_23102023.pdf](https://conselho.saude.gov.br/images/Manual_de_Orientacao_Pendencias_comuns_em_protocolos_CHS_versao_23102023.pdf) [acesso 30/03/2024]

des excessivas e promovendo a confiança e a interação. O registro deve garantir a liberdade da pessoa participante sobre sua participação e uso de dados, agora e no futuro. Informações sobre ressarcimento e formas de cobertura de despesas devem ser incluídas. O uso de imagem e/ou voz de participantes deve ser detalhadamente descrito e consentido. O consentimento e o assentimento devem considerar características individuais, sociais, econômicas e culturais das pessoas participantes. Deve-se descrever os procedimentos da pesquisa de forma clara e acessível para participantes. Informações sobre riscos, desconfortos, estresse ou cansaço durante a pesquisa devem ser claramente comunicadas, assegurando à pessoa participante o direito de não responder ou se retirar da pesquisa a qualquer momento. No caso de menores de idade ou pessoas com ausência de autonomia para consentir, deve-se respeitar a autonomia e a opção de participar, mesmo após a consulta aos responsáveis legais. Quando o registro do consentimento/assentimento for escrito, a pessoa participante deve receber uma via assinada, e o documento deve ser numerado para garantir sua integridade;

- V. *Pesquisas remotas*: É necessário detalhar todas as etapas não presenciais do estudo, incluindo modelos de formulários e termos apresentados para participantes;
- VI. *Dados de Participantes em Pesquisas com Colaboração Internacional e Uso de Banco de Dados*. Deve-se garantir proteção ao enviar dados para o exterior. Um Termo de Compromisso de Uso de Banco de Dados é Necessário para o compartilhamento de dados;
- VII. *Cronograma*: Deve descrever a duração total e as diferentes etapas da pesquisa, com compromisso explícito de que a pesquisa iniciará apenas após aprovação pelo Sistema CEP/Conep;
- VIII. *Orçamento Financeiro*: Todos os protocolos devem incluir um orçamento financeiro detalhado, mesmo em pesquisas com financiamento próprio.

#### **4.7. Onde? Como?**

A Plataforma Brasil (PB) é um sistema web. Com o projeto em mãos (tanto o documento fonte contendo o texto dos elementos que fazem parte do projeto, como a versão do projeto em PDF para *upload*), a pessoa pesquisadora deve submeter o projeto seguindo as instruções detalhadas no Manual do Pesquisador da PB (ver Figuras 4.1 e 4.2). O documento fonte é necessário porque os elementos textuais serão copiados para os campos correspondentes na plataforma utilizando a função copiar e colar do navegador web.

Uma versão na forma de roteiro com as imagens para submissão inicial de projeto na Grande àrea de Ciências Exatas e da Terra, na qual se enquadra a maioria dos projetos da área de Web e Multimídia, foi disponibilizada por Pimentel [35].

##### **4.7.1. Cadastro das pessoas pesquisadoras**

O primeiro passo é efetuar o cadastro das pessoas pesquisadoras, sendo necessários:

- Número do Cadastro de Pessoa Física (CPF);
- Currículo do pesquisador, que pode ser um arquivo em formato doc, docx, odt ou pdf, ou então *a referência do currículo na Plataforma Lattes*;
- Uma cópia digitalizada de um documento de identificação com foto (jpg ou pdf);
- Um endereço de e-mail válido e em uso.

A pessoa pesquisadora também deverá ter em mãos o CPF de cada membro da equipe para registro no projeto. Esse dado é suficiente para o cadastro, que é associado à Plataforma Lattes. Quando da submissão do projeto pelo pesquisador responsável, cada membro da equipe recebe um e-mail informando sua inclusão como membro da equipe.

Os demais passos correspondem ao preenchimento do formulário como detalhado na próxima seção. Esse preenchimento será facilitado se o projeto de pesquisa seguir a estrutura indicada na Seção 4.5.

#### **4.7.2. Criar e submeter projeto na Plataforma Brasil e o CAAE**

O preenchimento do formulário se dá em seis telas:

- *Tela 1*: Informações Preliminares: sobre a equipe e instituições envolvidas;
- *Tela 2*: Área de Estudo (áreas temáticas especiais, se aplicáveis (e.g. genética humana ou estudos com populações indígenas), e grandes áreas do conhecimento (e.g., Ciências Exatas e da Terra). Título principal (duas versões: público e interno). Contatos público e científico;
- *Tela 3*: Desenho de Estudo/Apoio Financeiro: desenho do estudo, fontes de financiamento e palavras-chave;
- *Tela 4*: Detalhamento do Estudo: Resumo, Introdução, Hipótese, Objetivo Primário, Objetivo Secundário, Metodologia Proposta, Critério de Inclusão, Critério de Exclusão, Riscos, Benefícios, Metodologia de Análise de dados, Desfecho Primário, Desfecho Secundário, Tamanho da Amostra no Brasil, Países de Recrutamento e o correspondente Número de Participantes na Pesquisa;
- *Tela 5*: Outras Informações: indicação do uso ou não de fontes secundárias de dados; o número de participantes recrutados que sofrerão algum tipo de intervenção no centro de pesquisa, e os grupos em que serão divididos; indicação se o estudo envolve ou não múltiplos centros, especificando quais são e quantas pessoas participantes serão recrutadas em cada um; manifestação sobre a solicitação de dispensa ou não do TCLE, com a justificativa correspondente, se aplicável; indicação de se haverá ou não retenção de amostras para armazenamento em bancos, e a justificativa, se houver retenção; detalhamento do cronograma de execução, incluindo cada etapa (nome, datas de início e fim, e atividades correspondentes); detalhamento do orçamento financeiro, incluindo identificação, tipo (por exemplo, custeio ou capital) e valor em reais; espaço para outras informações a critério do pesquisador; e a bibliografia;
- *Entre a Tela 5 e a Tela 6*: janela flutuante para anexar Folha de Rosto, Projeto Detalhado (Brochura Investigador), TCLE ou justificativa de ausência.
- *Tela 6*: Finalizar: Manter ou não sigilo do projeto e por quanto tempo, se aplicável. Aceite do termo de compromisso e envio ao CEP.

Para um exemplo de preenchimento na Grande Área de Ciências Exatas e da Terra, sugere-se o roteiro com imagens disponibilizado por Pimentel [35].

#### **4.7.3. Após a submissão**

A pessoa responsável pode acompanhar pelo sistema o andamento da avaliação do projeto na PB, que deve seguir os tempos indicados no Quadro 4.3.3. Em caso de pendências

serem identificadas e demandarem ação por parte da pessoa responsável, esta recebe um e-mail. A pessoa responsável também recebe e-mail ao final do processo de avaliação e o parecer estar liberado. Como exemplo, a Figura 4.3 ilustra um caso real de histórico de trâmites, cujos passos são como segue:

1. *Submetido pela CONEP para avaliação do CEP.* Após envio pela pessoa pesquisadora, o setor administrativo do Conep encaminha para um CEP, usualmente um CEP local da instituição proponente (ou outro indicado no TCLE);
2. *Resultado: Rejeição do PP.* A secretaria do CEP realiza a verificação de documentos. Nesse caso, o projeto de pesquisa foi rejeitado pelo CEP na etapa de verificação por necessidade de ajustes no TCLE. *A pessoa pesquisadora responsável é notificada por e-mail.*
3. *Submetido para avaliação do CEP.* A pessoa pesquisadora responsável realiza os ajustes conforme as orientações recebidas, e resubmete o projeto de pesquisa;
4. *Aceitação do PP.* A secretaria do CEP verifica a documentação e, neste caso, o projeto de pesquisa foi aceito para ser avaliado pelo CEP;
5. *Indicação de Relatoria.* A secretaria do CEP indica, no sistema, o nome de uma pessoa relatora para analisar o projeto de pesquisa;
6. *Confirmação de Indicação de Relatoria.* A pessoa coordenadora do CEP realiza a confirmação da designação do relator para analisar o projeto de pesquisa.
7. *Aceitação de Elaboração de Relatoria:* A pessoa relatora aceita a responsabilidade de elaborar o parecer sobre o projeto de pesquisa.
8. *Parecer do relator emitido:* A pessoa relatora emite seu parecer sobre o projeto de pesquisa.
9. *Parecer do colegiado emitido:* A pessoa coordenadora do CEP registra o parecer do CEP sobre o projeto de pesquisa.
10. *Parecer liberado:* A pessoa coordenadora do CEP libera o parecer, que indica se o projeto de pesquisa foi aprovado ou não para realização. *A pessoa pesquisadora responsável é notificada por e-mail.*

#### **4.7.4. Após a aprovação: execução do projeto e divulgação dos resultados**

Após aprovação do projeto por um CEP, que deve seguir os prazos de tramitação sumarizados no Quadro 4.3.3, o parecer de aprovação é acompanhado de um Certificado de Apresentação de Apreciação Ética (CAAE).

Quando do recrutamento de participantes, ou quando da entrega do TCLE a elas, o número do CAAE pode ser incluído junto com os detalhes de contato dos pesquisadores e do CEP que aprovou o projeto. Um exemplo é o TCLE de Rodrigues and Moreira [36].

Além disso, o número do CAAE pode ser referenciado nas publicações resultantes da pesquisa juntamente com uma declaração explícita dos cuidados tomados pelas pessoas pesquisadoras, como no trabalho de Verhalen and Rodrigues [37] intitulado *Design of Therapeutic Digital Games that Support Dialogue with Children about Death*:

*“O uso dos jogos com o público infantil foi aprovado pelo Comitê de Ética em Pesquisa, com o número CAAE: 52278121.1.0000.5154. Durante as avaliações deste projeto, os participantes deram seu consentimento para participação. Os textos dos termos de consentimento foram lidos em conjunto, e os voluntários foram infor-*

HISTÓRICO DE TRÂMITES							
Apreciação	Data/Hora	Tipo Trâmite	Versão	Perfil	Origem	Destino	Informações
PO	10/07/2023 10:39:26	Parecer liberado	1	Coordenador	Escola de Artes, Ciências e Humanidades da Universidade de São Paulo - EACH/USP	PESQUISADOR	
PO	10/07/2023 08:35:34	Parecer do colegiado emitido	1	Coordenador	Escola de Artes, Ciências e Humanidades da Universidade de São Paulo - EACH/USP	Escola de Artes, Ciências e Humanidades da Universidade de São Paulo - EACH/USP	
PO	09/07/2023 22:55:19	Parecer do relator emitido	1	Membro do CEP	Escola de Artes, Ciências e Humanidades da Universidade de São Paulo - EACH/USP	Escola de Artes, Ciências e Humanidades da Universidade de São Paulo - EACH/USP	
PO	09/07/2023 22:47:10	Aceitação de Elaboração de Relatoria	1	Membro do CEP	Escola de Artes, Ciências e Humanidades da Universidade de São Paulo - EACH/USP	Escola de Artes, Ciências e Humanidades da Universidade de São Paulo - EACH/USP	
PO	25/06/2023 19:53:57	Confirmação de Indicação de Relatoria	1	Coordenador	Escola de Artes, Ciências e Humanidades da Universidade de São Paulo - EACH/USP	Escola de Artes, Ciências e Humanidades da Universidade de São Paulo - EACH/USP	
PO	21/06/2023 08:36:59	Indicação de Relatoria	1	Secretária	Escola de Artes, Ciências e Humanidades da Universidade de São Paulo - EACH/USP	Escola de Artes, Ciências e Humanidades da Universidade de São Paulo - EACH/USP	
PO	21/06/2023 08:36:18	Aceitação do PP	1	Secretária	Escola de Artes, Ciências e Humanidades da Universidade de São Paulo - EACH/USP	Escola de Artes, Ciências e Humanidades da Universidade de São Paulo - EACH/USP	
PO	20/06/2023 19:20:52	Submetido para avaliação do CEP	1	Pesquisador Principal	PESQUISADOR	Escola de Artes, Ciências e Humanidades da Universidade de São Paulo - EACH/USP	
PO	20/06/2023 13:24:31	Rejeição do PP	1	Secretária	Escola de Artes, Ciências e Humanidades da Universidade de São Paulo - EACH/USP	PESQUISADOR	Prezado(a) pesquisador(a). <a href="#">Ver mais &gt;&gt;</a>
PO	20/06/2023 11:05:14	Submetido pela CONEP para avaliação do CEP	1	Administrador	CONEP	Escola de Artes, Ciências e Humanidades da Universidade de São Paulo - EACH/USP	

**Figura 4.3. Exemplo de Histórico de Trâmites da Plataforma Brasil (PB).**

*mados sobre os riscos e benefícios da pesquisa, além de que sua participação era voluntária e que poderiam se retirar do estudo a qualquer momento. Também foi garantido que suas identidades seriam anonimizadas, de modo que não fosse possível identificá-los.*

*Nos estudos conduzidos por webconferência, que requeriam a gravação das telas, os voluntários assinaram um termo de cessão de imagem e som. Os dados coletados estão sob a posse exclusiva das responsáveis pela pesquisa e serão armazenados até o ano de 2026, conforme o prazo estipulado.*

*Ressalta-se que nos artigos publicados os participantes foram anonimizados. A plataforma RUFUS é um sistema de código aberto e atualmente está em processo de registro de software junto aos órgãos responsáveis da Universidade de São Paulo, campus São Carlos. A plataforma está em conformidade com os princípios da Lei Geral de Proteção de Dados (LGPD).” [37, p.7].*

#### 4.7.5. Após a aprovação: emendas e extensões ao projeto

Após a aprovação do projeto, pode ser necessário realizar ajustes no desenho do estudo, por exemplo, ou para estender sua duração. A Norma Operacional N° 001/2013 CONEP/CNS define os conceitos de emenda e extensão [17, p.3].

- “*Emenda é toda proposta de modificação ao projeto original, apresentada com a justificativa que a motivou. As emendas devem ser apresentadas ao CEP de forma clara e sucinta, identificando a parte do protocolo a ser modificada e suas justificativas. A emenda será analisada pelas instâncias de sua aprovação final (CEP e/ou CONEP).*”
- “*Extensão é a proposta de prorrogação ou continuidade da pesquisa com os mes-*

*mos participantes recrutados, sem mudança essencial nos objetivos e na metodologia do projeto original. Havendo modificações importantes de objetivos e métodos, deve ser apresentado outro protocolo de pesquisa.”*

A pessoa pesquisadora pode enviar uma *emenda* ao Sistema CEP/Conep através da Plataforma Brasil. A análise da emenda será conduzida pelas instâncias responsáveis por sua aprovação anterior (CEP e/ou CONEP). É possível enviar emendas apenas para pesquisas que já foram aprovadas. O sistema permite o envio de apenas uma emenda por vez, apresentando a opção novamente somente quando a análise ética da emenda anterior tiver sido finalizada.

A necessidade de uma *extensão* em um projeto de pesquisa pode surgir quando o tempo originalmente previsto para a coleta de dados e análise se mostra insuficiente para alcançar os objetivos propostos. A extensão permite que a equipe tenha mais tempo para conduzir o estudo, sem a necessidade de recomeçar o processo de submissão e aprovação do protocolo de pesquisa. As modificações propostas pelo pesquisador responsável não podem descaracterizar o estudo originalmente proposto e aprovado.

Como exemplo, no projeto realizado para o desenvolvimento do método *Experience Sampling and Programmed Intervention Method* [38] foram utilizados os recursos de emendas e de extensão. Por exemplo, após a realização das entrevistas de pré-design com os especialistas em saúde, percebeu-se a importância de estender o modelo, o que demandou incluir novas funcionalidades nos aplicativos web e móvel. Além disso, durante o processo de design participativo houve sugestões que demandaram modificações no modelo e no protótipo de software correspondente. Foi necessário também realizar emendas para a inclusão de novos participantes em novos estudos de caso, exigindo ajustes na abordagem metodológica original. Além disso, devido à complexidade do projeto e durante sua evolução ao longo do processo, foi necessário solicitar uma extensão para o período de avaliação do método e do modelo ESPIM. Isso permitiu realizar uma análise mais abrangente dos resultados obtidos, além de fornecer suporte adicional aos estudos de caso reais e conduzir entrevistas pós-design com maior profundidade.

## **4.8. Exemplos de pesquisas que envolvem seres humanos**

### **4.8.1. Casos práticos na área de Web e Multimídia: biblioteca digital da ACM**

Para ilustrar como a submissão dos projetos Comitê de Ética em Pesquisa (CEP) é crucial para garantir a integridade e a responsabilidade das pesquisas que envolvem interações com seres humanos na área de web e multimídia, procurou-se trabalhos divulgados na biblioteca digital da ACM.

- Uma busca com os termos *media*, *user*, e *interaction* no resumo do artigo, e *Institutional Review Board (IRB)* (em português: *CEP*) no corpo do artigo, seleção dos artigos publicados pela ACM nos últimos cinco anos, retornou um conjunto de 78 artigos [39].
- A busca equivalente para os termos *web*, *user*, e *interaction* retornou um conjunto de 42 artigos [40].

As buscas retornam, por padrão, os artigos ordenados por relevância. O artigo que retorna em primeiro lugar na busca com os termos *media*, *user*, e *interaction* é intitulado



*Instagram Data Donation: A Case Study on Collecting Ecologically Valid Social Media Data for the Purpose of Adolescent Online Risk Detection* [41]. O artigo que retorna em primeiro lugar na busca com os termos *web*, *user*, e *interaction* tem o título: *Exploring Web-Based VR for Participatory Robot Design* [42]

Ao ajustar a apresentação dos resultados da busca pelos artigos mais citados, os resultados são *Algorithmic Folk Theories and Identity: How TikTok Users Co-Produce Knowledge of Identity and Engage in Algorithmic Resistance* [43] e *Covid-19 highlights the issues facing blind and visually impaired people in accessing data on the web* [44] para a primeira e a segunda buscas, respectivamente.

A menção a *IRB* nos artigos não necessariamente indica que os projetos tenham passado por um processo formal de submissão. Em alguns casos, pode ter ocorrido a dispensa da necessidade de submissão ao CEP devido à natureza específica da pesquisa, ou à avaliação de que os riscos potenciais para os participantes são mínimos ou inexistentes, e a isenção pode ter sido concedida pelo próprio CEP, como exigido pela legislação americana [45].

*Instagram Data Donation: A Case Study on Collecting Ecologically Valid Social Media Data for the Purpose of Adolescent Online Risk Detection*. O artigo tem como pessoas participantes uma população vulnerável: adolescentes. Razi et al. [41] registram um estudo de caso com o objetivo de melhorar a segurança online de adolescentes, envolvendo jovens com idades entre 13 e 21 anos que são usuários do Instagram. Adolescentes contribuíram doando e anotando seus dados do *Instagram*. Com foco em garantir a confidencialidade e a privacidade dos adolescentes participantes, Razi et al. [41], além de obterem aprovação do IRB de uma das instituições envolvidas para o estudo (Univ. Central Flórida, Estados Unidos da América (EUA)), explicitam que adotaram uma série de medidas para garantir a proteção dos adolescentes e a condução ética da coleta de dados:

- as pessoas pesquisadoras se identificaram como “notificadoras obrigatórias de abuso infantil para casos urgentes de risco a menores,” o que significa que elas eram obrigadas a entrar em contato com a Linha Direta de Abuso da Flórida para relatar qualquer caso de suspeita razoável de que uma criança tenha sido abusada, negligenciada ou ameaçada de dano no estado onde a pesquisa foi realizada (Flórida);
- a equipe adquiriu um *Certificado de Confidencialidade do National Institute of Health (NIH), EUA*, o que permite proteger as informações pessoais dos adolescentes participantes e evitar que sejam solicitadas legalmente através de intimações legais;
- decidiu-se remover qualquer informação pessoalmente identificável dos dados textuais ou de imagem, incluindo parafrasear ou editar o conteúdo de quaisquer dados apresentados. Devido à natureza sensível do conjunto de dados, não será disponibilizado publicamente para uso, mas poderá ser compartilhado como um conjunto de dados restrito. Para acessar os dados brutos mais sensíveis (desidentificados dentro de padrões razoáveis usando ferramentas de desidentificação automatizadas), será necessário que pessoas interessadas apresentem um histórico estabelecido de pesquisas relevantes e publicadas para justificar por que devem ter acesso a esses dados, além de obterem aprovação do Comitê de Ética em Pesquisa de suas instituições de origem, e de atenderem aos requisitos para reutilização e redistribuição;
- todas as pessoas pesquisadoras completaram o treinamento exigido pela universi-

- dade para participação em projetos de pesquisa envolvendo seres humanos;
- todas as pessoas pesquisadoras completaram o treinamento do Programa de Proteção à Juventude da instituição;
  - adolescentes participantes com menos de 18 anos precisaram obter o consentimento dos pais e preencher termos de assentimento, enquanto participantes maiores de 18 anos preencheram termos de consentimento. Nos formulários de consentimento e assentimento, foram incluídas informações detalhadas sobre a pesquisa, seu processo, benefícios e riscos potenciais;
  - esclarecimentos foram fornecidos sobre quais informações seriam coletadas, como seriam armazenadas e protegidas, bem como outras informações relevantes para a participação no estudo.

*Exploring Web-Based VR for Participatory Robot Design.* Neste caso, o grupo de pessoas participantes também era composto por adolescentes, como reportado por Bhatia et al. [42]. O estudo tinha como objetivo determinar a viabilidade e acessibilidade de ferramentas de realidade virtual baseadas na web como uma plataforma de pesquisa inovadora para facilitar sessões de design participativas e remotas, metodologia que os autores justificam ser apropriada para ser utilizada como participantes vulneráveis, como base no trabalho de Bell [46].

Bhatia et al. [42] explicitam que o estudo foi aprovado pelo CEP da universidade e que, antes das sessões de design, tanto adolescentes participantes consentiram em participar e em serem gravados no *Zoom* como seus pais ou responsáveis foram informados da participação dos adolescentes. Não há referência a termos de consentimento e de assentimento ou referência a riscos, por exemplo, mas quem lê o artigo assume que foi cumprido a exigência, nos EUA, de concordância explícita de participantes com um termo de consentimento escrito [45].

*Algorithmic Folk Theories and Identity: How TikTok Users Co-Produce Knowledge of Identity and Engage in Algorithmic Resistance.* Karizat et al. [43] descrevem um estudo em que conduziram entrevistas com 15 usuários da rede TikTok nos EUA, explorando teorias sobre o algoritmo “For You” em relação a identidades pessoais e sociais. O algoritmo analisa o comportamento do usuário visando aumentar o engajamento na plataforma.

Os autores empregaram um serviço de recrutamento de participantes para pesquisa e pagaram a cada participante uma compensação de 20 dólares americanos pelo tempo dedicado à pesquisa – uma prática proibida no Brasil. Karizat et al. [43] explicitam que o projeto foi aprovado pelo CEP da universidade, que os recrutados concordaram com o termo de consentimento, e que a idade mínima para participar era 18 anos.

*COVID-19 highlights the issues facing blind and visually impaired people in accessing data on the web.* Siu et al. [44] relatam as experiências de pessoas com deficiência visual (BVI) ao acessar representações de dados disponíveis na web durante o período inicial da pandemia de COVID-19. Os dados foram obtidos por meio de um questionário com 127 respondentes e de entrevistas com 12 respondentes.

Siu et al. [44] explicitam que o estudo foi aprovado pelo CEP da universidade e que a idade mínima para participar era 18 anos. Ainda, registram que respondentes do

questionário podiam optar por participar de um sorteio para ganhar um dos vinte cartões-presente no valor de 15 dólares cada e não era necessário completar a pesquisa para participar – prática não é permitida no Brasil. O texto não explicita que participantes concordaram com o TCLE, o que é assumido pelo leitor sendo isso obrigatório nos EUA [45].

#### **4.8.2. Exemplos de projetos da área de Web e Multimídia do ICMC-USP**

Ilustramos os destaques de três trabalhos relativamente a suas referências à aprovação do projeto correspondente por um CEP. Assim como os trabalhos utilizados como exemplos nas Seções 4.7.4 e 4.7.5, estes trabalhos reportados por autores vinculados ao Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo (ICMC-USP).

*Once upon a time... you! A game to support health professionals in the treatment of children with chronic diseases, helping them to express their feelings.* No artigo, Verhalen et al. [47] reportam o jogo sério intitulado *Narrativa Invertida*, desenvolvido para apoiar o tratamento de crianças com doenças crônicas e graves. Utilizando uma abordagem de design de jogos terapêuticos, o jogo permite que as crianças criem narrativas a partir de elementos pré-disponíveis, como balões de fala, objetos e personagens. Os profissionais de saúde podem usar o jogo como recurso terapêutico para iniciar diálogos ou procedimentos. Avaliações empíricas mostraram melhorias na jogabilidade e acessibilidade.

Verhalen et al. [47] ressaltam que o trabalho teve aprovação de Comitê de Ética e incluem o número do parecer, que os voluntários consentiram em participar da pesquisa, e que os seus dados foram anonimizados. Estudantes e profissionais da Computação foram convidados a jogar o jogo e fornecer suas impressões. A amostra, selecionada por conveniência, teve sete participantes.

*Towards Design Guidelines for IoT Applications considering Elderly Users* Envolvendo grupos vulneráveis (pessoas idosas), a pesquisa resultou em um conjunto de diretrizes visando auxiliar pessoas designers e desenvolvedoras na construção de soluções de IoT, com foco em pessoas idosas e considerando requisitos como usabilidade e acessibilidade.

Rodrigues et al. [48] registram que pesquisa foi aprovada por um CEP da universidade e informam o número do protocolo de estudo. Além disso, ressaltam que a coleta de dados ocorreu com o consentimento dos participantes para usar suas respostas, e foi garantido o anonimato completo dos participantes. O projeto aprovado no CEP previa a coleta de dados com pessoas idosas, empresas, especialistas em IHC e estudantes. Este artigo em particular registra os resultados de entrevistas com 11 pessoas idosas.

*Pesquisa sobre Dark Patterns em protótipo de jogo digital com conteúdo extraído da DB-Pedia.* Rodrigues and Moreira [36] registram que o projeto aprovado por um CEP e incluem o número do CAAE, e que projeto prevê que participantes instalem um aplicativo Android™ que implementa um jogo educacional. O TCLE informa, entre outros, que o jogo poderia ser utilizado a qualquer momento dentro do período de teste de aproximadamente 15 dias, sem obrigatoriedade de jogar em períodos específicos ou continuar jogando. O TCLE também informa que os dados seriam coletados em português e incluem um formulário para informações demográficas (idade, gênero, país e escolaridade), opcional para os participantes; um código alfanumérico anonimizado; registros da

interação com elementos do jogo; um formulário opcional para opiniões sobre o jogo; e o encerramento do experimento. Merece destaque o cuidado de Rodrigues and Moreira [36] com a coleta 100% anonimizada. Considerando que o estudo demandava instalação do aplicativo no dispositivos das pessoas participantes, o código do aplicativo não foi disponibilizado na *Google Play Store* porque esta teria acesso a quem instalou o jogo, o que violaria o pressuposto de “O convite para participar da pesquisa não deve ser feito com a utilização de listas que permitam a identificação dos convidados nem a visualização dos seus dados de contato (e-mail, telefone, etc.) *por terceiros*” [49, item 2.1].

#### **4.8.3. Dois exemplos que dispensaram aprovação por CEP**

Em seu trabalho, Galvão et al. [50] registram o desenvolvimento do framework conceitual CERCO (Componentes, Elementos, Requisitos e Conceitos) e artefatos relacionados, como glossários e diretrizes de design, para auxiliar na criação de jogos empáticos, visando abordar temas complexos de forma interativa. Galvão et al. [50] destacam que, devido à revisão extensa da literatura e metodologias aplicadas no trabalho, o estudo não demandou investigações com usuários ou estudos de campo, o que dispensou a necessidade de solicitar aprovação para um CEP. Além disso, Galvão et al. [50] observam que o projeto em cujo escopo o trabalho foi realizado tinha aprovação ética para pesquisas com seres humanos.

Pimentel et al. [51] relatam a proposta e os resultados obtidos com o projeto Meninas Programadoras, que objetiva incentivar a participação feminina na área de computação oferecendo cursos gratuitos online de Introdução à Programação com Python para alunas do Ensino Médio e concluintes. O relato explicita:

*As atividades dispensam avaliação pela Comissão Nacional de Ética em Pesquisa por serem realizadas com o intuito exclusivamente de ensino sem finalidade de pesquisa científica (Resolução CONEP nº 510 de 07/04/2016, Artigo 1º, parágrafo único, inciso VIII). As informações são coletadas e armazenadas respeitando a LGPD, e os dados são apresentados sem possibilidade de identificação das participantes.*

#### **4.9. Considerações finais**

Neste texto, procuramos orientar pesquisadores da área de Sistemas Interativos Web e Multimídia relativamente ao impacto de suas pesquisas quando estas envolvem participantes humanos. Ao contextualizarmos fundamentos históricos, morais e éticos associados às pesquisas, mostramos a necessidade de legislação que oriente e proteja participantes e pesquisadores.

Além de apresentar a legislação pertinente, ilustramos e detalhamos o processo de proteção dos participantes e de pesquisadores por meio de submissão de projetos de pesquisa ao sistema CEP/Conep. Ao registrarmos exemplos de trabalhos reportados na literatura, exemplificamos resultados de pesquisa consistentes com os padrões éticos da atualidade.

---

---

## Referências

- [1] John Leslie King. Humans in computing: growing responsibilities for researchers. *Commun. ACM*, 58(3):31–33, feb 2015. ISSN 0001-0782. URL <https://doi.org/10.1145/2723675>.
- [2] Tanya Estes, Linda Mallory, and Edward Sobiesk. The value of a one semester exposure to the institutional review board process. *J. Comput. Sci. Coll.*, 31(6):63–69, jun 2016. ISSN 1937-4771.
- [3] Johanna Phelps-Hillen. Institutional review boards: Human subjects and their texts. In *Proc. 32nd ACM International Conf. on The Design of Communication CD-ROM*. ACM, 2014. URL <https://doi.org/10.1145/2666216.2666235>.
- [4] Casey Fiesler, Michael Zimmer, Nicholas Proferes, Sarah Gilbert, and Naiyan Jones. Remember the human: A systematic review of ethical considerations in reddit research. *Proc. ACM Hum.-Comput. Interact.*, 8(GROUP), feb 2024. URL <https://doi.org/10.1145/3633070>.
- [5] Luiz Paulo Carvalho, José Antonio Suzano, Flávia Maria Santoro, Jonice Oliveira, and Maria da Graça Pimentel. Ethics: What is the research scenario in the brazilian symposium webmedia? In *Proc. Brazilian Symposium on Multimedia and the Web*, page 1–10. ACM, 2022. URL <https://doi.org/10.1145/3539637.3557932>.
- [6] Patrícia F Amorim, Carolina Sacramento, Eliane Pinheiro Capra, Patricia Zamprognio Tavares, and Simone Bacellar Leal Ferreira. Submit or not my HCI research project to the ethics committee, that is the question. In *Proc. 18th Brazilian Symposium on Human Factors in Computing Systems*. ACM, 2019. URL <https://doi.org/10.1145/3357155.3358473>.
- [7] Luiz Paulo Carvalho, José Antonio Suzano, Roberto Pereira, Flávia Maria Santoro, and Jonice Oliveira. Ethics: What is the research scenario in the brazilian symposium ihc? In *Proc. XX Brazilian Symposium on Human Factors in Computing Systems*. ACM, 2021. URL <https://doi.org/10.1145/3472301.3484324>.
- [8] Luiz Paulo Carvalho, José Antonio Suzano, Monica Anastassiu, Flávia Maria Santoro, Jonice Oliveira, and João Carlos Gonçalves. Ethics: What is the research scenario in the brazilian symposium sbqs? In *Proc. XX Brazilian Symposium on Software Quality*. ACM, 2021. URL <https://doi.org/10.1145/3493244.3493249>.
- [9] Luiz Paulo Carvalho, Rosa Maria M. Da Costa, Flávia Maria Santoro, and Jonice Oliveira. How to carry out a brazilian research in computing considering ethical or moral aspects? In *Proc. XIX Brazilian Symposium on Information Systems*, page 151–158. ACM, 2023. URL <https://doi.org/10.1145/3592813.3592900>.
- [10] Conselho Nacional de Saúde. Resolução 196/1996: diretrizes e normas regulamentadoras de pesquisas envolvendo seres humanos. [https://bvsms.saude.gov.br/bvs/saudelegis/cns/1996/res0196\\_10\\_10\\_1996.html](https://bvsms.saude.gov.br/bvs/saudelegis/cns/1996/res0196_10_10_1996.html), Outubro 1996. Acessado 24/01/2024.
- [11] Conselho Nacional de Saúde. Resolução 466/2012: diretrizes e normas regulamentadoras de pesquisas envolvendo seres humanos. <https://conselho.saude.gov.br/resolucoes/2012/Reso466.pdf>, Dezembro 2012. Acessado 24/01/2024.
- [12] Gabriel Sausen Feil and Júlia Rocha Paz. O que é ética? quatro possibilidades conceituais na pesquisa em comunicação. *Intercom: Revista Brasileira de Ciências da Comunicação*, 46:e2023202, 2023. ISSN 1809-5844. URL <https://doi.org/10.1590/1809-58442023202pt>.
- [13] Yves de La Taille. *Moral e ética: dimensões intelectuais e afetivas*. Artmed Editora, 2007.
- [14] Clóvis Barros Filho. O que é moral? — clóvis de barros filho. YouTube, March 2014. URL <https://www.youtube.com/watch?v=Jsjn49FxlLc>. vídeo (2min33s).

- [15] Sociedade Brasileira de Computação (SBC). Código de Ética e Conduta Profissional. Resolução nº 02, de 21 de março de 2024, 2024. Disponível em: <https://www.sbc.org.br/documentos-da-sbc/summary/245-codigo-de-etica-sbc/1487-codigo-de-etica-e-conduta-profissional>.
- [16] Instituto Nacional de Câncer (INCA). Legislação pertinente sobre o uso de animais em pesquisa 2008-2021 / Comitê de Ética no Uso de Animais (CEUA) - INCA, 2022. URL <https://www.gov.br/inca/pt-br/assuntos/pesquisa/comites-e-comissoes/comite-de-etica-no-uso-de-animais-ceua-inca/legislacao-pertinente>. Acessado 24/03/2024.
- [17] Ministério da Saúde - Conselho Nacional de Saúde. Norma operacional 001/2023, 2013. URL [https://conselho.saude.gov.br/images/comissoes/conep/documentos/NORMAS-RESOLUCOES/Norma\\_Operacional\\_n\\_001-2013\\_Procedimento\\_Submissao\\_de\\_Projeto.pdf](https://conselho.saude.gov.br/images/comissoes/conep/documentos/NORMAS-RESOLUCOES/Norma_Operacional_n_001-2013_Procedimento_Submissao_de_Projeto.pdf).
- [18] Paul Weindling. *Victims and Survivors of Nazi Human Experiments: Science and Suffering in the Holocaust*. Bloomsbury Publishing, 2014.
- [19] Kendall Smith. Edward Jenner and the Small Pox Vaccine. *Frontiers in Immunology*, 2, 2011. ISSN 1664-3224. URL <https://www.frontiersin.org/articles/10.3389/fimmu.2011.00021>.
- [20] Science History Institute. Marie Skłodowska Curie. *Scientific Biographies*, 2023. URL <https://sciencehistory.org/education/scientific-biographies/marie-sklodowska-curie>. © 2023 Science History Institute.
- [21] Stephen Gibson. Milgram's Experiments on Obedience to Authority. *Oxford Research Encyclopedia of Psychology*, 06 2020. URL <https://oxfordre.com/psychology/view/10.1093/acrefore/9780190236557.001.0001/acrefore-9780190236557-e-511>.
- [22] Augustine Brannigan, Ian Nicholson, and Frances Cherry. Introduction to the special issue: Unplugging the milgram machine. *Theory & Psychology*, 25(5):551–563, 2015. URL <https://doi.org/10.1177/0959354315604408>.
- [23] Ian Nicholson. “Torture at Yale”: Experimental subjects, laboratory torment and the “rehabilitation” of Milgram’s “Obedience to Authority”. *Theory & Psychology*, 21(6):737–761, 2011. URL <https://doi.org/10.1177/0959354311420199>.
- [24] Jharna Mandal, Subrat Acharya, and Subhash Chandra Parija. Ethics in Human Research. *Tropical Parasitology*, 1(1):2–3, January 2011. URL <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3593469>.
- [25] Conselho Nacional de Saúde. Resolução 510/2016: dispõe sobre as normas aplicáveis a pesquisas em ciências humanas e sociais cujos procedimentos metodológicos envolvam a utilização de dados diretamente obtidos com os participantes ou de informações identificáveis ou que possam acarretar riscos maiores do que os existentes na vida cotidiana. [https://bvsms.saude.gov.br/bvs/saudelegis/cns/2016/res0510\\_07\\_04\\_2016.html](https://bvsms.saude.gov.br/bvs/saudelegis/cns/2016/res0510_07_04_2016.html), Abril 2016. Acessado 24/01/2024.
- [26] A. S. Vázquez. *Ética*. Civilização Brasileira, 39th edition, 2018.
- [27] Juliana O S Miranda, Deisy V Santos, Climene L de Camargo, Carlito L Nascimento Sobrinho, Darci O S Rosa, and Gleyce M S Souza. Construção e aplicação de um termo de assentimento: relato de experiência. *Texto & Contexto-Enfermagem*, 26:e2460016, 2017.
- [28] Cristiane S Varejão, Fátima H do Espírito Santo, and Maria N S de Ribeiro. The importance of applying the statement of assent to children and adolescents: a qualitative study. *Investigación y Educación en Enfermería*, 40(2), 2022.
- [29] Conselho Nacional de Saúde Secretaria-Executiva do Conselho Nacional de Saúde CO-

- NEP. Ofício circular nº 17/2022/conep/secns/ms. [https://conselho.saude.gov.br/images/Of%C3%ADcio\\_Circular\\_17\\_SEI\\_MS\\_-\\_25000.094016\\_2022\\_10.pdf](https://conselho.saude.gov.br/images/Of%C3%ADcio_Circular_17_SEI_MS_-_25000.094016_2022_10.pdf), Julho 2022. Acessado 24/01/2024.
- [30] Conselho Nacional de Saúde. Resolução nº 674, de 6 de maio de 2022. <https://conselho.saude.gov.br/resolucoes-cns/2469-resolucao-n-674-de-06-de-maio-de-2022>, Maio 2022. Acessado 24/01/2024.
- [31] ACM. ACM Publications Policy on Research Involving Human Participants and Subjects, 2021. URL <https://www.acm.org/publications/policies/research-involving-human-participants-and-subjects>.
- [32] John W Creswell and J David Creswell. *Research design: Qualitative, quantitative, and mixed methods approaches*. Sage publications, 2017.
- [33] Mirian Goldenberg. *A arte de pesquisar: como fazer pesquisa qualitativa em Ciências Sociais*. Editora Record, 2011.
- [34] Jonathan Lazar, Jinjuan Heidi Feng, and Harry Hochheiser. *Research methods in human-computer interaction*. Morgan Kaufmann, 2017.
- [35] Maria G Pimentel. Roteiro para Submissão Inicial de Projeto na Plataforma Brasil grande área “Ciência Exatas e da Terra”. [https://drive.google.com/file/d/11V7u\\_o-rFFGXXkDxtDVDwyhfTO9EWpuIm/view?usp=sharing](https://drive.google.com/file/d/11V7u_o-rFFGXXkDxtDVDwyhfTO9EWpuIm/view?usp=sharing), 2024.
- [36] Marcos Vinícius Figueiredo Silva Rodrigues and Dilvan de Abreu Moreira. Pesquisa sobre Dark Patterns em protótipo de jogo digital com conteúdo extraído da DB-Pedia (CAAE: 69471223.3.0000.5504)). TCLE: [https://drive.google.com/drive/folders/1AbzUcAlz7dXr03dxMdV9Q\\_UO7OwFSCc9](https://drive.google.com/drive/folders/1AbzUcAlz7dXr03dxMdV9Q_UO7OwFSCc9) Acessado 20/04/2024, 2024.
- [37] Aline E C Verhalen and Kamila R H Rodrigues. Design of therapeutic digital games that support dialogue with children about death. In *Proc. XXII Brazilian Symposium on Human Factors in Computing Systems*. ACM, 2024. URL <https://doi.org/10.1145/3638067.3638112>.
- [38] Bruno Cunha, Keli Rodrigues, Ibrahim Zaine, Emerson da Silva, Cleber Viel, and Maria Pimentel. Experience sampling and programmed intervention method and system for planning, authoring, and deploying mobile health interventions: Design and case reports. *JMIR*, 23(7): e24278, 2021. doi: 10.2196/24278. URL <https://www.jmir.org/2021/7/e24278>.
- [39] 2024. URL <https://dl.acm.org/action/doSearch?fillQuickSearch=false&target=advanced&expand=dl&AllField=Abstract%3A%28user%29+AND+Fulltext%3A%28irb%29+AND+Abstract%3A%28interaction%29+AND+Abstract%3A%28media%29&startPage=&EpubDate=%5B20190418%20TO%20202404182359%5D&queryID=58/6785572614>. Recursos da ACM Digital Library com *media*, *user*, e *interaction* no *abstract*, *IRB* (*Institutional Review Board*) no corpo do texto, nos últimos cinco anos. Acesso 2024-04-18.
- [40] 2024. URL <https://dl.acm.org/action/doSearch?fillQuickSearch=false&target=advanced&expand=dl&AllField=Abstract%3A%28user%29+AND+Fulltext%3A%28irb%29+AND+Abstract%3A%28interaction%29+AND+Abstract%3A%28web%29&startPage=&EpubDate=%5B20190418%20TO%20202404182359%5D&queryID=26/6785646287>. Recursos da ACM Digital Library com *web*, *user*, e *interaction* no *abstract*, a *IRB* (*Institutional Review Board*) no corpo do texto, nos últimos cinco anos. Acesso 2024-04-18.
- [41] A Razi, A Alsoubai, S Kim, N Naher, S Ali, G Stringhini, M De Choudhury, and P J. Wisniewski. Instagram Data Donation: A Case Study on Collecting Ecologically Valid Social Media Data for the Purpose of Adolescent Online Risk Detection. In *Extended Abstracts*

- 2022 *ACM CHI*. ACM, 2022. URL <https://doi.org/10.1145/3491101.3503569>.
- [42] Simran Bhatia, Elin A. Björling, and Tanya Budhiraja. Exploring Web-Based VR for Participatory Robot Design. In *Companion 2021 ACM/IEEE Intl Conf. on Human-Robot Interaction*, page 109–112. ACM, 2021. URL <https://doi.org/10.1145/3434074.3447139>.
- [43] Nadia Karizat, Dan Delmonaco, Motahhare Eslami, and Nazanin Andalibi. Algorithmic Folk Theories and Identity: How TikTok Users Co-Produce Knowledge of Identity and Engage in Algorithmic Resistance. *Proc. ACM Hum.-Comput. Interact.*, 5(CSCW2), oct 2021. URL <https://doi.org/10.1145/3476046>.
- [44] Alexa F. Siu, Danyang Fan, Gene S-H Kim, Hrishikesh V. Rao, Xavier Vazquez, Sile O’Modhrain, and Sean Follmer. COVID-19 highlights the issues facing blind and visually impaired people in accessing data on the web. In *Proc. 18th International Web for All Conference*. ACM, 2021. URL <https://doi.org/10.1145/3430263.3452432>.
- [45] Title 21–Food and Drugs: Chapter I–Food and Drug Administration, Department of Health and Human Services, Subchapter A - General, Part 50 - Protection of Human Subjects, Subpart B - Informed Consent of Human Subjects. <https://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfcfr/CFRSearch.cfm?CFRPart=50&showFR=1&subpartNode=21:1.0.1.1.20.2>, 2023. Acessado em 21 de abril de 2024.
- [46] Beth T. Bell. Understanding adolescents. In Linda Little, Daniel Fitton, Beth T. Bell, and Nicola Toth, editors, *Perspectives on HCI Research with Teenagers*, pages 11–27. Springer International Publishing, 2016. URL [https://doi.org/10.1007/978-3-319-33450-9\\_2](https://doi.org/10.1007/978-3-319-33450-9_2).
- [47] Aline E C Verhalen, Tiago N Silva, and Kamila R H Rodrigues. Once upon a time... you! a game to support health professionals in the treatment of children with chronic diseases, helping them to express their feelings. In *Proceedings of the XXII Brazilian Symposium on Human Factors in Computing Systems*, IHC ’23, 2023. URL <https://doi.org/10.1145/3638067.3638129>.
- [48] Sandra S Rodrigues, Renata Pontin Fortes, and Kamila R H Rodrigues. Towards design guidelines for iot applications considering elderly users. In *Proc. XXII Brazilian Symp. Human Factors in Computing Systems*, 2023. URL <https://doi.org/10.1145/3638067.3638139>.
- [49] Conselho Nacional de Saúde Secretaria-Executiva do Conselho Nacional de Saúde CONEP. Carta circular nº 1/2021-conep/secns/ms: Orientações para procedimentos em pesquisas com qualquer etapa em ambiente virtual. [https://conselho.saude.gov.br/images/comissoes/conep/documentos/CARTAS/Carta\\_Circular\\_01.2021.pdf](https://conselho.saude.gov.br/images/comissoes/conep/documentos/CARTAS/Carta_Circular_01.2021.pdf), Março 2021. Acessado 20/04/2024.
- [50] Vinícius Ferreira Galvão, Cristiano Maciel, Eunice Pereira Dos Santos Nunes, and Kamila Rios Da Hora Rodrigues. A framework to support the development of empathic games. In *Proc. XXII Brazilian Symposium on Human Factors in Computing Systems*. ACM, 2024. URL <https://doi.org/10.1145/3638067.3638104>.
- [51] Maria G Pimentel, Juliana Eusebio, Rudinei Goularte, Uthant Leite, and Helen Picoli. Meninas Programadoras: Promovendo o Engajamento Feminino em Computação via Cursos Curtos Online de Programação. In *Anais Estendidos do XXIX WebMedia*, pages 107–110. SBC, 2023. URL [https://sol.sbc.org.br/index.php/webmedia\\_estendido/article/view/25685](https://sol.sbc.org.br/index.php/webmedia_estendido/article/view/25685).



Apêndice A: exemplo de TCLE  
*ilustra a demanda por participação remota (online) e colaboração entre participantes*

---

**TERMO DE CONSENTIMENTO LIVRE E ESCLARECIDO - TCLE**

Você está sendo convidado(a) a participar como voluntária(o) da pesquisa intitulada “.....”. Essa pesquisa tem como objetivo principal realizar avaliações quantitativas e qualitativas da percepção dos usuários em relação aos recursos .... disponibilizados no sistema ..... Para tanto, será solicitado que .... participantes *..tarefas a realizar..* [**de modo colaborativo**] utilizando o sistema de modo **remoto** .....

Nos comprometemos a seguir a Resolução CNS 466/12, relacionada à Pesquisa com Seres Humanos, respeitando o seu direito garantido de:

1. Ter liberdade de participar ou deixar de participar do estudo, sem que isso lhe traga algum prejuízo ou risco, podendo interromper sua participação a qualquer momento caso se sinta incomodado(a) com a mesma;
2. Manter o seu nome em sigilo, sendo que o que disser não lhe resultará em qualquer dano à sua integridade e privacidade;
3. Responder às questões levantadas por meio de formulários ao final de sua participação na sessão **remota** de avaliação dos recursos ...., sabendo que solicita-se que todos utilizem um local reservado a fim de preservar a privacidade de todos durante a sessão;
4. Garantia de receber uma resposta a alguma dúvida durante ou após sua participação;
5. Direito ao ressarcimento de despesas relativas à participação na pesquisa;
6. Direito à indenização, caso perceba prejuízo pela participação na pesquisa.

Sendo sua participação voluntária, você não receberá benefício financeiro. A pesquisa será feita de forma **remota** e você utilizará o seu ambiente usual de computador com acesso à Internet. Solicitamos que você utilize um local reservado a fim de preservar a privacidade de todos durante a sessão **remota**. Esclarecemos, ainda, que o tempo total estimado para a realização das atividades e responder os questionários é de ... minutos. Devido à natureza **online** do estudo, pode haver interrupções no acesso aos sistemas e plataformas utilizados, o que pode afetar sua participação e, nesse, caso consultaremos sua disponibilidade de participação em nova sessão.

Garantimos que você pode ser ressarcido pelas despesas relacionadas à sua participação no que tange custos adicionais de Internet, energia elétrica ou outra despesa decorrente do uso do computador e de conexão à Internet durante o período da pesquisa. O reembolso será limitado a R\$... por sessão **remota**. Para solicitar reembolso, envie e-mail com assunto “solicitado reembolso” para a pessoa pesquisadora responsável após a realização da sessão remota, informando data e hora de participação, e o identificador da sessão.

Neste estudo, utilizaremos os seguintes recursos: e-mail para convite, preparação e esclarecimentos; *Google Meet* para esclarecimentos adicionais e para acompanhamento de sua sessão ...; e o sistema .... para realização da tarefa realizada durante a sessão. Antes da sessão, você será convidado(a) a responder um questionário seu perfil incluindo sua área de atuação e sua experiência com o ..... Durante a sessão, após assistir um vídeo tutorial de aproximadamente .... minutos, sua tarefa será ....., [tarefa essa realizada de modo **colaborativo** com outro participante da pesquisa]. Ao término da sessão você será convidado a responder *três questionários* (contendo 12, 12 e 3 perguntas, respectivamente), que serão aplicados por meio do *Google Forms*.

Em relação à interação realizada por meio do *Google Meet*, serão registrados os dados relacionados

à duração da sessão, bem como os horários de entrada e saída dos participantes, e não serão realizadas gravações em vídeo da sessão. Os dados da interação dos participantes com o sistema, durante a realização da tarefa, serão registrados automaticamente pelo sistema e armazenados de forma segura e anônima, garantindo a privacidade dos participantes.

Devido ao fato das tarefas serem realizadas de forma colaborativa, formaremos grupos de dois participantes levando em consideração sua disponibilidade de agenda, área de interesse e existência ou não de experiência prévia com o sistema ..... Caso não se sinta confortável em trabalhar com a sua dupla, você poderá requisitar mudança. Solicitamos que, durante as sessões com outros participantes, você não revele nenhum dado pessoal e nem solicite dados pessoais dos outros participantes.

Por sua participação nesta pesquisa, você poderá adquirir conhecimento, desenvolver habilidades e refletir sobre experiência e práticas em sua área de atuação e ..... Os resultados dessa pesquisa servirão de base para aprimorar os recursos .... do sistema..., sendo assim, outras pessoas também poderão se beneficiar no futuro.

**Riscos:** Embora sejam tomadas medidas para garantir a segurança e a privacidade dos participantes, existem alguns riscos inerentes à natureza do estudo e do ambiente online.

- **Risco social:** Interações indesejadas entre participantes podem ocorrer, por isso as sessões serão compostas por indivíduos que já se conhecem. Os participantes têm o direito de solicitar a troca de dupla ou interromper sua participação;
- **Risco emocional:** Devido a sua natureza colaborativa, o estudo pode despertar emoções desconfortáveis durante as interações entre os participantes. Os participantes têm a opção de interromper sua participação a qualquer momento, e manterão um cópia deste termo que contém dados dos contatos para acompanhamento pós-estudo;
- **Risco de interrupção do serviço:** Possíveis interrupções no acesso aos sistemas e plataformas online podem ocorrer, o que pode afetar a participação. Será consultada a disponibilidade dos participantes para uma nova sessão, se necessário;
- **Risco de exigência de tempo:** O experimento requer um tempo significativo dos participantes dado que a sessão terá duração de aproximadamente ... minutos;
- **Risco econômico:** Não há custo direto para os participantes, mas a participação exige o uso de recursos pessoais, como tempo e acesso a computador e Internet;
- **Risco de confidencialidade:** Existe um risco mínimo de comprometimento da confidencialidade dos dados pessoais caso sejam divulgados durante a interação com outros participantes. Recomenda-se evitar o compartilhamento de dados pessoais e solicitação de informações de outros participantes. São utilizadas medidas de segurança, como o protocolo SSH, para proteger a privacidade e garantir confidencialidade;
- **Risco de exposição à radiação:** Dispositivos eletrônicos emitem radiação eletromagnética considerada segura em níveis típicos de uso. O experimento envolve o uso desses dispositivos, mas a exposição à radiação será limitada ao uso normal e dentro das normas de segurança estabelecidas para os equipamentos eletrônicos.

**Benefícios:** Embora os benefícios possam variar de acordo com cada indivíduo, potenciais benefícios para os participantes do estudo incluem:

- **Relevância e aplicabilidade:** Ao realizar uma tarefa em sua área de interesse, o participante pode perceber que suas habilidades e conhecimentos são aplicáveis e relevantes. Isso proporciona uma sensação de valorização e contribuição, aumentando a satisfação pessoal e o sentimento de que sua participação no estudo é significativa;
- **Aquisição de conhecimento e aprendizado:** Ao se envolver em uma tarefa relacionada à sua

área de interesse, o participante tem a oportunidade de expandir seu conhecimento e aprofundar sua compreensão sobre um tema específico. Isso contribui para seu desenvolvimento pessoal e profissional, beneficiando-o além da participação no estudo;

- **Aprendizado e desenvolvimento de habilidades:** Durante a participação, os participantes podem adquirir novos conhecimentos e habilidades relacionados ao uso do sistema e à criação de programas de intervenção, o que pode ser útil tanto em contextos profissionais quanto pessoais;
- **Reflexão sobre experiências e práticas:** O envolvimento no estudo pode permitir que os participantes reflitam sobre suas próprias experiências e práticas de autoria de programas de intervenção, promovendo uma maior compreensão e aprimoramento das mesmas;
- **Contribuição para a pesquisa:** Ao participar do estudo, os participantes têm a oportunidade de contribuir para o avanço do conhecimento científico na área...;
- **Possibilidade de *feedback*:** Os participantes terão a oportunidade de fornecer *feedback* sobre o sistema e as funcionalidades ..., contribuindo para o aprimoramento e desenvolvimento de futuras versões do sistema....

As informações desta pesquisa serão confidenciais e só poderão ser divulgadas em eventos ou publicações sem a identificação dos participantes, a não ser entre os responsáveis pelo estudo, sendo assegurado o sigilo sobre sua participação.

Este Termo de Consentimento é emitido em formato digital (PDF) e assinado digitalmente pela pessoa pesquisadora responsável. Solicitamos que você inclua suas assinatura digital e nos devolva a versão assinada por você e pela pessoa pesquisadora.

Para obter mais esclarecimentos ou informações sobre o estudo e sua participação, você poderá entrar em contato a qualquer momento com a pessoa pesquisadora responsável:

- **Nome completo**
- **Contato: e-mail:** . . . .  
**telefone:** ( . . ) . . .
- **Endereço:** . . . .  
Instituto . . . .  
Universidade . . . .  
*Telefone:* ( . . ) . . .

Além disso, para obter mais esclarecimentos ou informações sobre o estudo e sua participação, você poderá entrar em contato a qualquer momento com o Comitê de Ética em Pesquisa:

- **Comitê de Ética em Pesquisa NOME DO COMITE**
- **endereço completo ....**
- **horário de atendimento ao público.....**

### **Autorização**

Eu, \_\_\_\_\_, após a leitura ou a escuta da leitura deste documento e de ter tido a oportunidade de conversar com a pessoa pesquisadora responsável para esclarecer todas as minhas dúvidas, estou suficientemente esclarecido e informado sobre minha participação nesta pesquisa, tendo ficado claro que minha participação é voluntária e que posso retirar este consentimento a qualquer momento sem penalidades ou perda de qualquer benefício. Estou ciente também dos objetivos da pesquisa, dos procedimentos dos quais participarei, dos possíveis danos ou riscos deles provenientes e da garantia de

confidencialidade. Diante do exposto e de espontânea vontade, expresso minha concordância em participar deste estudo e assino digitalmente este termo, já previamente assinado digitalmente pela pessoa pesquisadora responsável, sendo que guardarei uma cópia com as duas assinaturas para meu registro e encaminharei uma cópia para a pessoa pesquisadora responsável.

---

Assinatura do(a) participante

Assinatura da Pessoa Pesquisadora Responsável  
nome da pessoa pesquisadora...  
nome da instituição...



# WebMedia2023

XXIX Simpósio Brasileiro de  
Sistemas Multimídia e Web

REALIZAÇÃO



ORGANIZAÇÃO



COOPERAÇÃO



PATROCÍNIO

