MINICURSOS

DO XVI ENCONTRO UNIFICADO DE COMPUTAÇÃO DO PIAUÍ (ENUCOMPI) E SEMANA NACIONAL DE CIÊNCIA E TECNOLOGIA (SNCT)

ENUCOMPI
ENCONTRO UNIFICADO DE COMPUTAÇÃO DO PIAUÍ

SBC
Sociedade Brasileira de Computação

20ª SNCT

MINICURSOS DO XVI ENCONTRO UNIFICADO DE COMPUTAÇÃO DO PIAUÍ (ENUCOMPI) E SEMANA NACIONAL DE CIÊNCIA E TECNOLOGIA (SNCT 2023)

19 e 20 de outubro de 2023, Piripiri - Piauí

Editora

Sociedade Brasileira de Computação

Organizadores

Mayllon Veras da Silva (IFPI)
Alcemir Rodrigues Santos (UESPI)
Rodrigo Augusto Rocha Souza Baluz (UESPI)

Realização

Instituto Federal de Educação, Ciência e Tecnologia do Piauí (IFPI)
Universidade Estadual do Piauí (UESPI)

Dados Internacionais de Catalogação na Publicação (CIP)

E56 Encontro Unificado de Computação do Piauí e Semana Nacional de Ciência e Tecnologia (23. : 19 – 20 outubro 2023 : Piripiri)
Minicursos do ENUCOMPI e SNCT 2023 [recurso eletrônico] / organização: Mayllon Veras da Silva ; Alcemir Rodrigues Santos ; Rodrigo Augusto Rocha Souza Baluz. Dados eletrônicos. – Porto Alegre: Sociedade Brasileira de Computação, 2023.
97 p. : il. : PDF ; 3.95 MB

Modo de acesso: World Wide Web.
Inclui bibliografia
ISBN 978-85-7669-559-2 (e-book)

1. Computação – Brasil – Evento. 2. Ciência e Tecnologia. 3. Navegação Web. 4. Inteligência Artificial. 5. Navegação Web. 6. Projetos de Softwares. I. Silva, Mayllon Veras da. II. Santos, Alcemir Rodrigues. III. Baluz, Rodrigo Augusto Rocha Souza. IV. Sociedade Brasileira de Computação. V. Título.

CDU 004(063)

Ficha catalográfica elaborada por Annie Casali – CRB-10/2339

Biblioteca Digital da SBC – SBC OpenLib

Índices para catálogo sistemático:

1. Ciência e tecnologia dos computadores : Informática – Publicação de conferências, congressos e simpósios etc. ... 004(063)

Capa (Edição):

José Maria Santos da Rocha Junior, IFPI

Editoração:
Antonio Santos de Sousa, IFPI, Mayllon Veras da Silva, IFPI

Organização do ENUCOMPI / SNCT 2023

Coordenadores Gerais

Mayllon Veras da Silva, IFPI

Alcemir Rodrigues Santos, UESPI

Coordenadores da Trilha de Minicursos

Antonio Santos de Sousa, IFPI

Realização

Apoio Institucional

Organizações de apoio

Empresas parceiras

Apresentação

O **Encontro Unificado de Computação do Piauí (ENUCOMPI)** se destaca no cenário tecnológico do Nordeste. Realizado em conjunto com a **Semana Nacional de Ciência e Tecnologia (SNCT 2023)**, o ENUCOMPI deste ano trilhou um caminho inovador com o tema "Ciência e Tecnologia na Geração do Conhecimento", refletindo seu compromisso com a vanguarda da inovação e a disseminação do saber. Com uma história rica de 16 anos, o ENUCOMPI, que teve suas origens na cidade de Parnaíba, litoral do Piauí, é fruto da colaboração de professores e pesquisadores das instituições de ensino de Computação. Reconhecido como um dos eventos mais influentes de computação na região, o ENUCOMPI ganhou especial notoriedade nos estados do Maranhão, Piauí e Ceará. Este ano, a cidade de Piripiri foi a anfitriã do evento, realizado nas instalações do Instituto Federal de Educação, Ciência e Tecnologia do Piauí (IFPI). ENUCOMPI 2023 se dedicou a promover a inovação tecnológica sustentável e a acelerar o empreendedorismo na área da informática e computação. Esta edição foi marcada por uma agenda diversificada, incluindo palestras, minicursos, apresentações de artigos e o salão de protótipos, onde projetos inovadores da região que utilizam tecnologias de ponta foram exibidos. Cada atividade foi planejada para permitir que os participantes mergulhassem nas mais recentes tendências da computação e se engajassem em um intercâmbio de experiências profundamente enriquecedor. Mais do que um evento, o ENUCOMPI é um catalisador para o networking, estabelecendo um ambiente propício para que estudantes, pesquisadores, profissionais e entusiastas da computação se encontrem, troquem ideias e formem colaborações estratégicas para projetos futuros.

Sinopse

O Livro de Minicursos ENUCOMPI/SNCT 2023 explora uma série de tópicos relevantes na área de tecnologia e programação. No primeiro capítulo, intitulado "Desenvolvimento de Extensão para o Chrome com Persistência em Nuvem Utilizando Firebase", os autores apresentam a criação de extensões para navegador, utilizando HTML, CSS, JavaScript e o Cloud Firestore do Google. Avançando para o segundo capítulo, "Explorando a Explicabilidade da Inteligência Artificial - Técnicas para Compreender e Interpretar Modelos de Aprendizado de Máquina", discute-se a importância da transparência e compreensibilidade dos modelos de IA. No terceiro capítulo, "Desenvolvimento de Jogos 2D de Plataforma: Explorando Unity e Chat GPT para Criação de Códigos Dinâmicos", é apresentada uma metodologia para o desenvolvimento de jogos, combinando a plataforma Unity com a tecnologia de inteligência artificial Chat GPT, focando na criação de jogos de plataforma com códigos dinâmicos. Por fim, no quarto capítulo, "Git e GitHub: Desenvolvendo Habilidades Essenciais para Colaboração e Controle de Versões", os autores delineiam um percurso de aprendizado sobre estas ferramentas, abordando desde os conceitos básicos até suas aplicações práticas no controle de versões e na colaboração em projetos de software.

Comitês

Comitês de Organização

ORGANIZAÇÃO GERAL

Mayllon Veras da Silva (IFPI/Piripiri)
Alcemir Rodrigues Santos (UESPI/Piripiri)

COMISSÃO DE MINICURSOS

Antonio dos Santos Souza (IFPI/Parnaíba)
Denylson Melo Pereira (IFPI/Parnaíba)
Anderson dos Reis Barros (IFPI/Pedro II)

COMISSÃO DE PALESTRAS

Alcemir Rodrigues Santos (UESPI/Piripiri)
Harilton da Silva Araújo (UESPI/Piripiri)

COMISSÃO DAS SEÇÕES TÉCNICAS DE ARTIGOS

Iallen Gábio de Sousa Santos (IFPI/Piripiri)
Ricardo Sekeff Moura Budaruiche (IFPI/Piripiri)
Manoel Gonçalves da Silva Neto (IFPI/Pedro II)

COMISSÃO DE INSCRIÇÕES E CERTIFICAÇÃO

Thiago Abreu de Moura (IFPI/Pedro II)

COMISSÃO DE PATROCÍNIO

Athânio de Souza Silveira(IFPI/Parnaíba)

COMISSÃO DE DIVULGAÇÃO, CRIAÇÃO E ARTE

Cléber Araújo da Silva (IFPI/Pedro II)

Sumário

Capítulo 1

Desenvolvimento de extensão para o Chrome com persistência em nuvem utilizando Firebase

Letícia D. N. Ximenes, Marcos Antônio B. P. Tenório de Oliveira, Francisco Kellviny Cruz Feitosa e Mayllon Veras da Silva

Abstract

In this chapter, we will cover the fundamental concepts of HTML, CSS, and JavaScript, followed by an introduction to cloud data persistence. The chosen tool for this purpose is Visual Studio Code, and we will utilize Cloud Firestore, a free service provided by Google. The goal is to create a simple extension for the Chrome browser, aiming to assist in the initial learning of web development.

Resumo

Neste capítulo, serão abordados os conceitos básicos de HTML, CSS e JavaScript, seguidos pela introdução à persistência de dados em nuvem. A ferramenta escolhida para este propósito é o Visual Studio Code, e utilizaremos o Cloud Firestore, um serviço gratuito fornecido pelo Google. O objetivo é criar uma extensão simples para o navegador Chrome, visando auxiliar no aprendizado inicial do desenvolvimento web.

1.1. Introdução

Neste capítulo, o principal objetivo é fornecer uma introdução acessível e prática às tecnologias fundamentais do desenvolvimento web. Explorando os conceitos básicos do HTML, CSS e JavaScript, que são as linguagens essenciais para construir páginas da web.

Além disso, o armazenamento de informações se dará por meio da nuvem utilizando o Cloud Firestore, um serviço gratuito oferecido pelo Google. Este que permite guardar e recuperar dados online, o que é de muita utilidade em projetos web.

Para uma melhor organização, o capítulo será dividido em três partes principais. Contendo cada uma delas os tópicos de HTML, CSS, JavaScript, o banco de dados em núvem Cloud Firestore e os passos necessários para o desenvolvimento de uma extensão para o navegador Chrome.

1.1.1. Construir uma base sólida

As primeiras páginas desse capítulo se dão à exploração dos alicerces fundamentais do **HTML, CSS e JavaScript**. Será fornecida uma base sólida para a construção de páginas web, estabelecendo a base de conhecimento para seu desenvolvimento. À medida que se adquire esses fundamentos, torna-se possível uma progressão de conhecimento web que é fundamental para o desenvolvimento de aplicações de maior complexidade.

Este conhecimento não é somente uma etapa inicial, como também uma preparação para a criação de uma extensão para o navegador Chrome. O início deste capítulo oferece a base essencial que servirá como um guia ao desenvolvimento de extensões e à expansão de habilidades na programação em geral.

1.1.2. Explorar o banco de dados em nuvem

Após estabelecida as bases sólidas no desenvolvimento web, é necessário um entendimento do armazenamento de dados na nuvem. Nesta parte, será conduzida uma análise do banco de dados em nuvem que servirá como diferencial de uma extensão para o navegador Chrome. A ferramenta escolhida foi o Cloud Firestore, uma solução oferecida pelo Google.

Aqui será detalhado como essa ferramenta permite armazenar e acessar dados de forma simples e intuitiva, tornando-a a escolha ideal para a introdução ao armazenamento de dados na nuvem. Começando com uma visão geral do Cloud Firestore, abordando seus principais conceitos e recursos. Em seguida, será fornecido o conhecimento de como criar e gerenciar bancos de dados, armazenar informações e acessá-las. Após a edificação base do assunto, será possível atrelá-lo ao desenvolvimento de uma extensão.

1.1.3. Entender o que é uma extensão e seu processo de desenvolvimente

Para concluir este capítulo, depois de abordados em detalhes os tópicos essenciais, será desenvolvido o entendimento das extensões. Desde o que exatamente é uma extensão, como funciona, o seu desenvolvimento específico para o navegador Chrome e as maneiras pelas quais o Cloud Firestore pode ser integrado a esse processo.

Além disso, será oferecido um exemplo de extensão que será construído, proporcionando uma visão concreta e tangível dos conceitos discutidos anteriormente. Este fim de capítulo servirá como um guia abrangente para a criação de extensões para o navegador Chrome, destacando algumas estratégias de integração com o Cloud Firestore.

1.2. HTML: Introdução e Conceito

Esta seção concentra-se nos princípios fundamentais do HTML (Hypertext Markup Language). O HTML é a principal linguagem usada para criar páginas da web e serve como a base essencial para qualquer desenvolvedor. Este conhecimento é fundamental, pois será aplicado posteriormente na criação de uma extensão simples para o navegador Chrome.

O HTML é uma linguagem de marcação que estrutura o conteúdo de uma página da web, fornecendo a estrutura básica para todas as páginas na internet. Com o HTML, você pode definir a estrutura de uma página, criar links, inserir imagens e muito mais. Além disso, todo documento HTML segue uma estrutura padrão, como demonstrado no

código abaixo.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
     initial-scale=1.0">
  <title>Document</title>
</head>
<body>

</body>
</html>
```

Listing 1.1. Estrutura básica HTML

1.2.1. HTML: <!DOCTYPE html>

O termo <DOCTYPE html> é uma instrução fundamental no HTML que especifica a versão e o tipo de documento que está sendo usado em uma página da web.

A tag <!DOCTYPE html> é uma forma de declarar que o documento é um documento HTML. Ela informa ao navegador que o conteúdo da página deve ser interpretado como um documento HTML.

O <!DOCTYPE html> não especifica uma versão específica do HTML, como o HTML5. Em vez disso, ele indica ao navegador que a página deve ser tratada de acordo com os padrões mais recentes e compatíveis com HTML.

A declaração <!DOCTYPE html> também desempenha um papel crucial na determinação do modo de renderização do navegador. Os navegadores têm modos diferentes de renderização para lidar com páginas mais antigas e com páginas compatíveis com os padrões modernos. O <!DOCTYPE html> instrui o navegador a usar o modo de renderização mais recente e compatível.

O uso do <!DOCTYPE html> é altamente recomendado em todos os documentos HTML. Ele ajuda a garantir que sua página seja renderizada corretamente nos navegadores modernos e seja compatível com os padrões atuais da web.

1.2.2. HTML: Tags de construção web

As tags HTML são as bases fundamentais da web. Elas são os elementos mais básicos de uma página web, pois tudo é desenvolvido a partir delas, incluindo os caracteres apresentados, informações de vídeo e imagem, ou campos de dados dos quais serão retiradas informações necessárias, que podem ou não ser usadas posteriormente.

Em resumo, elas definem a estrutura e o conteúdo de uma página. Elas são cercadas por colchetes angulares, como por exemplo, ’<tag>’, e geralmente vêm em pares, consistindo em uma tag de abertura e outra de fechamento com uma barra entre o primeiro colchete e a tag ’</tag>’, como a tag ’<h1>’, que tem a seguinte estrutura ’<h1>’

Título '</h1>'. Também existem aquelas que não necessitam de fechamento, como a tag '<input>'.

Tabela 1.1. Tags exemplo, que também serão utilizadas na extensão

Tags	Funcionalidade
<html>	**É o elemento raiz de uma página HTML. Ela engloba todo o conteúdo da página.**
<head>	**É usada para incluir informações sobre o documento que não são visíveis para os visitantes da página, como metadados, folhas de estilo CSS e meta tags.**
<body>	**Contém o conteúdo visível da página da web, incluindo texto, imagens, links e outros elementos que os visitantes veem e interagem.**
<button>	**Cria um botão clicável que pode ser usado para acionar ações ou eventos em uma página web.**
<iframe>	**Permite incorporar uma janela de navegador aninhada em uma página, útil para incorporar conteúdo externo, como mapas ou vídeos.**
<script>	**É usada para incluir código JavaScript diretamente em uma página, permitindo interatividade e funcionalidade dinâmica.**
<link>	**É usada para vincular folhas de estilo externas (CSS) a uma página, definindo a aparência e o layout do conteúdo.**
<input>	**Cria campos de entrada, como caixas de texto, botões de opção ou caixas de seleção, que permitem aos usuários inserir dados em um formulário.**
	É usada para criar listas não ordenadas, com itens listados com marcadores, como pontos ou símbolos.
	É usada dentro de elementos ou para definir itens de lista, como elementos de uma lista de itens.
<a>	**Cria links para outras páginas ou recursos, permitindo aos usuários navegar para diferentes partes do site ou para outros sites.**

1.2.2.1. HTML: Atributos, Classes e Ids

Atributos de tag em HTML são informações adicionais que podem ser incluídas dentro das tags HTML para fornecer detalhes ou instruções específicas sobre como um elemento deve ser processado ou exibido em um navegador da web.

Sintaxe: Os atributos de tag são especificados dentro das tags HTML e geralmente têm a seguinte sintaxe: nome-do-atributo = "valor". O nome do atributo é sempre colocado dentro do nome da tag, seguido por um sinal de igual (=) e, em seguida, o valor do atributo, que é colocado entre aspas duplas (") ou aspas simples (').

Validação e Acessibilidade: A escolha e o uso adequados de atributos de tag são importantes para garantir que seu código HTML seja válido e acessível. A atribuição de valores significativos a atributos, como alt em imagens e aria-label em elementos acessíveis, melhora a experiência do usuário.

Os atributos de tag em HTML desempenham um papel essencial na estruturação e no funcionamento das páginas da web, permitindo que os desenvolvedores controlem como os elementos são exibidos e interagem com os usuários. Também vale-se ressaltar que existem diversos tipos de atributos para HTML, porém alguns necessitam de uma tag específica para serem utilizados. Na seguinte tabela há um detalhamento desses atributos:

Além disso, é fundamental destacar que as Classes e IDs são atributos HTML

Tabela 1.2. Tipos de atributos HTML

Tipos de Atributos	Explicação
Atributos Globais	**Existem atributos que são considerados "globais"e podem ser usados em praticamente qualquer elemento HTML. Alguns exemplos de atributos globais incluem id, class, style, title, lang, data-*, entre outros.**
Atributos Específicos de Elementos	**Cada elemento HTML pode ter atributos próprios específicos que controlam seu comportamento e aparência. Por exemplo, a tag <img> tem o atributo src para especificar a fonte da imagem e o atributo alt para fornecer um texto alternativo.**
Atributos de Hiperlink	**As tags de hiperlink, como <a> e <area>, têm atributos especiais, como href para especificar o destino do link e target para determinar onde o link deve ser aberto (por exemplo, em uma nova janela ou na mesma janela).**
Atributos de Formulário	**Os formulários HTML, criados com as tags <form>, <input>, <select>, entre outras, utilizam atributos como name, type, value, placeholder e outros para coletar informações do usuário e enviar dados para o servidor.**
Atributos Booleanos	**Alguns atributos têm valores booleanos, o que significa que sua presença indica verdadeiro e sua ausência indica falso. Por exemplo, o atributo checked em uma caixa de seleção indica se ela está marcada.**

essenciais para a identificação e seleção de elementos específicos, principalmente com o propósito de estilização e manipulação por meio de CSS e/ou JavaScript.

O que são Classes: Classes são atributos que podem ser adicionados a elementos HTML para identificá-los de maneira específica. Diferentes elementos podem compartilhar a mesma classe, e um elemento pode ter várias classes.

Sintaxe: Para adicionar uma classe a um elemento HTML, utiliza-se o atributo "class"e atribui-se um nome de classe. Como por exemplo no seguinte código:

```
<p class="destaque">destaque.</p>
```

Listing 1.2. Sintaxe do atributo "class"

O que são IDs: IDs são atributos usados para identificar exclusivamente um único elemento em uma página HTML. Cada ID deve ser único na página.

Sintaxe: Para adicionar um ID a um elemento HTML, usa-se o atributo id e associa-se um nome de ID. Como por exemplo no código:

```
<div id="cabecalho">Div principal.</p>
```

Listing 1.3. Sintaxe do atributo "ID"

1.2.3. HTML: Inclusão de bibliotecas

As bibliotecas são conjuntos de código pré-escrito que fornecem funcionalidades específicas para facilitar o desenvolvimento de sites e aplicativos. Sua inclusão em HTML é uma prática importante na construção de sites mais complexos, principalmente quando se trata de desenvolvimento com banco de dados em nuvem, pois somente com essas bibliotecas se torna possível utilizar diversos dos serviços disponibilizados. Algumas informações sobre bibliotecas HTML são:

Reutilização de Código: As bibliotecas contêm código pré-escrito que aborda

diversas tarefas, como manipulação do DOM (Document Object Model), animações, validação de formulários, gráficos e, principalmente, a aplicação de um sistema de banco de dados em nuvem. Incluir bibliotecas permite que os desenvolvedores reutilizem esse código em seus projetos, economizando tempo e esforço.

Desenvolvimento Rápido: Com o uso de bibliotecas, é possível desenvolver sites e aplicativos mais rapidamente. Em vez de escrever código do zero para cada funcionalidade, os desenvolvedores podem aproveitar as bibliotecas existentes para acelerar o processo de desenvolvimento.

Confiabilidade e Testes: Bibliotecas amplamente utilizadas são frequentemente testadas pela comunidade de desenvolvedores, o que aumenta a confiabilidade do código. Isso reduz a probabilidade de erros e problemas de segurança.

1.2.3.1. HTML: Como Incluir Bibliotecas

Usando <script> para JavaScript: Para incluir uma biblioteca JavaScript em seu HTML, você pode usar a tag <script> no elemento <head> ou <body>, essas bibliotecas podem ser tanto na nuvem, quanto locais. Respectivamente como exemplo de inclusão em nuvem e local observa-se a Listing 1.4 e Listing 1.5:

```
<script src="https://www.gstatic.com/firebasejs/9.14.0/
   firebase-compat.js"><script>
```

Listing 1.4. Inclusão de biblioteca da nuvem.

```
<script src="sandbox.js"><script>
```

Listing 1.5. Inclusão de biblioteca da local

Ordem de Inclusão: A ordem em que você inclui bibliotecas no HTML pode ser importante, especialmente no caso de bibliotecas JavaScript. Normalmente, é uma boa prática incluir bibliotecas antes da utilização do código JavaScript para garantir que as dependências estejam disponíveis.

1.3. CSS: Introdução e Conceito

O tópico a ser retratado se dedica na explosão dos conceitos de CSS (Cascading Style Sheets). Representa uma linguagem de estilo (en-US) usada para a apresentação de um documento que está estruturado em HTML ou XML (abrangendo diversos tipos de XML, como SVG, MathML ou XHTML). O CSS determina a maneira como os elementos são exibidos em diferentes tipos de mídia, incluindo tela, papel, saída de áudio ou outros meios de apresentação.

CSS é considerada umas das principais liguagens da open web e incorporada nos navegadores da web de acordo com as espicificações da WC3 (World Web consortium). Possui diversas versões como CSS1, CSS2 e CSS3, a recomendada é a CSS2, porém a mais atual e que está em processo de padronização se trata da versão CSS3, que será discutida nessa parte do capítulo.

Um dos principais usos do CSS é se trata da estilização de elementos HTML, permitindo que os desenvolvedores determinem as cores, fontes, tamanhos de texto e espaçamentos. Isso garante que o conteúdo seja apresentado de forma atraente e legível, tornando a experiência do usuário mais agradável. Com a capacidade de adaptar a aparência de um site a diferentes tamanhos de tela, permitindo que os programadores construam interfaces que funcionem corretamente em dispositivos móveis, tablets e desktops, mantendo a consistência visual.

1.3.1. CSS: Estrutura do Codigo

A estrutura de um código CSS é fundamental para manter o projeto organizado e fácil de entender. Uma boa estrutura ajuda a evitar problemas de manutenção, facilitando a colaboração com outros desenvolvedores e melhora a eficiência do desenvolvimento. A estrutura de um código CSS segue uma série de elementos e regras que organizam as instruções de estilo para definir como os elementos HTML devem ser apresentados. A estrutura básica de um código CSS se baseia em:

Seletor: Um seletor é usado para identificar qual elemento HTML ou grupo de elementos será estilizado. Pode ser um elemento HTML específico, uma classe, um ID, ou até mesmo um seletor universal que se aplica a todos os elementos.

Exemplos de seletores:

Seletor de elemento HTML: 'p' estiliza todos os parágrafos.

Seletor de classe: '.destaque' estiliza todos os elementos com a classe "destaque".

Seletor de ID: '#cabecalho' estiliza o elemento com o ID cabecalho.

Propriedades: As propriedades são os atributos de estilo que você deseja aplicar ao(s) elemento(s) selecionado(s). Cada propriedade tem um nome que descreve a característica do estilo e um valor que define como essa característica deve ser aplicada, podem mudar desde a cor, a tamanho, margem, entre outras mudanças.

Exemplo de propriedades:

'color': Define a cor do texto.

'font-size': Define o tamanho da fonte.

'margin': Define margens externas.

'padding': Define espaçamento interno.

'background-color': Define a cor de fundo

Declaração: Uma declaração CSS é composta por um seletor e um conjunto de propriedades e valores. Ela define como um elemento selecionado deve ser estilizado.

Bloco de Regras: Um bloco de regras CSS é um conjunto de uma ou mais declarações CSS que são agrupadas entre chaves. Este bloco é associado a um seletor, que define a quais elementos HTML as declarações dentro do bloco se aplicam. Os blocos de regras permitem estilizar elementos HTML de acordo com as propriedades e valores definidos nas declarações CSS. Eles são a base da estilização em CSS, permitindo que você controle o layout e a aparência de sua página da web.

```
p {
    color: blue;
    font-size: 160px;
}
```

Listing 1.6. Exemplo de declaração e Bloco de regras

Comentários: Você pode incluir comentários no seu código CSS para documentação e esclarecimentos. Comentários começam com /* e terminam com */.

```
/*Exemnplo de um comentario CSS*/
```

Listing 1.7. Exemplo de comentário CSS

Arquivo externo: O código CSS também pode ser colocado em um arquivo externo com a extensão .css e, em seguida, referenciado no documento HTML usando a tag <link>.

```
<link rel="stylesheet" type="text/css" href="estilos.css">
```

Listing 1.8. Exemplo de referência de um arquivo CSS externo

Em resumo, a estrutura de um código CSS consiste em seletores que definem os elementos a serem estilizados, propriedades que descrevem o estilo e os valores que definem como esse estilo deve ser aplicado. As declarações são agrupadas em blocos de regras e podem ser organizadas em um arquivo externo para facilitar a manutenção e a reutilização dos estilos em um site. Comentários também podem ser usados para fornecer informações adicionais sobre o código CSS.

1.3.2. CSS: Demais funções

Além do que foi citado anteriormente o CSS ainda possui diversas outras funções que são do interesse de qualquer desenvolvedor web.

1.3.2.1. CSS: Especificidade

A especificidade CSS refere-se à maneira como os navegadores determinam qual regra de estilo deve ser aplicada a um elemento quando várias regras conflitam entre si. É um conceito fundamental em CSS e é usado para resolver conflitos e determinar a prioridade das regras de estilo em uma página da web. A especificidade é crucial para garantir que o design de uma página seja consistente e previsível.

- **Hierarquia de Seletores:** A hierarquia de seletores determina a ordem de prioridade das regras CSS. Quanto mais específico for o seletor, maior será a prioridade da regra. Por exemplo, uma regra aplicada a um seletor de classe (.classe) terá mais prioridade do que uma regra aplicada a um seletor de tipo (elemento) porque é mais específica.

- **ID vs. Classes vs. Elementos:** A especificidade também depende do tipo de seletor usado. Os seletores ID (#id) têm mais peso do que seletores de classe (.classe) e seletores de tipo (elemento). Portanto, uma regra com um seletor ID terá prioridade sobre regras com seletores de classe ou elementos.

- **Inline Styles:** As regras de estilo definidas diretamente em um elemento HTML usando o atributo style têm a maior prioridade e substituem todas as outras regras. Isso significa que os estilos inline prevalecerão sobre qualquer outra definição de estilo.

- **!important:** O valor !important pode ser anexado a uma declaração de estilo em uma regra CSS para dar a ela a mais alta prioridade. No entanto, o uso excessivo de !important não é uma prática recomendada, pois pode tornar o código CSS mais difícil de manter e depurar.

- **Especificidade de Combinação:** Quando há várias partes em um seletor, como combinação de seletores de classe, ID, elementos e pseudo-classes, a especificidade é calculada somando os valores correspondentes a cada parte. Por exemplo, um seletor como div#id .classe terá uma especificidade maior do que um seletor como .classe .classe.

- **Cascata:** A ordem em que as regras CSS são declaradas também é importante. As regras que aparecem posteriormente no código têm precedência sobre as regras anteriores, desde que tenham a mesma especificidade. Isso é conhecido como ordem de cascata.

1.3.2.2. CSS: Herança

A herança em CSS é um princípio fundamental que determina como as propriedades CSS são aplicadas a elementos HTML dentro de uma página da web. A herança permite que você defina estilos para elementos pai e permita que esses estilos sejam herdados por elementos filhos, economizando assim muito tempo e esforço no desenvolvimento de páginas web consistentes e bem projetadas.

- **Como Funciona:** A herança em CSS segue a hierarquia de elementos HTML. Isso significa que, quando você aplica um estilo a um elemento pai, os elementos filhos desse pai herdarão automaticamente esse estilo, a menos que um estilo específico seja aplicado diretamente a eles, substituindo a herança.

- **Propriedades Herdadas:** Nem todas as propriedades CSS são herdadas. As propriedades herdadas são aquelas que fazem sentido serem passadas dos elementos pai para os elementos filhos. Exemplos de propriedades herdadas incluem font-family, color, font-size, line-height e text-align.

- **Propriedades Não Herdadas:** Algumas propriedades CSS não são herdadas, o que significa que elas não são transmitidas aos elementos filhos. Exemplos de propriedades não herdadas incluem width, height, margin, padding e border.

1.4. JavaScript: Introdução e Conceito

Nesta seção será explorada os fundamentos do JavaScript. Esta que é uma linguagem de programação amplamente utilizada no desenvolvimento web para criar funcionalidades em páginas da web. Diferentemente do HTML, que é usado para estruturar o conteúdo de uma página, e do CSS, que é usado para estilizar a página, o JavaScript é usado para adicionar comportamento e interatividadde à web.

- **LIguaguem de script:** JavaScript é uma linguagem de script, o que significa que é executada diretamente no navegador da web do usuário. Isso permite que os desenvolvedores criem funcionalidades que respondem às ações dos usuários sem a necessidade de recarregar a página ou fazer solicitações ao servidor.

- **Interatividade:** Uma das principais funções do JavaScript é tornar as páginas da web interativas. Isso inclui validar formulários, criar animações, responder a cliques de botões etc..

- **Cliente-Side:** JavaScript é executado no lado do cliente, o que significa que o código JavaScript é baixado e executado no navegador do usuário. Isso o torna adequado para tarefas que não requerem acesso a recursos do servidor.

- **Manipulação do DOM:** O DOM (Document Object Model) é uma representação hierárquica da estrutura de uma página da web. JavaScript é usado para acessar e manipular o DOM, permitindo que os desenvolvedores alterem o conteúdo e a aparência de uma página.

- **Ampla Comunidade e Ecossistema:** JavaScript possui uma grande comunidade de desenvolvedores e um ecossistema rico de bibliotecas e frameworks que facilitam o desenvolvimento de aplicativos web complexos. Alguns dos frameworks populares incluem React, Angular e Vue.js.

- **Versatilidade:** Além de ser usado no desenvolvimento web, JavaScript também é usado em outras áreas, como desenvolvimento de jogos (usando bibliotecas como Phaser), desenvolvimento de aplicativos móveis híbridos (usando frameworks como o Apache Cordova) e até mesmo em extenções para o navegador chrome.

1.4.1. JavaScript: Linguagem de Script

Uma linguagem de script é uma linguagem de programação que é projetada para ser executada por um interpretador ou máquina virtual, em oposição a linguagens compiladas, que são traduzidas para código de máquina antes de serem executadas.

- **Interpretação:** As linguagens de script são geralmente interpretadas, o que significa que o código é lido linha por linha e executado diretamente por um interpretador, sem a necessidade de um processo de compilação separado. Isso torna a escrita e a execução de código mais flexíveis e rápidas.

- **Facilidade de Uso:** As linguagens de script são frequentemente projetadas para serem de fácil leitura e escrita, o que as torna acessíveis a programadores de diferentes níveis de habilidade. Elas geralmente têm uma sintaxe mais simples e abstrações de alto nível que simplificam tarefas comuns.
- **Tempo de Execução:**Como as linguagens de script são interpretadas em tempo de execução, elas permitem uma execução dinâmica do código, o que significa que as decisões podem ser tomadas com base em condições que só são conhecidas durante a execução.
- **Tempo de Execução:** Como as linguagens de script são interpretadas em tempo de execução, elas permitem uma execução dinâmica do código, o que significa que as decisões podem ser tomadas com base em condições que só são conhecidas durante a execução.
- **Linguagens de Script na Web:** Linguagens de script desempenham um papel crucial na criação de interatividade em páginas da web. JavaScript é a linguagem de script mais comumente usada na web para validar formulários, criar animações, manipular o DOM e muito mais.

1.4.2. JavaScript: Interativisdade

A interatividade em JavaScript refere-se à capacidade de criar interações em seu código. Isso é fundamental para realizar tarefas como processar listas de dados, criar animações, validar entradas do usuário, etc.

Variáveis: Em JavaScript são utilizadas para armazenar dados e valores que podem ser usados ao longo do programa. Elas são um conceito fundamental em programação e desempenham um papel crucial na manipulação de informações. Por se tratar de uma linguagem dinâmica, não é necessário declarar explicitamente o tipo de uma variável. já que o mesmo é determinado em tempo de execução com base no seu valor atribuído.

Existem três palavras-chave para declarar variáveis em JavaScript. Essas que são:

Tabela 1.3. Tipos de variáveis JavaScript

Variáveis	Explicação
var	**Foi a forma tradicional de declarar variáveis em JavaScript, mas possui escopo de função, o que pode levar a problemas de escopo. É raramente usado hoje em dia.**
let	**Introduzido no ES6 (ECMAScript 2015), o let permite declarar variáveis com escopo de bloco, o que significa que elas são visíveis apenas dentro do bloco onde foram declaradas.**
const	**Também introduzido no ES6, o const é usado para criar variáveis de somente leitura, ou seja, uma vez atribuído um valor, ele não pode ser alterado.**

```
let saldo = 1000;
const nome = "Paulo";
var idade = 25;
```

Listing 1.9. Exemplo das variáveis JavaScript

Há também algumas regras para a nomeclatura de variáves, como por exemplo, deve-se começar com uma letra, sublinhado (_) ou cifrão ($), pode-se conter letras, números, sublinhados e cifrões, sensíveis a maiúsculas e minúsculas (ex.: minhaVariavel e minhavariavel são diferentes).

Funções:A criação de funções em JavaScript é um conceito estrutural que permite o agrupamento de um conjunto de instruções em um bloco nomeado e reutilizável. Funções em JavaScript são usadas para organizar e modular o código, tornando-o mais legível e fácil de manter

A sintaxe básica para criar uma função em JavaScript é a seguinte. 'nomeDaFuncao:' Um identificador que representa o nome da função, 'parametro1', 'parametro2', etc.: Parâmetros opcionais que a função pode receber. Eles são usados para passar informações para a função, 'Código da função': O conjunto de instruções que a função executa quando é chamada.

```
function nomeDaFuncao(parametro1, parametro2, ...) {
  // C digo da function
}
```

Listing 1.10. Declaração de uma função em JavaScript

Para executar o código dentro de uma função, sendo assim necessário chamá-la usando o nome e, opcionalmente, passar argumentos para os parâmetros. Por exemplo:

```
function saudacao(nome) {
  console.log("Ol , " + nome + "!");
}
saudacao("Alice");
```

Listing 1.11. Chamando uma função em JavaScript

As funções podem retornar valores usando a palavra-chave return. Isso permite que seja obtido um resultado da função e o utilize em outras partes do código.

```
function soma(a, b) {
  return a + b;
}
let resultado = soma(5, 3);
```

Listing 1.12. Retorno de uma função em JavaScript

As variáveis declaradas dentro de uma função têm escopo local, o que significa que somente são visíveis e acessíveis dentro da função. Isso ajuda a evitar colisões de nomes de variáveis em diferentes partes do código.

If e Else: Em JavaScript, 'if' e 'else' são estruturas de controle de fluxo que permitem tomar decisões com base em condições. Elas são fundamentais para criar lógica condicional em programas.

A estrutura 'if' permite que seja executado um bloco de código se uma determinada condição for verdadeira. Se a condição não for verdadeira, o bloco de código não

será executado. A sintaxe geral é a seguinte:

```
let idade = 18;
if (idade >= 18) {
  console.log("alguma_coisa");
}
```

Listing 1.13. Exemplo do uso do if

A estrutura if...else permite que seja efetuado um bloco de código se a condição for verdadeira (if) e outro bloco de código se a condição for falsa (else). A sintaxe é a seguinte:

```
let idade = 15;
if (idade >= 18) {
  console.log("alguma_coisa");
} else {
  console.log("alguma_outra_coisa");
}
```

Listing 1.14. Exemplo do uso do if...else

Em resumo, as estruturas if e else são cruciais para criar lógica condicional em programas JavaScript, permitindo uma tomada de decisões com base nas condições especificadas. São amplamente utilizadas em scripts para tornar o código mais flexível e responsivo.

Manipulação do DOM: O gerenciamento do Document Object Model ou DOM, com Java=Script é um aspecto fundamental do desenvolvimento web. O DOM é uma representação em hierarquia da estrutura de uma página web, sua manipulação permite que os desenvolvedores acessem e modifiquem os elementos HTML e sua estrutura dinamicamente.

Para acessar elementos HTML no DOM, pode-se utilizar os métodos 'getElementById', 'getElementsByClassName', 'getElementsByTagName', 'querySelector', 'querySelectorAll' e outros. Aqui estão alguns exemplos:

- **getElementById:** Acessa um elemento pelo seu ID.
- **getElementsByClassName:** Acessa elementos por sua classe.
- **getElementsByTagName:** Acessa elementos por suas tags.
- **querySelector:** Acessa o primeiro elemento de um seletor CSS.
- **querySelectorAll** Acessa todos os elementos de um seletor CSS.

A manipulação do DOM com JavaScript é uma habilidade essencial para criar páginas da web interativas e dinâmicas. Com os comandos corretos, pode-se criar, modificar e remover elementos, tornando aplicações web mais flexíveis e responsivas. É importante

lembrar de usar essa manipulação com cuidado para evitar problemas de desempenho e acessibilidade.

addEventListener: É um método fundamental em JavaScript que permite adicionar ouvintes de eventos a elementos HTML. Eventos são ações que ocorrem em uma página web, como cliques do mouse, pressionamentos de teclas, carregamento da página, entre outros. O 'addEventListener' permite que sejam registradas funções (ouvintes) que serão executadas quando um evento específico ocorrer em um elemento HTML.

```
elemento.addEventListener(evento, function, opcional);
```

Listing 1.15. Sintaxe do addEventListener

elemento: É o elemento HTML ao qual deseja-se adicionar o ouvinte de evento.

evento: É uma string que especifica o tipo de evento que será ouvido (por exemplo, "click", "keydown", "load", "submit", etc.).

função: É a função que será executada quando o evento ocorrer.

opcional: Pode ser um objeto de opções que permite personalizar o comportamento do ouvinte, como usar a captura de eventos ou uma vez.

```
// Seleciona um elemento pelo ID
let meuBotao = document.getElementById("meu-botao");
// Adiciona um ouvinte de evento de clique ao bot o
meuBotao.addEventListener("click", function() {
  alert("clicado!");
});
```

Listing 1.16. Exemplo de Uso Básico

1.5. Banco de dados em nuvem: Cloud Firestore

Nesse segmento do capítulo, será introduzido o conceito fundamental de bancos de dados em nuvem, bem como informações essenciais sobre o Cloud Firestore, incluindo como integrá-lo ao seu projeto. Para incorporar o Cloud Firestore em aplicativos da web JavaScript, é necessário utilizar o Firebase, que é a plataforma que hospeda e fornece acesso ao Firestore. Você pode incluir o Firebase em seu projeto da web com um CDN (Content Delivery Network) ou instalá-lo via npm (Node Package Manager) para ter acesso às bibliotecas necessárias.

Uma vez integrado, você poderá utilizar as APIs e métodos fornecidos pelo Firebase para interagir com o Firestore no JavaScript da web. Essas funcionalidades permitem que você leia, escreva, atualize e exclua dados no Firestore, além de permitir a escuta de eventos de sincronização em tempo real. Essas capacidades tornam o Cloud Firestore uma ferramenta para o desenvolvimento de aplicativos web responsivos e dinâmicos na nuvem.

1.5.1. Introdução a banco de dados

Um banco de dados é um sistema organizado para armazenar, gerenciar e recuperar informações de maneira eficiente, que desempenha um papel crucial na maioria das aplicações de software, pois permite que os desenvolvedores armazenem e acessem dados de forma estruturada e confiável.

Bancos de dados são usados para armazenar uma variedade de informações, desde simples listas de tarefas até informações complexas em sistemas de gerenciamento de clientes, registros de vendas, registros médicos e muito mais. Desempenham um papel fundamental na persistência de dados, o que significa que os dados podem ser mantidos mesmo após a aplicação ser encerrada.

1.5.2. O que é o Cloud Firestore

O Cloud Firestore é um banco de dados NoSQL (Not Only SQL), ou seja, faz parte de uma categoria de bancos de dados que não seguem o modelo de bancos de dados relacionais tradicionais (SQL), que são baseados em tabelas com esquemas rigorosos. Em vez disso, os bancos de dados NoSQL adotam modelos de dados mais flexíveis e escaláveis, projetados para atender a diferentes necessidades de armazenamento e recuperação de informações.

Oferecido pela Google como parte da plataforma Google Cloud. Diferentemente dos bancos de dados tradicionais, o Firestore adota um modelo de dados flexível, onde os dados são organizados em documentos e coleções. Se destaca por sua escalabilidade, capacidade de sincronização em tempo real e integração com a infraestrutura de nuvem da Google, tornando-o uma escolha preferêcial para desenvolvedores que buscam construir aplicativos modernos e responsivos na nuvem. Com o Firestore, é possível armazenar, consultar e sincronizar dados de maneira eficaz, tornando-o uma ferramenta para uma variedade de aplicações em todo o mundo.

- **Modelo de Dados Flexível:** Os dados são armazenados em documentos que podem conter campos de diferentes tipos e serem organizados em coleções. Isso oferece flexibilidade significativa na estruturação de dados, permitindo que os desenvolvedores adaptem o banco de dados às necessidades de seus aplicativos.

- **Escalabilidade e Desempenho:** O Firestore é altamente escalável e pode lidar com grandes volumes de dados e tráfego de aplicativos, escalando automaticamente para atender às demandas crescentes.

- **Sincronização em Tempo Real:** Uma das características mais marcantes é a capacidade de sincronizar dados em tempo real entre os clientes e o banco de dados. Qualquer alteração nos dados é refletida imediatamente em todos os clientes conectados, tornando-o adequado para aplicativos que exigem colaboração em tempo real, como aplicativos de chat.

- **Consultas e Indexação:** O Firestore suporta consultas ricas que permitem buscar e filtrar dados com base em vários critérios, tornando-o adequado para aplicativos

que precisam de consultas complexas.O Firestore suporta consultas ricas que permitem buscar e filtrar dados com base em vários critérios, tornando-o adequado para aplicativos que precisam de consultas complexas.

- **Integração com Plataforma Google:** Se integra perfeitamente com outras ferramentas e serviços do Google Cloud, oferecendo uma solução completa para desenvolvimento de aplicativos na nuvem.

1.5.3. Cloud Firestore: Organização em documentos

A estrutura de organização de dados do Cloud Firestore é baseada em documentos, coleções e campos, e é projetada para ser altamente escalável e adaptável às necessidades dos desenvolvedores. Aqui está uma explicação mais detalhada:

- **Documentos:** O documento é a unidade básica de armazenamento de dados no Cloud Firestore. Cada documento é um registro individual que contém um conjunto de campos e seus valores associados. Os documentos são armazenados em coleções e podem ser pensados como equivalentes a registros ou linhas em uma tabela de banco de dados relacional.

- **Coleções:** As coleções são grupos de documentos relacionados. Podem ser criadas e nomeadas de acordo com as necessidades do aplicativo. Uma coleção pode conter vários documentos, e os documentos dentro de uma coleção não precisam ter a mesma estrutura de campos. Isso oferece uma grande flexibilidade na organização dos dados.

- **Campos:** Os campos são pares chave-valor que representam os dados em um documento. Cada campo tem um nome único (chave) e um valor associado. Os valores podem ser de diferentes tipos, como strings, números, datas, arrays ou mesmo outros documentos aninhados. O Cloud Firestore não impõe um esquema rígido, permitindo que diferentes documentos dentro da mesma coleção tenham campos diferentes.

- **Hierarquia:** Os documentos e coleções são organizados em uma hierarquia. Onde épossível aninhar coleções dentro de documentos ou documentos dentro de documentos, criando assim uma estrutura de dados hierárquica. Isso é especialmente útil para representar relacionamentos complexos entre os dados, como um usuário com várias postagens, onde cada usuário é um documento e as postagens podem ser coleções aninhadas dentro desse documento.

1.5.4. Cloud Firestore: Como incluir na web e códigos em JavaScript

Para incluir a biblioteca do Firebase (Cloud Firestore é um serviço do Firebase) em um projeto da web, é necessário seguir alguns passos. A biblioteca do Firebase é usada para interagir com o Cloud Firestore e outros serviços Firebase em aplicativos web. A versão utilizada será a do Firebase (versão 9.14.0).

- Primeiro, é necessário acessar o **Console Firebase**, após deve-se criar um novo projeto ou selecionar um projeto existente onde é desejado utilizar o Cloud Firestore.

- Dentro do projeto Firebase, acesse as configurações do projeto na guia "Geral". Na sequência, role a página até a seção "Seus aplicativos"e proceda com a adição de um novo aplicativo web para realizar a configuração adequada.
- Opte por um nome para o aplicativo da web. O Firebase gerará um objeto de configuração que conterá as informações essenciais para estabelecer a conexão com o projeto. Este objeto seguirá um formato semelhante ao exemplo a seguir:

```
    const firebaseConfig = {
      apiKey: "SUA_API_KEY",
      authDomain: "SEU_DOMNIO.firebaseapp.com",
      projectId: "SEU_ID_DO_PROJETO",
      storageBucket: "SEU_BUCKET.appspot.com",
      messagingSenderId: "SEU_SENDER_ID",
     appId: "SEU_APP_ID"
    };
```

Listing 1.17. Objeto exemplo do firebase

- Adicione a biblioteca do Firebase à sua página HTML, incluindo o seguinte código em seu arquivo HTML, normalmente dentro da tag <head>:

```
    <script src="https://www.gstatic.com/firebasejs
       /9.14.0/firebase-compat.js"></script>
```

Listing 1.18. Inclusão da biblioteca do firebase(9.14.0)

- Após incluir a biblioteca, você precisará inicializar o Firebase com as configurações do seu projeto. Coloque o seguinte código JavaScript em um arquivo ou script dentro do seu projeto:

```
import firebase from 'firebase/compat/app';\

import 'firebase/compat/firestore';

const firebaseConfig = {
  apiKey: "SUA_API_KEY",
  authDomain: "SEU_DOMNIO.firebaseapp.com",
  projectId: "SEU_ID_DO_PROJETO",
  storageBucket: "SEU_BUCKET.appspot.com",
  messagingSenderId: "SEU_SENDER_ID",
  appId: "SEU_APP_ID"
};

firebase.initializeApp(firebaseConfig);

const firestore = firebase.firestore();
```

Listing 1.19. Exemplo de inicialização do firebase

- Para adicionar um novo documento ao Cloud Firestore, utiliza-se o método 'add()' ou 'set()'. O 'add()' cria automaticamente um ID único para o documento, enquanto o 'set()' permite que você defina um ID personalizado.

```
const minhaColecao = collection(db,
'nome-da-sua-colecao');

const novoDocumento = { campo1: 'valor1',
campo2: 'valor2' };
const novoDocumentoRef = await addDoc(minhaColecao,
novoDocumento);

const documentoPersonalizadoRef = await
setDoc(minhaColecao, 'meu-id-personalizado',
novoDocumento);
```

Listing 1.20. Exemplo de como adicionar um documento a nuvem

- Para recuperar documentos do Cloud Firestore, pode-se usar o método get() para obter os dados de um documento específico ou onSnapshot() para ouvir atualizações em tempo real nos documentos.

```
const meuDocumentoRef = doc(db, 'nome-da-sua-colecao',
'id-do-documento');

const documento = await getDoc(meuDocumentoRef);
if (documento.exists()) {
  console.log('Dados do documento:', documento.data());
} else {
  console.log('O documento nao existe.');
}

const unsubscribe = onSnapshot(meuDocumentoRef,
(doc) => {console.log('Dados atualizados do documento:',
doc.data());
});

unsubscribe();
```

Listing 1.21. Exemplo de como vizualizar um documento na nuvem

- Para deletar documentos, utiliza-se o método delete().

```
const meuDocumentoRef = doc(db, 'nome-da-sua-colecao',
'id-do-documento');

await deleteDoc(meuDocumentoRef);
```

Listing 1.22. Exemplo de como deletar um documento da nuvem

1.6. Extensão Chrome: Conceito e Desenvolvimento

Nesta seção final do capítulo,com base no conhecimento adquirido sobre HTML, CSS, JavaScript e Cloud Firestore, será detalhado o conceito de uma extensão e como funciona seu processo de desenvolvimento para que a mesma consiga trabalhar ao lado do banco de dados do firebase. A forma utilizada para a implementação do Firebase em uma extensão será baseada no método do autor Aniket Das no site [1].

- **Conceito de uma extensão Chrome:** Uma extensão do Chrome é um pequeno programa de software que pode ser instalado no navegador Google Chrome. Essas extensões são projetadas para adicionar funcionalidades extras ao navegador, personalizar a experiência do usuário e melhorar a produtividade.

  As extensões do Chrome podem realizar uma variedade de tarefas, como bloquear anúncios, traduzir páginas da web, gerenciar senhas, verificar a ortografia, capturar capturas de tela, adicionar marcadores e muito mais. São criadas por desenvolvedores e estão disponíveis na Chrome Web Store, onde os usuários podem procurar, encontrar e instalar as extensões que desejam.

  Essas extensões geralmente aparecem na barra de ferramentas do navegador como ícones e podem ser ativadas ou desativadas de acordo com as necessidades do usuário. Ajudam a personalizar e melhorar a experiência de navegação no Chrome de acordo com as preferências individuais.

- **Definir o Objetivo da Extenção:** Antes de começar a escrever código, é importante ter uma compreensão clara do que a extensão deve fazer. deve-se ter um objetivo definido para a extensão, seja fornecer funcionalidades adicionais a sites, melhorar a experiência de navegação ou realizar outras tarefas específicas.

- **Cofigurar o Ambiente de Desenvolvimento:** Certificar-se de ter o ambiente de desenvolvimento adequado configurado. Isso geralmente inclui um editor de código, como o VS Code (Visual Studio Code), e o navegador Chrome instalado.

- **Criar a estrutura da extensão:** Uma extensão pode possuir diversos arquivos formadores da mesma dependendo de sua complexidade, porém devem ter no mínimo 4 arquivos principais: manifest.json‘: Este é o arquivo de manifesto da extensão, que contém informações sobre a extensão, como nome, descrição, versão e permissões, script.js : Este é um arquivo de código da extensão, onde está contido o JavaScript do programa, style.css : É um arquivo de folha de estilo que define a aparência visual da extensão, incluindo formatação, cores, fontes e layout, popup.html : Define o conteúdo de uma pequena janela pop-up que pode ser aberta pelo usuário clicando no ícone da extensão na barra de ferramentas do navegador. Esse arquivo HTML contém a interface do usuário que aparece quando a extensão é ativada e fornece uma maneira de interagir com a extensão ou acessar suas funcionalidades principais de forma rápida e conveniente.

- **Definir as permissões necessárias:** No arquivo ‘manifest.json‘, especifique as permissões necessárias para sua extensão funcionar. Por exemplo, se a extensão precisa acessar a página atual, precisará especificar a permissão ‘"activeTab"‘.

- **Criar a interface do usuário:** Usando HTML, CSS e JavaScript para criar a interface da extensão. Isso pode incluir botões, pop-ups, barras de ferramentas ou qualquer outra interface necessária para a interação com o usuário.
- **Escrever o código JavaScript:** Utiliza-se JavaScript para implementar a lógica da sua extensão. pode interagir com as APIs do Chrome para acessar funcionalidades específicas do navegador, como guias, histórico de navegação, cookies, entre outros.
- **Testar a extensão:** Carregar a extensão no Chrome para testá-la. Para fazer isso, é necessário acessar a página de extensões (chrome://extensions/), habilita o modo de desenvolvedor e carreguar a pasta da extensão.

1.6.1. Extensão Chrome: Desenvolvimento junto ao Cloud Firestore

Primeiro é necessário entender que para se utilizar o Firebase dentro de uma extensão é preciso uma série de permisões que só são adquiridas ao comprar o pacote de desenvolvedor exclusivo do Chrome, porém existe outra maneira de criar essa extensão, que se trata basicamente de utilizar um script como background do código, ou seja, o uso de um arquivo HTML e JavaScript extra permite que a extensão execute código em um ambiente controlado e isolado, útil para tarefas específicas que não devem afetar a página da web principal ou para garantir a segurança das operações da extensão.

Criando os arquivos da extensão: Ao abrir o VS Code (Visual Studio Code) se torna possível escolher uma pasta já existente no computador e utiliza-la para criação de novos arquivos, ao clicar em ´New File´, é importante saber que as terminações nos nomes dos arquivos indicam seu tipo, sendo respectivamente ´.js´,´.css´,´.html´,´.json´ para arquivos JavaScript, CSS, HTML e manifest. Para uma extensão básica com Cloud Firestore, serão 7 arquivos essenciais: ccs1.css, css2.css, manifest.json, popup.html, popup.js, sandbox.html e sandbox.js. Cada um será explicado de forma mais detalhada a seguir:

1.6.1.1. Extensão Chrome: manifest.json

O manifest.json é um arquivo de manifesto de extensão do Chrome, usado para configurar e definir as propriedades e comportamentos de uma extensão.

```
{
  "manifest_version": 3,
  "name": "My extension",
  "version": "1.0",
  "permissions": [
    "activeTab",
    "tabs"
  ],
  "action": {
    "default_popup": "popup.html"
  },
  "sandbox": {
```

```
    "pages": ["sandbox.html"]
  },
  "content_security_policy": {
    "sandbox": "sandbox allow-scripts allow-popups; script-
       src 'self' https://www.gstatic.com/ https://*.
       firebaseio.com https://www.googleapis.com"
  }
}
```

Listing 1.23. Arquivo manifest.json de uma extensão com Cloud Firestore

- **manifest_version (Versão do Manifesto):** Indica qual versão o manifesto segue. Cada versão tem suas próprias especificações e recursos, porém somente o manifest V3 é aceito pelo Chrome atualmente.

- **name (Nome):** Define o nome da extensão, que será exibido aos usuários na lista de extensões do Chrome e em outras interações com a extensão.

- **version (Versão):** Especifica a versão atual da extensão. Isso é útil para controlar e rastrear diferentes versões da extensão à medida que ela é atualizada.

- **permissions (Permissões):** Lista as permissões necessárias para a extensão. Neste caso, a extensão requer acesso às guias ativas ("activeTab") e acesso às guias em geral ("tabs"). Isso permite que a extensão interaja com as guias do navegador.

- **action:**Define as ações da extensão. Aqui, a ação padrão é configurada para abrir a página ´popup.html´ quando o ícone da extensão na barra de ferramentas do Chrome é clicado.

- **sandbox:** Especifica que a extensão utiliza uma página ´sandbox.html´ que é executada em um ambiente isolado. Isso ajuda a garantir que o código dentro dessa página não interfira na página da web atual e funcione de forma segura e isolada.

- **content_security_policy:** Define as políticas de segurança de conteúdo para a extensão. Aqui, é configurado para permitir a execução de scripts no ambiente sandbox (´"sandbox allow-scripts allow-popups"´) e também permite a execução de scripts a partir de fontes específicas, como o próprio domínio da extensão, ´https://www.gstatic.com´, ´https://*.firebaseio.com´, e ´https://www.googleapis.com´.

Em resumo, este arquivo manifest.json descreve os detalhes essenciais da extensão, incluindo seu nome, versão, permissões necessárias, comportamento padrão e configurações de segurança para a execução de scripts. Ele é fundamental para configurar e distribuir a extensão no navegador Chrome.

1.6.1.2. Extensão Chrome: popup.html e sandbox.html

O seguinte código HTML cria uma página da web simples com a inclusão de bibliotecas JavaScript (Firebase), um arquivo de script personalizado (sandbox.js), um arquivo de folha de estilo (css1.css), elementos de entrada de texto e botões, e uma lista não ordenada. A interação e funcionalidade específica da página dependerá do conteúdo do arquivo sandbox.js, que provavelmente contém lógica de JavaScript para manipular os elementos da página e realizar ações quando os botões são clicados.

```
<html>
<head>
  <script src="https://www.gstatic.com/firebasejs/9.14.0/
     firebase-compat.js"></script>

  <script src="sandbox.js"></script>

  <link rel="stylesheet" href="css1.css">
</head>
<body>
  <input id="addList" type="text"><br>
  <button id="addDocument">Adicionar</button>
  <button id="removeAll">Remover</button>
  <ul id="list"></ul>
</body>
</html>
```

Listing 1.24. Estrutura do sandbox.html

Este próximo código HTML cria uma página da web simples com um botão, um iframe para incorporar outra página chamada (sandbox.html), um arquivo JavaScript externo chamado "popup.js"e um arquivo de folha de estilo externo chamado (css2.css). É uma estrutura básica para criar uma página que pode ser interativa e estilizada com a ajuda de JavaScript e CSS.

```
<html>
        <body>
                <button id="init">Initialize Firebase</
                   button><br/>
                <iframe id="iframe" src="sandbox.html"></
                   iframe>
                <script src="popup.js"></script>
                <link rel="stylesheet" href="css2.css">
        </body>
</html>
```

Listing 1.25. Estrutura do popup.html

1.6.1.3. Extensão Chrome: css1.css e css2.css

Os arquivos CSS são puramente estilizações e, como tal, não contêm lógica de programação ou funcionalidades específicas. Eles são a ferramenta que permite aos desenvolvedores moldar a estética de uma página, desde o posicionamento de elementos até a escolha das cores, tipografia e animações. A flexibilidade inerente ao CSS significa que os desenvolvedores podem traduzir suas visões criativas em designs únicos, cativantes e, acima de tudo, funcionais, ou seja, dependem somente da preferência do desenvolvedor que cria a aplicação.

1.6.1.4. Extensão Chrome: popup.js e sandbox.js

Agora há dois arquivos JavaScript, cada um pro html de mesmo nome. Enquanto o popup.js somente inicializa o Firebase para que seja possível utiliza-lo na extensão, o sandbox.js faz todo o resto, tendo a função de adicionar, remover e mostrar todos os dados que estão presentes na nuvem.

```
const iframe = document.getElementById("iframe");

document.getElementById("init").onclick = () => {
        iframe.contentWindow.postMessage("init", "*");
};
```

Listing 1.26. Javascript do popup.js

Vale-se ressaltar também, que devemos incluir a configuração do Cloud Firestore no inicio do código dentro do sandbox.js e que após isso pode-se fazer uma extensão que inclua todas as funcionalidas do banco de dados em nuvem do Firebase.

```
const config = {
        projectId: "SEU_ID_DO_PROJETO",
        apiKey: "SUA_API_KEY",
        storageBucket: "SEU_BUCKET.appspot.com",
};

const app = firebase.initializeApp(config);
const db = firebase.firestore();

let isFirebaseInitialized = false;
```

Listing 1.27. Estrutura JavaScript básica do sandbox.js

1.7. Conclusão

Neste capítulo, foram explorados os fundamentos essenciais de desenvolvimento web, abordando os conceitos básicos de HTML, CSS e JavaScript. Além disso, foi introduzido o conceito de persistência de dados em nuvem, utilizando o Visual Studio Code como a

ferramenta de desenvolvimento e o Cloud Firestore, um serviço gratuito fornecido pelo Google, como a plataforma de armazenamento de dados em nuvem.

O principal objetivo deste capítulo é fornecer uma base sólida para aqueles que estão iniciando no desenvolvimento web, com foco na criação de uma extensão simples para o navegador Chrome que tenha como diferencial a utilização de um banco de dados em nuvem. Essa extensão tem a finalidade de auxiliar os iniciantes a compreenderem os princípios fundamentais dessa área da tecnologia.

Referências

[1] [10] Das, A. (2022). How to add firebase to a chrome extension on manifest v3.

[2] Documentação do Firebase sobre ler dados do Cloud Firestore, https://firebase.google.com/docs/firestore/query-data/get-data?hl=pt-br, Acessado em 20 de setembro de 2023.

[3] Mozilla Developer Network (MDN) para HTML, https://developer.mozilla.org/en-US/docs/Web/HTML, Acessado em 20 de setembro de 2023.

[4] Mozilla Developer Network (MDN) para JavaScript, https://developer.mozilla.org/en-US/docs/Web/JavaScript, Acessado em 20 de setembro de 2023.

[5] Mozilla Developer Network (MDN) para CSS, https://developer.mozilla.org/en-US/docs/Web/CSS, Acessado em 20 de setembro de 2023.

[6] Mozilla Developer Network (MDN) para HTML, https://developer.mozilla.org/en-US/docs/Web/HTML, Acessado em 20 de setembro de 2023.

[7] Documentação do Firebase sobre adicionar o Firebase ao projeto JavaScript, https://firebase.google.com/docs/web/setup?hl=pt-br, Acessado em 20 de setembro de 2023.

[8] Documentação do Firebase de introdução ao Cloud Firestore, https://firebase.google.com/docs/firestore?hl=pt-br, Acessado em 20 de setembro de 2023.

[9] Documentação do Firebase sobre adicionar dados ao Cloud Firestore, https://firebase.google.com/docs/firestore/manage-data/add-data?hl=pt-br, Acessado em 20 de setembro de 2023.

[10] Documentação do Firebase sobre excluir dados ao Cloud Firestore, https://firebase.google.com/docs/firestore/manage-data/delete-data?hl=pt-br, Acessado em 20 de setembro de 2023.

Capítulo 2

Explorando a Explicabilidade da Inteligência Artificial - Técnicas para Compreender e Interpretar Modelos de Aprendizado de Máquina

Júlio V. M. Marques, Clésio A. Gonçalves, Pablo de Abreu Vieira, Armando L. Borges, Viviane B. Leal Dias, Willians S. Santos e Romuere R. V. Silva

Abstract

Artificial Intelligence (AI) has become powerful, but explainability is crucial for its reliability in sensitive areas. Interpretable models offer transparent insights, while "black boxes"lack explanation. Visualization techniques, such as heatmaps, aid in understanding. Pursuing explainability is ethical and essential to ensure trust and accountability in applications, especially in medical diagnostics. In this context, this book chapter explores its significance, techniques, evaluation, applications, challenges, limitations, and key conclusions in this field.

Resumo

A Inteligência Artificial (IA) tornou-se poderosa, mas a explicabilidade é crucial para sua confiabilidade em áreas sensíveis. Modelos interpretáveis oferecem insights transparentes, enquanto as "caixas pretas"carecem de explicação. Técnicas de visualização, como mapas de calor, auxiliam na compreensão. Buscar explicabilidade é ético e essencial para garantir a confiança e a responsabilidade em aplicações, especialmente em diagnósticos médicos. Nesse contexto, este capítulo de livro explora sua importância, técnicas, avaliação, aplicações, desafios, limitações e as principais conclusões dessa área.

2.1. Introdução

Nos últimos anos, a Inteligência Artificial (IA) consolidou-se como uma poderosa ferramenta com aplicações que abrangem desde diagnósticos médicos [Grif and Avush 2018] até à condução autônoma de veículos [Nivas et al. 2016]. No entanto, à medida que a complexidade dos modelos de IA aumenta, a demanda por compreender e explicar suas decisões torna-se crítica [Arrieta et al. 2020]. A explicabilidade na IA desempenha um

papel fundamental na construção de sistemas responsáveis e éticos, garantindo a confiança dos usuários e a conformidade com regulamentações, especialmente em setores sensíveis.

Existem diferentes tipos de IA, algumas são interpretáveis e outras são consideradas caixas pretas. Os modelos interpretáveis são projetados para serem transparentes, proporcionando uma visão direta de como tomam decisões e fazem previsões, assim, revelando o processo de tomada de decisão e oferecendo *insights* compreensíveis sobre como chegam a suas conclusões [Agarwal and Das 2020], exemplos desse modelo são as árvores de decisão [Jou 1986], regressões lineares [Yan and Su 2009] e máquinas de vetores de suporte [Cortes and Vapnik 1995]. Os modelos consideradas "caixas pretas"produzem previsões ou decisões sem oferecer uma explicação clara do raciocínio subjacente [Ribeiro et al. 2016a]. Isso pode ser especialmente problemático em cenários críticos, como cuidados de saúde e justiça criminal, onde a compreensão das razões por trás das decisões é crucial. Além disso, a transparência é essencial para evitar que os modelos de IA perpetuem informações indesejadas. Portanto, a explicabilidade desempenha um papel essencial na construção de sistemas de IA confiáveis e morais. Isso levou à adoção de modelos de aprendizado de máquina intrinsecamente interpretáveis.

Adicionalmente, as técnicas de visualização, como os mapas de calor, desempenham um papel fundamental na compreensão de modelos complexos, como redes neurais. Os mapas de calor destacam as partes mais influentes dos dados de entrada, especialmente em tarefas de visão computacional, onde revelam quais regiões de uma imagem são mais relevantes para uma determinada classificação [Selvaraju et al. 2019].

A busca pela explicabilidade na IA é uma resposta essencial à crescente complexidade dos modelos atuais e às necessidades de setores críticos. A transparência não é apenas uma questão ética, mas também uma exigência crucial para construir uma aplicação de confiança. À medida que a tecnologia avança, a comunidade científica e os profissionais da área continuarão aprimorando as técnicas de explicabilidade, com o objetivo de impulsionar a confiança, a responsabilidade e a aplicação ética em uma ampla gama de campos, que incluem resolução de desafios complexos como diagnósticos médicos.

Este trabalho está estruturado da seguinte forma: a Seção 2.2 explora a Importância da Explicabilidade na Inteligência Artificial; a Seção 2.3 discute as Técnicas de Explicabilidade, incluindo Modelos Interpretáveis, Mapas de Calor (*Heatmaps*), *Class Activation Mapping* (Grad-CAM) e *Local Surrogate* (LIME); a Seção 2.4 aborda a Avaliação da Explicabilidade; a Seção 2.5 destaca Aplicações Práticas; a Seção 2.6 explora os Desafios e Limitações; por fim, a Seção 2.7 oferece principais *insights* e conclusões.

2.2. Importância da Explicabilidade na Inteligência Artificial

Modelos de IA interpretáveis tornam possível que suas previsões sejam examinadas e explicadas, sendo possível identificar quaisquer desvios indesejados e até mesmo antiéticos em seu comportamento e forma de analisar os dados. Neste contexto, esta seção aborda alguns fatores que destacam a importância da explicabilidade de modelos de IA e a análise de suas previsões para identificar comportamentos inadequados do ponto de vista ético e técnico.

Segundo [Pedreshi et al. 2008], modelos de classificação treinados com dados históricos podem ser discriminatórios no sentido social e negativo, visto que estes dados podem conter padrões preconceituosos fortemente enraizados na sociedade atual. Neste contexto, partindo da ideia de que esses modelos de IA podem, e são utilizados como forma de apoio a decisão em vários contextos, como liberação de crédito bancário ou acesso a serviços públicos, é evidente que eles podem incluir comportamentos socialmente, racialmente e etnologicamente segregativos, prejudicando negativamente classes menos favorecidas, replicando assim, comportamentos antiéticos tradicionais.

Nas décadas de 1970 e 1980, a Faculdade de Medicina do Hospital St George's usou um algoritmo para triagem de candidatos que concorriam a oportunidades de emprego na instituição. Este programa utilizava informações contidas nos formulários do candidatos, o qual não continha qualquer referência étnica. No entanto, descobriu-se que o programa discriminava de forma injusta as minorias étnicas e pessoas do sexo feminino, inferindo esta informação através dos seus nomes e locais de nascimento, reduzindo suas probabilidades de serem selecionados pelo algoritmo e para uma futura entrevista [Lowry and Macpherson 1988]. Comportamentos como este podem se repetir em muitos outros casos em que é aplicado o uso da IA para realizar tomadas de decisão. O estudo conduzido por [Caliskan et al. 2017] revela preconceitos humanos encontrados em textos e corpus da web, onde constatou-se que nomes de pessoas negras estavam mais associados a termos negativos e desagradáveis em comparação com nomes de pessoas brancas. Com isso, é possível notar que a principal causa dos comportamentos negativamente discriminatórios dos sistemas baseados em IA advém dos dados que são usados em seu treino, visto que eles podem conter vieses antiéticos, mesmo que sutis, mas que podem levar esses sistemas a tais tipos de atitude.

A IA está continuamente sendo aplicada e aprimorada nos mais diversos campos e áreas do conhecimento, principalmente na área da saúde para auxiliar no diagnóstico de doenças por imagens [Borges et al. 2022] [Gennatas and Chen 2021]. Aplicações baseadas em sistemas como esse tem um grande potencial de auxiliar profissionais no desempenho de suas atividades. No entanto, diversos modelos de IA, com ênfase naqueles baseados em redes neurais profundas, não oferecem esclarecimentos claros sobre como chegaram em determinada conclusão, e portanto, são conhecidos como "caixas-pretas"[Rudin 2019]. Esse problema pode se tornar mais grave quando trazido a contextos delicados, em que é de suma importância a garantia de que determinado fator ou objeto tenha sido analisado e levado em conta pelo modelo antes do mesmo dar um resposta a inferência, como por exemplo, na análise de exames médicos. Neste contexto, tais modelos podem aprender padrões presentes em sua base de treino que são irrelevantes ao diagnóstico enquanto descarta os padrões que deveriam ser unicamente levados em conta, podendo levar a erros ocasionais em futuras inferências.

Estas situações podem se replicar em quaisquer outros contextos, de outros campos e áreas. Portanto, é necessário entender como estes modelos funcionam e como seus resultados podem ser explicados e justificados, com o intuito de compreender os elementos que estão sendo analisados e considerados pelo sistema para formular suas respostas, evitando assim, graves erros, principalmente em contextos delicados, nos quais eles podem ocasionar em danos e/ou prejuízos para as pessoas. Para isso, existem diversas técnicas de explicabilidade, como Mapas de Calor (*Heatmaps*), *Class Activation Map-*

ping (Grad-CAM), *Local Surrogate* (LIME) e os próprios Modelos Interpretáveis. Tais técnicas serão discutidas e analisadas ao longo do capítulo.

2.3. Técnicas de Explicabilidade

Nesta seção, é explorado uma variedade de técnicas de explicabilidade em *machine learning* e inteligência artificial. Essas técnicas são projetadas para tornar os modelos de aprendizado de máquina mais transparentes e compreensíveis, permitindo que os usuários entendam como e por que os modelos tomam decisões.

2.3.1. Modelos Interpretáveis

Os modelos interpretáveis são modelos de aprendizado de máquina que têm a capacidade de serem explicados e compreendidos de forma direta. Esses modelos são construídos com a intenção de manter uma relação clara entre as características de entrada, as saídas ou previsões, tornando-os transparentes e interpretáveis. A interpretabilidade é uma característica importante em muitas aplicações de IA, especialmente em áreas onde a transparência, a responsabilidade e a confiança são críticas, como medicina, direito, finanças e sistemas de suporte à decisão. Aqui estão alguns exemplos de modelos interpretáveis:

- **Regressão Linear** [Yan and Su 2009]: Um modelo linear estabelece uma relação linear entre as características de entrada e a variável de saída. É possível ver diretamente como as características afetam a saída por meio dos coeficientes, contribuindo para uma interpretabilidade clara.
- **Árvores de Decisão** [Jou 1986]: As árvores de decisão dividem os dados em um conjunto de regras hierárquicas baseadas nas características de entrada. A interpretabilidade é natural, pois pode-se seguir o caminho da árvore para entender as decisões.
- **Regressão Logística** [Cox 1958]: A regressão logística é usada para problemas de classificação binária e fornece uma interpretação direta dos coeficientes em relação às características.
- **Regras de Associação** [Pei 2009]: Esses modelos geram regras do tipo "se... então..."com base nas relações entre as características de entrada e a variável de saída, Gerando assim a interpretabilidade.
- **Máquinas de Vetores de Suporte (SVM)** [Cortes and Vapnik 1995]: Quando um *kernel* linear é usado em SVMs, o limite de decisão é uma linha reta, tornando o modelo interpretável em problemas de classificação.

Algumas técnicas podem ser aplicadas a modelos interpretáveis para fornecer uma compreensão mais completa de como eles funcionam e tomam decisões. A escolha das técnicas específicas depende do modelo, do problema e do público-alvo. Dentre elas, destacam-se:

- **Visualização de Coeficientes**: Em modelos lineares, como a regressão linear ou logística, pode-se visualizar diretamente os coeficientes atribuídos a cada característica, destacando a influência de cada uma.

- **Gráficos de Barras de Importância de Características**: Para árvores de decisão e modelos baseados em regras, pode-se criar gráficos de barras que mostram a importância relativa de cada característica na tomada de decisões.
- **Gráficos de Dependência Parcial**: Esses gráficos mostram como a variável de saída se relaciona com uma variável de entrada específica, mantendo outras variáveis constantes. São úteis para ilustrar as relações entre as características e as previsões em modelos como árvores de decisão.
- **Matrizes de Correlação**: Em modelos interpretáveis, como regressões lineares, exibir matrizes de correlação pode ajudar a identificar relações lineares entre características e a variável de saída.
- **Análise de Resíduos**: Para modelos de regressão, a análise de resíduos ajuda a avaliar as suposições do modelo.
- **Regras de Explicação**: Para modelos baseados em regras, pode explicar as decisões por meio de regras simples do tipo "se... então...", facilitando a compreensão das decisões do modelo.

2.3.2. Mapas de Calor (*Heatmaps*)

Mapas de calor (*heatmaps*) são uma técnica de explicabilidade amplamente utilizada em modelos de IA para visualizar e comunicar como as características de entrada afetam as previsões ou saídas do modelo. Essa técnica é especialmente útil quando se deseja entender como as características contribuem para as decisões do modelo em dados tabulares.

Os mapas de calor são normalmente aplicados a uma única instância de dados de entrada ou a um conjunto de dados de entrada. Cada instância de dados contém características (atributos) que são alimentadas ao modelo de IA. Para cada característica de entrada, o mapa de calor calcula a contribuição relativa dessa característica para a saída ou previsão do modelo.

As contribuições são representadas visualmente por meio de cores em um mapa de calor. Geralmente, as cores quentes, como vermelho ou amarelo, indicam uma contribuição positiva da característica para a saída, enquanto as cores frias, como azul ou verde, indicam uma contribuição negativa ou nula. A intensidade da cor pode representar a magnitude da contribuição.

O mapa de calor é gerado de forma que cada característica seja mapeada em um eixo (horizontal ou vertical) e a saída do modelo ou a importância da característica seja mapeada no outro eixo. Isso cria uma representação visual que destaca quais características têm maior impacto nas previsões ou saídas do modelo. A Figura 2.1 abaixo ilustra um exemplo de mapa de calor. Podemos notar a concentração populacional por meio das cores, onde, quanto mais escura a tonalidade maior é a densidade populacional. Por meio desses mapas de calor podemos extrair informações visuais das regiões com maiores significância para a tarefa estudada.

O uso dos mapas de calor na explicabilidade de modelos de IA contribui principalmente na interpretação de modelos complexos, identificação de características impor-

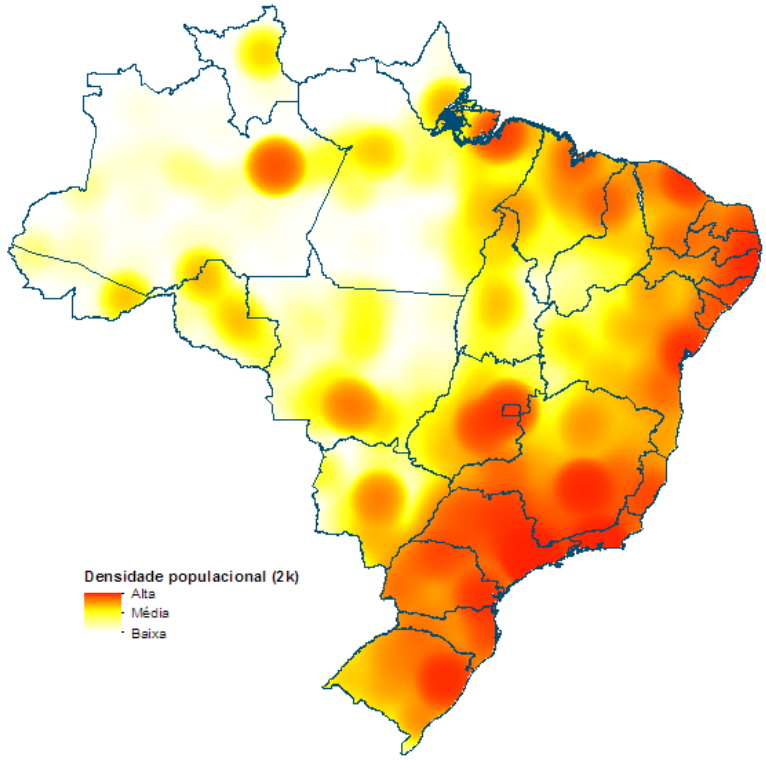


Figura 2.1: Densidade populacional do Brasil em 2010. As áreas com cores mais quentes representam uma maior densidade populacional. Fonte: [Forest-gis]

tantes, detecção de relações não lineares, diagnóstico de erros, além da validação e ajuste de modelos.

É importante notar que a interpretação de mapas de calor pode ser subjetiva, e os resultados podem depender da técnica específica de cálculo da contribuição das características. Portanto, é aconselhável usar mapas de calor em conjunto com outras técnicas de explicabilidade para obter uma compreensão completa do comportamento do modelo de IA.

2.3.3. *Class Activation Mapping* (Grad-CAM)

O Grad-CAM, ou *Class Activation Mapping*, é uma técnica de interpretação de modelos de aprendizado profundo que ajuda a visualizar quais partes de uma imagem são mais influentes na tomada de decisão do modelo durante a classificação [Selvaraju et al. 2017]. Ele se tornou uma ferramenta essencial para entender como as redes neurais convolucionais (CNNs) "olham"para as imagens e é amplamente utilizado em tarefas de visão computacional, como classificação de imagens e detecção de objetos.

O Grad-CAM é aplicado principalmente a modelos de CNN, que são amplamente usados em tarefas de visão computacional. Esses modelos são compostos por várias camadas convolucionais que extraem características relevantes das imagens. Essas camadas convolucionais produzem mapas de ativação, que destacam as regiões da imagem onde certas características foram detectadas. Quanto mais profunda a camada, mais abstratas

são as características que ela representa. Em um modelo de classificação de imagem, as camadas convolucionais são seguidas por camadas totalmente conectadas que fazem a decisão final sobre a classe à qual a imagem pertence. O Grad-CAM se concentra em entender como essas camadas totalmente conectadas ponderam as características das camadas convolucionais.

O processo pode ser explicado da seguinte forma: Primeiramente, uma imagem é alimentada ao modelo, que gera uma previsão. A classe alvo é determinada a partir dessa previsão. Em seguida, o Grad-CAM calcula os gradientes da classe alvo em relação às ativações na camada convolutiva mais profunda. Isso indica quais ativações foram mais influentes na decisão da classe. Então o Grad-CAM combina os gradientes com as ativações da camada convolutiva usando uma média ponderada [Chattopadhay et al. 2018]. Essa média ponderada é usada para obter os pesos de ativação de cada canal na camada convolutiva. Por fim, os pesos de ativação são usados para criar um mapa de ativação que destaca as regiões da imagem que mais contribuíram para a decisão da classe alvo. Na Figura 2.2, temos uma demonstração da utilização do Grad-CAM para visualizar as zonas de ativações do modelo. As zonas em azul, se concentram nas regiões que os modelos consideram mais importante para a tarefa, podemos notar que em sua maioria as regiões se concentram dentro do parênquima pulmonar, o que mostra que o modelo realmente está levando em consideração regiões que são de fato importante para a detecção da COVID-19.

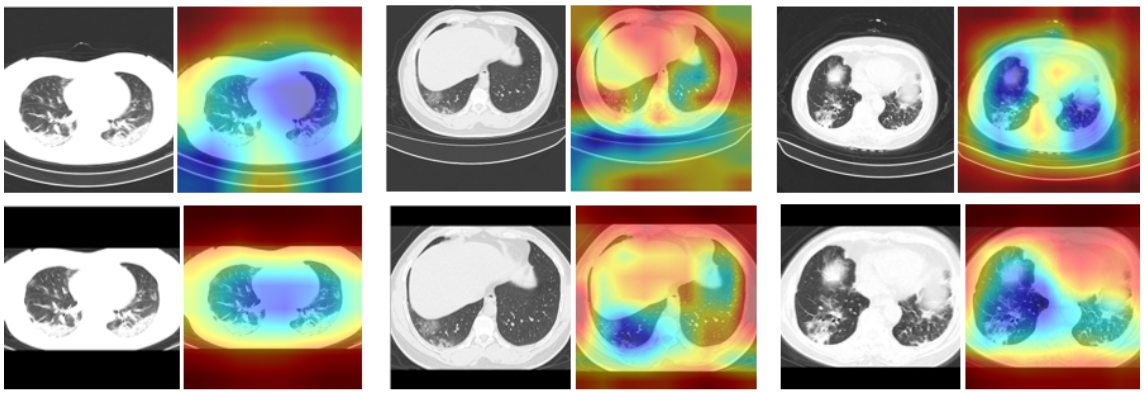


Figura 2.2: Zonas de ativações geradas pelo Grad-CAM para um modelo de detecção de COVID-19 em imagens de tomografia computadorizada. Fonte: [Marques et al. 2023]

O Grad-CAM oferece várias vantagens, ele torna as decisões dos modelos de *deep learning* mais transparentes, permitindo que os usuários compreendam quais características da imagem levaram à classificação. O Grad-CAM não apenas indica a classe alvo, mas também localiza as regiões da imagem que foram mais relevantes para essa classificação, fornecendo *insights* sobre como esses modelos tomam decisões. Sua capacidade de localizar características relevantes em imagens torna-o valioso em uma variedade de domínios, desde medicina, mostrando as regiões em destaques que revelam características relevantes para o problema, ate segurança, podendo ser utilizado para identificar regiões onde o modelo consideram importantes para manter a segurança. À medida que a pes-

quisa em interpretabilidade de modelos de aprendizado profundo avança, o Grad-CAM continua sendo uma das técnicas mais amplamente utilizadas.

2.3.4. *Local Surrogate* (LIME)

O *Local Interpretable Model-agnostic Explanations* (LIME) é uma técnica que visa tornar os modelos de aprendizado de máquina mais transparentes e interpretáveis, especialmente em nível local [Ribeiro et al. 2016b]. Ele se concentra em explicar as previsões de modelos de *machine learning* para instâncias de dados individuais, permitindo que os usuários entendam como o modelo chegou a uma decisão específica para um caso particular.

Modelos de aprendizado de máquina modernos, como redes neurais profundas, são frequentemente complexos e difíceis de interpretar. O LIME aborda essa complexidade fornecendo explicações compreensíveis para previsões de modelos de alta dimensionalidade. O LIME cria modelos locais (*surrogates*) que são muito mais simples do que o modelo de *machine learning* original, mas ainda são explicativos. Esses modelos locais são criados para imitar o comportamento do modelo original em torno da instância de dados de interesse. O LIME funciona introduzindo pequenas alterações ou modificações nos dados de entrada da instância que está sendo analisada e observando como as previsões do modelo original mudam em resposta a essas alterações. Isso ajuda a entender a sensibilidade do modelo à variação nos dados de entrada.

O processo do LIME envolve os seguintes passos: Primeiramente, uma instância de dados individual para a qual se deseja explicar a previsão é selecionada. O LIME cria uma série de instâncias com pequenas alterações a partir da instância de interesse. Essas variações e alterações podem ser feitas de várias maneiras, como adição de ruído ou remoção de recursos. As instâncias com alterações são alimentadas ao modelo de *machine learning* original, e suas previsões são registradas. Um modelo local, geralmente linear ou outro modelo, é treinado usando as instâncias com alterações e as previsões correspondentes do modelo original. Esse modelo local atua como um substituto ou *surrogates* do modelo original. O modelo local é interpretável, permitindo que os usuários entendam as relações entre os recursos de entrada e as previsões do modelo original para a instância de interesse.

O LIME oferece várias vantagens: Ele fornece explicações interpretáveis para previsões de modelos complexos em nível local, o que é útil quando se deseja entender o raciocínio do modelo para casos específicos. O LIME é "agnóstico"em relação ao tipo de modelo de *machine learning* usado, o que significa que pode ser aplicado a uma ampla variedade de modelos sem necessidade de conhecimento interno sobre eles. Pode ser aplicado em várias tarefas, incluindo classificação, regressão e até mesmo em tarefas de processamento de linguagem natural. Na Figura 2.3, podemos observar uma representação do modelo LIME para varias entradas. Nessa figura, temos as instâncias de entrada com variações e alterações, o modelo LIME interpreta essa variação para cada instância de entrada e tenta representa-la por meio de cores, mostrando a região de ativação para cada entrada.

O LIME é uma ferramenta valiosa para tornar os modelos de *machine learning* mais interpretáveis, fornecendo explicações compreensíveis para previsões em nível local. Sua capacidade de criar modelos locais simples que imitam o comportamento de

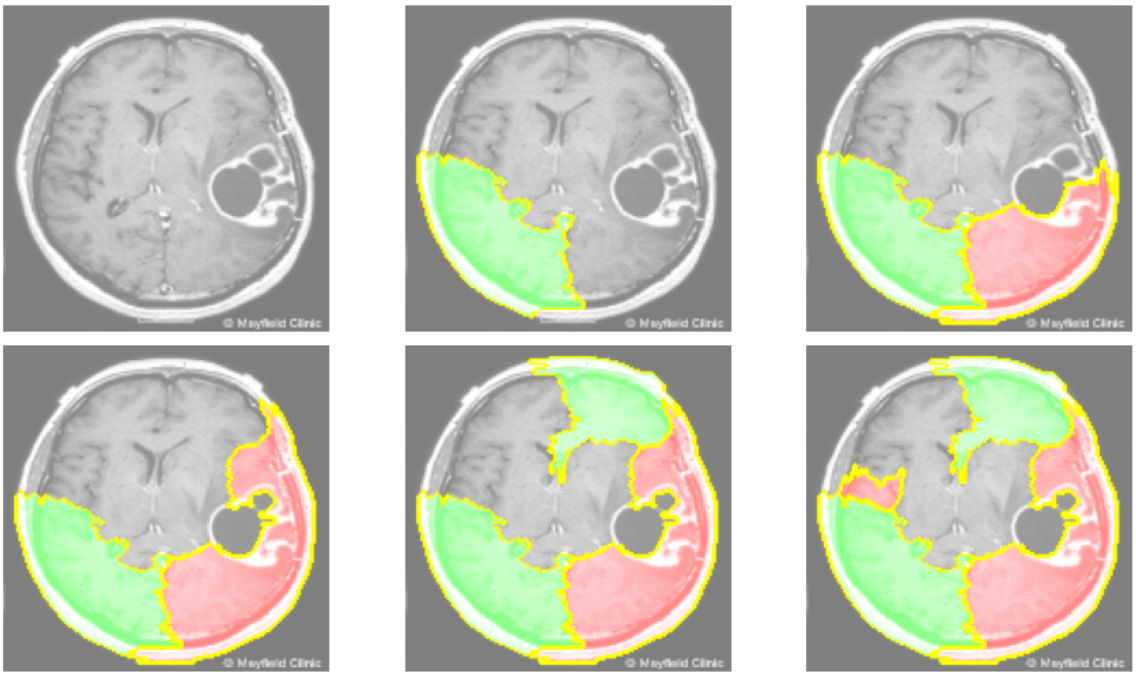


Figura 2.3: Zonas de ativações geradas pelo LIME para um modelo de detecção de tumores.

modelos complexos torna-o útil em uma variedade de domínios, como a medicina, onde é possivel obter uma explicação dos resultados de modelos para a detecção de glaucoma [Volkov and Averkin 2023], ajudando os usuários a entender como e por que os modelos tomam decisões específicas para casos individuais.

2.4. Avaliação da Explicabilidade

A interpretabilidade e explicabilidade desempenham um papel crucial na adoção e confiança em modelos de IA em uma ampla variedade de domínios, desde a medicina, a área jurídica, onde aplicações podem ser utilizadas para tratar de privacidade de dados e legislação [Thommandru et al. 2023] e até na área financeira, com modelos capazes de analisar contramedidas financeiras para o desenvolvimento da economia [Shuguang 2011]. Nesta seção, exploramos como avaliar a explicabilidade de modelos de IA e as técnicas discutidas anteriormente, como Modelos Interpretáveis, Mapas de Calor, Grad-CAM e LIME.

A avaliação da explicabilidade é essencial para determinar a eficácia das técnicas utilizadas e garantir que os resultados sejam confiáveis e úteis para os usuários finais. Várias abordagens podem ser consideradas ao avaliar a explicabilidade de um modelo: Uma maneira de avaliar a explicabilidade é por meio de métricas quantitativas que mensuram o desempenho do modelo na explicação de suas decisões. Por exemplo, para o Grad-CAM e Mapas de Calor, pode-se avaliar a precisão com que essas técnicas identificam as regiões de influência nas imagens em relação às previsões reais. Para o LIME, pode-se medir o quão bem os modelos locais criados se aproximam das previsões do modelo ori-

ginal para as instâncias de interesse. Métricas como precisão, *recall* e erro médio podem ser usadas dependendo da tarefa como classificação ou segmentação. Além das métricas quantitativas, a avaliação qualitativa desempenha um papel crucial na compreensão da explicabilidade. Isso envolve a revisão e interpretação das explicações fornecidas pelo modelo. Os usuários podem avaliar a adequação das explicações à tarefa em questão, verificar se as informações fornecidas são compreensíveis e se correspondem às expectativas. Essa avaliação geralmente envolve especialistas humanos que examinam as explicações geradas pelo modelo.

Uma abordagem útil na avaliação da explicabilidade é comparar o desempenho das técnicas utilizadas com modelos de referência ou *benchmarks*. Isso ajuda a determinar se as técnicas estão fornecendo explicações significativas e superando abordagens padrão. Por exemplo, ao usar o Grad-CAM para a detecção de objetos em imagens, pode-se comparar seu desempenho com métodos tradicionais de segmentação de objetos para verificar se ele fornece *insights* adicionais. Também podemos coletar *feedback* de usuários reais que interagem com o sistema alimentado por modelos de IA para fornecer informações valiosas sobre a eficácia das explicações. Isso pode ajudar a identificar áreas que requerem melhorias e ajustes nas técnicas de explicabilidade para atender às necessidades dos usuários.

A avaliação da explicabilidade deve levar em consideração o contexto e o domínio da aplicação. O que é considerado uma explicação eficaz pode variar dependendo da tarefa. Em algumas situações, uma explicação visual, como um mapa de calor, pode ser mais adequada, enquanto em outras, uma explicação textual ou baseada em regras pode ser preferível. É importante adaptar as técnicas de explicabilidade ao contexto específico. Em resumo, a avaliação da explicabilidade é um componente crítico na implantação de modelos de IA em aplicações do mundo real. Métricas quantitativas, avaliação qualitativa, comparação com modelos de referência, *feedback* dos usuários e consideração do contexto são todos aspectos importantes a serem considerados. A escolha das técnicas de explicabilidade também deve ser orientada pelo objetivo da interpretabilidade em um determinado cenário. Ao adotar uma abordagem abrangente de avaliação, é possível garantir que os modelos de IA sejam mais transparentes, confiáveis e éticos em sua tomada de decisão.

2.5. Aplicações Práticas

As aplicações práticas da IA são tecnologias que utilizam algoritmos e modelos de IA para solucionar uma variedade de problemas cotidianos, seja de forma autônoma ou com assistência, com a finalidade de emular a capacidade cognitiva humana. Estas aplicações operam através da aquisição e análise de dados, empregando o aprendizado automático para efetuar escolhas ou realizar ações com base nas informações disponíveis. Elas têm uma ampla variedade de finalidades e oferecem benefícios em diversas áreas na sociedade.

A IA explicável pode ser definida como aquela que produz informações ou argumentos que tornam seu funcionamento de fácil compreensão. Neste contexto, a interpretabilidade pode ser definida como a capacidade de um modelo de explicar ou fornecer o significado em termos compreensíveis para humanos, afirma [Arrieta et al. 2020]. A interpretabilidade da IA é crucial, visto que a IA está se tornando cada vez mais integrada à

vida cotidiana da sociedade. É fundamental que as pessoas possam entender e confiar nas decisões e ações executadas por esses sistemas. Nesta seção serão apresentados exemplos de aplicações com IA e sua interpretabilidade em diferentes setores.

2.5.1. Aplicação na área da saúde

Na área da medicina, a IA desempenha um papel essencial com diversas aplicações que transformam a prática médica. Uma das aplicações de destaque da IA é a detecção de doenças por meio da análise de imagens médicas, como radiografias e ressonâncias magnéticas. Essa abordagem automatizada é significativamente mais rápida do que os métodos tradicionais, uma vez que os computadores podem processar grandes volumes de dados em um intervalo de tempo muito menor em comparação com um médico humano.

No trabalho proposto por [Ahsan et al. 2020] o LIME é utilizado para identificar os recursos específicos em imagens de radiografia de tórax, as imagens tem duas classes, pacientes com COVID-19 e pacientes sem a doença. A Figura 2.4 apresenta a metodologia usada. Após as imagens de entrada, o LIME é aplicado e encontra o conjunto de *superpixels* que contém a ligação mais válida com um rótulo de previsão. Em outras palavras, o LIME está tentando entender quais partes da imagem são mais influentes para a previsão feita pelo modelo.

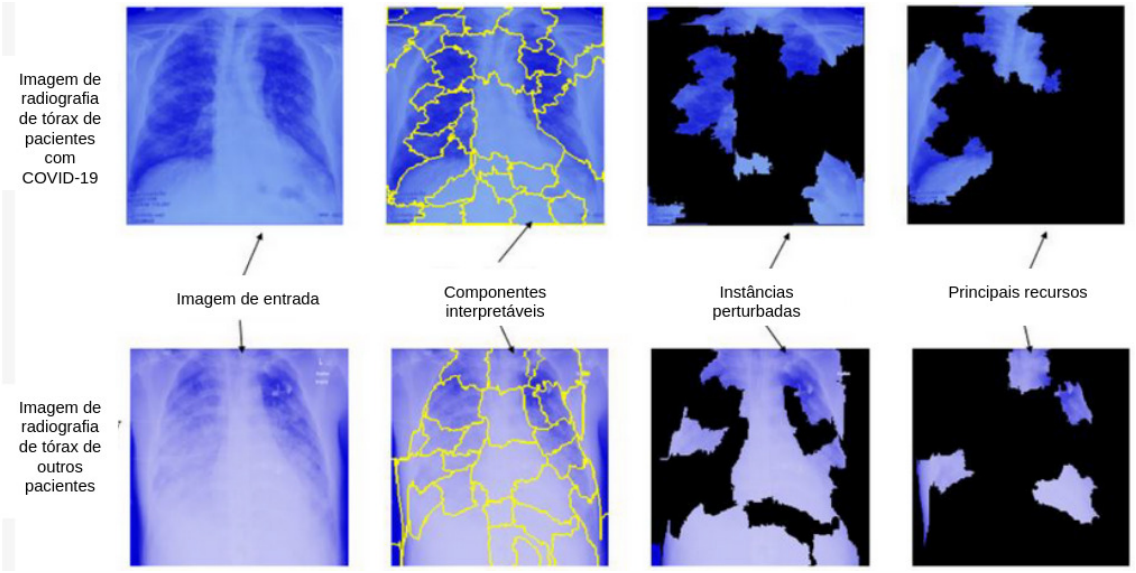


Figura 2.4: Uso do LIME em imagens de radiografia de tórax. Fonte: [Ahsan et al. 2020]

No entanto, a verdadeira eficácia dessa colaboração entre IA e especialistas em saúde reside na interpretabilidade das conclusões. A interpretabilidade reflete a capacidade de fornecer clareza sobre a localização das características indicativas da doença na imagem, permitindo aos médicos uma compreensão precisa e rápida dos resultados. Além da detecção de doenças, a IA na medicina abrange áreas como o desenvolvimento de tratamentos personalizados com base em genômica, a análise de dados de pacientes em larga escala para identificar tendências e aprimorar a gestão de hospitais e clínicas. A interpre-

tabilidade desempenha um papel vital nessas áreas, garantindo que as recomendações da IA sejam compreensíveis e confiáveis para os profissionais de saúde, promovendo, assim, uma assistência médica mais precisa e eficaz.

2.5.2. Aplicação na área da comunicação e marketing

A IA tem desempenhado um papel revolucionário no setor de comunicação e marketing, transformando a maneira como as empresas se relacionam com seus consumidores. Uma profunda análise de grandes conjuntos de informações pode fornecer às empresas valiosas informações sobre o comportamento, preferências e necessidades de seus públicos-alvo, resultando em estratégias de marketing mais eficazes.

A personalização de campanhas publicitárias é outra área em que a IA se destaca. Ao explorar minuciosamente os dados dos consumidores, a IA identifica padrões e características individuais, permitindo que as empresas adaptem suas mensagens de marketing com precisão para atender segmentos específicos do mercado. O resultado são campanhas mais direcionadas e envolventes, estabelecendo conexões mais sólidas entre os consumidores e os produtos ou serviços oferecidos.

A criação automatizada de conteúdo publicitário também desempenha um papel significativo na IA aplicada à comunicação e marketing. A IA tem a capacidade de gerar textos, imagens e vídeos com base em critérios definidos e dados disponíveis, acelerando consideravelmente o processo de produção de conteúdo, ao mesmo tempo em que garante consistência e eficiência na criação de mensagens de marketing. Na figura 2.5, podemos observar uma nuvem de palavras [Jin 2017]. Os modelos de processamento de linguagem natural (PLN) [Iqbal and Qureshi 2022], são bastantes utilizados para a geração de texto. Para entender o que esses modelos estão aprendendo ou levando em consideração é possível utilizar as nuvens de palavras para tentar identificar os padrões utilizados pelos modelos.

Figura 2.5: Nuvens de palavras. Fonte: [Jin 2017]

Em todas essas aplicações, a interpretabilidade da IA desempenha um papel crucial. É fundamental que as empresas compreendam de forma clara como a IA toma suas

decisões, possibilitando ajustes nas estratégias para que estejam alinhadas com seus objetivos e valores. A habilidade de explicar o processo de tomada de decisão da IA promove a transparência e confiança, elementos essenciais para a construção de relacionamentos sólidos com os consumidores e para garantir o sucesso a longo prazo das estratégias de marketing baseadas em IA. Em outras palavras, a interpretabilidade não só aprimora a eficácia da IA, mas também fortalece a conexão entre as empresas e seus públicos, moldando de forma positiva o futuro da comunicação e marketing.

2.5.3. Aplicação na área da agricultura

A agricultura moderna se beneficia da aplicação da IA, uma vez que desempenha um papel essencial na otimização dos recursos agrícolas, tais como irrigação, fertilizantes e pesticidas. A IA opera por meio da coleta de dados provenientes de uma variedade de fontes, incluindo sensores e satélites, permitindo um monitoramento detalhado das condições do solo, do clima, do crescimento das plantas e das pragas. Com base na interpretabilidade desses dados, os agricultores podem adquirir uma visão precisa das necessidades de suas culturas, o que lhes permite tomar decisões com o objetivo de maximizar a produtividade e reduzir o desperdício.

O trabalho de [Zhang et al. 2022] mostra a aplicação do Grad-CAM em imagens da *Spodoptera frugiperda* em uma plantação de milho. A Figura 2.6 apresenta o Grad-CAM da imagem de entrada, a aplicação da menor área retangular convexa, a localização do objeto e a remoção da área que não é de interesse. Podemos notar que o Grad-CAM nesse exemplo trás a cor vermelha para destacar a região de maior ativação do modelo, onde ele identifica que aquela região é a região mais importante para sua previsão, por fim, após o processamento notamos que realmente aquela região é onde apresenta a *Spodoptera frugiperda* na imagem.

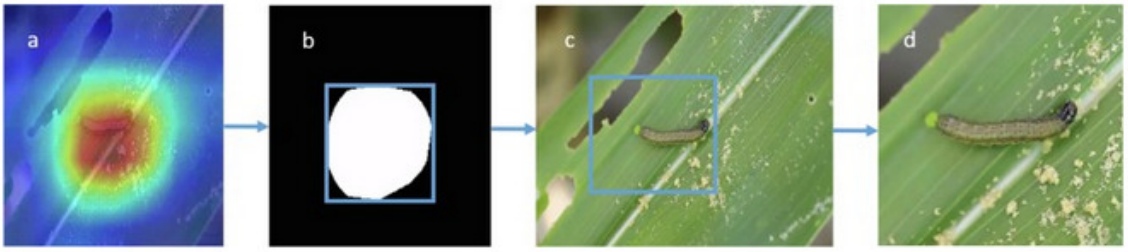


Figura 2.6: IA na agricultura. Ilustração do uso de Grad-CAM na detecção de *Spodoptera frugiperda*. Fonte: [Zhang et al. 2022]. Em a), temos a imagem gerada pelo Grad-CAM, destacando em vermelho a região mais importante para a previsão do modelo. Em b), a imagem segmentada dessa região. Em c), apresenta a *Spodoptera frugiperda* em destaque dentro da região de ativação feita pelo Grad-CAM e por fim, em d), temos a *Spodoptera frugiperda* segmentada e aproximada, mostrando que o modelo realmente é capaz de identificar as *Spodoptera frugiperda* em imagens.

A interpretabilidade da IA na agricultura é essencial, com ela é possivel que os agricultores compreendam de maneira clara e transparente como as recomendações da IA

são geradas. Essa compreensão é de importância vital, pois possibilita que os agricultores ajam de forma confiante em relação aos recursos agrícolas, resultando em uma gestão mais eficaz das operações. Em suma, a interpretabilidade da IA é uma peça fundamental que promove uma agricultura mais eficiente e sustentável, ao capacitar os agricultores a tirar o máximo proveito das vantagens oferecidas por essa tecnologia inovadora.

2.6. Desafios e Limitações

Embora existam diversas técnicas para a explicabilidade e modelos naturalmente interpretáveis, ainda existem alguns problemas relacionados ao entendimento de seus resultados, e que precisam ser levados em consideração antes de sua utilização. Neste sentido, esta seção descreve alguns pontos importantes a respeito desse contexto, tendo em vista a importância de estar ciente dessas limitações antes de utilizar os métodos abordados, pois elas podem afetar diretamente a qualidade e confiabilidade dos resultados.

No estudo conduzido por[Arrieta et al. 2020], os autores apresentam um quadro geral acerca do nível de explicabilidade de alguns modelos naturalmente interpretáveis. Neste sentido, com relação aos modelos de regressão linear/logística, é explicado que, apesar de serem legíveis por humanos e poderem ter sua complexidade reduzida para torná-los mais interpretáveis, suas variáveis e interações podem ser muito complexas para serem analisadas sem ferramentas matemáticas. Além disso, o número de interações entre variáveis independentes pode ser tão grande que seria necessário uma decomposição do modelo em partes menores para ser entendido.

Os mesmos autores ainda explanam acerca das árvores de decisão, onde eles as descrevem como modelos que podem ser facilmente simulados por humanos e oferecem uma explicação clara de como os dados estão sendo interpretados, permitindo um entendimento direto do processo de predição. No entanto, é importante enfatizar que a estrutura desse modelo pode conter muitos nós e regras, dificultando o processo de rastreio de cada decisão, afetando assim interpretação do modelo, até mesmo para especialistas. Por outro lado, as Máquinas de Vetores de Suporte (SVMs) geralmente necessitam que o modelo seja simplificado ou que técnicas de explicação locais sejam aplicadas [Arrieta et al. 2020]. Ainda assim, SVMs com kernel linear, não fornecem uma explicação direta e clara acerca de como os dados são relacionados às variáveis dependentes. Sendo assim, elas somente usam uma reta como limite para classificar os dados com base em um hiperplano de decisão, dado a linearidade de sua natureza.

Os *Heatmaps*, por sua vez, se baseiam em uma estimativa da importância que cada característica, que alimenta o modelo, tem para a composição de sua resposta. Neste sentido, por serem estimativas, tais cálculos podem ser imprecisos devido a fatores relativos a base de dados e ao modelo em si, como por exemplo a complexidade do modelo ou a presença de ruído nos dados de entrada. Outro ponto importante a se considerar é a potencial subjetividade presente na interpretação dos mapas de calor, ou seja, do ponto de vista técnico, a interpretação pode variar dependendo da ótica de quem esteja interpretando os dados exibidos nele.

O *Grad-CAM* consiste em realizar uma análise dos gradientes das ativações das camadas convolucionais da rede neural, com o objetivo de identificar as áreas de maior influência na decisão do modelo. No entanto, é importante reconhecer que esses gradien-

tes, embora relevantes, também estão sujeitos a restrições, como, resolução da imagem, arquitetura da rede neural, camadas convolucionais disponíveis, limitações de interpretação, dependência da qualidade do modelo e dependência da qualidade do modelo. Essas falhas podem surgir devido a características ambíguas nas ativações das camadas convolucionais, a sobreposição de características relevantes e irrelevantes nas imagens e a complexidade do próprio modelo.

O *LIME* de forma semelhante, fornece interpretações compreensíveis para as decisões de modelos de aprendizado de máquina em um nível local. Entretanto, é fundamental destacar os desafios do uso do modelo. Por exemplo, a eficácia do LIME pode depender da escolha adequada de hiperparâmetros e da seleção de instâncias de dados representativas para a explicação. Além disso, as interpretações geradas pelo LIME são aproximações e podem não capturar totalmente o comportamento do modelo original em todas as situações, essa característica pode gerar interpretações ambíguas em alguns cenários, como em modelos altamente não lineares, regiões de decisão complexas, conjuntos de dados desbalanceados, instabilidades nos modelos, modelos de alta dimensionalidade e ruído nos dados de entrada.

Levando em consideração os pontos ressaltados anteriormente, ao utilizar qualquer técnica de interpretabilidade ou modelo naturalmente interpretável, é importante estar ciente das limitações e desafios dessas abordagens. Uma compreensão aprofundada desses elementos possibilita uma avaliação mais precisa da confiabilidade das interpretações geradas. Portanto, a consideração cuidadosa sobre essas limitações desempenha um papel essencial na garantia de que as interpretações resultantes sejam não apenas relevantes, mas também confiáveis para contribuir na compreensão e na tomada de decisões fundamentadas em modelos de aprendizado de máquina.

2.7. Conclusões

Em conclusão, a interpretabilidade e a explicabilidade desempenham um papel fundamental no campo da Inteligência Artificial (IA). À medida que a IA se torna cada vez mais integrada em diversas áreas, desde a saúde até o marketing e a agricultura, compreender como os modelos de IA tomam decisões se torna crucial. Este entendimento não apenas promove a confiança dos usuários, mas também ajuda a evitar resultados indesejados e antiéticos.

No entanto, a busca pela interpretabilidade não é isenta de desafios e limitações. Modelos complexos, como redes neurais profundas, podem ser difíceis de interpretar, mesmo com técnicas de explicabilidade. Além disso, as interpretações geradas por essas técnicas podem ser aproximadas e sujeitas a imprecisões, especialmente em situações de alta complexidade ou com dados ruidosos.

Apesar dessas limitações, é essencial adotar uma abordagem abrangente para avaliar a interpretabilidade da IA em contextos específicos. Métricas quantitativas, avaliação qualitativa, comparação com modelos de referência e o *feedback* dos usuários desempenham um papel importante na avaliação da eficácia das técnicas de explicabilidade.

A interpretabilidade da IA continuará a ser uma área de pesquisa e desenvolvimento importante, à medida que os modelos de IA se tornam mais complexos e integrados

em nossa sociedade. A transparência e a compreensão das decisões de IA são essenciais para construir sistemas responsáveis, éticos e confiáveis, garantindo que a IA beneficie a humanidade em diversos setores e aplicações.

À medida que a tecnologia avança, a interpretabilidade da IA continuará a evoluir, ajudando a resolver desafios complexos e a promover uma tomada de decisão informada em diversas áreas, desde a medicina até a agricultura e além. Com a conscientização das limitações e a busca constante por melhorias, a interpretabilidade da IA continuará a desempenhar um papel crucial no progresso da IA e em sua integração responsável em nossa sociedade.

Portanto, o objetivo deste capítulo foi concluído ao expor de forma conceitual e intuitiva, uma compreensão relativamente ampla do domínio da interpretabilidade e a explicabilidade de IA, orientando assim o público interessado sobre os principais conceitos, técnicas e abordagens no contexto desse capítulo, de forma que não se limitem somente a isto, mas que também procurem outros contextos e formas de se trabalhar dentro deles utilizando todo o conhecimento abordado, sendo isto, a principal contribuição deste capítulo.

Como trabalhos futuros, pretende-se realizar um levantamento do estado da arte a fim de explorar novas técnicas e metodologias para a interpretabilidade e a explicabilidade de IA. Com isso, é possível realizar melhorias neste capítulo, contribuindo para o estado da arte e realizar comparações com as técnicas apresentadas.

Agradecimentos

Este capítulo foi realizado com o apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código Financeiro 001. Agradecemos também ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) e a Fundação de Amparo à Pesquisa do Estado do Piauí (FAPEPI). Todos os códigos podem ser encontrados em https://github.com/Julio-M39/SINFO2022MINICURSO.

Referências

[Jou 1986] (1986). Induction of decision trees. volume 1, pages 81–106. Kluwer Academic Publishers.

[Agarwal and Das 2020] Agarwal, N. and Das, S. (2020). Interpretable machine learning tools: A survey. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1528–1534. IEEE.

[Ahsan et al. 2020] Ahsan, M. M., Gupta, K. D., Islam, M. M., Sen, S., Rahman, M. L., and Shakhawat Hossain, M. (2020). Covid-19 symptoms detection based on nasnetmobile with explainable ai using various imaging modalities. *Machine Learning and Knowledge Extraction*, 2(4):490–504.

[Arrieta et al. 2020] Arrieta, A. B., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., García, S., Gil-López, S., Molina, D., Benjamins, R., et al. (2020). Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information fusion*, 58:82–115.

[Borges et al. 2022] Borges, A. L., Gonçalves, C. d. A., Dias, V. B. L., Sousa, E. A., Costa, C. H. N., and Silva, R. R. V. e. (2022). Visceral leishmaniasis detection using deep learning techniques and multiple color space bands. In *International Conference on Intelligent Systems Design and Applications*, pages 492–502. Springer.

[Caliskan et al. 2017] Caliskan, A., Bryson, J. J., and Narayanan, A. (2017). Semantics derived automatically from language corpora contain human-like biases. *Science*, 356(6334):183–186.

[Chattopadhay et al. 2018] Chattopadhay, A., Sarkar, A., Howlader, P., and Balasubramanian, V. N. (2018). Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 839–847.

[Cortes and Vapnik 1995] Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3):273–297.

[Cox 1958] Cox, D. R. (1958). The regression analysis of binary sequences. *Journal of the Royal Statistical Society: Series B (Methodological)*, 20(2):215–232.

[Forest-gis] Forest-gis. Densidade populacional no brasil – heatmaps. Acesso em 29 set. 2023.

[Gennatas and Chen 2021] Gennatas, E. D. and Chen, J. H. (2021). Chapter 1 - artificial intelligence in medicine: past, present, and future. In Xing, L., Giger, M. L., and Min, J. K., editors, *Artificial Intelligence in Medicine*, pages 3–18. Academic Press.

[Grif and Avush 2018] Grif, M. G. and Avush, Y. (2018). The development of medical diagnostic system based on integration of traditional and eastern medicines. In *2018 XIV International Scientific-Technical Conference on Actual Problems of Electronics Instrument Engineering (APEIE)*, pages 511–515.

[Iqbal and Qureshi 2022] Iqbal, T. and Qureshi, S. (2022). The survey: Text generation models in deep learning. *Journal of King Saud University - Computer and Information Sciences*, 34(6, Part A):2515–2528.

[Jin 2017] Jin, Y. (2017). Development of word cloud generator software based on python. *Procedia Engineering*, 174:788–792. 13th Global Congress on Manufacturing and Management Zhengzhou, China 28-30 November, 2016.

[Lowry and Macpherson 1988] Lowry, S. and Macpherson, G. (1988). A blot on the profession. *British medical journal (Clinical research ed.)*, 296(6623):657.

[Marques et al. 2023] Marques, J. V. M., de Araújo Gonçalves, C., de Carvalho Ferreira, J. F., de Melo Souza Veras, R., de Andrade Lira Rabelo, R., and Veloso e Silva, R. R. (2023). Detection of covid-19 in computed tomography images using deep learning. In Abraham, A., Pllana, S., Casalino, G., Ma, K., and Bajaj, A., editors, *Intelligent Systems Design and Applications*, pages 143–152, Cham. Springer Nature Switzerland.

[Nivas et al. 2016] Nivas, V. M., Krishnan, P. G., and Fredrhic, A. C. (2016). Automated guided car (agc) for industrial automation. In *2016 International Conference on Emerging Trends in Engineering, Technology and Science (ICETETS)*, pages 1–6.

[Pedreshi et al. 2008] Pedreshi, D., Ruggieri, S., and Turini, F. (2008). Discrimination-aware data mining. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, page 560–568, New York, NY, USA. Association for Computing Machinery.

[Pei 2009] Pei, J. (2009). *Association Rules*, pages 140–142. Springer US, Boston, MA.

[Ribeiro et al. 2016a] Ribeiro, M. T., Singh, S., and Guestrin, C. (2016a). "why should i trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 1135–1144, New York, NY, USA. Association for Computing Machinery.

[Ribeiro et al. 2016b] Ribeiro, M. T., Singh, S., and Guestrin, C. (2016b). "why should i trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 1135–1144, New York, NY, USA. Association for Computing Machinery.

[Rudin 2019] Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature machine intelligence*, 1(5):206–215.

[Selvaraju et al. 2017] Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. (2017). Grad-cam: Visual explanations from deep networks via gradient-based localization. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 618–626.

[Selvaraju et al. 2019] Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. (2019). Grad-CAM: Visual explanations from deep networks via gradient-based localization. *International Journal of Computer Vision*, 128(2):336–359.

[Shuguang 2011] Shuguang, W. (2011). Analysis on finance countermeasures of regional economy development in china. In *2011 2nd International Conference on Artificial Intelligence, Management Science and Electronic Commerce (AIMSEC)*, pages 6142–6145.

[Thommandru et al. 2023] Thommandru, A., Mone, V., Mitharwal, S., and Tilwani, R. (2023). Exploring the intersection of machine learning, money laundering, data privacy, and law. In *2023 International Conference on Innovative Data Communication Technologies and Application (ICIDCA)*, pages 149–155.

[Volkov and Averkin 2023] Volkov, E. N. and Averkin, A. N. (2023). Possibilities of explainable artificial intelligence for glaucoma detection using the lime method as an

example. In *2023 XXVI International Conference on Soft Computing and Measurements (SCM)*, pages 130–133.

[Yan and Su 2009] Yan, X. and Su, X. G. (2009). *Linear Regression Analysis: Theory and Computing*. World Scientific Publishing Co., Inc., USA.

[Zhang et al. 2022] Zhang, H., Zhao, S., Song, Y., Ge, S., Liu, D., Yang, X., and Wu, K. (2022). A deep learning and grad-cam-based approach for accurate identification of the fall armyworm (spodoptera frugiperda) in maize fields. *Computers and Electronics in Agriculture*, 202:107440.

Capítulo 3

Desenvolvimento de Jogos 2D de Plataforma: Explorando Unity e Chat GPT para Criação de Códigos Dinâmicos

Héder Pereira Rodrigues Silva, Carlos Heitor Marques da Silva Santos, Isabele Rodrigues de Souza, Artur Guilherme Silva Caldas, Iallen Gábio de Sousa Santos

Abstract

As technology advances, game development has become more accessible, allowing enthusiasts and independent developers to build their own virtual worlds. In addition to large companies, online tools like free-access game engines have democratized game development. In this course, we will focus on the Unity platform, a powerful game development engine for 2D and 3D games. We will explore how the artificial intelligence of Chat GPT can expedite the code creation process, freeing up time for creativity. We will discuss fundamental game development concepts, development tools, the C# programming language, and create a case study for developing a platform game.

Resumo

À medida que a tecnologia avança, a criação de jogos se tornou mais acessível, permitindo que entusiastas e desenvolvedores independentes construam seus próprios mundos virtuais. Além das empresas de grande porte, as ferramentas online, como as engines de acesso gratuito, democratizaram o desenvolvimento de jogos. Neste curso, focaremos na plataforma Unity, uma poderosa engine de desenvolvimento de jogos em 2D e 3D, e exploraremos como a inteligência artificial do Chat GPT pode acelerar o processo de criação de códigos, liberando tempo para a criatividade. Discutiremos os conceitos fundamentais do desenvolvimento de jogos, ferramentas de desenvolvimento, linguagem de programação C# e criaremos um estudo de caso para desenvolver um jogo de plataforma.

3.1. Introdução

Segundo o [SX group 2023], Pesquisa Game Brasil(PGB), de 2023, “Jogar jogos digitais é uma das mais importantes formas de diversão dentre os costumes de entretenimento brasileiro”. A diversão dos jogos está enraizada na variedade de emoções que eles podem evocar, desde a satisfação de superar obstáculos e resolver quebra-cabeças até a adrenalina da competição em tempo real. Os games oferecem uma variedade de experiências que despertam a imaginação, conhecimento e proporcionam entretenimento. Além disso, oferecem uma sensação de realização à medida que os jogadores progridem, ganham recompensas e alcançam objetivos dentro do jogo. Essa combinação de desafio, imersão e recompensa contribui para a diversão duradoura que os games proporcionam e torna esta forma de entretenimento cativante para pessoas de todas as idades.

O avanço da tecnologia, tem tornado a criação de seus seus próprios mundos virtuais mais acessível para entusiastas e desenvolvedores independentes. A 10ª edição da Pesquisa Game Brasil [SX group 2023], afirma no seu levantamento anual, visto na figura 3.1 consolidado sobre o consumo de jogos eletrônicos no país, que cerca de 70,1% afirmam ter o hábito de jogarem, jogos eletrônicos. O desenvolvimento de jogos eletrônicos tem algumas particularidades em relação ao desenvolvimento tradicional de software. Este fator é intensificado pela integração com o fator artistico impermeado neste meio.

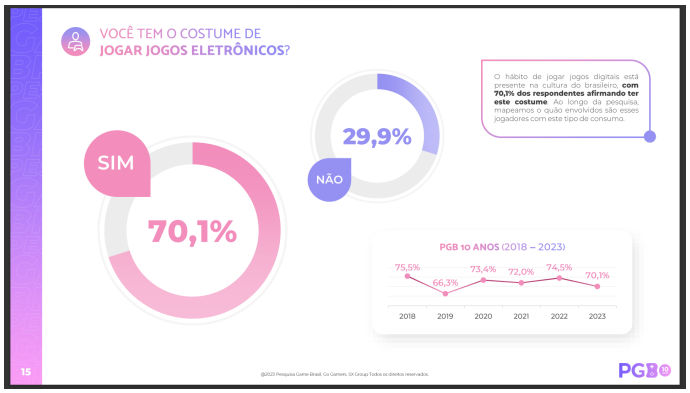


Figura 3.1. Consumo dos jogos eletrônicos, PGB 10 anos. [SX group 2023]

Os jogos desempenham um papel de relevância, abrangendo uma ampla gama de áreas como: educação, entretenimento, saúde e aplicações profissionais. Na educação, eles podem ser usados em salas de aula e atividades extracurriculares para tornar o aprendizado mais envolvente. Na saúde, os jogos têm sido aplicados para melhorar a atividade física e auxiliar no tratamento de distúrbios mentais. No campo profissional, servem como ferramenta de treinamento e até mesmo como estratégia de marketing. Finalmente, o entretenimento é uma das principais aplicações dos jogos, abrangendo desde passatempos casuais até competições globais online e experiências imersivas com narrativas envolventes.

Os videogames abrangem uma vasta variedade de gêneros e estilos, cada um com suas próprias características e desafios distintos. Por exemplo, jogos de estratégia desafiam a mente enquanto os jogos de tiro em primeira pessoa testam os reflexos do jogador. O mundo dos jogos oferece algo para todos os gostos.

Tendo em vista isso, os jogos de plataforma são destacados neste capitulo, uma vez que geralmente apresentam controles simples, com movimentos básicos, como pular e correr. Isso os torna acessíveis a uma ampla gama de jogadores, incluindo iniciantes e jogadores casuais. Mas fora isso, você basicamente se deslocava pela fase, coletava itens e pulava em blocos ou desviava de obstáculos. Ainda assim, à medida que o tempo progrediu, novas aptidões e ferramentas começaram a surgir, evoluindo desde a pré-renderização (também chamado 2,5d) até o total. Alguns jogos de plataforma 2d entraram para a história alcançando sucesso comercial e um grande número de fãs dando origem a franquias que duram até hoje. Entre eles podem ser destacados: Sonic[SEGA 1991], Mario [Nintendo 1983], Mega Man [Nintendo 1987], como visto na Figura 3.2 e tantos outros. Estes jogos são marcados por jogabilidade simples e desafiadora.

Figura 3.2. Cena de Batalha do jogo Mega Man. [Nintendo 1987]

A simplicidade pode ser o segredo para criar jogos divertidos e com uma elevada taxa de repetição. O ponto a considerar é que os jogos de plataforma estão presentes, mesmo após décadas desde o seu surgimento, e novos títulos continuam a ser lançados ano após ano, cativando a preferência do público, o que demonstra que esse gênero ainda é atraente e que uma fórmula pode ser ajustada, mantendo a mesma essência original, sem se tornar obsoleta. Adicionalmente, a indústria de jogos se transformou em uma potência financeira, movimentando bilhões de dólares e gerando oportunidades empolgantes de carreira para desenvolvedores talentosos.

Desde que emergiram, há algumas décadas, os videogames representam diversão. No entanto, para além daquele público jovem que historicamente se vinculou a essa forma de entretenimento, existe hoje uma audiência adulta que também está envolvida nesse passatempo. Nesse sentido, tal setor gera aproximadamente US$ 200 bilhões por ano, conforme estimativas da Newzoo, plataforma de informações sobre o universo dos jogos. [Michelle Rouhof 2023]. O desenvolvimento de jogos não é mais apenas uma forma de entretenimento; tornou-se um campo lucrativo e multifacetado, com aplicações que se estendem por uma ampla gama de áreas, incluindo educação, saúde e entretenimento.

No cerne desse crescimento, destacam-se as empresas que desempenham um papel vital na evolução da indústria. No ano fiscal de 2020/2021, a Sony, uma das empresas

Tabela 3.1. Ranking de empresas no mercado dos jogos e seus respectivos valores em outubro de 2023.

Empresa	Valor de mercado
Apple	US$ 3 Trilhões
Microsoft	US$ 2,63 Trilhões
Alphabet	US$ 1,53 Trilhão
Amazon	US$ 1,35 Trilhão
Meta, Inc.	US$ 780,2 Bilhões

notáveis na indústria de jogos, bateu recordes de receita e lucro na divisão de Games e Serviços de Internet, que inclui a PlayStation. De acordo com o seu último relatório financeiro, a empresa arrecadou impressionantes US$ 25,04 bilhões e teve um ganho de US$ 3,22 bilhões nessa área[Sony 2021]. Esses números destacam a robustez do mercado de jogos e a importância das grandes empresas na condução desse setor em constante crescimento. Suas vastas operações abrangem uma ampla variedade de gêneros e plataformas, abrindo oportunidades para uma indústria que continua a florescer.

A Tabela 3.1 apresenta um ranking de algumas das principais empresas no mercado de tecnologia e jogos, acompanhadas de seus respectivos valores de mercado em bilhões de dólares americanos, especificamente relacionados aos lucros obtidos no segmento de jogos dessas indústrias. A Apple lidera a lista com um valor de mercado de US$ 3 trilhões, seguida pela Microsoft com US$ 2,63 trilhões, a Alphabet (empresa-mãe da Google) com US$ 1,53 trilhão e a Amazon com US$ 1,35 trilhão. Além dessas quatro gigantes, a Meta, Inc. também está presente na lista, com um valor de mercado de US$ 780,2 bilhões.

A tabela apresenta uma coluna mostrando

O notável crescimento da indústria de jogos é um testemunho não apenas da sua capacidade de gerar lucros significativos para as empresas envolvidas, mas também da sua versatilidade e capacidade de impactar positivamente várias áreas da sociedade. No entanto, não são apenas as grandes corporações que se beneficiam desse crescimento.

A disponibilidade de ferramentas online, como as engines de acesso privado e gratuitas, que oferecem um conjunto de recursos e ferramentas para simplificar tarefas como gráficos, física e interatividade no desenvolvimento e execução de jogos de vídeo, democratizou o desenvolvimento de jogos, tornando-o acessível a um público mais amplo. A exemplo disso, temos plataformas como Unity [Unity Technologies 2023], Game Maker [Mark Overmars 2023], Unreal Engine[Epic Games 2023] e outras têm desempenhado um papel crucial nessa revolução. Elas tornaram a criação de jogos de alta qualidade mais acessível do que nunca, estimulando a inovação e a diversidade no setor. No entanto, neste minicurso, concentramos nossa atenção especificamente na Unity com o auxílio do Chat GPT, explorando como essas ferramentas podem colaborar para criar jogos dinâmicos de plataforma.

A Unity, em particular, destaca-se como uma plataforma de desenvolvimento de jogos e aplicativos em 2D e 3D. Ela oferece um ambiente de criação flexível e intuitivo, permitindo que programadores e artistas colaborem na criação de experiências interativas

para várias plataformas, incluindo PC, consoles, dispositivos móveis e realidade virtual. Com recursos avançados em gráficos, física realista, suporte a várias linguagens de programação e uma loja de ativos, a Unity é uma escolha popular para desenvolvedores em todo o mundo.

O Chat GPT, ou *Generative Pre-trained Transformer for Chat*, representa uma inovação significativa em inteligência artificial. Treinado em grandes volumes de texto, esse modelo de linguagem é capaz de compreender o contexto das conversas e gerar respostas coerentes e relevantes, tornando-o ideal para aplicações de chatbots, assistentes virtuais e enriquecimento da narrativa em jogos. Sua versatilidade e capacidade de entender a linguagem natural o tornam uma ferramenta valiosa em uma ampla variedade de campos, impulsionando a interatividade e a automação de tarefas relacionadas à geração de texto.

Neste minicurso, exploraremos como utilizar o Chat GPT como uma ferramenta para a criação de códigos, simplificando e acelerando o processo de desenvolvimento de jogos. pois a AI automatiza tarefas complexas, gerando códigos personalizados com base nas necessidades do desenvolvimento, assim, liberando tempo e recursos para se concentrar na criatividade e na jogabilidade.

Em conclusão, o notável crescimento da indústria de jogos é um testemunho não apenas da sua capacidade de gerar lucros significativos para as empresas envolvidas, mas também da sua versatilidade e capacidade de impactar positivamente várias áreas da sociedade, e ferramentas de desenvolvimento acessíveis. Com isso, Nos próximos capítulos, abordaremos os conceitos básicos para o desenvolvimento de um jogo, após isso uma visão sobre as ferramentas de desenvolvimento no mercado, então será apresentado o basico da linguagem C# que será utilizada e por fim um estudo de caso da criação de um jogo de plataformal.

3.2. Conceitos Básicos para o Desenvolvimento de Jogos

Diferentemente de outras áreas do desenvolvimento de software, a criação de jogos envolve conceitos específicos e peculiares. Neste capítulo, iremos explorar os fundamentos que servem como alicerce para a construção de videogames. É importante compreender os princípios que definem a essência de um jogo, incluindo a análise das regras que governam seu funcionamento, os objetivos que orientam os jogadores e os desafios que os mantêm imersos na experiência.

Considerando que os videogames são predominantemente utilizados para fins de entretenimento, os aspectos artísticos desempenham um papel essencial na criação de um jogo, incluindo elementos como trilha sonora e artes em 2D ou 3D. Estes aspectos contribuem para a composição de cenários, fases, personagens e outros elementos.

3.2.1. Mecânicas de jogo e Regras

Mecânicas de jogos, também conhecidas como mecânicas de gameplay, são sistemas e regras que definem como um jogo funciona do ponto de vista da interação do jogador. Elas estabelecem as ações que os jogadores podem realizar, as regras que regem essas ações e como essas ações afetam o mundo virtual do jogo. As mecânicas de jogos são

essenciais para criar desafios e os objetivos de um jogo. Aqui estão alguns exemplos das diferentes formas de mecânicas:

- **Definição de Ações Permitidas e Proibidas:** As mecânicas determinam quais ações os jogadores podem ou não realizar durante o jogo. Por exemplo, em um jogo de xadrez, as regras estipulam como cada peça pode se mover e as condições sob as quais uma peça pode ser capturada.
- **Estrutura de Turnos ou Fases:** Uma mecânica amplamente adotada é o sistema de batalhas em turnos, como pode ser visto na figura 3.3. Nesse sistema, os jogadores têm a oportunidade de escolher entre várias opções, como atacar, defender ou usar itens durante sua vez. Após a execução de suas ações, a vez passa para os inimigos, seguindo essa alternância. Esse sistema coloca ênfase no pensamento estratégico e no planejamento das jogadas, ao contrário da agilidade requerida em jogos de plataforma, por exemplo.

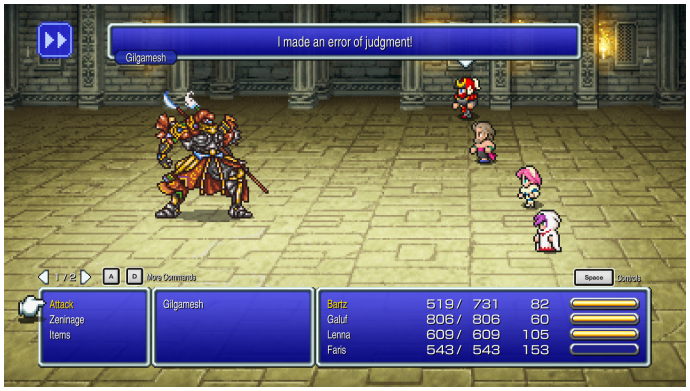


Figura 3.3. Cena de Batalha do jogo Final Fantasy V. [SquareSoft 1992]

- **Condições de Vitória e Derrota:** As regras definem as condições que determinam quando um jogador vence ou perde o jogo. Por exemplo, no jogo de tabuleiro "Catan", as regras estipulam que o primeiro jogador a atingir um certo número de pontos vence.
- **Limites de Recursos e Habilidades:** Muitos jogos têm recursos limitados ou habilidades específicas que os jogadores podem usar. As regras estabelecem como esses recursos são distribuídos e como as habilidades podem ser usadas. Por exemplo, em um jogo de estratégia em tempo real como "StarCraft II"[Entertainment 2010], as regras determinam a quantidade de recursos que os jogadores podem extrair e como as unidades podem ser treinadas e controladas.
- **Movimento:** A grande maioria dos jogos possui ou tem como base o controle de movimento de um personagem, assim, essa mecânica é explorada de varias formas diferentes. Por exemplo, no jogo "Sonic the Hedgehog"[SEGA 1991] o movimento consiste em correr em altas velocidades, enquanto no jogo "Celeste" [Thorson 2018] o foco é em pulos precisos da personagem e na escalada de paredes.

Em resumo, as regras e mecânicas de um jogo trabalham em conjunto para criar uma estrutura que dá forma à jogabilidade e à experiência geral do jogador. Elas estabelecem as diretrizes gerais e detalham como as ações dos jogadores se desenrolam e se conectam para criar a experiência de jogo desejada.

3.2.2. Objetivos

Os objetivos em jogos são as metas que os jogadores buscam alcançar durante a sua experiência de jogo. Essas metas dão propósito à jogabilidade, fornecendo aos jogadores um motivo para se envolverem e progredirem. Estes objetivos podem vir das mais diferentes formas, como por exemplo:

- **Motivação e Direcionamento:** Os objetivos fornecem motivação aos jogadores, dando-lhes um senso de propósito e direção. Por exemplo, em um jogo de aventura como "The Legend of Zelda"[Nintendo 1986], o objetivo principal é resgatar a princesa e derrotar o vilão, proporcionando um propósito claro para o jogador e uma direção para seguir em sua aventura.
- **Variedade de Objetivos:** Os objetivos podem variar amplamente, desde simples tarefas até conquistas complexas. Alguns jogos podem ter múltiplos objetivos ao longo da história. Em um jogo de simulação como "The Sims"[Arts 2000], os objetivos vão desde realizar interações sociais até conquistar metas de carreira e construir uma casa ideal.
- **Recompensas e Reconhecimento:** Ao atingir objetivos, os jogadores frequentemente são recompensados com pontos, itens, poderes especiais ou reconhecimento dentro do jogo. Em um jogo de RPG como "The Elder Scrolls V: Skyrim"[Bethesda 20], os objetivos de missões concedem recompensas valiosas, como equipamentos poderosos e habilidades adicionais. Assim servindo como motivação adicional para o jogador continuar jogando.

Em resumo, os objetivos em jogos são elementos cruciais para manter os jogadores engajados, proporcionar desafios e criar uma sensação de progresso. Eles variam de acordo com o tipo de jogo e podem ser personalizados para atender às necessidades específicas da experiência de jogo desejada.

3.2.3. Desafios

Os desafios desempenham um papel fundamental para tornar a experiência de jogo mais envolvente para os jogadores. Eles são essenciais para enriquecer a jornada em direção aos objetivos, tornando-a mais significativa e recompensadora.

Os desafios, muitas vezes, proporcionam um senso de realização à medida que os jogadores superam obstáculos. Eles podem testar as habilidades e conhecimentos dos jogadores, encorajando-os a desenvolver estratégias e táticas criativas para vencer. Essa superação progressiva dos desafios cria uma sensação de crescimento pessoal dentro do jogo à medida que os jogadores adquirem novas habilidades.

Além disso, os desafios também contribuem para a imprevisibilidade e variedade em jogos. Eles quebram a monotonia, garantindo que cada sessão de jogo seja única.

A incerteza de como um desafio será enfrentado ou resolvido adiciona um elemento de surpresa e emoção à experiência.

No entanto, é crucial que os desenvolvedores mantenham um equilíbrio cuidadoso ao criar desafios. Desafios excessivamente difíceis podem levar a uma experiencia frustrante, enquanto desafios muito fáceis resultam em tédio. Portanto, encontrar o ponto ideal de dificuldade para o jogo é o essencial, para assim manter os jogadores imersos e ansiosos por mais.

aqui estão alguns exemplos de como os desafios são aplicados em diferentes tipos de jogos e situações:

- **Jogos de Plataforma:** Em jogos como "Super Mario"[Nintendo 1983], os desafios frequentemente envolvem a navegação por níveis cheios de obstáculos, como buracos, inimigos e armadilhas, enquanto o jogador tenta alcançar a bandeira no final.
- **Quebra-Cabeças:** Jogos de quebra-cabeças, como "Tetris"[Pajitnov 1984] ou "Candy Crush Saga"[King 2012], desafiam os jogadores a pensar rapidamente e tomar decisões estratégicas para criar combinações ou encaixar peças em padrões específicos.
- **Jogos de RPG:** Em jogos de RPG como "The Witcher 3: Wild Hunt"[RED 2015], os desafios incluem batalhas contra inimigos formidáveis, que requerem planejamento tático, escolha de habilidades e uso de poções para vencer.
- **Jogos de Estratégia em Tempo Real (RTS):** Jogos como "StarCraft II"[Entertainment 2010] apresentam desafios que envolvem o gerenciamento de recursos, construção de bases e coordenação de unidades em batalha, exigindo habilidades estratégicas e táticas.
- **Jogos de Sobrevivência:** Em jogos de sobrevivência como "Minecraft"[Mojang 2011], os desafios incluem encontrar recursos, enfrentar monstros e construir abrigos para sobreviver em um ambiente hostil.

3.3. Ferramentas para o Desenvolvimento de Jogos

Para a criação e desenvolvimento de um jogo, são necessários diversos tipos de software e ferramentas. Essas ferramentas desempenham um papel fundamental, tanto na criação e edição de códigos para os algoritmos que operam dentro do jogo, quanto na criação dos elementos audiovisuais que compõem o videogame. Nesta seção, exploraremos alguns exemplos de recursos amplamente utilizados por desenvolvedores, desde iniciantes até os mais experientes.

3.3.1. Engines de jogos

Uma engine de jogos é uma plataforma de desenvolvimento que fornece um conjunto de ferramentas e recursos para criar jogos de forma mais eficiente. Ela inclui componentes essenciais, como mecanismos de física, renderização, áudio e lógica de jogo, permitindo que os desenvolvedores foquem na criação de conteúdo. As engines de jogos são amplamente utilizadas na indústria para acelerar o processo de desenvolvimento e oferecer

suporte a uma variedade de plataformas. Abaixo, apresentaremos algumas das engines de jogos mais conhecidas e suas características distintas.

- **Unity** é uma das engines de desenvolvimento de jogos mais populares e amplamente utilizadas no mundo. Ela é conhecida por sua versatilidade e suporte multiplataforma, o que a torna uma escolha ideal para desenvolvedores que desejam criar jogos para uma variedade de dispositivos, incluindo PC, consoles, dispositivos móveis e realidade virtual.
- **Gamemaker** é uma engine de desenvolvimento de jogos que ganhou destaque por sua abordagem acessível e amigável para iniciantes. Ela permite que desenvolvedores criem jogos 2D de forma rápida e eficaz, usando uma linguagem de programação chamada GameMaker Language (GML) ou uma interface de arrastar e soltar.
- **Godot Engine** é uma engine de código aberto que ganhou popularidade devido à sua simplicidade e facilidade de uso. É uma opção excelente para desenvolvedores iniciantes, mas também é poderosa o suficiente para projetos avançados. Godot possui seu próprio sistema de script chamado GDScript, que é semelhante a Python.

3.3.2. Artes e Música

Como mencionado na Seção 3.1, os elementos gráficos e sonoros desempenham um papel fundamental nos jogos. A trilha sonora, por exemplo, tem o poder de evocar emoções, criar atmosferas de mundo e fortalecer a narrativa e o ambiente do jogo. Por outro lado, as artes visuais dão vida ao mundo virtual, enriquecendo-o e tornando-o mais imersivo para os jogadores. A ausência desses elementos resultaria em experiências monótonas e desinteressantes. Portanto, o desenvolvimendo de videogames requer habilidades artísticas bem como o domínio de ferramentas para produção destes elementos. A seguir, são apresentadas algumas ferramentas que podem ser utilizadas para o a criação desses elementos.

- **Softwares para música:**
  - **Ableton Live:** O Ableton Live é um software de produção musical amplamente usado tanto em estúdio quanto em performances ao vivo Ele oferece uma ampla gama de instrumentos virtuais e efeitos em tempo real, assim também sendo eficiente para a criação de trilha sonora e efeitos de ambiente.
  - **FL Studio:** também conhecido como Fruity Loops, é um software de produção musical versátil apreciado por sua interface amigável. É uma escolha popular para iniciantes e produtores de música eletrônica devido à sua flexibilidade.
- **Softwares para desenho 2d/3d**
  - **Adobe Photoshop:** O Photoshop é uma ferramenta versátil amplamente utilizada para a criação de artes 2D. Ele oferece recursos avançados de edição de imagem, pintura digital e ilustração. Pode ser usado para criar personagens, cenários e elementos visuais em vários estilos artísticos, desde realismo até ilustrações estilizadas e pixel art.

– **Piskel:** O Piskel é um software focado em pixel art, tornando-o uma escolha ideal para artistas que desejam criar gráficos pixelados, comuns em jogos retrô e indie. Ele possui uma interface simples e ferramentas específicas para desenhar e animar sprites em pixel art.
– **Blender:**O Blender é uma ferramenta de modelagem e animação 3D. Embora seja mais conhecido pela criação de modelos tridimensionais, também é usado para criar ativos 3D para jogos, como personagens, objetos e ambientes. Oferece recursos avançados de modelagem, texturização e animação.

Por fim, tendo em vista as ferramentas demonstas, neste minicurso estaremos focando na plataforma Unity como a engine de desenvolvimento de jogos principal para nossos projetos práticos. Além disso, para auxiliar na criação de códigos e fornecer suporte adicional, também utilizaremos o chat GPT.

3.4. Sintaxe Básica de C#

O C# (pronunciado como "C sharp") é uma linguagem de programação versátil e amplamente usada desenvolvida pela microsoft. Uma vez que este minicurso se propõe a introduzir o desenvolvimento de jogos com a Unity, é importante conhecer a sintaxe básica de C# pois esta é a linguagem padrão de desenvolvimento nesta Engine.

As sintaxes são as regras que governam a estrutura gramatical e a forma correta de escrever código. À medida que exploramos esses conceitos básicos de sintaxe, também mencionaremos como eles são usados no contexto da programação com unity, oferecendo uma visão geral dos fundamentos necessários para começar o desenvolvimento de jogos.

3.4.1. Estrutura basica de C#

Todo programa C# começa com uma estrutura básica, essa estrura é a base pra cirar programas em C#. Na Unity, esta estrutura é criada automaticamente e não é alterada ou gerenciada pelo desenvolvedor de maneira direta. O Algoritmo 3.1 apresenta esta estrutura básica de um programa em C#.

Algoritmo 3.1. Estrutura básica de C#

```
1 using System;
2 class program {
3           static void Main(){
4          //Seu codigo aqui
5         }
6 }
```

- "USING SYSTEM"; isso inclui o namespace "SYSTEM", que contém classes e métodos essenciais.
- "CLASS PROGRAM’; declara uma classe chamada "PROGRAM".
- "STATIC VOID MAIN ()"; O ponto de entrada. Todo programa C# começa a ser executado a partir deste metodo.

3.4.2. Váriaveis e tipos de dado

Uma variável em C# permite que você organize e manipule valores, como números, textos ou datas, durante a execução do programa. Os tipos de dados definem o tipo de valor que uma variável pode conter. No Algoritmo 1.2. são apresentados exemplos de declaração de variável em C#.

Algoritmo 3.2. variaveis e tipos de dado

```
int age = 25;
double salary = 1000.50;
string name = "Joao";
bool isactive = true;
```

- "INT"; Tipos de dados para números inteiros.
- "STRING"; Tipos de dados para textos.
- "FLOAT"; Tipos de dados para números de pontos flutuante.
- "BOOL"; Tipos de dados para valores booleanos (verdadeiros ou falsos).

3.4.3. Estruturas Condicionais

As estruturas condicionais em C# permite que você tome decisões em seu programa com base em condições. Ela controla qual parte do código será executada com base em se uma condição é verdadeira ou falsa. O bloco mais comum de estrutura condicional em C# é o if. O algoritmo 3.3 exemplifica a estrutura condicional if/else em C#.

Algoritmo 3.3. Estrutura condicionais

```
if (idade >= 18){
    Console.WriteLine(nome + "E maior de idade.");
}else{
    Console.WriteLine(nome + "E menor de idade.");
}
```

- "IF (idade >= 18)": Aqui, você verifica se a idade é maior ou igual a 18.
- ’ Console.WriteLine(nome + "é maior de idade.")’; : Se a condição do if for verdadeira, esta linha será executada e imprimirá no console "João é maior de idade."
- "ELSE": Caso a condição do if seja falsa, o código dentro do bloco else será executado.
- Console.WriteLine(nome + "é menor de idade."); : Se a condição do if for falsa, esta linha será executada e imprimirá no console "João é menor de idade."

3.4.4. Estruturas de repetição

As estruturas de repetição em C# (e em outras linguagens de programação) são usadas para executar um bloco de código várias vezes. Isso é útil quando você precisa realizar tarefas repetitivas ou interar sobre uma coleção de dados. Existem várias estruturas de repetição em C#, dentre elas, "*For*"e "*while*",como visto no algoritmo 3.4

Algoritmo 3.4. Estrutura de repetição

```
for (int i = 0; i < 5; i++){
    // Codigo a ser repetido
}
while (condicao){
    // Codigo a ser repetido enquanto a condicao for
       verdadeira
}
```

- “FOR”: Usado para criar um loop com um computador.
- “WHILE”: Cria um loop enquanto uma condição for verdadeira.

3.4.5. Arrays

Um array em c# é uma estrutura de dados que armazena uma coleção de elementos do mesmo tipo em uma sequência unidimensional. Esses elementos são acessados por meio de índices numéricos, onde cada elemento tem um índice que representa sua posição na sequência. O primeiro elemento de um array geralmente tem um índice de 0, o segundo tem um índice de 1, o terceiro tem um índice de 2 e assim por diante, seguindo uma sequência numérica crescente.

Algoritmo 3.5. Arrays

```
int[] numeros = new int[5];
numeros[0] = 10;
numeros[1] = 20;
numeros[2] = 30;
numeros[3] = 40;
numeros[4] = 50;
foreach (int numero in numeros){
    Console.WriteLine(numero);
}
```

- "INT [] NUMEROS": declara um array de inteiros chamado numeros com espaço para 5 elementos.
- Os valores são atribuídos aos elementos individuais do array usando índices, como "NUMEROS [0] = 10", "NUMEROS [1] = 20", etc.
- O loop "FOREACH" é usado para interar pelos elementos do array e imprimir cada valor.

3.4.6. Classes, Instâncias, Atributos, Métodos e Herença

As classes são uma parte fundamental da Programação Orientada a Objetos (POO), que é um paradigma de programação que organiza o código em torno de objetos e suas interações. As classes permitem a encapsulação de dados e funcionalidades relacionados, facilitando a reutilização do código e o gerenciamento de complexidade em projetos de software.

Algoritmo 3.6. Classe e POO

```
// Exemplo de declaracao de classe simples
class Pessoa{
    // Campos (variaveis de instancia)
    public string Nome;
    public int Idade;
    // Metodo construtor
    public Pessoa(string nome, int idade){
        Nome = nome;
        Idade = idade;
    }
    // Metodo de exemplo
    public void Apresentar(){
        Console.WriteLine($"Ola, meu nome e {Nome} e tenho
            {Idade} anos.");
    }
}
```

3.4.6.1. Objeto (Instancias)

Um objeto é uma instância de uma classe. Cada objeto possui seu próprio conjunto de campos e pode chamar os métodos da classe.

Algoritmo 3.7. Objeto da Classe Pessoa

```
// Criando objetos da classe Pessoa
Pessoa pessoa1 = new Pessoa("Alice", 30);
Pessoa pessoa2 = new Pessoa("Bob", 25);
// Chamando o metodo Apresentar para cada objeto
pessoa1.Apresentar(); // Saida: Ola, meu nome e Alice e
   tenho 30 anos.
pessoa2.Apresentar(); // Saida: Ola, meu nome e Bob e tenho
    25 anos.
```

- Objeto da Classe Pessoa: 'PESSOA1' é uma instância (objeto) da classe 'PESSOA'.

3.4.6.2. Atributos

Os atributos, também conhecidos como campos, são variáveis que pertencem a uma classe e representam o estado ou as características dos objetos dessa classe, como foi visto no código de classe. 3.6

3.4.6.3. Métodos

Os métodos são funções que pertencem a uma classe e definem o comportamento ou as ações que os objetos dessa classe podem realizar.

Algoritmo 3.8. Métodos da Classe Calculadora

```
class Calculadora{
    // Metodo que adiciona dois numeros inteiros
    public int Somar(int a, int b){
        return a + b;
    }
}
```

- Neste exemplo, criamos um objeto 'CALCULADORA' da classe 'CALCULADORA' e chamamos seu método 'SOMAR()' para calcular a soma de dois números.
- 'SOMAR(INT a, INT b)' é um método público da classe 'CALCULADORA' que aceita dois argumentos (números inteiros 'A' e 'B') e retorna a soma deles.

3.4.6.4. Herança

A herança é um conceito fundamental na programação orientada a objetos (POO) que permite que uma classe (chamada de classe derivada ou subclasse) herde os atributos e métodos de outra classe (chamada de classe base ou superclasse). Isso promove a reutilização de código e estabelece uma relação hierárquica entre as classes.

Algoritmo 3.9. Herança da Classe "Animal"

```
class Animal{
    public void Respirar(){
        Console.WriteLine("O animal esta respirando.");
    }
}

class Cachorro : Animal{
    public void Latir(){
        Console.WriteLine("O cachorro esta latindo.");
    }
}
```

- Neste exemplo, a classe 'CACHORRO' herda da classe 'ANIMAL'. Isso significa que um objeto da classe 'CACHORRO' terá acesso ao método 'EMITIR SOM'() da classe 'ANIMAL'. Além disso, a classe Cachorro tem seu próprio método 'LATIR'() e substitui (sobrescreve) o método 'EMITIR SOM'(), fornecendo uma implementação específica para cães.

3.5. Estudo de Caso: Desenvolvendo seu primeiro jogo com o auxílio da unity e chat gpt.

Como citado anteriormente, a Unity é uma engine de game design utilizada por várias empresas e desenvolvedores independentes. Além disso, a unity possui e disponibiliza milhares de recursos e tutoriais gratuitos aos desenvolvedores. Utilizando-a em conjunto com o Chat GPT, inteligência artificial mundialmente conhecida por sua facilidade de compreensão e possibilidade de recursos disponibilizados, é possível a criação de milhões de jogos digitais.

3.5.1. Preparação do Ambiente de Desenvolvimento

Nesta seção, serão abordados os preparos necessários para criação do jogo.

Inicialmente, para o desenvolvimento de jogos utilizando a Unity & ChatGPT, é necessário fazer o download da plataforma de desenvolvimento unity, acessando o site oficial da plataforma [Unity Technologies 2023] Com a Unity Hub instalada, é necessário fazer o download do software de desenvolvimento. De acordo com o site oficial da Unity [Unity Technologies 2023], os requisitos de sistema mínimos para utilização da unity são:

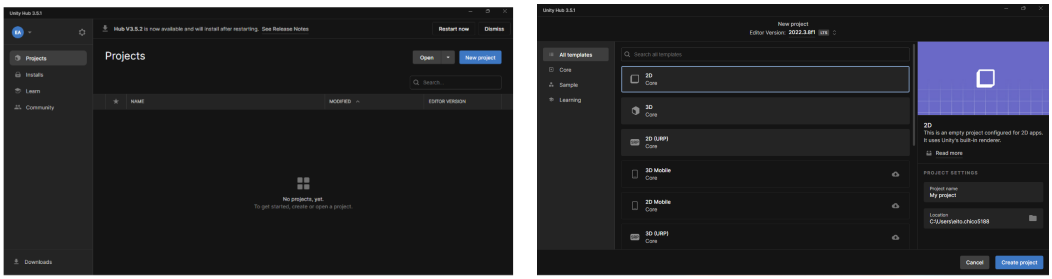
- OS: Windows 7 SP1+, 8, 10, 64-bit versions only; macOS 10.12+; Ubuntu 16.04, 18.04, and CentOS 7.
- GPU: Placa gráfica com recursos do DX10 (Shader Model 4.0).

Obs: No projeto utilizado para criação deste, foi utilizado a Unity versão 2023.3.8f1.

A unity requisita uma interface de programação, algum programa voltado para programação. Um dos mais famosos ,e que foi escolhido para este projeto é o Visual Studio Code. Para fazer seu download, acesse o site oficial da plataforma [Microsoft 2023].

Outra ferramenta necessária para o desenvolvimento será o Chat GPT, que como citado anteriormente, é um chatbot disponível online com capacidade para criar narrativas, orientar no uso de ferramentas (inclusive da unity) e gerar código em c#. Estas características fazem com que o chat GPT seja uma importante ferramenta auxiliar no desenvolvimento de jogos com a unity, especialmente para desenvolvedores iniciantes. Para acessar o Chat GPT, faça login na plataforma online, acessada pelo site oficial [OpenAI 2023].

Agora, para finalziar esta etapa preparatória, é necessária a criação do projeto do jogo na unity. para cria-lo, acesse o unity hub, localize o botão *New Project*, na aba de projetos, vista na figura 3.4(a). Na aba New Project (figura 3.4(b), selecione a opção *2D Core*, renomeie-o como preferir e clique no botão *Create Project*, assim, será criado seu projeto na unity.



(a) Aba Projects na Unity (b) Criação de um novo Projeto

Figura 3.4. Abas Project e New Project na Unity.

Ao fim deste processo de criação, será apresentada a interface de desenvolvimento básica da unity, vista na figura 3.5. Logo mais, a mesma será explicada.

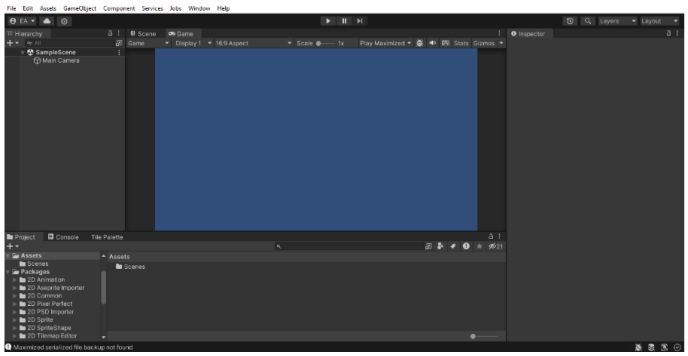


Figura 3.5. InterFace Básica de Desenvolvimento

3.5.2. Interface básica da Unity.

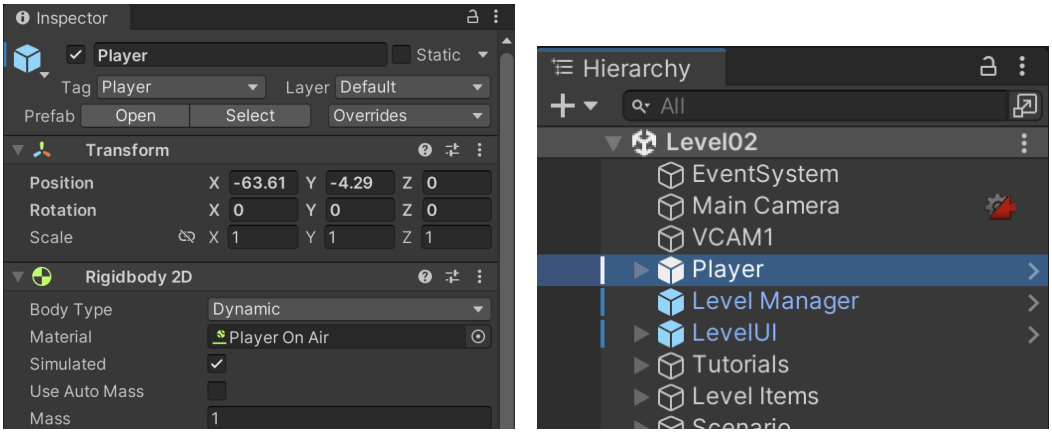
Já com as plataformas prontas, é indispensável a apresentação da interface básica de desenvolvimento da Unity.

3.5.2.1. Inspector

Geralmente localizada no lado direito do monitor, a janela inspector aborda as características e os atributos sobre os objetos. É nela onde são apresentadas as opções de configuração de um objeto dentro da Unity. Como pode-se ver na figura 3.6(a), o objeto “Main Camera”, possui os componentes “Transform”, “Camera” e “Audio Listener”.

3.5.2.2. Hierarchy

Nesta janela, localizada no lado esquerdo do visor, são organizados os objetos presentes na “cena”, tudo aquilo pertencente ao jogo, como players, objetos, paredes, etc. Os objetos podem possuir sub-objetos, que são organizados e separados nesta janela, como é visto na figura 3.6(b).

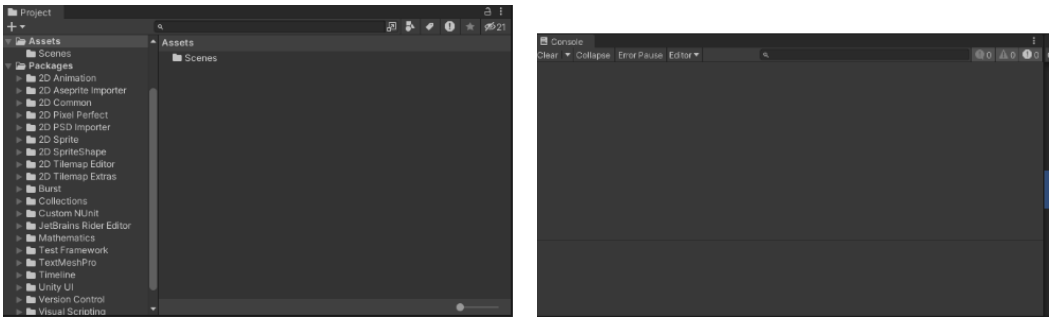


(a) Inspector (b) Hierarchy

Figura 3.6. Janelas Inspector e Hierarchy da Unity.

3.5.2.3. Project

Nesta janela, estão localizados os arquivos do jogo, sendo possível acessar/escolher arquivos. Ao lado pode-se ver na figura 3.7(a) que os arquivos ficam organizados em pastas(ficheiros), tornando mais fácil o acesso



(a) Project (b) Console

Figura 3.7. Janelas Project e Console da Unity.

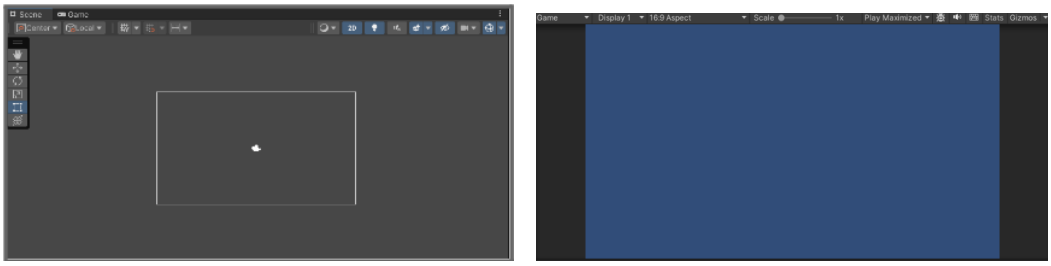
3.5.2.4. Console

A janela console (Figura 3.7(b)), geralmente localizada no canto inferior esquerdo da interface, tem a função de mostrar ao usuário os erros existentes em seu projeto. Caso algum script tenha uma sintaxe errada, algum erro de ortografia, por exemplo, nesta janela terá uma mensagem.

3.5.2.5. Scene

Em todo projeto, é necessário um local para organizar, posicionar, e visualizar objetos das cenas de um jogo. Na unity, esta janela é chamada de Scene (figura 3.8(a)). Nela,

podemos alterar características dos objetos, como suas posições, escalas, dentre outras funções



(a) Scene (b) Game

Figura 3.8. Janelas Scene e Game da Unity.

3.5.2.6. Game

A janela Game possui a função de simular como o jogo ficará,, a partir do que já estiver produzido (figura 3.8(b)).

3.5.3. Tutorial: criando seu primeiro jogo de plataforma 2d com unity e chat GPT

Neste tutorial, será criado um jogo de plataforma 2d que consiste em um player com capacidade de movimentar-se para frente e para trás e pular. O Objetivo do jogo é alcançar o final da fase. Neste minigame serão trabalhados 3 objetos, o chão, o player e o destino.

Na unity, o jogo é representado por uma cena contendo objetos. Estes objetos contêm componentes que representam características do objeto como: posicionamento, detector de colisões, corpo físico, elemento gráfico, elemento sonoro, etc. Além disso, é possível adicionar scripts para acessar estas propriedades e controlar os objetos na cena.

Para criá-los, com o projeto aberto, acesse o menu de contexto (geralmente acessado clicando no botão direito do mouse) na janela de “Hierarchy”, e procurar o menu *create 2D Objects*, em seguida, selecione o sprite desejado, como é visto na figura 3.9. (Utilizado no projeto: circle):

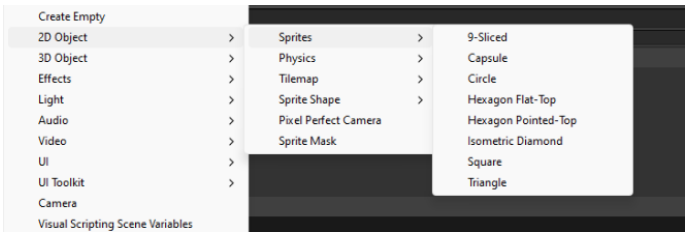


Figura 3.9. Criação do sprite Circle para representar o Player

Com ele criado, percebe-se que nas janelas Hierarchy e Scene foi adicionado um novo objeto, como o nome de Circle. Na Hierarchy, abrindo o menu de contexto, tendo selecionado este objeto, é possível renomeá-lo (recomendável o nome “Player”). Na janela de Inspector (com o objeto do player selecionado) adicione componentes a ele, para isto,

clique no botão “Add Component”, e procure o atributo “RigidyBody2D”, para que o objeto tenha a característica de corpo rígido, com física e etc. Também adicione um atributo “Collider2D”, que possui a função de indicar que o objeto estabelece colisão com outros objetos, tendo as opções de BoxCollider2D e CapsuleCollider2D (neste caso, é utilizado o capsule collider). Com isto, a parte física do player está feita Agora, é preciso que crie o objeto das plataformas, seguindo a mesma lógica. Crie o objeto de “Square” em Hierarchy, adicione RigidyBody2D e desta vez, BoxCollider2D. Assim também o objetivo deve ser criado, utilizando o sprite de “triangle”. Lembre-se de renomeá-lo No objeto da plataforma, acesse o componente RigidyBody2D, e na opção “Body Type”, troque para Static, assim, quando a fase for iniciada, o objeto ficará estático e não sofrerá força gravitacional. Com os objetos criados, pode-se posicionar da forma que for desejado. Também é possível duplicar a plataforma já com seus atributos, acessando o menu de contexto e duplicando. Para a parte lógica do player, inicialmente deve-se criar um script de controle do player. Crie um folder(menu de contexto, “Create”, “Folder”) na janela Project, com o nome de “script”, nela, acesse o menu de contexto, e selecione “Create”, “C# Script”, nomeie-o de PlayerController, pois ele possuirá o algoritmo de locomoção horizontal do player. É necessário associar o script ao objeto do player (para isto, vá na janela inspector do objeto, procure a opção “Add component” e procure o script desejado). Agora, o chat GPT será utilizado para criação de um simples prompt de movimentação horizontal do player. Neste caso, foi utilizado o seguinte prompt: "Olá, preciso de um simples prompt de movimentação horizontal de um player em um jogo de plataforma 2D na unity utilizando C#".

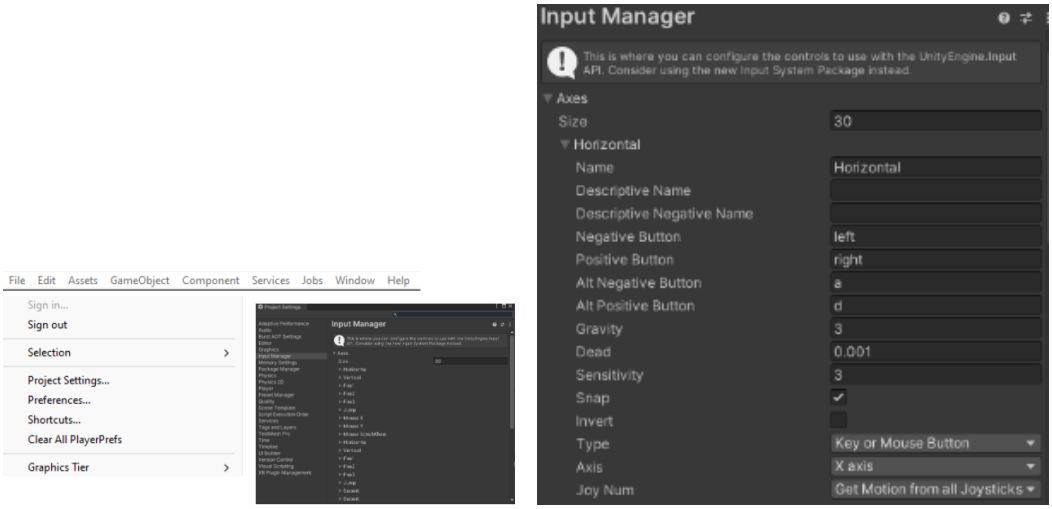
O mesmo respondeu anexando o algoritmo 3.10

Algoritmo 3.10. Movimentação Horizontal

```
using UnityEngine;

public class PlayerController : MonoBehaviour{
    public float speed = 5.0f;
    private Rigidbody2D rb;

    private void Start(){
        rb = GetComponent<Rigidbody2D>();
    }

    private void Update(){
        float moveInput = Input.GetAxis("Horizontal");
        Vector2 moveVelocity = new Vector2(moveInput *
           speed, rb.velocity.y);
        rb.velocity = moveVelocity;
    }
}
```

Antes de tudo é preciso entender que quando criado um projeto, a Unity gera uma série de funcionalidades. Uma delas é a pré-alocação de funções para os botões no teclado. Isto funciona da seguinte forma: Na unity, na barra superior, vá no botão de

“Edit” e procure o menu “Project Settings”. Clicando nele, será aberta uma nova janela com o mesmo nome, com diversas informações, nela, procure no menu esquerdo a opção “Input Manager”, como visto na figura 3.10(a).



(a) InputManager (b) Inputs Horizontais

Figura 3.10. Aba Input Manager e Inputs Horizontais

Neste menu, são apresentadas as configurações de entrada. Em cada um dos tópicos, temos uma pré-configuração de botões. Em horizontal, por exemplo, temos os botões que são chamados para movimentos horizontais, como é observado na figura 3.10(b). Em Positive button, temos a tecla “d” configurada, e em Negative button temos a tecla “a”. Isto significa que quando apertada (dependendo do código utilizado), a tecla “d”, o player andará para frente, e quando apertada “a”, andará para trás.

Com o Visual Studio Code aberto no script desejado, copie do chatgpt o código disponibilizado e cole-o no script. Com o texto pronto, sempre é necessário salvá-lo. Esa função pode ser feita pelo atalho no teclado (Ctrl + S).

Agora, é necessário um prompt para o player, com a função de pular entre as plataformas. Para criá-lo, faça como o anterior, nomeie-o de PlayerJump e associe-o ao player. Neste caso, foi utilizado o seguinte prompt: "Crie agora um script de pulo para o player, utilizando o input "Jump". O código resultante da requisição para o chat GPT é apresentado no Algoritmo 3.11.

Algoritmo 3.11. Atributos do script PlayerJump

```
1 public float jumpForce = 8.0f;
2 private Rigidbody2D rb;
3
4 private void Update(){
5     if (Input.GetButtonDown("Jump"))
6     {
7         rb.AddForce(Vector2.up * jumpForce, ForceMode2D.
              Impulse);
8     }
```

```
}
```

No Algoritmo 3.11, são criados dois atributos: jumpForce e rb. O jumpForce corresponde à força que irá impulsionar o player para cima. Já o rb representa o corpo físico do objeto onde será aplicada a força.

Neste passo, faça como no script anterior, copie, cole, e depois salve-o. Neste ponto, nosso player já se movimenta horizontalmente e já consegue pular. Para testar isto, a Unity disponibiliza a janela game onde é possível executar um *preview* do jogo.

Como em todo jogo, este também necessita de um aviso que o player alcançou seu objetivo. Na unity, isto pode ser feito por meio dos objetos chamados de panels. Para criá-lo, pode-se pedir um passo a passo ao Chat GPT, é propiciado um breve tutorial de como produzi-lo. Com o prompt: "Tenho um objeto que será o objetivo final do player, preciso que apareça na tela do jogador, um painel dizendo que ele ganhou ou jogo quando quando o mesmo colidir com o objetivo, como posso fazer isto?". A IA respondeu anexando as seguintes estruturas de algoritmo:

Algoritmo 3.12. Variáveis do algoritmo de Level End

```
public GameObject victoryPanel;
private bool hasWon = false;
```

No algoritmo 3.12, vemos a parte de variáveis e atributos do script de level end

Algoritmo 3.13. Métodos de Colisão

```
private void OnTriggerEnter2D(Collider2D other){
        if (other.CompareTag("Player") && !hasWon){
            hasWon = true;
            ShowVictoryPanel();
        }
    }
```

Algoritmo 3.14. Algoritmo Level End

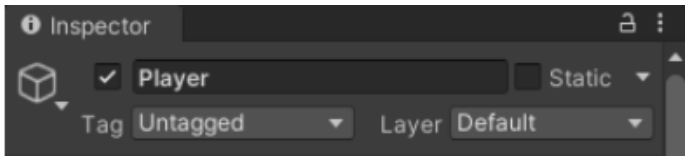
```

 private void ShowVictoryPanel(){
        victoryPanel.SetActive(true);
    }
```

No algoritmo 3.14, observamos o método "ShowVictoryPanel", que ativa o painel de vitória.

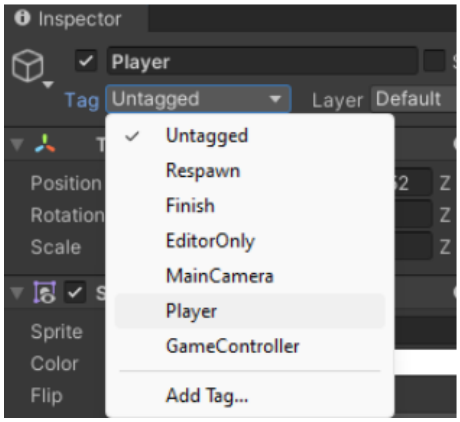
Com estas instruções, resta ainda a criaç ao de alguns componentes. Primeiramente, você precisa criar a UI (User Interface) geral do jogo. Para isto, crie um objeto do tipo canva, visto no menu de criação "UI".

Renomeie-o para UI, e logo após isto, com ele selecionado, crie um novo objeto do tipo *panel*, um pouco acima de canvas. Este painel criado se torna “herdeiro” do canva. No inspector, deve-se desmarcar a caixa ao lado do nome do objeto, assim, o objeto inicia

a fase estando desativado, e só é ativado caso algo seja estabelecido. Agora, com o script "GameManager" ativado, faz-se como nos outros scripts anteriores, todavia, associe este ao objeto de objetivo. Para que este script funcione, é necessário associar a tag player ao objeto player. No inspector do player, selecione a opção "Tag", e nela, escolha a tag player, como é visto nas figuras 3.11(a) e 3.11(b). Isto serve para diferenciar o player de objetos comuns.



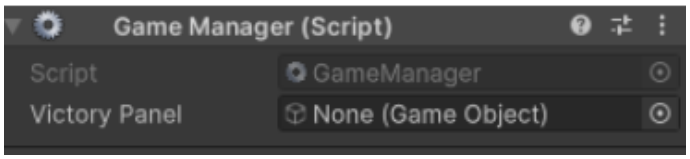
(a) Untagged



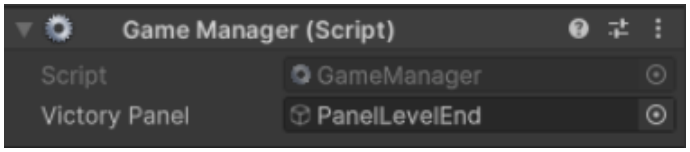
(b) Tag Player

Figura 3.11. Player com a tag "Player"

Além disso, também é necessário atribuir o panel ao gameobject do script. No inspector do objetivo com o script GameManeger, arraste o panel para a aba do componente, nas figuras 3.12(a) e 3.12(b). Verifique se no componente de capsule collider do objetivo, a caixa "Is trigger" está marcada(figura 3.13).



(a) Seleção de Panel Vazia



(b) Painel com PanelLevelEnd selecionado

Figura 3.12. Painel de Seleção de Objeto

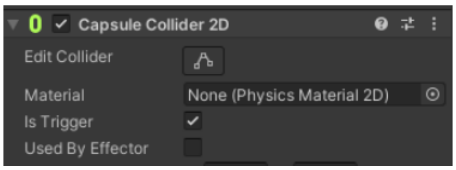
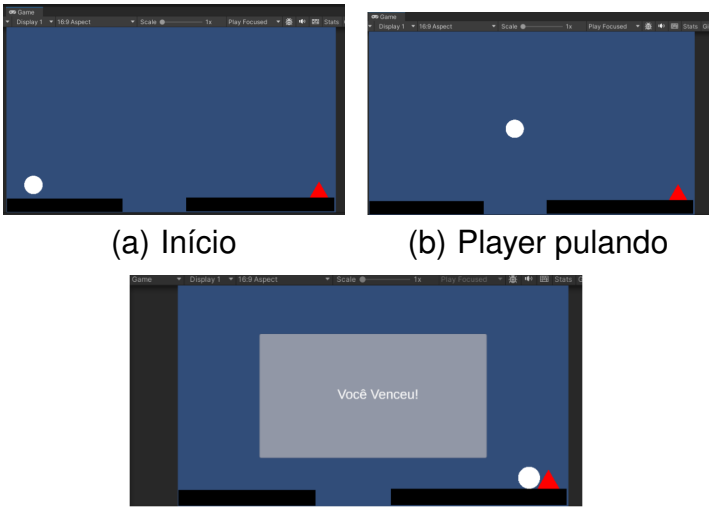


Figura 3.13. Opção IsTrigger



(a) Início (b) Player pulando

(c) Final

Figura 3.14. Capturas de tela do protótipo de jogo de plataforma funcional.

Por fim, é necessário a personalização do painel, como adição de texto, sendo feita por meio do componente adicionável *TextMeshPRO*. Adicione o componente no painel, como feito com os outros componentes adicionados. Na Hierarchy, com o painel selecionado, adicione o componente "TextMeshPRO", localizado no menu *UI*. Para personalizalo, vá ao inspetor do TextMeshPRO adicionado. A Figura 3.14 apresenta as capturas de tela de como o jogo deverá se comportar após a realização dos passsos descritos anteriormente.

3.6. Conclusão

Neste capítulo, foram explorados elementos essenciais que compõem um jogo, como as mecânicas, regras e desafios que o tornam envolvente. Tamém Investigamos as ferramentas e motores de jogo populares usados C#. Ao criar um estudo de caso para um jogo de plataforma, os participantes tiveram a oportunidade de aplicar o conhecimento adquirido em um projeto prático.

Portanto, espera-se que os conhecimentos adquiridos ao longo deste minicurso sejam não apenas valiosos para a compreensão do desenvolvimento de jogos, mas também funcionem como uma porta de entrada para aqueles que desejam ingressar nessa área profissionalmente.

Referências

[OpenAI 2023] OpenAI (2023). Chatgpt. Disponível em: https://chat.openai.com. Acesso em: 04 de outubro 2023.

[Arts 2000] Arts, E. (2000). The sims. Plataformas: Microsoft Windows, MacOS, Linux, PlayStation 2, Nintendo GameCube, Nintendo DS, Xbox, Mac OS Classic.

[Bethesda 20] Bethesda (20). The elder scrolls v: Skyrim. Plataformas: Nintendo Switch, PlayStation 4, Xbox One, Steam.

[Entertainment 2010] Entertainment, B. (2010). Starcraft ii. Plataforma: MacOS, Microsoft Windows.

[Epic Games 2023] Epic Games (2023). Unreal engine. Disponível em: https://www.unrealengine.com/pt-BR. Acesso em: 04 de outubro 2023.

[King 2012] King (2012). Candy crush saga. Plataformas: Android, IOS, Microsoft Windows.

[Mark Overmars 2023] Mark Overmars (2023). Gamemaker. Disponível em: https://gamemaker.io/pt-BR. Acesso em: 04 de outubro 2023.

[Michelle Rouhof 2023] Michelle Rouhof (2023). Newzoo. Disponível em: https://newzoo.com/. Acesso em: 21 de setembro 2023.

[Microsoft 2023] Microsoft (2023). Visual studio code. Disponível em: https://code.visualstudio.com. Acesso em: 04 de outubro 2023.

[Mojang 2011] Mojang (2011). Minecraft. Plataformas: IOS, Android, PlayStation 4, Nintendo Switch, Xbox One, Xbox Series X e Series S, Microsoft Windows.

[Nintendo 1983] Nintendo (1983). Super mario bros. Plataformas: Nintendo Switch, Nintendo Entertainment System.

[Nintendo 1986] Nintendo (1986). The legend of zelda. Plataformas: Nintendo Entertainment System.

[Nintendo 1987] Nintendo (1987). Mega man. Plataformas: Android, iOS, Nintendo Entertainment System.

[Pajitnov 1984] Pajitnov, A. (1984). Tetris. Plataformas: Game Boy, Nintendo Entertainment System.

[RED 2015] RED, C. P. (2015). The witcher 3: Wild hunt. Plataformas: PlayStation 5, PlayStation 4, Nintendo Switch, Xbox One, Xbox Series X e Series S, Microsoft Windows.

[SEGA 1991] SEGA (1991). Sonic the hedgehog. Plataformas: Mega Drive, Android, SG-1000 Mark III, PC, Playstation, Xbox.

[Sony 2021] Sony (2021). Suplemento de informações para os resultados financeiros consolidados para o quarto trimestre encerrado em 31 de março de 2021. Technical report, Sony.

[SquareSoft 1992] SquareSoft (1992). Final fantasy v. Plataforma: Game Boy Advance.

[SX group 2023] SX group (2023). Pesquisa game brasil. Disponível em: `https://www.pesquisagamebrasil.com.br`. Acesso em: 30 de setembro 2023.

[Thorson 2018] Thorson, M. (2018). Celeste. Plataformas: Steam, Epic Games, Xbox, PlayStation, Switch.

[Unity Technologies 2023] Unity Technologies (2023). Plataforma de desenvolvimento em tempo real do unity. Disponível em: `https://unity.com/pt`. Acesso em: 04 de outubro 2023.

Capítulo 4

Git e GitHub: Desenvolvendo Habilidades Essenciais para Colaboração e Controle de Versões

Ruan Victor Carreiro Gomes, Bruno Lima Pinheiro, João Carlos Pinto Carvalho, Jociel dos Santos Andrade e Anderson dos Reis Barros

Abstract

This book chapter thoroughly delves into the realm of Git and GitHub, providing an in-depth overview of these indispensable tools for developing essential collaboration and version control skills in the software development landscape. Starting with fundamental concepts, proceeding through installation and configuration, and emphasizing the significance of teamwork, the chapter aims to empower readers to effectively apply these skills in their development practices.

Resumo

Este capítulo de livro de minicurso explora de forma abrangente o mundo do Git e do GitHub, oferecendo uma visão detalhada dessas ferramentas cruciais para o desenvolvimento de habilidades essenciais de colaboração e controle de versões no cenário do desenvolvimento de software. Iniciando com conceitos fundamentais, passando pela instalação e configuração, e destacando a relevância da colaboração em equipe, o capítulo visa capacitar os leitores a aplicar eficazmente essas habilidades em suas práticas de desenvolvimento.

4.1. Introdução

Este minicurso estabelece o ponto de partida para uma análise abrangente do cenário do desenvolvimento de software, destacando as características do Git e do GitHub. Nesse contexto, a gestão e supervisão das alterações no código-fonte tornam-se tarefas simplificadas. Nessa situação, colaborar em projetos de equipe é análogo a conduzir habilmente uma orquestra, onde divergências são harmonizadas de forma cuidadosa, e cada etapa do processo é minuciosamente documentada. O poder intrínseco do Git e do GitHub concede aos envolvidos total controle sobre essa ampla gama de funcionalidades.

No decorrer deste capítulo aborda-se uma variedade de tópicos, incluindo a administração de código, a colaboração eficaz e o controle de versões, todos os quais são componentes essenciais que desempenham um papel importante no funcionamento das equipes de desenvolvimento. Os participantes adquirem habilidades em uma das linguagens universais dos desenvolvedores de software à medida que incorporam palavras como "commit", "push" e "pull request" em seu vocabulário.

A estrutura compreende as seguintes seções: a Seção 1.2 destaca a importância dos conceitos relacionados ao controle de versão e à colaboração em comunidades de código aberto. Subsequentemente, a Seção 1.3 explora a introdução ao Git, seguida pela Seção 1.4, que aborda o GitHub, ambas sendo ferramentas fundamentais para desenvolvedores. Por fim, a Seção 1.5 encerra o capítulo com algumas considerações finais, consolidando os princípios discutidos.

4.2. A Importância do Controle de Versão e da Colaboração em Comunidades de Código Aberto

Já foi pensado em como grandes projetos de software são desenvolvidos sem se tornarem caos de códigos confusos? Ou como as equipes de desenvolvimento distribuídas podem trabalhar juntas em projetos complexos? O controle de versão é a ferramenta que possibilita a resposta a essas questões.

Considerando as dimensões dos projetos de software modernos, que geralmente envolvem equipes de desenvolvimento distribuídas geograficamente, milhares de arquivos e milhões de linhas de código, surgem como um desafio significativo. Manter o código organizado, garantir a rastreabilidade das alterações e facilitar a colaboração eficaz nessas circunstâncias pode ser extremamente complicado [7]. Nesse contexto, o controle de versão, representado por sistemas robustos como o Git, desempenha um papel fundamental. Não se trata apenas de uma característica agradável, mas de um componente essencial no desenvolvimento de software contemporâneo [1]. O versionamento permite que os desenvolvedores trabalhem em conjunto em um único códigofonte, independentemente de sua localização geográfica. Além disso, garante o registro de cada alteração realizada, documenta as razões por trás dessas mudanças e, o mais importante, oferece a capacidade de reverter facilmente para qualquer ponto no histórico do projeto [9]. Um sistema de controle de versão robusto, como o Git, é o cerne do desenvolvimento de software moderno. Sem ele, os desenvolvedores teriam que recorrer a métodos mais arcaicos, como a cópia manual de versões de código, a criação de pastas com nomes incrementais ou depender da memória humana para monitorar todas as modificações.

Além de melhorar a eficiência, o controle de versão oferece maior segurança ao funcionar como um registro permanente de todas as mudanças no código. Ele fornece um método valioso de auditoria para identificar quando e por quem uma alteração específica foi implementada [10]. Encontrar erros, corrigir problemas e melhorar a qualidade do software dependem disso.

Todas essas vantagens do controle de versão são complementadas por uma característica ainda mais atraente no mundo do desenvolvimento de software moderno: a colaboração em comunidades de código aberto e a participação em uma "rede social" de inovadores [8]. Isso é uma extensão natural do controle de versão, permitindo conectar-se com outros entusiastas, desenvolvedores talentosos e especialistas de diversas regiões enquanto se gerencia o código. Ao registrar-se no GitHub, a maior

plataforma de colaboração e hospedagem de código-fonte do mundo, é possível juntar-se a uma comunidade de desenvolvedores, compartilhar trabalho, contribuir para projetos e aprender com outros usuários [5]. Essa é uma rede onde pessoas de diferentes locais trabalham juntas e geram novas ideias. Afinal, no GitHub, o código é social.

Ao longo do capítulo, serão discutidas as ferramentas Git e GitHub, essenciais para colaboração e controle de versão no desenvolvimento de software. Esta jornada aumentará o conhecimento, habilidades e auxiliará no gerenciamento de código de forma eficiente e eficaz.

4.3. Iniciando com o Git

Para contextualizar o Git, é necessário percorrer as raízes do desenvolvimento de software e a revolução que ele representou. Embora o Git tenha uma origem humilde, sua história é fascinante. Hoje é considerado uma das ferramentas de controle de versão mais poderosas.

No início do desenvolvimento de software, o gerenciamento de versões do códigofonte era uma das tarefas mais difíceis que os programadores enfrentavam. Antes do Git, os sistemas de controle de versão centralizados eram a norma, em que os desenvolvedores se conectavam a um servidor central para trabalhar em conjunto e o repositório principal era mantido por ele [2]. No entanto, esse método tinha limitações significativas, como pontos de falha e lentidão. Ao usar um sistema centralizado, o servidor era o principal componente do controle de versão. Qualquer problema com o servidor, como falhas de hardware, problemas de rede ou erros humanos, pode causar grandes interrupções no fluxo de trabalho de desenvolvimento. Isso pode resultar na perda de acesso ao código-fonte ou, pior ainda, na corrupção de dados, colocando em risco o projeto como um todo. As operações, como a obtenção de versões anteriores do código ou a integração de alterações, eram frequentemente lentas e sujeitas a atrasos devido ao fato de os desenvolvedores precisarem de uma conexão constante com o servidor central. A produtividade e a agilidade das equipes de desenvolvimento foram prejudicadas por isso, principalmente em projetos grandes ou com membros distribuídos. O criador do kernel Linux, Linus Torvalds, começou a desenvolver o Git por conta dessas restrições. Ele viu uma chance de desenvolver uma ferramenta de controle de versão que desafiasse os padrões estabelecidos e resolvesse esses principais problemas.

Com isso, a história[1] do Git começa no início dos anos 2000, quando Linus Torvalds, enfrentou as limitações de sua comunidade de desenvolvimento. Torvalds viu uma oportunidade para criar uma nova ferramenta de controle de versão que desafiaria as normas estabelecidas porque estava insatisfeito com as soluções existentes. Assim, o Git surgiu. Ele começou como um projeto de código aberto em 2005 e tem velocidade e descentralização como seus pilares. O Git chamou a atenção da comunidade rapidamente e ganhou uma base de usuários fiel. No entanto, como muitas histórias notáveis, o Git

[1]https://git-scm.com/book/en/v2/Getting-Started-A-Short-History-of-Git

teve seu início em meio a uma grande destruição criativa e uma intensa disputa. Com sua amplitude e importância no mundo do código aberto, o projeto do kernel Linux trabalhou por anos distribuindo modificações no código por meio de correções e arquivos. Porém, em 2002, o kernel Linux assumiu um caminho diferente ao usar a BitKeeper, uma DVCS (Distributed Version Control System) proprietária.

O conto teve um giro inesperado em 2005. A ruptura da conexão entre a empresa por trás do BitKeeper e a comunidade de desenvolvedores do Linux levou ao pagamento da ferramenta, obrigando a comunidade a procurar uma alternativa. O próprio arquiteto do Linux, Linus Torvalds, sentiu-se compelido a liderar o desenvolvimento de uma nova ferramenta, aproveitando as lições úteis que aprendeu com o BitKeeper. Velocidade, simplicidade, arquitetura completamente distribuída e suporte robusto para desenvolvimento não-linear com milhares de ramos paralelos eram os princípios fundamentais. Essa nova ferramenta deveria ser capaz de lidar com projetos de grande envergadura, como o núcleo do Linux, com eficiência e destreza.

Com sua rapidez, facilidade e arquitetura distribuída, o Git se estabeleceu como uma ferramenta revolucionária no mundo do desenvolvimento de software . Essencial para projetos de todos os tamanhos, ele permite um controle de versão ágil e adaptável [6]. A influência do Git transcendeu, remodelando a maneira como os desenvolvedores colaboram e gerenciam código. Desde sua fundação em 2005, o Git tem amadurecido e evoluído constantemente, mantendo os princípios que o tornaram notável [8]. Sua velocidade impressionante, eficiência em projetos extensos e capacidade de administrar desenvolvimento não linear, evidenciada por seu sistema de branches, são notáveis. Não é surpresa que, entre os sistemas de controle de versão disponíveis, o Git tenha conquistado uma parcela significativa do mercado, representando uma parte substancial dos sistemas de controle de versão em uso, com perspectivas de crescimento contínuo. Muitas organizações já abandonaram sistemas de controle de versão mais antigos em favor do Git, tornando-o seu VCS (Version Control System) preferido [7]. O Git simboliza a capacidade humana de superar desafios, inovar e, em última análise, criar algo que transforma a maneira como as comunidades de desenvolvedores colaboram e moldam o futuro do software [4].

4.3.1. Instalação e Configuração do Git

Por meio do site oficial[2] o Git pode ser instalado em ambientes Windows, Linux e MacOS, seguindo as instruções fornecidas pelo assistente de instalação. O objetivo desta seção é discutir em detalhes os passos essenciais necessários para instalar e configurar o Git nos ambientes Windows e Linux, fornecendo uma base sólida para começar o desenvolvimento usando essa poderosa ferramenta de controle de versão.

4.3.1.1. Ambiente Windows

- Através do Site Oficial: Para instalar o Git no ambiente Windows, é necessário acessar o site oficial do Git em https://git-scm.com/download/win através de um navegador web;

[2]https://git-scm.com/downloads

- Baixar o Instalador: Na página de downloads, será encontrado um link para baixar o instalador do Git para Windows. Ao clicar no link, o arquivo executável (.exe) será baixado para o computador;
- Executar o Instalador: Após a conclusão do download, o usuário deve navegar até a pasta onde o arquivo foi salvo e executá-lo. Geralmente, o nome do arquivo segue o formato Git-x.x.x.x-64-bit.exe, onde "x.x.x.x" representa a versão atual do Git;
- Configurações Iniciais Através do Instalador:
  - Definir o Caminho do Git: Ao aceitar os termos e condições do Git, o instalador permitirá escolher o local de instalação. O caminho padrão é recomendado na maioria dos casos, e o usuário pode prosseguir clicando em "Next";
  - Seleção de Componentes: Na próxima etapa, o instalador oferecerá a opção de selecionar os componentes a serem instalados. É aconselhável deixar todas as opções selecionadas, uma vez que são essenciais para o funcionamento adequado do Git. Clique em "Next" para continuar;
  - Selecionar Pasta no Menu Iniciar: O instalador permitirá escolher a pasta no menu Iniciar onde os atalhos do Git serão criados. A pasta padrão é recomendada, então o usuário pode clicar em "Next";
  - Escolher Editor Padrão: Nesta etapa, o usuário poderá selecionar o editor de texto padrão que o Git utilizará para abrir mensagens de commit e outras mensagens. O editor "Vim" é configurado por padrão, mas é possível escolher outro editor, se desejado;
  - Ajustar o Nome do Branch Inicial: O instalador perguntará se deseja ajustar o nome do branch inicial de "master" para "main." Essa mudança é uma prática comum para evitar termos historicamente carregados. Deixe a opção selecionada e clique em "Next";
  - Ambiente PATH: Nesta etapa, o instalador pergunta sobre a configuração do ambiente PATH. Deixe a opção "Use Git from the Windows Command Prompt" selecionada para que o Git seja acessível a partir do prompt de comando. Clique em "Next";
  - Escolher Executável SSH: O usuário poderá escolher o cliente SSH que o Git utilizará. A opção padrão é adequada para a maioria dos casos. Clique em "Next";
  - Escolher Backend de Transporte HTTPS: O instalador questionará sobre o backend de transporte HTTPS. Deixe a opção "Use the OpenSSL library" selecionada e clique em "Next";
  - Terminal a Usar com Git Bash: Selecione o emulador de terminal padrão a ser usado com o Git Bash. "Use MinTTY" é uma escolha comum. Clique em "Next";
  - Comportamento Padrão do git pull: Escolha o comportamento padrão para o comando "git pull." A opção "Default" é a recomendada. Clique em "Next";
- Após concluir todas as etapas, o Git estará instalado e o usuário terá acesso ao prompt do Git Bash no ambiente Windows;

4.3.1.2. Ambiente Linux

- Abrir o Terminal: O Terminal Linux pode ser acessado pressionando simultaneamente as teclas Ctrl + Alt + T ou procurando por "Terminal" no menu de aplicativos, dependendo da distribuição Linux utilizada;

- Atualizar o sistema: Antes de prosseguir com a instalação do Git, é aconselhável atualizar os repositórios de pacotes do sistema para obter as informações mais recentes sobre as versões disponíveis. O comando a ser executado é "sudo apt update";

- Instalar o Git: Com o sistema atualizado, o próximo passo é instalar o Git. Isso pode ser feito através do comando "sudo apt install git";

- Verificar a instalação: Após a conclusão da instalação, é importante verificar se o Git foi instalado corretamente. Isso pode ser feito digitando o comando "git –version";

Após a instalação do Git em um dos ambientes mencionados anteriormente, é necessário configurar o nome de usuário e o endereço de email. Isso pode ser realizado por meio dos seguintes comandos:

- git config –global user.name "Seu Nome";

- git config –global user.email "seu@email.com";

Essa configuração é crucial para que o Git registre com precisão as contribuições nos projetos em que estiver envolvido. É essencial inserir as informações necessárias para uma identificação adequada. Após concluir a instalação do Git em um dos ambientes e configurar corretamente o nome de usuário e o endereço de email, estará preparado para tirar o máximo proveito dessa ferramenta. Nos próximos tópicos desta seção, será abordado com mais detalhes sobre como utilizar o Git, bem como as melhores práticas que contribuirão para otimizar o trabalho de desenvolvimento.

4.3.2. Fundamentos do Git

Este tópico visa fornecer uma base sólida para entender os conceitos fundamentais do sistema de controle de versão Git. Ao iniciar, explica o que é um repositório Git e como inicializá-lo, enfatizando a importância dessa etapa crucial. Em seguida, discute os conceitos de stading e commit, mostrando como essas técnicas permitem um controle minucioso das mudanças em um projeto. Por fim, apresenta os comandos básicos do Git, como "git add" para fazer alterações, "git commit" para registrar essas mudanças no histórico e outros comandos úteis, como "git status" e "git log".

4.3.2.1. Inicialização de um repositório Git

É fundamental compreender o que realmente é um repositório Git quando se está estudando os fundamentos do Git. Simplesmente, um repositório Git é onde todo o códigofonte de um projeto é armazenado e rastreado. É como um enorme armário virtual que armazena um histórico de todas as mudanças feitas no código. Isso permite que os desenvolvedores naveguem pelo tempo e restaurem os arquivos anteriores quando necessário [8]. Este repositório é o núcleo do controle de versão Git e permite que as equipes de desenvolvimento trabalhem de forma organizada.

Além disso, é importante compreender o procedimento de inicialização de um repositório Git local, que é executado por meio do comando "git init". A Figura 4.1 mostra a criação de um repositório local na pasta "GitFundamentos" através do GitBash. Ao tomar essa iniciativa, seja ao dar início a um novo projeto ou ao desejar começar a rastrear um código já existente com o Git, o desenvolvedor estabelece um repositório local. Com essa ação, o projeto passa a ser supervisionado pelo Git, pronto para registrar com minúcia todas as mudanças e atualizações realizadas. A partir desse momento, o Git assume a responsabilidade de monitorar todas as operações que ocorrem nos arquivos do projeto, o que permite um controle exaustivo de todas as alterações que foram implementadas. A gestão eficaz do código-fonte e o desenvolvimento colaborativo dependem dessa capacidade de monitoramento. Assim, compreender esses princípios essenciais é um passo importante na jornada do domínio do Git e aproveitar suas vantagens no desenvolvimento de software.

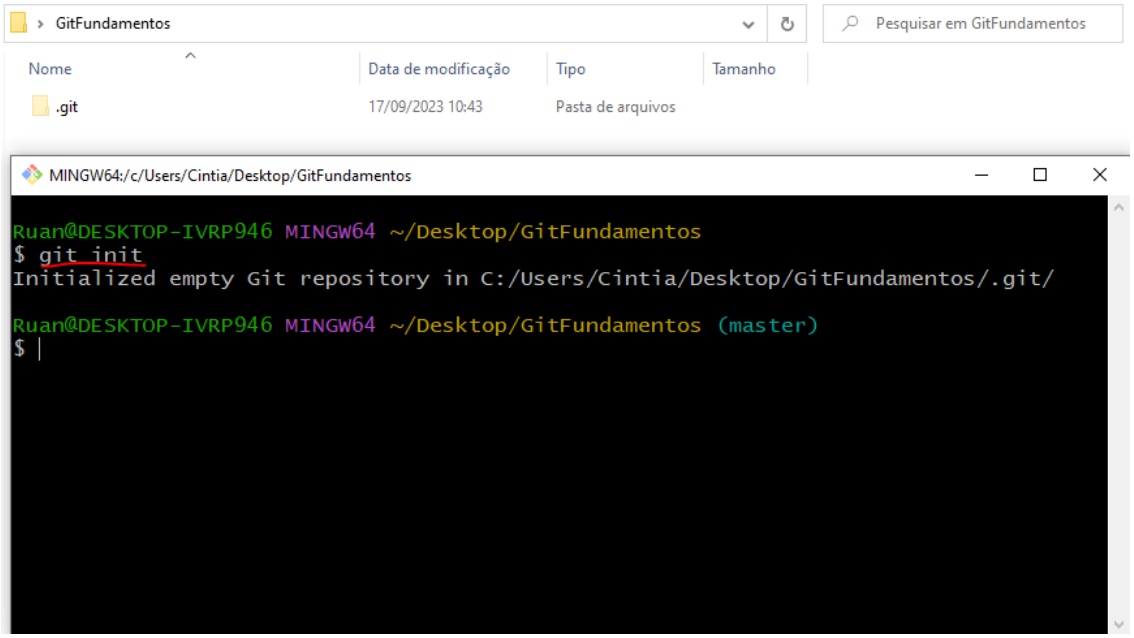


Figure 4.1. Criação de um Repositório Local. Fonte: Elaborada pelos autores

4.3.2.2. O Conceito de Staging Área e Commit

A área de preparação, conhecida como staging area, desempenha um papel fundamental no uso eficaz do Git. Nesse contexto, pode ser comparada a um espaço de seleção, onde as alterações feitas nos arquivos de um projeto são organizadas cuidadosamente antes de serem oficialmente registradas em um commit. Essa prática oferece aos desenvolvedores a oportunidade de revisar todas as modificações realizadas e, com precisão, escolher quais delas serão incorporadas a um commit específico. Essa abordagem não apenas promove uma organização estruturada, mas também se mostra altamente benéfica em cenários complexos de desenvolvimento de software.

Nessa área de preparação, em que as modificações passam por uma triagem e são organizadas antes de se tornarem permanentes no histórico de um projeto, possibilita que as equipes operem com maior eficiência, assegurando que somente as alterações desejadas sejam registradas. Esse aspecto torna-se particularmente valioso em projetos que contam com a participação de várias pessoas e contribuições, onde a clareza e a precisão no controle de versão desempenham um papel crucial para garantir o sucesso de um desenvolvimento colaborativo.

Já o processo de commit é o momento em que as alterações que foram previamente preparadas na área de staging são oficialmente registradas no histórico do repositório Git. Cada commit cria um novo ponto de referência para registrar quaisquer mudanças feitas no projeto até então. Um método recomendado é fornecer mensagens de commit significativas e descritivas que expliquem as alterações de forma simples e compreensível. Dessa forma, todos podem rapidamente entender quais mudanças foram feitas em um commit específico, facilitando a colaboração da equipe.

A Figura 4.2 ilustra visualmente o fluxo do processo, onde as modificações feitas nos arquivos no diretório de trabalho são selecionadas e movidas para a área de preparação (staging area) antes de serem registradas no repositório Git por meio de um commit. Isso cria um histórico claro e rastreável de todas as alterações feitas no projeto, permitindo que os desenvolvedores colaborem de forma mais eficiente e compreensível.

Figure 4.2. Fluxo do processo de um commit. Fonte: Elaborada pelos autores

Além disso, a Figura 4.3 apresenta um exemplo prático desse processo no prompt do Git Bash, demonstrando o uso dos comandos "git status," "git add .," e "git commit -a -m ’Adição de um novo arquivo’" para preparar e registrar alterações em um projeto Git:

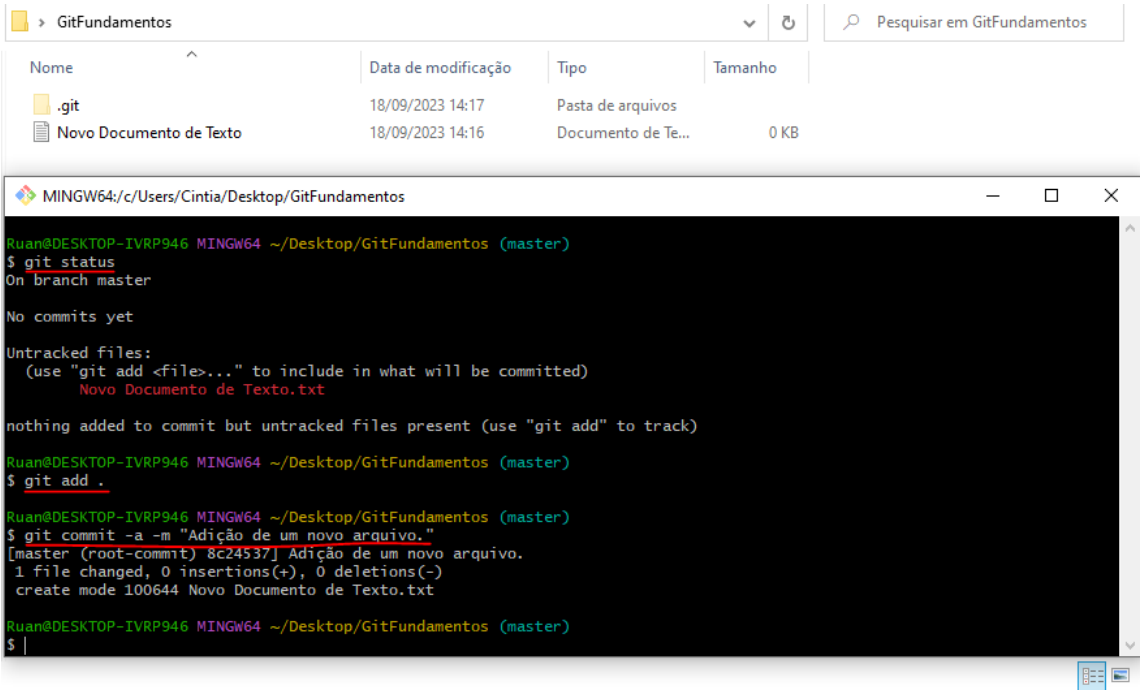


Figure 4.3. Demonstração do processo no Git Bash. Fonte: Elaborada pelos autores

- O comando **git status** é utilizado para verificar o estado atual do repositório, exibindo quais arquivos foram modificados e se estão prontos para o staging;
- Em seguida, o comando **git add .** é empregado para adicionar todas as mudanças ao staging area, preparando-as para o próximo commit;
- Por fim, o comando **git commit -a -m "Adição de um novo arquivo"** realiza o commit das alterações preparadas no staging area, juntamente com uma mensagem descritiva que documenta o propósito do commit;

4.3.3. Trabalhando com Branches

As branches são como "galhos" de uma grande árvore, permitindo que o projeto se divida em caminhos separados. Essa divisão possibilita que várias linhas de desenvolvimento ocorram simultaneamente, como se estivesse trabalhando em diferentes partes da floresta, sem interferências entre elas. A importância das branches se deve ao fato de que, ao construir software, é comum a necessidade de adicionar novos recursos, corrigir erros ou fazer melhorias. No entanto, fazer todas essas mudanças diretamente no projeto principal pode gerar confusão. É aí que as branches entram em cena.

Com as branches, é possível criar cópias separadas do projeto, cada uma representando uma ideia ou uma tarefa específica. Por exemplo, se deseja adicionar um novo

recurso, pode criar uma branch para isso, e se precisa corrigir um bug ao mesmo tempo, outra branch pode ser criada para a correção. Dessa forma, é possível trabalhar em cada tarefa de forma isolada, sem preocupações com interferências entre elas.

Quando todas essas tarefas estiverem concluídas, é possível reuní-las de volta ao projeto principal, como se estivesse unindo as trilhas de volta à estrada principal da floresta. Isso é conhecido como "merge" (mesclagem) e representa uma parte crucial do controle de versão. O uso adequado das branches torna o desenvolvimento mais organizado, evita confusões e, o mais importante, permite manter um histórico claro e rastreável de todas as mudanças no projeto.

Portanto, as branches funcionam como caminhos que orientam o desenvolvimento de software, garantindo a capacidade de explorar, inovar e melhorar o projeto de forma controlada e eficaz. À medida que este tópico é explorado, são apresentadas informações sobre como criar, gerenciar e utilizar branches para tornar o trabalho mais organizado e produtivo.

Na Figura 4.4, é possível visualizar um cenário típico de controle de versão com o uso de branches. A branch principal, representada como "master", constitui o tronco principal da árvore de desenvolvimento. Os círculos azuis ao longo da branch "master" representam os commits realizados nessa linha de desenvolvimento principal. Além disso, são observadas duas branches secundárias, representadas por círculos verdes e uma bolinha laranja. Cada uma dessas branches secundárias representa uma linha de desenvolvimento separada, onde novos recursos ou correções estão sendo trabalhados de forma independente.

A branch verde escura, que se destaca na figura, representa o processo de "merge" (mesclagem). Esse processo ocorre quando as mudanças efetuadas em uma das branches secundárias são integradas de volta à branch principal "master". O resultado é um único histórico de desenvolvimento que incorpora as alterações feitas nas branches secundárias.

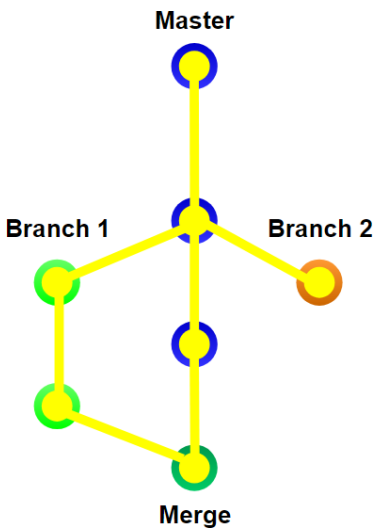


Figure 4.4. Ilustração das branches em um repositório Git. Fonte: Elaborada pelos autores

Essa representação gráfica demonstra como as branches permitem o desenvolvimento simultâneo de diferentes recursos ou correções, mantendo o histórico de cada linha de desenvolvimento de forma clara e organizada. Quando o trabalho em uma branch secundária é concluído e revisado, ele pode ser mesclado de volta à branch principal, garantindo que todas as alterações sejam incorporadas ao projeto principal de maneira controlada e eficiente. Esse método auxilia na preservação da integridade do código e facilita a colaboração entre membros da equipe.

A manipulação de branches no Git é uma parte essencial do controle de versão e do desenvolvimento colaborativo. Existem vários comandos que permitem criar, listar, alternar entre branches e mesclar alterações de diferentes branches. A Figura 4.5 ilustra visualmente, em vermelho, esses comandos no Git Bash seguidos de seus significados:

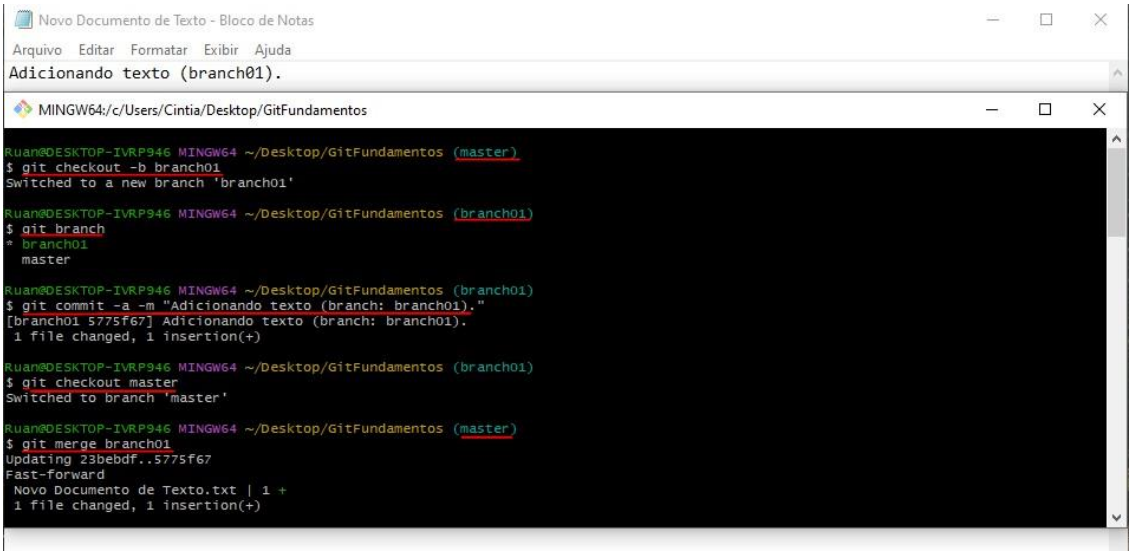


Figure 4.5. Comandos para gerenciar branches no Git Bash. Fonte: Elaborada pelos autores

- Primeiro, o comando **git checkout -b "nome da branch"** é usado para criar uma nova branch e mudar imediatamente para ela. Isso permite que você inicie um novo ramo de desenvolvimento com o nome especificado;

- O comando **git branch** é usado para listar todas as branches disponíveis no repositório. Isso ajuda a visualizar todas as linhas de desenvolvimento em andamento;

- Para confirmar as alterações em uma branch, você utiliza o comando **git commit -a -m "Mensagem de commit"**. Esse comando registra as alterações feitas na branch atual, fornecendo uma mensagem descritiva que explica o objetivo do commit;

- Quando é necessário alternar entre diferentes branches, o comando **git checkout nome-da-branch** é usado. Isso permite que você mude para a branch desejada e continue o trabalho nela;

- Por fim, o comando **git merge** nome-da-branch é utilizado para mesclar as alterações de uma branch em outra. Isso é particularmente útil quando você deseja incorporar o trabalho de uma branch de desenvolvimento de volta à branch principal, como a "master";

Esses comandos são essenciais para a organização do desenvolvimento de software com o Git, pois permitem a confirmação de alterações, a criação de várias linhas de desenvolvimento independentes e a integração de novos recursos ou correções de bugs no projeto principal.

4.3.4. Resolução de Conflitos

A resolução de conflitos no Git é um tópico fundamental para que os desenvolvedores possam gerenciar e trabalhar eficazmente com diferentes ramificações de um projeto. Os parágrafos a seguir apresentam explicações sobre esse tema, com o intuito de torná-lo acessível a iniciantes.

Quando vários desenvolvedores – ou até mesmo um único desenvolvedor – trabalham em partes diferentes de um projeto, é comum que as alterações feitas por um desenvolvedor entrem em conflito com as feitas por outro desenvolvedor. Esses conflitos podem ocorrer quando duas partes alteram uma seção específica de um arquivo ou quando uma parte exclui um arquivo que a outra parte está editando. Embora o Git seja capaz de detectar esses conflitos, é fundamental saber como resolvê-los corretamente.

A resolução de conflitos no Git é um processo que envolve examinar as diferenças entre as alterações conflitantes e decidir quais modificações serão mantidas. Esses conflitos nos arquivos com notações especiais, como "«««« HEAD" e "»»»»»» branch", são marcados pelo Git. Decidir qual versão do código deve ser mantida e qual deve ser descartada é responsabilidade do desenvolvedor. Inicialmente, isso pode parecer um pouco intimidante, mas quando você o faz, ele se torna uma tarefa comum.

A primeira coisa que deve ser feita para resolver conflitos é abrir o arquivo em conflito em um editor de texto. Em seguida, o desenvolvedor verá as partes que estão em conflito marcadas. A escolha de quais partes de cada alteração serão mantidas e como o arquivo deve ficar após a resolução são as principais responsabilidades. Para garantir que a versão final seja funcional e sem erros, pode ser necessário uma compreensão profunda do contexto das alterações e da lógica do código. Após as decisões corretas, as marcações de conflito são removidas e apenas as partes do código que o desenvolvedor deseja manter são mantidas. Para registrar a resolução do conflito, o arquivo alterado é adicionado ao Git e um novo commit é criado. É fundamental fornecer uma explicação detalhada das alterações feitas durante a resolução em uma mensagem de commit. Uma vez que o conflito tenha sido resolvido e documentado, o desenvolvedor pode começar o projeto normalmente.

Na Figura 4.6, é apresentado um exemplo prático que ilustra a resolução de conflitos no Git. Nesse cenário, duas branches estão envolvidas: a branch "main" e a branch "conflitos". Cada passo do processo é acompanhado por comandos Git relevantes e explicações claras.

A branch "main" é a linha principal de desenvolvimento, enquanto a branch "con-

flitos" representa uma linha de desenvolvimento secundária. Cada uma delas tem seus próprios commits e alterações específicas. No exemplo, as alterações conflitantes ocorrem na "Linha 1" de um arquivo de texto, onde ambas as branches fazem modificações diferentes. A resolução de conflitos envolve a análise dessas diferenças, a escolha das modificações a serem mantidas e a criação de um novo commit para registrar a resolução. No exemplo, você pode ver as mensagens de commit associadas a cada passo do processo, tornando a resolução de conflitos uma tarefa clara e compreensível.

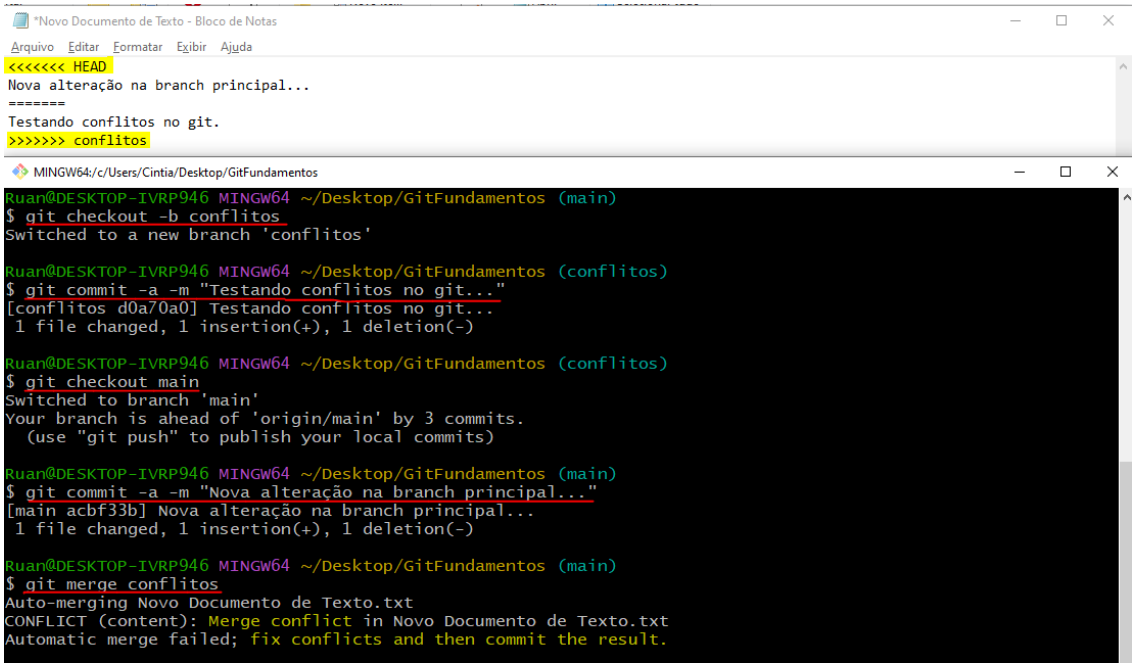


Figure 4.6. Resolvendo Conflitos. Fonte: Elaborada pelos autores

- **git checkout -b conflito**s: Um novo branch denominado "conflitos" é criado e, em seguida, o Git muda para essa branch. Nesse momento, o usuário está trabalhando na branch "conflitos";
- **git commit -a -m "Testando conflitos no git..."**: Um commit é realizado na branch "conflitos" com uma mensagem descritiva. Esse commit representa uma alteração específica feita nessa branch;
- **git checkout main**: A partir deste comando, o usuário retorna à branch principal "main". Agora, está novamente na branch principal;
- **git commit -a -m "Nova alteração na branch principal..."**: Um novo commit é criado na branch principal "main" com outra mensagem descritiva. Esse commit representa uma alteração diferente feita na branch principal.
- **git merge conflitos**: Neste ponto, o objetivo é mesclar as alterações da branch "conflitos" de volta à branch principal "main". No entanto, devido à existência de alterações conflitantes na mesma parte do arquivo em ambas as branches, isso resultará em um conflito que precisará ser resolvido.

Após a resolução do conflito, o desenvolvimento pode continuar de forma organizada e controlada, assegurando que as alterações sejam incorporadas ao projeto principal de maneira eficaz. Embora a resolução de conflitos possa parecer uma habilidade desafiadora no início, com o tempo, os desenvolvedores ganharão confiança e eficácia ao lidar com essas situações. Isso permite que o projeto prossiga de maneira fluida e com a colaboração de todos os envolvidos.

4.4. GitHub e Colaboração em Equipe

O GitHub[3], um serviço de hospedagem de código colaborativo construído sobre o sistema de controle de versão Git, desempenha um papel fundamental no mundo do desenvolvimento de software de código aberto (OSS). Esta plataforma é muito mais do que apenas um repositório para código-fonte; é uma comunidade vibrante que reúne milhões de desenvolvedores e projetos de OSS [3]. O desenvolvimento de software de código aberto sempre foi uma parte vital da indústria de tecnologia, alimentando inúmeras inovações e avanços. Através do GitHub, desenvolvedores de todo o mundo podem colaborar em projetos, contribuir com código, relatar e discutir problemas (bugs) e melhorar continuamente o software que é usado por milhões de pessoas em todo o mundo. [8].

Entretanto, a colaboração distribuída e descentralizada presente no universo do desenvolvimento de software de código aberto (OSS) também traz consigo desafios significativos. À medida que as comunidades de desenvolvimento OSS crescem, é preciso enfrentar a complexa tarefa de coordenar esforços entre os desenvolvedores, monitorar e registrar mudanças em diversos repositórios e manter padrões de qualidade consistentes.

É nesse cenário desafiador que o GitHub se destaca como uma plataforma fundamental para a comunidade OSS. O GitHub oferece uma plataforma robusta que fornece suporte a uma ampla gama de recursos essenciais para o desenvolvimento colaborativo. Entre esses recursos, destacam-se:

- Controle de Versão: O GitHub baseia-se no sistema de controle de versão Git, o que significa que oferece uma maneira eficiente e confiável de rastrear e gerenciar alterações no código-fonte. Isso é crucial para manter um histórico claro e rastreável de todas as mudanças realizadas ao longo do tempo;
- Gerenciamento de Distribuição de Software: O GitHub não se limita apenas ao armazenamento de código-fonte. Ele também facilita a distribuição de software, permitindo que os desenvolvedores compartilhem seus projetos com facilidade, tornando-os acessíveis a uma ampla audiência;
- Rastreamento de Bugs e Problemas: O GitHub oferece um sistema integrado de rastreamento de problemas e bugs. Isso permite que os desenvolvedores relatem problemas, sugiram melhorias e acompanhem o progresso das correções. Essa funcionalidade é vital para manter um software de alta qualidade;
- Integração Contínua (CI) e Entrega Contínua (CD): Por meio do GitHub Actions, os desenvolvedores podem automatizar a integração contínua e a entrega contínua

[3]https://github.com/

de seus projetos. Isso significa que as alterações de código são testadas automaticamente e, se aprovadas, podem ser implantadas sem intervenção manual, acelerando o ciclo de desenvolvimento;

Todas essas ferramentas e recursos são essenciais para manter o ritmo acelerado de produção e manutenção de software de alta qualidade em um ambiente de código aberto. O GitHub desempenha um papel crucial ao fornecer uma plataforma unificada que atende às necessidades da comunidade OSS, permitindo que os desenvolvedores colaborem de maneira eficaz, resolvam problemas rapidamente e entreguem software de alta qualidade para usuários em todo o mundo. Portanto, o GitHub não é apenas um repositório de código; é o epicentro da inovação e colaboração no desenvolvimento de software de código aberto.

4.4.1. Entrando no GitHub

Este tópico oferece orientações essenciais sobre como utilizar o GitHub após a criação de uma conta no site oficial[4]. Com o registro concluído, os usuários estão prontos para explorar as funcionalidades da plataforma, hospedar projetos de código e colaborar com outros desenvolvedores. A seguir, destacamos as etapas cruciais relacionadas à criação de repositórios (repos) e ao processo de clonagem de repositórios já existentes:

4.4.1.1. Criando um Repositório

Um repositório, frequentemente chamado de "repo", é onde o código-fonte e os arquivos relacionados a um projeto são armazenados. Criar um repositório no GitHub é o ponto de partida para compartilhar e colaborar com outros desenvolvedores. Aqui estão as etapas para criar um novo repositório:

- Após efetuar o login, o usuário deve acessar o seu perfil, clicando na foto de perfil localizada no canto superior direito e selecionando a opção "Your profile" (Seu perfil);
- Para criar um novo repositório, no perfil do usuário, é necessário clicar no botão "Repositories" (Repositórios) e, em seguida, escolher "New" (Novo) para iniciar o processo de criação do repositório;
- Na página de configuração do repositório, é necessário fornecer os dados essenciais, como o nome do repositório, uma descrição, a escolha entre torná-lo público ou privado, a opção de iniciar com um arquivo README (opcional) e outras configurações pertinentes;
- Após preencher os detalhes, o usuário deve clicar em "Create repository" (Criar repositório). Dessa forma, o novo repositório estará pronto para receber código e arquivos;

[4]https://github.com/signup

Após a criação, o repositório estará disponível no GitHub, e o usuário poderá começar a adicionar arquivos, gerenciar o código-fonte e colaborar com outros desenvolvedores. A Figura abaixo mostra visualmente o passo a passo visto anteriormente.

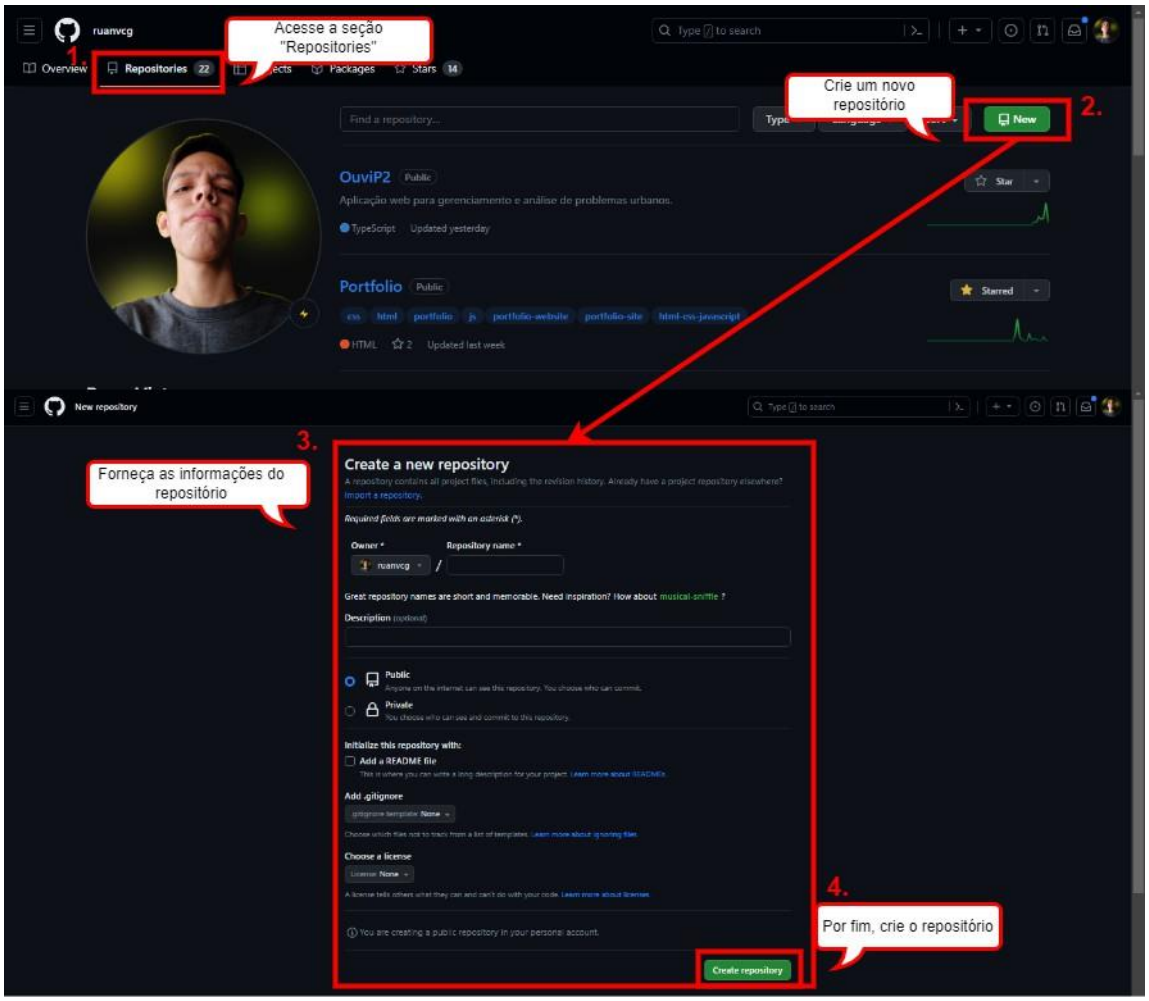


Figure 4.7. Criando um novo repositório. Fonte: Elaborada pelos autores

4.4.1.2. Clonando um Repositório Existente

Clonar um repositório existente no GitHub é um processo fundamental que permite aos usuários criar uma cópia local de um projeto em seu próprio ambiente de desenvolvimento. Isso possibilita que eles trabalhem no código, façam modificações e contribuam para o projeto de maneira eficiente. A seguir, um guia detalhado com todas as etapas necessárias para clonar um repositório existente:

- Localizar o Repositório: O usuário deve navegar até o repositório desejado. A barra de pesquisa na parte superior do GitHub pode ser utilizada para encontrar um repositório específico;

- Obter a URL do Repositório: Após clicar em "Code", uma janela pop-up aparecerá, mostrando a URL. É importante certificar-se de que a opção "HTTPS" esteja selecionada, pois essa é a forma mais comum de clonagem. A URL pode ser copiada clicando no ícone de área de transferência ao lado dela, ou é possível optar por "Download ZIP" para baixar o repositório como um arquivo ZIP compactado;

- Abrir o Terminal (Linux/Mac) ou Git Bash (Windows): Agora, o terminal ou Git Bash deve ser aberto no computador. É fundamental garantir que o Git esteja instalado e configurado corretamente no sistema;

- Clonar o Repositório: No terminal ou Git Bash, digite o seguinte comando: **git clone (URL do repositório desejado)**. Dessa forma, o repositório será clonado com sucesso;

Concluindo com êxito, o repositório do GitHub foi clonado para o computador do usuário, e agora ele está preparado para iniciar o desenvolvimento do projeto localmente. É fundamental manter em mente que ao efetuar alterações no código, é possível submeter essas modificações de volta ao repositório original utilizando as operações de "commits" e "pushes", simplificando assim a colaboração em equipe e a gestão de versões dos projetos.

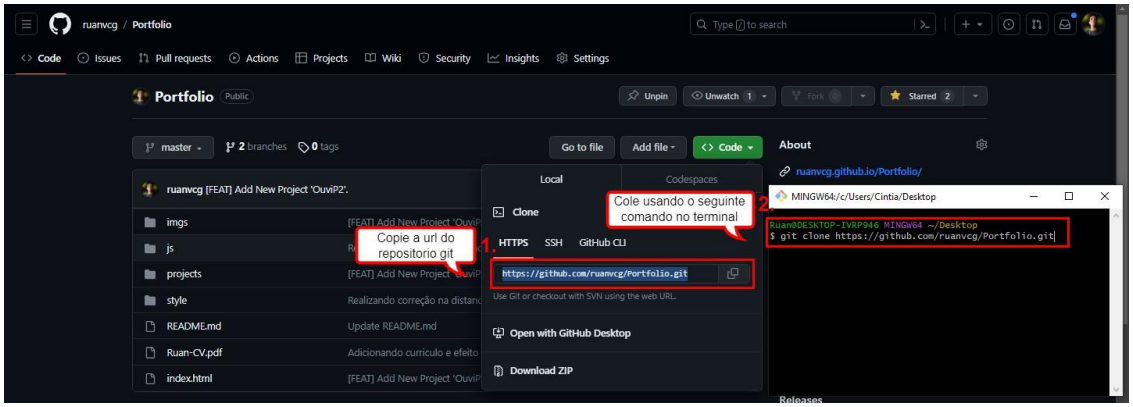


Figure 4.8. Clonando um repositório. Fonte: Elaborada pelos autores

4.4.1.3. Contribuindo com Repositórios

Neste tópico, será abordado detalhadamente como os usuários podem contribuir com repositórios já existentes no GitHub. Contribuir para projetos de código aberto é uma parte fundamental da colaboração na plataforma, permitindo que desenvolvedores de todo o mundo trabalhem juntos em projetos comuns. Abaixo, são apresentados os principais passos para contribuir de maneira eficaz (veja a Figura 4.9 para uma representação visual das principais funções):

- Encontrar um Projeto de Interesse: O primeiro passo para contribuir é encontrar um projeto que desperte interesse. O GitHub possui milhões de repositórios abertos, abrangendo uma ampla variedade de tecnologias e tópicos. É possível utilizar a

barra de pesquisa na parte superior do GitHub para localizar repositórios específicos ou explorar listas de projetos populares;

- Explorar o Repositório: Após identificar um projeto de interesse, é importante explorar o repositório correspondente para compreender melhor seu propósito, estrutura e diretrizes de contribuição. Geralmente, os repositórios contêm um arquivo "README" que oferece informações essenciais sobre o projeto, como sua finalidade, instruções de configuração e orientações para contribuição;
- Criar um Fork do Repositório: Para iniciar a contribuição, é necessário criar um "fork" (cópia) do repositório original. Isso criará uma duplicata do projeto na própria conta do usuário no GitHub, possibilitando trabalhar nele de forma independente;
- Clonar o Fork para o Ambiente Local: Após criar o "fork," o próximo passo é clonar o repositório duplicado para o ambiente de desenvolvimento local utilizando o Git. Essa ação permite que o usuário faça modificações no código e realize testes em seu próprio ambiente;
- Efetuar as Alterações: O desenvolvedor precisa efetuar as alterações requeridas no código, aderindo às melhores práticas de programação e às orientações específicas do projeto;
- Criar uma Solicitação de Mesclagem (Pull Request): Após concluir as alterações e realizar os testes adequados, é hora de criar uma "solicitação de mesclagem" no repositório original. Esse procedimento é realizado por meio da plataforma GitHub e permite que os mantenedores do projeto analisem e revisem as alterações propostas;
- Aguardar a Aprovação: Posteriormente, após a revisão e aprovação das alterações pelos mantenedores do projeto, a contribuição será incorporada ao repositório principal. Com isso, o colaborador terá concluído com sucesso o processo de contribuição;

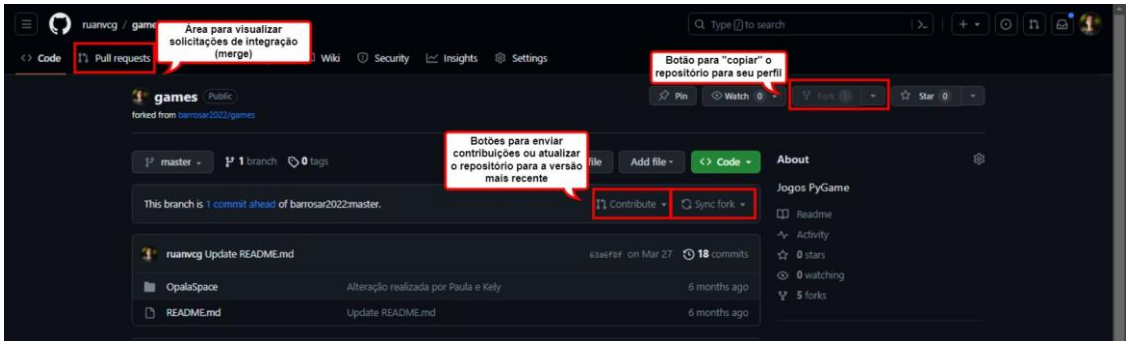


Figure 4.9. Contribuindo em repositórios. Fonte: Elaborada pelos autores

Contribuir para projetos de código aberto no GitHub é uma oportunidade valiosa para desenvolvedores adquirirem experiência significativa, enriquecerem seus conhecimentos ao interagir com outros membros da comunidade e se integrarem a uma rede

global de profissionais do desenvolvimento de software. É crucial manter um comportamento respeitoso, aderir estritamente às diretrizes estabelecidas pelo projeto e demonstrar abertura para receber e incorporar feedback. Essas práticas não apenas facilitam uma colaboração eficaz, mas também enriquecem a experiência, proporcionando um ambiente de aprendizado contínuo e gratificante.

4.5. Considerações Finais

Este minicurso explora profundamente o universo do Git e do GitHub, fornecendo uma visão abrangente dessas ferramentas essenciais para o desenvolvimento de habilidades de colaboração e controle de versões. Inicialmente, estabelece uma base conceitual sólida, esclarecendo os termos e fundamentos relacionados ao Git e ao GitHub. Em seguida, segue uma abordagem prática, detalhando passo a passo como instalar e configurar o Git, ao mesmo tempo em que explora suas funcionalidades centrais.

Ao longo deste minicurso, enfatiza-se a importância da colaboração em equipe e do controle de versões no desenvolvimento de software moderno. Destaca-se como o GitHub se tornou uma plataforma fundamental para desenvolvedores e equipes de software, possibilitando a colaboração global em projetos de pequeno e grande porte. Ao final, o objetivo é inspirar os participantes a aplicarem essas habilidades em suas práticas de desenvolvimento, capacitando-os para contribuir efetivamente em projetos de código aberto e melhorar suas capacidades de colaboração e controle de versões. Promover uma colaboração saudável e eficaz no desenvolvimento de software é uma responsabilidade compartilhada pela comunidade científica, formuladores de políticas e toda a sociedade, e este minicurso representa um passo na direção de alcançar esse objetivo.

Referências

[1] Blauw, F. F. (2018). The use of git as version control in the south african software engineering classroom. In *2018 IST-Africa Week Conference (IST-Africa)*, pages Page 1 of 8–Page 8 of 8.

[2] de Alwis, B. and Sillito, J. (2009). Why are software projects moving from centralized to decentralized version control systems? In *2009 ICSE Workshop on Cooperative and Human Aspects on Software Engineering*, pages 36–39.

[3] Decan, A., Mens, T., Mazrae, P. R., and Golzadeh, M. (2022). On the use of github actions in software development repositories. In *2022 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 235–245.

[4] Elsen, S. (2013). Visgi: Visualizing git branches. In *2013 First IEEE Working Conference on Software Visualization (VISSOFT)*, pages 1–4.

[5] Liberty, J. and Galloway, J. (2021).

[6] Parizi, R. M., Spoletini, P., and Singh, A. (2018). Measuring team members' contributions in software engineering projects using git-driven technology. In *2018 IEEE Frontiers in Education Conference (FIE)*, pages 1–5.

[7] Singh, V., Singh, A., Aggarwal, A., and Aggarwal, S. (2021). Devops based migration aspects from legacy version control system to advanced distributed vcs for deploying micro-services. In *2021 IEEE International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS)*, pages 1–5.

[8] Spinellis, D. (2012). Git. IEEE Software, 29(3):100–101.

[9] Xu, X., Cai, Q., Lin, J., Pan, S., and Ren, L. (2019). Enforcing access control in distributed version control systems. In *2019 IEEE International Conference on Multimedia and Expo (ICME)*, pages 772–777.

[10] Zhao, Y., Zhou, Y., Hu, C., Zhang, X., and Zhang, Z. (2018). Gitbased version control for beamline control system at the shanghai synchrotron radiation facility. In *2018 5th International Conference on Systems and Informatics (ICSAI)*, pages 134–138.