UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"
FACULDADE DE CIÊNCIAS - CAMPUS BAURU
DEPARTAMENTO DE COMPUTAÇÃO
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

MATHEUS DOS SANTOS RIBEIRO SILVA

DESENVOLVIMENTO DE UMA APLICAÇÃO INTEGRADA PARA GERENCIAMENTO DE PROJETOS E RECURSOS PARA PROGRAMADORES

BAURU
2023

MATHEUS DOS SANTOS RIBEIRO SILVA

DESENVOLVIMENTO DE UMA APLICAÇÃO INTEGRADA PARA GERENCIAMENTO DE PROJETOS E RECURSOS PARA PROGRAMADORES

Trabalho de Conclusão de Curso do Curso de Bacharelado em Ciência da Computação da Universidade Estadual Paulista "Júlio de Mesquita Filho", Faculdade de Ciências, Campus Bauru.
Orientador: Prof. Me. Juliana da Costa Feitosa

BAURU
2023

S586d

Silva, Matheus dos Santos Ribeiro
Desenvolvimento de uma Aplicação Integrada para Gerenciamento de Projetos e Recursos para Programadores / Matheus dos Santos Ribeiro Silva. -- Bauru, 2023
44 p. : tabs., fotos

Trabalho de conclusão de curso (Bacharelado - Ciência da Computação) - Universidade Estadual Paulista (Unesp), Faculdade de Ciências, Bauru
Orientadora: Juliana da Costa Feitosa

1. Software Desenvolvimento. 2. Software integrado. 3. Ciência da computação. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Biblioteca da Faculdade de Ciências, Bauru. Dados fornecidos pelo autor(a).

Essa ficha não pode ser modificada.

Matheus dos Santos Ribeiro Silva

DESENVOLVIMENTO DE UMA APLICAÇÃO INTEGRADA PARA GERENCIAMENTO DE PROJETOS E RECURSOS PARA PROGRAMADORES

Trabalho de Conclusão de Curso do Curso de Bacharelado em Ciência da Computação da Universidade Estadual Paulista “Júlio de Mesquita Filho”, Faculdade de Ciências, Campus Bauru.

Banca Examinadora

Prof. Me. Juliana da Costa Feitosa
Orientador
Universidade Estadual Paulista "Júlio de Mesquita Filho"
Faculdade de Ciências
Departamento de Ciência da Computação

Prof. Dra. Simone das Graças Domingues Prado
Universidade Estadual Paulista "Júlio de Mesquita Filho"
Faculdade de Ciências
Departamento de Ciência da Computação

Prof. Dr. Kelton Augusto Pontara da Costa
Universidade Estadual Paulista "Júlio de Mesquita Filho"
Faculdade de Ciências
Departamento de Ciência da Computação

Bauru, 17 de Novembro de 2023.

Agradecimentos

Em primeiro lugar, agradeço a Deus por me conceder a oportunidade de concluir mais esta etapa acadêmica e realizar um sonho.

Agradeço minha família, sobretudo minha mãe, Maria Regina, o meu Pai, Eliseu, e minha irmã, Larissa. Não poderia ter pessoas melhores me apoiando e torcendo por minhas conquistas.

Também gostaria de agradecer a minha namorada e companheira de vida, Silvania. Que esteve comigo e me apoiou em todos os momentos, inclusive os mais difíceis.

Obrigado aos amigos que fiz durante esses 4 anos de graduação, vocês fizeram esse ciclo ser inesquecível, agradeço por todo o apoio.

Obrigado ao professor Kelton Augusto Pontara da Costa, por possibilitar a minha participação e contribuição no projeto de extensão "Conexão Legal", e por aceitar ser um membro da banca avaliadora do meu trabalho.

Também quero agradecer ao professor Jose Remo Ferreira Brega, por disponibilizar o laboratório LIV para realização do trabalho.

E, por fim, agradeço a minha orientadora, e uma das melhores professoras que eu já tive, Juliana da Costa Feitosa. Obrigado por acreditar no meu projeto e estar sempre disposta a ajudar.

Resumo

A área de Desenvolvimento de Software experimentou um crescimento exponencial nas últimas décadas, impulsionando uma revolução tecnológica que tem tido um impacto significativo na vida cotidiana de bilhões de pessoas. Esse avanço se traduz em benefícios notáveis em setores como Educação, Medicina, Engenharia, Matemática, Indústria, entre outros. Em consonância com a crescente demanda, houve um aumento substancial na quantidade de profissionais de desenvolvimento de *software*. No contexto da programação no Brasil, modalidades de emprego mais flexíveis, a exemplo o trabalho como pessoa jurídica, oferece vantagens atrativas, como salários superiores e benefícios fiscais. No entanto, a gestão de projetos em diversas empresas, em múltiplos projetos e com uma variedade de tecnologias pode ser desafiadora, resultando na dispersão da atenção dos programadores, o que prejudica sua eficiência e desempenho. Com o advento e a disseminação do acesso às Inteligências Artificiais Generativas (IAG) para um público mais amplo, observou-se a integração de serviços e conjuntos de *software* com Inteligência Artificial (IA), unificando funcionalidades e dados para melhorar a eficiência e o desempenho dos usuários. Grandes empresas, como *Microsoft* e *Google*, buscam cada vez mais integrar seus diversos serviços e ambientes de trabalho com IAs, como o *ChatGPT* da *OpenAI* ou o *Bart* da *Google*, com o objetivo de disponibilizar produtos úteis para seus usuários e otimizar seus lucros por meio da melhoria contínua de suas ferramentas. Nesse contexto, este projeto propõe a criação de uma *dashboard* integrada que utiliza uma Application Programing Interface (API) do *ChatGPT* e do *Google Workspace*. Essa plataforma visa centralizar o gerenciamento de códigos e anotações em projetos de desenvolvimento, com o propósito de aprimorar a eficiência e a produtividade dos desenvolvedores de *software*.

Palavras-chave: Integração de serviços; *Dashboard*; *Application Programing Interface*; Desenvolvimento de Software.

Abstract

The field of Software Development has experienced exponential growth in recent decades, driving a technological revolution that has had a significant impact on the daily lives of billions of people. This advancement translates into noteworthy benefits across various sectors, including Education, Medicine, Engineering, Mathematics, and Industry, among others. In response to the growing demand, there has been a substantial increase in the number of software development professionals. In the context of programming in Brazil, more flexible employment modalities, such as working as "Pessoa Jurídica", offer attractive advantages such as higher salaries and tax benefits. However, managing projects across multiple companies, handling numerous projects, and dealing with a variety of technologies can be challenging, resulting in the dispersion of programmers' attention, which impairs their efficiency and performance. With the advent and widespread accessibility of Generative Artificial Intelligences to a broader audience, there has been an observed integration of services and software suites with AI, consolidating functionalities and data to enhance the efficiency and performance of users. Major corporations, such as Microsoft and Google, are increasingly seeking to integrate their diverse services and workspaces with AIs like OpenAI's ChatGPT or Google's Bart, aiming to provide valuable products to their users and optimize their profits through continuous improvement of their tools. In this context, this project proposes the creation of an integrated dashboard utilizing APIs from ChatGPT and Google Workspace. This platform aims to centralize code and project annotation management, with the goal of enhancing the efficiency and productivity of software developers.

Palavras-chave: *Software Integration*; *Dashboard*; *Application Programing Interface*; *Software Development*.

Lista de figuras

Figura 1 – Fluxograma da estrutura da aplicação 27
Figura 2 – Login com o Google 29
Figura 3 – Captura de tela de configuração do cliente no site do Google Cloud 30
Figura 4 – Exemplo de token de acesso 30
Figura 5 – Aceite de termos do Google Drive 31
Figura 6 – Captura de tela da barra lateral direita, mostrando o sistema de gerenciamento de arquivos 32
Figura 7 – Fluxo de autenticação e requisição do serviço do Google Drive . . . 33
Figura 8 – Captura de tela do exemplo de uma requisição para o Google Drive 33
Figura 9 – Captura de tela da visualização superior da barra lateral esquerda . 34
Figura 10 – Captura de tela da barra lateral esquerda com a opção do chat aberta 34
Figura 11 – Captura de tela exibindo os componentes principais da aplicação . . 35
Figura 12 – Captura de tela do Google Drive 36
Figura 13 – Visualização das opções de funções na edição de códigos e anotações 36
Figura 14 – Visualização da função de editor de código 37
Figura 15 – Visualização de algumas funções do componente 38

Lista de quadros

Quadro 1 – Software 24
Quadro 2 – Hardware 24

Lista de abreviaturas e siglas

API	*Application Programming Interface*
APP	Aplicativo
Bot	Robô
CI/CD	*Continuous Integration / Continuous Delivery*
CLT	Consolidação das Leis Trabalhistas
FGTS	Fundo de Garantia do Tempo de Serviço
GB	*Gigabyte*
HDD	Unidade de Disco Rígido
IA	Inteligência Artificial
IAG	Inteligência Artificial Generativa
IDE	Ambiente Integrado de Desenvolvimento
ID	Identificador
JS	*JavaScript*
LLM	*Large Language Model*
PJ	Pessoa Jurídica
SDK	*Software Development Kit*
TB	*TeraByte*
TCC	Trabalho de Conclusão de Curso
TI	Tecnologia da Informação
URI	*Uniform Resource Identifier*
VSCode	*Microsoft Visual Studio Code*

Sumário

1 INTRODUÇÃO 12
1.1 Problemática 13
1.1.1 PJ - Múltiplas Empresas, Projetos e Ferramentas 13
1.1.2 Dispersão de Recursos e Concentração de Tarefas 14
1.1.3 Integração de Inteligências Artificiais Generativas (IAGs) como Potencial Solução 14
1.2 Justificativa 15
1.3 Objetivos 16
1.3.1 Objetivos Específicos 16
1.4 Organização do Trabalho 16

2 FUNDAMENTAÇÃO TEÓRICA 17
2.1 Produtividade para Programadores 17
2.1.1 Conceito de performance para programadores 17
2.1.1.1 Interrupções e mudanças de contexto 18
2.1.1.2 Gerenciamento de tempo 19
2.2 IAG, LLM e Sistemas Integrados 19
2.2.1 IAG 19
2.2.2 Sistemas Integrados com LLM 20
2.2.2.1 Github Copilot 20
2.2.2.2 ChatGPT 21
2.3 Git e Github 22
2.3.1 Git: Controle de Versões Distribuídos 22
2.3.2 *Github*: Plataforma de Colaboração 22

3 FERRAMENTAS E MÉTODOS 24
3.1 Ferramentas Gerais 24
3.1.1 JavaScript 25
3.1.1.1 TypeScript 25
3.1.2 NodeJS 25
3.1.3 NestJS 25
3.1.4 NextJS 26

4 DESENVOLVIMENTO 27
4.1 Módulo de Autenticação 28
4.1.1 Acesso com o Google 28

4.2 Módulo de Gerenciamento de Arquivos 32
4.3 Módulo de IA 33
4.4 Módulo de *Dashboard* 35
4.4.1 Editor de Códigos e textos 36
4.4.2 Modo escuro e claro 38

5 CONSIDERAÇÕES FINAIS 40
5.1 Trabalhos Futuros 40

REFERÊNCIAS 42

1 Introdução

O campo da programação e desenvolvimento de *software* cresce rapidamente, acompanhando o cenário global de tecnologia. Segundo dados da ABES (2023) (Associação Brasileira de Empresas de *software*), impulsionado pela pandemia, o setor tecnológico (que engloba os mercados de *software*, serviços, *hardware* e as exportações do segmento) cresceu 23% no Brasil em 2020. A produção mundial da área de tecnologia apresentou em 2021, um crescimento de 11%, enquanto no Brasil o crescimento chegou a 17,4%. À medida que essa indústria se expande, surgem novos desafios para os profissionais envolvidos, especialmente no que diz respeito à gestão eficaz de projetos e recursos. Entre os possíveis modelos de contratação, destacam-se as modalidades CLT (Consolidação das Leis Trabalhistas) e PJ (Pessoa Jurídica). O modelo mais utilizado é o regime CLT, no qual um vínculo empregatício direto é criado entre o empregado e a empresa. Já o modelo PJ, apesar de menos comum se comparado com o CLT, vem crescendo rapidamente e cada vez mostra-se uma opção atrativa para ambas as partes. Ainda que esse tipo de contratação possua diversas vantagens, prestar serviços para múltiplas empresas e projetos, utilizando diferentes ferramentas e tecnologias, pode dificultar a organização e o gerenciamento de tempo, além de diminuir a eficiência do profissional de programação.

Nesse cenário de constante evolução, observa-se a ascensão das Inteligências Artificiais Generativas (IAGs) como parceiras valiosas no aprimoramento de tarefas complexas e na integração de serviços. De acordo com Velazco (2023), a empresa do setor de tecnologia *Microsoft* está desenvolvendo uma nova aplicação de bate-papo, com integração de uma Inteligência Artificial (IA) para adotar o papel de um suporte técnico do computador do usuário, no seu sistema operacional, o *Windows*. Empresas líderes no setor seguem o mesmo caminho, como o *Google* com o *Bart*, demonstrando o potencial transformador das IAGs, buscando integrá-las em seus ambientes de trabalho para aprimorar a experiência de seus usuários e impulsionar a eficiência operacional.

Este Trabalho de Conclusão de Curso (TCC) surge da necessidade de enfrentar os desafios da gestão de projetos e recursos na indústria de desenvolvimento de *software*, aproveitando o poder das IAGs para a criação de soluções inovadoras. O objetivo principal deste trabalho foi desenvolver uma plataforma integrada, que faça uso das APIs do *ChatGPT* da *OpenAI* e do *Google Workspace*, para centralizar e auxiliar a gestão de códigos, anotações e recursos em projetos de desenvolvimento de *software*. Com isso, ao unificar ferramentas e serviços amplamente utilizados por desenvolvedores, em um sistema focado em facilitar o gerenciamento de diversos pro-

jetos simultaneamente, esse trabalho busca fornecer uma solução em resposta às demandas crescentes dessa categoria de profissionais, contribuindo para a otimização de processos, melhora na gestão de tempo e aumento da produtividade.

1.1 Problemática

A área de desenvolvimento de *software* é um campo em constante evolução que desempenha um papel fundamental na transformação digital de empresas e na criação de soluções tecnológicas inovadoras. De acordo com Andreessen (2011), a necessidade de *software* continua superando a capacidade de produzir o *software*. Essa crescente demanda impulsionou um aumento significativo no número de programadores e desenvolvedores de *software* no Brasil e em todo o mundo. No entanto, esse crescimento também trouxe consigo uma série de desafios complexos que afetam diretamente a eficiência e a produtividade desses profissionais.

1.1.1 PJ - Múltiplas Empresas, Projetos e Ferramentas

No Brasil, o modelo de contratação PJ tem testemunhado um notável aumento de adeptos, tornando-se uma alternativa crescentemente atrativa tanto para contratantes quanto para contratados. Um estudo conduzido pela empresa Revelo (2020), revelou um aumento notável de 27% no número de profissionais de tecnologia interessados em adotar o modelo de contratação PJ no ano de 2020. Adicionalmente, aproximadamente 30% dos candidatos demonstraram disponibilidade para serem contratados sob essa modalidade, representando um aumento significativo em relação ao período pré-pandêmico, quando o interesse por esse regime era de apenas 7%. Esse modelo, caracterizado pela ausência de um vínculo empregatício robusto com a empresa, apresenta diversas vantagens tributárias quando comparado ao tradicional modelo de CLT. No entanto, os profissionais que optam por esse tipo de contratação enfrentam restrições no acesso a benefícios, garantias e direitos assegurados pelo modelo CLT. Segundo o Brasil (1943), entre esses benefícios estão o pagamento de Hora Extra, 13ž salário, período de férias remuneradas, Fundo de Garantia do Tempo de Serviço (FGTS), seguro-desemprego, licença maternidade, e outros direitos correlatos.

Como define o Brasil (1943):

> Considera-se empregado toda pessoa física que prestar serviços de natureza não eventual a empregador, sob a dependência deste e mediante salário.

Dentre as principais distinções do modelo PJ, destaca-se a flexibilidade proporcionada aos profissionais de desenvolvimento de *software*, que conseguem prestar

serviços para múltiplas empresas simultaneamente. Como estabelece a Brasil (2017), a contratação de profissionais PJ é permitida mas deve ser feita de forma adequada, respeitando os diretos dos trabalhadores, não impondo nenhuma restrição ao número de empresas para as quais o profissional pode prestar serviços. Dessa forma, é gerada uma flexibilidade maior nas definições do contrato de trabalho. Com exceção de possíveis impedimentos contratuais, os profissionais que adotam esse regime tornam-se prestadores de serviços independentes, desfrutando da liberdade de atuar em diferentes empresas de maneira flexível. Essa flexibilidade possibilita que os desenvolvedores trabalhem em diversos projetos de forma paralela, empregando uma ampla gama de tecnologias e enfrentando diferentes contextos de trabalho. No entanto, essa multiplicidade de compromissos acrescenta complexidade à organização e gestão desses projetos.

1.1.2 Dispersão de Recursos e Concentração de Tarefas

Segundo o estudo feito por Mark, Gudith e Klocke (2008), a troca de contexto durante o desenvolvimento de uma tarefa não tem uma diferença significativa se comparada com um cenário em que a mesma tarefa foi executada sem interrupções. Surpreendentemente, em alguns casos, a tarefa foi concluída de forma mais rápida quando interrupções eram adicionadas, pois as pessoas buscavam agilizar seu trabalho com a intenção de compensar as pausas realizadas. Ainda de acordo com os autores, apesar de um possível ganho em velocidade, mais estresse, frustração, pressão do tempo e esforço foram necessários, o que gera um desgaste significativamente maior durante a busca pela conclusão da atividade proposta.

1.1.3 Integração de Inteligências Artificiais Generativas (IAGs) como Potencial Solução

À medida em que as IAGs tornam-se mais acessíveis e poderosas, surge a oportunidade de explorar seu potencial na resolução dos desafios enfrentados pelos desenvolvedores de *software*. Grandes empresas como *Microsoft* e *Google* têm investido em IAGs para aprimorar suas soluções e melhorar a eficiência dos fluxos de trabalho. No entanto, até o momento, há uma falta de pesquisa específica sobre como essas IAGs podem ser aplicadas na gestão de projetos e recursos de desenvolvimento de *software* em um contexto mais amplo. IAs como do *Github Copilot* (desenvolvido pela *Microsoft*) e o *Google Bart* (desenvolvido pela *Google*), e *ChatGPT* da *OpenAI*, podem ser integradas em aplicações e serviços diversos, buscando impulsionar a produtividade e possibilitando a criação de ferramentas adicionais ou estruturais em outros *softwares*. Atualmente, grandes empresas já estão integrando essas ferramentas em suas plataformas, como é o caso da empresa Meta. De acordo com a Meta (2023),

um robô de bate-papo será lançado para as plataformas da empresa proprietária de plataformas como o Instagram, Facebook e Whatsapp.

1.2 Justificativa

No contexto brasileiro, onde modalidades de trabalho flexíveis e autônomas são cada vez mais atraentes para desenvolvedores de *software*, a necessidade de ferramentas que otimizem a produtividade e a colaboração torna-se ainda mais premente.

O *Github Copilot* é uma aplicação que utiliza uma IAG para auxiliar programadores no desenvolvimento de códigos. Além de eficiente, conforme apresentado a seguir, várias possíveis integrações com outros programas e aplicações já foram feitas ou estão sendo planejadas, como, por exemplo, o *Windows Copilot*. Uma utilização similar ao fluxo implementado na aplicação, é a do *Github Copilot Chat*, uma nova ferramenta fornecida pelo Github, que permite ao usuário realizar um bate-papo com a IA da empresa. De acordo com um estudo realizado por Rodriguez (2023), 85% dos desenvolvedores participantes, que utilizaram o serviço, sentiram-se mais confiantes com a qualidade de seus códigos, tiveram suas revisões de códigos completas 15% mais rápido, e 88% desses afirmaram ter mantido constância do ritmo de trabalho, pois se sentiram mais focados, menos frustrados, e desfrutaram melhor o processo de programação.

Segundo Mark, Gudith e Klocke (2008), interrupções e mudanças de contexto podem ter um impacto negativo significativo no bem-estar de profissionais. Pausas frequentes e inesperadas podem levar a sentimentos de frustração e estresse, o que pode afetar negativamente o desempenho no trabalho e a qualidade de vida em geral. Além disso, esses intervalos também podem exigir mais esforço e tempo para concluir as tarefas, uma vez que o profissional precisa reconectar-se com a tarefa após cada interrupção. Isso pode levar a uma sobrecarga de trabalho, afetando ainda mais o bem-estar do profissional.

De acordo com Aeon, Faber e Panaccio (2021), de maneira geral, o gerenciamento de tempo aprimora a performance no trabalho, conquista acadêmica, e bem-estar. Quando bem aplicado, pode aumentar significativamente a produtividade e eficiência na execução de tarefas. Nesse contexto, o desenvolvimento de uma aplicação integrada para gerenciamento de projetos e recursos para programadores pode ser uma solução eficiente para melhorar a organização e gestão desses profissionais, contribuindo para a melhoria da produtividade e do bem-estar no trabalho.

Segundo Harman (2012), algoritmos de IA já conseguem gerar análises de *software* inteligentes, desenvolvimento, testes e sistemas de suporte à decisão. A integração de uma IA na ferramenta também pode ajudar a automatizar algumas tarefas,

sugerir melhorias em códigos, resumindo textos, e diminuindo significativamente o tempo gasto pelos profissionais.

Diante disso, este trabalho justifica-se por mitigar as consequências dos problemas apresentados, desenvolvendo uma aplicação integrada que seja eficiente para melhorar o gerenciamento de *workspace* e projetos dos programadores, gerando benefícios para esses profissionais.

1.3 Objetivos

O objetivo principal é, portanto, implementar uma aplicação web integrada, que utilize as APIs do *ChatGPT* da empresa *OpenAI* e do *Google Workspace* da empresa *Google*, para auxiliar o desenvolvimento de *software* por meio da centralização e facilitação do gerenciamento de anotações, código e recursos em projetos de desenvolvimento de *software*, com o propósito de aprimorar a eficiência e a produtividade dos desenvolvedores de *software*.

1.3.1 Objetivos Específicos

- Avaliar e especificar as formas de integração de cada módulo e detalhar o seu funcionamento;
- Implementar o módulo de autenticação, autorização e integração da conta do Google, através do serviço de autenticação da empresa;
- Implementar o sistema de gerenciamento de projetos, anotações e códigos, utilizando a API do Google Drive;
- Implementar o módulo de IA integrando o ChatGPT como ferramenta auxiliar; e
- Desenvolvimento de uma *dashboard* para interação do usuário com o sistema.

1.4 Organização do Trabalho

O presente trabalho está estruturado da seguinte forma: no Capítulo 2 encontram-se as definições, contextualizações e principais informações dos temas mais importantes para o trabalho. Já o Capítulo 3, discorre sobre as tecnologias e métodos escolhidos, detalhando cada um. No Capítulo 4, são abordados aspectos como estrutura, detalhes de desenvolvimento e funcionamento do trabalho. O desenvolvimento da aplicação foi dividido em módulos, sendo cada um detalhado neste capítulo. Por fim, no Capítulo 5 são apresentadas as considerações finais, com uma visão geral do resultado final da aplicação e sugestão para os trabalhos futuros.

2 Fundamentação Teórica

Neste capítulo, os conceitos base para este trabalho são apresentados e definidos, como por exemplo, conceito de produtividade para programadores, gerenciamento de tempo e foco, IAG e sua eficiência, sistemas integrados, modelos de trabalho, crescimento do mercado de tecnologia e *workspaces*. Esta seção de fundamentação teórica oferece uma visão abrangente sobre as estratégias eficazes de gerenciamento de tempo, a evolução do mercado de trabalho brasileiro com o aumento do trabalho PJ e o impacto crescente da IA e *Large Language Model* (LLM), em várias áreas. As informações apresentadas aqui serviram como base sólida para o desenvolvimento das seções subsequentes deste trabalho, abrindo caminho para uma análise mais aprofundada, e aplicação prática das teorias e tendências discutidas.

2.1 Produtividade para Programadores

Nesta seção da metodologia, será explorado o conceito de produtividade na programação em profundidade. Serão analisados os fatores que influenciam a produtividade dos programadores, desde as ferramentas e práticas adotadas, até o ambiente de trabalho e a gestão de projetos. Além disso, há de ser examinado como a aplicação integrada que foi desenvolvida busca aprimorar a produtividade dos desenvolvedores de *software* no contexto do modelo PJ.

Ao entender e avaliar os pilares da produtividade na programação, há um melhor preparo para identificar como a aplicação pode efetivamente contribuir para melhorar a eficiência e a qualidade do trabalho e de vida dos programadores.

2.1.1 Conceito de performance para programadores

As definições de produtividade frequentemente buscam quantificar a eficiência por meio de uma ou mais métricas, assim como o termo "performance"é definido pelo Michaelis (2023) como "Conjunto de fatores que determinam o desempenho de algo". Essas definições são essenciais para a compreensão dos conceitos subjacentes à eficiência e ao desempenho, particularmente quando aplicadas ao contexto da produtividade no desenvolvimento de *software*.

No aspecto de determinação de métricas para definir a produtividade de um programador, ainda não existe um consenso. Existem diversos artigos com o objetivo central de tentar definir o conceito de performance para programadores. Meyer et al. (2014) destaca a ampla utilização de métricas objetivas no desenvolvimento de

software, como a contagem de linhas de código escritas em um período determinado e o número de tarefas completadas por mês, entre outras. No entanto, é importante reconhecer que tais métricas nem sempre conseguem capturar com precisão o desempenho em contextos diversos. Além disso, muitas das vezes, a percepção dos programadores sobre sua própria eficiência não recebe a devida atenção em estudos.

A avaliação da produtividade no desenvolvimento de *software* é um desafio complexo. Métricas puramente quantitativas podem negligenciar a qualidade do código e não refletir a complexidade inerente a diferentes tarefas. Portanto, é fundamental considerar não apenas o que pode ser medido, mas também o que os desenvolvedores percebem como eficaz.

2.1.1.1 Interrupções e mudanças de contexto

A pesquisa de DeMarco e Lister (1985) evidenciou, por meio de um experimento, que o ambiente de trabalho e a organização desempenham papéis cruciais na performance dos programadores. Este estudo destacou que fatores como interrupções, o tamanho do espaço de trabalho e o nível de ruído ambiental têm um impacto direto na eficiência da execução de tarefas. Além disso, esses elementos influenciam diretamente a redução da frustração ao longo da execução das tarefas. Ainda segundo o experimento, um ambiente de trabalho apropriado, com o espaço adequado e sereno, promove um cenário propício para a concentração, criatividade e produtividade. Essa condição se reflete em uma maior satisfação dos profissionais e em resultados de trabalho mais eficazes.

O estudo conduzido por Meyer et al. (2014) lança luz sobre uma questão recorrente no universo dos programadores. Enquanto os desenvolvedores demonstram uma preferência por manter um ambiente de trabalho livre de interrupções e mudanças frequentes de contexto, os dados coletados revelam uma realidade diferente, na qual essas mudanças ocorrem de forma frequente. Surpreendentemente, segundo os autores em seu experimento, é apontado que, apesar dessas interrupções constantes, os desenvolvedores ainda se veem como produtivos. Essa aparente contradição é explicada pela presença de um fator-chave: o custo variável associado às diferentes formas de mudança de contexto.

Somado a isso, a pesquisa conduzida por Mark, Gudith e Klocke (2008) oferece uma perspectiva esclarecedora sobre o custo das interrupções no ambiente de trabalho. Os dados coletados revelam uma descoberta notável: desenvolvedores cujas tarefas foram interrompidas conseguiram concluir suas atividades em um período mais curto, sem comprometer a qualidade.

Os resultados deste estudo apontam para uma reação interessante das pessoas diante das interrupções. Ao se depararem com essa situação, os profissionais

tendem a compensar o tempo perdido, trabalhando em um ritmo mais acelerado. Entretanto, como os autores alertam, essa vantagem aparente vem acompanhada de um preço a ser pago: um aumento significativo do estresse, maiores níveis de frustração, pressão do tempo intensificada, esforço adicional e um acréscimo na carga de trabalho.

As interrupções, portanto, constituem um dilema intrincado no contexto da produtividade dos desenvolvedores de *software*. Elas podem oferecer ganhos momentâneos, mas, se não forem gerenciadas adequadamente, também podem acarretar consequências negativas de longo prazo, afetando a saúde mental e a qualidade do trabalho.

2.1.1.2 Gerenciamento de tempo

De acordo com Aeon, Faber e Panaccio (2021), o gerenciamento de tempo, de maneira geral, emerge como um fator crítico que impulsiona não apenas a performance no ambiente de trabalho e a conquista acadêmica, mas também o bem-estar em sua totalidade. Quando aplicado com habilidade e discernimento, o gerenciamento de tempo tem o potencial de gerar um impacto expressivo na produtividade e eficiência durante a execução de tarefas. Ainda segundo os autores, o gerenciamento de tempo não é uma abordagem única que serve para todos os contextos. Pelo contrário, é uma habilidade flexível que deve ser adaptada a cada situação, levando em consideração as características específicas do trabalho, as metas pessoais e os limites individuais. Além disso, quando realizado com êxito, o gerenciamento de tempo não apenas aprimora a eficiência, mas também contribui para uma sensação de realização e equilíbrio entre a vida pessoal e profissional.

2.2 IAG, LLM e Sistemas Integrados

Nos últimos anos, a IA tem vivenciado um crescimento exponencial, moldando e revolucionando diversas áreas da sociedade. Esse crescimento está intrinsecamente ligado ao aumento da capacidade computacional, disponibilidade de dados em grande escala e avanços nas técnicas de aprendizado de máquina, como comentado por Sejnowski (2023). Como resultado, a IA tem encontrado aplicações em uma ampla gama de campos, oferecendo soluções inovadoras e eficazes.

2.2.1 IAG

Russell e Norvig (2010) definem a IA como a capacidade das máquinas de emular e simular os comportamentos cognitivos humanos. Isso engloba a habilidade de resolver problemas e adquirir conhecimento. Eles destacam que, quando a IA al-

cança seu máximo desenvolvimento, ela demonstra a capacidade de tomar decisões lógicas, aproximando-se da capacidade cognitiva humana.

Além disso, existe um conjunto significativo de evidências que sugerem que uma máquina abstrata e generativa, quando implementada de maneira apropriada, pode resultar em sistemas flexíveis capazes de operar independentemente em uma ampla variedade de situações e ambientes distintos de forma independente.

Conforme define Russell e Norvig (2010):

> A Inteligência Artificial Generativa se define como o campo da ciência que estuda a construção (totalmente) automatizada da inteligência. Isso contrasta com a IA contemporânea, que estuda a compreensão e construção da inteligência pelos seres humanos.

Resumidamente, a IAG abrange uma categoria de modelos e ferramentas de IA especializadas na criação de conteúdo novo, que pode incluir imagens, texto, música e até mesmo código.

2.2.2 Sistemas Integrados com LLM

Avanços no desenvolvimento e capacidade das LLM possibilitaram que a geração de código automático fosse possível em contextos de tarefas de programadores. Segundo Sejnowski (2023), LLM são modelos de redes neurais de IA que têm a capacidade de processar e gerar textos em linguagem natural. Para isso, esses modelos utilizam bases de dados massivas, realizando um treinamento para performar uma tarefa, permitindo que a IA reconheça palavras e contextos, realize traduções, predições, entre outros tipos de texto.

Empresas líderes no setor de tecnologia investem em inovação e tecnologia, buscando melhorar seus produtos e aumentar sua lucratividade. Entre as mais significativas recentes inovações, temos a integração de IA em diversas aplicações, ferramentas e, até mesmo, a disponibilidade dessas serviços através de APIs.

2.2.2.1 Github Copilot

Esse é o caso da *Microsoft*, que desenvolveu uma ferramenta chamada *Github Copilot*, uma aplicação que utiliza uma ferramenta de geração de código baseada em LLM para auxiliar programadores no desenvolvimento de aplicações. Segundo Velazco (2023), a grande empresa da área de tecnologia Microsoft, está desenvolvendo uma nova aplicação para usuários do seu sistema operacional, o Windows. O Windows Copilot, segundo o autor, será um bate-papo com uma IA, que adotou o papel de suporte técnico do computador do usuário. Quaisquer ações que se queira fazer ou saber sobre o seu computador, poderiam ser realizadas ou adquiridas com

uma simples mensagem. Já o *Github Copilot* faz recomendações de código baseadas no contexto do projeto, considerando suas estruturas e ferramentas sendo utilizadas. Como o nome diz, a ferramenta adota um papel de "copiloto"no desenvolvimento, cuja a função do desenvolvedor é aceitar ou não as sugestões propostas.

Atualmente, ela pode ser facilmente integrada diretamente em editores de códigos e IDEs amplamente utilizados, como o *Visual Studio Code* (VSCode), *Visual Studio*, *Neovim*, e *JetBrains IDEs*. Os resultados do estudo feito por Peng et al. (2023), sugerem que a utilização do *Github Copilot* aumenta significativamente a produtividade dos desenvolvedores. Segundo dados do estudo, os programadores que utilizaram a ferramenta durante o desenvolvimento do código conseguiram concluir as tarefas propostas 55,8% mais rápidas se comparados ao grupo que não utilizou. Apesar de os dados não avaliarem a qualidade do código desenvolvido, o ganho em produtividade e velocidade é notável.

Outro estudo feito por Vaithilingam, Zhang e Glassman (2022), demonstrou que, apesar de os resultados não terem demonstrado necessariamente uma redução no tempo para uma tarefa ser completada, a maioria dos desenvolvedores participantes expressaram grande preferência pelo uso da ferramenta ao desenvolvê-las, o que representa ganhos em satisfação e percepção de produtividade por parte dos programadores.

2.2.2.2 ChatGPT

Uma da principais ferramentas utilizadas na aplicação é o ChatGPT. Ele é um modelo de linguagem desenvolvido pela OpenAI, e foi projetado para compreender e gerar texto em linguagem natural. De acordo com OpenAI (2023), ele é programado para responder perguntas, fornecer informações, ajudar com tarefas e interagir em conversas sobre uma ampla variedade de tópicos, seguindo instruções através de textos providos pelo usuário e gerando respostas detalhadas.

O ChatGPT é uma ferramenta útil para desenvolvedores, pois ajuda a esclarecer conceitos de programação, depurar código, fornecer exemplos de código, explicar tópicos de aprendizado de máquina e IA, oferecer suporte técnico, gerar documentação e até mesmo criar protótipos rápidos.

Segundo OpenAI (2023), a empresa por trás do desenvolvimento do modelo GPT-3, oferece uma API que permite que desenvolvedores integrem os modelos GPT em suas próprias aplicações e sistemas. Através dessa API, é possível acessar e usar os modelos GPT para várias tarefas relacionadas à geração de texto e processamento de linguagem natural. Ainda segundo o autor, os GPT são treinados para entender linguagem natural e código, e como eles podem gerar texto em resposta a entradas, chamadas de *prompts*. Além disso, ele menciona várias aplicações possíveis para

esses modelos, como redação de documentos, escrita de código, respostas a perguntas, análise de texto, criação de agentes de conversação, tradução de idiomas, entre outras.

2.3 Git e Github

A gestão de código-fonte desempenha um papel crucial no mundo do desenvolvimento de *software* e em projetos de pesquisa acadêmica. Manter um registro claro e organizado das alterações no código, permitir a colaboração eficaz e garantir a rastreabilidade são elementos-chave na busca de resultados bem-sucedidos e na validação de descobertas. Duas ferramentas que se destacam nesse cenário são o Git e o GitHub.

2.3.1 Git: Controle de Versões Distribuídos

O *Git* é um sistema de controle de versão distribuído, amplamente adotado. Ele permite que os desenvolvedores acompanhem e gerenciem as mudanças no código-fonte de forma eficiente. Como citado por Blischak, Davenport e Wilson (2016), é possível fazer o controle de versão de diversos tipos de arquivos em um repositório *Git*, como textos, imagens, entre outros. Algumas das principais características do *Git* incluem:

- **Histórico Detalhado:** o *Git* registra cada alteração no código, proporcionando um histórico detalhado do desenvolvimento do projeto. Isso é essencial para a rastreabilidade e a documentação;
- ***Branching* e *Merging*:** o *Git* permite a criação de ramificações (*branches*) para desenvolver recursos ou correções separadamente, facilitando a colaboração em equipe. Posteriormente, essas ramificações podem ser mescladas (*merged*) para consolidar as alterações;
- **Controle de Conflitos:** o *Git* oferece ferramentas para resolver conflitos entre diferentes ramificações, garantindo a integridade do código; e
- **Acesso Remoto:** repositórios *Git* podem ser acessados e colaborados por meio da internet, tornando-os ideais para equipes distribuídas geograficamente.

2.3.2 *Github*: Plataforma de Colaboração

O *GitHub*, por outro lado, é uma plataforma que hospeda repositórios *Git*, além de oferecer funcionalidades adicionais para colaboração e gerenciamento de projetos. Suas principais características incluem:

- **Hospedagem de Código Aberta:** projetos de código aberto podem ser hospedados no *GitHub* gratuitamente, tornando-o um ponto de encontro para a comunidade de desenvolvedores e pesquisadores;
- **Colaboração Eficiente:** o *GitHub* simplifica a colaboração, permitindo que vários desenvolvedores trabalhem juntos em um único projeto. Ele oferece ferramentas para revisão de código (*code review*), rastreamento de problemas (*issue tracking*) e integração contínua (CI/CD);
- **Visibilidade e Transparência:** repositórios públicos no GitHub oferecem visibilidade e transparência, permitindo que outros pesquisadores e colaboradores examinem, validem e contribuam para o seu trabalho; e
- **Backup e Segurança:** o GitHub fornece um backup seguro dos repositórios, protegendo o código contra perdas acidentais.

No contexto de um TCC ou pesquisa acadêmica, o Git e o GitHub desempenham um papel crucial. Eles permitem manter um registro claro de todas as alterações feitas no seu código, tornando a sua pesquisa rastreável e bem documentada. Além disso, facilitam a colaboração com orientadores, colegas e outros pesquisadores, tornando o processo de revisão e validação mais eficiente.

3 Ferramentas e Métodos

Este capítulo discorre sobre as tecnologias e métodos escolhidos, detalhando e especificando seus funcionamentos. Além disso, padrões de código e boas práticas para garantir a qualidade, manutenibilidade e a escalabilidade do sistema serão explicadas com detalhes nesse capítulo. Em suma, é apresentada uma visão aprofundada do processo de desenvolvimento da aplicação, destacando a estrutura dos módulos, os padrões de código e as boas práticas aplicadas, bem como as ferramentas que desempenharam um papel crucial.

3.1 Ferramentas Gerais

Considerando o contexto descrito, as ferramentas utilizadas para o desenvolvimento do projeto foram escolhidas como descrito no Quadro 1 e Quadro 2:

Quadro 1 – Software

Sistema Operacional	Windows 11 Home para desktop
Ambiente Integrado de Desenvolvimento (IDE)	*Microsoft Visual Studio Code* (VSCode)
Linguagem de Programação	*Javascript* com *Typescript*
Runtime	NodeJS na versão 18.16.0
Frameworks	NextJS na versão 13.5.2
APIs	Drive API, ChatGPT API

Fonte: Elaborada pelo autor.

Quadro 2 – Hardware

Modelo do notebook	*Samsung Expert X50*
Processador	*Intel(R) Core(TM) i7-8550U*
Memória Primária	20 GB de Memória RAM DDR4 2666 *SODIMM*
Memória Secundária	*HDD* de 1TB, *SSD Crucial BX500* de 240GB
Placa de Vídeo	*NVIDIA GeForce MX110*

Fonte: Elaborada pelo autor.

A principais ferramentas utilizadas ao longo do desenvolvimento da aplicação serão descritas a seguir, como *Javascript*, *Typescript*, *NestJS*, *NextJS*, *Google Drive API*, *Google Auth*, *ChatGPT API*.

3.1.1 JavaScript

Segundo a Documentação disponibilizada por MDN (2023), JS é uma linguagem de programação interpretada, conhecida como a linguagem *script* para páginas *web*, mas que também é utilizada em diversos outros ambientes sem *browser*. O JS é uma linguagem baseada em protótipicos, multiparadigma e dinâmica.

De acordo com Flanagan (2012), a maioria dos sites modernos usa JS e, todos os navegadores modernos incluem interpretadores JS. Ainda segundo o autor, JS, CSS e HTML seriam a "trindade"da internet atual.

3.1.1.1 TypeScript

De acordo com Bierman, Abadi e Torgersen (2014), *TypeScript* é um *superset* da linguagem JS, ou seja, é uma extensão da linguagem, compartilhando a mesma base sintática do JS, mas adicionando estruturas e funcionalidades extras, com o objetivo de possibilitar um desenvolvimento facilitado de aplicações de grande escala. Ele complementa dizendo que, apesar do seu sucesso, JS não é uma boa linguagem para desenvolvimento e manutenção de grandes aplicações, o *TypeScript* tem o objetivo de resolver essa deficiência. Ainda segundo o autor, o *TypeScript* não tem a inteção de ser uma nova linguagem de programação, mas enriquecer o JS com sistema de módulos, classes, interfaces e um sistema de tipagem estática, sendo um suporte para ele. A utilização desse sistema de tipagem no *TypeScript* ajuda o programador na busca por erros, e permite um maior suporte da IDE utilizada, adicionando, por exemplo, sugestões de métodos que um objeto pode chamar.

3.1.2 NodeJS

Como definido por Foundation (2023), NodeJS é um ambiente de execução da linguagem de programação Javascript (JS). Com a grande popularidade do JS, surgiu a vontade por parte dos desenvolvedores de desenvolver aplicações fora do ambiente do navegador, o que era uma limitação da Linguagem. O NodeJS foi criado com o objetivo de permitir que o JS seja utilizado em ambientes fora do navegador, podendo, até mesmo, ser usado como uma aplicação do lado servidor.

3.1.3 NestJS

De acordo com Mysliwiec (2023), *NestJS* é uma *framework* para construção de aplicações eficientes e escaláveis com *NodeJS* no lado servidor. É utilizado JS progressivo com ele, e toda a sua base é estruturada para ter suporte total ao *Typescript*. Essa foi a ferramenta escolhida para o desenvolvimento do *"backend"* da aplicação, ou seja, o lado servidor, no qual as requisições serão feitas.

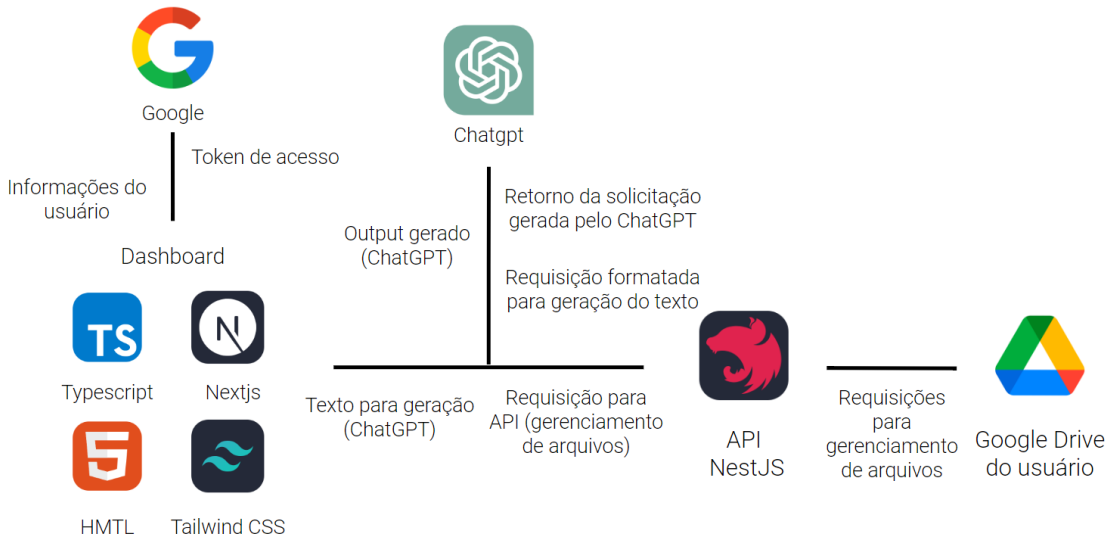
3.1.4 NextJS

Como define Vercel (2023b), o NextJS é um *framework* que utiliza a biblioteca *React* em sua base, com o intuito de construir aplicações *full-stack*, ou seja, do lado cliente e servidor. O *NextJS* utiliza um sistema modular, sua estrutura é baseada em componentes reutilizáveis, facilitando a construção de aplicações escaláveis.

4 Desenvolvimento

Nessa etapa será descrito o desenvolvimento da aplicação, suas funções e detalhes de implementação. O desenvolvimento foi feito seguindo uma abordagem modular, visando a otimização de funcionalidades. Cada módulo foi projetado para atender funções especificas, permitindo uma implementação estruturada e organizada. Essa divisão modular não apenas simplifica o processo de desenvolvimento, mas também promove a clareza e a manutenibilidade do sistema. A Figura 1, representa a estrutura, conexões e tecnologias utilizadas em cada módulo.

Figura 1 – Fluxograma da estrutura da aplicação



Fonte: Elaborada pelo autor.

A seguir, serão descritos em detalhes os quatro módulos principais que compõem a aplicação: Autenticação, Gerenciamento de Arquivos, IA e *Dashboard*. Cada um desses módulos desempenha um papel fundamental e estrutural no desenvolvimento:

- **Autenticação**: este módulo aborda a autenticação de usuários, permitindo o acesso seguro à aplicação. É explorada a implementação de métodos de autenticação, bem como medidas de segurança implementadas e integração do serviço de autenticação da *Google*;

- **Gerenciamento de Arquivos:** este módulo concentra-se na organização e gestão de arquivos relacionados a projetos de desenvolvimento. É detalhado como os usuários podem armazenar, recuperar e gerenciar arquivos relacionados aos seus projetos dentro e fora do sistema, por meio da integração com o *Google Drive*;
- **IA:** este módulo baseia-se na integração de ferramentas de IAGs para fornecer assistência no desenvolvimento de código e anotações, auxiliando em, por exemplo, geração de resumos de textos, melhorias e identificação de problemas em códigos, e utilização de um *Chatbot*. Será explorado o que é uma IAG, como a IA é integrada e como ela pode otimizar os processos de desenvolvimento; e
- ***Dashboard*:** o módulo de *Dashboard* é responsável por ser a interface gráfica, na qual o usuário do sistema interage, sendo o componente centralizador de todos os módulos e funcionalidades do projeto. Será discutido como a interface do usuário é projetada para melhorar a usabilidade e a produtividade dos desenvolvedores.

4.1 Módulo de Autenticação

No desenvolvimento da aplicação integrada, um elemento fundamental é o Módulo de Autenticação. Este módulo desempenha um papel crucial na garantia da segurança, praticidade e facilidade de uso para os usuários. Aqui, há de se explorar, como é utilizado o *Next Auth* para implementar uma autenticação eficaz e integrada com o *Google*.

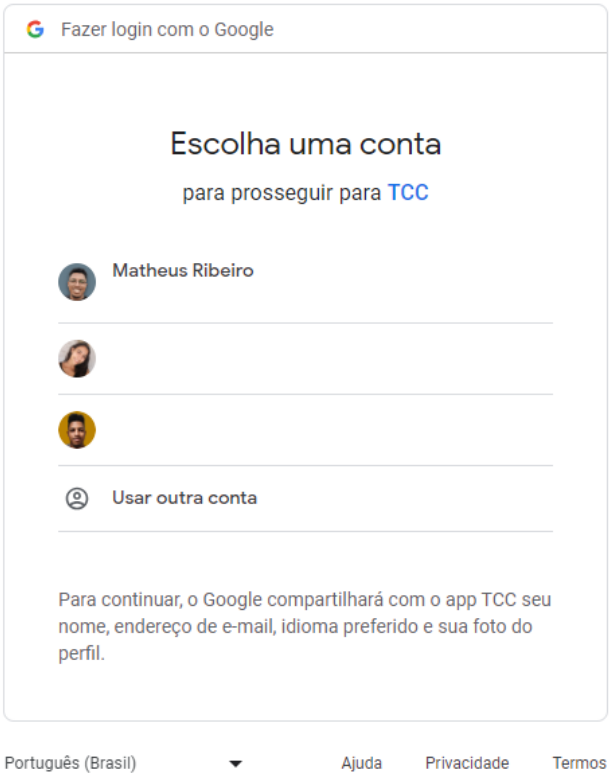
Para o desenvolvimento da *dashboard*, foi utilizado o *framework NextJS*, cujos detalhes foram especificados na subseção 3.1.4. Foi utilizada para integração do sistema de autenticação, a solução *Next Auth*, feita para ser empregada em projetos que usam o Nextjs. Como especifica Collins (2023), a ferramenta possui suporte para diversos sistemas de autenticação amplamente usados, como o do *Google*, que possui um provedor exclusivo.

4.1.1 Acesso com o Google

Neste módulo foi feita a escolha pela integração do serviço de autenticação do *Google*, tornando a experiência do usuário mais simplificada. Isso elimina a necessidade de criar uma nova conta com credenciais adicionais, uma vez que basta possuir uma conta no ecossistema do *Google*. Quando um usuário já estiver autenticado e logado em seu navegador com sua conta *Google*, não será necessário fornecer credenciais novamente. Assim, basta escolher qual conta será escolhida, conforme é

mostrado na Figura 2. Isso simplifica ainda mais o processo de acesso à aplicação, com o adicional de delegar a autenticação e gerenciamento de credencias para o robusto sistema de autenticação da empresa, aumentando a segurança.

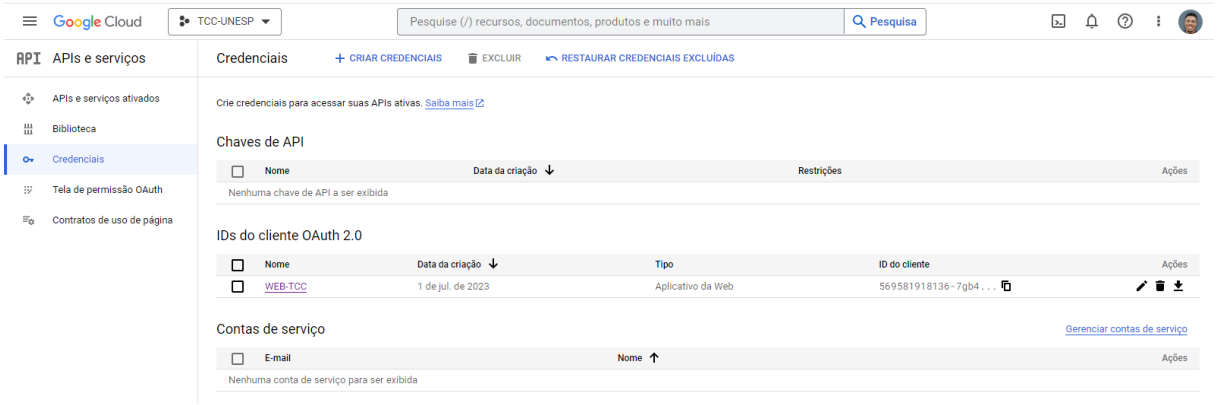
Figura 2 – Captura de tela da aplicação - Login com o Google



Fonte: Elaborada pelo autor.

Como apresentado por Google (2023), para realizar o acesso no sistema com a conta *Google*, foi necessário fazer a criação e configuração de um cliente na plataforma de nuvem da empresa, o *Google Cloud*, como é mostrado na Figura 3.

Figura 3 – Captura de tela de configuração do cliente no site do Google Cloud



Fonte: Elaborada pelo autor.

A utilização de credenciais válidas é confirmada pela geração de acesso da aplicação, usualmente chamado de *token* de acesso. Na aplicação, foi implementada a lógica para validar se o usuário está autenticado, utilizando o *token* de acesso, considerando que ele só é gerado após o usuário e suas credenciais serem autenticadas. Na Figura 4, encontra-se o código de um *token* exemplo. Caso o usuário esteja autenticado, é permitido o acesso e a aplicação realiza o redirecionamento para a tela principal. Caso contrário, o usuário é redirecionado para a tela de login novamente.

Cada informação do *token* de acesso é importante e pode ser utilizada em diversos contextos. Como mostra o exemplo da Figura 4. As informações básicas do usuário são nome, e-mail e endereço para visualizar a imagem perfil. Essas são utilizadas em diversas funções da aplicação, como a imagem de perfil sendo mostrada na barra lateral da esquerda, demonstrada na Figura 9.

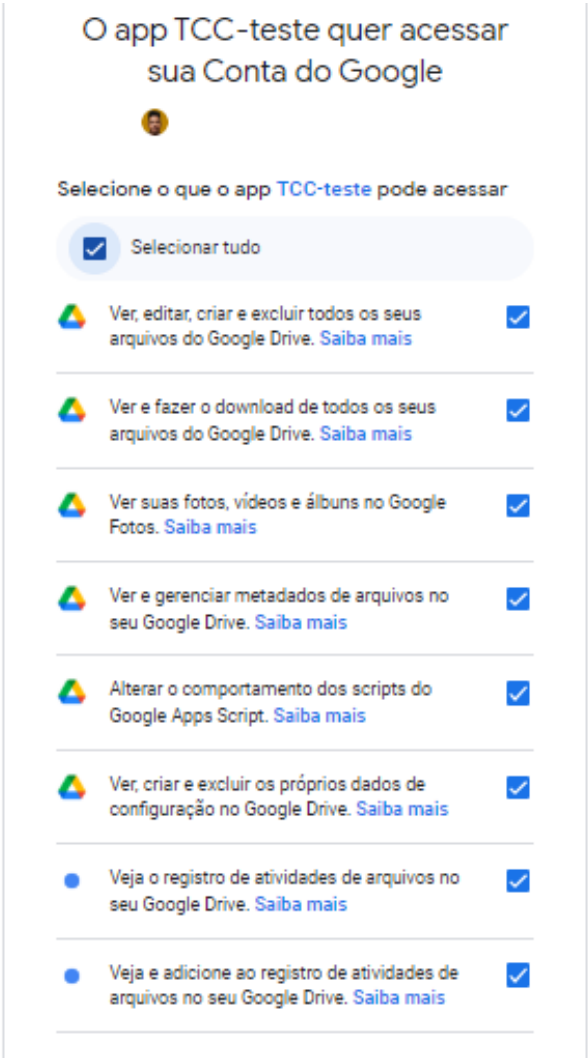
Figura 4 – Exemplo de token de acesso


```
{
  "token": {
    "name": "Nome Exemplo",
    "email": "exemplo@gmail.com",
    "picture": "https://lh3.googleusercontent.com/a/ACg8ocKdMk6HLoGFm
    "sub": "101313622516289128153",
    "access_token": "ya29.a0AfB_byCJZy3K9TFms2-B75a4z_QMib2XT7deMtf86
    "iat": 1695518828,
    "exp": 1698110828,
    "jti": "bc70caa1-5897-4a1f-9d66-4c47742359c8"
  }
}
```


Fonte: Elaborada pelo autor.

Caso seja a primeira entrada em determinada conta, um pedido de aceite de permissões é realizado para acesso de dados e funcionalidades para gerenciamento do serviço *Google Drive* do usuário, assim como mostra a Figura 5. Esse gerenciamento é feito pela aplicação e é explicado em detalhes na seção 4.2.

Figura 5 – Captura de tela de aceite de termos do Google Drive - Login com o Google



Fonte: Elaborada pelo autor.

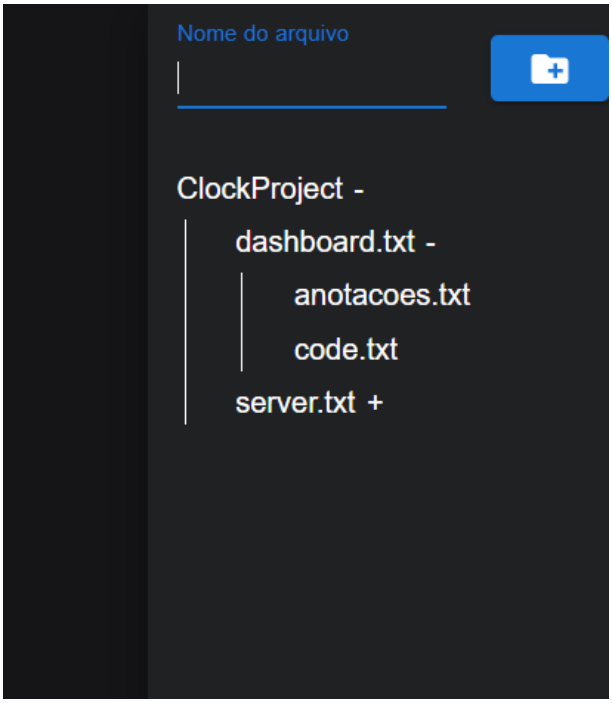
Após a configuração, as credenciais do cliente foram disponibilizadas, com informações importantes, como: identificador do cliente, identificador do projeto, Identificador Uniforme de Recursos (URI), provedor e segredo do cliente. Todas essas credencias são necessárias para que a aplicação integrada seja capaz de realizar e ter suas requisições validadas por parte do sistema da empresa.

4.2 Módulo de Gerenciamento de Arquivos

Por meio da *dashboard*, o usuário é capaz de criar, editar e excluir arquivos e pastas, sendo estas anotações, códigos ou outros arquivos de texto, como mostra a Figura 6. Com a integração do *Google Drive*, existe a alternativa de acessar esses arquivos diretamente do sistema do *Google*, podendo ser usado no computador, no celular e em outros dispositivos com acesso à internet.

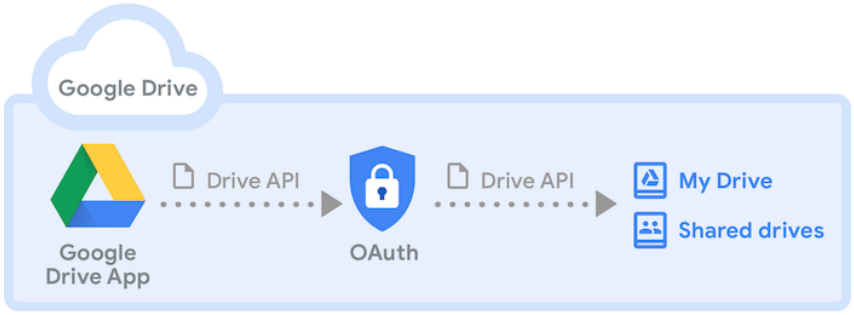
O *Google Drive* é um serviço de armazenamento de arquivos na nuvem, que oferece aos usuários um espaço de armazenamento pessoal chamado "Meu Drive". Além do serviço para usuários, uma API é disponibilizada pela empresa, sendo ela capaz de pegar, editar, criar e excluir arquivos do *Drive* de um ou mais usuários. A Figura 7 descreve o fluxo de requisições para gerenciamento dos arquivos do *Drive* através da API. É necessário que um *token* válido seja utilizado nas requisições para ser autenticado e autorizado pela API. Na Figura 8, é exposto um exemplo da estrutura de uma requisição para a API, sendo esta para a criação de uma pasta.

Figura 6 – Captura de tela da barra lateral direita, mostrando o sistema de gerenciamento de arquivos



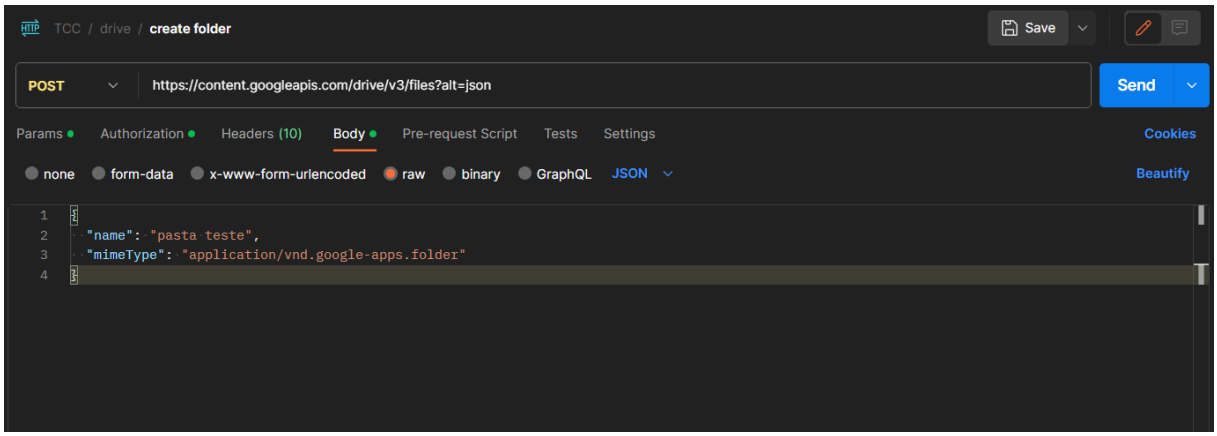
Fonte: Elaborada pelo autor.

Figura 7 – Imagem representando o fluxo de autenticação e requisições do serviço do Google Drive



Fonte: Google, 2023

Figura 8 – Captura de tela do exemplo de uma requisição para o Google Drive utilizando o *software* Postman



Fonte: Elaborada pelo autor.

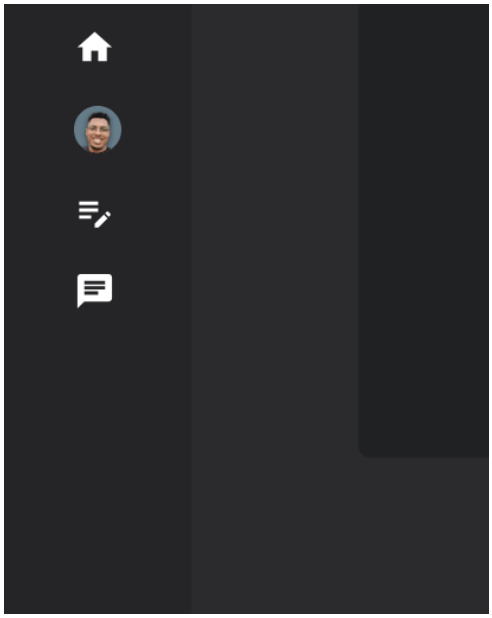
4.3 Módulo de IA

Para disponibilizar o *ChatGPT* como ferramenta auxiliar no projeto, foi utilizado o *Kit* de Desenvolvimento de *Software* (SDK) de IA da *Vercel*, o *Vercel AI SDK*. Como descreve a Vercel (2023a), essa ferramenta é uma biblioteca para construção de aplicações com serviços de texto e interfaces de *chat* alimentadas por IA, com o intuito de ajudar programadores no desenvolvimento de integrações como o *ChatGPT* em projetos baseados em JS.

Na aplicação, são disponibilizadas duas formas de utilização do serviço de IA do *ChatGPT*. A primeira por meio da barra lateral da esquerda, como mostra a Figura 9, na qual um *bot* de bate-papo pode ser acessado, permitindo que, ao utilizar o

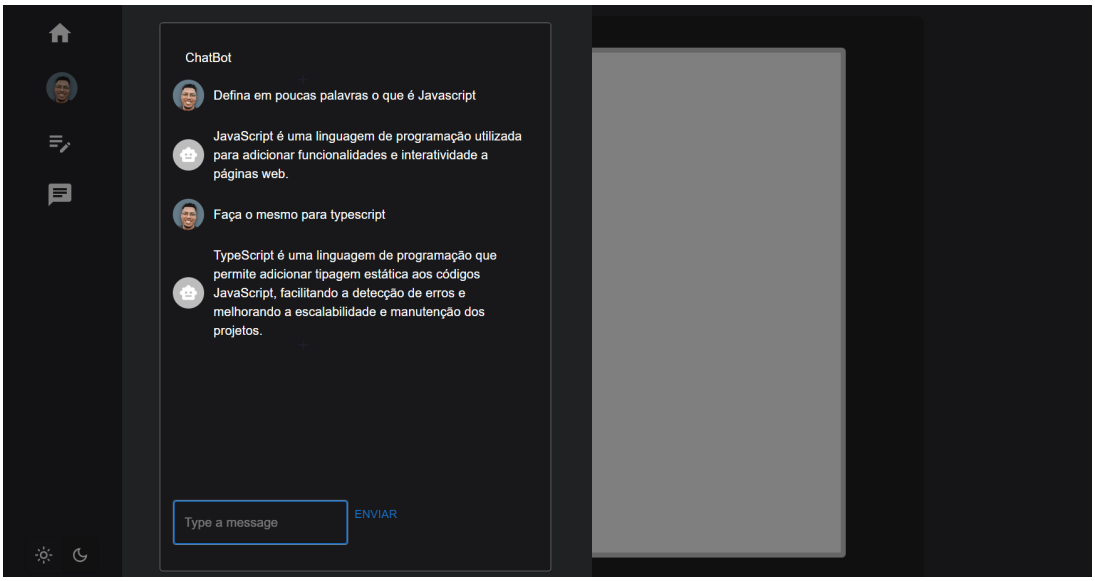
sistema, o usuário tenha a possibilidade de fazer perguntas sobre programação, pedir correções de ortografia em um texto, gerar resumos, separar e elencar os pontos mais importantes de qualquer tipo de texto, como mostra a Figura 10.

Figura 9 – Captura de tela da visualização superior da barra lateral esquerda



Fonte: Elaborada pelo autor.

Figura 10 – Captura de tela da barra lateral esquerda com a opção do chat aberta

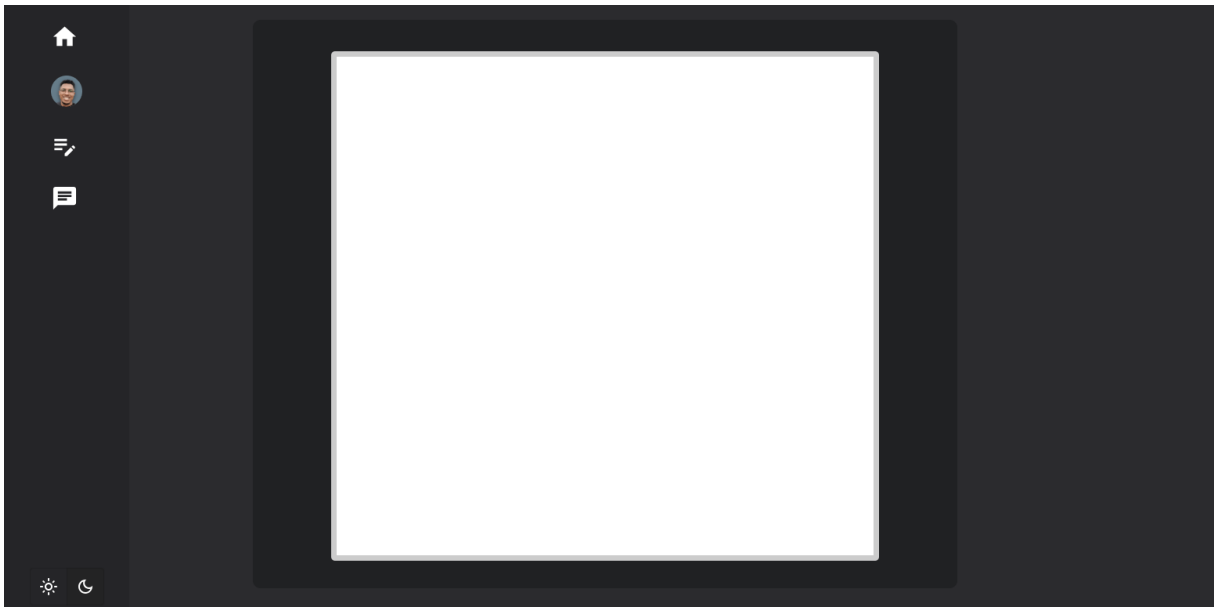


Fonte: Elaborada pelo autor.

4.4 Módulo de *Dashboard*

O módulo de *Dashboard* é o componente central da aplicação integrada para gerenciamento de projetos e recursos para desenvolvedores de *software*. Todos os módulos descritos anteriormente são integrados com ele. A Figura 11 demonstra a tela inicial da aplicação, quando o usuário está autenticado.

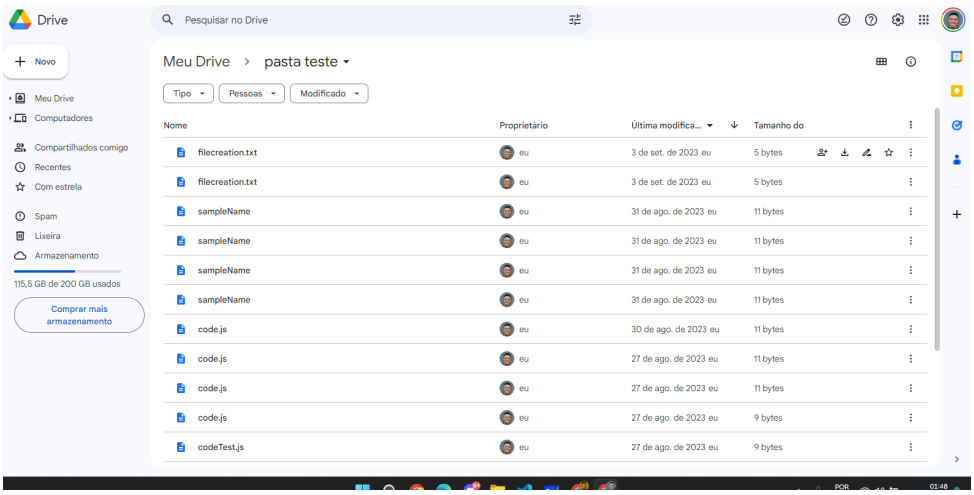
Figura 11 – Captura de tela exibindo os componentes principais da aplicação



Fonte: Elaborada pelo autor.

Na tela principal, existem três componentes centrais: as duas barras laterais (esquerda e direita), e o editor de textos e códigos localizado ao centro. A barra lateral esquerda contém quatro botões, sendo o primeiro utilizado para voltar a aplicação para o estado inicial, e o segundo com a foto de perfil do usuário, a qual é adquirida por meio do *token* de acesso, como descrito anteriormente. O terceiro abre a barra lateral direita, permitindo que o usuário da aplicação faça o gerenciamento dos arquivos, podendo criar, editar e excluir. Dessa forma, com a integração ao *Google* Drive, as ações realizadas nesse componente serão compartilhadas com o serviço do *Google*, como demonstrado no exemplo da Figura 12. O quarto botão permite o acesso ao bate-papo com a IA, servindo como ferramenta auxiliar.

Figura 12 – Captura de tela do Google Drive

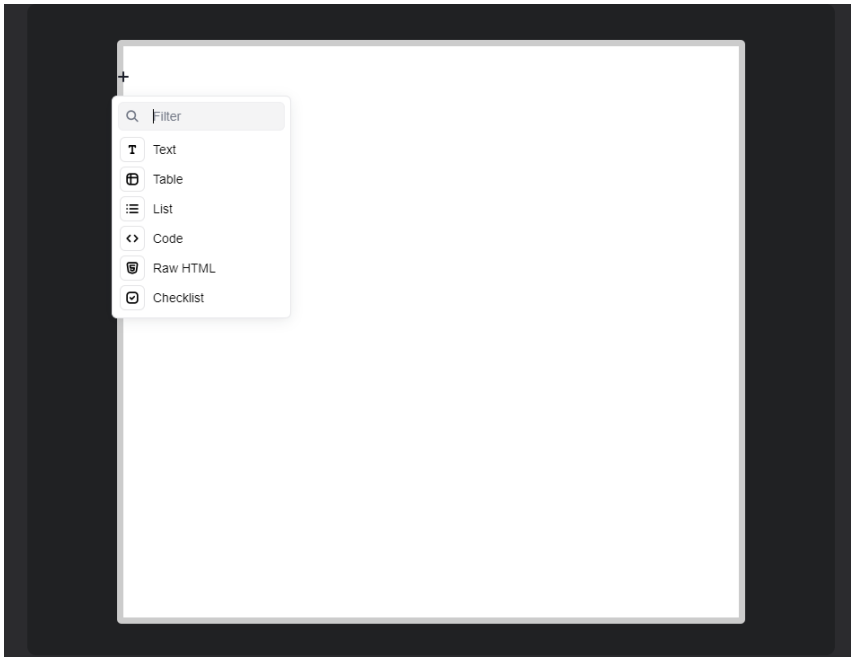


Fonte: Elaborada pelo autor.

4.4.1 Editor de Códigos e textos

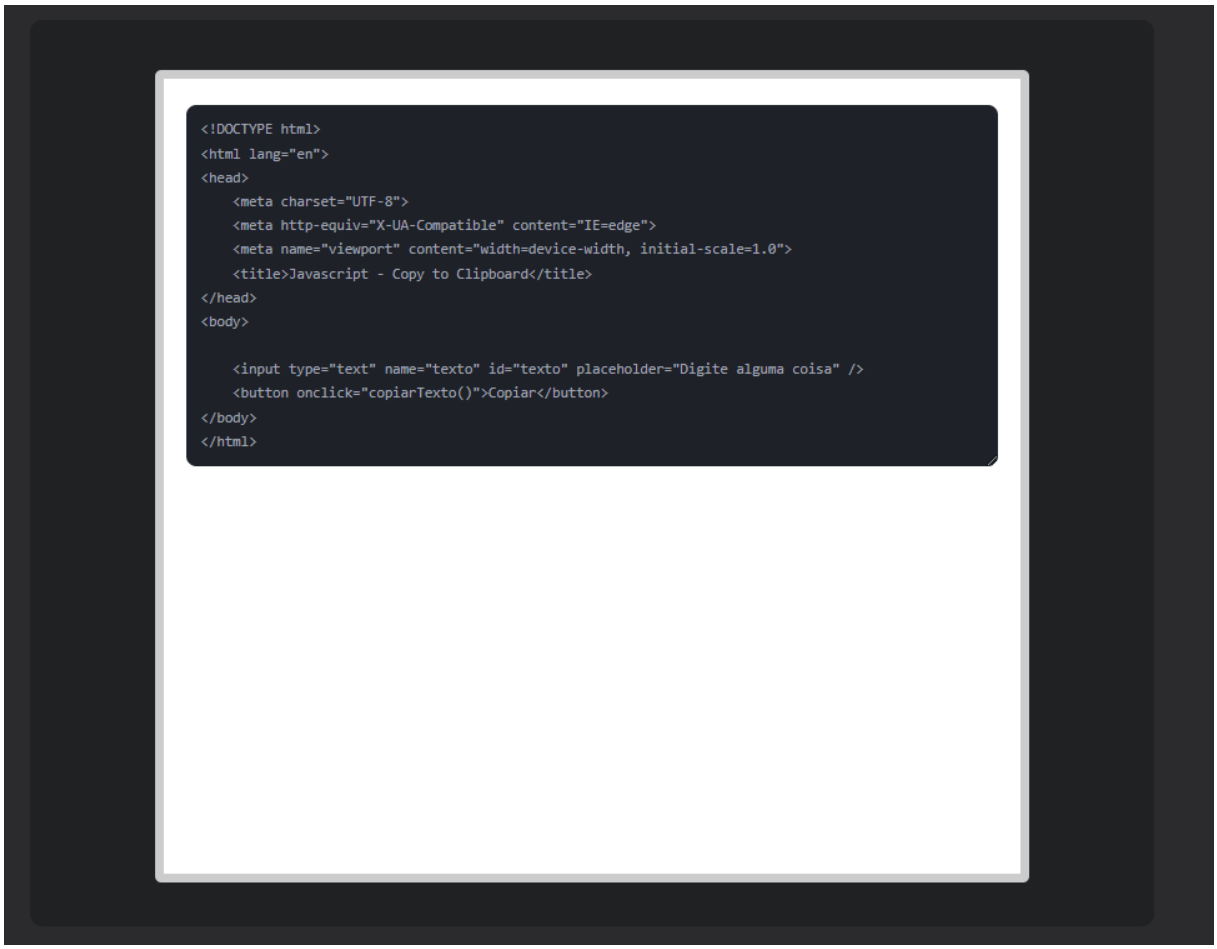
O componente central é responsável por mostrar e permitir a edição dos textos e códigos, fazer anotações e até mesmo gerar tabelas. Essas opções de componente podem ser encontradas em botões ao lado da caixa de texto, como mostra a Figura 13. Por meio das opções, é possível escolher o componente para escrever o código, como demonstrado na Figura 14.

Figura 13 – Visualização das opções de funções na edição de códigos e anotações



Fonte: Elaborada pelo autor.

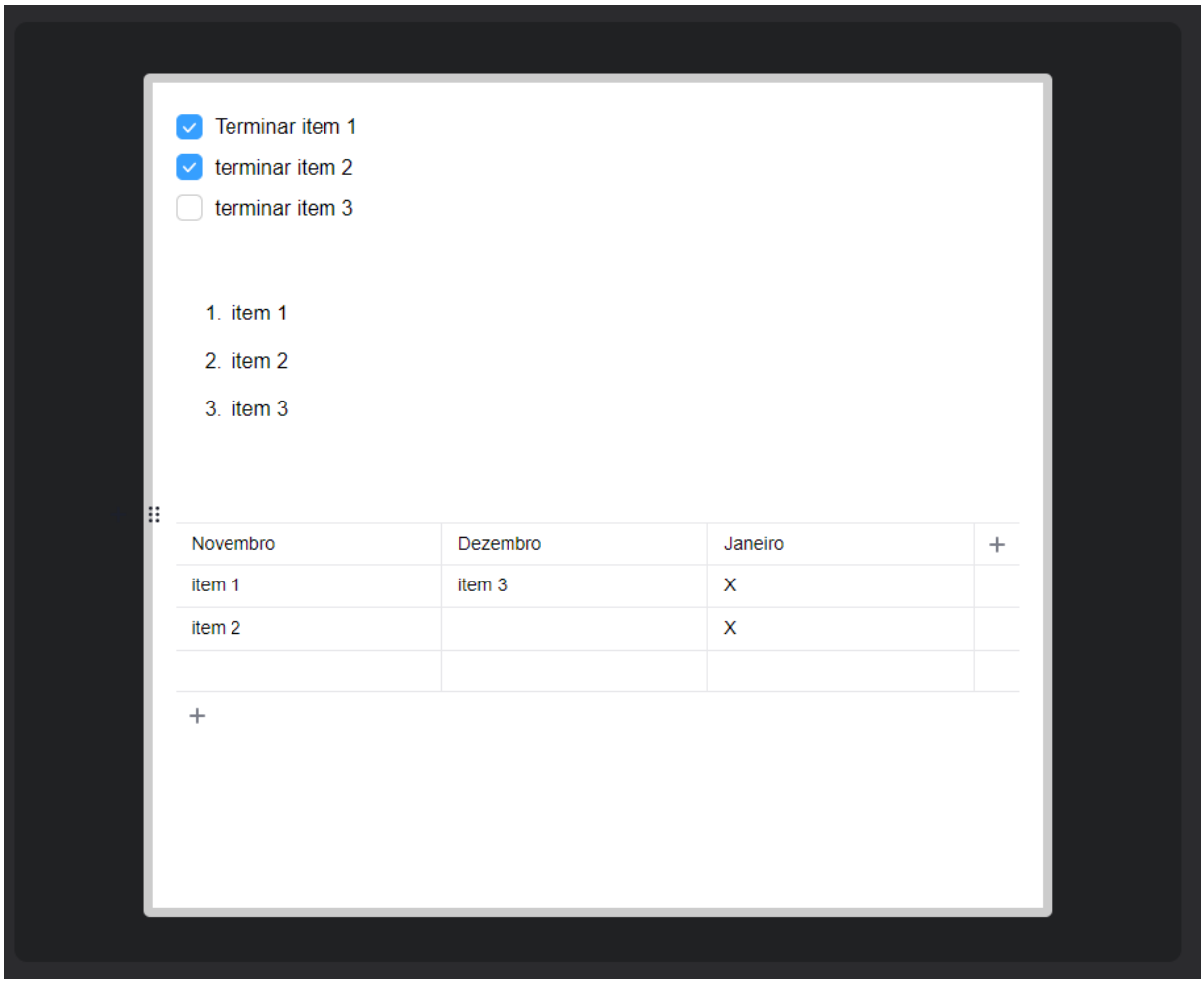
Figura 14 – Visualização da função de editor de código



Fonte: Elaborada pelo autor.

Existem outros recursos adicionais com o propósito de aprimorar a elaboração de anotações, organização e gerenciamento de projetos e códigos, conforme demonstrado na Figura 15. Essas ferramentas e funcionalidades complementares ampliam a capacidade de registro e estruturação de informações, permitindo uma gestão mais eficaz. Dessa forma, os desenvolvedores podem contar com um conjunto diversificado de instrumentos para otimizar sua organização.

Figura 15 – Visualização de algumas funções do componente



Fonte: Elaborada pelo autor.

4.4.2 Modo escuro e claro

Conforme define Dash e Hu (2021):

> O modo escuro pode ser visto como um caso especial de transformação de cores que reduz a luz da tela, alterando explicitamente as cores de fundo com tema branco, para cores com tema escuro.

Segundo Eisfeld e Kristallovich (2020), disponibilizar a versão de uma aplicação no modo escuro, ou seja, uma versão com uma paleta de cores escuras, aumenta significativamente a experiência do usuário em determinadas circunstâncias, especialmente em ambientes de baixa iluminação. Ainda segundo os autores, esse modo tem o potencial de economizar a bateria de dispositivos e aumentar o tempo de uso entre os carregamentos dos que usam bateria.

Considerando o contexto deste trabalho, de acordo com uma pesquisa online feita por Sarath (2016), em um site voltado aos desenvolvedores, 92% dos participante

demonstraram ter preferência pelo tema escuro em detrimento do claro em seu editores de código. No presente trabalho, foi desenvolvido um botão para que o usuário tenha a opção de escolher o modo de cores que deseja utilizar na aplicação, podendo escolher entre o claro ou o escuro.

5 Considerações Finais

O presente trabalho teve como objetivo o desenvolvimento de uma aplicação integrada voltada para atender às necessidades dos desenvolvedores de *software*. A aplicação proposta combina ferramentas de gerenciamento de projetos, recursos e serviços de IA para auxiliar em tarefas relacionadas à programação e gerenciamento de projetos pessoais e profissionais.

À medida em que os capítulos apresentados foram explorados, destacou-se a importância das tecnologias escolhidas, com foco no uso de JS e TypeScript, além da integração eficaz de serviços do Google, incluindo a API do Google Drive. Além disso, o controle de versões e a colaboração eficaz em relação ao uso do Git e do GitHub foi enfatizado, demonstrando como essas ferramentas desempenham um papel crucial na manutenção de um registro claro de todas as alterações feitas no código.

No contexto da autenticação do usuário e geração de *tokens* de acesso, destacou-se a importância da segurança. Ademais, a integração com a API do *Google Drive* foi um marco no aprimoramento do ambiente de gerenciamento de arquivos, possibilitando o acesso a eles em diversos dispositivos.

Somado a isso, a integração do módulo de IA, com foco no *ChatGPT*, adicionou uma camada de assistência aos desenvolvedores, oferecendo suporte na programação, revisões de texto, geração de resumos e outras tarefas relacionadas. Essa integração foi feita por meio do *Vercel AI SDK* e demonstra como a IA pode ser aplicada de maneira prática em projetos do mundo real.

Por fim, o módulo de *dashboard* forneceu aos desenvolvedores uma plataforma central para gerenciar projetos e recursos de maneira eficaz, demonstrando como todas as funcionalidades da aplicação se unem em um ambiente coeso.

Em síntese, essa aplicação integrada trata as dificuldades dos programadores, buscando o aumento de produtividade, foco e organização, centralizando funções e serviços amplamente utilizados em um único sistema. Portanto, a aplicação também é útil e otimiza o trabalho dos mais diversos tipos de desenvolvedores, tratando, em especifico, aqueles que prestam serviços para diversas companhias, pois os garante maior autonomia para gerirem e organizarem seus projetos.

5.1 Trabalhos Futuros

Considerando uma futura continuidade do trabalho, esses são alguns dos pontos que podem ser explorados:

- Integração de outras ferramentas do ecossistema Google, como a pesquisa (*Google Search*) e calendário (*Google Calendar*);
- Implementação de mais funcionalidades utilizando a API do ChatGPT;
- Implementação de mais métodos de Login, logar com Facebook e Github, por exemplo;
- Disponibilizar ferramentas colaborativas para múltiplas pessoas realizarem o gerenciamento de códigos e anotações simultaneamente; e
- Realização de testes e comparações com outros sistemas integrados com propostas similares.

Referências

ABES. *Estudo Mercado Brasileiro de Software Panorama e Tendências 2023*. 2023. Disponível em: https://abes.com.br/dados-do-setor/. Acesso em: 10 de outubro de 2023.

AEON, B.; FABER, A.; PANACCIO, A. Does time management work? a meta-analysis. *PloS one*, Public Library of Science San Francisco, CA USA, v. 16, n. 1, p. e0245066, 2021.

ANDREESSEN, M. Why software is eating the world. *Wall Street Journal*, v. 20, n. 2011, p. C2, 2011.

BIERMAN, G.; ABADI, M.; TORGERSEN, M. Understanding typescript. In: SPRINGER. *ECOOP 2014–Object-Oriented Programming: 28th European Conference, Uppsala, Sweden, July 28–August 1, 2014. Proceedings 28*. [S.l.], 2014. p. 257–281.

BLISCHAK, J. D.; DAVENPORT, E. R.; WILSON, G. A quick introduction to version control with git and github. *PLoS computational biology*, Public Library of Science, v. 12, n. 1, p. e1004668, 2016.

BRASIL. *Consolidação das Leis do Trabalho*. Diário Oficial da União, 1943. Disponível em: https://www.planalto.gov.br/ccivil_03/decreto-lei/Del5452.htm. Acesso em: 24 de outubro de 2023.

BRASIL. *Lei da Terceirização (Lei nž 13.429, de 31 de março de 2017)*. Diário Oficial da União, 2017. Disponível em: https://www.planalto.gov.br/ccivil_03/_ato2015-2018/2017/lei/L13429.htm. Acesso em: 24 de outubro de 2023.

COLLINS, I. *Documentação Next Auth*. 2023. Disponível em: https://next-auth.js.org/getting-started/introduction. Acesso em: 21 de outubro de 2023.

DASH, P.; HU, Y. C. How much battery does dark mode save? an accurate oled display power profiler for modern smartphones. In: *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services*. [S.l.: s.n.], 2021. p. 323–335.

DEMARCO, T.; LISTER, T. Programmer performance and the effects of the workplace. In: *Proceedings of the 8th international conference on Software engineering*. [S.l.: s.n.], 1985. p. 268–272.

EISFELD, H.; KRISTALLOVICH, F. *The rise of dark mode: A qualitative study of an emerging user interface design trend*. 2020.

FLANAGAN, D. *JavaScript: o guia definitivo*. [S.l.]: Bookman Editora, 2012.

FOUNDATION, T. O. *NodeJS documentation*. 2023. Disponível em: https://nodejs.org/en. Acesso em: 23 de outubro de 2023.

GOOGLE. *Login do Google para Web*. 2023. Disponível em: https://developers.google.com/identity/sign-in/web/sign-in?hl=pt-br. Acesso em: 23 de outubro de 2023.

HARMAN, M. The role of artificial intelligence in software engineering. In: IEEE. *2012 First International Workshop on Realizing AI Synergies in Software Engineering (RAISE)*. [S.l.], 2012. p. 1–6.

MARK, G.; GUDITH, D.; KLOCKE, U. The cost of interrupted work: more speed and stress. In: *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. [S.l.: s.n.], 2008. p. 107–110.

MDN. *Documentação JavaScript*. 2023. Disponível em: https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference. Acesso em: 21 de outubro de 2023.

META. *Introducing New AI Experiences Across Our Family of Apps and Devices*. 2023. Disponível em: https://about.fb.com/news/2023/09/introducing-ai-powered-assistants-characters-and-creative-tools/. Acesso em: 02 de novembro de 2023.

MEYER, A. N.; FRITZ, T.; MURPHY, G. C.; ZIMMERMANN, T. Software developers' perceptions of productivity. In: *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*. [S.l.: s.n.], 2014. p. 19–29.

MICHAELIS. *Dicionário Brasileiro da Língua Portuguesa*. 2023. Disponível em: https://michaelis.uol.com.br/busca?r=0&f=0&t=0&palavra=performance. Acesso em: 21 de outubro de 2023.

MYSLIWIEC, K. *Nestjs Documentation*. 2023. Disponível em: https://docs.nestjs.com/. Acesso em: 23 de outubro de 2023.

OPENAI. *ChatGPT Guide*. 2023. Disponível em: https://platform.openai.com/docs/guides/gpt. Acesso em: 23 de outubro de 2023.

PENG, S.; KALLIAMVAKOU, E.; CIHON, P.; DEMIRER, M. The impact of ai on developer productivity: Evidence from github copilot. *arXiv preprint arXiv:2302.06590*, 2023.

REVELO. *Relatório de salários*. 2020. Disponível em: https://mktcme.s3-sa-east-1.amazonaws.com/reports/relatorio-salario-2020.pdf?utm_source=crm&utm_medium=email. Acesso em: 10 de outubro de 2023.

RODRIGUEZ, M. *Research: Quantifying GitHub Copilots impact on code quality*. 2023. Disponível em: https://github.blog/2023-10-10-research-quantifying-github-copilots-impact-on-code-quality/. Acesso em: 02 de novembro de 2023.

RUSSELL, S. J.; NORVIG, P. *Artificial intelligence a modern approach*. [S.l.]: London, 2010.

SARATH. *Which color theme do you prefer in your code editor? Survey, Retrieved 2020- 04-28*. 2016. Disponível em: https://hashnode.com/post/which-color-theme-do-you-prefer-in-your-codeeditor-ciq9e3wbn1avb0053p48nozw0. Acesso em: 24 de outubro de 2023.

SEJNOWSKI, T. J. Large language models and the reverse turing test. *Neural computation*, MIT Press, v. 35, n. 3, p. 309–342, 2023.

VAITHILINGAM, P.; ZHANG, T.; GLASSMAN, E. L. Expectation vs. experience: Evaluating the usability of code generation tools powered by large language models. In: *Chi conference on human factors in computing systems extended abstracts*. [S.l.: s.n.], 2022. p. 1–7.

VELAZCO, C. Meet windows copilot, the ai coming to help you understand your pc. *The Washington Post*, The Washington Post, p. NA–NA, 2023.

VERCEL. *Documentação Vercel AI SDK*. 2023. Disponível em: https://sdk.vercel.ai/docs. Acesso em: 22 de outubro de 2023.

VERCEL. *NextJS Documentation*. 2023. Disponível em: https://nextjs.org/docs. Acesso em: 24 de outubro de 2023.