ETEC ABDIAS DO NASCIMENTO

CURSO TÉCNICO EM DESENVOLVIMENTO DE SISTEMAS

BRUNO FIGUEREDO SALES SANTOS
GABRIEL DA ROSA SOUZA
GABRIEL MORAES BEZERRA
GUILHERME ASSUNÇÃO RESENDE
MESK KESLEY PIRES FONSECA
MURILO GONZALEZ BEZ CHLEBA
RONALDO JUSTINO DOS SANTOS JUNIOR

EasyPlan

SÃO PAULO – SP
DEZEMBRO / 2024

BRUNO FIGUEREDO SALES SANTOS
GABRIEL DA ROSA SOUZA
GABRIEL MORAES BEZERRA
GUILHERME ASSUNÇÃO RESENDE
MESK KESLEY PIRES FONSECA
MURILO GONZALEZ BEZ CHLEBA
RONALDO JUSTINO DOS SANTOS JUNIOR

EasyPlan

Trabalho de Conclusão de Curso apresentado à Etec Abdias do Nascimento, do Centro Estadual de Educação Tecnológica Paula Souza, como requisito parcial para a obtenção da habilitação profissional de Técnico de Nível Médio em Desenvolvimento de Sistemas sob a orientação do(s) Professor(es) Stephany Martins Conceição e José Fernando Lino Santiago.

SÃO PAULO – SP
DEZEMBRO / 2024

BRUNO FIGUEREDO SALES SANTOS
GABRIEL DA ROSA SOUZA
GABRIEL MORAES BEZERRA
GUILHERME ASSUNÇÃO RESENDE
MESK KESLEY PIRES FONSECA
MURILO GONZALEZ BEZ CHLEBA
RONALDO JUSTINO DOS SANTOS JUNIOR

EASYPLAN

Aprovada em : 05 / 12 / 2024

Conceito: B

Banca de Validação:

Professor Stephany Martins Conceição
Etec Abdias do Nascimento
Orientador

Professor .....
Etec Abdias do Nascimento

Professor Eduardo Lopes
Etec Abdias do Nascimento

SÃO PAULO – SP
2024

AGRADECIMENTOS

*Agradecemos, primeiramente, à professora **Stephany Martins**, cuja orientação e dedicação foram fundamentais para o desenvolvimento e realização do projeto **EasyPlan**. Sua paciência, comprometimento e incentivo nos motivaram a enfrentar os desafios com determinação e a buscar a excelência em todas as etapas do trabalho.*

*À nossa equipe, manifestamos nossa gratidão pelo empenho, pela colaboração e pelo espírito de união que tornaram possível a concretização deste projeto. A troca de ideias, o esforço coletivo e a determinação de todos foram indispensáveis para o sucesso do **EasyPlan**.*

Por fim, agradecemos a todas as pessoas que, de alguma forma, contribuíram direta ou indiretamente para o desenvolvimento deste projeto, seja pelo suporte técnico, pelas críticas construtivas ou pelo encorajamento ao longo do processo. Este trabalho é resultado de um esforço conjunto, e somos gratos por cada contribuição que tornou essa realização possível.

*Agradecemos, primeiramente, à professora **Estefani**, cuja orientação e dedicação foram fundamentais para o desenvolvimento e realização do projeto **EasyPlan**. Sua paciência, comprometimento e incentivo nos motivaram a enfrentar os desafios com determinação e a buscar a excelência em todas as etapas do trabalho.*

*À nossa equipe, manifestamos nossa gratidão pelo empenho, pela colaboração e pelo espírito de união que tornaram possível a concretização deste projeto. A troca de ideias, o esforço coletivo e a determinação de todos foram indispensáveis para o sucesso do **EasyPlan**.*

RESUMO

O Trabalho de Conclusão de Curso (TCC) intitulado "EasyPlan" propõe a criação de um aplicativo destinado a melhorar a organização pessoal e profissional dos usuários, com foco na dificuldade de equilibrar afazeres diários e lazer, especialmente devido ao uso excessivo de celulares e suas constantes distrações. A pesquisa revelou que muitos, especialmente entre 15 e 22 anos, enfrentam problemas de produtividade e saúde mental, como estresse, ansiedade e desmotivação, devido à falta de organização. O aplicativo visa solucionar esses desafios ao ajudar os usuários a gerenciar suas tarefas de maneira mais eficaz, promovendo uma rotina equilibrada. O projeto busca melhorar a gestão do tempo, aumentar a produtividade e promover hábitos saudáveis, incorporando funcionalidades como lembretes, planejamento de metas e acompanhamento de progresso. Além disso, o EasyPlan oferecerá dicas para o uso consciente da tecnologia, com a intenção de reduzir as distrações digitais e melhorar o bem-estar emocional dos usuários. A metodologia de desenvolvimento envolveu a coleta de dados por meio de questionários, com foco na faixa etária mencionada, a fim de identificar as necessidades e comportamentos mais comuns entre os usuários. A justificativa do projeto se fundamenta na importância da organização para o sucesso pessoal e saúde mental, buscando criar uma ferramenta capaz de promover uma vida mais equilibrada, produtiva e saudável, ao mesmo tempo em que fomenta o interesse por atividades organizacionais.

Palavras-Chave: Organização, Produtividade, Gestão do Tempo.

ABSTRACT

The Course Completion Work (TCC) entitled "EasyPlan" proposes the creation of an application designed to improve users' personal and professional organization, focusing on the difficulty of balancing daily tasks and leisure, especially due to the excessive use of cell phones and their constant distractions. The research revealed that many, especially between 15 and 22 years old, face productivity and mental health problems, such as stress, anxiety and lack of motivation, due to a lack of organization. The app aims to solve these challenges by helping users manage their tasks more effectively, promoting a balanced routine. The project seeks to improve time management, increase productivity and promote healthy habits, incorporating features such as reminders, goal planning and progress tracking. In addition, EasyPlan will offer tips for the conscious use of technology, with the intention of reducing digital distractions and improving users' emotional well-being. The development methodology involved collecting data through questionnaires, focusing on the aforementioned age group, in order to identify the most common needs and behaviors among users. The justification for the project is based on the importance of organization for personal success and mental health, seeking to create a tool capable of promoting a more balanced, productive and healthy life, while at the same time fostering interest in organizational activities.

Keywords: Organization. Productivity. Time management.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS / SIGLAS

CLT – Consolidação das Leia Trabalhistas

DIEESE – Departamento Intersindical de Estatística e Estudos Socioeconômicos

RAIS – Relação Anual de Informações Sociais

VSM – *Value Stream Mapping (*Mapeamento do Fluxo de Valor)

SQL – Structured Query Language (Linguagem de Consulta Estruturada)

SPA – Single Page Application (Aplicação de Página Única)

CRUD – Create, Read, Update, Delete (Criar, Ler, Atualizar, Excluir)

EAP – Estrutura Analítica do Projeto

DER – Diagrama de Entidade e Relacionamento

SUMÁRIO

1. INTRODUÇÃO

Nosso Trabalho de Conclusão e Curso (TCC), cujo foi nomeado de EasyPlan, consiste na criação de um aplicativo onde o intuito dele é melhorar a organiza-ção pessoal e profissional das pessoas. Como resultado da nossa pesquisa, se observou que uma grande parte das pessoas tem uma certa dificuldade em estabelecer um equilíbrio tanto nos seus afazeres gerais da vida, quanto no seu tempo livre. Por conta que há vários estímulos, causados pelo uso excessivo do celular, que acabam tirando a atenção necessária que aquela pessoa precisava ter para fazer determinada tarefa.

1.1. PROBLEMATIZAÇÃO

A falta de organização tem se mostrado um problema crescente, especialmente entre os jovens, e pode afetar negativamente tanto a produtividade quanto a saúde mental. O uso excessivo do celular e as distrações involuntárias, como imprevistos ou desinteresse, são algumas das causas desse comportamento. Muitas vezes, o desinteresse surge pela ausência de um estímulo eficaz para manter a pessoa focada nas atividades, ou pela falta de uma rotina organizada que ajude a estruturar o dia a dia

Antes de propor soluções, é importante compreender como a falta de organização pode impactar a saúde mental. A psicóloga Katia Cristina Brito (2023) destaca: “Uma vida bem estruturada promove saúde mental, da mesma forma que uma vida desorganizada pode causar instabilidade mental”[1]. Esse desequilíbrio tem se refletido em diversas consequências negativas, especialmente entre os jovens, de forma trágica em algumas situações.

De acordo com a pesquisa da psicanalista Nicole Bonatti (2021), a falta de organização no ambiente de trabalho pode gerar uma série de problemas. A ausência de organização leva à repetição de tarefas, o que compromete a produtividade. A falta de planejamento dificulta a gestão do tempo e a execução das tarefas dentro dos prazos estabelecidos. Sem uma organização clara, torna-se difícil atingir objetivos, o que pode gerar frustração e desmotivação. A desorganização também pode resultar em sobrecarga, dificultando o foco e a execução eficiente das tarefas. Além disso,

sem uma estrutura definida, é difícil perceber o progresso, o que leva à perda de motivação e ao ciclo vicioso da desorganização [2].

Portanto, entender como a desorganização impacta diretamente a motivação, o foco e a saúde mental é fundamental para implementar soluções eficazes que promovam uma rotina mais equilibrada e produtiva.

1.2. HIPÓTESE

A falta de organização e de uma rotina estruturada entre os jovens contribui significativamente para a saúde mental, levando a um ciclo vicioso de desmotivação e falta de foco. Esse cenário se agrava pelo uso excessivo de dispositivos móveis, que provoca distrações e reduz a capacidade de concentração, resultando em um aumento do estresse, da ansiedade e da insatisfação pessoal. Assim, a implementação de práticas de organização pessoal, aliadas a técnicas de gerenciamento do tempo e uso consciente da tecnologia, pode melhorar a saúde mental e a produtividade, criando um ambiente propício para o desenvolvimento de habilidades e o alcance de metas.

No futuro, esta aplicação, ao promover práticas de organização pessoal e técnicas de gerenciamento do tempo, pode alcançar um impacto significativo na melhoria da saúde mental dos jovens, ajudando a reduzir níveis de estresse e ansiedade. Ao incentivar o uso consciente da tecnologia, pode criar hábitos mais saudáveis e produtivos, estimulando maior foco, motivação e satisfação pessoal. Com o tempo, isso poderá resultar em um ambiente mais equilibrado, onde os jovens podem desenvolver suas habilidades, alcançar metas e construir uma base sólida para o seu bem-estar e sucesso a longo prazo.

1.3. OBJETIVOS

1.3.1. Geral

Desenvolver o aplicativo EasyPlan, que tem como propósito transformar a organização pessoal e profissional dos usuários. O aplicativo visa ajudar aqueles que enfrentam dificuldades em equilibrar suas responsabilidades diárias e seu tempo livre, proporcionando uma abordagem intuitiva e envolvente. Reconhecendo que uma vida organizada vai além do trabalho e dos estudos, o EasyPlan permitirá que os usuários estabeleçam rotinas estruturadas e gerenciem sua vida de forma eficaz, mesmo em meio aos desafios impostos por distrações digitais.

1.3.2. Específico

- Facilitar a organização de tarefas acadêmicas, profissionais e pessoais, oferecendo uma interface amigável e acessível.
- Proporcionar recursos de planejamento que ajudem os usuários a definir metas claras e acompanhar seu progresso de forma visual.
- Aumentar a produtividade dos usuários por meio de lembretes personalizáveis e técnicas de gerenciamento de tempo.
- Incentivar a adoção de hábitos saudáveis, por meio de desafios interativos e recompensas que promovam a organização.
- Estimular o interesse pelo aprendizado e pela auto-organização, oferecendo dicas e conteúdos educacionais dentro do aplicativo.
- Criar uma comunidade virtual onde os usuários possam compartilhar experiências, dicas de organização e motivação mútua.
- Implementar ferramentas analíticas que permitam aos usuários monitorar seu desempenho e identificar áreas de melhoria na gestão do tempo.
- Reduzir os impactos das distrações digitais, oferecendo estratégias para minimizar a interferência do uso excessivo de celulares na realização de tarefas.

1.4. JUSTIFICATIVA

A proposta deste projeto surge da constatação de que a falta de organização é um problema crescente, especialmente entre os jovens, e impacta tanto a produtividade quanto a saúde mental. Como mencionado na problemática, distrações como o uso excessivo do celular, imprevistos e desinteresse são consequências diretas da falta de uma rotina estruturada, o que leva à baixa produtividade e ao aumento do estresse e da ansiedade. Ao observar essas dificuldades, percebemos a necessidade de fornecer suporte para o aprimoramento das habilidades de gestão do tempo e organização.

Ao oferecer ferramentas e orientações para uma gestão mais eficiente das tarefas diárias e a criação de uma rotina organizada, buscamos não apenas melhorar a produtividade e o foco, mas também promover o bem-estar emocional. Acreditamos que a redução do estresse e da ansiedade, causados pela falta de organização, contribui para uma vida mais equilibrada e satisfatória. Assim, este projeto tem como objetivo ajudar os indivíduos a desenvolverem uma estrutura que promova tanto o sucesso profissional quanto a saúde mental, minimizando os impactos negativos da desorganização.

2. METODOLOGIA

2.1 DETALHAMENTO DA PESQUISA

A metodologia adotada neste trabalho visou fornecer um detalhamento minucioso da pesquisa realizada, de modo que permita sua reprodução por terceiros. Inicialmente, foram conduzidas pesquisas na internet para a coleta de dados relevantes. Esses dados foram organizados e analisados, o que resultou na elaboração de um questionário que foi feito no google forms. Posteriormente, os dados adquiridos através deste questionário foram novamente analisados.

2.1.1 CAMINHOS PERCORRIDOS PARA ALCANÇAR OS OBJETIVOS PROPOSTOSE

Através das pesquisas preliminares, identificamos um problema significativo relacionado ao foco e ao desinteresse dos jovens, um fenômeno que se intensificou após a pandemia. Essa constatação evidenciou a necessidade de desenvolver um aplicativo que auxilie os jovens a superarem esses desafios.

2.1.2 ESPECIFICAÇÕES TÉCNICAS, MATERIAIS E EQUIPAMENTOS EMPREGADOS

Para a realização da pesquisa, utilizamos computadores e smartphones com acesso à internet. Adicionalmente, foram consultados diversos artigos jornalísticos para fundamentar o estudo.

2.1.3 SELEÇÃO DA AMOSTRA E PERCENTUAL EM RELAÇÃO À POPULAÇÃO ESTUDADA

A amostra foi selecionada em uma instituição de ensino cuja faixa etária predominante era de 15 a 18 anos. Consequentemente, a maior parte da nossa população estudada está nessa faixa etária. Um segmento menor da população estudada compreende indivíduos entre 18 e 22 anos, com uma parcela ainda menor composta por pessoas acima de 22 anos.

2.1.4 INSTRUMENTOS DE PESQUISA UTILIZADOS

Os instrumentos de pesquisa incluíram questionários, enviados para os grupos escolares, e consultas a sites de psicologia e notícias. Esses instrumentos permitiram uma coleta de dados abrangente e diversificada.

2.1.5 TRATAMENTO E ANÁLISE DOS DADOS

Após a coleta dos dados, realizamos reuniões para avaliar a pertinência de cada um deles em relação aos nossos objetivos. A análise criteriosa dos dados nos permitiu delinear de forma mais precisa as funcionalidades do aplicativo que pretendemos desenvolver.

2.1.6 PERIODO DA PESQUISA

O período da pesquisa foi de 30 de março a 22 de novembro de 2024. Durante esse tempo, os dados foram coletados e analisados, permitindo uma compreensão profunda sobre os desafios enfrentados pelos jovens em relação à organização pessoal e ao uso excessivo de tecnologia. Esse período permitiu o levantamento de informações relevantes que orientaram o desenvolvimento das funcionalidades do aplicativo EasyPlan.

2.1.7 PLATAFORNA ULTILIZADA

A plataforma utilizada para a coleta de dados foi o Google Forms. Por meio dessa ferramenta, foi possível criar um questionário online de fácil acesso para os participantes, garantindo uma coleta de respostas eficiente e organizada. O Google Forms permitiu a aplicação do questionário em diferentes grupos e facilitou o processo de análise das respostas, que foram automaticamente organizadas na plataforma.

2.1.8 QUANTIDADE DE RESPOSTAS E IMPACTO NO PROJETO

Foram obtidas 30 respostas ao questionário enviado. As respostas foram analisadas de forma criteriosa para identificar padrões e insights que pudessem contribuir para a criação do aplicativo. A amostra, composta principalmente por jovens de 15 a 18 anos, forneceu informações valiosas sobre os hábitos e dificuldades em relação à organização e ao gerenciamento do tempo. As perguntas do questionário abordaram diversos aspectos relacionados à organização pessoal, ao uso de tecnologia e ao impacto desses fatores na saúde mental e produtividade dos participantes. Entre os temas abordados, destacaram-se: dificuldades em manter o foco, gestão de tempo, uso de dispositivos móveis e como a desorganização impacta a vida cotidiana. Também foram investigadas as estratégias que os participantes utilizam para tentar se organizar, bem como suas preferências por ferramentas digitais que poderiam auxiliá-los nesse processo.

3. DESENVOLVIMENTO

3.1 REVISÃO LITERÁRIA

A falta de organização pode afetar profundamente a saúde mental. A disciplina e uma gestão clara do tempo são essenciais para a realização eficaz das atividades diárias. Quando o tempo não é gerido adequadamente, pode resultar em ansiedade e estresse, como discutido por vários estudiosos. De acordo com The Time Paradox: The New Psychology de Philip Zimbardo e John Boyd (2008) [3], a gestão do tempo e a percepção do tempo afetam diretamente o bem-estar psicológico e a saúde mental.

A falta de organização pessoal é um desafio comum que afeta a gestão do tempo, a disciplina e a rotina dos indivíduos. A literatura aponta que, embora as pessoas tenham motivação inicial, a ausência de organização leva à frustração, indisciplina e desistência de tarefas (Macan, 1994) [4]. A gestão de tempo é crucial para a produtividade e o sucesso, tanto pessoal quanto profissional, mas barreiras como procrastinação e falta de motivação dificultam sua implementação (Claessens et al., 2007; Steel, 2007) [5][6].

O foco, essencial para a execução eficaz das tarefas, é prejudicado por fatores como multitarefa e estresse (Posner & Petersen, 1990) [7]. A persistência é fundamental para a superação de obstáculos e o alcance de metas de longo prazo (Duckworth et al., 2007) [8]. Manter a motivação e o foco ao longo do tempo é um desafio constante. Uma solução prática que forneça incentivo constante pode ser determinante para o sucesso e para o desenvolvimento de hábitos saudáveis (Ryan & Deci, 2000) [9].

A análise das respostas de usuários em pesquisas revelou que muitos enfrentam dificuldades em manter a consistência em suas rotinas devido à falta de organização e motivação contínua. Com base nessas necessidades, o EasyPlan foi desenvolvido para integrar os pilares da gestão de tempo, foco e persistência, oferecendo uma plataforma que auxilia os usuários a manterem-se disciplinados e motivados ao longo do tempo. A ferramenta visa fornecer as estratégias necessárias para superar os desafios de organização e manter o progresso contínuo, por meio de lembretes, metas claras e recursos de monitoramento de desempenho.

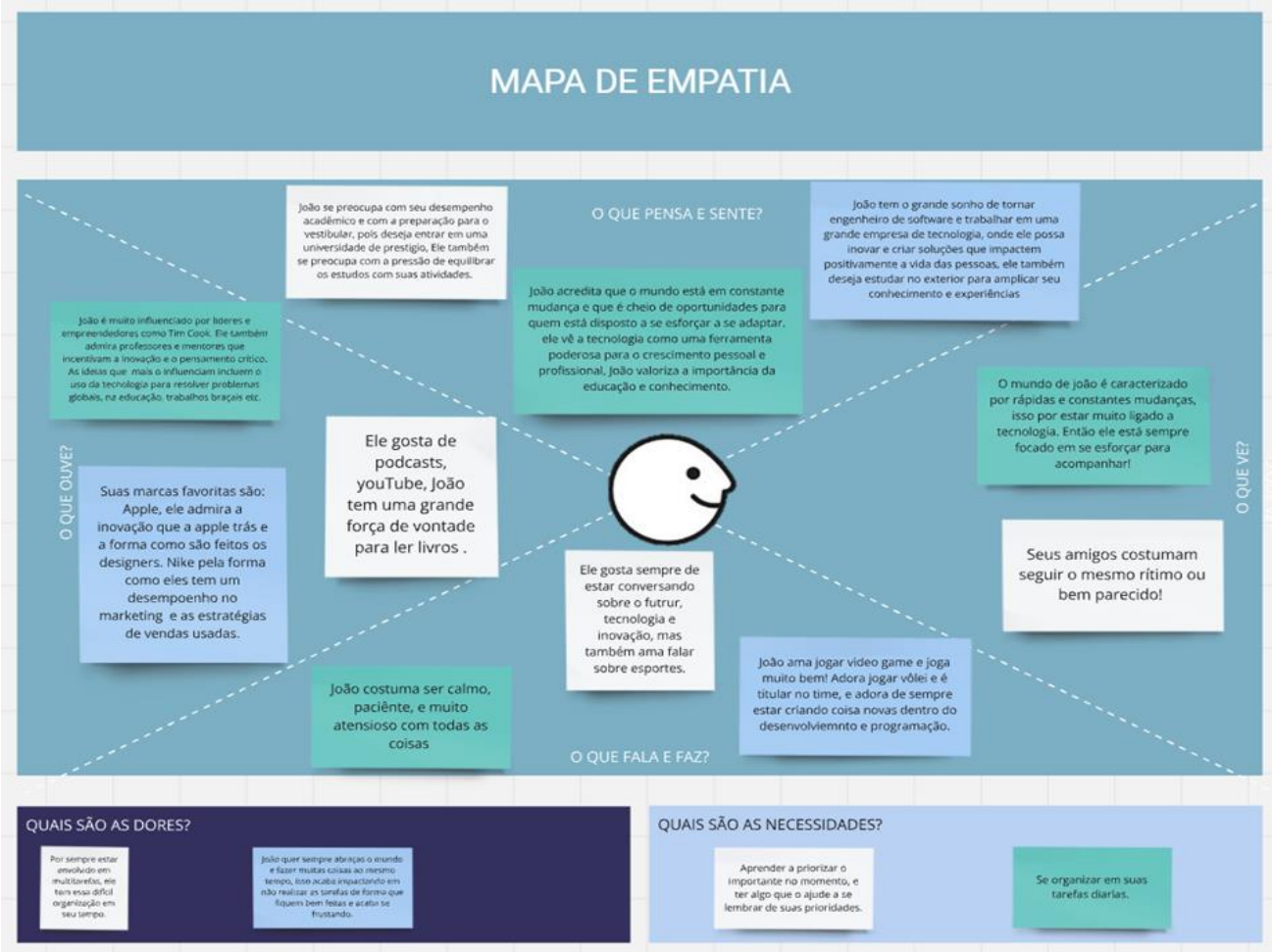
3.2 ANÁLISE DA PESQUISA DE CAMPO

3.2.1 Mapa de empatia

Um mapa de empatia é uma ferramenta visual usada para entender de maneira mais profunda as necessidades, desejos, dores e expectativas de um público-alvo. Ele ajuda a compreender o comportamento e as motivações dos usuários ou clientes, facilitando o desenvolvimento de soluções mais alinhadas com as necessidades reais das pessoas.

O mapa de empatia é uma ferramenta que ajuda a entender as emoções, necessidades e comportamentos do público-alvo. No caso do EasyPlan, ele foi crucial para identificar os principais desafios dos usuários, como distrações digitais e falta de organização. Com base nisso, o mapa de empatia ajudou a definir funcionalidades do aplicativo, como planejamento de tarefas, lembretes e estratégias para reduzir distrações, garantindo que o app atendesse às reais necessidades dos usuários e promovesse uma rotina mais equilibrada.

Figura 1 – Mapa de empatia



3.2.2 Personas

Persona é uma representação semi-ficcional do usuário ideal de um produto, serviço ou solução, baseada em dados reais de pesquisa, comportamento e características do público-alvo. A criação de uma persona envolve reunir informações demográficas, comportamentais, motivações, desafios e objetivos, permitindo que a equipe de desenvolvimento de um projeto compreenda melhor quem é o usuário, suas necessidades e como ele interage com o que está sendo oferecido.

A criação de personas foi essencial no desenvolvimento do EasyPlan, permitindo identificar os principais desafios dos usuários, como dificuldade de organização e distrações digitais. Isso ajudou a moldar o design e os recursos do app, como planejamento de tarefas, lembretes personalizados e estratégias contra distrações. Além disso, as personas influenciaram a inclusão de recursos interativos, como

desafios e recompensas, e conteúdos educativos para melhorar a gestão do tempo e a produtividade, tornando o EasyPlan uma solução eficaz para os problemas reais dos usuários, promovendo sua saúde mental e eficiência.

Figura 2 – Persona 1

Figura 3 – Persona 2

3.3 FERRAMENTAS DE DESENVOLVIMENTO

3.3.1 FERRAMENTAS DE INTEGRAÇÃO E BASE DE DADOS

- **MySQL** - O **MySQL** é um **sistema de gerenciamento de banco de dados relacional** de código aberto amplamente utilizado no desenvolvimento de aplicativos e sites. Ele armazena dados de maneira estruturada em tabelas, permitindo consultas complexas por meio da linguagem SQL (Structured Query Language). O MySQL é conhecido por sua robustez, escalabilidade e confiabilidade, sendo uma das escolhas mais populares para gerenciar grandes volumes de dados. Ele é usado por diversas empresas e plataformas, tanto em ambientes de desenvolvimento quanto em produção.

- **React Router** - React Router é uma biblioteca que fornece uma solução para navegação em aplicações React. Ele permite a criação de rotas dinâmicas, onde a interface do usuário é atualizada conforme a URL muda, sem recarregar a página. O React Router é usado para criar uma navegação fluida em aplicações de uma única página (SPA), permitindo que diferentes componentes sejam renderizados com base na URL, sem necessidade de recarregar a página.

- **Express** - Express é um framework minimalista para Node.js que facilita a criação de servidores e APIs. Ele fornece funcionalidades como roteamento, middleware e manipulação de requisições HTTP de forma simples e eficiente. O Express é amplamente usado no desenvolvimento de backends, pois ajuda a gerenciar rotas, processar requisições e responder a elas, além de permitir a integração com outras bibliotecas e serviços.

3.3.2 FERRAMENTAS DE FRONT-END

- **React -** React é uma biblioteca JavaScript desenvolvida pelo Facebook para a criação de interfaces de usuário. Ele permite a criação de componentes reutilizáveis que gerenciam seu próprio estado e reagem a mudanças de dados. O React é usado para construir interfaces dinâmicas em aplicações

web, onde as partes da interface podem ser atualizadas de forma eficiente sem recarregar a página inteira. Ele é especialmente útil em SPAs (aplicações de uma única página).

- **Git -** Git é um sistema de controle de versão distribuído que permite rastrear as alterações no código-fonte de um projeto ao longo do tempo. Ele é amplamente utilizado no desenvolvimento colaborativo. O Git é utilizado para gerenciar o histórico de código, possibilitando que desenvolvedores trabalhem de forma eficiente e segura em projetos, podendo reverter alterações, criar versões e facilitar o trabalho em equipe através de branches e fusões (merges).

3.3.3 FERRAMENTAS DE SERVIDOR E GERENCIAMENTO DE PACOTES

- **Node.js -** Node.js é um ambiente de execução JavaScript que permite rodar código JavaScript no lado do servidor. Ele é baseado no motor V8 do Google Chrome e usa um modelo de I/O assíncrono e orientado a eventos. O Node.js é utilizado para criar aplicações server-side escaláveis e rápidas, como servidores web, APIs e sistemas de tempo real, aproveitando sua arquitetura não bloqueante e eficiente para lidar com grandes volumes de tráfego simultâneo.

- **Trello -** O **Trello** é uma plataforma de **gestão de projetos** baseada em quadros e cartões. Ele usa uma estrutura visual intuitiva para organizar tarefas e acompanhar o progresso de um projeto. Os usuários podem criar quadros para diferentes projetos, adicionar listas (como "A Fazer", "Em Progresso", "Concluído") e cartões para tarefas específicas. É possível adicionar prazos, anexar documentos, deixar comentários e colaborar com outras pessoas. O Trello é amplamente utilizado por equipes para gerenciar fluxos de trabalho de forma simples e eficaz, seja para projetos pessoais ou profissionais.

- **Axios -** Axios é uma biblioteca JavaScript baseada em promessas que facilita a realização de requisições HTTP. É amplamente usada para interagir com APIs e servidores, tanto no lado do cliente (navegador) quanto no servidor

(Node.js). O Axios simplifica o envio de dados e a manipulação das respostas de servidores ou APIs. Ela oferece recursos como o controle de erros, interceptadores de requisição/resposta e a possibilidade de cancelar requisições.

3.3.4 Ferramentas de design:

- **Miro - Miro** é uma plataforma online colaborativa de **quadros brancos digitais** que permite equipes e indivíduos criar, planejar e compartilhar ideias, fluxos de trabalho, mapas mentais e protótipos. Ele oferece uma interface intuitiva onde é possível inserir post-its, textos, imagens, diagramas e outros tipos de conteúdos, facilitando o brainstorming, a organização de projetos e o planejamento estratégico. Ideal para reuniões virtuais e equipes remotas, o Miro facilita a colaboração em tempo real, permitindo que vários usuários interajam e contribuam simultaneamente no mesmo espaço de trabalho.

- **Figma** - **Figma** é uma plataforma de **design de interfaces** e **colaboração em tempo real** usada por designers de produtos, desenvolvedores e equipes para criar protótipos e interfaces de usuário. Ele permite que múltiplos usuários trabalhem simultaneamente em um mesmo arquivo, facilitando a colaboração em tempo real entre membros de equipes distribuídas. O Figma oferece ferramentas poderosas de design vetorial, prototipagem e animação, além de permitir que os designs sejam facilmente compartilhados com stakeholders ou desenvolvedores para revisão e implementação. Sua natureza baseada em nuvem também facilita o acesso e a edição dos projetos em qualquer lugar.

3.4 EAP

3.4.1 Iniciação do Projeto

Definição dos objetivos:

O projeto *EasyPlan* tem como objetivo principal o desenvolvimento de um aplicativo que promova a organização pessoal dos usuários. O aplicativo será composto por funcionalidades como cadastro, login, Pomodoro, gamificação, agenda, checklist, estatísticas e gestão de informações pessoais, visando atender às necessidades específicas dos usuários.

Análise da viabilidade:

Foi realizada uma análise detalhada da viabilidade do projeto, considerando fatores como recursos tecnológicos e humanos disponíveis, além do tempo necessário para a execução de todas as funcionalidades.

Revisão bibliográfica:

Realizou-se uma pesquisa abrangente sobre metodologias de organização pessoal, técnicas de produtividade, como o método Pomodoro, e estratégias de gamificação. O objetivo foi embasar teoricamente o desenvolvimento do projeto e adotar as melhores práticas disponíveis.

Aprovação do projeto:

Após a análise inicial, o escopo do projeto foi validado e aprovado para que as etapas de planejamento e desenvolvimento pudessem ser iniciadas.

3.4.2 Planejamento:

Levantamento de requisitos

Os requisitos necessários para o desenvolvimento do aplicativo foram definidos:

Front-End: Telas de cadastro, login, Pomodoro, gamificação, agenda, checklist, estatísticas e conta do usuário.

Back-End: Lógica para sistemas de cadastro, login, Pomodoro, gamificação (incluindo pontos, upgrades e punições), leitura de estatísticas e alterações na conta.

Banco de Dados: Estrutura para armazenar informações relacionadas a cadastro, login, Pomodoro e feedback dos usuários.
Elementos visuais adicionais: Sprites para os upgrades, animações de fundo e uma tela específica para o registro de compromissos.

Cronograma do projeto: Foi elaborado um cronograma contendo todas as etapas do desenvolvimento, desde o design e codificação até os testes e validação, com prazos e responsáveis definidos para cada fase.

Definição de ferramentas e tecnologias: Selecionaram-se ferramentas e tecnologias apropriadas para o desenvolvimento, contemplando linguagens e frameworks para front-end, back-end e banco de dados, além de ferramentas de design gráfico e controle de versionamento.

3.4.3 Desenvolvimento

Design do aplicativo:

Foram criados os designs das telas e dos elementos visuais do aplicativo, priorizando uma interface intuitiva e agradável ao usuário.

Wireframe e protótipos: Foram desenvolvidos protótipos simplificados das telas para testar a navegação e a usabilidade do sistema.

Desenvolvimento Front-End: Implementou-se a interface do usuário, contemplando telas como cadastro, login, Pomodoro, checklist e outras funcionalidades.

Desenvolvimento Back-End: Foi desenvolvida a lógica dos sistemas de cadastro, gamificação (pontos, upgrades e punições), leitura de estatísticas e gestão de conta, conectando as funcionalidades ao banco de dados.

Integração de APIs e banco de dados: O banco de dados foi integrado com as funcionalidades do aplicativo, garantindo o armazenamento seguro de informações sensíveis e a comunicação eficiente entre os sistemas.

3.4.4 Testes e Validação

Testes de funcionalidade: Realizaram-se testes para verificar se todas as funcionalidades do aplicativo estavam funcionando de acordo com o planejado.

Testes de usabilidade: Foram realizados testes com usuários para avaliar a facilidade de uso e a navegação nas telas do aplicativo.

Testes de segurança: Foram conduzidos testes específicos para garantir a proteção de informações armazenadas no banco de dados e evitar possíveis vulnerabilidades

.

3.4.5 Encerramento

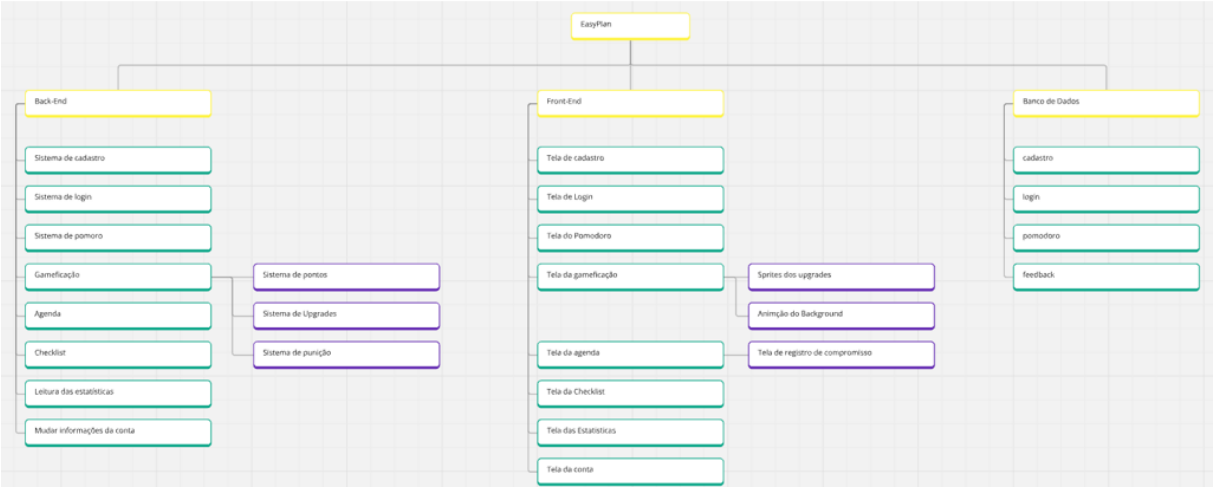
Documentação final do projeto Foi elaborada a documentação completa do projeto, contendo o código-fonte, manuais de uso e relatórios técnicos.

Apresentação do TCC O projeto *EasyPlan* foi apresentado formalmente, destacando as etapas do desenvolvimento e os resultados obtidos com a implementação do aplicativo.

Avaliação e feedback: Foram coletados feedbacks de avaliadores para identificar possíveis melhorias e ajustes futuros no projeto.

Entrega do relatório final: O relatório final foi produzido, consolidando todas as etapas do desenvolvimento, os resultados alcançados e as perspectivas futuras para o aplicativo.

FIGURA 4 - EAP



6. ANÁLISE DE REQUISITOS DO SOFTWARE

6.1 REQUISITOS FUNCIONAIS

RF. 001: Tela de Login/Cadastro

Descrição: Nela será onde o usuário irá colocar seus dados básicos para se ter uma conta, como o nome, e-mail e senha. Também terá uma opção em que o usuário poderá usar o seu login do google ou do facebook.

Entrada: Nome, e-mail e senha.

Processo: Após o usuário colocar os seus dados o sistema irá verificar se o e-mail e senha dele é valido e guardar os dados no banco de dados.

Saída: Se os dados de cadastro estiverem todos certos aparecer escrito para o usuário “Cadastro concluído com sucesso” e o usuário será redirecionado para a tela principal da aplicação. Caso o e-mail dele não seja valido aparecera um aviso na tela dizendo “E-mail invalido”, ou se a senha for invalida, como não ter determinado número de caracteres, irá aparecer para o usuário criar uma senha mais ‘forte’.

Prioridade:

[x] Essencial [ ] Importante [ ] Desejável

RF. 002: Tela Login

Descrição: Será uma forma resumida da tela de Cadastro, terá a área do nome/ e-mail e a área da senha.

Entrada: Nome, e-mail e senha.

Processo: O sistema receberá os dados que o usuário colocou, verificar no banco de dados o e-mail, nome e senha se são equivalentes aos que haviam sido salvos anteriormente e retornar o resultado para o usuário.

Saída: Caso os dados que o usuário colocou forem diferentes com o que o banco de dados salvou, aparecerá um aviso falando que um dos dados está errado, por exemplo, se e-mail estiver errado aparecerá um aviso na tela dizendo “E-mail invalido”. Caso o usuário esqueça a senha terá uma opção de nós enviarmos um e-mail de redefinição de senha.

Prioridade:

[x] Essencial [ ] Importante [ ] Desejável

RF. 003: Tela de Pomodoro

Descrição: A tela inicial da aplicação, onde após o usuário fizer seu login ele irá ser redirecionado direto para ela, a sua função base é servir como um timer onde o usuário poderá configurar o tempo que ele gastará para realizar determinada tarefa.

Entrada: O tempo que o usuário selecionou no timer.

Processo: O sistema irá analisar e armazenar, no banco de dados, o tempo que o usuário selecionou no timer para realizar suas tarefas, para que ele possa calcular o total de pontos que o usuário fez. Após concluir o seu “tempo” de estudo ele receberá pontos, caso ele não cumpra o tempo total, ele receberá só 80% dos pontos equivalentes ao tempo que ele cumpriu.

Saída: Após o sistema ter salvado e calculado os dados do timer, ele irá retornar o resultado dessa análise para o usuário tanto em forma de pontos, que estão associadas à parte gamificada, quanto em estatísticas na Tela de estatísticas da aplicação

Prioridade:

[x] Essencial [ ] Importante [ ] Desejável

RF. 004: Tela do Game

Descrição: O aplicativo tem como objetivo ajudar os usuários a manter o foco, desativando as distrações externas e utilizando um sistema visual de feedback para incentivar o cumprimento do tempo de foco estabelecido. O usuário define a quantidade de tempo que deseja se manter concentrado na tela de game. Ao iniciar o timer, todas as notificações de outros aplicativos são bloqueadas, criando um ambiente livre de distrações. Durante o período de foco, um objeto, como uma lua ou outro plano visual, começa a se mover ou a realizar uma ação simbólica (por exemplo, uma volta completa ao redor de um círculo) conforme o tempo passa. Quando o tempo de foco chega ao fim, o objeto atinge seu objetivo, indicando o término do tempo de concentração.

Entrada:

1. **Tempo desejado de foco**: O usuário define o tempo que deseja se concentrar na tarefa (por exemplo, 20 minutos).
2. **Início**: O usuário pressiona o botão "Iniciar" para começar o timer e bloquear notificações de outros aplicativos no celular.
3. **Seleção do objeto visual**: O usuário pode escolher entre diferentes símbolos ou planos visuais, como uma lua ou outro objeto, para acompanhar o progresso do tempo.

Processo:

1. O usuário define o tempo de foco na tela de game.
2. Ao pressionar "Iniciar", o aplicativo começa a contagem do tempo.
3. As notificações de outros aplicativos são automaticamente bloqueadas, impedindo distrações durante o período de foco.
4. Durante o tempo de concentração, o símbolo escolhido (exemplo: uma lua) começa a girar ou se mover ao redor de um círculo, representando o progresso do tempo.
5. A medida que o tempo passa, o símbolo se aproxima de seu objetivo final (por exemplo, a lua dá a volta completa ao redor do círculo).
6. O progresso é visível para o usuário, ajudando a manter a motivação e o foco.

Saida:

- Quando o tempo de foco chega ao fim, o símbolo (lua ou outro objeto visual) atinge seu objetivo final (como completar uma volta ao redor do círculo).
- O aplicativo então notifica o usuário de que o tempo de foco terminou.
- As notificações de outros aplicativos são novamente liberadas, permitindo que o usuário retome a interação com o celular.

Essa estrutura ajuda a promover a concentração, criando uma experiência visual e funcional que mantém os usuários motivados e sem distrações durante o período de foco estabelecido.

Prioridade:

[x] Essencial [ ] Importante [ ] Desejável

RF. 005: Tela de Estatísticas

Descrição: A tela em que terá todos os dados tanto de progresso quanto de retrocesso do usuário ao decorrer do uso do aplicativo.

Entrada: Os dados adquiridos pelo timer.

Processo: O sistema irá pegar os dados salvos pelo usuário no banco de dados para fazer os gráficos do progresso do usuário.

Saída: Após isso o sistema irá apresentar, como resposta para o usuário, os gráficos/estatísticas de progresso

Prioridade:

[x] Essencial [ ] Importante [ ] Desejável

RF. 006: Tela do Calendário

Descrição: A tela em que o usuário poderá marcar o que ele irá fazer em cada dia da semana em algum mês.

Entrada: Dados que o usuário for salvar no calendário

Processo: O sistema irá salvar, no banco de dados, todos as informações em que o usuário quiser salvar no calendário, como compromissos ou o que ele vai fazer em determinada semana.

Saída: No calendário irá aparecer as informações que o usuário colocou nos dias determinados por ele.

Prioridade:

[x] Essencial [ ] Importante [ ] Desejável

RF. 007: Tela de Checklist

Descrição: A tela em que o usuário poderá listar as coisas que ele pretende fazer da sua rotina.

Entrada: Dados que o usuário for salvar

Processo: O sistema irá salvar as informações que o usuário colocou na checklist no banco de dados.

Saída: Após ter salvo os dados, o sistema irá retornar para o usuário esses dados em uma espécie de lista para que o usuário possa marcar o que ele já fez ou não até aquele momento

Prioridade:

[x] Essencial [ ] Importante [ ] Desejável

RF. 008: Tela de Suporte

Descrição: A tela em que o usuário poderá se comunicar com os Administradores do Aplicativo.

Entrada: Informações que o usuário poderá compartilhar com os Admins, caso tenha algo de errado com o aplicativo

Processo: O sistema irá criar uma área onde o usuário poderá escrever as informações que ele deseja enviar para os Admins.

Saída: Após o sistema “criar” essa área, ele irá mandar as informações que o usuário escreveu e mandar diretamente para os Admins.

Prioridade:

[x] Essencial [ ] Importante [ ] Desejável

6.2 REQUISITOS NÃO FUNCIONAIS

NF001 - Usabilidade

A usabilidade do aplicativo será simples e bem intuitiva, para que o usuário se localize de forma fácil. Nele terá ícones pré-selecionados onde o usuário possivelmente entenderá o que cada um faz apenas vendo os uma única vez.

Prioridade:

[x] Essencial [ ] Importante [ ] Desejável

NF002 – Confiabilidade e Segurança

A confiabilidade e segurança do nosso aplicativo será feita com criptografia tendo como base o Firebase do Google para salvar os dados do usuário como a senha. Por exemplo, se o usuário quiser mudar a senha do login, ele recebera um e-mail pedindo uma confirmação onde se for o próprio usuário que está tentado mudar a senha.

Prioridade:

[x] Essencial [ ] Importante [ ] Desejável

NF003- Desempenho

Terá uma renderização rápida podendo carregar de forma ágil os ícones, as telas do aplicativo e outros recursos.

Prioridade:

[x] Essencial [ ] Importante [ ] Desejável

NF004 - Portabilidade

A portabilidade do aplicativo será feita para celulares, principalmente para Android.

Prioridade:

[x] Essencial [ ] Importante [ ] Desejável

NF005 - Hardware e Software

Ter um sistema operacional a partir do Android 1.5 (Cupcake).

Prioridade:

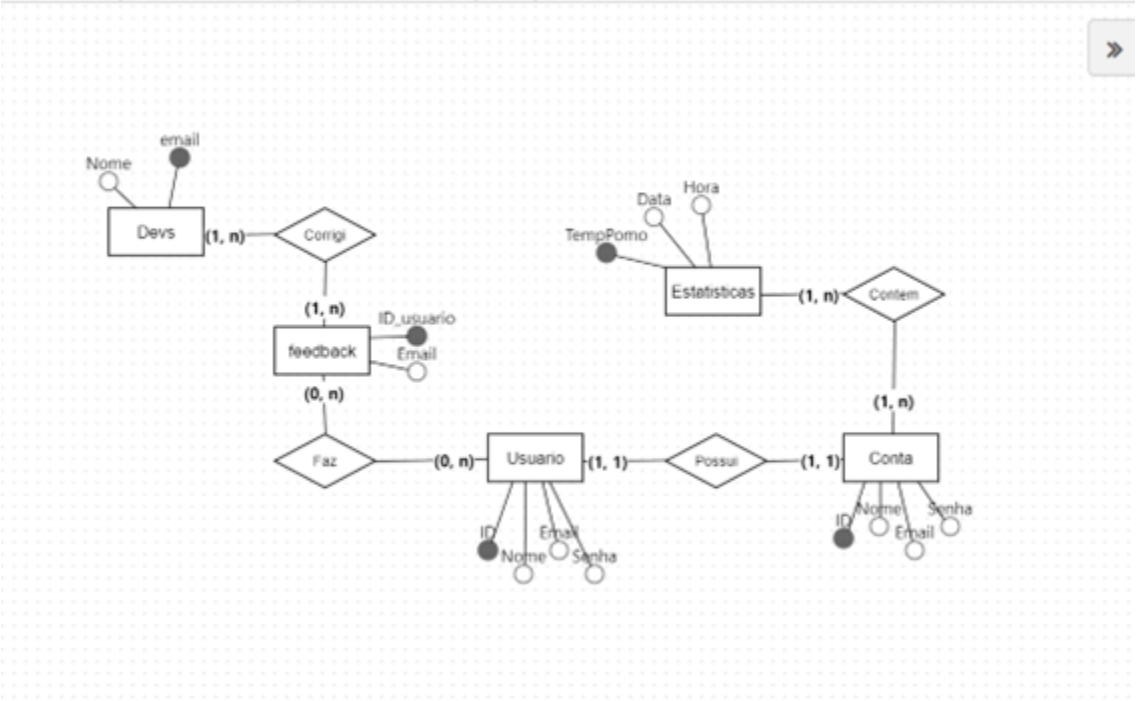
[x] Essencial [ ] Importante [ ] Desejável

7. MODELAGEM DO SISTEMA

7.1 BANCO DE DADOS

7.2 DIAGRAMA DE ENTIDADE E RELACIONAMENTO (DER):

Figura 5 - DER



8. BANCO DE DADOS: CÓDIGO:

Figura 6 – Banco de dados

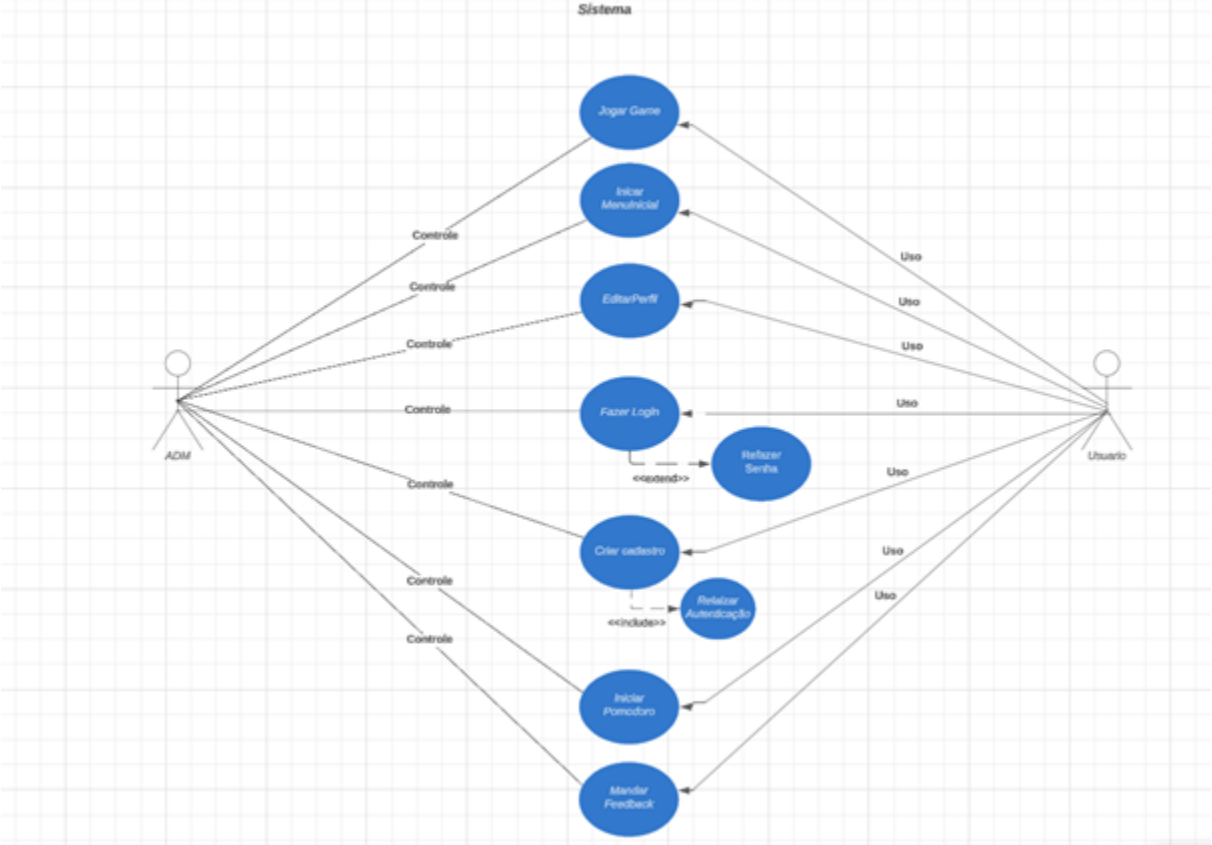
```sql
DROP DATABASE IF EXISTS easy_plan;
CREATE DATABASE IF NOT EXISTS easy_plan;
USE easy_plan;

CREATE TABLE tblogin (
    idu INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(200) NOT NULL,
    email VARCHAR(200) NOT NULL,
    senha VARCHAR(200) NOT NULL
);
CREATE TABLE feedback (
    idu INT,
    dataPost TIMESTAMP DEFAULT CURRENT_TIMESTAMP PRIMARY KEY,
    nota INT,
    comentario text,
    FOREIGN KEY (idu) REFERENCES tblogin(idu)
);
CREATE TABLE tbcadastro (
    idu INT,
    dataPost TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    nome VARCHAR(200) NOT NULL,
    email VARCHAR(200) NOT NULL,
    senha VARCHAR(200) NOT NULL,
    FOREIGN KEY (idu) REFERENCES tblogin(idu),
    FOREIGN KEY (dataPost) REFERENCES feedback(dataPost)
);
CREATE TABLE tbpomodoro (
    id INT AUTO_INCREMENT PRIMARY KEY,
    tempoAtiv INT NOT NULL,
    tempoTotal INT NOT NULL,
    horaAtiv DATETIME NOT NULL,
    idPlaneta INT DEFAULT 0
);
CREATE TABLE calendario (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    time TIME NOT NULL,
    location VARCHAR(255) NOT NULL,
    details TEXT,
    dataAtiv DATE NOT NULL,  -- Data do compromisso
    anotacoes TEXT,
```

Figura 7 – Banco de dados

```sql
CREATE TABLE listas (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(255) NOT NULL,
    data_criacao TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
CREATE TABLE tarefas (
    id INT AUTO_INCREMENT PRIMARY KEY,
    lista_id INT,
    nome VARCHAR(255) NOT NULL,
    concluida BOOLEAN DEFAULT FALSE,
    data_criacao TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (lista_id) REFERENCES listas(id) ON DELETE CASCADE
);
```

9. APLICAÇÃO

Figura 8 - Diagrama de caso de uso



9.1 PROTÓTIPAÇÃO

9.1.1 Análise de teste do protótipo

Figura 9 - Relatório Do Teste Usabilidade

RELATÓRIO DO TESTE DE USABILIDADE				
Projeto: Easy Plan			Moderador: Gabriel da Rosa	
Fase: Protótipo	Convidado: Aparecida da Rosa		Quantidade de Tarefas: 1	
Data: 7/9/2024	Local: Etec Abdias			
#	Tarefa	Local da Análise (Problema)	Descrição	Criticidade
1	Utilizar o timer Pomodoro	Tela do timer Pomodoro	não entendeu o significado do ícone de "pausar"	Baixo
2	Check-lists	Marcar as tarefas realizadas com sucesso.	Não foi possível excluir algumas listas e mais uma vez foi feita a reclamação das cores dos botões.	Alta
3				
4				

Figura 10 - Relatório do teste de usabilidade

RELATÓRIO DO TESTE DE USABILIDADE				
Projeto: Easy Plan			Moderador: Gabriel da Rosa	
Fase: Protótipo	Convidado: Daniel da Rosa		Quantidade de Tarefas: 4	
Data: 7/9/2024	Local: Etec Abdias			
#	Tarefa	Local da Análise (Problema)	Descrição	Criticidade
1	Cadastro de usuário	Usuário deve colocar seus dados: Nome; E-mail; Senha: Para realizar o cadastro.	Usuário sentiu dificuldade para progredir no momento de se cadastrar, Os campos estavam difícil de enxergar e não davam clareza ao usuário.	Média
2	Redefinir Senha.	Usuário deve receber em seu e-mail uma confirmação para a mudança de e-mail.	Não foi possível fazer a mudança de senha por motivo do app não estar 100% pronto.	Média
3	Adicionar novo compromisso.	Usuário deve adicionar uma data e hora, uma descrição para o compromisso, local, detalhes(opcional)	Usuário se deu bem com o processo de adicionar novo compromisso, e foi de simples usabilidade. Mas foi necessário a mudança de posição dos botões.	Média
4	Definição de tempo foco.	Usuário deve escolher um tempo limite para dar 100% foco em determinada área. Lembrando que deve ser em minutos.	Fácil usabilidade, a princípio o usuário encontrou problema no jogo de cores da tela.	Bixa

Figura 11 - Relatório do teste de usabilidade

RELATÓRIO DO TESTE DE USABILIDADE				
Projeto: Easy Plan			Moderador: Murilo Gonzalez	
Fase: Protótipo		Convidado: Matheus Juan	Quantidade de Tarefas: 1	
Data: 7/9/2024		Local: Etec Abdias		
#	Tarefa	Local da Análise (Problema)	Descrição	Criticidade
1	Criar um evento	Botão "Salvar"	não encontrou o botão "Salvar", trouxe também a questão do jogo de cores.	Alta
2				
3				

Figura 12 - Relatório do teste de usabilidade

RELATÓRIO DO TESTE DE USABILIDADE				
Projeto: Easy Plan			Moderador: Gabriel da Rosa	
Fase: Protótipo	Convidado: Leonardo da Rosa		Quantidade de Tarefas: 2	
Data: 7/9/2024	Local: Etec Abdias			
#	Tarefa	Local da Análise (Problema)	Descrição	Criticidade
1	Criar um novo evento	Tela de criação de evento	não encontrou a opção para adicionar um local ao evento.	Alto
2	Completar uma tarefa	Lista de tarefas	opção de marcar uma tarefa como "Concluída" não é intuitiva.	Média
3				
4				

10.8 DESCRIÇÃO DA APLICAÇÃO (DETALHAMENTO DAS TELAS)

10.8.1 Tela de Login e Cadastro:

Figura 13 – Tela Cadastro

Figura 14 – Tela Cadastro

A tela de login permite que o usuário acesse o sistema de maneira simples e intuitiva. Nela, há campos específicos para inserir o **e-mail** e a **senha** cadastrados. Caso o usuário não possua uma conta, há a opção de clicar em um botão **"Cadastrar"** que redireciona para a tela de cadastro. Nesta, o usuário pode preencher seus dados, como **nome**, **e-mail** e **senha**, para criar uma conta.

10.8.2 Tela de Agenda:

Figura 15 – Tela Agenda

A tela da agenda oferece funcionalidades para gerenciamento e acompanhamento de atividades e desempenho. Ela apresenta:

1. **Gráfico de Desempenho:**
    a. Um gráfico que exibe o desempenho do usuário nos últimos sete dias, fornecendo uma visão clara e visual dos dados.
2. **Resumo em Destaque:**
    a. Quadrados com informações resumidas que mostram:

   i. O **tempo total** gasto no aplicativo.
   ii. O tempo dedicado no **dia atual**.
   iii. O tempo registrado no **dia anterior**.

3. **Gestão de Compromissos:**
   a. Ferramenta para agendar compromissos, permitindo que o usuário insira:
      i. Nome do compromisso.
      ii. Horário.
      iii. Local (opcional).
      iv. Descrição detalhada, se necessário.
4. **Edição e Controle:**
   a. Funcionalidade de checklist para acompanhar os compromissos realizados.
   b. Opções para editar ou ajustar os compromissos previamente criados, como alterar nomes, adicionar ou remover itens, e atualizar o status de conclusão.

Essa tela é projetada para garantir que o usuário tenha controle total sobre sua agenda e acompanhe seu desempenho diário de maneira eficiente e prática.

10.8.3 Tela do timer:

Figura 16 – Tela do Tempo

A tela do timer permite que o usuário configure e gerencie períodos de tempo para suas atividades. As principais funcionalidades incluem:

1. **Configuração de Tempo:**
    a. O usuário pode inserir o **tempo desejado** para o cronômetro.
2. **Início do Timer:**

   a. Ao pressionar o botão de início, as **notificações são bloqueadas**, garantindo foco total na atividade.

   b. Uma animação visual, representando a **água girando ao redor da Terra**, é iniciada, indicando que o timer está em funcionamento.

3. **Conclusão do Timer:**
   a. Quando o tempo configurado termina, a animação da água para de girar.
   b. O tempo registrado é automaticamente **enviado para o banco de dados**, onde é armazenado e vinculado ao histórico de uso.

Essa tela foi projetada para oferecer uma experiência visual e funcional envolvente, ajudando o usuário a manter o foco enquanto acompanha seu progresso de forma intuitiva.

10.8.4 Tela do game:

Figura 17 – Tela do Tempo

A tela do jogo é uma experiência interativa que combina elementos lúdicos e de progresso com o uso do aplicativo. Suas principais características incluem:

1. **Seleção de Planetas:**
    a. O usuário pode escolher entre diversos planetas, que são desbloqueados progressivamente com base no **tempo total acumulado** no aplicativo.
    b. A evolução dos planetas funciona como um incentivo para manter a consistência no uso do app.

2. **Dinâmica de Foco e Rotação:**
    a. Após selecionar um planeta, ele começa a **girar**, simulando a rotação de um sistema solar.
    b. O tempo de rotação é configurado pelo usuário no **timer**, ajustado ao período definido de foco.
3. **Progresso Visual e Motivacional:**
    a. O movimento contínuo dos planetas reflete o tempo de foco ativo, criando uma representação visual do progresso.
    b. A mecânica gamificada serve como estímulo para os usuários alcançarem novos marcos e desbloquearem novos planetas, fortalecendo o engajamento.

Essa tela foi desenvolvida para transformar o tempo de uso em uma experiência motivadora e divertida, conectando os objetivos pessoais do usuário a uma jornada cósmica de evolução.

10.8.5 Tela do Usuário:

Figura 18 – Tela usuário

Figura 19 – Tela usuário/Email

Figura 20 – Tela usuário/Senha

Figura 21 – Tela usuário/Nome

Figura 22 – Tela usuário/Feedback

A tela do usuário é projetada para fornecer uma visão centralizada das informações pessoais e opções de personalização. Suas principais funcionalidades incluem:

1. **Perfil Pessoal:**
    a. Exibição de dados do usuário, como:
        i. **Foto de perfil.**
        ii. **Nome.**
        iii. **E-mail.**
2. **Edição de Informações:**

a. Opção para editar os seguintes itens:
    i. **Foto de perfil:** o usuário pode alterar ou atualizar a imagem.
    ii. **Nome:** possibilita ajustes no nome cadastrado.
    iii. **E-mail:** para atualizações ou correções.
    iv. **Senha:** opção para redefinir a senha com segurança.

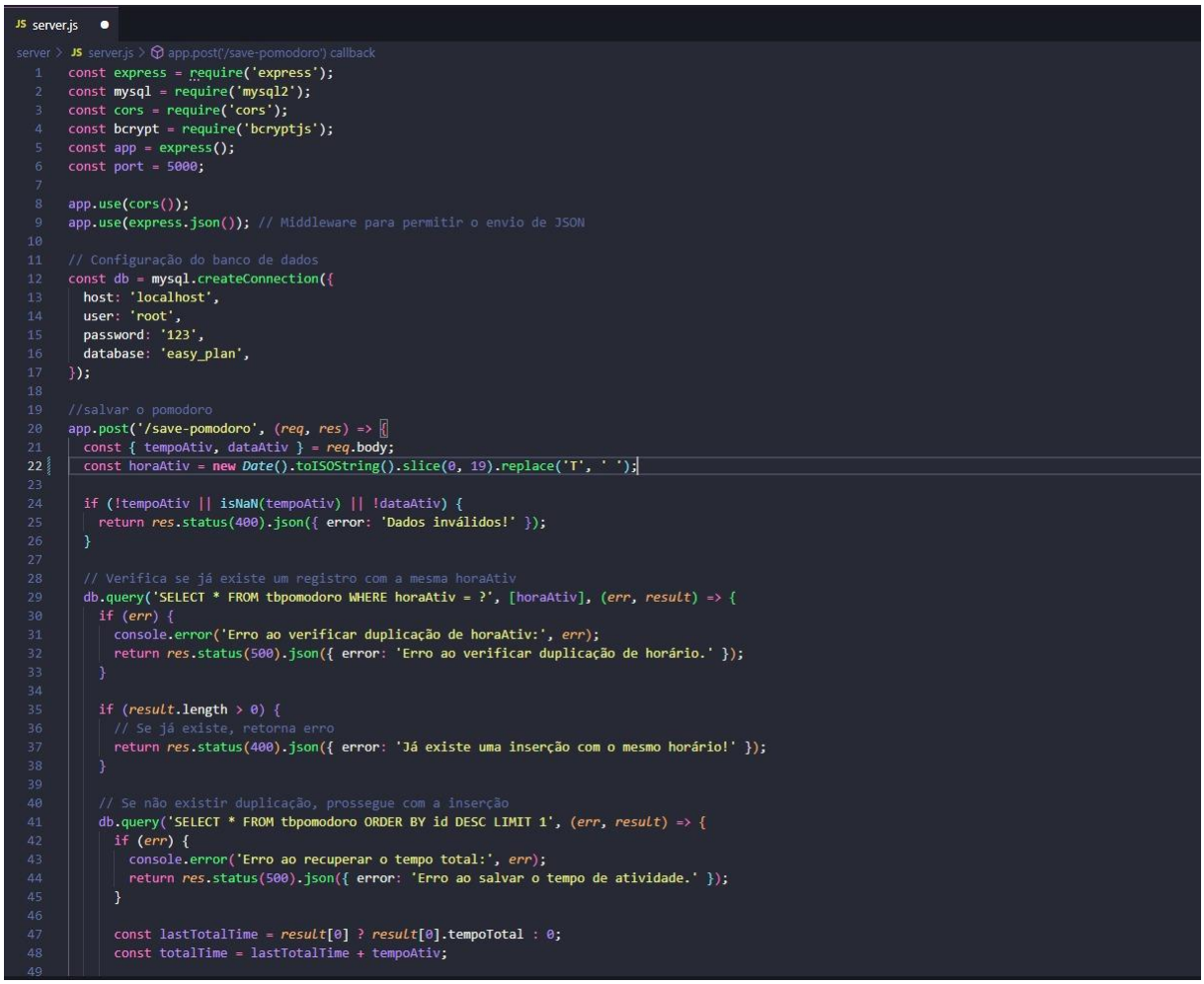
3. **Feedback de Evolução:**
    a. Uma seção dedicada para mostrar o **progresso e evolução** do usuário dentro do aplicativo, como:
        i. Resumo de metas alcançadas.
        ii. Estatísticas de desempenho recente.

Essa tela foi criada para oferecer um espaço personalizado, onde o usuário pode gerenciar suas informações, acompanhar sua evolução e manter suas configurações sempre atualizadas.

11. TESTES (QUALIDADE E TESTE DE SOFTWARE) E CÓDIGOS DA APLICAÇÃO

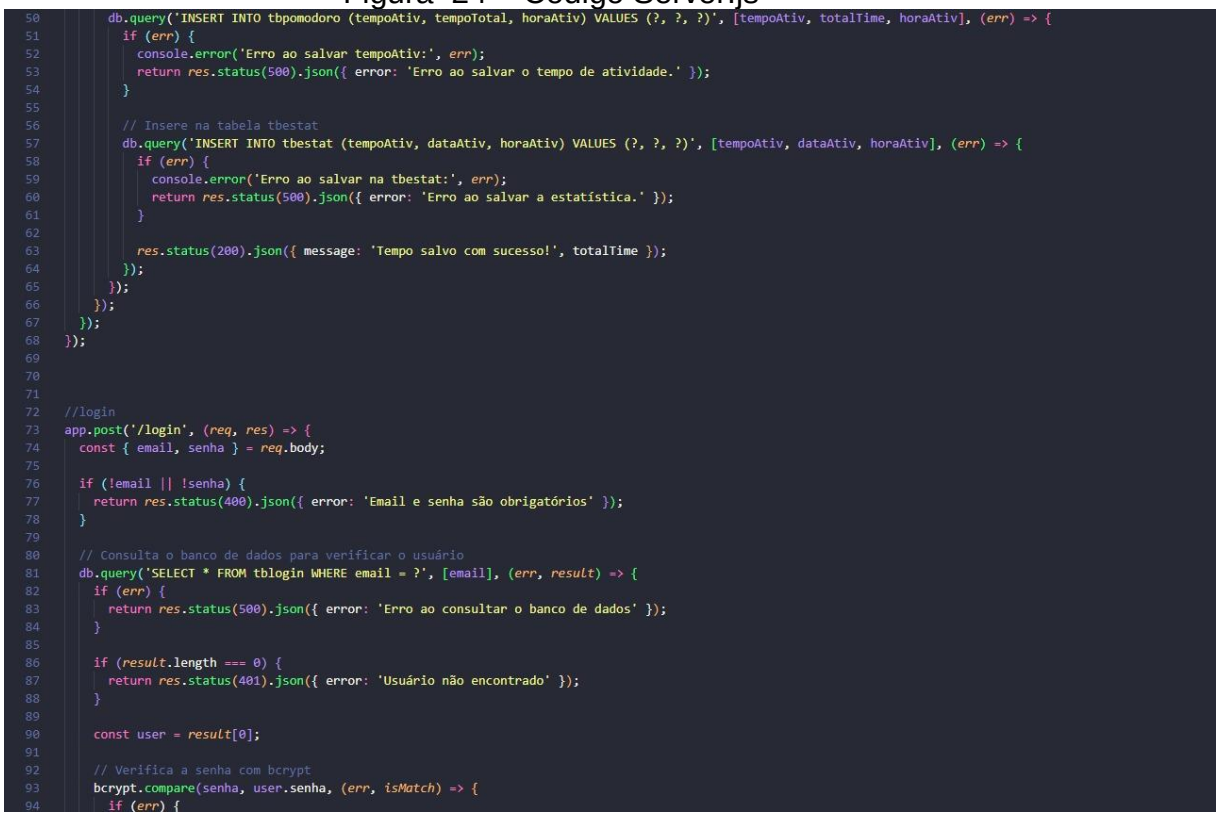
11.1 SERVER.JS

Figura 23 – Código Server.js



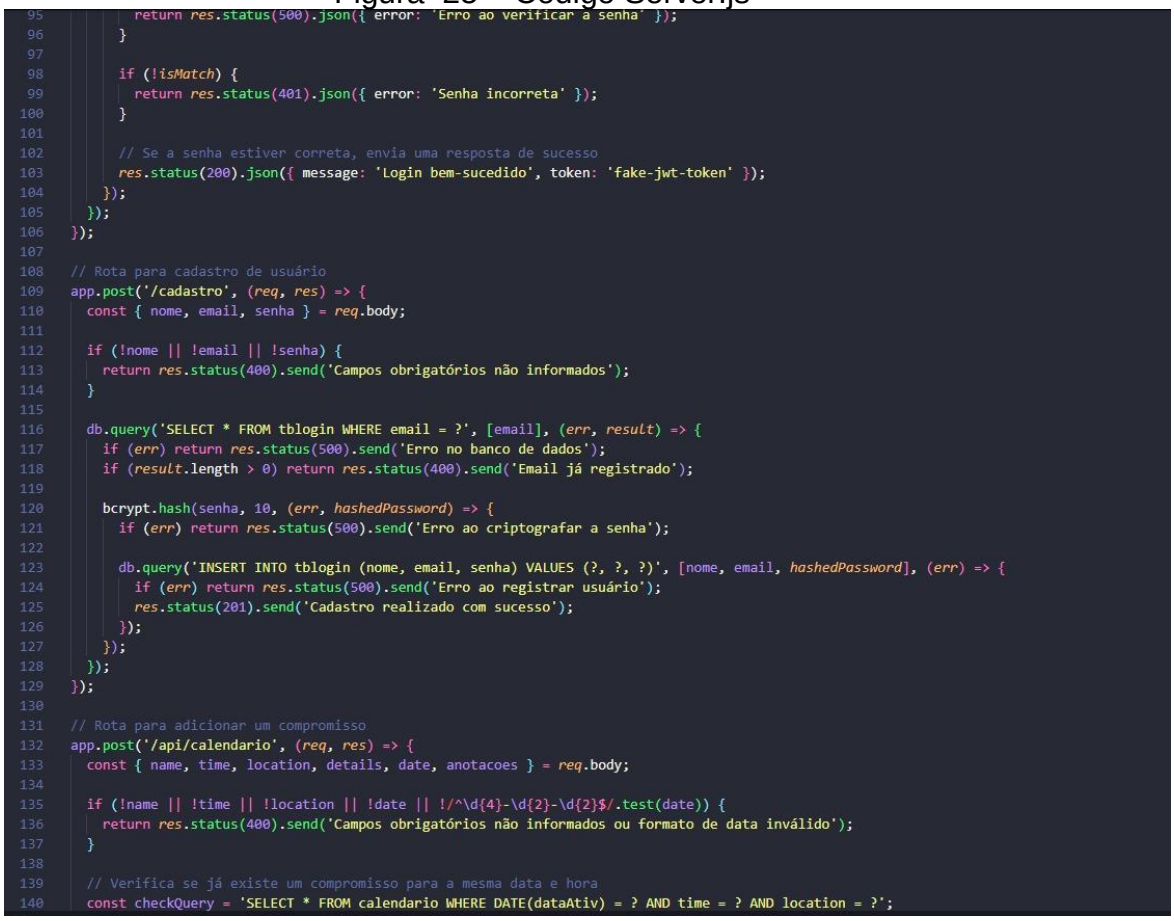
```
server.js
server > server.js > app.post('/save-pomodoro') callback
const express = require('express');
const mysql = require('mysql2');
const cors = require('cors');
const bcrypt = require('bcryptjs');
const app = express();
const port = 5000;

app.use(cors());
app.use(express.json()); // Middleware para permitir o envio de JSON

// Configuração do banco de dados
const db = mysql.createConnection({
  host: 'localhost',
  user: 'root',
  password: '123',
  database: 'easy_plan',
});

//salvar o pomodoro
app.post('/save-pomodoro', (req, res) => {
  const { tempoAtiv, dataAtiv } = req.body;
  const horaAtiv = new Date().toISOString().slice(0, 19).replace('T', ' ');

  if (!tempoAtiv || isNaN(tempoAtiv) || !dataAtiv) {
    return res.status(400).json({ error: 'Dados inválidos!' });
  }

  // Verifica se já existe um registro com a mesma horaAtiv
  db.query('SELECT * FROM tbpomodoro WHERE horaAtiv = ?', [horaAtiv], (err, result) => {
    if (err) {
      console.error('Erro ao verificar duplicação de horaAtiv:', err);
      return res.status(500).json({ error: 'Erro ao verificar duplicação de horário.' });
    }

    if (result.length > 0) {
      // Se já existe, retorna erro
      return res.status(400).json({ error: 'Já existe uma inserção com o mesmo horário!' });
    }

    // Se não existir duplicação, prossegue com a inserção
    db.query('SELECT * FROM tbpomodoro ORDER BY id DESC LIMIT 1', (err, result) => {
      if (err) {
        console.error('Erro ao recuperar o tempo total:', err);
        return res.status(500).json({ error: 'Erro ao salvar o tempo de atividade.' });
      }

      const lastTotalTime = result[0] ? result[0].tempoTotal : 0;
      const totalTime = lastTotalTime + tempoAtiv;
```

Figura 24 – Código Server.js



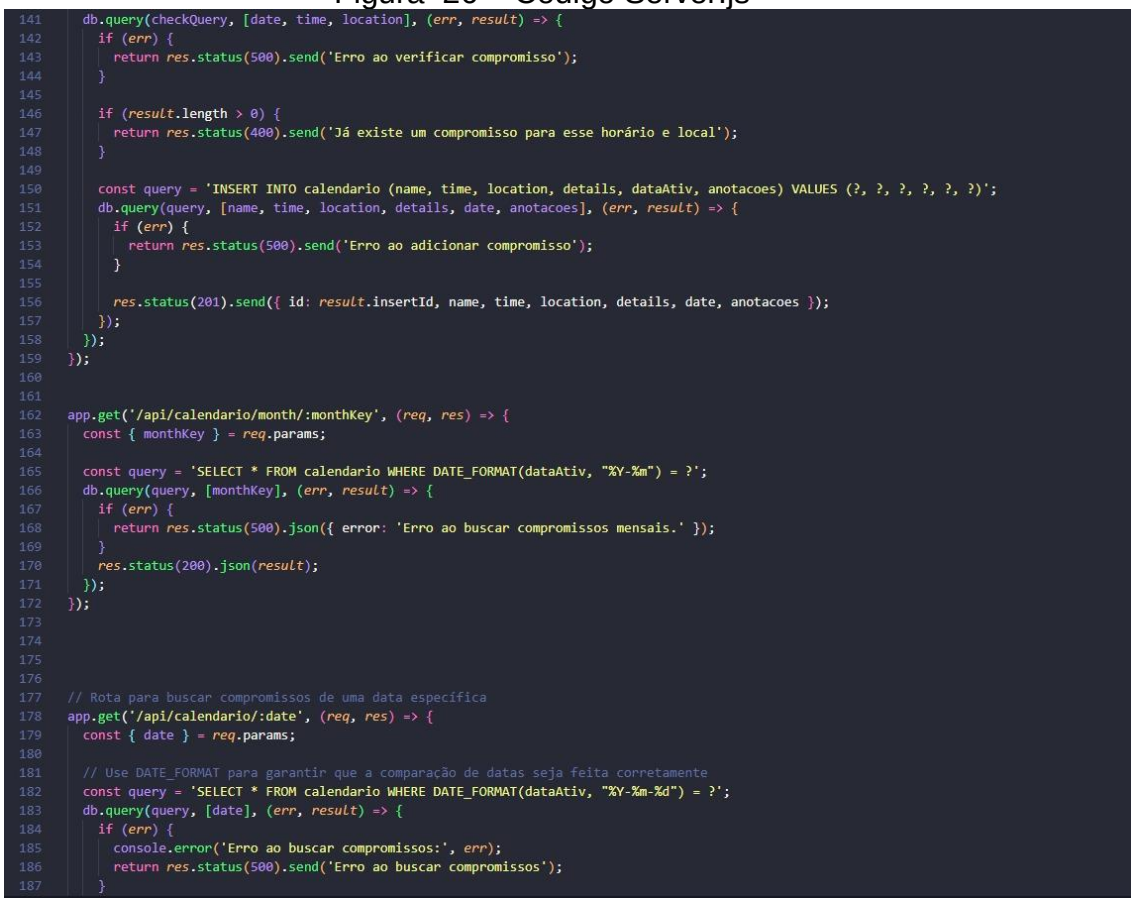
```
      db.query('INSERT INTO tbpomodoro (tempoAtiv, tempoTotal, horaAtiv) VALUES (?, ?, ?)', [tempoAtiv, totalTime, horaAtiv], (err) => {
        if (err) {
          console.error('Erro ao salvar tempoAtiv:', err);
          return res.status(500).json({ error: 'Erro ao salvar o tempo de atividade.' });
        }

        // Insere na tabela tbestat
        db.query('INSERT INTO tbestat (tempoAtiv, dataAtiv, horaAtiv) VALUES (?, ?, ?)', [tempoAtiv, dataAtiv, horaAtiv], (err) => {
          if (err) {
            console.error('Erro ao salvar na tbestat:', err);
            return res.status(500).json({ error: 'Erro ao salvar a estatística.' });
          }

          res.status(200).json({ message: 'Tempo salvo com sucesso!', totalTime });
        });
      });
    });
  });
});


//login
app.post('/login', (req, res) => {
  const { email, senha } = req.body;

  if (!email || !senha) {
    return res.status(400).json({ error: 'Email e senha são obrigatórios' });
  }

  // Consulta o banco de dados para verificar o usuário
  db.query('SELECT * FROM tblogin WHERE email = ?', [email], (err, result) => {
    if (err) {
      return res.status(500).json({ error: 'Erro ao consultar o banco de dados' });
    }

    if (result.length === 0) {
      return res.status(401).json({ error: 'Usuário não encontrado' });
    }

    const user = result[0];

    // Verifica a senha com bcrypt
    bcrypt.compare(senha, user.senha, (err, isMatch) => {
      if (err) {
```

Figura 25 – Código Server.js



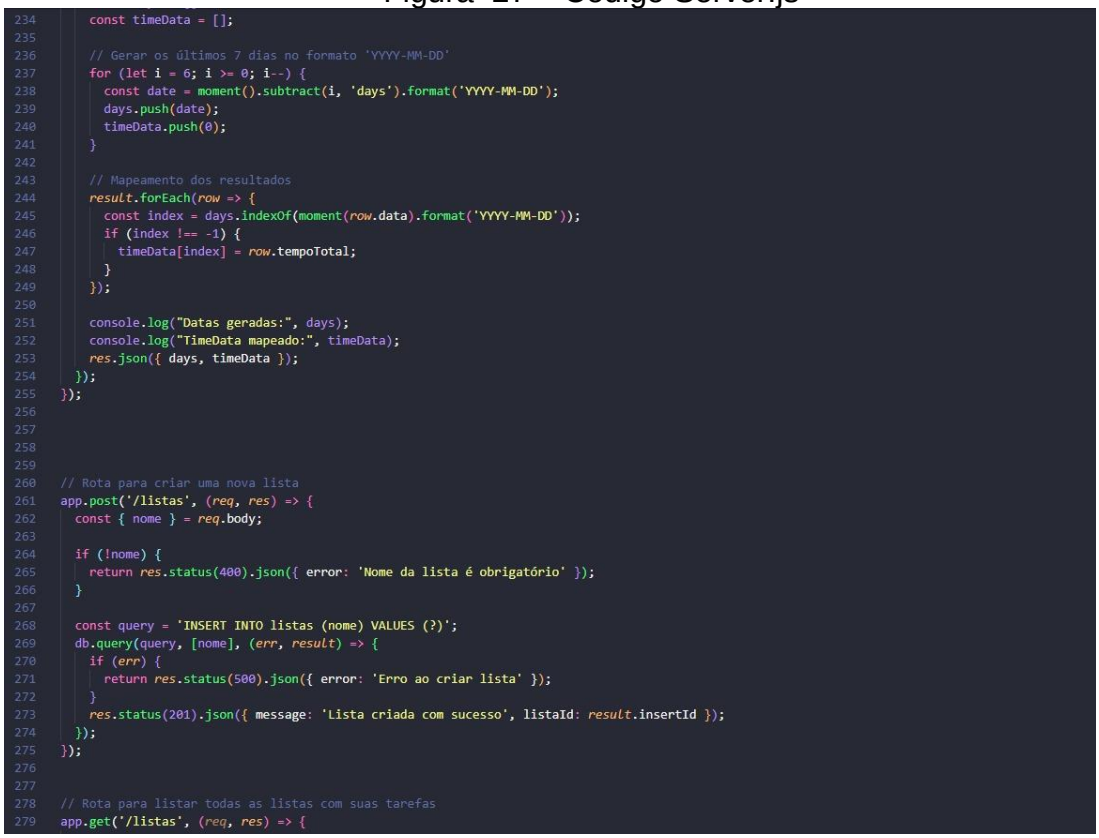
```
        return res.status(500).json({ error: 'Erro ao verificar a senha' });
      }

      if (!isMatch) {
        return res.status(401).json({ error: 'Senha incorreta' });
      }

      // Se a senha estiver correta, envia uma resposta de sucesso
      res.status(200).json({ message: 'Login bem-sucedido', token: 'fake-jwt-token' });
    });
  });
});

// Rota para cadastro de usuário
app.post('/cadastro', (req, res) => {
  const { nome, email, senha } = req.body;

  if (!nome || !email || !senha) {
    return res.status(400).send('Campos obrigatórios não informados');
  }

  db.query('SELECT * FROM tblogin WHERE email = ?', [email], (err, result) => {
    if (err) return res.status(500).send('Erro no banco de dados');
    if (result.length > 0) return res.status(400).send('Email já registrado');

    bcrypt.hash(senha, 10, (err, hashedPassword) => {
      if (err) return res.status(500).send('Erro ao criptografar a senha');

      db.query('INSERT INTO tblogin (nome, email, senha) VALUES (?, ?, ?)', [nome, email, hashedPassword], (err) => {
        if (err) return res.status(500).send('Erro ao registrar usuário');
        res.status(201).send('Cadastro realizado com sucesso');
      });
    });
  });
});

// Rota para adicionar um compromisso
app.post('/api/calendario', (req, res) => {
  const { name, time, location, details, date, anotacoes } = req.body;

  if (!name || !time || !location || !date || !/^\d{4}-\d{2}-\d{2}$/.test(date)) {
    return res.status(400).send('Campos obrigatórios não informados ou formato de data inválido');
  }

  // Verifica se já existe um compromisso para a mesma data e hora
  const checkQuery = 'SELECT * FROM calendario WHERE DATE(dataAtiv) = ? AND time = ? AND location = ?';
```

Figura 26 – Código Server.js



```
    db.query(checkQuery, [date, time, location], (err, result) => {
      if (err) {
        return res.status(500).send('Erro ao verificar compromisso');
      }

      if (result.length > 0) {
        return res.status(400).send('Já existe um compromisso para esse horário e local');
      }

      const query = 'INSERT INTO calendario (name, time, location, details, dataAtiv, anotacoes) VALUES (?, ?, ?, ?, ?, ?)';
      db.query(query, [name, time, location, details, date, anotacoes], (err, result) => {
        if (err) {
          return res.status(500).send('Erro ao adicionar compromisso');
        }

        res.status(201).send({ id: result.insertId, name, time, location, details, date, anotacoes });
      });
    });
  });


  app.get('/api/calendario/month/:monthKey', (req, res) => {
    const { monthKey } = req.params;

    const query = 'SELECT * FROM calendario WHERE DATE_FORMAT(dataAtiv, "%Y-%m") = ?';
    db.query(query, [monthKey], (err, result) => {
      if (err) {
        return res.status(500).json({ error: 'Erro ao buscar compromissos mensais.' });
      }
      res.status(200).json(result);
    });
  });




  // Rota para buscar compromissos de uma data específica
  app.get('/api/calendario/:date', (req, res) => {
    const { date } = req.params;

    // Use DATE_FORMAT para garantir que a comparação de datas seja feita corretamente
    const query = 'SELECT * FROM calendario WHERE DATE_FORMAT(dataAtiv, "%Y-%m-%d") = ?';
    db.query(query, [date], (err, result) => {
      if (err) {
        console.error('Erro ao buscar compromissos:', err);
        return res.status(500).send('Erro ao buscar compromissos');
      }
```

Figura 27 – Código Server.js



```
    const timeData = [];

    // Gerar os últimos 7 dias no formato 'YYYY-MM-DD'
    for (let i = 6; i >= 0; i--) {
      const date = moment().subtract(i, 'days').format('YYYY-MM-DD');
      days.push(date);
      timeData.push(0);
    }

    // Mapeamento dos resultados
    result.forEach(row => {
      const index = days.indexOf(moment(row.data).format('YYYY-MM-DD'));
      if (index !== -1) {
        timeData[index] = row.tempoTotal;
      }
    });

    console.log("Datas geradas:", days);
    console.log("TimeData mapeado:", timeData);
    res.json({ days, timeData });
  });
});




// Rota para criar uma nova lista
app.post('/listas', (req, res) => {
  const { nome } = req.body;

  if (!nome) {
    return res.status(400).json({ error: 'Nome da lista é obrigatório' });
  }

  const query = 'INSERT INTO listas (nome) VALUES (?)';
  db.query(query, [nome], (err, result) => {
    if (err) {
      return res.status(500).json({ error: 'Erro ao criar lista' });
    }
    res.status(201).json({ message: 'Lista criada com sucesso', listaId: result.insertId });
  });
});


// Rota para listar todas as listas com suas tarefas
app.get('/listas', (req, res) => {
```

Figura 28– Código Server.js

```
  const query = `
    SELECT listas.id, listas.nome, listas.data_criacao,
      (SELECT JSON_ARRAYAGG(
          JSON_OBJECT('id', tarefas.id, 'nome', tarefas.nome, 'concluida', tarefas.concluida)
        )
       FROM tarefas WHERE tarefas.lista_id = listas.id) AS tarefas
    FROM listas
  `;

  db.query(query, (err, lists) => {
    if (err) {
      return res.status(500).json({ error: 'Erro ao buscar listas' });
    }
    res.status(200).json(lists);
  });
});

// Rota para alterar o nome de uma lista
app.put('/listas/:id', (req, res) => {
  const listaId = req.params.id;
  const { nome } = req.body;

  if (!nome) {
    return res.status(400).json({ error: 'Nome da lista é obrigatório' });
  }

  const query = 'UPDATE listas SET nome = ? WHERE id = ?';
  db.query(query, [nome, listaId], (err, result) => {
    if (err) {
      return res.status(500).json({ error: 'Erro ao atualizar lista' });
    }

    if (result.affectedRows === 0) {
      return res.status(404).json({ error: 'Lista não encontrada' });
    }

    res.status(200).json({ message: 'Lista atualizada com sucesso' });
  });
});



// Rota para criar uma nova tarefa
app.post('/tarefas', (req, res) => {
  const { lista_id, nome } = req.body;

  if (!lista_id || !nome) {
```

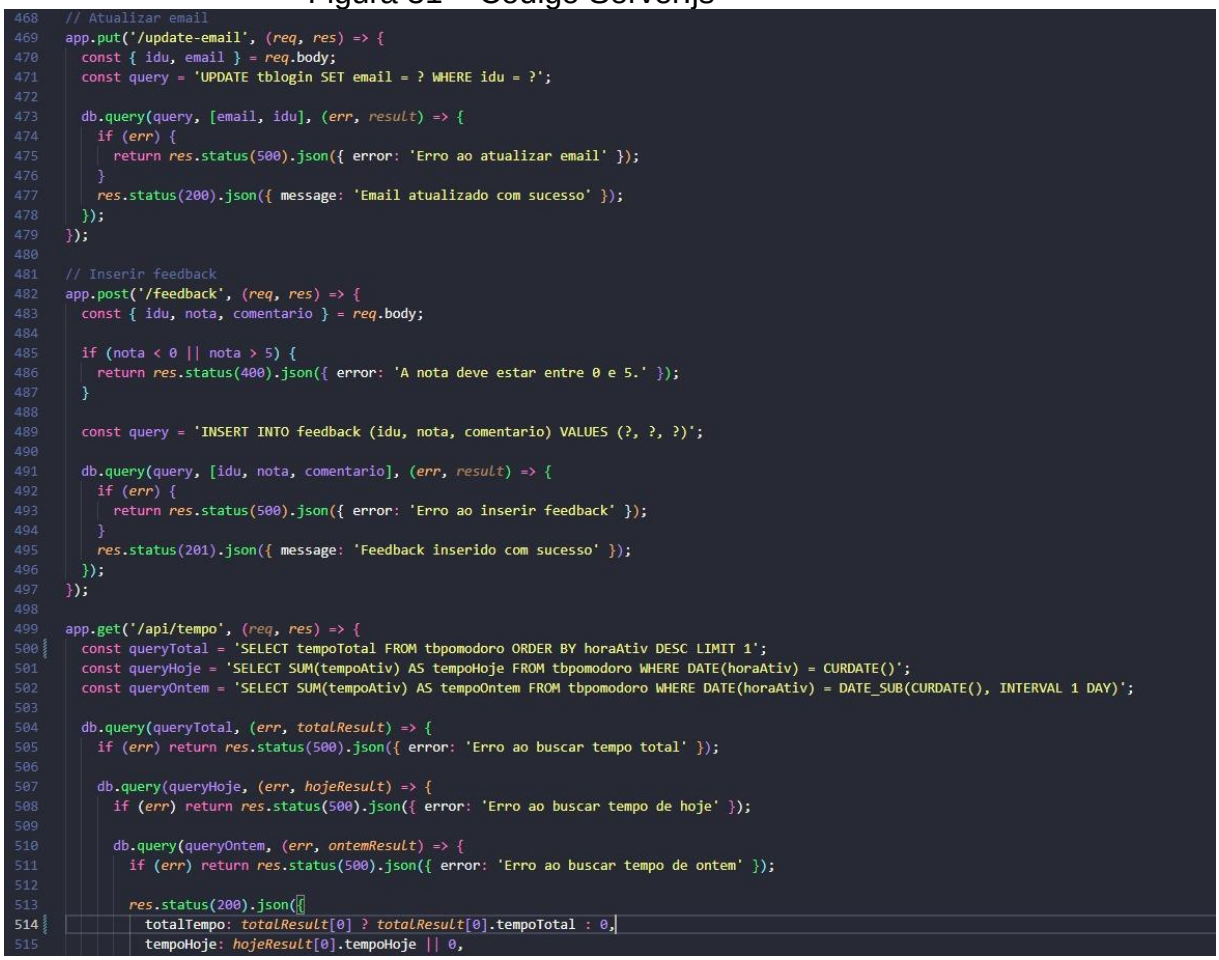
Figura 29 – Código Server.js

```
    return res.status(400).json({ error: 'Lista ID e nome da tarefa são obrigatórios' });
  }

  const query = 'INSERT INTO tarefas (lista_id, nome) VALUES (?, ?)';
  db.query(query, [lista_id, nome], (err, result) => {
    if (err) {
      return res.status(500).json({ error: 'Erro ao criar tarefa' });
    }
    res.status(201).json({ message: 'Tarefa criada com sucesso', tarefaId: result.insertId });
  });
});

// Rota para listar as tarefas de uma lista específica
app.get('/listas/:id/tarefas', (req, res) => {
  const listaId = req.params.id;
  const query = 'SELECT * FROM tarefas WHERE lista_id = ?';

  db.query(query, [listaId], (err, tasks) => {
    if (err) {
      return res.status(500).json({ error: 'Erro ao buscar tarefas' });
    }
    res.status(200).json(tasks);
  });
});

// Rota para atualizar o status da tarefa (concluída ou não)
app.put('/tarefas/:id', (req, res) => {
  const tarefaId = req.params.id;
  const { concluida } = req.body;

  if (typeof concluida !== 'boolean') {
    return res.status(400).json({ error: 'O status de conclusão deve ser um valor booleano' });
  }

  const query = 'UPDATE tarefas SET concluida = ? WHERE id = ?';
  db.query(query, [concluida, tarefaId], (err, result) => {
    if (err) {
      return res.status(500).json({ error: 'Erro ao atualizar tarefa' });
    }
    res.status(200).json({ message: 'Tarefa atualizada com sucesso' });
  });
});

// Rota para excluir uma tarefa
app.delete('/tarefas/:id', (req, res) => {
  const tarefaId = req.params.id;
  const query = 'DELETE FROM tarefas WHERE id = ?';
```

Figura 30 – Código Server.js

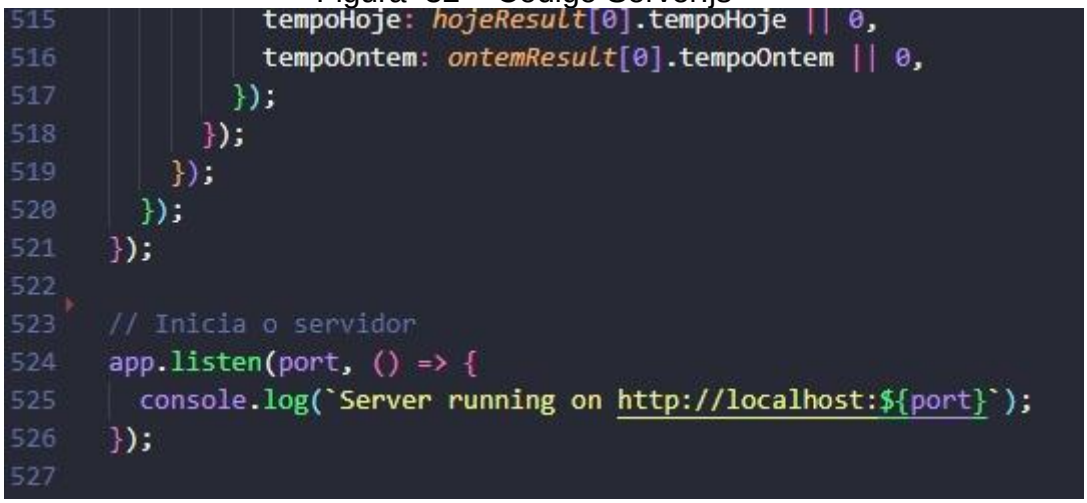
```
374
375     db.query(query, [tarefaId], (err, result) => {
376       if (err) {
377         return res.status(500).json({ error: 'Erro ao excluir tarefa' });
378       }
379       res.status(200).json({ message: 'Tarefa excluída com sucesso' });
380     });
381   });
382
383
384   // Rota para excluir uma lista e suas tarefas associadas
385   app.delete('/listas/:id', (req, res) => {
386     const listaId = req.params.id;
387
388     // Exclui as tarefas associadas antes de excluir a lista
389     const deleteTasksQuery = 'DELETE FROM tarefas WHERE lista_id = ?';
390     db.query(deleteTasksQuery, [listaId], (err) => {
391       if (err) {
392         return res.status(500).json({ error: 'Erro ao excluir tarefas da lista' });
393       }
394
395       // Exclui a lista
396       const deleteListQuery = 'DELETE FROM listas WHERE id = ?';
397       db.query(deleteListQuery, [listaId], (err) => {
398         if (err) {
399           return res.status(500).json({ error: 'Erro ao excluir lista' });
400         }
401         res.status(200).json({ message: 'Lista excluída com sucesso' });
402       });
403     });
404   });
405
406   app.get('/get-last-pomodoro', (req, res) => {
407     const query = 'SELECT idPlaneta FROM tbpomodoro ORDER BY id DESC LIMIT 1';
408     db.query(query, (err, result) => {
409       if (err) return res.status(500).json({ error: 'Erro ao buscar último pomodoro' });
410       res.status(200).json(result[0] || { idPlaneta: null });
411     });
412   });
413
414   app.put('/update-planet', (req, res) => {
415     const { idPlaneta } = req.body;
416     const query = 'UPDATE tbpomodoro SET idPlaneta = ? ORDER BY id DESC LIMIT 1';
417     db.query(query, [idPlaneta], (err) => {
418       if (err) return res.status(500).json({ error: 'Erro ao atualizar planeta' });
419       res.status(200).json({ message: 'Planeta atualizado com sucesso' });
420     });
```

Figura 31 – Código Server.js



```
// Atualizar email
app.put('/update-email', (req, res) => {
  const { idu, email } = req.body;
  const query = 'UPDATE tblogin SET email = ? WHERE idu = ?';

  db.query(query, [email, idu], (err, result) => {
    if (err) {
      return res.status(500).json({ error: 'Erro ao atualizar email' });
    }
    res.status(200).json({ message: 'Email atualizado com sucesso' });
  });
});

// Inserir feedback
app.post('/feedback', (req, res) => {
  const { idu, nota, comentario } = req.body;

  if (nota < 0 || nota > 5) {
    return res.status(400).json({ error: 'A nota deve estar entre 0 e 5.' });
  }

  const query = 'INSERT INTO feedback (idu, nota, comentario) VALUES (?, ?, ?)';

  db.query(query, [idu, nota, comentario], (err, result) => {
    if (err) {
      return res.status(500).json({ error: 'Erro ao inserir feedback' });
    }
    res.status(201).json({ message: 'Feedback inserido com sucesso' });
  });
});

app.get('/api/tempo', (req, res) => {
  const queryTotal = 'SELECT tempoTotal FROM tbpomodoro ORDER BY horaAtiv DESC LIMIT 1';
  const queryHoje = 'SELECT SUM(tempoAtiv) AS tempoHoje FROM tbpomodoro WHERE DATE(horaAtiv) = CURDATE()';
  const queryOntem = 'SELECT SUM(tempoAtiv) AS tempoOntem FROM tbpomodoro WHERE DATE(horaAtiv) = DATE_SUB(CURDATE(), INTERVAL 1 DAY)';

  db.query(queryTotal, (err, totalResult) => {
    if (err) return res.status(500).json({ error: 'Erro ao buscar tempo total' });

    db.query(queryHoje, (err, hojeResult) => {
      if (err) return res.status(500).json({ error: 'Erro ao buscar tempo de hoje' });

      db.query(queryOntem, (err, ontemResult) => {
        if (err) return res.status(500).json({ error: 'Erro ao buscar tempo de ontem' });

        res.status(200).json({
          totalTempo: totalResult[0] ? totalResult[0].tempoTotal : 0,
          tempoHoje: hojeResult[0].tempoHoje || 0,
```

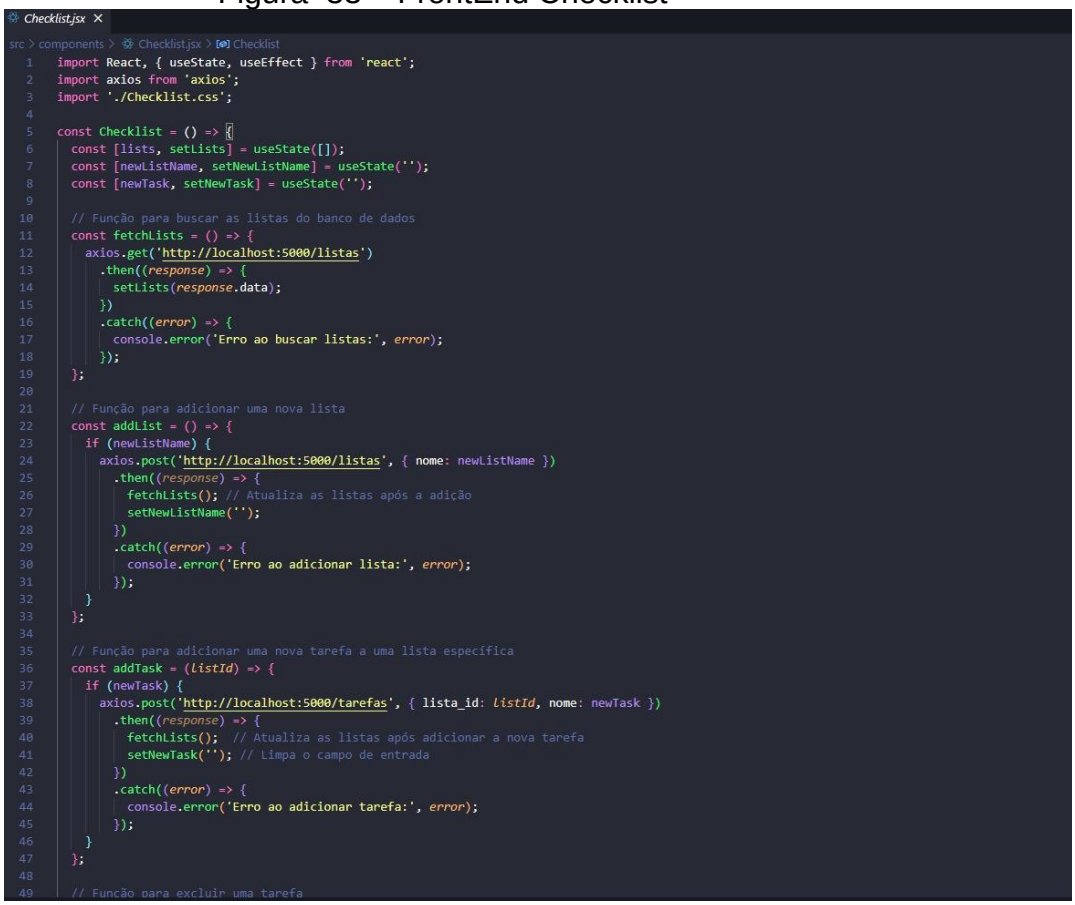
Figura 32 – Código Server.js



```
515                 tempoHoje: hojeResult[0].tempoHoje || 0,
516                 tempoOntem: ontemResult[0].tempoOntem || 0,
517               });
518             });
519           });
520         });
521       });
522
523       // Inicia o servidor
524       app.listen(port, () => {
525         console.log(`Server running on http://localhost:${port}`);
526       });
527
```

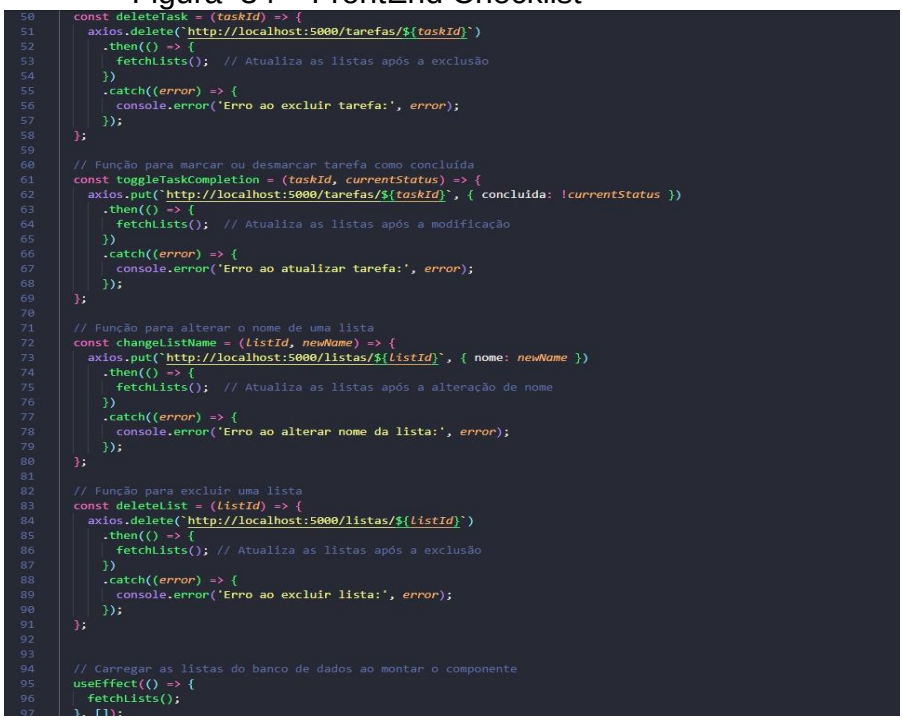
11.2 FRONTEND/CHECKLIST:

Figura 33 – FrontEnd Checklist



```jsx
import React, { useState, useEffect } from 'react';
import axios from 'axios';
import './Checklist.css';

const Checklist = () => {
  const [lists, setLists] = useState([]);
  const [newListName, setNewListName] = useState('');
  const [newTask, setNewTask] = useState('');

  // Função para buscar as listas do banco de dados
  const fetchLists = () => {
    axios.get('http://localhost:5000/listas')
      .then((response) => {
        setLists(response.data);
      })
      .catch((error) => {
        console.error('Erro ao buscar listas:', error);
      });
  };

  // Função para adicionar uma nova lista
  const addList = () => {
    if (newListName) {
      axios.post('http://localhost:5000/listas', { nome: newListName })
        .then((response) => {
          fetchLists(); // Atualiza as listas após a adição
          setNewListName('');
        })
        .catch((error) => {
          console.error('Erro ao adicionar lista:', error);
        });
    }
  };

  // Função para adicionar uma nova tarefa a uma lista específica
  const addTask = (listId) => {
    if (newTask) {
      axios.post('http://localhost:5000/tarefas', { lista_id: listId, nome: newTask })
        .then((response) => {
          fetchLists();  // Atualiza as listas após adicionar a nova tarefa
          setNewTask(''); // Limpa o campo de entrada
        })
        .catch((error) => {
          console.error('Erro ao adicionar tarefa:', error);
        });
    }
  };

  // Função para excluir uma tarefa
```

Figura 34 – FrontEnd Checklist

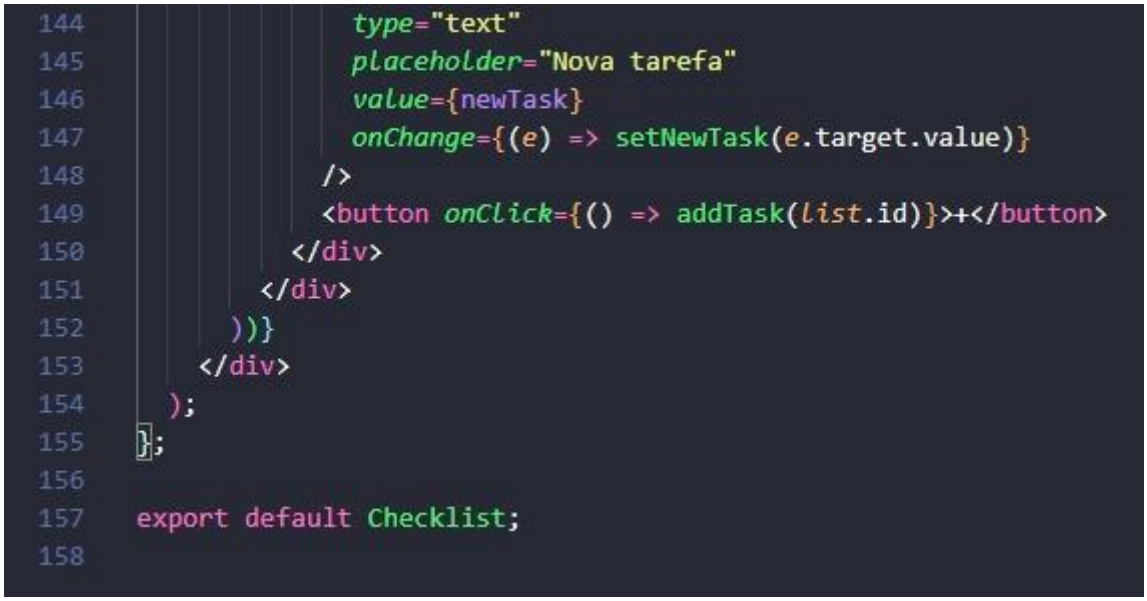


```jsx
  const deleteTask = (taskId) => {
    axios.delete(`http://localhost:5000/tarefas/${taskId}`)
      .then(() => {
        fetchLists();  // Atualiza as listas após a exclusão
      })
      .catch((error) => {
        console.error('Erro ao excluir tarefa:', error);
      });
  };

  // Função para marcar ou desmarcar tarefa como concluída
  const toggleTaskCompletion = (taskId, currentStatus) => {
    axios.put(`http://localhost:5000/tarefas/${taskId}`, { concluída: !currentStatus })
      .then(() => {
        fetchLists();  // Atualiza as listas após a modificação
      })
      .catch((error) => {
        console.error('Erro ao atualizar tarefa:', error);
      });
  };

  // Função para alterar o nome de uma lista
  const changeListName = (listId, newName) => {
    axios.put(`http://localhost:5000/listas/${listId}`, { nome: newName })
      .then(() => {
        fetchLists();  // Atualiza as listas após a alteração de nome
      })
      .catch((error) => {
        console.error('Erro ao alterar nome da lista:', error);
      });
  };

  // Função para excluir uma lista
  const deleteList = (listId) => {
    axios.delete(`http://localhost:5000/listas/${listId}`)
      .then(() => {
        fetchLists(); // Atualiza as listas após a exclusão
      })
      .catch((error) => {
        console.error('Erro ao excluir lista:', error);
      });
  };


  // Carregar as listas do banco de dados ao montar o componente
  useEffect(() => {
    fetchLists();
  }, []);
```

Figura 35 – FrontEnd Checklist

```
}, []);

return (
  <div className="checklist-container">
    <h1>Check-lists</h1>

    <div className=' nomeLista'>
      <input
        type="text"
        placeholder="Nome da nova lista"
        value={newListName}
        onChange={(e) => setNewListName(e.target.value)}
      />
      <button onClick={addList}>+</button>
    </div>

    {lists.map((list) => (
      <div key={list.id} className="list">
        <div className="list-header">
          <input
            type="text"
            value={list.nome}
            onChange={(e) => changeListName(list.id, e.target.value)}
          />
          <button onClick={() => deleteList(list.id)}>Excluir Lista</button>
        </div>

        <div className="tasks">
          {list.tarefas && list.tarefas.length > 0 ? (
            list.tarefas.map((task) => (
              <div key={task.id} className="task">
                <input
                  type="checkbox"
                  checked={task.concluida}
                  onChange={() => toggleTaskCompletion(task.id, task.concluida)}
                />
                <span className={task.concluida ? 'completed' : ''}>{task.nome}</span>
                <button onClick={() => deleteTask(task.id)}>X</button>
              </div>
            ))
          ) : (
            <p>Nenhuma tarefa nesta lista.</p>
          )}
        </div>

        <div className="add-task">
          <input
            type="text"
```

Figura 36 – FrontEnd Checklist



```
                    type="text"
                    placeholder="Nova tarefa"
                    value={newTask}
                    onChange={(e) => setNewTask(e.target.value)}
                  />
                  <button onClick={() => addTask(list.id)}>+</button>
                </div>
              </div>
            ))}
          </div>
        );
      };

      export default Checklist;
```

Figura 37 – FrontEnd Checklist

Checklist.css

src > components > # Checklist.css > .completed

```css
.checklist-container {
  font-family: 'Arial', sans-serif;
  max-width: 700px;
  margin: 0 auto;
  padding: 20px;
  border-radius: 10px;
  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
}
.nomeLista button{
  margin-left: 15px;
  border:2px solid #4caf50;
  background-color: #388e3c00;
  color: #4caf50;
}
.nomeLista input {
  padding: 4px;
  flex: 0.2;
  border: 1px solid #ddd;
  border-radius: 5px;
  font-size: 0.9rem;
}
h1 {
  text-align: center;
  color: white;
  font-size: 2rem;
  margin-bottom: 20px;
}
.list {
  border: 1px solid #ddd;
  color: white;
  border-radius: 8px;
  padding: 15px;
  margin: 10px 0;
  box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
}
.list-header {
  display: flex;
  justify-content: space-between;
  align-items: center;
  margin-bottom: 15px;
}
.list-header input {
  font-size: 1rem;
  border: none;
  color: white;
  background: transparent;
  flex: 0.5;
  padding: 5px;
}
```

Figura 38 – FrontEnd Checklist

```
}
.list-header button {
  background-color: rgba(0, 0, 0, 0);
  color: rgb(255, 0, 0);
  border: 2px solid rgb(255, 0, 0);
  padding: 8px 10px;
  cursor: pointer;
  border-radius: 5px;
  font-size: 0.7rem;
}
.list-header button:hover {
  background-color: #d32f2f;
}
.tasks {
  margin-top: 10px;
}
.task {
  display: flex;
  align-items: center;
  margin: 8px 0;
}
.task span {
  margin-left: 10px;
  flex: 1;
  font-size: 1rem;
}
.task input[type="checkbox"] {
  margin-right: 10px;
}
.task button {
  background-color: #f4433600;
  color: rgb(255, 0, 0);
  border: 2px solid rgb(255, 0, 0);
  padding: 6px 12px;
  cursor: pointer;
  margin-left: 10px;
  border-radius: 5px;
  font-size: 0.7rem;
}
.task button:hover {
  background-color: #d32f2f;
}
.add-task {
  margin-top: 15px;
  display: flex;
  justify-content: space-between;
  align-items: center;
}
```

Figura 39 – FrontEnd Checklist

```
}
.add-task input {
  padding: 8px;
  flex: 0.5;
  border: 1px solid #ddd;
  border-radius: 5px;
  font-size: 1rem;
}
.add-task button {
  background-color: #4caf4f00;
  color: #4caf50;
  border: 2px solid #4caf50;
  padding: 8px 10px;
  cursor: pointer;
  border-radius: 13px;
  font-size: 1rem;
  margin-top: 0px;
}
.add-task button:hover {
  background-color: #388e3c;
}
.completed {
  text-decoration: line-through;
  color: #888;
}
```

11.3 FRONTEND/GAME

Figura 40 – Game

Game.jsx

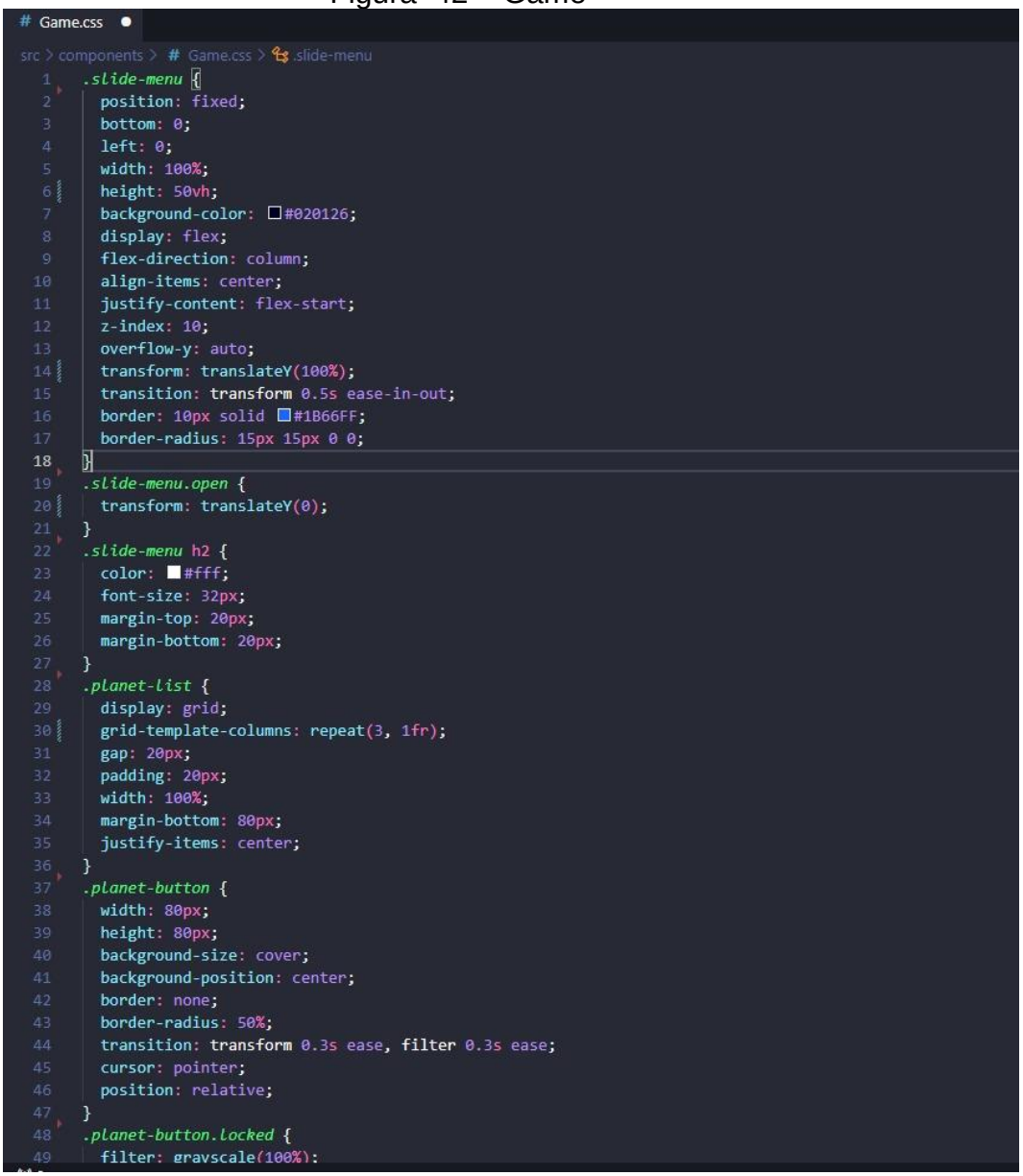
src > components > Game.jsx > Game > useEffect() callback > handleClickOutside

```jsx
import React, { useState, useEffect, useRef } from 'react';
import './Game.css'; // Estilização do menu deslizante
import axios from 'axios';

const planets = [
  { id: 0, image: '/planetas/0.svg', requiredHours: 0 },
  { id: 1, image: '/planetas/1.svg', requiredHours: 2 },
  { id: 2, image: '/planetas/2.svg', requiredHours: 5 },
  { id: 3, image: '/planetas/3.svg', requiredHours: 10 },
  { id: 4, image: '/planetas/4.svg', requiredHours: 15 },
  { id: 5, image: '/planetas/5.svg', requiredHours: 20 },
  { id: 6, image: '/planetas/6.svg', requiredHours: 25 },
  { id: 7, image: '/planetas/7.svg', requiredHours: 50 },
  { id: 8, image: '/planetas/8.svg', requiredHours: 60 },
  { id: 9, image: '/planetas/9.svg', requiredHours: 80 },
];

const Game = ({ onSelectPlanet }) => {
  const [totalHours, setTotalHours] = useState(0);
  const [menuOpen, setMenuOpen] = useState(false);
  const [selectedPlanetId, setSelectedPlanetId] = useState(null);
  const menuRef = useRef(null);

  useEffect(() => {
    // Fetch tempoTotal em minutos e converte para horas
    axios.get('http://localhost:5000/get-total-time')
      .then((response) => {
        const totalMinutes = response.data.tempoTotal || 0;
        setTotalHours(totalMinutes / 60);
      })
      .catch((error) => console.error('Erro ao buscar tempo total:', error));

    // Função para fechar o menu ao clicar fora
    const handleClickOutside = (event) => {
      if (menuRef.current && !menuRef.current.contains(event.target)) {
        setMenuOpen(false);
      }
    };

    // Adiciona o evento de clique fora do menu
    document.addEventListener('mousedown', handleClickOutside);

    // Remove o evento quando o componente for desmontado
    return () => {
      document.removeEventListener('mousedown', handleClickOutside);
    };
  }, []);

  return (
```

Figura 41 – Game

```
49    return (
50      <div>
51        {/* Este botão está sendo chamado com uma imagem */}
52        <div
53          className="large-ball"
54          onClick={() => setMenuOpen(true)} // Abre o menu ao clicar na bola
55        />
56
57        {/* Menu deslizante com planetas */}
58        <div className={`slide-menu ${menuOpen ? 'open' : 'closed'}`} ref={menuRef}>
59          <h2>Selecione um Planeta</h2>
60          <div className="planet-list">
61            {planets.map((planet) => (
62              <button
63                key={planet.id}
64                className={`planet-button ${totalHours >= planet.requiredHours ? '' : 'locked'}`}
65                onClick={() => {
66                  if (totalHours >= planet.requiredHours) {
67                    setSelectedPlanetId(planet.id);
68                    onSelectPlanet(planet.id);
69                  }
70                }}
71                disabled={totalHours < planet.requiredHours}
72                style={{ backgroundImage: `url(${planet.image})` }}
73              >
74                {totalHours < planet.requiredHours && (
75                  <span className="locked-message">{planet.requiredHours}h</span>
76                )}
77              </button>
78            ))}
79          </div>
80        </div>
81      </div>
82    );
83  };
84
85  export default Game;
86
```

Figura 42 – Game



Game.css

src > components > # Game.css > .slide-menu

```css
.slide-menu {
  position: fixed;
  bottom: 0;
  left: 0;
  width: 100%;
  height: 50vh;
  background-color: #020126;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: flex-start;
  z-index: 10;
  overflow-y: auto;
  transform: translateY(100%);
  transition: transform 0.5s ease-in-out;
  border: 10px solid #1B66FF;
  border-radius: 15px 15px 0 0;
}
.slide-menu.open {
  transform: translateY(0);
}
.slide-menu h2 {
  color: #fff;
  font-size: 32px;
  margin-top: 20px;
  margin-bottom: 20px;
}
.planet-list {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  gap: 20px;
  padding: 20px;
  width: 100%;
  margin-bottom: 80px;
  justify-items: center;
}
.planet-button {
  width: 80px;
  height: 80px;
  background-size: cover;
  background-position: center;
  border: none;
  border-radius: 50%;
  transition: transform 0.3s ease, filter 0.3s ease;
  cursor: pointer;
  position: relative;
}
.planet-button.locked {
  filter: grayscale(100%);
```

Figura 43 – Game

```
    filter: grayscale(100%);
    cursor: not-allowed;
  }
  .planet-button:not(.locked):hover {
    transform: scale(1.2);
  }
  .planet-button:disabled::after {
    content: attr(data-hours) "h";
    display: block;
    color: #fff;
    text-align: center;
    font-size: 14px;
    font-weight: bold;
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
  }
  .locked-message {
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
    color: #fff;
    font-size: 16px;
    font-weight: bold;
  }
```

11.4 FONTEND/LOGIN

Figura 44 – Fontend/log

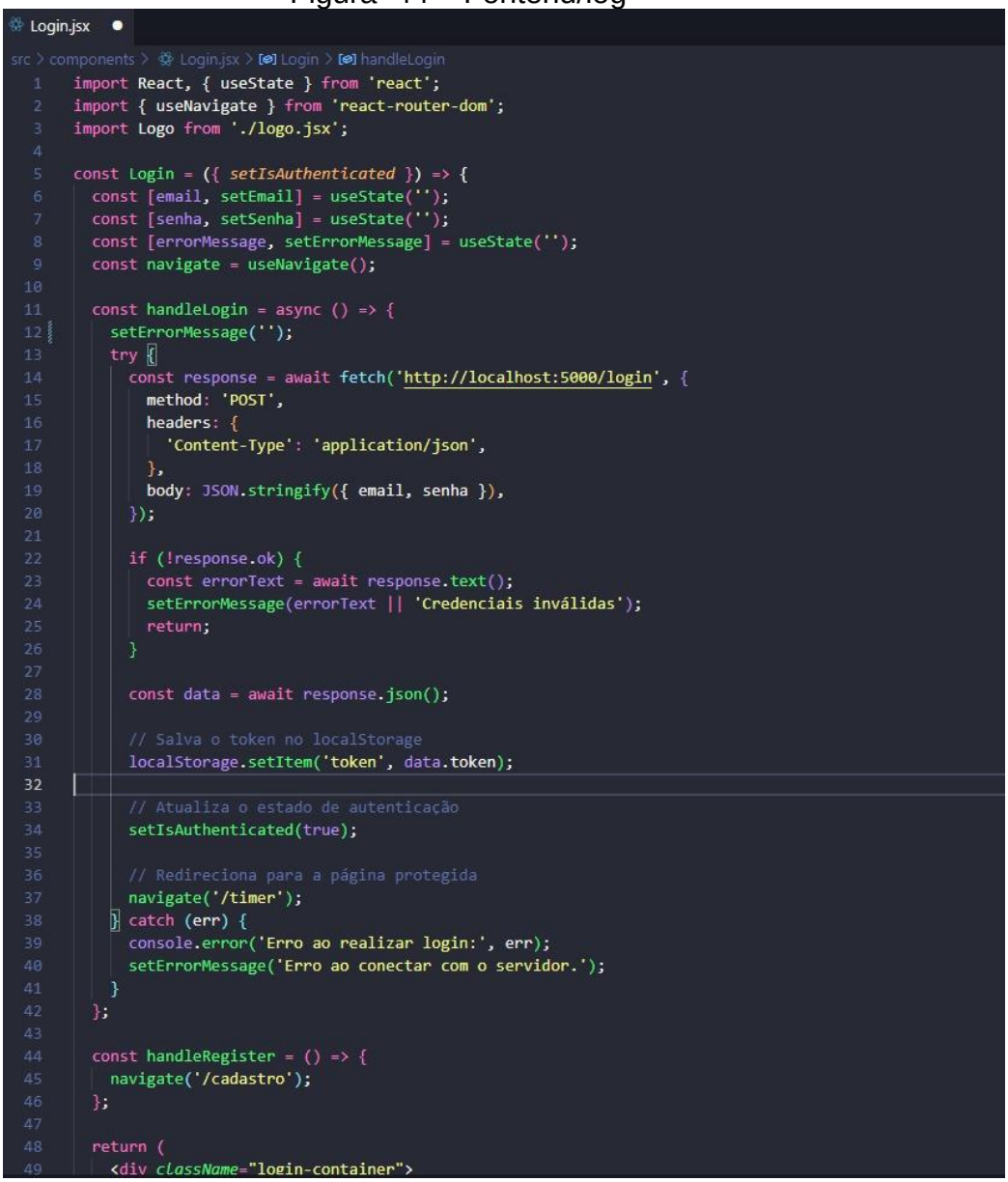
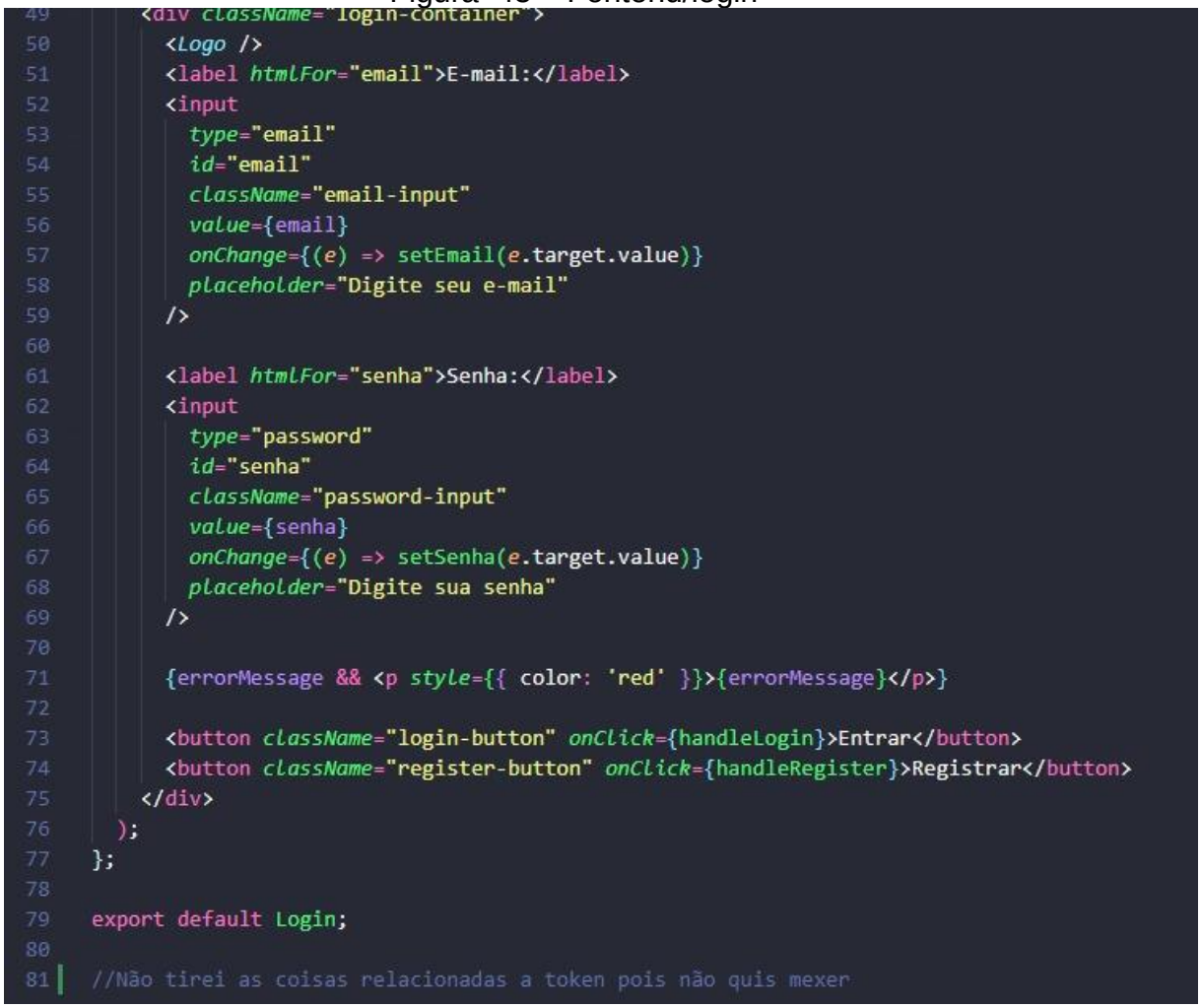


Figura 45 – Fontend/login



```
      <div className="login-container">
        <Logo />
        <label htmlFor="email">E-mail:</label>
        <input
          type="email"
          id="email"
          className="email-input"
          value={email}
          onChange={(e) => setEmail(e.target.value)}
          placeholder="Digite seu e-mail"
        />

        <label htmlFor="senha">Senha:</label>
        <input
          type="password"
          id="senha"
          className="password-input"
          value={senha}
          onChange={(e) => setSenha(e.target.value)}
          placeholder="Digite sua senha"
        />

        {errorMessage && <p style={{ color: 'red' }}>{errorMessage}</p>}

        <button className="login-button" onClick={handleLogin}>Entrar</button>
        <button className="register-button" onClick={handleRegister}>Registrar</button>
      </div>
    );
};

export default Login;

//Não tirei as coisas relacionadas a token pois não quis mexer
```

Figura 46 – Fontend/login

Login.css

src > components > # Login.css > button:disabled

```css
.login-container {
  display: flex;
  flex-direction: column;
  max-width: 400px;
  margin: 0 auto;
  padding: 20px;
  border-radius: 8px;
  text-align: left;
}
label {
  font-weight: bold;
  margin-top: 10px;
  margin-bottom: 5px;
  color: aliceblue;
}
.email-input,
.password-input {
  width: 100%;
  padding: 10px;
  margin-top: 5px;
  margin-bottom: 15px;
  border: 1px solid #ccc;
  border-radius: 5px;
}
button {
  padding: 10px 15px;
  border: none;
  border-radius: 15px;
  cursor: pointer;
  margin-top: 15px;
}
.login-button {
  background-color: white;
  color: black;
}
.register-button {
  background-color: white;
  color: black;
}
button:hover {
  opacity: 0.8;
}
button:disabled {
  background-color: #ccc;
  cursor: not-allowed;
}
```

11.5 FRONTEND/LOGO:

Figura 47 – Frontend/logo

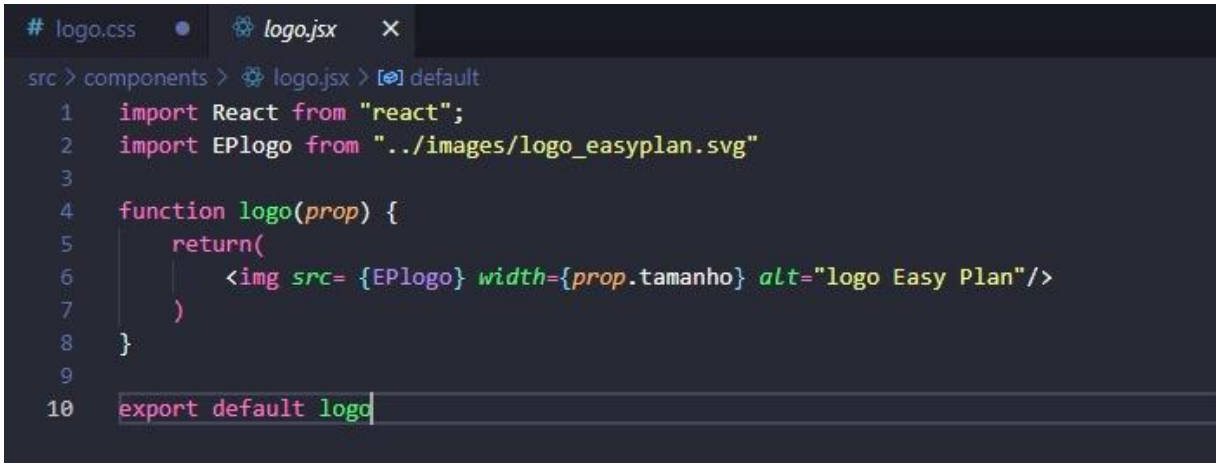
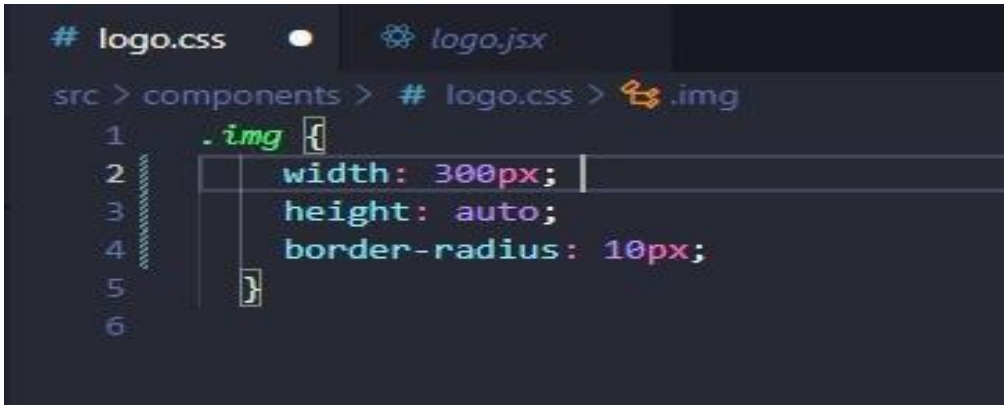
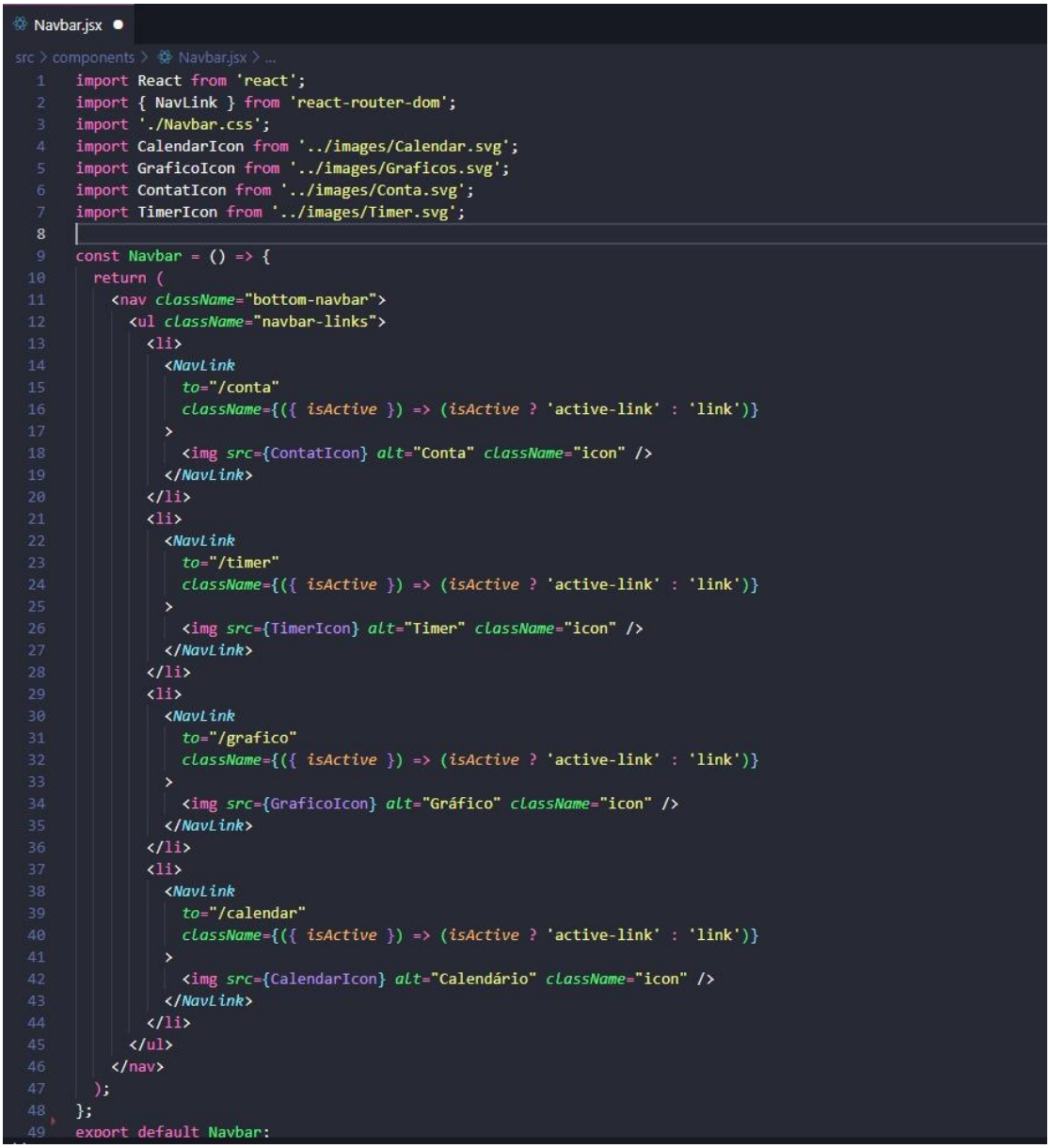


Figura 48 – Frontend/logo



11.6 FONTEND/NAVBAR:

Figura 49 - Fontend/navbar



Navbar.jsx

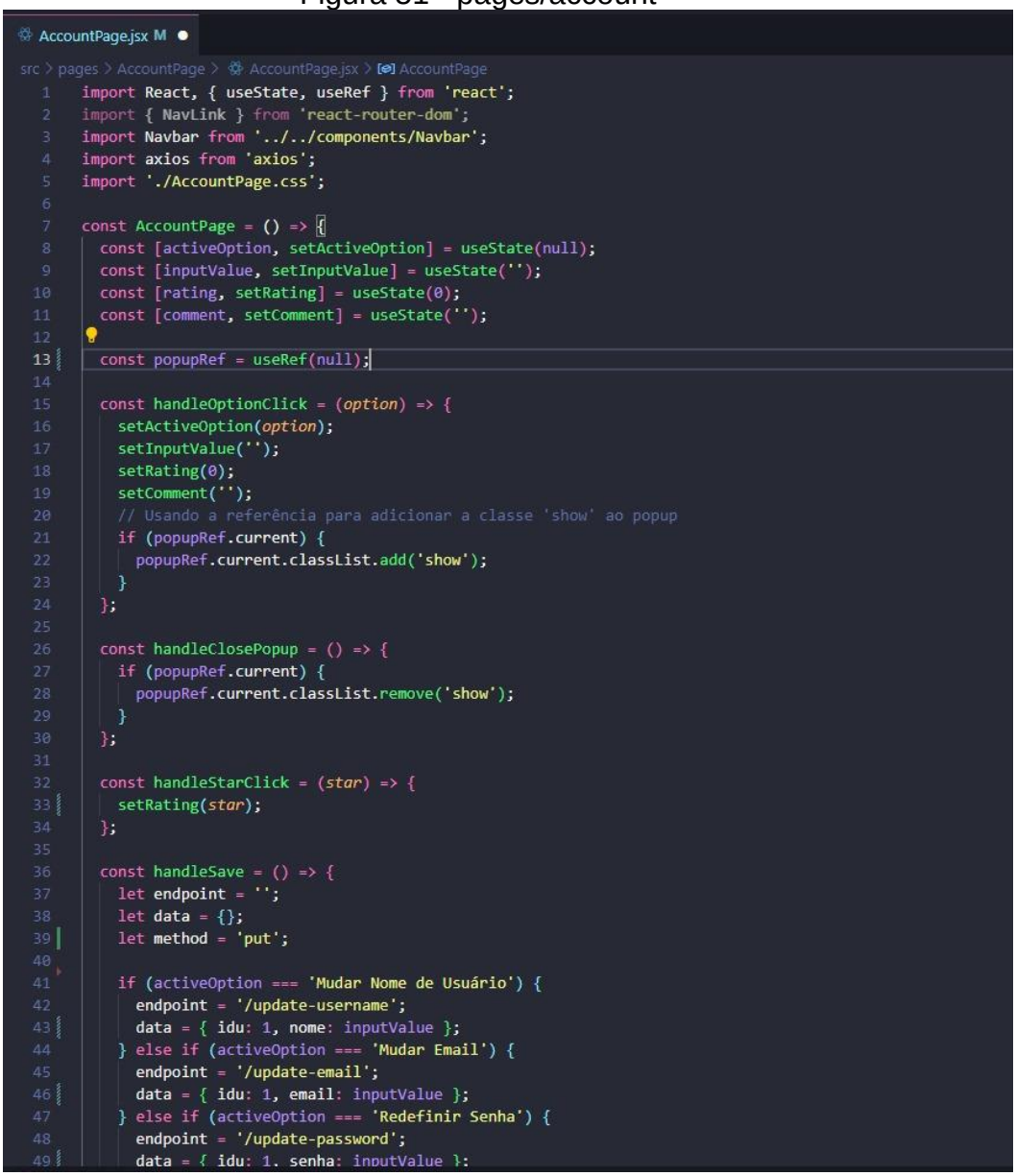
src > components > Navbar.jsx > ...

```jsx
import React from 'react';
import { NavLink } from 'react-router-dom';
import './Navbar.css';
import CalendarIcon from '../images/Calendar.svg';
import GraficoIcon from '../images/Graficos.svg';
import ContatIcon from '../images/Conta.svg';
import TimerIcon from '../images/Timer.svg';

const Navbar = () => {
  return (
    <nav className="bottom-navbar">
      <ul className="navbar-links">
        <li>
          <NavLink
            to="/conta"
            className={({ isActive }) => (isActive ? 'active-link' : 'link')}
          >
            <img src={ContatIcon} alt="Conta" className="icon" />
          </NavLink>
        </li>
        <li>
          <NavLink
            to="/timer"
            className={({ isActive }) => (isActive ? 'active-link' : 'link')}
          >
            <img src={TimerIcon} alt="Timer" className="icon" />
          </NavLink>
        </li>
        <li>
          <NavLink
            to="/grafico"
            className={({ isActive }) => (isActive ? 'active-link' : 'link')}
          >
            <img src={GraficoIcon} alt="Gráfico" className="icon" />
          </NavLink>
        </li>
        <li>
          <NavLink
            to="/calendar"
            className={({ isActive }) => (isActive ? 'active-link' : 'link')}
          >
            <img src={CalendarIcon} alt="Calendário" className="icon" />
          </NavLink>
        </li>
      </ul>
    </nav>
  );
};
export default Navbar;
```

Figura 50 - Fontend/navbar

11.7 FONTEND/PAGES/ACCOUNT:

Figura 51 - pages/account

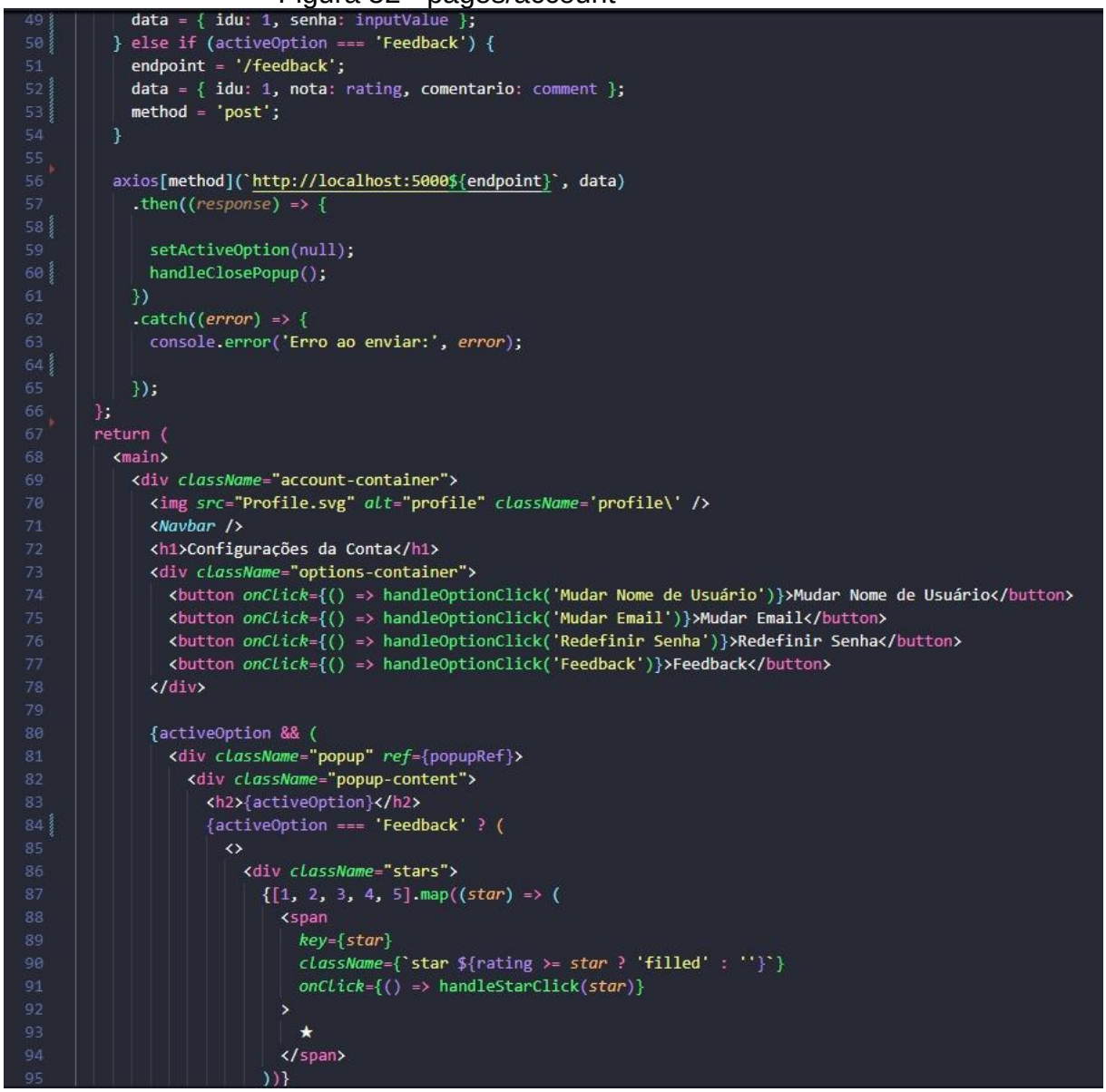


AccountPage.jsx M

src > pages > AccountPage > AccountPage.jsx > AccountPage

```jsx
import React, { useState, useRef } from 'react';
import { NavLink } from 'react-router-dom';
import Navbar from '../../components/Navbar';
import axios from 'axios';
import './AccountPage.css';

const AccountPage = () => {
  const [activeOption, setActiveOption] = useState(null);
  const [inputValue, setInputValue] = useState('');
  const [rating, setRating] = useState(0);
  const [comment, setComment] = useState('');

  const popupRef = useRef(null);

  const handleOptionClick = (option) => {
    setActiveOption(option);
    setInputValue('');
    setRating(0);
    setComment('');
    // Usando a referência para adicionar a classe 'show' ao popup
    if (popupRef.current) {
      popupRef.current.classList.add('show');
    }
  };

  const handleClosePopup = () => {
    if (popupRef.current) {
      popupRef.current.classList.remove('show');
    }
  };

  const handleStarClick = (star) => {
    setRating(star);
  };

  const handleSave = () => {
    let endpoint = '';
    let data = {};
    let method = 'put';

    if (activeOption === 'Mudar Nome de Usuário') {
      endpoint = '/update-username';
      data = { idu: 1, nome: inputValue };
    } else if (activeOption === 'Mudar Email') {
      endpoint = '/update-email';
      data = { idu: 1, email: inputValue };
    } else if (activeOption === 'Redefinir Senha') {
      endpoint = '/update-password';
      data = { idu: 1, senha: inputValue };
```

Figura 52 - pages/account



```
    data = { idu: 1, senha: inputValue };
  } else if (activeOption === 'Feedback') {
    endpoint = '/feedback';
    data = { idu: 1, nota: rating, comentario: comment };
    method = 'post';
  }

  axios[method](`http://localhost:5000${endpoint}`, data)
    .then((response) => {

      setActiveOption(null);
      handleClosePopup();
    })
    .catch((error) => {
      console.error('Erro ao enviar:', error);

    });
};
return (
  <main>
    <div className="account-container">
      <img src="Profile.svg" alt="profile" className='profile\' />
      <Navbar />
      <h1>Configurações da Conta</h1>
      <div className="options-container">
        <button onClick={() => handleOptionClick('Mudar Nome de Usuário')}>Mudar Nome de Usuário</button>
        <button onClick={() => handleOptionClick('Mudar Email')}>Mudar Email</button>
        <button onClick={() => handleOptionClick('Redefinir Senha')}>Redefinir Senha</button>
        <button onClick={() => handleOptionClick('Feedback')}>Feedback</button>
      </div>

      {activeOption && (
        <div className="popup" ref={popupRef}>
          <div className="popup-content">
            <h2>{activeOption}</h2>
            {activeOption === 'Feedback' ? (
              <>
                <div className="stars">
                  {[1, 2, 3, 4, 5].map((star) => (
                    <span
                      key={star}
                      className={`star ${rating >= star ? 'filled' : ''}`}
                      onClick={() => handleStarClick(star)}
                    >
                      ★
                    </span>
                  ))}
```

Figura 53 - pages/account

```
                    ))}
                  </div>
                  <textarea
                    placeholder="Deixe seu comentário aqui..."
                    value={comment}
                    onChange={(e) => setComment(e.target.value)}
                  />
                </>
              ) : (
                <input
                  type="text"
                  placeholder={`${activeOption.toLowerCase()}`}
                  value={inputValue}
                  onChange={(e) => setInputValue(e.target.value)}
                />
              )}
              <div className="popup-buttons">
                <button onClick={handleSave}>Salvar</button>
                <button onClick={handleClosePopup}>Cancelar</button>
              </div>
            </div>
          </div>
        )}
      </div>
    </main>
  );
};
export default AccountPage;
```

Figura 54 - pages/account

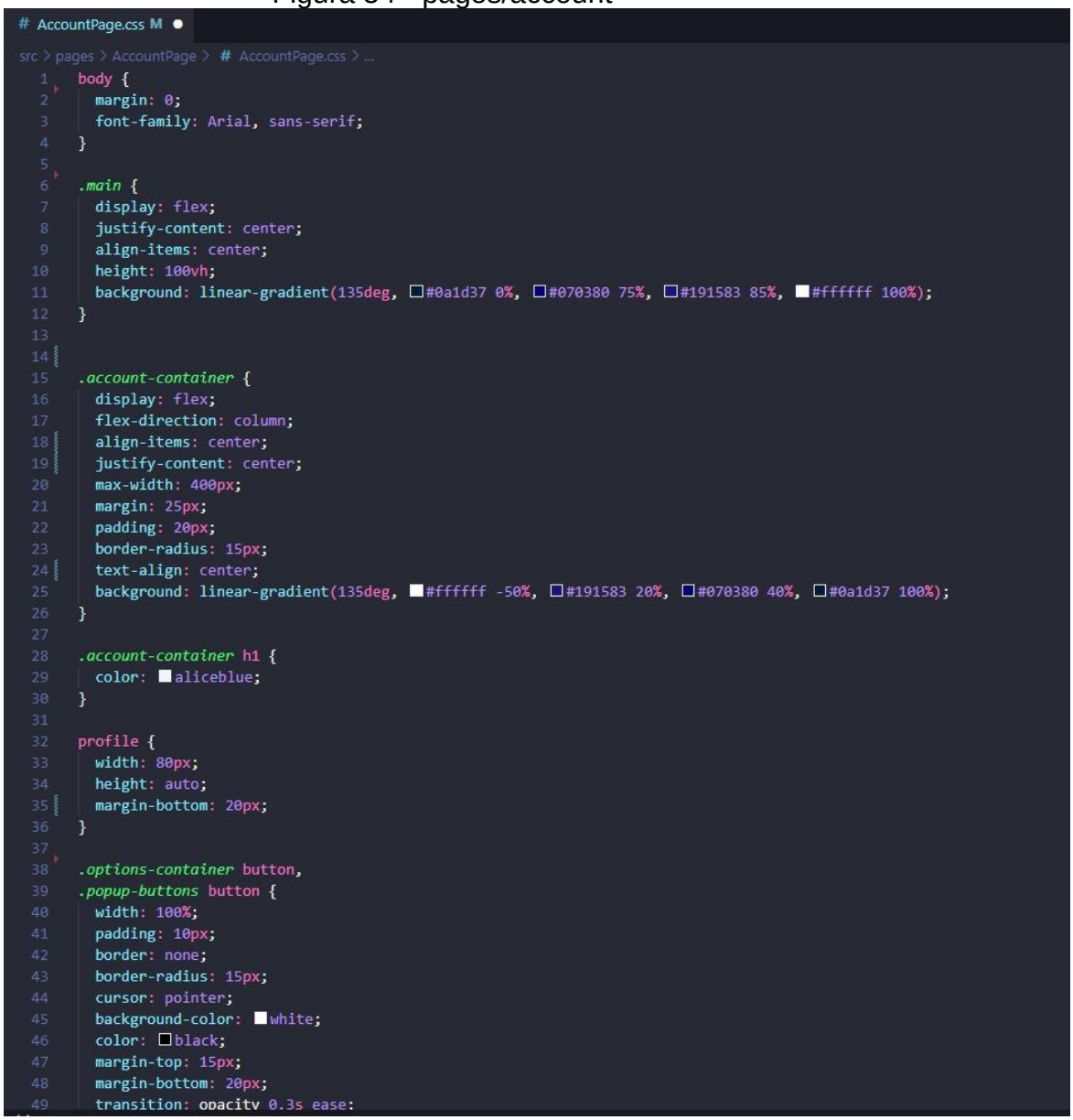
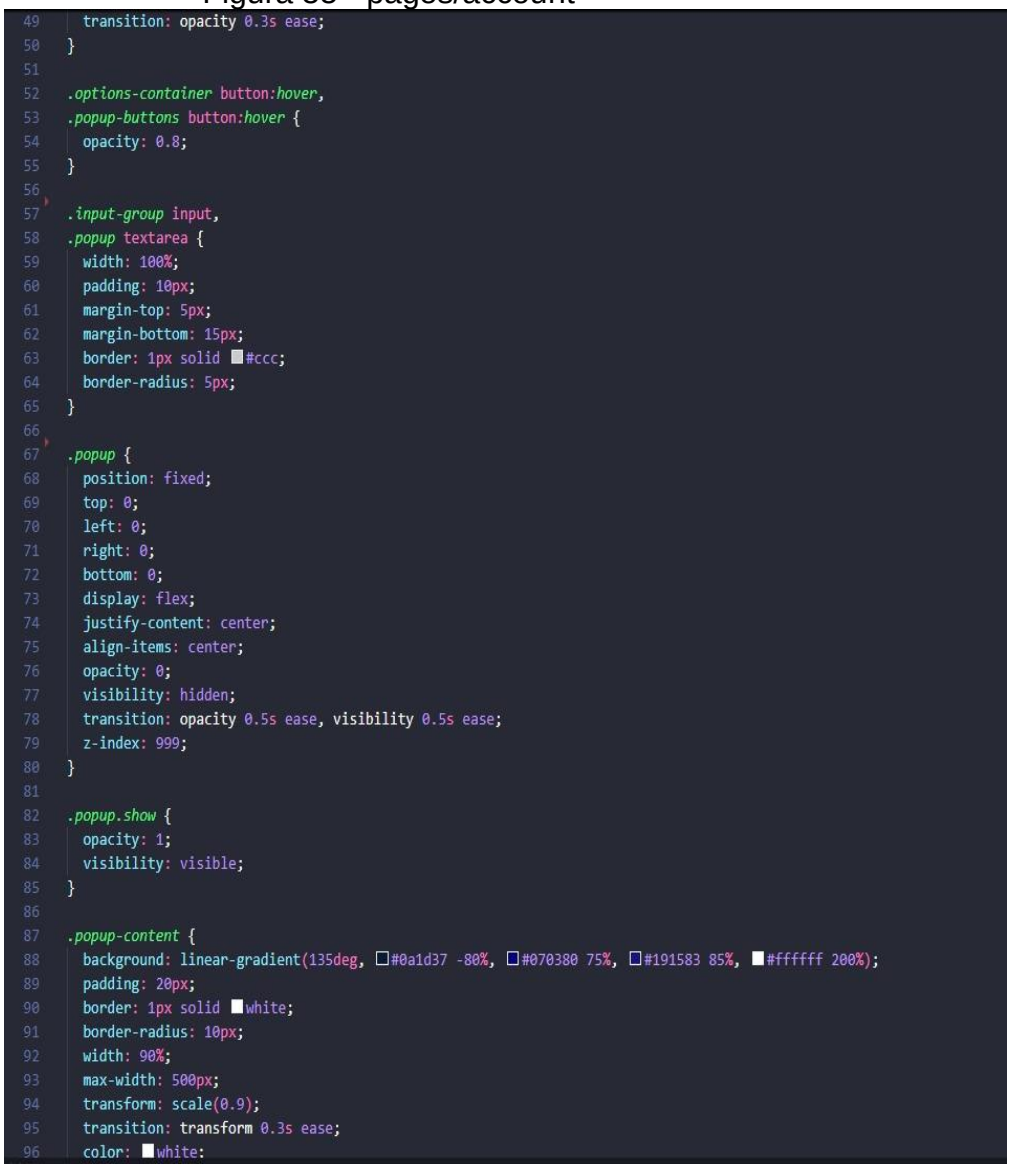


Figura 55 - pages/account



```
  transition: opacity 0.3s ease;
}

.options-container button:hover,
.popup-buttons button:hover {
  opacity: 0.8;
}

.input-group input,
.popup textarea {
  width: 100%;
  padding: 10px;
  margin-top: 5px;
  margin-bottom: 15px;
  border: 1px solid #ccc;
  border-radius: 5px;
}

.popup {
  position: fixed;
  top: 0;
  left: 0;
  right: 0;
  bottom: 0;
  display: flex;
  justify-content: center;
  align-items: center;
  opacity: 0;
  visibility: hidden;
  transition: opacity 0.5s ease, visibility 0.5s ease;
  z-index: 999;
}

.popup.show {
  opacity: 1;
  visibility: visible;
}

.popup-content {
  background: linear-gradient(135deg, #0a1d37 -80%, #070380 75%, #191583 85%, #ffffff 200%);
  padding: 20px;
  border: 1px solid white;
  border-radius: 10px;
  width: 90%;
  max-width: 500px;
  transform: scale(0.9);
  transition: transform 0.3s ease;
  color: white;
```

Figura 56 - pages/account

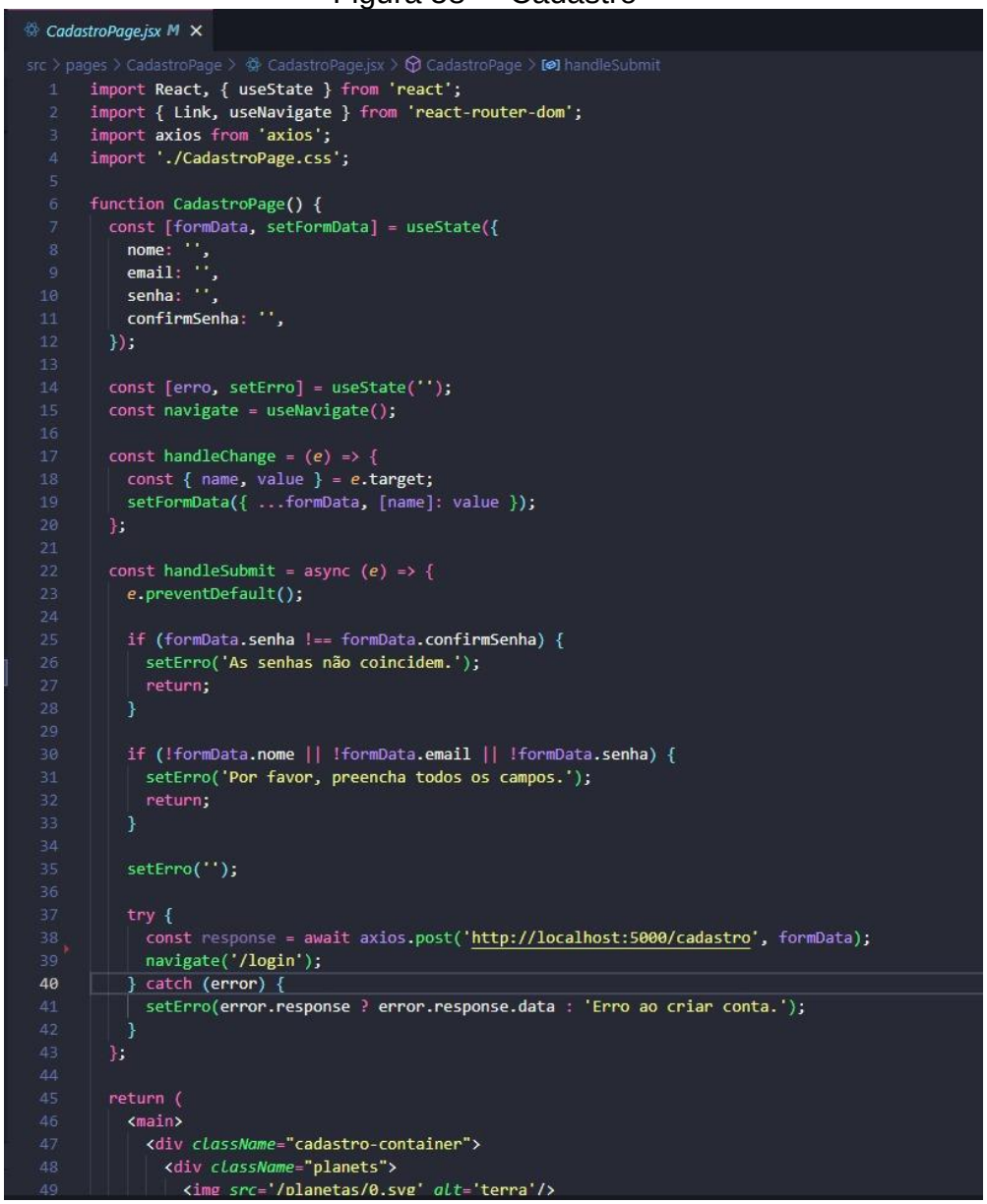
```
    color: white;
}

.popup.show .popup-content {
    transform: scale(1);
}

.popup .stars {
    text-align: center;
}

.popup .star {
    font-size: 24px;
    cursor: pointer;
    color: #ccc;
    margin: 0 5px;
}

.popup .star.filled {
    color: #ffcc00;
}

.popup-buttons button,
.popup input,
.popup textarea {
    width: 100%;
    padding: 10px;
    border: none;
    border-radius: 15px;
    cursor: pointer;
    background-color: white;
    color: black;
    margin-top: 15px;
    margin-bottom: 20px;
    transition: opacity 0.3s ease;
}

.popup-buttons button:hover,
.options-container button:hover {
    opacity: 0.8;
}

.popup-overlay {
    position: absolute;
    top: 0;
    left: 0;
    right: 0;
    bottom: 0;
```

Figura 57 - pages/account

```
    bottom: 0;
    background: rgba(0, 0, 0, 0.8);
    transition: opacity 0.5s ease;
    opacity: 0;
}

.popup.show .popup-overlay {
    opacity: 1;
}

.cadastro-error {
    color: red;
    font-size: 14px;
    margin-bottom: 10px;
    text-align: center;
}

.cadastro-link {
    color: #4CAF50;
    text-decoration: none;
    text-align: center;
    display: block;
    margin-top: 10px;
}

p {
    color: aliceblue;
}

```

11.8 FONTEND/PAGES/CADASTRO:

Figura 58 – Cadastro



```jsx
import React, { useState } from 'react';
import { Link, useNavigate } from 'react-router-dom';
import axios from 'axios';
import './CadastroPage.css';

function CadastroPage() {
  const [formData, setFormData] = useState({
    nome: '',
    email: '',
    senha: '',
    confirmSenha: '',
  });

  const [erro, setErro] = useState('');
  const navigate = useNavigate();

  const handleChange = (e) => {
    const { name, value } = e.target;
    setFormData({ ...formData, [name]: value });
  };

  const handleSubmit = async (e) => {
    e.preventDefault();

    if (formData.senha !== formData.confirmSenha) {
      setErro('As senhas não coincidem.');
      return;
    }

    if (!formData.nome || !formData.email || !formData.senha) {
      setErro('Por favor, preencha todos os campos.');
      return;
    }

    setErro('');

    try {
      const response = await axios.post('http://localhost:5000/cadastro', formData);
      navigate('/login');
    } catch (error) {
      setErro(error.response ? error.response.data : 'Erro ao criar conta.');
    }
  };

  return (
    <main>
      <div className="cadastro-container">
        <div className="planets">
          <img src='/planetas/0.svg' alt='terra'/>
```

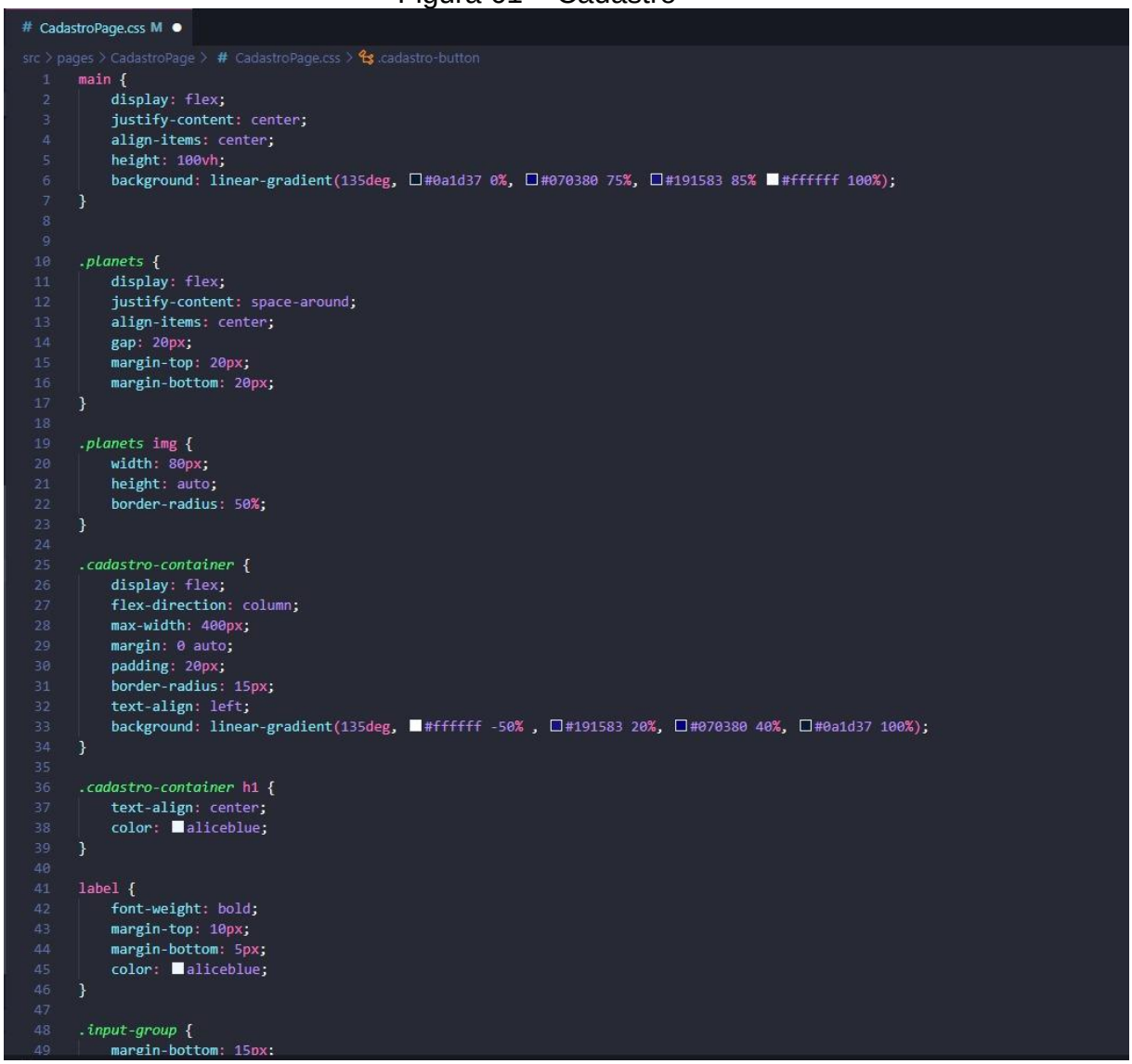
Figura 59 – Cadastro

```
          <img src='/planetas/9.svg' alt='planeta9'/>
          <img src='/planetas/1.svg' alt='planeta1'/>
        </div>
        {erro && <p className="cadastro-error">{erro}</p>}
        <form onSubmit={handleSubmit}>
          <div className="input-group">
            <label htmlFor="nome">Nome:</label>
            <input
              type="text"
              id="nome"
              name="nome"
              value={formData.nome}
              onChange={handleChange}
              className="cadastro-input"
              placeholder="Digite seu nome"
            />
          </div>
          <div className="input-group">
            <label htmlFor="email">Email:</label>
            <input
              type="email"
              id="email"
              name="email"
              value={formData.email}
              onChange={handleChange}
              className="cadastro-input"
              placeholder="Digite seu email"
            />
          </div>
          <div className="input-group">
            <label htmlFor="senha">Senha:</label>
            <input
              type="password"
              id="senha"
              name="senha"
              value={formData.senha}
              onChange={handleChange}
              className="cadastro-input"
              placeholder="Digite sua senha"
            />
          </div>
          <div className="input-group">
            <label htmlFor="confirmSenha">Confirmar Senha:</label>
            <input
              type="password"
              id="confirmSenha"
              name="confirmSenha"
```

Figura 60 – Cadastro

```
                name="confirmSenha"
                value={formData.confirmSenha}
                onChange={handleChange}
                className="cadastro-input"
                placeholder="Confirme sua senha"
              />
            </div>
            <button type="submit" className="cadastro-button">Cadastrar</button>
          </form>
          <p>
            Já tem uma conta? <Link to="/login" className="cadastro-link">Faça login</Link>
          </p>
        </div>
      </main>
    );
  }

  export default CadastroPage;
```

Figura 61 – Cadastro



CadastroPage.css M

src > pages > CadastroPage > # CadastroPage.css > .cadastro-button

```css
main {
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    background: linear-gradient(135deg, #0a1d37 0%, #070380 75%, #191583 85% #ffffff 100%);
}


.planets {
    display: flex;
    justify-content: space-around;
    align-items: center;
    gap: 20px;
    margin-top: 20px;
    margin-bottom: 20px;
}

.planets img {
    width: 80px;
    height: auto;
    border-radius: 50%;
}

.cadastro-container {
    display: flex;
    flex-direction: column;
    max-width: 400px;
    margin: 0 auto;
    padding: 20px;
    border-radius: 15px;
    text-align: left;
    background: linear-gradient(135deg, #ffffff -50% , #191583 20%, #070380 40%, #0a1d37 100%);
}

.cadastro-container h1 {
    text-align: center;
    color: aliceblue;
}

label {
    font-weight: bold;
    margin-top: 10px;
    margin-bottom: 5px;
    color: aliceblue;
}

.input-group {
    margin-bottom: 15px;
```

Figura 62 – Cadastro

```
    margin-bottom: 15px;
}

.cadastro-input {
    width: 100%;
    padding: 10px;
    margin-top: 5px;
    margin-bottom: 15px;
    border: 1px solid #ccc;
    border-radius: 5px;
}

.cadastro-button {
    width: 100%;
    padding: 10px;
    border: none;
    border-radius: 15px;
    cursor: pointer;
    background-color: white;
    color: black;
    margin-top: 15px;
    margin-bottom: 20px;
    transition: opacity 0.3s ease;
}

.cadastro-button:hover {
    opacity: 0.8;
}

.cadastro-error {
    color: red;
    font-size: 14px;
    margin-bottom: 10px;
    text-align: center;
}

.cadastro-link {
    color: #4CAF50;
    text-decoration: none;
}

.cadastro-container p{
    color: aliceblue;
}
```

11.9 FRONTEND/PAGES/CALENDARIO:`

Figura 63 – Calendário

CalendarPage.jsx M

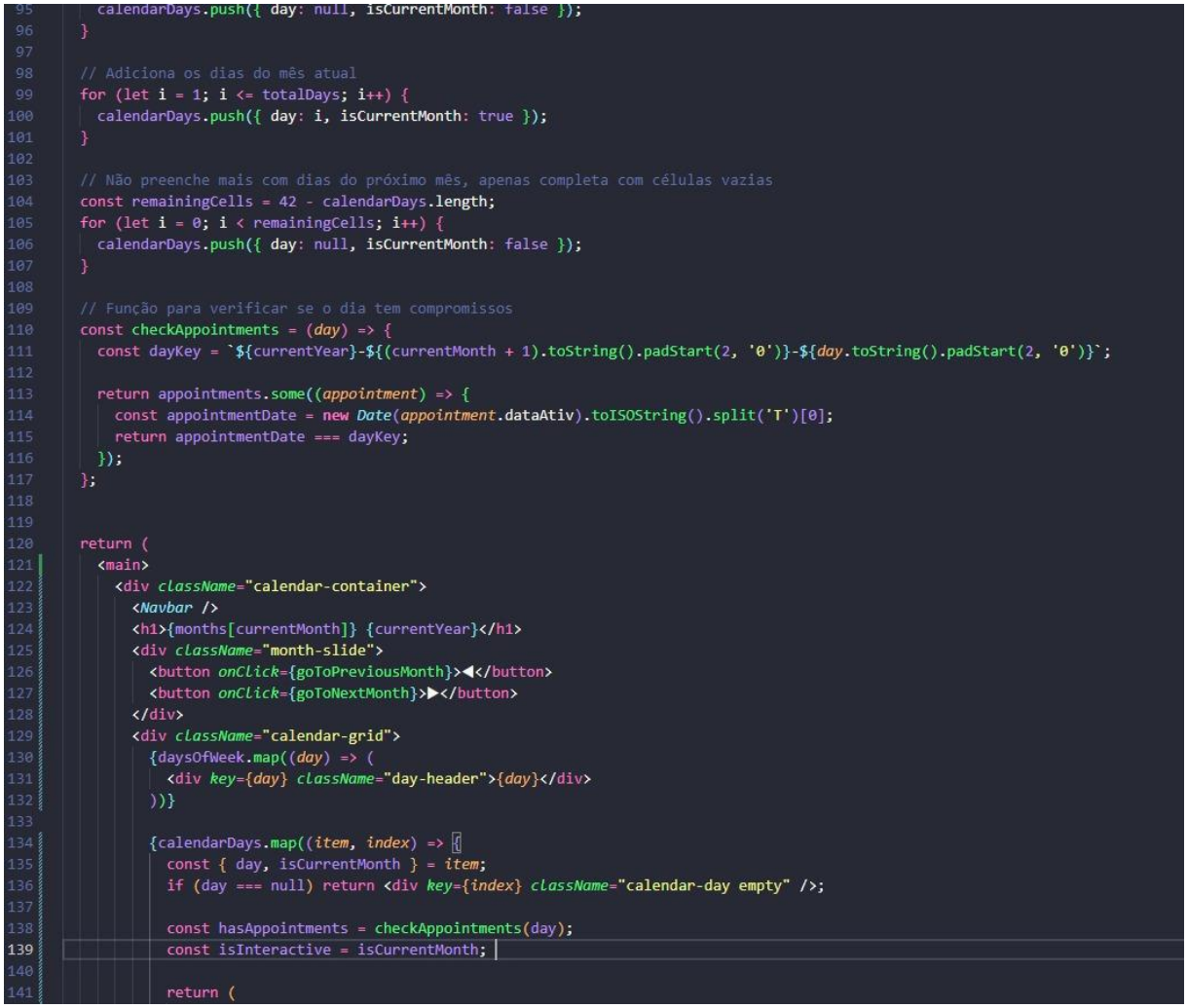
src > pages > CalendarPage > CalendarPage.jsx > CalendarPage > calendarDays.map() callback

```jsx
import React, { useState, useEffect } from 'react';
import { useNavigate } from 'react-router-dom';
import axios from 'axios';
import Navbar from '../../components/Navbar';
import './CalendarPage.css';
import Checklist from '../../components/Checklist';

const CalendarPage = () => {
  const [appointments, setAppointments] = useState([]);
  const [currentMonth, setCurrentMonth] = useState(new Date().getMonth());
  const [currentYear, setCurrentYear] = useState(new Date().getFullYear());
  const [selectedDate, setSelectedDate] = useState('');
  const [loading, setLoading] = useState(false);
  const navigate = useNavigate();

  // Hook para buscar compromissos sempre que selectedDate mudar
  useEffect(() => {
    const fetchMonthlyAppointments = async () => {
      setLoading(true);
      try {
        const monthKey = `${currentYear}-${(currentMonth + 1).toString().padStart(2, '0')}`;
        const response = await axios.get(`/api/calendario/month/${monthKey}`);
        setAppointments(response.data);
      } catch (error) {
        console.error("Erro ao buscar compromissos mensais:", error);
      } finally {
        setLoading(false);
      }
    };

    fetchMonthlyAppointments();
  }, [currentMonth, currentYear]);


  const months = [
    'Janeiro', 'Fevereiro', 'Março', 'Abril', 'Maio', 'Junho',
    'Julho', 'Agosto', 'Setembro', 'Outubro', 'Novembro', 'Dezembro',
  ];

  const daysOfWeek = ['Dom', 'Seg', 'Ter', 'Qua', 'Qui', 'Sex', 'Sáb'];

  const daysInMonth = (month, year) => new Date(year, month + 1, 0).getDate();
  const firstDayOfMonth = (month, year) => new Date(year, month, 1).getDay();

  const lastDayOfMonth = (month, year) => new Date(year, month + 1, 0).getDay();

  const getNextMonthStartDay = (month, year) => {
    const lastDay = new Date(year, month + 1, 0).getDay();
    return (lastDay + 1) % 7;
```

Figura 64 – Calendário

```
    return (lastDay + 1) % 7;
  };

  const handleDayClick = (day) => {
    if (day) {
      const dateString = `${currentYear}-${(currentMonth + 1).toString().padStart(2, '0')}-${day.toString().padStart(2, '0')}`;
      setSelectedDate(dateString);
      navigate(`/day/${currentMonth + 1}/${day}`);
    }
  };

  const goToNextMonth = () => {
    let newMonth = currentMonth + 1;
    let newYear = currentYear;

    if (newMonth > 11) {
      newMonth = 0;
      newYear++;
    }

    setCurrentMonth(newMonth);
    setCurrentYear(newYear);
  };

  const goToPreviousMonth = () => {
    let newMonth = currentMonth - 1;
    let newYear = currentYear;

    if (newMonth < 0) {
      newMonth = 11;
      newYear--;
    }

    setCurrentMonth(newMonth);
    setCurrentYear(newYear);
  };

  const totalDays = daysInMonth(currentMonth, currentYear);
  const startDay = firstDayOfMonth(currentMonth, currentYear);

  const nextMonthStartDay = getNextMonthStartDay(currentMonth, currentYear);

  const calendarDays = [];

  // Adiciona os dias em branco antes do primeiro dia do mês
  for (let i = 0; i < startDay; i++) {
    calendarDays.push({ day: null, isCurrentMonth: false });
```

Figura 65 – Calendário



```
    calendarDays.push({ day: null, isCurrentMonth: false });
  }

  // Adiciona os dias do mês atual
  for (let i = 1; i <= totalDays; i++) {
    calendarDays.push({ day: i, isCurrentMonth: true });
  }

  // Não preenche mais com dias do próximo mês, apenas completa com células vazias
  const remainingCells = 42 - calendarDays.length;
  for (let i = 0; i < remainingCells; i++) {
    calendarDays.push({ day: null, isCurrentMonth: false });
  }

  // Função para verificar se o dia tem compromissos
  const checkAppointments = (day) => {
    const dayKey = `${currentYear}-${(currentMonth + 1).toString().padStart(2, '0')}-${day.toString().padStart(2, '0')}`;

    return appointments.some((appointment) => {
      const appointmentDate = new Date(appointment.dataAtiv).toISOString().split('T')[0];
      return appointmentDate === dayKey;
    });
  };


  return (
    <main>
      <div className="calendar-container">
        <Navbar />
        <h1>{months[currentMonth]} {currentYear}</h1>
        <div className="month-slide">
          <button onClick={goToPreviousMonth}>◀</button>
          <button onClick={goToNextMonth}>▶</button>
        </div>
        <div className="calendar-grid">
          {daysOfWeek.map((day) => (
            <div key={day} className="day-header">{day}</div>
          ))}

          {calendarDays.map((item, index) => {
            const { day, isCurrentMonth } = item;
            if (day === null) return <div key={index} className="calendar-day empty" />;

            const hasAppointments = checkAppointments(day);
            const isInteractive = isCurrentMonth;

            return (
```

Figura 66 – Calendário

```
                return (
                  <div
                    key={index}
                    className={`calendar-day ${hasAppointments ? 'has-appointments' : ''} ${!isCurrentMonth ? 'not-current-month' : ''}`}
                    onClick={isInteractive ? () => handleDayClick(day) : null}
                  >
                    {isCurrentMonth ? day : ''}
                  </div>
                );
              })}
            </div>
            <Checklist/>
            <br/>
            <br/>
            <br/>
            <br/>
            <br/>
            <br/>
          </div>
        </main>
    );
};

export default CalendarPage;

```

Figura 67 – Calendário

CalendarPage.css M

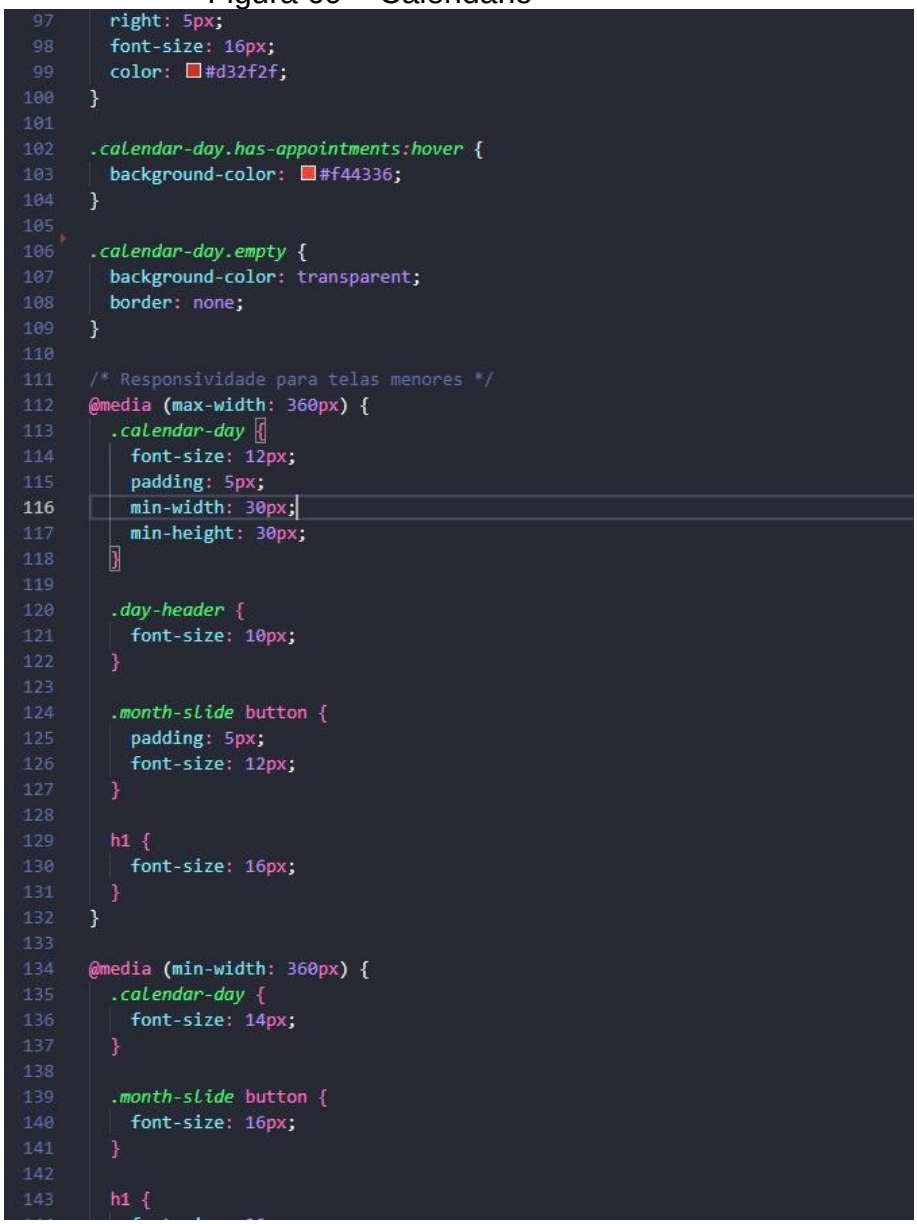
src > pages > CalendarPage > # CalendarPage.css > .calendar-grid

```css
* {
  margin: 0;
  padding-bottom: 0px;
  box-sizing: border-box;
}

body {
  font-family: Arial, sans-serif;
  background-color: #f9f9f9;
  color: #333;
}

.calendar-container {
  padding: 0px;
  text-align: center;
  width: 100%;
}

h1 {
  font-size: 22px;
  margin: 20px 0 20px 0;
  font-weight: bold;
  color: white;
}

.month-slide button {
  margin: 0px 40px 0 40px ;
  padding: 10px;
  font-size: 14px;
  cursor: pointer;
  border: none;
  background-color: #4caf4f00;
  color: white;
  border-radius: 5px;
}

.calendar-grid {
  display: grid;
  grid-template-columns: repeat(7, 1fr);
  gap: 1px;
  margin-top: 20px;
  margin-left: 8px;
  margin-right: 8px;
  border: 1px solid #ccc;
  border-radius: 10px;
  grid-template-rows: repeat(6, auto);
  padding: 8px;
}
```

Figura 68 – Calendário

```
.day-header {
  font-weight: bold;
  text-align: center;
  color: #fff;
  padding: 5px;
  font-size: 14px;
}

.calendar-day {
  cursor: pointer;
  padding: 8px;
  text-align: center;
  border: 1px solid #ddd;
  background-color: #fff;
  border-radius: 5px;
  color: #333;
  transition: background-color 0.3s ease, color 0.3s ease;
  display: flex;
  justify-content: center;
  align-items: center;
  font-size: 14px;
  min-width: 40px;
  min-height: 40px;
}

.calendar-day:hover {
  background-color: #e0e0e0;
  color: #4CAF50;
}

.calendar-day.not-current-month {
  background-color: #d3d3d3;
  cursor: not-allowed;
  color: #ccc;
}

.calendar-day.has-appointments {
  background-color: #ff6347;
  color: white;
  font-weight: bold;
  position: relative;
}

.calendar-day.has-appointments::after {
  content: '•';
  position: absolute;
  bottom: 5px;
  right: 5px;
```

Figura 69 – Calendário



```
  right: 5px;
  font-size: 16px;
  color: #d32f2f;
}

.calendar-day.has-appointments:hover {
  background-color: #f44336;
}

.calendar-day.empty {
  background-color: transparent;
  border: none;
}

/* Responsividade para telas menores */
@media (max-width: 360px) {
  .calendar-day {
    font-size: 12px;
    padding: 5px;
    min-width: 30px;
    min-height: 30px;
  }

  .day-header {
    font-size: 10px;
  }

  .month-slide button {
    padding: 5px;
    font-size: 12px;
  }

  h1 {
    font-size: 16px;
  }
}

@media (min-width: 360px) {
  .calendar-day {
    font-size: 14px;
  }

  .month-slide button {
    font-size: 16px;
  }

  h1 {
```

Figura 70 – Calendário

```
    h1 {
      font-size: 20px;
    }
  }

```

11.10 FRONTEND/PAGES/DAY:

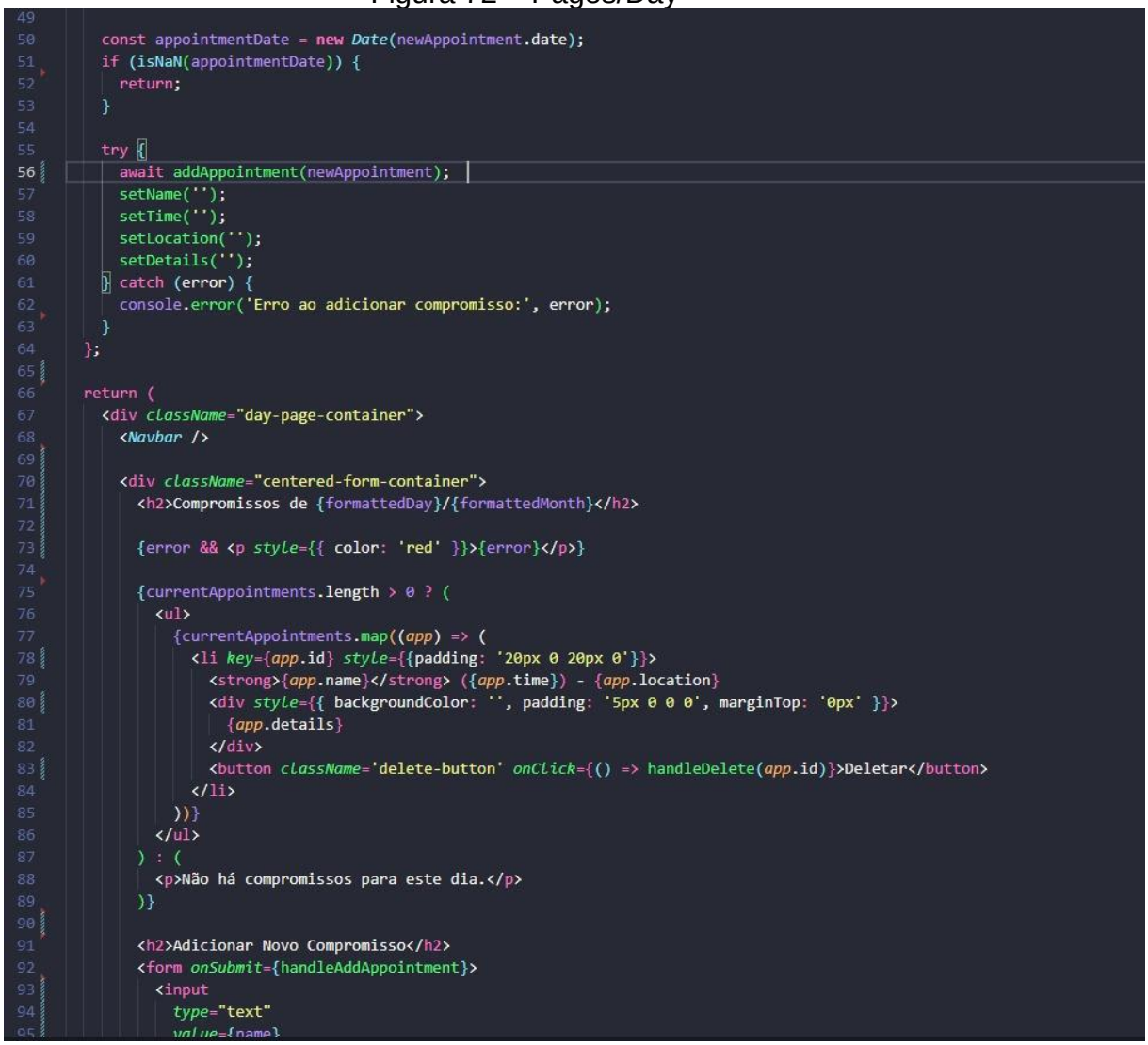
Figura 71 – Pages/Day

DayPage.jsx M

src > pages > DayPage > DayPage.jsx > DayPage > handleAddAppointment

```jsx
import React, { useState, useEffect } from 'react';
import { useParams } from 'react-router-dom';
import { useAppointments } from '../../context/AppointmentsContext';
import './DayPage.css';
import Navbar from '../../components/Navbar';

const DayPage = () => {
  const { month, day } = useParams();
  const { appointments, fetchAppointments, addAppointment, deleteAppointment, loading, error } = useAppointments();

  const [name, setName] = useState('');
  const [time, setTime] = useState('');
  const [location, setLocation] = useState('');
  const [details, setDetails] = useState('');


  const formattedMonth = month?.padStart(2, '0');
  const formattedDay = day?.padStart(2, '0');
  const dateKey = `${new Date().getFullYear()}-${formattedMonth}-${formattedDay}`;

  useEffect(() => {
    if (!appointments.length) {
      fetchAppointments(dateKey);
    }
  }, [dateKey, fetchAppointments, appointments.length]);

  // Filtra os compromissos com base na data correta
  const currentAppointments = appointments.filter((app) => {
    const appointmentDateString = new Date(app.dataAtiv).toISOString().split('T')[0];
    return appointmentDateString === dateKey;
  });

  const handleDelete = (id) => {
    if (id) {
      deleteAppointment(id);
    } else {
      console.error('ID não fornecido');
    }
  };

  const handleAddAppointment = async (e) => {
    e.preventDefault();

    if (!name || !time || !location) {
      return;
    }

    const newAppointment = { name, time, location, details, date: dateKey };
```

Figura 72 – Pages/Day



```
49
50      const appointmentDate = new Date(newAppointment.date);
51      if (isNaN(appointmentDate)) {
52        return;
53      }
54
55      try {
56        await addAppointment(newAppointment);
57        setName('');
58        setTime('');
59        setLocation('');
60        setDetails('');
61      } catch (error) {
62        console.error('Erro ao adicionar compromisso:', error);
63      }
64    };
65
66    return (
67      <div className="day-page-container">
68        <Navbar />
69
70        <div className="centered-form-container">
71          <h2>Compromissos de {formattedDay}/{formattedMonth}</h2>
72
73          {error && <p style={{ color: 'red' }}>{error}</p>}
74
75          {currentAppointments.length > 0 ? (
76            <ul>
77              {currentAppointments.map((app) => (
78                <li key={app.id} style={{padding: '20px 0 20px 0'}}>
79                  <strong>{app.name}</strong> ({app.time}) - {app.location}
80                  <div style={{ backgroundColor: '', padding: '5px 0 0 0', marginTop: '0px' }}>
81                    {app.details}
82                  </div>
83                  <button className='delete-button' onClick={() => handleDelete(app.id)}>Deletar</button>
84                </li>
85              ))}
86            </ul>
87          ) : (
88            <p>Não há compromissos para este dia.</p>
89          )}
90
91          <h2>Adicionar Novo Compromisso</h2>
92          <form onSubmit={handleAddAppointment}>
93            <input
94              type="text"
95              value={name}
```

Figura 73 – Pages/Day

```
            value={name}
            onChange={(e) => setName(e.target.value)}
            placeholder="Nome do compromisso"
            required
          />
          <input
            type="time"
            value={time}
            onChange={(e) => setTime(e.target.value)}
            required
          />
          <input
            type="text"
            value={location}
            onChange={(e) => setLocation(e.target.value)}
            placeholder="Local do compromisso"
            required
          />
          <textarea
            value={details}
            onChange={(e) => setDetails(e.target.value)}
            placeholder="Detalhes do compromisso"
          />
          <button className='add-button' type="submit">Adicionar Compromisso</button>
        </form>
      </div>
    </div>
  );
};

export default DayPage;
```

Figura 74 – Pages/Day

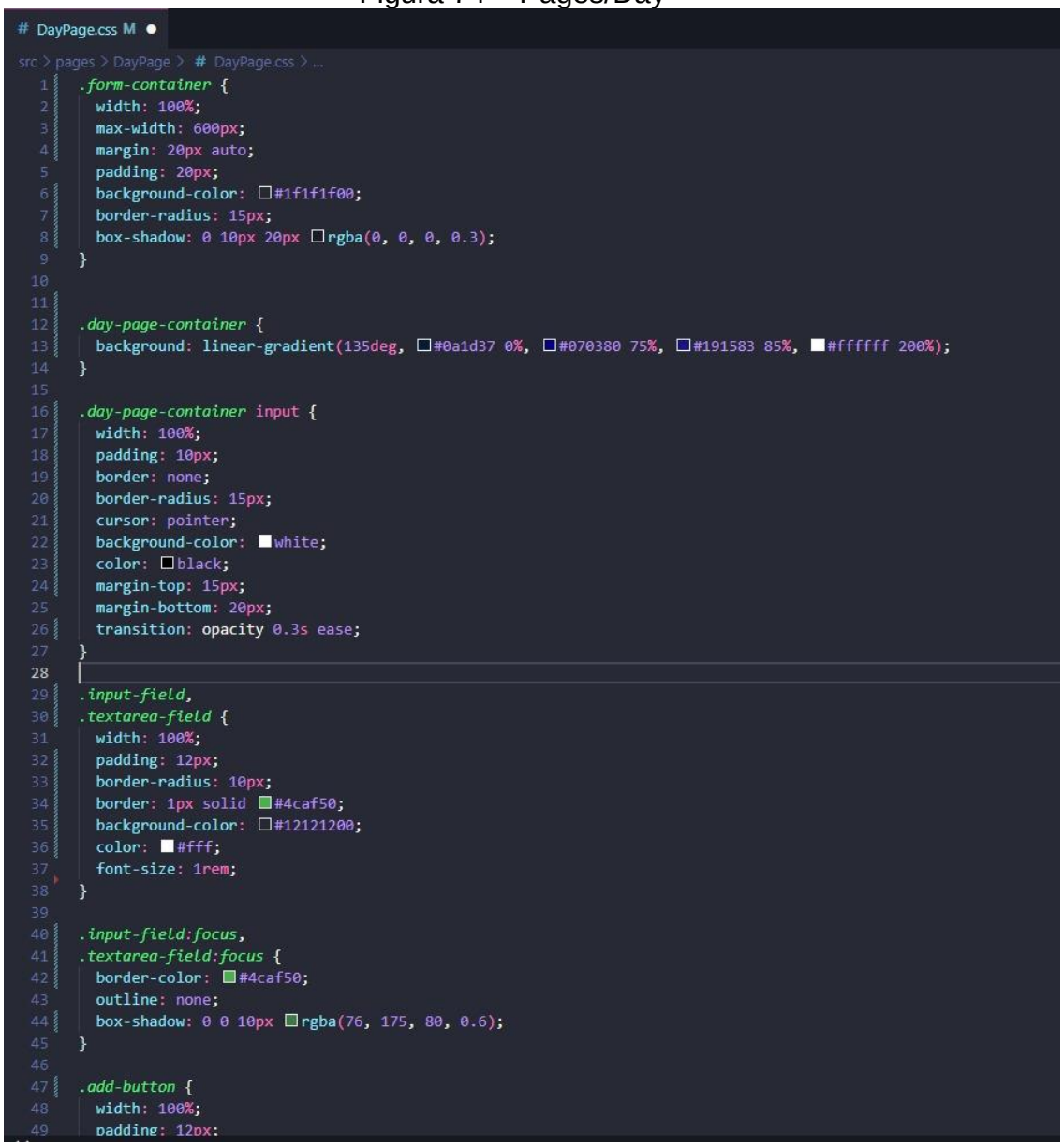


Figura 75 – Pages/Day

```
    padding: 12px;
    background-color: #4caf50;
    color: white;
    border: none;
    border-radius: 12px;
    font-size: 1.2rem;
    cursor: pointer;
    transition: background-color 0.3s;
}

.add-button:hover {
    background-color: #45a049;
}

.delete-button:hover{
    background-color: #ff0000;
}

.form-container h2 {
    text-align: center;
    color: #4caf50;
    margin-bottom: 20px;
    color: #ffffff;
}

.input-group {
    margin-bottom: 20px;
}

.input-group label {
    display: block;
    font-size: 1rem;
    color: #fff;
    margin-bottom: 8px;
}

.day-page-container p{
    margin: 10px 0 10px 0;
    color: #ffffff;
}

.day-page-container {
    display: flex;
    flex-direction: column;
    color: #ffffff;
    justify-content: center;
    align-items: center;
    min-height: 100vh;
```

Figura 76 – Pages/Day

```
    min-height: 100vh;
    padding: 20px;
    background-color: #12121200;
}

.centered-form-container {
    width: 100%;
    max-width: 600px;
    background-color: #1f1f1f00;
    padding: 20px;
    border-radius: 15px;
    border: 1px solid white;
    box-shadow: 0 10px 20px rgba(0, 0, 0, 0.3);
    text-align: center;
}

.centered-form-container form {
    display: flex;
    flex-direction: column;
    gap: 15px;
}

.centered-form-container input,
.centered-form-container textarea,
.centered-form-container button {
    width: 100%;
    padding: 12px;
    border-radius: 10px;
    border: 1px solid #4caf50;
    background-color: #ffffff;
    color: #000000;
    font-size: 1rem;
}

.centered-form-container input:focus,
.centered-form-container textarea:focus {
    border-color: #0c6deb;
    outline: none;
    box-shadow: 0 0 10px rgba(12, 94, 187, 0.6);
}

.centered-form-container button {
    background-color: #4caf4f00;
    color: white;
    border: 1px solid white;
    cursor: pointer;
    font-size: 1.2rem;
    transition: background-color 0.3s;
```

11.11 FRONTEND/PAGES/GRAFICO:

Figura 77 – Grafico

GraficoPage.jsx M

src > pages > GraficoPage > GraficoPage.jsx > GraficoPage

```
import React, { useState, useEffect } from 'react';
import { Line } from 'react-chartjs-2';
import { Chart as ChartJS, CategoryScale, LinearScale, PointElement, LineElement, Title, Tooltip, Legend } from 'chart.js';
import Navbar from '../../components/Navbar';
import { format } from 'date-fns';
import './GraficoPage.css';

ChartJS.register(CategoryScale, LinearScale, PointElement, LineElement, Title, Tooltip, Legend);

const GraficoPage = () => {
  const [graficoData, setGraficoData] = useState({ days: [], timeData: [] });
  const [tempoData, setTempoData] = useState({ totalTempo: 0, tempoHoje: 0, tempoOntem: 0 });

  useEffect(() => {
    fetch('http://localhost:5000/api/grafico')
      .then(response => response.json())
      .then(data => {
        const formattedDays = data.days.map(day => format(new Date(day), 'dd/MM'));
        setGraficoData({ days: formattedDays, timeData: data.timeData });
      })
      .catch(error => console.error('Erro ao buscar dados para o gráfico:', error));

    fetch('http://localhost:5000/api/tempo')
      .then(response => response.json())
      .then(data => setTempoData(data))
      .catch(error => console.error('Erro ao buscar dados de tempo:', error));
  }, []);


  const formatarTempo = (minutos) => {
    const horas = Math.floor(minutos / 60);
    const mins = minutos % 60;
    return horas > 0 ? `${horas} horas` : `${mins} minutos`;
  };

  return (
    <div className="grafico-page">
      <Navbar />
      <h2>Gráfico dos Últimos 7 Dias</h2>

      {/* Gráfico de Linha */}
      <div className="grafico-container">
        <Line
          data={{
            labels: graficoData.days,
            datasets: [{
              label: 'Tempo total de Timer (em minutos)',
              data: graficoData.timeData,
              borderColor: '#3498db',
```

Figura 78 – Grafico

```
                borderColor: '#3498db',
                backgroundColor: 'rgba(52, 152, 219, 0.2)',
                tension: 0.4,
                pointBackgroundColor: '#3498db',
                pointBorderColor: '#fff',
                pointBorderWidth: 2,
              }],
            }}
            options={{
              responsive: true,
              scales: {
                x: {
                  title: { display: true, text: 'Data' },
                  grid: { color: '#ddd' }
                },
                y: {
                  title: { display: true, text: 'Tempo (minutos)' },
                  min: 0,
                  grid: { color: '#ddd' },
                },
              },
              plugins: {
                tooltip: {
                  backgroundColor: '#333',
                  titleColor: '#fff',
                  bodyColor: '#fff',
                },
              },
            }}
          />
        </div>
        <div className="tempo-containers">
          <div className="tempo-box">
            <h3>Tempo Total</h3>
            <p>{formatarTempo(tempoData.totalTempo)}</p>
          </div>
          <div className="tempo-box">
            <h3>Tempo Hoje</h3>
            <p>{formatarTempo(tempoData.tempoHoje)}</p>
          </div>
          <div className="tempo-box">
            <h3>Tempo Ontem</h3>
            <p>{formatarTempo(tempoData.tempoOntem)}</p>
          </div>
        </div>
      </div>
```

Figura 79 – Grafico

```
            <p>{formatarTempo(tempoData.tempoOntem)}</p>
          </div>
        </div>
      </div>
    );
};

export default GraficoPage;

```

Figura 80 – Gráfico

GraficoPage.css M

src > pages > GraficoPage > # GraficoPage.css > {} @media (max-width: 768px)

```
.grafico-page{
  background: linear-gradient(135deg, #0a1d37 0%, #070380 75%, #191583 85%, #ffffff 200%);
}

.grafico-container {
  width: 70%;
  height: 600px;
  margin: 0 auto;
}

.tempo-containers {
  display: flex;
  justify-content: space-around;
  margin-top: 20px;
  color: black;
  gap: 20px;
  justify-content: space-evenly;
  padding-bottom: 40px;
}

.tempo-box {
  background-color: #fff;
  color: black;
  padding: 20px;
  border-radius: 12px;
  text-align: center;
  box-shadow: 0 2px 12px rgba(0, 0, 0);
  width: 180px;
  height: 250px;
  display: flex;
  flex-direction: column;
  justify-content: center;
  margin: 0px;
  transition: transform 0.3s ease;
}

.grafico-page h2 {
  color: white;
  text-align: center;
  margin-top: 40px;
  margin-bottom: 40px;
}

@media (max-width: 480px) {
  .grafico-page h2{
    margin-left:10px ;
    margin-right: 10px;
  }

```

Figura 81 – Grafico

```
.grafico-page{
  padding-bottom: 250px;
}
.grafico-container {
  width: 100%;
  height: 360px;
  padding-top: 20px;
  margin-bottom: 10px;
}

.tempo-containers {
  flex-direction: row;
  align-items: center;
  justify-content: space-between;
  height: auto;
  gap: 15px;
  margin-left: 10px;
  margin-right: 10px;
  margin-top: -250px;
}

.tempo-box {
  width: 30%;
  height: 150px;
  margin-bottom: 0;
}
}

@media (max-width: 768px) {
.grafico-container {
  width: 90%;
  height: 450px;
}

.tempo-box {
  width: 150px;
}
}
p{
color: black;
}
```

11.12 FRONTEND/PAGES/TIMER:

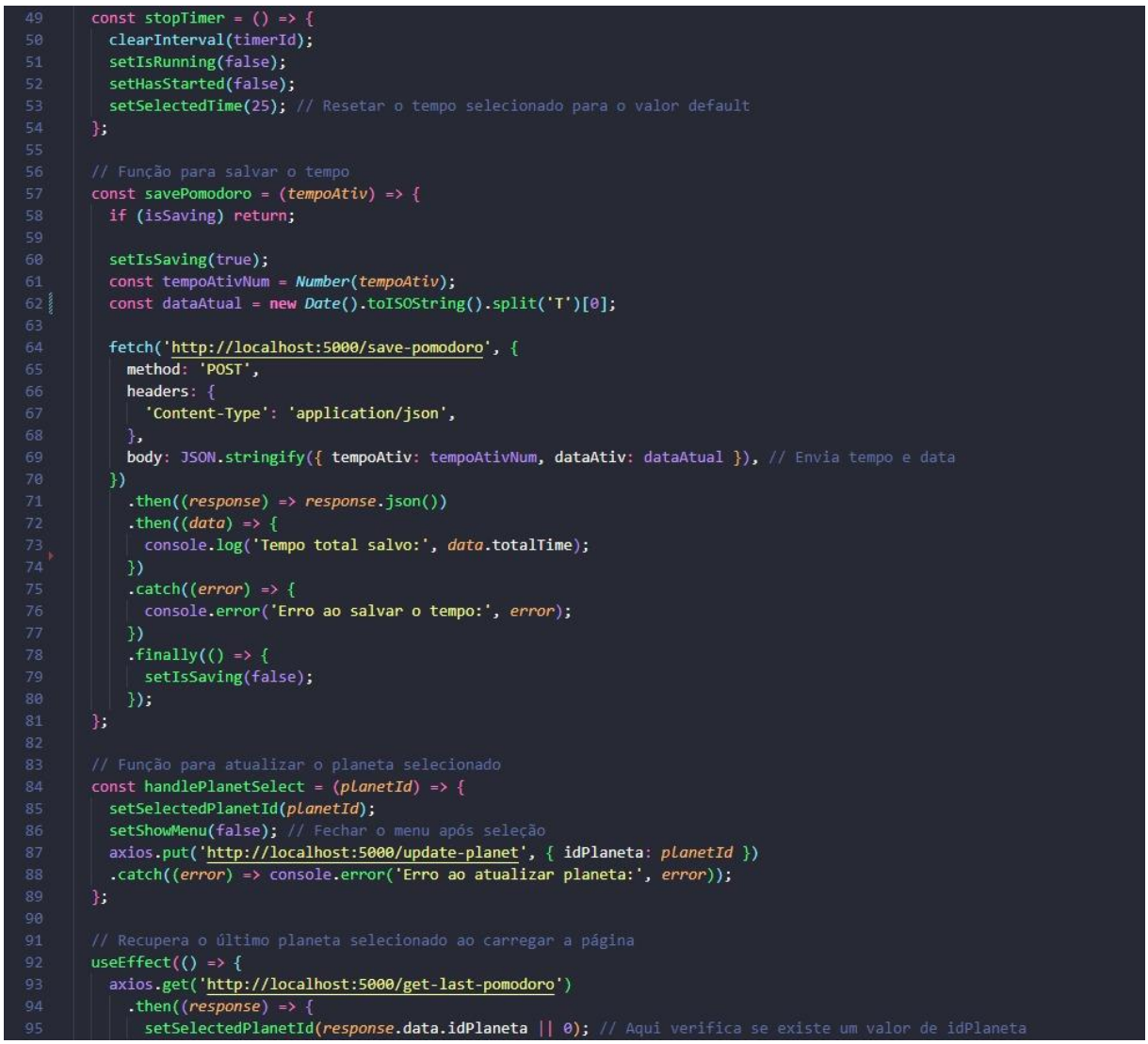
Figura 82 – Timer

TimerPage.jsx M

src > pages > TimerPage > TimerPage.jsx > TimerPage > savePomodoro > headers

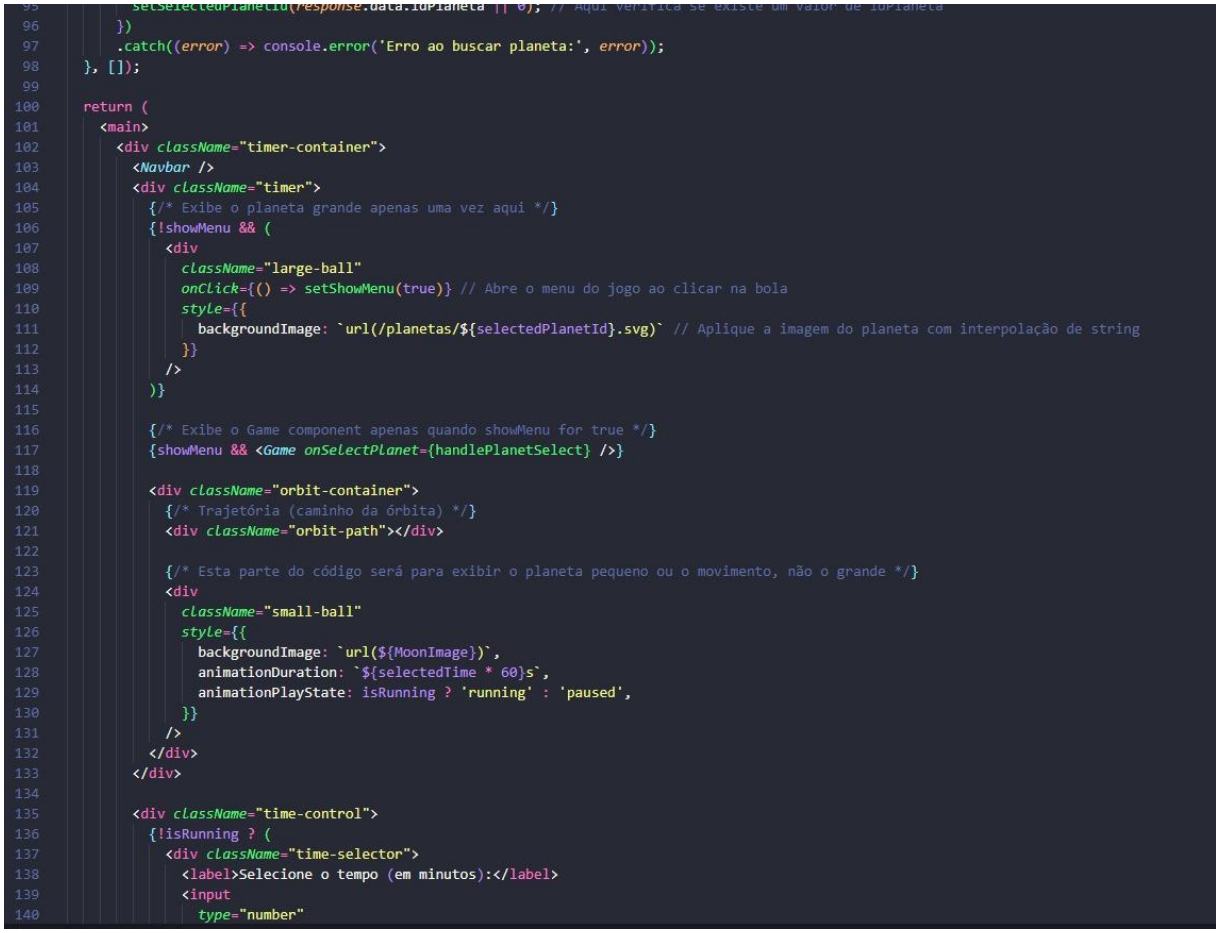
```jsx
import React, { useState, useEffect } from 'react';
import './TimerPage.css';
import Navbar from '../../components/Navbar';
import MoonImage from '../../images/lua.svg';
import { useNavigate } from 'react-router-dom';
import axios from 'axios';
import Game from '../../components/Game';

const TimerPage = () => {
  const [time, setTime] = useState(0);
  const [isRunning, setIsRunning] = useState(false);
  const [selectedTime, setSelectedTime] = useState(25);
  const [timerId, setTimerId] = useState(null);
  const [isSaving, setIsSaving] = useState(false);
  const [showMenu, setShowMenu] = useState(false);
  const [selectedPlanetId, setSelectedPlanetId] = useState(1);
  const [hasStarted, setHasStarted] = useState(false);
  const navigate = useNavigate();

  // Função para iniciar o timer
  const startTimer = () => {
    if (selectedTime <= 0 || isRunning) return; // Verifica se o timer já está rodando

    setIsRunning(true);
    setTime(selectedTime * 60);
    setHasStarted(true);

    const id = setInterval(() => {
      setTime((prevTime) => {
        if (prevTime <= 1) {
          clearInterval(id);
          setIsRunning(false);
          setHasStarted(false);

          if (!isSaving) {
            savePomodoro(selectedTime);
          }

          return 0;
        }
        return prevTime - 1;
      });
    }, 1000);

    setTimerId(id); // Armazena o id do timer
  };

  // Função para parar o timer
  const stopTimer = () => {
```

Figura 83 – Timer



```
49  const stopTimer = () => {
50    clearInterval(timerId);
51    setIsRunning(false);
52    setHasStarted(false);
53    setSelectedTime(25); // Resetar o tempo selecionado para o valor default
54  };
55
56  // Função para salvar o tempo
57  const savePomodoro = (tempoAtiv) => {
58    if (isSaving) return;
59
60    setIsSaving(true);
61    const tempoAtivNum = Number(tempoAtiv);
62    const dataAtual = new Date().toISOString().split('T')[0];
63
64    fetch('http://localhost:5000/save-pomodoro', {
65      method: 'POST',
66      headers: {
67        'Content-Type': 'application/json',
68      },
69      body: JSON.stringify({ tempoAtiv: tempoAtivNum, dataAtiv: dataAtual }), // Envia tempo e data
70    })
71      .then((response) => response.json())
72      .then((data) => {
73        console.log('Tempo total salvo:', data.totalTime);
74      })
75      .catch((error) => {
76        console.error('Erro ao salvar o tempo:', error);
77      })
78      .finally(() => {
79        setIsSaving(false);
80      });
81  };
82
83  // Função para atualizar o planeta selecionado
84  const handlePlanetSelect = (planetId) => {
85    setSelectedPlanetId(planetId);
86    setShowMenu(false); // Fechar o menu após seleção
87    axios.put('http://localhost:5000/update-planet', { idPlaneta: planetId })
88    .catch((error) => console.error('Erro ao atualizar planeta:', error));
89  };
90
91  // Recupera o último planeta selecionado ao carregar a página
92  useEffect(() => {
93    axios.get('http://localhost:5000/get-last-pomodoro')
94      .then((response) => {
95        setSelectedPlanetId(response.data.idPlaneta || 0); // Aqui verifica se existe um valor de idPlaneta
```

Figura 84 – Timer

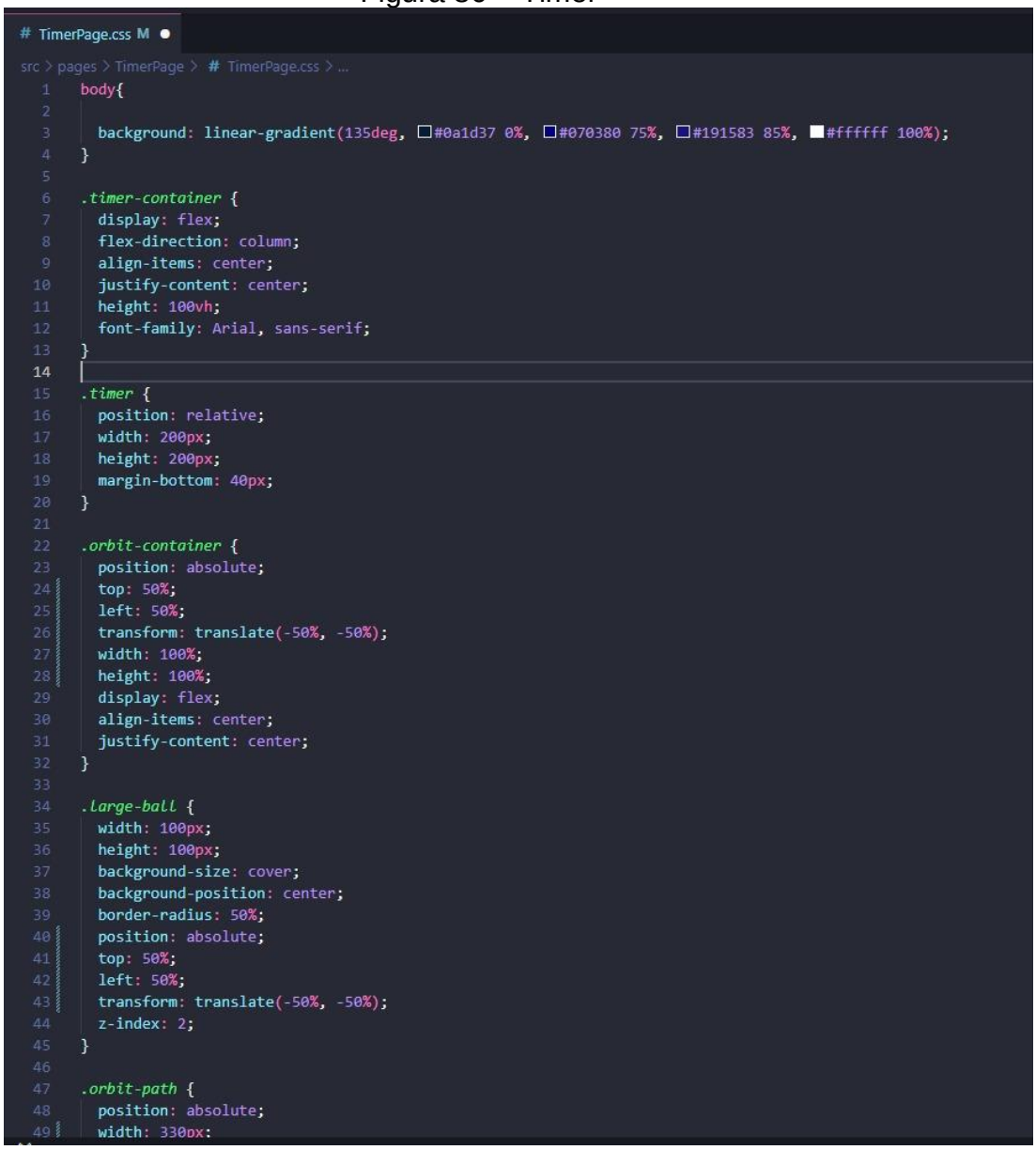


```
        setSelectedPlanetId(response.data.idPlaneta || 0); // Aqui verifica se existe um valor de idPlaneta
      })
      .catch((error) => console.error('Erro ao buscar planeta:', error));
  }, []);

  return (
    <main>
      <div className="timer-container">
        <Navbar />
        <div className="timer">
          {/* Exibe o planeta grande apenas uma vez aqui */}
          {!showMenu && (
            <div
              className="large-ball"
              onClick={() => setShowMenu(true)} // Abre o menu do jogo ao clicar na bola
              style={{
                backgroundImage: `url(/planetas/${selectedPlanetId}.svg)` // Aplique a imagem do planeta com interpolação de string
              }}
            />
          )}

          {/* Exibe o Game component apenas quando showMenu for true */}
          {showMenu && <Game onSelectPlanet={handlePlanetSelect} />}

          <div className="orbit-container">
            {/* Trajetória (caminho da órbita) */}
            <div className="orbit-path"></div>

            {/* Esta parte do código será para exibir o planeta pequeno ou o movimento, não o grande */}
            <div
              className="small-ball"
              style={{
                backgroundImage: `url(${MoonImage})`,
                animationDuration: `${selectedTime * 60}s`,
                animationPlayState: isRunning ? 'running' : 'paused',
              }}
            />
          </div>
        </div>

        <div className="time-control">
          {!isRunning ? (
            <div className="time-selector">
              <label>Selecione o tempo (em minutos):</label>
              <input
                type="number"
```

Figura 85 – Timer

```
                    type="number"
                    value={selectedTime}
                    onChange={(e) => setSelectedTime(Number(e.target.value))}
                    min="1"
                  />
                  <button onClick={startTimer} disabled={selectedTime <= 0}>
                    Iniciar
                  </button>
                </div>
              ) : (
                <div className="time-display">
                  <span>
                    {Math.floor(time / 60)}:{String(time % 60).padStart(2, '0')}s
                  </span>
                  <button onClick={stopTimer}>Parar</button>
                </div>
              )}
            </div>
          </div>
        </main>
    );
  };

  export default TimerPage;

```

Figura 86 – Timer



TimerPage.css M

src > pages > TimerPage > # TimerPage.css > ...

```css
body{

  background: linear-gradient(135deg, #0a1d37 0%, #070380 75%, #191583 85%, #ffffff 100%);
}

.timer-container {
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  height: 100vh;
  font-family: Arial, sans-serif;
}

.timer {
  position: relative;
  width: 200px;
  height: 200px;
  margin-bottom: 40px;
}

.orbit-container {
  position: absolute;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
  width: 100%;
  height: 100%;
  display: flex;
  align-items: center;
  justify-content: center;
}

.large-ball {
  width: 100px;
  height: 100px;
  background-size: cover;
  background-position: center;
  border-radius: 50%;
  position: absolute;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
  z-index: 2;
}

.orbit-path {
  position: absolute;
  width: 330px;
```

Figura 87 – Timer

```
    width: 330px;
    height: 330px;
    border-radius: 50%;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
}

.small-ball {
    border: 2.5px solid #007bff;
    width: 20px;
    height: 20px;
    background-size: cover;
    background-position: center;
    border-radius: 50%;
    position: absolute;
    top: 50%;
    left: 50%;
    transform-origin: -160px 0;
    animation-name: orbit;
    animation-timing-function: linear;
    animation-iteration-count: infinite;
}

@keyframes orbit {
    0% {
        transform: rotate(0deg) translateX(-160px) rotate(0deg);
    }
    100% {
        transform: rotate(360deg) translateX(-160px) rotate(-360deg);
    }
}

.time-control {
    text-align: center;
}

.time-selector label {
    display: block;
    margin-top: 50px;
    margin-bottom: 10px;
}

.time-selector input {
    width: 80px;
    padding: 5px;
    margin-right: 10px;
    width: 50%;
```

Figura 88 – Timer

```
  width: 50%;
  padding: 10px;
  margin-top: 5px;
  margin-bottom: 15px;
  border: 1px solid #ccc;
  border-radius: 5px;
}

.time-selector button,
.time-display button {
  padding: 10px;
  background-color: #007bff;
  color: white;
  border: none;
  border-radius: 4px;
  cursor: pointer;
}

.time-selector button:disabled {
  background-color: #d3d3d3;
}

.time-display span {
  margin-top: 50px;
  font-size: 24px;
  font-weight: bold;
  color: aliceblue;
}

.time-display {
  display: flex;
  flex-direction: column;
  align-items: center;
}

.time-display button {
  margin-top: 10px;
}

```

11.13 SPLASHSCREEN:

Figura 89 – Splashscreen

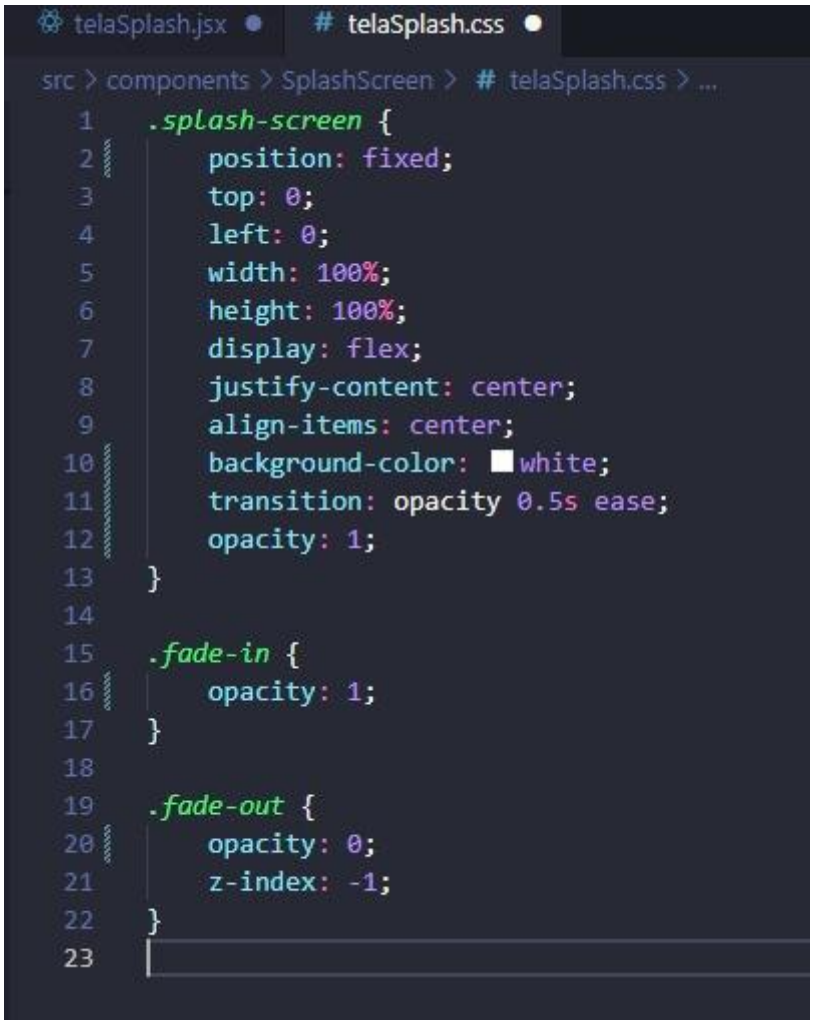


Figura 90 – Splashscreen

telaSplash.jsx ● # telaSplash.css ●

src > components > SplashScreen > telaSplash.jsx > ...

```jsx
import React, { useEffect, useState } from "react";
import Tela from "./telaLogo.svg";
import './telaSplash.css';

function TelaSplash({ onSplashEnd }) {
    const [isVisible, setIsVisible] = useState(true);

    useEffect(() => {
        const timer = setTimeout(() => {
            setIsVisible(false);
            onSplashEnd();
        }, 1500);

        return () => clearTimeout(timer);
    }, [onSplashEnd]);

    return (
        <div className={`splash-screen ${isVisible ? 'fade-in' : 'fade-out'}`}>
            <img src={Tela} alt="Tela de Splash" />
        </div>
    );
}

export default TelaSplash;
```

11.14 FRISTPAGE:

Figura 91 – Fristpage

FirstPage.jsx

src > pages > FirstPage > FirstPage.jsx > FirstPage

```
1   import React, { useState } from "react";
2   import "./FirstPage.css";
3   import TelaSplash from "../../components/SplashScreen/telaSplash";
4   import "../../components/Login.css";
5   import Login from "../../components/Login";
6
7   const FirstPage = () => {
8     const [showContent, setShowContent] = useState(false);
9
10    const handleSplashEnd = () => {
11      setShowContent(true);
12    };
13
14    return (
15      <main>
16        <div>
17
18          <TelaSplash onSplashEnd={handleSplashEnd} />
19        </div>
20
21
22        {showContent && (
23          <div className="conteudo-principal">
24            <Login setIsAuthenticated={() => {}} />
25          </div>
26        )}
27      </main>
28    );
29  };
30
31  export default FirstPage;
32
```

FirstPage.jsx ● # FirstPage.css ●

src > pages > FirstPage > # FirstPage.css > ...

```css
body{
    margin-top: 0px;
    background: linear-gradient(135deg, #0a1d37 0%, #070380 75%, #191583 85%, #ffffff 100%);
}

main {
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    background: linear-gradient(135deg,
                                #0a1d37 0%,
                                #070380 75%,
                                #ffffff 200%);
  }

  .conteudo-principal {
    width: 100%;
    text-align: center;
    padding: 20px;
  }

  .logo {
    justify-content: center;
  }
```

index.css ● JS App.js ×

src > JS App.js > ...

```
import React from 'react';
import { Routes, Route } from 'react-router-dom';
import CadastroPage from './pages/CadastroPage/CadastroPage';
import TimerPage from './pages/TimerPage/TimerPage';
import CalendarPage from './pages/CalendarPage/CalendarPage';
import DayPage from './pages/DayPage/DayPage';
import AccountPage from './pages/AccountPage/AccountPage';
import Login from './pages/FirstPage/FirstPage';
import GraficoPage from './pages/GraficoPage/GraficoPage';
import { AppointmentsProvider } from './context/AppointmentsContext';

function App() {
  return (
    <AppointmentsProvider>
      <Routes>
        {/* Página inicial de login */}
        <Route path="/" element={<Login />} />

        {/* Página de cadastro */}
        <Route path="/cadastro" element={<CadastroPage />} />

        {/* Páginas principais */}
        <Route path="/timer" element={<TimerPage />} />
        <Route path="/calendar" element={<CalendarPage />} />
        <Route path="/day/:month/:day" element={<DayPage />} />
        <Route path="/conta" element={<AccountPage />} />
        <Route path="/grafico" element={<GraficoPage />} />

        {/* Rota padrão para páginas não encontradas */}
        <Route path="*" element={<Login />} />
      </Routes>
    </AppointmentsProvider>
  );
}

export default App;
```

12. DISCUSSÃO DOS RESULTADOS

Os dados coletados foram obtidos por meio de questionários distribuídos entre jovens de 15 a 22 anos, abrangendo questões sobre hábitos organizacionais e o impacto das distrações digitais na produtividade e saúde mental. No total, 30 respostas foram analisadas, revelando padrões importantes:

Falta de Organização:A maioria dos participantes relatou dificuldades em gerenciar o tempo devido ao uso excessivo de dispositivos móveis, o que interfere diretamente em suas rotinas acadêmicas e profissionais.
Saúde Mental e Produtividade: Grande parte dos respondentes indicou sentir-se sobrecarregada, desmotivada e ansiosa, fatores que reforçam a importância de uma ferramenta que promova organização e foco.

Preferências por Ferramentas Digitais: Os dados mostram um interesse significativo em aplicativos que ajudem na criação de rotinas e que ofereçam funcionalidades gamificadas para manter o engajamento.

Gráficos e tabelas foram utilizados para ilustrar os resultados, destacando a relação direta entre organização pessoal e o bem-estar emocional dos participantes. Além disso, os dados forneceram subsídios para o desenvolvimento das funcionalidades do aplicativo EasyPlan, como lembretes personalizados, gameficação e estatísticas de progresso.

13. CONSIDERAÇÕES FINAIS / CONCLUSÃO

13.1 CONCLUSÃO

Este trabalho conclui que o aplicativo **EasyPlan** tem o potencial de ajudar jovens a lidar melhor com a desorganização e as distrações digitais em suas vidas diárias. A pesquisa revelou que muitos enfrentam dificuldades em equilibrar suas responsabilidades, o que afeta tanto a produtividade quanto a saúde mental.

O **EasyPlan** oferece soluções práticas, como lembretes, planejamento de tarefas e recursos gameficados, que ajudam a melhorar a gestão do tempo e promover uma rotina mais saudável e equilibrada. Além disso, a ferramenta proporciona uma experiência mais personalizada, permitindo que os usuários adaptem o app às suas necessidades específicas.

Com base nos resultados obtidos, é evidente que o **EasyPlan** pode ser uma grande aliada para aqueles que desejam alcançar mais organização, foco e bem-estar emocional. A proposta vai além de apenas gerenciar tarefas, ajudando os usuários a desenvolver hábitos mais saudáveis e a manter uma vida mais produtiva.

Esse projeto também abre espaço para melhorias contínuas, sugerindo que novas versões do aplicativo possam oferecer ainda mais recursos, mantendo-se relevante e útil para diferentes públicos ao longo do tempo. Assim, o **EasyPlan** se consolida como uma ferramenta não apenas prática, mas também cheia de potencial para transformar a rotina de seus usuários.

13.2 REFERÊNCIAS

[1] BRITO, Kátia Cristina. A importância de uma vida estruturada para a saúde mental. 2023. Disponível em: < PDF - Kátia Cristina Barbosa Ferreira.pdf >. Acesso em: 22 nov. 2024.

[2] BONATTI, Nicole. O impacto da desorganização no ambiente de trabalho e na saúde mental. 2021.

[3] ZIMBARDO, P. G.; BOYD, J. N. The Time Paradox: The New Psychology of Time That Will Change Your Life. New York: Free Press, 2008. Disponível em: <The Time Paradox - Internet Archive>. Acesso em: 22 nov. 2024.

[4] MACAN, T. H. Time-management: test of a process model. Journal of Applied Psychology, v. 79, n. 3, p. 381-391, 1994. Disponível em: <Time Management: test of a Process Model - ResearchGate>. Acesso em: 22 nov. 2024.

[5] CLAESSENS, B. J. C.; VAN EDE, E. R.; WERK, L. M. van. A review of time management research: A test of the time-management-performance model. Personnel Psychology, v. 60, p. 559-577, 2007. Disponível em: <https://www.researchgate.net/publication/228664480_A_Review_of_Time_Management_Literature>. Acesso em: 22 nov. 2024.

[6] STEEL, P. The nature of procrastination: a meta-analytic and theoretical review of quintessential self-regulatory failure. Psychological Bulletin, v. 133, n. 1, p. 65-94, 2007. Disponível em: <https://pubmed.ncbi.nlm.nih.gov/17201571/>. Acesso em: 22 nov. 2024.

[7] POSNER, M. I.; PETERSEN, S. E. The attention system of the human brain. Annual Review of Neuroscience, v. 13, p. 25-42, 1990. Disponível em: <https://pubmed.ncbi.nlm.nih.gov/2183676/>. Acesso em: 22 nov. 2024.

[8] DUCKWORTH, A. L.; SULTAN, D.; GILLHAM, J.; KIM, S.; QUAO, E. Self-discipline outdoes IQ in predicting academic performance of adolescents. Personality and Social

Psychology Bulletin, v. 33, n. 8, p. 1067-1079, 2007. Disponível em: <https://www.researchgate.net/publication/7454641_Self-Discipline_Outdoes_IQ_in_Predicting_Academic_Performance_of_Adolescents>. Acesso em: 22 nov. 2024.

[9] RYAN, R. M.; DECI, E. L. Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being. American Psychologist, v. 55, n. 1, p. Disponível em: <https://selfdeterminationtheory.org/SDT/documents/2000_RyanDeci_SDT.pdf>. Acesso em: 22 nov. 2024.

Cirillo, Francesco. The Pomodoro Technique: The Life-Changing Time-Management System. 2006. [ISBN se disponível]

Clear, James. Atomic Habits: An Easy & Proven Way to Build Good Habits & Break Bad Ones. Avery, 2018.

Morgenstern, Julie. Time Management from the Inside Out: The Foolproof System for Taking Control of Your Schedule—and Your Life. Henry Holt and Co., 2004.