CENTRO PAULA SOUZA
FACULDADE DE TECNOLOGIA DE FRANCA
“Dr. THOMAZ NOVELINO”

TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

LEONARDO MENDONÇA FERREIRA

ROUTINE – GERENCIAMENTO DE HÁBITOS

Trabalho de Graduação apresentado à Faculdade de Tecnologia de Franca - “Dr. Thomaz Novelino”, como parte dos requisitos obrigatórios para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientador: Carlos Alberto Lucas

FRANCA/SP
2024

ROUTINE - GERENCIAMENTO DE HÁBITOS

Leonardo Mendonça Ferreira[1]
Carlos Alberto Lucas [2]

Resumo

Na correria da vida moderna, a atual geração enfrenta desafios significativos para cumprir prazos e demandas necessárias no cotidiano. A sobrecarga de informações e distrações constantes provenientes de dispositivos eletrônicos e redes sociais contribuem para a falta de foco e procrastinação. Além disso, a natureza multitarefa da vida moderna pode resultar em uma divisão desigual de atenção, comprometendo a eficiência na conclusão de atividades simples. A urgência de enfrentar esse problema é crucial, pois a incapacidade de cumprir prazos pode ter implicações diretas na produtividade individual e coletiva. Felizmente, os aplicativos de gestão de hábitos emergem como ferramentas valiosas para mitigar esses desafios. Essas aplicações oferecem funcionalidades que permitem a criação de listas de hábitos, definição de prazos e visualização semanal, mensal e anual das tarefas pendentes, proporcionando uma abordagem estruturada para o gerenciamento do tempo. Além disso, muitos desses aplicativos incorporam técnicas de produtividade comprovadas, como a técnica Pomodoro, para incentivar a concentração e a conclusão eficiente dos hábitos. Ao fornecer uma interface intuitiva e acessível, essas ferramentas capacitam os usuários a organizarem suas responsabilidades diárias de maneira mais eficaz, superando as barreiras que a era digital impôs à gestão do tempo.

Palavras-chave: gerenciamento de hábitos; monitoramento do tempo; administração de tarefas; prazos.

Abstract

In the hustle of modern life, the current generation faces significant challenges in meeting deadlines and daily demands. The overload of information and constant distractions from electronic devices and social media contribute to a lack of focus and procrastination. Moreover, the multitasking nature of modern life can lead to an uneven division of attention, compromising efficiency in completing simple tasks. Addressing this issue is crucial, as the inability to meet deadlines can have direct implications on individual and collective productivity. Fortunately, habit management apps are emerging as valuable tools to mitigate these challenges. These applications offer features that allow the creation of habit lists, deadline setting, and weekly, monthly, and yearly task tracking, providing a structured approach to time management. Additionally, many of these apps incorporate proven productivity techniques, such as the Pomodoro technique, to encourage focus and the efficient completion of habits. By offering an intuitive and accessible interface, these tools empower users to organize their daily responsibilities more effectively, overcoming the barriers that the digital age has imposed on time management.

[1] Graduando em Análise e Desenvolvimento de Sistemas pela Fatec Dr Thomaz Novelino – Franca/SP. Endereço eletrônico: leonardo.ferreira17@fatec.sp.gov.br
[2] Docente do CST em Análise e Desenvolvimento de Sistemas da Faculdade de Tecnologia Dr Thomaz Novelino – Fatec Franca/SP. Endereço eletrônico: carlos.lucas@fatec.sp.gov.br

Keywords: *habit management; time tracking; task management; deadlines.*

1 Introdução

Um dos maiores desafios da atualidade é controlar o tempo e os hábitos diários devido às distrações constantes presentes no cotidiano. Pode ser difícil manter o foco e priorizar tarefas importantes, seja ao responder a e-mails, participar de reuniões, lidar com problemas inesperados ou realizar hábitos diários básicos, como arrumar a cama. Isso pode ocasionar a sensação de opressão e de que não há progresso nas incumbências mais importantes, gerando uma fadiga mental que prejudica ainda mais o foco e a produtividade, além da tendência à procrastinação ou evitar hábitos considerados difíceis ou desagradáveis.

Tudo isso leva a um acúmulo de tarefas que precisam ser concluídas, o que aumenta ainda mais a sensação de sobrecarga e dificulta o início e a finalização dos objetivos. Para superar esses desafios, é importante priorizar os hábitos, dividi-los em etapas menores e definir metas realistas. De acordo com *Prodest* (s.d.), existe uma lista de atividades e ações que podem colaborar para o total controle das tarefas e, uma delas, é o gerenciamento do tempo e dos hábitos que deverão ser cumpridos.

Pensando no problema em questão, o presente projeto tem por objetivo oferecer uma ferramenta para controle de hábitos diários e possibilitar que o usuário execute sua rotina com eficiência. O aplicativo de gerenciamento de hábitos possibilita facilmente criar e priorizar hábitos e acompanhar o seu progresso ao longo do tempo. Essas funcionalidades auxiliarão o usuário a se manter organizado, focado e produtivo, mesmo com uma rotina muito atribulada.

1.1 Termo da Abertura do Projeto (TAP)

O Termo de Abertura de Projeto (TAP), segundo o INPE (2021) é o documento que marca o início do projeto e estabelece as bases para o seu gerenciamento.

A partir da definição do Guia PMBOK (2013), entende-se que o TAP é usado para gerenciar os recursos organizacionais que serão utilizados ao longo da proposta, além de vital para o garantir o ciclo de vida do sistema e para a gestão de qualquer imprevisto que possa acontecer. Logo, todas as partes envolvidas devem estar totalmente cientes do TAP, assinando-o assim que estiver pronto.

Este documento tem como objetivo informar e documentar as partes do software *Routine*, onde serão detalhadas as etapas para o desenvolvimento do

sistema, detalhes do software e de quem o projetou. Além disso, fornecerá a documentação completa dos artefatos do software, constando todos os diagramas, requisitos funcionais e não funcionais, regras de negócios do sistema e os casos de uso, assim como a entrega da ferramenta em pleno funcionamento, para que o usuário possa usufruir integralmente da plataforma com todas as suas funções, após a realização do cadastro.

O software foi baseado em uma *NLW*, desenvolvida pela *Rocketseat*, que apresenta o desenvolvimento básico do software. Como proposta para o presente Trabalho de Graduação, foi selecionado o software em questão, porém com o aprimoramento de algumas funções que garantem a viabilidade do projeto, como login, para garantir maior segurança ao usuário, e banco de dados, para armazenar e atualizar os hábitos do usuário, como a inserção de um novo objetivo ou a exclusão de outro.

Durante a realização do curso de Análise e Desenvolvimento de Sistemas na Fatec - Franca/SP, um dos maiores desafios encontrados pelos alunos foi conciliar o curto tempo disponível com as tarefas do curso, o que muitas vezes os levava a perder-se nos assuntos, ocasionando atrasos na entrega das atividades. Algumas das dificuldades na utilização de softwares já disponíveis no mercado ocorrem por conta dos preços inacessíveis, e o fato de que os sistemas são destinados à utilização profissional, havendo diversas funções que normalmente não são utilizadas, tornando seu uso mais trabalhoso.

A partir da identificação de tais dificuldades e da necessidade de um software mais simples para o acompanhamento de metas pessoais diárias, a opção *Routine* será uma escolha acessível, prática e viável para qualquer perfil de usuário, entregando o resultado esperado.

Foi desenvolvido um software básico de gestão e controle de hábitos cotidianos, que deverá aceitar o cadastro e a exclusão de hábitos em um ou mais dias da semana; realizar o login para garantir a segurança dos dados do usuário; e mostrar o progresso diário, semanal, mensal e anual dos hábitos realizados.

As restrições do projeto se resumem ao curto prazo de entrega, dificultando o aperfeiçoamento e os altos custos para a utilização de ferramentas pagas necessárias no desenvolvimento da aplicação.

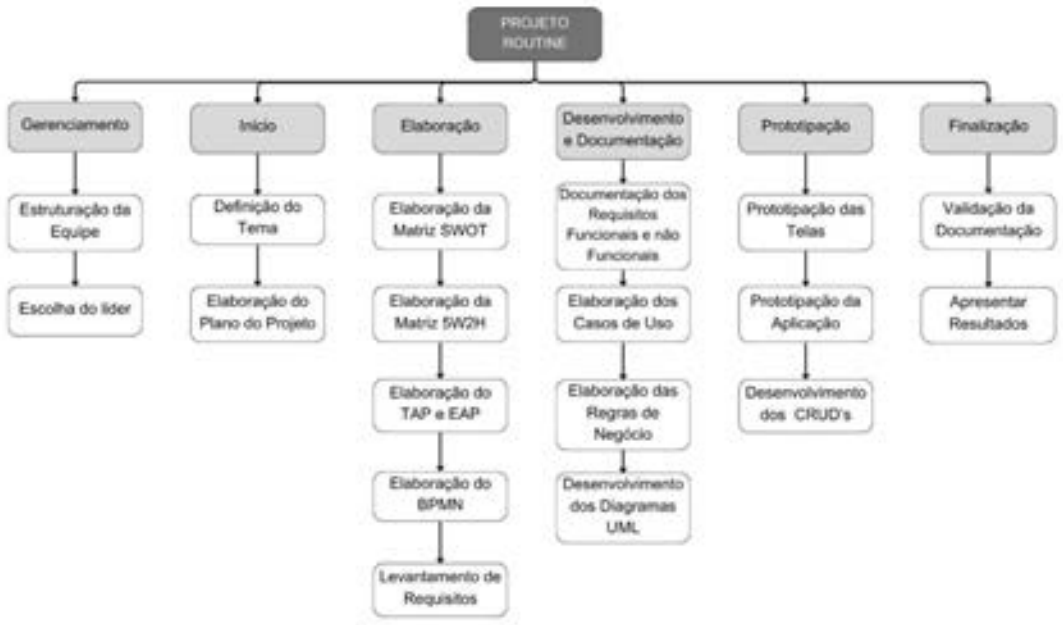
Quanto aos fatores de riscos externos, nota-se a dificuldade de inserção de um software novo em um mercado saturado, a dificuldade de conquista do usuário e o

fato da utilização ser de total responsabilidade dele, o que muitas vezes faz com que ele deixe de acompanhar os hábitos e o software. Os fatores de riscos internos envolvem dificuldade na parte da programação por falta de conhecimento mais aprofundado em tal linguagem.

Os produtos e artefatos esperados são: TAP, Matriz SWOT, 5W2H, diagramas UML, documentação dos requisitos, prototipação de telas e um protótipo do software.

Segundo o Guia PMBOK (2013), a EAP é a decomposição hierárquica orientada à entrega do escopo total do trabalho a ser executado pela equipe do projeto para atingir os objetivos e entregar o que foi requerido, e o seu principal benefício é fornecer uma visão geral do que deve ser entregue. É um modelo que exibe os componentes ou atividades necessárias para o desenvolvimento do projeto em divisões hierárquicas, trazendo maior facilidade no seu gerenciamento e entendimento dos passos que devem ser adotados. A Figura 1 apresenta o EAP do projeto deste estudo.

Figura 1: EAP do projeto



Fonte: Elaborado pelo autor

A Estrutura Analítica do Projeto é um dos documentos iniciais, utilizada para facilitar o gerenciamento do projeto e as etapas que devem ser cumpridas. A primeira etapa definida foi o Gerenciamento, que abrange as pesquisas iniciais, a realização do curso *NLW*, da *Rocketseat*, e a definição da bibliografia.

A fase inicial do projeto foi responsável por definir o tema e elaborar o Plano do Projeto. Foi importante refletir sobre as necessidades encontradas pelo usuário (público-alvo), levando em consideração as dificuldades identificadas. O roteiro a ser utilizado foi elaborado a fim de atingir as metas definidas com as metodologias que serão aplicadas no projeto.

Na segunda fase, a Elaboração, foi realizado o levantamento do problema em geral, utilizando dados colhidos através de pesquisas e estudos; dos processos que serão realizados; e os resultados esperados. As metodologias escolhidas na fase anterior foram utilizadas para maior compreensão do projeto, observando as potencialidades e as fragilidades da aplicação. Com os dados colhidos, foram criados artefatos fundamentais para o avanço do projeto, como a Matriz SWOT que é responsável por levantar os pontos fracos e fortes da aplicação e a elaboração do plano 5W2H que detalha as características da aplicação e de atividades que ela realizará.

Na etapa em questão, também houve a elaboração do Termo de Abertura do Projeto (TAP), da Estrutura Analítica do Projeto (EAP), da Notação de Modelagem de Processos de Negócios (BPMN - *Business Process Model and Notation)*, além do Levantamento de Requisitos. Os modelos mencionados acima são responsáveis por detalhar ainda mais os procedimentos em que a aplicação atuará, através da diagramação e modelagem dos processos de acordo com o que foi estudado ao longo do curso e padrão definido pela Engenharia de Software (ES).

Com os dados colhidos e os diagramas montados, teve início a fase de Desenvolvimento e Documentação, com a aplicação de práticas e métodos especificados pela Engenharia de Software, com a utilização de ferramentas já disponíveis no mercado, documentando o funcionamento dos processos a fim de facilitar o desenvolvimento da aplicação. Os diagramas UML são responsáveis por facilitar a visualização do projeto antes que seja iniciada a fase da prototipação, tornando mais fácil aplicar as regras estudadas dentro do desenvolvimento da aplicação. Os diagramas entregues são: BPMN do sistema, Caso de Uso, Requisitos Funcionais, Requisitos não Funcionais, Regras de Negócio e o Canvas do Negócio.

Após realizado o levantamento dos dados e a criação de todos os diagramas definidos no projeto, teve início a fase da Prototipação, começando o desenvolvimento do software. A etapa em questão é utilizada para apresentarmos ao público-alvo a solução em desenvolvimento, através da prototipação das telas e da aplicação, antes

mesmo do início do projeto. Assim como outros métodos utilizados ao longo do projeto, a prototipação é responsável por baratear os custos e facilitar a representação dos requisitos e das regras impostas à aplicação, garantindo que ela atingirá os parâmetros levantados ao longo do estudo e modelagem do projeto. Nesta fase, também é realizado o Desenvolvimento dos 3 CRUD, com a implementação das funcionalidades básicas do código.

2 Viabilidade do Projeto

Esta seção tem como objetivo apresentar os estudos realizados e a viabilidade do projeto. Para isso, foram utilizadas as ferramentas e os métodos estudados ao longo do curso Análise e Desenvolvimento de Sistemas da Fatec – Franca/SP, nas matérias de Engenharia de Software. Serão apresentados a matriz SWOT do negócio, o plano 5W2H e o Canvas do negócio.

2.1 Canvas de Negócio (*Business Model Canvas* - BMC)

Segundo a Inatel (2012), o desenvolvimento de um novo produto pode se beneficiar muito da metodologia Canvas. Nos tempos modernos, em que a velocidade das mudanças, a grande competitividade e a vontade do cliente são importantes agentes que influenciam fortemente na necessidade de inovações constantes, esta tecnologia pode colaborar para o sucesso do empreendimento.

O modelo baseia-se nos processos de realimentação e testes de campo para sua validação, anterior à sua aplicação na efetivação do negócio. Para a criação do modelo Canvas, foi utilizada a ferramenta Canva. Canva (s.d) é uma plataforma online de design e comunicação visual. O Quadro 1 evidencia o modelo Canvas para a aplicação em questão.

Quadro 1: Canvas do Negócio

Fonte: Elaborado pelo autor

A aplicação foi projetada para a população em geral. Pensando na problemática da questão que é a dificuldade de gerenciamento do tempo e de hábitos, a proposta de valor da aplicação se resume a um software que facilite o acompanhamento e a visualização dos hábitos diários e que traga uma maior simplicidade em sua utilização, evitando um processo que exija mais tempo dedicado ao controle da rotina.

A ferramenta será disponibilizada ao usuário por meio de aplicação na WEB e divulgada a partir de anúncios no Flyer do Instagram e impulsionamentos pelo Google. Para maior conectividade com o usuário, será oferecido atendimento via e-mail de suporte e através das mídias sociais.

O diferencial proposto pela aplicação é a simplicidade no uso e a ausência de custos para o usuário; logo, a rentabilidade provém de anúncios existentes dentro da página inicial da ferramenta, assim como a estrutura de custos do projeto se resume à hospedagem da página na internet.

No processo de desenvolvimento do software, foram necessários o conhecimento técnico da equipe de desenvolvimento e o acesso a um computador. As atividades principais oferecidas pela equipe envolvem a entrega e a manutenção contínua da ferramenta.

2.2 Matriz SWOT

De acordo com Nakagawa (2015), a Matriz SWOT é uma ferramenta clássica da Administração e de extrema importância para análises estratégicas do projeto, sendo utilizada para avaliar sua situação, identificando e reconhecendo suas forças e fraquezas, além de novas oportunidades e ameaças a que ele está submetido.

É uma técnica que avalia todo o contexto e que considera o mercado onde o projeto estará exposto, avaliando a situação e quais diferenciais a oferecer quanto a concorrentes já estabelecidos ou não. No Quadro 2, está a matriz SWOT do projeto em questão.

Quadro 2: Matriz SWOT

Fonte: Elaborado pelo autor

Com a Matriz SWOT, foram identificadas as maiores dificuldades e oportunidades com a aplicação em relação ao mercado, como uma certa dificuldade em aplicar hábitos simples e repetitivos em softwares já estabelecidos dentro do mercado atual, trazendo uma grande oportunidade à aplicação da proposta deste projeto, onde é possível criar hábitos simultaneamente e com uma maior facilidade, além de facilitar a visualização de conclusão dos hábitos especificados.

Por ser um programa remodelado e incrementado, uma das maiores ameaças encontradas foi a dificuldade de diferenciar os softwares de outras plataformas já

disponíveis atualmente, mas criando uma grande oportunidade de alteração e entrega de uma aplicação não genérica.

2.3 Plano de Ação 5W2H do Projeto

Para o Sebrae (2023), o Plano de ação 5W2H é um checklist das atividades preventivas e corretivas que precisam ser desenvolvidas, e deve ser organizado de forma prática, simples, eficiente e clara. O plano de ação tem por objetivo definir os desdobramentos da estratégia em ações com monitoramento. Por meio dele, deve-se planejar tudo a ser executado, criar uma metodologia, elaborar um cronograma e indicar os responsáveis para dar o andamento de cada etapa.

O Plano de ação 5W2H foi uma das metodologias aplicadas no decorrer do projeto por definir um roteiro para a execução dos hábitos. Ainda segundo o Sebrae (2023), o plano de ação é composto por sete perguntas, as quais foram aplicadas ao projeto. São elas:

- *What*? (O que?): define o que deve ser feito pelos autores do projeto;
- *Why*? (Por quê?): define-se a justificativa, o motivo da proposta ser desenvolvida para a resolução do problema;
- *Where*? (Onde?): informa onde as ações propostas devem acontecer ou afetar;
- *Who*? (Quem?): define quem são os atores e responsáveis pela ação a ser tomada;
- *When*? (Quando): serve para definir quando as ações serão tomadas;
- *How*? (Como): informa como tais ações serão tomadas ou implementadas;
- *How Much*? (Quanto custa?): define quanto deverá custar tais medidas.

Quadro 3: Plano de ação 5W2H

	O que? (What?)	Por que? (Why?)	Onde? (Where?)	Quando? (When?)	Por quem? (Who?)	Como? (How?)	Quanto? (How much?)
1	Realizar cadastro no software.	Garantir a segurança ao usuário.	Na internet, utilizando um navegador WEB.	Quando houver um usuário acessando o software.	Usuário	Colocando o usuário e a senha.	Tempo
2	Cadastrar hábito.	Realizar acompanhamento das metas através de hábitos no software.	Na internet, utilizando um navegador WEB.	Quando houver um usuário adicionando um hábito no software.	Usuário	1. Clicando no botão para cadastrar o hábito. 2. Definindo um nome e uma data para o hábito.	Tempo
3	Excluir hábito.	Excluir alguma meta no software.	Na internet, utilizando um navegador WEB.	Quando houver um usuário solicitando a edição do hábito no software.	Usuário	Clicando no botão para excluir o hábito.	Tempo
4	Exibir progresso diário.	Facilitar o acompanhamento da conclusão de metas diárias.	Na internet, utilizando um navegador WEB.	Quando houver um usuário solicitando a exibição do progresso realizado.	Usuário	Clicando em cima do dia que deseja visualizar.	Tempo
5	Exibir progresso semanal.	Facilitar o acompanhamento da conclusão de metas na semana.	Na internet, utilizando um navegador WEB.	Quando houver um usuário solicitando a exibição do progresso realizado.	Usuário	Visualizando na aplicação.	Tempo
6	Exibir progresso mensal/anual.	Facilitar o acompanhamento da conclusão de metas no mês/ano.	Na internet, utilizando um navegador WEB.	Quando houver um usuário solicitando a exibição do progresso realizado.	Usuário	Clicando no botão de acompanhamento mensal/anual.	Tempo

Fonte: Elaborado pelo autor

Diante do descrito, foi imprescindível a utilização desta ferramenta no decorrer do projeto devido a sua importância e descrição textual das tarefas realizadas pelo software.

A primeira atividade foi o cadastro do cliente para que possa acessar o programa, sendo ele o responsável por seu cadastro e a segurança de sua conta e dados. Na segunda ação, são iniciadas as funções que o software apresentará aos seus usuários, com a possibilidade de registrar sua rotina, incluindo suas metas e o prazo para que ela seja realizada, podendo ser recorrente ou não.

Na terceira ação, é definida a opção de excluir um hábito, caso o usuário considere que não precisa realizá-lo. A quarta ação é uma aba que será responsável por exibir o progresso dos hábitos diariamente, completando a barra de progresso no decorrer das conclusões dos hábitos. Da mesma forma que é exibido o progresso diário, um adendo ainda maior, na quarta e na quinta ação, foi a criação de uma tabela

de progresso semanal, assim como mensal e anual do usuário, que permite ver o desempenho dele ao longo do tempo.

3 Levantamento de Requisitos

No decorrer do projeto, foram selecionadas as ferramentas atendendo aos critérios de eficiência e adaptação ao código base e as premissas do projeto. Essas ferramentas, em grande maioria, foram abordadas ao longo do curso, com diversas documentações, tutoriais e recursos na comunidade de desenvolvedores, o que auxiliou na implementação e procura de soluções.

A escolha das ferramentas também teve influência pessoal do desenvolvedor. As licenças das ferramentas são de código aberto, podendo ser utilizadas para fins comerciais e pessoais. Os documentos gerados são a Matriz SWOT, Plano de Ação 5W2H, BPMN do projeto, Casos de Uso, documentação, diagramas e protótipo do projeto.

3.1 Elicitação e especificação dos Requisitos

Segundo Martins (2001), através da elicitação de requisitos, é possível compreender as necessidades do usuário que devem ser atendidas pela aplicação, identificando os maiores obstáculos entre os desenvolvedores e os usuários.

No levantamento de requisitos, foram utilizadas as técnicas de pesquisa e observação para identificar falhas ou até complexidades em outras aplicações disponíveis no mercado, a fim de criar uma aplicação com maior simplicidade na utilização e eficiência nos processos.

Para melhor visualização do produto, foram realizadas prototipagens das telas iniciais com o intuito de nortear o desenvolvimento da aplicação e fazendo as alterações necessárias.

3.2 BPMN

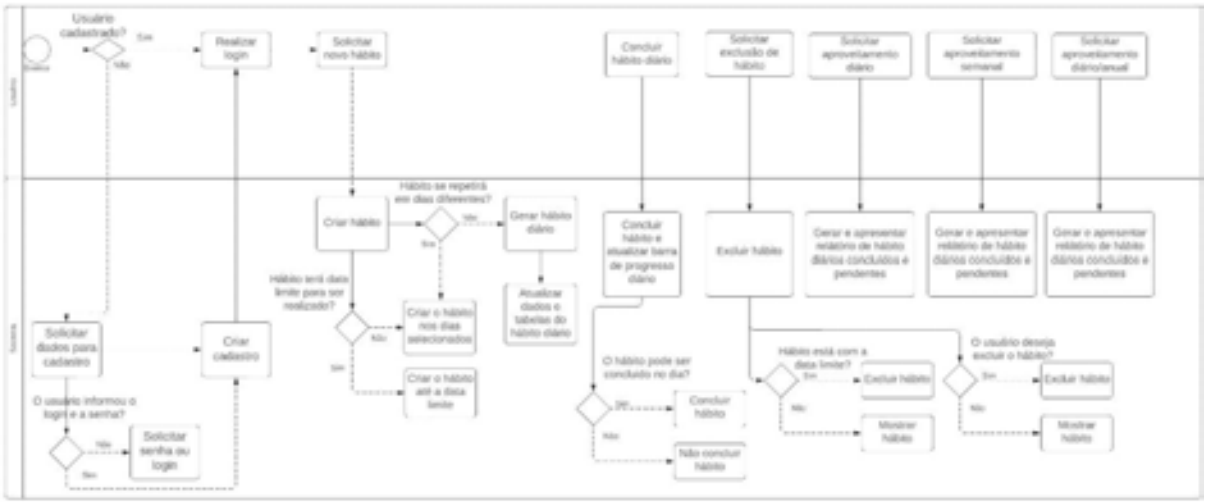
De acordo com a Controladoria Geral da União (CGU) (2020), *Business Process Model and Notation* (BPMN), cujo significado é Notação de Gerenciamento de Processos de Negócio, é um padrão desenvolvido pela *Business Process Management Initiative* (BPMI) e atualmente é mantido pelos órgãos responsáveis por estabelecer normas para sistemas de informações.

Trata-se de uma notação gráfica com símbolos padronizados para simplificar e colocar o projeto de uma maneira compreensível para profissionais de negócios e para profissionais da tecnologia, auxiliando na comunicação e compreensão dos processos

dentro de empresas. Para viabilizar a criação do diagrama, foi seleciona a ferramenta *Lucidchart* um aplicativo de diagramação inteligente.

O BPMN fala sobre o funcionamento do sistema o processo de criação de usuário, ilustrado na Figuras 2.

Figura 2: Processo BPMN do Sistema



Fonte: Elaborado pelo autor

3.3 Requisitos Funcionais

A criação dos Requisitos Funcionais é uma etapa extremamente importante dentro da Engenharia de Software para o desenvolvimento de projetos, pois, com eles, é possível definir a base do projeto, suas implementações, testes e validação de soluções. A sua principal contribuição é definir a meta de utilização do recurso e do comportamento que o sistema deverá ter de acordo com o processamento das entradas e saídas das informações.

Para *Sommerville* (2011), os Requisitos Funcionais são declarações de serviços que o sistema deve fornecer, sobre como o sistema deve reagir a entradas específicas e como deve se comportar em determinadas situações. Além disso, os requisitos funcionais também podem explicitar o que o sistema não deve fazer. Com isso em mente e a partir do que foi estudado ao longo do curso, nota-se que os requisitos descrevem as funções e operações que o sistema realizará.

No Quadro 4, são abordadas as funções sistêmicas planejadas para o software deste estudo, incluindo acesso ao sistema, cadastro de usuário, cadastro de hábitos, edição de hábitos e emissão de um controle diário.

Quadro 4: Requisitos Funcionais do sistema

RF001 - Cadastro de Usuários	Categoria: () Oculto (X)Evidente	Prioridade: (X) Altíssima () Alta () Média () Baixa
Descrição: deverá ser feito um cadastro de usuários com um nome de usuário e senha para acesso ao sistema.		
RF002 - Alterar a Senha	Categoria: () Oculto (X) Evidente	Prioridade: (X) Altíssima () Alta () Média () Baixa
Descrição: deverá ser possível, a recuperação de senha de um cadastro, de acordo com a solicitação do usuário.		
RF003 - Marcação de hábitos como completos/incompletos	Categoria: () Oculto (X) Evidente	Prioridade: (X) Altíssima () Alta () Média () Baixa
Descrição: o usuário pode marcar ou desmarcar hábitos como concluídos para o dia selecionado.		
RF004 - Exibição de hábitos	Categoria: () Oculto (X) Evidente	Prioridade: (X) Altíssima () Alta () Média () Baixa
Descrição: o sistema deve exibir uma lista de hábitos diários para o usuário com base na data atual.		
RF005 - Criar hábito	Categoria: () Oculto (X) Evidente	Prioridade: (X) Altíssima () Alta () Média () Baixa
Descrição: deverá ser possível a criação de um novo hábito que possua um título e as datas que ele se repetirá, de acordo com a solicitação do usuário.		
RF006 – Concluir hábito	Categoria: () Oculto (X) Evidente	Prioridade: (X) Altíssima () Alta () Média () Baixa
Descrição: deverá ser possível a marcar hábitos como concluídos para o dia selecionado		
RF007 - Excluir Hábito	Categoria: () Oculto (X) Evidente	Prioridade: (X) Altíssima () Alta () Média () Baixa
Descrição: a exclusão do hábito só será permitida pelo sistema para hábitos do dia atual e de dias futuros.		
RF008 - Exibir Progresso Diário	Categoria: () Oculto (X) Evidente	Prioridade: () Altíssima (X) Alta () Média () Baixa
Descrição: deverá ser possível exibir o progresso diário de hábitos através de uma seleção na página inicial, distinguindo hábitos realizados dos não realizados.		
RF009 - Exibir Progresso Semanal	Categoria: () Oculto (X) Evidente	Prioridade: () Altíssima () Alta (X) Média () Baixa
Descrição: deverá ser exibido o progresso semanal de hábitos.		

RF010 - Exibir Progresso Mensal	Categoria: () Oculto (X) Evidente	Prioridade: () Altíssima () Alta (X) Média () Baixa
Descrição: deverá ser exibido o progresso mensal de hábitos.		
RF010 - Exibir Progresso Anual	Categoria: () Oculto (X) Evidente	Prioridade: () Altíssima () Alta (X) Média () Baixa
Descrição: deverá ser exibido o progresso anual de hábitos.		

Fonte: Elaborado pelo autor

3.4 Requisitos Não Funcionais

Requisitos Não Funcionais, segundo *Sommerville* (2011), são restrições aos serviços ou funções oferecidos pelo sistema. Incluem restrições de *timing*, restrições no processo de desenvolvimento e restrições impostas pelas normas. Ao contrário das características individuais ou dos serviços do sistema, os requisitos não funcionais, muitas vezes, aplicam-se ao sistema como um todo. Entende-se que os Requisitos Não Funcionais descrevem características do projeto que não estão diretamente relacionadas às funcionalidades que ele executará, como segurança, desempenho, usabilidade, entre outros.

No Quadro 5, são abordadas as funcionalidades planejadas para a aplicação, incluindo recuperação de senha, restrição do design das telas, utilização de internet e o acesso por navegadores.

Quadro 5: Requisitos Não Funcionais do sistema

RNF001-Desempenho	A aplicação deve carregar e exibir os dados de hábitos rapidamente, mesmo em tabelas mensais ou semanais longas.	Tipo	() Desejável (X) Obrigatório	(X) Permanente () Transitório
RNF002-Usabilidade	A interface deve ser intuitiva, permitindo ao usuário marcar ou desmarcar hábitos com facilidade e visualizar seu progresso.	Tipo	() Desejável (X) Obrigatório	(X) Permanente () Transitório
RNF003-Utilizar internet	O sistema somente funcionará caso o usuário esteja conectado à internet.	Tipo	() Desejável (X) Obrigatório	(X) Permanente () Transitório
RNF004-Segurança	As rotas de autenticação e modificação de hábitos devem ser protegidas por mecanismos de segurança, como autenticação de token do usuário.	Tipo	() Desejável (X) Obrigatório	(X) Permanente () Transitório
RNF005-Responsividade	A aplicação deve ser responsiva, adaptando-se bem a diferentes tamanhos de tela (mobile first).	Tipo	() Desejável (X) Obrigatório	(X) Permanente () Transitório

RNF006-Escalabilidade	A aplicação deve suportar um grande número de usuários sem comprometer o desempenho.	Tipo	(X) Desejável () Obrigatório	(X) Permanente () Transitório

Fonte: Elaborado pelo autor

3.5 Regras de Negócio

Para *HISATOMI* et. alt. (2014), as regras de negócio são responsáveis por traduzir as necessidades que o projeto apresenta, trazendo tais informações com regras lógicas e fáceis de visualizar. Sua principal importância é servir como um guia para a equipe de desenvolvimento, garantindo uma boa interpretação e aplicação de tais regras ao longo do projeto.

De acordo com o objetivo das Regras de Negócio, é importante atentar-se a como se expressar, determinar as limitações e procedimentos que serão utilizados e garantir que todas as informações e procedimentos estejam disponíveis de forma sucinta e clara a todos os integrantes do projeto, desde os desenvolvedores até os próprios usuários do sistema. O Quadro 6 traz as regras de negócios pensadas para o melhor funcionamento da aplicação.

Quadro 6: Regras de Negócio do sistema.

RN001 – Bloqueio de dias passados
Descrição: O sistema não permite que o usuário marque ou altere hábitos de dias passados, apenas os exibe como finalizados ou não.
RN002 – Acesso ao site
Descrição: O sistema deverá restringir acesso dos usuários não cadastrados.
RN003 – Criar hábitos
Descrição: Cada hábito criado deve ser adicionado nos dias selecionados e aparecer na barra de progresso.
RN004 – Excluir hábitos
Descrição: Quando um hábito é excluído, ele deve ser removido da lista de possíveis hábitos e das rotinas diárias associadas.
RN005 – Conclusão dinâmica de hábitos
Descrição: Um hábito só pode ser marcado como completo para o dia atual, e o progresso é atualizado em tempo real com base na conclusão dos hábitos.
RN006 – Rotinas criadas por usuário
Descrição: Cada rotina está vinculada a um usuário específico, e somente o usuário que a criou pode completá-la ou excluí-la.

Fonte: Elaborado pelo autor

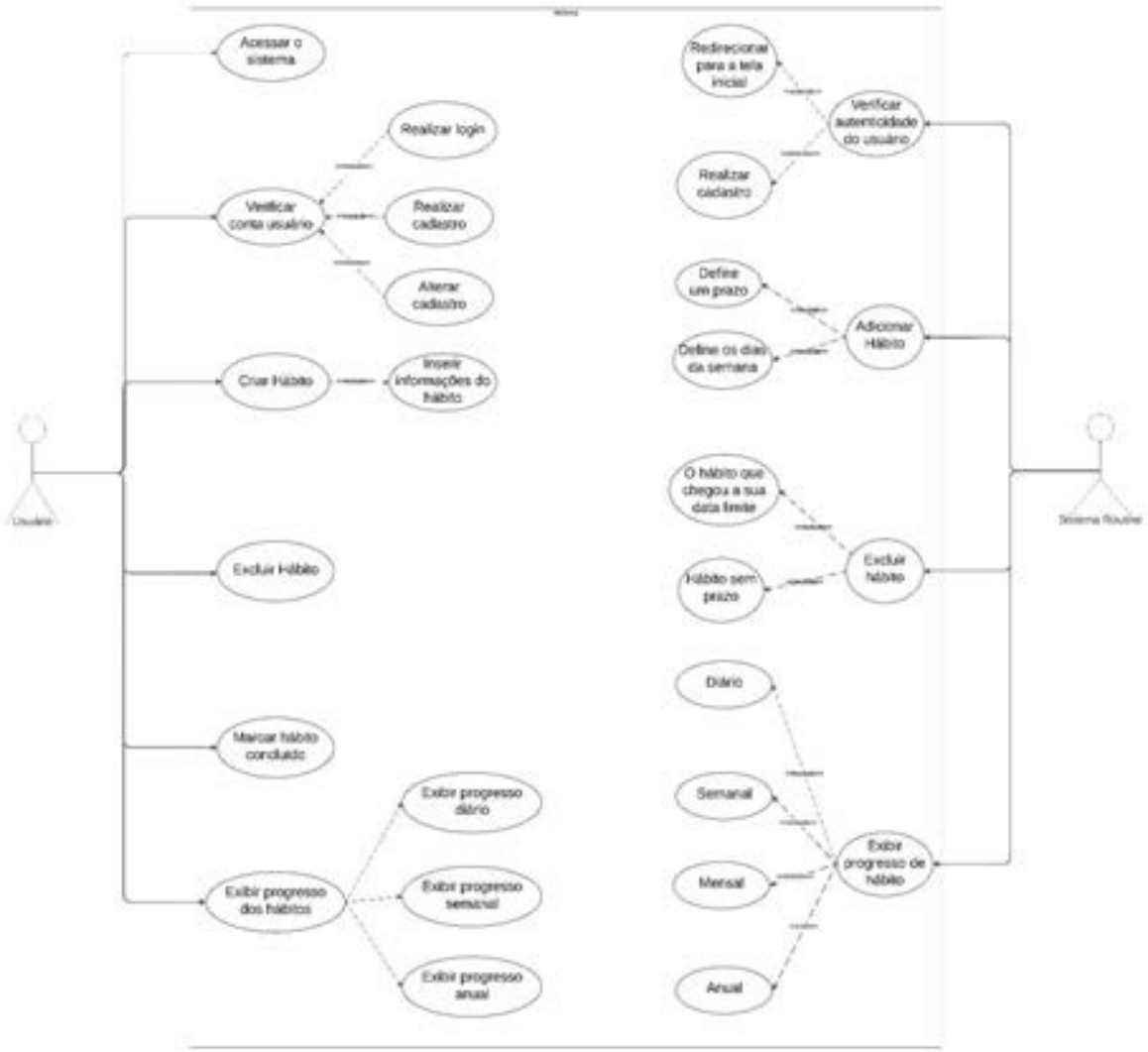
3.6 Casos de Uso

Ainda segundo *HISATOMI* et. alt. (2014), o Diagrama de Caso de Uso, um dos primeiros diagramas criados, desempenha um papel crucial no ciclo de vida do

desenvolvimento de um sistema, contribuindo diretamente na economia de tempo e recursos, além de fornecer uma visão geral das interações entre os usuários e o sistema. Este diagrama é responsável por representar visualmente as interações, e sua real importância está na documentação destas interações que, caso seja bem-feita, facilita o entendimento da funcionalidade do sistema para as partes interessadas, como desenvolvedores e analistas do projeto, podendo auxiliar, inclusive, na documentação dos requisitos do sistema.

Devido a sua importância dentro da ER (Engenharia de Requisitos) e a uma necessidade para o projeto, a Figura 4 e o Quadro 6 apresentam, respectivamente, o Diagrama de Casos de Uso e sua documentação detalhada.

Figura 3: Diagrama de Caso de Uso



Fonte: Elaborado pelo autor

Quadro 6: Use Case

Caso de Uso – Criar cadastro de usuário	
ID	UC 001
Descrição	Este caso de uso tem por objetivo cadastrar os dados do usuário no sistema
Ator Primário	Usuário do sistema
Pré-condição	Nenhuma
Cenário Principal	1. O use case inicia quando o usuário inicia a aplicação. 2. A aplicação irá carregar a tela de login/cadastro automaticamente. 3. O usuário irá selecionar a aba “criar conta” e deverá adicionar um nome e senha. 4. O sistema irá validar os dados para a criação do usuário. 5. O sistema encerra o caso de uso.
Pós-condição	Não possui.
Cenário Alternativo	4a – O usuário deixa de informar algum dado para a criação do cadastro 4a.1 O sistema identifica o campo e retorna uma mensagem para que ele seja preenchido.
Caso de Uso – Usuário já cadastrado	
ID	UC 002

Descrição	Este caso de uso tem por objetivo instruir o usuário a realizar o cadastro com outro usuário realizar login
Ator Primário	Usuário do sistema
Pré-condição	O usuário já ser cadastrado no sistema
Cenário Principal	1. O use case inicia quando o usuário tenta realizar um cadastro com um usuário já cadastrado 2. O sistema irá validar os dados e negar a criação do cadastro 3. O usuário deverá selecionar entre “realizar login” ou “esqueci minha senha” 4. O sistema encerra o caso de uso
Pós-condição	Não possui.
Cenário Alternativo	Não possui
Caso de Uso – Esqueci minha senha	
ID	UC 003
Descrição	Este caso de uso tem por objetivo realizar a alteração da senha do cadastro do usuário
Ator Primário	Usuário do sistema
Pré-condição	O usuário já ser cadastrado no sistema
Cenário Principal	1. O use case inicia quando o usuário seleciona a opção “Esqueci minha senha”. 2. O sistema irá validar o usuário da conta e seguir para a página de recuperação de senha. 3. O usuário realizará a alteração pelo sistema. 4. O sistema encerra o caso de uso.
Pós-condição	Não possui.
Cenário Alternativo	Não possui
Caso de Uso – Realizar login	
ID	UC 004
Descrição	Este caso de uso tem por objetivo fazer login do usuário.
Ator Primário	Usuário
Pré-condição	O usuário já ser cadastrado no sistema
Cenário Principal	1. O use case inicia quando o usuário abre a aplicação. 2. O sistema irá carregar a tela de login automaticamente 3. O usuário deverá inserir o nome de usuário e logo abaixo sua senha 4. Após essas etapas é necessário clicar no botão “Entrar” 5. O sistema irá validar os dados para o acesso 6. O sistema encerra o caso de uso
Pós-condição	Não possui.
Cenário Alternativo	Não possui
Caso de Uso – Criar hábito	
ID	UC 005
Descrição	Este caso de uso tem por objetivo criar o hábito na aplicação.
Ator Primário	Usuário
Pré-condição	O usuário possuir um cadastro.
Cenário Principal	1. O use case inicia quando o usuário abre o site. 2. O sistema irá carregar a tela dos *Routines*. 3. O usuário irá selecionar a opção “novo hábito”. 4. O usuário deverá preencher o nome 5. O usuário deverá preencher os dias que deseja realizar o hábito 6. O sistema irá validar os dados para a criação do hábito 7. O sistema encerra o caso de uso

Pós-condição	Não possui.
Cenário Alternativo	5a – O usuário poderá criar metas de curto prazo selecionando apenas um dia para a repetição da tarefa. 5b – O usuário poderá criar metas de longo prazo selecionando os dias que ela se repetirá.
Caso de Uso – Definir metas de curto prazo	
ID	UC 006
Descrição	Este caso de uso tem por objetivo criar um hábito de curto prazo
Ator Primário	Usuário
Pré-condição	O usuário possuir um cadastro.
Cenário Principal	1. O use case inicia quando o usuário seleciona “novo hábito” 2. O sistema irá carregar a tela de “Criar hábito” 3. O usuário deverá preencher o nome e a recorrência do hábito 4. O sistema irá validar os dados para a criação do hábito 5. O sistema encerra o caso de uso
Pós-condição	Não possui.
Cenário Alternativo	Não possui
Caso de Uso – Definir metas de longo prazo	
ID	UC 007
Descrição	Este caso de uso tem por objetivo criar um hábito de longo prazo
Ator Primário	Usuário
Pré-condição	O usuário possuir um cadastro.
Cenário Principal	6. O use case inicia quando o usuário seleciona “novo hábito” 7. O sistema irá carregar a tela de “Criar hábito” 8. O usuário irá preencher o nome e a recorrência do hábito 9. O sistema irá validar os dados para a criação do hábito 10. O sistema encerra o caso de uso
Pós-condição	Não possui.
Cenário Alternativo	Não possui
Caso de Uso – Excluir hábito	
ID	UC 008
Descrição	Este caso de uso tem por objetivo excluir um hábito
Ator Primário	Usuário
Pré-condição	O usuário deve ter um hábito cadastrado
Cenário Principal	1. O use case inicia quando o usuário seleciona a exibição dos hábitos diários. 2. O usuário irá selecionar a opção “lixeira” 3. O sistema processará a exclusão do hábito 4. Caso cumpra as regras de negócio, o sistema deverá excluir o hábito. 5. O sistema encerra o caso de uso
Pós-condição	Não possui
Cenário Alternativo	3a – Caso o hábito seja de longo prazo, a exclusão de um dos hábitos acarretará a exclusão dos seguintes.
Caso de Uso – Concluir hábito	
ID	UC 009
Descrição	Este caso de uso tem por objetivo concluir um hábito
Ator Primário	Usuário
Pré-condição	O usuário deve ter um hábito cadastrado
Cenário Principal	1. O use case inicia quando o usuário realiza a tarefa do hábito

	2. O usuário irá selecionar a lista de hábitos diários 3. O usuário irá marcar o hábito como concluído 4. O sistema deverá receber e atualizar a barra de progresso 5. O sistema encerra o caso de uso
Pós-condição	Não possui
Cenário Alternativo	4a – Caso o hábito seja de longo prazo, a conclusão de um dos hábitos não modificará os hábitos de dias seguintes.
Caso de Uso – Visualizar progresso	
ID	UC 010
Descrição	Este caso de uso tem por objetivo visualizar o progresso do usuário quanto aos hábitos concluídos e não concluídos
Ator Primário	Usuário
Pré-condição	O usuário deve ter um hábito cadastrado
Cenário Principal	1. . O use case inicia quando o usuário seleciona a página inicial 2. O sistema irá carregar a tela de início junto às metas cadastradas 3. O sistema deverá consumir os dados de hábitos realizados e não realizados 4. O sistema deverá exibir de forma gráfica a progressão semanal dos hábitos realizados pelo usuário 5. O sistema encerra o caso de uso
Pós-condição	Não possui
Cenário Alternativo	Não possui
Caso de Uso – Visualizar metas diárias	
ID	UC 011
Descrição	Este caso de uso tem por objetivo acompanhar a barra de progresso do usuário em um dia específico
Ator Primário	Usuário
Pré-condição	O usuário deve ter um hábito cadastrado
Cenário Principal	1. O use case inicia quando o usuário seleciona um dia 2. O sistema consumirá os hábitos cadastrados no dia selecionado 3. O sistema deverá mostrar uma barra de progresso de acordo com a conclusão dos hábitos pelo usuário 4. O sistema encerra o caso de uso
Pós-condição	Não possui
Cenário Alternativo	Não possui
Caso de Uso – Visualizar metas mensais	
ID	UC 012
Descrição	Este caso de uso tem por objetivo acompanhar a tabela de progresso do usuário em mês em específico
Ator Primário	Usuário
Pré-condição	O usuário deve ter um hábito cadastrado
Cenário Principal	5. O use case inicia quando o usuário seleciona a página inicial 6. O sistema consumirá os hábitos cadastrados no mês selecionado 7. O sistema deverá mostrar uma tabela de progresso 8. O sistema encerra o caso de uso
Pós-condição	Não possui
Cenário Alternativo	Não possui
Caso de Uso – Visualizar metas anuais	

ID	UC 013
Descrição	Este caso de uso tem por objetivo acompanhar a barra de progresso do usuário em um dia específico
Ator Primário	Usuário
Pré-condição	O usuário deve ter um hábito cadastrado
Cenário Principal	9. O use case inicia quando o usuário seleciona a página de acompanhamento mensal 10. O sistema consumirá os hábitos cadastrados no ano 11. O sistema deverá mostrar uma tabela de progresso 12. O sistema encerra o caso de uso
Pós-condição	Não possui
Cenário Alternativo	Não possui

Fonte: Elaborado pelo autor

4 Ferramentas, Métodos e Prototipação de Telas

O sistema foi desenvolvido utilizando a linguagem de programação *TypeScript*, com a utilização de Frameworks, como *React*, *Node.js* e *Tailwind CSS*. A modelagem de dados foi feita a partir do Prisma, que simplifica a interação com o banco de dados *My SQLlite*.

As ferramentas escolhidas para o projeto foram selecionadas, inicialmente, a partir do curso no qual o projeto se baseia, porém, adaptadas de acordo com as necessidades que foram surgindo ao longo do desenvolvimento. A escolha das ferramentas foi baseada em sua eficiência, facilidade de manejo e em um grande repertório de documentações, recursos e suporte a comunidades, disponíveis na internet. Nos próximos parágrafos, serão apresentadas as tecnologias escolhidas e suas extensões, assim como o motivo para tal seleção.

Para lidar com os dados da aplicação, foi escolhida a ferramenta *Prisma* (versão 5.3.1), uma ferramenta *ORM* (*Object Relational Mapping*) responsável por facilitar a interação do *TypeScript*/*Node.js* com o banco de dados, deixando a cargo dos desenvolvedores a definição dos modelos e o esquema de dados através do arquivo *schema.prisma*, além de realizar migrações ao banco de dados de forma detalhada e versionada, podendo acessar atualizações passadas. A ferramenta foi escolhida por contar com diversas funcionalidades que simplificam a visualização da interação com o banco de dados, tal como o pacote *Prisma Client*, que fornece uma maneira segura de interagir com os dados cadastrados.

A biblioteca *React* (v18.2.0) é a base do *Front-End* da aplicação. De acordo com *React.dev* (s.d), essa famosa biblioteca em *JavaScript* fornece diversos recursos

de tipagem estática, que são potencializados com a utilização do *TypeScript* e com componentes que podem ser reutilizados ao longo do código. A biblioteca em questão foi escolhida por sua vasta rede de suporte e recursos disponíveis na internet, tendo um auxílio gigantesco da comunidade que a utiliza.

Como essa aplicação trabalha em exibição na *Web*, foi necessária a utilização da biblioteca *React DOM*, que é usada para renderizar os componentes *React* no navegador e, para facilitar o desenvolvimento do código, também foi adicionado o *Plugin Vite.js/plugin-react*, que tem por função otimizar a construção do código da aplicação por permitir a atualização automática, na *Web,* das alterações realizadas no código, sem a necessidade de recarregar a página, tornando a visualização dos efeitos da mudança do código mais fácil.

Outra biblioteca utilizada dentro do *React* foi a *Phosphor React* (v1.4.1), composta por um conjunto de ícones de códigos abertos que foram projetados com o intuito de serem modernos, trazendo maior sofisticação para a aparência da aplicação.

Como linguagem de programação, foi selecionada a *TypeScript* (versão 4.9.3). Essa linguagem, de acordo com Typescriptlang.org (2012), é desenvolvida pela Microsoft, que adiciona recursos de tipagem estática à linguagem *JavaScript*, permitindo, aos desenvolvedores, especificar variáveis, propriedades de objetos, entre outros. Sua principal diferença é justamente a tipagem estática que permite detectar erros de complicação antes mesmo de sua execução, auxiliando no desenvolvimento da aplicação. A escolha desta linguagem se deu por ela ser crescente no mercado atual, com grande quantidade de documentos e bibliotecas disponíveis, além de ter uma manutenção mais fácil e tornar o código mais claro e legível.

Para a estilização das telas e plugins contidos na aplicação, foi selecionada a biblioteca *Tailwind* CSS (versão 3.2.4). Segundo *Tailwindcss* (s.d), este *framework* CSS permite criar layouts para as aplicações de forma personalizável, fornecendo componentes para sua estilização. Uma das principais características é a extensa coleção de CSS utilitárias para a construção de estilos de interfaces, sendo maleável de acordo com o interesse do desenvolvedor. Essa tecnologia foi escolhida por permitir maior personalização e por consumir pequeno espaço de armazenamento se comparado a outros *frameworks*, cumprindo o propósito da aplicação e elevando sua eficiência e agilidade, além de possuir uma documentação extremamente detalhada, facilitando a busca por soluções.

A necessidade de gerenciar as solicitações HTTP e lidar com as políticas de *Cross Origin Resource Sharing,* levou à escolha do *Fastify* (versão 4.12.0). Segundo o *Fastify* (2016), essa ferramenta é um *framework* para linguagem *JavaScript* ambientado com o Node.js, que é conhecido pelo seu desempenho e eficiência por utilizar uma maneira otimizada para lidar com as solicitações HTTP, resultando em um processamento mais rápido. Também conta com diversos *plugins*, o que permite adicionar funcionalidades ao software sem demandar muito esforço. Uma das extensões mais utilizadas foi a *Fastify* CORS. Esse *plugin* é um mecanismo de segurança responsável por controlar recursos Web de domínios diferentes. Neste caso, ele foi utilizado para lidar com as políticas de *Cross Origin Resource Sharing*, permitindo que os usuários acessem o *back-end* de diferentes origens.

Ainda para atender as solicitações HTTP, foi utilizada a biblioteca *Axios* (versão 1.5.0). Segundo a biblioteca *Axios* (2016), ela é uma ferramenta versátil para fazer requisições HTTP e é conhecida pela simplicidade em lidar com certos recursos avançados como o cancelamento de requisições e o suporte a promessas. Ela trabalha de maneira eficiente enviando promessas, facilitando o trabalho com operações assíncronas e tratativas com requisições e respostas antes de serem manipuladas. Essa biblioteca foi utilizada tanto no desenvolvimento do *back-end* quanto no *front-end*, trabalhando com requisições de dados nas duas frentes.

Por ser uma ferramenta voltada ao gerenciamento de tempo e hábitos, houve a necessidade da utilização da biblioteca Day.js (versão1.11.17). Segundo Day.js (s.d), essa biblioteca minimalista serve para a manipulação de datas e horas no *back-end*, uma alternativa poderosa por sua simplicidade, além de ter um arquivo significativamente menor e uma melhor performance em relação a outras disponíveis. A biblioteca foi escolhida por lidar de maneira diferente com as operações, em que retorna novos objetos sem modificar outros existentes, além de contar com suporte a sistemas de *Plugins*, estendendo algumas de suas funcionalidades.

Além do Day.js, foram utilizadas outras bibliotecas para gerenciar datas de forma eficiente no sistema.

A biblioteca *date-fns* (versão 4.0) foi escolhida devido ao seu suporte completo para manipulação de datas com fusos horários, algo fundamental para garantir precisão em diferentes regiões. De acordo com *date-fns* (s.d), essa biblioteca oferece uma variedade de funções para manipulação de datas, mantendo uma abordagem modular e funcional que facilita a manutenção e extensibilidade do código. Ela se

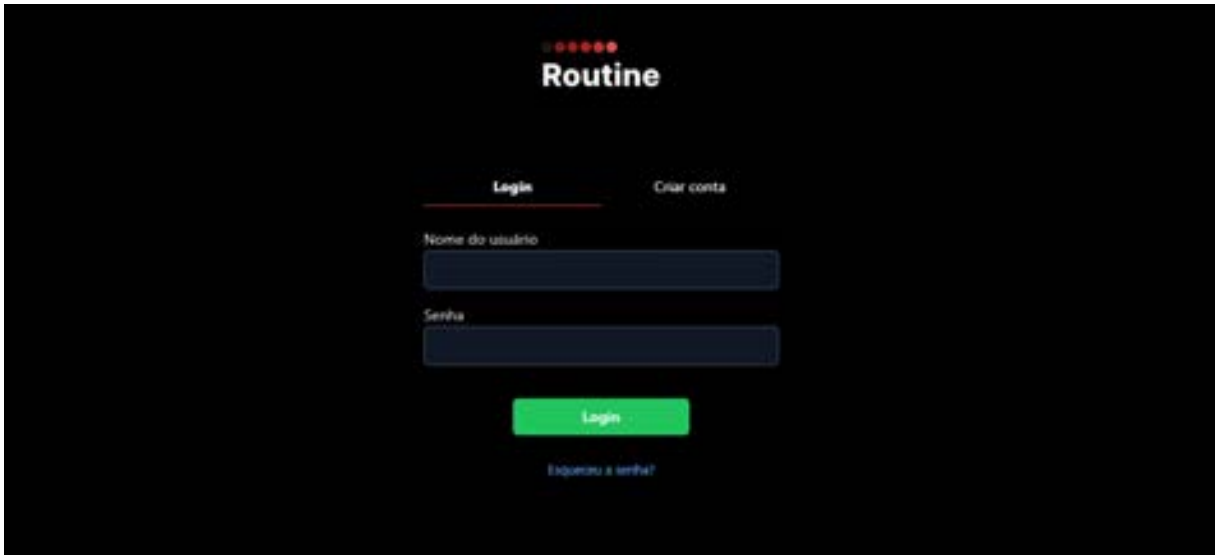
destaca por sua estrutura leve e independente de classes, tornando as operações mais diretas e menos propensas a erros de mutabilidade. Além disso, a biblioteca possui excelente performance e integrações com sistemas modernos de gerenciamento de tempo.

Outra ferramenta importante no projeto é o *React Datepicker*, um componente específico para seleção de datas em aplicações *React*. Esta biblioteca foi escolhida por sua interface intuitiva e altamente customizável, permitindo uma experiência de usuário mais amigável e flexível na seleção de datas para criação de hábitos e rotinas. De acordo com *React Datepicker* (s.d), o componente é projetado para funcionar bem com bibliotecas de manipulação de datas, como o *Day.js* e *date-fns*.

Para a validação e manutenção dos dados utilizados na ferramenta, foi selecionada a biblioteca *Zod* (versão 3.20.2). De acordo com *Zod.dev* (s.d), essa biblioteca é voltada ao TypeScript aproveitando ao máximo os recursos de tipagem que a linguagem oferece. É responsável por validar os dados recebidos pela aplicação, garantindo que sejam manipulados apenas dados corretos e que eles sigam o caminho e formatos esperados dentro do sistema. Além de todos os seus pontos positivos, essa biblioteca também suporta a composição de certos esquemas, permitindo ao desenvolvedor criar tipos e validações personalizadas para a solução do problema em questão.

Quanto à prototipação das telas, foram pensadas e desenvolvidas para o usuário visando maior simplicidade, garantindo uma usabilidade de fácil entendimento e maior fluidez, sem a necessidade de ficar trocando de telas diversas vezes. Dentro da tela de login, mostrada na figura 4, o usuário poderá criar um cadastro, realizar o acesso ao cadastro onde está salvo no banco de dados e poderá recuperar a senha do cadastro, caso tenha esquecido.

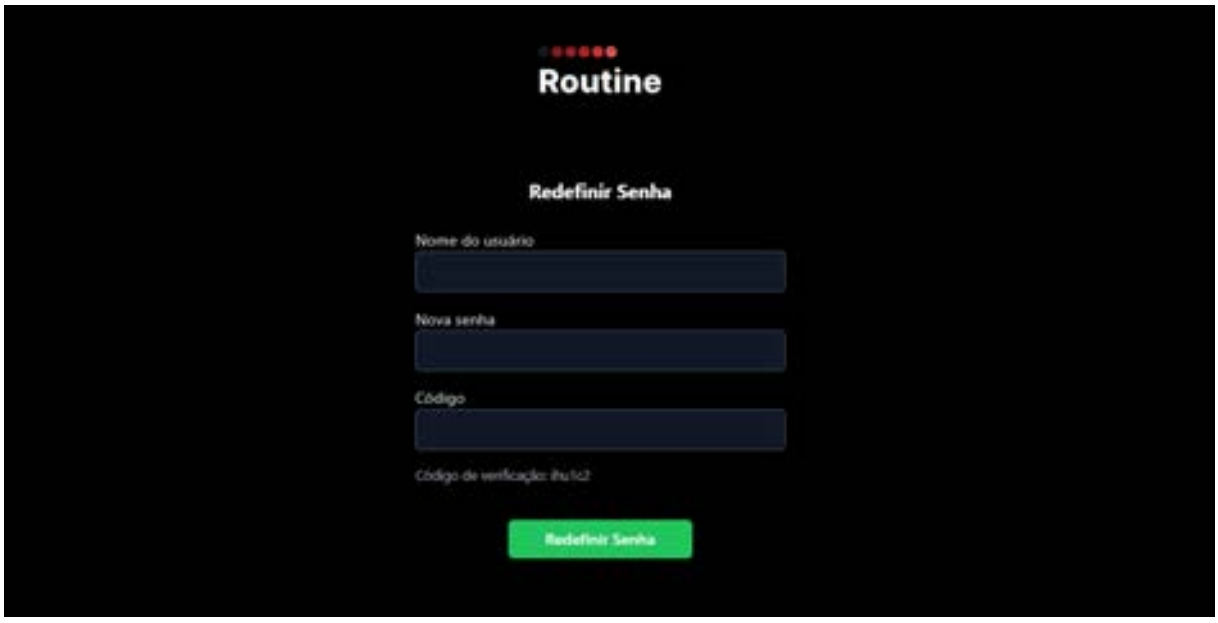
Figura 4: Tela de Login



Fonte: Elaborado pelo autor

Com o processo de login, é de extrema necessidade uma opção para a recuperação de senha dos cadastros criados. A opção é disponibilizada na tela de Login e, caso o usuário opte por recuperar a senha, ele será redirecionado a outra tela, exemplificada na figura 5, onde será necessário o seu nome de usuário e o código de verificação para realizar o processo.

Figura 5: Tela inicial



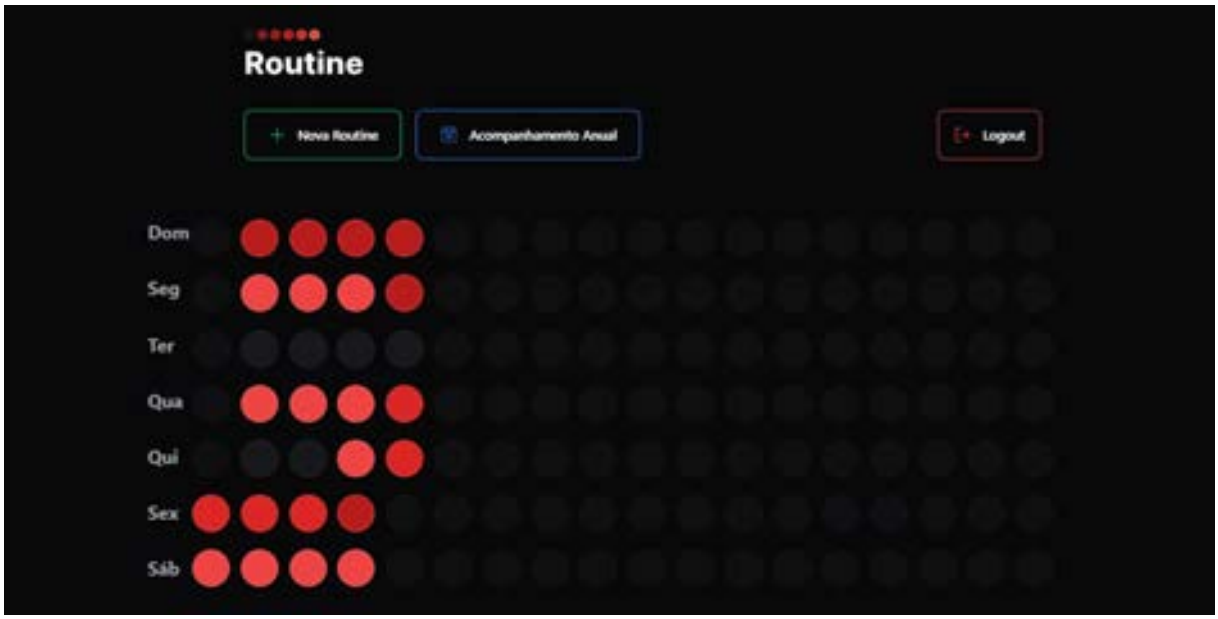
Fonte: Elaborado pelo autor

Após realizado o acesso ao cadastro, o usuário será direcionado à página principal do sistema, a tela inicial que é exibida na figura 6, onde o usuário fará a maior parte das interações oferecidas pela aplicação. Com a parte estética desenvolvida através de componentes, o usuário tem a opção de visualizar hábitos diários, e poderá acessar o menu para a criação de hábitos, para obter uma relação dos hábitos de

meses anteriores e sair de sua conta através de cliques na página inicial, exibida na figura 6.

O sistema exibirá os hábitos criados pelo usuário de três formas diferentes. A primeira é semanalmente, na própria tela inicial, onde há o preenchimento dos quadrados que se referem a um dia específico do mês, de acordo com o progresso de hábitos cumpridos pelo usuário. Quanto mais adesão aos hábitos criados, mais claro o quadrado ficará.

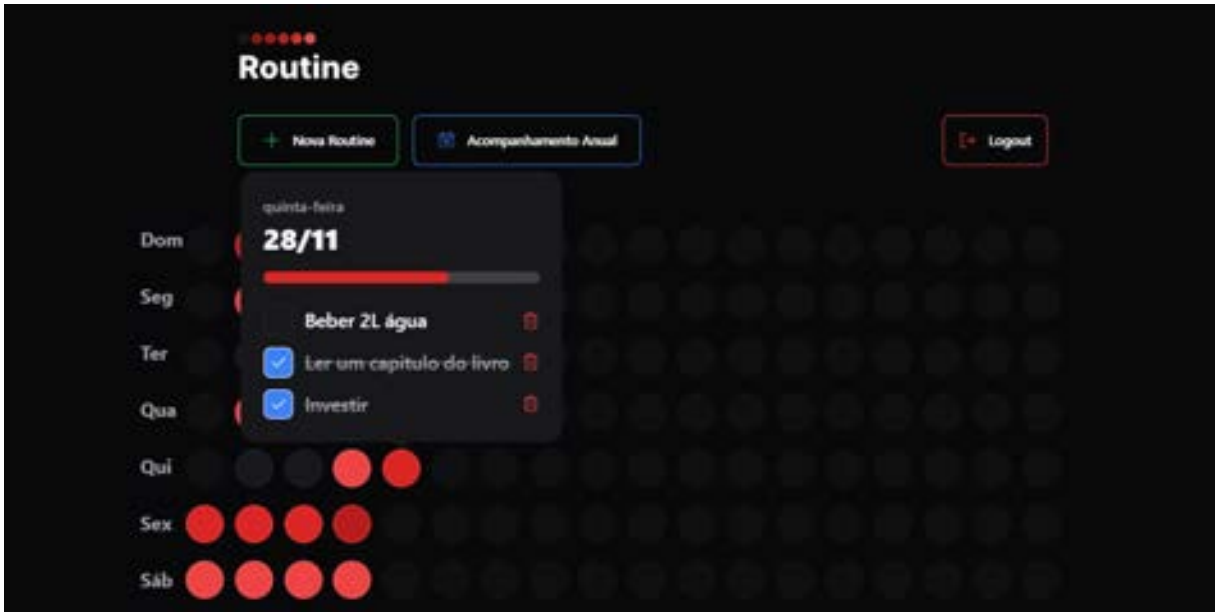
Figura 6: Tela inicial



Fonte: Elaborado pelo autor

A segunda forma de exibição será diária, que é exemplificada na figura 7. Para acessar o menu de visualização de hábitos diários, é necessário clicar no dia que deseja acessar. Dentro desse menu, é possível visualizar uma barra de progresso que progredirá quando o usuário completar um hábito. Ele também pode excluir um hábito, caso haja necessidade, desde que cumpra as regras de exclusão para hábitos do dia atual ou de dias posteriores.

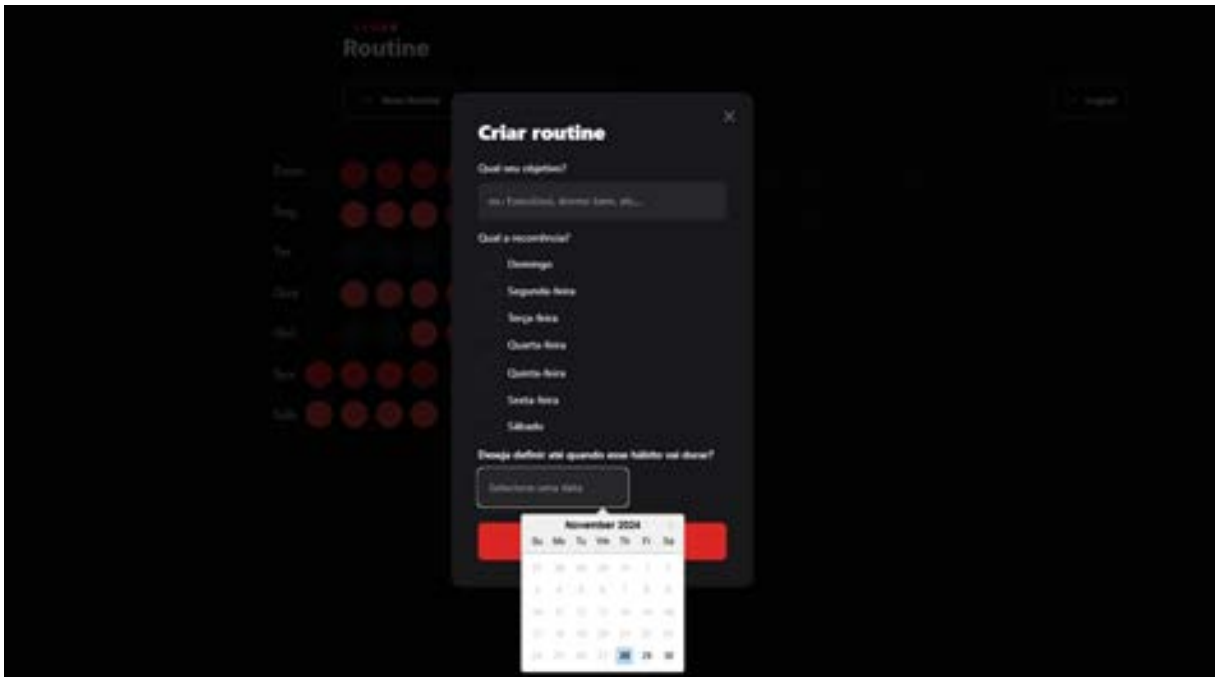
Figura 7: Visualização de Hábitos por dia



Fonte: Elaborado pelo autor

O usuário poderá criar hábitos de curto ou longo prazo de acordo com seu interesse. Para isso, é necessário incluir o objetivo, dias da semana que deverá repetir e quantas semanas irá durar, através de um calendário disponibilizado no menu, conforme mostrado na figura 8.

Figura 8: Menu de criação de Hábitos

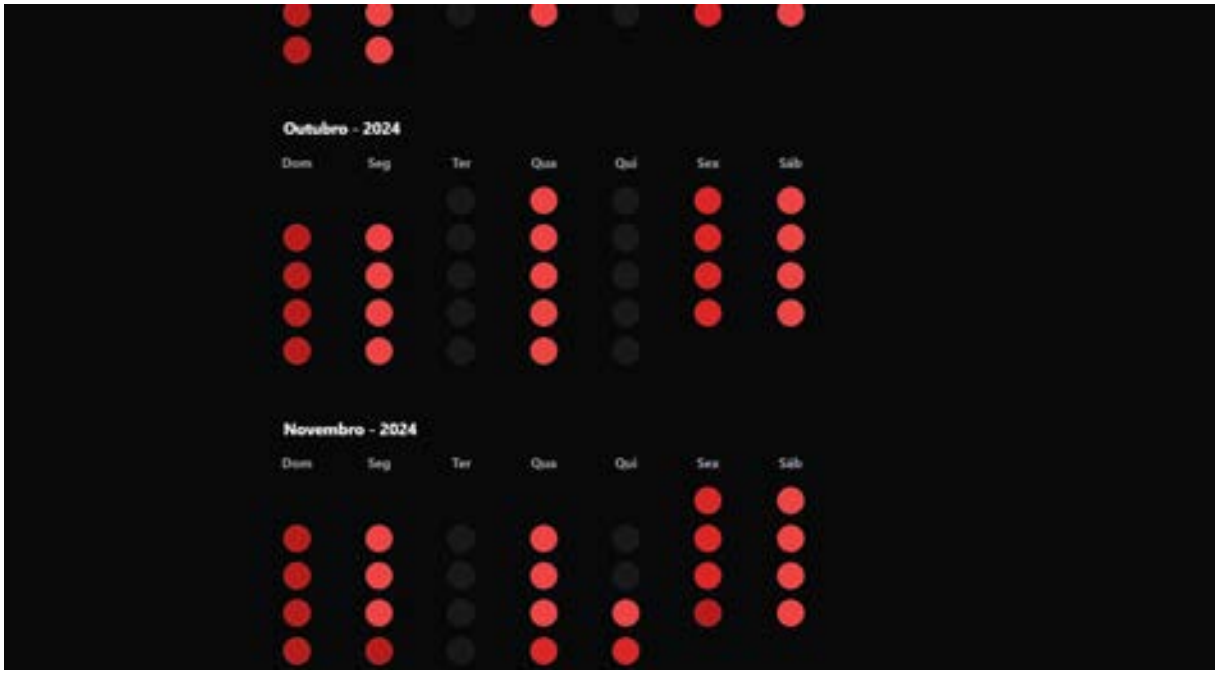


Fonte: Elaborado pelo autor

Verificou-se a necessidade de uma maneira de visualizar hábitos de meses passados para um maior controle das metas e rotinas do usuário. Para isso foi criado um botão chamado Acompanhamento Anual. Para essa funcionalidade foi utilizado

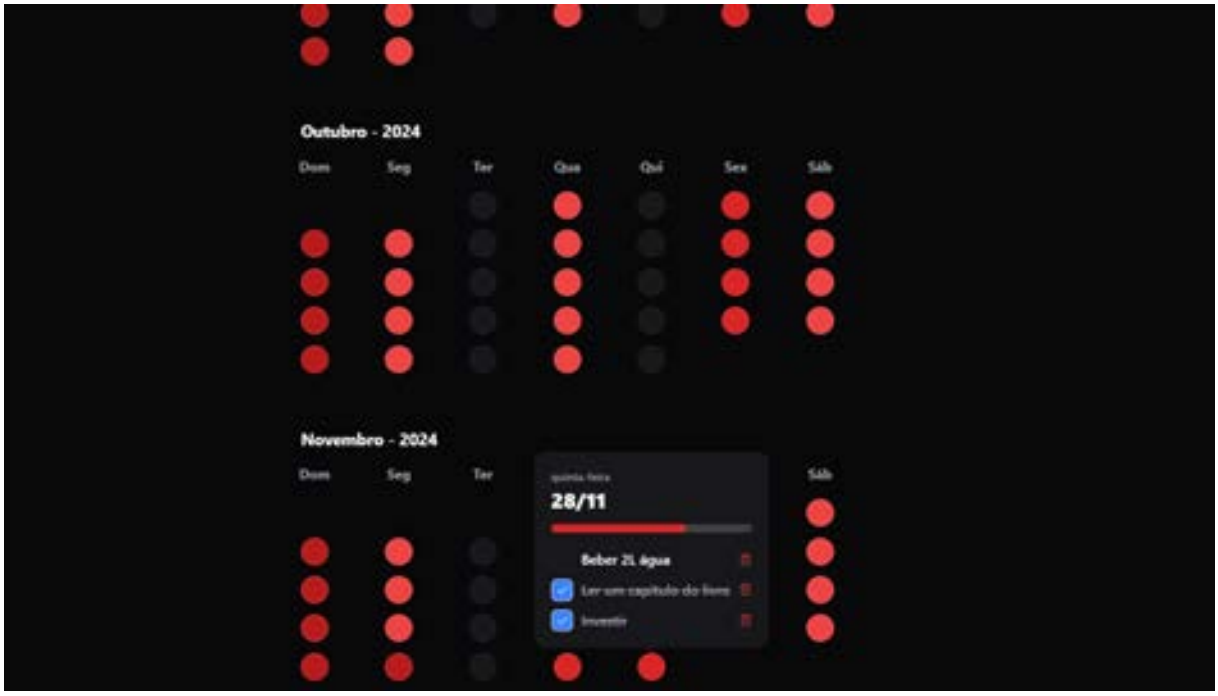
um layout similar à tela inicial, onde o usuário terá a visualização do mês e a opção de acessar a visualização diária, conforme exibido nas figuras 9 e 10.

Figura 9: Acompanhamento anual exibindo hábitos mensais



Fonte: Elaborado pelo autor

Figura 10: Acompanhamento anual exibindo hábitos diários



Fonte: Elaborado pelo autor

5 Desenvolvimento

Para iniciar o desenvolvimento do site, foi necessário entender como a programação seria feita e, para isso, foi utilizado o modelo de arquitetura de construção de sites *MVC (Model, View e Controller).* Esse modelo divide a construção da aplicação em três componentes principais, com responsabilidades distintas que conversam entre si através de rotas no desenvolvimento. O *Model* é responsável por representar a lógica dos dados da aplicação, então, ele acessa, manipula e gerencia todos os dados através de um banco de dados. O *View* é a camada responsável por receber os dados e apresentá-los ao usuário através de uma interface. Essa parte normalmente não agrega à lógica do negócio, apenas à parte visual. O *Controller* é responsável por intermediar o contato entre o *Model* e o *View*, atuando no recebimento das entradas de informações e processando-as, retornando o resultado esperado à *View* que será mostrado ao usuário.

O desenvolvimento teve início na parte do *Model*, onde foi adicionado o Prisma como banco de dados por sua maior volatilidade no desenvolvimento e facilidade de visualização e modificação das tabelas já criadas, colocando algumas *seeds* e dependências disponíveis da ferramenta. A primeira parte do código foi criar a tabela que receberá os *hábitos* criados pelo usuário através do arquivo *schema.prisma*, que é uma ferramenta do Prisma para modelar os dados que o banco irá receber.

As principais tabelas desenvolvidas são:

Routine: responsável por guardar os dados dos hábitos já criados e relaciona-se com as tabelas *dayRoutine* e *weekDays*. A Figura 11 exibe o código da tabela *Routine*.

Figura 11: Tabela Routine criada a partir do Schema.prisma

```
model Routine {
  id          String   @id @default(uuid())
  title       String
  created_at  DateTime
  user_id     String
  user User @relation(fields: [user_id], references: [id])

  dayRoutines DayRoutine[]
  weekDays  RoutineWeekDays[]

  @@map("routines")
}
```

Fonte: Elaborado pelo autor

DayRoutine: responsável por definir quando os hábitos devem aparecer para o usuário e retornar as informações das *Routines* já cadastrados para o dia. A Figura 12, mostra como a tabela foi criada.

Figura 12: Tabela DayRoutine criada a partir do Schema.prisma

```
model DayRoutine {
  id String @id @default(uuid())

  day_id   String
  routine_id String

  day   Day   @relation(fields: [day_id], references: [id])
  routine Routine @relation(fields: [routine_id], references: [id])

  @@unique([day_id, routine_id])
  @@map("day_routines")
}
```

Fonte: Elaborado pelo autor

RoutineWeekDays: responsável por guardar os dados dos hábitos da semana para o usuário, retornando quais semanas esses hábitos devem aparecer. A Figura 13 mostra como a tabela foi criada.

Figura 13: Tabela RoutineWeekDays criada a partir do Schema.prisma

```
model RoutineWeekDays {
  id       String @id @default(uuid())
  routine_id String
  week_day Int

  routine Routine @relation(fields: [routine_id], references: [id])

  @@unique([routine_id, week_day])
  @@map("routine_week_days")
}
```

Fonte: Elaborado pelo autor

Uma das maiores dificuldades encontradas foi testar as tabelas e rotas criadas no *back-end* com dados reais e, para isso, foi usada uma função disponível na ferramenta Prisma. A ferramenta para criação de *seeds*, responsável por popular o banco de dados com informações pré-definidas, podendo ser dados iniciais ou até dados complexos no desenvolvimento, sem ter a necessidade de adicionar dados

manualmente. A *seed* é de extrema importância, pois permite consultas de como o banco está agindo com a entrada de dados pelo usuário.

Com as tabelas criadas e o banco de dados funcional, teve início a criação das rotas que serão responsáveis por consumir os dados que estão no Banco de Dados. Esta modalidade é conhecida como o *Controller* da aplicação, assim, para cada tabela criada, é necessária uma rota para que a informação seja consumida.

A primeira rota criada foi a *Routine*, como mostra a figura 14. Dentro desta rota foram adicionadas diversas funções, como uma consulta SQL, responsável por povoar os hábitos criados como *Title* do Routine, e os dias que eles se repetirão.

Figura 14: Rota Routines

```
app.post('/user/:id/routines', async (request) => {
  const createRoutineBody = z.object({
    title: z.string(),
    weekDays: z.array(
      z.number().min(0).max(6)
    ),
  })

  const routineUser = z.object({
    id: z.string().uuid(),
  })

  const { title, weekDays } = createRoutineBody.parse(request.body)
  const { id } = routineUser.parse(request.params)

  const today = dayjs().startOf('day').toDate()

  await prisma.routine.create({
    data: {
      user_id: id,
      title,
      created_at: today,
      weekDays: {
        create: weekDays.map(weekDay => {
          return {
            week_day: weekDay,
          }
        }),
      }
    }
  })
})
```

Fonte: Elaborado pelo autor

Outra rota criada foi denominada “*Reset* p*assword para alteração de cadastro”*. Ela será responsável por receber o chamado de alteração de cadastro e atualizar os novos dados dentro do banco de dados, como mostra a figura 15.

Figura 15: Reset password para alteração de cadastro.

```
app.post('/reset-password', async (request, reply) => {
  const resetPasswordSchema = z.object({
    username: z.string(),
    newPassword: z.string(),
  });

  const { username, newPassword } = resetPasswordSchema.parse(request.body);

  const user = await prisma.user.findUnique({
    where: {
      username,
    },
  });

  if (!user) {
    reply.status(404).send('Usuário não encontrado');
    return;
  }

  await prisma.user.update({
    where: {
      username,
    },
    data: {
      password: newPassword,
    },
  });

  reply.send('Senha alterada com sucesso');
});
```

Fonte: Elaborado pelo autor

Finalizado o *Model* e *Controller*, teve início a criação da parte *View*, ou seja, o *front-end* da aplicação, sua parte estética, e que irá consumir todos os dados e as rotas criadas. A aplicação foi dividida em diferentes componentes que se relacionam: os dois primeiros componentes criados no *front-end* foram o *Header* e o *NewRoutineForm*, que, juntos, são responsáveis por criar as *routines* e guardar os dados adicionados pelo usuário. O primeiro, executará a parte visual para que o usuário tenha a opção de criar a *routine*; o segundo, é o responsável por receber e guardar esses dados, realizando a comunicação com o banco de dados através das rotas criadas. A Figura 16 mostra parte do código do *NewRoutineForm*.

Figura 16: Formulário para a criação de Routine

```
async function createNewRoutine(event: FormEvent) {
  event.preventDefault()

  if (!title || weekDays.length === 0) {
    return(
      alert('Digite o nome do seu Routine, e selecione um dia da semana!')
    )
  }

  console.log('User ID:', userId);

  await api.post(`/user/${userId}/routines`, {
    title,
    weekDays,
  })

  setTitle('')
  setWeekDays([])

  alert('Routine criado com sucesso!')

}

function handleToggleWeekDay(weekDay: number) {
  if(weekDays.includes(weekDay)) {
    const weekDaysWithRemovedOne = weekDays.filter(day => day !== weekDay)

    setWeekDays(weekDaysWithRemovedOne)
  } else {
    const weekDaysWithAddedOne = [...weekDays, weekDay]

    setWeekDays(weekDaysWithAddedOne)
  }
}
```

Fonte: Elaborado pelo autor

Após a criação dos dois componentes anteriores, houve a necessidade de buscar uma solução para lidar com os dados do usuário. Assim, foi criada a *RoutinesList*, um componente responsável por comunicar com o Banco de Dados e fazer uma busca de todas as *routines* já criados e disponíveis no banco.

O próximo módulo criado foi a *RoutineDay*, extremamente relevante por conter regras de negócio, como o fato de o usuário não conseguir excluir *routines* no dia atual ou não conseguir completar *routines* de dias anteriores, assim como é responsável por retornar as *routines* criadas para aquele determinado dia. Portanto, ele faz a separação entre todos as *routines* recebidas pelo componente anterior e exibe apenas os que são do dia em questão. A Figura 17 exibe o código deste componente.

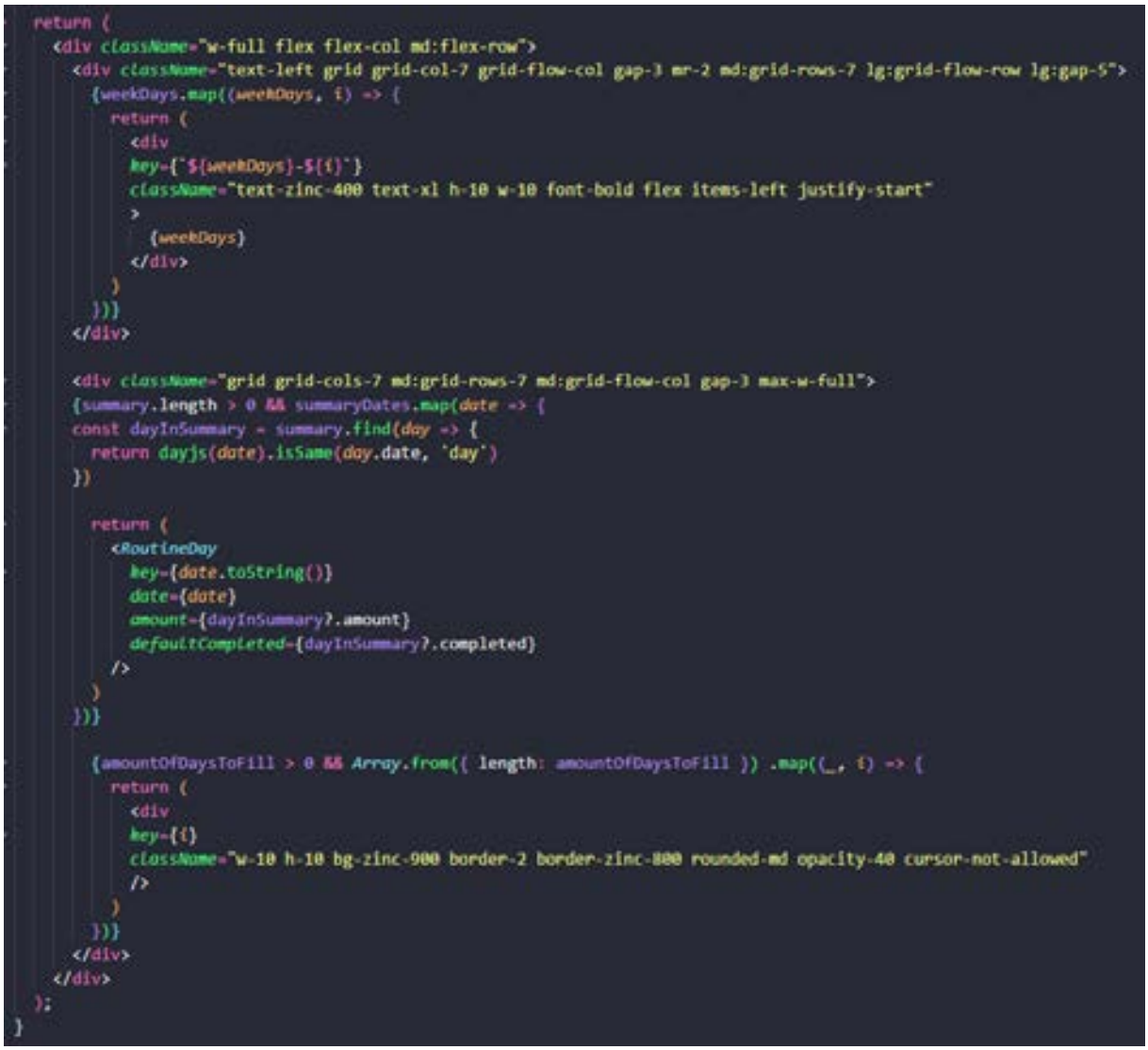
Figura 17: Componente RoutineDay

```tsx
export function RoutineDay({ defaultCompleted = 0, amount = 0, date }: RoutineDayProps){
  const [completed, setCompleted] = useState(defaultCompleted)

  const completedPercentage = amount > 0 ? Math.round((completed/ amount) * 100) : 0

  const dayAndMonth = dayjs(date).format('DD/MM')
  const dayOffWeek = dayjs(date).format('dddd')

  function handleCompletedChanged(completed: number) {
    setCompleted(completed)
  }

  return (
    <Popover.Root>
      <Popover.Trigger
      className={clsx('w-10 h-10  border-2  rounded-lg', {
          'bg-zinc-900 border-zinc-800': completedPercentage === 0,
          'bg-green-900 border-green-700': completedPercentage > 0 && completedPercentage < 40,
          'bg-green-800 border-green-600': completedPercentage >= 20 && completedPercentage < 40,
          'bg-green-700 border-green-500': completedPercentage >= 40 && completedPercentage < 60,
          'bg-green-600 border-green-500': completedPercentage >= 60 && completedPercentage < 80,
          'bg-green-500 border-green-400': completedPercentage >= 80,
        })}
      />

      <Popover.Portal>
          <Popover.Content className="min-w-[320px] p-6 rounded-2xl bg-zinc-900 flex flex-col">
          <span className="font-semibold text-zinc-400">{dayOffWeek}</span>
          <span className="mt-01 font-extrabold leading-tight text-3xl">{dayAndMonth}</span>

          <ProgressBar progress={completedPercentage} />

          <RoutinesList date={date} onCompletedChanged={handleCompletedChanged} />

          <Popover.Arrow height={8} width={16} className="fill-zinc-900" />
        </Popover.Content>
      </Popover.Portal>
    </Popover.Root>
  )
}
```

Fonte: Elaborado pelo autor

Após todas essas etapas, ocorreu o desenvolvimento da *SummaryTable*, o principal componente do *front-end* da aplicação, e o responsável por mostrar todos os dados ao usuário. Esse componente relaciona-se com os anteriores, mostrando somente os hábitos do mês em vigência, e ainda seguindo as regras criadas no componente anterior. Ele apresenta uma tabela com os dias e as *routines* contidas neles, e é onde o usuário consegue fazer a interação com nossa aplicação, podendo visualizar *routines* já concluídas, concluir *routines* e excluir as *routines* já criadas. A Figura 18 exibe parte do código deste componente.

Figura 18: Parte do componente SummaryTable



```
return (
  <div className="w-full flex flex-col md:flex-row">
    <div className="text-left grid grid-col-7 grid-flow-col gap-3 mr-2 md:grid-rows-7 lg:grid-flow-row lg:gap-5">
      {weekDays.map((weekDays, i) => {
        return (
          <div
          key={`${weekDays}-${i}`}
          className="text-zinc-400 text-xl h-10 w-10 font-bold flex items-left justify-start"
          >
            {weekDays}
          </div>
        )
      })}
    </div>

    <div className="grid grid-cols-7 md:grid-rows-7 md:grid-flow-col gap-3 max-w-full">
    {summary.length > 0 && summaryDates.map(date => {
    const dayInSummary = summary.find(day => {
      return dayjs(date).isSame(day.date, 'day')
    })

      return (
        <RoutineDay
          key={date.toString()}
          date={date}
          amount={dayInSummary?.amount}
          defaultCompleted={dayInSummary?.completed}
        />
      )
    })}

      {amountOfDaysToFill > 0 && Array.from({ length: amountOfDaysToFill }) .map((_, i) => {
        return (
          <div
          key={i}
          className="w-10 h-10 bg-zinc-900 border-2 border-zinc-800 rounded-md opacity-40 cursor-not-allowed"
          />
        )
      })}
    </div>
  </div>
);
}
```

Fonte: Elaborado pelo autor

Por ser um sistema para acompanhamento de metas diárias e semanais, foi identificada a necessidade de guardar as metas já concluídas dos meses anteriores. Para isso, foi criado um componente chamado *YearlySummary* que, através de uma função que organiza a forma de visualização dos dias, exibe as tarefas cadastradas nos meses do ano em questão.

Figura 19: Parte do componente YearlySummary

```
const groupDatesByWeek = (dates: Date[]) => {
  const weeks: (Date | null)[][] = [];
  let currentWeek: (Date | null)[] = [];


  const firstDayOfMonth = dayjs(dates[0]);
  const firstDayOfWeek = firstDayOfMonth.day();


  for (let i = 0; i < firstDayOfWeek; i++) {
    currentWeek.push(null);
  }

  dates.forEach((date) => {
    const dayOfWeek = dayjs(date).day();

    currentWeek.push(date);

    if (dayOfWeek === 6) {
      weeks.push(currentWeek);
      currentWeek = [];
    }
  });

  if (currentWeek.length > 0) {
    while (currentWeek.length < 7) {
      currentWeek.push(null);
    }
    weeks.push(currentWeek);
  }

  return weeks;
};
```

Fonte: Elaborado pelo autor

Com a programação concluída, foi detectada a necessidade de garantir a segurança do usuário. Após pesquisas e testes, foram identificadas duas maneiras de realizar o processo de Cadastro de Usuários na aplicação: a primeira, através da Ferramenta *Firebase*, disponibilizada pela Google; a segunda, dentro da própria aplicação, através da criação de rotas e tabelas. A utilização da ferramenta *Firebase* acarretaria um alto custo, incompatível com o orçamento para o desenvolvimento da aplicação. Dessa forma, foi escolhido o processo de cadastro dentro da própria programação da aplicação, salvando os dados como usuário e senha, para que ele possa realizar o cadastro e o *login*.

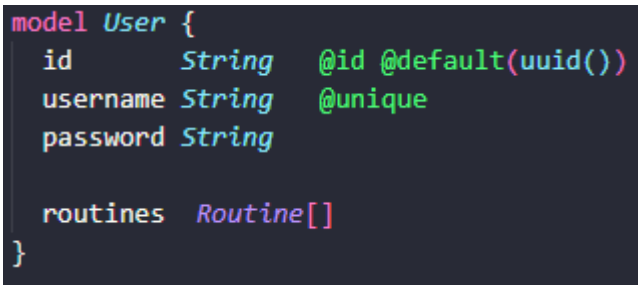
Para que a criação do usuário fosse possível, foram adicionadas novas tabelas, rotas e a parte visual no front-end. O maior desafio dentro deste processo foi relacionar os novos arquivos criados com as tabelas já existentes dentro do banco, para que, assim, a *routines* fosse vinculada ao usuário. As Figuras 20, 21 e 22 exibem parte do código criado.

Figura 20: Criação do Usuário no arquivo *Routes.ts*

```
app.post('/register', async (request, reply) => {
  const createUser = z.object({
    username: z.string(),
     password: z.string()
 })
  const { username, password } = createUser.parse(request.body)

  const user = await prisma.user.create({
    data: {
      username,
      password
    },
  });

  reply.send(user);
});

app.post('/login', async (request, reply) => {
  const User = z.object({
    username: z.string(),
     password: z.string()
 })
  const { username, password } = User.parse(request.body)

  const user = await prisma.user.findUnique({
    where: {
      username,
      password
    },
  });

  if (!user) {
    reply.status(401).send('Usuário não encontrado');
    return;
  }

  reply.send({ id: user.id, username: user.username });
});
```

Fonte: Elaborado pelo autor

Figura 21: Nova tabela dentro do arquivo *Schema.prisma*



```
model User {
  id       String   @id @default(uuid())
  username String   @unique
  password String

  routines  Routine[]
}
```

Fonte: Elaborado pelo autor

Figura 22: Parte do código para a criação do usuário no *Front-End*

```
const handleAuth = async (e: React.FormEvent) => {
  e.preventDefault();
  try {
    if (isLogin) {
      const response = await api.post('/login', { username, password });
      localStorage.setItem('userId', response.data.id);
      navigate('/app');
    } else {
      const response = await api.post('/register', { username, password });
      alert('Usuário registrado com sucesso');
      setIsLogin(true);
    }
  } catch (error) {
    alert('Erro ao autenticar usuário');
  }
};
```

Fonte: Elaborado pelo autor

Com o processo de cadastro e login adicionados, foi identificada a necessidade de haver uma opção para que o usuário pudesse realizar a alteração ou a recuperação da senha de seu cadastro. Para isso, foi criado um gerador de código que verifica o formulário para a alteração de senha, responsável por atualizar o cadastro do usuário no banco de dados através da rota *Reset-Password*, que atualizará os dados de acordo com o nome do usuário, como mostram as figuras 23 e 24.

Figura 23: Gerador de código no *Front-end*

```
const ForgotPassword: React.FC = () => {
  const [username, setUsername] = useState('');
  const [newPassword, setNewPassword] = useState('');
  const [code, setCode] = useState('');
  const [generatedCode, setGeneratedCode] = useState(Math.random().toString(36).substring(7));
  const navigate = useNavigate();

  const handleResetPassword = async (e: React.FormEvent) => {
    e.preventDefault();
    if (code !== generatedCode) {
      alert('Código incorreto!');
      return;
    }

    try {
      await api.post('/reset-password', { username, newPassword });
      alert('Senha alterada com sucesso!');
      navigate('/');
    } catch (error) {
      alert('Erro ao redefinir a senha');
    }
  };
```

Fonte: Elaborado pelo autor

Figura 24: Formulário de Atualização de Cadastro

```
return (
  <div className="w-screen h-screen bg-black">
    <div classNam="flex justify-center p-8">
      <a href="/">
        <img classNam="w-28 h-16 lg:w-32 lg:h-24" src={logoImage} alt="Routine Logo"/>
      </a>
    </div>
    <div classNam="w-screen max-w-md m-auto p-8 space-y-6 text-white shadow-md rounded-lg">
      <h1 classNam="text-xl font-bold text-center">Redefinir Senha</h1>
      <form onSubmit={handleResetPassword} classNam="space-y-4">
        <div>
          <label classNam="block text-gray-300">Nome do usuário</label>
          <input
            type="text"
            value={username}
            onChange={(e) => setUsername(e.target.value)}
            classNam="w-full px-3 py-2 bg-gray-900 border border-gray-600 rounded-md shadow-sm focus:outline-none focus:ring-1 focus:ring-blue-500"
          />
        </div>
        <div>
          <label classNam="block text-gray-300">Nova senha</label>
          <input
            type="password"
            value={newPassword}
            onChange={(e) => setNewPassword(e.target.value)}
            classNam="w-full px-3 py-2 bg-gray-900 border border-gray-600 rounded-md shadow-sm focus:outline-none focus:ring-1 focus:ring-blue-500"
          />
        </div>
        <div>
          <label classNam="block text-gray-300">Código</label>
          <input
            type="text"
            value={code}
            onChange={(e) => setCode(e.target.value)}
            classNam="w-full px-3 py-2 bg-gray-900 border border-gray-600 rounded-md shadow-sm focus:outline-none focus:ring-1 focus:ring-blue-500"
          />
        </div>
        <p classNam="text-gray-400 text-sm">Código de verificação: {generatedCode}</p>
        <div classNam="flex justify-center mt-8">
          <button
            type="submit"
            classNam="w-1/2 mx-auto mt-5 py-2 px-2 mt-5 font-semibold bg-green-500 text-white rounded-md hover:bg-green-600"
          >
            Redefinir Senha
          </button>
        </div>
      </form>
```

Fonte: Elaborado pelo autor

6 Resultados e Discussão

Para compreender os resultados do projeto, é essencial mencionar os objetivos iniciais que nortearam esta pesquisa: a criação de um aplicativo de gerenciamento de hábitos de fácil utilização e que permite acompanhar e organizar os hábitos diários com eficiência.

Inicialmente, o resultado esperado era a criação de um aplicativo simples para gerenciar as tarefas pessoais. Porém, durante a jornada de desenvolvimento, com a estruturação do código e o entendimento do curso base, melhorias foram realizadas, como o processo de *login* para a criação do usuário e a redefinição de senha, visando a segurança.

Além disso, também foi criada uma barra de progresso para acompanhar a evolução, sendo possível visualizar os hábitos, diária, semanal, mensal e anualmente. Para complementar, foi acrescentada uma ferramenta de definição de prazo para os hábitos, caso seja necessário. Tais melhorias elevaram as expectativas iniciais e, consequentemente, proporcionarão uma melhor experiência ao usuário.

A aplicação desenvolvida ainda está em processo de construção, com a possibilidade de implementação de novas funcionalidades. Atualmente, o sistema trabalha de forma local, mas o intuito é ambientar e permitir que esteja em um domínio na internet para que assim todos os usuários possam acessar de forma fácil e prática.

Por ser uma aplicação de utilização simples, não é exigida grande potência de Hardware para seu funcionamento; logo tem maior compatibilidade com computadores de utilização comum. A arquitetura desenvolvida na aplicação é categorizada como fácil e objetiva, tendo um método conhecido para seu desenvolvimento, o que facilita a compreensão e modificação por parte de desenvolvedores. Por ser uma aplicação desenvolvida por um estudante, não foram necessários custos e investimentos para sua implementação, apenas tempo e esforço.

Considerações finais

O trabalho em questão foi desenvolvido com o intuito de gerenciar os hábitos e metas ao longo do curso, pois, durante a graduação, foi observada uma dificuldade dos graduandos em conciliar todas as tarefas e obrigações do dia a dia, assim como

criar e gerenciar hábitos que gostariam de implementar em suas rotinas. Dessa forma, nem todos os objetivos almejados eram concluídos com êxito, pois não dispunham de uma ferramenta que os auxiliasse nesse processo de iniciar um novo projeto em suas vidas.

Com o intuito de ajudar a todos que enfrentam esse mesmo desafio, surgiu a ideia de criar um aplicativo simples e acessível para o acompanhamento dos hábitos e objetivos que deveriam ser cumpridos em um determinado prazo.

Por conseguinte, foi idealizada uma aplicação que desse assistência aos usuários para evitar a procrastinação e permitir a efetivação da rotina planejada pelos usuários. Para concretizar a proposta, surgiu a ideia de transformá-la em um trabalho acadêmico.

Como esperado, alguns desafios surgiram durante todo o processo, como relacionar cada usuário com seus respectivos dados; fazer a criação de rotas; entender sobre uma nova linguagem de programação; e buscar um diferencial para a aplicação.

Pensando no futuro, é possível identificar inúmeras possibilidades de melhorias para a aplicação, como um painel para acompanhar a meta de outros usuários (no caso de uma equipe de trabalho, por exemplo); um campo para adicionar notas e lembretes aos hábitos já criados; notificações ao usuário; e personalização do fundo da página inicial do projeto.

Agradeço a todos que estiveram envolvidos para tornar o projeto possível, à Rocketseat por trazer a ideia através de NWLs disponíveis em sua página e ao orientador Prof. Carlos Alberto Lucas que me auxiliou sempre que necessário para a conclusão do referido Trabalho de Conclusão de Curso.

Referências

AXIOS. **Cliente HTTP baseado em promessas para o navegador e node.js**, 2016. Disponível em: https://axios-http.com/ptbr/docs/intro. Acesso em: 12 Set. 2023.

CGU - Controladoria Geral da União. **Guia de Modelagem de Processos da BPMN**, 2020. Disponível em: https://repositorio.cgu.gov.br/bitstream/1/66339/3/Guia_de_Modelagem_de_Processos.pdf. Acesso em: 22 Abr. 2024.

DATE-FNS. **Biblioteca para manipulação de datas com suporte a fusos horários, v4.0**. Disponível em: https://date-fns.org/. Acesso em: 12 Set. 2023.

DAY. **Alternativa rápida de 2kB para Moment.js com a mesma API moderna,** s.d. Disponível em: https://day.js.org/. Acesso em: 12 Set. 2023.

FASTIFY. **Framework web rápido e de baixo custo, para Node.js**, 2016. Disponível em: https://fastify.dev/. Acesso em: 12 Set. 2023.

HISATOMI, Marco Ikuro**;** GÓES, Anderson de Souza; BARROS, Rodolfo Mirando de**. Gestão de Regras de Negócios**, 2014. Engenharia de Software Magazine, EDIÇÃO 66. Disponível em: https://gaia.uel.br/download.php?q=ZC9mNGZkYTE1NWU4ZGRlOGU5MmU3N2ZhMDFjZTgyZTI2OS5wZGY=. Acesso em: 08 Mai. 2024.

INATEL - Incubadora de empresas e projetos**. Modelo de Negócios Canvas**: teoria e prática final, 2012. Disponível em: https://edisciplinas.usp.br/pluginfile.php/374839/mod_resource/content/1/Modelo%20de%20Negocio%20Canvas%20-%20Teoria%20e%20Pratica%20Final.pdf. Acesso em: 22 Abr. 2024.

INPE - Instituto Nacional de Pesquisas Espaciais. **Guia Prático de Preenchimento do TAP – Termo de Abertura do Projeto**, 2021. Disponível em: http://mtc-m21d.sid.inpe.br/col/sid.inpe.br/mtc-m21d/2022/04.20.17.24/doc/publicacao.pdf?metadatarepository=sid.inpe.br/mtc-m21d/2022/04.20.17.24.31&mirror=urlib.net/www/2021/06.04.03.40.25&languagebutton=en. Acesso em: 22 Abr. 2024.

MARTINS, Luiz Eduardo. **Uma metodologia de elicitação de requisitos de software baseada na teoria da atividade.** 2001. Disponível em: https://repositorio.unicamp.br/Busca/Download?codigoArquivo=466908. Acesso em: 15 Mai. 2024.

NAKAGAWA, Marcelo. SEBRAE, 2015. **Ferramenta: ANÁLISE SWOT (CLÁSSICO)**. Disponível em: https://sebrae.com.br/Sebrae/Portal%20Sebrae/Anexos/ME_Analise-Swot.PDF. Acesso em: 22 Abr. 2024.

PRISMA. **Simplifique o trabalho e a interação com o banco de dados**, s.d. Disponível em: https://www.prisma.io/. Acesso em: 12 Set. 2023.

PRODEST – Instituto de Tecnologia da Informação e Comunicação do Espirito Santo, s.d. **Como evitar que o acúmulo de funções prejudique a sua qualidade de vida no trabalho.** Disponível em: https://prodest.es.gov.br/como-evitar-que-o-acumulo-de-funcoes-prejudique-a-sua-qualidade-de-vida-no-trabalho. Acesso em: 22 Abr. 2024.

Project Management Institute. **Um Guia do Conhecimento em Gerenciamento de Projetos** (Guia PMBOK), 2013. Disponível em: https://www.facom.ufu.br/~william/Disciplinas%202018-2/BSI-GSI030-EngenhariaSoftware/Livro/engenhariaSoftwareSommerville.pdf. Acesso em: 12 Set. 2023.

REACT. **A biblioteca para interfaces de usuário web e nativas**, s.d. Disponível em: https://react.dev/. Acesso em: 12 Set. 2023.

REACT DATEPICKER. **Biblioteca de seleção de datas para React**. Disponível em: https://reactdatepicker.com/. Acesso em: 12 Set. 2023.

SEBRAE. **5W2H: o que é, para que serve e por que usar na sua empresa,** 2023. Disponível em: https://www.sebrae-sc.com.br/blog/5w2h-o-que-e-para-que-serve-e-por-que-usar-na-sua-empresa. Acesso em: 05 Mai. 2024.

SOMMERVILLE, Ian. **Engenharia de Software**. São Paulo: Person, 2011. Disponível em: https://www.facom.ufu.br/~william/Disciplinas%202018-2/BSI-GSI030-EngenhariaSoftware/Livro/engenhariaSoftwareSommerville.pdf. Acesso em: 06 Mai. 2024.

TAILWINDCSS. **Crie sites modernos rapidamente, sem sair do HTML**, s.d. Disponível em: https://tailwindcss.com/. Acesso em: 12 Set. 2023.

TYPESCRIPTLANG. **TypeScript é JavaScript com sintaxe para tipos**, 2012. Disponível em: https://www.typescriptlang.org/. Acesso em: 12 Set. 2023.

ZOD. **Validação de esquema primeiro TypeScript com inferência de tipo estático**, s.d. Disponível em: https://zod.dev/. Acesso em: 12 Set. 2023.