RESIGeT On-Line
Revista Eletrônica de Sistemas de Informação e Gestão Tecnológica
Vol. 14 Nro. 1 2024
Uni-FACEF
Centro Universitário Municipal de Franca

e-Time - Desenvolvimento de um Aplicativo de Controle de Horário para Profissionais

Lucas Castellan
Graduando em Sistemas de Informação – Uni-FACEF
lucascastelanph@gmail.com

Vinicius Freitas Pinheiro
Graduando em Sistemas de Informação – Uni-FACEF
viniciusfpinheiro0@gmail.com

Geraldo Henrique Neto
Mestre em Ciências com ênfase em informática Médica – Uni-FACEF
geraldo.henriqueneto@gmail.com

RESUMO
O avanço do trabalho remoto trouxe desafios no controle e gerenciamento de horas trabalhadas, especialmente com equipes distribuídas em diferentes fusos horários. Há uma crescente necessidade de soluções que garantam a precisão na marcação de ponto e promovam uma gestão justa do tempo de trabalho. O objetivo deste projeto é desenvolver o *e-Time*, uma ferramenta que registra e gerencia as horas trabalhadas, controlando tanto a carga horária diária quanto às horas extras, além de facilitar a administração de um banco de horas. Essa solução visa oferecer uma experiência integrada e automatizada para empresas e colaboradores, garantindo notificações sobre o início e término da jornada. A justificativa para o desenvolvimento do *e-Time* está na carência de plataformas que atendam de maneira eficaz às particularidades do trabalho remoto, proporcionando maior transparência e eficiência na gestão do tempo. A metodologia adotada foca na criação de uma aplicação *desktop* (interface gráfica de um sistema operacional), intuitiva e precisa, com funcionalidades que incluem o registro automatizado de jornada e ferramentas de monitoramento em tempo real para gestores. Os principais resultados esperados incluem maior precisão no controle de horas, satisfação dos colaboradores e uma gestão mais eficiente do tempo de trabalho, refletindo-se em maior produtividade. Concluindo, o *e-Time* surge como uma solução que não apenas resolve a necessidade imediata do mercado, mas também estabelece um novo padrão para o gerenciamento do trabalho remoto, promovendo um ambiente mais justo, transparente e produtivo.

Palavras-chave: Trabalho remoto. Controle de horas. *e-Time*. Registro de ponto

ABSTRACT
The advancement of remote work has brought challenges in controlling and managing hours worked, especially with teams distributed across different time zones. There is a growing need for solutions that guarantee precision in timekeeping and promote fair management of working time. The objective of this project is to develop e-Time, a tool that records and manages hours worked, controlling both daily workload and overtime, in addition to facilitating the administration of a time bank. This solution aims to offer

an integrated and automated experience for companies and employees, ensuring notifications about the beginning and end of the journey. The justification for the development of e-Time is the lack of platforms that effectively meet the particularitiesof remote work, greater transparency and efficiency in time management. The methodology adopted focuses on creating an intuitive and accurate desktop application, with features that include automated journey recording and real-time monitoring tools for managers. The main expected results include greater precision in time tracking, employee satisfaction and more efficient management of working time, resulting in greater productivity. In conclusion, e-Time emerges as a solution that not only solves an immediate market need, but also establishes a new standard for remote work management, promoting a fairer, more transparent and productive environment.

***Keywords**: Remote work. Time tracking.* e-Time. *Timekeeping.*

1. INTRODUÇÃO

No cenário atual, com a ascensão do trabalho remoto, a importância de uma ferramenta eficaz para o registro e gerenciamento de horas trabalhadas torna-se indiscutível. A flexibilidade e os desafios singulares do trabalho à distância, destacam a carência de soluções integradas, de maneira eficiente como o controle de ponto, especialmente frente às complexidades apresentadas pelos diferentes fusos horários das equipes distribuídas globalmente. Visto que o trabalho remoto, que já era adotado em várias áreas profissionais, tem se consolidado como uma modalidade de emprego formal, especialmente no contexto do teletrabalho (Santana, 2018, p. 172).

Essa necessidade evidencia uma lacuna no mercado por uma plataforma capaz de oferecer um acompanhamento preciso e simplificado das horas de trabalho, peça-chave para a administração eficaz de equipes remotas. Partindo disto, como a tecnologia em conjunto com esta configuração de trabalho pode nos auxiliar no controle e registro de horas trabalhadas?

Com isto temos como objetivo principal o desenvolvimento de uma ferramenta que abranja toda esta necessidade, que seria controle as horas trabalhadas pelo colaborador, quanto tempo resta para o cumprimento de sua carga horária diária, como também registrar as horas extras que o funcionário desempenhou em favor da empresa, gerando-se um banco de horas. Assim oferecemos um conforto ao colaborador já que o *software* também visa sempre lembrar ao colaborador de ligar o marcador, como também notificá-lo de que sua jornada de oito horas diárias terminou, promovendo um trabalho mais justo para ambas as partes: o gestor da equipe e o funcionário.

Diante desse cenário, surge a *e-Time*, uma solução pioneira que promete revolucionar o gerenciamento do tempo em ambientes de trabalho remotos. Com sua aplicação *desktop*, o *e-Time* é projetado para superar os desafios do trabalho à distância, permitindo que os usuários registrem o início e término de suas jornadas com facilidade, independentemente de sua localização ou fuso horário. A plataforma não apenas facilita a medição acurada das horas trabalhadas, mas também proporciona aos gestores ferramentas avançadas para o manejo eficiente dessas horas, incluindo monitoramento em tempo real, ajustes de carga de trabalho e

administração do banco de horas, fomentando assim uma distribuição de tarefas mais equitativa e o aumento da produtividade.

A adesão ao *e-Time* marca um progresso significativo no gerenciamento do trabalho remoto, superando barreiras impostas pela distância e variação de fusos horários. Ao garantir registros precisos e facilitar uma gestão de tempo integrada, a plataforma não só responde a uma demanda imediata do mercado, mas também estabelece as bases para um ambiente de trabalho mais justo, transparente e eficaz. O e-Time não é só uma ferramenta de controle de tempo, é um aliado para empresas e profissionais no trabalho remoto.

2 HOME-OFFICE: CONCEITOS E TENDÊNCIAS

O termo *home office* vem da língua inglesa e significa trabalho feito em casa, podendo ter um sentido um pouco mais amplo, como um trabalho que pode ser feito em qualquer lugar e de forma remota. O que se tornou possível por meio das evoluções tecnológicas que foram acontecendo com o passar do tempo. O teletrabalho é um termo que surgiu quando o mundo passava pela crise do petróleo, fazendo com que os gastos com deslocamento ao trabalho se tornassem uma grande despesa para as empresas e assim o *home office* começou a ganhar espaço como uma alternativa viável para certas atividades (Froehlich, 2020).

Esta configuração vem se tornando parte das opções de trabalho, visto que no período de pandemia ela teve um grande avanço, fazendo com que muitas empresas adotassem esta nova forma de se trabalhar, esta que segue com grande tendência atualmente. Um estudo realizado pela Fundação Getúlio Vargas (FGV), por exemplo, concluiu que a “prática do *home office* deve crescer em 30% após o término da pandemia” (Boehm Camila, 2020, p.1).

Com tudo há a promessa de que esta modalidade forneça aos colaboradores, mais flexibilidade e liberdade, pois os meios utilizados para o *home office*, permitem que se trabalhe de qualquer local, tendo apenas uma conexão com a internet. Em contrapartida, o uso do teletrabalho pode ultrapassar os limites, atrapalhando viagens de férias e o tempo dedicado para o descanso do colaborador.

No desenvolvimento da sociedade, a forma de trabalhar tem mudado suas características, passando de atividades físicas para atividades intelectuais. O problema é que este esquema não transfere lucros para os trabalhadores pro meio de apropriação do uso de tecnologia financiada pelo investimento público, mas cria uma maior concentração de rendimentos e uma perda de direitos laborais.

Antes desta modalidade de trabalho, os colaboradores tinham que enfrentar longas horas se deslocando até as suas respectivas empresas, seja por metrô, trem e até mesmo ônibus, com isto muitos funcionários chegam em suas empresas já cansados antes mesmo de iniciar o expediente, desta forma as empresas deixam de desfrutar de total desempenho de seus colaboradores (Rocha & Amador, 2018).

Navarini e Pereira (2021, p.15) observam que a possibilidade de trabalho remoto, principalmente em ambientes híbridos, pode representar risco de perdas, pois os funcionários precisam transportar equipamentos e documentos até suas residências e esse movimento pode levar ao vazamento de informações. Estudos disponíveis que contemplam a um referencial teórico que traga debates sobre a modalidade de trabalho *home office* ainda são contestáveis. Grande parte dos

materiais e estudos existentes com essa temática são ligados à educação e como ela atua em relação ao ensino híbrido.

2.1 CONTROLE DE HORÁRIO: IMPORTÂNCIA E DESAFIOS

Dentro de um mundo em que o *home office* se estruturou de forma impactante na sociedade a gestão do tempo de forma eficiente aparece como uma estrutura de extrema importância para correlacionar a vida social com a profissional. Mesmo oferecendo flexibilidade e um grande conforto, podendo muitas vezes obter uma escolha de horário e local de trabalho com amplitude, os desafios estão presentes principalmente dentro do ambiente de produção no qual majoritariamente é utilizado o espaço doméstico. Neste âmbito há necessidade de acolher novas estratégias de controle de tempo para alcançar um bom nível de produtividade, sempre mantendo o bem-estar do indivíduo (Sousa & Saraiva, 2023).

Um dos maiores obstáculos enfrentados no regime de *home office* é a propensão ao sobretrabalho, decorrente da dificuldade de estabelecer limites rígidos entre o horário de trabalho e o tempo destinado ao descanso e lazer. A adoção de ferramentas tecnológicas de gerenciamento de tarefas e a prática do *time-blocking* surgem como soluções viáveis para estruturar o dia de maneira que o trabalho seja realizado de forma eficaz, permitindo períodos de descanso e recuperação adequados (Ferreira Certo, 2023).

Adicionalmente, a questão da supervisão a distância e o monitoramento da produtividade colocam em pauta a importância da comunicação e da transparência nas relações de trabalho. A implementação de reuniões de alinhamento e o uso de plataformas colaborativas *online* são essenciais para a manutenção de uma comunicação eficaz, ajudando a construir um ambiente de confiança mútua entre equipe e gestores (Silva, 2023).

A ergonomia no *home office* é um aspecto frequentemente subestimado, mas de vital importância para prevenir problemas de saúde e aumentar a produtividade. Trabalhar em casa pode levar ao uso de instalações inadequadas, como cadeiras desconfortáveis ou mesas de altura imprópria, aumentando o risco de problemas musculoesqueléticos. Portanto, é crucial que os empregadores forneçam diretrizes ergonômicas e, quando possível, suporte para a montagem de estações de trabalho adequadas em casa (Araújo & João Junior, 2022). Interrupções frequentes é outra característica do ambiente doméstico que pode complicar a gestão eficaz do tempo. Membros da família, tarefas domésticas e até animais de estimação podem se tornar fontes constantes de distração, dificultando a manutenção do foco. Isso ressalta a necessidade de criar um espaço dedicado ao trabalho e estabelecer limites claros com os outros membros da casa, além de implementar técnicas de gestão de tempo, como a técnica Pomodoro, para segmentar o trabalho em períodos de alta concentração alternados com pausas curtas (Ferreira Certo, 2023).

Finalmente, as políticas organizacionais devem ser adaptadas para abordar os desafios e oportunidades do trabalho remoto. Isso inclui a revisão das políticas de trabalho, a introdução de flexibilidade nos horários de trabalho e a criação de canais de comunicação eficazes que ajudem a manter a coesão da equipe e a cultura organizacional. Adotar uma abordagem mais flexível pode ajudar a maximizar os benefícios do *home office* enquanto minimiza seus desafios, promovendo um ambiente de trabalho mais adaptável e resiliente (Silva, 2023).

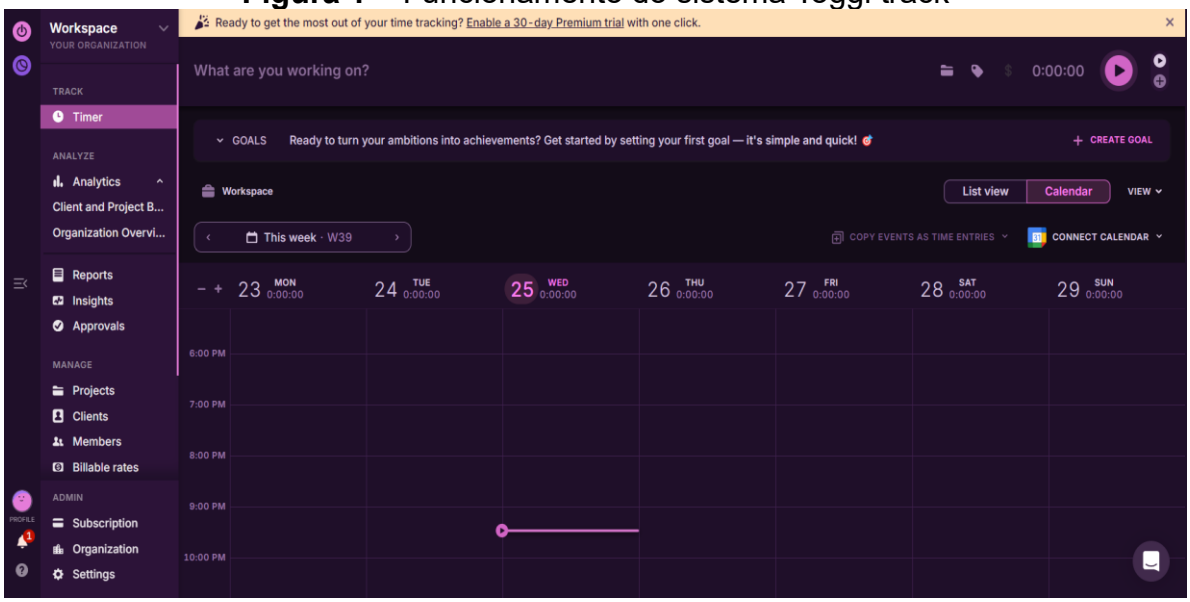
2.2 REVISÃO DE APLICATIVOS SIMILARES

A análise de aplicativos similares é uma etapa crucial para entender o mercado atual, identificar lacunas e oportunidades, e estabelecer *marcos comparativos* para o desenvolvimento do *e-Time*. Serão revisados dois dos principais aplicativos de controle de horário disponíveis no mercado, com foco em suas funcionalidades, usabilidade, tecnologia empregada e avaliações de usuários.

2.2.1 TOGGL TRACK

Toggl Track é um aplicativo popular de rastreamento de tempo que permite aos usuários monitorarem suas horas de trabalho com facilidade, como apresentado na Figura 1. Ele é utilizado tanto por indivíduos quanto por equipes em diversas indústrias e oferece funcionalidades como rastreamento de tempo em tempo real, relatórios detalhados e exportáveis. Toggl Track utiliza uma combinação de tecnologias *web* e *mobile)*, incluindo *React* para a interface de usuário e Node.js no *backend* (parte de um sistema ou aplicativo que é responsável pelo processamento de dados, lógica de negócios e comunicação com o banco de dados). Além disso, oferece uma *API RESTful* para integração com outros sistemas. Os usuários destacam a simplicidade e a eficácia do Toggl Track, mencionando a facilidade de uso e a qualidade dos relatórios. No entanto, alguns relatam dificuldades com a sincronização entre dispositivos e a curva de aprendizado inicial (Toggl, 2024).

Figura 1 – Funcionamento do sistema Toggl track



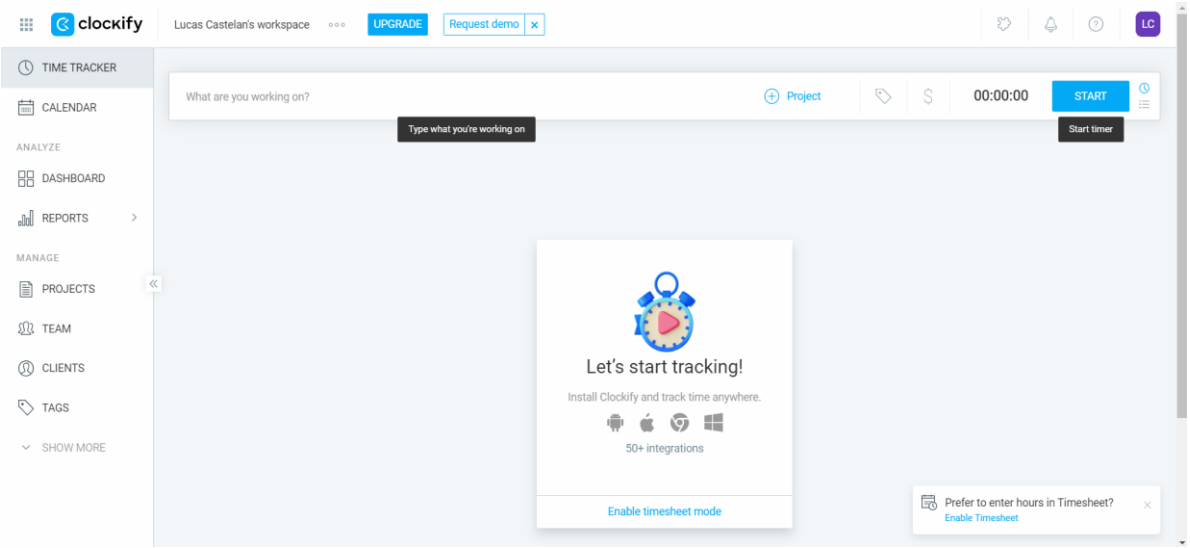
FONTE: Toggl, 2024.

2.2.2 CLOCKIFY

Clockify é uma ferramenta gratuita de rastreamento de tempo e monitoramento de produtividade, projetada para profissionais autônomos e pequenas empresas. Entre suas principais funcionalidades, destacam-se o rastreamento de tempo ilimitado e relatórios detalhados de produtividade como apresentado na Figura 2. O Clockify é construído utilizando tecnologias modernas da internet, com uma

arquitetura baseada em micro serviços. Ele utiliza React para a interface do usuário e Node.js para o sistema de *backend*, garantindo escalabilidade e desempenho. Os usuários apreciam a funcionalidade gratuita ilimitada e a facilidade de uso do Clockify. No entanto, alguns apontam como desvantagens a complexidade na configuração inicial e as limitações nas funcionalidades de relatórios avançados (Clockify, 2024).

Figura 2 – Funcionamento do sistema Toggl track



FONTE: clockify, 2024

3 TECNOLOGIAS PARA DESENVOLVIMENTO DE APLICATIVOS

Para o desenvolvimento do aplicativo *e-Time*, foram escolhidas tecnologias modernas com excelente integração, permitindo a construção de um *software fullstack* eficiente e de alta usabilidade. O Next.js foi adotado como base da aplicação, o que possibilitou um desenvolvimento unificado em um único ambiente, proporcionando uma experiência fluida e otimizada para os usuários. A seguir, são apresentados os detalhes das principais tecnologias utilizadas no desenvolvimento do aplicativo.

- TAURI COM NEXT.JS (*DESKTOP FRONTEND*): O Tauri permite desenvolver aplicações *desktop* usando tecnologias web com baixo consumo de recursos. No *e-Time*, foi implementado com o Next.js para criar uma interface eficiente e responsiva, aproveitando recursos de geração de páginas estáticas (SSG, *Static Site Generation*) e renderização no servidor (SSR, *Server-Side Rendering*). A escolha do Tauri se deu por sua leveza em comparação com o *Electron*, enquanto o Next.js foi escolhido por sua performance e facilidade no desenvolvimento de interfaces (Tauri, 2024; Vercel, 2024).

- NEXT.JS (FRONTEND): Next.js é um framework utilizado para o desenvolvimento do *frontend* web do *e-Time*, focado em criar interfaces dinâmicas e interativas. Ele foi implementado com recursos como renderização híbrida e roteamento automático, melhorando o carregamento

das páginas e a performance. A escolha do Next.js foi feita por sua capacidade de otimizar SEO (*Search Engine Optimization*) através da renderização no servidor, além de proporcionar uma abordagem *fullstack*, facilitando a manutenção e escalabilidade do projeto (Vercel, 2024).

- NODE.JS (*BACKEND*): Node.js é usado no *backend* do *e-Time* por sua capacidade de desenvolver aplicações web escaláveis com arquitetura baseada em eventos e operações assíncronas. Foi implementado para gerenciar múltiplas requisições simultâneas e manter a responsividade do sistema. A escolha do Node.js se deve à sua integração com o *frontend* desenvolvido em Next.js, além de permitir o uso de JavaScript e TypeScript em toda a aplicação, simplificando a comunicação entre as camadas e reduzindo a complexidade do desenvolvimento (Node.js, 2024).

- PRISMA COM SQLITE: Prisma é o ORM (*Object-Relational Mapping*) utilizado para gerenciar o banco de dados do *e-Time*, integrado com o SQLite. Ele foi implementado para simplificar a interação com o banco, fornecendo uma camada de abstração que facilita a escrita de consultas e gerência de migrações automáticas. A escolha do SQLite se deu por sua leveza e simplicidade, adequadas para o desenvolvimento local, enquanto o Prisma garante que a transição futura para bancos mais robustos, como MySQL ou PostgreSQL, seja facilitada (SQLITE, 2024; Prisma, 2024).

- ZUSTAND PARA GERENCIAMENTO DE ESTADO: Zustand foi utilizado no *frontend* do *e-Time* para gerenciar o estado global. Ele foi escolhido por sua simplicidade e menor necessidade de código em comparação com alternativas como Redux, tornando o desenvolvimento mais rápido e fácil de manter. A leveza e a API (*Application Programming Interface*) intuitiva do Zustand permitem gerenciar estados reativos sem comprometer a performance, o que é ideal para a eficiência do *e-Time*. Além disso, sua integração com React no ambiente Next.js facilita o gerenciamento de estados de maneira escalável e previsível (PMNDRS, 2024).

- CLERK PARA GERENCIAMENTO E AUTENTICAÇÃO DE USUÁRIOS: Clerk é utilizado no *e-Time* para gerenciar a segurança e autenticação de usuários. Ele foi implementado para facilitar processos como login, registro, recuperação de senha e autenticação de dois fatores, garantindo a segurança do sistema. A escolha do Clerk se deu por sua fácil integração com Next.js e pela simplicidade na implementação da autenticação, além de oferecer uma experiência amigável e personalizável para os usuários (Clerk, 2024).

4 REQUISITOS FUNCIONAIS E NÃO FUNCIONAIS DO SISTEMA

Para a construção do sistema proposto, foi realizada uma especificação detalhada dos requisitos funcionais necessários para o seu pleno funcionamento. Os requisitos funcionais descrevem as funcionalidades essenciais que o *software* deve oferecer, garantindo que as expectativas dos usuários e dos administradores sejam atendidas de maneira clara e objetiva. Nas Tabelas 1 e 2 a identificação, o nome, a

descrição, a categoria, a prioridade e informações adicionais sobre cada um dos requisitos funcionais e não funcionais respectivamente.

TABELA 1 - 4.1 REQUISITOS FUNCIONAIS

ID	Nome	Descrição	Prioridade	Dependências
RF001	Autenticação de Usuário	O sistema deve permitir que o usuário faça login ou crie uma conta. O login pode ser feito por email ou número de telefone.	Alta	Nenhuma
RF002	Recuperação de Senha	O sistema deve permitir que o usuário redefina sua senha ou recupere o acesso à conta caso esqueça a senha.	Média	RF001
RF003	Criação de Equipes	O sistema deve permitir que o usuário crie uma equipe, preenchendo o nome da equipe e assim definindo o criador automaticamente como administrador da equipe.	Alta	RF001
RF004	Entrada em Equipe	O sistema deve permitir que o usuário entre em uma equipe já existente, após receber um convite ou permissão do administrador.	Alta	RF001
RF005	Administração de Equipe	O sistema deve permitir que o administrador da equipe adicione e remova membros, bem como gerencie permissões dentro da equipe.	Média	RF003
RF006	Registro de Jornada de Trabalho	O sistema deve permitir que o usuário registre o início e término de suas jornadas de trabalho, incluindo a funcionalidade de adicionar pausas e intervalos com durações definidas ou customizadas.	Alta	RF001, RF004
RF007	Temporizador de Trabalho	O sistema deve exibir um temporizador para que o usuário inicie, pause e finalize suas sessões de trabalho, com a opção de adicionar intervalos a qualquer momento.	Alta	RF006
RF008	Visualização de Estatísticas de Produtividade	O sistema deve exibir gráficos e dados relacionados à produtividade do usuário, incluindo horas trabalhadas, horas extras e metas atingidas.	Média	RF006, RF007
RF009	Monitoramento de Equipe	O sistema deve permitir que o usuário visualize as atividades de outros membros da equipe, como progresso de trabalho e localização, dentro da página de "Equipe".	Média	RF004

RF010	Notificações de Tarefas e Horários	O sistema deve enviar notificações ao usuário relacionadas a tarefas, início e término de jornada, pausas e intervalos programados.	Média	RF006
RF011	Exportação de Relatórios	O sistema deve permitir que o usuário exporte relatórios das horas trabalhadas e produtividade em formatos como PDF ou CSV.	Baixa	RF008
RF012	Sincronização de Dados	O sistema deve sincronizar os dados do usuário e da equipe em tempo real entre diferentes dispositivos, garantindo que todas as informações estejam atualizadas.	Alta	RF006, RF004
RF014	Gerenciamento de Permissões	O sistema deve permitir que administradores alterem permissões específicas para usuários de sua equipe.	Alta	RF003
RF015	Relatórios Detalhados	O sistema deve permitir que relatórios detalhados sejam gerados com dados específicos, como tarefas concluídas e tempo por tarefa.	Baixa	RF011
RF016	Histórico de Atividades	O sistema deve permitir que os usuários visualizem um histórico completo de atividades, incluindo horários de entrada, saída e pausas registradas.	Baixa	RF008,RF012

TABELA 2 - 4.1.2 REQUISITOS NÃO FUNCIONAIS

ID	Nome	Descrição	Prioridade	Dependências
RNF001	Segurança de Autenticação	Todas as informações de login e dados sensíveis devem ser criptografadas usando algoritmos modernos de segurança, como fornecido pelo Clerk.	Alta	Nenhuma
RNF002	Tempo de Resposta	O sistema deve responder a qualquer solicitação do usuário (como iniciar uma jornada ou visualizar um gráfico) em menos de 2 segundos.	Alta	Nenhuma
RNF003	Disponibilidade do Sistema	O sistema deve estar disponível 99% do tempo, exceto durante os períodos de manutenção programada.	Alta	Nenhuma
RNF004	Compatibilidade Multi-Plataforma	O sistema deve ser compatível tanto com navegadores web quanto com a versão desktop (Tauri), sem perda de funcionalidades.	Alta	Nenhuma

RNF005	Design Responsivo	A interface do sistema deve ser responsiva e adaptável a diferentes tamanhos de tela, garantindo uma boa experiência tanto em *desktop* quanto em dispositivos móveis.	Média	Nenhuma
RNF006	Escalabilidade	O sistema deve ser capaz de escalar para suportar até 1000 usuários simultâneos sem perda de desempenho.	Alta	Nenhuma
RNF007	Usabilidade	O sistema deve ser intuitivo e fácil de usar, com uma interface que não exija treinamento intensivo para novos usuários.	Média	Nenhuma
RNF008	Privacidade de Dados	O sistema deve estar em conformidade com as normas de proteção de dados, como a LGPD (Lei Geral de Proteção de Dados), garantindo que os dados dos usuários não sejam compartilhados sem consentimento.	Alta	Nenhuma

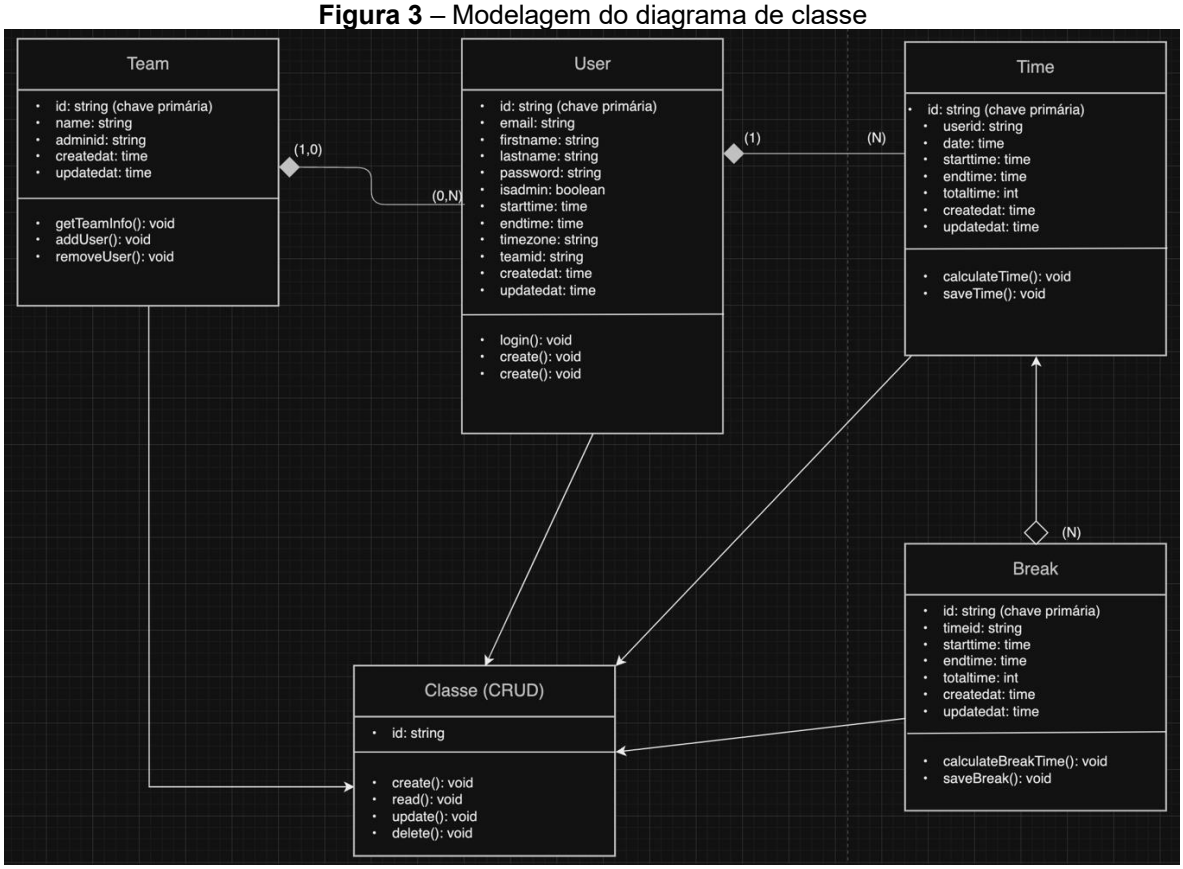
4.2 DIAGRAMA DE CLASSE

O diagrama representado pela Figura 3, apresenta a modelagem das principais classes envolvidas no sistema, demonstrando as estruturas das classes e seus respectivos relacionamentos. Ilustra também os principais métodos utilizados, a citar: operações de criação, leitura, atualização e exclusão que podem ser realizadas nas entidades Team, User e Time.

4.3 MODELAGEM LÓGICA DO BANCO DE DADOS

Na Figura 4 contém a modelagem lógica do banco de dados que descreve as tabelas do *software*, assim como seus respectivos atributos. As tabelas representam as principais funcionalidades do sistema, como a gestão de usuários, equipes, jornadas de trabalho e pausas. Cada tabela inclui atributos específicos, como nome, e-mail, horário de início e término de trabalho, além de informações sobre o tempo total de atividade e intervalos. Esse modelo é essencial para o gerenciamento eficiente das funcionalidades do *software*, permitindo controle e organização dos dados.

Figura 3 – Modelagem do diagrama de classe



FONTE: OS AUTORES, 2024.

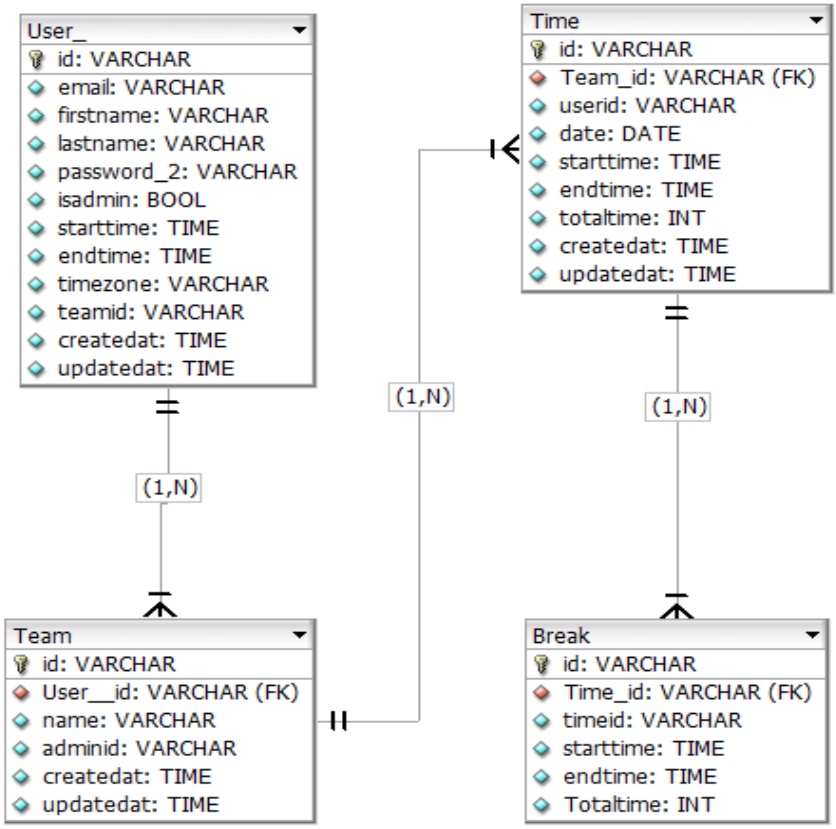
4.4 DIAGRAMA DE CASOS DE USO

O diagrama de casos de uso apresentado na Figura 5 representa as interações dos usuários com o sistema, destacando os principais casos de uso disponíveis tanto para usuários comuns quanto para administradores. Ele é dividido em dois níveis, autenticação, na qual o usuário pode realizar login ou registrar-se, e o sistema em si, em que, uma vez autenticado, o usuário pode executar diversas funcionalidades, como criar ou ingressar em uma equipe, registrar e gerenciar tempo, e obter dados.

5. ARQUITETURA DO SISTEMA

A arquitetura do sistema *e-Time* foi projetada para otimizar a integração entre o *frontend* e o *backend*, aproveitando as vantagens oferecidas pelo Next.js (Vercel, 2024). Diferentemente de arquiteturas que separam essas duas camadas em ambientes distintos, o *e-Time* utiliza uma abordagem unificada, na qual tanto o *frontend* quanto o *backend* coexistem dentro do mesmo *framework*. Esse modelo simplifica o desenvolvimento e a manutenção do sistema, além de permitir uma comunicação mais eficiente entre as partes.

Figura 4 – Modelagem lógica do banco de dados

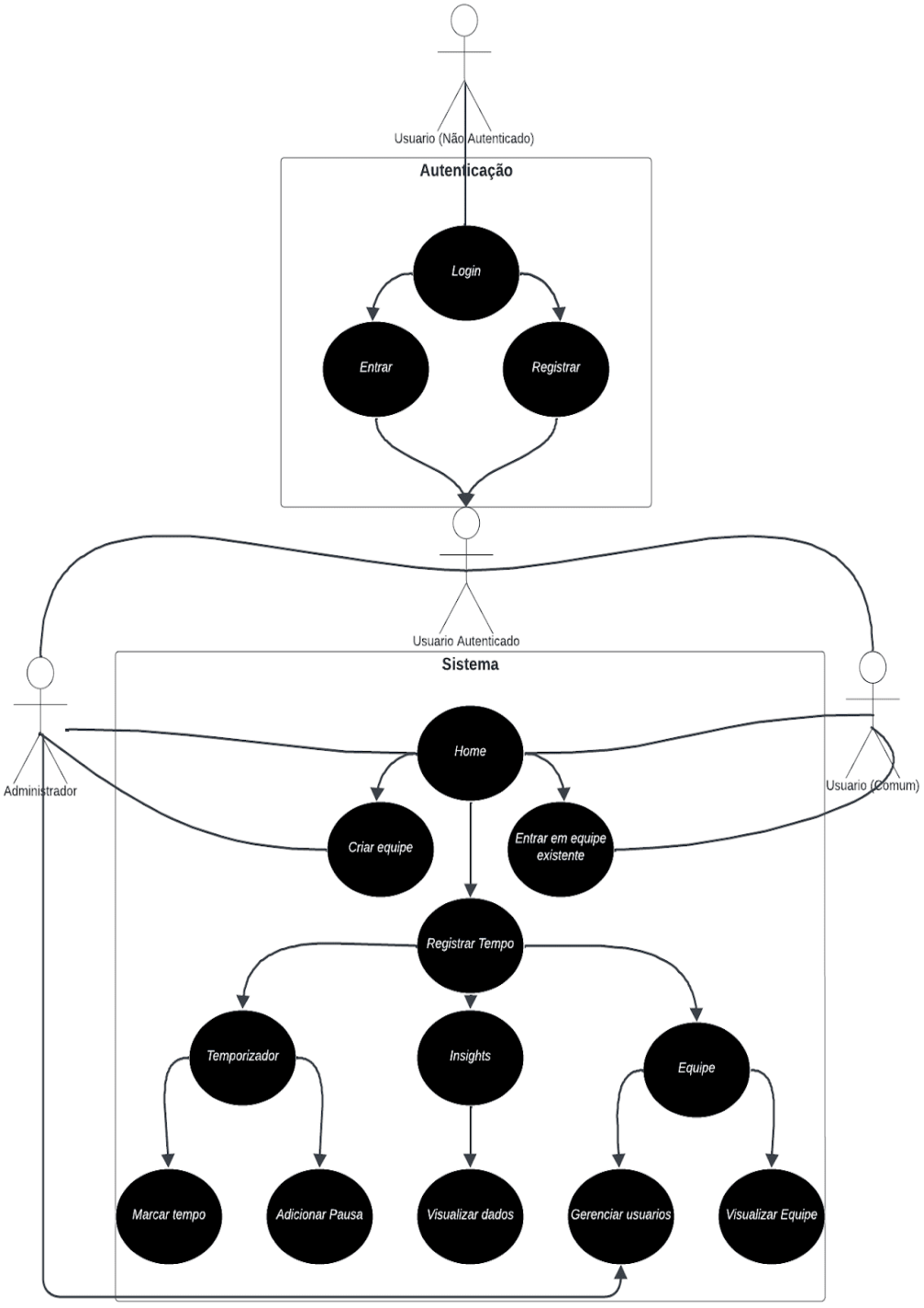


FONTE: OS AUTORES, 2024.

5.1 VISÃO GERAL DA ARQUITETURA

O uso das *Server Actions* no Next.js é um dos pontos-chave dessa arquitetura, permitindo que a lógica de negócios seja executada diretamente no servidor dentro do ambiente do próprio Next.js (Vercel, 2024). Isso elimina a necessidade de gerenciar um *backend* separado, tornando o fluxo de dados entre o *frontend* e o *backend* mais direto e eficiente. Além disso, *APIs* personalizadas são utilizadas para lidar com funcionalidades específicas, como a manipulação de dados no banco de dados ou a gestão de autenticação de usuários. Esse *design* centralizado não apenas melhora a performance do sistema, como também facilita a evolução do sistema, permitindo que novas funcionalidades sejam adicionadas sem a complexidade de integrar diferentes ambientes.

Figura 5 – Modelagem do diagrama de caso de uso



FONTE: OS AUTORES, 2024.

5.2 COMPONENTES DO SISTEMA

A camada de apresentação, desenvolvida com Next.js, oferece uma interface dinâmica e responsiva para os usuários (Vercel, 2024). Componentes como *shadcn-ui* e *Radix-UI* foram integrados para proporcionar uma experiência de usuário

moderna e eficiente, com foco em acessibilidade e usabilidade. Ao usar a renderização no servidor (SSR), o sistema consegue garantir tempos de resposta rápidos, o que melhora a experiência geral de uso, especialmente em casos em que o volume de dados processados é elevado. Com o Next.js atuando como *frontend* e *backend*, o ciclo de comunicação entre a interface do usuário e o processamento de dados se torna mais curto e eficiente.

Por outro lado, a camada de aplicação, que lida com as regras de negócio do sistema, também é implementada dentro do Next.js, com as Server *Actions* desempenhando um papel central (Vercel, 2024). Essas ações permitem que a lógica seja executada no servidor assim que um usuário interage com a interface, garantindo que operações como o registro de horas ou a criação de equipes sejam tratadas rapidamente e com segurança. O *backend*, embora simplificado pela utilização do Next.js, ainda utiliza *APIs* específicas para funções mais complexas e manipuladoras de dados, oferecendo flexibilidade e controle em áreas críticas do sistema.

O gerenciamento de estado do aplicativo é feito utilizando Zustand, uma biblioteca leve para gerenciar estados globais e locais da aplicação (PMNDRS, 2024). Essa solução é integrada de forma a garantir que as informações sobre o estado do usuário, suas jornadas de trabalho, pausas e intervalos, sejam sincronizadas de maneira eficiente e rápida, proporcionando uma experiência de usuário sem interrupções.

5.3 INTEGRAÇÃO ENTRE AS TECNOLOGIAS UTILIZADAS

O banco de dados SQLite é responsável por armazenar todas as informações relacionadas ao sistema, desde os usuários até os registros de jornada e pausas (SQLite, 2024). Para otimizar a interação com o banco de dados, o sistema utiliza o Prisma ORM, que facilita as consultas e atualizações de dados (PRISMA, 2024). A integração do Prisma com o Next.js é fluida, garantindo que as operações de leitura e escrita ocorram de maneira eficiente e segura, mesmo em cenários de alta demanda.

A segurança e o gerenciamento de usuários são tratados de forma robusta no sistema, utilizando o Clerk para autenticação e controle de acesso (CLERK, 2024). Essa solução oferece uma integração direta com o Next.js, garantindo que a proteção de dados seja mantida em todas as interações entre o usuário e o sistema. Isso é especialmente importante em um sistema como o *e-Time*, em que informações sensíveis, como registros de trabalho e dados de equipe, precisam ser protegidas de maneira eficaz.

O uso de Tauri para criar uma versão *desktop* da aplicação, utilizando a mesma base de código do *frontend* web, é outro diferencial na arquitetura do *e-Time* (TAURI, 2024). Essa abordagem permite que os usuários acessem o sistema de forma nativa em seus *desktop*s, sem sacrificar a performance ou a funcionalidade. Isso não apenas amplia o alcance do sistema, mas também simplifica o desenvolvimento, já que ambas as versões (web e *desktop*) compartilham o mesmo código base, minimizando a necessidade de manter dois sistemas distintos.

A arquitetura integrada do *e-Time*, com *frontend* e *backend* centralizados no Next.js, juntamente com o uso de *Server Actions*, *APIs*, Prisma e Zustand, resulta em um sistema ágil, eficiente e escalável. Essa estrutura permite que o *e-Time* atenda de maneira eficaz às necessidades dos profissionais, proporcionando uma

experiência de uso fluida e segura, ao mesmo tempo em que facilita o desenvolvimento contínuo e a adição de novas funcionalidades à medida que o sistema evolui.

6. IMPLEMENTAÇÃO

Nesta seção, será detalhado o processo de desenvolvimento do sistema *e-Time*, desde a escolha das tecnologias até a implementação de funcionalidades específicas no *frontend* e *backend*. O foco será na integração dessas tecnologias para criar um aplicativo eficiente e escalável para o controle de horários de profissionais, desenvolvido exclusivamente para *desktop*.

6.1 DESCRIÇÃO DO PROCESSO DE DESENVOLVIMENTO

O processo de desenvolvimento do *e-Time* foi conduzido utilizando uma abordagem ágil, com iterações rápidas e constantes revisões. O uso de ferramentas como Next.js para o *frontend* e Node.js para o *backend* permitiu uma integração fluida entre cliente e servidor, utilizando TypeScript em todo o *stack* para garantir maior segurança e legibilidade no código.

Para o gerenciamento de estado no *frontend*, foi adotado *Zustand*, uma biblioteca leve e eficiente para gerenciar o estado global da aplicação. O banco de dados escolhido foi o SQLite, gerenciado por meio do *Prisma ORM*, garantindo flexibilidade na manipulação de dados. Além disso, o aplicativo será disponibilizado exclusivamente como uma aplicação *desktop*, utilizando Tauri para essa versão.

6.2 DETALHES DA IMPLEMENTAÇÃO DO *FRONTEND* EM NEXT.JS

O *frontend* foi desenvolvido utilizando Next.js, um framework que oferece suporte para a renderização no servidor, mas neste projeto foi utilizado apenas para criar a interface do aplicativo *desktop*, em conjunto com Tauri para empacotamento e distribuição. Zustand foi utilizado para gerenciar o estado da aplicação, principalmente em relação à manipulação de tempos e intervalos dos usuários. O código de configuração da store do Zustand, que gerencia o estado dos tempos, pode ser visto na Figura 6.

Essa *store* mostrada na Figura 6 permite controlar o estado de um temporizador de maneira eficiente, mantendo o estado salvo localmente com o auxílio do *middleware persist*. Funções como *toggleRunning*, *togglePaused* e *toggleStopped* ajudam a gerenciar o ciclo de vida do temporizador, enquanto *setGoal* controla a meta de trabalho que o usuário tem.

6.3 DESENVOLVIMENTO DO *BACKEND* EM NODE.JS

O *backend* foi desenvolvido utilizando Node.js, uma plataforma leve e eficiente, ideal para aplicações que requerem escalabilidade e performance. O *backend* foi implementado diretamente no ambiente Next.js, utilizando ações de servidor para facilitar a comunicação entre o *frontend* e o *backend* em tempo real.

Figura 6 – Desenvolvimento frontend

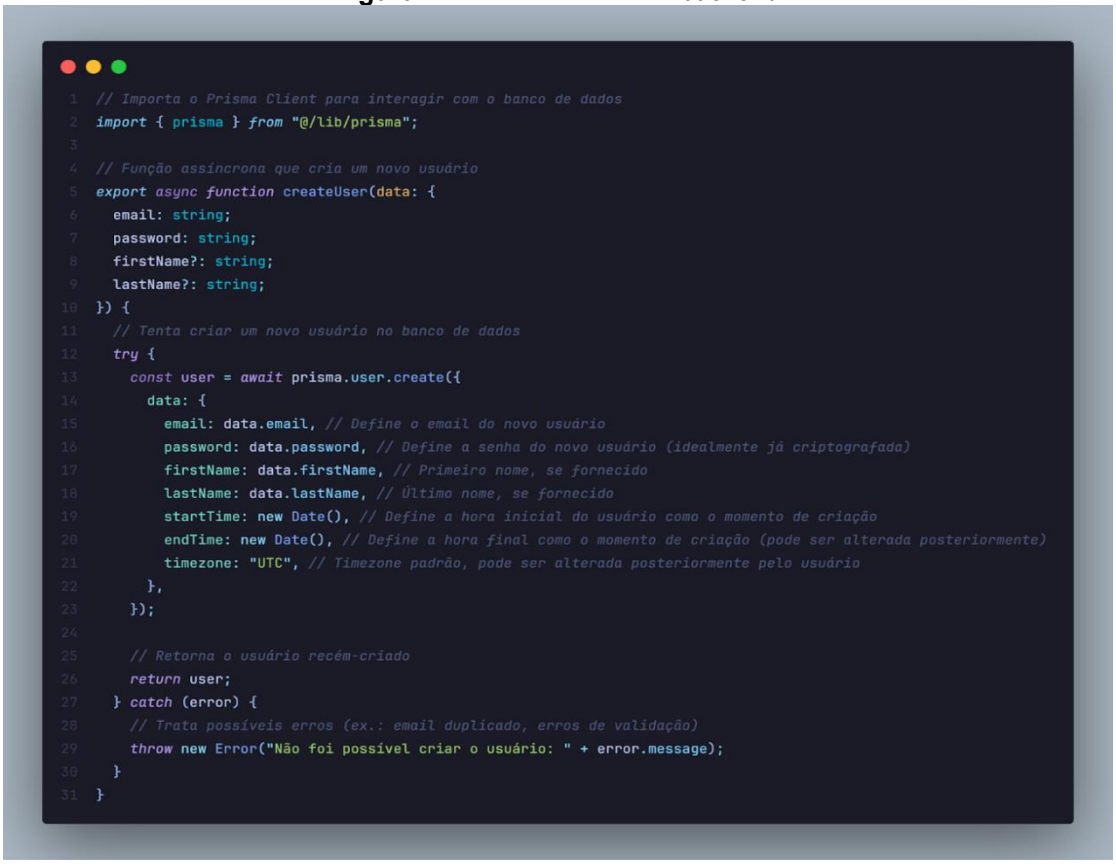
```
import { create } from "zustand"; // Importa a função create da biblioteca Zustand para criar uma store.

type TimerState = {
  time: number; // Tempo atual.
  isRunning: boolean; // Indica se o temporizador está rodando.
  setTime: (time: number) => void; // Define o valor de time.
  toggleRunning: () => void; // Alterna o estado de isRunning.
  reset: () => void; // Reseta o estado ao valor inicial.
};

// Cria a store do temporizador usando Zustand.
const useTimer = create<TimerState>((set) => ({
  time: 0,
  isRunning: false,
  setTime: (time: number) => set({ time }),
  toggleRunning: () => set((state) => ({ isRunning: !state.isRunning })),
  reset: () => set({ time: 0, isRunning: false }),
}));

export default useTimer; // Exporta a store para ser usada em outros componentes.
```

FONTE: OS AUTORES, 2024.

A arquitetura do *backend* foi implementada utilizando *server actions* do Next.js, o que permite realizar operações diretamente no servidor sem a necessidade de criar *APIs* em todos os pontos. As *APIs* RESTful foram utilizadas apenas em alguns casos específicos, enquanto a maioria das operações *CRUD* (*Create, Read, Update, Delete*) no banco de dados SQLite é gerenciada por meio das *server actions*. Para essas operações, foi utilizado o *Prisma ORM*, que oferece uma interface de manipulação de dados intuitiva e otimizada para interagir com o banco de dados de maneira eficiente. Na Figura 7 podemos ver um exemplo de uma rota que realiza a criação de um registro de tempo no banco de dados utilizando Prisma dentro de uma ação de servidor no Next.js.

Este código ilustra de maneira clara como o Prisma facilita a interação com o banco de dados. Tudo começa com a importação do Prisma Client, que torna mais simples a execução de operações como a criação de registros. Em seguida, temos a função assíncrona createUser, responsável por receber dados como e-mail e senha, e registrar um novo usuário no banco. Por ser assíncrona, essa função permite que o código continue rodando sem interrupções, otimizando o desempenho. O Prisma realiza a inserção dos dados, preenchendo os campos necessários e configurando automaticamente os horários de criação. Se ocorrer algum erro, como um e-mail já cadastrado, o código está preparado para lançar uma exceção e tratar o problema de forma adequada. Esse processo, ilustrado na Figura 7, mostra como uma *server action* do Next.js pode ser utilizada para manipular dados no banco de maneira eficiente e segura.

Figura 7 – Desenvolvimento *backend*



```
// Importa o Prisma Client para interagir com o banco de dados
import { prisma } from "@/lib/prisma";

// Função assíncrona que cria um novo usuário
export async function createUser(data: {
  email: string;
  password: string;
  firstName?: string;
  lastName?: string;
}) {
  // Tenta criar um novo usuário no banco de dados
  try {
    const user = await prisma.user.create({
      data: {
        email: data.email, // Define o email do novo usuário
        password: data.password, // Define a senha do novo usuário (idealmente já criptografada)
        firstName: data.firstName, // Primeiro nome, se fornecido
        lastName: data.lastName, // Último nome, se fornecido
        startTime: new Date(), // Define a hora inicial do usuário como o momento de criação
        endTime: new Date(), // Define a hora final como o momento de criação (pode ser alterada posteriormente)
        timezone: "UTC", // Timezone padrão, pode ser alterada posteriormente pelo usuário
      },
    });

    // Retorna o usuário recém-criado
    return user;
  } catch (error) {
    // Trata possíveis erros (ex.: email duplicado, erros de validação)
    throw new Error("Não foi possível criar o usuário: " + error.message);
  }
}
```

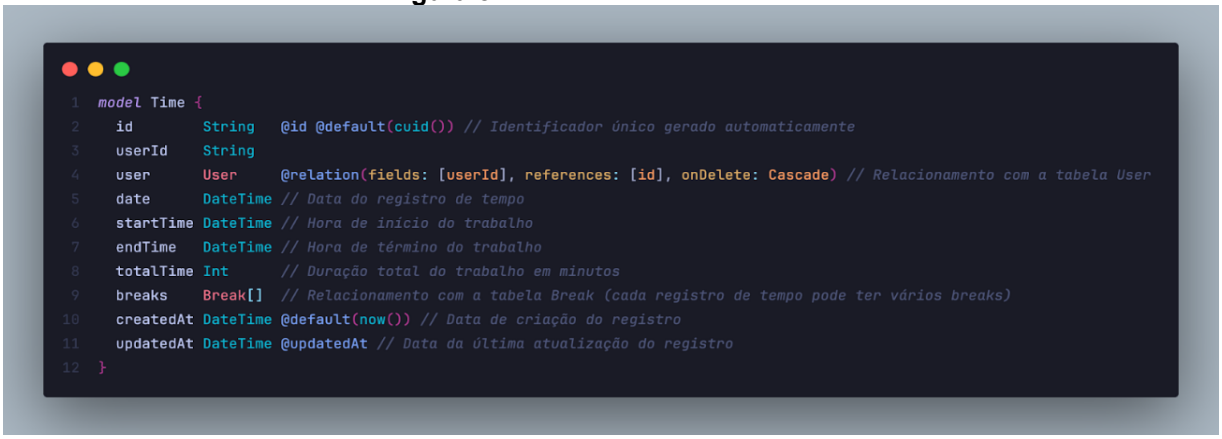
FONTE: OS AUTORES, 2024.

O *e-Time* utiliza SQLite como banco de dados, com o *Prisma ORM* para a gestão de entidades e operações no banco. O SQLite foi escolhido por sua simplicidade e eficiência para aplicativos de pequeno e médio porte, sendo perfeitamente adequado para o cenário do *e-Time*, que gerencia dados de usuários, times e tempos de trabalho.

O Prisma facilita a modelagem das tabelas, a criação de consultas complexas e a migração de dados. Para manter a consistência, as entidades como users, teams, times e breaks foram definida dentro do Prisma, assegurando que o sistema seja capaz de escalar conforme o número de usuários cresça. Na Figura 8 podemos ver como está a definição da tabela Time no Prisma Schema.

A tabela "Time" apresentada na Figura 8, registra as informações sobre o tempo de trabalho de um usuário. O campo ID é o identificador único, gerado automaticamente, enquanto o userId faz referência ao usuário que realizou o registro, ligando-se à tabela User. O campo date armazena a data do registro, e os campos startTime e endTime indicam os horários de início e término do trabalho. A duração total, em minutos, é calculada e armazenada no campo totalTime. Além disso, a tabela se relaciona com a tabela Break, que registra os intervalos ou pausas feitas durante o período de trabalho.

Figura 8 – Desenvolvimento frontend



```
model Time {
  id        String   @id @default(cuid()) // Identificador único gerado automaticamente
  userId    String
  user      User     @relation(fields: [userId], references: [id], onDelete: Cascade) // Relacionamento com a tabela User
  date      DateTime // Data do registro de tempo
  startTime DateTime // Hora de início do trabalho
  endTime   DateTime // Hora de término do trabalho
  totalTime Int      // Duração total do trabalho em minutos
  breaks    Break[]  // Relacionamento com a tabela Break (cada registro de tempo pode ter vários breaks)
  createdAt DateTime @default(now()) // Data de criação do registro
  updatedAt DateTime @updatedAt // Data da última atualização do registro
}
```

FONTE: OS AUTORES, 2024.

7 APRESENTAÇÃO DOS RESULTADOS OBTIDOS

O aplicativo *e-Time* foi testado internamente para garantir sua funcionalidade no controle de horas trabalhadas por colaboradores em *home office*. Durante a fase de testes, realizamos avaliações de desempenho e usabilidade utilizando o *software* em situações reais. Ao longo das sessões de teste, foram coletadas imagens do aplicativo em funcionamento, demonstrando o uso cotidiano do *e-Time* para o registro e acompanhamento das horas de trabalho. As capturas de tela apresentadas nas Figuras 9A e 9B, demonstram o aplicativo em ação, registrando horas de trabalho de forma precisa e sem interrupções.

Na Figura 9A, é possível visualizar o *layout* da interface e como ela facilita o acompanhamento do tempo de forma clara e objetiva.

Os testes internos realizados com o *e-Time* demonstraram sua eficiência no controle de horas trabalhadas. O aplicativo registrou com precisão o tempo dedicado às atividades, além de fornecer gráficos e comparativos claros sobre o progresso e as horas acumuladas, como ilustrado na Figura 9B. A simplicidade na visualização dos dados contribuiu para uma análise rápida e prática do desempenho.

Um dos recursos mais importantes do *e-Time* é sua simplicidade de uso, que se reflete diretamente no *feedback* positivo que tivemos durante os testes. Na Figura 10B está apresentada a funcionalidade de controle de horas, na qual o *modal* de adicionar pausas ao trabalho está aberto, facilitando a inserção de intervalos durante o monitoramento das atividades diárias.

Mesmo que o *e-Time* tenha demonstrado um bom desempenho durante os testes, algumas limitações técnicas foram identificadas no desenvolvimento. A integração entre as tecnologias utilizadas, como Next.js, Tauri e SQLite, apresentou desafios, especialmente em termos de sincronização e desempenho em ambientes de *hardware* mais simples ou em condições de conexão de internet instável. Esses fatores impactaram o tempo de resposta do *software* em certas situações, e melhorias futuras serão focadas em otimizar a performance nesses cenários.

Figura 9A– Tela de registro de tempo

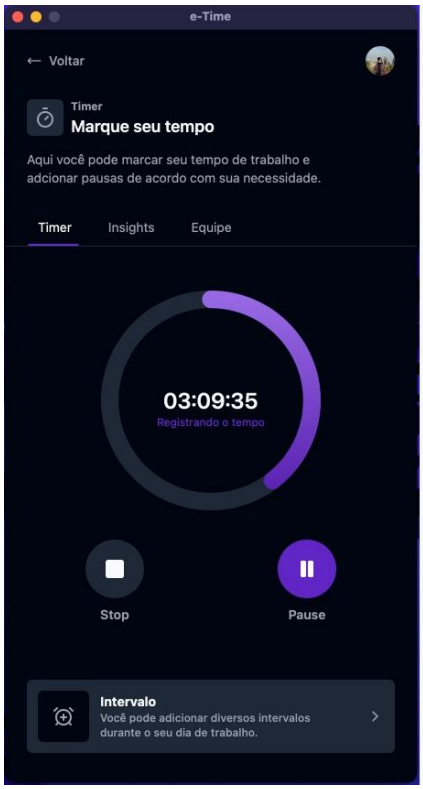


Figura 9B– Tela de exibição dos insights

FONTE: OS AUTORES, 2024.

Figura 10A– Tela de exibição da equipe

Figura 10B– Seletor de intervalo

FONTE: OS AUTORES, 2024.

Embora a interface seja intuitiva, algumas melhorias podem ser implementadas para aumentar a clareza de certos recursos. A área destacada na Figura 10A representa um ponto que pode ser aprimorado para proporcionar uma melhor experiência ao usuário ao acompanhar o progresso do grupo de trabalho.

8 CONSIDERAÇÕES FINAIS

Com base no desenvolvimento do *e-Time*, acredita-se que a solução aqui apresentada tem um grande potencial para atender às demandas do mercado de trabalho remoto, conforme discutido ao longo deste projeto. O processo de criação desta plataforma proporcionou uma série de aprendizados e experiências valiosas, permitindo-nos compreender como estruturar, documentar e implementar um *software* capaz de lidar com a gestão precisa do tempo de trabalho em um ambiente distribuído globalmente. Embora o projeto ainda esteja em fase de desenvolvimento contínuo, foram construídas funcionalidades essenciais, como o registro de horas trabalhadas, a adaptação automática a fusos horários e o gerenciamento de banco de horas. Esses elementos, embora ainda passíveis de refinamentos, já demonstram a aplicabilidade e o valor da solução, com grande potencial de serem aprimorados e reutilizados em futuras atualizações.

Além disso, o desenvolvimento do *e-Time* nos proporcionou uma visão prática das tecnologias e metodologias utilizadas no mercado atual, confirmando a relevância de ferramentas como Next.js, Prisma e Zustand na construção de aplicações robustas e seguras.

A experiência de planejar, desenvolver e testar a plataforma trouxe não apenas conhecimento técnico, mas também a importância de uma gestão eficiente do tempo e a flexibilidade necessária para lidar com os desafios que surgem ao longo do processo.

Esse projeto nos permitiu crescer tanto academicamente quanto profissionalmente, reforçando nossa capacidade de adaptação e nos preparando para enfrentar futuros desafios no mercado de trabalho. A experiência adquirida com o *e-Time* nos deu uma visão clara da importância de soluções tecnológicas bem planejadas e documentadas, bem como da necessidade de contínua evolução para atender às exigências de um cenário de trabalho em constante mudança.

Além de seu impacto prático no mercado de trabalho remoto, o desenvolvimento do *e-Time* também evidenciou a importância de criar soluções tecnológicas que sejam acessíveis e escaláveis. A capacidade de adaptar a plataforma a diferentes realidades empresariais e contextos geográficos, sem comprometer a precisão no controle das jornadas, é um diferencial que posiciona o *e-Time* como uma ferramenta estratégica para possíveis inserções em outros *softwares*. Esse nível de flexibilidade e personalização torna a plataforma versátil para empresas de todos os portes, ajudando-as a otimizar seus processos de gestão de tempo de maneira intuitiva e eficiente.

Por fim, o desenvolvimento do e-Time nos proporcionou uma compreensão mais aprofundada sobre o ciclo de vida de um projeto. Desde a concepção até a implementação, o processo nos permitiu adaptar o sistema às necessidades reais dos usuários e ao ambiente dinâmico do mercado. Esse aprendizado será essencial não apenas para melhorias futuras no e-Time, mas também para novos projetos que venhamos a desenvolver, consolidando uma visão

mais estratégica e focada no aprimoramento contínuo da tecnologia e da experiência do usuário.

REFERÊNCIAS

ARAÚJO, J. V.; JUNIOR, A. L. A ergonomia no home office: a relevância da ergonomia no trabalho em casa. 2022. Disponível em: <https://www.semanticscholar.org/paper/A-Ergonomia-no-Home-Office%3A-a-Relev%C3%A2ncia-da-no-em-Ara%C3%BAjo-Junior/15c79890cec307b0cba384bd065e740c996a4c0f>. Acesso em: 2 ago. 2024.

HAUBRICH, D. B.; FROEHLICH, C. Benefícios e desafios do home office em empresas de tecnologia da informação. *Gestão & Conexões - Management and Connections Journal*, Vitória (ES), v. 9, n. 1, p. 167-184, jan./abr. 2020. Disponível em: https://doi.org/10.13071/regec.2317-5087.2020.9.1.27901.167-184. Acesso em: 23 mai. 2024.

BOEHM, Camila. Número de empresas que adotam home office deve crescer 30% após pandemia. 2020. Disponível em: <https://agenciabrasil.ebc.com.br/geral/noticia/2020-04/numero-de-empresas-adotam-home-office-deve-crescer-30-apos-pandemia>. Acesso em: 30 jul. 2024.

CERTO, D. R. M.; FERREIRA, A. L. O home-office na percepção dos trabalhadores de uma indústria automobilística. 2023. Disponível em: <https://www.semanticscholar.org/paper/O-home-office-na-percep%C3%A7%C3%A3o-dos-trabalhadores-de-uma-Certo-Ferreira/91631eb2445f5085f2743990d69acdae520c23e9>. Acesso em: 4 fev. 2024.

CLERK. Clerk Documentation. Disponível em: https://clerk.dev/docs. Acesso em: 14 abr. 2024.

CLOCKIFY Reviews on Capterra. Disponível em: <https://www.capterra.com/p/175045/Clockify/reviews/>. Acesso em: 15 mai 2024.

HERRON, A. Node.js 8 the Right Way: Practical, Server-Side JavaScript That Scales. Pragmatic Bookshelf, 2018.

NAVARINI, M. P.; PEREIRA, M. S. da C. de P. Modelo de trabalho híbrido: análise dos impactos e perspectivas. 2021. Trabalho de Conclusão de Curso (Graduação em Engenharia da Produção) – Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2021. Disponível em: http://www.repositorio.poli.ufrj.br/monografias/monopoli10032974.pdf. Acesso em: 19 jun. 2024

NODE.JS FOUNDATION. Node.js Documentation. Disponível em: <https://nodejs.org/en/docs/>. Acesso em: 4 abr. 2024.

ORACLE CORPORATION. MySQL Documentation. Disponível em: <https://dev.mysql.com/doc/>. Acesso em: 15 mar. 2024.

PMNDRS. Zustand Documentation. Disponível em: https://docs.pmnd.rs/zustand/getting-started/introduction. Acesso em: 28 jul. 2024.

PRISMA. Prisma Documentation. Disponível em: <https://www.prisma.io/docs/>. Acesso em: 2 ago. 2024.

ROCHA, C. T. M.; AMADOR, F. S. O teletrabalho: conceituação e questões para análise. Cad. EBAPE.BR, Rio de Janeiro, v. 16, n. 1, 2018. Disponível em: https://www.scielo.br/j/cebape/a/xdbDYsyFztnLT5CVwpxGm3g/abstract/?lang=pt#. Acesso em: 1 ago. 2024.

SANTANA, M. M. B. R. HOME OFFICE: Uma análise acerca dos benefícios e prejuízos proporcionados aos empregados por esta modalidade empregatícia após a reforma trabalhista. 2018. 28 f. Trabalho de Conclusão de Curso (Bacharelado em Direito) - Centro Universitário Tabosa de Almeida – ASCES/UNITA, Caruaru, 2018. Disponível em: . Acessado em: 28 maio 2023.

SILVA, G. P.; SILVA, M. A. Riscos para as empresas com o home office e as horas extras. 2023. Disponível em: <https://www.semanticscholar.org/paper/RISCOS-PARA-AS-EMPRESAS-COM-O-HOME-OFFICE-E-AS-SILVA-SILVA/f614dc152dcb6692e00600f69041d39bfc0c2cd2>. Acesso em: 15 ago. 2024.

SOUSA, P. S.; SARAIVA, C. F. O controle da jornada de trabalho do profissional home office durante a pandemia da Covid-19. 2023. Disponível em: <https://www.semanticscholar.org/paper/O-CONTROLE-DA-JORNADA-DE-TRABALHO-DO-PROFISSIONAL-A-Sousa-Saraiva/812f38869c90f6a7680782efc603680d59f60c64>. Acesso em: 22 ago. 2024.

SQLite Documentation. Disponível em: https://www.SQLite.org/docs.html. Acesso em: 30 mai. 2024.

TAURI. Tauri Documentation. Disponível em: https://tauri.app/. Acesso em: 30 fev. 2024.

TOGGL Track. Disponível em: <https://toggl.com/track/>. Acesso em: 15 maio 2024. TOGGL Track Reviews on G2. Disponível em: <https://www.g2.com/products/toggl-track/reviews>.

VERCEL Next.js Documentation. Disponível em: https://nextjs.org/docs. Acesso em: 27 ago. 2024.

Repositório do software:

PINHEIRO, Vinícius. e-Time: Sistema de gerenciamento de tempo para trabalho remoto. Disponível em: https://github.com/vinifpinheiro/tcc-e-Time. Acesso em: 27 set. 2024.