MAURÍCIO JÚNIOR SANTOS FREIRE

RELATÓRIO DE ESTÁGIO – DESENVOLVIMENTO DE SOFTWARE DE COLETA DOMICILIAR NA EMPRESA MGCODE

LAVRAS – MG

2023

MAURÍCIO JÚNIOR SANTOS FREIRE

RELATÓRIO DE ESTÁGIO – DESENVOLVIMENTO DE *SOFTWARE* DE COLETA DOMICILIAR NA EMPRESA MGCODE

INTERNSHIP REPORT – DEVELOPMENT OF HOME COLLECTION SOFTWARE AT MGCODE COMPANY

Relatório de estágio supervisionado apresentado à Universidade Federal de Lavras, como parte das exigências do Curso de Sistemas de Informação, para obtenção do título de Bacharel.

APROVADA EM 04 de dezembro de 2023.

Dr. Antônio Maria Pereira de Resende

Dr. Rafael Serapilha Durelli

Dra. Renata Teles Moreira

gov.br
Documento assinado digitalmente
ANTONIO MARIA PEREIRA DE RESENDE
Data: 11/12/2023 11:10:53-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Antônio Maria Pereira de Resende
Orientador

LAVRAS – MG

2023

AGRADECIMENTOS

Primeiramente, sou imensamente grato a Deus por ter me guiado, dado forças e sabedoria ao longo dessa árdua trajetória.

À minha família, quero expressar minha profunda gratidão por seu amor incondicional, apoio constante e compreensão durante as etapas da minha jornada acadêmica. Vocês são a razão disso aqui.

Aos meus amigos com quem dividi minha moradia na cidade de Lavras: Carlos Eduardo Leopoldino Santos, Gustavo Antônio Alves Dutra, Ryan Newber Silva, Deyne Dehon de Oliveira, David Miller de Oliveira, Rafael Campidelli e Euller Júnior.

Aos meus amigos de faculdade, Gustavo Alvarenga Marzoque, Pedro Gabriel Camargo Calisto, Gabriel Fernandes Vieira, Maurício Farah Cordeiro Júnior e Gabrielly Camargo Corrêa, pelos conhecimentos, experiências e muitas risadas. Foi um gás conviver com vocês, sentirei falta.

Não posso deixar de agradecer também ao professor Doutor Antônio Maria Pereira de Rezende, por aceitar me orientar durante este trabalho.

À empresa MGCode e ao Jacson Soares, por me conceder a oportunidade de adentrar ao mercado profissional pela primeira vez, onde ganhei conhecimento técnico e pessoal. São profissionais ímpares.

Em resumo, este trabalho não teria sido possível sem o apoio e contribuição de cada um de vocês. Minha gratidão é imensa e eterna.

RESUMO

Este trabalho apresenta o relatório do estágio supervisionado realizado na MGCode, uma empresa de consultoria e desenvolvimento de *softwares*. Ele detalha as principais atividades realizadas, porém com ênfase no desenvolvimento de um *software* de coleta domiciliar. O relatório abrange diversas áreas de conhecimento exploradas e demonstra como a experiência em uma empresa cujas atividades se concentram em programação e manipulação de dados pode contribuir para o desenvolvimento das habilidades técnicas e de trabalho em equipe de um profissional.

Palavras-chave: Relatório de estágio. MGCode. Desenvolvimento *web*.

ABSTRACT

This paper presents the report of the supervised internship carried out at MGCode, a software development and consultancy company. It details the main activities carried out, but with an emphasis on the development of home collection software. The report covers the various areas of knowledge explored and demonstrates how experience in a company whose activities focus on programming and data manipulation can contribute to the development of a professional's technical and teamwork skills.

Keywords: Internship report. MGCode. Web development.

LISTA DE FIGURAS

LISTA DE SIGLAS

SQL	Standard Query Language
YARN	Yet Another Resource Negotiator
DBA	Database Administrator
API	Application Programming Interface
HTTP	Hypertext Transfer Protocol
REST	Representational State Transfer
CRUD	Create Read Update Delete
URL	Uniform Resource Locator

SUMÁRIO

1 INTRODUÇÃO

Um dos requisitos para conclusão e obtenção do título de Bacharel em Sistemas de Informação na UFLA é a aprovação do Trabalho de Conclusão de Curso (TCC). Este pode ser apresentado em diferentes modelos, sendo um deles o Relatório de Estágio. O estágio profissional é uma das possibilidades para a aplicação dos conhecimentos adquiridos durante o período acadêmico. Este relatório apresenta atividades aplicadas e conhecimentos obtidos na MGCode durante o período de 18 de maio a 18 de dezembro de 2022.

A MGCode é uma *startup* situada em Perdões (MG). Ela iniciou sua trajetória como parte da provedora de Internet MinasNet, atualmente Sempre Internet. Posteriormente, desvinculou-se e tornou-se uma *startup* em consultoria e desenvolvimento de *software*, especializada em integrações HL7[1], DICOM[2] e HealthCare[3], atuando fortemente na área da saúde com empresas e hospitais mais conceituados do mercado. Também oferece uma gama de soluções para os mais diversos comércios, desenvolvendo soluções adequadas para cada negócio.

O foco da empresa num primeiro momento consistiu na prestação de serviços em integração de sistemas e migração de dados, sendo essa a área de experiência do sócio proprietário. A MGCode também desenvolve soluções tecnológicas para empresas regionais, oferecendo um sistema de vendas *desktop* desenvolvido em Delphi. Essa solução atende às necessidades de administração de vendas no comércio de produtos diversos e na gestão de sócios em clubes esportivos.

1.1 Contextualização

O FEMME, um renomado laboratório médico especializado na saúde da mulher, abordou a MGCode com o intuito de adotar métodos modernos para otimizar seus procedimentos e garantir um atendimento de qualidade e conveniência às pacientes. Uma das áreas atuantes do FEMME é a coleta domiciliar, permitindo a

[1] O HL7 é um "padrão" utilizado pelo setor de saúde para permitir a troca de informações em saúde através de mensagens

[2] DICOM é a sigla para *Digital Imaging and Communications in Medicine* (Comunicação de Imagens Digitais em Medicina) e representa um conjunto de normas criado para padronizar o formato eletrônico utilizado no armazenamento e na comunicação das imagens.

[3] É um conjunto de filosofia operacionais e método que ajudam a criar o máximo de valor para os pacientes.

realização de exames dos pacientes no conforto de seus lares, evitando deslocamentos desnecessários.

A coleta domiciliar de amostras representa uma solução inovadora para os obstáculos logísticos que, muitas vezes, impedem as pacientes de acessar atendimentos médicos de qualidade. No entanto, coordenar essas atividades demanda uma abordagem meticulosa e organizada. Assim, em parceria com a FEMME, a MGCode desenvolveu um *software* de monitoramento e gestão de coleta domiciliar. Foi projetado para otimizar a coordenação de atendimentos a domicílio, fornecendo uma plataforma centralizada para o planejamento de rotas, agendamento de visitas, registro de informações dos pacientes e monitoramento em tempo real.

1.2 Objetivos

São apresentados nesta seção os objetivos gerais e específicos deste trabalho.

Objetivos gerais

O objetivo do trabalho é descrever e analisar as atividades realizadas durante o estágio como desenvolvedor *front-end* na empresa MGCode. Além disso, enfatizar o papel desempenhado no projeto de desenvolvimento do sistema do Femme.

Objetivos específicos

a) Apresentar as tecnologias e métodos de desenvolvimento utilizados na empresa;
b) Relatar detalhadamente algumas atividades desenvolvidas como desenvolvedor *front-end* durante o estágio;
c) Concluir o relatório, ressaltando as principais conclusões e reflexões sobre a experiência como estagiário desenvolvedor *front-end* na empresa MGCode.

1.3 Estrutura do trabalho

A estrutura do trabalho está organizada da seguinte maneira: no primeiro capítulo, são descritos os objetivos do trabalho; no capítulo dois, são apresentadas as tecnologias empregadas no desenvolvimento das atividades, detalhando as ferramentas e linguagens utilizadas para o projeto; no capítulo três, são descritas e demonstradas algumas atividades realizadas durante o período de estágio. Por fim, no capítulo quatro, é apresentada a conclusão do trabalho.

2 REFERENCIAL TEÓRICO

Esta seção descreve os conceitos e tecnologias utilizadas no desenvolvimento do *software* pelo estagiário, sendo elas: React, TypeScript, SCSS, GitHub e o *Scrum*.

2.1 React

Antes de explicar o React, é preciso entender o que é *front-end*. O termo *front-end* refere-se à parte de um sistema ou aplicação de software que os usuários interagem diretamente. É responsável por apresentar dados e funcionalidades de uma forma que seja visualmente atraente e fácil de usar para os usuários finais. O *front-end* engloba uma variedade de elementos, incluindo cores, animações, imagens, ícones e fontes (JAISWAL; HELIWAL, 2022). O React, também conhecido como React.js, é uma biblioteca *front-end* de JavaScript de código aberto, amplamente utilizada para o desenvolvimento de interfaces de usuário interativas e dinâmicas.

O cerne do React é a ideia de construir interfaces de usuário por meio de componentes, blocos autônomos e reutilizáveis de código, podendo ser combinados para formar uma interface completa. Esses componentes podem conter *HyperText Markup Language* (HTML), *Cascading Style Sheets* (CSS) e lógica de JavaScript, permitindo uma abordagem modular e organizada para o desenvolvimento. O React oferece uma separação clara entre as preocupações de apresentação e de estado através da sua estrutura de componentes, resultando em um código mais fácil de entender, manter e testar.

O ecossistema do React é extenso e inclui diversas ferramentas e bibliotecas complementares, como o React Router, utilizado para roteamento, e o Axios, utilizado

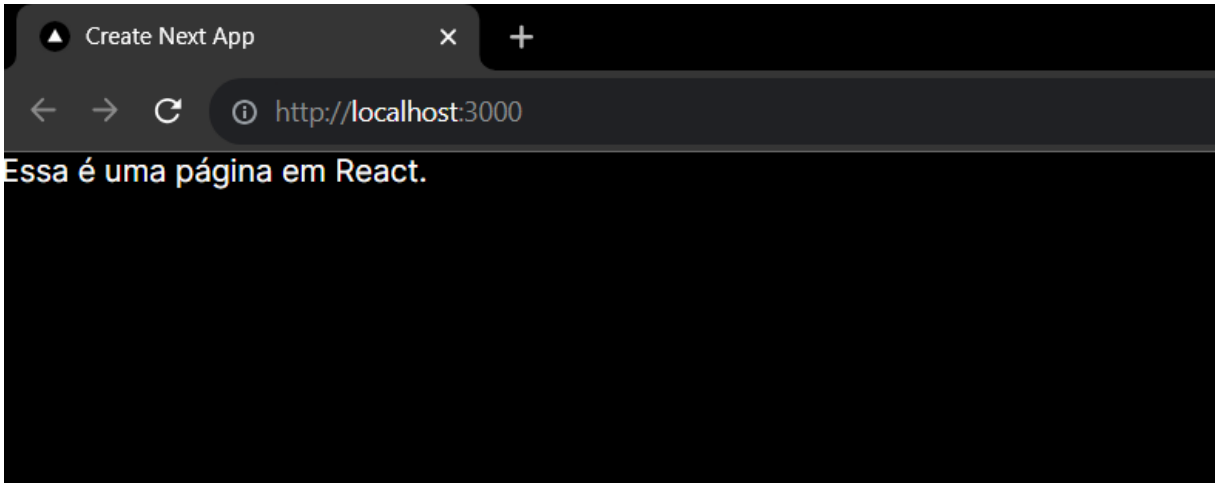
para realizar requisições HTTP. Na Figura 1 – Exemplo de um código em React, é possível visualizar um exemplo de código implementado no arquivo principal do React, e, na Figura 2 – Página exibida pelo código da Figura 1, a página *web*.

Figura 1 – Exemplo de um código em React

```
export default function Home() {
  return (
    <main>
      <p>
        Essa é uma página em React.
      </p>
    </main>
  )
}
```

Fonte: Elaborada pelo autor (2023).

Figura 2 – Página exibida pelo código da Figura 1



Fonte: Elaborada pelo autor (2023).

A escolha do React foi devido à estruturação das pastas serem mais simples. O React foi usado como parte principal do sistema de *front-end* durante o estágio. Usou-se, principalmente, para construir as telas do sistema, requisições HTTP e roteamento entre as páginas do *software*.

2.2 TypeScript

Antes de explicar o TypeScript, é preciso entender o JavaScript. Segundo Suehring e Valade (2013), o JavaScript é usado na criação de páginas da *web* a fim de deixar a experiência do usuário mais interessante e interativa. O fato de ser executado diretamente no navegador dos usuários, permite que os desenvolvedores criem experiências ricas e responsivas, tornando-o uma das tecnologias mais influentes no campo do desenvolvimento *front-end*. Uma das principais características do JavaScript é sua natureza interpretada e baseada em eventos. O código JavaScript é executado à medida que o usuário interage com a página, respondendo a eventos como cliques, digitação e movimentos do *mouse*. Essa abordagem permite a criação de interfaces dinâmicas e fluidas, contribuindo para a experiência do usuário.

No entanto, o JavaScript também apresenta desafios. Sua tipagem dinâmica permite as variáveis mudarem de tipo durante a execução, podendo levar a erros difíceis de rastrear. Além disso, a falta de recursos de estruturação de código pode dificultar a manutenção e escalabilidade. E é nesse contexto que o TypeScript entra em cena. O TypeScript é uma extensão do JavaScript onde aborda muitas dessas limitações, fornecendo tipagem estática, suporte a recursos modernos do JavaScript e ferramentas avançadas de desenvolvimento. TypeScript é uma linguagem de programação de código aberto desenvolvida pela Microsoft que estende o JavaScript, acrescentando recursos de tipagem estática e outras funcionalidades avançadas ao ecossistema de desenvolvimento *web* (TYPESCRIPT, 2023).

O diferencial entre o TypeScript e o JavaScript é a introdução de tipos estáticos. Enquanto o JavaScript (Figura 3 – Exemplo de um código em JavaScript) é uma linguagem de tipagem dinâmica, o TypeScript permite aos desenvolvedores definir explicitamente os tipos de variáveis, parâmetros de função e outros elementos do código (Figura 4 – Exemplo do mesmo trecho de código da Figura 3 em TypeScript).

Figura 3 – Exemplo de um código em JavaScript

```javascript
function greet(name) {
  return "Hello, " + name + "!";
}

console.log(greet("Maurício"));
```

Fonte: Elaborada pelo autor (2023).

Figura 4 – Exemplo do mesmo trecho de código da Figura 3 em TypeScript

```typescript
function greet(name: string): string {
  return "Hello, " + name + "!";
}

console.log(greet("Maurício"));
```

Fonte: Elaborada pelo autor (2023).

Observe a diferença: no código TypeScript, adiciona-se tipos às variáveis e aos parâmetros de função. No exemplo citado, a função *greet* aguarda um argumento do tipo *string* e retorna uma *string*. Ao definir explicitamente o tipo dos parâmetros e do retorno da função, o TypeScript pode realizar verificações de tipo em tempo de compilação para ajudar a evitar erros comuns. Isso é uma das principais vantagens do TypeScript sobre o JavaScript padrão, desempenhando um papel fundamental na criação da lógica das telas do sistema.

2.3 SCSS

Antes de se adentrar no SCSS, é preciso entender o CSS. *Cascading Style Sheets* ou Folhas de Estilo em Cascata são uma linguagem de marcação utilizada para definir a apresentação e o *layout* de documentos HTML e XML. Segundo a W3C Escritório Brasil (2016), o CSS prepara as informações do site para consumir da melhor maneira possível. Introduzido nos anos 90, o CSS permitiu uma separação eficaz entre a estrutura e o estilo dos elementos da página *web*. Isso possibilitou a criação de *designs* visualmente atraentes e aprimorou a manutenção, uma vez que as alterações de estilo podiam ser aplicadas globalmente sem a necessidade de modificar diretamente o conteúdo HTML.

No entanto, à medida que as demandas aumentaram, os desafios do CSS também se tornaram evidentes. A manutenção de estilos em projetos e a necessidade de reutilização de código CSS levaram à busca de soluções mais avançadas. É nesse contexto que a linguagem SCSS entra em jogo.

SCSS é uma extensão do CSS que introduz recursos de programação, tornando a escrita de estilos mais modular, flexível e eficiente. Ele é conhecido como um pré-processador CSS, ou seja, é convertido em CSS padrão antes de ser interpretado pelos navegadores.

Se apresenta como uma evolução natural do CSS, oferecendo recursos avançados, melhorando a organização, a reutilização e a escalabilidade dos estilos. Com sua sintaxe semelhante à do CSS tradicional (Figura 5 – Exemplo de um CSS) e sua capacidade de ser convertido em CSS padrão, o SCSS conquistou uma base sólida de adeptos na comunidade de desenvolvimento *web*, contribuindo para uma abordagem mais eficiente e inovadora (Figura 6 – Exemplo do mesmo trecho da Figura 5, porém com SCSS) na estilização de páginas da internet.

Figura 5 – Exemplo de um CSS

```css
.button {
  background-color: #007bff;
  color: #fff;
  padding: 10px 20px;
  border: none;
}

.button:hover {
  background-color: #0056b3;
}
```

Fonte: Elaborada pelo autor (2023).

Figura 6 – Exemplo do mesmo trecho da Figura 5, porém com SCSS

```scss
$primary-color: #007bff;
$secondary-color: #0056b3;

.button {
  background-color: $primary-color;
  color: #fff;
  padding: 10px 20px;
  border: none;

  &:hover {
    background-color: $secondary-color;
  }
}
```

Fonte: Elaborada pelo autor (2023).

Nesse exemplo (Figura 6 – Exemplo do mesmo trecho da Figura 5, porém com SCSS), consiste no uso de variáveis para as cores primária e secundária, mantendo a estrutura de regras e pseudosseletores. O SCSS permite aproveitar as vantagens da organização modular sem aumentar a complexidade do código. Quando compilado para CSS, o resultado será o mesmo em ambos os casos (Figura 8 – Página exibida pela junção do CSS da Figura 6 e o trecho da Figura 7).

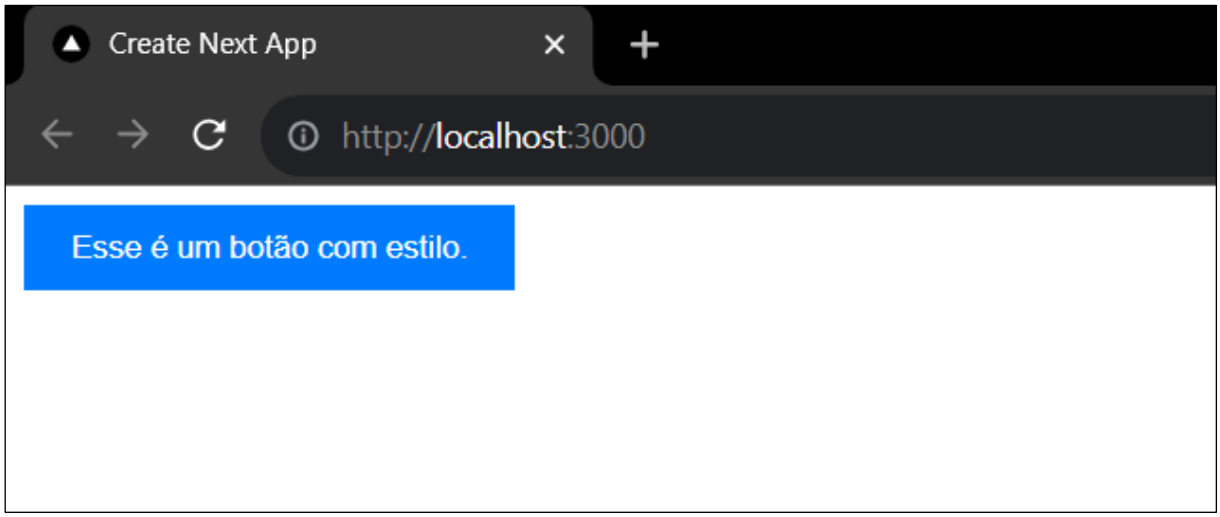
Figura 7 – Exemplo de código em React com um botão estilizado por CSS

```jsx
import './styles.css';

export default function Home() {
  return (
    <main>
      <button className="button">
        Esse é um botão com estilo.
      </button>
    </main>
  )
}
```

Fonte: Elaborada pelo autor (2023).

Figura 8 – Página exibida pela junção do CSS da Figura 6 e o trecho da Figura 7



Fonte: Elaborada pelo autor (2023).

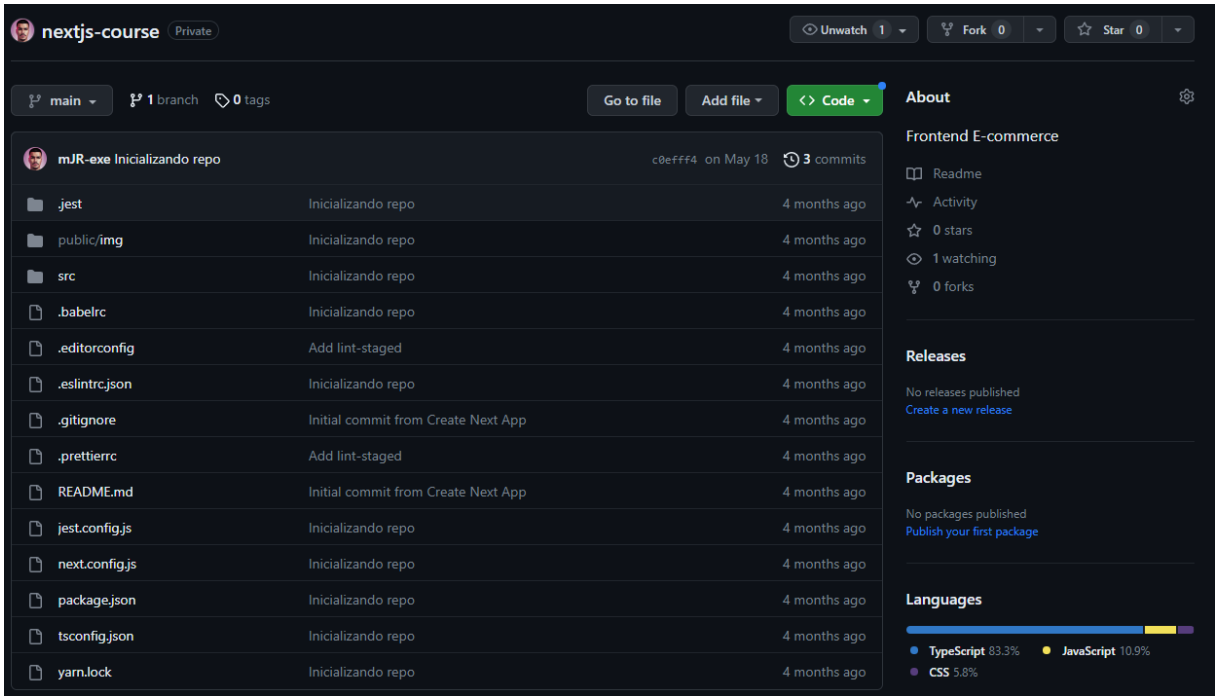
O SCSS desempenhou um papel fundamental na estilização das páginas do sistema durante o estágio, deixando-as mais bonitas e de forma agradável para o usuário, além de ajudar a construir um *software* responsivo.

2.4 GitHub

Segundo a documentação oficial do GitHub (2023), o GitHub é uma plataforma de hospedagem de código para controle de versão e colaboração. Criado em 2008, transformou a forma como os desenvolvedores colaboram, compartilham e contribuem para projetos de código aberto e privado. O GitHub se tornou um ponto central para a comunidade de desenvolvedores, combinando controle de versão, gerenciamento de projetos e colaboração em um único ambiente.

Uma característica fundamental do GitHub é a utilização do Git, um sistema de controle de versão distribuído. O Git permite acompanhar as alterações feitas no código-fonte temporariamente, facilitando a colaboração entre equipes, ramos de desenvolvimento e a manutenção de diferentes versões do *software*. O GitHub fornece uma interface amigável para o Git (Figura 9 – Exemplo de um repositório no Github), tornando a criação de repositórios, a clonagem, a colaboração e o rastreamento de problemas mais acessíveis a uma ampla variedade de desenvolvedores.

Figura 9 – Exemplo de um repositório no Github



Fonte: Elaborada pelo autor (2023).

Utilizou-se o GitHub a fim de possibilitar uma colaboração eficiente entre os desenvolvedores da empresa, facilitando o desenvolvimento colaborativo. Além de ajudar a estabelecer padrões de colaboração.

2.5 *Scrum*

O *Scrum* é um *framework* de gerenciamento de projetos criado por Ken Schwaber e Jeff Sutherland na década de 1990. A metodologia possui um documento oficial, disponível na internet, que exemplifica seus conceitos, definições e regras (SCHWABER; SUTHERLAND, 2020). O *Scrum* é uma metodologia ágil amplamente adotada. Consiste em um *framework* de gerenciamento utilizado pelas equipes para alcançar um objetivo coletivo através da auto-organização. Esse método descreve uma série de encontros, recursos e papéis, visando garantir a eficácia na entrega de projetos (AWS, 2023).

O *Scrum* é utilizado como a metodologia de gerenciamento na MGCode. O *framework Scrum* consiste em equipes associadas a papéis e eventos, como é apresentado na Figura 10 – Funcionamento de um sprint. Esses componentes e a ferramenta utilizada para aplicá-los são descritos nas subseções seguintes.

2.5.1 Equipe *Scrum*

O time *Scrum* possui os seguintes papéis definidos: a) *Product Owner*; b) Time de Desenvolvimento; c) e o *Scrum Master*. O *Product Owner* é o integrante encarregado por maximizar o valor agregado dos entregáveis, gerenciando o *backlog* do produto e garantindo que a sua visualização seja transparente, organizada e priorizada. Também é responsável pelo sucesso do produto, incorporando nele as necessidades descritas pelo mercado e pelos clientes de forma a solucionar problemas explícitos e implícitos. Personifica/representa os potenciais clientes para o time de desenvolvimento, assumindo suas vontades e desejos. Frente ao mercado e potenciais clientes, possui a missão de defender os interesses do seu time de desenvolvimento, respeitando seus limites e coletando a maior quantidade de detalhes possível, na menor granularidade possível.

O time de desenvolvimento é composto por integrantes que assumem a responsabilidade de entregar uma versão utilizável do produto ao final de cada *sprint*, com uma quantidade de incrementos acordada com os outros constituintes do time *Scrum*. Os times possuem liberdade e são autorizados a coordenar suas funções, se responsabilizando pela sinergia, eficiência e eficácia.

O *Scrum Master* é o integrante que assume o compromisso de certificar que a teoria, prática e regra do *Scrum* sejam compreendidas, adotadas e praticadas na rotina do time *Scrum*. Ele possui a missão de auxiliar os membros a aperfeiçoarem as interações a fim de maximizar os benefícios trazidos pelo *framework*. Também faz parte do seu escopo de atuação remover os impedimentos para o progresso do time de desenvolvimento e transmitir a visão, os objetivos e itens do *backlog* de forma clara.

2.5.2 Eventos *Scrum*

Os eventos do *Scrum* são reuniões realizadas pela equipe regularmente para avaliar o andamento das tarefas, o método de execução e o sucesso na conclusão das atividades. Os eventos utilizados pela MGCode são *Sprint Planning*, S*print*, *Daily*, *Sprint Review* e *Sprint Retrospective*.

a) *Sprint Planning* é a reunião do *Scrum*, por onde a equipe decide quais tarefas do *Product Backlog* devem ser realizadas. A definição deve ser

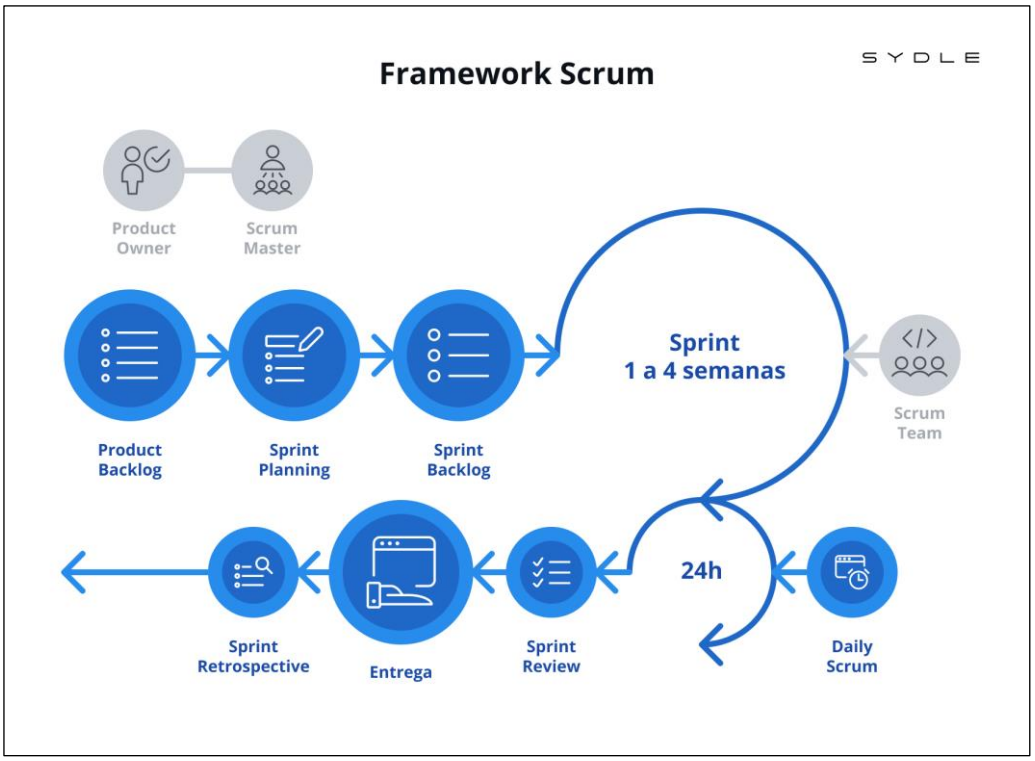
mensurável e alcançável para que, ao fim da reunião, cada integrante da equipe saiba qual produto deve ser entregue ao fim do *sprint*. Essa reunião é feita antes de iniciar qualquer *sprint*. *Sprint* é o período no qual a equipe realizará as tarefas atribuídas durante o *Sprint Planning*. O período é geralmente de duas semanas, mas pode ser determinado pela equipe. A MGCode optou por usar um *sprint* de 15 dias;

b) *Daily* é uma reunião curta realizada diariamente na qual os membros da equipe apresentam suas atividades e solicitam ajuda em caso de dificuldades;

c) *Sprint Review* é a reunião realizada no final de cada *sprint* para avaliar se os objetivos definidos para o *sprint* em questão foram realizados com sucesso;

d) *Sprint Retrospective* é a última reunião realizada em um *sprint* para avaliar o funcionamento e a estagnação no *sprint* em questão. Seu principal objetivo é melhorar o andamento de *Sprints* futuras.

Figura 10 – Funcionamento de um sprint



Fonte: Sydle (2023).

2.5.3 Ferramenta utilizada para aplicar o método ágil *Scrum*

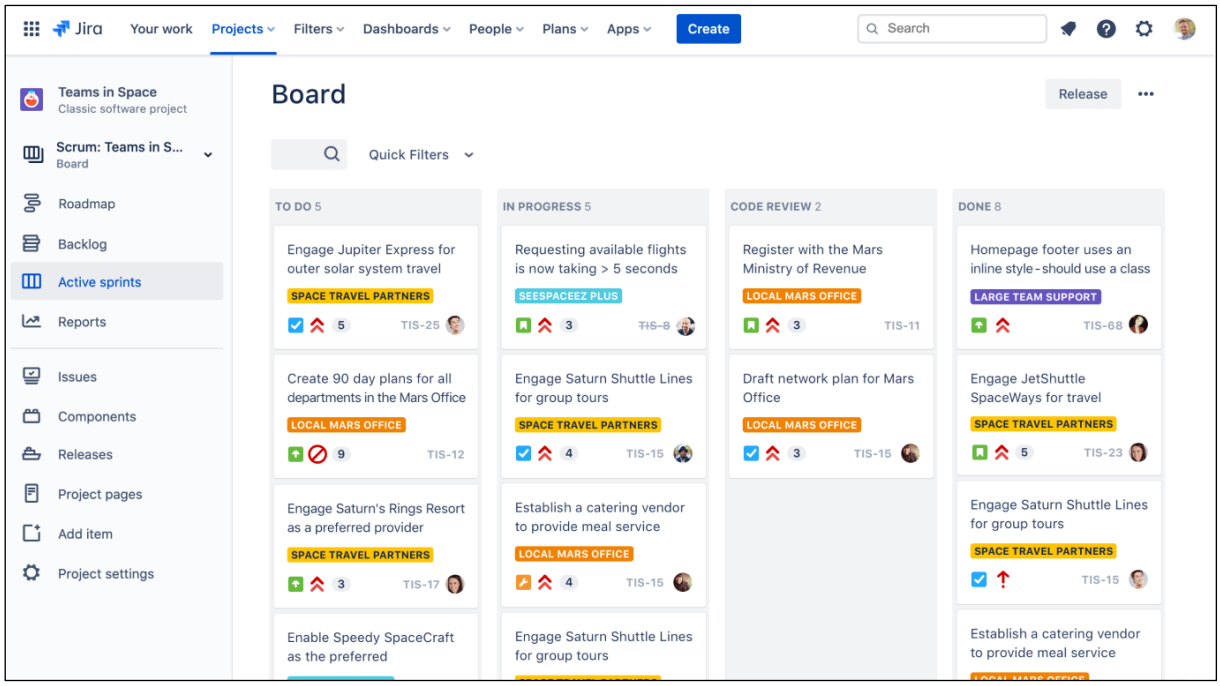
Na MGCode utiliza-se a ferramenta *online* Jira, uma plataforma popular de gerenciamento de projetos desenvolvida pela Atlassian[4]. Essa ferramenta facilita a implementação de metodologias ágeis, como destacado por Garrett (2021). O Jira oferece recursos específicos para ajudar as equipes a gerenciar *sprints* no contexto do *Scrum*:

a) Quadro *Scrum*: O Jira possui um quadro específico para o *Scrum*, como é apresentado na Figura 11 – Distribuição de tarefas e os progressos na plataforma Jira, por onde as tarefas são organizadas em colunas, representando diferentes estágios do fluxo de trabalho, como "Para Fazer", "Em Progresso" e "Concluído";
b) Criação de *sprints*: No Jira, você pode criar *sprints* definindo um intervalo de datas e associando tarefas a esse *sprint*. As tarefas atribuídas ao *sprint* serão exibidas no quadro *Scrum* e os membros da equipe podem mover essas tarefas pelas colunas, conforme o progresso;
c) Planejamento de *sprints*: Antes de iniciar um *sprint*, as equipes geralmente conduzem uma reunião de planejamento do *sprint* para definir quais tarefas serão abordadas durante aquele período;
d) Acompanhamento do Progresso: Durante um *sprint*, os membros da equipe podem atualizar o status das tarefas diretamente no quadro *Scrum*. Isso permite a visibilidade do progresso e ajuda a identificar possíveis problemas ou gargalos;
e) Revisão e Retrospectiva: No final de um *sprint*, as equipes realizam uma revisão para demonstrar o trabalho concluído e coletar um *feedback*. Em seguida, conduzem uma retrospectiva para analisar o *sprint* e identificar melhorias para os próximos *sprints*.

O Jira também oferece relatórios e métricas para acompanhar o desempenho da equipe temporariamente, como velocidade do *sprint*, histórias concluídas e gráficos de *Burndown*.

[4] Empresa australiana de software que desenvolve produtos para desenvolvedores de software e gerentes de projeto.

Figura 11 – Distribuição de tarefas e os progressos na plataforma Jira



Fonte: Atlassian (2023).

3 ATIVIDADES DESENVOLVIDAS

Esta seção é composta pelas atividades realizadas durante o estágio na MGCode. A subseção 3.1 é uma sintetização das atividades para uma noção evolutiva, respeitando a ordem dos acontecimentos. Na subseção 3.2 são detalhadas as atividades de desenvolvimento do Sistema de Coleta Domiciliar da FEMME.

3.1 Síntese das atividades

A síntese das atividades aborda as atividades e treinamentos técnicos realizados durante o período de estágio na empresa. Ofertaram-se duas vagas de estágio para desenvolvedores pela empresa. Os candidatos enviaram um currículo para demonstrar suas habilidades técnicas e suas aspirações de carreira para participar do processo de seleção. Os requisitos da vaga não incluíam habilidades técnicas, no entanto, enfatizou-se a necessidade de pessoas proativas e autodidatas.

O processo de seleção consistiu em um teste prático realizado presencialmente no escritório da empresa. O teste proposto consistiu no desenvolvimento de um CRUD, um sistema para consulta, cadastro, alteração e exclusão de usuários. Como exigência, deveria ter um *front-end* e *back-end* básicos. As linguagens e tecnologias utilizadas no processo consiste em: banco de dados MySQL, PHP para o *back-end* e o *front-end*, JavaScript, HTML5 e CSS3. Uma semana antes do teste prático, disponibilizaram-se cursos para essas tecnologias, os quais tinham material suficiente para os participantes conseguirem realizar o teste.

Fez-se uma análise de desempenho dos participantes constituída das seguintes características como critério de aprovação: a) tempo gasto pelo participante para desenvolver o sistema; b) clareza do código; c) e usabilidade do sistema.

3.1.1 Escolha de atuação

O autor deste relatório recebeu como proposta duas áreas de atuação ao se juntar à equipe de desenvolvedores da MGCode: *front-end* e *back-end*. O estagiário tem a liberdade de escolher sua área inicial para a realização de cursos em tecnologias específicas disponibilizados pela empresa, porém, posteriormente, é

necessário da parte do colaborador aprender as duas frentes de desenvolvimento. Optou-se, inicialmente, por se capacitar em *front-end*, utilizando a tecnologia React. O estagiário recebeu treinamento para compreender a estrutura do banco de dados e as regras de negócios, com o objetivo de adquirir um conhecimento e relacioná-lo ao entendimento das tabelas desse banco de dados. O DBA em questão consistia em um conjunto reduzido de duas tabelas. A tabela mais extensa continha cinco campos e a mais compacta continha quatro campos. Empregava chaves estrangeiras como parte da sua estrutura. A tabela com o maior número de chaves estrangeiras continha cinco, utilizadas principalmente para estabelecer relações entre uma tabela de usuários e outra de carros.

No geral, a complexidade do banco de dados era baixa, adequada para a fase de treinamento do estagiário. Não havia procedimentos (*stored procedures*) nem transações definidas. As tabelas eram projetadas principalmente para armazenar e recuperar informações básicas relacionadas a usuários e carros. Continham campos de controle padrão, como "criado em" e "alterado em", uma parte fundamental das boas práticas de gerenciamento de dados. Não havia campos que se destacassem por sua complexidade ou dificuldade de lidar. Os tipos de dados incluam informações como texto, números inteiros e datas. Fez-se esse treinamento com o auxílio do DBA em conjunto com um de seus colegas de equipe que prestava esse serviço.

3.1.2 Cursos introdutórios realizados

Cursos de programação do portal de ensino B7Web[5] chamados **“**Introdução ao ReactJS**”** e **“**Introdução ao Typescript**”** (disponíveis em https://b7web.com.br) foram realizados nas três primeiras semanas. Esses cursos foram adquiridos pela empresa para o treinamento de novos colaboradores e capacitá-los nas novas tecnologias de mercado. Realizaram-se aproximadamente 45 horas de treinamento, alternando entre a prática, teoria e implementação de pequenos projetos para a fixação do conteúdo com a supervisão dos colaboradores para acompanhar o progresso do estagiário.

[5] A B7Web é uma plataforma online que disponibiliza cursos de desenvolvimento.

3.1.3 Projeto de desenvolvimento

Surgiu a ideia na empresa de desenvolver um novo sistema em ReactJS com o propósito de gerenciar empresas de forma eficiente e escalável. Além de ser um projeto para treinamento e avaliação das habilidades adquiridas pelo curso B7Web, abordaria necessidades reais da empresa. Esse sistema não se limitaria apenas ao controle básico, mas também poderia incorporar módulos adicionais no futuro, destinados a atender às diversas necessidades administrativas dos clientes. Essas funcionalidades abrangem:

a) Controle de Colaboradores: o sistema permite o cadastro e a gestão de informações sobre funcionários, incluindo dados pessoais, informações de contato, cargos, e histórico profissional;
b) Gestão de Clientes: registro de informações detalhadas sobre os clientes da empresa, como histórico de compras, preferências e informações de contato;
c) Controle de Vendas: capacidade de registrar vendas realizadas, gerar faturas, acompanhar o status dos pagamentos e manter um histórico completo das transações comerciais;
d) Contabilidade e Finanças: recursos para gestão financeira, como controle de despesas, receitas, balanços e relatórios financeiros.

O sistema de controle empresarial utilizava uma abordagem mais tradicional até então, com o *front-end* desenvolvido em HTML, CSS e JavaScript puro. No entanto, tornou-se evidente a necessidade de uma modernização. A escolha pelo ReactJS foi feita com base em sua flexibilidade e poder de escalabilidade. Além de atender à necessidade de substituir sistemas legados, a construção de um novo sistema em ReactJS também padroniza a linguagem para o *back-end* e *front-end*, uma vez que ambos usam o Typescript. Isso reduz a curva de aprendizado do desenvolvedor ao mudar de *stack*.

A execução do projeto envolveu um estagiário praticando o *front-end* e outro o *back-end*. A equipe consiste somente por esses desenvolvedores, sem o uso de metodologias ágeis. Vários desafios foram enfrentados durante a execução, considerando a transição de tecnologias legadas para tecnologias mais modernas, como ReactJS e TypeScript. Criaram-se componentes reutilizáveis e eficientes,

melhorando a manutenção e a escalabilidade do sistema. Aprender a gerenciar o estado da aplicação foi fundamental na garantia de um fluxo de dados eficaz entre os componentes.

O estagiário foi o responsável pelos testes de utilização do *software* de maneira instintiva. Não houve controle de versões do sistema, apenas a hospedagem dos arquivos em um repositório no GitHub. Um *commit* com os novos arquivos era realizado a cada atualização. O desenvolvedor sênior fazia uma nova bateria de testes ao terminar uma demanda e, de maneira manual, a atualização do sistema em produção. O curso da B7Web forneceu uma base para o pontapé inicial no projeto, mas a maior parte do aprendizado aconteceu durante a implementação. A empresa incentivou o desenvolvimento contínuo e a pesquisa independente. O desenvolvimento desse *software* iniciou-se logo após o estagiário ser admitido, estendendo-se até o término do estágio, já que o sistema entrou em operação e necessitava de manutenções.

Participar desse projeto interno foi extremamente valioso para o autor, permitindo que adquirisse habilidades essenciais em desenvolvimento *web*, além de lhe preparar efetivamente para participar do projeto Femme, aumentando a sua confiança e competência como desenvolvedor.

3.2 Sistema de Coleta Domiciliar FEMME

O projeto Sistema de Coleta Domiciliar FEMME contou com a metodologia ágil SCRUM e tem em sua primeira etapa um escopo para registrar os pacientes, cadastrar atendimentos e um sistema de usuários para acesso ao *software*.

3.2.1 Principais funcionalidades do sistema

- Registro de Pacientes: permite o cadastro detalhado de pacientes, incluindo informações pessoais, endereço e detalhes de contato. Antes de realizar o cadastro de um atendimento, é necessário ter um paciente para que o atendimento possa existir;
- Cadastro de Atendimentos: possibilita o registro e o acompanhamento de atendimentos domiciliares, incluindo agendamento e detalhes do atendimento;

- Sistema de Usuários: oferece diferentes níveis de acesso para profissionais de saúde e outros desenvolvedores, garantindo a segurança e a privacidade dos dados.

3.2.2 Equipe

A equipe foi composta por um *Product Owner* (PO) responsável por definir os requisitos e prioridades do projeto, um *Scrum Master* para garantir a aplicação eficaz do *Scrum*, e membros do *Dev Team*, incluindo dois desenvolvedores *back-end*, dois *front-end* e um administrador de banco de dados (DBA). O estagiário desenvolveu o papel de desenvolvedor *front-end*.

3.2.3 Metodologia *Scrum*

O projeto adotou a metodologia *Scrum*, utilizando a ferramenta Jira para auxiliar, como parte de seu processo de desenvolvimento. Isso incluiu:

- Ritos do *Scrum*: realizaram-se reuniões de *Planning*, *Review* Retrospectiva e Diárias para garantir o acompanhamento do progresso do projeto e ajustes contínuos. O *sprint* durava 15 dias;
- *Scrum Master*: desempenhou um papel fundamental na orientação da equipe, garantindo os princípios e práticas do *Scrum;*
- *Product Owner*: foi responsável por definir os requisitos do sistema, priorizar o *backlog* de histórias e garantir as necessidades dos usuários;
- *Sprint Planning*: as funcionalidades foram planejadas em *sprints*, e a equipe quantificou o esforço necessário para cada história de usuário;
- Testes: os testes eram feitos manualmente pela PO;
- Melhorias contínuas: durante as reuniões e *sprints*, a equipe identificou áreas de melhoria e implementou ajustes no processo de desenvolvimento, como o planejamento da *sprint*, que contava com a inclusão de itens mal definidos no *backlog*, levando a atrasos.

3.2.4 GitHub

O projeto começou com a criação de dois repositórios no GitHub, um para o *front-end* e outro para o *back-end*, servindo como um local central para armazenar

o código-fonte do projeto. Os membros da equipe tinham acesso a esses repositórios. É importante observar que a empresa não adotou o modelo Gitflow[6] em seu fluxo de trabalho de gerenciamento de código-fonte.

Os repositórios foram organizados no GitHub com duas *branches*, sendo essas:

- *Develop*: usada como um ambiente de desenvolvimento contínuo. Novos recursos eram desenvolvidos e testados nessa *branch*;
- *Release*: usada para preparar o código para uma versão de produção. Quando a equipe acredita que a *develop* está pronta para ser lançada como uma versão estável, uma nova *branch* de *release* é criada a partir da *develop.*

Isso permitiu a colaboração da equipe efetivamente, rastreando mudanças no código-fonte e mantendo um histórico completo das alterações.

3.2.5 Implantação em produção

A implantação em produção do projeto FEMME era um processo cuidadosamente planejado e executado, seguindo alguns passos como:

1) Testes pela PO: a versão de desenvolvimento era submetida a testes funcionais e manuais pela *Product Owner*. A PO avaliou a funcionalidade, garantindo os requisitos definidos e a inexistência de problemas críticos. A ênfase nesta etapa era garantir a qualidade das alterações antes de prosseguir com a implantação em produção;
2) Preparação para implantação: isso inclui a compilação do código-fonte e a atualização da base de dados;
3) Utilização do FTP para transferência: os arquivos eram transferidos por meio do FTP para o servidor de produção;
4) Verificação Final: após a implantação em produção, a equipe realizou uma verificação final para confirmar se os recursos foram transferidos com sucesso e que o sistema estava funcionando conforme o esperado; qualquer problema crítico nessa fase seria abordado imediatamente para garantir a estabilidade do sistema;

[6] Modelo alternativo de ramificação do Git que consiste no uso de ramificações de recursos e várias ramificações primárias.

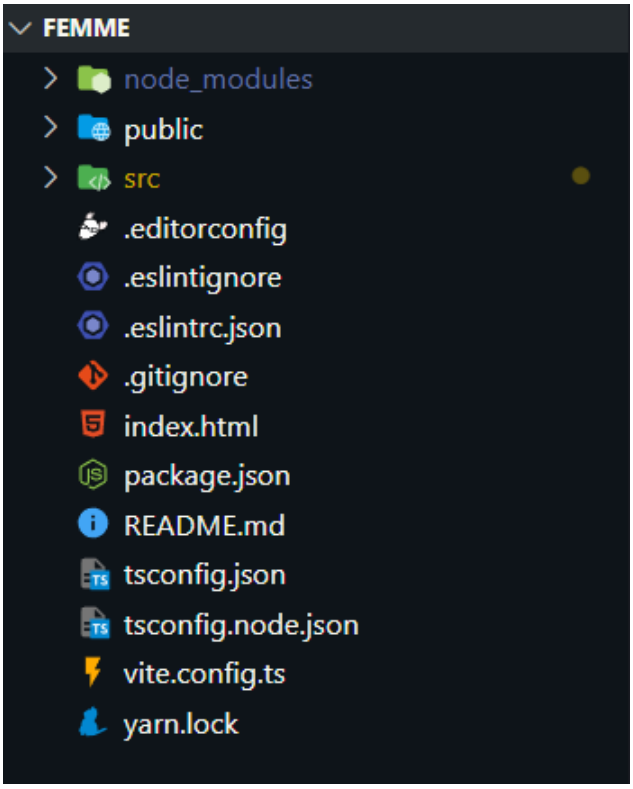
5) Monitoramento e ajustes: após a implantação bem-sucedida, a equipe implementou um processo de monitoramento contínuo para identificar e abordar quaisquer problemas no ambiente de produção. Isso garantia o funcionamento do sistema de forma confiável após a implantação.

3.2.6 Configurações iniciais do projeto

As configurações iniciais envolvem a instalação do ReactJS. O primeiro comando executado foi “*yarn create vite femme --template react-ts”*, esse comando utiliza o Yarn para criar um projeto, utilizando o gerenciador de pacotes Yarn e o *template* "react-ts" (React com TypeScript) fornecido pelo Vite, uma ferramenta de construção de projetos *web* rápida e moderna. Após o início do projeto, instalou-se o ESLint com o comando “*yarn add eslint -D”,* inicializando-o com o comando “*yarn eslint –init”*. O ESLint é uma ferramenta de análise de código estático que ajuda a identificar e corrigir problemas no código. Ele também ajuda a manter um padrão de código consistente e a evitar erros comuns. De acordo com a documentação do ESLint de 2023, o ESLint verifica o código em busca de problemas de formatação, erros de estilo, práticas não recomendadas e até mesmo possíveis erros de lógica.

A estrutura inicial do projeto ficou da seguinte forma, como observa-se na Figura 12 – Estrutura inicial do projeto:

Figura 12 – Estrutura inicial do projeto



Fonte: Elaborada pelo autor (2023).

- A pasta “node_modules” é onde as dependências do projeto são instaladas pelo gerenciador de pacotes Yarn;
- A “public” é onde estão os recursos públicos do aplicativo que não precisam ser processados pelo processo de compilação, como o arquivo HTML principal, imagens, fontes e outros recursos estáticos;
- A “src” contém o código-fonte principal do aplicativo. É onde estão os componentes, estilos, *scripts* e outros arquivos relacionados à implementação do aplicativo;
- O arquivo “.editorconfig” é usado para manter a consistência de formatação do código entre diferentes editores e IDEs usados por membros da equipe. Ele define regras de estilo, como indentação, espaçamento e quebras de linha;
- O “.eslintignore” é usado para definir quais arquivos ou pastas o ESLint deve ignorar ao analisar o código;
- O arquivo “.eslintrc.json” é onde configuram-se as regras, incluindo as necessárias para a análise de código estático do ESLint;

- O “.gitignore” é usado para listar arquivos e pastas para o Git ignorar ao controlar as versões do seu projeto. Isso inclui arquivos temporários, *logs*, pastas de dependências e outros itens que não devem ser versionados;
- O “index.html” é o ponto de entrada do aplicativo. Ele contém a estrutura básica do documento HTML;
- O “package.json” é um arquivo de configuração do projeto com informações, dependências, *scripts* de gerenciamento e outras configurações;
- O “README.md” é um documento de leitura com informações sobre o projeto;
- Os arquivos **“**tsconfig.json” e “tsconfig.node.json” contêm as configurações do compilador TypeScript para o projeto em geral;
- O arquivo “vite.config.ts” é usado para configurar as opções de construção e desenvolvimento da ferramenta Vite;
- O “yarn.lock” é um arquivo gerado pelo Yarn com um registro das versões específicas das dependências instaladas. Isso ajuda a garantir consistência nas versões das dependências entre diferentes ambientes de desenvolvimento.

3.2.7 Registrar os pacientes

O principal objetivo é simplificar e otimizar o processo de registro de pacientes no laboratório. O cadastro do paciente é feito com base no CPF, conforme ilustra a Figura 13 – Caixa de texto de pesquisa do CPF.

Figura 13 – Caixa de texto de pesquisa do CPF

Fonte: Sistema de Coleta Domiciliar FEMME (2023).

Para o cadastro do usuário irá se expandir novos campos de entrada de dados caso o CPF procurado pelo atendente não existir no banco de dados, conforme apresentados e ilustrados nas Figura 14 – Campo de cadastro de informações

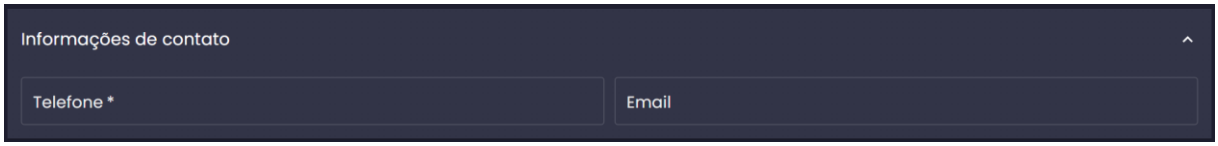
pessoais do paciente, Figura 15 – Campo de cadastro de informações de contato do paciente e Figura 16 – Campo de cadastro do(s) endereço(s) do paciente:

a) Informações pessoais: nome, CPF, RG, data de nascimento, peso, altura e gênero;
b) Informações de contato: telefone e *e-mail*;
c) Endereço(s): CEP, rua, número, bairro, complemento, tipo, cidade, estado, referência e a marcação exata no mapa da localidade.

Figura 14 – Campo de cadastro de informações pessoais do paciente

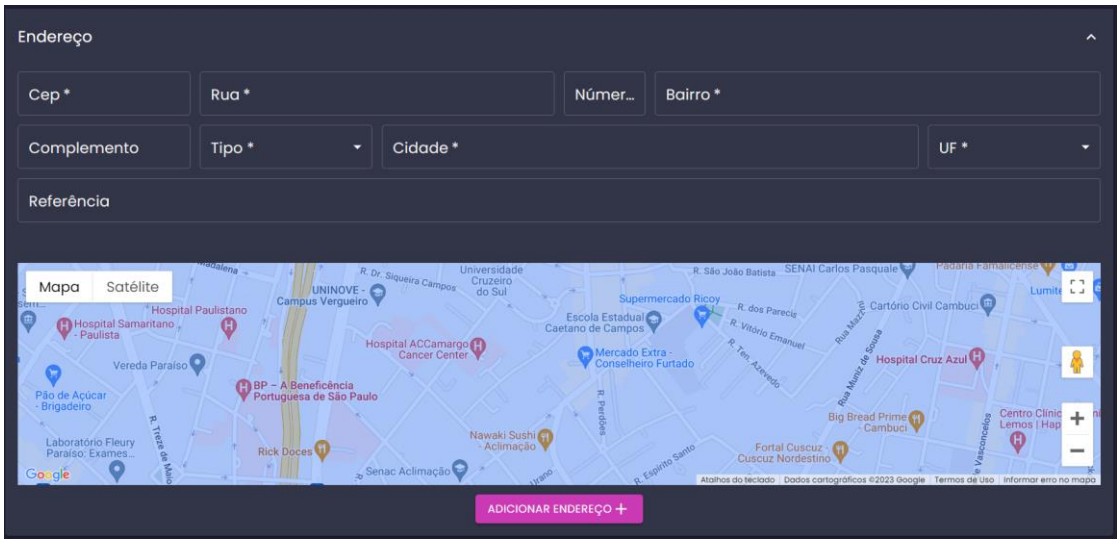
Fonte: Sistema de Coleta Domiciliar FEMME (2023).

Figura 15 – Campo de cadastro de informações de contato do paciente



Fonte: Sistema de Coleta Domiciliar FEMME (2023).

Figura 16 – Campo de cadastro do(s) endereço(s) do paciente



Fonte: Sistema de Coleta Domiciliar FEMME (2023).

O *front-end* desenvolvido pelo estagiário prepara uma requisição do tipo POST para o *back-end,* após o preenchimento dos campos, com os seguintes dados apresentados na Figura 17 – Dados enviados para o *back-end* para cadastro do paciente:

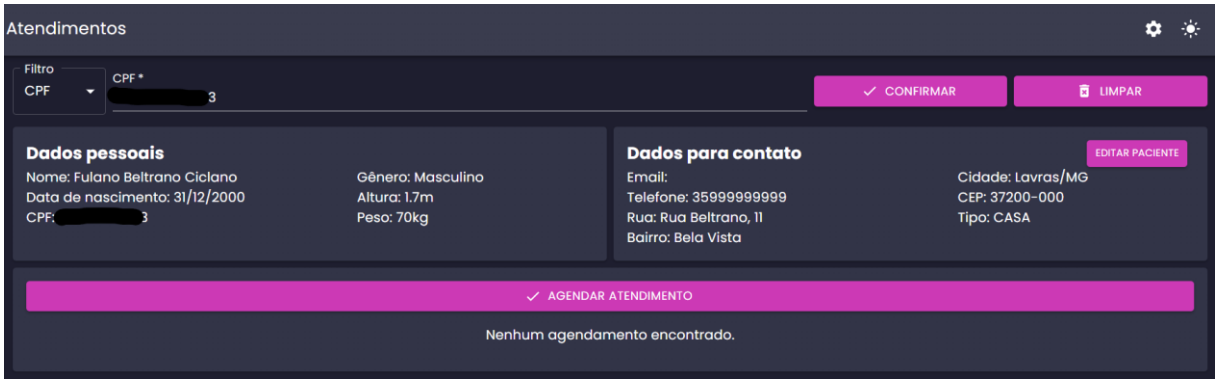
Figura 17 – Dados enviados para o *back-end* para cadastro do paciente

```
const data: TypePacientePost = {
  nome: capitalize(nome),
  cpf: cpf.replace(/[^\d]+/g, ""),
  rg,
  dataNascimento: dataNasc,
  sexo,
  peso,
  altura,
  contato: {
    telefones: [
      {
        numero: telefone,
      },
    ],
    emails: [
      {
        email,
      },
    ],
  },
  enderecos: arrayAddress,
};
```

Fonte: Elaborada pelo autor (2023).

Caso o usuário esqueça de preencher algum dos dados obrigatórios antes de realizar a requisição um *pop-up* de alerta é apresentado. Há também a validação do CPF. É feita a requisição POST ao *back-end* após as entradas de dados, cadastrando o paciente e apresentando uma mensagem que o paciente foi cadastrado com sucesso, levando-o diretamente para a tela de cadastro dos atendimentos (Figura 18 – Tela de cadastro de atendimentos do paciente), tornando o processo mais rápido.

Figura 18 – Tela de cadastro de atendimentos do paciente



Fonte: Sistema de Coleta Domiciliar FEMME (2023).

3.2.8 Registrar os atendimentos

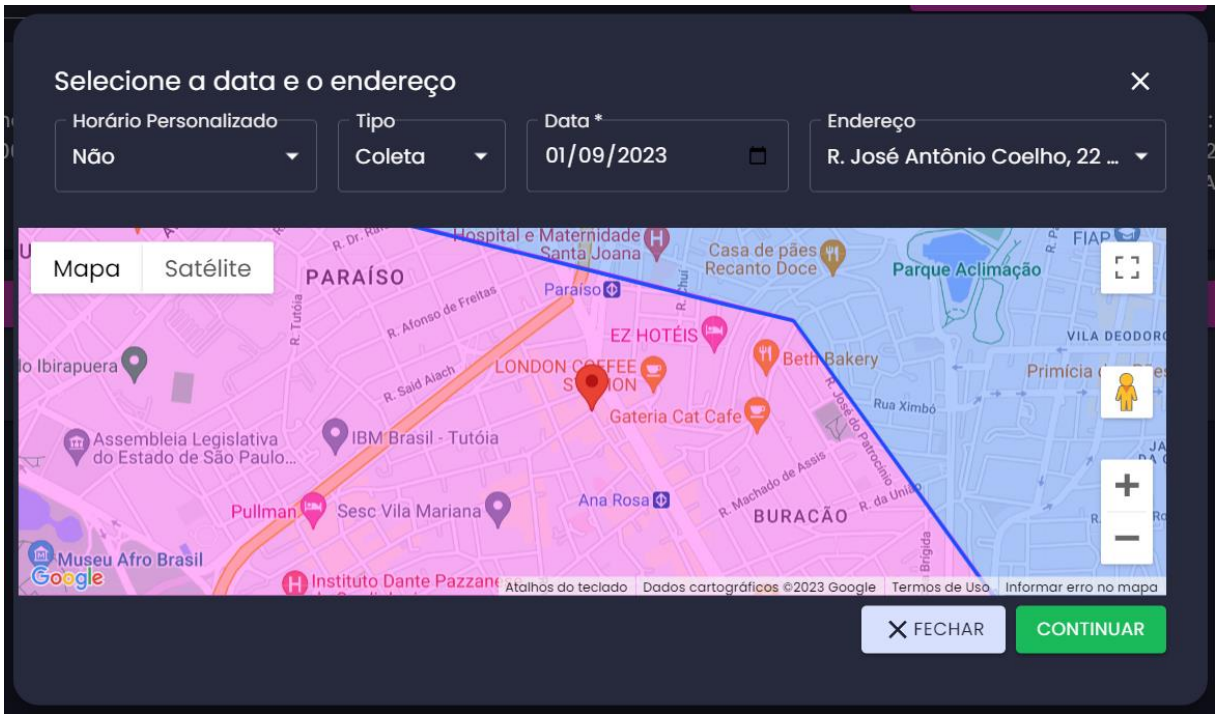
A tela de registro dos atendimentos foi uma das principais solicitadas pelo laboratório, razão pela qual surgiu a necessidade de uma atenção maior e, também, maior rapidez no desenvolvimento para que já suprisse suas demandas. O cadastro dos atendimentos é feito a partir de algumas regras passadas pelo gestor do laboratório. Sendo elas:

a) O endereço do paciente deve estar nas regiões de atendimento pré-estabelecidas pelo laboratório;
b) O agendamento deve estar cadastrado em um horário diferente dos outros agendamentos, para evitar conflitos de rotas.

O atendente irá preencher os dados dos atendimentos em um modal onde são preenchidas informações por etapas. A primeira etapa consiste no preenchimento dos seguintes campos:

- Horário Personalizado: permite ao atendente selecionar um horário já preenchido no sistema. Isso foi pedido posteriormente pelo laboratório;
- Tipo: esse campo diz respeito ao tipo do atendimento a ser realizado, se é coleta domiciliar ou se é vacina;
- Data: como próprio nome já diz, é o campo onde será selecionada a data do atendimento;
- Endereço: endereço do paciente onde será realizada a coleta ou vacina. O endereço selecionado é exibido no mapa, conforme observa-se na Figura 19 – Primeira etapa de criação do atendimento.

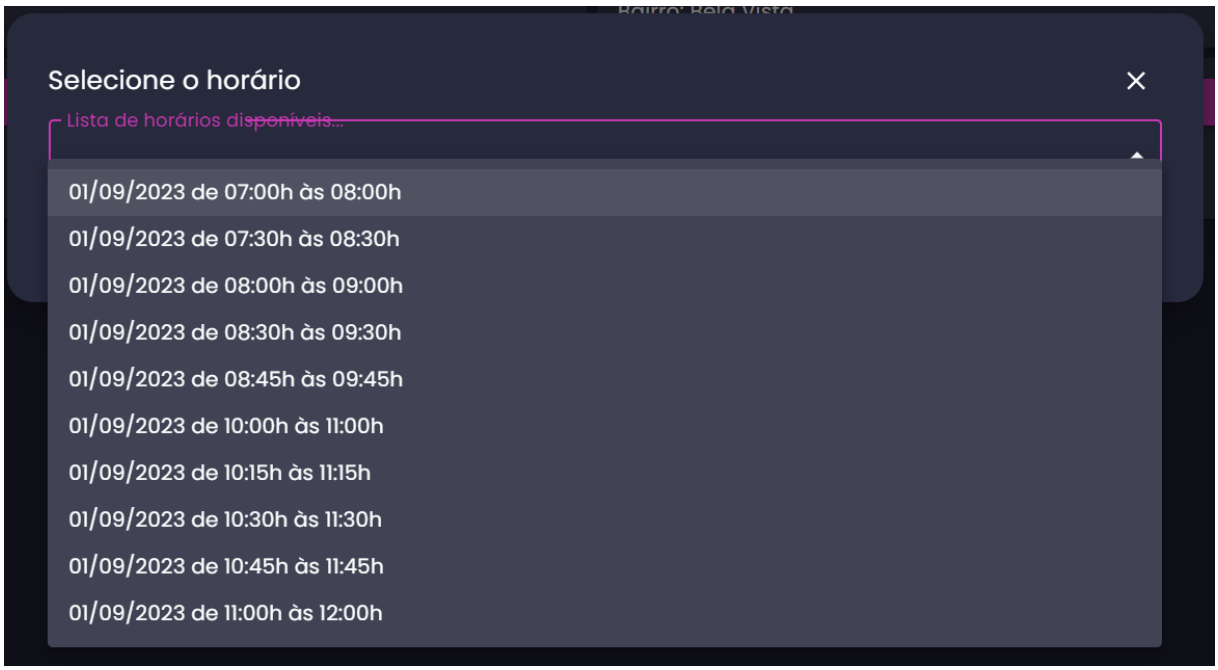
Figura 19 – Primeira etapa de criação do atendimento



Fonte: Sistema de Coleta Domiciliar FEMME (2023).

A segunda etapa diz respeito apenas à lista de horários disponíveis (Figura 20 – Segunda etapa da criação do atendimento), podendo o atendente escolher um horário que melhor atende ao paciente.

Figura 20 – Segunda etapa da criação do atendimento

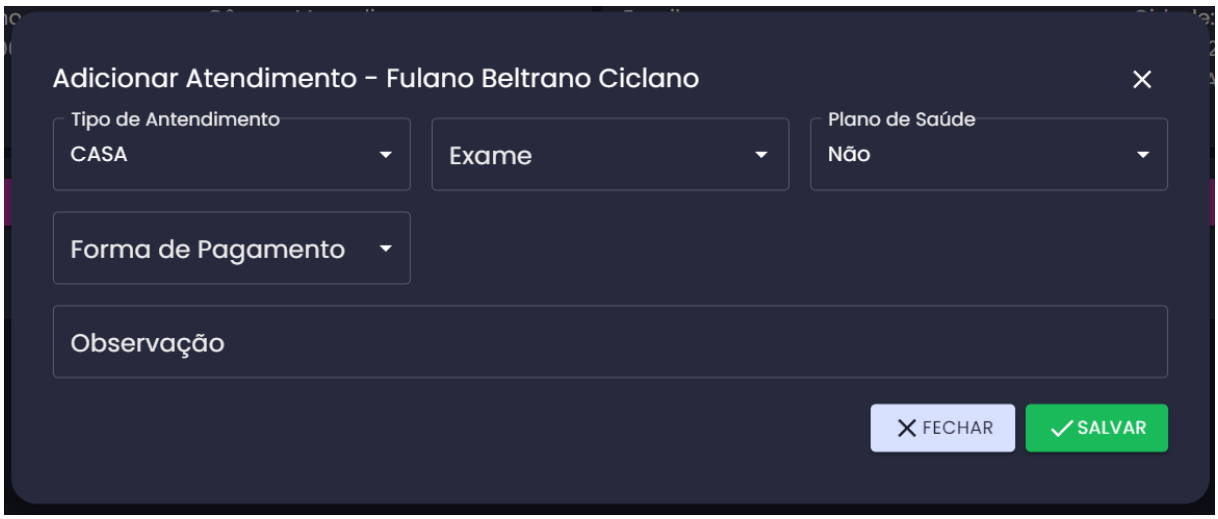


Fonte: Sistema de Coleta Domiciliar FEMME (2023).

A terceira (e final) etapa apresenta alguns campos a serem preenchidos (Figura 21 – Terceira etapa da criação do atendimento (sem plano de saúde) e Figura 22 – Terceira etapa da criação do atendimento (com plano de saúde)). São eles:

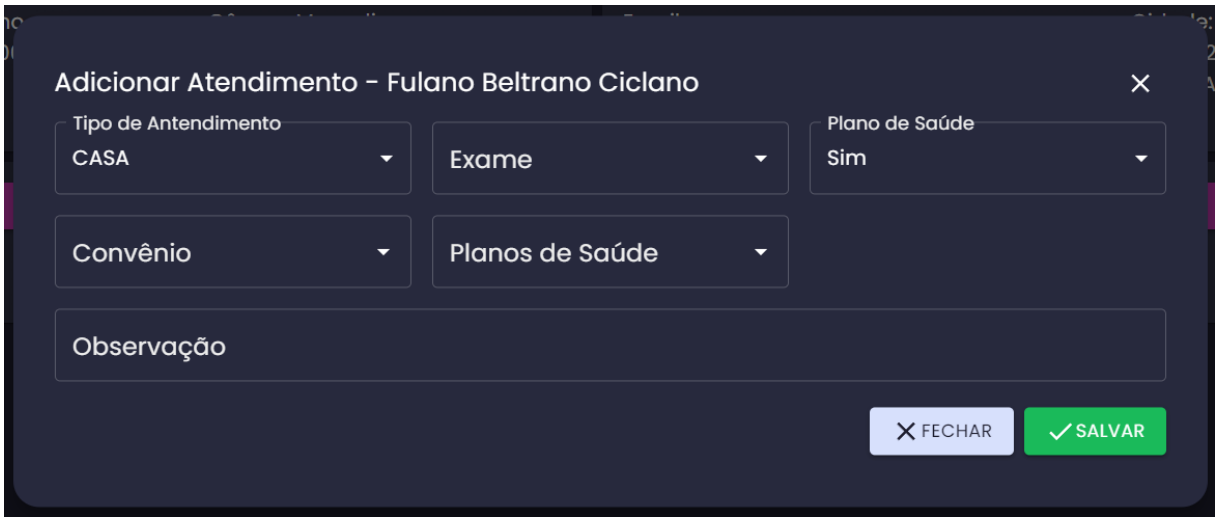
- Tipo de Atendimento: esse campo apresenta o local onde o atendimento será feito. Seja ele apartamento, casa, residência de idosos ou comercial;
- Exame: apresenta uma lista de exames informados pelo laboratório por onde será escolhido o exame a ser realizado pelo paciente;
- Plano de Saúde: apenas para informar se o paciente possui plano de saúde ou não;
- Forma de Pagamento: caso o paciente não possua plano de saúde, abre-se um novo campo para indicar qual a forma de pagamento a ser utilizada, seja ela Pix, cartão ou dinheiro;
- Convênio: caso o paciente possua plano de saúde, abre-se um campo com diversos convênios disponíveis, podendo o atendente selecionar o convênio do paciente;
- Planos de Saúde: a partir do convênio selecionado, busca-se planos de saúde desse convênio, onde é selecionado o plano de saúde do paciente;
- Observação: campo para o caso do atendente desejar adicionar uma observação para aquele atendimento.

Figura 21 – Terceira etapa da criação do atendimento (sem plano de saúde)



Fonte: Sistema de Coleta Domiciliar FEMME (2023).

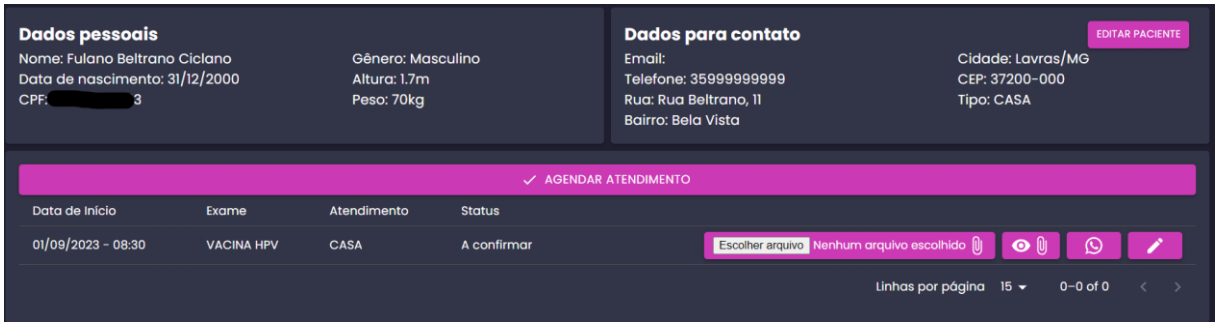
Figura 22 – Terceira etapa da criação do atendimento (com plano de saúde)



Fonte: Sistema de Coleta Domiciliar FEMME (2023).

As requisições para o *back-end* são feitas etapa por etapa, respeitando o modelo de negócio proposto pelo laboratório. Após clicar no botão “Salvar”, aparece um *pop-up* com a mensagem de que o atendimento foi cadastrado com sucesso, e, logo após, o mesmo é mostrado na tela do paciente onde o atendimento foi feito (Figura 23 – Tela de atendimento do paciente quando já cadastrado).

Figura 23 – Tela de atendimento do paciente quando já cadastrado



Fonte: Sistema de Coleta Domiciliar FEMME (2023).

3.2.9 Cadastro de usuários

Os sistemas de autenticação e gerenciamento de usuários desempenham um papel fundamental em garantir a segurança e a acessibilidade da plataforma. A tela de cadastro de usuários permite aos colaboradores do laboratório criar contas para seus atendentes. Inicialmente, a tela apresenta a lista de usuários cadastrados (Figura

24 – Tela inicial da página de usuários), com opções de filtrar os mesmos por nome ou *e-mail*. A cada usuário listado há a opção de editar os dados e/ou excluir o usuário do sistema, impossibilitando seu acesso ao sistema.

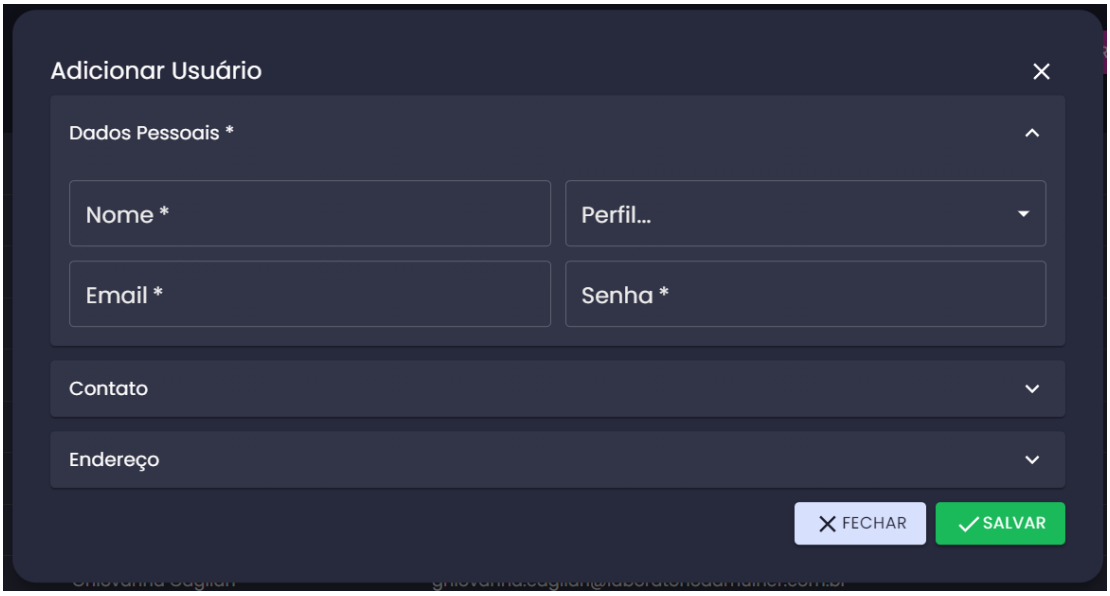
Figura 24 – Tela inicial da página de usuários

Fonte: Sistema de Coleta Domiciliar FEMME (2023).

Ao adicionar um usuário faz-se necessário clicar no botão “+ usuário”, em seguida, abre-se um modal com campos para serem preenchidos (Figura 25 – Acordeão de dados pessoais para o cadastro do usuário). Esses campos estão separados em acordeões, onde cada acordeão é referente à um assunto.

O primeiro acordeão é o de dados pessoais, onde estão os campos de: nome, perfil (onde será atribuído algum perfil, sendo que cada perfil possui diferentes permissões ante o sistema: administração, liderança, motorista e operação), *e-mail* e senha.

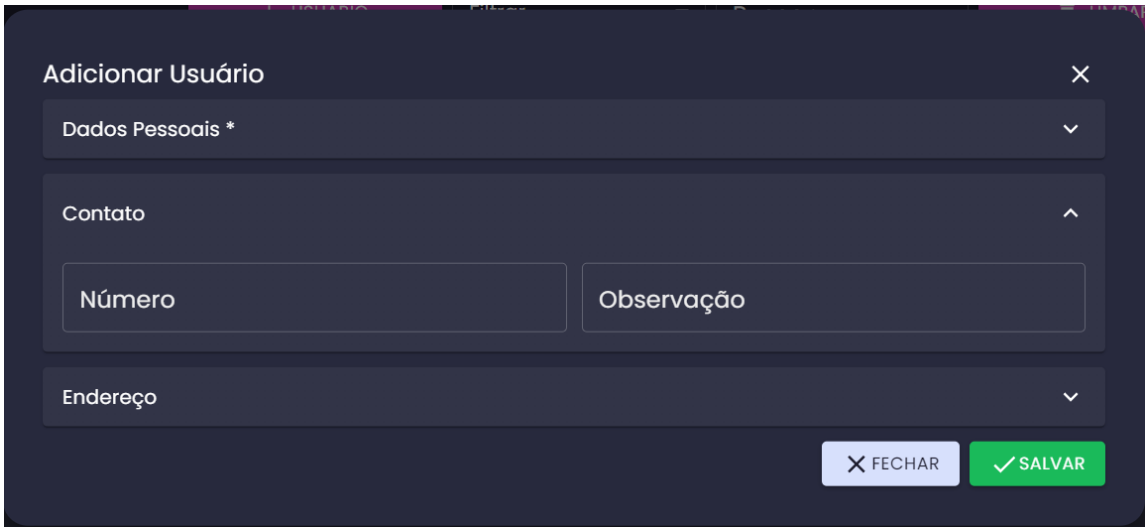
Figura 25 – Acordeão de dados pessoais para o cadastro do usuário



Fonte: Sistema de Coleta Domiciliar FEMME (2023).

O segundo acordeão é o de contato, onde estão os campos de número e observação, como é apresentado na Figura 26 – Acordeão de contato para o cadastro do usuário.

Figura 26 – Acordeão de contato para o cadastro do usuário



Fonte: Sistema de Coleta Domiciliar FEMME (2023).

O terceiro acordeão (Figura 27 – Acordeão de endereço para o cadastro do usuário) é o de endereço, onde estão os campos de CEP, cidade, bairro, rua, número, complemento e referência.

Figura 27 – Acordeão de endereço para o cadastro do usuário

Fonte: Sistema de Coleta Domiciliar FEMME (2023).

O *front-end* desenvolvido pelo estagiário prepara uma requisição do tipo POST para o *back-end* para o cadastro do usuário após o preenchimento dos campos. Depois da requisição com sucesso, retorna ao usuário uma mensagem de que o cadastro foi realizado.

3.2.10 Aprendizados

A participação no projeto FEMME foi uma experiência significativa, considerando as oportunidades de aprendizado e desenvolvimento profissional. Várias melhorias foram identificadas e implementadas, contribuindo para aprimorar as práticas de desenvolvimento e implantação. A experiência no projeto permitiu uma compreensão mais profunda dos processos de implantação em produção, resultando em uma abordagem mais eficaz e organizada para garantir a estabilidade e a qualidade das implantações, além da interação constante com a *Product Owner* (PO) e outros membros da equipe, fortalecendo as habilidades de comunicação e a compreensão das necessidades do cliente.

É importante destacar que a equipe contava com um time incompleto, incluindo a ausência de *Quality Assurance* (QA) e *DevOps*[7]. Isso gerou problemas relacionados a atrasos na implantação, aumento da carga de trabalho para desenvolvedores, falta de automação e desafios de segurança.

[7] É uma cultura na engenharia de software que aproxima os desenvolvedores de software e os administradores do sistema

4 CONCLUSÃO

Este relatório de estágio descreve as atividades desenvolvidas durante o período de 18 de maio a 18 de dezembro de 2022 na empresa MGCode pelo aluno Maurício Júnior Santos Freire. Durante esse período, o estagiário trabalhou com as tecnologias React, TypeScript, SCSS e GitHub, atuando nos sistemas de gerenciamento empresarial e coleta domiciliar. O sistema de gerenciamento empresarial propiciou o controle de colaboradores, vendas, clientes e finanças, enquanto o sistema de coleta domiciliar proporcionou o controle de pacientes, atendimentos e usuários.

As atividades propostas pela empresa foram desenvolvidas com sucesso pelo estagiário. O ambiente de trabalho era amigável e colaborativo, contribuindo para a resolução eficaz dos desafios enfrentados. É evidente que a experiência no mundo real e a aplicação prática dos conhecimentos adquiridos durante o curso são fatores cruciais para o crescimento profissional e a contribuição efetiva para a sociedade. A oportunidade de participar ativamente no desenvolvimento de um *software* proporcionou uma visão realista das demandas, desafios e complexidades do mercado de trabalho.

Durante o processo de desenvolvimento do programa, pôde-se aplicar conceitos teóricos e práticos aprendidos ao longo do curso universitário, como os ensinados nas disciplinas de Engenharia de *Software*, Qualidade de *Software*, Gerência de Projetos de *Software*, Segurança, Auditoria e Avaliação de Sistemas de Informação, entre outras presentes na matriz do curso de Sistemas de Informação. Desde a análise de requisitos até a implementação de funcionalidades específicas e a manutenção da segurança dos dados, os fundamentos adquiridos em aulas foram cruciais para a tomada de decisões informadas e criar uma solução de alta qualidade.

O desenvolvimento deste *software* resultou em benefícios tangíveis para o laboratório. A otimização dos processos de registro de pacientes e gestão de atendimentos domiciliares contribuiu para uma melhoria na qualidade dos serviços prestados. A automação de tarefas rotineiras e a centralização de informações também tiveram impactos positivos na eficiência operacional.

Este projeto não apenas trouxe resultados concretos para a organização, mas também proporcionou ao autor deste trabalho valiosas lições e preparação para desafios futuros. Através das dificuldades enfrentadas durante o desenvolvimento,

aprendeu-se a importância da resiliência, da colaboração em equipe e da busca constante por soluções inovadoras.

O desenvolvimento deste *software* como estagiário para um laboratório médico especializado na saúde da mulher foi uma jornada enriquecedora, reforçando a relevância da educação superior e da experiência prática em sinergia. Este trabalho não apenas representou a aplicação dos conhecimentos adquiridos, mas também sinalizou o compromisso contínuo com a aprendizagem ao longo da carreira profissional. Com a base sólida obtida durante o curso, está-se preparado para enfrentar desafios e continuar contribuindo de maneira significativa para o campo da tecnologia e da saúde.

REFERÊNCIAS

ATLASSIAN. **Jira Software**: Quadros Scrum e Kanban. 2023. Disponível em: https://www.atlassian.com/br/software/jira/guides/boards/overview#board-vs-project. Acesso em: 23 ago. 2023.

AWS. **O que é o Scrum?** 2023. Disponível em: https://aws.amazon.com/pt/what-is/scrum/. Acesso em: 28 ago. 2023.

B7*WEB*. **Quer aprender a programar do zero?** 2019. Disponível em: https://b7*web*.com.br/. Acesso em: 25 ago. 2023.

ESLINT. **Documentation**. [ca. 2023]. Disponível em: https://eslint.org/docs/latest/. Acesso em: 30 ago. 2023.

FRAMEWORKS. **Springer**, Singapore, v. 93, p. 709-717, jan. 2022. Disponível em: https://doi.org/10.1007/978-981-16-6605-6_53. Acesso em 19 de setembro de 2023.

GARRETT, F. **Ferramentas de gestão de projetos**: veja seis opções grátis. 22 set. 2021. Techtudo. Disponível em: https://www.techtudo.com.br/noticias/2021/09/ferramentas-de-gestao-de-projetos-veja-seis-opcoes-gratis.ghtml. Acesso em: 19 set. 2023.

GITHUB. **GitHub Docs**. 2023. Disponível em: https://docs.github.com/pt. Acesso em: 25 ago. 2023.

JAISWAL, P.; HELIWAL, S. **Competitive Analysis of *Web* Development**, 2022.

REACT. **Learn React**. 2023. Disponível em: https://react.dev/learn. Acesso em: 25 ago. 2023.

SASS. **Sass Basics**. 2023. Disponível em: https://sass-lang.com/guide/. Acesso em: 25 ago. 2023.

SUEHRING, S.; VALADE, J. **PHP, MySQL, JavaScript & HTML5 All-in-One for Dummies**. Hoboken, New Jersey: John Wiley & Sons, Inc, 2013.

SYDLE. **Framework Scrum**: o que é e como funciona? 2023. Gestão por Processos. Disponível em: https://www.sydle.com/br/blog/framework-scrum-5f6dc45f320703787497f887. Acesso em: 19 set. 2023.

SCHWABER, Ken. SUTHERLAND, Jeff. **Guia do Scrum: o guia definitivo para o Scrum: as regras do Jogo**. 2020. Disponível em: https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-Portuguese-European.pdf. Acesso em: 25 ago. 2023.

TYPESCRIPT. **TypeScript Documentation**. 2023. Disponível em: https://www.typescriptlang.org/docs/. Acesso em: 25 ago. 2023.

W3C Escritório Brasil. **CSS**: Curso W3C Escritório Brasil. 2016. Disponível em: https://www.w3c.br/pub/Cursos/CursoCSS3/css-*web*.pdf. Acesso em: 19 set. 2023.

YARN. **Getting Started**. 2023. Disponível em: https://classic.yarnpkg.com/lang/en/docs/getting-started/. Acesso em: 30 ago. 2023.