LembraMed - Desenvolvimento de um Sistema para Lembretes e Gerenciamento de Medicações

Guilherme S. Kuhn, Ana P. Canal
Curso de Ciência da Computação
UFN - Universidade Franciscana
Santa Maria - RS
guilherme.kuhn@ufn.edu.br, apc@ufn.edu.br

***Resumo*—Este trabalho apresenta o desenvolvimento de um sistema web com o uso da linguagem Typescript, junto com o framework React para o frontend, com Node.JS e framework Express no backend. A API da Twilio foi utilizada para envio de notificações SMS, e o MongoDB para o armazenamento de dados. O desenvolvimento fez o uso das boas práticas da metodologia ágil chamada *Feature-Driven Development*. Seguindo suas etapas, foi entregue um sistema que lembra o usuário a consumir seus medicamentos corretamente, com ajuda da visualização dos medicamentos que precisam ser utilizados no dia que se encontra e o envio de mensagens SMS no horário correto de uso. O sistema também deixa possível o gerenciamento do seu catálogo de fármacos, funcionando tanto para cuidar a si, quanto para melhorar a qualidade de vida de pessoas ou animais que o usuário cuide.**

***Palavras-chave* : React; Medicações; Saúde Digital; Desenvolvimento Web**

I. Introdução

Os medicamentos são muito importantes para o nosso bem estar, previnem doenças, epidemias e salvam vidas [1], mas há um problema, segundo o CIATOX (Centro de Informação e Assistência Toxicológica), cerca de 33,62% dos casos de intoxicação [2], na maioria em idosos e crianças, são associados com o uso errado de medicamentos, não tomando medicamentos na hora certa, utilizando doses erradas [3], ou até mesmo utilizando remédios vencidos. O controle de medicamentos se faz necessário para ajudar a população, tanto para as que precisam ser educadas sobre o uso correto, quanto as pessoas que precisam de ajuda para se lembrar de medicações.

Este trabalho, tem como função auxiliar na solução de problemas como esse a partir de um sistema web, ajudando pacientes a não cometerem erros, educando e alertando o paciente. O projeto será detalhado ao decorrer deste artigo.

A. Justificativa

Um dos problemas existentes em relação à remédios prescritos, é que por muitas vezes, as pessoas esquecem do horário correto de utilizar, ou até mesmo, param de utilizar completamente, comprometendo a eficácia do medicamento. Isso pode ser um problema maior ainda, quando remédios de alta importância são utilizados incorretamente, o que pode comprometer a vida do paciente [3].

A justificativa deste trabalho, é criar um sistema que possibilite aos usuários catalogar medicações (para eles mesmos ou para outras pessoas que estão cuidando), visualizar os medicamentos do dia, registrar a utilização e a lembrá-los por meio de notificações SMS, de usar corretamente seus remédios no momento exato, e impedir o consumo de fármacos vencidos. Profissionais da saúde consequentemente serão auxiliados com o sistema, pois facilitará o acompanhamento de seus pacientes quanto aos seus medicamentos, ajudando na relação entre o profissional e o paciente. Com a possibilidade de catalogar medicações não somente para si, é possível melhorar a qualidade de vida não apenas nossa, mas também de pessoas que cuidamos, reduzindo erros de utilização e aumentando a adesão de tratamentos.

B. Objetivo Geral

Este trabalho tem como objetivo geral desenvolver uma aplicação web, que possibilite aos pacientes catalogar e lembrar-se de utilizar seus remédios no momento exato, evitando o consumo incorreto de medicamentos e lembrando-os de descartá-los quando vencidos.

C. Objetivos Específicos

Os objetivos específicos deste trabalho, são:

- Descrever o problema de controle de administração de fármacos.
- Desenvolver um sistema web que faça o gerenciamento de medicamentos, a partir das tecnologias React e Typescript.
- Alertar usuários via notificações, quanto aos seus medicamentos prescritos.

II. Referencial Teórico

Nessa seção, serão apresentados conceitos importantes que tornaram o desenvolvimento deste trabalho possível, como o conceito de *Digital Health* e o controle de medicações. As tecnologias que foram utilizadas também serão apresentadas, como o *framework* React junto com a linguagem Typescript, a parte *backend* da aplicação com Node.JS

junto do *framework* Express, o banco de dados MongoDB e por fim, a API da Twilio para envio de SMS.

A. Digital Health

Digital Health (saúde digital) é um termo geral para o uso de tecnologia voltada para a área da saúde. Existem várias aplicações possíveis, como sistemas para cuidar de doenças crônicas, softwares para computadores que servem para monitorar epidemias e até aplicações móveis para gerenciar medicamentos. Como o mundo está cada vez mais conectado e o uso de dispositivos móveis e computadores só aumentam, a chamada *Digital Health* se torna muito conveniente tanto para pacientes, quanto para profissionais, melhorando a qualidade de vida de pacientes e até mesmo salvando vidas [4].

A saúde digital pode ajudar os usuários a se informarem melhor sobre a necessidade de se cuidar, assim como ajudar no gerenciamento de condições fora de ambientes hospitalares e facilitar a prevenção de pioras de quadro, e até mesmo diagnosticar precocemente doenças graves, por meio de análise de dados e inteligência artificial [5].

Com o tempo ela vem se atualizando lado a lado com as novas tecnologias sendo descobertas e aprimoradas, sendo possível, por exemplo, o acompanhamento e a coleta de dados de pacientes, como movimentos, padrões de sono e batimentos cardíacos, tudo remotamente. Com dados relevantes de pacientes, profissionais podem fazer acompanhamento e intervenções [6]. A aplicação da saúde digital para controle de medicações pode auxiliar na redução de usos incorretos, problemas estes citados na seção a seguir.

B. Controle de Medicamentos

É evidente que os remédios são essenciais para cuidar de nossa saúde e vida, mas para isso, é necessário ter um controle sobre seus medicamentos. De acordo com Salama e Abd-Elfattah [7], pacientes precisam utilizar o remédio correto e no tempo correto, para não se expor a situações de risco de saúde.

Ainda segundo Salama e Abd-Elfattah [7], a parte da população que corre mais riscos é a de pessoas idosas, pois ao envelhecer, tendemos a utilizar vários medicamentos recorrentes. Somando o número de medicações, com os possíveis problemas de habilidades decorrentes da idade, o risco de cometer um erro ao consumir o medicamento, é alto, pois lembrar de utilizar o medicamento certo, no dia certo e na hora correta, pode ser difícil [7].

Outros dois problemas identificados por Salama e Abd-Elfattah [7], é que há problemas de administração de medicamentos em pacientes que estão em hospitais e casas de repouso, pela quantidade de medicações gerenciadas, e também o problema da inexperiência quanto a fármacos controlados por parte do paciente ou falta de comunicação entre o paciente e profissional.

Diante dessa necessidade, foi pensado em um sistema web para o controle e gerenciamento de medicações, com um sistema de notificações, auxiliando pacientes a usar suas medicações corretamente, assim como administrar fármacos adequadamente para outras pessoas. Para o desenvolvimento desse sistema, foram utilizados as tecnologias que serão descritas a seguir.

C. Linguagem JavaScript e o Framework React

JavaScript é uma linguagem leve e dinâmica que teve seu início em 1995, com o intuito de melhorar interfaces de páginas web, fornecendo uma linguagem de *script* para web, possibilitando interações que não eram possíveis utilizando apenas HTML. Com o passar dos anos, a linguagem também começou a ser utilizada fora de ambientes web e também foram surgindo *frameworks* para esta linguagem, como o React [8].

O React é um dos principais *frameworks* existentes para a linguagem JavaScript. Foi criado pelo Facebook em 2011, possuindo foco principal no desenvolvimento de interfaces de usuário para aplicações web e é atualmente utilizado, como por exemplo, em empresas grandes como o Airbnb, Instagram e o próprio Facebook [9].

A biblioteca deixa possível a criação de páginas atualizadas em tempo real (ou reativas, como o próprio nome sugere) e seus componentes podem ser compartilhados/reutilizados, deixando o sistema escalável de uma forma simplificada. Por ser código aberto, são feitas variadas contribuições de desenvolvedores, para aprimorar ainda mais a biblioteca [9].

D. TypeScript

Criada pela Microsoft, o TypeScript tem como utilidade trazer mais recursos sobre a linguagem JavaScript. Seu principal recurso é a tipagem estática, tornando o código mais documentável, escalável e legível, por descrever melhor os objetos e funções. Ao contrário do JavaScript, o TypeScript pode ajudar o desenvolvedor a identificar erros antes mesmo da execução de um projeto, por meio de checagens de tipos de valores [10].

E. Back-end Node.JS e Framework Express

Em relação à parte *backend*/servidor da aplicação, responsável pela entrega de informações solicitadas no *frontend*, foi selecionado o Node.JS, criado em 2009 e atualizado até atualmente, é um ambiente de execução de JavaScript orientado a objetos. A tecnologia é assíncrona/não bloqueante, significando que o projeto pode realizar mais de uma operação ao mesmo tempo, por exemplo, uma requisição de um recurso online, enquanto faz uma outra operação, o que torna o Node.JS perfeito para projetos de quaisquer escala [11].

Express é um dos *frameworks* mais populares para o Node.JS, que descomplica e agiliza a criação de aplicativos e APIs. O *framework* possui diversas bibliotecas e ferramentas

para o desenvolvedor, como funcionalidades relacionadas a requisições HTTP (como GET, POST, DELETE e SET), autenticação, cookies e segurança do sistema [12]. Com estas tecnologias, também se faz possível a entrega de dados ao usuário a partir de um banco de dados (neste caso o MongoDB) e a utilização da API de SMS Twilio necessária para o trabalho, explicadas nas seções a seguir.

F. Banco de Dados MongoDB

Para o armazenamento e acesso das informações necessárias do sistema, como armazenamento de usuários e de medicações cadastradas, foi escolhido um banco de dados não-relacional, de código aberto, chamado MongoDB. É baseado em NoSQL, robusto, multiusuário e rápido [13], sendo possível armazenar os dados necessários em arquivos JSON flexíveis, assim como definir, controlar e requisitar essas informações facilmente. Além disso, o MongoDB disponibiliza uma nuvem gratuita para poder hospedar o banco de dados, o que facilita o desenvolvimento de projetos pequenos.

G. API de SMS Twilio

Para ajudar no aviso de medicações via mensagens, foi escolhido a API da Twilio [14], que possui um período de testes gratuito, podendo enviar mensagens SMS sem limites para um único número escolhido, ou para outros números não cadastrados, mas descontando do limite de gasto total que o teste permite. Este serviço possui várias funções bem documentadas para o uso no servidor Node.JS, facilitando a implementação e sua compreensão. Fora o uso para notificações SMS, que é o foco desta tecnologia neste trabalho, é também possível enviar mensagens via WhatsApp, encaminhar códigos de verificação de duas etapas, mandar promoções e também ser programada para responder certas mensagens recebidas [14].

III. Trabalhos Correlatos

Neste tópico, serão apresentados três trabalhos que contribuem com a compreensão e desenvolvimento do sistema, por possuírem temas semelhantes.

A. Trabalho Correlato I - PRATELEIRA: desenvolvimento de uma solução mobile para controle de medicamentos e rotinas medicinais

O trabalho de Filho, Oliveira e Lucas [15], apresenta um aplicativo para dispositivos móveis, chamado Prateleira, para pacientes e profissionais, onde é possível gerenciar um estoque compartilhado de medicamentos entre familiares e programar rotina de medicação dos usuários. O trabalho foi feito a partir da metodologia SCRUM, junto com as tecnologias: *framework* React Native, Node.JS e o sistema de gerenciamento de banco de dados PostgreSQL.

B. Trabalho Correlato II - MYTREAT: um aplicativo para o auxílio e controle de ingestão de medicamentos

Os autores Rosa e Sulzbach [16], apresentam um aplicativo (MyTreat) apenas para o uso pessoal, para sistemas móveis Android, que auxilia o usuário a utilizar os medicamentos no horário exato, por meio de notificações. O aplicativo foi desenvolvido pela ferramenta de desenvolvimento Android Studio, com SQLite, um banco de dados que armazena informações localmente no dispositivo do usuário. Segundo o caso de uso desenvolvido, o usuário pode ver o histórico de medicação, o cadastro de remédio, cadastrar dados pessoais, ver a agenda e receber notificações.

C. Trabalho Correlato III - Aplicativo mobile para controle e agendamento de consumo de medicamentos

O autor Junior [17], apresenta um aplicativo mobile de uso pessoal, para auxiliar pessoas a usarem seus remédios corretamente. O sistema permite o usuário a cadastrar medicamentos, cadastrar usuários, cadastrar prescrições, configurar alarmes e também visualizar seus medicamentos. Foi desenvolvido utilizando as tecnologias HTML, CSS, JavaScript, Bootstrap, JQuery e o *framework* Apache Cordova para gerar a aplicação mobile.

D. Considerações

Em relação ao trabalho de Filho, Oliveira e Lucas [15], foi percebido que ele é mais focado no compartilhamento de estoque de medicamentos entre familiares. Propõe em sua conclusão, implementar futuramente as funções de rotinas e também de notificações, sendo que as funções de autenticação, prateleira, categorias e medicamento foram concluídas.

No trabalho de Rosa e Sulzbach [16], foi notado que o aplicativo serve apenas para uso *offline*. Em sua conclusão, o aplicativo fornece o gerenciamento de medicamentos e o agendamento de horários, e que o sistema de notificações é funcional, a partir da frequência do medicamento, que é definida apenas por um intervalo de horas especificado.

Já no terceiro trabalho correlato, de Junior [17], foi observado que em sua conclusão, o trabalho consegue cumprir as suas funcionalidades, notificando o usuário para usar o seu medicamento cadastrado no sistema, a partir da frequência de uso, que também é definida apenas por um intervalo de horas definido. O autor propõe, após a conclusão, uma série de sugestões, como por exemplo, disponibilizar uma interface web para o usuário ter acesso e a implementação de notificações por SMS.

Todos os autores dos trabalhos correlatos colocaram a ênfase no quão importante é o desenvolvimento de sistemas voltadas no auxílio da saúde para a sociedade, assim como a importância da orientação para utilização correta de medicamentos por pacientes. Visto os três trabalhos correlatos, junto com as pesquisas realizadas, notou-se uma falta de aplicações web com este mesmo intuito - já que um site

abrangeria tanto os dispositivos móveis quanto computadores.

Este trabalho visa a desenvolver um sistema web, onde o usuário pode gerenciar medicamentos (para ele mesmo ou para alguém que ele esteja cuidando), podendo visualizar as medicações que deverá utilizar no dia e também seu catálogo completo de medicações. Se fundamentará também, a partir de uma sugestão no trabalho de Junior [17], implementando o envio de notificações SMS, quando for hora de utilizar a medicação.

IV. METODOLOGIA *Feature-Driven Development*

Para um projeto ter mais chances de suceder, é recomendado o uso de uma metodologia, que oriente no desenvolvimento do sistema a ser feito. A partir de princípios e processos, as metodologias de gestão de projetos deixam claro o que deve ser executado pela pessoa ou equipe, a modo de evitar problemas de comunicação, problemas envolvendo o tempo de entrega e problemas de desenvolvimento.

A metodologia escolhida para desenvolver e documentar este projeto proposto, foi uma metodologia ágil, chamada *Feature-Driven Development*, que foi criada originalmente por Peter Coad e seus colegas, sendo aperfeiçoada posteriormente por Stephen Palmer e John Felsing, resultando então em um método ágil que se adapta a projetos pequenos à grandes, tendo seu foco principal no desenvolvimento incremental, por funcionalidade de sistemas. As funcionalidades, conforme a metodologia, são funções do sistema que podem ser desenvolvidas em no máximo duas semanas, mas caso extrapole este tempo, devem ser divididas em blocos menores [18].

Segundo Pressman [18], esta metodologia possui cinco etapas para a realização do desenvolvimento com sucesso, cujas serão explicadas e mostradas ao longo do próximo tópico:

- Primeira Etapa - Construção de modelo geral.
- Segunda Etapa - Construir lista de funcionalidades.
- Terceira Etapa - Planejar por funcionalidades.
- Quarta Etapa - Projetar por funcionalidade.
- Quinta Etapa - Desenvolver por funcionalidade.

V. DESENVOLVIMENTO DO SISTEMA

Neste tópico, é abordado a parte de desenvolvimento do projeto, chamado provisoriamente de lembraMed, seguindo as cinco etapas da metodologia *Feature-Driven Development*, iterando a quarta e quinta fase (projetar e desenvolver, respectivamente), para cada funcionalidade.

A. Construção de Modelo Geral

A primeira etapa da metodologia consiste em construir um modelo geral, que resulta em um Diagrama de Domínio, como mostrado na Figura 1, que serve para ajudar na visualização das principais entidades e suas relações, assim como auxiliar no entendimento inicial de como a aplicação deve ser desenvolvida, definindo o problema a ser resolvido [19].

Como resultado desta etapa, é possível ter uma ideia inicial do sistema, que o usuário gerencia suas medicações, e que cada medicamento possui um histórico de utilização. O usuário também recebe notificações SMS, sobre suas medicações.

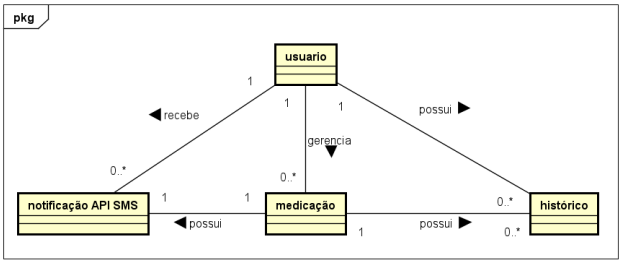


Figura 1. Diagrama de Domínio.

B. Construção de Lista de Funcionalidades

A segunda etapa consiste em construir a lista de funcionalidades, onde deve ser listado todos os requisitos necessários (funcionais e não funcionais) que o sistema precisa entregar, a partir das regras de negócios [19]. Como resultado, tem-se as Regras de Negócio listadas na Figura 2, com uma breve descrição sobre cada regra e um Diagrama de Caso de Uso.

Regra de Negócio	Descrição
RN01	Deve ser disponibilizado um sistema web
RN02	Deve existir um banco de dados para salvar os dados de usuários
RN03	Deve permitir o usuário a se cadastrar
RN04	Deve permitir o usuário a gerenciar/visualizar os seus dados
RN05	Os medicamentos serão adicionados pelos usuários
RN06	Os medicamentos serão gerenciados pelos usuários
RN07	Os medicamentos poderão ser vinculados a outras pessoas
RN08	O paciente poderá marcar/desmarcar que utilizou o medicamento no dia
RN09	O paciente poderá visualizar seu catálogo de medicamentos
RN10	Deve mostrar ao paciente os remédios que deve utilizar no dia em que se encontra
RN11	Deve ser possível orientar o paciente a utilizar o medicamento no horário correto por meio de notificações SMS no horário especificado
RN12	Deve disponibilizar uma opção para adicionar data de vencimento, para medicamentos
RN13	Deve alertar o paciente quando algum medicamento vencer
RN14	Deve ser possível trocar o horário/dia de uso de remédios
RN15	Deve ser possível visualizar o histórico de uso de medicações

Figura 2. Regras de Negócio

A partir das regras de negócios, foram geradas duas tabelas, uma citando os requisitos funcionais (Figura 3) e a outra, requisitos não funcionais (Figura 4), todas exibindo a descrição, relevância e complexidade de cada requisito.

Requisito Funcional	Descrição	Relevância	Complexidade
RF01	O sistema deve permitir o cadastro e controle de usuários	Alta	Baixa
RF02	Permitir a adição e controle de medicamentos com suas informações	Alta	Média
RF03	Permitir o usuário cadastrar e gerenciar medicamentos para outras pessoas além dele mesmo	Alta	Baixa
RF04	O sistema deve apresentar para o usuário, o histórico de uso das medicacões no catálogo	Alta	Baixa
RF05	Permitir o paciente ver os medicamentos que precisa tomar no dia	Alta	Alta
RF06	Permitir o paciente a marcar que utilizou o remédio no dia	Alta	Média
RF07	O sistema deve notificar por SMS o usuário no horário exato de utilização de medicamentos cadastrados	Alta	Alta
RF08	Para os pacientes, mostrar um catálogo de todos os remédios atuais e anteriores	Média	Média
RF09	Possibilitar o usuário a adicionar uma data de vencimento de um medicamento	Média	Baixa
RF10	Alertar o usuário caso um medicamento esteja vencido	Alta	Média

Figura 3. Requisitos Funcionais

Requisito Não-Funcional	Descrição	Relevância	Complexidade
RNF01	O aplicativo será desenvolvido a partir da linguagem JavaScript com Typescript	Essencial	Média
RNF02	O banco de dados a ser utilizado será o MongoDB	Essencial	Média
RNF03	Com o *framework* React, será desenvolvido o layout do sistema	Essencial	Média
RNF04	O *backend* da aplicação para ajudar no desenvolvimento, é Node.js com ajuda do *framework* Express	Essencial	Alta
RNF05	O envio de mensagens SMS será feito a partir da API da Twilio	Essencial	Alta

Figura 4. Requisitos Não-Funcionais

O Diagrama de Caso de Uso, exibido na Figura 5, contém o ator principal (usuário) e as atividades que pode exercer no sistema desenvolvido. O diagrama ajudou na compreensão do escopo do projeto, assim como deixar ilustrado as tarefas que os usuários podem realizar e consequentemente, as que não. Os principais casos de uso encontrados, foram o 'Gerenciar Medicamento' (UC07) e 'Visualizar medicamentos do dia' (UC03).

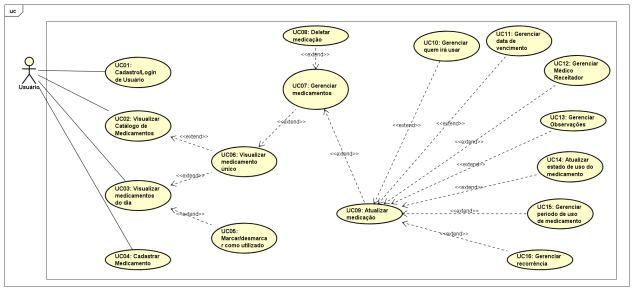


Figura 5. Diagrama de Caso de Uso.

C. Planejar por Funcionalidades

A terceira etapa da metodologia, consiste na criação da sequência do desenvolvimento. A partir da complexidade das funcionalidades do projeto [19], é estimado um tempo para o desenvolvimento de cada uma, como mostrado na Figura 6.

Número	Função	Tempo Estimado para Desenvolver
1	Gerenciamento de Usuários	30 horas
2	Gerenciamento de Medicações	60 horas
3	Gerenciamento de Histórico de Uso	30 horas
4	Catálogo de Medicações	30 horas
5	Visualização de Medicamentos para Hoje	60 horas
6	Notificações SMS	60 horas

Figura 6. Planejamento por Funcionalidade

D. Projetar por Funcionalidades

Na quarta etapa, é feito o detalhamento de cada funcionalidade, sendo executada apenas uma vez para cada função, assim como o refinamento do modelo abrangente do sistema, que é feito logo na primeira fase da metodologia [19]. A partir desta etapa, foi gerado, o diagrama de classes.

O Diagrama de Classes, mostrado na Figura 7, é o resultado do refinamento do modelo abrangente, adicionando também atributos e métodos para as classes do sistema. Com ele, foi possível mapear e entender as relações entre as classes, e o que cada uma pode armazenar e fazer dentro do sistema web.

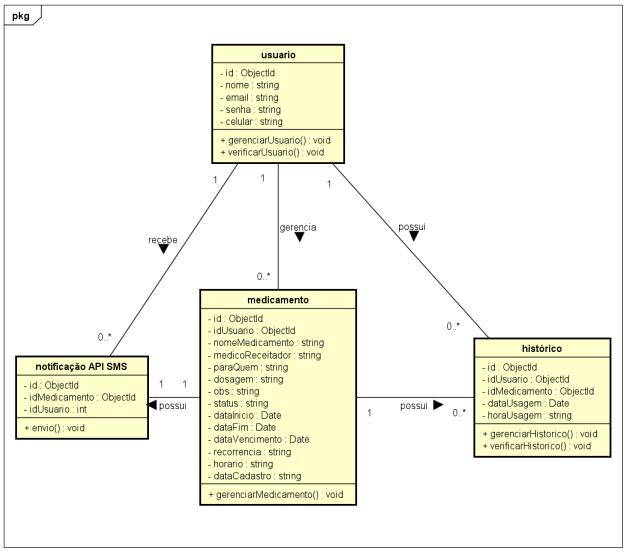


Figura 7. Diagrama de Classes.

E. Desenvolver por Funcionalidades

A quinta e última etapa da metodologia, é onde ocorre o desenvolvimento, se repetindo para cada funcionalidade do sistema, as quais foram detalhadas nas etapas anteriores. A partir das iterações desta etapa, foi gerado o projeto, de forma incremental [19].

A Figura 8 abaixo, mostra como ficou estruturado os arquivos do *frontend* (lembraMed) que entrega a as telas do sistema para o usuário interagir e do servidor *backend* (lembraMedServer) que deixa possível a autenticação, requi-

sição e interação com dados no MongoDB e a entrega de dados para o usuário.

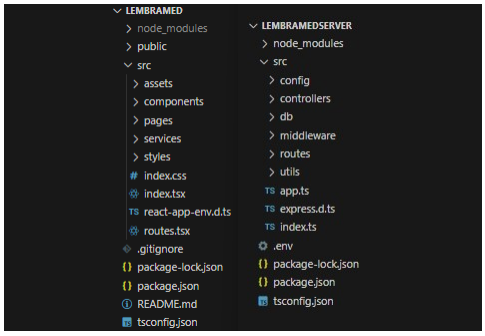


Figura 8. Pastas do projeto.

Nas seções a seguir, são mostrados detalhadamente os resultados do desenvolvimento das principais funcionalidades, entregando ao usuário, um sistema de lembretes e gerenciamento de medicações funcional.

1) Gerenciamento de Usuários: O primeiro produto dessa etapa, foi a parte de login e cadastro de usuários do sistema, a Figura 9 representa as interfaces geradas nesta etapa. Na tela de cadastro, o usuário faz a inserção de seus dados e ao clicar para fazer o cadastro, o sistema checa se os dados são válidos, para assim fazer contato com o *backend*. Utilizando a biblioteca AXIOS, o sistema envia os dados digitados pelo usuário ao servidor Node.JS + Express, que por sua vez checa se o email já existe no banco de dados MongoDB e caso não exista, a senha é encriptada usando a biblioteca BCrypt e assim, o usuário é cadastrado com sucesso.

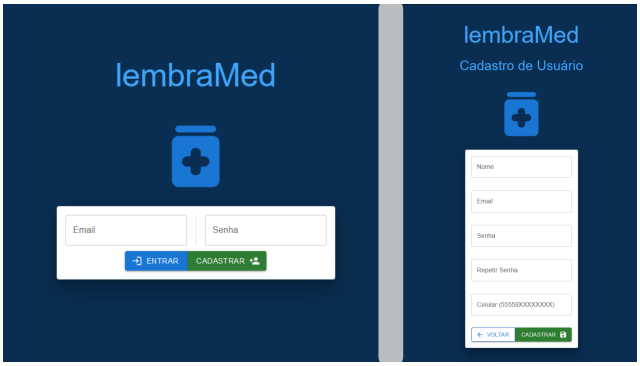
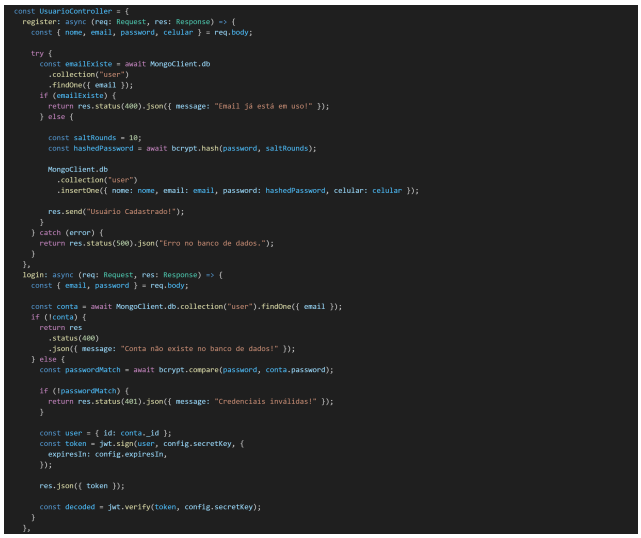


Figura 9. Interfaces de Login e de Cadastro.

Na parte de login de usuário, ao clicar para fazer login, é validado os dados e enviado uma requisição POST ao servidor, com o que foi digitado nos campos de usuário e senha. O *backend* recebendo a requisição de um login, verifica primeiramente se existe um email correspondente e caso exista, ele pega a senha existente no banco de dados que está encriptada e compara com a senha digitada pelo usuário, caso esteja tudo certo, é criada um *token* de sessão JWT. O *token* então é pego pelo *frontend* e salvo no localStorage do navegador, para manter a sessão, assim podendo acessar o sistema e rotas onde é necessário a autenticação. Na Figura 10, está o trecho de código referente às funções de login e cadastro no servidor Node, referenciadas anteriormente.



```
const UsuarioController = {
  register: async (req: Request, res: Response) => {
    const { nome, email, password, celular } = req.body;

    try {
      const emailExiste = await MongoClient.db
        .collection("user")
        .findOne({ email });
      if (emailExiste) {
        return res.status(400).json({ message: "Email já está em uso!" });
      } else {

        const saltRounds = 10;
        const hashedPassword = await bcrypt.hash(password, saltRounds);

        MongoClient.db
          .collection("user")
          .insertOne({ nome: nome, email: email, password: hashedPassword, celular: celular });

        res.send("Usuário Cadastrado!");
      }
    } catch (error) {
      return res.status(500).json("Erro no banco de dados.");
    }
  },
  login: async (req: Request, res: Response) => {
    const { email, password } = req.body;

    const conta = await MongoClient.db.collection("user").findOne({ email });
    if (!conta) {
      return res
        .status(400)
        .json({ message: "Conta não existe no banco de dados!" });
    } else {
      const passwordMatch = await bcrypt.compare(password, conta.password);

      if (!passwordMatch) {
        return res.status(401).json({ message: "Credenciais inválidas!" });
      }

      const user = { id: conta._id };
      const token = jwt.sign(user, config.secretKey, {
        expiresIn: config.expiresIn,
      });

      res.json({ token });

      const decoded = jwt.verify(token, config.secretKey);
    }
  },
```

Figura 10. Código do Node para Login e Cadastro.

2) Gerenciamento de Medicações: Na segunda etapa de construção, foi realizado o desenvolvimento das funções de cadastro, visualização, atualização e remoção das medicações do usuário. A Figura 11, representa as interfaces de cadastro e de edição, respectivamente.

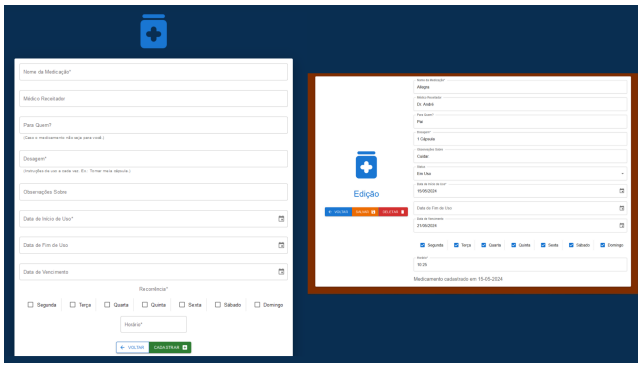


Figura 11. Interfaces de Cadastro e Edição de Medicações.

Ao clicar para adicionar um novo medicamento, o usuário é levado a uma página para fazer o cadastro. Ao inserir os dados e clicar para registrar, o sistema checa se os dados são válidos, avisando o usuário via alertas caso houver algo inválido.

Com tudo validado, é chamado a API de cadastro de medicações do servidor Node, pelo *frontend*, como mostra a Figura 12. A chamada envia os dados da medicação, junto

com o *token* JWT no *header*, que serve para identificar quem está cadastrando. O servidor ao receber a chamada POST, realiza por fim, a inserção do medicamento no banco de dados, antes checando se o medicamento já não existe e também valida os dados e o *token* recebido.

```
if (
      dadosValidosCadastro(
        nomeMedicamento,
        dosagem,
        dataInicio,
        horario,
        recorrencia,
        dataFim,
        dataVencimento
      )
    ) {
      try {
        const token = localStorage.getItem("authToken");

        const response = await axios.post(
          "http://localhost:3030/medication/register",
          {
            nomeMedicamento: nomeMedicamento,
            medicoReceitador: medicoReceitador,
            paraQuem: paraQuem,
            dosagem: dosagem,
            obs: obs,
            status: status,
            dataInicio: dataInicio,
            dataFim: dataFim,
            dataVencimento: dataVencimento,
            recorrencia: recorrencia,
            horario: horario,
            dataCadastro: dataCadastro,
          },
          {
            headers: {
              Authorization: `Bearer ${token}`,
            },
          }
        );
        toast.success("Cadastro Completado.");
        navigate("/medicamentos/catalogo");
      } catch (error) {
        toast.error(
          "Houve algum erro, verifique os campos preenchidos ou se algum campo obrigatório está vazio!"
        );
      }
    }
```

Figura 12. Trecho do *frontend* chamando a API de cadastro de medicações.

3) Gerenciamento de Histórico de Uso: Esta terceira fase de desenvolvimento, teve como produto a possibilidade do usuário registrar ou remover do registro atual a utilização do fármaco no dia em que ele se encontra (Figura 13a), assim como disponibilizar uma visualização do histórico de uso para cada medicação (Figura 13b). Ao interagir com o botão para cadastrar, o *frontend* envia uma requisição POST com ajuda do AXIOS, enviando o identificador do medicamento junto com o *token* de autorização, para o servidor.

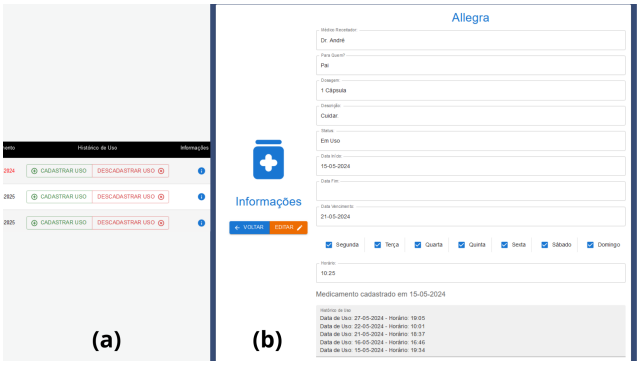


Figura 13. Partes do sistema referentes ao histórico de uso.

O servidor verifica a validez do *token* e identifica o usuário. Caso ainda não tenha sido registrado no dia em que se encontra, o *backend* se conecta com o banco de dados e insere no histórico uma entrada que possui: a identificação do usuário, a identificação do medicamento, a data e a hora de uso. A parte da API de registro de utilização, como explicada anteriormente, é mostrada na Figura 14.

```
register: async (req: Request, res: Response) => {
  const { idMedicamento } = req.body;

  const token = req.header("Authorization");

  if (!token) {
    return res.status(401).json({ message: "Token não encontrado." });
  }

  const tokenWithoutBearer = token.replace("Bearer ", "");

  const decoded = jwt.verify(tokenWithoutBearer, config.secretKey) as any;
  const idUsuario = decoded.id;

  //Pega a data e hora atual, e formata, para poder inseri-los no banco.
  const dataUsagem = moment().format("DD-MM-YYYY");
  const horaUsagem = moment().format("HH:mm");

  //Consulta o banco de dados, para ver se já foi registrado.
  const historico = await MongoClient.db
    .collection("historico")
    .find({
      idUsuario: new ObjectId(idUsuario),
      idMedicamento: new ObjectId(idMedicamento),
      dataUsagem: dataUsagem,
    })
    .toArray();

  //Checa se já existe no histórico, a partir do length.
  if (historico.length === 0) {
    try {
      //Então, insere o uso do medicamento no histórico do banco de dados.
      MongoClient.db.collection("historico").insertOne({
        idUsuario: new ObjectId(idUsuario),
        idMedicamento: new ObjectId(idMedicamento),
        dataUsagem: dataUsagem,
        horaUsagem: horaUsagem,
      });
      res.send("Usagem Cadastrada!");
    } catch (error) {
      return res.status(500).json({ message: "Erro!" });
    }
  } else {
    return res.status(500).json({ message: "Erro, já existe no histórico!" });
  }
```

Figura 14. API de registro no histórico.

4) Catálogo de Medicações: A quarta etapa de desenvolvimento deixou possível o usuário ver seu catálogo completo de medicações, como mostra a Figura 15. A interface possui uma tabela, evidenciando o nome do medicamento, mostra para quem é a medicação, a data de vencimento e o status de cada uma medicação. A cor da data de vencimento muda para amarelo caso esteja perto, ou vermelho caso já esteja vencida, para ajudar na visualização. O usuário pode preferir por ver mais informações, indo então para a interface de visualização do medicamento.

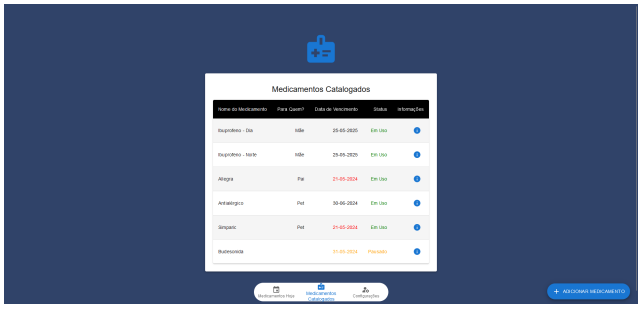


Figura 15. Interface do catálogo de medicações.

Ao usuário iniciar e renderizar a página, o sistema chama uma função para realizar uma chamada GET ao *backend*, enviando o *token* do usuário no *header*, requisitando o catálogo do usuário logado. O início deste fluxo é mostrado na Figura 16. O servidor se comunica com o banco de dados, buscando os medicamentos a partir do identificador do usuário, retornando um documento no formato JSON em array do catálogo para o *frontend*. Em seguida, os dados são ordenados primeiro pelo *status*, depois ordenado por para quem a medicação vai e por último, é ordenado pelo nome da medicação. Após a requisição e o tratamento dos dados, a página é atualizada, inserindo cada medicamento na tabela.

Figura 16. Trecho do *frontend* que dá inicio a chamada da API.

5) Visualização de Medicamentos para Hoje: A funcionalidade entregue na penúltima etapa, permite o usuário visualizar medicações que precisa utilizar ou administrar no dia em que se encontra, como mostra a Figura 17. Para ser possível essa visualização, a página ao ser renderizada, é feito uma chamada GET ao servidor, enviando o *token* de autenticação, para requisitar as medicações a serem utilizadas no dia. Logo, o servidor verifica o *token* e se conecta ao banco de dados, para buscar as medicações do usuário conectado que estão em uso.

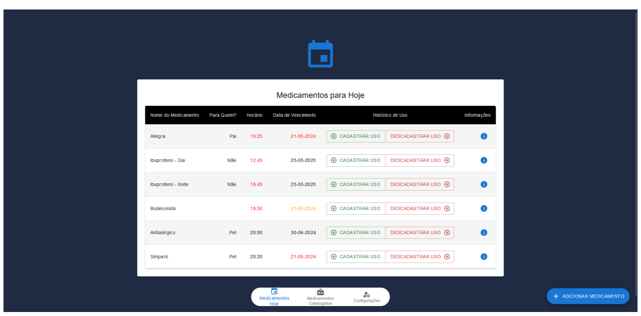


Figura 17. Interface mostrando as medicações para o dia atual.

Depois disso, o servidor filtra os resultados, deixando apenas as medicações que tem uma recorrência para o dia atual, e que está dentro do período indicado de uso, retornando ao *frontend* os resultados em JSON, como mostra a Figura 18. Estes dados então são pegos, e ordenados da mesma forma da tabela do catálogo. Após este fluxo, a página é atualizada, inserindo cada medicamento que precisa ser utilizado no dia atual, na tabela para o usuário.

Nesta tabela, é possível ter uma visualização do nome da medicação, para quem é medicação, o horário que deve ser usado e a data de vencimento. O usuário também pode registrar/descadastrar o uso da medicação no dia que se encontra ou ver mais informações sobre um fármaco em específico.

Figura 18. Trecho da API que retorna as medicações do dia.

6) Notificações SMS: Na sexta e última etapa do trabalho, foi implementado a notificação, enviando mensagens para o celular dos usuários, quando está na hora de utilizar certa medicação que tenha cadastrado em seu catálogo. Com ajuda da biblioteca NODE-CRON no servidor, a cada minuto que se passa, ela chama uma função, onde é iniciado uma busca no banco de dados por medicações que estão em uso e filtra os resultados, deixando apenas medicações que estão dentro do período de uso e que precisa utilizar hoje, também filtrando-os pelo horário em que o sistema se encontra.

Após o filtro, para cada resultado encontrado é buscado o número de celular do usuário, para então montar a mensagem SMS, contendo o nome da medicação, o horário de uso, a dosagem registrada, mostrando para quem é e alertando sobre a data de vencimento se o usuário tenha evidenciado. Com a mensagem montada, é feito o uso da API da Twilio para o envio das mensagens por SMS, para o número do usuário. A Figura 19 mostra o código *backend* de montagem e envio pela API, referenciados neste parágrafo, e ao lado direito, as mensagens sendo recebidas.

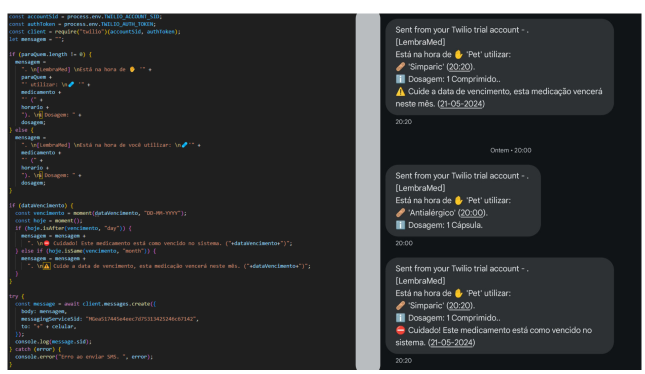


Figura 19. Código da montagem e envio de SMS, com resultado ao lado.

VI. Conclusão e Trabalhos Futuros

Desenvolvidas as etapas finais do trabalho de conclusão do curso, foi possível a partir dos moldes criados, desenvolver e entregar um sistema capaz de gerenciar medicações e enviar lembretes SMS aos usuários. Os referenciais teóricos ajudaram a manter o escopo do projeto, mostrando conceitos importantes sobre o controle de medicações e o uso da tecnologia voltada para a saúde. Também contribuíram para uma compreensão inicial das tecnologias que foram utilizadas, sendo elas JavaScript com React, o *framework* TypeScript, Node.JS com *framework* Express e por fim a API de SMS da Twilio.

Os trabalhos relacionados deram um norte no desenvolvimento, mostrando maneiras de implementação e deixando propostas para desenvolver um sistema de gerenciamento de medicações completo. Contribuíram também com o entendimento teórico sobre a importância de sistemas relacionados à área da saúde. Notou-se a partir das pesquisas, uma carência na parte de sistemas web de gerenciamento de medicações, dando uma oportunidade para o trabalho suprir esta falta, usando também tecnologias mais atuais. Como principal diferencial diante dos trabalhos correlatos, este trabalho implementou o alerta por SMS.

Guiando-se pela metodologia *Feature-Driven Development*, as primeiras três etapas ajudaram a ter um entendimento da visão geral do sistema, para então na quarta e quinta etapa, desenvolver o sistema de forma organizada e com as tecnologias propostas, iterando para cada função.

Finalizado o desenvolvimento de todas as funcionalidades, o produto foi um sistema onde o usuário em seu navegador pode cadastrar e gerenciar suas medicações, para que então, possa visualizar seu catálogo completo, ou visualizar apenas as medicações que precisa utilizar no dia em que se encontra. O sistema também conta com um registro de histórico de uso, podendo o usuário registrar o uso de um certo fármaco e visualizar os registros. Outra parte importante entregue, foi a de notificações por SMS por uma API de mensagens da Twilio, não precisando estar conectado para receber alertas na hora correta sobre qual medicamento precisa usar ou administrar para uma outra pessoa, também mostrando se o medicamento está vencido ou perto de vencer.

Com este produto, é possível ajudar pacientes a se organizarem quanto aos seus medicamentos e lembrá-los de utilizar seus medicamentos corretamente - no horário correto e dentro da validade. O sistema conta com a opção de administrar medicações para outras pessoas, caso cuide, por exemplo, de um *pet* ou outro alguém, melhorando a qualidade de vida não só do usuário, como de quem ele cuida.

Como sugestões de melhoria, para trabalhos futuros:

- Implementar a função para médicos poderem ter acesso ao catálogo e controlar medicações de seus pacientes.
- Disponibilização de outras opções de recorrência.
- Aprimorar a visualização para dispositivos móveis.
- Exportação do histórico de uso e do catálogo de medicações.
- Categorização de medicações.

Referências

[1] Lenita Wannmacher. *Importância dos Medicamentos Essenciais em Prescrição e Gestão Racionais*. 2010. URL: https://www.paho.org/ (acesso em 06/2022).

[2] UNICAMP. *Medicamento inadequado responde por 33,62 dos casos de intoxicação*. 2018. URL: https://www.unicamp.br/unicamp/clipping/2018/01/24/medicamento-inadequado-responde-por-3362-dos-casos-de-intoxicacao (acesso em 06/2022).

[3] Kamila Garan. *Qual é a importância do uso correto de medicamentos?* 2020. URL: https://www.farmajunior.com.br/farmacia/qual-e-a-importancia-do-uso-correto-de-medicamentos-2/ (acesso em 06/2022).

[4] Digital Health Association. *What is Digital Health?* 2020. URL: https://www.dha.org.nz/pages/what-is-digital-health (acesso em 05/2023).

[5] Food Drug Administration. *What is Digital Health?* 2020. URL: https://www.fda.gov/medical-devices/digital-health-center-excellence/what-digital-health (acesso em 05/2023).

[6] Quinn Grundy. *A Review of the Quality and Impact of Mobile Health Apps*. 2022. URL: https://doi.org/10.1146/annurev-publhealth-052020-103738 (acesso em 05/2022).

[7] Diaa Salama e Mohamed Abd-Elfattah. *Smart drugs:*Improving healthcare using Smart Pill Box for Medicine Reminder and Monitoring System. 2018. URL: https://www.sciencedirect.com/science/article/pii/S2314728818300230 (acesso em 06/2022).

[8] Adam D. Scott. *JavaScript Everywhere:* Building Cross-Platform Applications with GraphQL, React, React Native, and Electron. Paperback. O'Reilly Media, 2020.

[9] Vinicios Neves. *React:* o que é, como funciona e um Guia dessa popular ferramenta JS. 2023. URL: https://www.alura.com.br/artigos/react-js (acesso em 05/2023).

[10] TypeScriptLang. *TypeScript for the New Programmer*. 2023. URL: https://www.typescriptlang.org/docs/handbook/typescript-from-scratch.html (acesso em 10/2023).

[11] Alex Young, Bradley Meck e Mike Cantelon. *Node.js in Action*. 2th. Manning Publications, 2017.

[12] Mozilla Developer Network. *Introdução Express/-Node*. 2023. URL: https://developer.mozilla.org/pt-BR/docs/Learn/Server-side/Express_Nodejs/Introduction#introduzindo_o_express (acesso em 11/2023).

[13] MongoDB. *O que é o MongoDB?* 2023. URL: https://www.mongodb.com/pt-br/what-is-mongodb (acesso em 10/2023).

[14] Twilio. *APIs para SMS, voz, e-mail e autenticação - Twilio*. 2024. URL: https://www.twilio.com/pt-br (acesso em 05/2024).

[15] Wilson Filho, José Oliveira e Carlos Lucas. *PRATELEIRA:* desenvolvimento de uma solução mobile para controle de medicamentos e rotinas medicinais. 2020.

[16] Dioni Rosa e Maurício Sulzbach. *MYTREAT:* um aplicativo para o auxílio e controle de ingestão de medicamentos. 2017.

[17] José Borba Junior. *Aplicativo mobile para controle e agendamento de consumo de medicamentos*. 2015.

[18] Roger Pressman. *Engenharia de Software - Uma Abordagem Profissional*. 7th. AMGH, 2011.

[19] Alexandre Magno Figueiredo. *FDD Numa Casca de Banana - Um guia de rápido aprendizado para a Feature-Driven Development*. Autopublicação, 2007.