UNIVERSIDADE DE TRÁS-OS-MONTES E ALTO DOURO

ELTON PENICELA

elton.penicela@ispt.ac.mz

Mestrado em Engenharia Informática e Tecnologia Web

PROGRAMAÇÃO WEB AVANÇADA.

Link do projecto (backend, frontend Mysql), Mockup, wireframe e prototipo

https://github.com/epenicela/ProjectoFinal.git

Sistema de gestão de vacinas

Introducação

Frontend e *Backend* são duas partes fundamentais de uma aplicação web. O *frontend* é a camada responsável pela interface com o usuário, ou seja, é a parte visível do sistema. Já o *backend* é a camada que cuida da lógica do sistema, como processamento de dados, armazenamento e gerenciamento de informações.

Para desenvolver um sistema web completo, é necessário combinar essas duas camadas de forma harmoniosa, de modo que a experiência do usuário seja agradável e a navegação seja rápida e segura. As tecnologias mais usadas na actualidade são VueJs (Frontend) e NodeJs(Backend), por possuir uma vantagem de usarem ambas usam JavaScript como linguagem principal, o que permite que você use a mesma linguagem em todo o aplicativo, desde o lado do cliente até o lado do servidor. Isso ajuda a simplificar o desenvolvimento e a manutenção do aplicativo.

Entretanto fazendo o uso das tecnologias acima citadas, no presente trabalho pretende-se criar um sistema de gestão de vacinas que tem como objetivo facilitar o registro e acompanhamento das vacinas aplicadas em indivíduos, bem como auxiliar no planejamento e distribuição das doses.

Essa plataforma poderá ser utilizada por profissionais da saúde, como enfermeiros e médicos, que inserem os dados das vacinas aplicadas em um sistema informatizado. Dessa forma, será possível monitorar quantas doses já foram aplicadas em determinada região, controlar o estoque de vacinas e identificar possíveis faltas de doses em alguma área específica.

Além disso, o sistema eletrônico de gestão de vacinas possibilitará um melhor controle de campanhas de vacinação, já que é possível identificar quais grupos de pessoas ainda precisam ser imunizados e quais vacinas são indicadas para cada faixa etária.

Trazendo assim mais segurança e eficiência na gestão de vacinas, pois diminui as chances de erros no registro e garante um melhor acompanhamento da cobertura vacinal da população. Com isso, é possível combater doenças e epidemias com mais efetividade, contribuindo para a saúde pública.

Backend

O Node.js é uma plataforma de software livre, baseada no interpretador V8 do Google, que permite a criação de aplicações server-side utilizando JavaScript. Ele é altamente escalável e orientado a eventos, e é muito utilizado para construir APIs RESTful, sistemas de gerenciamento de conteúdo, aplicações de streaming de vídeo, dentre outras soluções web.

```
npm init -y
npm install express
npm install -g nodemon
npm install mysql
```

- **npm init -y**: Esse comando cria um arquivo package.json na pasta em que foi executado. Ele contém as informações sobre o projeto, como nome, descrição, autor, versão, dependências e scripts.
- **npm install express**: Esse comando instala o framework Express no projeto. O Express é um framework Node.js que ajuda a construir aplicações web de forma mais fácil e rápida, oferecendo uma série de recursos e funcionalidades para lidar com rotas, middlewares, templates, etc.
- **npm install -g nodemon**: Esse comando instala o Nodemon globalmente no sistema. O Nodemon é uma ferramenta que reinicia automaticamente o servidor Node.js toda vez que detecta alterações no código fonte, o que é muito útil durante o desenvolvimento.

- **npm install mysql**: Esse comando instala o driver MySQL para Node.js no projeto. O MySQL é um sistema de gerenciamento de banco de dados relacional, e o driver permite que o Node.js se conecte e interaja com o MySQL, fazendo consultas e manipulando os dados.

Rotas

Rotas em Node.js são definidas para mapear URLs específicas para funções de manipulação de requisições (request handlers) em um servidor web. É uma forma de definir como as requisições feitas por um cliente serão tratadas pelo servidor. Essas rotas podem ser definidas utilizando o framework Express.js, que é uma das opções mais populares para o desenvolvimento de aplicações web em Node.js.

Para definir uma rota em Node.js utilizando Express.js, é preciso seguir alguns passos:

i. Instale o pacote do Express.js no seu projeto:

```
npm install express
```

ii. Crie uma instância do Express.js:

```
const express = require('express');
const app = express();
```

iii. Defina uma rota para o seu servidor:

```
app.get('/', function (req, res) {
  res.send('Olá, mundo!');
});
```

Neste exemplo, a rota '/' (raiz) é definida para responder a uma requisição GET enviando a mensagem "Olá, mundo!" como resposta.

iv. Inicie o servidor:

```
app.listen(3000, function () {
  console.log('Servidor iniciado na porta 3000!');
});
```

Neste exemplo, o servidor é iniciado na porta 3000.

É possível definir rotas para diferentes tipos de requisição HTTP, como GET, POST, PUT, DELETE, entre outros. Além disso, também é possível definir rotas com parâmetros dinâmicos, que permitem que as URLs sejam mais flexíveis e possam receber valores diferentes a cada requisição.

Exemplo de rota com parâmetro dinâmico:

```
app.get('/usuario/:id', function (req, res) {
  res.send('Usuário ' + req.params.id);
});
```

Neste exemplo, a rota '/usuario/:id' é definida para responder a uma requisição GET enviando a mensagem "Usuário " seguido do valor do parâmetro "id" como resposta. Esse valor será diferente a cada requisição.

Em resumo, rotas em Node.js são utilizadas para definir como as requisições feitas por um cliente serão tratadas pelo servidor. Elas são definidas utilizando o framework Express.js e permitem que as URLs sejam mapeadas para funções específicas que serão responsáveis por manipular as requisições.

Frontend

Vue.js é um framework JavaScript de código aberto que permite a criação de interfaces de usuário dinâmicas e reativas. Ele é voltado para o desenvolvimento de aplicações de página única (SPA), o que significa que o conteúdo é carregado uma única vez e, em seguida, é atualizado dinamicamente conforme o usuário interage com a aplicação.

```
npm install -g @vue/cli
vue create frontend
vue add vuetify
npm install axios
```

"npm install -g @vue/cli" é usado para instalar globalmente a CLI (Interface de Linha de Comando) do Vue, que é uma ferramenta para desenvolver aplicativos Vue.js mais facilmente.

"vue create frontend" é usado para criar um novo projeto Vue.js chamado "frontend". Isso criará uma estrutura de arquivos básica para o projeto, incluindo um arquivo "package.json" que gerencia as dependências do projeto.
"vue add vuetify" é usado para instalar o framework Vuetify no projeto Vue.js. O Vuetify é um framework de UI (User Interface) que ajuda a criar interfaces de usuário consistentes e atraentes.
"npm install axios" é usado para instalar a biblioteca Axios, que é uma biblioteca para fazer requisições HTTP (Ajax) no navegador. Isso é útil para acessar APIs e enviar dados para um servidor em um aplicativo Vue.js.

Rotas

As rotas em Vue.js são usadas para navegar em diferentes componentes da aplicação com base na URL atual. As rotas são definidas no objeto router que é passado para a instância Vue, que é criada para renderizar a aplicação.

Para definir rotas no Vue.js, é necessário primeiro importar o Vue Router e usar a função Vue.use para instalá-lo:

```
import Vue from 'vue'
import VueRouter from 'vue-router'

Vue.use(VueRouter)
```

Components

O Vue.js é um framework JavaScript que permite a criação de interfaces de usuário interativas e reativas em páginas web. Para construir essas interfaces, o Vue.js oferece diversos componentes que podem ser utilizados para criar elementos visuais, manipular dados, lidar com eventos, entre outras funcionalidades.

Alguns dos principais componentes do Vue.js incluem:

Componentes: permitem a criação de elementos reutilizáveis e organizados em hierarquias para a construção de interfaces mais complexas.

Diretivas: são instruções que permitem modificar o comportamento ou a aparência dos elementos HTML.
Filtros: permitem a formatação e manipulação de dados que serão exibidos na interface.

Mixins: permitem a reutilização de comportamentos e métodos entre diferentes componentes.

Watchers: permitem a observação de mudanças em variáveis ou propriedades de um componente, executando ações em resposta a essas mudanças.
Eventos: permitem a comunicação entre diferentes componentes da interface, disparando ações e atualizações de dados.

Esses e outros componentes do Vue.js são usados para criar interfaces reativas e dinâmicas, permitindo uma experiência de usuário mais fluida e interativa.

Consumindo o backend como API

Axios é uma biblioteca JavaScript baseada em Promises para fazer solicitações HTTP do lado do cliente. Ele é amplamente utilizado em aplicações front-end para enviar e receber dados de um servidor.
Para consumir uma API no VueJS, você pode utilizar a biblioteca axios, que é uma biblioteca de clientes HTTP baseada em promessas que pode ser usada tanto no navegador quanto no Node.js.
Para utilizá-la, primeiro você precisa instalá-la no seu projeto:
npm install axios
Depois disso, é importado arquivo JavaScript Vue e utilizando-a para fazer requisições HTTP. Por exemplo, para obter uma lista de vacinas do backend:

```
import axios from 'axios';

const api = axios.create({
```

```
    baseURL: 'http://localhost:3000/'
});

export default api;
```

‘

O axios.create é um método da biblioteca Axios que permite criar uma nova instância de um objeto Axios com configurações predefinidas. Isso pode ser útil em projetos Vue.js para definir configurações de solicitação HTTP padrão, como a URL base da API, os headers de autenticação ou os interceptadores de solicitação e resposta.

Por exemplo, ao criar uma nova instância Axios com axios.create, é possível definir a baseURL da API como 'http://localhost:3000/'.
O metodo listar através do arquivo acima que faz o uso do axio ela faz a requisição para o backend(API) que por sua vez retorna a lista das vacinas em formato adequado.

```
import api from ./api';


export default{
    listar:() =>{
        return api.get('vacinas');
    },
}
```

Por sua vez no código a seguir e passa a resposta na colecao de um componente da View

```
listar() {
      vacina.listar().then(response => {
        this.items = response.data
      })
    }
```

Também foram utilizar outras funções HTTP, como POST, PUT e DELETE, e passar parâmetros para a requisição, como cabeçalhos e dados do corpo.

Construção de ambiente gráfico

Vuetify é um framework de componentes de interface de usuário baseado no Vue.js. Ele oferece uma ampla gama de componentes pré-construídos e personalizáveis, como botões, barras de

navegação, listas, caixas de diálogo, formulários e muito mais. Com o Vuetify, os desenvolvedores podem criar aplicativos da web elegantes e responsivos com rapidez e facilidade, sem a necessidade de criar cada componente do zero.

Nesta fase segui-se com a criação do menu de administração, para tal foram seguidos os seguintes passos:

1. Instacao do Vuetify no projecto:

   npm install vuetify

2. Importar o Vuetify em arquivo main.js:

```
import Vue from 'vue'
import Vuetify from 'vuetify'
import 'vuetify/dist/vuetify.min.css'

Vue.use(Vuetify)
```

3. Criar um componente de menu, por exemplo PacienteView.vue, com o seguinte código:

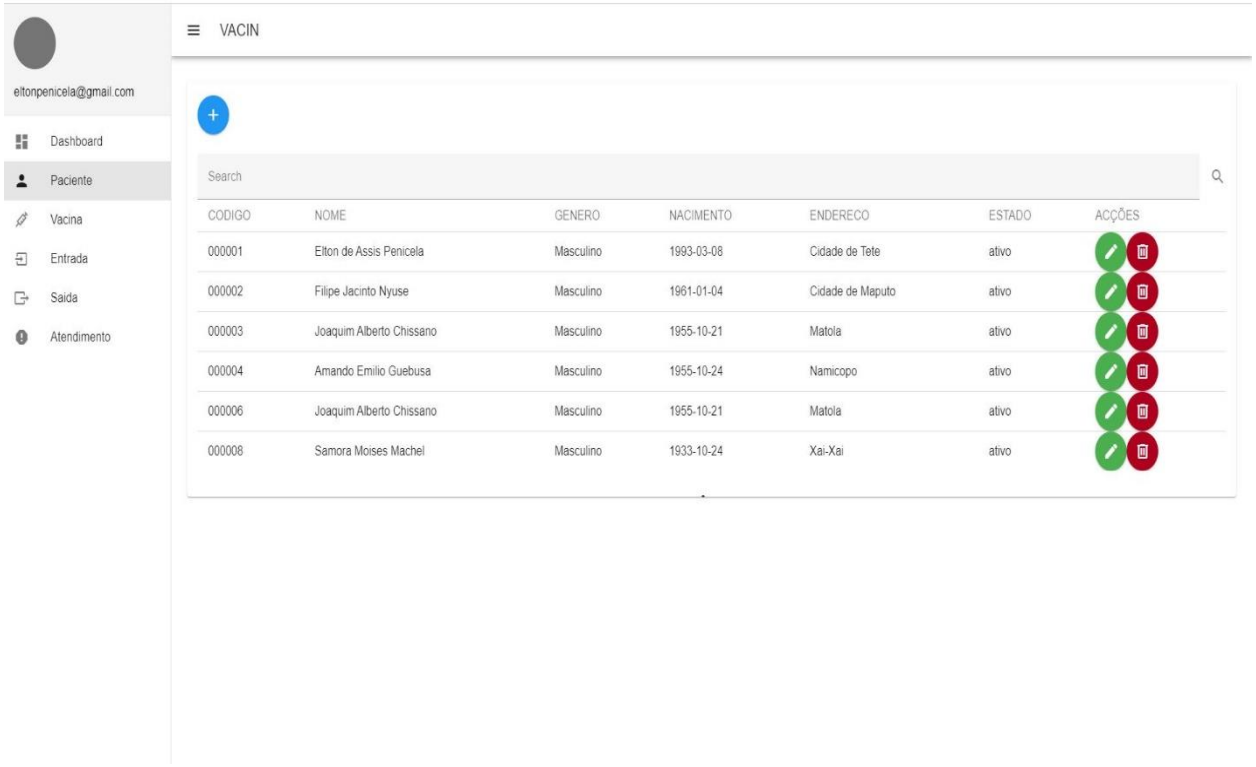
```
<template>
  <v-navigation-drawer app :value="drawer">
    <v-list>
      <v-list-item    v-for="item    in    items"    :key="item.title"
:to="item.link">
        <v-list-item-icon>
          <v-icon>{{ item.icon }}</v-icon>
        </v-list-item-icon>
        <v-list-item-content>
          <v-list-item-title>{{ item.title }}</v-list-item-title>
        </v-list-item-content>
      </v-list-item>
    </v-list>
  </v-navigation-drawer>
</template>

<script>
export default {
  data() {
    return {
      drawer: true,
```

```
        items: [
          { title: 'Dashboard', icon: 'mdi-view-dashboard', link: '/' },
          { title: 'Usuários', icon: 'mdi-account', link: '/users' },
          { title: 'Produtos', icon: 'mdi-shopping', link: '/products' },
          { title: 'Configurações', icon: 'mdi-settings', link: '/settings'
},
        ],
      }
    },
}
</script>
```

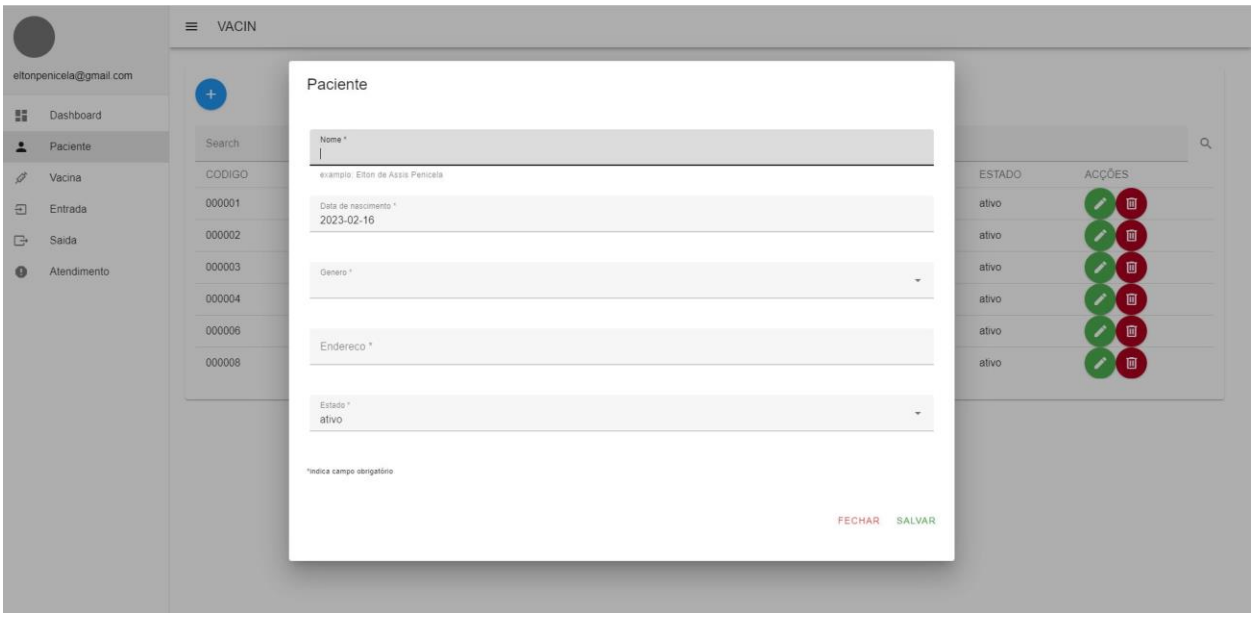
4. Adicione o componente PacienteView ao componente App.vue, por exemplo:

```
<template>
  <v-app>
    <admin-menu />
    <v-main>
      <!-- Conteúdo da página aqui -->
    </v-main>
  </v-app>
</template>

<script>
import AdminMenu from './AdminMenu.vue'

export default {
  components: {
    AdminMenu,
  },
}
</script>
```

5. Desta forma procegui-se criando todas outras telas e formulário.



Nota: os formulários foram criados usando como recurso o modal.



Conexão com Banco de Dados

MySQL é um sistema de gerenciamento de banco de dados relacional de código aberto. Ele utiliza a linguagem SQL (Structured Query Language) para manipular dados em bancos de dados relacionais.

Nesta fase foi criado um pacote designado “Infrastutura” que sera responsável por criar todos recuros para estabelecer uma conexão entra o banco de dados (**Mysql**) e a aplicação eem **nodejs.**

Dentro do pacote “infrastutura” fora criados dois arquivos “conexão.js” e “tabela.js” que por sua ves eram requisitado por outro “model” onde continha alguns instruções em sql.

Abaixo segue um exemplo de conexão (arquivo: conexão.js) com banco de dados Mysql:

```
const mysql = require("mysql");


const conexao = mysql.createConnection({
    host: "localhost",
    port: 3306,
    user: "root",
    password: "",
    database: "u868151405_pweb"
});

module.exports = conexao;
```

De lembrar que existe um prerequisito para tornar essas intrucoes funcionais que é a intalacao da dependência do Mysql:

```
npm install mysql
```

De seguida criou-se uma classe de nome tabela que tinha

```
class Tabelas {
    init(conexao) {
        this.conexao = conexao;
        this.criarTabelaVacina();
        this.criarTabelaEntrada();
        this.criarTabelaSaida();

        this.criarTabelaEntradaTriggerInserted();
        this.criarTabelaEntradaTriggerDeleted();
        this.criarTabelaSaidaTriggerInserted();
    }
```

```
    criarTabelaVacina() {
        const sql =
            'CREATE TABLE IF NOT EXISTS vacina (' +
            'id INT(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,' +
            'nome_vac VARCHAR(50) NOT NULL,' +
            'validade DATE NOT NULL,' +
            'nr_dose INT NOT NULL,' +
            'quantidade INT NOT NULL,' +
            'STATUSVAC ENUM("ativo","inativo") NULL DEFAULT "ativo")' +
            ' COLLATE="latin1_swedish_ci" ENGINE=InnoDB AUTO_INCREMENT=1;';

        this.conexao.query(sql, (error) => {
            if (error) {
                console.log("Erro ao criar a tabela VACINA");
                console.log(error.message);
                return;
            }

            console.log("");
            console.log("CRIANDO TABELAS...");
            console.log("Tabela VACINA criada com sucesso...");
        });
    }

}


module.exports = new Tabelas();
```

O código em Node.js apresentado é apenas uma pequena amostra do código no seu todo, que é responsável por criar tabelas em um banco de dados MySQ, logo que aplicação esteja no ar, caso ela não encontre asa tabelas necessárias o *beckend* cria de forma automática, garantindo assim maior portabilidade do codigo. As tabelas que são criadas pelo código são: vacina, entrada, saida, paciente e atendimento. Além disso, o código também cria gatilhos (triggers) para essas tabelas, que são executados após a inserção ou deleção de dados nas tabelas. O método init é responsável por inicializar a conexão com o banco de dados, e em seguida, chama os métodos para criar as tabelas e os gatilhos. Cada método é responsável por criar uma tabela específica, sendo que cada tabela é composta por colunas que armazenam diferentes informações relacionadas a vacinas, pacientes e atendimentos. O código utiliza comandos SQL para criar as tabelas, como CREATE

TABLE IF NOT EXISTS e FOREIGN KEY, que definem as relações entre as tabelas. Por fim, o código exibe mensagens de sucesso ou erro para indicar se a criação das tabelas e gatilhos foi realizada com sucesso ou não.

Conclusão

Em conclusão, Full Stack é uma abordagem de desenvolvimento de software que se concentra em criar aplicativos que abrangem todo o ciclo de vida do desenvolvimento. Isso significa que um desenvolvedor Full Stack tem conhecimentos tanto de back-end quanto de front-end, permitindo que ele trabalhe em todas as camadas da aplicação e crie soluções de ponta a ponta. Isso se traduz em um desenvolvimento mais eficiente, maior velocidade de entrega, melhor qualidade de código e uma experiência de usuário aprimorada. A abordagem Full Stack é altamente valorizada pelas empresas e continua a ser uma habilidade muito procurada no mercado de trabalho.

Todavia, a combinação do Node.js no backend e Vue.js no frontend é uma solução poderosa para o desenvolvimento de aplicativos web modernos e escaláveis. Com o Node.js, é possível construir servidores robustos e escaláveis com código JavaScript no lado do servidor, e com o Vue.js, é possível criar interfaces de usuário modernas e reativas com uma curva de aprendizado relativamente fácil. Juntos, o Node.js e o Vue.js permitem o desenvolvimento de aplicativos full stack altamente eficientes, tornando-os uma escolha popular para projetos de desenvolvimento de software modernos.

Com o uso do banco de dados relacional Mysql foram alcançados os objectivos do projecto que se pretendia desenvolver, onde no final quem podesse geria a plicacao em causa conseguia ter a relação de quantas doses de vacinas já foram aplicadas em determinada região, controlar o estoque de vacinas e identificar possíveis faltas de doses em alguma área específica, como acompanhar os paciente vacinados.

Link do projecto no git::
https://github.com/epenicela/ProjectoFinal.git

Referencias

"Web Development with Node and Express: Leveraging the JavaScript Stack" de Ethan Brown.

"Learning React: A Hands-On Guide to Building Web Applications Using React and Redux" de Kirupa Chinnathambi.

"CSS Secrets: Better Solutions to Everyday Web Design Problems" de Lea Verou.

"Designing Web Interfaces: Principles and Patterns for Rich Interactions" de Bill Scott e Theresa Neil.

Site oficial do Vuetify: https://vuetifyjs.com/

Documentação do Vuetify: https://vuetifyjs.com/en/getting-started/quick-start

"Vuetify: A Vue.js Material Component Framework" - artigo do site Medium: https://medium.com/tech-laboratory/vuetify-a-vue-js-material-component-framework-54cfbf16f44a

"Vuetify Tutorial: Build an App with Vue.js, Firebase, and Material Design Components" - artigo do site Scotch.io: https://scotch.io/tutorials/vuetify-tutorial-build-an-app-with-vuejs-firebase-and-material-design-components.

"Vue.js: Up and Running" de Callum Macrae (O'Reilly Media, 2018)

"Learning Vue.js 2" de Olga Filipova (Packt Publishing, 2016)

Documentação oficial do Vue.js: https://vuejs.org/v2/guide/