

Projeto Nest.js e Vue.js

Alunos: Augusto Castejon, Álvaro Dias, Luiz Felipe e Vinicius Dogonski

1. Criando as funções no Backend

1.1. Em chave.service.ts, coloque:

```
import { Injectable, NotFoundException } from
 '@nestjs/common';
import { Model } from 'mongoose';
import { InjectModel } from '@nestjs/mongoose';
import { Chave } from '../model/chave.model';

@Injectable()
export class ChaveService {
  constructor(@InjectModel('Chave') private readonly
    chaveModel: Model<Chave>) { }

  async listarChaves(): Promise<Chave[]> {
    return await this.chaveModel.find({ status: true
    }).exec();
  }

  async criarChave(nome: string): Promise<Chave> {
    try {
      const chave = new this.chaveModel({ nome,
situacao: "DISPONIVEL", status: true });
      return await chave.save();
    } catch (Error) {
      console.log(Error.message)
      throw new NotFoundException(
        Error.message
      );
    }
  }

  async buscarChavePorNome(nome: string): Promise<Chave> {
    const chave = await this.chaveModel.findOne({ nome
    });

    if (!chave) {
```

```

        throw new NotFoundException('Chave não encontrada');
    }
    return chave;
}

    async alterarChave(nome: string, novosDados: Partial<Chave>): Promise<Chave>{
        const chave = await this.chaveModel.findOne({nome});

        if(!chave){
            throw new NotFoundException('Chave não encontrada');
        }

        Object.assign(chave, novosDados);

        return await chave.save();
    }

    async desativarChave (nome : string): Promise<Chave> {
        const chave = await this.chaveModel.findOne({ nome });
    });
        if (!chave) {
            throw new NotFoundException('Chave não encontrada');
        }
        chave.status = false;

        return await chave.save();
    }
}

```

1.2. Em chave.controller.ts, coloque:

```

import { Controller, Get, Post, Body, Param, Put, Patch }
from '@nestjs/common';
import { ChaveService } from '../chave.service';
import { Chave } from '../model/chave.model';

```

```
@Controller('chave')
export class ChaveController {
  constructor(private readonly chaveService: ChaveService) {}

  @Get()
  async listarChaves(): Promise<Chave[]> {
    return this.chaveService.listarChaves();
  }

  @Post()
  async criarChave(@Body() data: Chave): Promise<Chave> {
    console.log(data);
    return this.chaveService.criarChave(data.nome);
  }

  @Get('/:nome')
  async buscarChavePorNome(@Param('nome') nome: string) {
    try {
      const chave = await
this.chaveService.buscarChavePorNome(nome);
      return {chave};
    } catch (error) {
      return {error: error.message};
    }
  }

  @Put('/:nome')
  async alterarChave(
    @Param('nome') nome : string,
    @Body() novosDados : Partial<Chave>
  ) {
    try {
      const chaveAlterada = await
this.chaveService.alterarChave(
        nome,
        novosDados,
      );
      return { chave: chaveAlterada, message: 'Chave alterada
com sucesso' };
    } catch (error) {
      return { error: error.message };
    }
  }
}
```

```

    }
  }

  @Patch('desativar/:nome')
  async desativarChave(@Param('nome') nome: string) {
    try {
      const chaveDesativada = await
this.chaveService.desativarChave(nome);
      return { chave: chaveDesativada, message: 'Chave
desativada com sucesso' };
    } catch (error) {
      return { error: error.message };
    }
  }
}

```

2. Criando as funções no Frontend

- 2.1. Antes de iniciar o front, vamos fazer downloads de alguns itens para estilizar nosso projeto:
- <https://github.com/luiz-felippelb/vuejs-emprestimoChave-daw1/blob/main/src/assets/add.png>
 - <https://github.com/luiz-felippelb/vuejs-emprestimoChave-daw1/blob/main/src/assets/edit.png>
 - <https://github.com/luiz-felippelb/vuejs-emprestimoChave-daw1/blob/main/src/assets/find.png>
 - <https://github.com/luiz-felippelb/vuejs-emprestimoChave-daw1/blob/main/src/assets/key.png>

- 2.2. Em `\src\components\modals` crie `AlterarChave.vue`, `BuscarChave.vue` e `RemoverChave.vue`.

- 2.3. Em `AlterarChave.vue` coloque:

```

<script setup lang="ts">
import { ref, onMounted, onUpdated, defineProps } from
'vue';
import axios from 'axios';

interface Chave {
  nome: string;
  situacao: string;
  status: boolean;
}

```

```

}

const props = defineProps({
  show: Boolean,
});

const key = ref('');
const listKey = ref<Chave[]>([]);

const fetchData = async () => {
  try {
    const response = await
axios.get('http://localhost:3000/chave');
    listKey.value = response.data as Chave[];
  } catch (error) {
    console.error('Erro na requisição à API: ', error);
  }
};

onMounted(() => {
  fetchData();
});

onUpdated(() => {
  fetchData();
});

const nomeChave = ref('');
const Message = ref('');

const alterarNomeChave = async () => {
  if (key.value.trim() !== '' && nomeChave.value.trim()
!== '') {
    try {
      Message.value = '';
      const chaveSelecionada =
listKey.value.find((chave) => chave.nome === key.value);

      if (chaveSelecionada) {
        const response = await
axios.put(`http://localhost:3000/chave/${chaveSelecionada.no
me}`, {
          nome: nomeChave.value,

```

```

    });

    // Lide com a resposta da API, por exemplo,
    exibindo uma mensagem de sucesso
    console.log('Nome da chave alterado com
sucesso:', response.data);

    // Atualizar a lista de chaves após a
alteração

    fetchData();

    // Limpar os campos após a alteração
    key.value = '';
    nomeChave.value = '';
    Message.value = 'Nome alterado com sucesso';
  } else {
    Message.value = 'Chave não encontrada';
  }
} catch (error) {
  Message.value = 'Erro ao alterar o nome da
chave';
  console.error('Erro ao alterar o nome da chave
no banco', error);
}
} else {
  Message.value = 'Selecione uma chave e forneça um
novo nome';
}
};
</script>

<template>
  <link rel="stylesheet" href="src/assets/modal.css">
  <Transition name="modal">
    <div v-if="show" class="modal-mask">
      <div class="modal-container">

        <div class="modal-header">
          <h3>Editar Chave</h3>
        </div>

        <div class="modal-body">

```

```

        <form
@submit.prevent="alterarNomeChave">
            <select v-model="key"
id="selectKey">
                <option value="" disabled
selected hidden>Selecione uma chave</option>
                <option v-for="chave in listKey"
:value="chave.nome">{{ chave.nome }}</option>
            </select>

            <label for="nomeChave">Novo Nome da
Chave:</label>

            <input type="text" id="nomeChave"
v-model="nomeChave" required>
            <button
type="submit">Alterar</button>

            <p :class="{ 'visible': Message !==
''}" >{{ Message }}.</p>
        </form>
    </div>

    <button class="modal-default-button"
@click="$emit('close') ">X</button>
</div>
</div>
</Transition>
</template>

<style scoped>
</style>

```

2.4. Em BuscarChave.vue coloque:

```

<script setup lang="ts">
import { ref, getCurrentInstance } from 'vue';
import axios from 'axios';

const props = defineProps({
    show: Boolean
})

```

```

const nomeChave = ref('')
const Message = ref('')

const buscarChave = async () => {
  if (nomeChave.value.trim() !== '') {
    try {
      Message.value = ''
      const response = await
axios.get(`http://localhost:3000/chave/${nomeChave.value}`);
      console.log('Chave: ', response.data);
      // Emitir evento para o componente pai com o
valor da chave
      const instance = getCurrentInstance();
      if (instance) {
        instance.emit('chave-encontrada',
response.data);
      }
      Message.value = 'Chave selecionada'
    } catch (error) {
      Message.value = 'Chave não encontrada'
      console.error('Erro ao buscar a chave no banco',
error);
    }
  }
}
</script>

<template>
  <link rel="stylesheet" href="src/assets/modal.css">
  <Transition name="modal">
    <div v-if="show" class="modal-mask">
      <div class="modal-container">

        <div class="modal-header">
          <h3>Buscar chave</h3>
        </div>

        <div class="modal-body">
          <form @submit.prevent="buscarChave">
            <label for="nomeChave">Nome da
Chave:</label>

```



```

        <input type="text" id="nomeChave"
v-model="nomeChave" required>
        <button
type="submit">Buscar</button>
        <p :class="{ 'visible': Message !==
''}" >{{ Message }}!</p>
        </form>
    </div>

    <button class="modal-default-button"
@click="$emit('close')">X</button>
    </div>
</div>
</Transition>
</template>

<style scoped>
</style>

```

2.5. Em RemoverChave.vue coloque:

```

<script setup lang="ts">
import { ref, onMounted, onUpdated, defineProps } from
'vue';
import axios from 'axios';

interface Chave {
    nome: string;
    situacao: string;
    status: boolean;
}

const props = defineProps({
    show: Boolean,
});

const key = ref('');
const listKey = ref<Chave[]>([]);

const fetchData = async () => {
    try {
        const response = await
axios.get('http://localhost:3000/chave');

```

```

        listKey.value = response.data as Chave[];
    } catch (error) {
        console.error('Erro na requisição à API: ', error);
    }
};

onMounted(() => {
    fetchData();
});

onUpdated(() => {
    fetchData();
});

const nomeChave = ref('');
const Message = ref('');

const removerChave = async () => {
    if (key.value.trim() !== '') {
        try {
            Message.value = '';
            const chaveSelecionada =
listKey.value.find((chave) => chave.nome === key.value);

            if (chaveSelecionada) {
                const response = await
axios.patch(`http://localhost:3000/chave/desativar/${chaveSe
lecionada.nome}`);

                // Lide com a resposta da API, por exemplo,
exibindo uma mensagem de sucesso
                console.log('Chave removida com sucesso:',
response.data);

                // Atualizar a lista de chaves após a
remoção

                fetchData();

                // Limpar os campos após a remoção
                key.value = '';
                Message.value = 'Chave removida';
            } else {
                Message.value = 'Chave não encontrada';
            }
        }
    }
};

```

```

    }
    } catch (error) {
        Message.value = 'Erro ao remover a chave';
        console.error('Erro ao remover a chave no
banco', error);
    }
    } else {
        Message.value = 'Selecione uma chave para remover';
    }
};
</script>

<template>
    <link rel="stylesheet" href="src/assets/modal.css">
    <Transition name="modal">
        <div v-if="show" class="modal-mask">
            <div class="modal-container">

                <div class="modal-header">
                    <h3>Remover Chave</h3>
                </div>

                <div class="modal-body">
                    <form @submit.prevent="removerChave">
                        <select v-model="key"
id="selectKey">
                            <option value="" disabled
selected hidden>Selecione uma chave</option>
                            <option v-for="chave in listKey"
:value="chave.nome">{{ chave.nome }}</option>
                        </select>

                        <button
type="submit">Remover</button>

                        <p :class="{ 'visible': Message !==
''}" >{{ Message }}.</p>
                    </form>
                </div>

                <button class="modal-default-button"
@click="$emit('close')">X</button>
            </div>

```

```

        </div>
      </Transition>
    </template>

    <style scoped>
    </style>

```

- 2.6. Para fazer a estilização dos nossos modais vamos criar nosso arquivo modal.css em \src\assets\modal.css, com:

```

.modal-mask {
  position: fixed;
  z-index: 9998;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  background-color: rgba(0, 0, 0, 0.8);
  display: flex;
  transition: opacity 0.3s ease;
}

.modal-container {
  position: relative;
  width: 450px;
  margin: auto;
  padding: 1rem;
  background-color: #fff;
  border-radius: 1rem;
  box-shadow: 0 2px 8px rgba(0, 0, 0, 0.33);
  transition: all 0.3s ease;
}

.modal-header h3 {
  font-size: 1.5rem;
  text-align: center;
  color: #45A452;
  margin-top: 0.2rem;
}

.modal-body {
  margin-top: 2rem;
}

```

```
.modal-default-button {
  position: absolute;
  right: 1rem;
  top: 1rem;
  cursor: pointer;
  background-color: #B05E31;
  padding: 0.5rem 0.8rem;
}

.modal-enter-from {
  opacity: 0;
}

.modal-leave-to {
  opacity: 0;
}

.modal-enter-from .modal-container,
.modal-leave-to .modal-container {
  -webkit-transform: scale(1.1);
  transform: scale(1.1);
}

form {
  padding: 1rem;
}

label {
  display: block;
}

form input {
  border: 1px solid black;
  padding: 0.5rem;
  margin-right: 1rem;
  margin-bottom: 1rem;
  width: 210px;
  height: max-content;
}

form button {
  margin-top: 0.2rem;
}
```

```

font-size: 1rem;
background-color: #42b983;
padding: 0.5rem 1rem;
cursor: pointer;
width: 140px;
}

form select {
border: 1px solid black;
padding: 0.5rem;
margin-right: 1rem;
margin-bottom: 1rem;
width: 210px;
}

p {
visibility: hidden;
font-size: 0.9rem;
color: red;
}

.visible {
visibility: visible;
}

```

- 2.7. Em `\src\components\Header.vue` atualize o código (recomenda-se copiar e colar por cima) com:

```

<script setup lang="ts">
import BuscarChave from './modals/BuscarChave.vue';
import AddKey from './modals/AddKey.vue'
import AlterarChave from './modals/AlterarChave.vue';
import RemoverChave from './modals/RemoverChave.vue';
import { ref } from 'vue'

const title = ref('Where\'s the key?')

const showModalAdd = ref(false)
const showModalFind = ref(false)
const showModalEdit = ref(false)
const showModalDelete = ref(false)

</script>

```

```

<template>
  <header>
    

    <div class="title">
      <h1>{{ title }}</h1>
    </div>

    <nav>
      <div class="dropdown">
        <button class="dropbtn"><span>Chaves</span></button>
        <div class="dropdown-content">
          <a @click="showModalAdd = true"
title="Inserir"><span>Inserir</span></a>
          <a @click="showModalFind = true"
title="Buscar"><span>Buscar</span></a>
          <a @click="showModalEdit = true"
title="Alterar"><span>Alterar</span></a>
          <a @click="showModalDelete = true"
title="Remover"><span>Remover</span></a>
        </div>
      </div>
    </nav>
  </header>

  <Teleport to="body">
    <AddKey :show="showModalAdd" @close="showModalAdd =
false"></AddKey>
    <BuscarChave :show="showModalFind" @close="showModalFind
= false"></BuscarChave>
    <AlterarChave :show="showModalEdit"
@close="showModalEdit = false"></AlterarChave>

```

```

        <RemoverChave :show="showModalDelete"
@close="showModalDelete = false"></RemoverChave>
        </Teleport>
    </template>

<style scoped>
header {
    display: flex;
    padding: 0.5rem 3rem 0.5rem 3rem;
    align-items: center;
    background-color: #50BF84;
}

.title {
    font-size: 2rem;
    width: 80%;
}

h1 {
    margin-top: auto;
    margin-left: 1rem;
}

/* Style The Dropdown Button */
.dropbtn {
    padding: 5px;
    margin-top: auto;
    background-color: transparent;
    display: flex;
    align-items: center;
    width: 9rem;
}

.dropbtn span {
    margin-left: 0.5rem;
    font-size: 1rem;
}

/* The container <div> - needed to position the dropdown
content */
.dropdown {
    position: relative;

```



```
display: inline-block;
}

/* Dropdown Content (Hidden by Default) */
.dropdown-content {
  display: none;
  position: absolute;
  background-color: white;
  box-shadow: 0px 8px 16px 0px rgba(0, 0, 0, 0.2);
  z-index: 1;
}

/* Links inside the dropdown */
.dropdown-content a {
  position: relative;
  color: black;
  padding: 10px;
  text-decoration: none;
  display: block;
  width: 9rem;
}

.dropdown-content a img {
  width: 40px;
  height: 40px;
}

.dropdown-content a span {
  position: absolute;
  top: 1.3rem;
  left: 4rem;
  font-size: 1rem;
}

/* Change color of dropdown links on hover */
.dropdown-content a:hover {
  background-color: #B3DE5D;
  cursor: pointer;
}

/* Show the dropdown menu on hover */
.dropdown:hover .dropdown-content {
  display: block;
}
```

```

}

/* Change the background color of the dropdown button when
the dropdown content is shown */
.dropdown:hover .dropbtn {
  background-color: #B3DE5D;
}</style>

```

- 2.8. Em `\src\components\Main.vue` atualize o código (recomenda-se copiar e colar por cima) com:

```

<script setup lang="ts">
import BuscarChave from './modals/BuscarChave.vue';
import { ref, onMounted } from 'vue';
import axios from 'axios';

interface Chave {
  nome: string;
  situacao: string;
  status: boolean;
}

const listKey = ref<Chave[]>([]);
const key = ref<string>('');

const fetchData = async () => {
  try {
    const response = await
axios.get('http://localhost:3000/chave');
    listKey.value = response.data as Chave[];
  } catch (error) {
    console.error('Erro na requisição à API: ', error);
  }
};

onMounted(() => {
  fetchData();
});

const chaveRetornada = (chave: Chave) => {
  key.value = chave.nome;
};
</script>

```

```

<template>
  <main>
    <BuscarChave
@chave-encontrada="chaveRetornada"></BuscarChave>

    <select v-model="key" id="selectKey">
      <option value="" disabled hidden>Selecione uma
chave</option>
      <option v-for="chave in listKey" :value="chave">
        {{ chave.nome }}
      </option>
    </select>
  </main>
</template>

<style scoped>
main {
  display: flex;
  padding: 4rem;
  align-items: center;
  justify-content: space-around;
}

select {
  cursor: pointer;
  appearance: none;
  padding: 1rem 4rem;
  border-radius: 2rem;
  text-align: center;
  font-size: 18px;
  background-color: #50BF84;
}

select:hover {
  background-color: #B3DE5D;
}

</style>

```