Projeto Nest.js e Vue.js

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

1. Configurando o backend para fornecer o serviço responsável por buscar as chaves no BD.

Abra o projeto backend no VSCode. Em seguida:

1.1. Primeiro precisamos adicionar um módulo ao projeto que será essencial para fazer a conexão com o mongoDB. No terminal, no diretório backend/ execute o comando:

→ nest i --save @nestjs/mongoose

1.2. Abra o arquivo backend/src/chave/service/chave/**chave.service.ts** e insira as seguintes linhas de código:

```
import { Injectable } 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().exec();
   }
}
```

1.3. Abra o arquivo backend/src/chave/service/chave/**chave.controller.ts** e insira as seguintes linhas de código:

```
import { Controller, Get } 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();
   }
}
```

1.4. Abra o arquivo backend/src/chave/service/chave/**chave.module.ts** e insira as seguintes linhas de código:

```
import { Module } from '@nestjs/common';
import { ChaveController } from './chave.controller';
import { ChaveService } from './chave.service';

@Module({
 controllers: [ChaveController],
 providers: [ChaveService]
})

export class ChaveModule {}
```

1.5. Abra o arquivo backend/src/**app.module.ts** e insira as seguintes linhas de código:

```
import { Module } from '@nestjs/common';
import { AppController } from './app.controller';
import { AppService } from './app.service';
import { ChaveModule } from './chave/chave.module';
import { EmprestimoModule } from
'./emprestimo/emprestimo.module';
import { ServidorModule } from './servidor/servidor.module';
import { ChaveController } from './chave/chave.controller';
import { ChaveService } from './chave/chave.service';
import { MongooseModule } from '@nestjs/mongoose';
import { ChaveSchema } from './chave/model/chave.model';
import { EmprestimoSchema } from
'./emprestimo/model/emprestimo.model';
```

```
import { ServidorSchema } from
'./servidor/model/servidor.model';

@Module({
 imports: [
   //ChaveModule, ServidorModule, EmprestimoModule,
   MongooseModule.forRoot('mongodb://localhost:27017'),
   MongooseModule.forFeature([{name: 'Chave', schema :
ChaveSchema}]),
   MongooseModule.forFeature([{ name: 'Emprestimo', schema:
EmprestimoSchema }]),
   MongooseModule.forFeature([{ name: 'Servidor', schema:
ServidorSchema }]),
 ],
 controllers: [AppController, ChaveController],
 providers: [AppService, ChaveService],
})
export class AppModule {}
```

1.6. Abra o arquivo backend/src/**main.ts** e insira as seguintes linhas de código:

```
import { NestFactory } from '@nestjs/core';
import { AppModule } from './app.module';

async function bootstrap() {
 const app = await NestFactory.create(AppModule);

 // Configurando o CORS
 app.enableCors({
   origin: 'http://localhost:5173'
 })

 await app.listen(3000);
}
bootstrap();
```

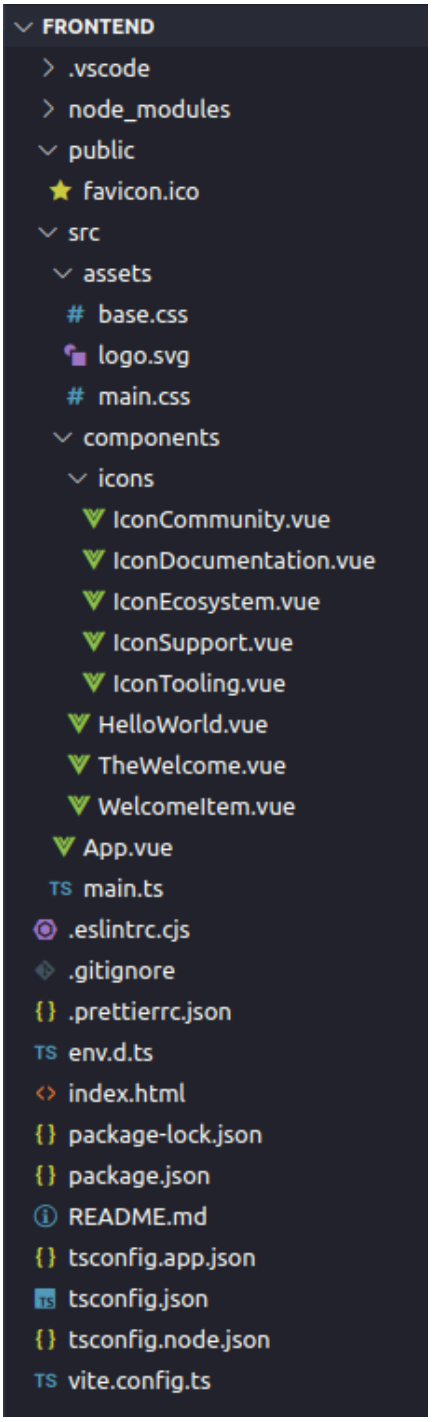
Agora, ao inicializar nosso servidor referente ao backend (**npm run start**) já conseguimos visualizar todas as chaves inseridas no banco através da rota **http://localhost:3000/chave**.
OBS: Certifique-se de que a instância do mongo esteja em execução via docker.

→ sudo docker start mongo

2. Configurando o frontend para fazer a requisição via URL ao nosso backend para obter as chaves inseridas no BD.

Abra o projeto frontend no VSCode.

2.1. Segue abaixo a estrutura inicial do nosso projeto frontend.



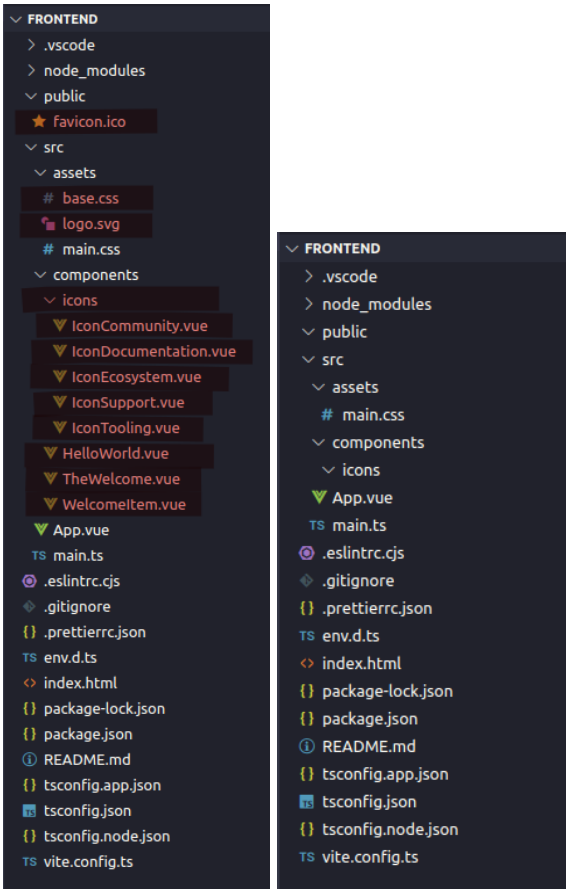
2.2. Um projeto Vue é construído a partir de componentes. Na construção dos nossos componentes utilizaremos uma estrutura de arquivo chamado Single-File Component (também conhecido como arquivos .vue, abreviados como SFC). Um Vue SFC, como o nome sugere, encapsula a lógica do componente (JavaScript), modelo (HTML) e estilos (CSS) em um único arquivo. Aqui temos um exemplo:

```
<script setup>
import { ref } from 'vue'
const count = ref(0)
</script>

<template>
  <button @click="count++">Count is: {{ count }}</button>
</template>

<style scoped>
button {
  font-weight: bold;
}
</style>
```

2.3. Para iniciarmos na construção do design do nosso frontend, faremos alguns ajustes nos arquivos iniciais do projeto. Primeiro vamos excluir alguns componentes que já foram criados por padrão. Os arquivos marcados em vermelho podem ser excluídos:



2.4. Faremos pequenas alterações nos arquivos **index.html**, **main.css** e **App.vue**.

2.4.1. index.html

Em <head></head> adicione:

```
 <link rel="preconnect"
href="https://fonts.googleapis.com">
 <link rel="preconnect"
href="https://fonts.gstatic.com" crossorigin>
 <link
href="https://fonts.googleapis.com/css2?family=Chivo:w
ght@300&display=swap" rel="stylesheet">
```

Substituir o conteúdo da tag **<title></title>** para : **Where's the key?**
Substituir a linha da tag **<link>** por:
<link rel="icon" href="./public/favicon.ico">

2.4.2. main.css

Substituir todo o conteúdo do arquivo pelo seguinte trecho de código:

```
* {
 padding: 0;
 margin: 0;
 list-style: none;
 border: 0;
 box-sizing: border-box;
 font-weight: normal;
 font-family: 'Chivo', sans-serif;
}

#app {
 display: flex;
 flex-direction: column;
}
```

2.4.3. App.vue

```
<script setup lang="ts">

</script>

<template>
```

```
</template>

<style scoped></style>
```

2.5. Feito essas alterações, adicionaremos alguns arquivos ao projeto.

2.5.1. logo.png

Faça o download do arquivo:
https://github.com/luiz-felippelb/vuejs-emprestimoChave-daw1/blob/main/src/assets/logo.png

Adicione o arquivo no diretório:
/emprestimo-chave/frontend/src/assets/

2.5.2. favicon.ico

Faça o download do arquivo:
https://github.com/luiz-felippelb/vuejs-emprestimoChave-daw1/blob/main/public/favicon.ico

Adicione o arquivo no diretório: /emprestimo-chave/frontend/public/

2.6. Agora, vamos fazer uma prévia construção do cabeçalho da nossa página.

2.6.1. No diretório:
/emprestimo-chave/frontend/src/components/

→ touch Header.vue

Adicione as seguintes linhas de código ao arquivo **Header.vue:**

```
<script setup lang="ts">
import { ref } from 'vue'

const title = ref('Where\'s the key?')
</script>

<template>
 <header>
   <img class="logo" src="../assets/logo.png" alt="Key
logo" width="150" height="150" />

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

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

.title {
 font-size: 1.5rem;
 width: 80%;
}

h1 {
 padding-top: 1em;
}
</style>
```

Agora, para que nosso componente seja renderizado quando executarmos nosso projeto frontend, vamos importá-lo e utilizá-lo no arquivo **App.vue**. Que deve ficar dessa forma:

```
<script setup lang="ts">
import Header from './components/Header.vue';
</script>

<template>
   <Header></Header>
</template>

<style scoped></style>
```

2.7. Agora, de fato, iniciaremos o desenvolvimento do frontend da nossa primeira funcionalidade do projeto, que é fazer uma requisição via URL ao servidor do backend para obter as chaves inseridas no BD.

2.7.1. No diretório:
/emprestimo-chave/frontend/src/components/

→ touch Main.vue

Adicione as seguintes linhas de código ao arquivo **Main.vue:**

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

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

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

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()
})
</script>

<template>
 <main>
   <select v-model="key" id="selectKey">
     <option value="" disabled selected 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;
}
</style>
```

Neste componente definimos a função **fetchData** que faz uma solicitação http get para http://localhost:3000/chave utilizando axios.

E quando nosso componente é montado (onMounted()), nosso método é ativado aguardando a requisição ao nosso servidor backend.

Para que o componente seja renderizado, adicione-o em **App.vue**.

```
<script setup lang="ts">
import Main from './components/Main.vue'
import Header from './components/Header.vue'
</script>

<template>
 <Header></Header>

 <Main></Main>
</template>

<style scoped></style>
```

Para testar:

→ npm run dev

Lembrando que o servidor backend deve estar em execução juntamente com a nossa instância do mongo via Docker.