Tutorial Listando Chaves Disponiveis

Projeto: Controle de empréstimo de chaves

Tecnologias: Spring Boot Rest e Svelte

Alunos:
Gabriela Silva Rodrigues;
Gustavo Machado Pontes;
Mateus Alves Silva;
Nathan Rodrigues dos Santos.

Iniciar o container

- Abra o terminal e digite o seguinte comando para iniciar o container com o mongo:

  sudo docker start mongo

Configuração de acesso ao banco e modificação da porta do server

- Abra o projeto no eclipse e procure o arquivo application.properties localizados na pasta src/main/resources/application.properties e digite:

```
spring.data.mongodb.host=localhost
spring.data.mongodb.port=27017
spring.data.mongodb.database=emprestimo_chaves
server.port=8081
```

- Essas configurações configuram o host, porta e a database do Banco de dados MongoDB a última linha modifica a porta do servidor Tomcat que está embutido no SpringBoot para que a backend e frontend não utilizem a mesma porta;

Criação do DTO, Repository e Controller

- Crie os seguinte arquivos nos packages fique atentos aos tipos de arquivos que são pedidos para selecionar corretamente no menu;

- Crie os seguinte arquivos dentro dos packages:
  - Um Record dentro da pasta dto com o nome ChaveRecordoDto para armazenar temporariamente os dados com o seguinte conteudo:

```
package com.GGMN.backend.dto;
public record ChaveRecordDto(String nome, String situacao,  boolean
status) {

        }
```

  - Uma interface no package repository com o nome ChaveRepository que herda os métodos do MongoRepositoy que contem os comandos de persistência e busca no mongodb, digite o seguinte conteúdo:

```
package com.GGMN.backend.repository;

import java.util.List;

import org.springframework.data.mongodb.repository.MongoRepository;
import org.springframework.stereotype.Repository;

import com.GGMN.backend.model.Chave;

@Repository
public interface ChaveRepository extends MongoRepository<Chave, String>{
    List<Chave> findByStatus(boolean status);
    List<Chave> findBySituacao(String situacao);
}
```

- Uma classe com o nome de ChaveController com o seguinte conteúdo:

```
package com.GGMN.backend.controller;

import java.util.ArrayList;
import java.util.List;

import org.springframework.beans.BeanUtils;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.HttpStatusCode;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Controller;
import org.springframework.stereotype.Service;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.GGMN.backend.dto.ChaveRecordDto;
import com.GGMN.backend.model.Chave;
import com.GGMN.backend.repository.ChaveRepository;

@RestController
@RequestMapping("/chaves")
@CrossOrigin(origins = "http://localhost:8080")
public class ChaveController {

    @Autowired
    private ChaveRepository chaveRepository;

    @PostMapping
```

```
    public ResponseEntity<Chave> saveChave(@RequestBody ChaveRecordDto
chaveRecordDto) {

            Chave chave = new Chave();
            BeanUtils.copyProperties(chaveRecordDto, chave);


            return
ResponseEntity.status(HttpStatus.CREATED).body(chaveRepository.save(chave));
    }

    @GetMapping
    public ResponseEntity<List<Chave>> getAllChaves() {

            return
ResponseEntity.status(HttpStatus.OK).body(chaveRepository.findAll());
    }

    @GetMapping("/status/{status}")
    public ResponseEntity<List<Chave>> getPerStauts(@PathVariable(value="status")
boolean status) {


            return
ResponseEntity.status(HttpStatus.OK).body(chaveRepository.findByStatus(status));
    }

    @GetMapping("/situacao/{situacao}")
    public ResponseEntity<List<Chave>>
getPerSituacao(@PathVariable(value="situacao") String situacao) {

            List<Chave> chaves = new ArrayList<>();

            for (Chave chave : chaveRepository.findBySituacao(situacao)) {
                    if(chave.getStatus() == true) {
                            chaves.add(chave);
                    }
            }



            return ResponseEntity.status(HttpStatus.OK).body(chaves);
    }
}
```

- Esse arquivo contem o tratamento de cada chamado da API fazendo a ligação do frontend com o banco de dados.

Criação do DTO, Repository e Controller

- Abra o projeto frontend no vscode e navegue até a pasta src abra o arquivo App.Svelte e copie o seuinte código:

```
<script>
//Importando as páginas como se fossem componentes
import InserirPage from "./pages/InserirPage.svelte";
import ListarPage from "./pages/ListarPage.svelte";
//Selecionando a página inicial
let page = 'InserirPage';
function navigate(to) {
page = to;
}
</script>
<body>
<h1>IFTM - Empréstimo de Chaves</h1>
<!--Criando o menu-->
<nav>
<a on:click={() => navigate('InserirPage')}>Inserir Chave</a>
<a on:click={() => navigate('ListarPage')}>Lista de Chaves</a>
</nav>
{#if page === 'InserirPage'}
<InserirPage />
{:else if page === 'ListarPage'}
<ListarPage />
{/if}
</body>

<style>
*{
border: 0;
margin: 0;
padding: 0;
}
body {
text-align: center;
justify-content: center;
display: block;
background-color: rgb(34, 40, 49);
color: rgb(238, 238, 238);
overflow: auto;
}

h1{
margin-bottom: 20px;
}

a{
color:rgb(214, 90, 49);
text-decoration: none;
border: 3px solid rgb(214, 90, 49) ;
padding: 15px;
```

```
}

a:hover{
background-color: rgb(214, 90, 49);
color: white;
cursor: pointer;
}
</style>
```

- Crie uma pasta camada pages e dentro dela um arquivo com o nome ListarPage.svelte e copie o seguinte código:

```
<script>
let chave = { nome: "", situacao: "disponivel", status: true };

//Lista para mostrar as chaves
let Listachaves = [];

async function carregarChaves() {
try {
const response = await fetch(
"http://localhost:8081/chaves/situacao/disponivel"
); // Adicione "http://" ao URL
if (response.ok) {
const chaves = await response.json();
Listachaves = chaves;
console.log(chaves); // Mude para "chaves" em vez de "Listachaves"
} else {
console.error(
"Erro ao carregar as chaves:",
response.statusText
);
}
} catch (error) {
console.error("Erro ao carregar as chaves:", error);
}
}

carregarChaves();

</script>
<main>
<h2>Lista de Chaves Disponíveis:</h2>
<ul>
{#each Listachaves as chave}
{#if chave.situacao == "disponivel"}
<li>🗝{chave.nome} - Situação: {chave.situacao} ✔</li>
{:else}
<li>🗝{chave.nome} - Situação: {chave.situacao} ⛔</li>
{/if}
{/each}
```

```
</ul>
</main>
<style>
main {
text-align: center;
justify-content: center;
display: block;
background-color: rgb(34, 40, 49);
color: rgb(238, 238, 238);
overflow: auto;
}

h2 {
font-size: 2.5em;
font-family: "Courier New", Courier, monospace;
color: rgb(214, 90, 49);
font-weight: bolder;
}

button {
background-color: rgb(214, 90, 49);
color: rgb(238, 238, 238);
padding: 1vh 2vh;
border-radius: 5%;
cursor: pointer;
margin: 1vh;
transition: background-color 0.3s;
font-size: 1.5em;
}

button:hover {
background-color: rgb(57, 62, 70);
}

label {
margin-bottom: 5vh;
font-family: Arial, Helvetica, sans-serif;
font-weight: bold;
font-size: 1.3em;
}

input {
padding: 1vh;
margin: 3vh;
border: 5px solid rgb(93, 104, 122);
border-radius: 4px;
}
li {
list-style: none;
margin: 2vh;
font-size: 2em;
```

```
}
</style>
```