Tutorial Completando CRUD Chaves

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.

Adição dos métodos de update e delete na API

- Acesse a classe com o nome de ChaveController com o seguinte conteúdo, para melhor entendimento recomendo ler os comentários:

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

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

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.DeleteMapping;
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.PutMapping;
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;

	//Persiste no banco
	@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));
	}


	//Busca todas a chaves sem filtro
	@GetMapping
	public ResponseEntity<List<Chave>> getAllChaves() {

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


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


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

	//Filtro Situação
	@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);
	}
```

```
	//Filtro Nome
	@GetMapping("/nome/{nome}")
	public ResponseEntity<List<Chave>> getPerNome(@PathVariable(value="nome") String
nome) {

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

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

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

	//Alteração
	@PutMapping("/{id}")
	public ResponseEntity<Chave> updateChave(@PathVariable(value="id") String id,
@RequestBody ChaveRecordDto chaveRecordDto) {

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

		chave.setId(id);

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

	}

	//"Deleção"
	@DeleteMapping("/{id}")
	public ResponseEntity<Chave> deleteChave(@PathVariable(value="id") String id,
@RequestBody ChaveRecordDto chaveRecordDto) {

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

		chave.setStatus(false);
		chave.setId(id);

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

	}
}
```

iniciando o servidor do SpringBoot

- clique o ícone de play no canto superior direito do Visual Studio Code.

Modificando a tela para listar chaves

- Abra o projeto front-end
- Entre na pasta pages
- Abra o arquivo ListarPage.svelte
- Copie o seguinte código, para melhor entendimento recomendo ler os comentários:

```
<!-- ListarPage.svelte -->
<script>
    let chave = { nome: "", situacao: "disponivel", status: true };
    let filtroNome = ""; // para a busca pelo nome
    let Listachaves = [];

    async function carregarChaves() {
        // Implementação de busca no backend com filtro de nome
        let url = "http://localhost:8081/chaves";

        try {
            const response = await fetch(url);
            if (response.ok) {
                Listachaves = await response.json();
            } else {
                console.error(
                    "Erro ao carregar as chaves:",
                    response.statusText,
                );
            }
        } catch (error) {
            console.error("Erro ao carregar as chaves:", error);
        }
    }

     // Implementação de busca no backend com filtro de situação
    async function carregarChavesDisponiveis() {
        let url = "http://localhost:8081/chaves/situacao/disponivel";

        try {
            const response = await fetch(url);
            if (response.ok) {
                Listachaves = await response.json();
            } else {
                console.error(
                    "Erro ao carregar as chaves:",
                    response.statusText,
                );
            }
        } catch (error) {
```

```
        console.error("Erro ao carregar as chaves:", error);
    }
}

// Função atualizada para carregar chaves baseada no filtro de nome
async function carregarChavesNome() {
    // Implementação de busca no backend com filtro de nome
    let url = "http://localhost:8081/chaves";
    if (filtroNome.trim()) {
        url += `/nome/${encodeURIComponent(filtroNome.trim())}`;
    }

    try {
        const response = await fetch(url);
        if (response.ok) {
            Listachaves = await response.json();
        } else {
            console.error(
                "Erro ao carregar as chaves:",
                response.statusText,
            );
        }
    } catch (error) {
        console.error("Erro ao carregar as chaves:", error);
    }
}

// Função para alterar a situação da chave
async function alterarSituacaoChave(chave) {
    try {
        const response = await fetch(
            `http://localhost:8081/chaves/${chave.id}`,
            {
                method: "PUT",
                headers: { "Content-Type": "application/json" },
                body: JSON.stringify({
                    ...chave,
                    situacao:
                        chave.situacao === "disponivel"
                            ? "indisponivel"
                            : "disponivel",
                }),
            },
        );

        if (response.ok) {
            carregarChaves(); // Recarrega a lista de chaves
        } else {
            console.error(
                "Erro ao alterar a situação da chave:",
                response.statusText,
            );
```

```
        }
    } catch (error) {
        console.error("Erro ao alterar a situação da chave:", error);
    }
}

// Função para reativar a chave
async function reativarChave(chave) {
    try {
        const response = await fetch(
            `http://localhost:8081/chaves/${chave.id}`,
            {
                method: "PUT",
                headers: { "Content-Type": "application/json" },
                body: JSON.stringify({
                    ...chave,
                    status:
                        chave.status === false
                            ? true
                            : false,
                }),
            },
        );

        if (response.ok) {
            carregarChaves(); // Recarrega a lista de chaves
        } else {
            console.error(
                "Erro ao alterar a situação da chave:",
                response.statusText,
            );
        }
    } catch (error) {
        console.error("Erro ao alterar a situação da chave:", error);
    }
}

// Função para desativar (remover) uma chave
async function desativarChave(chave) {
    try {
        const response = await fetch(
            `http://localhost:8081/chaves/${chave.id}`,
            {
                method: "DELETE",
                headers: { "Content-Type": "application/json" },
                body: JSON.stringify(chave),
            },
        );

        if (response.ok) {
            carregarChaves(); // Recarrega a lista de chaves
        } else {
```

```
            console.error(
              "Erro ao desativar a chave:",
              response.statusText,
            );
          }
        } catch (error) {
          console.error("Erro ao desativar a chave:", error);
        }
    }

    carregarChavesDisponiveis()
</script>

<main>
    <h2>Lista de Chaves Disponíveis:</h2>
    <br>
    <button on:click={carregarChavesDisponiveis}>Disponíveis</button>
    <button on:click={carregarChaves}>Todas</button>
    <br>
    <input
      type="text"
      bind:value={filtroNome}
      placeholder="Digite o nome da chave"
      onchange="carregarChavesNome"
    />
    <button on:click={carregarChavesNome}>Buscar</button>

    <ul>
      {#each Listachaves as chave}
        <li>
          🗝 {chave.nome} - Situação: {chave.situacao}

          {#if chave.status === true}
            {#if chave.situacao === "disponivel"}
              ✅
              <button on:click={() => alterarSituacaoChave(chave)}
                >Alterar</button
              >
              <button on:click={() => desativarChave(chave)}
                >Desativar</button
              >
            {:else}
              ⛔
              <button on:click={() => alterarSituacaoChave(chave)}
                >Alterar</button
              >
              <button on:click={() => desativarChave(chave)}
                >Desativar</button
              >
            {/if}
          {:else}
            ⭕
```

```
                <button on:click={() => reativarChave(chave)}
                    >Reativar</button
                >
              {/if}
            </li>
        {/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);
    }

    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>

iniciando o projeto Svelte

- Abra o terminal vá até a pasta frontend utilizando o cd e digite:
  npm run dev