**README.md**

**Projeto de Controle de Estoque de Carros com Spring Boot**

**Descrição do Projeto**

Este é um sistema de controle de estoque de carros, desenvolvido usando Spring Boot. Ele inclui funcionalidades básicas, como adição, listagem e atualização de carros, além do gerenciamento de fotos associadas a cada veículo.

**Requisitos do Sistema**

- JDK 8 ou superior

- Maven

- Banco de dados MySQL

- Node.js (para o front-end, se aplicável)

**Configuração do Banco de Dados**

1. Crie um banco de dados MySQL chamado `estoque\_carros`.

2. Configure as propriedades do banco de dados no arquivo `src/main/resources/application.properties`.

**properties**

spring.datasource.url=jdbc:mysql://localhost:3306/estoque\_carros

spring.datasource.username=seu\_usuario

spring.datasource.password=sua\_senha

**Executando a Aplicação**

1. Clone o repositório.

```bash

git clone https://github.com/seu-usuario/estoque-carros-spring-boot.git

```

2. Navegue até o diretório do projeto.

```bash

cd estoque-carros-spring-boot

```

3. Execute o aplicativo Spring Boot.

```bash

./mvnw spring-boot:run

```

A aplicação estará disponível em [http://localhost:8080](http://localhost:8080).

**Endpoints da API**

- \*\*Listar Carros:\*\*

- GET /api/carros

- \*\*Adicionar Carro:\*\*

- POST /api/carros

- Corpo da requisição:

```json

{

"modelo": "Nome do Modelo",

"marca": "Nome da Marca"

// Outros campos, se aplicável

}

```

- \*\*Adicionar Foto:\*\*

- POST /api/fotos

- Corpo da requisição:

```json

{

"nomeArquivo": "nome\_da\_foto.jpg"

// Outros campos, se aplicável

}

```

**Desenvolvimento Adicional**

Para o desenvolvimento front-end, você pode integrar uma tecnologia como React, Angular ou Vue.js. Certifique-se de configurar corretamente os recursos necessários no diretório `src/main/resources/static` ou `src/main/resources/templates` para integração com o back-end.

refletir as tecnologias específicas que você está usando, e inclua qualquer outra informação relevante para os desenvolvedores que possam contribuir para o projeto.

- Para a segurança, considere a implementação de autenticação e autorização usando Spring Security.

**Contribuições**

Contribuições são bem-vindas! Sinta-se à vontade para enviar pull requests ou relatar problemas.

**Licença**

Este projeto é licenciado sob a [Licença MIT](LICENSE).

---

Roteiro

1. **Modelagem do Banco de Dados:**
   * Defina as entidades necessárias, como **Carro**, **Fotos**, etc.
   * Relacione as entidades conforme necessário.
2. **Configuração do Spring Boot:**
   * Configure o Spring Boot em seu projeto.
   * Configure o acesso ao banco de dados.
3. **Desenvolvimento da Camada de Repositório:**
   * Crie interfaces de repositório para cada entidade usando Spring Data JPA.
   * Defina consultas personalizadas para operações de entrada e saída no estoque.
4. **Desenvolvimento da Camada de Serviço:**
   * Crie serviços para manipular as operações de estoque.
   * Implemente lógica de negócios para adicionar fotos, informações, etc.
5. **Desenvolvimento da Camada de Controlador:**
   * Crie controladores para manipular solicitações da interface do usuário.
   * Use anotações **@RestController** e **@RequestMapping** para mapear URLs para métodos de controle.
6. **Desenvolvimento da Interface do Usuário:**
   * Use uma tecnologia front-end (como React, Angular, ou Thymeleaf) para criar páginas de gerenciamento de estoque.
   * Integre com o backend usando chamadas de API REST.
7. **Tratamento de Imagens:**
   * Implemente um serviço para lidar com o upload e armazenamento de fotos.
   * Considere o uso de um serviço de armazenamento em nuvem para armazenar as imagens.
8. **Atualização Dinâmica da Tela:**
   * Utilize WebSocket para permitir a comunicação em tempo real entre o servidor e o cliente.
   * Atualize dinamicamente os componentes da tela quando houver alterações no estoque.
9. **Segurança:**
   * Implemente autenticação e autorização para proteger as operações sensíveis.
10. **Testes:**
    * Desenvolva testes unitários e de integração para garantir a robustez do sistema.
11. **Documentação:**
    * Documente seu código e API para facilitar a manutenção.
12. CRIAR AS TABELAS para inserção, CRUD

@Entity

public class Carro {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String modelo;

private String marca;

// Outros campos e relacionamentos

// getters e setters

}

@Entity

public class Foto {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String nomeArquivo;

// Outros campos e relacionamentos

// getters e setters

}

Repository

public interface CarroRepository extends JpaRepository<Carro, Long> {

// Consultas personalizadas se necessário

}

public interface FotoRepository extends JpaRepository<Foto, Long> {

// Consultas personalizadas se necessário

}

Serviços

@Service

public class CarroService {

@Autowired

private CarroRepository carroRepository;

public List<Carro> listarCarros() {

return carroRepository.findAll();

}

public Carro adicionarCarro(Carro carro) {

// Lógica de negócios para adicionar um carro

return carroRepository.save(carro);

}

// Outros métodos para atualizar, excluir, etc.

}

@Service

public class FotoService {

@Autowired

private FotoRepository fotoRepository;

public Foto adicionarFoto(Foto foto) {

// Lógica de negócios para adicionar uma foto

return fotoRepository.save(foto);

}

// Outros métodos para recuperar, excluir, etc.

}

@RestController

@RequestMapping("/api/carros")

public class CarroController {

@Autowired

private CarroService carroService;

@GetMapping

public List<Carro> listarCarros() {

return carroService.listarCarros();

}

@PostMapping

public Carro adicionarCarro(@RequestBody Carro carro) {

return carroService.adicionarCarro(carro);

}

// Outros métodos para atualizar, excluir, etc.

}

@RestController

@RequestMapping("/api/fotos")

public class FotoController {

@Autowired

private FotoService fotoService;

@PostMapping

public Foto adicionarFoto(@RequestBody Foto foto) {

return fotoService.adicionarFoto(foto);

}

// Outros métodos para recuperar, excluir, etc.

}