Documentação de Software - Sistema de Locadora de Automóveis

Autores

Bruno Gomes

Danielle Sanches

Fábio Macedo

Yasmin

Higor

2024

Visão Geral

Este documento descreve a implementação de um sistema de locadora de automóveis utilizando linguagem JAVA e framework Spring. O sistema permite que clientes potenciais e cadastrados realizem ações como cadastro de cliente, escolha de veículo para aluguel, e efetivação do aluguel de veículos. O projeto é desenvolvido com foco em modularidade, segurança e usabilidade.

Tecnologias Utilizadas

- Linguagem: Java

- Hibernate
- Framework: Spring Boot
- Banco de Dados: MySQL
- Controle de Versão: Git
- Maven

Estrutura do Projeto

O projeto é dividido nas seguintes camadas:

- Controller

  Responsável por receber as requisições HTTP e retornar as respostas adequadas, através de chamadas entre as classes responsáveis pela lógica de negócio e acesso ao banco.
  Criamos os seguintes controllers: AcessorioController, AluguelController, CarroController e MotoristaController

- Repositories

  Interfaces para acesso ao banco de dados, usando Spring Data JPA.
  Neste projeto foram criadas as seguintes interfaces de acesso: AcessorioRepository, AluguelRepository, ApoliceRepository, CarroRepository, ModeloCarroRepository, MotoristaRepository

- Entities

  Representa as entidades do sistema que são persistidas no banco de dados.
  Foram divididas nos seguintes pacotes: Aluguel, Carro, Pessoa

- DTO (Data Transfer Object):

DTO Objetos que carregam dados entre as camadas do sistema, para que se exiba ao usuário apenas aqueles dados tidos como relevantes naquele momento.
Para este projeto foram vistos como necessários as seguintes classes DTO: AcessorioDTO, AluguelDTO, CarroDisponivelDTO, CarroDTO, CategoriaDTO

- Response:

  Pacote criado para que abrigasse as classes de validações, a fim de que houvesse um melhor tratamento a determinados erros mais comuns.

  Foram Criadas as seguintes classes neste pacote: ErrorResponse, ResourceNotFound, ValidationException.

- Migrations:

  Após o mapeamento pelo Hibernate das entidades para o banco, as migrations tiveram um papel importantíssimo neste projeto. O seu uso se deu para inserção de dados no banco, o que serviu para que nenhum dos colaboradores necessitasse ter de acessar o banco de dados para inserir dados, serviço feito pelas migrations quando se roda o programa.

Casos de Uso

1. Cadastro de Cliente

Descrição: Permite que um cliente em potencial se cadastre na locadora de automóveis para acessar os serviços de aluguel.
- Controller: ClienteController
- Model: Cliente
- Service: ClienteService
- Repository: ClienteRepository

Fluxo:

1. O cliente acessa a página inicial e preenche o formulário de cadastro.
2. O sistema valida as informações, incluindo a verificação de duplicidade de e-mail.
3. Após validação, os dados são salvos no banco de dados.
4. O cliente recebe uma confirmação na tela e é redirecionado para a página inicial.
5. Ao cliente é dada a opção de buscar um cliente (motorista) pelo seu ID, ou mesmo listar todos os clientes cadastrados.
6. Também é possível fazer a busca do email de um determinado motorista

Critérios de Aceitação:
- O formulário deve conter os campos: nome completo, data de nascimento, CPF, e número da CNH.
- Após o cadastro bem-sucedido, deve ser exibida uma mensagem de confirmação.
- O sistema deve evitar registros duplicados com base no e-mail.

2. Escolha de Veículo para Aluguel

Descrição: Permite que um cliente cadastrado selecione um veículo para alugar.
- Controller: CarroController
- Model: Carro
- Service: VeiculoService
- Repository: CarroRepository
- View: selecaoVeiculo.html, detalhesVeiculo.html

Fluxo:

1. O cliente acessa a seção de seleção de veículos na página inicial.
2. O cliente por listar todos os veículos, como também buscar um veículo específico pelo seu id.
3. O cliente também tem a opção de buscar veículos por determinada categoria, ou por algum acessório que tenha preferência.
4. Uma lista de veículos disponíveis é exibida, com a opção de aplicar filtros.
5. Ao clicar em um veículo, o cliente é redirecionado para a página de detalhes.
6. O cliente seleciona o período de aluguel e adiciona o veículo ao carrinho.
7. O carrinho exibe um resumo dos veículos selecionados e o custo estimado.
8. O cliente pode revisar o carrinho e confirmar a reserva.

Critérios de Aceitação:
- A página inicial deve exibir uma lista de veículos disponíveis com informações detalhadas.
- Filtros para categoria e acessórios devem estar disponíveis.
- O cliente deve poder visualizar os detalhes do veículo e selecionar as datas de aluguel.

3. Efetivação do Aluguel de Veículo

Descrição: Permite que o cliente confirme e efetive a reserva de um veículo selecionado.
- Controller: AluguelController
- Model: Aluguel
- Service: AluguelService
- Repository: AluguelRepository
- View: resumoAluguel.html, confirmacaoAluguel.html

Fluxo:

1. Após revisar o carrinho, o cliente confirma a reserva.

2. O cliente é redirecionado para uma página de resumo, onde revisa todos os detalhes do aluguel.
3. O cliente concorda com os termos e condições e escolhe o método de pagamento.
4. O pagamento é processado (simulado) e a reserva é confirmada.
5. O veículo é marcado como reservado no sistema e as datas são bloqueadas no calendário.
6. O cliente recebe uma confirmação na tela com os detalhes do aluguel.

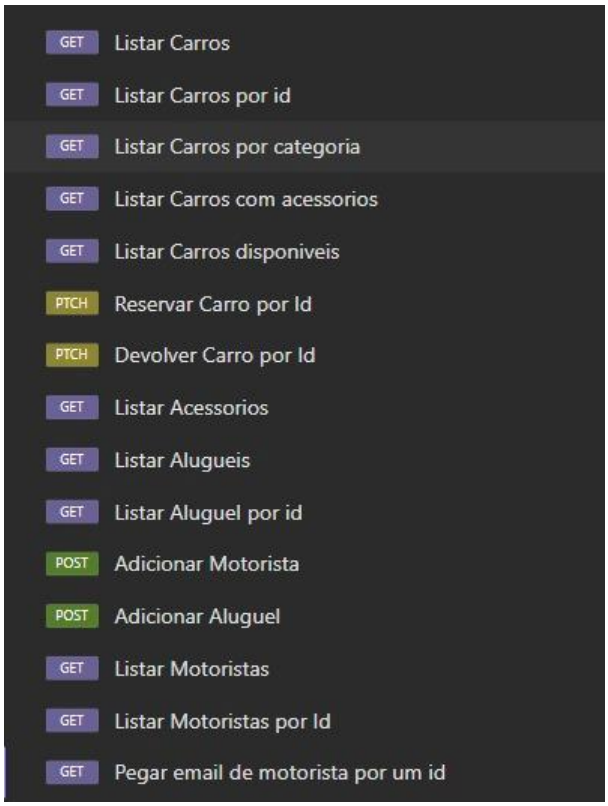
Critérios de Aceitação:
- O cliente deve revisar e concordar com os termos de aluguel antes de efetivar o pagamento.
- O sistema deve processar o pagamento e confirmar a reserva.
- O veículo deve ser marcado como reservado e indisponível para outras datas.

Considerações Finais

Este sistema foi desenvolvido para ser escalável e modular, facilitando futuras manutenções e extensões. A documentação deve ser atualizada conforme novas funcionalidades sejam adicionadas ou modificadas.

Acrescentar as validações:

Criar os links de mapeamento do insomnia/postman e nome dos métodos, para quem vai usar



Carros

Listar todos os Carros: http://localhost:8080/carros

Listar Carros por Id: http://localhost:8080/carros/{id}

Listar Carros por categoria: http://localhost:8080/carros/categoria/{NOME_CATEGORIA}

Listar Carros por acessórios: http://localhost:8080/carros/acessoriosid/{id}

Listar Carros disponíveis: http://localhost:8080/carros/disponiveis

Reservar Carro por Id: http://localhost:8080/carros/reservar-carro/{id}

Devolver Carro por Id: http://localhost:8080/carros/devolver-carro/{id}

Aluguel:

Listar Alugueis: http://localhost:8080/alugueis

Listar Aluguel por Id: http://localhost:8080/alugueis/{id}

Motorista:

Adicionar Motorista: http://localhost:8080/motoristas

Listar todos os Motoristas: http://localhost:8080/motoristas

Listar Motorista por Id: http://localhost:8080/motoristas/{id}

Obter o email de um Motorista: http://localhost:8080/motoristas/{id}/email