E-COMMERCE

DESENVOLVIDO POR: JEFFERSON SILVA

Jeff.jp1998@gmail.com

(98)98491-1746

O referido projeto implementa Swagger para auxiliar na documentação, sua integra pode ser acessada pela url /swagger-ui.html.

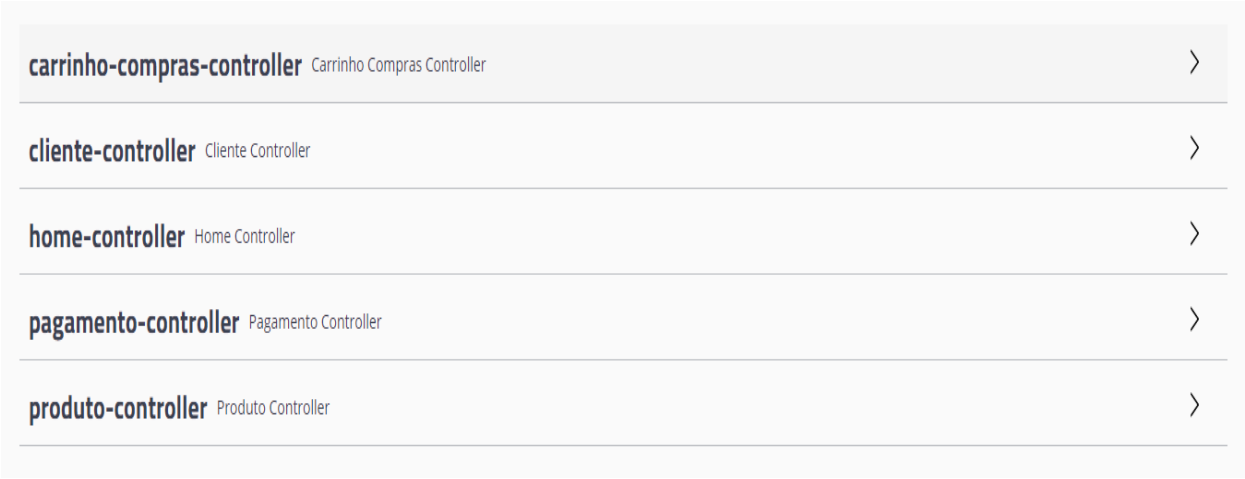
Foi Utilizado o banco de dados MySQL, o ddl-auto do hibernate foi definido como update no application.properties da aplicação. Para inicializar o projeto é necessário criar um database intitulado “ecommerce”. Ao inicializar a aplicação pela primeira vez o banco é copulado.

Para acessar alguns end-points é necessário utilizar o usuário padrão criado no projeto (usuário: lorem@teste.com, senha:12345) ou cadastrar um novo cliente, pois foi implementado o Spring Security no projeto.

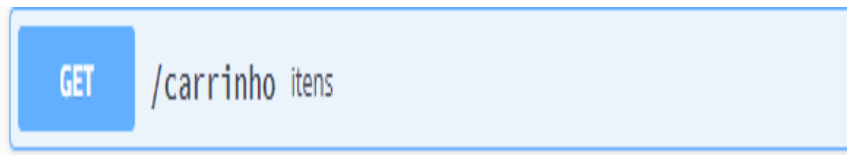
O Projeto conta em seu front-end com um navbar que auxilia no acesso aos end-points.

1. CONTROLLERS

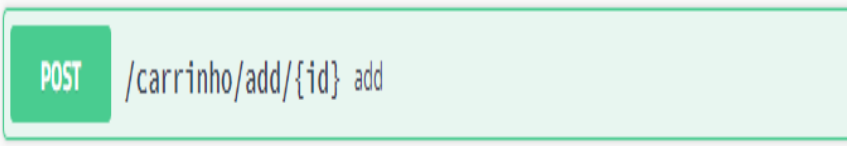
A imagem a seguir apresenta os controllers da aplicação.



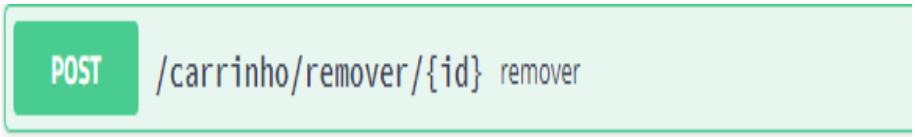
1.1 CarrinhoComprasController



Mapeado com um GetMapping para a url “/carrinho”, devolve para a view um objeto “CarrinhoCompra” que contém uma linkedHashMap com os itens presentes no carrinho.



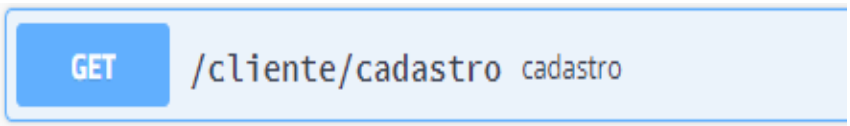
Mapeado com um PostMapping para a url “/carrinho/add/{id}”, recebe um id de “ProdutoDTO”, criar um objeto “CarrinhoItem” e adicionando ao “CarrinhoCompras”. Caso o produto já esteja no carrinho aumenta em 1 sua quantidade.



Mapeado com um PostMapping para a url “/carrinho/remover/{id}”, recebe um id de “ProdutoDTO”, e remove do “CarrinhoComprar” o “CarrinhoItem” que contém o objeto.

1.2 ClienteController

Foi implementado Spring Security no projeto, portanto alguns end-points precisam de autenticação, o cadastro é acessível através da url “/cliente/cadastro”.

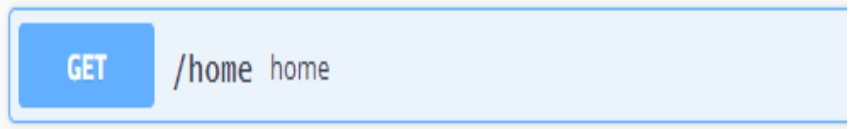


Ao submeter o formulário, a url “/cliente” mapeada como PostMapping, recebe o “ClienteDTOForm”, chama os métodos necessários para salvar no banco de dados.

Ao cadastrar um cliente, a api devolve como resposta os dados do cliente cadastrado para fins de validação.

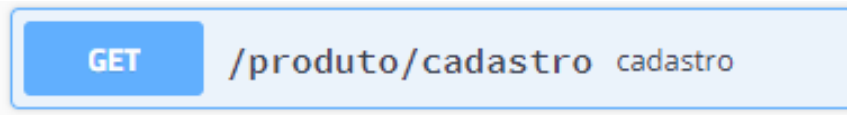
1.3 HomeController

Chamada para home pelo método GET pela url “/home”, essa tela apresenta todos os produtos cadastrados.

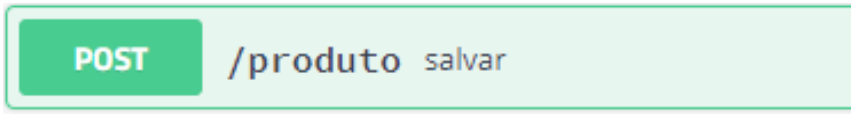


1.4 ProdutoController

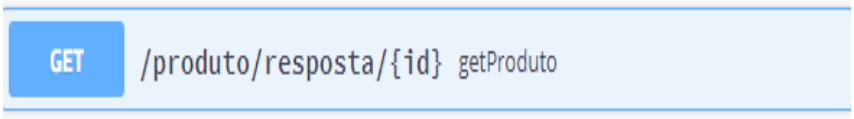
É necessário o cadastro de produtos, que serão salvos no banco de dados, o cadastro pode ser realizado por meio do endereço “/produto/cadastro”



Ao submeter o formulário, a url “/produto” mapeada com PostMapping recebe o “ProdutoDTOForm” e chama os métodos necessários para salvar no banco de dados.

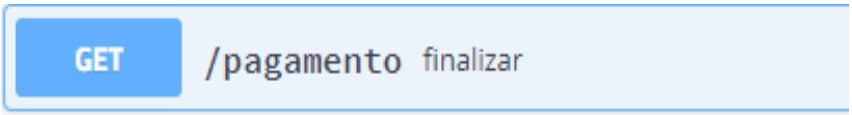


Ao cadastrar um produto, a API devolve como resposta os dados do produto cadastrado para fins de validação.



1.5 PagamentoController

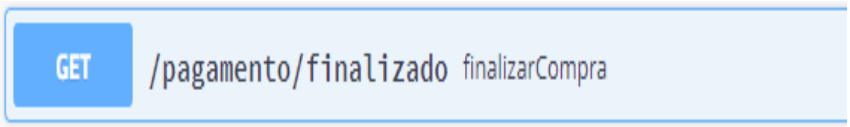
Caso tenha itens no carrinho, através da url “/pagamento” pode-se escolher o endereço e frete.



Saindo da tela anterior e indo para o “pagamento/confirmarCompra” pode se escolher os meios de pagamento e numero de parcelas, esse metodo recebe o total do carrinho de comprar

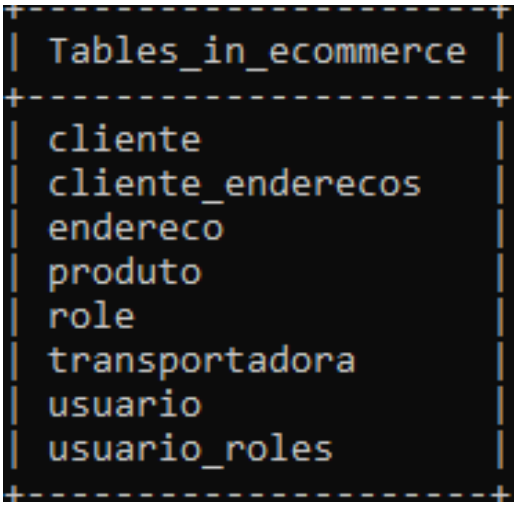
No endereço “pagamento/pagar” pode se confirmar o pagamento, esse end-point recebe o numero de parcelas e seu valor.

Tela que confirma o pagamento, apresenta o numero do pedido (um numero randomico de 6 digitos) e o codigo de rastreio.



2 MODELS E BANCO

Não foram feitas tabelas para todas as entididades no pacote models, a imagem abaixo apresenta as tabelas do banco de dados.



Foi utilizado o Spring Data JPA para a persistencia dos objetos.