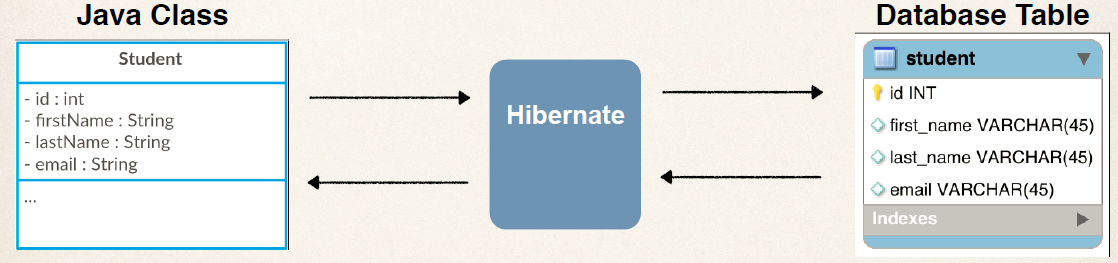
**Hibernate & JPA**



São 2 bibliotecas do módulo **Spring Data** voltadas para acesso a bancos de dados relacionais através do padrão **ORM – Mapeamento Objeto Relacional**.



**ORM – Mapeamento objeto relacional**

Recurso através do qual podemos configurar as classes JavaBean de entidade do projeto de forma que elas sejam interpretadas como tabelas do banco de dados. Por exemplo, fizemos o mapeamento ORM da entidade Cliente configurando a tabela do banco de dados para esta entidade.

package br.com.cotiinformatica.entities;

import java.util.Date;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.Table;

import javax.persistence.Temporal;

import javax.persistence.TemporalType;

import lombok.AllArgsConstructor;

import lombok.Getter;

import lombok.NoArgsConstructor;

import lombok.Setter;

import lombok.ToString;

**@Entity**

**@Table(name = "cliente")**

@Setter

@Getter

@NoArgsConstructor

@AllArgsConstructor

@ToString

public class Cliente {

**@Id**

**@GeneratedValue(strategy = GenerationType.IDENTITY)**

**@Column(name = "idcliente")**

private Integer idCliente;

**@Column(name = "nome", length = 150, nullable = false)**

private String nome;

**@Column(name = "cpf", length = 11, nullable = false, unique = true)**

private String cpf;

**@Column(name = "email", length = 50, nullable = false,**

**unique = true)**

private String email;

**@Column(name = "telefone", length = 14, nullable = false)**

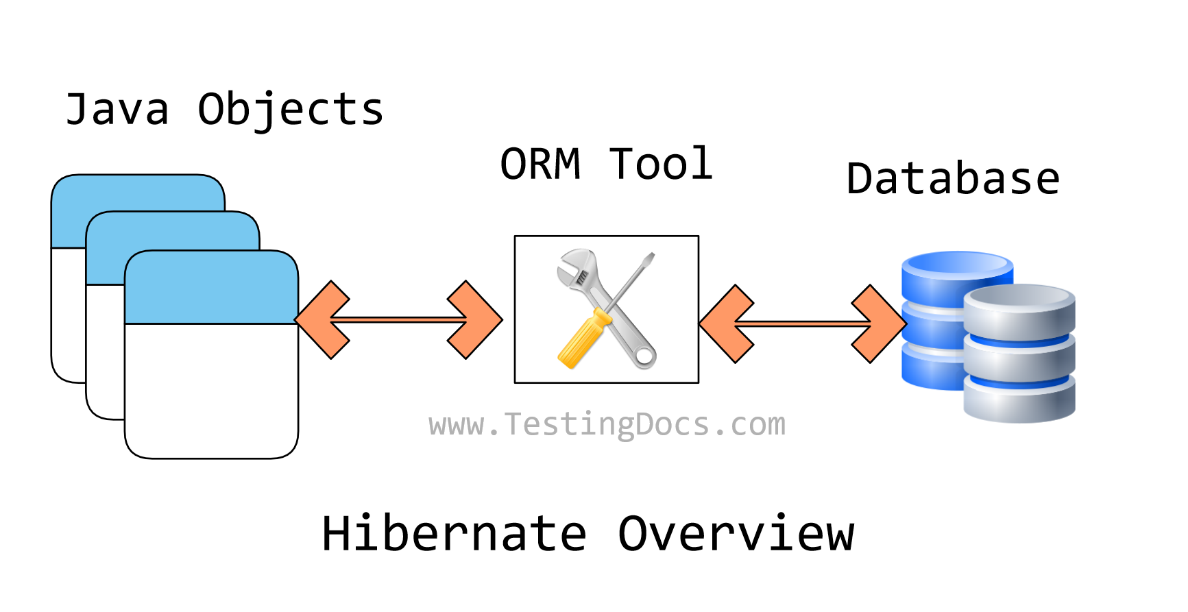
private String telefone;

**@Temporal(TemporalType.DATE)**

**@Column(name = "datacadastro", nullable = false)**

private Date dataCadastro;

}



Em seguida, configuramos a conexão com o banco de dados   
(DATA SOURCE) através do arquivo **application.properties**

server.port=8082

spring.datasource.url=**jdbc:postgresql://localhost:5432/bd\_apiclientes**

spring.datasource.driver-class-name=**org.postgresql.Driver**

spring.datasource.username=**postgres**

spring.datasource.password=**coti**

spring.jpa.properties.hibernate.dialect=**org.hibernate.dialect.PostgreSQLDialect**

spring.jpa.hibernate.ddl-auto=**update**

spring.jpa.show-sql=**true**

spring.jpa.properties.hibernate.format\_sql=**true**

**JpaRepository**

Interface que já disponibiliza os métodos básicos para desenvolvimento de um repositório com JPA & Hibernate, ou seja, já fornece um CRUD completo.

package br.com.cotiinformatica.repositories;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.stereotype.Repository;

import br.com.cotiinformatica.entities.Cliente;

**@Repository**

public interface IClienteRepository **extends JpaRepository<Cliente, Integer>** {

}

**Criando no projeto uma camada de serviços:**

Ou seja, vamos criar uma classe que será responsável por implementar todas as **regras de negócio** de uma determinada entidade, por exemplo: Cliente.

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

**package** br.com.cotiinformatica.services;

**import** java.util.Date;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Service;

**import** br.com.cotiinformatica.entities.Cliente;

**import** br.com.cotiinformatica.repositories.IClienteRepository;

**@Service**

**public** **class** ClienteService {

// injeção de dependência

@Autowired

**private** IClienteRepository clienteRepository;

// Método para realizar o cadastro de um cliente

**public** **void** cadastrar(Cliente cliente) {

// gerando a data de cadastro do cliente

cliente.setDataCadastro(**new** Date());

// gravando no banco de dados

clienteRepository.save(cliente);

}

}

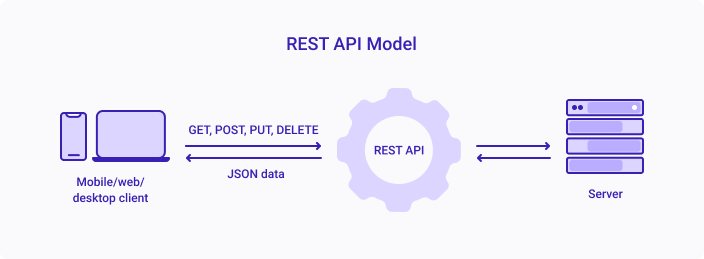
Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

/controllers/ClienteController.java

Voltando no controlador para implementar o serviço

de cadastro de cliente na API (HTTP POST)



Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

**package** br.com.cotiinformatica.controllers;

**import** java.util.List;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.http.HttpStatus;

**import** org.springframework.http.ResponseEntity;

**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.RestController;

**import** br.com.cotiinformatica.dtos.GetClientesDTO;

**import** br.com.cotiinformatica.dtos.PostClientesDTO;

**import** br.com.cotiinformatica.dtos.PutClientesDTO;

**import** br.com.cotiinformatica.entities.Cliente;

**import** br.com.cotiinformatica.services.ClienteService;

**import** io.swagger.annotations.Api;

**import** io.swagger.annotations.ApiOperation;

@Api(tags = "Clientes")

@RestController

**public** **class** ClientesController {

//injeção de dependência

@Autowired

**private** ClienteService clienteService;

@ApiOperation("Serviço para cadastro de clientes.")

@PostMapping("/api/clientes")

**public** ResponseEntity<GetClientesDTO> post

(@RequestBody PostClientesDTO dto) {

**try** {

Cliente cliente = **new** Cliente();

cliente.setNome(dto.getNome());

cliente.setEmail(dto.getEmail());

cliente.setTelefone(dto.getTelefone());

cliente.setCpf(dto.getCpf());

clienteService.cadastrar(cliente);

**return** ResponseEntity.*status*

(HttpStatus.***CREATED***).body(**null**);

}

**catch**(Exception e) {

**return** ResponseEntity.*status*

(HttpStatus.***INTERNAL\_SERVER\_ERROR***).body(**null**);

}

}

@ApiOperation("Serviço para edição de clientes.")

@PutMapping("/api/clientes")

**public** ResponseEntity<GetClientesDTO> put

(@RequestBody PutClientesDTO dto) {

**return** **null**;

}

@ApiOperation("Serviço para exclusão de clientes.")

@DeleteMapping("/api/clientes/{id}")

**public** ResponseEntity<GetClientesDTO> delete

(@PathVariable("id") Integer idCliente) {

**return** **null**;

}

@ApiOperation("Serviço para consulta de clientes.")

@GetMapping("/api/clientes")

**public** ResponseEntity<List<GetClientesDTO>> getAll() {

**return** **null**;

}

@ApiOperation("Serviço para consulta de cliente por id.")

@GetMapping("/api/clientes/{id}")

**public** ResponseEntity<GetClientesDTO> getById

(@PathVariable("id") Integer idCliente) {

**return** **null**;

}

}

**Executando o projeto:**

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Interface gráfica do usuário, Aplicativo, Tabela, Excel

Descrição gerada automaticamente

**Executando:**

<http://localhost:8082/swagger-ui/index.html>

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

**POSTMAN**

Principal ferramenta para realização de testes em APIs (Serviços)

<https://www.postman.com/downloads/>

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

**Criando um workspace:**

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

**Importando o link da documentação da API:**

Interface gráfica do usuário, Texto, Aplicativo, Site

Descrição gerada automaticamente

<http://localhost:8082/v2/api-docs>

Texto

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

No banco de dados:

**PgAdmin**

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

**Model Mapper**

Biblioteca do Spring utilizada para facilitar a transferência de dados entre objetos, fazendo com que o desenvolvedor possa em uma única linha de código transferir dados de um JAVABEAN para outro.



Instalando a biblioteca:

/pom.xml

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

<dependency>

<groupId>**org.modelmapper.extensions**</groupId>

<artifactId>**modelmapper-spring**</artifactId>

<version>**3.0.0**</version>

</dependency>

**Voltando no controlador:**

**package** br.com.cotiinformatica.controllers;

**import** java.util.List;

**import** org.modelmapper.ModelMapper;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.http.HttpStatus;

**import** org.springframework.http.ResponseEntity;

**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.RestController;

**import** br.com.cotiinformatica.dtos.GetClientesDTO;

**import** br.com.cotiinformatica.dtos.PostClientesDTO;

**import** br.com.cotiinformatica.dtos.PutClientesDTO;

**import** br.com.cotiinformatica.entities.Cliente;

**import** br.com.cotiinformatica.services.ClienteService;

**import** io.swagger.annotations.Api;

**import** io.swagger.annotations.ApiOperation;

@Api(tags = "Clientes")

@RestController

**public** **class** ClientesController {

//injeção de dependência

@Autowired

**private** ClienteService clienteService;

@ApiOperation("Serviço para cadastro de clientes.")

@PostMapping("/api/clientes")

**public** ResponseEntity<GetClientesDTO> post

(@RequestBody PostClientesDTO dto) {

**try** {

**ModelMapper modelMapper = new ModelMapper();**

**Cliente cliente = modelMapper.map**

**(dto, Cliente.class);**

clienteService.cadastrar(cliente);

**return** ResponseEntity.*status*

(HttpStatus.***CREATED***).body(**null**);

}

**catch**(Exception e) {

**return** ResponseEntity.*status*

(HttpStatus.***INTERNAL\_SERVER\_ERROR***).body(**null**);

}

}

@ApiOperation("Serviço para edição de clientes.")

@PutMapping("/api/clientes")

**public** ResponseEntity<GetClientesDTO> put

(@RequestBody PutClientesDTO dto) {

**return** **null**;

}

@ApiOperation("Serviço para exclusão de clientes.")

@DeleteMapping("/api/clientes/{id}")

**public** ResponseEntity<GetClientesDTO> delete

(@PathVariable("id") Integer idCliente) {

**return** **null**;

}

@ApiOperation("Serviço para consulta de clientes.")

@GetMapping("/api/clientes")

**public** ResponseEntity<List<GetClientesDTO>> getAll() {

**return** **null**;

}

@ApiOperation("Serviço para consulta de cliente por id.")

@GetMapping("/api/clientes/{id}")

**public** ResponseEntity<GetClientesDTO> getById

(@PathVariable("id") Integer idCliente) {

**return** **null**;

}

}

**Executando e testando:**

Interface gráfica do usuário, Tabela

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Criando uma classe DTO para refinar o retorno da API para os métodos POST, PUT e DELETE de clientes:

{

"**status**" : 201,

"**cliente**" : {

"**idCliente**" : 1,

"**nome**" : "Sergio Mendes",

"**cpf**" : "12345678900",

"**email**" : "sergio.coti@gmail.com",

"**telefone**" : "21969575900",

"**dataCadastro**" : "2023-03-01"

},

"**mensagem**" : "Cliente cadastrado com sucesso."

}

{

"**status**" : 400,

"**cliente**" : null,

"**mensagem**" : "O CPF informado já está cadastrado

no sistema, tente outro."

}

/dtos/ResponseClientesDTO.java

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

**package** br.com.cotiinformatica.dtos;

**import** lombok.AllArgsConstructor;

**import** lombok.Getter;

**import** lombok.NoArgsConstructor;

**import** lombok.Setter;

**import** lombok.ToString;

@Setter

@Getter

@NoArgsConstructor

@AllArgsConstructor

@ToString

**public** **class** ResponseClientesDTO {

**private** Integer status;

**private** GetClientesDTO cliente;

**private** String mensagem;

}

**Voltando no controlador:**

**package** br.com.cotiinformatica.controllers;

**import** java.util.List;

**import** org.modelmapper.ModelMapper;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.http.HttpStatus;

**import** org.springframework.http.ResponseEntity;

**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.RestController;

**import** br.com.cotiinformatica.dtos.GetClientesDTO;

**import** br.com.cotiinformatica.dtos.PostClientesDTO;

**import** br.com.cotiinformatica.dtos.PutClientesDTO;

**import** br.com.cotiinformatica.dtos.ResponseClientesDTO;

**import** br.com.cotiinformatica.entities.Cliente;

**import** br.com.cotiinformatica.services.ClienteService;

**import** io.swagger.annotations.Api;

**import** io.swagger.annotations.ApiOperation;

@Api(tags = "Clientes")

@RestController

**public** **class** ClientesController {

//injeção de dependência

@Autowired

**private** ClienteService clienteService;

@ApiOperation("Serviço para cadastro de clientes.")

@PostMapping("/api/clientes")

**public** ResponseEntity<ResponseClientesDTO> post

(@RequestBody PostClientesDTO dto) {

ResponseClientesDTO response = **new** ResponseClientesDTO();

**try** {

ModelMapper modelMapper = **new** ModelMapper();

Cliente cliente = modelMapper.map

(dto, Cliente.**class**);

clienteService.cadastrar(cliente);

response.setStatus(201);

response.setMensagem

("Cliente cadastrado com sucesso.");

response.setCliente

(modelMapper.map(cliente, GetClientesDTO.**class**));

**return** ResponseEntity.*status*

(HttpStatus.***CREATED***).body(response);

}

**catch**(Exception e) {

response.setStatus(500);

response.setMensagem(e.getMessage());

**return** ResponseEntity.*status*

(HttpStatus.***INTERNAL\_SERVER\_ERROR***).body(response);

}

}

@ApiOperation("Serviço para edição de clientes.")

@PutMapping("/api/clientes")

**public** ResponseEntity<GetClientesDTO> put

(@RequestBody PutClientesDTO dto) {

**return** **null**;

}

@ApiOperation("Serviço para exclusão de clientes.")

@DeleteMapping("/api/clientes/{id}")

**public** ResponseEntity<GetClientesDTO> delete

(@PathVariable("id") Integer idCliente) {

**return** **null**;

}

@ApiOperation("Serviço para consulta de clientes.")

@GetMapping("/api/clientes")

**public** ResponseEntity<List<GetClientesDTO>> getAll() {

**return** **null**;

}

@ApiOperation("Serviço para consulta de cliente por id.")

@GetMapping("/api/clientes/{id}")

**public** ResponseEntity<GetClientesDTO> getById

(@PathVariable("id") Integer idCliente) {

**return** **null**;

}

}

**Executando e testando:**

Interface gráfica do usuário, Tabela

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

**REQUEST BODY:**

Dados enviados para a API na requisição.

**{**

**"cpf": "12345678999",**

**"email": "sergio.coti@bol.com",**

**"nome": "Sergio Mendes",**

**"telefone": "21969575900"**

**}**

**RESPONSE:**

Conteúdo retornado pela API após o processamento da requisição.

**{**

**"status": 201,**

**"cliente": {**

**"idCliente": 3,**

**"nome": "Sergio Mendes",**

**"cpf": "12345678999",**

**"email": "sergio.coti@bol.com",**

**"telefone": "21969575900",**

**"dataCadastro": 1677716733743**

**},**

**"mensagem": "Cliente cadastrado com sucesso."**

**}**

**Implementando as regras de negócio para cadastro de clientes:**

* Todos os campos são obrigatórios
* O CPF não pode ser duplicado no banco de dados
* O Email não pode ser duplicado no banco de dados

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

package br.com.cotiinformatica.services;

import java.util.Date;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import br.com.cotiinformatica.entities.Cliente;

import br.com.cotiinformatica.repositories.IClienteRepository;

@Service

public class ClienteService {

// injeção de dependência

@Autowired

private IClienteRepository clienteRepository;

// Método para realizar o cadastro de um cliente

public void cadastrar(Cliente cliente) {

//verificando se os campos foram preenchidos

if(cliente.getNome() == null || cliente.getNome().trim().length() == 0) {

throw new IllegalArgumentException

("Por favor, informe o nome do cliente.");

}

else if(cliente.getCpf() == null || cliente.getCpf().trim().length() == 0) {

throw new IllegalArgumentException

("Por favor, informe o cpf do cliente.");

}

else if(cliente.getTelefone() == null

|| cliente.getTelefone().trim().length() == 0) {

throw new IllegalArgumentException

("Por favor, informe o telefone do cliente.");

}

else if(cliente.getEmail() == null || cliente.getEmail().trim().length() == 0) {

throw new IllegalArgumentException

("Por favor, informe o email do cliente.");

}

// gerando a data de cadastro do cliente

cliente.setDataCadastro(new Date());

// gravando no banco de dados

clienteRepository.save(cliente);

}

}

**HTTP STATUS CODES**

Sempre que uma API retorna resposta para o cliente, esta resposta é acompanhada por um código HTTP que indica se a requisição foi processada com sucesso ou com erro.

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

**package** br.com.cotiinformatica.controllers;

**import** java.util.List;

**import** org.modelmapper.ModelMapper;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.http.HttpStatus;

**import** org.springframework.http.ResponseEntity;

**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.RestController;

**import** br.com.cotiinformatica.dtos.GetClientesDTO;

**import** br.com.cotiinformatica.dtos.PostClientesDTO;

**import** br.com.cotiinformatica.dtos.PutClientesDTO;

**import** br.com.cotiinformatica.dtos.ResponseClientesDTO;

**import** br.com.cotiinformatica.entities.Cliente;

**import** br.com.cotiinformatica.services.ClienteService;

**import** io.swagger.annotations.Api;

**import** io.swagger.annotations.ApiOperation;

@Api(tags = "Clientes")

@RestController

**public** **class** ClientesController {

//injeção de dependência

@Autowired

**private** ClienteService clienteService;

@ApiOperation("Serviço para cadastro de clientes.")

@PostMapping("/api/clientes")

**public** ResponseEntity<ResponseClientesDTO> post

(@RequestBody PostClientesDTO dto) {

ResponseClientesDTO response = **new** ResponseClientesDTO();

**try** {

ModelMapper modelMapper = **new** ModelMapper();

Cliente cliente = modelMapper.map

(dto, Cliente.**class**);

clienteService.cadastrar(cliente);

response.setStatus(201);

response.setMensagem

("Cliente cadastrado com sucesso.");

response.setCliente(modelMapper.map

(cliente, GetClientesDTO.**class**));

**return** ResponseEntity.*status*(HttpStatus.***CREATED***)

.body(response);

}

**catch**(IllegalArgumentException e) {

response.setStatus(400);

response.setMensagem(e.getMessage());

**return** ResponseEntity.*status*

(HttpStatus.***BAD\_REQUEST***).body(response);

}

**catch**(Exception e) {

response.setStatus(500);

response.setMensagem(e.getMessage());

**return** ResponseEntity.*status*

(HttpStatus.***INTERNAL\_SERVER\_ERROR***).body(response);

}

}

@ApiOperation("Serviço para edição de clientes.")

@PutMapping("/api/clientes")

**public** ResponseEntity<GetClientesDTO> put

(@RequestBody PutClientesDTO dto) {

**return** **null**;

}

@ApiOperation("Serviço para exclusão de clientes.")

@DeleteMapping("/api/clientes/{id}")

**public** ResponseEntity<GetClientesDTO> delete

(@PathVariable("id") Integer idCliente) {

**return** **null**;

}

@ApiOperation("Serviço para consulta de clientes.")

@GetMapping("/api/clientes")

**public** ResponseEntity<List<GetClientesDTO>> getAll() {

**return** **null**;

}

@ApiOperation("Serviço para consulta de cliente por id.")

@GetMapping("/api/clientes/{id}")

**public** ResponseEntity<GetClientesDTO> getById

(@PathVariable("id") Integer idCliente) {

**return** **null**;

}

}

**Executando e testando:**

Interface gráfica do usuário, Aplicativo, Tabela, Excel

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

**Atualizando o GITHUB:**

Samsung@DESKTOP-P9F6D9F MINGW64 ~/Desktop/COTI - Aulas/2023 - Java WebDeveloper SQS 18h as 22h (Início em 09.01)/workspace/api\_clientes (main)

**$ git add .**

warning: in the working copy of 'pom.xml', LF will be replaced by CRLF the next time Git touches it

Samsung@DESKTOP-P9F6D9F MINGW64 ~/Desktop/COTI - Aulas/2023 - Java WebDeveloper SQS 18h as 22h (Início em 09.01)/workspace/api\_clientes (main)

**$ git commit -m 'Desenvolvimento da API de clientes'**

[main d8c1eb1] Desenvolvimento da API de clientes

4 files changed, 155 insertions(+), 6 deletions(-)

create mode 100644 src/main/java/br/com/cotiinformatica/dtos/ResponseClientesDTO.java

create mode 100644 src/main/java/br/com/cotiinformatica/services/ClienteService.java

Samsung@DESKTOP-P9F6D9F MINGW64 ~/Desktop/COTI - Aulas/2023 - Java WebDeveloper SQS 18h as 22h (Início em 09.01)/workspace/api\_clientes (main)

**$ git push -u origin main**

Enumerating objects: 26, done.

Counting objects: 100% (26/26), done.

Delta compression using up to 8 threads

Compressing objects: 100% (11/11), done.

Writing objects: 100% (15/15), 2.48 KiB | 845.00 KiB/s, done.

Total 15 (delta 4), reused 0 (delta 0), pack-reused 0

remote: Resolving deltas: 100% (4/4), completed with 4 local objects.

To https://github.com/smendescoti/api\_clientes.git

76dfa62..d8c1eb1 main -> main

branch 'main' set up to track 'origin/main'.