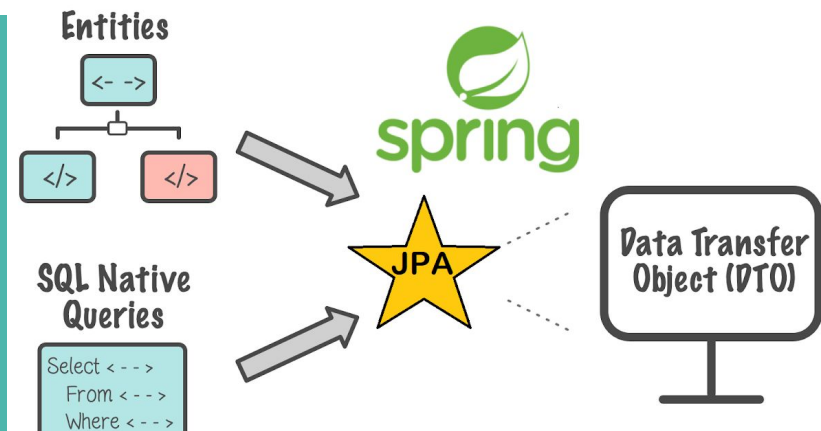

Desenvolvimento WEB

-

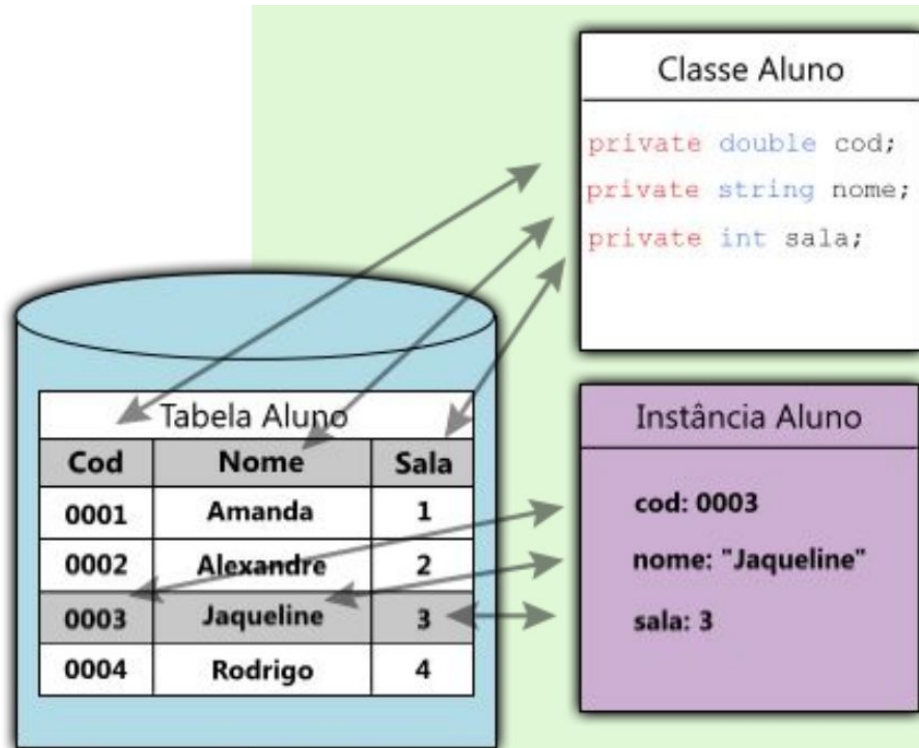
Full Stack Completo: Java + React

Acesso ao Banco de Dados com Spring Data JPA



Spring Data JPA

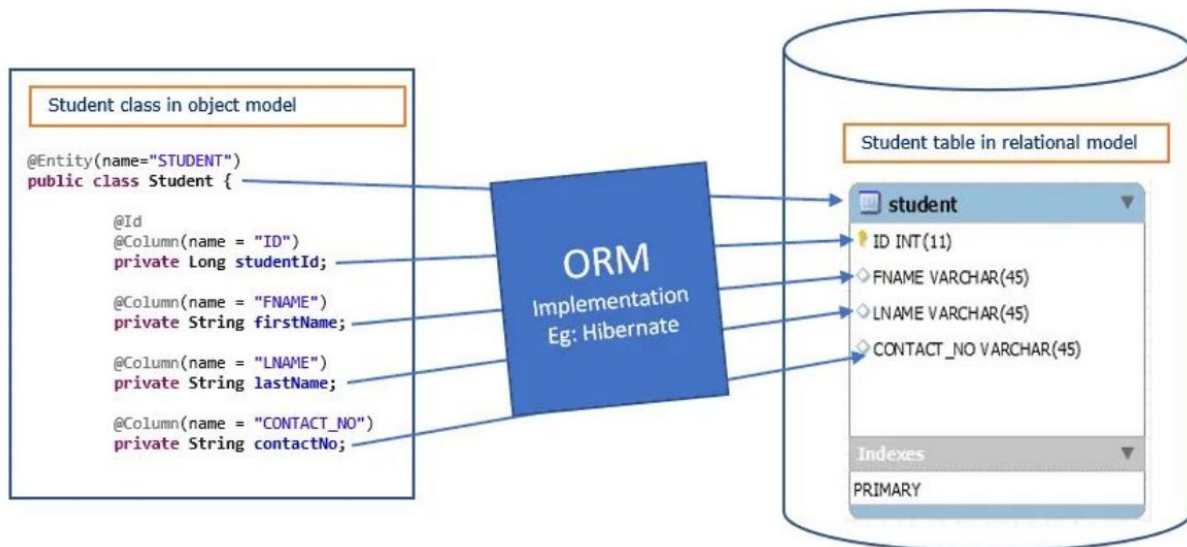
Como salvar
os dados de
um objeto no
banco de
dados?



Spring Data JPA



Mapeamento Objeto-Relacional (ORM)



ORM implements responsibility of mapping the Object to Relational Model.

Spring Data JPA

O que precisamos para usar o JPA em nosso projeto?

Verifique se as dependências do JPA e o Driver do banco de dados estão presentes no arquivo `pom.xml`. (Se não tiver, acrescente)

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
```

```
<dependency>
    <groupId>org.postgresql</groupId>
    <artifactId>postgresql</artifactId>
    <scope>runtime</scope>
</dependency>
```

Spring Data JPA

O que precisamos para usar o JPA em nosso projeto?

Implemente as configurações do banco de dados no arquivo `application.properties`

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

```
spring.datasource.url=jdbc:postgresql://localhost:5435/oxefood-noite
```

```
spring.datasource.username=postgres
```

```
spring.datasource.password=oxefood
```

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

```
spring.jpa.show-sql=false
```

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

```
spring.jpa.properties.hibernate.temp.use_jdbc_metadata_defaults = false
```

```
spring.jpa.database-platform=org.hibernate.dialect.PostgreSQL9Dialect
```

```
spring.jpa.generate-ddl=true
```

Spring Data JPA

O que precisamos para usar o JPA em nosso projeto?

Adicione as anotações do JPA nas classes de domínio (JavaBeans)

```
@Entity
```

```
@Table(name = "CategoriaProduto")
```

```
public class CategoriaProduto {
```

```
    @Id
```

```
    @GeneratedValue(strategy = GenerationType.SEQUENCE)
```

```
    private Long id;
```

```
    @Column
```

```
    private String chaveEmpresa;
```

```
    @Column(nullable = false, length = 100)
```

```
    private String descricao;
```

Spring Data JPA

O que precisamos para usar o JPA em nosso projeto?

Para cada classe de domínio, criar o respectivo repositório da entidade usando a interface **JpaRepository**

```
package br.com.ifpe.oxefoodnoite.modelo.produto;

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

public interface CategoriaProdutoRepository extends JpaRepository<CategoriaProduto, Long> {

}
```


Implementando a Inclusão no back-end

Início > Cadastro

E-mail *

Repetir e-mail *

Crie uma senha *

Repetir senha *

Dados cadastrais

☒ Pessoa física ☐ Pessoa jurídica

Nome *

Sobrenome *

Sexo *

CPF *

Telefone *

Celular *

Nascimento *

☒ Receber ofertas e novidades por e-mail

Endereço de entrega

CEP

Rua

Número

☐ Sem número

Complemento

Referência

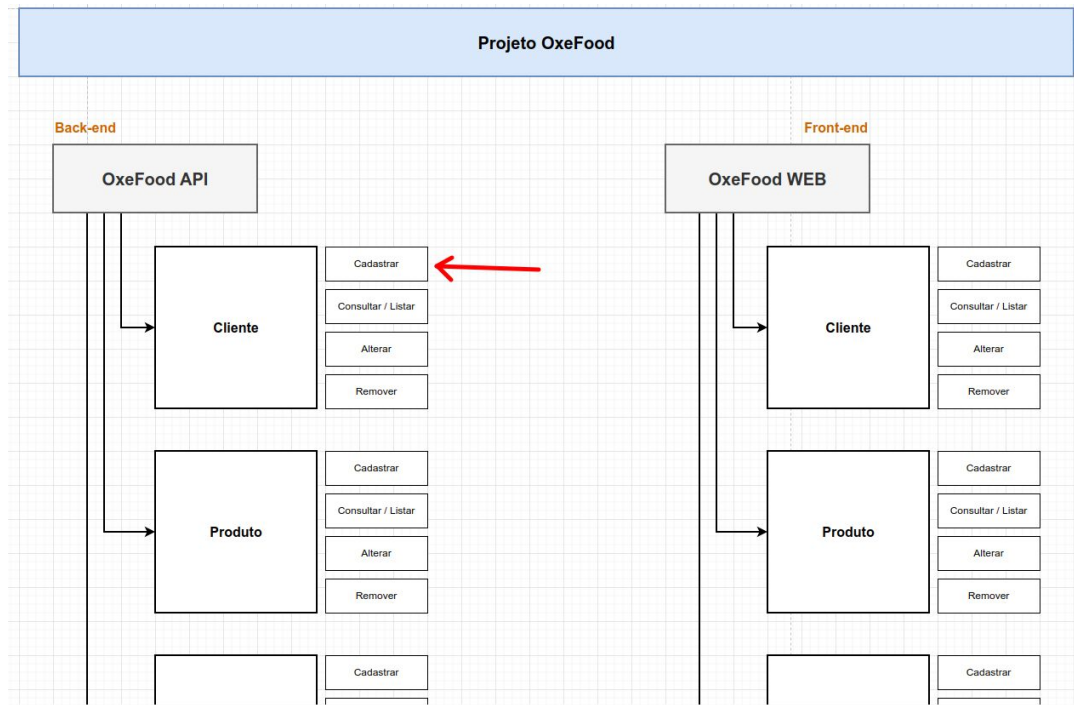
Bairro

Cidade

Estado

Implementando a Inclusão no Back-end

Meta: Implementar o cadastro de clientes no projeto do back-end do nosso sistema.



Implementando a Inclusão no Back-end

Meta: Implementar o cadastro de clientes no projeto do back-end do nosso sistema.

Cliente » Cadastro

Nome *

CPF

Fone Celular

Fone Fixo

Data Nascimento

Ex: 20/03/1985



Voltar



Salvar

Implementando a Incl

Meta: Implementar o cadastro de clientes no projeto do back-end do nosso sistema.

Cliente		
Atributo / Coluna	Tipo	Classe
id	Long	EntidadeNegocio
Habilitado	Boolean	EntidadeNegocio
versao	Long	EntidadeAuditavel
dataCriacao	LocalDate	EntidadeAuditavel
dataUltimaModificacao	LocalDate	EntidadeAuditavel
criadoPor	Long	EntidadeAuditavel
ultimaModificacaoPor	Long	EntidadeAuditavel
nome	String	Cliente
dataNascimento	LocalDate	Cliente
cpf	String	Cliente
foneCelular	String	Cliente
foneFixo	String	Cliente

Implementando a Inclusão no Back-end

Crie os arquivos abaixo (**classe** / **interface**):

- `br.com.ifpe.oxefood`
 - `api`
 - `cliente`
 - `ClienteController.java`
 - `ClienteRequest.java`
 - `config`
 - `modelo`
 - `cliente`
 - `Cliente.java`
 - `ClienteRepository.java`
 - `ClienteService.java`
 - `seguranca`
 - `util`

Implementando a Inclusão no Back-end

1.1) Classe Cliente:

Entidade Básica

JavaBean

Entidade de Domínio

```
package br.com.ifpe.oxefood.modelo.cliente;

import java.time.LocalDate;

import br.com.ifpe.oxefood.util.entity.EntidadeAuditavel;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

@Builder
@Getter
@Setter
@AllArgsConstructor
@NoArgsConstructor
public class Cliente extends EntidadeAuditavel {

    private String nome;

    private LocalDate dataNascimento;

    private String cpf;

    private String foneCelular;

    private String foneFixo;

}
```

Implementando

1.2) Classe Cliente:

Adicione as
anotações do JPA
na Entidade
Básica

```
package br.com.ifpe.oxefood.modelo.cliente;

import java.time.LocalDate;
import org.hibernate.annotations.SQLRestriction;
import br.com.ifpe.oxefood.util.entity.EntidadeAuditavel;
import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.Table;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

@Entity
@Table(name = "Cliente")
@SQLRestriction("habilitado = true")
@Builder
@Getter
@Setter
@AllArgsConstructor
@NoArgsConstructor
public class Cliente extends EntidadeAuditavel {

    @Column
    private String nome;

    @Column
    private LocalDate dataNascimento;

    @Column
    private String cpf;

    @Column
    private String foneCelular;

    @Column
    private String foneFixo;

}
```

Todas as anotações do JPA
devem ser importadas do pacote
`jakarta.persistence`

Implementando a Inclusão

Acrescente EntidadeAuditavel anotações do JPA:

```
package br.com.ifpe.oxefood.util.entity;

import java.time.LocalDate;
import org.springframework.data.annotation.CreatedDate;
import org.springframework.data.annotation.LastModifiedDate;
import com.fasterxml.jackson.annotation.JsonIgnore;
import jakarta.persistence.Column;
import jakarta.persistence.MappedSuperclass;
import jakarta.persistence.Version;
import lombok.Getter;
import lombok.Setter;

@SuppressWarnings("serial")
@Getter
@Setter
@MappedSuperclass
public abstract class EntidadeAuditavel extends EntidadeNegocio {

    @JsonIgnore
    @Version
    private Long versao;

    @JsonIgnore
    @CreatedDate
    private LocalDate dataCriacao;

    @JsonIgnore
    @LastModifiedDate
    private LocalDate dataUltimaModificacao;

    @JsonIgnore
    @Column
    private Long criadoPor; // Id do usuário que o criou

    @JsonIgnore
    @Column
    private Long ultimaModificacaoPor; // Id do usuário que fez a última alteração

}
```


Implementando a Inclusão

Acrescente EntidadeNegocio anotações do JPA:

```
package br.com.ifpe.oxefood.util.entity;

import java.io.Serializable;

import com.fasterxml.jackson.annotation.JsonIgnore;

import jakarta.persistence.Column;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.MappedSuperclass;
import lombok.EqualsAndHashCode;
import lombok.Getter;
import lombok.Setter;

@SuppressWarnings("serial")
@Getter
@Setter
@EqualsAndHashCode(of = { "id" })
@MappedSuperclass
public abstract class EntidadeNegocio implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE)
    private Long id;

    @JsonIgnore
    @Column
    private Boolean habilitado;

}
```

Implementando a Inc

2.1) Classe ClienteRequest:

```
package br.com.ifpe.oxefood.api.cliente;

import java.time.LocalDate;

import br.com.ifpe.oxefood.modelo.cliente.Cliente;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
public class ClienteRequest {

    private String nome;

    private LocalDate dataNascimento;

    private String cpf;

    private String foneCelular;

    private String foneFixo;

    public Cliente build() {

        return Cliente.builder()
            .nome(nome)
            .dataNascimento(dataNascimento)
            .cpf(cpf)
            .foneCelular(foneCelular)
            .foneFixo(foneFixo)
            .build();
    }
}
```

Implementando a Inc

2.2) Classe ClienteRequest:

Caso o atributo só armazene uma data (LocalDate), utilizar o formato:

```
@JsonFormat(pattern = "dd/MM/yyyy")
```

Caso o atributo armazene uma data e uma hora (LocalDateTime), utilizar o formato:

```
@JsonFormat(pattern = "dd/MM/yyyy HH:mm:ss")
```

```
package br.com.ifpe.oxefood.api.cliente;

import java.time.LocalDate;

import br.com.ifpe.oxefood.modelo.cliente.Cliente;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;
import lombok.NoArgsConstructor;

@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
public class ClienteRequest {

    private String nome;

    @JsonFormat(pattern = "dd/MM/yyyy")
    private LocalDate dataNascimento;

    private String cpf;

    private String foneCelular;

    private String foneFixo;

    public Cliente build() {

        return Cliente.builder()
            .nome(nome)
            .dataNascimento(dataNascimento)
            .cpf(cpf)
            .foneCelular(foneCelular)
            .foneFixo(foneFixo)
            .build();

    }

}
```

Adicione acima do campo de data a anotação **@JsonFormat** que especifica o formato em String do valor recebido que será convertido no LocalDate ou LocalDateTime.

Implementando a Inclusão no Back-end

3) Interface `ClienteRepository`:

```
package br.com.ifpe.oxefood.modelo.cliente;

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

public interface ClienteRepository extends JpaRepository<Cliente, Long> {

}
```

Spring Data JPA

O que precisamos para usar o JPA em nosso projeto?

<<Java Interface>>  JpaRepository<T,ID> org.springframework.data.jpa.repository	
● findAll(): List<T>	
● findAll(Sort): List<T>	
● findAllById(Iterable<ID>): List<T>	
● saveAll(Iterable<S>): List<S>	
● flush(): void	
● saveAndFlush(S): S	
● deleteInBatch(Iterable<T>): void	
● deleteAllInBatch(): void	
● getOne(ID): T	
● findAll(Example<S>): List<S>	
● findAll(Example<S>, Sort): List<S>	
● findAllById(Iterable): Iterable	
● saveAll(Iterable): Iterable	
● findAll(Example): Iterable	

JpaRepository	CRUD
save(T) : T	C-CREATE
findById(ID) : T findAll() : List<T>	R-READ
save(T) : T	U-UPDATE
deleteById(ID)	D-DELETE
https://docs.spring.io/spring-data/jpa/docs/current/api/org/springframework/data/jpa/repository/JpaRepository.html	

Implementando a Inclusão no Back-end

4) Classe ClienteService:

O pacote a ser importado para a anotação `@Transaction` é a do pacote `jakarta.transaction`

```
package br.com.ifpe.oxefood.modelo.cliente;

import jakarta.transaction.Transactional;

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

@Service
public class ClienteService {

    @Autowired
    private ClienteRepository repository;

    @Transactional
    public Cliente save(Cliente cliente) {

        cliente.setHabilitado(Boolean.TRUE);
        cliente.setVersao(1L);
        cliente.setDataCriacao(LocalDate.now());
        return repository.save(cliente);
    }
}
```

Implementando a Inclusão no Back-end

5) Classe ClienteController:

```
package br.com.ifpe.oxefood.api.cliente;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import br.com.ifpe.oxefood.modelo.cliente.Cliente;
import br.com.ifpe.oxefood.modelo.cliente.ClienteService;

@RestController
@RequestMapping("/api/cliente")
@CrossOrigin
public class ClienteController {

    @Autowired
    private ClienteService clienteService;

    @PostMapping
    public ResponseEntity<Cliente> save(@RequestBody ClienteRequest request) {

        Cliente cliente = clienteService.save(request.build());
        return new ResponseEntity<Cliente>(cliente, HttpStatus.CREATED);
    }
}
```

Implementando a Inclusão no Back-end

Abra o `pom.xml` e comente a dependência do `SpringSecurity`:

```
<!--  
    <dependency>  
        <groupId>org.springframework.boot</groupId>  
        <artifactId>spring-boot-starter-security</artifactId>  
    </dependency>  
-->
```


Implementando a Inclusão no Back-end

Resumo :: Classes que precisam ser implementadas:

- 1) Entidade Básica
- 2) ...Request.java
- 3) ...Repository.java
- 4) ...Service.java
- 5) ...Controller.java

Exemplo:

- 1) Produto.java
- 2) ProdutoRequest.java
- 3) ProdutoRepository.java
- 4) ProdutoService.java
- 5) ProdutoController.java

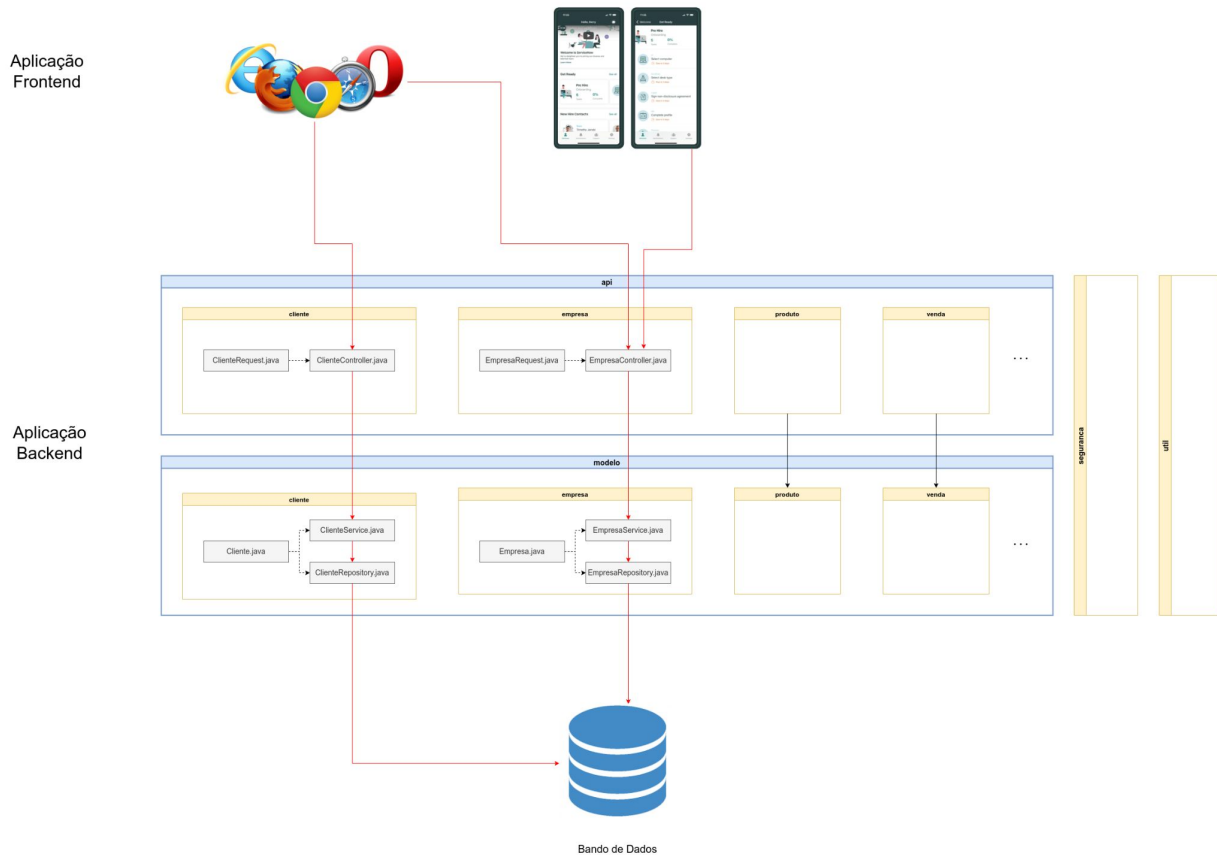
Exemplo:

- 1) Cliente.java
- 2) ClienteRequest.java
- 3) ClienteRepository.java
- 4) ClienteService.java
- 5) ClienteController.java

Exemplo:

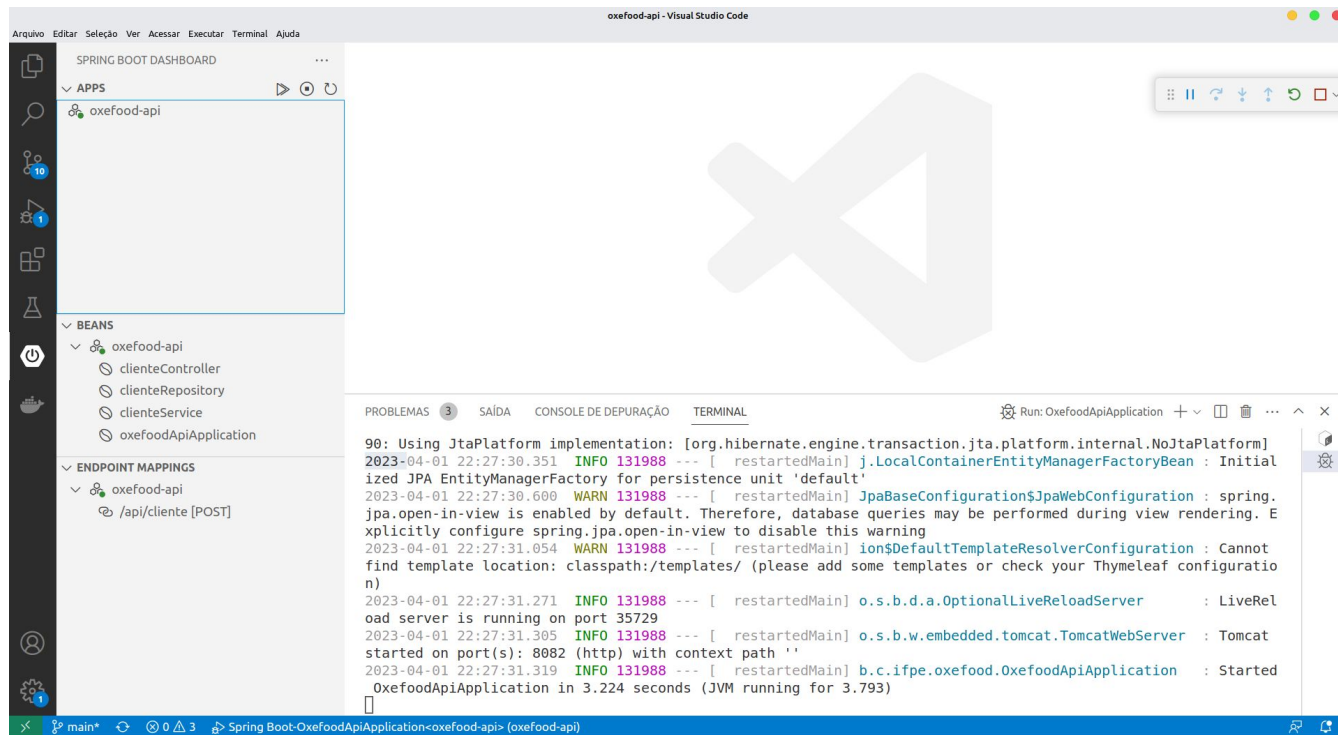
- 1) Entregador.java
- 2) EntregadorRequest.java
- 3) EntregadorRepository.java
- 4) EntregadorService.java
- 5) EntregadorController.java

Implementando a Inclusão no Back-end



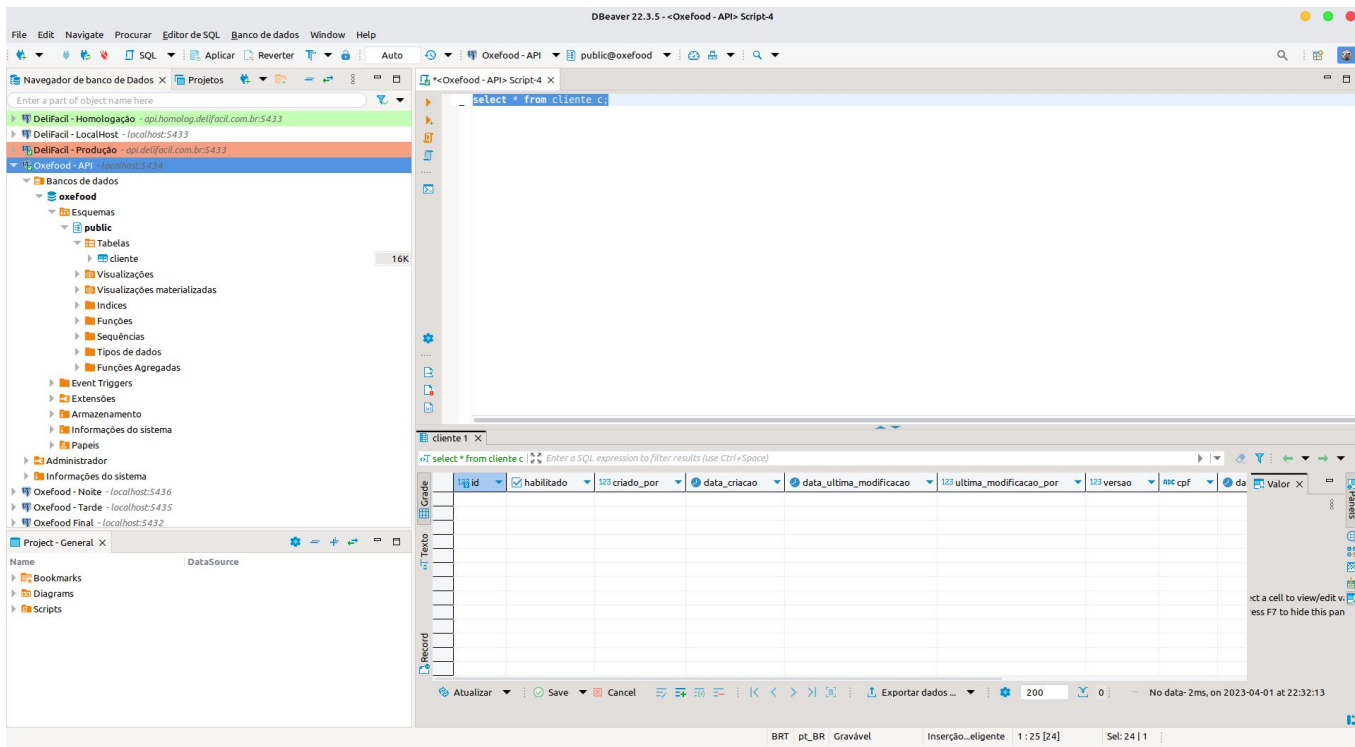
Implementando a Inclusão no Back-end

Levante a aplicação e teste o end-point do cadastro de cliente com a ferramenta Postman



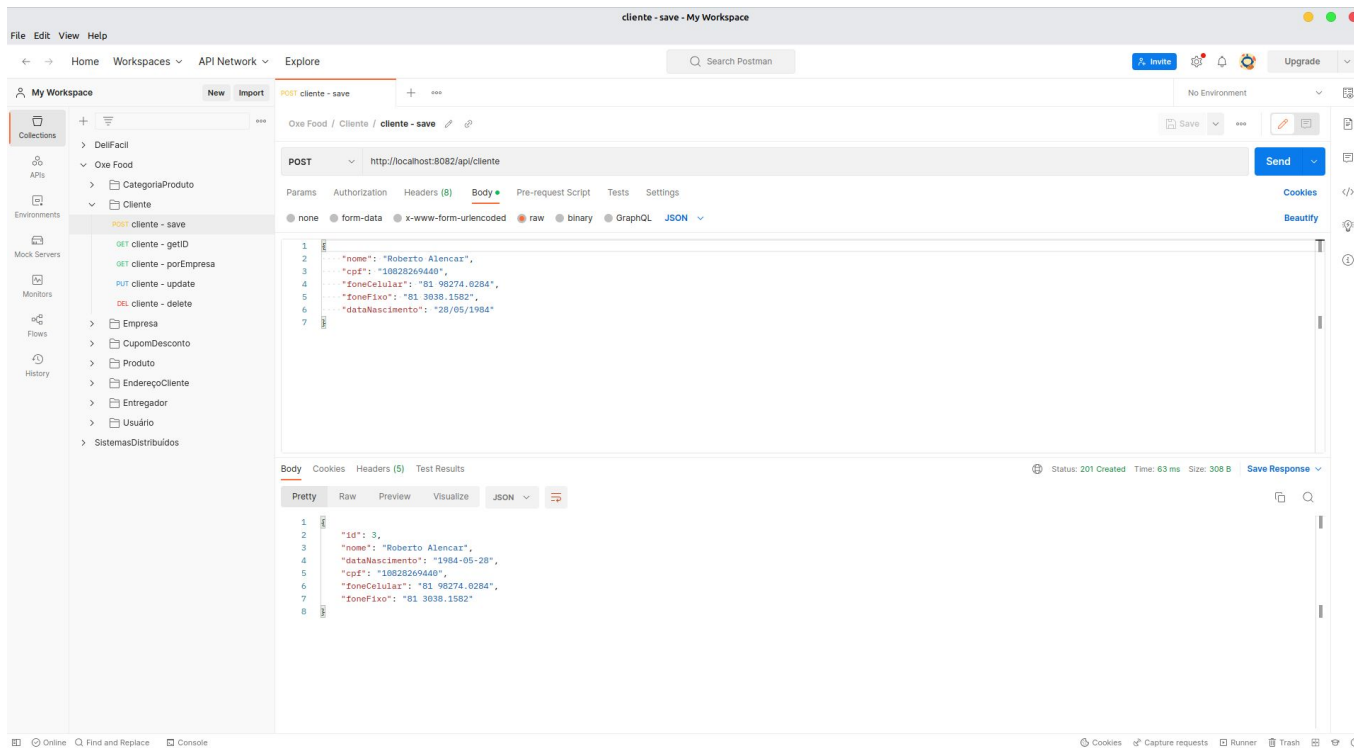
Implementando a Inclusão no Back-end

Levante a aplicação e teste o end-point do cadastro de cliente com a ferramenta Postman



Implementando a Inclusão no Back-end

Levante a aplicação e teste o end-point do cadastro de cliente com a ferramenta Postman



Implementando a Inclusão no Back-end

Levante a aplicação e teste o end-point do cadastro de cliente com a ferramenta Postman



DBeaver 22.3.5 - <Oxfood - API> Script-4

File Edit Navigate Procurar Editor de SQL Banco de dados Window Help

Navegador de banco de Dados X Projetos X <Oxfood - API> Script-4 X

Enter a part of object name here

- Delifaci - Homologação - api.homolog.delifaci.com.br:5433
- Delifaci - Localhost - localhost:5433
- Delifaci - Produção - api.delifaci.com.br:5433
- Oxfood - API - localhost:5434
 - Bancos de dados
 - oxfood
 - Esquemas
 - public
 - Tabelas
 - cliente 16K
 - Visualizações
 - Visualizações materializadas
 - Índices
 - Funções
 - Sequências
 - Tipos de dados
 - Funções Agregadas
 - Event Triggers
 - Extensões
 - Armazenamento
 - Informações do sistema
 - Papeis
 - Administrador
 - Informações do sistema
 - Oxfood - Noite - localhost:5436
 - Oxfood - Tarde - localhost:5435
 - Oxfood Final - localhost:5432

Project - General X

Name DataSource

Bookmarks

Diagrams

Scripts

select * from cliente c;

cliente 1 X

select * from cliente c; Enter a SQL expression to filter results (use Ctrl+Space)

Id	habilitado	123 criado_por	data_criacao	data_ultima_modificacao	123 ultima_modificacao_por	123 versao	not cpf	da	Valor X
1	M	[NULL]	2023-04-01	2023-04-01	[NULL]	1	108282694		

Atualizar Save Cancel Exportar dados ... 200 1

1 row(s) fetched - 2ms, on 2023-04-01 at 22:35:31

BRT pt_BR Gravável Inserção_eligente 1:25 [24] Sel: 24 | 1

Dúvidas



Exercício

1) Implemente no projeto do back-end o cadastro de Cliente



Cliente		
Atributo / Coluna	Tipo	Classe
id	Long	EntidadeNegocio
Habilitado	Boolean	EntidadeNegocio
versao	Long	EntidadeAuditavel
dataCriacao	LocalDate	EntidadeAuditavel
dataUltimaModificacao	LocalDate	EntidadeAuditavel
criadoPor	Long	EntidadeAuditavel
ultimaModificacaoPor	Long	EntidadeAuditavel
nome	String	Cliente
dataNascimento	LocalDate	Cliente
cpf	String	Cliente
foneCelular	String	Cliente
foneFixo	String	Cliente

Exercício

2) Implemente no projeto do back-end o cadastro de Produto



Produto		
Atributo / Coluna	Tipo	Classe
id	Long	EntidadeNegocio
...		
versao	Long	EntidadeAuditavel
...		
codigo	String	Produto
titulo	String	Produto
descricao	String	Produto
valorUnitario	Double	Produto
tempoEntregaMinimo	Integer	Produto
tempoEntregaMaximo	Integer	Produto

Exercício

3) Implemente no projeto do back-end o cadastro de Entregador

continua



Entregador		
Atributo / Coluna	Tipo	Classe
id	Long	EntidadeNegocio
...		
versao	Long	EntidadeAuditavel
...		
nome	String	Entregador
cpf	String	Entregador
rg	String	Entregador
dataNascimento	LocalDate	Entregador
foneCelular	String	Entregador
foneFixo	String	Entregador
qtdEntregasRealizadas	Integer	Entregador
...		

Exercício

3) Implemente no projeto do back-end o cadastro de Entregador



Entregador		
Atributo / Coluna	Tipo	Classe
...		
valorFrete	Double	Entregador
enderecoRua	String	Entregador
enderecoNumero	String	Entregador
enderecoBairro	String	Entregador
enderecoCidade	String	Entregador
enderecoCep	String	Entregador
enderecoUf	String	Entregador
enderecoComplemento	String	Entregador
ativo	Boolean	Entregador

The background features two large, solid green abstract shapes. One is a semi-circle on the left side, and the other is a more complex, organic shape on the right side.

Obrigado !