# Curso Java COMPLETO

devsuperior.com.br

**Dr. Nelio Alves**

## Projeto web services com Spring Boot e JPA / Hibernate

### Objetivos

- Criar projeto Spring Boot Java
- Implementar modelo de domínio
- Estruturar camadas lógicas: resource, service, repository
- Configurar banco de dados de teste (H2)
- Povoar o banco de dados
- CRUD - Create, Retrieve, Update, Delete
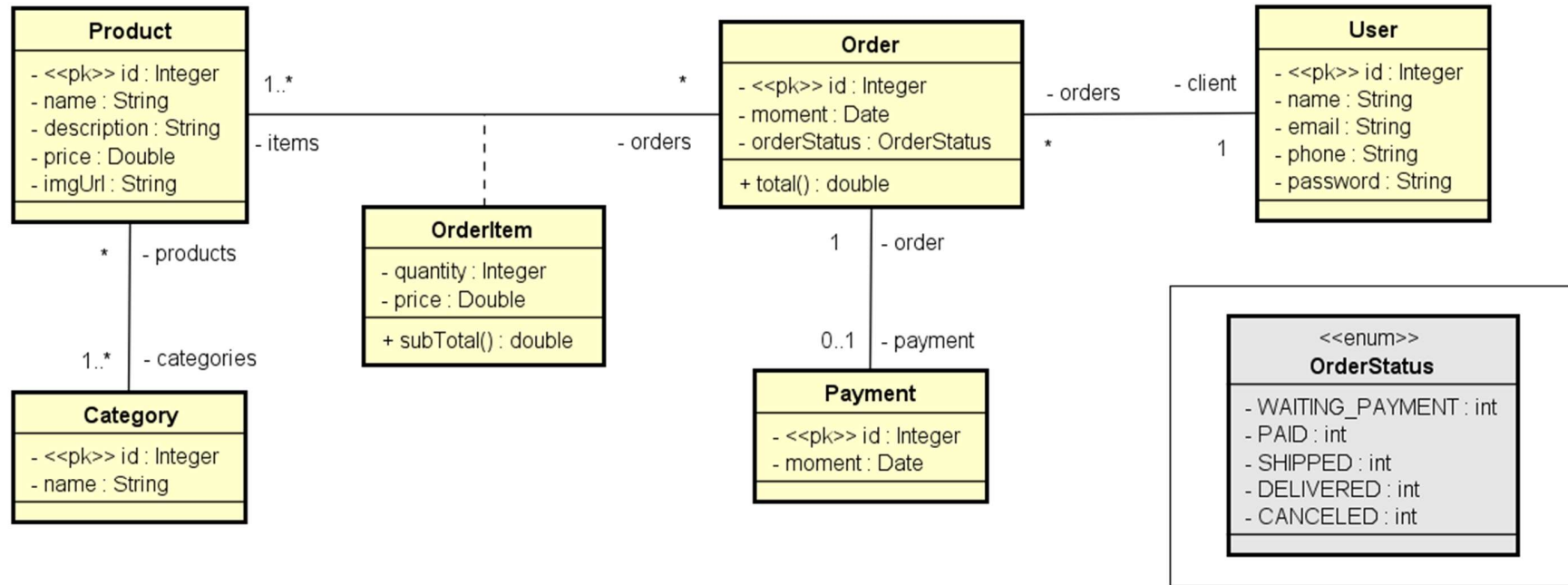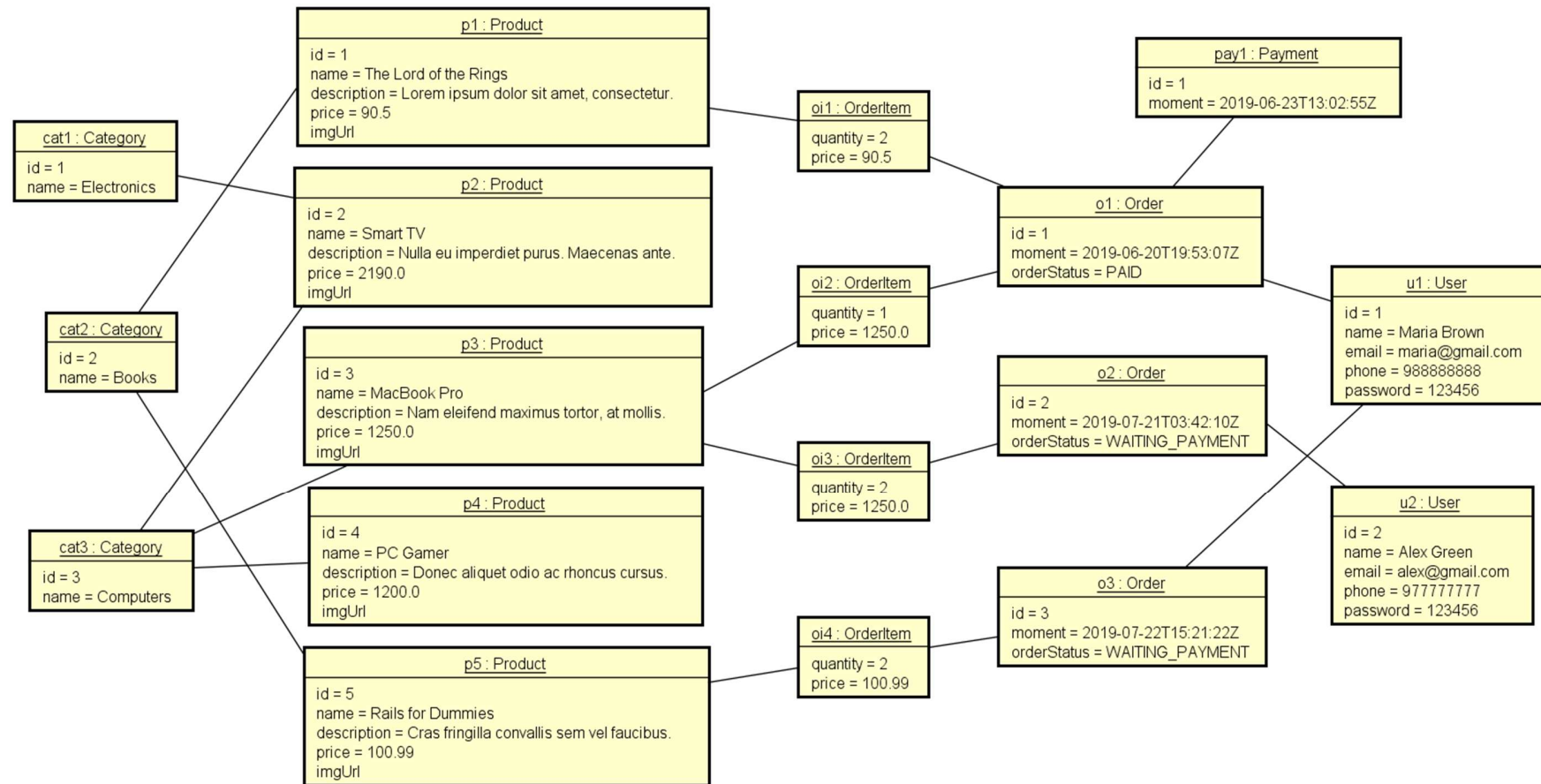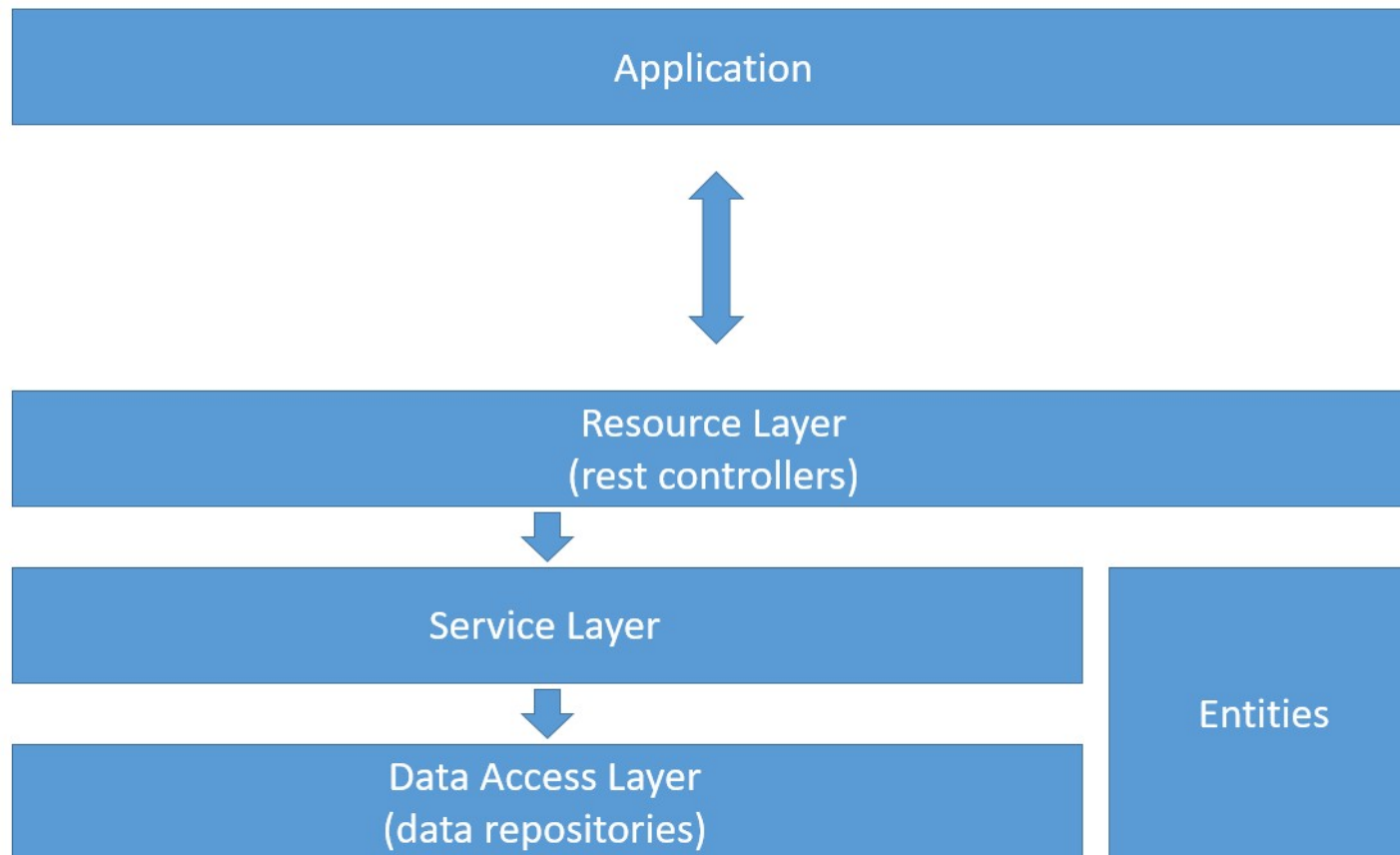- Tratamento de exceções

### Github:

https://github.com/acenelio/workshop-springboot4-jpa

https://github.com/acenelio/workshop-springboot3-jpa

https://github.com/acenelio/workshop-springboot2-jpa

# Domain Model

# Domain Instance

**p1 : Product**

id = 1
name = The Lord of the Rings
description = Lorem ipsum dolor sit amet, consectetur.
price = 90.5
imgUrl

**p2 : Product**

id = 2
name = Smart TV
description = Nulla eu imperdiet purus. Maecenas ante.
price = 2190.0
imgUrl

**p3 : Product**

id = 3
name = MacBook Pro
description = Nam eleifend maximus tortor, at mollis.
price = 1250.0
imgUrl

**p4 : Product**

id = 4
name = PC Gamer
description = Donec aliquet odio ac rhoncus cursus.
price = 1200.0
imgUrl

**p5 : Product**

id = 5
name = Rails for Dummies
description = Cras fringilla convallis sem vel faucibus.
price = 100.99
imgUrl

**cat1 : Category**

id = 1
name = Electronics

**cat2 : Category**

id = 2
name = Books

**cat3 : Category**

id = 3
name = Computers

**oi1 : OrderItem**

quantity = 2
price = 90.5

**oi2 : OrderItem**

quantity = 1
price = 1250.0

**oi3 : OrderItem**

quantity = 2
price = 1250.0

**oi4 : OrderItem**

quantity = 2
price = 100.99

**pay1 : Payment**

id = 1
moment = 2019-06-23T13:02:55Z

**o1 : Order**

id = 1
moment = 2019-06-20T19:53:07Z
orderStatus = PAID

**o2 : Order**

id = 2
moment = 2019-07-21T03:42:10Z
orderStatus = WAITING_PAYMENT

**o3 : Order**

id = 3
moment = 2019-07-22T15:21:22Z
orderStatus = WAITING_PAYMENT

**u1 : User**

id = 1
name = Maria Brown
email = maria@gmail.com
phone = 988888888
password = 123456

**u2 : User**

id = 2
name = Alex Green
email = alex@gmail.com
phone = 977777777
password = 123456

**Logical Layers**

# Project created

**Checklist:**
- Spring Initializr
    - Maven
    - Java 25
    - Packing JAR
    - Dependencies: Spring Web

# User entity and resource

**Basic entity checklist:**
- Basic attributes
- Associations (instantiate collections)
- Constructors
- Getters & Setters (collections: only get)
- hashCode & equals
- Serializable

# H2 database, test profile, JPA

**Checklist:**
- JPA & H2 dependencies
- application.properties
- application-test.properties
- Entity: JPA mapping

**Dependencies:**

**ATENÇÃO: não precisa adicionar se estas dependências já estiverem no seu pom.xml:**

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-h2console</artifactId>
</dependency>
<dependency>
    <groupId>com.h2database</groupId>
    <artifactId>h2</artifactId>
    <scope>runtime</scope>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
```

**application.properties:**

```
spring.profiles.active=test

spring.jpa.open-in-view=true
```

**application-test.properties:**

```
# DATASOURCE
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.url=jdbc:h2:mem:testdb
spring.datasource.username=sa
spring.datasource.password=

# H2 CLIENT
spring.h2.console.enabled=true
spring.h2.console.path=/h2-console

# JPA, SQL
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
spring.jpa.defer-datasource-initialization=true
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true
```

## JPA repository, dependency injection, database seeding

**Checklist:**

- UserRepository extends JPARepository<User, Long>
- Configuration class for "test" profile
- @Autowired UserRepository
- Instantiate objects in memory
- Persist objects

**Objects:**

```
User u1 = new User(null, "Maria Brown", "maria@gmail.com", "988888888", "123456");
User u2 = new User(null, "Alex Green", "alex@gmail.com", "977777777", "123456");
```

## Service layer, component registration

## Order, Instant, ISO 8601

**Basic new entity checklist:**

- Entity
  - "To many" association, lazy loading, JsonIgnore
- Repository
- Seed
- Service

- Resource

**Objects:**
```java
Order o1 = new Order(null, Instant.parse("2019-06-20T19:53:07Z"), u1);
Order o2 = new Order(null, Instant.parse("2019-07-21T03:42:10Z"), u2);
Order o3 = new Order(null, Instant.parse("2019-07-22T15:21:22Z"), u1);
```

## OrderStatus enum

## Category

**Objects:**
```java
Category cat1 = new Category(null, "Electronics");
Category cat2 = new Category(null, "Books");
Category cat3 = new Category(null, "Computers");
```

## Product

**Objects:**
```java
Product p1 = new Product(null, "The Lord of the Rings", "Lorem ipsum dolor sit amet, consectetur.", 90.5, "");
Product p2 = new Product(null, "Smart TV", "Nulla eu imperdiet purus. Maecenas ante.", 2190.0, "");
Product p3 = new Product(null, "Macbook Pro", "Nam eleifend maximus tortor, at mollis.", 1250.0, "");
Product p4 = new Product(null, "PC Gamer", "Donec aliquet odio ac rhoncus cursus.", 1200.0, "");
Product p5 = new Product(null, "Rails for Dummies", "Cras fringilla convallis sem vel faucibus.", 100.99, "");
```

## Many-to-many association with JoinTable

## OrderItem, many-to-many association with extra attributes

**Checklist:**
- OrderItemPK
- OrderItem
- Order one-to-many association
- Seed

**Objects:**
```java
OrderItem oi1 = new OrderItem(o1, p1, 2, p1.getPrice());
OrderItem oi2 = new OrderItem(o1, p3, 1, p3.getPrice());
OrderItem oi3 = new OrderItem(o2, p3, 2, p3.getPrice());
OrderItem oi4 = new OrderItem(o3, p5, 2, p5.getPrice());
```

# Product-OrderItem one-to-many association

# Payment, one-to-one association

# Subtotal & Total methods

# User insert

**Checklist:**
- UserService
- UserResource

**Test:**
```
{
    "name": "Bob Brown",
    "email": "bob@gmail.com",
    "phone": "977557755",
    "password": "123456"
}
```

# User delete

**Checklist:**
- UserService
- UserResource

# User update

**Checklist:**
- UserService
- UserResource

**Test:**
```
{
    "name": "Bob Brown",
    "email": "bob@gmail.com",
    "phone": "977557755"
}
```

# Exception handling - findById

**Checklist:**

- NEW CLASS: services.exceptions.ResourceNotFoundException
- NEW CLASS: resources.exceptions.StandardError
- NEW CLASS: resources.exceptions.ResourceExceptionHandler
- UserService

# Exception handling - delete

**Checklist:**

- NEW CLASS: services.exceptions.DatabaseException
- ResourceExceptionHandler
- UserService
    - EmptyResultDataAccessException
    - DataIntegrityViolationException

# Exception handling - update

**Checklist:**

- UserService
    - EntityNotFoundException

# Deploy Heroku

## Create Heroku app & provision PostgreSQL

**Checklist:**
- Heroku Sign Up
- Create app
- Provision PostgreSQL
  - App dashboard -> Resources
  - Search "postgres" -> select "Heroku Postgres"

## Install local PostgreSQL

**Checklist:**
- Download and install: https://www.postgresql.org/download/
  - Super user: postgres
  - Password: 1234567
  - Port: 5432
- Start/stop service: Task manager -> Services
- Check instalation
  - Start pgAdmin
  - Databases -> Create -> Database
    - Encoding: UTF8

## Dev profile

**Checklist:**
- PgAdmin: create local database: `create database springboot_course`

- Add PostgreSQL Maven dependency

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

- Create file: application-dev.properties

```
spring.datasource.url=jdbc:postgresql://localhost:5432/springboot_course
spring.datasource.username=postgres
spring.datasource.password=1234567

spring.jpa.properties.hibernate.jdbc.lob.non_contextual_creation=true
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true

jwt.secret=MYJWTSECRET
jwt.expiration=3600000
```

- Update application.properties: `spring.profiles.active=dev`

- Run application

# Get SQL script from local PostgreSQL

- PgAdmin: get SQL script:
  - Select database
  - Tools -> Backup
    - Format: Plain
    - Encoding: UTF8
    - Dump options:
      - Only schema: YES
      - Blobs: NO
      - Do not save: (ALL)
      - Verbose messages: NO

- Delete instructions before CREATE statements

# Run SQL Script

**Checklist:**
- App dashboard -> Settings - > Config Vars

  EXAMPLE:

  postgres://wavglvupbdad:358f443aafe452eca4c58fbc15d02e50b08130c7aaea3aff6c4f59c
  13f9abb@ec2-23-21-106-266.compute-1.amazonaws.com:5432/d7u9ub86cdsu

  **user**: wavglvupbdad

  **password**: 358f443aafe452eca4c58fbc15d02e50b08130c7aaea3aff6c4f59c13f9abb

  **server**: ec2-23-21-106-266.compute-1.amazonaws.com

**port**: 5432

**database**: d7u9ub86cdsu

- PgAdmin: Servers -> Create -> Server
  - o Advanced -> DB rescriction: (database)

- Database -> Query Tool
  - o Load and run SQL Script

## Heroku CLI

**Google: Heroku CLI**

**Terminal**
```
heroku login

winpty heroku.cmd login
```

## Deploy app to Heroku

- Heroku app dashboard -> Deploy
```
heroku git:remote -a myapp
git remote -v
```

- Setup Heroku app Config Vars
  - o DATABASE_URL
  - o JWT_EXPIRATION
  - o JWT_SECRET

- Create: application-prod.properties

```
spring.datasource.url=${DATABASE_URL}

spring.jpa.hibernate.ddl-auto=none
spring.jpa.show-sql=false
spring.jpa.properties.hibernate.format_sql=false

jwt.secret=${JWT_SECRET}
jwt.expiration=${JWT_EXPIRATION}
```

- Update application.properties: `spring.profiles.active=prod`

- Create files: system.properties

```
java.runtime.version=17
```

- Send to Heroku:
  ```
  git add .
  git commit -m "Deploy app to Heroku"
  git push heroku main
  ```