# Solução Passo-a-Passo de Exercício 9.2

(Sistema de Gestão de Clientes e Pedidos)

## 1. Início do Projeto com Spring Initializr

1. Acesse [Spring Initializr](https://start.spring.io/).
2. Configure o projeto:
   * **Project**: Maven
   * **Language**: Java
   * **Spring Boot**: Escolha a versão mais recente estável.
   * **Dependencies**:
     + Spring Web
     + Spring Data JPA
     + H2 Database
3. Clique em **"Generate"** para gerar o projeto e faça o download do ficheiro .zip.
4. Extraia o .zip e abra o projeto na sua IDE (recomendado: Visual Studio Code).

## 2. Estrutura Final do Projeto

O seu projeto deve ter a seguinte estrutura no final:

src  
├── main  
│ ├── java  
│ │ └── com.example.demo  
│ │ ├── controller  
│ │ │ ├── ClienteControlador.java  
│ │ │ └── PedidoControlador.java  
│ │ ├── model  
│ │ │ ├── Cliente.java  
│ │ │ └── Pedido.java  
│ │ ├── repository  
│ │ │ ├── ClienteRepositorio.java  
│ │ │ └── PedidoRepositorio.java  
│ │ ├── DatabaseLoader.java  
│ │ └── DemoApplication.java  
│ └── resources  
│ ├── application.properties  
│ └── data.sql (opcional)

## 3. Configuração do application.properties

Adicione estas configurações no ficheiro application.properties:

spring.datasource.url=jdbc:h2:file:./data/testdb  
spring.datasource.driver-class-name=org.h2.Driver  
spring.datasource.username=sa  
spring.datasource.password=  
spring.jpa.hibernate.ddl-auto=update  
spring.h2.console.enabled=true  
spring.h2.console.path=/h2-console  
spring.jpa.show-sql=true

## 4. Criar as Pastas e Ficheiros

Crie as seguintes pastas dentro do diretório src/main/java/com/example/demo:

* **controller**
* **model**
* **repository**

Depois, crie os ficheiros Java indicados nas próximas secções.

## 5. Implementação do Código

### Entidade Cliente

package com.example.demo.model;  
  
import jakarta.persistence.\*;  
import java.util.ArrayList;  
import java.util.List;  
import com.fasterxml.jackson.annotation.JsonManagedReference;  
  
@Entity  
public class Cliente {  
 @Id  
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 private Long id;  
 private String nome;  
 private String email;  
  
 @OneToMany(mappedBy = "cliente", cascade = CascadeType.ALL, orphanRemoval = true)  
 @JsonManagedReference  
 private List<Pedido> pedidos = new ArrayList<>();  
  
 public Cliente() {}  
 public Cliente(String nome, String email) {  
 this.nome = nome;  
 this.email = email;  
 }  
 // Getters e setters  
}

### Entidade Pedido

package com.example.demo.model;  
  
import com.fasterxml.jackson.annotation.JsonBackReference;  
import jakarta.persistence.\*;  
  
@Entity  
public class Pedido {  
 @Id  
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 private Long id;  
 private String descricao;  
  
 @ManyToOne  
 @JoinColumn(name = "cliente\_id")  
 @JsonBackReference  
 private Cliente cliente;  
  
 public Pedido() {}  
 public Pedido(String descricao, Cliente cliente) {  
 this.descricao = descricao;  
 this.cliente = cliente;  
 }  
 // Getters e setters  
}

### Repositórios

**ClienteRepositorio.java**

package com.example.demo.repository;  
  
import com.example.demo.model.Cliente;  
import org.springframework.data.jpa.repository.JpaRepository;  
  
public interface ClienteRepositorio extends JpaRepository<Cliente, Long> {}

**PedidoRepositorio.java**

package com.example.demo.repository;  
  
import com.example.demo.model.Pedido;  
import org.springframework.data.jpa.repository.JpaRepository;  
  
public interface PedidoRepositorio extends JpaRepository<Pedido, Long> {}

### Controladores

**ClienteControlador.java**

package com.example.demo.controller;  
  
import com.example.demo.model.Cliente;  
import com.example.demo.repository.ClienteRepositorio;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.\*;  
  
import java.util.List;  
  
@RestController  
@RequestMapping("/api/clientes")  
public class ClienteControlador {  
 private final ClienteRepositorio clienteRepositorio;  
  
 public ClienteControlador(ClienteRepositorio clienteRepositorio) {  
 this.clienteRepositorio = clienteRepositorio;  
 }  
  
 @GetMapping  
 public List<Cliente> listarClientes() {  
 return clienteRepositorio.findAll();  
 }  
  
 @PostMapping  
 public Cliente adicionarCliente(@RequestBody Cliente cliente) {  
 return clienteRepositorio.save(cliente);  
 }  
}

**PedidoControlador.java**

package com.example.demo.controller;  
  
import com.example.demo.model.Pedido;  
import com.example.demo.repository.PedidoRepositorio;  
import org.springframework.web.bind.annotation.\*;  
  
import java.util.List;  
  
@RestController  
@RequestMapping("/api/pedidos")  
public class PedidoControlador {  
 private final PedidoRepositorio pedidoRepositorio;  
  
 public PedidoControlador(PedidoRepositorio pedidoRepositorio) {  
 this.pedidoRepositorio = pedidoRepositorio;  
 }  
  
 @GetMapping  
 public List<Pedido> listarPedidos() {  
 return pedidoRepositorio.findAll();  
 }  
  
 @PostMapping  
 public Pedido adicionarPedido(@RequestBody Pedido pedido) {  
 return pedidoRepositorio.save(pedido);  
 }  
}

## 6. Adicionar Dados de Teste

Implemente o ficheiro DatabaseLoader.java:

package com.example.demo;  
  
import com.example.demo.model.Cliente;  
import com.example.demo.model.Pedido;  
import com.example.demo.repository.ClienteRepositorio;  
import com.example.demo.repository.PedidoRepositorio;  
import org.springframework.boot.CommandLineRunner;  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.Configuration;  
  
@Configuration  
public class DatabaseLoader {  
 @Bean  
 CommandLineRunner initDatabase(ClienteRepositorio clienteRepositorio, PedidoRepositorio pedidoRepositorio) {  
 return args -> {  
 Cliente cliente1 = new Cliente("João Silva", "joao.silva@example.com");  
 clienteRepositorio.save(cliente1);  
 pedidoRepositorio.save(new Pedido("Compra de teclado", cliente1));  
 };  
 }  
}

## 7. Testar a Aplicação

1. Execute a aplicação.
2. Acesse os seguintes endpoints no Postman:
   * **Listar clientes**: GET http://localhost:8080/api/clientes
   * **Adicionar cliente**: POST http://localhost:8080/api/clientes com um corpo JSON:
   * {  
      "nome": "Maria Oliveira",  
      "email": "maria.oliveira@example.com"  
     }