### Exercício 11.2 – Testes Unitários de Repositório e Serviço

**Objetivo:** Implementar um sistema simples de gestão de clientes com testes unitários para repositórios e serviços, utilizando **H2** para persistência em memória e **Mockito** para mockar dependências.

### Parte 1: Configuração do Projeto no Spring Initializr

1. **Acessar o Spring Initializr:**
   * Vá ao site [Spring Initializr](https://start.spring.io/).
2. **Configurar o projeto:**
   * **Project:** Maven
   * **Language:** Java
   * **Spring Boot:** Última versão estável
   * **Group:** com.example
   * **Artifact:** demo
   * **Name:** demo
   * **Package Name:** com.example.demo
   * **Packaging:** Jar
   * **Java Version:** 21
3. **Adicionar dependências:**
   * **Spring Web:** Para criar APIs REST.
   * **Spring Data JPA:** Para integração com a base de dados.
   * **H2 Database:** Para usar uma base de dados em memória nos testes.
   * **Spring Boot Starter Test:** Para suporte a testes com JUnit e Mockito.
   * **Spring Boot DevTools (opcional):** Para facilitar o desenvolvimento.
4. **Gerar o projeto:**
   * Clique em **Generate** para descarregar o projeto como um ficheiro ZIP.
5. **Abrir o projeto no VS Code:**
   * Extraia o ficheiro ZIP.
   * No VS Code, clique em **File > Open Folder** e selecione a pasta do projeto.

### Parte 2: Estrutura do Projeto

1. **Criar as pastas necessárias:**
   * No VS Code, crie as seguintes pastas em src/main/java/com/example/demo:
     + entity para a classe Cliente.
     + repository para o repositório Cliente.
     + service para a lógica de negócios.
   * Em src/test/java/com/example/demo, crie:
     + repository para os testes do repositório.
     + service para os testes dos serviços.
2. A estrutura do projeto deverá ficar assim:

src/  
├── main/  
│ ├── java/  
│ │ └── com.example.demo/  
│ │ ├── DemoApplication.java  
│ │ ├── entity/  
│ │ ├── repository/  
│ │ ├── service/  
├── test/  
│ └── java/  
│ └── com.example.demo/  
│ ├── repository/  
│ └── service/

### Parte 3: Implementação do Modelo de Dados

1. **Criar a classe Cliente em entity:**

package com.example.demo.entity;  
  
import jakarta.persistence.Entity;  
import jakarta.persistence.GeneratedValue;  
import jakarta.persistence.GenerationType;  
import jakarta.persistence.Id;  
  
@Entity  
public class Cliente {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 private Long id;  
 private String nome;  
 private String email;  
  
 public Cliente() {}  
  
 public Cliente(String nome, String email) {  
 this.nome = nome;  
 this.email = email;  
 }  
  
 public Long getId() {  
 return id;  
 }  
  
 public void setId(Long id) {  
 this.id = id;  
 }  
  
 public String getNome() {  
 return nome;  
 }  
  
 public void setNome(String nome) {  
 this.nome = nome;  
 }  
  
 public String getEmail() {  
 return email;  
 }  
  
 public void setEmail(String email) {  
 this.email = email;  
 }  
}

### Parte 4: Repositório Cliente

1. **Criar a interface ClienteRepository em repository:**

package com.example.demo.repository;  
  
import com.example.demo.entity.Cliente;  
import org.springframework.data.jpa.repository.JpaRepository;  
  
public interface ClienteRepository extends JpaRepository<Cliente, Long> {  
 Cliente findByEmail(String email);  
}

### Parte 5: Implementar o Serviço

1. **Criar a classe ClienteService em service:**

package com.example.demo.service;  
  
import com.example.demo.entity.Cliente;  
import com.example.demo.repository.ClienteRepository;  
import org.springframework.stereotype.Service;  
  
import java.util.List;  
  
@Service  
public class ClienteService {  
  
 private final ClienteRepository clienteRepository;  
  
 public ClienteService(ClienteRepository clienteRepository) {  
 this.clienteRepository = clienteRepository;  
 }  
  
 public List<Cliente> listarClientes() {  
 return clienteRepository.findAll();  
 }  
  
 public Cliente buscarPorEmail(String email) {  
 return clienteRepository.findByEmail(email);  
 }  
  
 public Cliente salvarCliente(Cliente cliente) {  
 return clienteRepository.save(cliente);  
 }  
}

### Parte 6: Testes do Repositório

1. **Criar ClienteRepositoryTest em test/repository:**

package com.example.demo.repository;  
  
import com.example.demo.entity.Cliente;  
import org.junit.jupiter.api.Test;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.boot.test.autoconfigure.orm.jpa.DataJpaTest;  
  
import static org.assertj.core.api.Assertions.assertThat;  
  
@DataJpaTest  
public class ClienteRepositoryTest {  
  
 @Autowired  
 private ClienteRepository clienteRepository;  
  
 @Test  
 public void testSalvarEObterCliente() {  
 Cliente cliente = new Cliente("João Silva", "joao.silva@example.com");  
 clienteRepository.save(cliente);  
  
 Cliente encontrado = clienteRepository.findByEmail("joao.silva@example.com");  
  
 assertThat(encontrado).isNotNull();  
 assertThat(encontrado.getNome()).isEqualTo("João Silva");  
 assertThat(encontrado.getEmail()).isEqualTo("joao.silva@example.com");  
 }  
}

### Parte 7: Testes do Serviço

1. **Criar ClienteServiceTest em test/service:**

package com.example.demo.service;  
  
import com.example.demo.entity.Cliente;  
import com.example.demo.repository.ClienteRepository;  
import org.junit.jupiter.api.BeforeEach;  
import org.junit.jupiter.api.Test;  
import org.mockito.InjectMocks;  
import org.mockito.Mock;  
import org.mockito.MockitoAnnotations;  
  
import static org.junit.jupiter.api.Assertions.assertEquals;  
import static org.mockito.Mockito.when;  
  
public class ClienteServiceTest {  
  
 @Mock  
 private ClienteRepository clienteRepository;  
  
 @InjectMocks  
 private ClienteService clienteService;  
  
 @BeforeEach  
 public void setup() {  
 MockitoAnnotations.openMocks(this);  
 }  
  
 @Test  
 public void testBuscarPorEmail() {  
 Cliente cliente = new Cliente("João Silva", "joao.silva@example.com");  
 when(clienteRepository.findByEmail("joao.silva@example.com")).thenReturn(cliente);  
  
 Cliente resultado = clienteService.buscarPorEmail("joao.silva@example.com");  
  
 assertEquals("João Silva", resultado.getNome());  
 assertEquals("joao.silva@example.com", resultado.getEmail());  
 }  
}

### Parte 8: Executar os Testes

1. **Executar testes no VS Code:**
   * Clique com o botão direito sobre os ficheiros de teste e selecione **Run Test**.
2. **Confirmar resultados:**
   * Verifique se os testes passam sem erros.