**TALLER 2 Dia 2**

**OBJETIVO: Implementar desde 0 un proyecto**

* **CONFIGURAR POM.XML**

<dependencies>

<!-- Spring Boot Starter Web -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<!-- Spring Boot Starter Data JPA -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<!-- H2 Database -->

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<scope>runtime</scope>

</dependency>

<!-- Spring Boot Starter Test -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

<!-- Mockito -->

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-core</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

* **IMPLEMENTAR LAS CLASES**

**User.java** (Modelo de usuario)

package com.example.model;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

@Entity

public class User {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

// Getters y Setters

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

}

**UserRepository.java** (Interfaz del repositorio)

package com.example.repository;

import com.example.model.User;

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

public interface UserRepository extends JpaRepository<User, Long> {

}

**UserService.java** (Clase de servicio)

package com.example.service;

import com.example.model.User;

import com.example.repository.UserRepository;

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

import org.springframework.stereotype.Service;

import java.util.Optional;

@Service

public class UserService {

@Autowired

private UserRepository userRepository;

public User getUserById(Long id) {

Optional<User> user = userRepository.findById(id);

return user.orElse(null);

}

public User saveUser(User user) {

return userRepository.save(user);

}

public void deleteUser(Long id) {

userRepository.deleteById(id);

}

}

**Escribir pruebas con JUnit y Mockito**

**UserServiceTest.java** (Pruebas unitarias)

package com.example.service;

import com.example.model.User;

import com.example.repository.UserRepository;

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 org.springframework.boot.test.context.SpringBootTest;

import java.util.Optional;

import static org.junit.jupiter.api.Assertions.assertEquals;

import static org.mockito.Mockito.\*;

@SpringBootTest

class UserServiceTest {

@Mock

private UserRepository userRepository;

@InjectMocks

private UserService userService;

@BeforeEach

void setUp() {

MockitoAnnotations.openMocks(this);

}

@Test

void testGetUserById() {

User user = new User();

user.setId(1L);

user.setName("John Doe");

when(userRepository.findById(1L)).thenReturn(Optional.of(user));

User result = userService.getUserById(1L);

assertEquals("John Doe", result.getName());

verify(userRepository, times(1)).findById(1L);

}

@Test

void testSaveUser() {

User user = new User();

user.setName("Jane Doe");

when(userRepository.save(user)).thenReturn(user);

User result = userService.saveUser(user);

assertEquals("Jane Doe", result.getName());

verify(userRepository, times(1)).save(user);

}

@Test

void testDeleteUser() {

doNothing().when(userRepository).deleteById(1L);

userService.deleteUser(1L);

verify(userRepository, times(1)).deleteById(1L);

}

}

Configurar**application.properties**

Añade la configuración de la base de datos H2 src/main/resources/application.properties

spring.datasource.url=jdbc:h2:mem:testdb

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=password

spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

spring.h2.console.enabled=true