

1. Estrutura do Projeto

Visão geral da estrutura de pastas do projeto:

sistema-gerenciamento-usuarios/

```
|
|
|— src/
|   |— main/
|       |— java/
|           |— com/
|               |— exemplo/
|                   |— usermanagement/
|                       |— User.java
|                       |— UserController.java
|                       |— UserService.java
|                       |— UserRepository.java
|                   |— resources/
|                       |— application.properties
|   |— test/
|       |— java/
|           |— com/
|               |— exemplo/
|                   |— usermanagement/
|                       |— UserServiceTest.java
|   |— pom.xml
```

2. Código de Implementação

User.Java (Modelo do Usuario):

```
package com.exemplo.usermanagement;
```

```
import javax.persistence.Entity;
```

```
import javax.persistence.GeneratedValue;
```

```
import javax.persistence.GenerationType;
```

```
import javax.persistence.Id;
```

```
import javax.validation.constraints.Email;
```

```
import javax.validation.constraints.Pattern;
```

```
import javax.validation.constraints.Size;
```

```
@Entity
```

```
public class User {
```

```
    @Id
```

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

```
    private Long id;
```

```
    @Size(min = 1, message = "Name cannot be empty")
```

```
    private String name;
```

```
    @Email(message = "Email should be valid")
```

```
    private String email;
```

```
    @Pattern(regex = "(?=.*\\d)(?=.*[a-zA-Z]).{8,}", message = "Password must be at least 8  
characters long and contain at least one number")
```

```
    private String password;
```

```
// Getters and Setters  
  
}
```

UserRepository.java
(Interface do Repositório):

```
package com.exemplo.usermanagement;  
  
import org.springframework.data.jpa.repository.JpaRepository;  
  
public interface UserRepository extends JpaRepository<User, Long> {  
    boolean existsByEmail(String email);  
    User findByEmail(String email);  
}
```

UserService.java (Lógica de
Negócio):

```
package com.exemplo.usermanagement;  
  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
  
import javax.transaction.Transactional;  
import java.util.Optional;  
  
@Service  
public class UserService {  
  
    @Autowired  
    private UserRepository userRepository;  
  
    @Transactional  
    public User registerUser(String name, String email, String password) {  
        if (userRepository.existsByEmail(email)) {  
            throw new IllegalArgumentException("Email already in use");  
        }  
        User user = new User();  
        user.setName(name);  
        user.setEmail(email);  
        user.setPassword(password);  
        return userRepository.save(user);  
    }  
  
    public User login(String email, String password) {
```

```

        User user = userRepository.findByEmail(email);
        if (user == null || !user.getPassword().equals(password)) {
            throw new IllegalArgumentException("Invalid credentials");
        }
        return user;
    }
}

```

UserController.java (Controlador REST):

```
package com.exemplo.usermanagement;
```

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;
```

```

@RestController
@RequestMapping("/api/users")
public class UserController {

    @Autowired
    private UserService userService;

    @PostMapping("/register")
    public User register(@RequestParam String name, @RequestParam String email,
        @RequestParam String password) {
        return userService.registerUser(name, email, password);
    }

    @PostMapping("/login")
    public User login(@RequestParam String email, @RequestParam String password) {
        return userService.login(email, password);
    }
}

```

application.properties
(Configuração):

```

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

```

3. Testes Unitários

UserServiceTest.java (Testes de Unidade para UserService):

```

package com.exemplo.usermanagement;

import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import static org.mockito.Mockito.*;
import static org.junit.jupiter.api.Assertions.*;

class UserServiceTest {

    private UserRepository userRepository;
    private UserService userService;

    @BeforeEach
    void setUp() {
        userRepository = mock(UserRepository.class);
        userService = new UserService();
        userService.setUserRepository(userRepository);
    }

    @Test
    void testRegisterUserSuccess() {
        User user = new User();
        user.setName("John");
        user.setEmail("john@example.com");
        user.setPassword("password123");

        when(userRepository.existsByEmail(user.getEmail())).thenReturn(false);
        when(userRepository.save(any(User.class))).thenReturn(user);

        User registeredUser = userService.registerUser(user.getName(), user.getEmail(),
user.getPassword());
        assertNotNull(registeredUser);
        assertEquals(user.getEmail(), registeredUser.getEmail());
    }

    @Test
    void testRegisterUserEmailAlreadyExists() {
        User user = new User();
        user.setEmail("john@example.com");

        when(userRepository.existsByEmail(user.getEmail())).thenReturn(true);

        IllegalArgumentException thrown = assertThrows(IllegalArgumentException.class, () ->
{
            userService.registerUser(user.getName(), user.getEmail(), user.getPassword());
});

        assertEquals("Email already in use", thrown.getMessage());
    }
}

```

```
}
```

```
@Test
```

```
void testLoginSuccess() {
```

```
    User user = new User();
```

```
    user.setEmail("john@example.com");
```

```
    user.setPassword("password123");
```

```
    when(userRepository.findByEmail(user.getEmail())).thenReturn(user);
```

```
    User loggedInUser = userService.login(user.getEmail(), user.getPassword());
```

```
    assertNotNull(loggedInUser);
```

```
}
```

```
@Test
```

```
void testLoginFailure() {
```

```
    when(userRepository.findByEmail("john@example.com")).thenReturn(null);
```

```
    IllegalArgumentException thrown = assertThrows(IllegalArgumentException.class, () ->
```

```
{
```

```
        userService.login("john@example.com", "wrongpassword");
```

```
    });
```

```
    assertEquals("Invalid credentials", thrown.getMessage());
```

```
}
```

```
}
```

4. Arquivo pom.xml

(Dependências do

Maven):

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
```

```
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
```

```
http://maven.apache.org/xsd/maven-4.0.0.xsd">
```

```
    <modelVersion>4.0.0</modelVersion>
```

```
    <groupId>com.exemplo</groupId>
```

```
    <artifactId>usermanagement</artifactId>
```

```
    <version>0.0.1-SNAPSHOT</version>
```

```
    <name>usermanagement</name>
```

```
    <description>Projeto para gerenciamento de usuários</description>
```

```
    <dependencies>
```

```
        <dependency>
```

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

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

<build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
        </plugin>
    </plugins>
</build>
</project>
```