

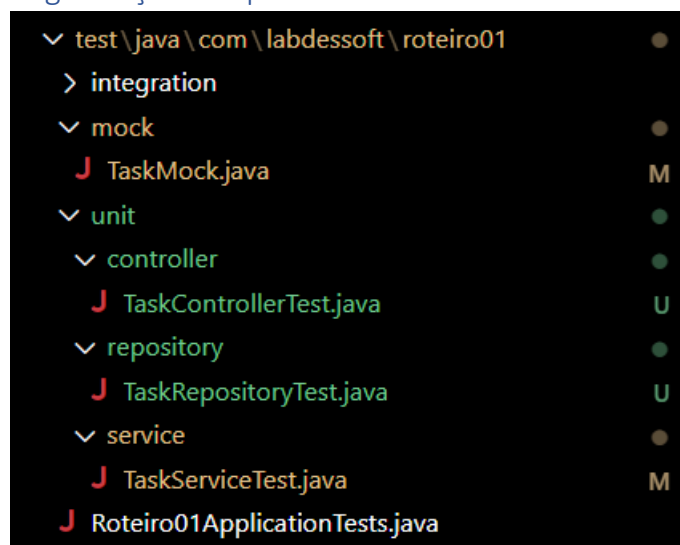
Laboratório 3 – Aula 03

Testes Unitários

O Teste Unitário é uma forma de verificar o funcionamento de pequenos pedaços de código, seu principal objetivo é verificar o quão bem seu código está escrito e, caso não esteja funcionando da maneira esperada, permitir que seja corrigido.

O ideal é que todas as suas classes sejam cobertas pelo teste.

Organização das pastas de teste



Exemplo

```
package com.labdessoft.roteiro01.unit.service;

import com.labdessoft.roteiro01.entity.Task;
import com.labdessoft.roteiro01.mock.TaskMock;
import com.labdessoft.roteiro01.repository.TaskRepository;
import com.labdessoft.roteiro01.service.TaskService;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.DisplayName;
import org.junit.jupiter.api.Test;
import org.junit.jupiter.api.extension.ExtendWith;
import org.junit.platform.runner.JUnitPlatform;
import org.junit.runner.RunWith;
import org.mockito.Mock;
import org.mockito.Mockito;
import org.mockito.junit.jupiter.MockitoExtension;
import org.springframework.data.domain.Page;
```

```

import org.springframework.data.domain.PageRequest;
import org.springframework.data.domain.Pageable;
import org.springframework.data.domain.Sort;

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

@ExtendWith(MockitoExtension.class)
@RunWith(JUnitPlatform.class)
public class TaskServiceTest {

    @Mock
    TaskRepository tasksRepository;

    private TaskService taskService;

    @BeforeEach
    public void setup() {

        taskService = new TaskService(tasksRepository);
        Pageable pageable = PageRequest.of(0, 5, Sort.by(
            Sort.Order.asc("name"),
            Sort.Order.desc("id")));

        Mockito.lenient().when(tasksRepository.findAll(pageable)).thenReturn(TaskMock.createTasks());
    }

    @Test
    @DisplayName("Should return all tasks")
    public void should_list_all_tasks_repository() {

        Pageable pageable = PageRequest.of(0, 5, Sort.by(
            Sort.Order.asc("name"),
            Sort.Order.desc("id")));

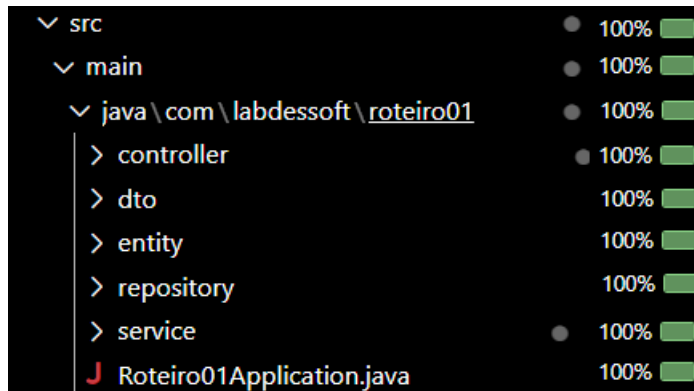
        Page<Task> tasks = taskService.listAll(pageable);

        assertEquals(tasks.getTotalPages(), 1);
        assertEquals(tasks.getNumberOfElements(), 2);
        assertNotNull(tasks);
    }
}

```

Implementação necessária para próxima entrega

- Agora você deverá implementar **todos os testes unitários** necessários na sua aplicação.
 - Neste momento, podemos usar a cobertura (Coverage) para ter certeza se está tudo coberto. O ideal é que todas as classes alcancem o 100%, conforme a imagem abaixo.



▼ src	100%	<div></div>
▼ main	100%	<div></div>
▼ java \ com \ labdessoft \ roteiro01	100%	<div></div>
> controller	100%	<div></div>
> dto	100%	<div></div>
> entity	100%	<div></div>
> repository	100%	<div></div>
> service	100%	<div></div>
J Roteiro01Application.java	100%	<div></div>

Materiais de apoio

- <https://intellij-support.jetbrains.com/hc/en-us/community/posts/115000791190-IntelliJ-does-not-run-Junit5-tests>
- <https://medium.com/@husnapoyraz88/spring-data-jpa-unit-test-repository-layer-9e875390645e>
- <https://www.bezkoder.com/spring-boot-unit-test-jpa-repo-datajpa-test/>
- <https://medium.com/javarevisited/restful-api-testing-in-java-with-mockito-controller-layer-f4605f8ffaf3>
- <https://www.freecodecamp.org/portuguese/news/como-testar-servicos-endpoints-e-repositorios-com-o-springboot/>
- <https://felipeborgesdev.medium.com/testes-unit%C3%A1rios-com-spring-junit-mockito-e-jacoco-5e39dc6bd7fe>
- <https://dev.to/luizleite /testes-realmente-unitarios-no-spring-boot-3gm8>