INFORME DE TESTING - UNION

Curso: Ingeniería de Software 1 (2016701)
Grupo:

- Daniel Alejandro Duitama Correa (dduitama@unal.edu.co)
- Edwin Felipe Pinilla Peralta
- Miguel Angel Martinez Fernandez (miamartinezfe@unal.edu.co)
- Juan Sebastián Umaña Camacho (juumanac@unal.edu.co)

Introducción

UNión es una plataforma de comunicación académica diseñada para la comunidad de la Universidad Nacional de Colombia, que centraliza y optimiza la interacción entre estudiantes y profesores. Su objetivo principal es resolver la dispersión de información en múltiples canales (correos, redes sociales, etc.) mediante un entorno seguro y organizado, enfocado en cursos específicos.

Funcionalidades clave:

- Gestión de cursos: Creación, edición y eliminación de cursos por parte de profesores.
- Mensajería en tiempo real: Chat individual y grupal, notificaciones instantáneas.
- Roles y permisos: Acceso diferenciado para estudiantes y profesores.
- Seguridad integrada: Autenticación mediante cuentas institucionales y protocolos de privacidad.
- Herramientas académicas: Encuestas con fecha límite.

Beneficio principal:

Simplifica la colaboración académica, garantizando que toda la comunicación y recursos estén disponibles en un solo lugar, con un diseño intuitivo inspirado en Google Classroom.

Resumen de Tests Realizados

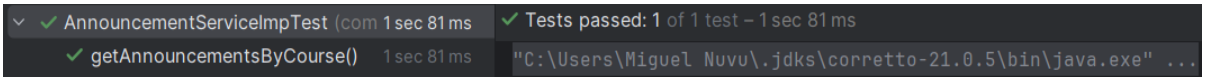
- Miguel Martinez

- Tipo de prueba: Unitaria
- Componente probado: Servicio: AnnouncementService
- Herramienta: JUnit 5 + Mockito

- Código del test:

```java
@Test
  void getAnnouncementsByCourse() {
    Long courseId = 1L;
    Course course = new Course();
    course.setId(courseId);

    Announcement announcement1 = new Announcement();
    announcement1.setId(1L);
    announcement1.setCourse(course);

    Announcement announcement2 = new Announcement();
    announcement2.setId(2L);
    announcement2.setCourse(course);

    List<Announcement> announcements = Arrays.asList(announcement1,
announcement2);

    when(courseService.existsById(courseId)).thenReturn(true);

when(announcementRepository.findAllByCourseId(courseId)).thenReturn(announce
ments);

    List<Announcement> result =
announcementService.getAnnouncementsByCourse(courseId);

    assertNotNull(result);
    assertEquals(2, result.size());
    assertEquals(announcement1, result.get(0));
    assertEquals(announcement2, result.get(1));

    verify(courseService, times(1)).existsById(courseId);
    verify(announcementRepository, times(1)).findAllByCourseId(courseId);
  }
```

- Resultados:



Lecciones Aprendidas y Dificultades

Aprendizajes clave:

- Una de las ventajas de usar inyección por constructor es que en el test podemos incluir solo la anotación @InjectMocks y Mockito se encargara de inyectar las dependencias necesarias.

Dificultades:

- El diseño del test requiere tiempo incluyendo la creación e inicialización de instancias puntuales.

Mejoras futuras:

- Aumentar la cantidad de pruebas para cada servicio, en especial para funciones críticas del sistema.
