Taller 3 - Testing

Realizado por:
Andres Felipe Ramirez Montana
Isabella Garces Acosta
Johan David Lozano Leiva
Maria Luisa Bautista Arango

Ingeniería de sistemas y computación

Universidad Nacional de Colombia
Sede Bogotá
Ingeniería de software I

Profesor:
Oscar Eduardo Alvarez

Introducción breve:

Learnhub es una plataforma web diseñada para centralizar información académica relevante. Facilita la búsqueda de **electivas**, optimizando la experiencia frente al sistema actual del SIA, permite explorar **cursos de Coursera** relacionados con asignaturas o intereses personales, y conecta a estudiantes con **canales de actividades extracurriculares**, fomentando la participación en grupos afines.

En cuanto a sus funcionalidades principales, el acceso es a través del correo institucional. Los usuarios pueden explorar electivas, cursos y canales mediante sistemas de búsqueda eficientes. Las electivas cuentan con filtros intuitivos para una selección más ágil. Los cursos de Coursera incluyen detalles como calificación y enlace de inscripción. En los canales, los estudiantes pueden unirse, ver información relevante y comunicarse con otros miembros a través de un chat integrado.

Resumen de los tests

Nombre del integrante: Maria Luisa Bautista Arango

Tipo de prueba realizada: prueba unitaria ya que no se accede a Firebase ni a otras dependencias externas para hacer el test.

Descripción breve del componente probado: Se probó el método `buscarPorCreditosTest` de la clase `MateriaRepository`, el cual filtra materias en una lista en memoria según la cantidad de créditos.

Herramienta o framework usado:

- **JUnit 5** (para ejecutar las pruebas).
- **Spring Boot Test** (para que la prueba sea reconocida dentro del contexto de Spring)

Screenshot del código del test:
Este es el test realizado

```java
@SpringBootTest  mbautistaa02 *
class LearnHubApplicationTests {


    @Test  new *
    public void testBuscarPorCreditos() throws ExecutionException, InterruptedException {
        // Datos de prueba
        Materia materia1 = new Materia();
        materia1.setCodigo("ART101");
        materia1.setNombre("Apreciación e historia del arte");
        materia1.setCreditos(3);

        Materia materia2 = new Materia();
        materia2.setCodigo("FIS101");
        materia2.setNombre("Objetos astrofísicos");
        materia2.setCreditos(3);

        Materia materia3 = new Materia();
        materia3.setCodigo("len101");
        materia3.setNombre("Lenguaje de señas");
        materia3.setCreditos(2);

        List<Materia> materias = Arrays.asList(materia1, materia2, materia3);

        MateriaRepository materiaRepository = new MateriaRepository();
        List<Materia> result = materiaRepository.buscarPorCreditosTest( creditos: 3, materias);

        // Imprimimos para depurar
        System.out.println("Materias encontradas: " + result.size());

        // Validamos
        assertEquals( expected: 2, result.size(),  message: "Deberían devolverse 2 materias con 3 créditos");
    }

}
```

Y este es el método sobre el que se realizó el test:

```java
public List<Materia> buscarPorCreditosTest(int creditos, List<Materia> materias)  2 usages  new *
        throws ExecutionException, InterruptedException {
    List<Materia> result = new ArrayList<>();
    for (Materia materia : materias) {
        if (materia.getCreditos() == creditos) {
            result.add(materia);
            System.out.println("la electiva "+ materia.getNombre()+" tiene " +  creditos+  " creditos");
        }
    }
    return result;
}
```

Resultado de la ejecución: Screenshot del test ejecutado y su salida:

```
la electiva Apreciación e historia del arte tiene 3 creditos
la electiva Objetos astrofísicos tiene 3 creditos
Materias encontradas: 2
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 12.18 s -- in
com.example.learnhub.LearnHubApplicationTests
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  22.572 s
[INFO] Finished at: 2025-02-21T23:32:23-05:00
[INFO] ------------------------------------------------------------------------
```

Nombre del integrante: Isabella Garcés Acosta

Tipo de prueba realizada: Prueba Unitaria, porque no requiere de interacción con otros módulos ni con la base de datos real, y por lo tanto no depende de una lista real.

Descripción breve del componente probado: Se probó el método `buscarPorArea` de la clase `CanalRepository`, el cual filtra los canales en una lista en memoria según el área/categoría en el que se clasifican.

Herramienta o framework usado:

- **JUnit 5** (para ejecutar las pruebas).
- **Mockito** (Para simular las dependencias)
- **Spring Boot Test** (para que la prueba sea reconocida dentro del contexto de Spring)

Screenshot del código del test:

- Test realizado

```
@Mock
private CanalRepository canalRepository; // Mockeamos el repositorio
no usages
@InjectMocks
private LearnHubApplicationTests testInstance; // Clase bajo prueba
new *
@Test
public void testBuscarPorArea() throws ExecutionException, InterruptedException {

    // Datos de prueba
    Canal canal1 = new Canal( id: "1",  nombre: "Minería de Datos",  area: "Tecnología",  capacity: 100,  currentSize: 50,  descripcion: "Seminario sobre minería de datos");
    Canal canal2 = new Canal( id: "2",  nombre: "Biología Marina",  area: "Ciencia",  capacity: 80,  currentSize: 40,  descripcion: "Investigación de la vida marina");
    Canal canal3 = new Canal( id: "3",  nombre: "Criptografía",  area: "Tecnología",  capacity: 150,  currentSize: 75,  descripcion: "Todo sobre criptografía");

    List<Canal> canales = Arrays.asList(canal1, canal2, canal3);
    String area = "Tecnología";

    // Simular el comportamiento del repositorio
    when(canalRepository.buscarPorArea(area)).thenReturn(
            canales.stream().filter(c -> c.getArea().equalsIgnoreCase(area)).toList()
    );

    List<Canal> resultado = canalRepository.buscarPorArea(area);

    // Imprimir resultados
    System.out.println("Cantidad de canales en " + area + ": " + resultado.size());
    System.out.println("Nombres de los canales: " + resultado.stream().map(Canal::getNombre).toList());

    // Assert
    assertEquals( expected: 2, resultado.size(),  message: "Deberían devolverse 2 canales de Tecnología");
}
```

- Método sobre el que se realizó el test:

```java
public List<Canal> buscarPorArea(String area) throws ExecutionException, InterruptedException {
    Firestore dbFirestore = FirestoreClient.getFirestore();
    CollectionReference collectionReference = dbFirestore.collection(COLLECTION_NAME);

    // Obtener todos los documentos
    ApiFuture<QuerySnapshot> future = collectionReference.get();
    List<QueryDocumentSnapshot> documents = future.get().getDocuments();

    // Normalizar el área de búsqueda
    String areaNormalized = StringUtils.removeTildes(area.toLowerCase());

    // Filtrar los canales por área
    List<Canal> canales = new ArrayList<>();
    for (QueryDocumentSnapshot document : documents) {
        Canal canal = document.toObject(Canal.class);
        String canalAreaNormalized = StringUtils.removeTildes(canal.getArea() != null ? canal.getArea().toLowerCase() : "");

        if (canalAreaNormalized.equals(areaNormalized)) {
            canales.add(canal);
        }
    }

    return canales;
}
```

Resultado de la ejecución: Screenshot del test ejecutado y su salida:

```
Cantidad de canales en Tecnología: 2
Nombres de los canales: [Minería de Datos, Criptografía]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 10.05 s -- in com.example.learnhub.LearnHubApplicationTests
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  17.925 s
[INFO] Finished at: 2025-02-28T01:10:38-05:00
[INFO] ------------------------------------------------------------------------
PS C:\Users\LENOVO\Repositorio_grupal-Ingenieria_de_software_1-2024-2\Proyecto\back\learnHub>
```

Nombre de Integrante: Andres Felipe Ramirez Montaña

Tipo de prueba realizada: Prueba de test unitaria para la prueba de si el método `testEstudianteJsonToEstudiante` de la clase `EstudianteRepository` convierte correctamente una entidad Estudiante. Además tiene en cuenta si vienen datos nulos.

Herramienta o framework usado:

- **JUnit 5** (para ejecutar las pruebas).
- **Spring Boot Test** (para que la prueba sea reconocida dentro del contexto de Spring)

Screenshot del código del test:

- **Text 1:**

```java
@SpringBootTest  mbautistaa02 *
class LearnHubApplicationTests {


    @Test  new *
    public void textToStudent() {
        ObjectMapper mapper = new ObjectMapper();
        EstudianteRepository estudianteRepository = new EstudianteRepository();
        Map<String, Object> data1 = new HashMap<>();
        data1.put("hash", "abc123-xyz789");
        data1.put("correo", "maria.gomez@example.com");
        data1.put("give_name", "Maria");
        data1.put("family_name", "Gomez");
        data1.put("picture", "url_de_perfil_maria");


        Estudiante estudiante = mapper.convertValue(data1, Estudiante.class);
        Estudiante EstudiantePrueba = estudianteRepository.testEstudianteJsonToEstudiante(data1);
        assertEquals(estudiante, EstudiantePrueba, message: "Los objetos no son iguales");
    }
```

- **Text 2 (Para valor nulo):**

```java
    @Test  new *
    public void testComparacionMapNull() {
        ObjectMapper mapper = new ObjectMapper();
        EstudianteRepository estudianteRepository = new EstudianteRepository();
        Map<String, Object> data1 = new HashMap<>();
        data1.put("hash", "abc123-xyz789");
        data1.put("correo", "maria.gomez@example.com");
        data1.put("give_name", "Maria");
        data1.put("family_name", "Gomez");
        data1.put("picture",null);

        Estudiante estudiante = mapper.convertValue(data1, Estudiante.class);
        Estudiante test = estudianteRepository.testEstudianteJsonToEstudiante(data1);
        assertEquals(estudiante, test,  message: "    Los objetos no son iguales");
    }
```

- **Método:**

```java
public Estudiante testEstudianteJsonToEstudiante(Map<String,Object> data) {  2 usages  new *

    Estudiante estudiante1 = Estudiante.builder()
            .hash(data.get("hash") != null ? data.get("hash").toString() : null)
            .correo(data.get("correo") != null ? data.get("correo").toString() : null)
            .give_name(data.get("give_name") != null ? data.get("give_name").toString() : null)
            .family_name(data.get("family_name") != null ? data.get("family_name").toString() : null)
            .picture(data.get("picture") != null ? data.get("picture").toString() : null)
            .build();

    return estudiante1;
}
```

Resultado de la ejecución: Screenshot del test ejecutado y su salida:

```
seconds (process running for 4.282)
Java HotSpot(TM) 64-Bit Server VM warning: Sharing is only supported for boot loader classes because bootstrap classpath has been appended
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 4.498 s -- in com.example.learnhub.LearnHubApplicationTests
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  8.493 s
[INFO] Finished at: 2025-02-27T23:57:42-05:00
[INFO] ------------------------------------------------------------------------
PS C:\Users\ANDRES FELIPE\IdeaProjects\Proyecto Ingesoft\isjoluan\Proyecto\back\learnHub>
```

Nombre del integrante: Johan David Lozano Leiva

Tipo de prueba realizada: Revisa si los componentes necesarios se renderizar.

Descripción breve del componente probado: Se probó el componente “Index.astro”, donde debe estar la página principal con varias secciones clave.

Herramienta o framework usado:

- **Vitest** (para testear los componentes ui)

Screenshot del código del test:
Este es el test realizado

MainPage.test.ts U | ClubsCard.astro M | CourseraCard.astro | index.astro M | vitest.config.ts U

tests > MainPage.test.ts > test("Verifica la estructura general del Layout") callback

```ts
import { experimental_AstroContainer as AstroContainer } from 'astro/container';
import { expect, test } from 'vitest';
import MainPage from "../src/pages/index.astro"; // Ajusta la ruta si es necesario


test("Renderiza la página principal con todos los componentes esperados", async () => {
  const container = await AstroContainer.create();
  const result = await container.renderToString(MainPage);

  // Verificar que el componente SearchSection está presente
  expect(result).toContain('Encontrar electivas nunca fue tan fácil');

  // Verificar que los tabs están presentes
  expect(result).toContain('id="default-styled-tab"');
  expect(result).toContain('Electivas');
  expect(result).toContain('Coursera');
  expect(result).toContain('Grupos');

  // Verificar que todas las secciones de contenido de tabs existen
  expect(result).toContain('id="styled-sia"');
  expect(result).toContain('id="styled-coursera"');
  expect(result).toContain('id="styled-settings"');
  expect(result).toContain('id="styled-clubs"');
});
```

MainPage.test.ts U | ClubsCard.astro M | CourseraCard.astro | index.astro M | vitest.config.ts U

tests > MainPage.test.ts > test("Verifica la estructura general del Layout") callback

```ts
test("Verifica la estructura general del Layout", async () => {
  expect(result).toContain('<html');
  expect(result).toContain('</html>');

  // Si el Layout incluye navbar, podemos verificarlo
  expect(result).toContain('LearnHub');
});

test("Verifica que las secciones específicas están renderizadas", async () => {
  const container = await AstroContainer.create();
  const result = await container.renderToString(MainPage);

  // Verificar componentes específicos
  expect(result).toContain('SIASection');
  expect(result).toContain('CourseraSection');
  expect(result).toContain('ClubsSection');
});
```

Resultado de la ejecución: Screenshot del test ejecutado y su salida:

PROBLEMS 5 | OUTPUT | DEBUG CONSOLE | TERMINAL | PORTS | CODE REFERENCE LOG ... node

```
√ tests/MainPage.test.ts (3 tests) 122ms
  √ Renderiza la página principal con todos los componentes esperados
  √ Verifica la estructura general del Layout
  √ Verifica que las secciones específicas están renderizadas

 Test Files  1 passed (1)
      Tests  3 passed (3)
   Start at  09:04:22
   Duration  2.13s
```

Lecciones aprendidas y dificultades:

Maria Luisa Bautista:
Al momento de crear el test no tenía del todo claro en donde se creaba este y me estaba funcionando en todos los casos (Incluso cuando le mandaba un fail inevitable).
Después descubrí que había una clase especial para los tests en spring boot y ahí estaba creado un test que era el que me decía que estaba pasando, así que ya cuando moví mi código del test a este apartado, me empezó a funcionar correctamente. Además mi intención inicial era realizar la prueba directamente con los elementos de la base de datos, pero esto me dio varios errores que no logré solucionar y por esto lo hice con datos de prueba. Me pareció muy interesante la creación de los tests en este framework y aprendí sobre el uso de Junit5 y Spring boot test.

Isabella Garcés Acosta

Inicialmente me costó un poco de trabajo entender la lógica que debía tener el test de tal manera que no necesitara de conectarse directamente a la base de datos para cumplir con el objetivo del método principal. Sin embargo con la ayuda de los frameworks usados se me hizo más fácil completar el test y entender así mismo como este funcionaba.

Adicionalmente me costó un poco distinguir si se trataba de una prueba unitaria o integrada, puesto que el framework simulaba la dependencia del método, llegué a considerar que la existencia de esa dependencia lo hacía una prueba unitaria,pero en vista de que es una dependencia simulada y no real, la interacción con los otros módulos no se ve evaluada, sino únicamente la lógica del método como tal.

Andres Felipe Ramirez Montaña:

Al principio, me interesaba mucho probar si una función realmente convertía bien los datos nulos en un JSON. Quería asegurarme de que el código manejara correctamente estos valores sin causar errores. Sin embargo, entender cómo escribir pruebas en Spring me pareció complicado al inicio. No solo era cuestión de validar los resultados, sino también de armar bien los datos para que el test fuera efectivo y, al mismo tiempo, evitar errores como un `NullPointerException`.

Con el tiempo, fui comprendiendo mejor la lógica detrás de los tests y cómo estructurarlos para que realmente sirvieran para detectar fallos. Ahora veo que testear no solo ayuda a comprobar que el código funciona, sino que también me obliga a escribirlo de manera más robusta y preparada para distintos escenarios.

Johan David Lozano Leiva

El proyecto actualmente es relativamente simple, por lo que los tests no son tan necesarios, sin embargo, conforme la complejidad de la app aumente, serán más importantes. Ya sea para probar si un componente efectivamente funciona, se renderiza sin problemas ya sea de nulos, de lógica o cualquier otro.
La documentación de Astro para tests es horrible y no pude sacar mucho con ella, tuve que leer bastantes posts en foros y usar gpt.