Práctica y Trabajo Tutelado de Programación Avanzada

3º Curso – Grado en Ingeniería Informática – Ingeniería del Software
Curso académico 2021-2022

1 Introducción

La práctica y el trabajo tutelado de la asignatura "Programación Avanzada" girarán en torno al diseño e implementación de una sencilla aplicación web SPA de subastas. El apartado 2 especifica la funcionalidad de la práctica. El apartado 3 especifica la funcionalidad del trabajo tutelado. Finalmente, el apartado 4 contiene la normativa y evaluación de la práctica.

2 Práctica

2.1 Modelo de información

Un **usuario** se registra en el subastador especificando su seudónimo (userName), contraseña, nombre, apellidos y dirección de correo electrónico.

Un usuario anuncia un **producto** especificando: nombre, descripción, número de minutos que durará la puja (en un caso real podrían ser horas o días, pero ello dificultaría la evaluación de la práctica), el precio de salida, información de envío y la categoría a la que pertenece. La información de envío contiene una descripción textual proporcionada por el usuario vendedor para especificar el medio o medios por los que puede enviar el producto, la duración aproximada, etc. (e.g. "envío por correo postal, dentro del territorio nacional, con plazo de entrega de una semana apróx."). Cada producto pertenece a una **categoría** ("Libros", "Música", "Películas", etc.). El sistema de categorías será plano, es decir, las categorías no tendrán subcategorías.

Un usuario puede realizar una **puja** sobre un producto cuyo periodo de puja todavía no haya terminado. Para realizar la puja, el usuario tiene que introducir la cantidad máxima que está dispuesto a pagar. El subastador le indica la cantidad mínima que debe introducir. Si todavía no existen pujas por el producto, el subastador indicará que la cantidad mínima será el precio de salida (es decir, se puede pujar con una cantidad igual o superior al precio de salida). En otro caso, el subastador indicará que la cantidad mínima debe ser estrictamente mayor que el precio actual. Cada vez que un usuario realiza una puja por un producto, el subastador calculará el "precio actual" de cada producto y el usuario ganador de la puja usando un sistema de incrementos automáticos. El sistema de incrementos automáticos se puede especificar examinando los siguientes ejemplos:

- **Ejemplo 1** (Tabla 1). *Un vendedor anuncia un producto con precio de salida de 10 euros. A continuación, un comprador desea pujar por ese producto. El subastador le indica que debe pujar, al menos, con 10 euros. El comprador especifica una puja máxima de 12 euros. En este momento, el precio del*

producto continúa siendo de 10 euros. Cuando vence el plazo de la puja, y asumiendo que no hubo otros pujadores, el único pujador resulta el ganador, que adquiere el producto por 10 euros.

	valor puja	precio actual	ganador
vendedor	10 €	10 €	
compradorA	12 €	10 €	*compradorA*

Tabla 1. Ejemplo 1.

- **Ejemplo 2** (Tabla 2). *Un vendedor anuncia un producto con precio de salida de 10 euros. A continuación, el comprador A desea pujar por ese producto. El subastador le indica que debe pujar, al menos, con 10 euros. El comprador A especifica una puja máxima de 12 euros. En este momento, el precio del producto continúa siendo de 10 euros. Otro comprador, B, desea pujar por el mismo producto. El sistema le indica que debe pujar con más de 10 euros. El comprador B especifica una puja máxima de 11 euros. En este momento, el precio del producto sube a 11,5 euros, dado que existe otro comprador (A) que está dispuesto a pagar más.* ***Obsérvese que se ha usado un incremento de 0,5 euros****. Cuando vence el plazo de la puja, y asumiendo que el comprador B no realiza una nueva puja y que ningún otro lo hace, el pujador A resulta ganador, y adquiere el producto por 11,5 euros*.

	valor puja	precio actual	ganador
vendedor	10 €	10 €	
compradorA	12 €	10 €	*compradorA*
compradorB	11 €	11,5 €	*compradorA*

Tabla 2. Ejemplo 2.

- **Ejemplo 3** (Tabla 3). *Un vendedor anuncia un producto con precio de salida de 10 euros. A continuación, el comprador A desea pujar por ese producto. El subastador le indica que debe pujar, al menos, con 10 euros. El comprador A especifica una puja máxima de 12 euros. En este momento, el precio del producto continúa siendo de 10 euros. Otro comprador, B, desea pujar por el mismo producto. El sistema le indica que debe pujar con más de 10 euros. El comprador B especifica una puja máxima de 14 euros. En este momento, el precio del producto sube a 12,5 euros, dado que el comprador B está dispuesto a pagar más. Cuando vence el plazo de la puja, y asumiendo que el comprador A no realiza una nueva puja y que ningún otro lo hace, el pujador B resulta ganador, y adquiere el producto por 12,5 euros*.

	valor puja	precio actual	ganador
vendedor	10 €	10 €	
compradorA	12 €	10 €	*compradorA*
compradorB	14 €	12,5 €	*compradorB*

Tabla 3. Ejemplo 3.

En los anteriores ejemplos, es importante observar que:

- Los incrementos automáticos en el precio de los productos son de 0,5 euros.
- El sistema de incrementos no se aplica cuando el nuevo valor del producto superaría la máxima puja. En ese caso, el nuevo valor será precisamente el de la máxima puja.
- Cuando exista más de una puja ganadora, el producto será adjudicado a la puja más antigua.
- Para evitar que un comprador "mal intencionado" intente subir artificialmente el precio actual de un producto, el subastador nunca permite que un usuario conozca las pujas máximas que han especificado el resto de compradores (dado que el comprador "mal intencionado" podría especificar una cantidad ligeramente inferior a la puja máxima, incrementando el precio actual al máximo y sabiendo que él no va a ganar). Por el mismo motivo, cuando se realiza una puja, el subastador no permite que un usuario puje con una cantidad igual al precio actual del producto si existe al menos una puja por ese producto (dado que el comprador "mal intencionado" podría especificar una cantidad igual al precio actual del producto, quizás subiendo en 0,5 unidades el precio y sabiendo que él no va a ganar porque pujó más tarde; el comprador "mal intencionado" podría incluso repetir esta operación varias veces para subir al máximo el precio).

2.2 Interacción con el usuario

A continuación, se detalla la funcionalidad que el subastador deberá ofrecer al usuario:

- **Registro de usuarios**. Debe permitir registrar nuevos usuarios, así como permitir cambiar la información básica de registro.
- **Autenticación y salida**. Un usuario se autenticará indicando su seudónimo y contraseña. El usuario podrá salir explícitamente del subastador.
- **Anunciar (insertar) un producto**. Un usuario autenticado podrá anunciar (insertar) productos. Una vez insertado un producto en el subastador, no será posible modificarlo.

- **Búsquedas de productos**. Cualquier usuario (autenticado o no) podrá buscar productos **(a los que todavía no les haya vencido su plazo de puja)** por palabras clave del nombre del producto y la categoría (mediante un desplegable). Así, por ejemplo, si existe un producto con nombre "Canon PowerShot SX50 HS", deberá aparecer cuando se busca "canon sx50", "sx50 CAN", etc., es decir, las palabras clave tienen que estar **todas** contenidas en el nombre del producto como palabras o parte de palabras, sin distinguir entre mayúsculas y minúsculas, y en cualquier orden. Si el usuario especifica la categoría, la búsqueda se restringirá a los productos de dicha categoría. Si el usuario no especifica palabras clave ni categoría, se mostrarán todos los productos. Los productos que aparecen tras el resultado de una búsqueda se visualizan mostrando: nombre de la categoría, nombre del producto, precio actual y tiempo (en minutos) que resta para que venza el plazo de puja. Un usuario (autenticado o no) podrá hacer clic sobre el nombre de un producto para ver los detalles del mismo, que incluyen: nombre de la categoría, nombre del producto, descripción, seudónimo del vendedor, fecha (día, mes y año) y hora (hora y minuto) en la que se anunció, tiempo (en minutos) que resta para que venza el plazo de la puja (o una indicación de que terminó el periodo de puja), precio de salida, precio actual e información de envío.

- **Realización de una puja**. Si el usuario está autenticado, la página que muestra los detalles de un producto incluirá un formulario para poder pujar por él si el periodo de puja todavía está abierto. Si la puja se ha realizado con éxito, la información del producto se refrescará y se indicará si perdió o va ganando.

- **Consulta de las pujas realizadas**. Un usuario autenticado podrá consultar las pujas que lleva realizado a lo largo del tiempo, mostrándose primero las pujas más recientes. Por cada puja se mostrará la fecha (día, mes y año) y hora (hora, minuto y segundo) en la que se hizo, el nombre del producto (con un enlace que permite ver su información detallada), cantidad máxima que el usuario estaba dispuesto a pagar y una indicación del estado de la puja: perdedora (había una puja mejor), ganando (el periodo de puja todavía está abierto) o ganadora (periodo de puja ya finalizó).

- **Consulta de los productos anunciados**. Un usuario autenticado podrá visualizar los productos que ha anunciado a lo largo del tiempo, mostrando primero los que les falta más tiempo para que concluya su periodo de puja. Por cada producto, se mostrará: su nombre (con un enlace que permite ver su información detallada), su precio actual, el tiempo que resta para que venza el plazo de puja (o una indicación de que ya venció) y la dirección de correo electrónico del comprador que ganó o va ganando (o una indicación de que no existe ninguna puja). La dirección de correo electrónico se presupone que la usaría el vendedor para contactar con el ganador de la puja.

Para no alargar innecesariamente la práctica, la información sobre categorías se introducirá mediante sentencias `INSERT INTO` en el script de inserción de datos (`backend/src/sql/2-MySQLCreateData.sql`).

2.3 Pruebas de integración de la capa modelo

Para verificar el correcto funcionamiento de la capa modelo, se implementarán pruebas automatizadas de los servicios de la capa modelo. **Será necesario implementar los casos de prueba más representativos (tanto de éxito como de error) de cada caso de uso**.

2.4 Internacionalización

La aplicación debe tratar la internacionalización de los mensajes, fechas y cantidades monetarias.

3 Trabajo tutelado

El objetivo del trabajo tutelado es introducir al alumno en el desarrollo de pruebas automatizadas contra la interfaz de usuario, comúnmente conocidas con el nombre "End-To-End (E2E) Testing". En el caso de una aplicación web, en este tipo de pruebas, cada caso de prueba arranca una instancia del navegador, reproduce automáticamente los pasos que realiza el usuario final para ejecutar un caso de uso y comprueba que éste funciona correctamente, verificando que en la página que visualiza el navegador aparece la información esperada. Por ejemplo, en pa-shop, el caso de prueba correspondiente al caso de uso "login" podría constar de los siguientes pasos (a continuación, se asume que en la base de datos ya existe el usuario "test" con contraseña "test"):

- Acceder a la URL http://localhost:3000.
- Hacer clic en el enlace de login.
- Introducir el seudónimo de usuario ("test") y contraseña ("test") en los campos del formulario de login.
- Hacer clic en el botón de autenticación.
- Localizar el enlace asociado al seudónimo del usuario en la cabecera.
- Comprobar que el texto del enlace es igual al seudónimo del usuario ("test").

En este trabajo tutelado se usará:

- La versión Java de Selenium WebDriver (https://www.selenium.dev/documentation/webdriver). Esta librería permite arrancar un navegador, cargar una página, realizar interacciones sobre ella (e.g. rellenar campos de un formulario, hacer clic en botones y enlaces, etc.), así como acceder a los elementos HTML que muestra la página. Conceptualmente, WebDriver sigue un diseño similar al de la API de JDBC. WebDriver ofrece una API independiente del navegador y para poder usarla con un navegador concreto se necesita un "driver", es decir una librería que implementa la API de WebDriver para ese navegador.

- JUnit 5 como librería de desarrollo de pruebas automáticas. Cada caso de prueba (anotado con `@Test`) usará Selenium WebDriver para implementar las interacciones que realiza el usuario con el navegador y el acceso a los elementos de la página. Para realizar las aserciones, usará JUnit.

El trabajo tutelado es INDIVIDUAL y OPCIONAL. Concretamente, se proponen 4 tareas, numeradas de 0 a 3. La tarea 0 debe ser la primera en realizarse y se hará colaborativamente entre todos los miembros que deseen hacer el trabajo tutelado. Posteriormente, cada miembro que desee hacer el trabajo tutelado escogerá una de las tres tareas restantes y la completará individualmente.

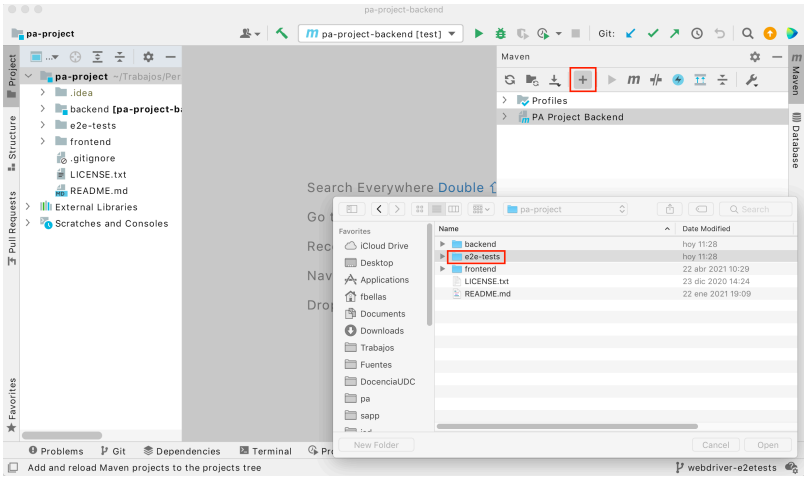
Finalmente, es importante tener en cuenta que para que el trabajo tutelado se considere como válido, el código Java de las pruebas E2E NO puede generarse automáticamente con Selenium IDE (otro producto de Selenium, orientado a la creación rápida de scripts que automatizan secuencias de navegación).

3.1 Tarea 0: creación del esqueleto del proyecto de pruebas E2E

Esta tarea debe ser la primera en implementarse y su objetivo es crear la base necesaria para la implementación del resto de tareas. Consta de los siguientes pasos:

- Crear el proyecto Maven que contendrán las pruebas E2E. El proyecto debe corresponder a una carpeta hermana de las carpetas “backend” y “frontend". Para ello, puede usarse el arquetipo maven-archetype-quickstart (https://maven.apache.org/archetypes/maven-archetype-quickstart) desde el directorio raíz de la práctica, introduciendo “es.udc.paproject.e2etests” como groupId, “e2e-tests” como artifactId y aceptando el resto de valores propuestos por defecto. Este comando debería crear la carpeta (proyecto Maven) “2e2-tests”, hermana de las carpetas “backend” y “frontend".

- Agregar el proyecto Maven “2e2-tests” al proyecto de la práctica en IDEA. En la pestaña “Maven”, hacer clic en “+” y seleccionar la carpeta “2e2-tests”:



- 2e2-tests/pom.xml: por defecto asume Java 1.7 y JUnit 4. Debe modificarse para que use Java 11 y JUnit 5 (la dependencia de JUnit 5 debe especificarse con groupId=org.junit.jupiter, artifactId=junit-jupiter-engine, version=5.7.1).

- 2e2-tests/pom.xml: incluir la dependencia de Selenium WebDriver (https://www.selenium.dev/documentation/webdriver/getting_started/install_library).

- Instalar un driver para Chrome. Existen varias posibilidades para implementar este paso. En particular, se usará una librería de gestión de drivers, más concretamente, WebDriverManager. Para ello, basta incluir la dependencia de esta librería (https://bonigarcia.dev/webdrivermanager/#setup) en 2e2-tests/pom.xml.

- 2e2-tests/src/test/java/es/udc/paproject/e2etests/AppTest: incluir el test de ejemplo básico de uso de WebDriverManager que figura en https://bonigarcia.dev/webdrivermanager/#driver-management. Al hacer copy-paste del ejemplo a AppTest es importante observar que:

    - La clase debe seguir llamándose AppTest (en lugar de ChromeTest).
    - El paquete de la clase AppTest debe seguir siendo "es.udc.paproject.e2etests".
    - El ejemplo usa la librería "assertj". Para no tener que incluir una dependencia de esta librería, basta hacer un par de cambios:

```
~~import static org.assertj.core.api.Assertions.assertThat;~~
import static org.junit.jupiter.api.Assertions.*;

...

~~assertThat(title).contains("Selenium WebDriver");~~
assertTrue(title.contains("Selenium WebDriver"));

...
```

    - Seguramente será necesario recargar el proyecto Maven "e2e-tests": pestaña "Maven" → e2e-tests → botón derecho → Reload Project.

- Probar a ejecutar el ejemplo:

```
cd e2e-tests
mvn test
```

- AppTest: renombrar el método "test" a "testLogin" e implementar el caso de prueba del caso de uso "login" descrito anteriormente. Para ello, es necesario tener en cuenta que:

    - Por sencillez, se usará la base de datos "paproject" para la ejecución de las pruebas de integración. Los datos necesarios para los casos de prueba

se insertarán en la base de datos mediante el script backend/src/sql/2-MySQLCreateData.sql.

- backend/src/sql/2-MySQLCreateData.sql: incluir una sentencia SQL para insertar el usuario "test" con contraseña "test" (obsérvese que la contraseña debe especificarse cifrada en el script SQL). Este es el usuario que se empleará en el caso de prueba.

- Para saber cómo implementar cada uno de los pasos del caso de prueba debe consultarse la documentación de WebDriver (https://www.selenium.dev/documentation/webdriver). Un buen punto de partida puede ser leer completamente https://www.selenium.dev/documentation/webdriver/getting_started/first_script y consultar bajo demanda https://www.selenium.dev/documentation/webdriver/elements.

- Se aconseja modificar el código de los componentes React para que se use el atributo "id" en los elementos HTML que se deseen localizar desde el caso de prueba (`WebElement loginLink = driver.findElement(By.id("loginLink"))`).

- Cuando se desea obtener un elemento HTML desde un caso de prueba usando la API de WebDriver, hay que tener en cuenta que WebDriver puede no ser siempre capaz de devolverlo inmediatamente. Por ejemplo, en el caso de prueba de login, tras hacer clic en el botón de autenticarse, si se intenta acceder al enlace que contiene el userName en la cabecera, es bastante probable que WebDriver no pueda devolverlo inmediatamente porque quizás el backend todavía no envío la respuesta, o si lo hizo, quizás al frontend todavía no le dio tiempo a añadir el enlace. Existen varias estrategias para solucionar este problema. Por sencillez, se aconseja usar la estrategia "implicit wait" (https://www.selenium.dev/documentation/webdriver/waits/#implicit-wait).
  Esta estrategia puede especificarse en el método anotado con `@BeforeEach` (`setupTest`), y así se aplicará a todos los casos de prueba. Con esta estrategia, cuando se le pide un elemento HTML a WebDriver, la llamada correspondiente (e.g. `driver.findElement(By.id("userNameLink"))`) se quedará bloqueada hasta que aparezca el elemento o transcurra el deadline especificado con la estrategia "implicit wait" (un deadline de 10 segundos en la práctica real será más que suficiente).

- Ejecutar los tests:

```
cd backend
mvn sql:execute spring-boot:run
cd frontend
npm start
```

```
cd e2e-tests
mvn test
```

3.2 Tarea 1: prueba automatizada del caso de uso "ver información detallada de un producto"

En esta tarea se añadirá a AppTest un caso de prueba para comprobar el funcionamiento de la visualización de la información detallada de un producto cuando el usuario está autenticado. Concretamente, constará de los siguientes pasos:

- Autenticar al usuario "test". Por sencillez, puede invocarse a "testLogin" (tarea 0) como parte del caso de prueba.
- En el formulario de búsqueda de productos (seguramente en la cabecera de la página de la aplicación), introducir el nombre de un producto que figure en el script SQL de inserción de datos (con un periodo de puja muy grande).
- Hacer clic en el botón de buscar.
- Localizar el enlace asociado al nombre del producto en el área de resultados de la búsqueda y hacer clic sobre él.
- Comprobar que se incluyen todos los valores de los campos esperados (nombre de la categoría, nombre del producto, descripción, etc.), sin comprobar los valores concretos de cada uno. Por ejemplo, si el nombre de la categoría se muestra dentro del elemento `<p id="productCategory" ...>`, hay que comprobar que la página contiene ese elemento.
- Comprobar que incluye el formulario para pujar.
- Comprobar que el valor del campo nombre del producto es del producto buscado.

3.3 Tarea 2: prueba automatizada del caso de uso "anunciar (insertar) un producto"

En esta tarea se añadirá a AppTest un caso de prueba para comprobar el funcionamiento de una inserción correcta de un producto. Concretamente, constará de los siguientes pasos:

- Autenticar al usuario "test". Por sencillez, puede invocarse a "testLogin" (tarea 0) como parte del caso de prueba.
- Hacer clic en el enlace u opción que permite insertar un producto.
- Rellenar el formulario con los datos de un producto que tenga un periodo de puja muy grande.
- Hacer clic en el botón de añadir.
- Comprobar que aparece el mensaje que indica que el producto se añadió correctamente.
- Hacer clic en el enlace u opción que muestra los productos que lleva anunciados el usuario.

- Comprobar que el nombre del primer producto coincide con el del producto añadido.

3.4 Tarea 3: prueba automatizada del caso de uso "pujar por un producto"

En esta tarea se añadirá a AppTest un caso de prueba para comprobar el funcionamiento de una puja correcta. Concretamente, constará de los siguientes pasos:

- Autenticar al usuario "test". Por sencillez, puede invocarse a "testLogin" (tarea 0) como parte del caso de prueba.
- Acceder a la URL de la información detallada de un producto que aparezca en el script SQL de inserción de datos y que tenga un periodo de puja muy grande.
- Especificar una cantidad válida en el formulario de puja.
- Hacer clic en el botón de pujar.
- Comprobar que aparece el mensaje que indica que la puja se ha realizado.
- Hacer clic en el enlace u opción que muestra las pujas realizadas.
- Comprobar que el nombre del producto de la primera puja coincide con el del producto por el que se pujó.

4 Normativa y evaluación

4.1 Composición de los grupos

La práctica se realizará en grupos de 3 personas.

4.2 Distribución de tareas de desarrollo de la práctica

Los miembros de un grupo serán responsables de dividir el desarrollo de la práctica según consideren que es más adecuado para su caso. Dentro de un mismo grupo, distintos miembros pueden contribuir al desarrollo de una misma funcionalidad si lo consideran necesario. Lo único que se exigirá de forma obligatoria es:

- **En cada una de las iteraciones de la práctica será necesario implementar todas las funcionalidades**.

- **Cada miembro del grupo debe contribuir de forma significativa al desarrollo de la práctica. Concretamente, cada miembro debe haber contribuido de forma relevante al desarrollo del backend y al frontend en cada una de sus capas**. Para determinar la contribución de cada alumno a la práctica, el profesor que la corrija utilizará EXCLUSIVAMENTE el grafo de commits del repositorio Git asociado (en caso de dudas podrá concertar una reunión con los miembros que estime necesario).

- **Los mensajes de los commits deberán seguir OBLIGATORIAMENTE el formato que se explica más abajo**.

Los mensajes de los commits seguirán el formato *"[<<etiqueta>>] Mensaje detallado*". La etiqueta identifica el caso de uso al que afecta el commit y el mensaje detallado aporta información específica. Como etiquetas se usarán las siguientes:

- FUNC-1: anunciar (insertar) un producto.
- FUNC-2: buscar productos.
- FUNC-3: ver la información detallada de un producto.
- FUNC-4: pujar por un producto.
- FUNC-5: consultar las pujas realizadas.
- FUNC-6: consultar los productos anunciados.
- VARIOS: commits de aspectos que aplican a **varios** casos de uso.

Ejemplos de commits válidos:

- [VARIOS] Añadida entidad Producto.
- [VARIOS] Añadidas reglas de control de acceso en el backend.
- [VARIOS] Recuperación de la lista de categorías implementada en el backend.
- [FUNC-2] Búsqueda de productos implementada en el backend.
- [FUNC-3] Visualización de la información detallada de un producto en el frontend.
- [FUNC-3] Mejora en la visualización de la información detallada de un producto en el frontend.

4.3 Distribución de tareas de desarrollo del trabajo tutelado

Todos los commits relativos a una tarea deben ser realizados por el alumno que haya elegido esa tarea, excepto la tarea 0. Los mensajes de los commits deben seguir el formato especificado en el apartado 4.2. Como etiquetas se emplearán:

- TT-0: Tarea 0.
- TT-1: Tarea 1.
- TT-2: Tarea 2.
- TT-3: Tarea 3.

4.4 Iteraciones y entregas

Para la realización de la aplicación se seguirá un enfoque basado en iteraciones, de manera que cada iteración incorpora más funcionalidad sobre la anterior, hasta que en la última iteración se termina con un software que implementa toda la funcionalidad. En particular, la aplicación se hará en tres iteraciones:

- **[OBLIGATORIA] Primera iteración**. Se implementará el backend con la funcionalidad especificada en el apartado 2. **Plazo de entrega: 20 de marzo**. El profesor enviará un correo a los miembros de cada grupo con los defectos que haya identificado al analizar el código de su práctica.

- **[OBLIGATORIA] Segunda iteración**. Se corregirán los defectos detectados en la primera iteración que se estimen oportunos (**se sugiere que se corrijan todos aquellos que pueden impactar negativamente en el desarrollo de la segunda iteración**) y se implementará el frontend con la funcionalidad especificada en el apartado 2. Los alumnos que lo deseen pueden presentar también el trabajo tutelado especificado en el apartado 3. **Plazo de entrega: 15 de mayo**.

- **[OPCIONAL] Tercera iteración**. Si no se presentó el trabajo tutelado en la segunda iteración, se puede presentar en la tercera iteración. **Plazo de entrega: 13 de junio**.

4.5 Evaluación

La práctica y el trabajo tutelado se evaluarán de la siguiente manera:

- **Práctica**. Cada una de las dos primeras iteraciones se puntúan de 0 a 5, siendo la nota final de la práctica la suma de ambas. Tal y como especifican las normas de evaluación de la asignatura, se recuerda que en la evaluación de cada una de las dos iteraciones es necesario obtener un mínimo de 2,5 puntos y en caso de no alcanzarse esa nota mínima en alguna de ellas, la nota máxima de la práctica será un 4. Además, para poder presentar la segunda iteración, es necesario haber obtenido la nota mínima en la primera iteración. Dentro de un grupo, la nota puede ser diferente para cada miembro del grupo en función de su contribución y la calidad del diseño e implementación de la parte que haya hecho.

- **Trabajo tutelado**. Puntuación: de 0 a 10 puntos. En este caso no es preciso obtener ninguna nota mínima.

Para la corrección de cada iteración y el trabajo tutelado, el profesor descargará el código del repositorio Git y evaluará su correcto funcionamiento, la calidad del diseño y la calidad del código.

Una práctica copiada significará un suspenso para el grupo que ha dejado copiar y para el que ha copiado; a todos los efectos, no se hará ninguna distinción. Los suspensos por práctica copiada tendrán que realizar una práctica distinta para la segunda oportunidad, que además deberán proponer (y ser aceptada).

Si alguno de los miembros de un grupo no supera la segunda iteración (habiendo superado la primera), pero el resto sí lo hace, el alumno suspenso (o no presentado) deberá desarrollar en solitario para la segunda oportunidad una versión extendida de la práctica.

En la convocatoria correspondiente a la segunda oportunidad se presentará la misma práctica y trabajo tutelado (excepto los suspensos en alguno de los dos casos anteriores), sin posibilidad de entregar la primera iteración: se presentará directamente la versión final de la práctica y del trabajo tutelado.