BATTLESHIP

UNIVERSIDAD
POLITÉCNICA
DE MADRID

Entrega 2: Diseño, Implementación y Validación

Fundamentos de Ingeniería del Software 2025

Universidad Politécnica de Madrid
E.T.S. de Ingeniería en Sistemas Informáticos

CASO DE ESTUDIO: DESARROLLO DE BATTLESHIP

Durante las primeras fases del proceso de diseño, el equipo experto ha detectado que se puede reutilizar un módulo llamado ***Battleship*** que ya resuelve la parte fundamental de la lógica de las partidas, barcos, cálculos de puntuaciones y movimientos.

Por otro lado, se nos ha proporcionado por la UPM una librería ***externals*** que se encarga de la conexión con el LDAP para poder gestionar la conexión y validación de los usuarios contra el actor externo.

Debido a ello, hay que tener en cuenta las librerías "Battleship-1.X.jar" y "externals-X.jar". El uso de la librería ***externals*** es necesario para el buen funcionamiento del proyecto (por tanto de obligatorio uso e inclusión) mientras que el uso de la librería ***Battleship*** quedara supeditado a la aceptación del cliente (el profesor de prácticas podrá pedir o no su uso) . El uso de librerías deberá realizarse mediante la herramienta Maven dentro del proyecto. Como parte de entender el manejo de estas librerías, se proporcionan los ficheros manual_Battleship.md y manual_externals.md, que , junto con los "Battleship-1.X-javadoc" y "externals-X-javadoc", definen el funcionamiento del componente y sus características. Deberá adaptarse el entorno y compilación a estas librerías siendo el mínimo el sistema JDK-17 sobre el que las librerías están compiladas.

Para esta segunda entrega **se deberán tener en cuenta los componentes externos dados por el profesor y las funcionalidades del sistema** asignadas por el profesor de prácticas en clase a implementar.

Se deberán realizar las siguientes actividades:

1. DISEÑO

A. **Diagrama de clases de diseño.** El diagrama de clases de análisis previamente elaborado (y, si procede, corregido atendiendo a las correcciones del profesor de prácticas y las nuevas librerías) deberá modificarse para formar un diagrama de clases de diseño que abarque todo el modelo y métodos necesarios para resolver las funcionalidades asignadas por el profesor.
    - Deberá añadirse la información correspondiente como tipos de datos de los atributos, especificación de los parámetros y su tipo en los métodos y cualquier otro tipo de información que se considere necesaria.
    - El diagrama también deberá extenderse para añadir las interfaces y nuevas clases que se consideren necesarias (vistas, etc.).
    - La aplicación "BATTLESHIP" deberá seguir un patrón arquitectónico concreto sugerido por el profesor.

- La aplicación "BATTLESHIP" deberá aplicar patrones de diseño en las situaciones en las que sea posible y pertinente, así como evitar los antipatrones de diseño.

B. **Diagrama de componentes**. Se debe organizar la aplicación en componentes arquitectónicos y generar un diagrama de componentes que los relacione a través de sus interfaces. El diagrama de componentes debe ser coherente con el diagrama de clases de diseño previamente elaborado.

C. **Diagrama de despliegue.** Se deberá elaborar un diagrama de despliegue que represente la implantación del sistema.

2. IMPLEMENTACIÓN

A. **Código fuente** en Java sobre Eclipse (24.06 o superior).

- A partir de los modelos creados se codificará una parte de la aplicación "BATTLESHIP", que contendrá algunas o todas las funcionalidades que el profesor ha indicado en la fase de diseño.
- Puede que la aplicación necesite en ciertos momentos interactuar con actores externos (LDAP, redes sociales, etc.) que serán proporcionadas por el profesor.
- Puede que la aplicación necesite en ciertos componentes ya desarrollados, y que requieran de que sean integrados, que serán proporcionadas por el profesor.
- La aplicación deberá crear objetos al iniciarse (usuario y administradores) a fin de soportar dichas funcionalidades.
- El proyecto por entregar debe tener la estructura generada por un proyecto Java básico de Maven, usando la plantilla *maven-archetype-quickstart*.
- El código fuente generado deberá ser coherente con el diseño previamente elaborado. Toda implementación que no sea acorde al diseño se calificará con una nota de 0 al igual que proyectos que no compilen.

3. VERIFICACIÓN Y VALIDACIÓN

A. **Pruebas unitarias**. Se deben especificar las pruebas de caja negra y caja blanca de todos los métodos de las clases controladoras a determinar por el profesor e implementar dichas pruebas con JUnit. Las pruebas se proporcionarán junto con el código fuente dentro de la estructura creada por el proyecto Maven. Se entregará un documento PDF que contendrá un apartado dedicado a la **solución teórica** para la obtención de los casos de prueba de las pruebas de **caja negra y caja blanca** en la **carpeta "Pruebas"** del GitLab.

B. **Pruebas de aceptación.** Se deben especificar, ejecutar y documentar las pruebas de validación de la funcionalidad responsable a casos de uso especificados por el profesor. Se entregará un documento PDF que contendrá un apartado dedicado a las pruebas de aceptación y que contendrá las ***capturas de pantalla*** sobre los diferentes casos de prueba de aceptación por parte del equipo, la tipología de usuarios creados como aceptación y los códigos de los casos de uso en RedMine, la entrega estará alojada en la en la **carpeta "Pruebas"** del GitLab.

C. **Trazabilidad**. Permitir la trazabilidad desde el requisito responsable de la creación de un usuario en el sistema hasta las pruebas de aceptación de dicho requisito, pasando por todos los artefactos relacionados generados a lo largo de todo el proyecto, esto implica que deberán usarse los códigos necesarios para poder trazarlas adecuadamente la validación y **deberá poder quedar claro la conexión entre las pruebas y los requisitos en RedMine**.

INSTRUCCIONES DE ENTREGA

El plazo para esta segunda entrega de la práctica finaliza el **25 de mayo de 2025 a las 23:59** horas. El trabajo debe entregarse vía Moodle y debe quedar registrado de forma completa en GitLab en la fecha indicada**.** El *repositorio del proyecto* en GitLab se estructurará en las siguientes carpetas e incluirá los siguientes artefactos:

- **modelado**: contendrá el fichero con los diagramas en PNG o JPG y un documento PDF de recopilación (mismo que se entrega en Moodle).
- **construcción**: contendrá el **código fuente** junto con la carpeta de test que contiene las **pruebas unitarias**.
- **pruebas**: documentos PDF de verificación y validación.

El *repositorio del proyecto* deberá gestionarse haciendo uso del sistema de control de versiones git.

Además, los estudiantes deberán realizar las siguientes tareas:

- El **líder del equipo de la fase de pruebas** deberá subir **un archivo PDF resumen** a la tarea de la entrega 2. En este archivo se detallará el nombre y los enlaces del proyecto en Redmine y GitLab. Además, se indicará el nombre y apellidos de todos los integrantes del equipo que realizan la entrega, identificando la fase del ciclo de vida del software en la que cada uno actúa como líder.

- Cada miembro del equipo deberá co-evaluar a cada uno de sus compañeros de grupo (que firman la entrega) a través de la tarea de **Moodle "evaluación de competencias transversales".**