UPMFit

Salud deportiva del campus SUR.

UNIVERSIDAD
POLITÉCNICA
DE MADRID

Entrega 2: Diseño, Implementación y Validación

Fundamentos de Ingeniería del Software 2023

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

CASO DE ESTUDIO: DESARROLLO DE UPMFit

El plazo para esta segunda entrega de la práctica finaliza el **28 de mayo de 2023 a las 23:59** horas. Para esta segunda entrega **se deberán tener en cuenta únicamente las siguientes funcionalidades del sistema**:

- Alta de cliente.
- Alta de curso.
- Inscripción de un cliente a un curso.
- Visualización de los detalles de un curso (incluyendo un listado con los clientes inscritos a él).

Se deberán realizar las siguientes actividades:

1. DISEÑO

A. **Diagrama de clases de diseño.** El diagrama de clases de análisis previamente elaborado deberá modificarse para formar un diagrama de clases de diseño que abarque únicamente las funcionalidades previamente indicadas. Para ello, 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 (por ejemplo, controladores) que se consideren necesarias o convenientes. Se valorará positivamente aplicar patrones de diseño en las situaciones en las que sea posible y pertinente y evitar aplicar antipatrones de diseño. La aplicación UPMFit deberá seguir el patrón arquitectónico MVC (Modelo-Vista-Controlador).

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. **Generación automática de código.** A partir del diagrama de clases de diseño y mediante las posibilidades que ofrece Enterprise Architect, crear el esqueleto del proyecto.

B. **Código fuente** en Java sobre Eclipse (Oxigen 3 o superior). A partir de los modelos creados se codificará la parte de la aplicación correspondiente a las funcionalidades indicadas previamente en este documento **(alta de cliente, alta de curso, inscripción de un cliente a un curso y visualización de cursos)**. La aplicación podrá crear objetos al iniciarse (p. ej., monitores y salas) a fin de soportar

dichas funcionalidades. No se debe implementar ninguna otra funcionalidad. Por ejemplo, si el acceso a una funcionalidad requiere autenticar al usuario, al no tener que implementarse la autenticación, la aplicación permitirá acceder a dicha funcionalidad a cualquier usuario. Las funcionalidades implementadas deberán ser usables a través de una **interfaz de texto** con introducción de instrucciones por **línea de comandos**. El proyecto a entregar debe tener la estructura generada por un proyecto Java básico de Maven, usando la plantilla *maven-archetype-quickstart*.

Para la implementación de la funcionalidad **alta de cliente**, se facilitará una librería que deberéis utilizar para:

- Simular la comprobación de la validez de los correos institucionales en el servidor de la UPM.
- El cifrado de las contraseñas.

Dicha librería, así como las instrucciones necesarias para su uso, se pondrán accesibles la semana de prácticas en la que se vaya a comenzar con la fase de implementación de la práctica.

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.

C. **Valoración crítica** sobre la aportación que ha tenido para la implementación la realización de las fases previas de ingeniería de software. Esta valoración deberá ser realizada de forma colectiva por todos los integrantes del grupo y deberá ser entregada en un documento PDF .

3. VERIFICACIÓN Y VALIDACIÓN

A. **Pruebas unitarias**. Se deben especificar las pruebas de caja negra de todos los métodos de la clase Cliente 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 Word/PDF que contendrá un apartado dedicado a la **solución teórica** para la obtención de los casos de prueba de las pruebas de caja negra en la **carpeta "Pruebas"** del GitLab.

B. **Pruebas de aceptación.** Se deben especificar, ejecutar y documentar las pruebas de validación de la funcionalidad responsable de la creación de nuevos clientes en el sistema. Se entregará undocumento Word/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 clientes 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 cliente 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.

INSTRUCCIONES DE ENTREGA

El trabajo quedará registrado en Redmine y GitLab a fecha **28 de mayo de 2023 a las 23:59.** El *repositorio del proyecto* en GitLab se estructurará en las siguientes carpetas e incluirá los siguientes artefactos:

- **modelado**: contendrá el fichero con extensión .qea correspondiente al proyecto de Enterprise Architect, así como los diagramas en PNG o JPG y un documento PDF de recopilación.
- **construccion**: contendrá el **código fuente** junto con la carpeta de test que contiene las **pruebas unitarias**.
- **valoración critica**: documento PDF con la valoración critica anteriormente indicada.
- **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 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 coevaluar a cada uno de sus compañeros de grupo (que firman la entrega) a través de la tarea de Moodle “evaluación de competencias transversales”.