Práctica 1 Programación Web Campamentos de Programación

Autores:
Francisco Arroyo Alcaide
Luis Molina Manzano
Felipe López Castro
Alvaro Serrano Rosado

Contents

1 Organización

Para la realización de la práctica el grupo se ha reunido varias veces a lo largo de la semana. En estas reuniones se diseñó el modelo de la aplicación (diagrama UML) para tener una visual clara y precisa de lo que se pide. Además, se repartieron las tareas de forma equitativa usando Trello para el control de estas. Para trabajar en equipo se ha usado GitHub para subir las versiones del código. En ciertas ocasiones se ha usado la extensión Live Share de VSCode y así hemos podido trabajar de forma simultanéa sobre nuestros archivos y, posteriormente, realizando pull requests para aplicar cambios en la rama main.

2 Implementación de la Práctica

2.1 Estructura de Carpetas

La estructura ha sido organizad en carpetas o "packages". Para ello nos hemos basado en los ejemplos vistos en clase, de forma que una vez ingresados a la ruta "src/es/uco/pw" nos encontramos las carpetas:

- **Clases:**En esta carpeta se encuentran las carpetas de las clases pedidas en el ejercicio 1 y 2 de la práctica (asistente, monitro, actividad, campamento e inscripcion).
- **Data:**En esta carpeta se encuentra los repositorios implementados con el "patron repositrio", del cual hablaremos más adelante en este documento.
- **Display:**En esta carpeta hemos alojado los diferentes menu (main's) implementados en el programa. De los cinco menus existentes hay uno principal y el resto son elecciones que se realizan en el menu principal.
- **Gestores:**En esta carpeta hemos recogido todos los gestores, los cuales son pedidos en el ejercicio 3 de la práctica y contienen las funcoines necesarias para que los distintos menus funcionen correctamente.

2.2 Patrones de Diseño

2.2.1 Patrón Factoría

Se ha seguido el guión de la práctica y se han analizado los ejemplos vistos en clase para tratar de implementar el patrón en la práctica. En este se nos pedía poder crear inscripciones parciales o completas, y despues de esto, implimentar si eran tardias o tempranas. .

2.2.2 Patrón Singleton

Este diseño no se detalló en el enunciado, pero lo estuvimos viendo durante la clase y consideramos que es una buena práctica utilizarlo en los manejadores (como los gestores de asistentes, monitores e inscripciones). Esto se debe a

que la información debe mantenerse en una única instancia; tener más de una instancia podría llevar a la pérdida de integridad de la información.

2.2.3 Patrón Repositorio

El patrón Repository no estaba definido en el enunciado de la práctica, sin embargo, consideramos que es una buena práctica aplicar este patrón en nuestros repositorios(InterfazRepositorio.java, RepositorioAsistentes.java, RepositorioCampamentos.java y RepositorioInscripciones.java). La razón principal es garantizar que la gestión de datos se realice de manera coherente y centralizada. La mentablemente hemos tenido que recurrir a ficheros para guardar los datos, debido a una incompleta implementación de este patrón.

2.3 Otras Decisiones de Importancia

2.3.1 Implementación del Patrón Repositorio

Hemos decidido usar el patrón Repository debido a que nos parecía de gran ayuda cuando tengamos que realizar la Practica 2, ya que en ella debemos implementar una base de datos, por lo que al tener ya implementado ete patrón nos facilitará las cosas en gran medida.

3 Dificultades Encontradas

Estas son las mayores dificultades o problemas que hemos tenido a la hora de elaborar el trabajo:

- **Patrón Repository**:Aunque dicho patrón ha sido una idea nuestra, que en ningun momento era obligatoria, nos ha costado más de lo esperado su implementación. Esto se ha debido a confusiones nuestra al implementar el patrón y enlazar los repositorios con los datos. Lo que nos ha llevado bastante más tiempo de lo esperado, pero todo sea por facilitarnos el trabajo de cara a la siguiente práctica.

- **Gestor Campamentos**: Nos ha costado bastante terminar esta clase, ya que como se puede observar en el archivo, es la que más líneas de codigo contiene con diferencia. Por no decir que al contener las funciones de dos de los mains hemos tenido que trabajar más en ella para que todo funcionara como debiese.

- **ArrayList**: Hemos experimentado problemas al usar ArrayLists debido a problemas con la integridad de los datos y la consistencia en su almacenamiento. En algunas instancias, encontramos dificultades para asegurarnos de que los datos se mantuvieran coherentes en la colección, lo que resultó en posibles inconsistencias en la información. Además, no siempre se lograba una persistencia adecuada de los datos en los vectores. Debido a esto hemos tenido que revisar y mejorar nuestras estrategias de uso de

ArrayLists en la aplicación de forma que pudiéramos garantizar su correcto funcionamiento y evitar esos problemas mencionados anteriormente. Muchos de estos problemas han venido derivados de algunas funciones que trabajaban con más de un ArrayList.

- **Menus**: La implementación de las funcoines de los diferentes gestores en los menus ha sido algo que en ningun momento pensamos que nos llevaría tanto tiempo. Gran parte de los problemas de los ArrayList y del Gestor Campamentos han sido descubiertos aquí, lo que ha implicado que esta parte final del trabajo haya sido lo más complejo.

4 Bibliografía

- **Documentación oficial de Java:** https://docs.oracle.com/en/java/
- **Documentación ArrayList:** https://refactorizando.com/uso-arraylist-en-java/
- **Documentación mediante IA:** https://chat.openai.com/