Documento de reflexión – Proyecto 3

Ernesto Pérez – 202112530
Germán Moreno – 202116701
Sofia Velásquez – 202113334

Tras haber completado el proyecto 3 y mirando la evolución de nuestra aplicación junto con el proceso de diseño es fácil llegar a una importante conclusión, el diseño de una aplicación es la parte más importante de su desarrollo. En este documento de reflexión comenzaremos analizando nuestro proceso de desarrollo de una aplicación de proyecto desde su etapa inicial con el documento de análisis, el primer documento de diseño y el primer 'prototipo' pasando por una evolución de la interfaz y extensiones al proyecto.

Entrega 1

En la primera entrega de este proyecto se desarrolló el documento de análisis, el cual se enfocaba en entender el problema plantea y contenía un esquema básico UML de como luciría nuestro proyecto junto con los requerimientos funcionales (con sus respectivos casos de uso e historias de usuario) y no funcionales (enfocados a COMO hacemos las cosas) de nuestro programa en adición a las restricciones de nuestro proyecto.

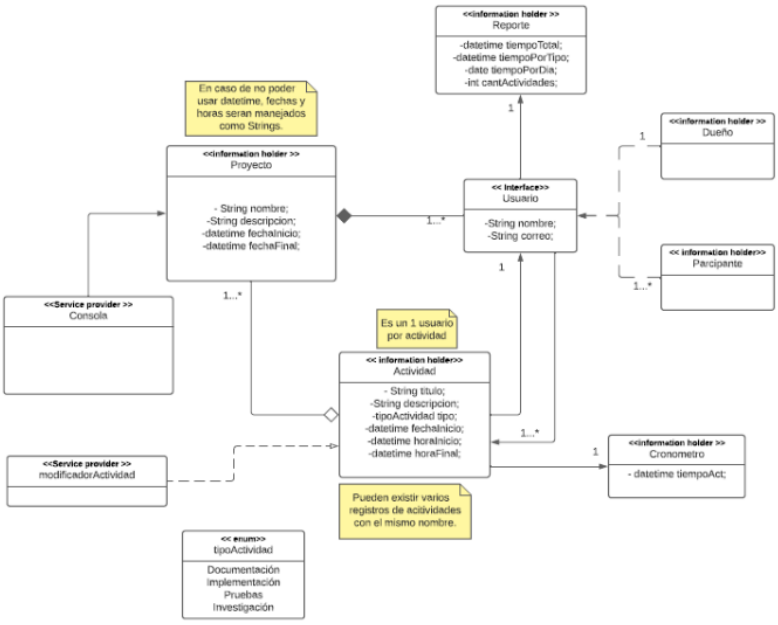


Diagrama UML del document de analisis

Asimismo, tras haber creado el documento de análisis se creó un documento de diseño. Este se creó de manera iterativa, es decir, empezamos desde lo más general de la aplicación y por cada iteración fuimos especificando mucho más las responsabilidades y roles de cada componente en nuestro proyecto, un diagrama de mundo mucho más avanzado con UML y diagramas de secuencia y especifico y colaboraciones detalladas entre los roles de nuestro proyecto.

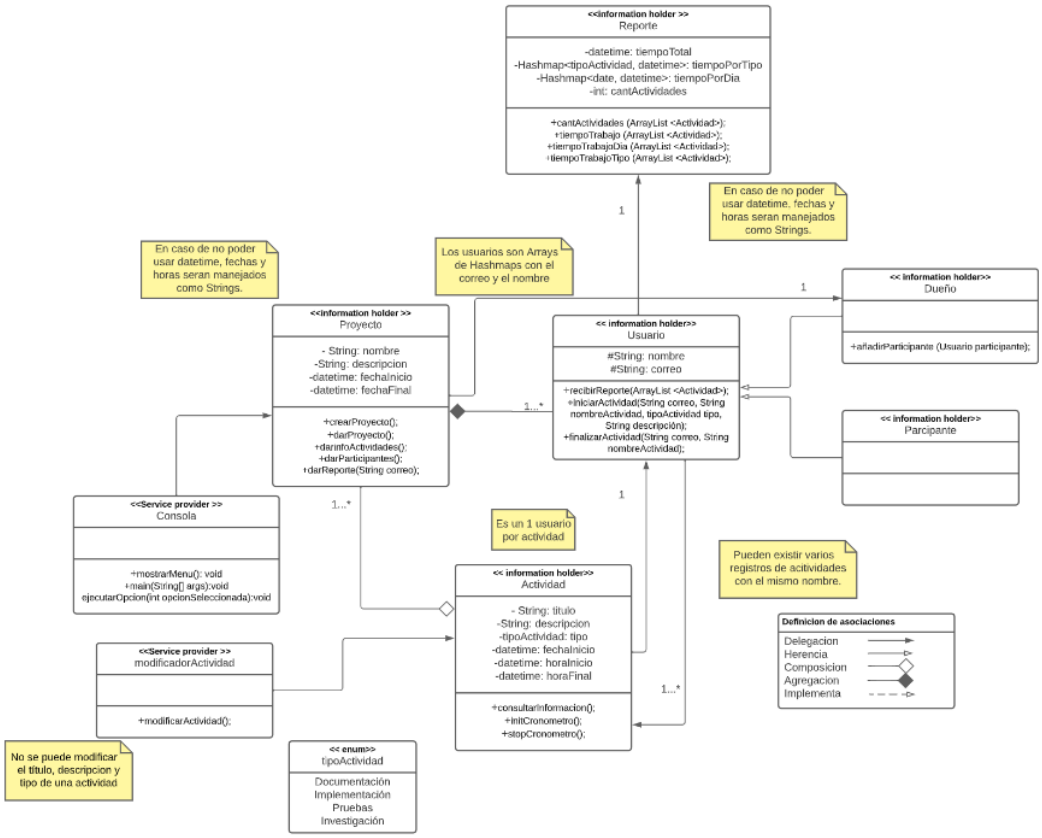


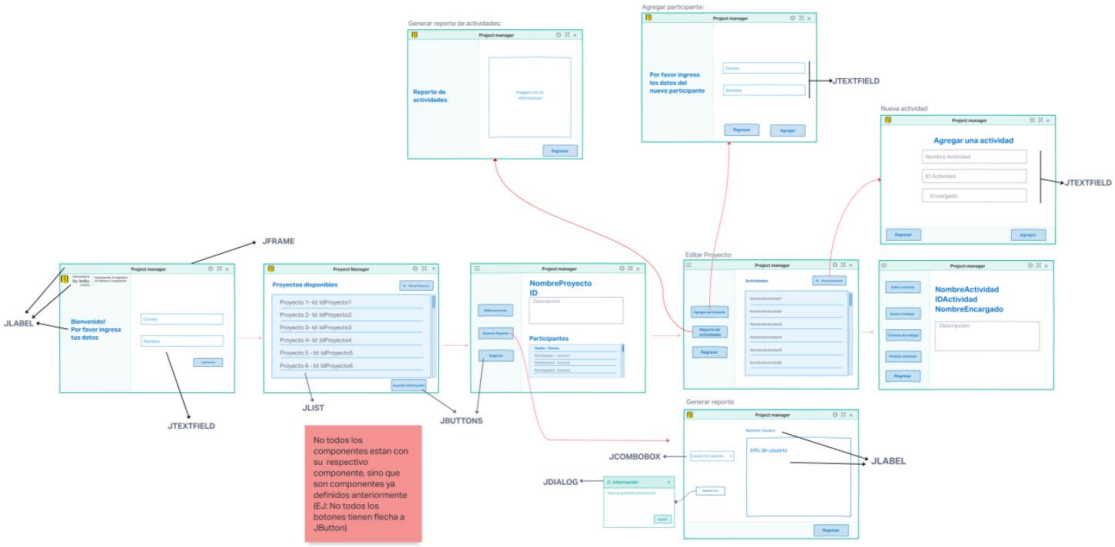
Diagrama UML del primer documento de diseño

A la hora de hacer la implementación de nuestro proyecto, esta resulto relativamente sencilla pues nos apegamos a nuestro diseño y el trabajo previo donde ya teníamos metodos y atributos de cada clase. Sin embargo, **todavía surgieron** algunos problemas que no se habían tenido en cuenta, notablemente la falta de una clase para manejar el proyecto. Un manejador de proyecto para poder movernos entre proyectos guardados y sacar la información necesaria de estos dados su id, fue algo fácil de implementar pero que incluso en las últimas modificaciones del proyecto 3 resulto siendo una de las clases más útiles de todo el proyecto, aunque no se planeó desde el principio.

Aquí es cuando nos dimos cuenta que el documento de análisis hecho sin bien simple era útil pues en torno a este formamos la estructura de nuestro proyecto y en torno a este iríamos extendiéndolo a medida que se fueran requiriendo nuevas funcionalidades. Asimismo, **el documento de diseño** resulto siendo nuestra **herramienta más útil**, siempre y cuando pudiéramos prever las funcionalidades de nuestro programa podíamos crear una solución eficiente a este. Así, durante la implementación solo sería necesario hacer ajustes menores sin alterar la estructura general del programa.

Entrega 2

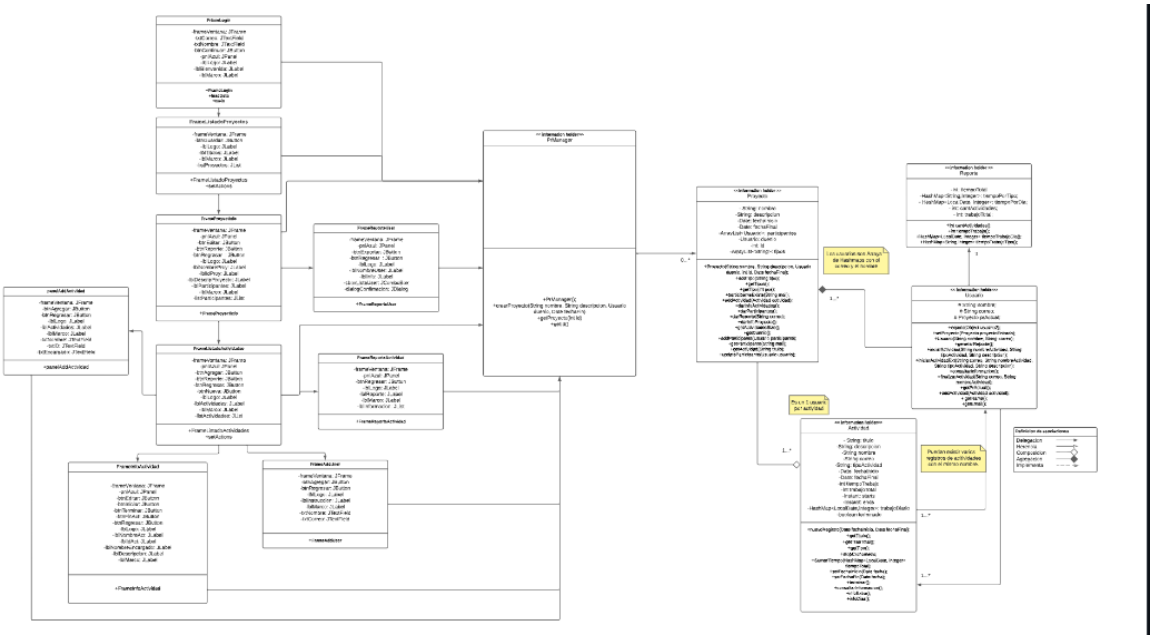
Para esta entrega el reto era implementar una interfaz basada no en consola sino una interfaz gráfica. Lo primero como siempre era modificar el documento de diseño no sin antes hacer un boceto de como queríamos que fuera nuestra interfaz gráfica la cual quedo de la siguiente manera:



Boceto de la interfaz

Con el boceto ya creado surgió un problema. La interfaz de consola implementaba previamente solicitaba al usuario el id del proyecto, id de la actividad e incluso el correo del usuario para poder realizar casi cualquier operación mientras que en la nueva interfaz planeada el correo del usuario se debía preguntar una sola vez, el proyecto se seleccionaba al inicio y ya no se le debía preguntar al usuario ninguna otra cosa y lo mismo con la actividad.

Aquí es cuando la primera tarea era modificar el documento de diseño y restructurar el codigo para que estos datos se pidieran una sola vez. **Esto fue la parte más demandante de la segunda entrega del proyecto**, y se deriva de un flojo proceso de diseño e implementación en la entrega 1 que dependía en que el usuario diera correctamente los códigos para cada actividad, proyecto y su correo en vez de manejar esto internamente eliminando así al usuario como posible fuente de error y mejorando la experiencia. Una vez resuelto este problema, se creó el diseño de como interactuarían los elementos de la interfaz con la logica del proyecto.



Modelo del mundo con la interfaz + modelo.

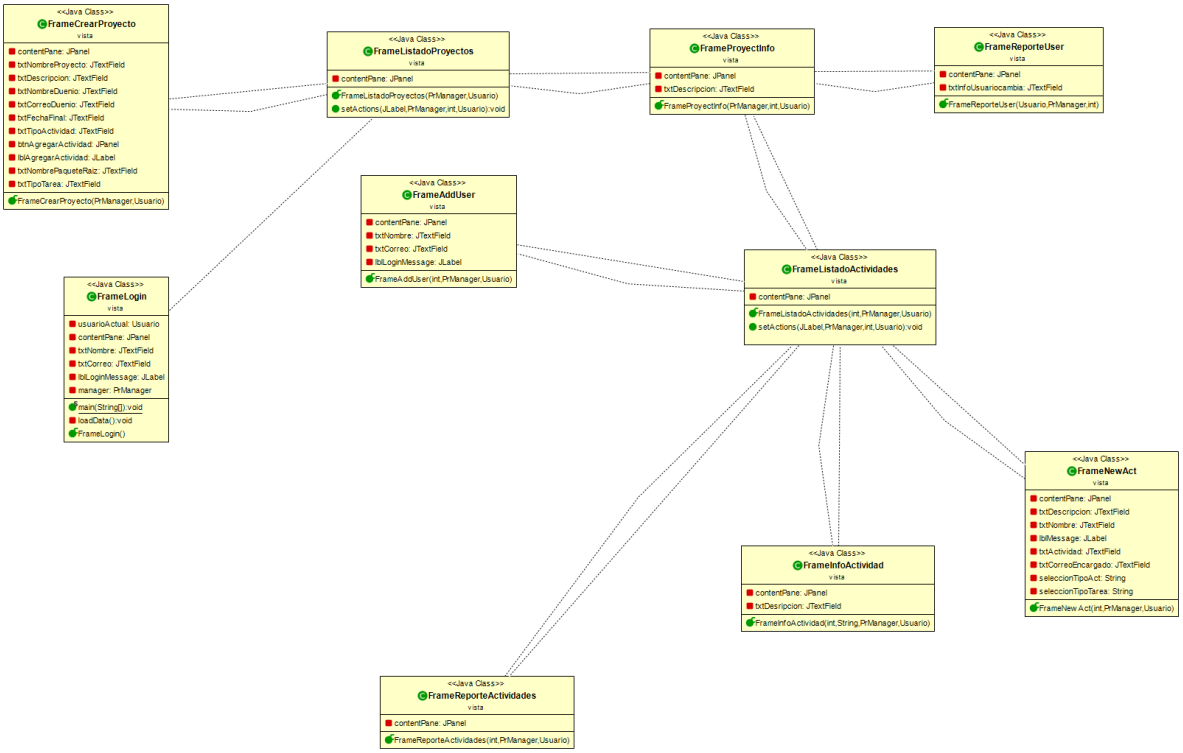
Cabe mencionar aquí que lo que nos permite conectar toda la interfaz con el modelo es la clase PrManager, de la que se habló anteriormente. Esto nos permite tener **una interfaz completamente desacoplada de la logica** del programa y seguir así el modelo MVC que se estudiaría más tarde pero que, gracias a una buena etapa de diseño, se respeta desde la primera implementación del codigo. Una vez se tuvo este diseño y se hicieron las correcciones necesarias en el codigo implementar la interfaz fue más aprender a manejar esta nueva herramienta que solucionar problemas de codigo por lo que fue bastante rápido y sencillo.

De todo este proceso sale una conclusión y recomendación: Diseñar la interfaz y corregir el codigo antes de implementar la interfaz es la mejor decisión de diseño que un grupo puede tomar.

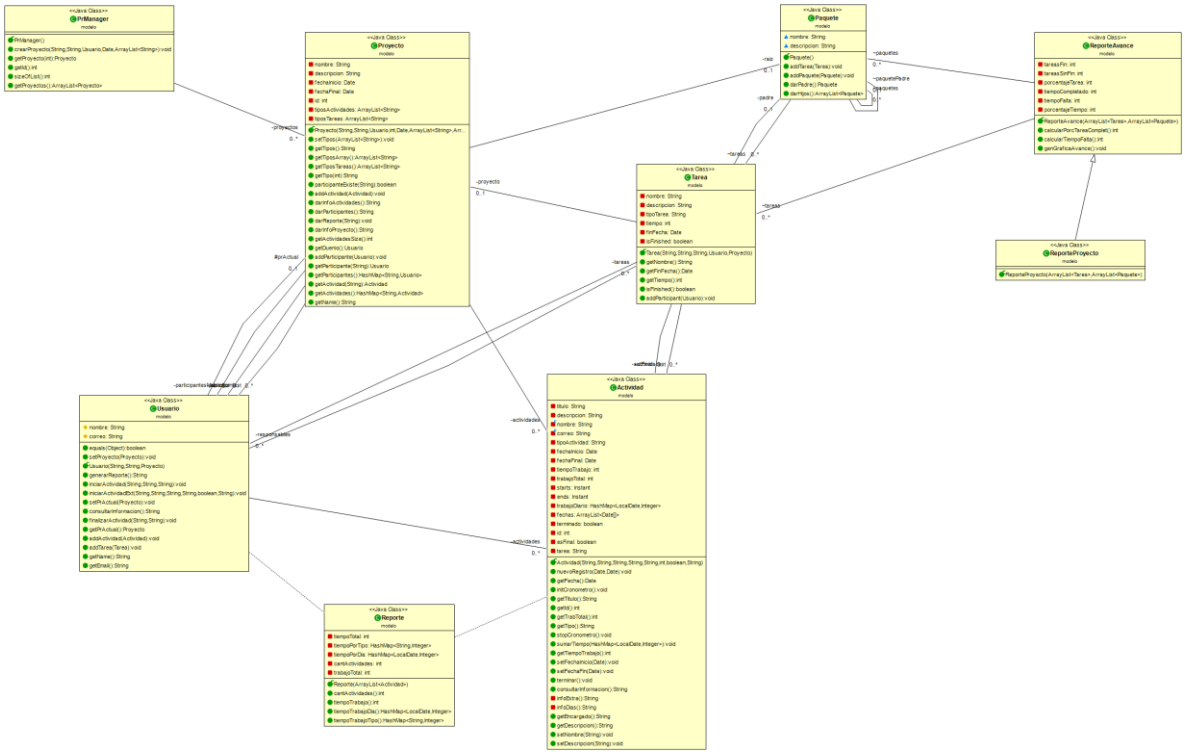
Entrega 3

Esta entrega consistió en añadir funcionalidades más que nada, principalmente un nuevo componente de tareas y paquetes de tareas y otro de reportes relacionados a varios aspectos del proyecto. Era claro que antes de tocar el codigo debíamos no solo entender el problema, sino que teníamos la tarea de encontrar la solución más óptima que se adaptara al codigo ya implementado con el fin de evitar modificar lo que ya existe, sino que solo se **agreguen nuevas funcionalidades sin tocar lo ya implementado.** Para esto creamos tres nuevas clases Tarea, Paquete y Reporte. Las primeras dos se encargarían de manejar lo relacionado a el primer componente y colaborando con las clases proyecto y actividad y la tercera usaría herencia para poder generar reportes para el proyecto y los paquetes, pero **además que extendiera las funcionalidades de una clase ya creada**, el reporte del usuario que ya calculaba información de los usuarios. De esta manera podíamos **reutilizar codigo sin dañar lo ya implementado.**

Un factor que afecto bastante esta última entrega fue la falta de tiempo y mala planeación de este por temas de otros exámenes, proyectos, etc. Sin embargo, nos dimos cuenta **que para poder implementar estas nuevas funcionalidades no tocaba modificar absolutamente nada del codigo**, al contrario, muchas cosas ya implementadas nos facilitaban añadir estas funcionalidades por lo que aparte de crear las clases solo tocaba añadir algunos botones y otras pocas cosas en la interfaz para poder completar estos componentes. Todo lo anterior se debe a que en el diseño de la entrega 2 se dejó con muy bajo acoplamiento el codigo y que desde un inicio se hizo un adecuado manejo de errores.



UML para la interfaz



UML del modelo(logica)

Teniendo todo la anterior en cuenta, la mejor recomendación que le podemos hacer a otro grupo aspirando crear su aplicación y añadirle funcionalidades en el futuro es la de dedicarle el suficiente tiempo al diseño de su aplicación para hacer la implementación un proceso más sencillo. De lo contrario estarán vulnerables a, en un futuro, reestructurar toda su aplicación o incluso se podrán ver en la necesidad de borrar y empezar de cero, por ejemplo, debido a mucho acoplamiento/dependencia entre las clases.