DOCUMENTO DE ANÁLISIS – Proyecto 3, entrega 2

Gabriel Villabón
Felipe Rueda
Mariana Ruiz

Analisis entrega 1

En esta primera entrega, solo se implementó un lenguaje orientado a objetos dentro de nuestra aplicación, por lo que solo teníamos un Diagrama de clases y de escenarios como se ven a continuación:

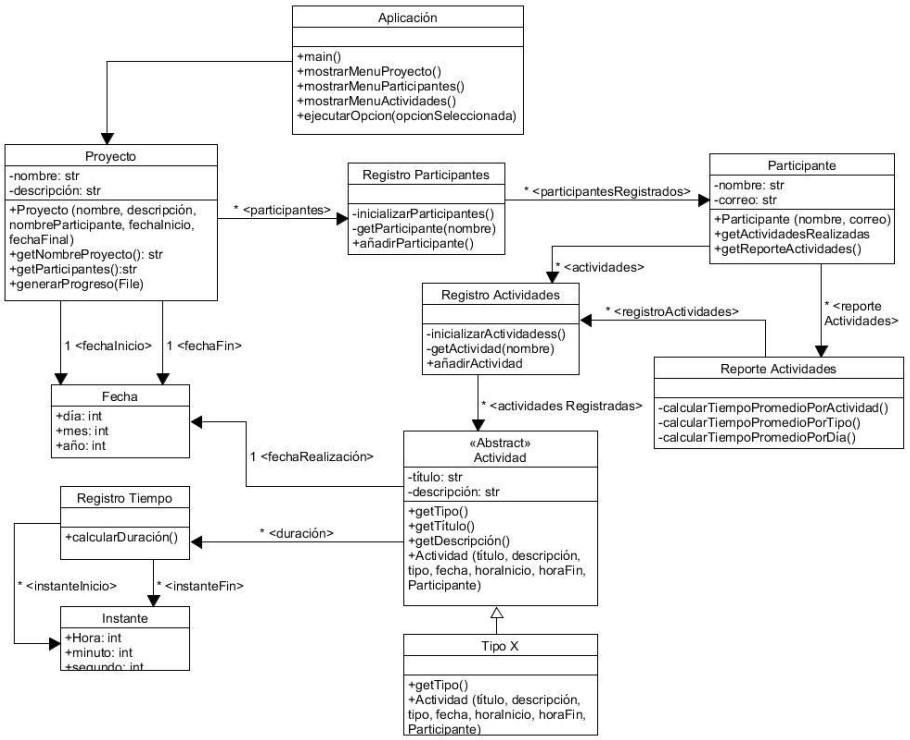

Figura 1. Diagrama clase entrega 1

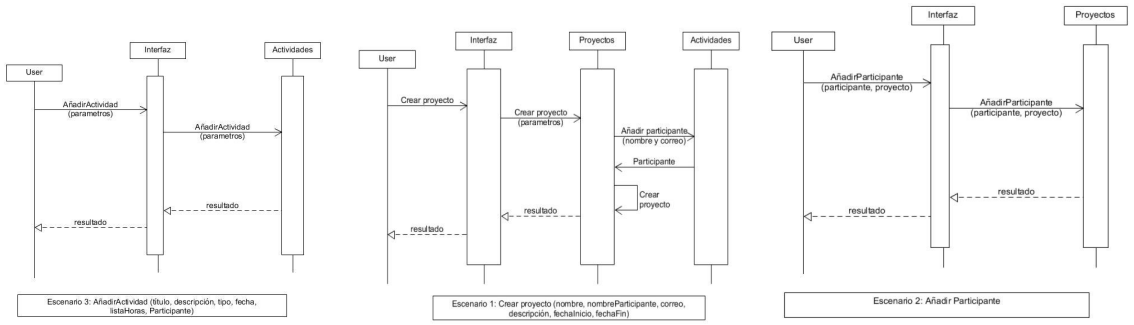

Figura 3. Diferentes diagramas de escenarios de la entrega 1

Resultados Entrega 1:

A nivel general, tuvimos buenos resultados en esta primera entrega. Sin embargo, abusamos mucho de dividir todo en demasiadas clases, algo que corregimos en siguientes

entregas. Hicimos buenos análisis y como resultado buenos diseños. Además de haber procurado implementar todo lo que vimos en clase.

Analisis entrega 2

En esta entrega ya comenzamos a implementar nuestras propias interfaces gráficas. Además, mejoramos nuestro diseño de lógica, por lo que nuestro diagrama de clases inicial fue modificado. A continuación se pueden ver los diagramas de este entrega:

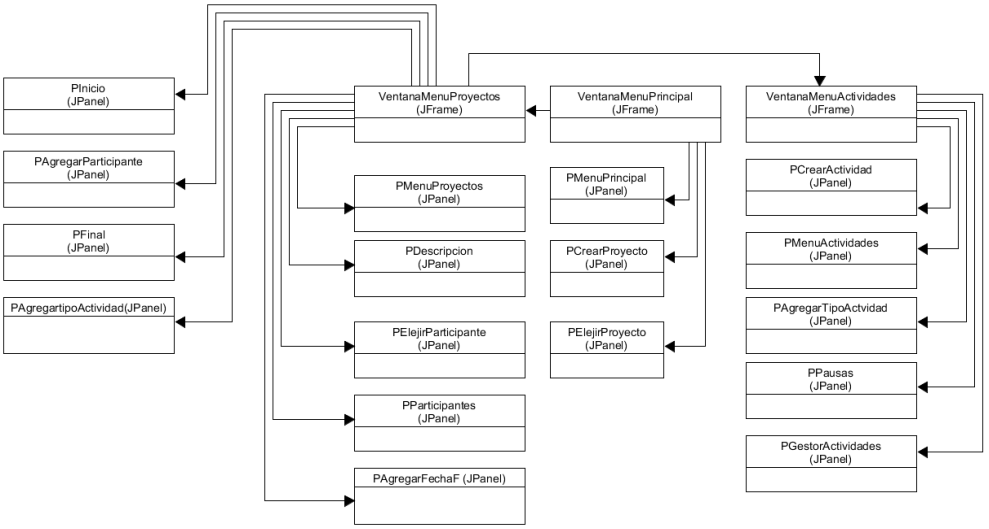

Figura 3. Diagrama interfaz gráfica entrega 2

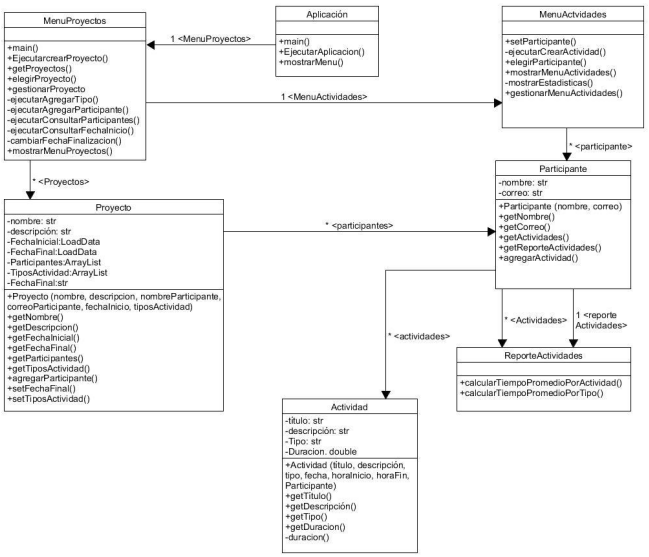

Figura 3. Diagrama clase entrega 2

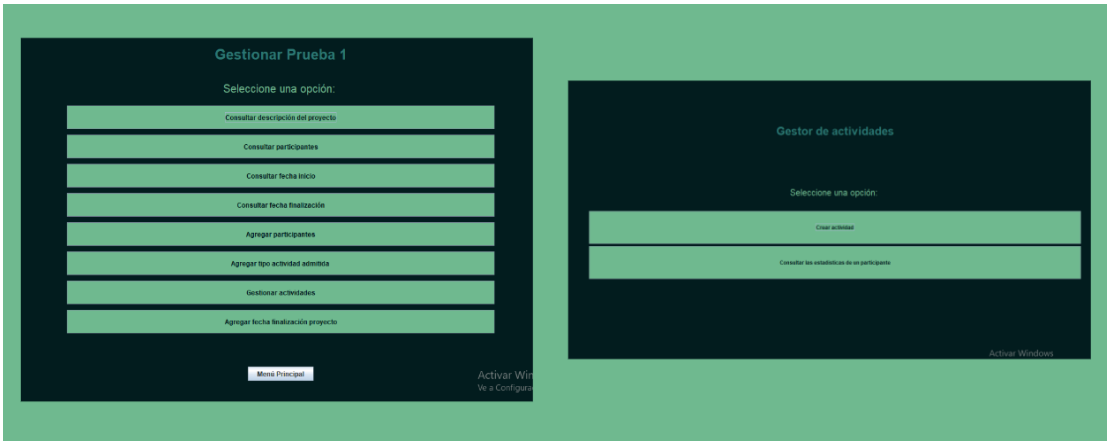

Figura 4. Algunas imágenes de la interfaz entrega 2

Resultados entrega 2:

En esta entrega tenemos más complicaciones. Por un lado, tuvimos que eliminar varias clases del diseño pasado ya que vimos que solo aumentaba la complejidad de la aplicación, diseños tales como los Registros y los tipo X de las actividades, entre otros. Estas clases eliminadas eran innecesarias y se podían implementar en otras clases como atributos. Por otro lado, Con respecto al diseño de la interfaz, aunque en un principio nosotros planteamos diseños mucho más arriesgados, mientras se realizaba el proyecto comprendimos la complejidad de crear diseños en Java, por lo que decimos crear una interfaz más estándar y minimalista, con 3 ventanas principales y varios subpaneles que derivan de estas. Desafortunadamente, por el tipo de diseño que habíamos implementado en la lógica, no pudimos crear un cronómetro de las actividades dentro de la interfaz, pues habría tocado hacer varías modificaciones con tiempo del que no pudimos disponer. Aquí entendimos la importancia del principio “Open/Closed” de solid, estar abierto a extensiones y cerrado a modificaciones.

Analisis entrega 3

En esta última entrega se añadieron varias extensiones. Por un lado, las nuevas clases y funcionalidades derivadas de “Paquetes de trabajo” y “tareas”, junto con todas sus pestañas y ventanas en la interfaz gráfica. Por otro lado, se implementaron formas de manejar posibles errores que pueda cometer el usuario. A continuación los diseños de esta entrega final:

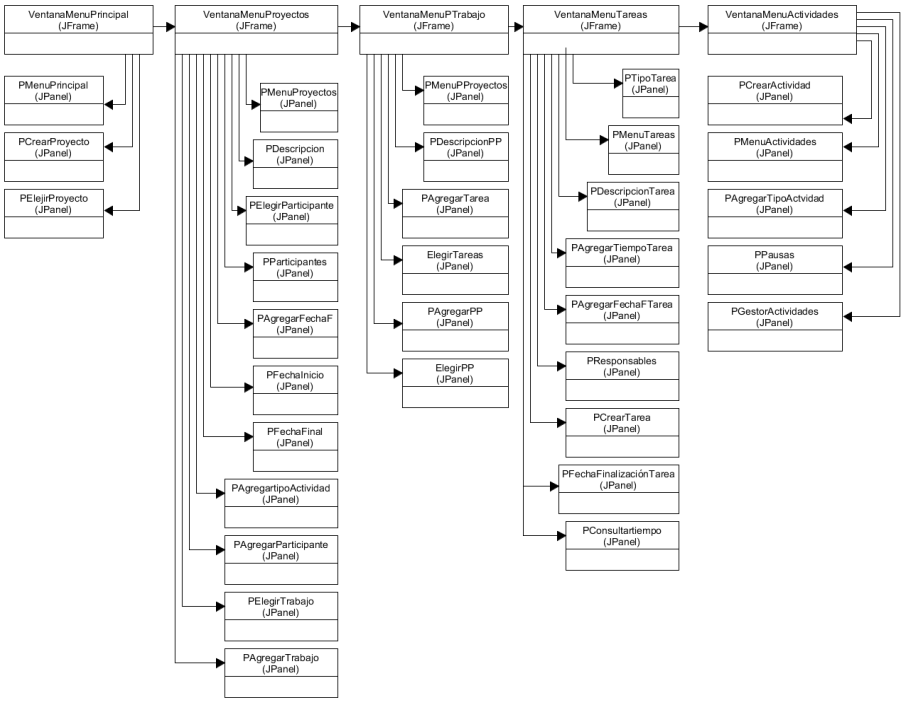

Figura 5. Diagrama interfaz gráfica entrega 3

Diseño de la interfaz: Con respecto a entregas anteriores, no cambiamos demasiado la interfaz salvo por la creación de otras dos ventanas: VentanaPProyectos y VentanaTareas. Estas dos se añadieron a la actualización con el fin de interactuar con las clases PaquetesProyecto y Tarea respectivamente. Por esa razón, estas están ubicadas de forma jerárquica entre Proyectos y actividades como se ve en el diagrama, cada una con diferentes paneles para cumplir sus propios requerimientos funcionales

Diseño de la aplicación: El diagrama de clases se ve en la siguiente imagen, en esta actualización. Por un lado, tenemos en cuenta las excepciones del programa que ocurren exclusivamente en los métodos ejecutarCrearProyecto() y en ejecutarCrearActividad() dentro de las clases “ControladorProyectos” y “ControladorActividades” respectivamente. Por otro lado, para implementar las tareas que están dentro de los paquetes de trabajo, que así mismo, este último está dentro de proyectos, creamos la clase “PaqueteTrabajo” y “Tarea” con sus correspondientes relaciones como se ve a continuación:

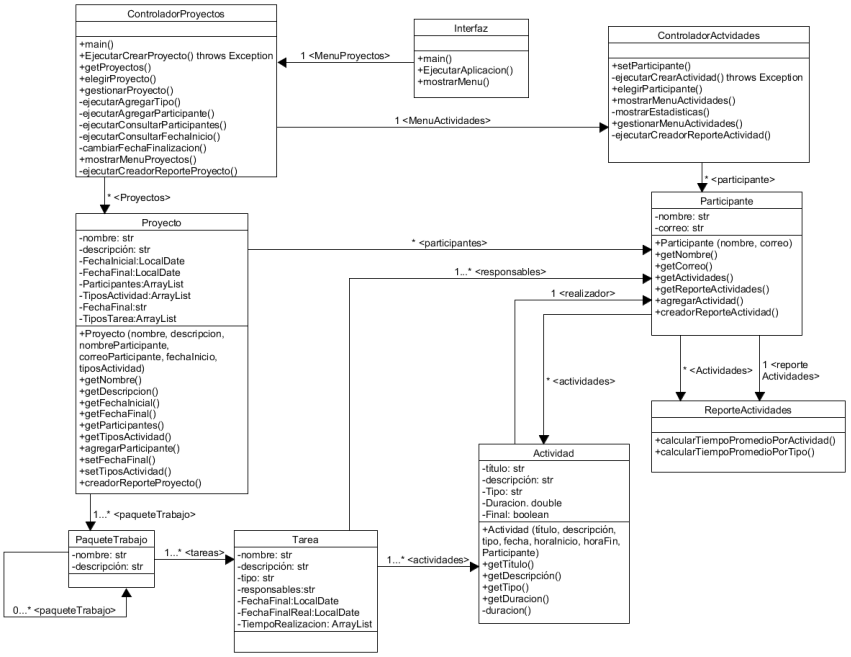

Figura 6. Diagrama interfaz clases entrega 3

Los Exception dentro de “ejecutarCrearProyecto()” que se solucionan son :

- Poner los @ en los correos para verificar que están escritos correctamente
- Verificar que la fecha agregada por el usuario exista en el calendario gregoriano
- Verificar que las fechas de inicio estén antes de fecha fin
- No hayan nombres completos repetidos

Los Exception dentro de “ejecutarCrearActividad()” que se solucionan son :

- Verificar que la hora agregada por el usuario exista la hora militar
- Verificar que la hora de inicio esté antes de hora fin
- No haya títulos repetidos

Resultados entrega 3:

En esta entrega aplicamos todo lo que vimos durante este semestre. Afortunadamente, las nuevas clases de “Paquete de trabajo” y “Tarea” no eran tan diferentes a las antiguas clases “Proyecto” y “actividad”, por lo que gracias a las buenas prácticas que habíamos implementado en entregas pasadas, tuvimos la posibilidad de reutilizar antiguo código de anteriores entregas para las nuevas implementaciones, tanto a nivel de lógica como de interfaz gráfica.

Con respecto al manejo de errores, utilizamos la mayor cantidad de botones posibles. Así, era casi imposible para un usuario crear errores en la aplicación, pues los botones pueden ser presionados o no, pero causan errores. De igual forma, para los Widgets tipo “JTextField” nos aseguramos que se detectara cuando estuvieran vacíos, tuvieran información repetida o no tuvieran sentido. Por ejemplo, en el caso de las fechas, no se permite escribir fechas que no estén dentro del calendario gregoriano o no tengan sentido cronológico.

Para finalizar, los cambios que se hicieron con respecto a la entrega 2 del diagrama de clases y de interfaz gráfica solo fueron extensiones, no hubo modificación alguna en lo que ya existía. Por lo que se cumplió el objetivo inicial.