**Iteración Final Proyecto 2:**

Elementos relacionados con el dominio:

* Proyecto (structurer)
* Reporte (service provider)
* Participante (information holder)
* Cronometro (service provider)

Elementos relacionados con el sistema:

* Aplicación (coordinator)
* FileManagaer (service provider)
* Imágenes(service provider)

Cosas que fluyen dentro del sistema:

* Actividad (information holder)

Interfaces:

* VentanaPrincipal (interfaz)
* CrearActividad (Interfaz)
* CrearProyecto(Interfaz)
* CronometrarActividad(interfaz)
* EscogerActModificar(interfaz)
* Loggeo(interfaz)
* MenuProyecto(interfaz)
* ModificarActividad(interfaz)

Responsabilidades:

* Iniciar un nuevo proyecto
* Ver los proyectos creados
* Añadir actividades a un proyecto
* Añadir nuevos participantes a un proyecto.
* Cronometrar el trabajo
* Empezar el cronometro
* Pausar el cronometro
* Finalizar el cronometro
* Introducir información de la actividad
* Iniciar la actividad
* Finalizar la actividad
* Modificar fecha/hora de la actividad
* Cambiar persona que hizo actividad
* Escoger un proyecto especifico
* Escoger una actividad especifica
* Visualizar el tiempo que tardó una actividad en realizarse.
* Medir cuánto tarda en realizarse una actividad
* Calcular el tiempo total de una actividad
* Generar un reporte
* Visualizar el reporte
* Calcular la información del reporte
* Clasificar la actividad por tipo.
* Tener persistencia
* Guardar archivos planos
* Leer archivos planos
* Cargar archivos
* Escribir archivos
* Mostar la ventana principal
* Mostrar el menu del proyecto
* Mostar el menu de modificar actividad
* Mostrar el menu de modificar actividad
* Mostrar el menu del cronometro
* Mostrar el reporte
* Mostrar el menu de añadir participantes
* Hacer la grafica de trabajo

**Colaboraciones**

* Aplicación, Proyecto, Actividad y Participante: Para poder hacer una actividad necesitan colaborar aplicación, proyecto, actividad y participante. Aplicación tiene o crea un proyecto, en el cual proyecto crea una actividad con su información correspondiente y finalmente al acabar se le asigna al participante que la realizo. Para crear una actividad tambien se necesita la ayuda de Cronometro para saber su tiempo de trabajo.
* Aplicación, Proyecto, Actividad: Para poder modificar una actividad necesitan colaborar aplicación, proyecto y actividad. Aplicación tiene el proyecto, en el cual proyecto tiene una actividad la cual toca modificar según su información. Para modificar tambien se necesita Participante ya que se puede modificar el encargado.
* Aplicación, Proyecto, Actividades, Participante y Reporte: Para producir y mostrar un reporte se necesita que Participante tenga la información de sus actividades (Actividad), luego Proyecto coge esa información y genera un reporte con ayuda de la clase Reporte y finalmente Aplicación coge el reporte y lo muestra.
* Aplicación, Proyecto y Participante: Para añadir participantes a los proyectos. Aplicación debe tener un proyecto actual, luego proyecto añade a un participante con su información de la clase.
* Aplicación y Proyecto: Para crear un proyecto, aplicación crea el proyecto, con su información correspondiente de la clase.
* Para manejar la persistencia, se necesitó de las clases FileManager, Proyecto, Actividad y Participante. FileManager es el principal protagonista de esa responsabilidad. Para la escritura, FileManager.write() es quien crea el archivo y escribe en él, y para obtener la información para escribir, se usan las variadas funciones get() de Proyecto, Actividad y Participante. Para leer, FileManager.read() lee el archivo por líneas y en cada línea se van separando los datos jerárquicamente, y se organizan en variables, arrays y mapas. Nuevamente, para hacer esta “traducción” se utilizan varios métodos de Actividad, Participante y Proyecto.
* En general aplicación interactua con todas las diferententes interfaces para que estas puedan mostrar y/o actualizar informacion. Esto ocurre de la siguiente forma, las interfaces llaman a un metodo ligado a lo que quieren hacer, aplicación se conecta con el proyecto y se realiza el metodo. Se le devuelve la informacion a la interfaz como puede ser en el caso de generarReporte o se actualiza la informacion del modelo como es en el caso de crearProyecto.

**Reflexion:**

Una reflexión sobre el diseño obtenido:

* Un trade-off, es tener un estilo de control centralizado, ya que por una parte este nos permite tener todo controlado desde la clase Proyecto, pero en un futuro si crece o cambia la aplicación se puede dificultar que Proyecto maneje tantos métodos y conozca tanta información.
* Una decisión de diseño que tomamos fue que participante no fuera el que modificara las actividades, sino que proyecto fuera el que lo hiciera, esto para evitar relaciones innecesarias y preservar la cohesión.
* Otra decisión de diseño fue no tener una clase externa dueño, ya que complicaba las cosas innecesariamente, sino que tener que los participantes tengan un booleano que defina si es dueño o no y actuara consecuente a esto
* Nos dimos cuenta de que tener clases “auxiliares” que sean service provider como Cronometro y File Manager para funciones es muy útil, ya que ayuda a la especialización, acoplamiento y cohesión.
* Nos dimos cuenta que tener un buen diseño original y un buen codigo base fueron fundamentales para poder realizar esat parte del proyecto correctamente, esto ya que al estar todo bien organizado y estructurado fue más un trabajo de conectar y expandir que de actualizar el codigo y/o cambiar el codigo.

Reflexión sobre el proceso de diseño:

* Un elemento que presento dificultad fue decidir si debiese existir la clase dueño y cual debería de ser su relación con participante y actividades. Al final decidimos no implementarla para evitar enredos y solamente incorporarla a Participante como un booleano
* Otro elemento que presento gran análisis fue la clase Proyecto, esto debido a que al inicio no sabíamos que responsabilidades darle o que propósito ponerle. Solo que contenga información del proyecto o que realice acciones sobre él y la actividad. Al final nos decantamos porque proyecto fuera nuestra clase central ya que tenía que conocer la información de casi todos y era practico de una vez asignarle los métodos correspondientes.
* Finalmente, Reporte, presentó una leve dificultad debido a que no sabíamos con qué relacionarla o si debería de saber alguna información. Al final decidimos que como ya teníamos la clase central de proyecto, que esta se encargara de hacer el reporte, debido a que ya tenía la información disponible y que reporte fuera un service provider que se permitiera construir y obtener.
* El diseño influye mucho sobre cómo va a salir un proyecto, ya que en base a él se modela la implementación, por eso es bueno tener un diseño bueno, especifico y diciente; para que todo el proceso sea más eficiente.

**Cambios en la implementación del GUI**

* El primer cambio realizado fue, que se le dio la opción al usuario de decidir en qué momento iniciar el cronometraje de su actividad. Es decir, el usuario puede crearla, sin iniciar el cronometraje y decidir iniciarla cuando desee. Esta decisión se tomó para darle más flexibilidad al usuario.
* Al momento de hacer la interfaz del reporte se cambió el diseño, esto debido a que nos dimos cuenta de que el tamaño de los reportes va a ser muy variable dependiendo del proyecto y su contenido. Por eso en vez de una lista por cada tipo y por cada persona, nos decantamos por tener una lista desplegable con todos los tipos y una lista desplegable con todas las personas. Esto hace que sin importar el tamaño del reporte se pueda ver de la mejor manera posible.
* Otro cambio que realizamos en el diseño fue en la parte de modificar actividad. Inicialmente, el usuario debía hacer input del correo y nombre del nuevo encargado (si deseaba cambiarlo). Pero para darle más comodidad al usuario, decidimos hacer un comboBox que desplegara los participantes del proyecto. Esto nos aseguraba una mejor experiencia para el usuario, y adicional a esto, nos aseguramos de que el usuario no metiera un nombre o correo que no fueran participantes ya creados dentro del proyecto.