

Documento de diseño– Proyecto 3

Ernesto Pérez – 202112530

Germán Moreno – 202116701

Sofia Velásquez – 202113334

** Para este documento de diseño se toma como base el documento de diseño del proyecto 2.*

Roles:

1. Proyecto
2. Dueño
3. Participante
4. Actividad
5. Reporte

Responsabilidades:

| Responsabilidad | Rol |
|---|--------------------------------|
| Crear proyecto | Proyecto |
| Finalizar proyecto | |
| Añadir participantes | Dueño |
| Iniciar actividad | Dueño o participante (USUARIO) |
| Modificar actividad | |
| Finalizar actividad | |
| Mostrar descripción de actividad | Actividad |
| Cronometrar tiempo de trabajo | |
| Mostrar la cantidad de actividades realizadas por usuario | Reporte |
| Mostrar reporte del tiempo trabajado de cada usuario | |
| Mostrar reporte de tiempo trabajado por tipo de actividad de cada usuario | |

Colaboraciones:

- Proyecto colabora con actividad y con usuarios para obtener su información y poder crearse.
- Dueño colabora con proyecto para añadir participantes.
- Reporte colabora con actividad y con usuario para poder generarse.

Justificación de decisiones:

- Decidimos remover la clase cronómetro, debido a que los métodos que se planeaban implementar en ella resultaron estar mejor ubicados en la clase actividad.
- Para optimizar el futuro desarrollo de la aplicación, decidimos mantener las dos horas tanto de inicio como de fin en un solo ArrayList conteniendo a ambos.

- Para calcular los tiempos el usuario tiene una lista con todas las actividades realizadas y la actividad misma es un diccionario con llave= nombre de la actividad, valor = atributos de la actividad.
- Decidimos cambiar el estereotipo de la clase usuario a information holder, debido a que si fuera una interface solo podríamos definir el comportamiento de Dueño y Participante, pero no especificarlo. En cambio, decidimos que Dueño y Participante heredaran los métodos de Usuario, debido a que ambos tienen los mismos requerimientos (dueño tiene uno adicional).

Reflexión

Ventajas y desventajas (tradeoffs)

El diseño iterativo es muy bueno en el sentido en que es muy sistemático y permite elaborar un proyecto grande con un alto nivel de detalle. Una desventaja es el tiempo que consume esta iteración tan minuciosa de todos los elementos pues nos dimos cuenta de que podíamos hacer la primera iteración con un alto nivel de detalle y en las siguientes iteraciones más que desglosar nuestros elementos en elementos más pequeños nos hubiéramos concentrado en ‘pulir’ responsabilidades, roles, colaboraciones y demás, haciendo del proceso mucho más rápido sin mayores complicaciones.

Elementos problemáticos.

Por un lado, tuvimos problemas con el reporte y la manera en que este calcularía el tiempo total, por actividad y por día y como conocería esta información del usuario y también el tipo de estructura datos más apto que nos permitiera guardar los tiempos y hacer la suma total dependiendo del método llamado.

Ubicar el método de modificadorActividad dentro de la clase fue importante porque era posible tener este método dentro de la misma clase actividad, pero decidimos que enfocarlo en una clase dedicada era la mejor opción para mantener todo más simple.

Glosario de Métodos:

Clase Proyecto:

- CrearProyecto(); Crea el proyecto
- darProyecto(); Da nombre, descripción, fecha inicial y final del proyecto y el dueño.
 - Falta dar participantes
- darInfoActividades(); Da lista de actividades por su nombre y un registro de todas las actividades con el mismo nombre para saber si se hizo durante varios días.
- darParticipantes(); da los participantes del proyecto, los participantes son
- darReporte(String correo); Pide el reporte de un usuario dado su correo

Clase Usuario:

- recibirReporte(ArrayList <Actividad>); pide a la clase Reporte dar un reporte para el correo dado como parámetro.
- iniciarActividad(String correo, String nombreActividad, tipoActividad tipo, String descripción); inicia una actividad en el proyecto.
- finalizarActividad(String correo, String nombreActividad); finaliza una actividad en el proyecto.

Clase Dueño:

- añadirParticipante (Usuario participante); Añade un participante a el proyecto con el correo y su nombre.

Clase Participante:

Nota: hereda los métodos de la clase usuario

Clase ReporteUsuario:

- cantActividades (ArrayList <Actividad>); cantidad de actividades realizadas por usuario dado su correo.
- tiempoTrabajo (ArrayList <Actividad>); reporte del tiempo trabajado del usuario
- tiempoTrabajoDia (ArrayList <Actividad>); reporte de tiempo trabajado por día del usuario
- tiempoTrabajoTipo (ArrayList <Actividad>); tiempo trabajado por tipo de actividad del usuario

Clase Actividad:

- consultarInformacion (); Imprime toda la información de la actividad (titulo, descripción, tipo, fecha inicio, hora inicio, hora final y participante que la realizo. Adicionalmente, exporta esta información en un archivo de texto txt.
- initCronometro(); inicia el cronometro para tomar el tiempo de realización de una actividad.
- stopCronometro(); termina el cronometro y recibe el tiempo en que se demoró la actividad.

Clase modificadorActividad:

- modificarActividad(); Modifica la fecha, hora de inicio y fin para las actividades, se pueden dar varias modificaciones para una sola actividad. No se puede modificar el título, descripción y tipo de una actividad.

Clase Consola:

- mostrarMenu(): void Muestra el menú de opciones
- main(String[] args):void
- ejecutarOpcion(int opcionSeleccionada):void Ejecuta una de las opciones del menú.

Modificaciones durante la implementación

1. Se creo la clase llamada PrManager. Como su nombre lo dice simplemente se encarga de manejar el proyecto actual en el que se encuentra dado que el programa puede guardar múltiples proyectos.

Con los atributos:

- ArrayList<Proyecto> proyectos -> La lista de todos los proyectos creados

Y con los métodos:

- crearProyecto (String nombre, String descripción, Usuario duenio, Date fechaFin)
 - Se encarga de crear el proyecto y añadir el proyecto a la lista de todos los proyectos

- `getProyecto(int id)`
 - Conseguir un proyecto dado el id
 - `getId()`
 - Encargada de conocer el número de proyectos actuales para saber cuál es el siguiente id.
2. Se usa la librería serializable para guardar los objetos directamente por lo que no se trabaja con archivos de texto ni csv's.
 3. Se añadieron métodos de consola y métodos para retornar informaciones que no estaban descritos en el diseño pero que su funcionalidad es muy simple y concreta.

Interfaz Gráfica:

Boceto: <https://ernestoperez603950.invisionapp.com/freehand/Boceto-oW87QUomK>

Clase `FrameLogin`:

- `nombre:JTextField`
- `correo:JTextField`
- `bienvenida:JLabel`
- `ImgUniandes:JFrame`

Esta clase pide el nombre y correo del usuario para poder ingresar a todo el contenido

Métodos:

- `FrameLogin`: El constructor se encarga de crear todos los componentes de la interfaz, validar el login del usuario y llevarlo al siguiente frame.
- `Main`: Es la primera ventana y tiene el método `main` que comienza la ejecución del programa
- `LoadData`: Se encarga de cargar los archivos (en caso de que existan) de proyectos creados anteriormente.

Clase `FrameListadoProyectos`:

- `Proyectos:JLabel`
- `NuevoProyecto:JButton`
- `GuardarInfo:JButton`

Muestra todos los proyectos que se han guardado en el gestor

Métodos:

- `FrameListadoProyectos` : El constructor se encarga de crear todos los componentes de la interfaz, y de llevar al usuario al frame para crear un proyecto o a la información de un proyecto según sea su elección.
- `SetActions`: Le añade la funcionalidad a la lista de proyectos, permitiendo al usuario clickearlas para ver la información del proyecto.

Clase FrameProjectInfo:

- infoProyecto: JLabel
- participantes: JLabel
- editarProyecto: JButton
- generarReporte: JButton
- Regresar: JButton

Da la información de cada proyecto

Metodos:

- FrameProjectInfo : El constructor se encarga de crear todos los componentes de la interfaz para mostrar la información del proyecto seleccionado. Además muestra 3 opciones para que el usuario avance a editar el proyecto, generar reporte de usuario o regresar al frame anterior.

Clase FrameListadoActividades:

- AgregarParticipantes: JButton
- ReporteActividades: JButton
- Regresar: JButton
- Actividades: JFrame

Muestra las actividades y da opciones para editar el proyecto

Metodos:

- FrameListadoActividades : El constructor se encarga de crear todos los componentes de la interfaz y la funcionalidad de los 4 botones.
- SetActions: Le añade la funcionalidad a la lista de actividades, permitiendo al usuario clickearlas para ver la información del proyecto.

Clase FrameInfoActividad:

- EditarActividad: JButton
- IniciarATrabajar: JButton
- TerminarTrabajo: JButton
- FinalizarActividad: JButton
- Regresar: JButton

Muestra la información de una actividad y permite iniciar y finalizar una sesión de trabajo

Metodos:

- FrameInfoActividad : El constructor se encarga de crear todos los componentes de la interfaz, es decir, mostrar la información de la actividad y añadir la funcionalidad a los 5 botones del frame.

Clase FrameNewAct:

- NombreActividad:JLabel
- IdActividad:JLabel
- Encargado:JLabel
- Agregar:JButton
- Regresar:JButton

Permite añadir una nueva actividad al proyecto

Metodos:

- FrameNewAct : El constructor se encarga de crear todos los componentes de la interfaz, crea 4 textFields y crea tanto el usuario (si no existe) como la actividad.

Clase FrameReporteUser:

- NombreUsuario:JLabel
- InfoUsuario:JLabel
- ListadoUsuario:JComboBox
- ExportarTxt:JButton

Exporta un reporte de usuario a un archivo txt

Metodos:

- FrameReporteUser : El constructor se encarga de crear todos los componentes de la interfaz y mostrar la información asociada al reporte.

Clase FrameReporteActividades:

- Reporte:JFrame
- Titulo:JLabel

Muestra en un recuadro la información de cuándo ha habido actividades realizadas en un proyecto.

Metodos:

- Frame : El constructor se encarga de crear todos los componentes de la interfaz y mostrar la frecuencia de actividades por día para un proyecto.

Clase FrameAddUser:

- user:JTextField
- Correo:JTextField
- Titulo:JLabel

Esta clase pide el nombre y correo de un Usuario y lo añade al proyecto en el que estamos.

Metodos:

- Frame : El constructor se encarga de crear todos los componentes de la interfaz y crear 2 text fields para que se ingrese la información del nuevo usuario.

Reflexión

Ventajas y desventajas (tradeoffs)

De cara al usuario, tener una interfaz gráfica solo tiene ventajas frente a una interfaz de consola. Si bien esta nueva interfaz resulta en una experiencia mucho más fluida y amena para el usuario trae sus desventajas a la hora de desarrollarla puesto que es más tiempo que se invierte en el diseño (pensando cómo se verá la interfaz y como será su interacción) y más tiempo que se toma en el desarrollo completo del programa ya que es mucho más difícil y demorado tener este tipo de interfaces a una simple consola.

Elementos problemáticos

La forma en que se muestra la información y la manera en que el usuario interactúa con el programa. Cuando era interfaz de consola, se pedía constantemente el ID del proyecto y nombre/correo del Usuario al ejecutar opciones sin embargo para la consola esto se cambió a que el usuario solo debe ingresar esta información al inicio cuando se logea y cuando selecciona el proyecto lo cual hace de una experiencia más fluida.

Modificaciones para el proyecto 3:

Requerimientos a implementar:

- Tareas
- Generar reportes

Modificaciones a clases anteriores

Clase Proyecto

Atributos:

- ArrayList<String> tiposTareas;

Metodos

- getTiposTareas

Clase Actividad:

Atributo:

- Boolean esFinTarea

Clase Tareas:

Atributos:

- String nombre
- String descripción
- String tipoTarea – Definida en cada proyecto

- Time tiempo
- Date finFecha – depende de la actFinal
- Actividad actFinal – Actividad que completa la tarea
- ArrayList<Usuario> responsables
- ArrayList<Actividad> actividades -

Metodos:

- DarNombre
- DarFinFecha
- Terminada
- SetFinFecha
-
- darTiempoTotal
 - Se encarga de dar el tiempo total en base a todas las actividades a las que estamos asociadas.

Interfaz

- Las tareas se crearán en el mismo panel que muestra la información del proyecto.
 - Un nuevo panel será usado para ingresar toda la información correspondiente a la tarea.
- Al crear una nueva actividad se le pedirá al usuario vincular la actividad con una Tarea e indicar si la actividad es la final o no.
- Los tipos de tarea se crean al crear el proyecto, de manera similar a como se crean los tipos de actividad.

Clase Paquete:

El paquete raíz se creará al crear un proyecto, solamente se creará con un nombre y la descripción

Para paquetes raíces tocara crearlos en el menú del proyecto y se tiene que especificar cual es el paquete padre.

Atributos:

- String Nombre
- Boolean esRaiz
- String Descripción
- Paquete paquetePadre
- ArrayList <Paquete> Paquetes – Todos los hijos directos
- ArrayList <Tarea> Tareas – Todas las tareas directas

La clase estará compuesta por un árbol en donde cada nodo será el paquete de trabajo/tarea y la raíz seria estrictamente un paquete de trabajo

Métodos:

- agregarTarea
- AgregarPaquete
- DarPadre
- DarHijos

- DarReporte

Clase ReporteAvance:

Como decisión de diseño y siguiendo las instrucciones del enunciado se tomó que desde la clase Paquete se hace un reporte únicamente del avance.

La calidad de planeación, equipo y resumen del proyecto son solamente para el proyecto.

El tiempo se maneja en minutos

Atributos:

Todos son atributos protegidos

- int tareasFin
- int tareasSinFin
- int porcentajeTarea
- int tiempoCompletado
- int tiempoFalta
- int porcentajeTiempo
- ArrayList <Paquete> Paquetes – Todos los hijos directos
- ArrayList <Tarea> Tareas – Todas las tareas directas

Métodos:

- calcularPorcTareaCompleta
- calcularTiempoFalta
- generarGraficaAvance

Interfaz:

- Desde el panel proyecto se añadirá la opción de generar el reporte. El usuario podrá seleccionar si quiere avance del proyecto total o elegir un paquete de tareas para generar su reporte.
- Una vez seleccionado esto se mostrará el panel con la respectiva información
- **AvanceProyecto:** Mostrar en una barra que muestra el total de actividades completadas sobre la cantidad de actividades planeadas y la cantidad de tiempo dedicado sobre cantidad de tareas completadas sobre el total, cantidad de tiempo planeado en las tareas completadas sobre la cantidad de tiempo total planeada

Clase ReporteProyecto extiende ReporteAvance extiende ReporteUsuario:

Esta clase hace el reporte del avance más calidad de planeación, equipo y resumen del proyecto.

Atributos:

- int tareasFin
- int tareasSinFin

- int porcentajeTarea
- int tiempoCompletado
- int tiempoFalta
- int porcentajeTiempo
- ArrayList <Paquete> Paquetes – Todos los hijos directos
- ArrayList <Tarea> Tareas – Todas las tareas directas
- ArrayList<Usuarios> users
-

Métodos:

- genGraficaAvance
- genGraficaCalidad
- genGraficaEquipo
- genGraficaResumen

Interfaz:

- Desde el panel proyecto se añadirá la opción de generar el reporte. El usuario podrá seleccionar, aparte del avance, si quiere visualizar la gráfica de equipo, la de calidad o el resumen del proyecto para generar su reporte. Una vez seleccionado esto se mostrará el panel con la respectiva información.
- **AvanceProyecto:** Mostrar en una barra que muestra el total de actividades completadas sobre la cantidad de actividades planeadas y la cantidad de tiempo dedicado sobre cantidad de tareas completadas sobre el total, cantidad de tiempo planeado en las tareas completadas sobre la cantidad de tiempo total planeada
- **Equipo:** Se mostrará un histograma donde se puede ver el desempeño del equipo, mostrando la cantidad de tiempo que cada uno ha invertido, la cantidad de tareas pendientes de las que es responsable y la cantidad de tiempo planeada para esas tareas.
- **Calidad:** En este aspecto es importante entender qué tan acertada ha sido la planeación (tiempo planeado sobre el tiempo real) por esto hemos decidido, para evaluar la eficiencia de una actividad, desarrollar una gráfica que muestre la cantidad de días faltantes hasta la fecha estimada de finalización.
- **Resumen:** En este aspecto se debe ver un resumen general del proyecto, incluyendo los tiempos totales, la cantidad de tareas pendientes.

Clase ReporteUsuario (ya está implementada, solo es añadir algunos métodos):

Métodos:

- DarTareasPendientes
- DarTiempoPendiente

Manejo de errores

Posibles errores:

- Al ejecutar el programa, puede no encontrar los datos del guardado anterior, resultando en un error del tipo nullPointer, para manejarlo basta con usar un try Catch que capture la excepción y en lugar de ella envíe un simple mensaje indicando que no se ha podido cargar la información e iniciar todo desde cero.

- Al momento de generar los reportes, debido a que es necesario iterar sobre todos los sub-Paquetes de cada paquete, en un momento determinado dejará de haber más y al no encontrarlo, nuevamente se envía un `nullPointerException`, para el manejo de esta excepción, solo es necesario detener el bucle cuando ocurre sin necesidad de informarle al usuario.
- Cuando se genera el gráfico de avance, de no haber datos al momento de hacerlo se genera una excepción que termina con el programa, para evitarlo hay dos maneras posibles, que es hacer el botón disponible únicamente si hay datos para generar el gráfico, o simplemente enviar un pop up que indique al usuario que por falta de datos no es posible graficar y luego se devuelva a la pantalla anterior.
- Cuando se crea el proyecto y las actividades, es necesario ingresar la fecha estimada de finalización. Debido a que estas fechas pueden estar incorrectas, el programa puede llegar a detenerse cuando se hace el `Parse de String a Date`, por lo que se lanza un pop up al usuario indicando que la fecha ingresada es inválida y es necesario intentar de nuevo, con una fecha que sí funcione.
- Ingresar datos de tipos inesperados, como Strings en campos de enteros puede resultar en una finalización anticipada de la ejecución actual, por lo que para evitarlo la idea principal es capturar la excepción obtenida para en su lugar enviar un pop up, regresar a la ventana que provocó el error y pedir al usuario que ingrese nuevamente sus datos.