**Proyecto 3**

**Entrega 1**

**Contexto del Proyecto**

Este proyecto consiste en construir una aplicación capaz de generar reportes que muestren el avance del proyecto y el desempeño del equipo de trabajo. Hay libertad sobre los elementos que deben tener los reportes, pero deberían cubrir al menos los siguientes aspectos:

* Avance del proyecto. En este aspecto es importante saber qué tal va el proyecto con respecto a las fechas planeadas y reales de terminación de las tareas, y también saber qué tal va el proyecto en cuanto a su avance (cantidad de tareas completadas sobre el total, cantidad de tiempo planeado en las tareas completadas sobre la cantidad de tiempo total planeada). El avance debería poder conocerse tanto para el proyecto entero como para cualquiera de los paquetes de trabajo.
* Calidad de la planeación. En este aspecto es importante entender qué tan acertada ha sido la planeación (tiempo planeado sobre el tiempo real) y encontrar características que nos ayuden a mejorar la planeación (por ejemplo, las tareas del tipo X tienen un error más grande en la estimación).
* Equipo. En este aspecto se debe poder 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.
* Resumen del proyecto. En este aspecto se debe ver un resumen general del proyecto, incluyendo los tiempos totales, la cantidad de tareas pendientes, el avance, etc.
* Cada proyecto tendrá una WBS, compuesta por paquetes de trabajo y tareas organizados de forma jerárquica: un paquete de trabajo puede tener por dentro otros paquetes de trabajo o puede tener tareas.
* Cada tarea debe tener un nombre, una descripción y un tipo. A medida que avance el proyecto, para cada tarea se podrá definir también una cantidad estimada de tiempo que tomará (el tiempo planeado), una fecha estimada de finalización y uno o varios responsables.
* De ahora en adelante, cuando se registre una actividad realizada en un proyecto, debe indicarse la tarea dentro de la WBS a la que quedará asociada. Una tarea podría tener varias actividades asociadas. Por ejemplo, si en la WBS del proyecto había una tarea llamada “Escribir documento de diseño” esta tarea podría tener asociadas varias actividades con los nombres “Construir estructura del documento de diseño”, “Escribir versión inicial del documento”, “Corregir el documento” y “Terminar el documento”. Fíjese que las actividades no se planean cuando se construye la WBS, sino que se van asociando a las tareas a medida que se van realizando. Al asociar una actividad a una tarea debe indicarse si con esa actividad se completa la tarea o no.

**Análisis y Diseño Del Proyecto**

* **Diagrama de Clases**

A continuación, presentamos el modelo del dominio del proyecto. En él pueden identificarse las entidades que participan en el caso, sus atributos y las respectivas relaciones entre cada una de ellas, incluyendo asociaciones y relaciones de herencia. Por cuestiones de Visualización puede ser consultado en el siguiente link: <https://app.moqups.com/3OroEs1P6YbAVBoon9I5XZbWjMgoPO9K/view/page/aa9df7b72>

* **Glosario y Descripciones**

**Consola**

**Atributos**

* proyecto: atributo que contiene la información de un proyecto.

**Métodos**

* ejectutarOpciones(): void: ejecuta las opciones disponibles del menú.
* mostrarMenu(): void: Muestra las opciones del menú.
* input(String): String: Recibe un tipo de dato del usuario.
* generarProyecto(): void: genera un proyecto nuevo.
* anadirParticipante(): void: añade un participante nuevo a un proyecto.
* agregarActividad(): void: agregar una actividad a un proyecto.
* buscarParticipante(): void: busca un participante dentro de un proyecto.
* buscarActividad(): void: busca una actividad dentro de un proyecto.
* editarActividad(Actividad): void: permite que los usuarios editen una actividad dentro de un proyecto.
* guardarAlSalir(): void: guarda los cambios realizados en un proyecto y cierra la aplicación.

**dialEditar**

**Atributos**

* lblTitulo: etiqueta del título.
* lblFechaInicial: etiqueta de la fecha inicial
* lblDescripcion: etiqueta de la descripción
* lblTipo: etiqueta del tipo
* txtTitulo: texto del título.
* txtDescripcion: texto de la descripción.
* txtFechainicial: texto de la fecha inicial
* btnEditar: botón para editar.
* radioTitulo: opción del título.
* radioFechaInicial: opción de la fecha inicial
* radioTipo: opción del tipo.

**Métodos**

* dialEditar(VentanaActividades, Ventana Principal): constructor de la clase *Métodos*
* actualizar(Actividad): actualiza la actividad actual.

**dialFecha**

**Atributos**

* fechaIngresada: fecha ingresada por el usuario.
* petición: etiqueta de petición.
* btnIngresar: botón para ingresar.
* btnFechaHoy: botón de la fecha diaria.
* txtFechaIngresada: campo de texto para la fecha ingresada.
* tipo: tipo de acción que realizará el botón.

**Métodos**

* dialFecha(String, String, VentanaActividades): constructor de la clase *dialFecha.*

**dialNuevaActividad**

**Atributos**

* lblTitulo: etiqueta del título.
* lblEncargado: etiqueta del usuario encargado.
* lblDescripcion: etiqueta de la descripción de la actividad.
* lblTipo: etiqueta de tipo.
* txtTitulo: campo de texto del título.
* txtDescripción: campo de texto de la descripción.
* comboTipo: cuadro combinado de strings.
* comboParticipantes: cuadro combinado de participantes.

**Métodos**

* dialNuevaActividad(VentanaPrincipal): método constructor de la clase *dialNuevaActividad*

**dialogoCargarProyecto**

**Atributos**

* lblTitulo: etiqueta del título.
* lblFechaInicial: etiqueta de la fecha inicial.
* lblDescripcion: etiqueta de la descripción.
* lblDuenoNombre: etiqueta del nombre del dueño del proyecto.
* lblDuenoCorreo: etiqueta del correo del nombre del dueño del proyecto.
* lblTipos: etiqueta de los tipos.
* lblArchivo: etiqueta del archivo.
* txtTitulo: texto del título
* txtDescripcion: texto de la descripción.
* txtFechaInicial: texto de la fecha inicial.
* txtDuenoNombre: texto del nombre del dueño.
* txtDuenoCorreo: texto del correo del dueño del proyecto.
* txtTipos: cuadro de texto de los tipos.
* txtArchivo: cuadro de texto del archivo.
* seccion: contenedor de las secciones.
* btnCrear: botón de crear.
* btnAbrir: botón de abrir.

**Métodos**

* dialogoCargarProyecto(Ventana Principal, Integer): método constructor de la clase *dialogoCargarProyecto.*

**dialWarning**

**Atributos**

* textoWarning: etiqueta del texto de advertencia.
* btnCerrar: botón de cerrar.
* juntar: contenedor de juntar.

**Métodos**

* dialWarning(String): método constructor de la clase *dialWarning.*

**PanelCentroVP**

**Atributos**

* lblProyecto: etiqueta del proyecto
* lblDescripcion: etiqueta de la descripción.
* lblFechaInicial: etiqueta de la fecha inicial.
* lblFechaFinal: etiqueta de la fecha final.
* txtProyecto: texto del proyecto.
* txtDescripcion: etiqueta de la descripción.
* txtFechaInicial: campo de texto de la fecha inicial.
* txtFechaFinal: campo de texto de la fecha final.
* btnParticipante: botón del participante.
* btnActividad: botón de la actividad.
* panelInfo: contenedor de la información.

**Métodos**

* panelCentro(VentanaPrincipal): método constructor de la clase *panelCentroVP.*
* actualizar(): se encarga de actualizar la información dada por el usuario.

**panelDerechoVP**

**Atributos**

* panelListas: contenedor de las listas.
* panelBotones: contenedor de botones.
* btnParticipantes: botón de los participantes.
* btnActividades: botón de las actividades.
* btnRegistro: botón de registro.
* listaActividades: lista de las actividades.
* listaParticipantes: lista de los participantes.

**Métodos**

* panelDerechaVP(VentanaPrincipal): método constructor de la clase *panelDerechaVP*
* cargarInfo(): se encarga de cargar la información.
* actualizarParticipantes: actualiza los participantes en un proyecto.
* actualizarActividades: actualiza las actividades en un proyecto.

**panelInfoActividad**

**Atributos**

* panelInfoActi: contenedor de la información de una actividad
* panelBotones: contenedor de botones.
* btnPausar: botón de pausa.
* btnEditar: botón de editar.
* btnSiguiente: botón de siguiente.
* btnAnterior: botón de anterior.
* btnFinalizar: botón de finalizar.
* lblTitulo: etiqueta de título.
* lblTipo: etiqueta de tipo
* lblTerminado: etiqueta de terminado.
* lblFechaInicial: etiqueta de la fecha inicial.
* lblFechaFinal: etiqueta de la fecha final.
* lblEncargado: etiqueta del encargado.
* lblDescripcion: etiqueta de la descripción.
* txtTitulo: campo de texto del título.
* txtTipo: campo de texto del tipo.
* txtTerminado: campo de texto de terminado.
* txtFechaInicial: campo de texto de fecha inicial.
* txtFechaFinal: campo de texto de fecha inicial.
* txtEncargado:  campo de texto del encargado.
* txtDescripcion: campo de texto de la descripción.

**Métodos**

* panelInfoActividad(VentanaActividades): método constructor de la clase *paneInfoActividad.*
* actualizar(Actividad): se encarga de actualizar una actividad.

**panelIzquierdaVP**

**Atributos**

* lblDueño: etiqueta de Dueño.
* lblUsuario: etiqueta de usuario.
* lblLogo: etiqueta de logo.
* txtDueño: campo de texto del dueño.
* panelDueño: contenedor del dueño.
* panelCargar: contenedor de cargar.
* panelImagen: contenedor de las imágenes.
* iconLogo: imagen del logo.
* iconUsuario: imagen del usuario.
* btnAbrir: botón de abrir.
* btnCrear: botón de crear.

**Métodos**

* panelIzquierdaVP(VentanaPrincipal): método constructor de la clase *panelIzquierdaVP.*
* actualizar(): se encarga de actualizar la información ingresada por el usuario.

**VentanaActividades**

**Atributos**

* actividades: lista de actividades de un proyecto.
* txtBusqueda: campo de texto de búsqueda.
* btnBuscar: botón de buscar.
* lblBusqueda: etiqueta de búsqueda.
* lblTipo: etiqueta de tipo.
* lblParticipante: etiqueta de participante.
* lblTitulo: etiqueta de título.
* lblLogo: etiqueta del logo.
* radioTipo: selección del tipo.
* radioTitulo: selección del título.
* radioParticipante: selección del participante.
* panelBusqueda: contenedor de búsqueda.
* subPanelBusqueda: sub-contenedor de búsqueda.
* panelUnionBusqueda: panel de unión de la búsqueda.
* panelActividades:  contenedor de actividades de una actividad.
* iconLogo: imagen del logo.

**Métodos**

* editarActividad(Integer, String): se encarga edita la actividad con la información dada por el usuario.
* actualizarSiguiente(): actualiza la siguiente actividad.
* finalizarAcitividad(Date): finaliza la actividad.
* estadoCronometro(Integer, Date): da el estado del cronometro.
* cargarActividadesEncontradas(): carga las actividades encontradas.
* gerPrincipal(): retorna la actividad principal del proyecto.

**VentanaParticipantes**

**Atributos**

* panelInfo: contenedor de la información de los participantes.
* panelBotonoes: contenedor de botones
* btnAnterior: botón anterior.
* btnDueno: botón de dueño.
* btnSiguiente: botón de siguiente.

**Métodos**

* VentanaParticipantes(VentanaPrincipal): constructor de la clase *ventanaPParticipantes.*
* actualizarAnterior(): actualiza la información del participante anterior.
* actualizarSiguiente(): actualiza la información del participante siguiente.
* setDueno(): establece el dueño.

**VentanaPrincipal**

**Atributos**

* panelIzquierda: contenedor izquierdo.
* panelCentro: contenedor del centro.
* panelDerecha: contenedor derecho.

**Métodos**

* VentanaPrincipal(): método constructor de la clase *ventanaPrincipal.*
* nuevoParticipante(String, String): añade un nuevo participante a un proyecto.
* nuevaActividad(String, String, Integer, Integer): agrega una nueva actividad a un proyecto.
* cargarProyecto(String): carga la información de un proyecto.
* crearProyecto(String, String, String, String, String, String): crea un proyecto.
* guardarAlSalir(): guarda al salir los cambios de un proyecto.

**Actividad**

**Atributos**

* titulo: Título de una actividad.
* fechaInicio: fecha de inicio de una actividad.
* fechaFinal: fecha final de una actividad.
* tipo: tipo de actividad.
* descripcion: descripción de la actividad.
* encargado: participante encargado de realizar la actividad.
* cronometro: cronometro de la actividad.
* terminado: estado de terminado de la actividad.

**Métodos**

* Actividad (String, String, String, Participante): método constructor de la clase *actividad.*
* getFechaInicial(): retorna la fecha inicial de la actividad.
* getFechaFinal(): retorna la fecha final de la actividad.
* getTitulo(): retorna el título de la actividad.
* getTipo(): retorna el tipo de actividad.
* getDescripcion(): retorna la descripción de la actividad.
* getEncargado(): retorna el participante encargado de la actividad.
* getTerminado(): retorna el estado de terminado de una actividad.
* getCronometro(): retorna el cronometro de un actividad.
* setFechaInicial(Date): modifica la fecha inicial de una actividad.
* setFechaFinal(Date): modifica la fecha final de una actividad.
* setTitulo(String): modifica el título de una actividad.
* setTipo(String): modifica el tipo de actividad.
* setDescripcion(String): modifica la descripción de la actividad.
* estadoCronometro(Integer, Date): retorna el estado del cronometro.
* setTerminado(Date): modifica estado de terminado de la actividad.

**Cronometro**

**Atributos**

* fechaPausa: fecha de pausa.
* tiempoTotalPausado: tiempo que una actividad duró en pausa.
* tiempoCronometrado: tiempo que tardó en completarse una actividad.
* pausa: estado de la actividad.

**Métodos**

* Cronometro(): método constructor de la clase *cronometro.*
* setPausa(pauseDate): modifica la pausa de una actividad.
* getTiempoInvertido(): muestra el tiempo invertido de una actividad.
* getTiempoCronometrado(): muestra el tiempo cronometrado.
* getCantPausado(): muestra la cantidad de tiempo pausado.
* getEstado(): muestra el estado de la actividad.
* calcularTiempoCronometrado(Date, Date): calcula el tiempo cronometrado de una actividad.

**Participante**

**Atributos**

* nombre: nombre del participante.
* correo: correo del participante.
* dueño: muestra si el participante es dueño o no.
* miReporte: Reporte de participante.
* misActividades: lista de actividades en las que esta un participante.

**Métodos**

* Participante(String, String, String): método constructor de la clase *participante.*
* getNombre(): muestra el nombre de un participante.
* getCorreo(): muestra el correo de un participante.
* isDueño(): muestra si un participante es dueño o no.
* getReporte(): muestra el reporte de un participante.
* getActividades(): muestra la lista de actividades en las que esta un participante.

**Proyecto**

**Atributos**

* titulo: título del proyecto
* descripcion: descripción del proyecto
* fechaInicio: fecha de inicio de un proyecto.
* fechaFinal: fecha de finalización de un proyecto.
* dueno: dueño del proyecto.
* participante: lista de participantes de un proyecto.
* actividades: lista de actividades de un proyecto.

**Métodos**

* Proyecto(): método constructor de la clase *proyecto*.
* setTitulo(String): modifica el título de un proyecto.
* setDescripcion(String): modifica la descripción de un proyecto.
* setFechaFinal(Date): modifica la fecha final de un proyecto.
* setDueño(Participante): modifica el dueño de un proyecto.
* getTitulo(): muestra el título de un proyecto.
* getDescripcion(): muestra la descripción de un proyecto.
* getFechaInicial(): muestra la fecha inicial de un proyecto.
* getFechaFinal(): muestra la fecha final de un proyecto.
* getParticipantes(): muestra la lista de participantes de un proyecto.
* getActividades(): muestra la lista de actividades de un proyecto.
* añadirParticipante(String, String): añade un participante a un proyecto.
* buscarParticipante(String): busca un participante dentro de un proyecto.
* crearActividad(String, String, String, String): crea una nueva actividad.
* buscarActividad(String, String): busca una actividad dentro de un proyecto.
* aEditar(Actividad, Integer, String): edita una actividad.

**Reporte**

**Atributos**

* elReporte: reporte.
* nombreParticipante: participante del reporte.
* correoParticipante: correo del participante.

**Métodos**

* Reporte(String, String): método constructor de la clase  *reporte.*
* getReporte(): muestra el reporte.
* calcularTiempoInvertido(ArrayList<Actividad>): calcula el tiempo que invirtió un participante en una actividad.
* calcularTiempoDia(ArrayList<Actividad>): calcula el tiempo por día.
* generarTextoReporte(ArrayLList<Actividad>): genera el texto del reporte.
* deMilisegundosATiempoCompleto(Long): convierte los milisegundos a tiempo completo.

**Reglas del Dominio**

* + Cada proyecto, actividad y participante DEBE tener la información completa requerida para hacer el respectivo registro. De lo contrario no se permitirá la creación de este.
  + Debe ser posible modificar las fechas y horas dadas por defecto por el programa de las actividades.
  + Todos los miembros del proyecto deben ser capaces de registrar y editar las actividades que se realicen dentro de este.
  + Los tipos disponibles se deben definir para cada proyecto.
  + Debe haber un archivo plano por proyecto y la aplicación debe funcionar si estos archivos se eliminan.
  + La interfaz de la aplicación debe estar basada en consola.
  + Descripción de los requerimientos funcionales de la aplicación.
  + Cada proyecto tendrá una WBS, compuesta por paquetes de trabajo y tareas organizados de forma jerárquica: un paquete de trabajo puede tener por dentro otros paquetes de trabajo o puede tener tareas.
  + la raíz de cada WBS siempre debe ser un paquete de trabajo.
  + Cada paquete de trabajo debe tener un nombre y una descripción.
  + Cada tarea debe tener un nombre, una descripción y un tipo.
  + Los tipos de las tareas se definirán en cada proyecto.
  + cuando se registre una actividad realizada en un proyecto, debe indicarse la tarea dentro de la WBS a la que quedará asociada. Una tarea podría tener varias actividades asociadas.
  + El tiempo real de una tarea es la suma del tiempo de las actividades que se realizaron para cumplir con la tarea. La fecha de finalización de una tarea es la fecha en la cual se realizó la actividad que completó la tarea.

**Historias de Usuario**

**Lideres del proyecto**

Como líder de proyecto, me gustaría tener registro de los nuevos proyectos que se vayan a realizar, donde se pueda ver las descripciones detalladas, las actividades e integrantes que los conforman. Además, requiere que los integrantes del proyecto puedan añadir y editar las actividades que les corresponden sin afectar las actividades que ya han sido realizadas. Todo esto para poder tener una mayor de coordinación entre los integrantes del proyecto y tener una mayor eficiencia para realizar el trabajo propuesto.

Para cumplir con las necesidades del usuario, la aplicación debe:

* Debe dejar registrar a cualquier participante con la información adecuada.
* Los participantes pueden editar actividades activas, pero no las finalizadas.
* Las descripciones deben incluir: título, dueño y propósito del proyecto.

**Integrantes del proyecto**

Como integrante de un proyecto, me gustaría poder registrar el tiempo que me tomó realizar una actividad, sí que tome en cuenta el tiempo en el que no estuve al tanto del mismo, para poder tener conocimiento de cuanto me estoy tardando en mis labores y poder ser más eficiente. A su vez me gustaría tener una lista de todas las actividades que haya realizado y otra, para las actividades que me faltan por finalizar, para poder estimar los tiempos de entrega. Finalmente, me gustaría que mis compañeros pudieran editar mis actividades cuando yo no pueda o no tenga el tiempo de acceder al programa, y así siempre tener un registro de las actividades que he estado realizando sin que se alteren los tiempos del proyecto.

Para cumplir con las necesidades de usuario, la aplicación debe:

* Darle al usuario un cronómetro por actividad donde pueda pausar y continuar el tiempo cuando lo requiera.
* Mostrar las listas de actividades activas y por finalizar.
* Dar acceso y habilitar la edición, a cualquier miembro del equipo, a las actividades que se encuentran dentro del proyecto.
* **Responsabilidades**

|  |  |  |
| --- | --- | --- |
| No | Responsabilidad | Componente |
| 1 | Interactuar con el usuario | Consola |
| 2 | Inicializar un nuevo proyecto | Proyecto |
| 3 | Cargar archivos planos | Proyecto |
| 4 | Almacenar información clave de un proyecto p | Proyecto |
| 5 | Definir los tipos a usar en el proyecto | Proyecto |
| 6 | Añadir Participantes | Proyecto |
| 7 | Crear Actividades | Proyecto |
| 8 | Editar Actividades | Proyecto |
| 9 | Almacenar información clave de una actividad | Actividad |
| 10 | Crear un cronometro | Actividad |
| 11 | Almacenar la información de un cronómetro | Actividad |
| 12 | Iniciar tiempo | Actividad |
| 13 | Parar tiempo | Actividad |
| 14 | Promediar tiempos | Cronometro |
| 15 | Contabilizar tiempo | Cronometro |
| 16 | Almacenar información de tiempos de un Participante | Reporte |
| 15 | Gestionar WBS | PaqueteTareas |

\*\*\*

* **Colaboraciones**

Existen 4 escenarios principales en los cuales se ve la colaboración entre los componentes de la solución:

1. Cuando se da uso del cronometro para una actividad determinada.Diagrama

   Descripción generada automáticamente
2. Cuando se crea una nueva actividad del proyecto.

Diagrama

Descripción generada automáticamente con confianza media

1. Cuando se edita una nueva actividad del proyecto.

Diagrama

Descripción generada automáticamente

1. Cuando se actualiza la información del proyecto

Diagrama

Descripción generada automáticamente