**Documento de análisis**

**a) Modelo de Dominio:**

Diagrama

Descripción generada automáticamente

**b) Descripción de los requerimientos funcionales:**

1. calcularTiempoDiario(): Información correspondiente al tiempo por días dedicado por un participante para realizar una tarea. Como participante del proyecto deseo tener las horas por día dedicadas a la realización de una tarea.
2. calcularTiempoTotal(): Información correspondiente al tiempo total dedicado por un participante para realizar una tarea. Como participante del proyecto deseo tener el tiempo total tomado por un participante para finalizar la tarea.
3. calcularTiempoPromedio(): Información correspondiente al tiempo promedio que le dedico un participante a la elaboración de una actividad. Como participante del proyecto deseo tener el tiempo promedio dedicado por día para la elaboración de la tarea asignada.
4. crearReporte(): Reporte que contiene la información de las actividades asignadas a cada participante junto con el tiempo diario, el tiempo total y el tiempo promedio de la elaboración de dicha actividad. Como participante del proyecto deseo tener un reporte que me diga las actividades asignadas a cada participante junto con sus estadísticas como las horas diarias dedicadas, el tiempo total y el promedio de tiempo en el que se completó la tarea.
5. crearActividad(): Crear una nueva tarea a realizar en el proyecto general. Como participante quiero crear una nueva actividad que aporte al desarrollo del proyecto ya sea ejecutada por mi o por otro participante del proyecto.
6. modificarFecha(): Cambiar las fechas que aparecen en el registro correspondientes al inicio y fin de una tarea. Como participante deseo cambiar las fechas de inicio y finalización de una tarea.
7. modificarHora(): Cambiar las horas que aparecen en el registro correspondientes al inicio y fin de una tarea. Como participante deseo cambiar las horas de inicio y finalización de una tarea.
8. empezarCronometro(): Iniciar con la medición del tiempo que durará la elaboración de la actividad asignado. Como participante del proyecto deseo iniciar un cronometro que me ayudar a saber el tiempo que me demora realizar una tarea.
9. pausarCronometro(): Suspender la medición del tiempo de realización de una actividad. Como participante del proyecto deseo pausar el cronometro para indicar que hay una pausa en la realización del proyecto.
10. reanudarCronometro(): Retomar la medición del tiempo de realización de una actividad. Como participante del proyecto deseo continuar con la medición de tiempo del cronometro para indicar que la realización del proyecto continuará.
11. finalizarCronometro(): Finalizar con la medición de tiempo del cronometro y registrar automáticamente ese tiempo como el total de horas dedicado a completar la actividad asignada. Como participante del proyecto deseo finalizar la medición del cronometro para indicar que la actividad fue terminada y guardar la medición del cronometro como el tiempo usado para la realización de la actividad.

**c) Restricciones**

* No hay fiabilidad de la mayoría de los datos ingresados puesto que no hay forma de autorizar los datos
* Los participantes del proyecto pueden modificar todos los datos del registro puesto que no hay sistemas de rangos que restrinja que persona es encargada de modificar los datos y quien no puede modificar los datos.
* El programa no contempla la posibilidad de que una actividad sea realizada por dos o mas personas.
* El programa no contempla la posibilidad de delegar tareas a otros participantes.
* El programa no es capaz de reconocer por si solo que una tarea fue realizada en varios días, sino que este debe realizarse por interpretación humana. Es necesario una unión en bloques para actividades que llevan más de un día de trabajo, esto ayudaría a mejorar los cálculos de tiempos.

**d) Diagrama de secuencia**

**Diagrama, Esquemático

Descripción generada automáticamente**

**e) Interfaces**

Página de Inicio de sesión

Diagrama

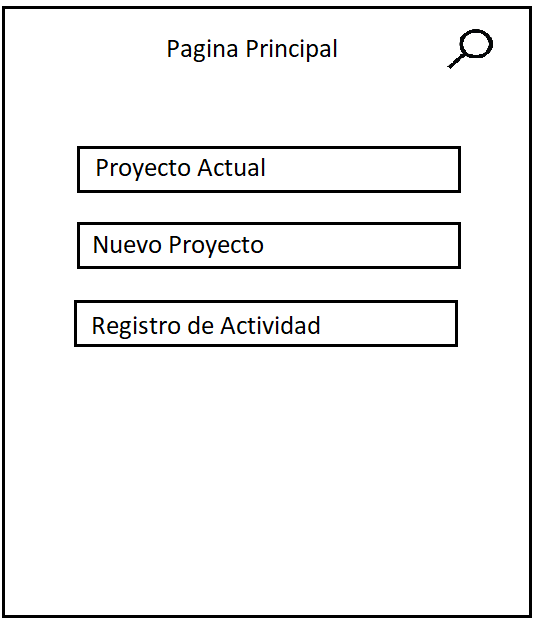
Descripción generada automáticamente con confianza media

Creación de Usuario

Una captura de pantalla de un celular con texto

Descripción generada automáticamente

Página principal

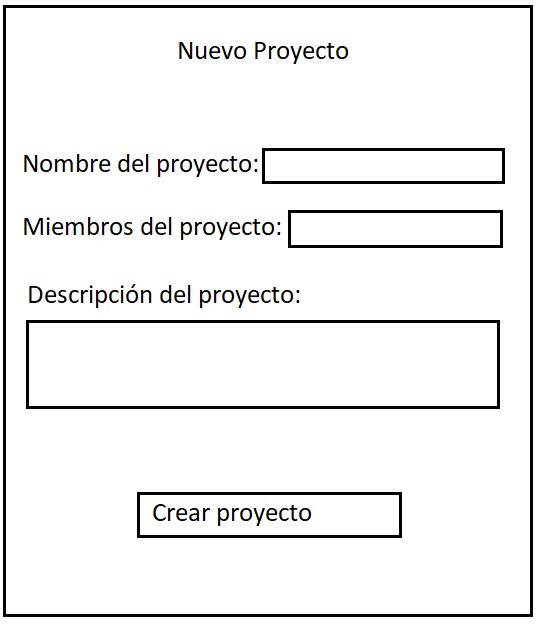
 : Buscar Proyecto

Proyecto actual

Imagen que contiene Tabla

Descripción generada automáticamente

Nuevo proyecto:



Desde la página de inicio de sesión puedo ingresar con un usuario o crear un nuevo usuario, posterior a esto tenemos la página principal en la cual podemos buscar un proyecto en específico, crear un proyecto nuevo, mirar el proyecto actual en el que nos encontramos y mirar todos los proyectos del usuario. Los proyectos cuentan con los miembros del proyecto, cuentan con un tiempo total de desarrollo y botones para controlar las jornadas de trabajo, además, cuenta con un gráfico de actividad del proyecto. Y también está la página nuevo proyecto a la cual podemos acceder desde la página principal y está nos sirve para elaborar la iniciación de un proyecto nuevo junto con todos sus parámetros.

**f) Actualizaciones para la entrega del proyecto 3**

Para este proyecto implementamos nuevas clases y las describiremos a continuación:

* Tarea: Esta nueva clase es la encargada de poseer toda la información alrededor de las tareas. Información como: nombre, descripción, tipo de tarea, tiempo, fecha, una lista de personas encargadas de la elaboración de esta entrega, una lista de actividades asociadas y por último una variable que marca si la tarea ya fue terminada o no. Los métodos asociados a esta clase son:
  + getNombre(): Retorna el nombre asociado a la tarea.
  + getTiempo(): Retorna el tiempo que lleva la tarea.
  + getFecha(): Retorna la fecha asociada con la tarea.
  + getParts(): Retorna la lista de personas encargadas de la tarea.
  + getActs(): Retorna la lista de actividades asociadas a la tarea.
  + getProgreso(): Retorna el estado de la tarea, si esta ya fue terminada o no.
  + setProgreso(): Cambia el estado por defecto de la tarea por terminada.
* Paquete: La clase paquetes es la encargada de contener tareas, almacenas tareas que estén asociadas de una manera u otra. Contiene: un nombre, una descripción, una lista de tareas y una lista de sub-paquetes. Los métodos que posee la clase paquete son:
  + getNombre(): Retorna el nombre actual del paquete.
  + setNombre(): Dado un nuevo nombre, modifica el nombre que poseía el paquete.
  + setDescr(): Dado un String, modifica la descripción que posee el paquete.
  + getTareas(): Retorna la lista de todas las tareas que posee el paquete.
  + getPaquetes(): Retorna la lista de los sub-paquetes que posee el paquete.
  + AgregarTareas(): Dada una tarea, agrega a la lista de tareas del paquete esta nueva tarea.
  + AgregarPacks(): Agrega un sub-paquete a la lista que posee el paquete.
  + EliminarTareas(): Dada una tarea, elimina de la lista de todas las tareas que tiene el paquete esta tarea.
  + EliminarPacks(): Dado un sub-paquete, elimina este sub-paquete de la lista total de todos los sub-paquetes que posee el paquete.

El nuevo UML quedaría de esta forma:

Diagrama, Esquemático

Descripción generada automáticamente

Se decidió agregar únicamente las dos clases correspondientes a cada elemento del nuevo mundo, y unirlas con las que ya existían dentro de la solución. Esto permitió simplificar significativamente el proceso de modificación y reestructuración del programa. Para vincular de la forma más orgánica a la clase tarea, se optó por eliminar la relación que existía previamente entre Proyecto y Actividad, para en su lugar establecer una entre Actividad y Tarea, debido a que en este nuevo contexto estos dos elementos se encuentran más estrechamente relacionados, y dejan en evidencia una relación más directa. Por otro lado, la clase Paquete fue considerada como un elemento principalmente de organización, en el que se puedan categorizar de forma más clara las diferentes tareas y las actividades que vienen dentro de ellas. Como se puede ver, cada paquete también puede estar compuesto por otros paquetes, de manera que se crea una estructura de árbol n-ario bastante interesante, pero que puede resultar contraproducente si no se usan métodos de búsqueda de información adecuados para navegarlo. A nivel general, el diseño de esta tercera implementación es bastante sencillo, pero concreto en sus objetivos, lo que permite afrontar de forma directa los requisitos que se deben cumplir, pero como consecuencia puede que surjan nuevos elementos o situaciones que no se hayan tenido en cuenta y que deban manejarse.

Errores Principales:

Al existir tantas estructuras de almacenamiento de información, uno de los principales errores es que alguna de estas estructuras se encuentre vacía y el programa se vea incapaz de continuar con esta información, para manejar esto, se procura garantizar que todas las estructuras tengan al menos un elemento al intentar realizar una búsqueda dentro de ellas, y así evitar esta dificultad.

Otro error común en este tipo de soluciones son los objetos repetidos, especialmente si se manejan volúmenes de datos grandes; afortunadamente, el programa cuenta con un mecanismo para detectar elementos repetidos, y se lo deja saber al usuario cuando este intenta agregar un elemento que ya se encuentra agregado.

Estrechamente ligado al punto anterior, se debe considerar el error marcado al intentar modificar un elemento que no hace parte de una estructura de información, cosa que no será posible ya que este no existe. Para darle un manejo a esto, existe un mecanismo que compara el elemento del que se quieren hacer las modificaciones con todos los que existen dentro de la estructura de información, y en caso de que no encuentre alguno similar, se lo dejará saber al usuario; adicionalmente, le informará de aquellos que existen dentro del contexto, para que éste pueda elegir uno que cumpla con los requisitos.