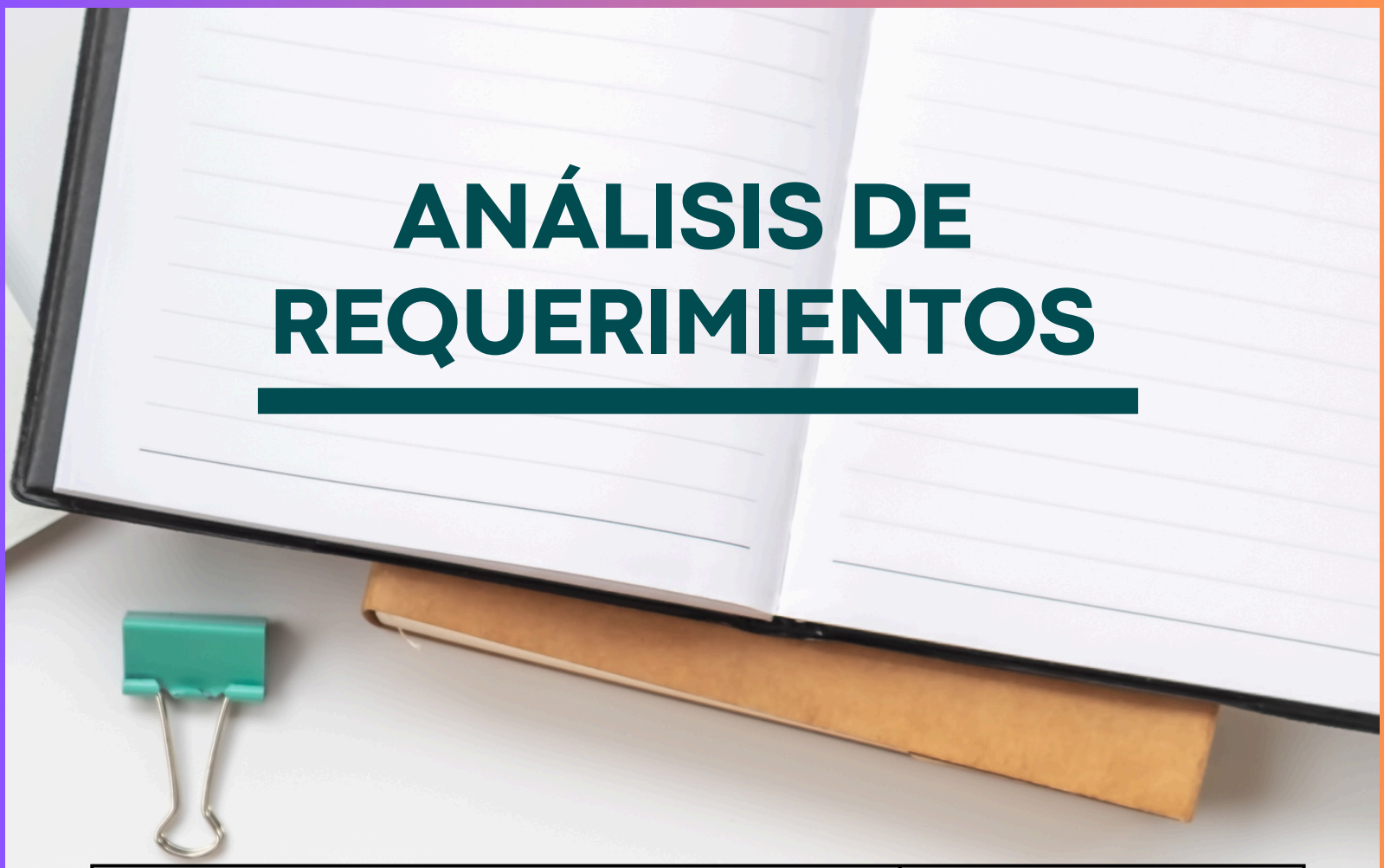


ANÁLISIS DE REQUERIMIENTOS

Nombres: Viktor Raúl Hernández Vargas Leonardo Cerda Rodriguez Joel Sanchez Campos	Matrículas: AL03003728 AL07024282 AL07098869
Nombre del curso: Programación orientada a objetos	Nombre del profesor: HECTOR ANTONIO AGUILAR MOGOLLAN
Módulo: 1	Avance de proyecto
Fecha: 07/02/2025	Equipo: N/A
Bibliografía:	

Índice

Introducción

En esta actividad, trabajaremos en el diseño y modelado de un Sistema de Gestión de Proyectos de Software (SGP) que permita a una empresa de desarrollo de software gestionar sus proyectos, equipos y tareas de manera más eficiente. Actualmente, la organización utiliza herramientas no especializadas, como hojas de cálculo y correos electrónicos, lo que genera desorganización y pérdida de información. Nuestro objetivo con este sistema es optimizar la administración de proyectos y mejorar la comunicación entre los miembros del equipo.

Para lograrlo, nos basaremos en los requisitos funcionales y no funcionales previamente establecidos. Identificaremos las clases y objetos necesarios para representar los elementos clave del SGP, como los proyectos, tareas, equipos de trabajo, miembros del equipo y sus roles. Además, definiremos los atributos y métodos de cada clase, así como las relaciones entre ellas, utilizando conceptos como asociación, agregación, composición y herencia para modelar correctamente la estructura del sistema.

También elaboraremos diagramas UML que representen visualmente el funcionamiento del sistema, incluyendo diagramas de clases, casos de uso, secuencia y estado. Estos diagramas servirán como guía para la implementación, asegurando que todos los componentes y sus interacciones estén bien definidos y alineados con los requerimientos.

Finalmente, documentaremos el diseño y modelado, proporcionando una explicación detallada de cada clase, objeto, relación y comportamiento. Esto garantizará que todos los elementos del sistema sean comprensibles y reflejen correctamente los requisitos planteados, facilitando su implementación y uso dentro de la empresa.

Avance de proyecto – Sistema de Gestión de Proyectos de Software

Una empresa de desarrollo de software requiere un Sistema de Gestión de Proyectos (SGP) que permita organizar sus proyectos, equipos y tareas de manera eficiente. Actualmente, el proceso se gestiona a través de hojas de cálculo y correos electrónicos, lo que genera desorganización y pérdida de información. El objetivo es diseñar un sistema web que facilite la administración de proyectos y la comunicación entre los miembros del equipo

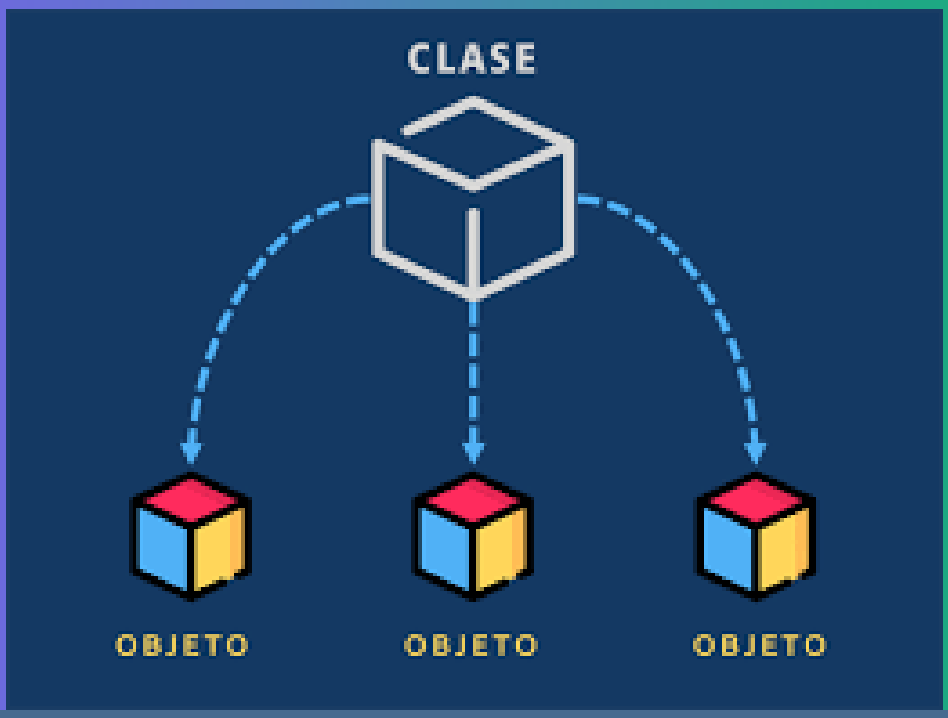
Requisitos Funcionales del Sistema de Gestión de Proyectos de Software (SGP):

1.-Debe permitir crear, editar y eliminar proyectos con atributos como nombre, descripción, fecha de inicio, fecha de finalización, estado y prioridad.
2.-Debe permitir asignar proyectos a los miembros del equipo para que cada uno tenga claro sus responsabilidades dentro de cada proyecto.
.3.-Debe permitir crear, asignar, editar y eliminar tareas dentro de los proyectos, con detalles como nombre, descripción, fecha de inicio, fecha límite, prioridad y estado.
4.-Debe permitir asignar tareas a los miembros del equipo de acuerdo con sus roles y responsabilidades, y permitir que actualicen el estado de las mismas.
5-Debe permitir adjuntar archivos y comentarios a las tareas, para que los miembros del equipo puedan agregar información relevante sobre su ejecución.

Avance de proyecto - Sistema de Gestión de Proyectos de Software

Requisitos Funcionales del Sistema de Gestión de Proyectos de Software (SGP):

6.-Debe permitir visualizar el avance de los proyectos mediante informes que muestren tareas completadas, tareas pendientes y cumplimiento de plazos.
7.-Debe permitir la comunicación entre los miembros del equipo mediante un sistema de mensajería interna y notificaciones automáticas.
8.-Debe permitir la creación de equipos de trabajo y la asignación de roles específicos a los miembros (como desarrollador, líder de proyecto, tester, etc.).
9.-Debe permitir gestionar los archivos relacionados con los proyectos, facilitando el almacenamiento y acceso a documentos como código fuente, informes, etc.
10.-Debe permitir gestionar permisos de acceso a proyectos, tareas y documentos según los roles y responsabilidades de cada miembro del equipo.
11.-Debe permitir generar reportes sobre el estado de los proyectos, el avance de las tareas y el desempeño del equipo.



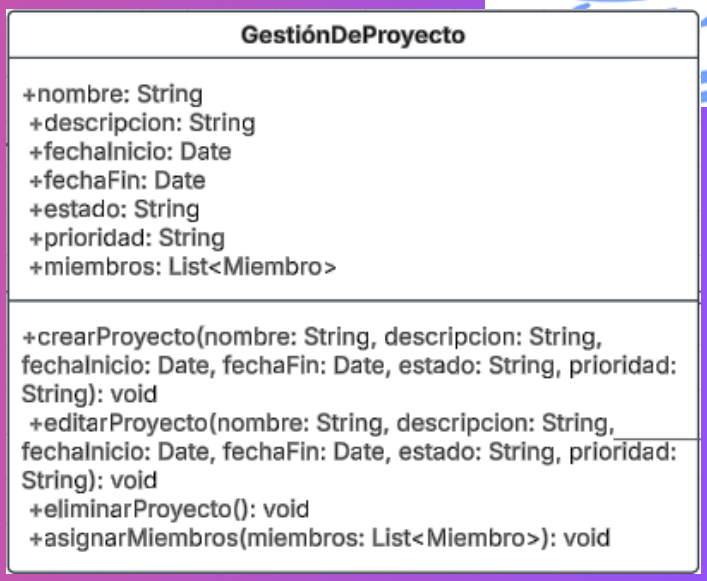
Requisitos No Funcionales del Sistema de Gestión de Proyectos de Software (SGP)

1.-Debe ser fácil de usar e intuitivo, permitiendo que los usuarios sin experiencia técnica puedan navegar sin dificultades.
2.-Debe ser escalable, permitiendo el manejo de un creciente número de proyectos, tareas y usuarios sin que el rendimiento se vea afectado.
3.-Debe garantizar la seguridad de los datos, utilizando encriptación HTTPS para la transmisión de información y protegiendo datos sensibles.
4.-Debe ofrecer tiempos de respuesta rápidos, asegurando que la carga de páginas y la ejecución de tareas se realicen en menos de 3 segundos.
5.-Debe garantizar una alta disponibilidad, con un tiempo de funcionamiento superior al 99.9% y un plan de recuperación ante fallos.
6.Debe permitir un mantenimiento sencillo, con un código modular y bien documentado para facilitar actualizaciones y mejoras.
7.-Debe ser compatible con los principales navegadores web, asegurando que funcione correctamente en Google Chrome, Mozilla Firefox, Safari y Microsoft Edge.
8.-Debe ser responsive, adaptándose a dispositivos con diferentes tamaños de pantalla como computadoras de escritorio, tablets y smartphones.
9.- Debe contar con soporte multilingüe, permitiendo que los usuarios seleccionen su idioma preferido para una mejor experiencia de uso.
10.-Debe incluir documentación completa y detallada tanto para usuarios, administradores y desarrolladores.

Clases

1. Clase: GestiónDeProyecto

Esta clase centraliza la administración de proyectos, incluyendo atributos clave como nombre, descripción, fechas, estado y prioridad. Permite crear, editar y eliminar proyectos, además de asignar miembros. Su diseño responde al requisito funcional de crear y gestionar proyectos con toda la información relevante. Se separa de otras clases para mantener una estructura clara y organizada que facilita la manipulación de proyectos sin interferir con otras funcionalidades.

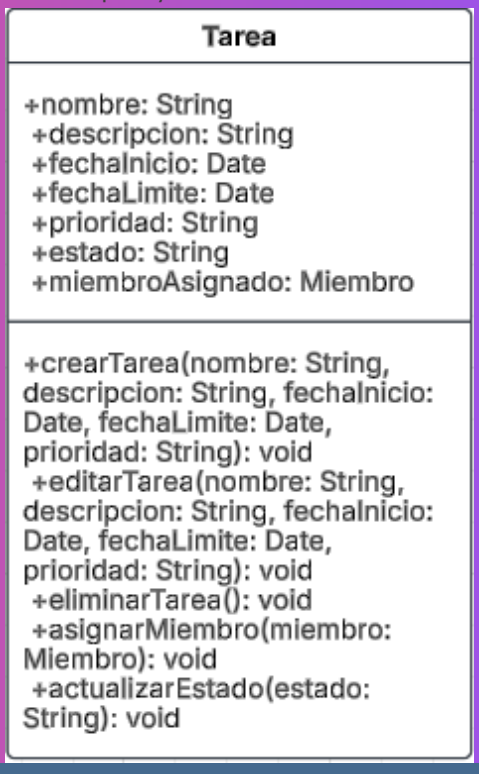


2. Clase: AsignacionMiembros

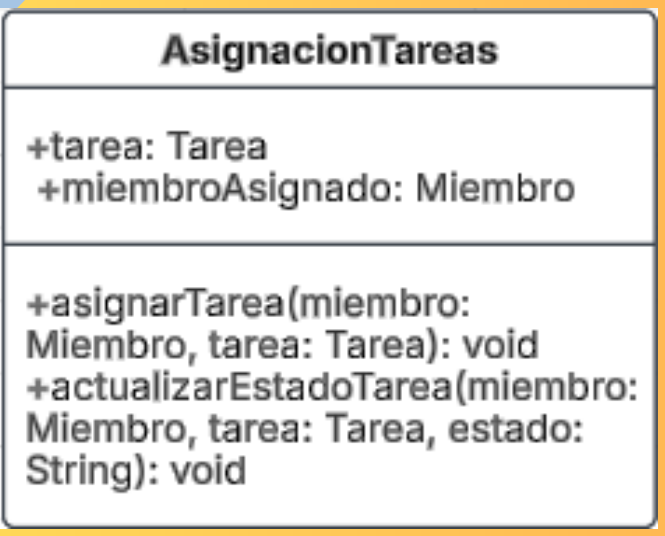
Esta clase gestiona la relación entre los proyectos y los miembros del equipo, permitiendo asignar miembros a proyectos y listar los miembros de un proyecto específico. Responde a la necesidad de asignar responsabilidades claras a los miembros del equipo dentro de cada proyecto, asegurando que las tareas sean realizadas por las personas adecuadas. Su diseño modular permite manejar las asignaciones de manera independiente de la gestión directa de proyectos.

3. Clase: Tarea

Representa las tareas dentro de los proyectos, con detalles como nombre, descripción, fechas, prioridad y estado. Permite crear, editar, eliminar tareas y asignarlas a miembros. Esto cubre el requisito de crear, editar, asignar y seguir el progreso de las tareas. El diseño de esta clase asegura un seguimiento detallado de cada tarea y una asignación específica de responsabilidades, lo cual es esencial para el control del avance del proyecto.



Clases

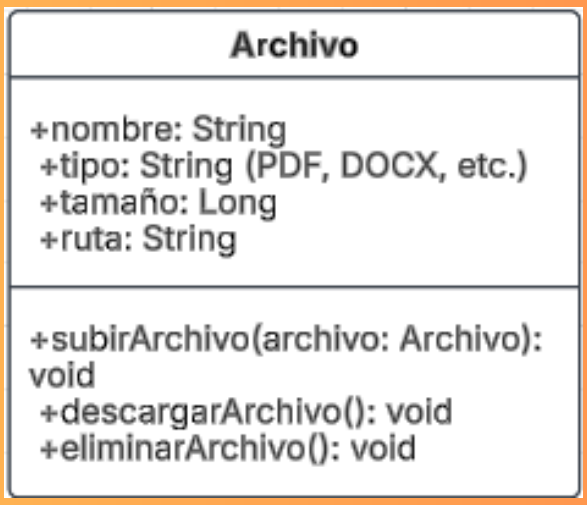


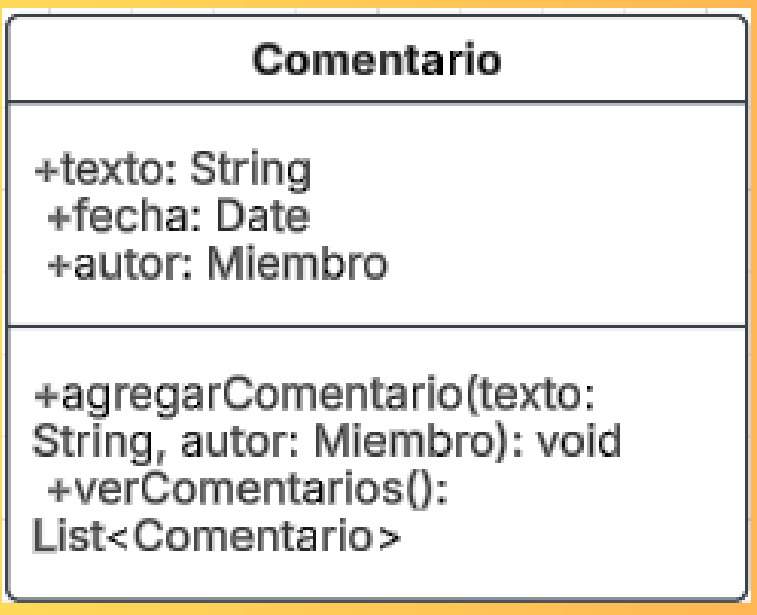
4. Clase: AsignacionTareas

Esta clase relaciona tareas con miembros específicos y permite asignar tareas y actualizar su estado. Se diseñó para gestionar de manera efectiva la distribución de tareas y el seguimiento de su progreso. Así, cumple con los requisitos de asignar tareas a miembros según sus roles y responsabilidades, además de permitirles actualizar el estado de las mismas.

5. Clase: Archivo

Permite la gestión de archivos relacionados con proyectos y tareas, como documentos y código fuente. Responde al requisito de almacenar y acceder a archivos relevantes de proyectos. El diseño de esta clase facilita la carga, descarga y eliminación de archivos, asegurando que la información clave esté siempre accesible para los miembros del equipo.





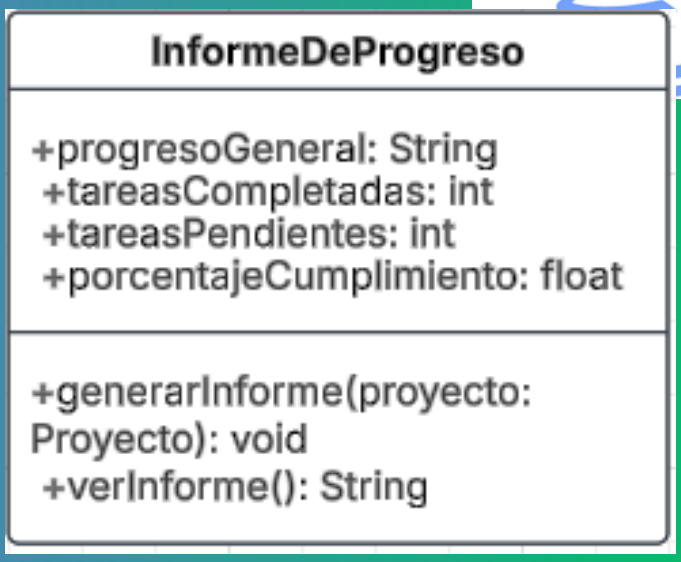
6. Clase: Comentario

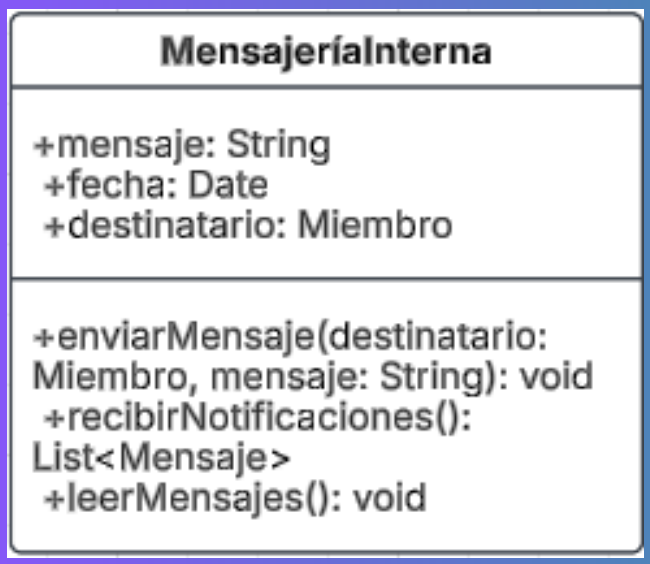
Facilita la comunicación entre miembros a través de comentarios dentro de tareas o proyectos. Permite agregar y visualizar comentarios, lo que mejora la colaboración y el seguimiento de los proyectos. Esto cumple con el requisito de permitir a los miembros agregar información relevante sobre la ejecución de las tareas. La clase está diseñada para que los comentarios sean fácilmente accesibles y gestionables.

Clases

7. Clase: InformeDeProgreso

Esta clase genera informes sobre el progreso del proyecto, incluyendo tareas completadas, tareas pendientes y porcentaje de cumplimiento. Responde a la necesidad de monitorear y visualizar el avance de los proyectos. El diseño de la clase facilita la generación de informes detallados, que son cruciales para los informes de avance requeridos por los usuarios del sistema.



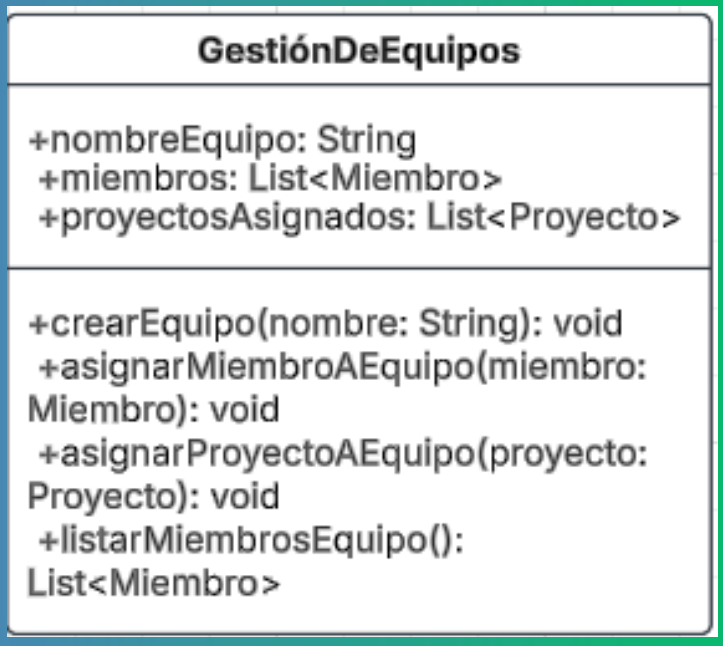


8. Clase: MensajeríaInterna

Permite la comunicación directa entre miembros mediante mensajes internos y notificaciones automáticas. Cumple con el requisito de facilitar la comunicación entre los miembros del equipo. Su diseño modular permite enviar, recibir y leer mensajes sin interferir en otras funcionalidades del sistema, lo que optimiza la comunicación.

9. Clase: GestiónDeEquipos

Esta clase gestiona la creación y asignación de equipos de trabajo, así como la asignación de proyectos a esos equipos. Responde al requisito de crear equipos de trabajo y asignar roles específicos a los miembros. Su diseño asegura que los equipos estén correctamente organizados y puedan ser asignados a proyectos según las necesidades de la organización.



Clases

PermisosAcceso
+usuario: Miembro +proyecto: Proyecto +nivelAcceso: String (Lectura, Escritura, Administrador)
+asignarPermiso(usuario: Miembro, proyecto: Proyecto, nivelAcceso: String): void +verPermisos(usuario: Miembro, proyecto: Proyecto): String +revocarPermiso(usuario: Miembro, proyecto: Proyecto): void

10. Clase: PermisosAcceso

Gestiona los permisos de acceso de los usuarios a proyectos, tareas y documentos, de acuerdo con su rol (lectura, escritura, administrador). Responde a los requisitos de gestionar los accesos según los roles y responsabilidades de los miembros del equipo, lo que asegura la seguridad y el control adecuado de la información dentro del sistema.

11. Clase: ReporteDeDesempeño

Genera informes sobre el desempeño de los miembros del equipo, incluyendo tareas completadas, pendientes y su productividad. Responde al requisito de generar reportes sobre el desempeño del equipo. El diseño de esta clase permite generar reportes claros y detallados que permiten evaluar el rendimiento de los miembros del equipo y su contribución a los proyectos.

ReporteDeDesempeño
+proyecto: Proyecto +miembro: Miembro +tareasCompletadas: int +tareasPendientes: int +productividad: float
+generarReporteDesempeño(miembro: Miembro): void +verReporteDesempeño(): String

Explicación de clases

Para diseñar el Sistema de Gestión de Proyectos de Software (SGP), nos basamos en los requisitos funcionales establecidos, asegurando que cada clase cumpliera un rol específico dentro del sistema para gestionar proyectos, tareas y equipos de manera eficiente.

La clase GestiónDeProyecto centraliza la administración de los proyectos, permitiendo crearlos, editarlos y eliminarlos, así como asignar miembros. Sus atributos clave, como nombre, descripción, fechas de inicio y fin, estado y prioridad, facilitan la gestión de la información esencial para cada proyecto. Esto responde al requisito de estructurar y manejar proyectos con toda la información relevante.

Por otro lado, la clase AsignacionMiembros gestiona la relación entre proyectos y miembros, asegurando que cada proyecto tenga asignadas las personas adecuadas. Esto garantiza que las tareas sean realizadas por quienes corresponda y se mantenga una distribución clara de responsabilidades. La clase Tarea representa las actividades dentro de los proyectos, permitiendo crearlas, asignarlas, editarlas y eliminarlas. Sus atributos, como nombre, descripción, fechas, prioridad y estado, son fundamentales para su correcta gestión. Además, la posibilidad de asignar miembros y actualizar el estado de cada tarea facilita el control del progreso del proyecto.

Para vincular tareas con miembros específicos y hacer seguimiento a su avance, la clase AsignacionTareas se encarga de gestionar esta relación, asegurando que cada tarea esté correctamente asignada. También consideramos la importancia de manejar información adicional, por lo que incorporamos la clase Archivo, que permite cargar, descargar y eliminar documentos relacionados con proyectos y tareas. Esto facilita el intercambio de información clave dentro del equipo, como documentos y código fuente, cumpliendo con la necesidad de gestionar archivos en el sistema.

La comunicación es un aspecto esencial en la gestión de proyectos, por lo que implementamos la clase Comentario, que permite a los miembros del equipo agregar y visualizar comentarios en tareas y proyectos, promoviendo la colaboración y el seguimiento adecuado. Además, la clase InformeDeProgreso permite generar reportes sobre el estado de los proyectos, mostrando tareas completadas, pendientes y el cumplimiento de los plazos, lo que resulta fundamental para monitorear el avance.

Explicación de clases

Para mejorar la comunicación interna del equipo, la clase MensajeríaInterna facilita el envío de mensajes directos y notificaciones automáticas, asegurando que la información relevante llegue a los destinatarios oportunos. A su vez, la clase GestiónDeEquipos permite la creación de equipos de trabajo y la asignación de roles específicos a sus miembros, cumpliendo con el requisito de organizar los equipos y distribuir responsabilidades dentro de los proyectos.
En términos de seguridad, consideramos clave la clase PermisosAcceso, que gestiona los permisos de los usuarios según su rol dentro del equipo, asegurando que cada persona tenga acceso solo a la información necesaria. Finalmente, incorporamos la clase ReporteDeDesempeño, que permite evaluar el rendimiento de los miembros del equipo mediante informes sobre la cantidad de tareas completadas y pendientes, proporcionando métricas clave para analizar la productividad y contribución de cada integrante al éxito del proyecto.
En conjunto, todas estas clases fueron diseñadas para cumplir con los requisitos del SGP, garantizando que el sistema sea funcional, eficiente y capaz de cubrir todas las necesidades de gestión de proyectos. Desde la creación de proyectos hasta el seguimiento del desempeño del equipo, cada clase y sus métodos están alineados con el objetivo de proporcionar un control claro y estructurado sobre los proyectos, tareas, miembros, archivos y comunicaciones, asegurando así una gestión integral del proceso.

Relaciones entre las clases y los objetos

1. **Asociaciones (Relación estándar con multiplicidad):**
GestiónDeProyecto (1) —— (0..) AsignacionMiembros
GestiónDeProyecto (1) —— (0..) Tarea
GestiónDeProyecto (1) —— (0..) InformeDeProgreso
GestiónDeProyecto (1) —— (0..) PermisosAcceso
GestiónDeEquipos (1) —— (1..) AsignacionMiembros
GestiónDeEquipos (1) —— (0..) GestiónDeProyecto
Tarea (1) —— (0..) Comentario
Tarea (1) —— (0..) Archivo
AsignacionTareas (1) —— (1) AsignacionMiembros
AsignacionTareas (1) —— (1) Tarea
ReporteDeDesempeño (1) —— (1) GestiónDeProyecto
ReporteDeDesempeño (1) —— (1) AsignacionMiembros
MensajeríaInterna (1) —— (1..*) AsignacionMiembros

2. Agregación (Relación débil, objetos pueden existir independientemente):
GestiónDeEquipos —◆ (1..) AsignacionMiembros
AsignacionMiembros —◆ (1..) AsignacionMiembros

3. Composición (Relación fuerte, objetos dependen de su contenedor):
GestiónDeProyecto —◆ (1..) Tarea
GestiónDeProyecto —◆ (1..) InformeDeProgreso
GestiónDeProyecto —◆ (1..) PermisosAcceso
Tarea —◆ (0..) Comentario
Tarea —◆ (0..*) Archivo

4. Herencia (Generalización / Especialización):
ReporteDeDesempeño —△ (1) InformeDeProgreso
GestiónDeEquipos —△ (1) AsignacionMiembros
PermisosAcceso —△ (1) GestiónDeProyecto

Relaciones entre las clases y los objetos explicación

Lo que hicimos fue seleccionar distintos tipos de relaciones entre las clases con el objetivo de reflejar el comportamiento real del sistema y las interacciones entre sus diferentes entidades. A continuación, explicamos cómo utilizamos cada tipo de relación y por qué las elegimos.

En primer lugar, aplicamos la asociación para conectar clases que pueden existir de manera independiente, pero que están relacionadas entre sí, permitiendo que una clase contenga múltiples instancias de otra. Un ejemplo de esto es la relación entre GestiónDeProyecto y AsignacionMiembros: un proyecto puede tener múltiples miembros asignados, aunque no es obligatorio que todos los proyectos tengan miembros en todo momento. Por ello, establecimos una multiplicidad de (0..), brindando flexibilidad en la asignación de miembros. Lo mismo ocurre en la relación entre GestiónDeProyecto y Tarea. Un proyecto puede contar con varias tareas, pero no todas las tareas son esenciales para un proyecto, por lo que definimos una relación de uno a muchos (1..).

Además, en GestiónDeProyecto, un proyecto puede contar con múltiples informes de progreso y permisos de acceso, aunque estos no necesariamente deben existir desde el inicio, lo que refleja un comportamiento dinámico dentro del sistema. En el caso de la relación entre GestiónDeEquipos y AsignacionMiembros, establecimos que un equipo debe tener al menos un miembro, pero no todos los equipos necesitan estar asignados a proyectos, justificando una relación de uno a muchos (1..*).

Para la relación entre Tarea y Comentario/Archivo, definimos que las tareas pueden contener comentarios o archivos, pero estos no son obligatorios, por lo que la relación es opcional con multiplicidad (0..*). Finalmente, en la relación entre AsignacionTareas, AsignacionMiembros y Tarea, optamos por modelarlas como relaciones uno a uno, ya que cada tarea debe estar asignada a un solo miembro, y cada asignación de tarea está vinculada a una tarea específica.

Por otro lado, utilizamos la agregación para representar relaciones más débiles, en las que los objetos pueden existir de manera independiente pero están agrupados de alguna forma. En el caso de GestiónDeEquipos y AsignacionMiembros, un equipo tiene miembros asignados, pero estos pueden existir sin necesidad de estar en un equipo o incluso pertenecer a múltiples equipos al mismo tiempo. Esta relación nos permitió reflejar que los miembros siguen siendo entidades independientes, aunque formen parte de un equipo.

Relaciones entre las clases y los objetos explicación

También modelamos una relación recursiva en AsignacionMiembros, lo que nos permitió representar que la asignación de miembros a proyectos puede involucrar múltiples asignaciones, ya que un mismo miembro puede estar asignado a varios proyectos y tareas de manera independiente.

Para las relaciones de composición, las utilizamos en aquellos casos en los que los objetos no pueden existir sin su entidad contenedora. En estas relaciones, los objetos dependen completamente de la existencia del objeto principal. Por ejemplo, en GestiónDeProyecto con Tarea, InformeDeProgreso y PermisosAcceso, estos elementos no pueden existir sin un proyecto. Si un proyecto es eliminado, sus tareas, informes de progreso y permisos también lo serán, lo que indica una fuerte dependencia entre estas clases.

Lo mismo ocurre en la relación entre Tarea y Comentario/Archivo. Los comentarios y archivos no tienen sentido fuera de una tarea, por lo que si esta se elimina, ellos también desaparecerán. Esta relación nos permitió modelar una dependencia total entre las tareas y los elementos que contienen.

Finalmente, utilizamos la herencia para modelar clases que comparten atributos o comportamientos comunes, pero que tienen particularidades que las hacen únicas en su implementación. Por ejemplo, en el caso de ReporteDeDesempeño e InformeDeProgreso, el ReporteDeDesempeño es una especialización del informe de progreso, ya que ambos comparten características como los datos sobre el avance del proyecto, pero el reporte de desempeño añade una dimensión de análisis del rendimiento de los miembros, lo que justifica el uso de herencia.

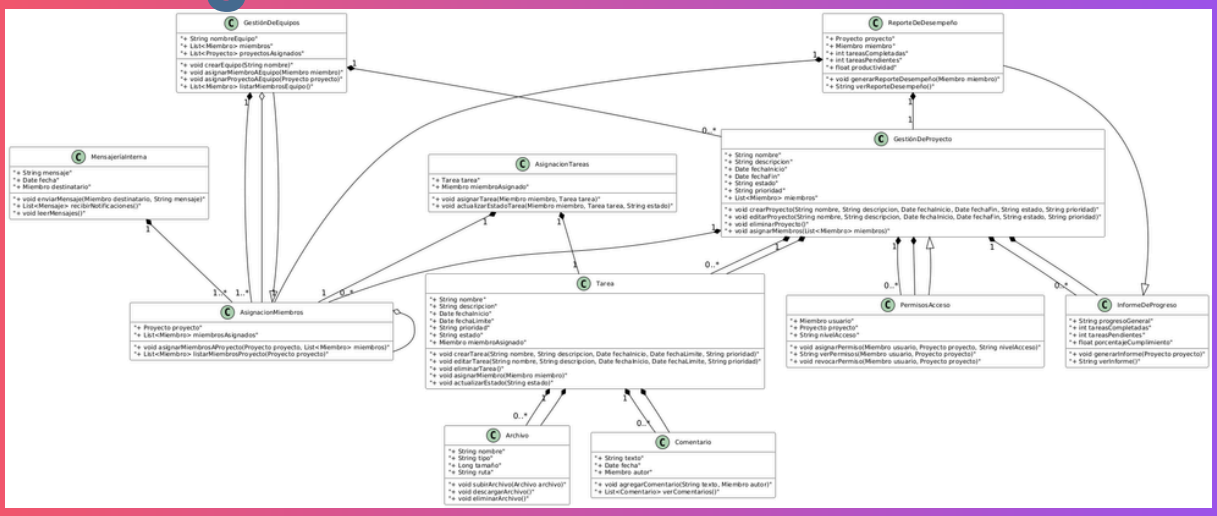
Otra relación de herencia la aplicamos entre GestiónDeEquipos y AsignacionMiembros, ya que la asignación de miembros es un aspecto clave dentro de la gestión de equipos. Aunque el concepto de asignación de miembros es general, dentro del contexto de equipos representa una especialización.

En cuanto a la relación entre PermisosAcceso y GestiónDeProyecto, modelamos los permisos de acceso como una especialización de la gestión de proyectos, ya que los permisos solo tienen sentido dentro de este contexto y deben ser gestionados como parte del mismo.

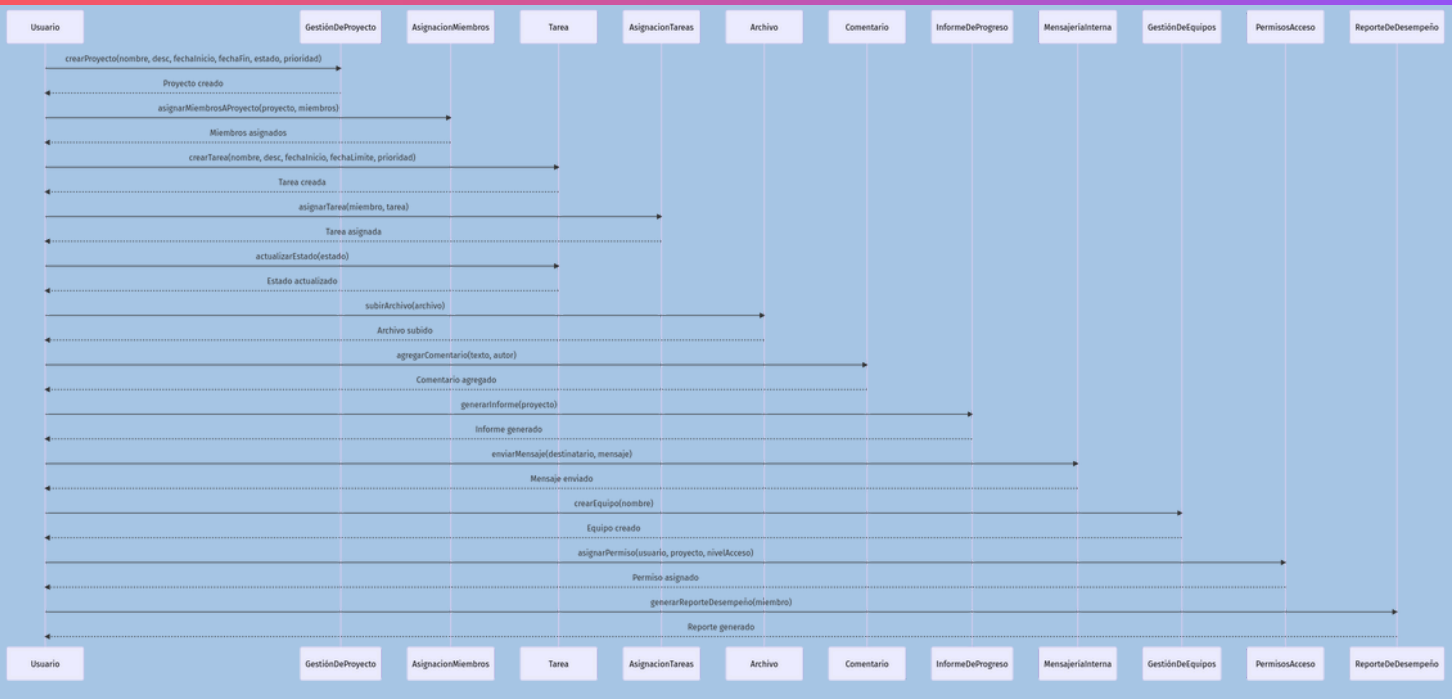
Estas decisiones nos permitieron diseñar un modelo sólido que refleja la naturaleza de los objetos y sus interacciones dentro del sistema, asegurando flexibilidad, dependencia adecuada y especialización cuando es necesario. Además, nos aseguramos de que el modelo sea robusto y escalable, permitiendo su adaptación a medida que el sistema crezca y evolucionen los requisitos.

Diagramas

1.-Diagrama de clases



2.-Diagrama de secuencia.



Conclusión

En esta actividad, aprendimos la importancia de contar con un Sistema de Gestión de Proyectos de Software (SGP) especializado para abordar los desafíos de una empresa de desarrollo de software. Inicialmente, identificamos que el uso de herramientas no especializadas, como hojas de cálculo y correos electrónicos, dificultaba la organización y el seguimiento eficiente de los proyectos. Esto nos llevó a diseñar una solución integral basada en una arquitectura robusta y estructurada.

Para la implementación, definimos once clases principales, cada una con responsabilidades específicas y bien delimitadas. Estas clases abarcan desde la gestión básica de proyectos (GestiónDeProyecto) hasta aspectos más detallados como la mensajería interna y los reportes de desempeño. A lo largo del proceso, nos enfocamos en establecer relaciones adecuadas entre las clases, utilizando distintos tipos de asociaciones, incluyendo agregación, composición y herencia. Esto nos permitió lograr una estructura flexible pero coherente, asegurando que el sistema pudiera escalar y adaptarse a futuras necesidades.

El diseño se vio respaldado por el uso de diagramas UML, específicamente el diagrama de clases y el diagrama de secuencia. El primero nos permitió visualizar de manera clara la arquitectura del sistema y las conexiones entre clases como GestiónDeProyecto, AsignacionMiembros, Tarea e InformeDeProgreso. También modelamos la herencia entre ReporteDeDesempeño e InformeDeProgreso, así como entre GestiónDeEquipos y AsignacionMiembros. Por otro lado, el diagrama de secuencia nos ayudó a comprender el flujo dinámico de interacciones dentro del sistema, garantizando que cada funcionalidad estuviera correctamente implementada.

Además, consideramos aspectos clave como la seguridad, mediante la clase PermisosAcceso, y la comunicación efectiva dentro del equipo, implementando MensajeríaInterna y Comentario. También aseguramos un seguimiento detallado del progreso del proyecto con las clases InformeDeProgreso y ReporteDeDesempeño.

Por lo tanto, con este sistema logramos una solución más organizada, eficiente y profesional, que facilitará la colaboración entre equipos y permitirá un mejor seguimiento de los proyectos de desarrollo de software. Esta experiencia nos permitió reforzar nuestros conocimientos en modelado de sistemas, diseño orientado a objetos y gestión de proyectos, validando la importancia de una estructura bien definida en el desarrollo de software.

Links a github

https://github.com/ViktorHernandez/Programacion-Orientada-a-Objetos.git

git@github.com:ViktorHernandez/Programacion-Orientada-a-Objetos.git