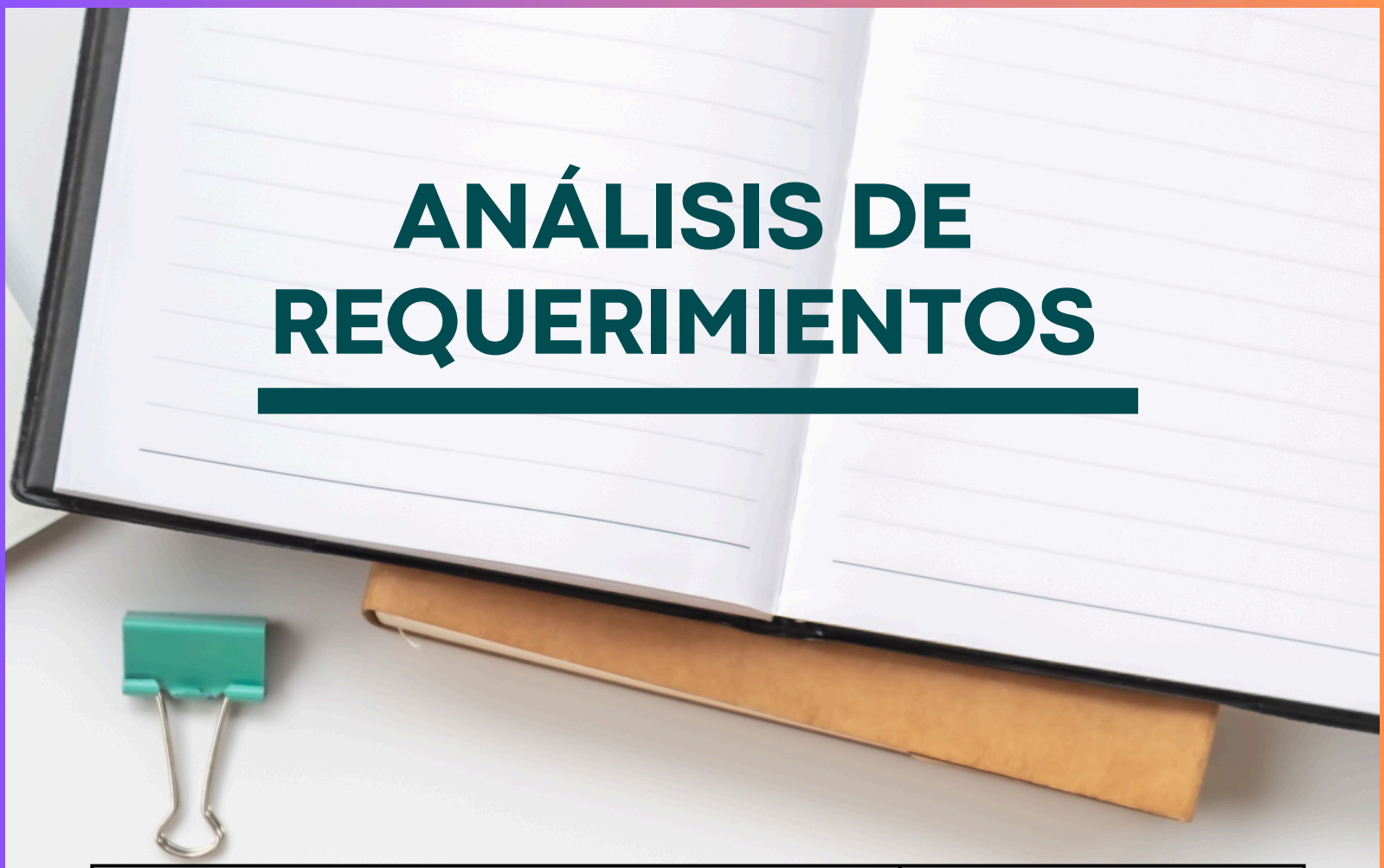


ANÁLISIS DE REQUERIMIENTOS

Nombres: Viktor Raúl Hernández Vargas Joel Sanchez Campos	Matrículas: AL03003728 AL07098869
Nombre del curso: Programación orientada a objetos	Nombre del profesor: HECTOR ANTONIO AGUILAR MOGOLLAN
Módulo: 1	Proyecto Final De Programación Orientada a Objetos
Fecha: 28/02/2025	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.- El sistema debe permitir el inicio de sesión de los usuarios con credenciales únicas y validar su nivel de permisos antes de conceder acceso a las funcionalidades correspondientes.
2.- Debe permitir la creación, edición y eliminación de proyectos, asegurando que cada proyecto tenga un nombre, descripción, fecha de inicio, fecha de finalización, estado y prioridad.
3.- Los administradores deben poder asignar y eliminar miembros de un proyecto, especificando el rol de cada miembro dentro del mismo.
4.- El sistema debe proporcionar la funcionalidad de gestionar tareas dentro de los proyectos, permitiendo su creación, asignación a miembros, actualización de estado y eliminación.
5.- Cada tarea debe contener un nombre, descripción, fecha de creación, fecha límite, prioridad y estado.

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

6.- Los usuarios deben poder agregar comentarios a las tareas asignadas para mejorar la comunicación y el seguimiento del progreso.
7.- El sistema debe permitir la generación de informes de progreso para cada proyecto, incluyendo el número de tareas completadas y pendientes, así como el porcentaje de avance.
8.- Los usuarios deben poder crear y administrar equipos de trabajo, asignando miembros y gestionando su composición.
9.- Se debe incluir un módulo de mensajería interna para la comunicación entre los miembros del sistema.
10.- Los administradores deben poder cambiar los niveles de permisos de los miembros y gestionar sus cuentas dentro del sistema.
11.- El sistema debe permitir la visualización de todas las tareas, equipos y proyectos asignados a cada usuario.
12.- Los usuarios deben poder actualizar su perfil, incluyendo el cambio de nombre de usuario y contraseña.
13.- Se debe permitir la asignación de tareas a proyectos específicos para mantener un orden en la planificación.
14.- Los administradores deben poder generar reportes de desempeño de los miembros basados en su progreso en los proyectos.
15.- Debe existir un control de permisos para garantizar que solo los usuarios con los niveles adecuados puedan realizar ciertas acciones dentro del sistema.

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

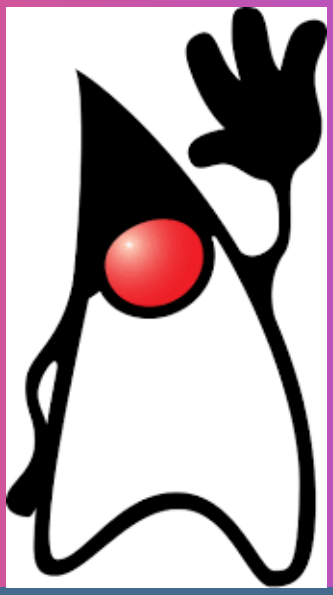
1. El sistema debe ejecutarse en un entorno basado en consola utilizando Java y NetBeans, garantizando compatibilidad con la ejecución en terminales.
2. La interfaz debe ser intuitiva, mostrando menús estructurados y opciones numeradas para una navegación clara y eficiente.
3. Debe manejar correctamente las excepciones para evitar errores inesperados y proporcionar mensajes de error comprensibles que ayuden a los usuarios a corregir problemas.
4. El sistema debe ser capaz de gestionar múltiples proyectos, tareas y equipos sin degradar el rendimiento, permitiendo operaciones simultáneas sin afectar la velocidad de respuesta.
5. La arquitectura del sistema debe ser modular, dividiendo funcionalidades en clases bien definidas para facilitar el mantenimiento y futuras ampliaciones.
6. Las operaciones del sistema, como la creación, edición o eliminación de proyectos y tareas, deben ejecutarse en tiempo real sin demoras perceptibles para el usuario.
7. El sistema debe validar las entradas de los usuarios, asegurando que los datos ingresados sean correctos y evitando inconsistencias o valores inválidos.
8. Debe permitir la asignación y gestión flexible de miembros en proyectos y tareas, garantizando que los cambios sean reflejados de manera instantánea.
9. La información de los proyectos, tareas y usuarios debe permanecer disponible durante la ejecución del programa, evitando la pérdida de datos en la sesión actual.
10. Debe ser estable y confiable, asegurando una ejecución continua sin necesidad de reinicios frecuentes debido a errores o bloqueos.
11. El sistema debe utilizar nombres de variables y métodos descriptivos que reflejen su propósito dentro del código, mejorando su mantenibilidad y legibilidad.
12. El sistema debe garantizar consistencia en la experiencia de usuario, asegurando que todas las funcionalidades operen de manera uniforme en todas las interacciones.

Clases

1. Clase: SistemaGestionProyectos

La clase SistemaGestionProyectos se encarga de facilitar la interacción del usuario con el sistema de gestión de proyectos. Se diseñó con métodos y atributos que permiten crear proyectos, asignar miembros, gestionar tareas y equipos, así como manejar la mensajería entre usuarios. Esto proporciona una interfaz amigable que permite a los usuarios realizar diversas acciones de manera intuitiva. Su relación con GestorProyectos es crucial, ya que esta clase maneja la lógica de negocio y el almacenamiento de datos, mientras que Usuario representa a los individuos que utilizan el sistema.

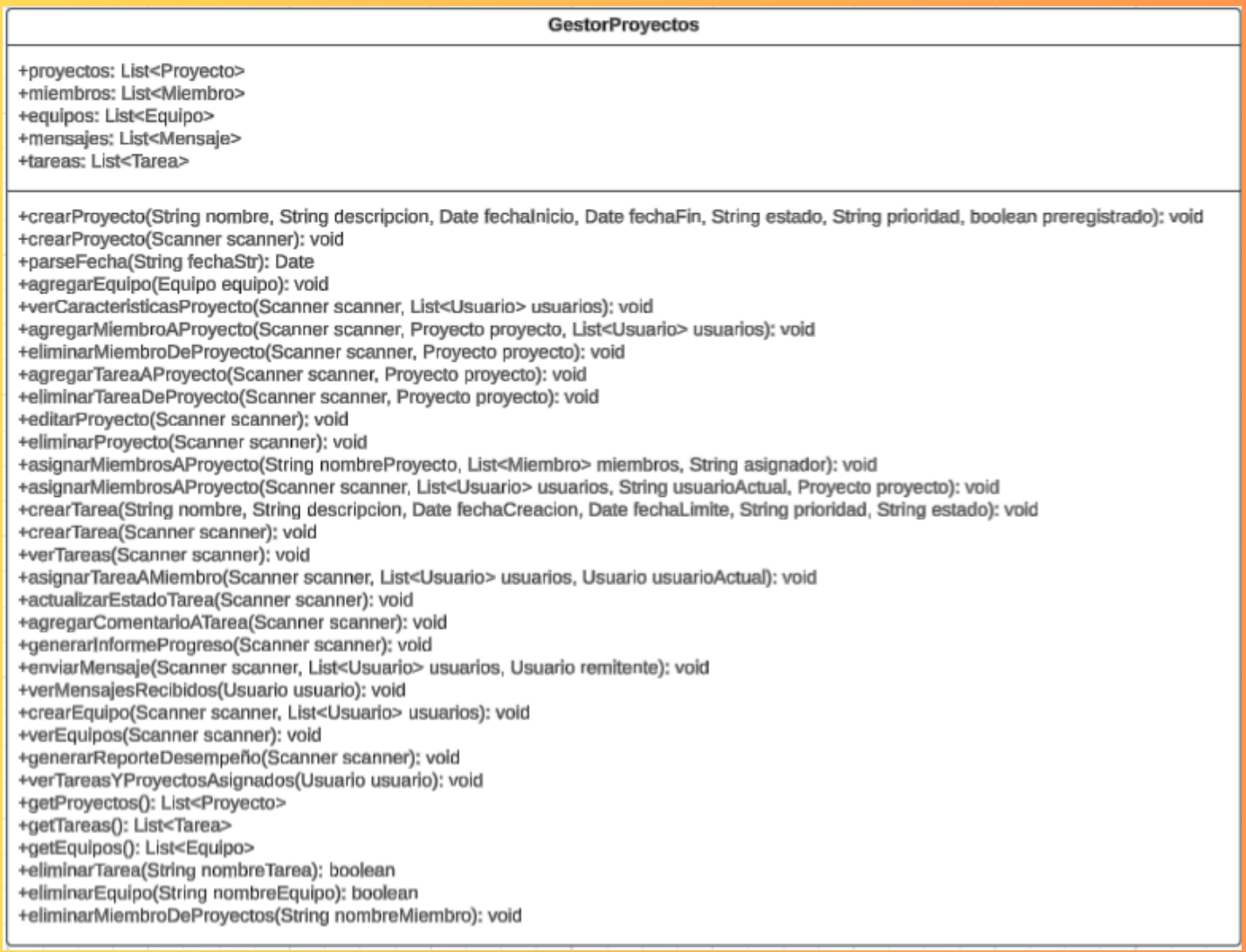
SistemaGestionProyectos
No tiene atributos porque controla el flujo de la aplicación y gestiona la interacción del usuario, utilizando instancias de otras clases para manejar datos y lógica. Sus métodos son estáticos, lo que simplifica el diseño y facilita el mantenimiento.
+main(String[] args): void +crearProyectosPreregistrados(GestorProyectos gestorProyectos): void +asignarMiembrosPreregistrados(GestorProyectos gestorProyectos): void +crearTareasPreregistradas(GestorProyectos gestorProyectos): void +crearEquiposPreregistrados(GestorProyectos gestorProyectos): void +menuPrincipal(Scanner scanner, GestorProyectos gestorProyectos, Usuario usuario, List<Usuario> usuarios): void +verTareasAsignadas(Usuario usuario, GestorProyectos gestorProyectos): void +verEquiposAsignados(Usuario usuario, GestorProyectos gestorProyectos): void +verProyectosAsignados(Usuario usuario, GestorProyectos gestorProyectos): void +ajustesPerfil(Scanner scanner, Usuario usuario, GestorProyectos gestorProyectos): void +actualizarNombreMiembroEnProyectos(String antiguoNombre, String nuevoNombre, GestorProyectos gestorProyectos): void +crearNuevoMiembro(Scanner scanner, List<Usuario> usuarios): void +iniciarSesion(Scanner scanner, List<Usuario> usuarios): Usuario +menuMensajeria(Scanner scanner, GestorProyectos gestorProyectos, Usuario usuario, List<Usuario> usuarios): void +asignarTareaAProyecto(Scanner scanner, GestorProyectos gestorProyectos): void +cambiarNivelPermisos(Scanner scanner, List<Usuario> usuarios): void +eliminarTarea(Scanner scanner, GestorProyectos gestorProyectos): void +eliminarEquipo(Scanner scanner, GestorProyectos gestorProyectos): void +eliminarMiembro(Scanner scanner, List<Usuario> usuarios, GestorProyectos gestorProyectos, Usuario usuario): void +verMiembrosDisponibles(List<Usuario> usuarios, GestorProyectos gestorProyectos): void



Clases

2. Clase: GestorProyectos

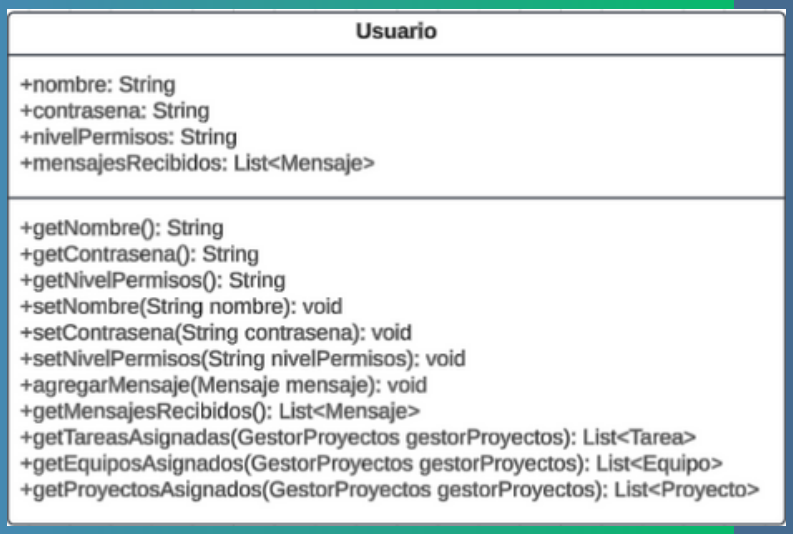
Dentro de GestorProyectos, se ha creado una estructura que almacena y gestiona toda la información relacionada con proyectos, miembros, tareas, equipos y mensajes. Los métodos y atributos de esta clase permiten crear, editar y eliminar proyectos, así como asignar tareas y miembros. Esto asegura una gestión integral y organizada de la información del sistema. Se relaciona con Proyecto, Miembro, Tarea, Equipo y Mensaje, ya que contiene listas de estos objetos y facilita la interacción entre ellos.



Clases

3. Clase: Usuario

Con la clase Usuario, se establecen las características que definen a los usuarios del sistema, como su nombre, contraseña y nivel de permisos. Los métodos y atributos permiten gestionar la información del usuario y su interacción con el sistema, incluyendo la recepción de mensajes y la obtención de tareas, equipos y proyectos asignados. Esto garantiza que cada usuario tenga un perfil único y que sus permisos sean gestionados adecuadamente. La relación con GestorProyectos es clave, ya que esta clase utiliza la información del usuario para asignar tareas y proyectos, y también se vincula con Mensaje para facilitar la comunicación.



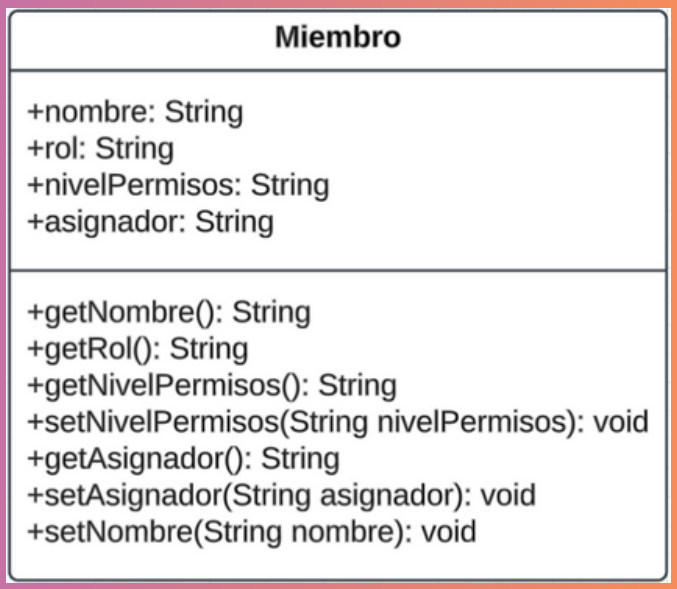
4. Clase: Proyecto

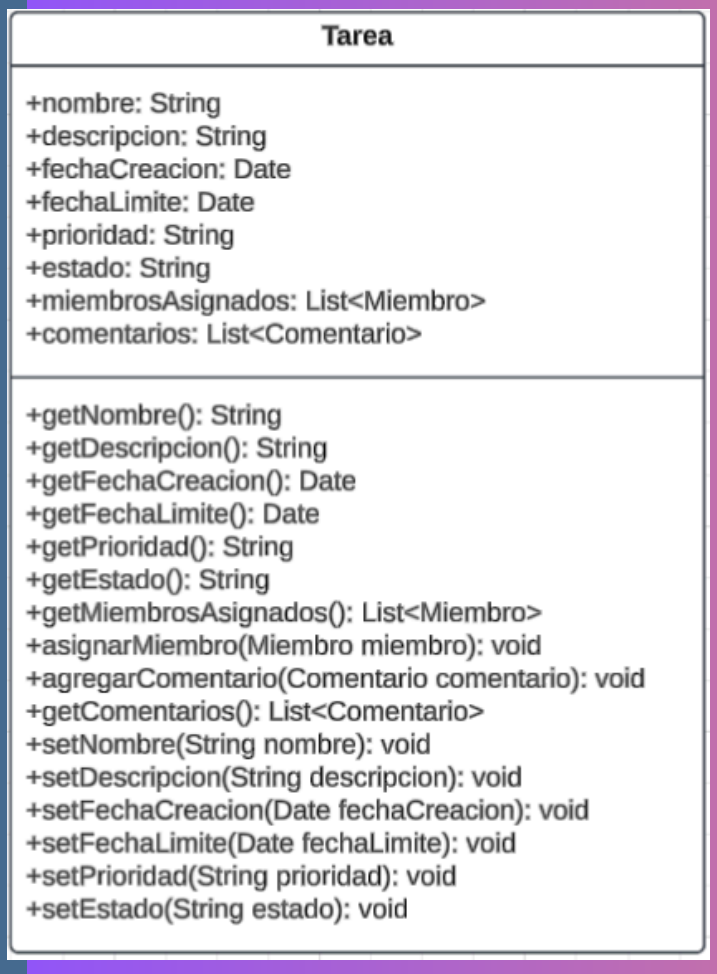
En Proyecto, se definen y gestionan las características de un proyecto, como su nombre, descripción, fechas, estado, prioridad, miembros y tareas. Los métodos y atributos permiten realizar acciones como asignar tareas y miembros a un proyecto. Esto proporciona una estructura clara y organizada para cada proyecto, facilitando su gestión. Se relaciona con Miembro y Tarea, ya que puede contener listas de estos objetos, y también con ReporteDesempeño, que permite generar informes sobre el desempeño del proyecto.

Clases

5. Clase: Miembro

La representación de los integrantes de un proyecto se realiza en la clase Miembro, que incluye atributos como nombre, rol y nivel de permisos. Los métodos y atributos permiten gestionar la información de cada miembro y su relación con los proyectos. Esto asegura que cada miembro tenga un perfil único y que sus permisos sean gestionados adecuadamente. Esta clase se relaciona con Proyecto, ya que los miembros son asignados a proyectos, y con Tarea, ya que pueden ser asignados a tareas específicas.





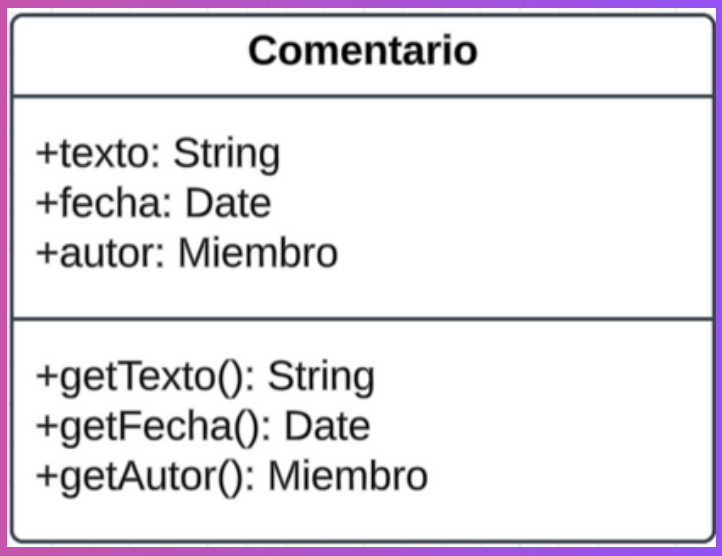
6. Clase: Tarea

Tarea se centra en definir las características de una tarea, incluyendo su nombre, descripción, fechas, prioridad, estado, miembros asignados y comentarios. Los métodos y atributos permiten gestionar la información de cada tarea y su relación con los miembros. Esto proporciona una estructura clara para cada tarea, facilitando su gestión. Se relaciona con Miembro y Proyecto, ya que las tareas pueden ser asignadas a miembros y están asociadas a proyectos específicos.

Clases

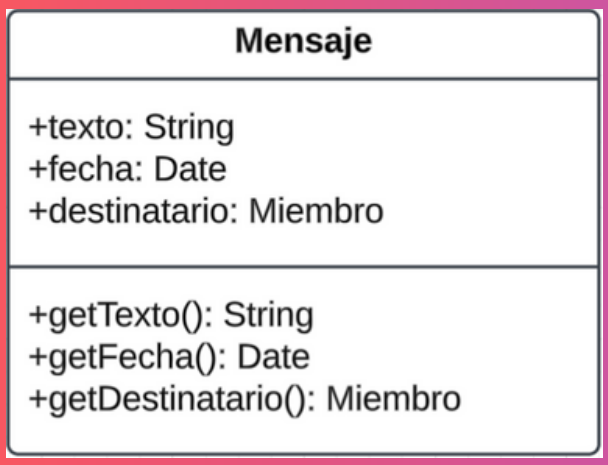
7. Clase: Comentario

Un comentario realizado por un miembro sobre una tarea se representa en la clase Comentario, que incluye el texto del comentario, la fecha y el autor. Los métodos y atributos permiten gestionar la retroalimentación y la comunicación sobre las tareas. Esto asegura que cada comentario esté asociado a un autor y a una fecha específica. La relación con Tarea es importante, ya que los comentarios son parte de las tareas y permiten un seguimiento más detallado del progreso y la comunicación.



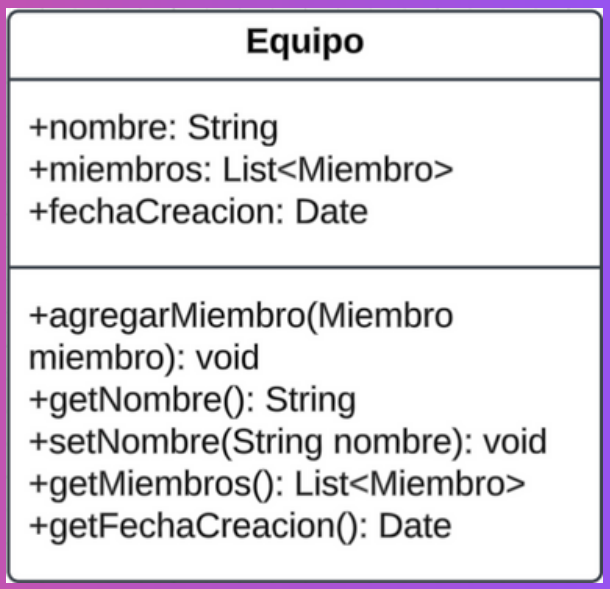
8. Clase: Mensaje

La gestión de la comunicación entre miembros del equipo se realiza en la clase Mensaje, que incluye el texto del mensaje, la fecha y el destinatario. Los métodos y atributos permiten enviar y recibir mensajes entre los usuarios. Esto asegura que cada mensaje esté asociado a un destinatario y a una fecha específica. La relación con Usuario es fundamental, ya que los mensajes son enviados y recibidos por los usuarios del sistema.



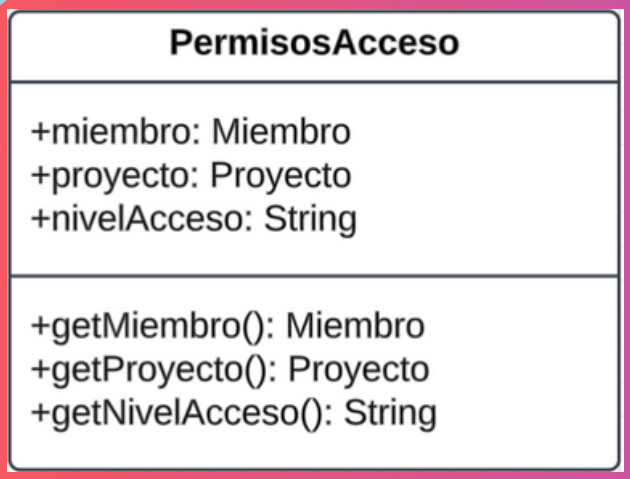
9. Clase: Equipo

En Equipo, se definen las características de un equipo, incluyendo su nombre, miembros y fecha de creación. Los métodos y atributos permiten gestionar la información de cada equipo y su composición. Esto organiza a los miembros en equipos para facilitar la gestión de proyectos. La relación con Miembro es clave, ya que los equipos están compuestos por miembros, y con GestorProyectos, que puede gestionar equipos dentro del sistema.



Clases

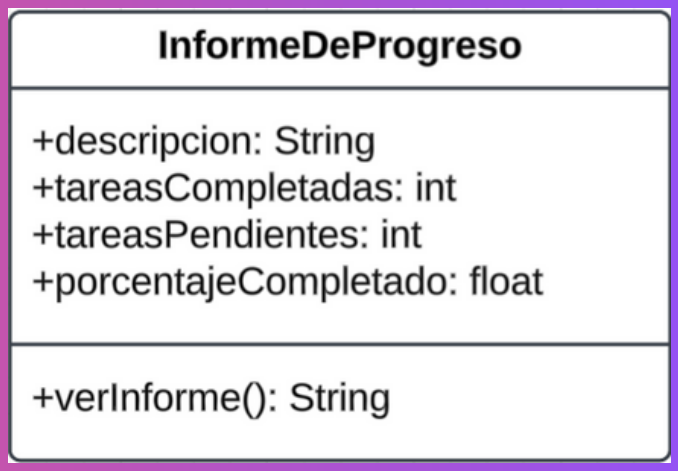
10. Clase: PermisosAcceso



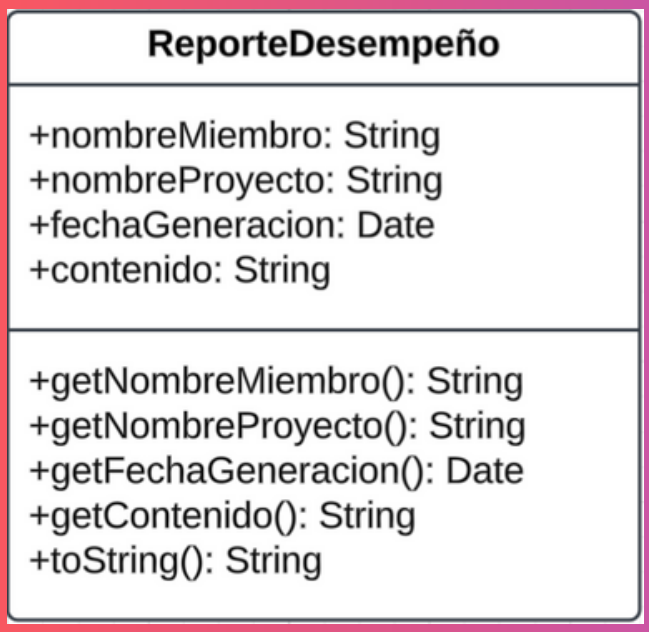
La gestión de los permisos de acceso de un miembro a un proyecto se realiza en la clase PermisosAcceso, que incluye el miembro, el proyecto y el nivel de acceso. Los métodos y atributos permiten definir y gestionar los permisos que tiene cada miembro sobre los proyectos. Esto asegura que se controle el acceso a la información y las funcionalidades del sistema. La relación con Miembro y Proyecto es esencial, ya que los permisos están asociados a un miembro específico y a un proyecto determinado.

11. Clase: InformeDeProgreso

Un informe que detalla el progreso de un proyecto se representa en la clase InformeDeProgreso, que incluye una descripción, el número de tareas completadas, tareas pendientes y el porcentaje completado. Los métodos y atributos permiten gestionar y visualizar el avance de un proyecto de manera clara. Esto proporciona un seguimiento del progreso de los proyectos. La relación con Proyecto es importante, ya que los informes de progreso están asociados a proyectos específicos.



12. Clase: ReporteDesempeño



Finalmente, en ReporteDesempeño, se define un reporte que evalúa el desempeño de un miembro en un proyecto, incluyendo el nombre del miembro, el nombre del proyecto, la fecha de generación y el contenido del reporte. Los métodos y atributos permiten gestionar y visualizar el desempeño de los miembros en los proyectos. Esto refleja el desempeño de los miembros en los proyectos. La relación con Proyecto es clave, ya que los reportes de desempeño están vinculados a proyectos, y con Miembro, ya que reflejan el desempeño de los miembros en esos proyectos.

Explicación de clases

Para el desarrollo del Sistema de Gestión de Proyectos, adoptamos un enfoque integral que nos permitió diseñar y estructurar diversas clases, cada una con un propósito específico y una interrelación clara. A continuación, describimos cómo cada clase contribuye al funcionamiento del sistema y cómo se relaciona con las demás.

Esta clase, SistemaGestionProyectos, fue diseñada con el objetivo de ser la interfaz principal del sistema, facilitando la interacción del usuario con todas las funcionalidades necesarias para gestionar proyectos. Esto nos llevó a implementar métodos que permiten crear proyectos, asignar miembros y gestionar tareas y equipos. Al hacerlo, buscamos que los usuarios puedan realizar diversas acciones de manera intuitiva. La conexión con GestorProyectos es esencial, ya que esta clase se encarga de la lógica de negocio y el almacenamiento de datos, mientras que Usuario representa a los individuos que utilizan el sistema, creando un flujo de trabajo eficiente.

GestorProyectos se desarrolló para centralizar y organizar toda la información relacionada con proyectos, miembros, tareas, equipos y mensajes. Esto nos llevó a establecer una estructura que permite crear, editar y eliminar proyectos, así como asignar tareas y miembros. Al hacerlo, garantizamos que la información se mantenga organizada y accesible. Esta clase se relaciona con Proyecto, Miembro, Tarea, Equipo y Mensaje, ya que contiene listas de estos objetos y facilita la interacción entre ellos, asegurando que todos los componentes del sistema funcionen de manera cohesiva.

Con la clase Usuario, definimos las características que identifican a los usuarios del sistema, como su nombre, contraseña y nivel de permisos. Esto nos llevó a implementar métodos que gestionan la información del usuario y su interacción con el sistema, incluyendo la recepción de mensajes y la obtención de tareas, equipos y proyectos asignados. La relación con GestorProyectos es clave, ya que esta clase utiliza la información del usuario para asignar tareas y proyectos, y también se vincula con Mensaje para facilitar la comunicación, creando un entorno colaborativo.

En Proyecto, se gestionan las características de un proyecto, como su nombre, descripción, fechas, estado y prioridad. Esto nos llevó a implementar métodos que permiten asignar tareas y miembros a un proyecto, proporcionando una estructura clara y organizada. La relación con Miembro y Tarea es fundamental, ya que los miembros son asignados a proyectos y las tareas están asociadas a estos, lo que permite un seguimiento efectivo del progreso y la distribución de responsabilidades.

Explicación de clases

La clase Miembro se centra en representar a los integrantes del equipo, definiendo su rol y nivel de permisos. Esto nos llevó a crear métodos que gestionan la información de cada miembro y su relación con los proyectos. Al hacerlo, aseguramos que cada miembro tenga un perfil único y que sus permisos sean gestionados adecuadamente. La conexión con Proyecto es esencial, ya que los miembros son asignados a proyectos específicos, y con Tarea, ya que pueden ser asignados a tareas concretas, facilitando así la colaboración.

Tarea se diseñó para definir las características de una tarea, incluyendo su nombre, descripción, fechas, prioridad, estado y miembros asignados. Esto nos llevó a implementar métodos que permiten gestionar la información de cada tarea y su relación con los miembros. La relación con Miembro y Proyecto es crucial, ya que las tareas pueden ser asignadas a miembros y están asociadas a proyectos específicos, lo que proporciona una estructura clara para la gestión de actividades.

Los comentarios realizados por un miembro sobre una tarea se representan en la clase Comentario, que incluye el texto del comentario, la fecha y el autor. Esto nos llevó a crear métodos que gestionan la retroalimentación y la comunicación sobre las tareas. La relación con Tarea es importante, ya que los comentarios son parte de las tareas y permiten un seguimiento más detallado del progreso y la comunicación, enriqueciendo la colaboración entre los miembros del equipo.

La gestión de la comunicación entre miembros del equipo se realiza en la clase Mensaje, que incluye el texto del mensaje, la fecha y el destinatario. Esto nos llevó a implementar métodos que permiten enviar y recibir mensajes entre los usuarios. La relación con Usuario es fundamental, ya que los mensajes son enviados y recibidos por los usuarios del sistema, asegurando que todos estén informados sobre los avances y cambios en los proyectos.

En Equipo, se definen las características de un equipo, incluyendo su nombre, miembros y fecha de creación. Esto nos llevó a crear métodos que gestionan la información de cada equipo y su composición. La relación con Miembro es clave, ya que los equipos están compuestos por miembros, y con GestorProyectos, que puede gestionar equipos dentro del sistema, facilitando así la organización del trabajo.

Explicación de clases

La gestión de los permisos de acceso de un miembro a un proyecto se realiza en la clase PermisosAcceso, que incluye el miembro, el proyecto y el nivel de acceso. Esto nos llevó a implementar métodos que permiten definir y gestionar los permisos que tiene cada miembro sobre los proyectos. La relación con Miembro y Proyecto es esencial, ya que los permisos están asociados a un miembro específico y a un proyecto determinado, asegurando que se controle el acceso a la información y las funcionalidades del sistema.

Un informe que detalla el progreso de un proyecto se representa en la clase InformeDeProgreso, que incluye una descripción, el número de tareas completadas, tareas pendientes y el porcentaje completado. Esto nos llevó a crear métodos que permiten gestionar y visualizar el avance de un proyecto de manera clara. La relación con Proyecto es importante, ya que los informes de progreso están asociados a proyectos específicos, proporcionando un seguimiento del avance.

Finalmente, en ReporteDesempeño, se define un reporte que evalúa el desempeño de un miembro en un proyecto, incluyendo el nombre del miembro, el nombre del proyecto, la fecha de generación y el contenido del reporte. Esto nos llevó a implementar métodos que permiten gestionar y visualizar el desempeño de los miembros en los proyectos. La relación con Proyecto es clave, ya que los reportes de desempeño están vinculados a proyectos, y con Miembro, ya que reflejan el desempeño de los miembros en esos proyectos.

En conjunto, todas estas clases interactúan para crear un sistema de gestión de proyectos cohesivo y eficiente. Cada clase aporta funcionalidades específicas que, al integrarse, permiten una planificación, ejecución y seguimiento de proyectos de manera colaborativa y organizada, asegurando que todos los miembros del equipo estén alineados y trabajando hacia los mismos objetivos.

Relaciones entre las clases y los objetos

1. Asociaciones (Relación estándar con multiplicidad):

- SistemaGestionProyectos (1) ——— (0..1) GestorProyectos
- GestorProyectos (1) ——— (0..*) Proyecto
- GestorProyectos (1) ——— (0..*) Miembro
- GestorProyectos (1) ——— (0..*) Equipo
- GestorProyectos (1) ——— (0..*) Tarea
- GestorProyectos (1) ——— (0..*) Mensaje
- Usuario (1) ——— (0..*) Mensaje
- Usuario (1) ——— (0..*) Tarea
- Usuario (1) ——— (0..*) Proyecto
- Usuario (1) ——— (0..*) Equipo
- Proyecto (1) ——— (0..*) Tarea
- Proyecto (1) ——— (0..*) Miembro
- Tarea (1) ——— (0..*) Comentario
- Tarea (1) ——— (0..*) Miembro
- Equipo (1) ——— (0..*) Miembro
- PermisosAcceso (1) ——— (1) Miembro
- PermisosAcceso (1) ——— (1) Proyecto

2. Agregación (Relación débil, objetos pueden existir independientemente):

- GestorProyectos ———◇ (1..*) Proyecto
- GestorProyectos ———◇ (1..*) Miembro
- GestorProyectos ———◇ (1..*) Equipo
- GestorProyectos ———◇ (1..*) Tarea
- Proyecto ———◇ (1..*) Miembro
- Proyecto ———◇ (1..*) Tarea
- Equipo ———◇ (1..*) Miembro
- Usuario ———◇ (0..*) Mensaje

3. Composición (Relación fuerte, objetos dependen de su contenedor):

- SistemaGestionProyectos ———◆ (1..*) GestorProyectos
- GestorProyectos ———◆ (1..*) Tarea
- GestorProyectos ———◆ (1..*) Mensaje
- Proyecto ———◆ (1..*) Tarea
- Tarea ———◆ (0..*) Comentario
- Mensaje ———◆ (1) Miembro
- Comentario ———◆ (1) Miembro
- **ReporteDesempeño ———◆ (1) Proyecto**

4. Herencia (Generalización / Especialización):

- PermisosAcceso ———△ (1) Miembro
- PermisosAcceso ———△ (1) Proyecto
- InformeDeProgreso ———△ (1) ReporteDesempeño

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. Por ejemplo, un GestorProyectos puede manejar múltiples Proyecto, Miembro, Equipo, Tarea y Mensaje, lo que refleja la flexibilidad y la naturaleza dinámica del sistema. En el caso de Usuario, este puede estar asociado a múltiples Mensaje, Tarea, Proyecto y Equipo, lo que permite una gestión integral de las interacciones del usuario dentro del sistema.

Además, un Proyecto puede tener múltiples Tarea y Miembro asignados, pero no todos los proyectos necesitan tener tareas o miembros en todo momento, lo que se refleja en la multiplicidad (0..*). Esto proporciona la flexibilidad necesaria para la gestión de proyectos en diferentes estados.

Para la relación entre Tarea y Comentario, definimos que las tareas pueden contener múltiples comentarios, pero estos no son obligatorios, lo que se representa con una multiplicidad (0..*). Esto permite que las tareas se gestionen de manera eficiente, sin la necesidad de que cada tarea tenga comentarios asociados.

Por otro lado, utilizamos la agregación para representar relaciones más débiles, donde los objetos pueden existir de manera independiente pero están agrupados de alguna forma. Por ejemplo, un GestorProyectos puede contener múltiples Proyecto, Miembro, Equipo y Tarea, pero estos pueden existir sin el gestor. Esto refleja que los miembros, proyectos y tareas son entidades independientes, aunque estén organizadas bajo un gestor.

También modelamos una relación recursiva en AsignacionMiembros, lo que nos permite representar que un mismo miembro puede estar asignado a varios proyectos y tareas de manera independiente, reflejando la complejidad de las asignaciones en un entorno de trabajo colaborativo.

Para las relaciones de composición, las utilizamos en aquellos casos en los que los objetos no pueden existir sin su entidad contenedora. Por ejemplo, en el caso de Proyecto con Tarea, InformeDeProgreso y PermisosAcceso, estos elementos dependen completamente de la existencia del 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. Los comentarios no tienen sentido fuera de una tarea, por lo que si esta se elimina, los comentarios también desaparecerán. Esta relación nos permitió modelar una dependencia total entre las tareas y los elementos que contienen.

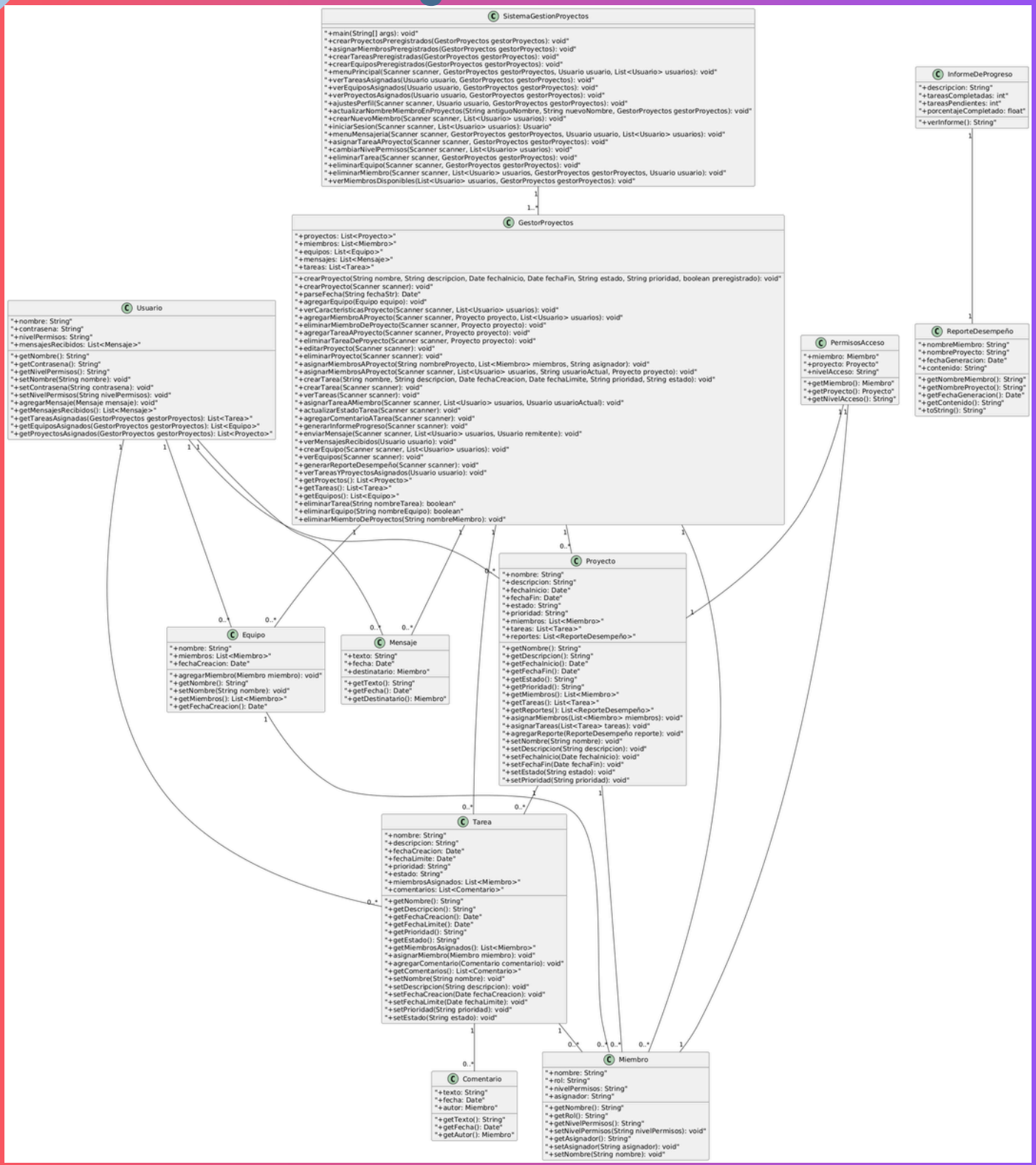
Finalmente, utilizamos la herencia para modelar clases que comparten atributos o comportamientos comunes, pero que tienen particularidades que las hacen únicas en su implementación. Por ejemplo, InformeDeProgreso y ReporteDesempeño comparten características relacionadas con el avance del proyecto, pero 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 se aplica entre PermisosAcceso y las entidades relacionadas, ya que los permisos solo tienen sentido dentro del contexto de un proyecto 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.

Diagrama

1.-Diagrama de clases



Explicación del código

Para nuestro código del proyecto, hicimos un sistema de gestión de proyectos que permite a los usuarios interactuar con diferentes funcionalidades relacionadas con la creación y gestión de proyectos, tareas y equipos. Lo que nos llevó a desarrollar este sistema fue la necesidad de facilitar la colaboración y organización en un entorno de trabajo. El flujo del programa comienza en la clase principal SistemaGestionProyectos, donde se inicializa un Scanner para la entrada de datos y se crea una instancia de GestorProyectos, que es el componente central que maneja todos los proyectos, tareas, equipos y usuarios. Al iniciar el programa, se crean algunos usuarios predefinidos y se añaden a una lista, junto con proyectos, tareas y equipos predefinidos. Esto permite que el sistema tenga datos iniciales con los que trabajar. Luego, se presenta un menú de inicio donde los usuarios pueden elegir entre iniciar sesión o terminar el programa. Si eligen iniciar sesión, se llama al método iniciarSesion, que solicita el nombre de usuario y la contraseña. Si las credenciales son correctas, el usuario es autenticado y se le da la bienvenida, mostrando su nivel de permisos.

Una vez que el usuario ha iniciado sesión, se accede al menuPrincipal, donde se presentan diferentes opciones basadas en el nivel de permisos del usuario. Si el usuario es un administrador, tiene acceso a una gama más amplia de funcionalidades, mientras que los usuarios regulares tienen acceso a funciones más limitadas. Al ejecutar el código y elegir la opción de iniciar sesión, por ejemplo, se solicita el nombre de usuario y la contraseña, y al ingresar las credenciales correctas, se muestra un mensaje de bienvenida junto con el nivel de permisos del usuario. En el menú principal, las opciones disponibles para un administrador son las siguientes: Crear proyecto permite al administrador crear un nuevo proyecto. Se solicita al usuario que ingrese el nombre, la descripción, las fechas de inicio y fin, el estado y la prioridad. Al elegir esta opción y proporcionar los datos, se crea un nuevo objeto Proyecto, que se añade a la lista de proyectos en el GestorProyectos. Editar proyecto permite al administrador editar un proyecto existente. Se muestra una lista de proyectos disponibles y se solicita al usuario que seleccione uno. Luego, se presentan los atributos del proyecto que se pueden editar, como el nombre, la descripción, las fechas, el estado y la prioridad. Al elegir un atributo y proporcionar un nuevo valor, se actualiza el proyecto. Eliminar proyecto permite al administrador eliminar un proyecto existente. Se muestra una lista de proyectos y se solicita al usuario que seleccione el número del proyecto que desea eliminar. Si se confirma la eliminación, el proyecto se elimina de la lista. Asignar miembros a un proyecto permite al administrador asignar miembros a un proyecto. Se muestra una lista de proyectos y se solicita al usuario que seleccione uno. Luego, se presentan los miembros disponibles y se solicita al administrador que seleccione a los miembros que desea asignar, junto con su rol. Crear tarea permite al administrador crear una nueva tarea. Se solicita al usuario que ingrese el nombre, la descripción, la fecha límite, la prioridad y el estado de la tarea.

Estos datos se utilizan para crear un nuevo objeto Tarea, que se añade a la lista de tareas en el GestorProyectos. Asignar tarea a miembro permite al administrador asignar una tarea a un miembro específico. Se muestra una lista de tareas disponibles y se solicita al usuario que seleccione una. Luego, se presentan los miembros disponibles y se solicita al administrador que seleccione a un miembro para asignarle la tarea. Actualizar estado de tarea permite al administrador actualizar el estado de una tarea existente. Se muestra una lista de tareas y se solicita al usuario que seleccione una. Luego, se solicita al usuario que ingrese el nuevo estado de la tarea. Agregar comentario a tarea permite a los usuarios agregar comentarios a una tarea. Se solicita el nombre de la tarea y el comentario, así como el nombre del autor. Si la tarea existe, el comentario se añade a la lista de comentarios de la tarea. Generar informe de progreso permite al administrador generar un informe sobre el progreso de un proyecto

Se solicita el nombre del proyecto y se calcula el número de tareas completadas y pendientes, así como el porcentaje de progreso. Mensajería permite a los usuarios enviar y recibir mensajes.

Explicación del código

Al seleccionar esta opción, se presenta un submenú donde se puede elegir enviar un mensaje, ver mensajes recibidos o volver al menú principal. Crear equipo permite al administrador crear un nuevo equipo. Se solicita el nombre del equipo y se permite agregar miembros al equipo seleccionando de la lista de usuarios disponibles. Generar reporte de desempeño permite al administrador generar un reporte de desempeño para un miembro en un proyecto específico. Se solicita el nombre del miembro y el proyecto, y se genera un reporte basado en el progreso del proyecto. Ver características de un proyecto permite a los usuarios ver los detalles de un proyecto específico. Se muestra una lista de proyectos y se solicita al usuario que seleccione uno para ver su nombre, descripción, fechas, estado, prioridad, miembros asignados y tareas. Ver tareas permite a los usuarios ver una lista de todas las tareas disponibles en el sistema. Ver equipos creados permite a los usuarios ver una lista de todos los equipos creados en el sistema. Asignar tarea a proyecto permite al administrador asignar una tarea a un proyecto específico. Se muestra una lista de tareas y proyectos, y se solicita al usuario que seleccione ambos para realizar la asignación. Cambiar nivel de permisos de un miembro permite al administrador cambiar el nivel de permisos de un miembro existente. Se muestra una lista de miembros y se solicita al administrador que seleccione uno y el nuevo nivel de permisos. Ajustes de perfil permite a los usuarios cambiar su nombre de usuario o contraseña. Se presenta un submenú con estas opciones. Eliminar tarea permite al administrador eliminar una tarea existente. Se muestra una lista de tareas y se solicita al usuario que seleccione una para eliminar. Eliminar equipo permite al administrador eliminar un equipo existente. Se muestra una lista de equipos y se solicita al usuario que seleccione uno para eliminar. Crear nuevo miembro permite al administrador crear un nuevo miembro en el sistema. Se solicita el nombre de usuario, contraseña y nivel de permisos. Eliminar miembro permite al administrador eliminar un miembro existente. Se muestra una lista de miembros y se solicita al usuario que seleccione uno para eliminar. Ver miembros disponibles permite a los usuarios ver una lista de todos los miembros disponibles en el sistema. Ver tareas asignadas permite a los usuarios ver las tareas que tienen asignadas. Ver equipos asignados permite a los usuarios ver los equipos a los que están asignados. Ver proyectos asignados permite a los usuarios ver los proyectos en los que están asignados. Salir permite a los usuarios salir del sistema y volver al menú de inicio.

Al elegir una opción en el menú, el sistema responde de acuerdo a los permisos del usuario. Por ejemplo, si un administrador selecciona la opción para eliminar un proyecto, se le permite hacerlo, mientras que un usuario regular no tendrá esa opción disponible. Esto asegura que solo los usuarios con los permisos adecuados puedan realizar acciones críticas en el sistema. Cuando un usuario con nivel de permisos "Usuario" inicia sesión, las opciones disponibles son diferentes. En lugar de las opciones de gestión de proyectos, el menú para un usuario regular incluye: Crear tarea permite al usuario crear una nueva tarea, ingresando detalles como el nombre, la descripción, la fecha límite, la prioridad y el estado. Asignar tarea a miembro permite al usuario asignar una tarea a un miembro, pero solo si tiene acceso a las tareas disponibles. Actualizar estado de tarea permite al usuario actualizar el estado de una tarea que le ha sido asignada. Agregar comentario a tarea permite al usuario agregar un comentario a una tarea específica. Ver características de un proyecto permite al usuario ver los detalles de un proyecto específico. Mensajería permite al usuario enviar y recibir mensajes. Ver tareas permite al usuario ver todas las tareas disponibles. Generar informe de progreso permite al usuario generar un informe sobre el progreso de un proyecto, pero con acceso limitado. Ajustes de perfil permite al usuario cambiar su nombre de usuario o contraseña. Ver tareas asignadas permite al usuario ver las tareas que tiene asignadas. Ver equipos asignados permite al usuario ver los equipos a los que está asignado. Ver proyectos asignados permite al usuario ver los proyectos en los que está asignado. Salir permite al usuario salir del sistema.

Explicación del código

Al entrar a un perfil de nivel miembro, las opciones cambian porque los usuarios regulares no tienen la capacidad de crear o eliminar proyectos, ni de gestionar miembros, lo que limita su interacción a tareas y proyectos en los que están directamente involucrados. Esto asegura que la gestión del sistema se mantenga en manos de los administradores, mientras que los usuarios pueden concentrarse en completar tareas asignadas y colaborar en proyectos. Este diseño modular y estructurado permite que el sistema sea escalable y fácil de mantener, facilitando la adición de nuevas funcionalidades en el futuro. En resumen, el sistema de gestión de proyectos está diseñado para ser intuitivo y eficiente, permitiendo a los usuarios gestionar sus tareas y colaborar con otros de manera efectiva.

Conclusión de Viktor

Como conclusión tras realizar este proyecto, puedo decir que he adquirido una comprensión más profunda de la programación orientada a objetos y su aplicación en el desarrollo de un sistema de gestión de proyectos. Lo que me llevó a aprender sobre la importancia de la planificación y la organización en el desarrollo de software. Para esto, tuve que llevar a cabo un diseño cuidadoso, identificando los requisitos funcionales y no funcionales del sistema, lo que me llevó a definir claramente las clases y sus relaciones.

Además, la creación de diagramas UML fue fundamental para visualizar la estructura del sistema y entender cómo interactúan los diferentes componentes. Implementar el sistema utilizando principios de programación orientada a objetos me permitió comprender mejor conceptos como la encapsulación, la herencia y el polimorfismo, al crear clases como Usuario, Proyecto, Tarea y Miembro, cada una con su propia responsabilidad y relaciones definidas.

A través de la simulación de un entorno de gestión de proyectos, entendí la importancia de facilitar la comunicación y la colaboración entre los miembros del equipo, lo que me llevó a implementar funcionalidades como la asignación de tareas, la gestión de equipos y la generación de informes de progreso, esenciales para el éxito de cualquier proyecto. Durante el desarrollo, enfrenté varios desafíos, desde la gestión de excepciones hasta la implementación de la lógica de negocio, lo que me llevó a abordar problemas de manera sistemática, buscando soluciones y optimizando el código para mejorar la eficiencia y la usabilidad del sistema.

Links a github

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