A4-220501095-AA2- Elaborar artefactos usando el paradigma de programación orientada a objetos

GA4-220501095-AA2-EV01 – Taller de conceptos y principios de programación orientada a objetos

GA4-220501095-AA2-EV02 - Informe de entregables para el proyecto de desarrollo de software

GA4-220501095-AA2-EV03 bases conceptuales acerca del lenguaje unificado de modelado (UML) y patrones de diseño

GA4-220501095-AA2-EV04 - Diagrama de clases del proyecto de software

GA4-220501095-AA2-EV05 - Desarrollar la arquitectura de software de acuerdo con el patrón de diseño seleccionado

GA4-220501095-AA2-EV06 - Taller arquitectura de software

Yeisson Mauricio Torres Patiño

Ficha: 2977422

Instructor

Jimmy Perea Gámez

Tecnólogo en análisis y desarrollo de software

Centro de la construcción regional valle, servicio nacional de aprendizaje

27 noviembre de 2024

Contenido

Introducción

En el presente documento se encontrarán todas las actividades propuestas de la guía GA4-220501095, en ellas se incluye las evidencias calificables requeridas por la guía de aprendizaje las cuales son:

1. Taller de conceptos y principios de programación orientada a objetos (POO) Programación orientada a objetos, el cual contiene el glosario de términos.
2. Informe de entregables para el proyecto de desarrollo de software, actividades que ya se habían hecho anteriormente.
3. Diagrama de clases del proyecto de software, actividad importante que se desarrolló hace poco.
4. Desarrollar la arquitectura de software de acuerdo con el patrón de diseño seleccionado, actividad de mucha importancia y requerida por la guía.

Objetivos

- ✓ Subir actividades anteriores requeridas por la guía de aprendizaje.
- ✓ Repasos sobre elaboración de diagramas del proyecto.
- ✓ Conocer y aprender la arquitectura del software.
- ✓ Aprender a construir la arquitectura del proyecto de software a desarrollar.

GA4-220501095-AA2-EV01 - Taller de conceptos y principios de programación orientada a objetos (POO) Programación orientada a objetos.

Es un método de programación en el cual los programas se organizan en colecciones cooperativas de objetos, cada uno de los cuales representa una instancia de alguna clase, y cuyas clases son, todas ellas, miembros de una jerarquía de clases unidas mediante relaciones de herencia.

Glosario

POO	Consiste en simplificar la complejidad del mundo real modelando solo los aspectos relevantes para un propósito particular, mientras se ocultan los detalles innecesarios.
Clases	Es un modelo o plantilla que define las características y comportamientos de un objeto.
Herencias	Es un mecanismo que permite que una clase herede atributos y comportamientos de otra clase. Es una de las características principales de la POO, junto con la encapsulación y el polimorfismo.
Objetos	Es una instancia de una clase que combina datos y funcionalidad en una sola entidad. Los objetos son la base del paradigma de la POO, y permiten simplificar programas complejos al dividirlos en unidades.
Métodos	Son funciones que permiten realizar acciones y manipular los atributos de una clase.
Eventos	Son el medio como interactúa una clase con otras o con el propio usuario, se encargan de avisar que algo ha ocurrido y de manejarlo de una forma o de otra.
Atributos	Son la información que describe las características de los objetos o instancias de una clase.
Abstracción	Es la capacidad de representar y manejar conceptos complejos de manera simplificada.
Encapsulamiento	Es esencial para construir sistemas de software seguros, flexibles y mantenibles.
Polimorfismo	Es la capacidad que tienen ciertos lenguajes para hacer que, al enviar el mismo mensaje desde distintos objetos, cada uno de esos objetos pueda responder a ese mensaje de forma distinta.

GA4-220501095-AA2-EV02 - Informe de entregables para el proyecto de desarrollo de software

Requisitos funcionales:

- Permitir al administrador del sistema crear todos los usuarios que necesite.
- Permitir la creación de máximo 2 usuarios administradores.
- Solicitar al administrador actualización de datos al cambio de año con el aumento del SMMLV.
- Permitir descargar informes de manera ágil al administrador.
- Permitir al administrador la actualización de las plantillas para la descarga de documentos.
- Envío rápido y de manera oportuna de las solicitudes realizadas por todos los colaboradores por mensaje de correo electrónico y WhatsApp.
- La página enviara un correo electrónico y WhatsApp al usuario confirmando que la solicitud ha sido realizada con éxito.
- Permitir al usuario solicitar todos los permisos que requiera.
- La página web debe enviar alertas al administrador sobre fechas especiales con un día de anticipación y el mismo día.
- El sistema deberá bloquear las solitudes ya realizadas por los usuarios, las cuales solo tienen permisos específicos (una vez al año, una vez por semestre, entre otros).

Requisitos no funcionales:

- La página web debe cargar la página web en menos de 3 segundos así estén 20 personas conectadas.
- Permitir al usuario actualizar la contraseña en su primer ingreso.

- Los colaboradores nunca podrán actualizar su información salarial. En este caso debe ser informado al administrador del sistema.
- Los usuarios solo tendrán acceso a su propia información.
- La página web debe estar disponible solo en el horario laboral.
- La página web debe ser funcional en Windows y IOS.
- La página web debe funcionar en Firefox, Chrome, internet explored y safari.
- La longitud de la clave de usuario debe ser mínimo de 7 caracteres y puede ser solo números, letras, en mayúsculas o minúscula.
- El sistema debe exigir un respaldo de seguridad cada 5 días.
- Solo el administrador puede crear usuarios nuevos.

Características de los usuarios.

La aplicación web va dirigida a todo tipo de público mayor de edad que labore en la empresa, en cualquier cargo desde gerencial hasta operativo y con el nivel operativo correspondiente al cargo.

Restricciones.

- Solo el usuario administrador tendrá permiso para crear usuarios nuevos.
- Solo se permitirá una sesión abierta por dispositivo y una sesión abierta como usuario.
- Si el usuario ingresa la contraseña dos veces errada se le notificara que el siguiente intento se bloqueara el usuario.
- Si se bloquea el usuario solo el administrador lo podrá desbloquear.
- Se realizará auditoria mensual por parte del administrador, para garantizar que el programa este cumpliendo con las restricciones inicialmente estipuladas.

|Registro de formato de ficha técnica	Responsable

CARACTERÍSTICAS DEL PRODUCTO

Nombre del producto: San Fernando Plaza contigo.

Línea del producto: Administración.

Versiones anteriores: 1.0

Versión actual: 1.0

Módulo: 1.0

DESCRIPCIÓN DEL PRODUCTO

Descripción general del producto: Software para la gestión de citas, permisos laboralesy descarga de documentos como cartas laborales, colillas de pago entre otros.

Objetivo: Mejorar los procesos y tiempos de respuesta en el área de gestión humana.

ARQUITECTURA

Descripción: HTML, CSS, JAVASCRIPT.

REQUERIMIENTOS DEL PRODUCTO

Requisitos del sistema(servidor)

Hardware: memoria RAM de 32gb o más, Almacenamiento SSD rápido, Soporte para tecnologías de virtualización (Intel VD-x o AMD-V) Servicio de internet, sistema de alimentación ininterrumpida. Sistema para copia de seguridad.

Software: Sistema Operativo Windows de 2000 en adelante, IOS, Linux con navegadores
Google Chrome, Mozilla, Firefox y otros reconocidos

Otros: Memoria ROM de 30Gb (Mínimo)

Requisitos del sistema(cliente)

Hardware: Computador con acceso a internet.

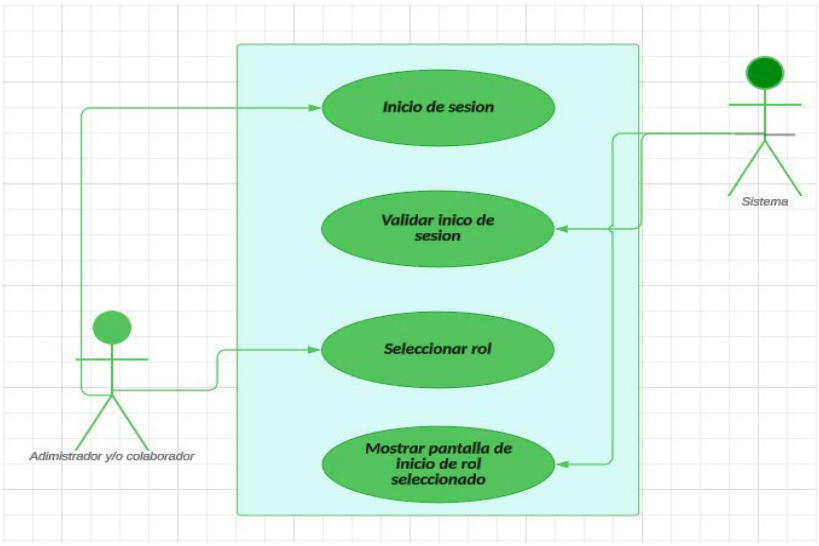
Software: Navegadores Web reconocidos: Google Chrome, Mozilla, Firefox y otros.

Otros: Cuenta de correo para enviar confirmación de la reservación.

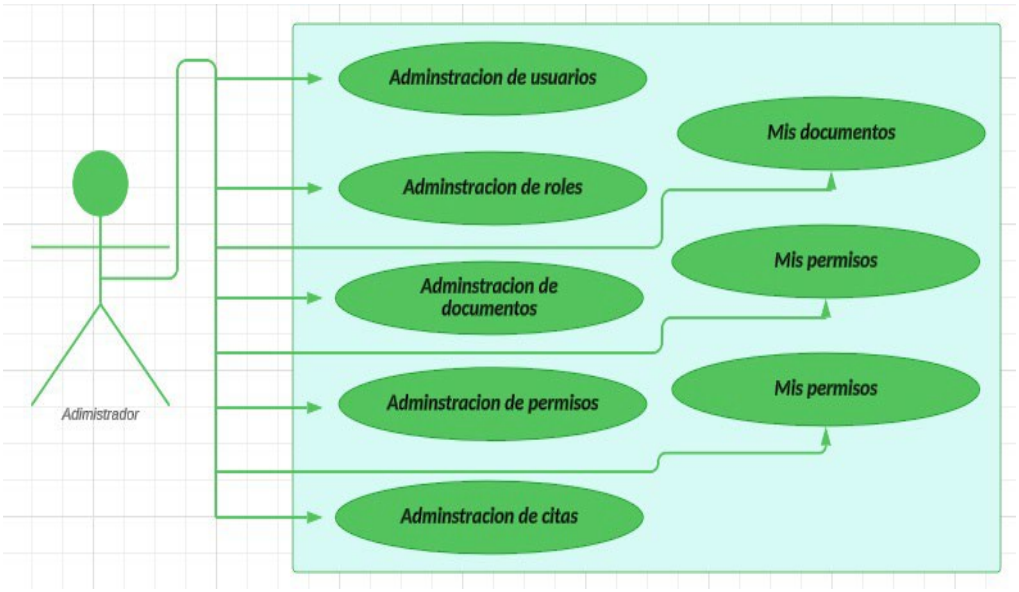
REQUERIMIENTOS

1. **Requerimientos funcionales generales:** Permitir al administrador del sistema crear todos los usuarios que necesite.
2. Permitir la creación de máximo 2 usuarios administradores.
3. Solicitar al administrador actualización de datos al cambio de año con el aumento del SMMLV.
4. Permitir descargar informes de manera ágil al administrador.
5. Permitir al administrador la actualización de las plantillas para la descarga de documentos.
6. Envío rápido y de manera oportuna de las solicitudes realizadas por todos los colaboradores por mensaje de correo electrónico y WhatsApp.
7. La página enviara un correo electrónico y WhatsApp al usuario confirmando que la solicitud ha sido realizada con éxito.
8. Permitir al usuario solicitar todos los permisos que requiera.
9. La página web debe enviar alertas al administrador sobre fechas especiales con un día de anticipación y el mismo día.
10. El sistema deberá bloquear las solitudes ya realizadas por los usuarios, las cuales solo tienen permisos específicos (una vez al año, una vez por semestre, entre otros).

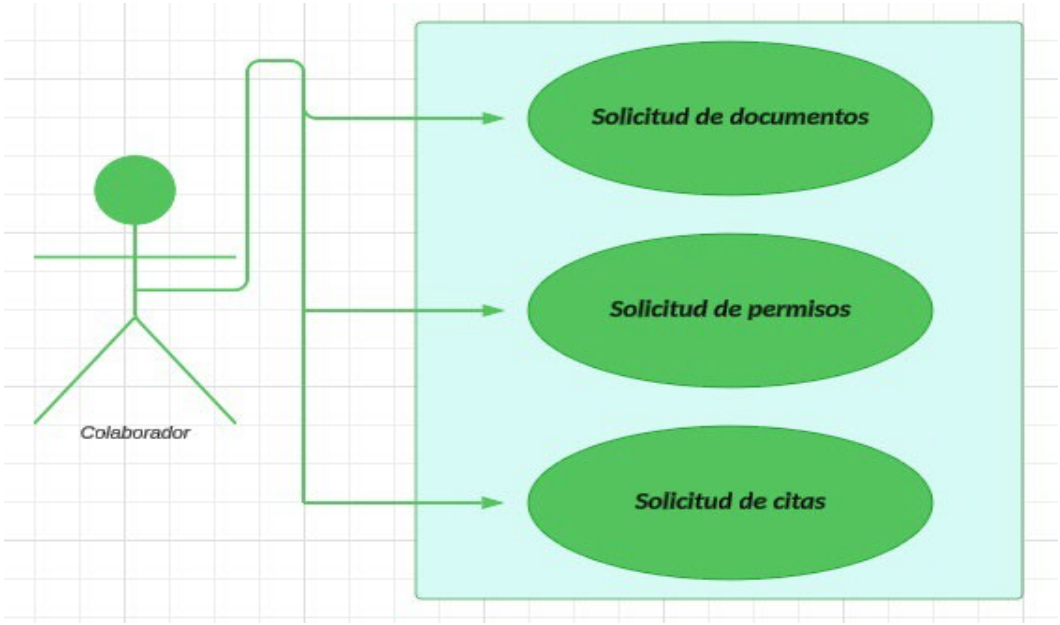
Clientes del producto: San Fernando Plaza.



Plantillas de caso de uso	
Nombre del caso de uso	Iniciar sesión
Autor	Yeison Torres
Fecha	26/09/2024
Descripción	Iniciar sesión como administrador y/o colaborador.
Actores	Colaboradores de la empresa.
Precondiciones	Para iniciar sesión el usuario debe tener aprobados los datos de ingreso.
Flujo normal	1. El actor ingresa usuario. 2. El actor ingresa contraseña. 3. El actor elige rol de ingreso. 4. El sistema verifica datos de ingreso. 5. El sistema abre página de inicio de rol seleccionado.
Flujo alternativo	• El sistema comprueba que los datos de ingreso sean los correctos. • El sistema le da acceso para recuperación de contraseña.
Postcondiciones	Se almacenan los datos de ingres y se abre página de inicio correctamente según el rol seleccionado.



Plantillas de caso de uso	
Nombre del caso de uso	Ingreso a página de administrador
Autor	Yeison Torres
Fecha	26/09/2024
Descripción	Entrada a la página de inicio como rol de administrador.
Actores	Especialista de gestión humana (administrador).
Precondiciones	Para iniciar sesión el usuario debe tener aprobados los datos de ingreso y seleccionar rol de ingreso.
Flujo normal	1. El actor inicia sección con el rol de administrador 2. El sistema valida datos de ingreso. 3. El actor elige la tarea a realizar 4. El sistema verifica opciones seleccionadas. 5. El sistema abre opciones seleccionadas. 6. El actor procedes a realizar actividades.
Flujo alternativo	• El sistema comprueba que los datos de ingreso sean los correctos. • El sistema le da acceso a la página del rol seleccionado.
Postcondiciones	Se almacenan los datos de ingres y se abre página de inicio correctamentesegún el rol seleccionado.



Plantillas de caso de uso	
Nombre del caso de uso	Ingreso a página de administrador
Autor	Yeison Torres
Fecha	26/09/2024
Descripción	Entrada a la página de inicio como rol de Colaborador.
Actores	Especialista financiera (colaborador).
Precondiciones	Para iniciar sesión el usuario debe tener aprobados los datos de ingreso yseleccionar rol de ingreso.
Flujo normal	7. El actor inicia sección con el rol de colaborador 8. El sistema valida datos de ingreso. 9. El actor elige la tarea a realizar 10. El sistema verifica opciones seleccionadas. 11. El sistema abre opciones seleccionadas. 12. El actor procedes a realizar actividades.
Flujo alternativo	• El sistema comprueba que los datos de ingreso sean los correctos. • El sistema le da acceso a la página del rol seleccionado.
Postcondiciones	Se almacenan los datos de ingres y se abre página de inicio correctamente según el rol seleccionado.

GA4-220501095-AA2-EV03 bases conceptuales acerca del lenguaje unificado de modelado (UML) y patrones de diseño

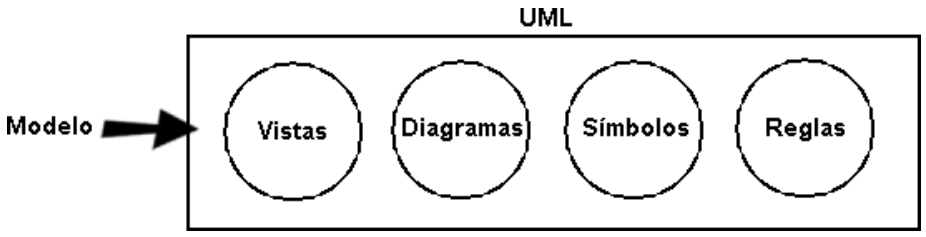
LENGUAJE UNIFICADO DE MODELADO (UML)

Como su nombre y sus siglas lo indican (Unified Modeling Language) es conocido como un lenguaje de modelado o gráfico visual, es diferente de los métodos de análisis y diseño más conocidos, pero más que un lenguaje es un estándar de normas y gráficos en los que se basa el cómo se deben representar los esquemas de desarrollo de software. Su función principal es visualizar los estados y las interaccionesentre objetos dentro de un sistema.

Características de una herramienta UML:

- ✓ Capacidad de diagramación, y los diferentes tipos de diagramas que soporta laherramienta.
- ✓ Sus esquemas de apoyo de diseño, documentación, construcción e implantaciónde sistema.
- ✓ Flexibilidad para admitir cambios no previstos durante el diseño o el rediseño.

Un modelo es expresado en un lenguaje de modelado. Este consiste de vistas, diagramas, elementos de modelo y un conjunto de mecanismos generales o reglas queindican cómo utilizar los elementos, las reglas son sintácticas, semánticas y pragmáticas.



- ✓ **Vistas:** Muestran varios aspectos del sistema modelado. No es una gráfica sino una abstracción que consiste en un número de diagramas que muestran un panorama completo del sistema. Las vistas también ligan el lenguaje de modeladoa los métodos o procesos elegidos para el desarrollo. Las diferentes vistas de UML son:

    - **Vista Use-Case:** Muestra la funcionalidad del sistema según como la percibenlos actores externos.
    - **Vista Lógica:** Muestra cómo se diseña la funcionalidad dentro del sistema, encontexto de la estructura estática y conducta dinámica del sistema.
    - **Vista de Componentes:** Muestra la organización de los componentes decódigo.
    - **Vista Concurrente:** Muestra la concurrencia en el sistema, dirigiendo los problemas con la comunicación y sincronización que están presentes en unsistema concurrente.
    - **Vista de Distribución:** Muestra la distribución del sistema en la arquitecturafísica con computadoras y dispositivos nodos.

- ✓ **Diagramas:** Gráficas que describen el contenido de una vista. UML tiene nueve tipos de diagramas que proveen en conjunto todas las vistas de un sistema comodiagramas de caso de uso, de clases, de objetos, de estados, de secuencia, de colaboración, de actividad, de componentes y de distribución.
- ✓ **Símbolos o Elementos de modelo:** Representan conceptos comunes orientadosa objetos, tales como clases, objetos y mensajes, y las relaciones entre estos como asociación, dependencia y generalización. Un elemento de modelo es utilizado en varios diagramas diferentes, pero siempre tiene el mismo significado ysimbología.
- ✓ **Reglas o Mecanismos generales:** Proveen comentarios extras, información osemántica acerca del elemento de modelo; además proveen mecanismos de extensión para adaptar o extender UML a un método o proceso específico, organización o usuario.

Tipos de diagramas en UML

En este estándar de modelado se pueden crear bastantes diagramas, entre esosse definirán algunos:

- ✓ Diagramas de casos de uso: representan a los actores y casos de uso queintervienen en un desarrollo de software.
- ✓ Diagramas de clases: para UML una clase es una entidad, no una clase software, así que puede ser un diagrama del dominio o representación de conceptos que intervienen en un problema, o también un diagrama de clasessoftware. El sentido de un diagrama UML se lo da quien lo construye.
- ✓ Diagramas de secuencia: Se usan para representar objetos software y elintercambio de mensajes entre ellos.
- ✓ Diagramas de colaboración: Se usan para representar objetos o clases y la forma en que se transmiten mensajes o se colaboran entre ellos para cumplirun objetivo.
- ✓ Diagramas de estados: Se usan para representar cómo un sistema cambiade estado a medida que se producen determinados eventos.
- ✓ Diagrama de componentes: Se crean cuando un sistema es complejo yconsta de demasiadas clases.
- ✓ Diagrama de despliegue: Esto proporciona la representación real delhardware del sistema y sigue un enfoque más real sobre cómo se desplegaría el sistema.
- ✓ Diagrama de objetos: Es como un desglose de un diagrama de clases, sebasan principalmente en entidades del mundo real.
- ✓ Diagrama de paquetes: Esto proporciona una representación de un sistemacon sus módulos y subsistemas de más alto nivel. Aquí se incluyen varias estructuras de diagrama de despliegue.
- ✓ Diagrama de perfil: Se introdujo en UML 2. Aunque estos tipos de diagramasUML no son tan populares, pueden utilizarse para representar la meta estructura del sistema.

- ✓ Diagrama de la estructura compuesta: Se usan para representar la estructurainterna de una clase. Representa cómo se relacionan las diferentes entidades de una clase entre sí y cómo se asocia la propia clase con la entidad o sistema exterior.
- ✓ Diagramas de comportamiento: Cubren las especificaciones restantes bajo UML. A diferencia de los diagramas estructurales, no son estáticos, sino querepresentan procesos y situaciones dinámicas.
- ✓ Diagramas de interacción: Son un subtipo de los diagramas de comportamiento, representan situaciones dinámicas. En particular, sonadecuados para modelar el comportamiento en el que los elementos intercambian información.

Los diagramas UML se utilizan para representar los siguientes componentes del sistema:

- Objetos individuales - elementos básicos.
- Clases - combina elementos con las mismas propiedades.
- Relaciones entre objetos - jerarquía y comportamiento o comunicación entreobjetos.
- Actividad - combinación compleja de acciones/módulos de comportamiento.
- Interacciones entre objetos e interfaces.

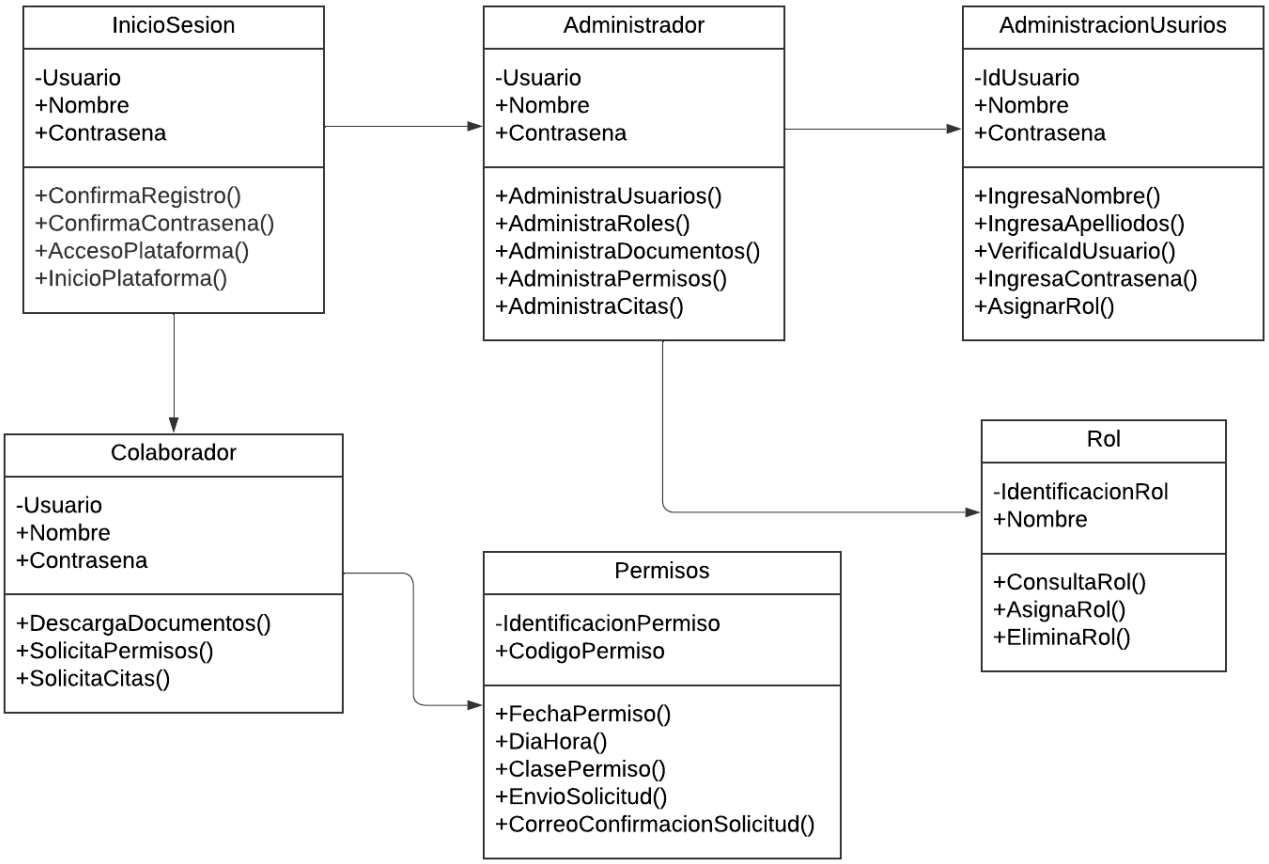
Categoría	Tipo de diagrama	Aplicación
Estructura	Diagrama de clases	**Visualizar clases**
	Diagrama de objetos	**Estado del sistema en un momentodado**
	Diagrama decomponentes	**Estructurar componentes y mostrar relaciones**
	Diagrama de estructura compositiva	**Divide los componentes o clases en sus componentes y aclara sus relaciones.**
	Diagrama de paquete	**Agrupa las clases en paquetes, muestra la jerarquía y la estructura de los paquetes.**
	Diagrama de distribución	**Distribución de componentes a los nodos informáticos**
	Gráfica de perfil	**Ilustra contextos de uso a través de estereotipos, condiciones límite, etc.**
Comportamiento	Diagrama de casos deuso	**Representa varias aplicaciones**
	Diagrama de actividades	**Describe el comportamiento de diferentes procesos (paralelos) en un sistema.**
	Diagrama de máquina deestados	**Documenta cómo un objeto es movidode un estado a otro por un evento.**
Comportamiento :interacción	Diagrama secuencial	**Secuencia temporal de lasinteracciones entre objetos**
	Diagrama decomunicación	**Distribución de roles de los objetos dentro de una interacción**
	Diagrama de tiempos	**Limitación de tiempo para los acontecimientos que conducen a un cambio de estado**
	Diagrama de interacción	**Secuencias y actividades interactivas**

Glosario:

- Objetos: Representan una entidad y el componente básico.
- Clase: Plano de un objeto.
- Abstracción: Comportamiento de una entidad del mundo real
- Encapsulación: Mecanismo para enlazar los datos y ocultarlos del mundoexterior.
- Herencia: Mecanismo para crear nuevas clases a partir de una existente.
- Polimorfismo: Define el mecanismo para salidas en diferentes formas.
- Actividad: llevar a cabo el comportamiento en un diagrama de máquina deestados.
- Actor: un rol que asume un usuario cuando invoca un caso de uso.
- Agregación: Tipo de asociación que se utiliza para representar una relación másfuerte entre dos clases.
- Asociación: Una relación con 2 o más extremos, donde cada extremo está enuna clase.
- Atributo: un dato significativo propiedad de una clase, que a menudo contienevalores que describen cada instancia de la clase.
- Bloque: el bloque es un lugar donde todos los agregados se recolectan en unsolo lugar.
- Diagrama de clases: Tipo de diagrama de estructura estática que describe la estructura de un sistema mostrando las clases del sistema, sus atributos y lasrelaciones entre las clases.
- Componente: Representa un módulo de con una interfaz bien definida.

- Restricción: Condición booleana del lenguaje de restricción de objetos que puedeno ser falsa si una clase se considera válida.
- Dependencia: existe una dependencia entre dos elementos definidos si uncambio en la definición de uno da como resultado un cambio en el otro.
- Evento: cuando ocurre en un objeto, puede causar una transición en un diagrama de máquina de estado.
- Herencia: donde una nueva clase más específica deriva parte de su definición deuna clase más general existente.
- Paquete: colección o agrupación de clases relacionadas o de clases con funcionalidad relacionada.
- Caso de uso: un caso de uso se puede definir como una secuencia de acciones,incluidas las variaciones, que el sistema puede ejecutar y que produce un resultado observable que tiene algún valor para un actor que interactúa con el sistema.

GA4-220501095-AA2-EV04 - Diagrama de clases del proyecto de software



https://lucid.app/lucidchart/d804d876-636f-4578-8eef-efafd18ec055/edit?viewport_loc=60%2C-60%2C2038%2C1011%2CHWEp-vi-RSFO&invitationId=inv_136150b4-c618-427b-a256-95b0c8b2c08d

GA4-220501095-AA2-EV05 - Desarrollar la arquitectura de software de acuerdo con el patrón de diseño seleccionado

Diagrama de nodos

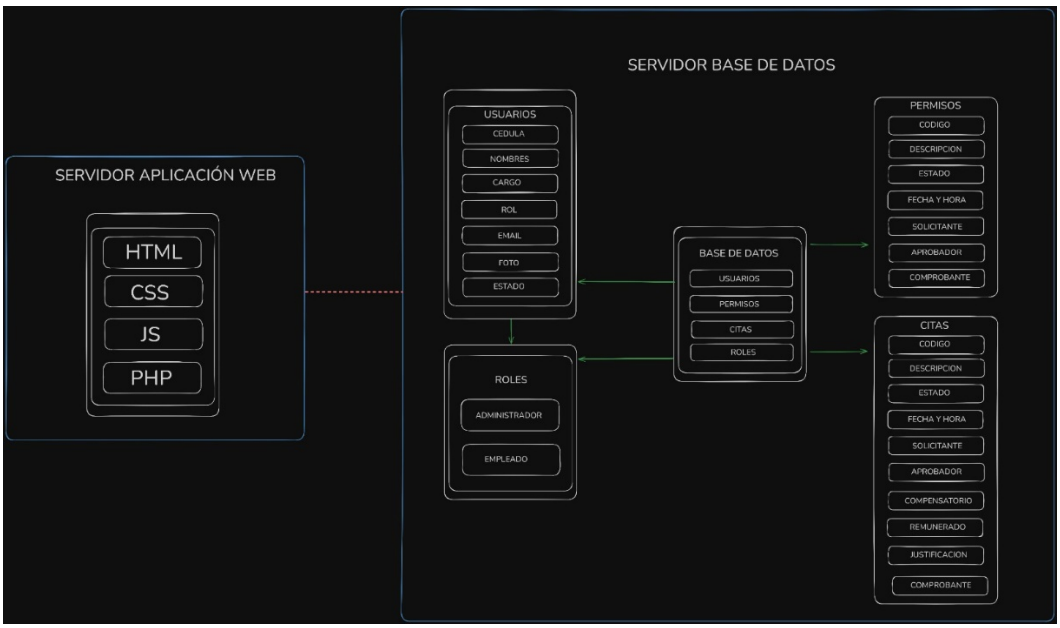
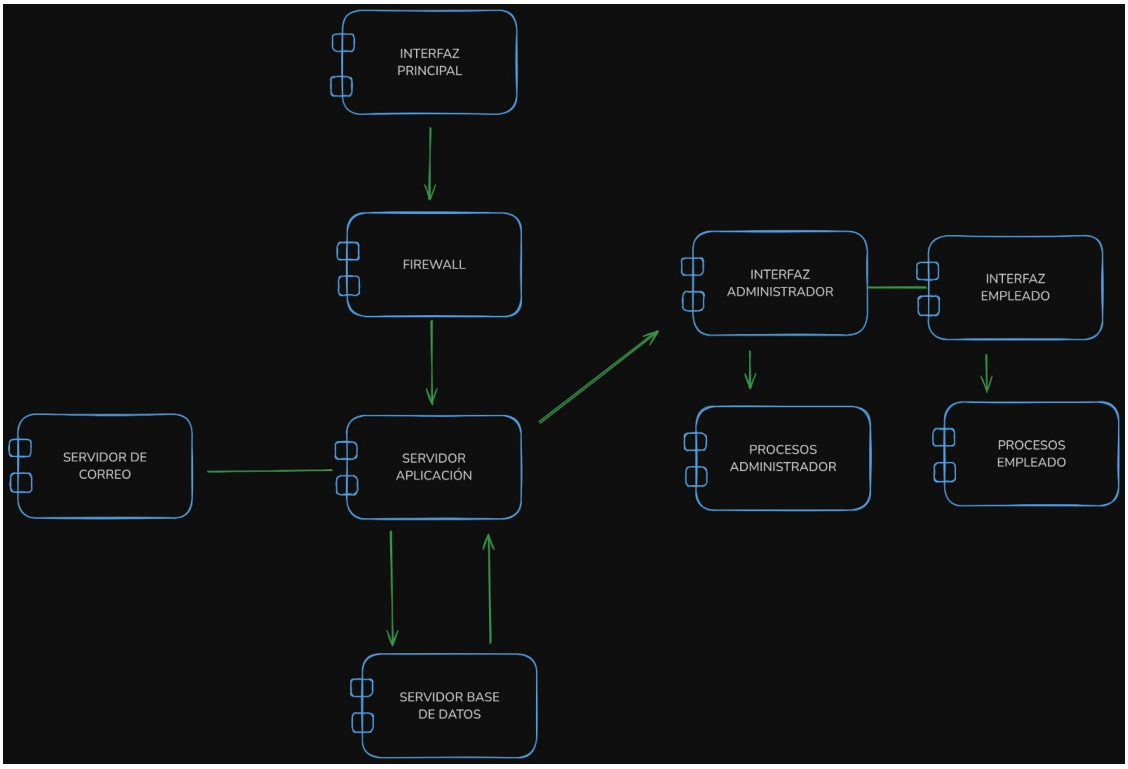


Diagrama de componentes



GA4-220501095-AA2-EV06 - Taller arquitectura de software

1. **¿Qué se entiende por arquitectura de software?**

Se refiere a la estructura fundamental y la organización de un sistema de software, incluyendo sus componentes, módulos y las relaciones entre ellos. Esta define la forma en que el software esta diseñado y como interactúan sus partes para cumplir con los requisitos funcionales y no funcionales del sistema.

2. **¿Cuál es su función?**

   Es proporcionar una estructura organizada y coherente para un sistema de software, esto incluye definir como se dividen las funcionalidades en módulos o componentes, como se comunican entre si y como se gestionan las dependencias. Además, la arquitectura del software busca asegurar que el sistema cumpla con requisitos como la escalabilidad, el rendimiento, la seguridad y la facilidad de mantenimiento. En resumen, su función es guiar y facilitar el diseño, desarrollo y mantenimientos efectivos del software.

3. **¿Cómo se elabora la arquitectura del software?**

   - ✓ Comprender los requisitos del sistema.
   - ✓ Identificar las componentes clave y sus interacciones.
   - ✓ Usar patrones de diseño y definir interfaces.
   - ✓ Mantener una separación clara de las responsabilidades.
   - ✓ Utilizar diagramas y documentación para visualizar la arquitectura.
   - ✓ Revisar y ajustar la arquitectura según sea necesario.
   - ✓ Documentar de manera concisa.
   - ✓ Implementar y probar el sistema según la arquitectura definida.

4. **¿Cómo lograr una buena arquitectura?**

- **Requisitos claros:** Comprender los requisitos funcionales y no funcionales del sistema.
- **Separación de preocupaciones:** Evitar que los componentes realicen múltiples tareas.
- **Modularidad:** Dividir el sistema en módulos o componentes independientes y reutilizables.
- **Escalabilidad:** Diseñar la arquitectura de manera que pueda crecer y adaptarse a medida que las necesidades del sistema evolucionan.
- **Flexibilidad:** Asegurarse de que la arquitectura sea flexible como para permitir cambios y actualizaciones sin ningún impacto a sus demás componentes.
- **Coherencia:** Mantener una estructura y estilo de diseño coherentes en todo el sistema para facilitar la comprensión y mantenimiento.
- **Patrones de diseño:** Utilizar patrones de diseño probados para abordar problemas comunes de diseño de software y mejorar la calidad de la arquitectura.
- **Documentación:** Documentar de manera clara y completa para que los miembros del equipo y demás interesados puedan comprenderla y colaborar de manera efectiva.
- **Evaluación continua:** Revisar y evaluar en todo el ciclo de desarrollo para identificar y abordar posibles problemas con respecto a los requisitos.
- **Colaboración:** Mantener una buena comunicación con los miembros del equipo.

5. ¿Cuáles son los elementos de diseño de una arquitectura de software?

- ✓ Componentes
- ✓ Interfaces
- ✓ Dependencias
- ✓ Patrones de diseño
- ✓ Capas
- ✓ Modelos de datos

- ✓ Modelos de comportamiento
- ✓ Seguridad
- ✓ Rendimiento
- ✓ Escalabilidad
- ✓ Documentación
- ✓ Mantenibilidad

Conclusiones

En conclusión, es de vital importancia repasar todos los temas que incluyen el proceso del desarrollo del software y así fortalecer nuestros conocimientos y tener presente todos esos términos a la hora de realizar actividades futuras.

En resumen, la arquitectura de software es como la estructura de una casa, pero más divertida y emocionante. Planificar y diseñar previamente cómo será la estructura del software garantiza su calidad, eficiencia y mantenibilidad a largo plazo.