Practica11

Nombres: Alex Travez, Mateo Oviedo

Fecha de realización: 15 /01/2025

Fecha de entrega: 16/01/2025

**Introducción**

La práctica 11 de Programación Avanzada tiene como objetivo desarrollar un sistema de gestión de vacaciones, utilizando diagramas UML para representar la estructura y el comportamiento del sistema. En este documento se presenta la explicación detallada de los diagramas elaborados: casos de uso, secuencia y clases.

**1. Diagrama de Casos de Uso**

**Descripción:** El diagrama de casos de uso identifica a los actores principales que interactúan con el sistema y los casos de uso asociados. Los actores definidos son:

* **Talento Humano (TH):** Responsable de asignar días de vacaciones y gestionar departamentos.
* **Empleado:** Usuario que solicita y consulta días de vacaciones.
* **Jefe del Departamento:** Encargado de aprobar o rechazar las solicitudes de vacaciones.

**Casos de Uso Incluidos:**

* Asignar días de vacaciones anuales.
* Solicitar días de vacaciones.
* Aprobar/rechazar solicitudes de vacaciones.
* Consultar días disponibles.
* Consultar solicitudes pendientes.
* Asignar empleado a departamento.

**Relaciones Principales:**

* Los actores interactúan directamente con los casos de uso.
* Existe una relación de dependencia entre "Solicitar días de vacaciones" y "Aprobar/Rechazar solicitud".

**Visualización:** El diagrama fue diseñado utilizando PlantUML para garantizar claridad y precisión en las interacciones.

Diagrama

Descripción generada automáticamente

Ilustración 1. Diagrama de casos de uso de gestión de vacaciones.

**2. Diagrama de Secuencia**

**Caso de Uso Representado:** Solicitar días de vacaciones.

**Descripción:** Este diagrama detalla el flujo de interacción entre los actores y el sistema durante el proceso de solicitud de vacaciones. Los elementos principales son:

* **Actor:** Empleado, quien inicia el proceso de solicitud.
* **Sistema:** Encargado de verificar la disponibilidad de días, gestionar la aprobación y actualizar el saldo.
* **Jefe del Departamento:** Revisa y decide sobre la solicitud.

**Pasos del Flujo Principal:**

1. El empleado inicia la solicitud.
2. El sistema verifica la disponibilidad de días de vacaciones.
3. El sistema envía la solicitud al jefe del departamento.
4. El jefe aprueba o rechaza la solicitud.
5. El sistema notifica la decisión al empleado y actualiza los datos.

**Visualización:** El diagrama enfatiza la secuencia lógica y temporal de los eventos.

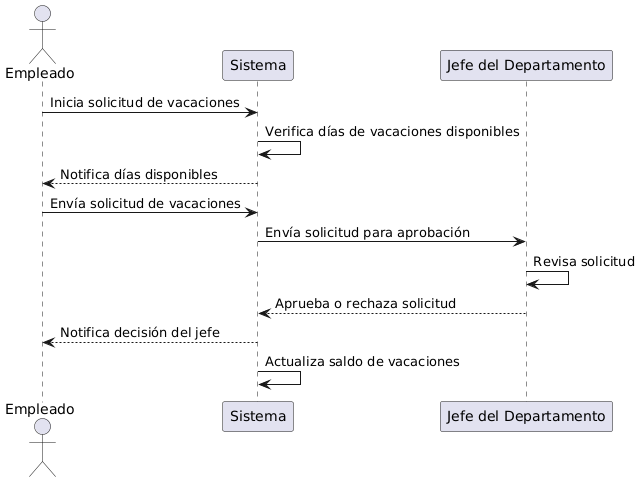


Ilustración 2. Diagramas de secuencia

**3. Diagrama de Clases**

**Descripción:** El diagrama de clases representa la estructura estática del sistema, incluyendo las clases principales, sus atributos, métodos y relaciones. Las clases modeladas son:

* **Empleado:** Clase base que contiene información general del empleado y permite la solicitud y consulta de vacaciones.
* **Funcionario:** Subclase de Empleado, que incluye funcionalidad adicional para pedir préstamos de días de vacaciones.
* **Contratado:** Subclase de Empleado, que gestiona la vigencia del contrato.
* **Jefe:** Clase que permite aprobar o rechazar solicitudes de vacaciones.
* **Departamento:** Clase que organiza empleados y asigna un jefe.
* **SistemaGestionVacaciones:** Clase principal que administra la asignación y procesamiento de solicitudes.
* **SolicitudVacaciones:** Clase que modela cada solicitud de vacaciones, incluyendo su estado y empleado asociado.

**Relaciones:**

* Empleado se relaciona con Jefe y Departamento.
* Funcionario y Contratado heredan de Empleado.
* SistemaGestionVacaciones interactúa con SolicitudVacaciones para procesar datos.

**Visualización:** El diagrama ilustra claramente las relaciones de herencia, composición y asociaciones entre las clases.

Diagrama

Descripción generada automáticamente

**Conclusiones**

1. Los diagramas UML permiten modelar la estructura y comportamiento del sistema de manera clara, facilitando la comunicación entre desarrolladores y partes interesadas.
2. La creación de un diagrama de clases bien definido garantiza un diseño sólido y estructurado para el sistema.
3. La representación de los casos de uso y las secuencias del sistema asegura que los flujos funcionales estén alineados con los requisitos planteados.

**Recomendaciones**

1. Validar continuamente los diagramas UML con los requisitos funcionales para evitar malentendidos durante la implementación.
2. Incluir comentarios y documentación detallada en cada diagrama para asegurar su comprensión por parte de todos los involucrados.
3. Usar herramientas colaborativas para mantener los diagramas actualizados y accesibles durante el ciclo de desarrollo.

**Referencias**

* **PlantUML:** Herramienta utilizada para la generación de diagramas. [Sitio oficial](https://plantuml.com/).
* **Documentación oficial Java:** Referencia para conceptos de programación orientada a objetos. [Oracle Java Documentation](https://docs.oracle.com/en/java/).
* **Práctica 11:** Documento base para el desarrollo de esta actividad. Facultad de Ingeniería Eléctrica y Electrónica, ITID433. Programación Avanzada.