DISEÑO ARQUITECTÓNICO DEL

SOFTWARE



**Curso:**  Administración de Proyectos de Software

**Sesión:** 05

**Docente:** Ing. José Vásquez Pereyra

# Nombre del Proyecto

Proyecto de Implementación de un Sistema de Asistente Virtual para una Empresa del Sector Hotelero.

# Introducción

En el presente documento se presenta el diseño arquitectónico propuesto para el "Proyecto de Implementación de un Sistema de Asistente Virtual para una Empresa del Sector Hotelero". Este diseño tiene como objetivo establecer una estructura técnica y funcional que respalde la solución, asegurando su eficiencia, escalabilidad y capacidad de adaptación a las necesidades del negocio. Se describen en detalle los componentes clave, las interacciones entre ellos y las tecnologías elegidas, todas alineadas con los objetivos estratégicos de modernización y mejora en la atención al cliente de la empresa hotelera. Documento presenta el diseño arquitectónico del sistema de software propuesto. Proporciona una visión general de los componentes principales del sistema, sus relaciones y cómo se integran para cumplir con los requisitos definidos.

# Objetivo del Diseño Arquitectónico

Desarrollar una arquitectura tecnológica sólida y eficiente para la implementación de un Sistema de Asistente Virtual en una empresa del sector hotelero, que potencie la atención al cliente, agilice los procesos internos y enriquezca la experiencia del usuario a través de soluciones innovadoras y escalables.

# Estilo Arquitectónico

El diseño arquitectónico propuesto adopta un enfoque basado en microservicios, lo que garantiza modularidad, escalabilidad y un mantenimiento simplificado del sistema de asistente virtual. Se implementa una arquitectura orientada a servicios (SOA), que facilita la integración entre los diferentes módulos del sistema. Además, se opta por mantener la operación en servidores locales de la empresa, lo que proporciona un mayor control, seguridad y disponibilidad interna.

# Diagrama de Arquitectura General del Sistema

El sistema se organiza en los siguientes componentes principales:

* Interfaz de Usuario (Front-end): Aplicación web que permite a los clientes interactuar con el asistente virtual para realizar consultas, reservas y solicitudes de servicios.
* Módulo de Asistente Virtual: Motor central que procesa las consultas de los usuarios utilizando procesamiento de lenguaje natural (NLP) y gestiona las respuestas automáticas.
* API Gateway: Componente que centraliza las solicitudes externas y dirige el tráfico hacia los servicios internos correspondientes de forma segura y eficiente.
* Servicios de Aplicación (Microservicios):
* Servicio de Reservas
* Servicio de Atención al Cliente
* Servicio de Gestión de Habitaciones

Cada uno de estos servicios gestiona funciones específicas de manera independiente.

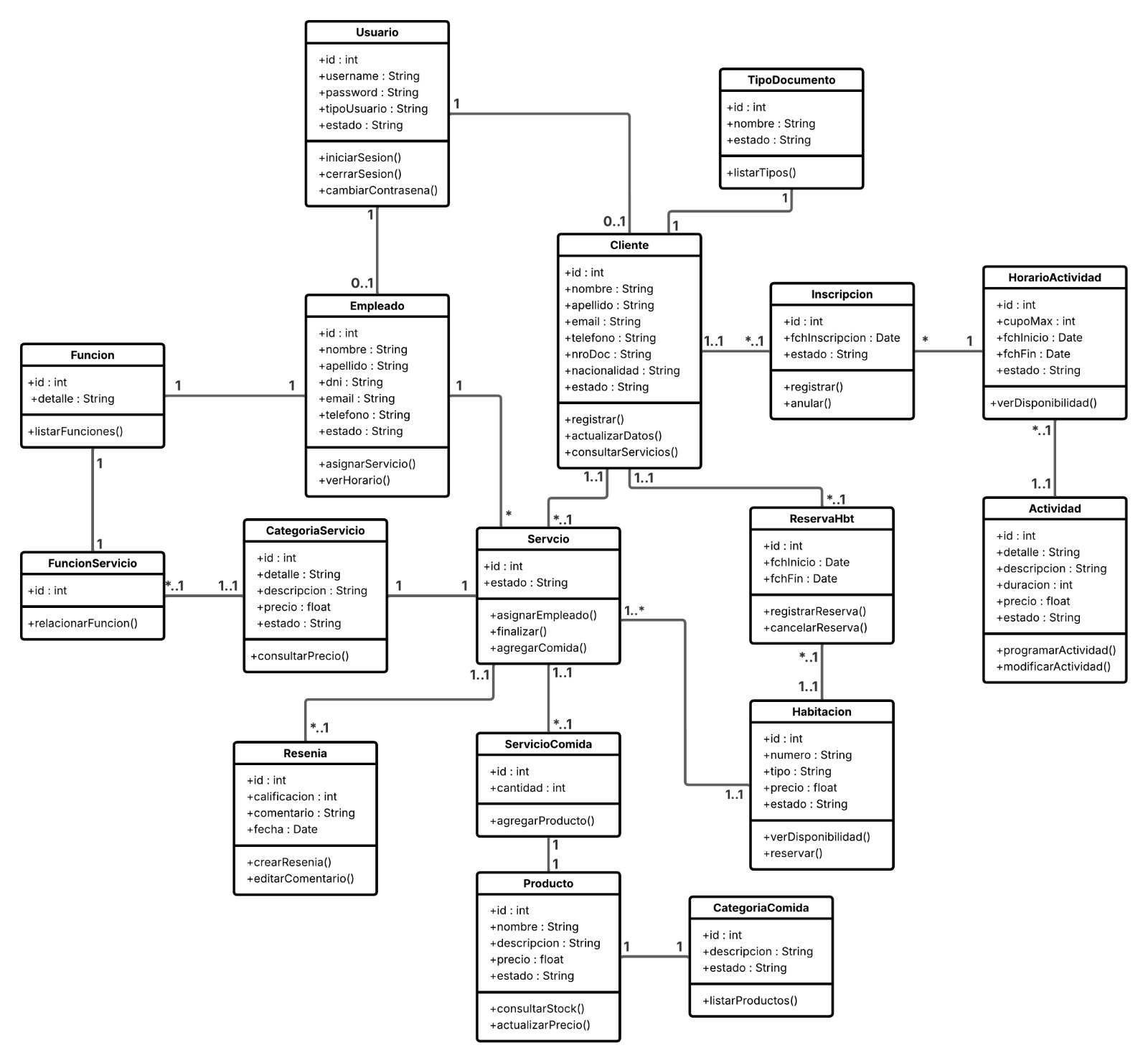
* Base de Datos: Almacena información de clientes, reservas, servicios y registros de conversaciones.
* Sistema de Administración: Plataforma interna para la gestión de contenidos, supervisión del asistente y monitoreo de interacciones.

# Descripción de Componentes

Detalle de los módulos o componentes que conforman el sistema y sus responsabilidades.

|  |  |  |  |
| --- | --- | --- | --- |
| **Componente** | **Descripción** | **Responsabilidades** | **Relaciones con otros componentes** |
| Frontend Web | Interfaz de usuario | Permitir interacción con el asistente | API Gateway |
| API Gateway | Enrutador de solicitudes | Gestionar comunicación entre usuarios y microservicios | Frontend, Microservicios |
| Servicio de Procesamiento de Lenguaje Natural | Módulo de interpretación de mensajes | Procesar lenguaje humano y generar respuestas | API Gateway, Base de Datos |
| Servicio de Atención al Cliente | Módulo de ayuda y consultas | Gestionar dudas y sugerencias de clientes | API Gateway |
| Base de Datos | Repositorio de datos | Almacenar información de usuarios, reservas y conversaciones | Microservicios |

# Diagramas de Diseño (UML)

* 1. **Diagrama de clases.**

1. **Cliente**

* **Atributos**: idCliente, nombre, email, telefono, estado.
* **Métodos**:
  + solicitarServicio()
  + cancelarServicio()
  + consultarHistorial()
* **Relación**:
  + Un cliente puede solicitar **cero o muchos servicios** (0..\*).

1. **Servicio**

* **Atributos**: idServicio, nombre, descripcion, precio, disponibilidad.
* **Métodos**:

activarServicio()

editar()

verDetalles()

* **Relaciones**:

Cada servicio es solicitado por **exactamente un cliente** (1).

Cada servicio está **ligado a una habitación obligatoriamente** (1).

Cada servicio es **atendido por exactamente un empleado** (1).

Un servicio puede estar relacionado con **una reserva** (opcional) (0..1).

1. **Empleado**

* **Atributos:** idEmpleado, nombre, email, telefono, cargo, estado.
* **Método:**

aceptarTarea()

* **Relación:**

Cada empleado atiende uno o varios servicios (1..\*).

1. **Habitación**

* **Atributos:** idHabitacion, numeroHabitacion, tipo, estado.
* **Método:**

consultarEstado()

* **Relación:**

Cada habitación puede estar asociada a varios servicios (0..\*).

1. **Reserva**

* **Atributos:** idReserva, descripcion, fechaInicio, fechaFin, estado.
* **Método:**

verDetalles()

* **Relación:**

Una reserva puede incluir cero o varios servicios (0..\*).

* 1. **Diagrama de secuencia.**

1. **Nombre:** Diagrama de Secuencia - Atención de Servicios y Reservas con Apoyo de Empleado
2. **Descripción**: El diagrama muestra el flujo de interacción entre el Cliente, el Asistente Virtual, la Base de Datos y el Empleado (Recepcionista) para atender una solicitud de servicio o realizar una reserva en el hotel.
3. El proceso general es el siguiente:

* El Cliente envía una solicitud al Asistente Virtual.
* El Asistente Virtual interpreta la solicitud usando procesamiento de lenguaje natural (NLP).
* El Asistente Virtual consulta a la Base de Datos para verificar la disponibilidad de los servicios o habitaciones.
* La Base de Datos, en ciertos casos, asigna directamente una consulta a un Empleado para su gestión manual.
* El Empleado confirma o responde la solicitud a la Base de Datos, que a su vez informa al Asistente Virtual.
* En situaciones donde se requiere aprobación adicional (por ejemplo, reservas especiales o solicitudes complejas), el Asistente Virtual solicita a la Base de Datos la asignación de un Recepcionista disponible.
* El Empleado revisa los detalles, realiza las validaciones necesarias y confirma o rechaza la solicitud.
* Finalmente, el Asistente Virtual notifica al Cliente el estado de su solicitud o reserva.

Diagrama de secuencia de solicitudes de servicio

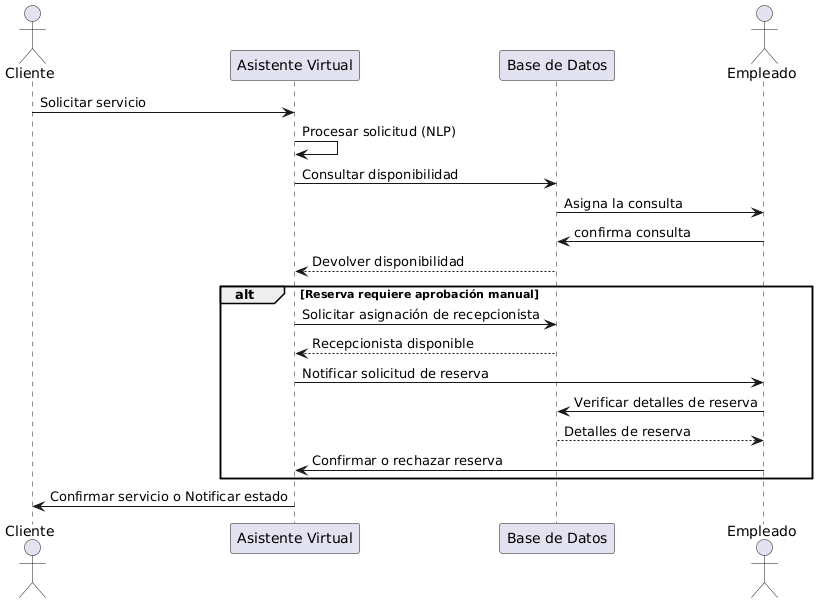
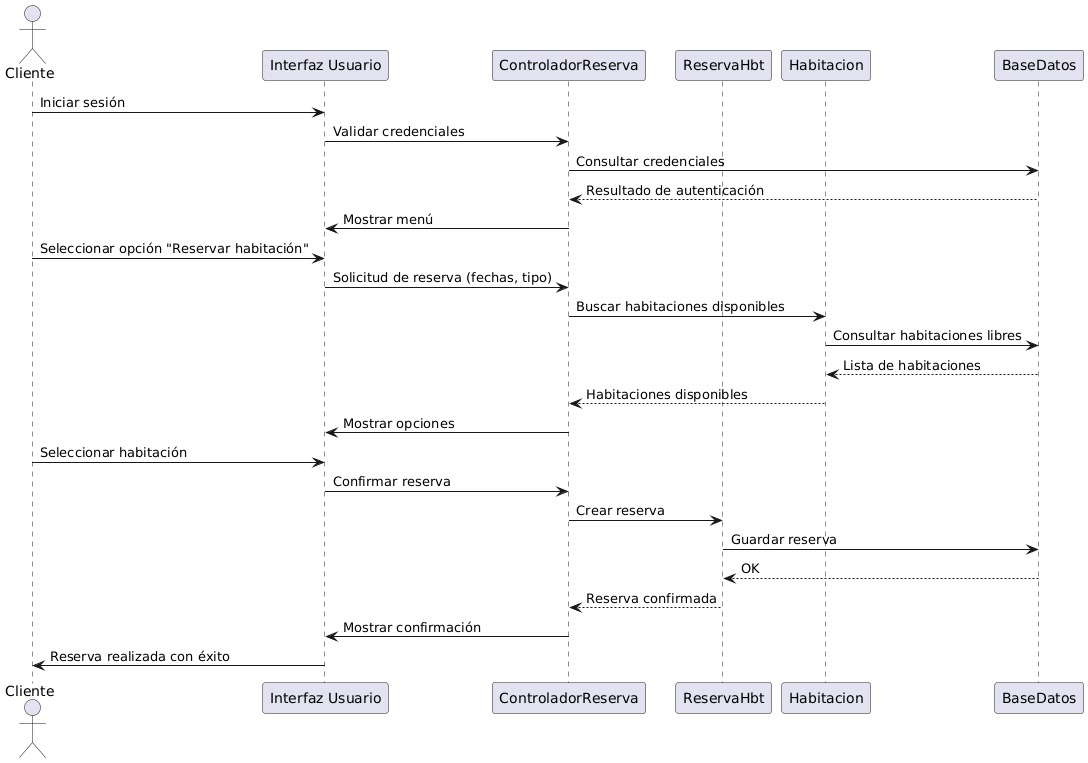


Diagrama de secuencia reservas

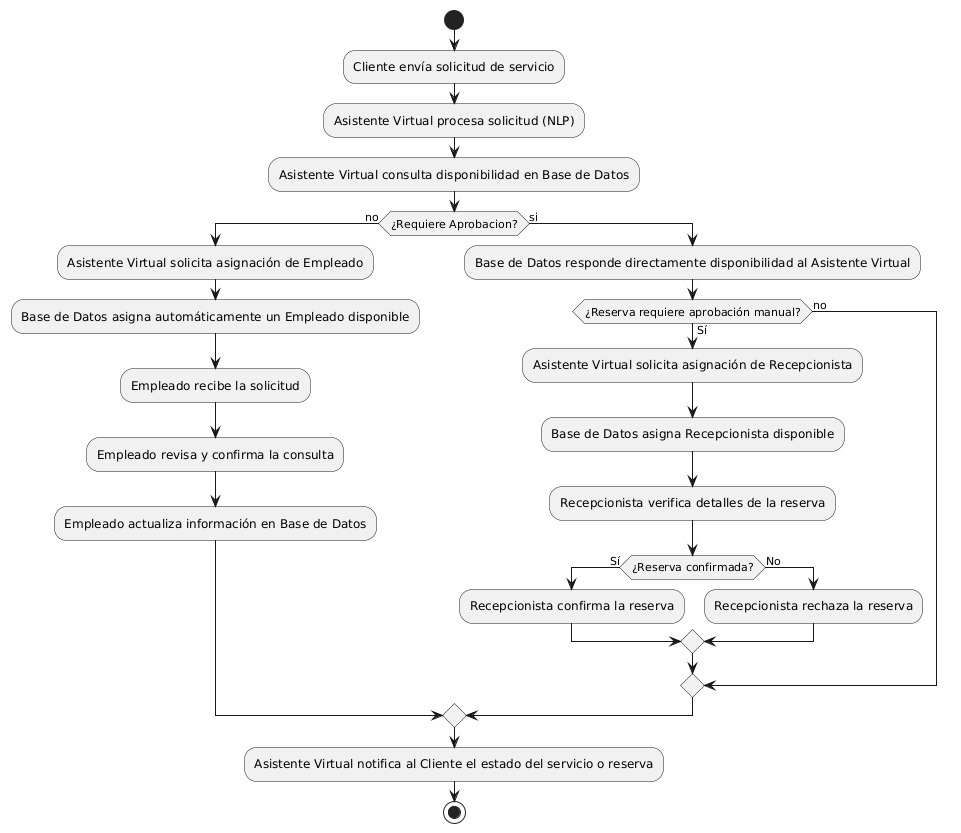


* 1. Diagrama de actividad.

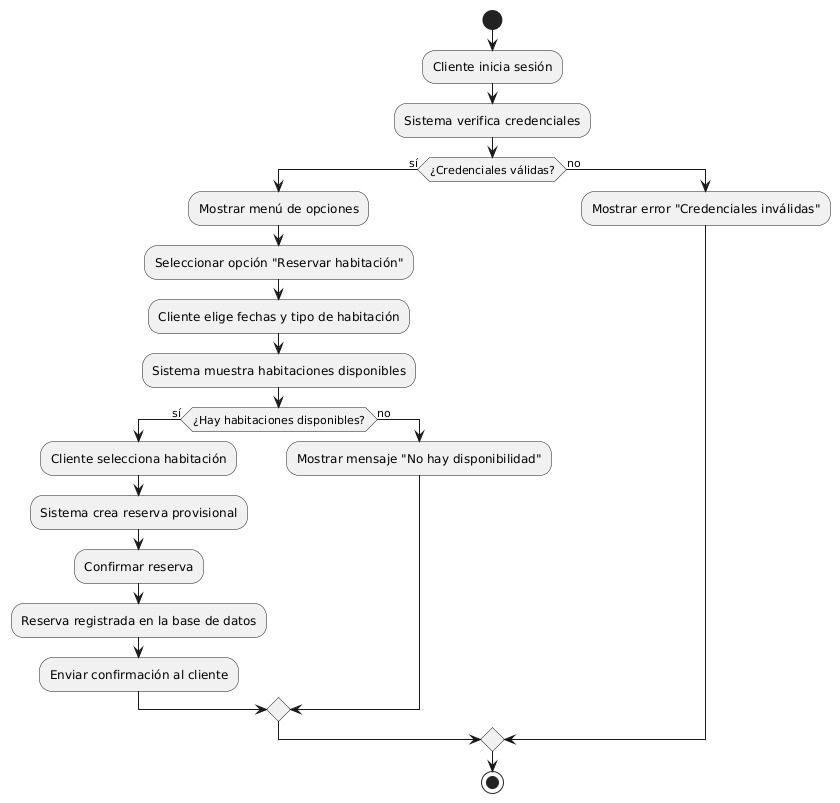
Este diagrama representa el flujo de un proceso en el que un Asistente Virtual maneja solicitudes de servicio o reservas de clientes.

* 1. **Cliente envía solicitud de servicio**: El cliente inicia el proceso enviando una solicitud para algún tipo de servicio.
  2. **Asistente Virtual procesa solicitud (NLP)**: El Asistente Virtual utiliza procesamiento de lenguaje natural (NLP) para entender y analizar la solicitud del cliente.
  3. **Asistente Virtual consulta disponibilidad en Base de Datos**: El Asistente Virtual consulta la Base de Datos para verificar si hay disponibilidad para el servicio solicitado.
  4. **¿Requiere Aprobación?**: Se verifica si la solicitud requiere aprobación antes de continuar con el proceso.
* **No**:
  + Base de Datos asigna automáticamente un Empleado disponible
  + Empleado recibe la solicitud.
  + Empleado revisa y confirma la consulta.
  + Empleado actualiza información en Base de Datos.
* **Sí**:
  + **¿Reserva requiere aprobación manual?**:
    - **Sí**:
      * Base de Datos asigna Recepcionista disponible:
      * Recepcionista verifica detalles de la reserva:
      * ¿Reserva confirmada?:
        + **Sí**: Si es confirmada, el recepcionista confirma la reserva.
        + **No**: Si no es confirmada, el recepcionista rechaza la reserva.
  1. **Asistente Virtual notifica al Cliente el estado del servicio o reserva**: Finalmente, el Asistente Virtual notifica al cliente el estado de su solicitud o reserva, ya sea confirmada o rechazada.

DIA Solicitud de servicios



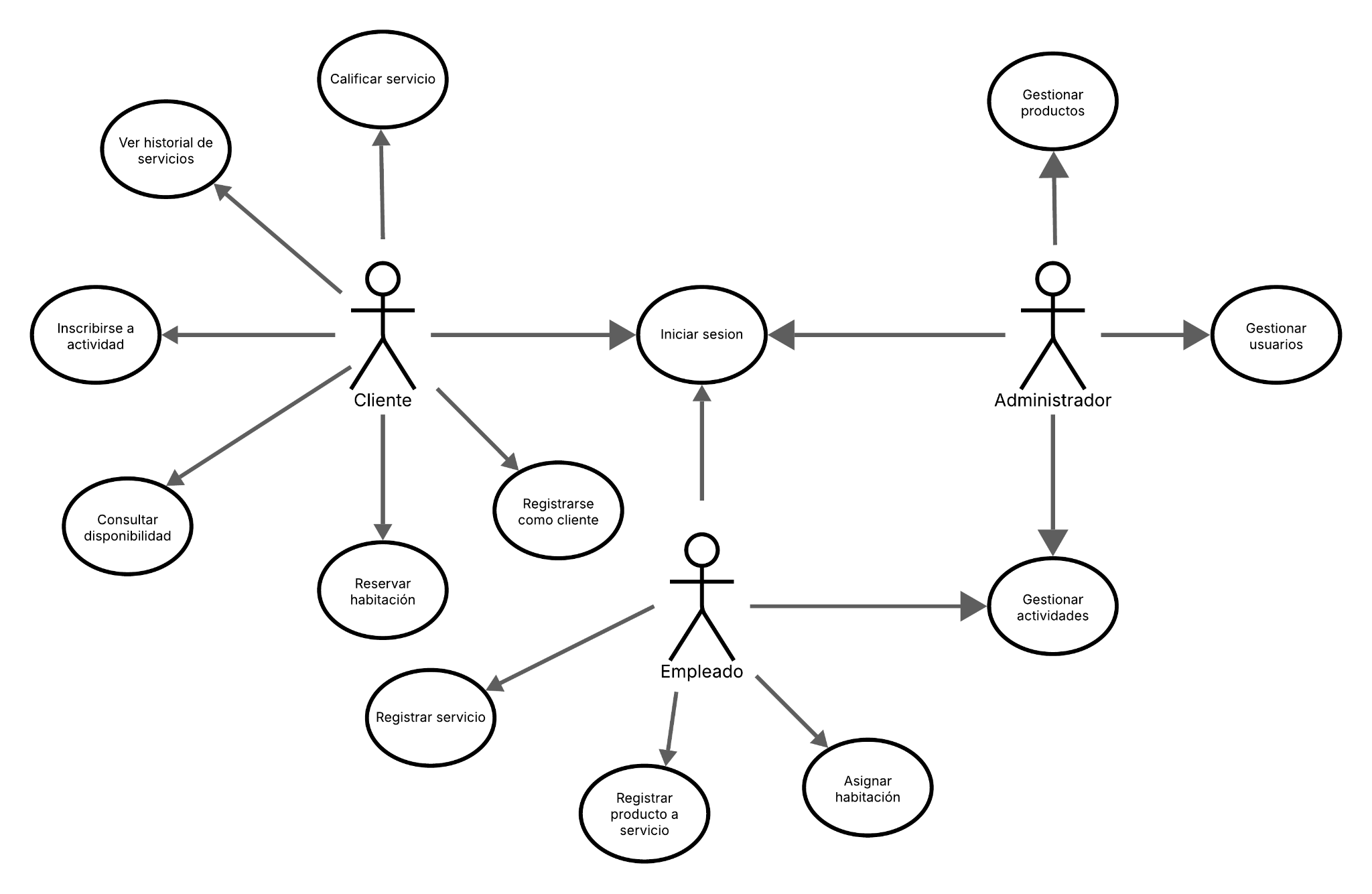
DIA Reservas



* 1. Diagrama de casos de uso.

1. Actores:

* **Cliente**: El usuario que solicita servicios y consulta su historial.
* **Empleado**: El que atiende los servicios solicitados.
* **Sistema**: Para manejar las operaciones generales como activar servicios, editar, ver detalles, etc.

1. Casos de uso:

* **Solicitar servicio**: El cliente puede solicitar un servicio.
* **Cancelar servicio**: El cliente puede cancelar un servicio previamente solicitado.
* **Consultar historial**: El cliente puede consultar el historial de servicios solicitados.
* **Activar servicio**: El sistema o empleado activa un servicio solicitado.
* **Editar servicio**: El empleado puede editar los detalles de un servicio.
* **Ver detalles del servicio**: Tanto el cliente como el empleado pueden ver los detalles de un servicio.
* **Aceptar tarea**: El empleado puede aceptar una tarea o servicio asignado.
* **Consultar estado de habitación**: El cliente o empleado puede consultar el estado de una habitación asociada a un servicio.
* **Ver detalles de la reserva**: El cliente puede consultar los detalles de una reserva asociada a un servicio.

1. Descripción de relaciones:

**Cliente**:

* + **Solicitar servicio**: El cliente solicita un servicio al sistema.**Cancelar servicio**: El cliente puede cancelar un servicio previamente solicitado.
  + **Consultar historial**: El cliente puede ver el historial de los servicios solicitados.
  + **Ver detalles del servicio**: El cliente puede consultar los detalles de un servicio solicitado.
  + **Ver detalles de la reserva**: El cliente puede consultar los detalles de la reserva asociada a un servicio.

**Empleado**:

* + **Aceptar tarea**: El empleado acepta el servicio asignado por el sistema.
  + **Editar servicio**: El empleado tiene la capacidad de editar los detalles de los servicios.
  + **Ver detalles del servicio**: El empleado puede consultar los detalles de un servicio para gestionarlo.
  + **Consultar estado de habitación**: El empleado puede verificar el estado de las habitaciones asociadas a los servicios.

**Sistema**:

* + **Activar servicio**: El sistema o el empleado activa el servicio solicitado por el cliente.
  + **Ver detalles del servicio**: El sistema permite tanto a clientes como empleados ver los detalles de los servicios.
  + **Consultar estado de habitación**: El sistema consulta el estado de las habitaciones asociadas a un servicio.

# Patrones de Diseño Utilizados

* 1. Singleton

Propósito: Asegurar que una clase tenga solo una instancia y proporcionar un punto de acceso global a ella.

Cómo ayuda: El patrón Singleton garantiza que el sistema no cree múltiples instancias de clases críticas, lo que optimiza el uso de recursos y asegura que todas las operaciones accedan a la misma fuente de configuración o base de datos.

b. Factory Method

Propósito: Proveer una interfaz para crear objetos en una clase base, pero permitir que las subclases decidan qué tipo de objeto instanciar.

Cómo ayuda: Facilita la creación de objetos de una manera flexible, manteniendo el código desacoplado y permitiendo la fácil expansión del sistema en el futuro, sin necesidad de modificar clases existentes.

c. Observer

Propósito: Permitir que un objeto (el sujeto) notifique a otros objetos (observadores) cuando su estado cambie.

Cómo ayuda: Este patrón asegura que el sistema reaccione automáticamente a ciertos eventos, lo que mejora la interacción en tiempo real entre los componentes sin que el código dependa directamente de la notificación.

d. Strategy

Propósito: Definir una familia de algoritmos, encapsular cada uno de ellos y hacerlos intercambiables.

Cómo ayuda: Permite cambiar la lógica de procesamiento sin alterar el código que usa esa lógica, lo que hace que el sistema sea más flexible y fácil de modificar o expandir según se desarrollen nuevas estrategias de machine learning.

# Consideraciones de Seguridad, Rendimiento y Escalabilidad

**9.1 Consideraciones de Seguridad**

1. **Autenticación y Autorización:**

El sistema implementará una autenticación robusta mediante OAuth2 o JWT (JSON Web Tokens) para asegurar que solo los usuarios autenticados accedan a funcionalidades sensibles.

Roles y permisos se definirán para los usuarios (Clientes, Empleados, Administradores) para garantizar que cada uno tenga acceso solo a las funcionalidades que le corresponden.

1. **Protección contra Inyecciones SQL:**

Se utilizarán consultas parametrizadas y ORMs (Object Relational Mappers) para evitar inyecciones SQL, lo que protege al sistema de ataques maliciosos.

* 1. **Consideraciones de Rendimiento**

1. **Optimización de Consultas a la Base de Datos:**

Se utilizarán índices en la base de datos para acelerar las consultas más frecuentes, como las relacionadas con las reservas y los servicios.

Se optimizarán las consultas SQL para reducir el tiempo de ejecución, utilizando técnicas como la paginación en las consultas de grandes volúmenes de datos.

1. **Uso de Caché:**

Se implementará un sistema de caché utilizando Redis o Memcached para almacenar en memoria las consultas más frecuentes (como la información de servicios disponibles), lo que reducirá la carga en la base de datos y mejorará los tiempos de respuesta.

1. **Optimización de Recursos en el Frontend:**

Se utilizarán técnicas de mitificación y compresión para los archivos JavaScript y CSS, así como el uso de imágenes optimizadas para reducir los tiempos de carga del frontend y mejorar la experiencia del usuario.

1. **Asincronía en Procesos Lentos:**

Las operaciones de larga duración, como el procesamiento de reserva de servicios o el análisis de datos, se manejarán de manera asíncrona para no bloquear la respuesta al usuario, utilizando colas de mensajes o procesos en segundo plano.

* 1. **Consideraciones de Escalabilidad**
     1. **Escalabilidad Horizontal:**

El sistema se diseñará para ser escalable horizontalmente, permitiendo agregar más instancias de servidores según sea necesario para manejar aumentos en el tráfico de usuarios.

Se utilizarán tecnologías de balanceo de carga (como Nginx o HAProxy) para distribuir el tráfico entre varias instancias del servidor.

* + 1. **Microservicios (Opcional):**

El sistema puede adoptar una arquitectura de **microservicios** en el futuro, separando funcionalidades clave como la gestión de reservas, servicios, y autenticación en servicios independientes que puedan escalar de forma independiente según las demandas del negocio.

* + 1. **Base de Datos Escalable:**

Se utilizará una base de datos distribuida o sharding para poder manejar grandes volúmenes de datos y permitir que la base de datos se escale fácilmente cuando sea necesario.

Se evaluará el uso de bases de datos NoSQL (como MongoDB) para gestionar datos no estructurados que puedan crecer rápidamente.

* + 1. **Elasticidad en la Nube:**

El sistema estará preparado para ser desplegado en infraestructuras en la nube (como AWS, Google Cloud o Azure) para aprovechar la elasticidad y la capacidad de ajustarse automáticamente a los picos de demanda, con recursos que se ajustan de manera dinámica.

* + 1. **Redundancia y Recuperación ante Desastres:**

Se implementarán soluciones de backup automático y se asegurarán estrategias de recuperación ante desastres para garantizar la disponibilidad del sistema ante fallos.

# Supuestos y Restricciones Técnicas

|  |  |
| --- | --- |
| **Supuesto o Restricción** | **Descripción** |
| Uso de tecnología web (Frontend) | El sistema solo contará con una interfaz web. Esto implica que el diseño debe ser completamente adaptativo y optimizado para varios dispositivos y navegadores. |
| Base de Datos Relacional | Se utilizará una base de datos relacional para almacenar los datos del sistema. Esto limita la estructura a tablas con relaciones predefinidas y puede no ser ideal para datos no estructurados. |
| Escalabilidad Horizontal | Se asume que el sistema podrá escalar horizontalmente. Esto requiere que la infraestructura esté preparada para aumentar las instancias de servidores sin problemas, lo que impacta en la elección de tecnologías. |
| Alta disponibilidad y redundancia | El sistema debe ser altamente disponible, con tolerancia a fallos. Esto afecta la decisión de infraestructura, garantizando copias de seguridad y mecanismos de recuperación ante desastres. |
| Tiempo de respuesta bajo para el cliente | El sistema debe asegurar tiempos de respuesta bajos en la interfaz de usuario. Esto influye en las decisiones sobre caché, optimización de consultas y arquitectura general del backend. |
| Accesibilidad y compatibilidad web | La interfaz web debe ser accesible y compatible con los principales navegadores y dispositivos. Esto afecta el diseño de la interfaz, la elección de tecnologías frontend y la validación de accesibilidad. |
| Uso de APIs externas | Se utilizarán APIs de terceros (por ejemplo, para pagos o análisis de datos). Esto obliga a la integración con servicios externos y puede generar dependencias que deben ser gestionadas adecuadamente. |
| Capacidad de integrar nuevos servicios | Se debe permitir la integración de nuevos servicios a medida que el negocio crezca (por ejemplo, nuevos servicios de hotel). Esto requiere una arquitectura modular que facilite la adición de nuevas funcionalidades. |

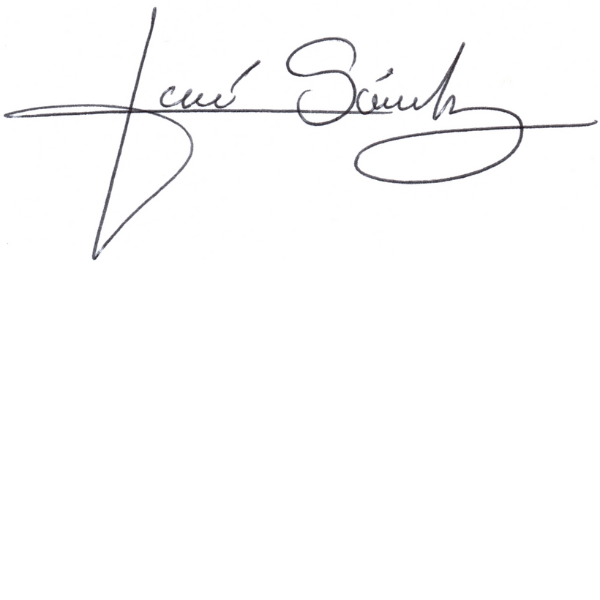
# Aprobaciones y Validaciones del Documento

Nombre: Martinez Rodriguez, Nilver Jefferson

Firma: Diagrama

El contenido generado por IA puede ser incorrecto.

Fecha: 25/04/25

Nombre: Sánchez Saavedra, Brayan José 

Firma:

Fecha: 25/04/25

# Conclusiones

Las decisiones arquitectónicas tomadas en el desarrollo del Sistema de Asistente Virtual para una Empresa del Sector Hotelero están orientadas a garantizar una solución escalable, segura y eficiente. Se ha optado por una arquitectura basada en la web, utilizando patrones de diseño como Singleton, Factory y Observer para asegurar flexibilidad y mantenimiento fácil. La escalabilidad horizontal y las consideraciones de seguridad, rendimiento y cumplimiento regulatorio aseguran que el sistema sea capaz de manejar el crecimiento de la demanda, mantener tiempos de respuesta bajos y proteger los datos sensibles de los usuarios, cumpliendo con los objetivos del proyecto..

# Aprendizaje del Equipo

|  |  |
| --- | --- |
| **Aspecto Aprendido** | **Comentario del Equipo** |
| Diseño de Arquitectura Web Escalable | El equipo ha aprendido a crear una arquitectura que soporta escalabilidad horizontal, lo cual es crucial para asegurar que el sistema pueda crecer sin comprometer su rendimiento. |
| Patrones de Diseño | La implementación de patrones como Singleton, Factory y Observer ha permitido al equipo comprender cómo aplicar soluciones estructuradas para mantener el sistema flexible y modular. |
| Seguridad en Aplicaciones Web | El equipo ha profundizado en técnicas de seguridad, como cifrado de datos y autenticación robusta, lo cual es esencial para proteger los datos de los usuarios y cumplir con regulaciones. |
| Optimización de Rendimiento | Aprendimos a implementar estrategias de caché y optimización de consultas para mejorar el tiempo de respuesta del sistema, lo que es vital para una buena experiencia de usuario. |