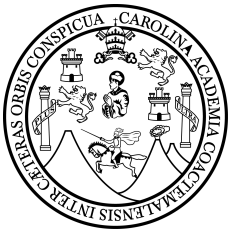
Teoría de Sistemas 1
Ingeniería en Ciencias y Sistemas
División de Ciencias de la Ingeniería
Centro Universitario de Occidente
Universidad de San Carlos de Guatemala



SISTEMA HOTELERÍA

Enlace Repositorio

https://github.com/Isaac03483/Hoteru

Marco Teórico.

11/21/2,023

Presentación del Proyecto y Pensamiento Sistémico:

En este análisis del sistema de recepción de un hotel, se ha adoptado un enfoque sistémico para comprender y evaluar su funcionamiento. El sistema de recepción es un componente crítico de la operación hotelera, y su eficiencia y eficacia son fundamentales para garantizar una experiencia positiva para los huéspedes. A través de este enfoque, se exploraron los roles, interacciones y subsistemas que conforman este sistema complejo.

Principales Objetivos:

El objetivo principal del sistema de recepción de un hotel es proporcionar un servicio eficiente y eficaz a los huéspedes. Este objetivo se desglosa en los siguientes subobjetivos:

- Facilitar el proceso de registro (check-in) y salida (check-out) de los huéspedes de manera rápida y sin complicaciones.
- Ofrecer información precisa y detallada sobre el hotel y sus servicios.
- Satisfacer las necesidades y solicitudes de los huéspedes, como reservas de actividades, transporte y otras consultas.
- Gestionar las reservas de habitaciones de manera eficiente y mantener actualizada la información sobre los huéspedes.
- Fomentar una comunicación efectiva entre los empleados del hotel para brindar un servicio coordinado y cohesivo con el uso de las tecnologías que se implementarán.

Tecnologías a Utilizar:

Laravel y PHP:

PHP es un lenguaje de programación de propósito general, interpretado y de código abierto, diseñado principalmente para el desarrollo web. Se utiliza ampliamente en la creación de sitios web y aplicaciones web.

Laravel es un framework de PHP de código abierto, creado por Taylor Otwell en 2011. Está basado en el patrón de arquitectura MVC (Modelo-Vista-Controlador) y proporciona una gran cantidad de características y funcionalidades para el desarrollo de aplicaciones web.

Ventajas de Laravel

- **Facilidad de uso:** Laravel es un framework muy bien diseñado y documentado, lo que lo hace fácil de aprender y usar.
- **Productividad:** Laravel proporciona una gran cantidad de características y funcionalidades que pueden ayudar a los desarrolladores a ser más productivos.
- **Seguridad:** Laravel tiene una gran atención a la seguridad, lo que puede ayudar a los desarrolladores a crear aplicaciones web seguras.

Características de Laravel

- **ORM (Object-Relational Mapping):** Laravel incluye un ORM (Object-Relational Mapping) integrado, que facilita la interacción con las bases de datos.
- **Enrutamiento:** Laravel proporciona un sistema de enrutamiento flexible y potente, que permite a los desarrolladores definir cómo se mapean las URL a las acciones de las aplicaciones.
- **Vistas:** Laravel proporciona una forma sencilla de crear y gestionar vistas, que son las páginas web que se muestran a los usuarios.
- **Controladores:** Los controladores son responsables de procesar las solicitudes de los usuarios y devolver las respuestas.
- **Middleware:** El middleware es una forma de añadir funcionalidad adicional a las aplicaciones web.

Usos de Laravel

Laravel se puede utilizar para crear una amplia gama de aplicaciones web, incluyendo:

- **Sitios web dinámicos**
- **Aplicaciones web de comercio electrónico**
- **Aplicaciones web de gestión de contenido**
- **Aplicaciones web de redes sociales**
- **Aplicaciones web de juegos**

Puedes instalar y obtener más información sobre Laravel en su página web: Laravel

Angular:

Angular es un marco (framework) de desarrollo para construir aplicaciones web de una sola página (SPA, por sus siglas en inglés). Fue desarrollado por Google y lanzado como AngularJS en 2010, pero posteriormente se reescribió por completo y se lanzó como Angular 2 en 2016.

Puedes instalar Angular y obtener más información desde su página oficial: Angular

Ventajas de Angular

- **Facilidad de uso:** Angular es un framework muy bien diseñado y documentado, lo que lo hace fácil de aprender y usar.
- **Productividad:** Angular proporciona una gran cantidad de características y funcionalidades que pueden ayudar a los desarrolladores a ser más productivos.
- **Seguridad:** Angular tiene una gran atención a la seguridad, lo que puede ayudar a los desarrolladores a crear aplicaciones web seguras.

Características de Angular

- **Componentes:** Angular utiliza componentes para construir la interfaz de usuario de las aplicaciones. Los componentes son unidades de código reutilizables que pueden agruparse para crear aplicaciones complejas.
- **Data binding:** Angular proporciona un sistema de data binding que permite a los desarrolladores vincular los datos de la aplicación a la interfaz de usuario.

- **Routing:** Angular proporciona un sistema de enrutamiento que permite a los desarrolladores definir cómo se mapean las URL a las diferentes partes de la aplicación.

Angular material:

Angular Material es una biblioteca de componentes de interfaz de usuario (UI) desarrollada por el equipo de Angular en colaboración con Google Material Design. Está diseñada para facilitar la creación de aplicaciones web con una apariencia y experiencia de usuario coherentes y atractivas, siguiendo los principios de diseño de Material Design.

Para importar o agregar angular material a nuestro proyecto de angular lo podemos hacer mediante el siguiente comando:

ng add @angular/material

Ventajas de Angular Material

- **Diseño:** Angular Material proporciona una amplia gama de componentes de interfaz de usuario con un diseño moderno y atractivo.
- **Facilidad de uso:** Angular Material es muy fácil de usar. Los componentes se pueden importar y utilizar fácilmente en cualquier aplicación Angular.
- **Documentación:** Angular Material tiene una excelente documentación que ayuda a los desarrolladores a aprender a utilizar los componentes.

Características de Angular Material

- **Componentes:** Angular Material proporciona una amplia gama de componentes de interfaz de usuario, incluyendo botones, campos de formulario, listas, menús, etc.
- **Temática:** Angular Material proporciona un sistema de tematización que permite a los desarrolladores personalizar el aspecto de los componentes.

PostgreSQL:

PostgreSQL es un sistema de gestión de bases de datos relacional de código abierto y potente. Es conocido por su capacidad de manejar grandes cantidades de datos y por su conformidad con los estándares SQL.

postgreSQL es una opción popular para aplicaciones empresariales y proyectos que requieren robustez, escalabilidad y conformidad con los estándares SQL. Puedes descargar PostgreSQL y encontrar más información en el sitio web oficial: PostgreSQL.

NGX Charts:

Ngx-Charts es una biblioteca de gráficos de código abierto para aplicaciones Angular. Está basada en la biblioteca de gráficos Chart.js y proporciona una amplia gama de tipos de gráficos, incluyendo:

- **Barra**
- **Punto**
- **Línea**
- **Pastel**
- **Dona**
- **Radar**
- **Scatter**
- **Heatmap**

Ngx-Charts es muy fácil de usar. Los gráficos se pueden crear simplemente importando el componente correspondiente y pasando los datos al componente.

Puedes importar o agregar esta biblioteca con el siguiente comando:

npm install @swimlane/ngx-charts --save

Puedes obtener más información y ejemplos en su sitio web: ngx-charts

Sistema de Hotelería Hoteru:

El sistema de hotelería **Hoteru** se creó con la finalidad de poder manejar de manera eficiente toda la información necesaria para el hotel. Este sistema cuenta con distintos módulos que explicaremos más adelante. Este sistema cuenta con los siguientes módulos:

- Módulo de Administrador.
- Módulo de Recepcionista.
- Módulo de Empleado.

Módulo de Administrador:

El administrador es el empleado más importante del sistema. Esto quiere decir que él será el encargado de manejar toda la información dentro del sistema. El administrador cuenta con las siguientes funcionalidades:

1. **Crear empleados:** El administrador puede crear varios empleados que sean capaces de ingresar al sistema en base a su tipo de empleado.

2. **Modificar empleados:** El administrador puede cambiar la información de los empleados. Entre los campos que se pueden actualizar se encuentran: Nombre del empleado, tipo de empleado y contraseña.
3. **Eliminar empleados:** El administrador puede eliminar permanentemente a otro empleado del sistema.
4. **Crear tipo de empleado:** El administrador puede agregar distintos tipos de empleados al sistema.
5. **Eliminar tipo de empleados:** El administrador puede eliminar tipos de empleados del sistema. Sin embargo, este no puede eliminar los tipos de empleados: admin y recepcionista, esto ya que esos tipos de empleados son importantes para el sistema.
6. **Listar empleados y tipos de empleados:** El administrador puede listar los empleados y tipos de empleados en el sistema. Además puede filtrar la información para hacer una búsqueda más rápida.
7. **Crear habitación:** El administrador puede agregar habitaciones al sistema.
8. **Eliminar habitación:** El administrador puede eliminar habitaciones del sistema.
9. **Crear tipo de habitación:** El administrador puede agregar distintos tipos de habitación al sistema. La información a ingresar es: Nombre del tipo de habitación y precio por día.
10. **Eliminar tipo de habitación:** El administrador puede eliminar un tipo de habitación del sistema siempre y cuando este no se haya asignado a alguna habitación.
11. **Listar habitaciones y tipos de habitación:** El administrador puede listar todas las habitaciones y tipos de habitación registradas en el sistema. Además este puede filtrar la información para hacer una búsqueda más rápida.
12. **Crear tarea:** El administrador puede crear tareas que los demás tipos de empleados deben realizar.
13. **Modificar tarea:** El administrador puede modificar la información sobre la tarea.
14. **Ver detalles:** El administrador puede visualizar los detalles de una tarea previamente ingresada.
15. **Eliminar tarea:** El administrador puede eliminar permanentemente una tarea del sistema.
16. **Listar tareas:** El administrador puede visualizar todas las tareas ingresadas en el sistema así como su estado actual (pendiente, asignada o realizada).
17. **Listar reservaciones:** El administrador puede visualizar todas las reservaciones hechas en el sistema.
18. **Dashboard:** La página principal nos muestra 4 reportes. Estos son: Gráfica de tareas de hoy, gráfica de tipos de habitación más solicitadas, Reporte de ganancias en un intervalo de tiempo y Clientes que más han visitado el hotel.

Módulo de Recepcionista:

La recepcionista será la encargada del manejo de todas las reservaciones dentro del sistema. Para esto ella deberá contar con las siguientes funcionalidades:

1. **Agregar reservación:** La recepcionista puede crear reservaciones en el sistema.
2. **Listar tipos de habitación:** La recepcionista podrá listar los distintos tipos de habitaciones dentro del sistema. Esto con el fin de poder proporcionar información de los costos de las habitaciones a las personas que quieran realizar reservaciones.

3. **Listar reservaciones para hoy:** La recepcionista podrá listar las reservaciones en base al día para poder saber qué reservaciones se encuentran agendadas para dicho día.

Módulo de Empleado:

Todos los tipos de empleados que no tengan un módulo propio serán redirigidos a este módulo. En este módulo se mostrarán las tareas que los administradores hayan ingresado al sistema. Este módulo cuenta con las siguientes funcionalidades:

1. **Listar tareas pendientes:** Al empleado se le listará todas las tareas que se encuentren en estado **"pendiente"** y un par de botones con los que podrá ver detalles sobre la tarea y asignarse la misma.
2. **Listar tareas asignadas:** Al empleado se le listará todas las tareas que éste haya decidido asignarse. De igual manera se le mostrarán un par de botones para visualizar los detalles de la tarea así como un botón para poder marcarla como **"finalizada".**

Diagrama Entidad Relacion Base de Datos:

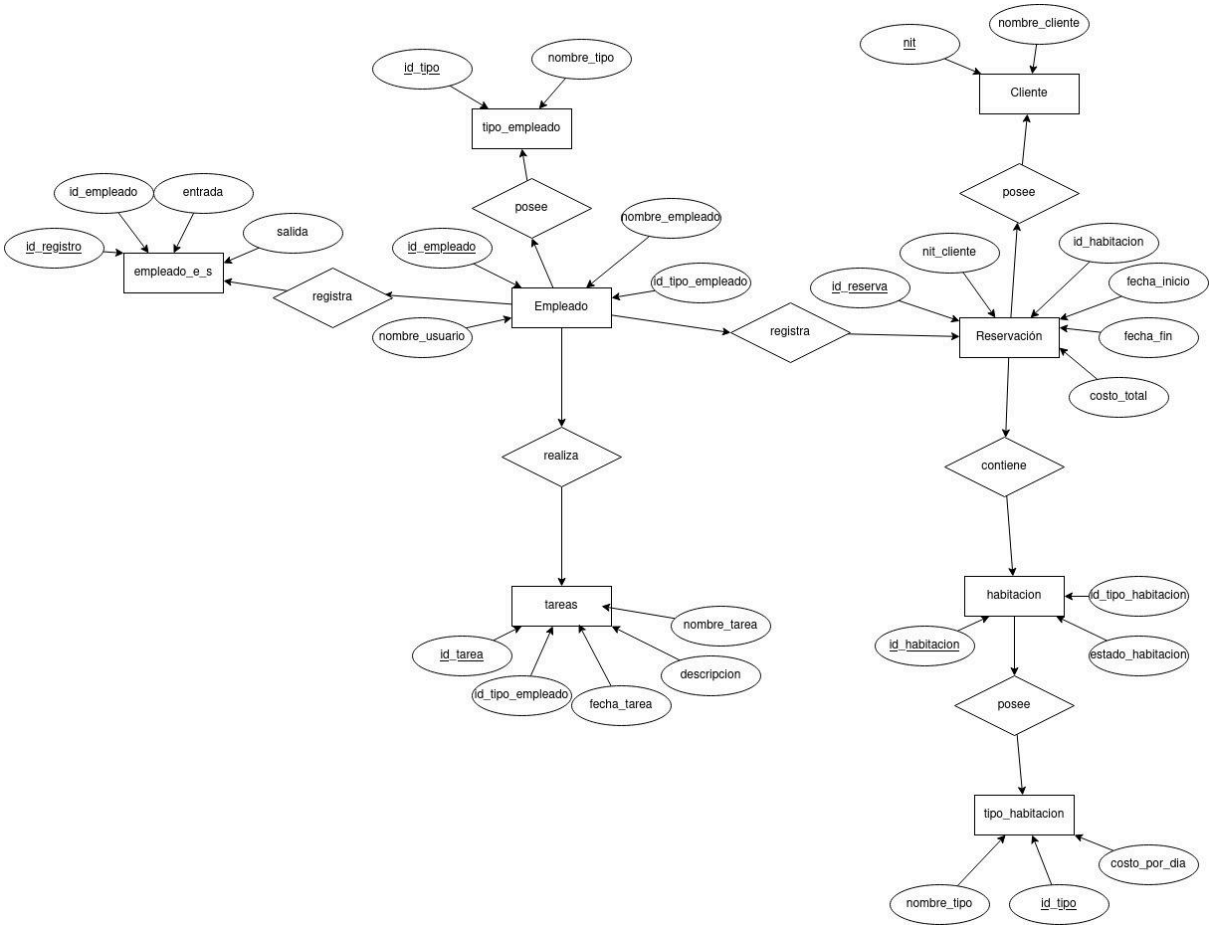
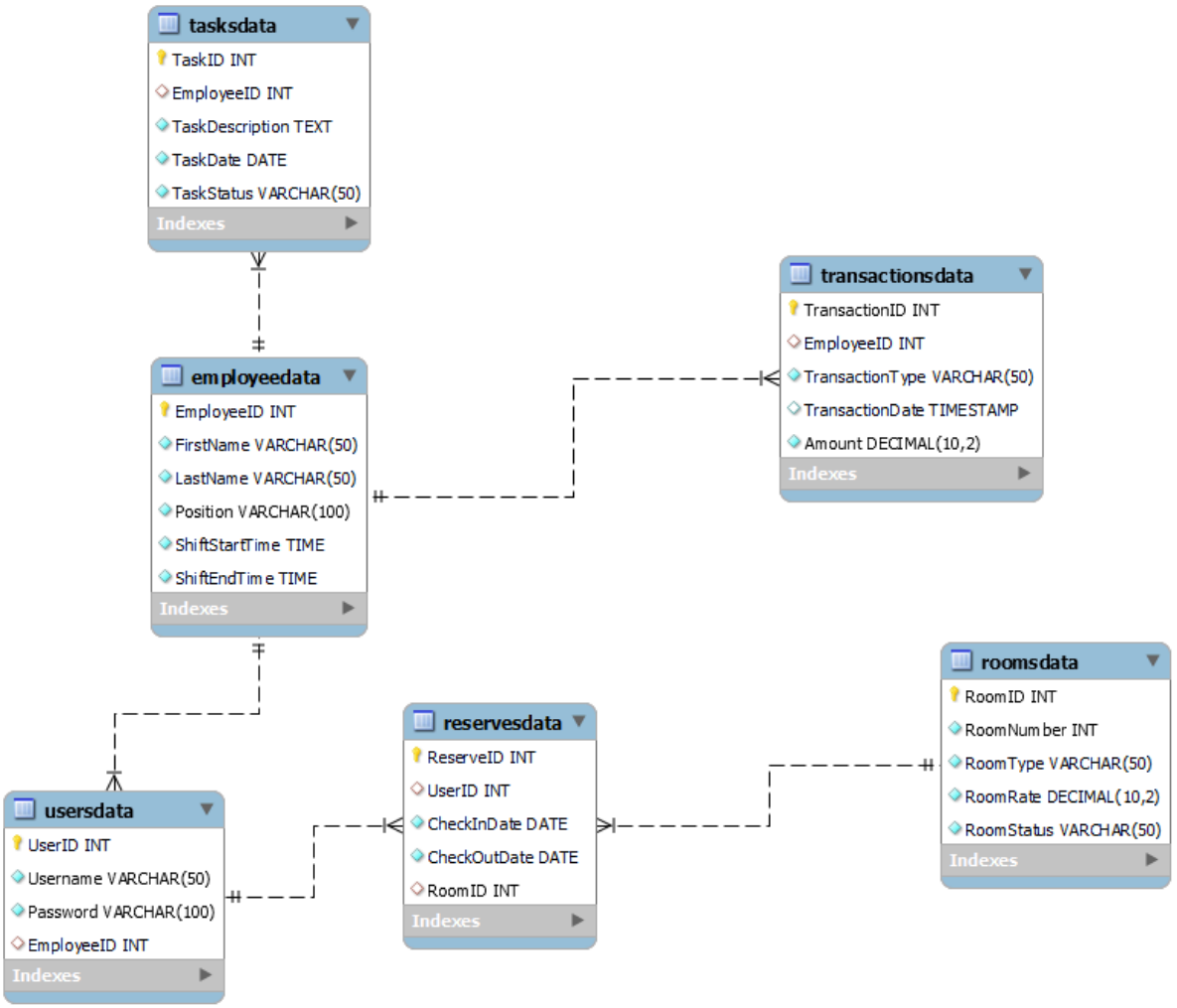


Diagrama Relacional Base de Datos:



Casos de Uso:

Caso de uso principal

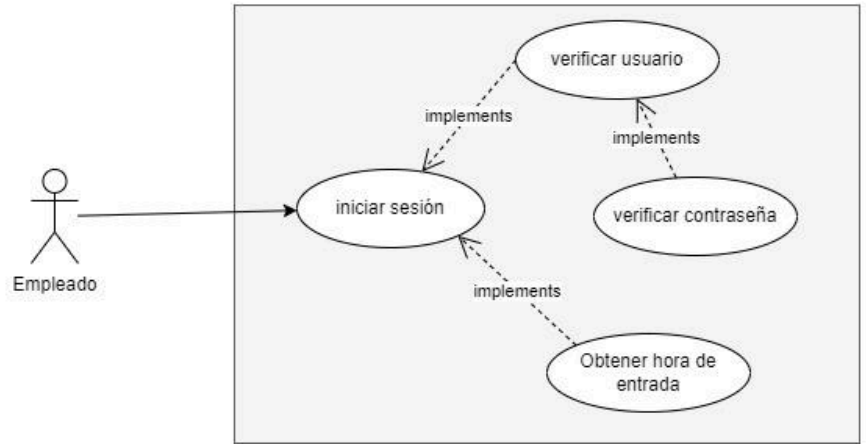
Caso de uso	Iniciar sesión
Descripción	El empleado ingresa sus credenciales para poder ingresar a la aplicación
Entidades	Empleado

Casos de uso incluidos

Caso de uso	Verificar usuario
Descripción	El sistema debe verificar que el usuario ingresado exista en la base de datos.
Entidades	

Caso de uso	Verificar contraseña
Descripción	Después de haber confirmado la existencia del usuario se debe verificar que la contraseña ingresada sea la misma.
Entidades	

Caso de uso	Obtener hora de entrada
Descripción	El sistema debe almacenar la hora a la que el usuario ingresó por primera vez al sistema durante el día
Entidades	

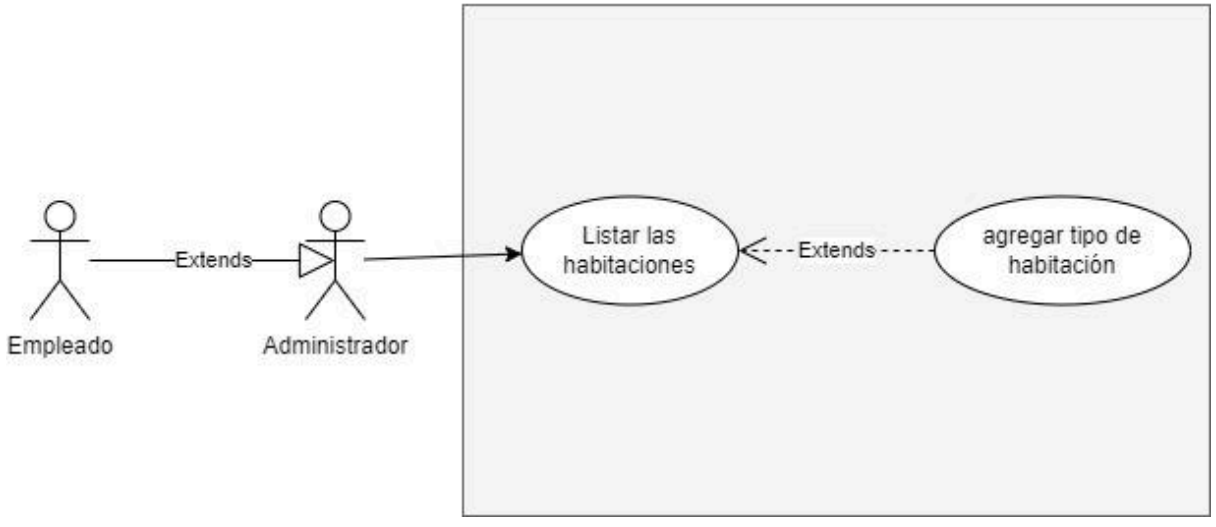


Caso de uso principal

Caso de uso	Listar las habitaciones
Descripción	El administrador puede listar todas las habitaciones agregadas al sistema
Entidades	Administrador

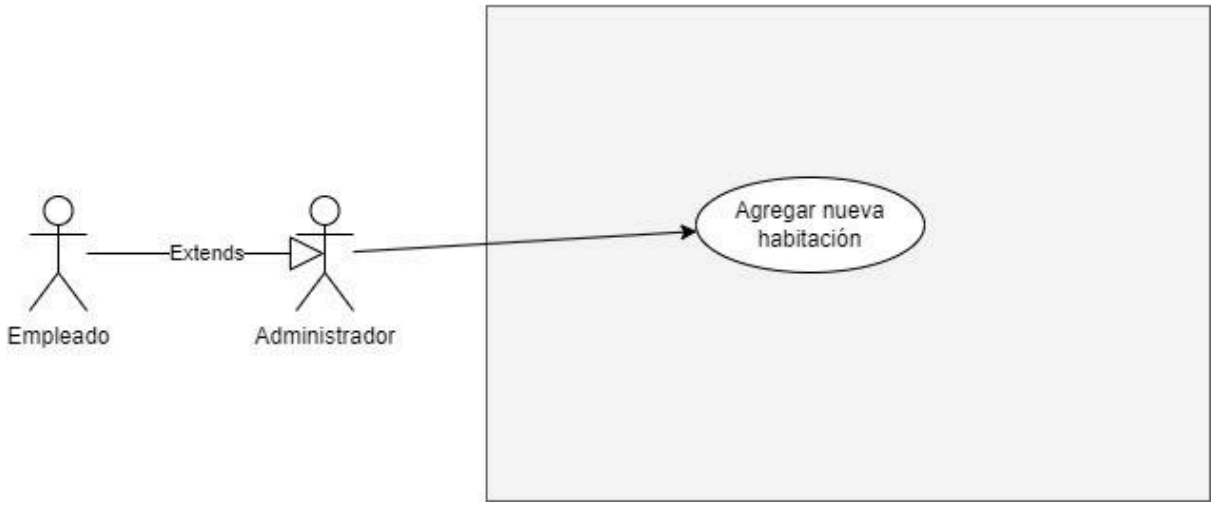
Caso de uso extendido

Caso de uso	Agregar tipo de habitación
Descripción	El administrador puede agregar un filtro en base al tipo de habitación que desea obtener
Entidades	Administrador



Caso de uso principal

Caso de uso	Agregar nueva habitación
Descripción	El administrador agrega habitaciones al sistema
Entidades	Administrador

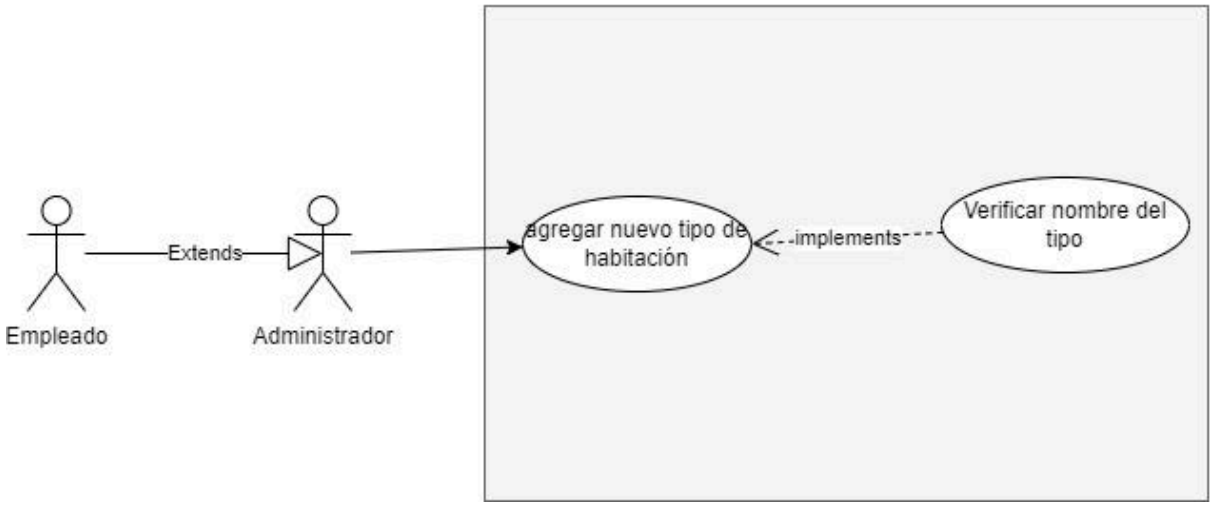


Caso de uso principal

Caso de uso	Agregar nuevo tipo de habitación
Descripción	El administrador puede agregar nuevos tipos de habitación al sistema
Entidades	Administrador

Casos de uso incluidos

Caso de uso	Verificar nombre del tipo
Descripción	El sistema debe confirmar que el nombre del nuevo tipo de habitación sea único y no se encuentre ya ingresado alsistema
Entidades	



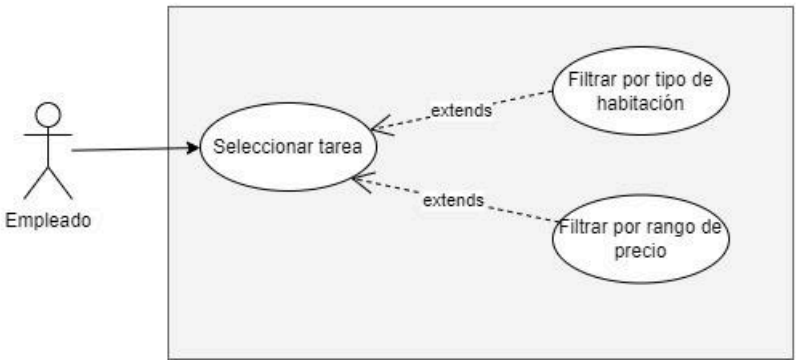
Caso de uso principal

Caso de uso	Listar Habitaciones
Descripción	El usuario es capaz de ver todas las habitaciones existentes en el sistema
Entidades	Usuario

Casos de uso extendidos

Caso de uso	filtrar por tipo de habitación
Descripción	El usuario es capaz de filtrar las habitaciones por tipo de habitación
Entidades	Usuario

Caso de uso	filtrar por rango de precio
Descripción	El usuario es capaz de filtrar las habitaciones por rango de precio
Entidades	Usuario



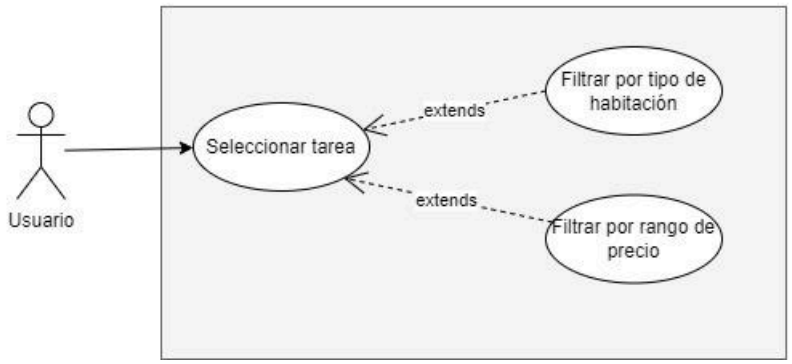
Caso de uso principal

Caso de uso	Ver información de reservación
Descripción	El usuario es capaz de ver la información sobre la reservación hecha en el sistema en base a algún código
Entidades	Usuario

Casos de uso extendidos

Caso de uso	filtrar por tipo de habitación
Descripción	El usuario es capaz de filtrar las habitaciones por tipo de habitación
Entidades	

Caso de uso	filtrar por rango de precio
Descripción	El usuario es capaz de filtrar las habitaciones por rango de precio
Entidades	

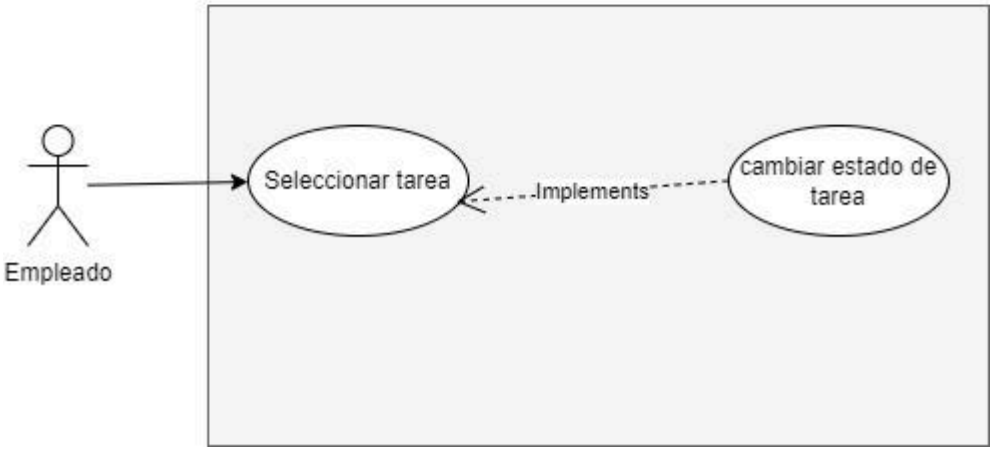


Caso de uso principal

Caso de uso	Finalizar tarea
Descripción	Una vez completada la tarea el empleado marca su tarea asignada como FINALIZADA
Entidades	Empleado

Caso de uso implementado

Caso de uso	Cambiar estado de tarea
Descripción	El sistema cambia el estado de la tarea de ASIGNADA a FINALIZADA.
Entidades	

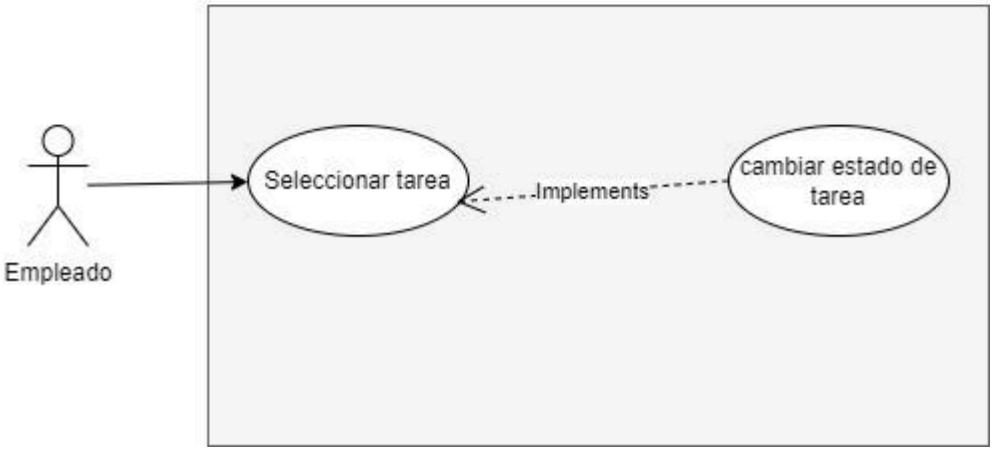


Caso de uso principal

Caso de uso	Seleccionar tarea
Descripción	Los empleados listan las tareas pendientes por realizar
Entidades	Empleado

Caso de uso implementado

Caso de uso	Cambiar estado de tarea
Descripción	El sistema cambia el estado de la tarea de 'Pendiente' a 'ASIGNADA' también agrega el código del empleado que realizará la tarea
Entidades	

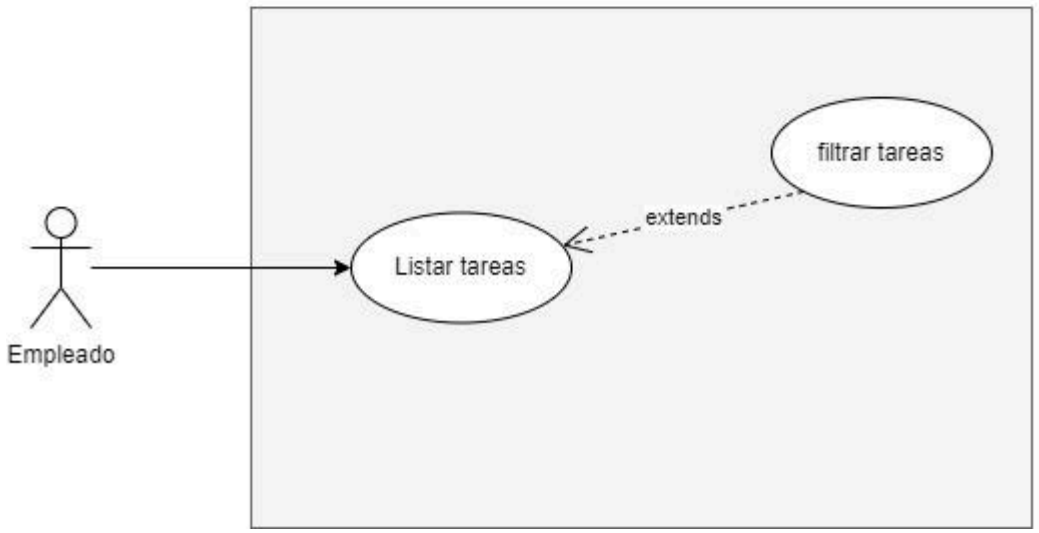


Caso de uso principal

Caso de uso	Listar tareas
Descripción	Los empleados listan las tareas que el administrador ha asignado durante el día
Entidades	empleado

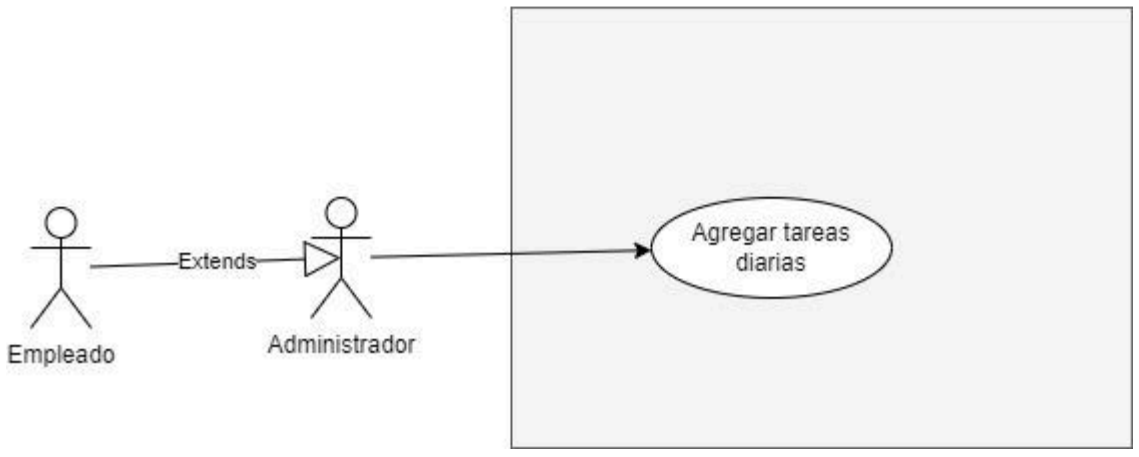
Caso de uso extendido

Caso de uso	Filtrar tareas
Descripción	El empleado es capaz de filtrar la tarea, puede ser por: PENDIENTE o ASIGNADA
Entidades	empleado



Caso de uso principal

Caso de uso	Agregar tareas diarias
Descripción	El administrador puede agregar tareas que serán realizadas por los demás empleados
Entidades	Administrador

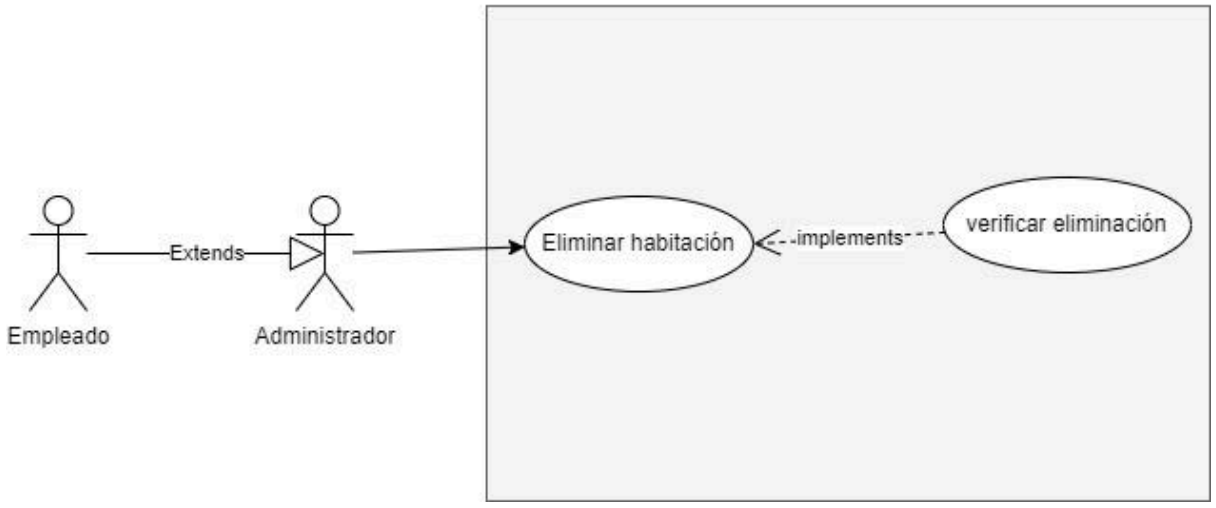


Caso de uso principal

Caso de uso	Eliminar habitación
Descripción	El administrador puede eliminar una habitación registrada anteriormente
Entidades	Administrador

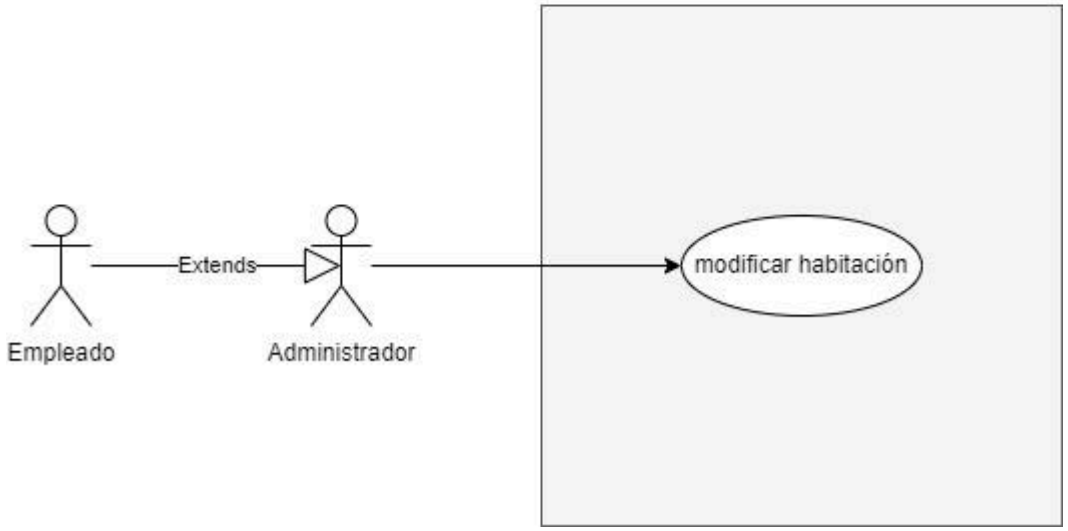
Casos de uso incluidos

Caso de uso	Verificar eliminación
Descripción	El sistema debe confirmar que la habitación a eliminar del sistema no se encuentre ocupada con alguna reserva vigente.
Entidades	



Caso de uso principal

Caso de uso	modificar habitación
Descripción	El administrador puede modificar la información sobre alguna habitación ingresada anteriormente
Entidades	Administrador

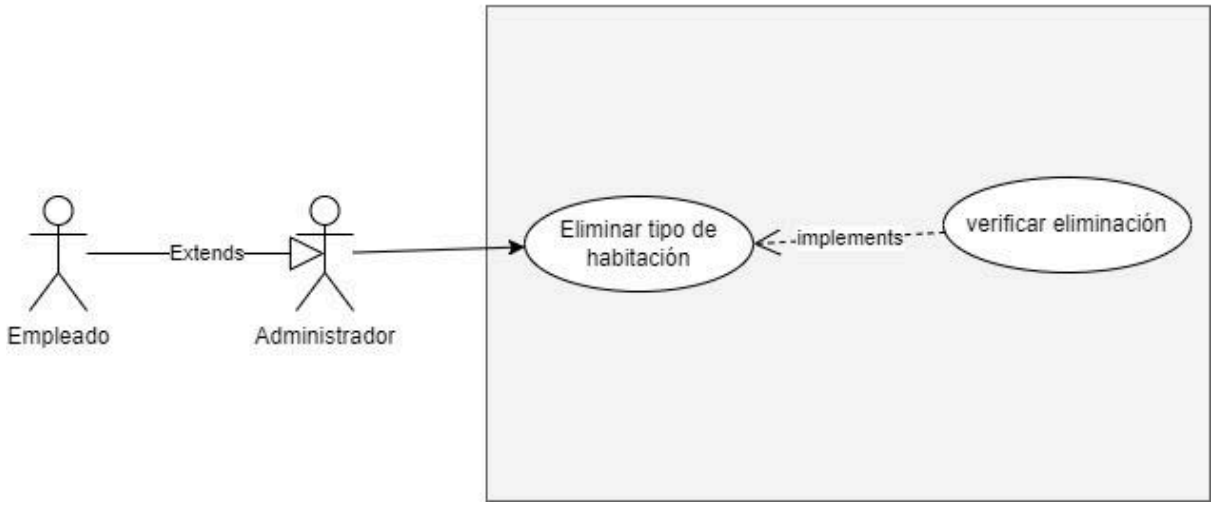


Caso de uso principal

Caso de uso	Eliminar tipo de habitación
Descripción	El administrador puede Eliminar algún tipo de habitación que ya se encuentre registrado en el sistema.
Entidades	Administrador

Casos de uso incluidos

Caso de uso	Verificar eliminación
Descripción	El sistema debe verificar que la eliminación sea posible. Para esto debe corroborar que este tipo de habitación no se encuentre asignado a una habitación registrada.
Entidades	

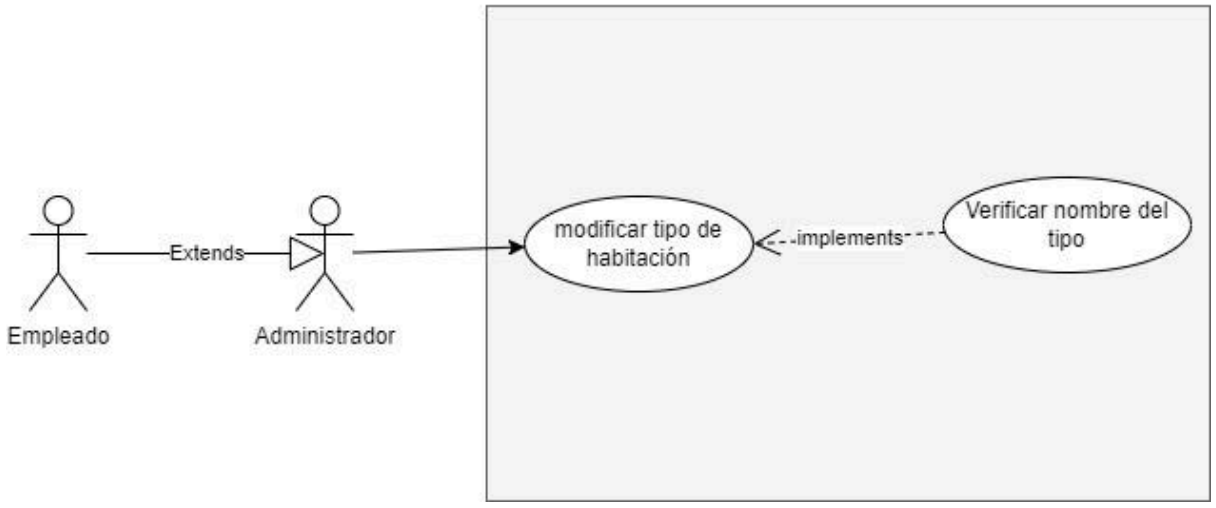


Caso de uso principal

Caso de uso	Modificar tipo de habitación
Descripción	El administrador puede modificar algún tipo de habitación que ya se encuentre registrado en el sistema.
Entidades	Administrador

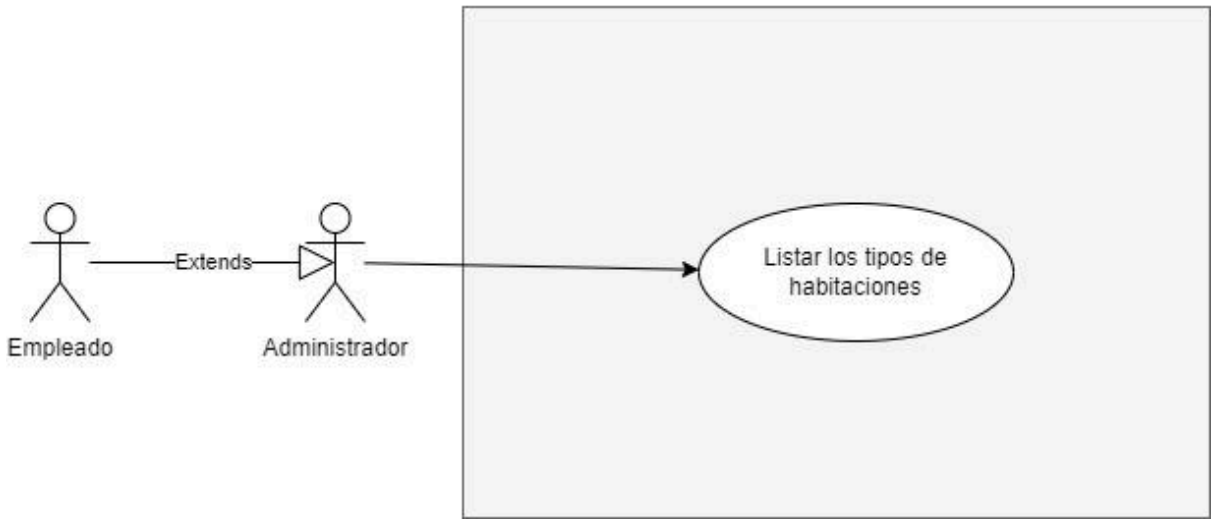
Casos de uso incluidos

Caso de uso	Verificar nombre del tipo
Descripción	El sistema debe confirmar que el nombre del nuevo tipo de habitación sea único y no se encuentre ya ingresado alsistema
Entidades	



Caso de uso principal

Caso de uso	Listar los tipos de habitaciones
Descripción	El administrador puede ver el listado de todos los tipos de habitaciones ingresadas anteriormente al sistema
Entidades	Administrador

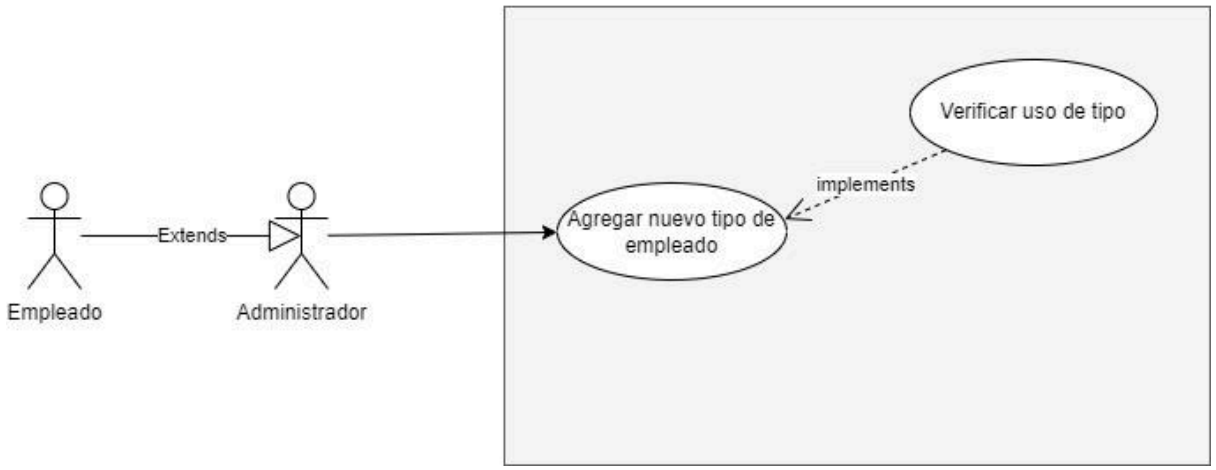


Caso de uso principal

Caso de uso	Eliminar tipo de empleado
Descripción	El administrador puede directamente eliminar un tipo de empleado en caso de que este ya no se necesite en la aplicación
Entidades	Administrador

Casos de uso incluidos

Caso de uso	Verificar uso de tipo
Descripción	El sistema debe confirmar que ningún empleado registrado y activo se encuentre usando ese tipo de empleado actualmente
Entidades	

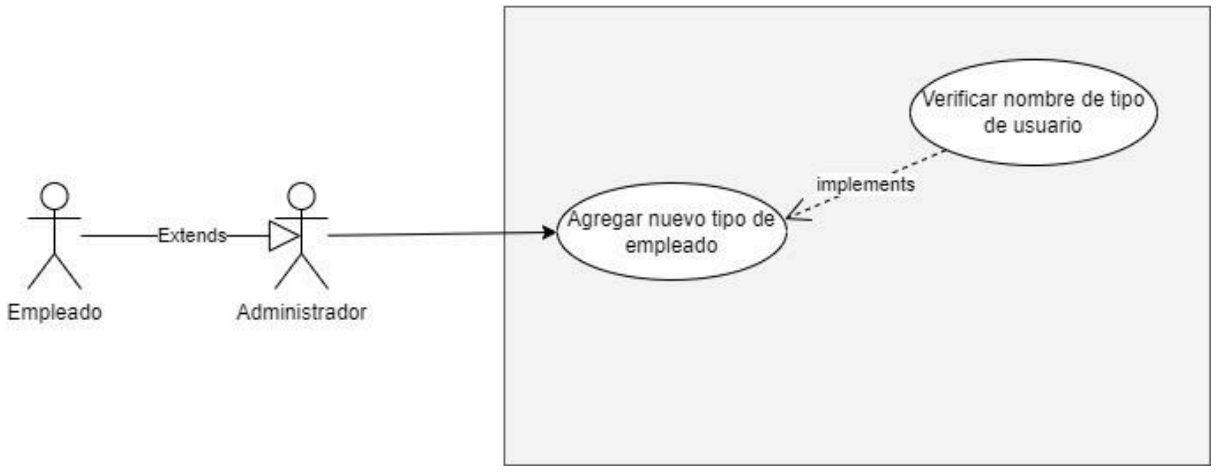


Caso de uso principal

Caso de uso	Actualizar tipo de empleado
Descripción	El administrador modifica el nombre de un tipo de empleado registrado anteriormente en caso de que haya existido un error al momento de registrarlo
Entidades	Administrador

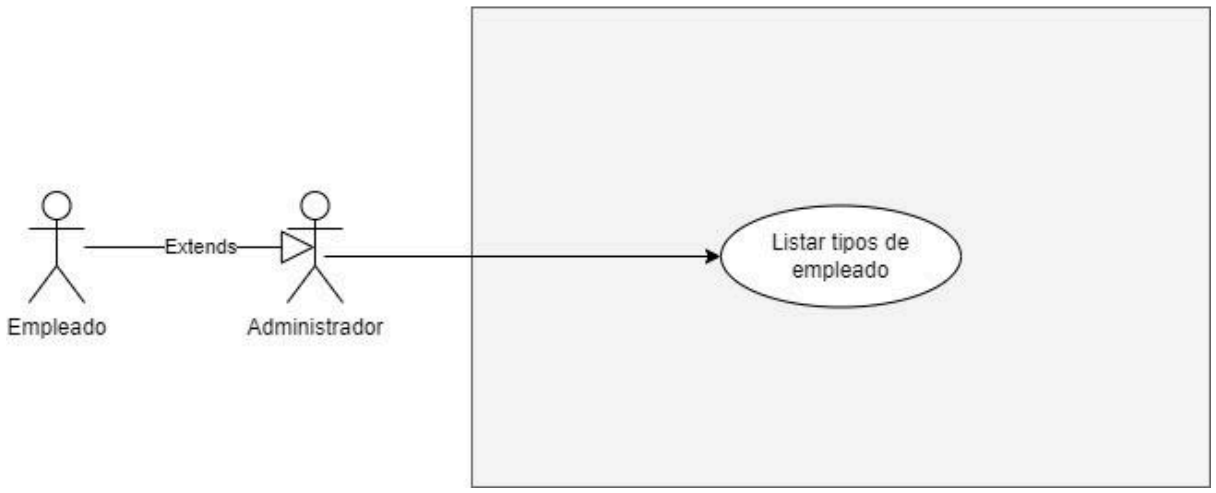
Casos de uso incluidos

Caso de uso	Verificar nombre de tipo de usuario
Descripción	El sistema debe verificar que la actualización de nombre se pueda realizar ya que este es un campo único
Entidades	



Caso de uso principal

Caso de uso	Listar tipos de empleado
Descripción	El administrador puede obtener una lista de todos los tipos de empleado registrados en el sistema
Entidades	Administrador

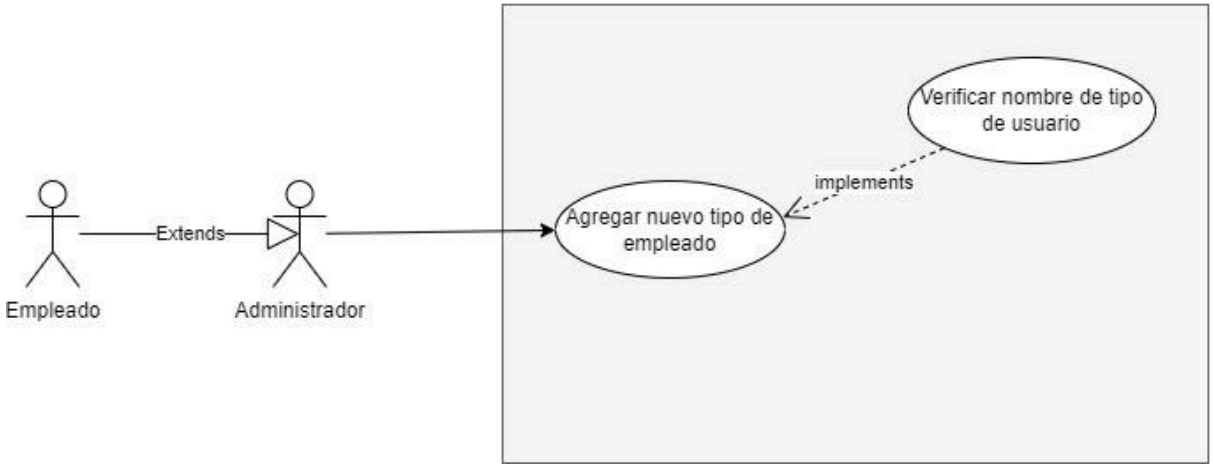


Caso de uso principal

Caso de uso	Agregar nuevo tipo de empleado
Descripción	El administrador agrega un nuevo tipo de empleado al sistema.
Entidades	Administrador

Casos de uso incluidos

Caso de uso	Verificar nombre de tipo de usuario
Descripción	El sistema debe confirmar que el nombre del nuevo tipo sea único para evitar ambigüedades.
Entidades	



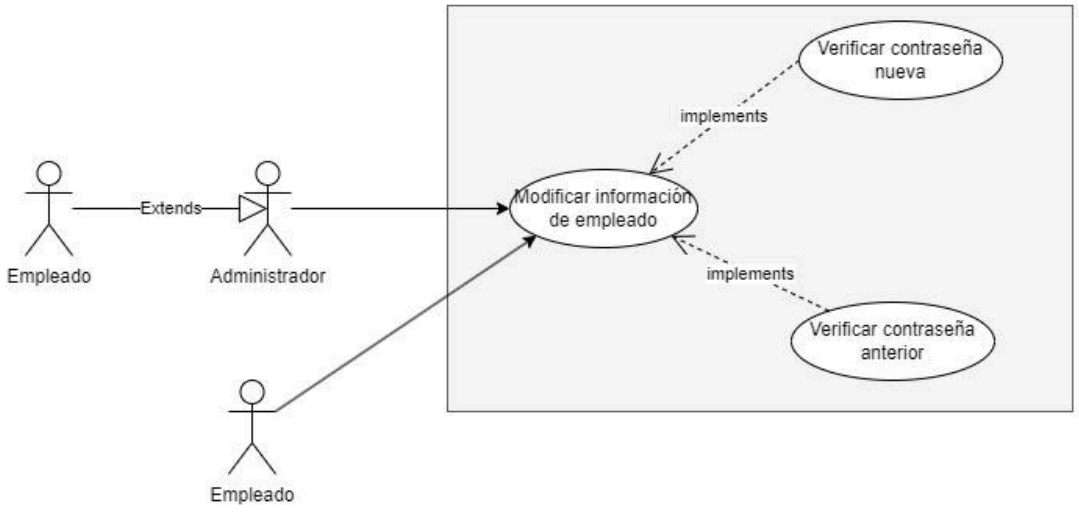
Caso de uso principal

Caso de uso	Modificar información de empleado
Descripción	El administrador puede modificar cosas como el nombre del empleado o la contraseña. Si se desea modificar la contraseña el empleado debe ingresar la contraseña anterior seguido de la nueva contraseña
Entidades	Administrador, Empleado

Casos de uso incluidos

Caso de uso	Verificar contraseña nueva
Descripción	El sistema debe confirmar que la información ingresada en los campos "contraseña" y "confirmar contraseña" coincidan.
Entidades	

Caso de uso	Verificar contraseña anterior
Descripción	El sistema debe confirmar que la contraseña anterior sea la correcta para poder hacer el cambio a la nueva.
Entidades	

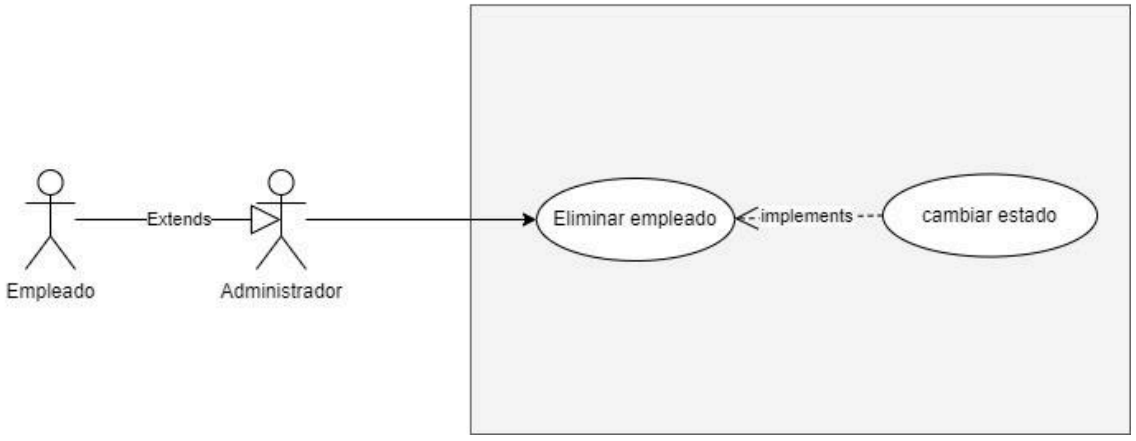


Caso de uso principal

Caso de uso	Eliminar empleado
Descripción	El administrador elimina a un empleado.
Entidades	Administrador

Casos de uso incluidos

Caso de uso	Cambiar estado
Descripción	El sistema cambia el estado del emplado a "inactivo"
Entidades	



Caso de uso principal

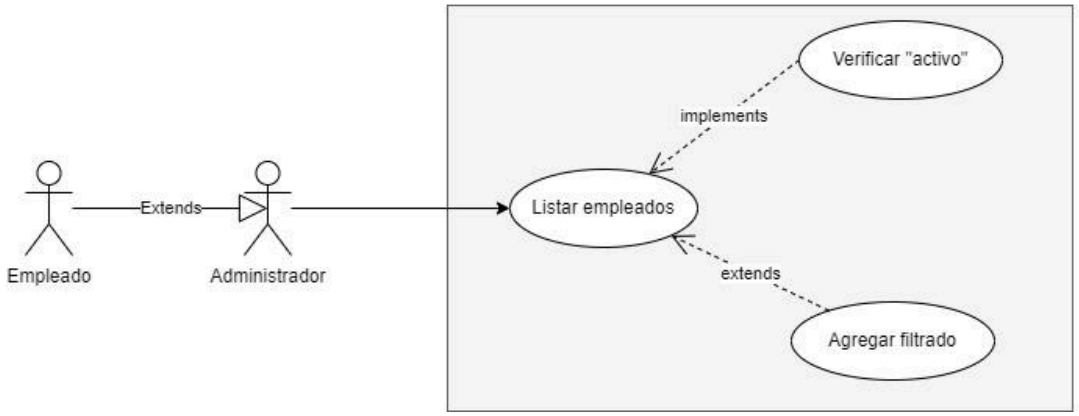
Caso de uso	Listar empleados
Descripción	El administrador lista todos los usuarios registrados en el sistema
Entidades	Administrador

Casos de uso incluidos

Caso de uso	Verificar "activo"
Descripción	El sistema obtiene todos los empleados registrados en el sistema que se encuentren "activos"
Entidades	

Casos de uso incluidos

Caso de uso	Agregar filtro
Descripción	El administrador puede filtrar los usuarios en base al tipo de empleado.
Entidades	



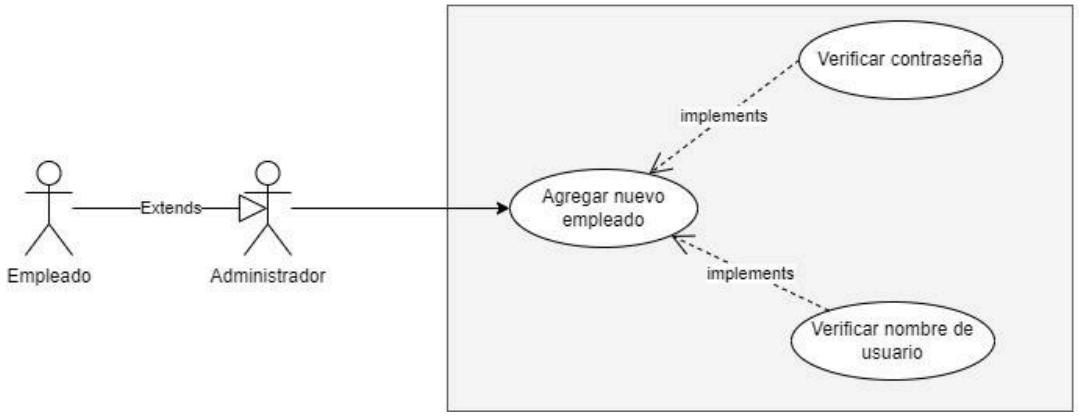
Caso de uso principal

Caso de uso	Agregar nuevo empleado
Descripción	El administrador registra un nuevo empleado al sistema ingresando toda la información necesaria como: nombre del empleado, nombre de usuario, contraseña, tipo de empleado
Entidades	Administrador

Casos de uso incluidos

Caso de uso	Verificar contraseña
Descripción	El sistema debe confirmar que la información ingresada en los campos "contraseña" y "confirmar contraseña" coincidan.
Entidades	

Caso de uso	Verificar nombre de usuario
Descripción	El sistema debe confirmar que el usuario ingresado se encuentre disponible ya que este debe ser único para evitar ambiguedades.
Entidades	



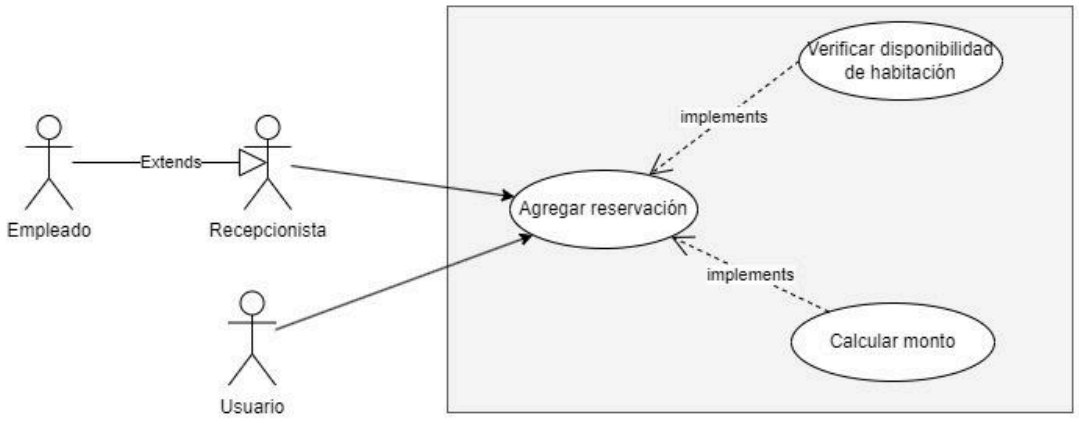
Caso de uso principal

Caso de uso	Agregar reservación
Descripción	La recepcionista o usuario (en caso de reservación en linea) registra una nueva reservación al sistema
Entidades	Recepcionista, usuario

Casos de uso incluidos

Caso de uso	Verificar disponibilidad de habitación
Descripción	El sistema debe verificar la disponibilidad del tipo de habitación que el empleado desee
Entidades	

Caso de uso	Calcular monto
Descripción	El sistema debe calcular el monto total en base a la habitación seleccionada y el número de días que el huésped reservará la habitación.
Entidades	



Diagramas Completos:
https://app.diagrams.net/#G10wYx6_heI6sIrfudJwC5FpRNfa6rypKB

Diagrama de Despliegue:

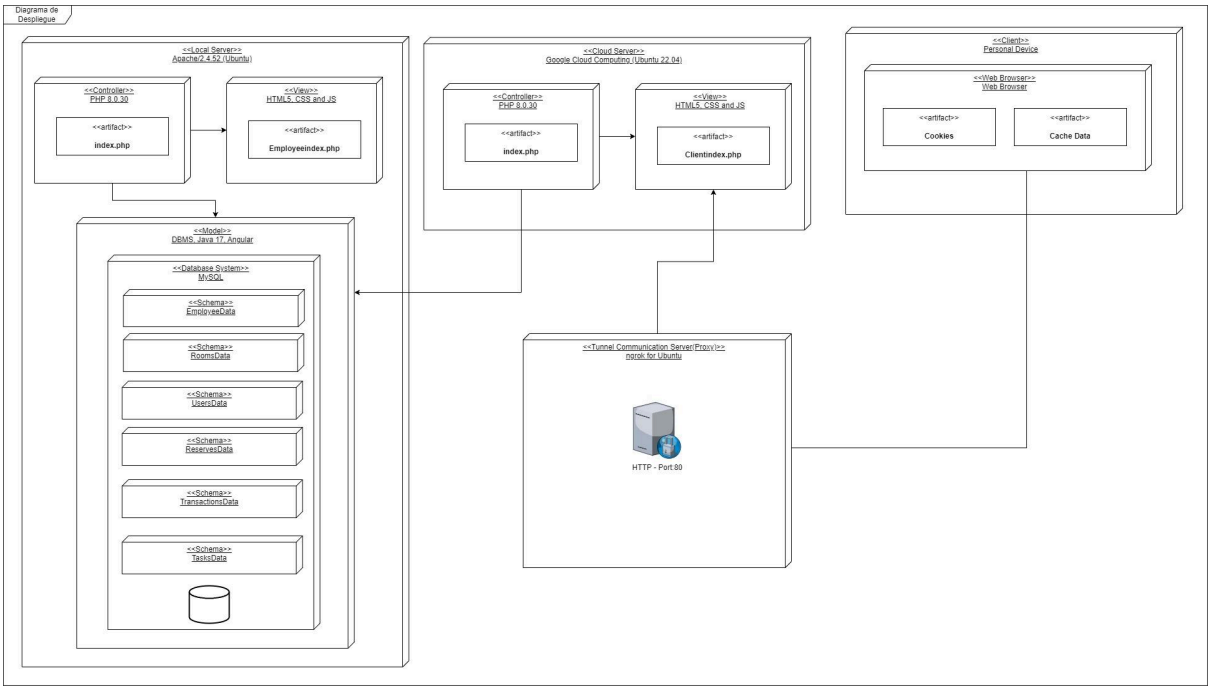


Diagrama:
https://app.diagrams.net/#G1HiHVAmjFF6Pd_VKOhQJPoAv4crxGk-UD

Diagrama de Componentes (UsersWeb):

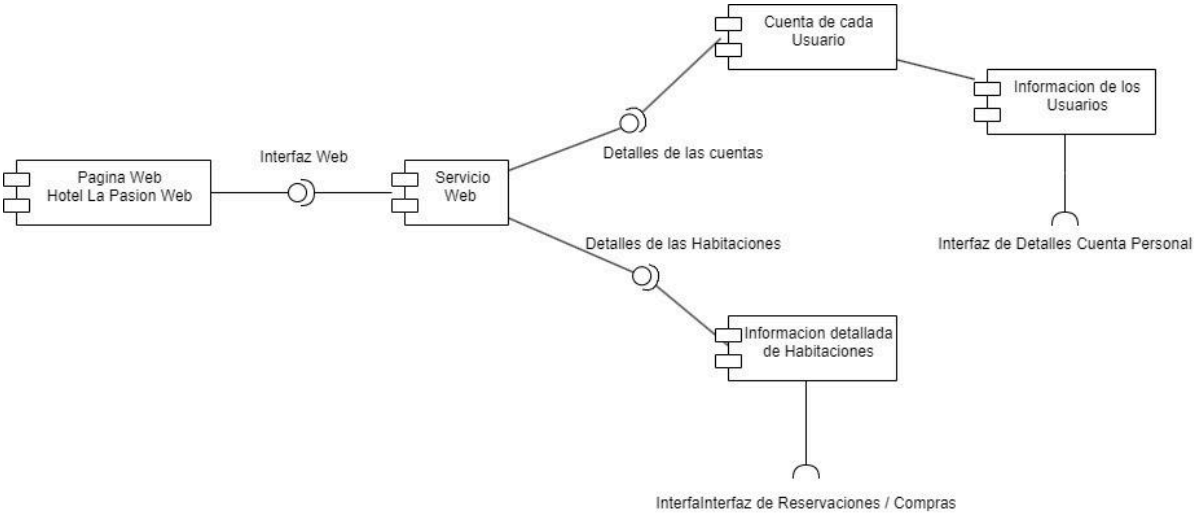


Diagrama de Componentes (AdminWeb):

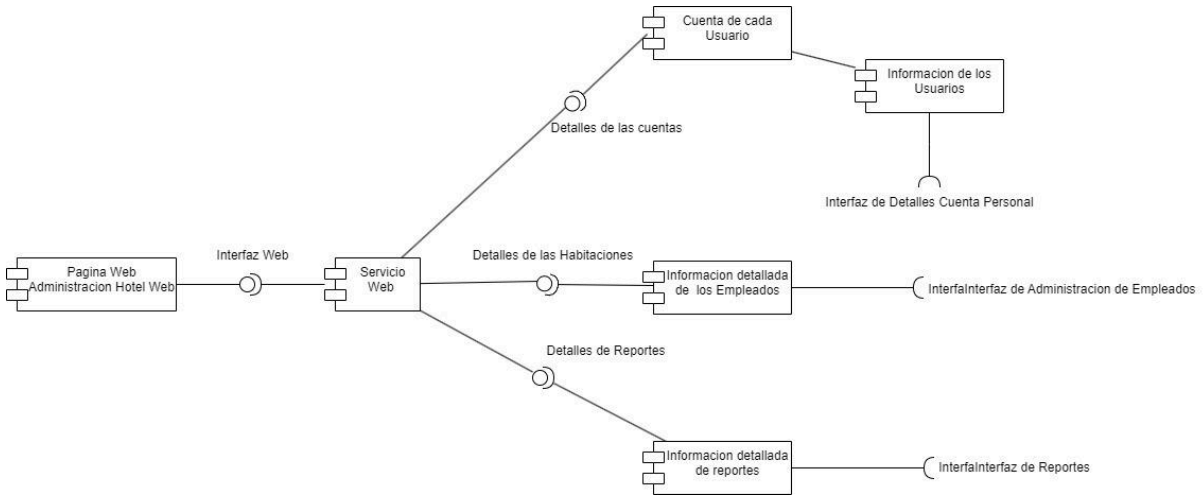
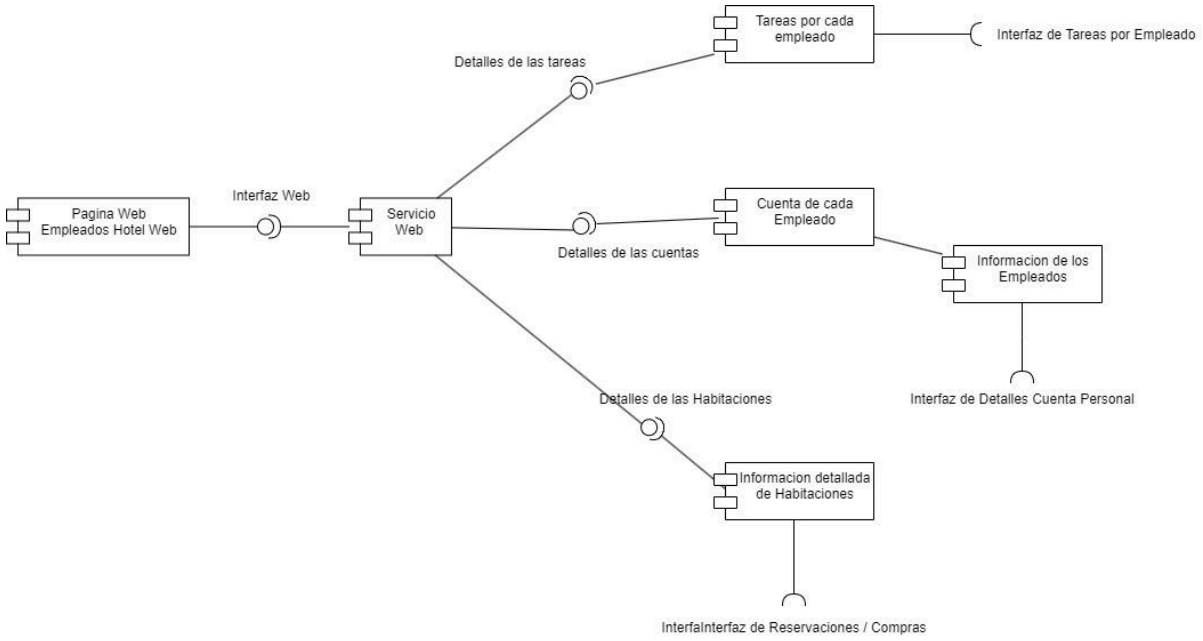


Diagrama de Componentes (EmployeeWeb):



Diagramas:
https://app.diagrams.net/#G1HiHVAmjFF6Pd_VKOhQJPoAv4crxGk-UD#%7B%22pageId%22%3A%22RzWk7jUkcD9xkoclXsJE%22%7D
https://app.diagrams.net/#G1HiHVAmjFF6Pd_VKOhQJPoAv4crxGk-UD#%7B%22pageId%22%3A%22_wnwLJzLY11zwsl_h76T%22%7D
https://app.diagrams.net/#G1HiHVAmjFF6Pd_VKOhQJPoAv4crxGk-UD#%7B%22pageId%22%3A%22uOmQKgz_6JZfp2VdKNX9%22%7D

Script Base de Datos:

```sql
CREATE DATABASE IF NOT EXISTS hoteru;
USE hoteru;

-- Tabla que almacena los diferentes tipos de empleados.
CREATE TABLE IF NOT EXISTS employee_types (
   id INT AUTO_INCREMENT PRIMARY KEY,
   type VARCHAR(255) NOT NULL
);

-- Tabla que almacena la información de cada empleado, incluyendo a qué
tipo pertenece.
CREATE TABLE IF NOT EXISTS employees (
   id INT AUTO_INCREMENT PRIMARY KEY,
   username VARCHAR(255) NOT NULL,
   password VARCHAR(255) NOT NULL,
   name VARCHAR(255) NOT NULL,
   employee_type_id INT,
   FOREIGN KEY (employee_type_id) REFERENCES employee_types (id)
);

-- Tabla que define los tipos de habitaciones disponibles y su costo por
día.
CREATE TABLE IF NOT EXISTS room_types (
   id INT AUTO_INCREMENT PRIMARY KEY,
   type VARCHAR(255) NOT NULL,
   cost_per_day DECIMAL(10,2) NOT NULL
);

-- Tabla que almacena las habitaciones disponibles, su estado y a qué tipo
pertenecen.
CREATE TABLE IF NOT EXISTS rooms (
   id INT AUTO_INCREMENT PRIMARY KEY,
   state VARCHAR(50) NOT NULL,
   room_type_id INT,
   FOREIGN KEY (room_type_id) REFERENCES room_types(id)
   );

-- Tabla que registra las reservaciones hechas por los clientes, habitación
reservada,
--  empleado  que  realizó  la  reserva  y  fechas  de  inicio  y  fin  de  la
reservación.
CREATE TABLE IF NOT EXISTS reservations (
   id INT AUTO_INCREMENT PRIMARY KEY,
   client_nit INT,
   room_id INT,
   date DATE,
   init_date DATE NOT NULL,
   end_date DATE NOT NULL,
   total DECIMAL(10,2),
   FOREIGN KEY (room_id) REFERENCES rooms(id)
```

```sql
   );

-- tabla para las tareas de los empleados
CREATE TABLE IF NOT EXISTS tasks(
   id INT AUTO_INCREMENT PRIMARY KEY,
   name VARCHAR(255) NOT NULL,
   task_description TEXT,
   date DATE NOT NULL,
   employee_id INT,
   state VARCHAR(255)
);
```