

Proyecto: Sistema de Gestión de Reservas para Espacios de Trabajo

Descripción

Una empresa con múltiples oficinas desea implementar un sistema donde los empleados puedan reservar espacios de trabajo (escritorios, salas de reuniones, estaciones de coworking, etc.). Se debe desarrollar un sistema web con Django y PostgreSQL para gestionar estas reservas de manera eficiente.

Requisitos Funcionales

1. Autenticación de Usuarios:

- Inicio de sesión y registro de usuarios con Django Auth.
- Roles: Administrador y Empleado.

2. Gestión de Espacios de Trabajo:

- CRUD para los espacios de trabajo (solo accesible para administradores).
- Cada espacio tiene: nombre, tipo, capacidad, disponibilidad, ubicación.

3. Reservas:

- Un empleado puede reservar un espacio para una fecha y hora específicas.
- No se permite reservar un espacio que ya esté ocupado en la misma franja horaria.
- Cancelación de reservas antes de su inicio.

4. Dashboard:

- Vista para empleados con sus reservas activas y pasadas.
- Vista para administradores con un resumen de la ocupación y estadísticas.

5. Notificaciones:

- Envío de correo de confirmación al realizar una reserva.
- Notificación de recordatorio antes de la reserva.







Requisitos Técnicos






- **Django:** Framework principal.
- **PostgreSQL:** Base de datos.
- **Django Rest Framework (DRF):** Para exponer una API (opcional).
- **Celery + Redis:** Para tareas en segundo plano (envío de correos).
- **Template Engine de Django:** Para las vistas.
- **Bootstrap o Tailwind:** Para la interfaz.








Desafío Extra (Opcional)





















- Implementar autenticación con Google OAuth.
- Usar WebSockets para actualizar el estado de las reservas en tiempo real.

• **Fases del Proyecto: Paso a Paso**

- Este será nuestro **plan profesional** de desarrollo, asegurando escalabilidad, mantenimiento y calidad.
-  **Fase 1: Configuración Inicial**
-  Crear el proyecto Django y configurar PostgreSQL.
-  Configurar `AUTH_USER_MODEL` con `AbstractUser` para personalizar usuarios.
-  Implementar autenticación con **Django Auth (login, registro y logout)**.
-  Definir los roles (`ADMIN` y `EMPLOYEE`).
-  **Entrega esperada:** Proyecto base con autenticación funcional.

-
-  **Fase 2: Gestión de Espacios de Trabajo**
 -  Crear el modelo `Workspace` (ID, nombre, tipo, capacidad, disponibilidad, ubicación).
 -  Implementar CRUD de espacios (solo accesible para administradores).
 -  Crear vistas y templates para visualizar los espacios.
 -  **Entrega esperada:** Los administradores pueden gestionar los espacios de trabajo.

-
-  **Fase 3: Sistema de Reservas**
 -  Crear el modelo `Reservation` (usuario, espacio, fecha inicio/fin, estado).
 -  Implementar la lógica de **validación de reservas solapadas**.
 -  Permitir que empleados reserven espacios disponibles.
 -  Permitir que los empleados **cancelen reservas antes de su inicio**.
 -  Implementar vistas para listar y administrar reservas.
 -  **Entrega esperada:** Sistema funcional de reservas con validaciones.
-

-  **Fase 4: Notificaciones Automáticas**
-  Crear el modelo `Notification` (usuario, mensaje, fecha de envío, estado).
-  Integrar **Celery + Redis** para enviar notificaciones por correo.
-  Enviar **confirmación de reserva** al usuario.
-  Enviar **recordatorios antes de la reserva**.
-  **Entrega esperada:** Notificaciones automáticas en funcionamiento.
- ---
-  **Fase 5: Dashboard y Estadísticas**
-  Implementar un **dashboard para empleados** con sus reservas activas y pasadas.
-  Implementar un **dashboard para administradores** con ocupación y estadísticas.
-  Usar gráficos para visualizar la ocupación de espacios.
-  **Entrega esperada:** Dashboard funcional con estadísticas en tiempo real.
- ---
-  **Fase 6: Exponer una API con Django Rest Framework (Opcional)**
-  Crear endpoints REST para gestionar usuarios, espacios y reservas.
-  Proteger la API con **tokens o JWT**.
-  Permitir que la app pueda integrarse con otras plataformas.
-  **Entrega esperada:** API funcional con autenticación y seguridad.
- ---
-  **Metodología de Trabajo**
-  **Versionamiento con Git:** Crearemos ramas (`feature/reservas`, `feature/notificaciones`, etc.).
-  **Deploy en un servidor:** Usaremos **Docker + PostgreSQL** en producción.
-  **Pruebas unitarias:** Implementaremos tests con `pytest` para garantizar calidad.