Especificaciones de requerimientos
Crear una plataforma web para gestionar las reservas de eventos internos de Grupo Coppel, permitiendo a los empleados solicitar reservas de salas de reuniones, auditorios y otros espacios corporativos, así como a los administradores gestionar y aprobar estas solicitudes.

Requerimientos Funcionales Portal de Reservas (Empleado)

Pantalla principal "RESERVA DE ESPACIOS"

- Opción para registrar una nueva solicitud de reserva.
- Opción para consultar las reservas existentes y el estado de la solicitud.
- Visualización de la disponibilidad de los espacios en tiempo real (calendario de reservas).

PASO 1: Selección del Espacio

- **Espacios disponibles:**
  - **Sala de reuniones pequeña**
  - **Sala de reuniones grande**
  - **Auditorio**
  - **Área de descanso**
- **Capacidad: Al seleccionar un espacio, se muestra la capacidad máxima del lugar.**

PASO 2: Fecha y Hora

- **Calendario para seleccionar la fecha y hora de la reserva.**
- **Validar disponibilidad en tiempo real (sin solapamientos).**

PASO 3: Detalles de la Reserva

- **Nombre del solicitante**
- **Departamento**
- **Duración estimada**

- **Descripción del evento: Breve resumen del propósito de la reserva.**

PASO 4: Confirmación

- **Validación de los datos y generación de un folio único de reserva.**
- **Mostrar mensaje de "Reserva exitosa" junto con el número de folio.**

Requerimientos Funcionales Portal de Administración (Administrador de Reservas)

Login de Administrador

- **Usuarios autorizados**: Administrador de reservas y jefes de departamento.
- **Autenticación**: Uso de **JWT** para la gestión segura de sesiones.

Gestión de Reservas

- Visualización de todas las reservas solicitadas, con los siguientes datos:
  - **Folio de reserva**, **Espacio reservado**, **Fecha y hora**, **Estado** (Pendiente, Aprobada, Rechazada).
- **Acción de aprobación/rechazo**: El administrador puede aprobar o rechazar solicitudes de reserva.
- **Acción de modificación**: El administrador puede modificar las reservas (cambiar la fecha, el espacio, etc.).
- **Historial de cambios**: Ver todos los cambios realizados en una reserva, junto con la fecha y el usuario que los realizó.

Casos de Uso

#	Nombre	Descripción
CU1	Registrar Nueva Reserva	El empleado ingresa los datos necesarios (espacio, fecha, detalles) y recibe un folio único. Validación de datos y generación del folio numérico.
CU2	Consulta de Reserva	El empleado ingresa el número de folio y consulta el estado de la solicitud (Aprobada, Rechazada, Pendiente). Mostrar detalles de la reserva y el historial de cambios.
CU3	Gestión de Reservas (Administrador)	El administrador puede ver todas las reservas, aprobarlas o rechazarlas, y modificar sus detalles. Modificación de la reserva (espacio, hora, fecha) y actualización de su estado.

Requerimientos Tecnicos

1. Base de Datos ● Estructura de Tablas: Crear tablas en SQL para las siguientes entidades: ● Espacios (nombre, capacidad, descripción). ● Reservas (folio, espacio, fecha, hora, solicitante, estado, descripción). ● Usuarios (empleados y administradores). ● Historial de Cambios (fecha, usuario, acción realizada, folio de reserva).
2. API REST ● La comunicación con la base de datos debe realizarse a través de una API REST utilizando los métodos GET, POST, PUT y DELETE: ○ POST: Crear nueva reserva. ○ GET: Consultar el estado de una reserva. ○ PUT: Aprobar, rechazar o modificar una reserva. ○ DELETE: Cancelar una reserva (solo administradores)
3. Cliente Web ● Lenguaje y Framework: El cliente web debe ser desarrollado utilizando al menos uno de los siguientes lenguajes y frameworks: ○ Frontend: React.js, Angular, o Vue.js. ○ Backend: Node.js con Express, Django, o Java Spring Boot. ● Interfaz Responsiva: La interfaz debe adaptarse a diferentes dispositivos (computadoras de escritorio, laptops, tabletas y teléfonos).
4. Patrones Arquitectónicos ● Implementar patrones arquitectónicos como MVC (Modelo-Vista-Controlador) o RESTful. ● Utilizar un enfoque basado en microservicios si el sistema es parte de una infraestructura más grande.
5. Programación Orientada a Objetos ● Uso de clases y objetos para estructurar la lógica del sistema (por ejemplo, clases para Reserva, Espacio, Usuario, etc.).
6. Autenticación y Seguridad ● Implementar autenticación segura para el portal de administración utilizando JWT o similar. ● Cifrado de contraseñas: Utilizar bcrypt o una librería de cifrado similar.

Sugerencias para Actualización

- **Interfaz de Usuario (UI/UX)**: Desarrollar una interfaz moderna y fácil de usar utilizando herramientas como **Tailwind CSS** o **Bootstrap**.

- **Automatización de pruebas**: Implementar **pruebas unitarias** y de **integración** para asegurar que los cambios no rompan funcionalidades existentes.

- **Gestión de Errores**: Implementar un adecuado **manejo de errores**, devolviendo códigos de estado HTTP apropiados y mensajes de error claros.
