### 1. ****Sistema de Gestión de Tareas****

**Descripción**: Aplicación web que permite a los usuarios crear, editar y eliminar tareas. Este proyecto está diseñado para evaluar las habilidades del candidato en la implementación de interfaces de usuario, bases de datos, y autenticación de usuarios.

**Objetivos del Proyecto**: Evaluar la capacidad del candidato para trabajar con interfaces web simples, gestionar el estado de la aplicación en el frontend y manipular datos en el backend.

**Alcance**:

* **Funcionalidades Incluidas**: Registro e inicio de sesión, creación, edición y eliminación de tareas, y visualización de tareas pendientes.
* **Funcionalidades No Incluidas**: Integraciones complejas, como recordatorios, tareas compartidas o sincronización en tiempo real.

**Dependencias**:

* Base de datos relacional (MySQL/PostgreSQL) para almacenamiento de tareas.
* Framework frontend (React.js, Vue.js, Angular).
* Sistema de autenticación (JWT o OAuth).

**Criterios de Aceptación**:

* El sistema debe permitir la creación, edición y eliminación de tareas correctamente.
* La interfaz debe ser funcional y accesible, con tiempos de carga inferiores a 2 segundos.
* La autenticación debe ser segura, con encriptación de contraseñas.

**Riesgos**:

* **Tiempo limitado**: El tiempo de desarrollo podría ser ajustado si no se gestionan bien los plazos de entrega.
* **Problemas de escalabilidad**: El sistema debe ser sencillo, pero si se añaden funcionalidades complejas podría afectar la calidad del código.

**Plan de implementación y despliegue**:

* **Iteración 1 (1 semana)**: Diseño UI/UX y estructura inicial.
* **Iteración 2 (2 semanas)**: Implementación de la funcionalidad de usuario y tareas.
* **Iteración 3 (2 semanas)**: Desarrollo del backend y persistencia de datos.
* **Iteración 4 (1 semana)**: Pruebas, optimización y corrección de errores.

### 2. ****Plataforma de Encuestas****

**Descripción**: Plataforma para crear y participar en encuestas, diseñada para evaluar la habilidad del candidato en manejar bases de datos, interfaces dinámicas y gráficos interactivos.

**Objetivos del Proyecto**: Evaluar la habilidad en la creación de interfaces interactivas, gestión de formularios dinámicos y presentación de resultados en tiempo real.

**Alcance**:

* **Funcionalidades Incluidas**: Registro de usuarios, creación de encuestas con opciones múltiples, participación en encuestas y visualización de resultados en gráficos.
* **Funcionalidades No Incluidas**: Funciones de análisis avanzado o integración con plataformas externas para compartir encuestas.

**Dependencias**:

* API para la visualización de gráficos (Chart.js, D3.js).
* Base de datos para almacenar respuestas (MongoDB, MySQL).
* Framework frontend (React, Vue.js).

**Criterios de Aceptación**:

* Los usuarios deben poder crear encuestas con múltiples opciones y votar.
* Los resultados deben mostrarse correctamente en gráficos dentro de 3 segundos.
* La interfaz debe ser intuitiva y cumplir con las pautas de accesibilidad WCAG 2.1.

**Riesgos**:

* **Dependencia de APIs**: El uso de bibliotecas para gráficos podría no integrarse bien con el resto de la arquitectura.
* **Escalabilidad limitada**: Si se incrementa el número de encuestas o usuarios, el sistema podría requerir optimización.

**Plan de implementación y despliegue**:

* **Iteración 1 (1 semana)**: Diseño de la interfaz de usuario y flujo de encuestas.
* **Iteración 2 (2 semanas)**: Implementación de la creación y participación en encuestas.
* **Iteración 3 (2 semanas)**: Desarrollo de la visualización de resultados y gráficos interactivos.
* **Iteración 4 (2 semanas)**: Pruebas, optimización y corrección de errores.

### 3. ****Sistema de Reservas para Restaurantes****

**Descripción**: Sistema para gestionar reservas en un restaurante. Este proyecto se centra en evaluar la habilidad del candidato para trabajar con bases de datos, gestión de reservas, y lógica de negocios.

**Objetivos del Proyecto**: Evaluar la capacidad para manejar la lógica de negocios de reservas, la gestión de usuarios y la integración con sistemas de correo electrónico.

**Alcance**:

* **Funcionalidades Incluidas**: Registro de usuarios, creación y gestión de reservas, confirmaciones por correo electrónico.
* **Funcionalidades No Incluidas**: Integración con sistemas de punto de venta (POS) o análisis avanzado de las reservas.

**Dependencias**:

* API de correo electrónico para confirmaciones (e.g., SendGrid).
* Base de datos (MySQL o PostgreSQL) para almacenar reservas.
* Framework frontend (React.js, Vue.js).

**Criterios de Aceptación**:

* Los usuarios pueden realizar, editar y cancelar reservas sin errores.
* El sistema debe enviar confirmaciones por correo electrónico.
* El sistema debe manejar el flujo de reservas en tiempo real sin retrasos importantes.

**Riesgos**:

* **Errores en la lógica de reservas**: Si la lógica para manejar las reservas no está bien estructurada, se podrían generar errores o inconsistencias.
* **Dependencia de correo electrónico**: Si el sistema de correo falla, la confirmación de reservas no se enviará a los usuarios.

**Plan de implementación y despliegue**:

* **Iteración 1 (1 semana)**: Diseño de la estructura de reservas y flujo de usuario.
* **Iteración 2 (2 semanas)**: Implementación del sistema de registro y reservas.
* **Iteración 3 (2 semanas)**: Desarrollo de las confirmaciones por correo electrónico y gestión administrativa.
* **Iteración 4 (1 semana)**: Pruebas, optimización y corrección de errores.

### 4. ****Aplicación de Clima****

**Descripción**: Aplicación para mostrar el clima actual y pronósticos. Este proyecto está diseñado para evaluar la capacidad de manejar APIs externas y representar datos en una interfaz de usuario.

**Objetivos del Proyecto**: Evaluar la habilidad para consumir APIs externas, gestionar la visualización de datos en una interfaz dinámica y optimizar la experiencia en dispositivos móviles.

**Alcance**:

* **Funcionalidades Incluidas**: Búsqueda del clima por ciudad, visualización de clima actual y pronóstico extendido a 7 días.
* **Funcionalidades No Incluidas**: Alertas meteorológicas en tiempo real o predicciones hiperlocalizadas.

**Dependencias**:

* API de clima (OpenWeatherMap, AccuWeather).
* Framework frontend (React.js).
* Framework backend (Node.js) si se requiere procesamiento adicional.

**Criterios de Aceptación**:

* La aplicación debe cargar los datos meteorológicos en menos de 3 segundos.
* La información del clima debe ser precisa y mostrarse correctamente en el frontend.
* La aplicación debe ser totalmente responsiva y funcionar bien en dispositivos móviles.

**Riesgos**:

* **Dependencia de la API externa**: Si la API de clima no responde correctamente, el sistema no podrá mostrar datos relevantes.
* **Problemas de responsividad**: Si la aplicación no es responsiva, no se podrá utilizar correctamente en dispositivos móviles.

**Plan de implementación y despliegue**:

* **Iteración 1 (1 semana)**: Diseño de la interfaz y flujo de la aplicación.
* **Iteración 2 (2 semanas)**: Implementación de la API para obtener datos meteorológicos.
* **Iteración 3 (1 semana)**: Desarrollo del frontend para mostrar los datos climáticos.
* **Iteración 4 (1 semana)**: Pruebas, optimización y corrección de errores.