**Documento de Requerimientos del Sistema de Reservas para Restaurante**

**1. Requerimientos Funcionales (RF)**

* **RF1: Gestión de Mesas**
  + **RF1.1:** El sistema debe inicializar un arreglo de objetos en localStorage al cargar la página, representando las mesas del restaurante.
  + **RF1.2:** Cada objeto de mesa debe contener las siguientes propiedades:
    - id: Un identificador único de la mesa (e.g., mesa1, mesa2).
    - capacidad: Un número que indica la capacidad máxima de personas (e.g., 2, 4, 6).
    - ubicacion: Un string que describe la ubicación de la mesa (e.g., "Ventana", "Jardín").
    - Estado: en este campo se va manejar 3 estados que son disponible, ocupada o deshabilitada
* **RF2: Gestión de Reservas**
  + **RF2.1:** El sistema debe permitir al usuario crear una nueva reserva.
  + **RF2.2:** Cada reserva debe ser un objeto con las siguientes propiedades:
    - idReserva: Un identificador único para la reserva.
    - nombreCliente: El nombre del cliente.
    - numeroPersonas: El número de personas de la reserva.
    - fechaReserva: La fecha de la reserva.
    - horaReserva: La hora de la reserva.
    - ocasionEspecial: Un string que indique la ocasión (opcional).
    - notasAdicionales: Un string para notas adicionales (opcional).
    - idMesaAsignada: El id de la mesa asignada a la reserva.
    - estado: El estado actual de la reserva (Pendiente, Confirmada, Cancelada, Finalizada, No Show).
* **RF3: Validaciones de Datos**
  + **RF3.1:** El **nombre del cliente** debe ser un campo obligatorio.
  + **RF3.2:** El **número de personas** debe ser un campo obligatorio y un número positivo mayor que cero.
  + **RF3.3:** La **fecha de la reserva** debe ser un campo obligatorio y debe ser una fecha posterior a la fecha actual.
  + **RF3.4:** La **hora de la reserva** debe ser un campo obligatorio y debe estar en el rango de 8:00 AM a 8:00 PM.
  + **RF3.5:** El sistema debe validar que la **mesa seleccionada esté disponible** en la fecha y hora de la reserva. Una mesa se considera "ocupada" si ya tiene una reserva activa en el mismo rango de tiempo.
* **RF4: Visualización**
  + **RF4.1:** El sistema debe mostrar un **plano del restaurante** (una representación visual de las mesas será en forma de una card que lleve todos los datos y si su estado es ocupada que sea de color azul y si está disponible color verde si esta deshabilitada de color negro y letra blanca).
    - **REF 4.1.1:** La card de la mesa debe tener tres acciones uno para editar los datos de la mesa, otro botón que sirva para reservar y el botón de eliminar
  + **RF4.2:** Cada mesa en el plano debe indicar visualmente si está **disponible u ocupada** en la fecha y hora seleccionada.
  + **RF4.3:** El sistema debe mostrar una lista de todas las reservas en una grilla de tarjetas.
  + **RF4.4:** Al momento de hacer una reserva el campo mesa debe ser una lista desplegable y debe mostrar las mesas disponibles (deben llenar la lista con el vector mesas en estado disponible)
  + **RF4.5:** El sistema debe mostrar una lista desplegable en el formulario de reserva con diferentes tipos de "ocasión especial" (ej. "Cumpleaños", "Aniversario", "Reunión de Negocios", "Ninguna") mínimo 8 tipos en la lista, dependiendo de la ocasión le van a poner una imagen a la card de reserva.
  + **RF4.6:** La card reserva debe tener tres acciones, la de editar, la de pagar cuenta y eliminar
    - **REF 4.1.2:** La card llevará un botón que diga pagar y al momento de pagar la reserva pasa a estado Finalizada
    - **REF 4.1.3:** Al momento de pagar también el estado de la mesa asignada debe pasar a disponible
* **RF5: Modificación de Reservas**
  + **RF5.1:** El sistema debe permitir al usuario **cambiar el estado** de una reserva (e.g., de "Pendiente" a "Confirmada").
  + **RF5.2:** El sistema debe permitir al usuario **anular o cancelar** una reserva, cambiando su estado a "Cancelada".
* **RF6: Filtrado de Reservas**
  + **RF6.1:** El sistema debe tener un filtro por **fecha** para mostrar solo las reservas de un día específico.
  + **RF6.2:** El sistema debe tener un filtro por **estado** para mostrar solo las reservas con estados específicos (e.g., "Pendientes" o "Confirmadas").

**2. Requerimientos No Funcionales (RNF)**

* **RNF1: Interfaz de Usuario**
  + **RNF1.1:** La interfaz debe ser **intuitiva y fácil de usar**.
  + Losformularios van en modales
  + **RNF1.2:** De manera opcional se puede utilizar un **framework CSS** para el diseño de la interfaz. Se sugieren opciones como:
    - **Bootstrap**: Muy popular y con muchos componentes.
    - **Tailwind CSS**: Ideal para un control total sobre el diseño.
    - **Materialize**: Basado en los principios de diseño de Google Material.
* **RNF2: Persistencia de Datos**
  + **RNF2.1:** Todos los datos de mesas y reservas deben ser almacenados en el **localStorage** del navegador.
  + **RNF2.2:** Al recargar la página, el sistema debe recuperar la información de localStorage.
* **RNF3: Manejo de Errores**
  + **RNF3.1:** El sistema debe mostrar **mensajes de error claros y concisos** cuando una validación falle. Por favor no usar las alertas de HTML o las alertas de Windows