Documento de Funcionalidades

Web App de Turnos para Peluquería

**1. Descripción General**

Esta aplicación web permitirá a los clientes de una peluquería solicitar turnos y consultar el estado de su solicitud de manera sencilla. Además, el administrador podrá gestionar los turnos, modificar su estado y notificar a los clientes sobre cualquier actualización.

**2. Funcionalidades**

**2.1 Funcionalidades para Usuarios (Clientes)**

* **Solicitar un turno:**
  + Ingresar nombre, teléfono o correo electrónico.
  + Seleccionar la fecha y el horario deseado.
  + Enviar la solicitud.
* **Consultar estado del turno:**
  + Ingresar ID de turno o correo electrónico.
  + Visualizar los detalles del turno y su estado actual.
* **Recibir notificaciones:**
  + Recibir un correo electrónico cuando el estado del turno cambie.

**2.2 Funcionalidades para Administrador (Peluquería)**

* **Gestión de turnos:**
  + Acceder a un panel con todos los turnos solicitados.
  + Filtrar turnos por estado (pendiente, confirmado, cancelado, etc.).
* **Actualización de turnos:**
  + Modificar el estado de un turno (ejemplo: de "pendiente" a "confirmado").
  + Enviar una notificación automática al cliente con la actualización.

**2.3 Funcionalidades Generales**

* **Interfaz Responsiva:**
  + Diseño adaptable para dispositivos móviles y escritorio.
* **Gestión de Base de Datos:**
  + Almacenamiento seguro de los turnos y sus datos asociados.
* **Seguridad y Privacidad:**
  + Validación de datos en formularios.
  + Acceso restringido para el administrador mediante autenticación.

**3. Diagrama de Flujo**

**3.1 Proceso de Solicitud de Turno**

1. El usuario ingresa a la web.
2. Completa el formulario con sus datos.
3. Envía la solicitud.
4. El sistema almacena el turno en la base de datos.
5. Se muestra un mensaje de confirmación con el estado del turno.
6. El usuario puede consultar su turno posteriormente.

**3.2 Proceso de Gestión de Turnos (Administrador)**

1. El administrador accede al panel de control.
2. Visualiza la lista de turnos registrados.
3. Cambia el estado de un turno si es necesario.
4. El sistema envía automáticamente una notificación al cliente con la actualización.

**4. Próximos Pasos**

Diseñar las pantallas principales de la interfaz.

Definir la estructura de la base de datos.

Implementar el sistema de autenticación para el administrador.

**Arquitectura de la Aplicación**

Una arquitectura típica para este tipo de aplicaciones sigue el modelo **Cliente-Servidor** con los siguientes componentes:

**Frontend (Interfaz de Usuario)**

Este será el sitio web donde los clientes podrán solicitar y consultar turnos, y donde el administrador podrá gestionarlos.  
**Opciones tecnológicas:**

* **React.js** (moderno, rápido y escalable).
* **Vue.js** (más sencillo y progresivo).
* **HTML, CSS, JavaScript con Bootstrap o Tailwind CSS** (si prefieres algo más básico para comenzar).

**Backend (Lógica y API)**

El backend se encargará de procesar solicitudes, validar datos y comunicarse con la base de datos.  
**Opciones tecnológicas:**

* **Flask (Python)**: Simple y fácil de aprender. Ideal para una primera API REST.
* **Django (Python) con Django Rest Framework (DRF)**: Más robusto, con autenticación lista para usar y ORM integrado.
* **Node.js con Express.js**: Si te gusta JavaScript y quieres una solución rápida y moderna.

**Tabla: usuarios**

| **Campo** | **Tipo de Dato** | **Descripción** |
| --- | --- | --- |
| id | INTEGER (PK, autoincrement) | Identificador único. |
| nombre | VARCHAR(100) | Nombre completo del cliente. |
| telefono | VARCHAR(20) | Número de teléfono del cliente. |
| email | VARCHAR(100) | Correo electrónico del cliente. |

**Tabla: turnos**

| **Campo** | **Tipo de Dato** | **Descripción** |
| --- | --- | --- |
| id | INTEGER (PK, autoincrement) | Identificador único del turno. |
| usuario\_id | INTEGER (FK) | Relación con la tabla usuarios. |
| fecha | DATETIME | Fecha y hora del turno. |
| estado | VARCHAR(20) | Estado del turno (Pendiente, Confirmado, Cancelado). |
| creado\_en | DATETIME | Fecha de creación del turno. |

**Tabla: administradores**

| **Campo** | **Tipo de Dato** | **Descripción** |
| --- | --- | --- |
| id | INTEGER (PK, autoincrement) | Identificador único del administrador. |
| usuario | VARCHAR(50) | Nombre de usuario del administrador. |
| password | VARCHAR(255) | Contraseña encriptada. |

**Frontend:**

* **React.js con Vite** para una experiencia moderna y rápida.
* **TailwindCSS** para el diseño.

**Backend:**

* **FastAPI (Python)** → Rápido y compatible con AWS Lambda. Alternativa: **Node.js (Express.js)** si prefieres JavaScript.
* **Base de datos:** PostgreSQL (en producción) o SQLite (para desarrollo).

**Despliegue:**

* **Docker** → Para empaquetar la aplicación y ejecutarla en cualquier servidor.
* **AWS (EC2, S3, RDS, Lambda, etc.)** → Si quieres aprender más sobre la nube.
* **Alternativa:** Heroku o Railway para una solución más fácil.

**¿Cómo avanzar con la arquitectura y el modelo de datos?**

**Paso 1: Diseño de la base de datos**

1. **Elige la tecnología:** ¿Vas a usar **SQL (PostgreSQL, MySQL, SQLite)** o una opción NoSQL como **MongoDB o DynamoDB**?
2. **Define los modelos** en una herramienta como **Draw.io** o en papel.
3. **Configura el ORM** en el backend, por ejemplo, **SQLAlchemy** para Python o **Prisma (Node.js)** para trabajar con bases de datos de forma más sencilla.

**Paso 2: Configuración del Backend**

1. **Crea la API REST con FastAPI o Express.js**.
2. **Define las rutas básicas** para:
   * Solicitar un turno (POST /turnos),
   * Consultar turnos (GET /turnos/{id}),
   * Cambiar estado de un turno /turnos/{id}/estado (para el administrador).
3. **Configura la conexión a la base de datos** y prueba con SQLite o PostgreSQL.
4. **Añade autenticación con JWT (JSON Web Tokens)** para la parte del administrador.

**Paso 3: Creación del Frontend**

1. **Diseña las pantallas con Figma o papel y lápiz.**
2. **Elige entre HTML+CSS o React.js para la interfaz.**
3. **Crea los formularios básicos:**
   * Formulario para solicitar turnos.
   * Página para consultar estado de turnos.
   * Panel de administración.

**Paso 4: Integración con Docker y AWS (Opcional)**

1. **Dockerizar la aplicación** (crear un Dockerfile y una docker-compose.yml).
2. **Configurar AWS EC2 o RDS** para la base de datos y el backend.
3. **Desplegar la aplicación en la nube** utilizando AWS Elastic Beanstalk o Fargate.