
APP IOS

Requerimientos Funcionales

1. **Reserva de espacios de parqueadero**:
    - El usuario debe poder seleccionar una **fecha**, **hora de entrada**, **hora de salida**, y **número de placa** para realizar una reserva.
    - La app debe mostrar las franjas horarias disponibles y las ocupadas en tiempo real, obteniendo esta información de Firebase.
2. **Modificación y cancelación de reservas**:
    - Los usuarios deben tener la posibilidad de **modificar** o **cancelar** una reserva siempre y cuando sea antes del inicio del tiempo reservado.
    - **Restricción**: Si el usuario llega tarde a su reserva, esta se debe cancelar automáticamente.
3. **Notificaciones push**:
    - Los usuarios deben recibir notificaciones push automáticas cuando se completen acciones importantes, tales como:
        - **Reserva confirmada**.
        - **Penalización aplicada**.
        - **Cancelación automática de la reserva por tardanza**.
        - **Acceso aprobado o denegado** cuando lleguen al parqueadero.
4. **Historial de reservas**:
    - La app debe permitir al usuario ver un historial de reservas pasadas y próximas. Este historial debe incluir información sobre si se le impuso una penalización o si hubo algún cambio en la reserva.
5. **Verificación de penalizaciones**:
    - Los usuarios deben ser notificados si tienen **penalizaciones activas** que les impidan hacer nuevas reservas. La app debe mostrar claramente la razón de la penalización y la duración de la misma.
6. **Consulta del estado de la reserva en tiempo real**:
    - Los usuarios deben poder consultar el estado actual de su reserva en tiempo real. Por ejemplo, si su reserva está activa y si el sistema está listo para permitir el acceso cuando lleguen al parqueadero.
7. **Verificación de salida del parqueadero**:
    - La app debe permitir al usuario verificar si el sistema ha registrado su salida correctamente y si su tiempo en el parqueadero ha excedido el reservado.

Requerimientos Técnicos

1. **Conexión a Firebase**:
    - La app debe estar configurada para interactuar con los servicios de Firebase:
        - **Firestore** para la gestión de las reservas y los datos.
        - **Firebase Cloud Messaging (FCM)** para las notificaciones push.
2. **Manejo de datos en tiempo real**:
    - La app debe actualizar en tiempo real la disponibilidad de los espacios en el parqueadero mediante Firebase Firestore.

Requerimientos de Integración con el Hardware

1. **Validación en tiempo real**:
    - La app debe estar conectada a Firebase de manera que cuando el usuario llegue al parqueadero, el sistema pueda verificar en tiempo real si tiene una reserva válida y permitir el acceso del vehículo.
2. **Comunicación con el sistema de reconocimiento de placas**:
    - La app no necesita interactuar directamente con el sistema de hardware, pero debe registrar las reservas en Firestore, donde el sistema de hardware (cámara y Raspberry Pi) consultará las placas y permitirá o denegará el acceso.

Hardware

Requerimientos Funcionales del Sistema de Hardware

1. **Captura de la matrícula del vehículo**:
    - El sistema debe contar con una **cámara** capaz de capturar imágenes nítidas de las placas de los vehículos que entran y salen del parqueadero.
    - La cámara debe estar **conectada a un microcontrolador** (como Raspberry Pi) que capture las imágenes y las procesa en tiempo real para extraer el número de la matrícula.
2. **Procesamiento de imágenes para reconocimiento de placas**:
    - Utilizar **OpenCV** (librería de procesamiento de imágenes) para analizar las imágenes capturadas y reconocer los caracteres de la matrícula del vehículo.
    - El microcontrolador debe ser capaz de procesar las imágenes en tiempo real, con un tiempo de respuesta adecuado (idealmente menos de 2 segundos).
3. **Comparación con la base de datos (Firebase)**:

    - El sistema debe conectarse a **Firebase** para verificar si la matrícula capturada tiene una reserva activa en ese momento.
    - Utilizar **Firebase Admin SDK** para realizar consultas en tiempo real y determinar si el vehículo tiene permiso para acceder al parqueadero.
    - El sistema debe verificar:
        - Si la placa tiene una reserva activa.
        - Si el vehículo llega dentro de la franja horaria reservada.
4. **Control de la barrera o láser**:
    - Dependiendo del resultado de la verificación en Firebase, el sistema debe **activar o desactivar un relé** que controla un láser.
    - Si la matrícula está autorizada, el relé debe desactivar el láser para permitir el paso del vehículo.
    - Si la matrícula no está autorizada, el sistema debe mantener el láser activado, bloqueando el acceso.
5. **Detección de vehículos (sensores de proximidad)**:
    - El sistema debe contar con un **sensor de proximidad** (como un sensor PIR o ultrasónico) que detecte cuando un vehículo llega a la entrada del parqueadero.
    - Al detectar un vehículo, el sistema debe activar la cámara para capturar la imagen de la placa.
6. **Registro de salida**:
    - El sistema debe repetir el proceso al momento de la salida del parqueadero, capturando la matrícula del vehículo que sale.
    - Comparar la hora de salida registrada con la hora reservada para aplicar posibles penalizaciones si el vehículo ha excedido el tiempo permitido.
7. **Aplicación de penalizaciones**:
    - Si un vehículo permanece más tiempo del reservado, el sistema debe registrar la penalización en Firebase y actualizar el estado del usuario, restringiendo su acceso futuro según las reglas definidas (por ejemplo, bloqueo de reservas durante el día siguiente).

Requerimientos de Componentes Hardware

1. **Cámara**:
    - **Raspberry Pi Camera Module** o una **cámara USB** de alta resolución, capaz de capturar imágenes de placas de vehículos.
    - Debe estar montada en una posición fija y alineada con la entrada para asegurar la captura precisa de la matrícula.
2. **Microcontrolador**:

    - **Raspberry Pi 4** (o modelo superior) es recomendado para manejar el procesamiento de imágenes (OpenCV), las conexiones a Firebase, y controlar el relé para la barrera.
    - El Raspberry Pi debe estar conectado a internet (WiFi o Ethernet) para poder realizar las verificaciones en Firebase en tiempo real.
3. **Relé**:
    - Un **módulo de relé de 5V o 12V** que permita controlar el láser.
    - El relé debe estar conectado a los pines GPIO del Raspberry Pi para ser activado o desactivado según los resultados de la verificación de la placa.
4. **Barrera física o láser**:
    - Un **láser** que se active o desactive para permitir o bloquear el acceso.
    - Debe ser resistente y adecuada para el entorno en el que se instala (interior o exterior).
5. **Sensores de proximidad**:
    - **Sensor PIR** (sensor de infrarrojos pasivo) o **sensor ultrasónico** para detectar cuando un vehículo se aproxima a la entrada del parqueadero.
    - Debe estar alineado correctamente para activar la cámara en el momento adecuado.
6. **Fuente de alimentación**:
    - **5V** para el Raspberry Pi y **12V** para los dispositivos de la barrera o láser.
    - Debe ser confiable y tener capacidad suficiente para alimentar todos los dispositivos conectados.

Requerimientos de Software en el Sistema de Hardware

1. **OpenCV para reconocimiento de imágenes**:
    - La librería **OpenCV** se utilizará en Python para procesar las imágenes de las placas capturadas por la cámara y extraer los números de matrícula.
2. **Firebase Admin SDK**:
    - El microcontrolador debe tener instalado el **Firebase Admin SDK** para poder realizar las consultas y verificar las reservas en tiempo real.
3. **Control GPIO**:
    - Utilizar **RPi.GPIO** o **pigpio** para controlar los pines GPIO del Raspberry Pi, encargados de activar el relé que maneja la barrera física o láser.
4. **Sistema de registro**:
    - El sistema debe registrar las entradas y salidas de los vehículos en Firebase, indicando la hora de entrada, hora de salida, y si el acceso fue autorizado o denegado.