ESPECIFICACIÓN DE REQUISITOS DEL SOFTWARE (SRS)

Sistema de Control de Acceso con IoT para Áreas Restringidas

Dr. Ray Brunett Parra Galaviz

Diseño e Implementación de un Sistema de Seguridad Inteligente

TSU En Tecnologías de la información

Meza Osuna Juan Manuel

Alvares Salazar Erick Sidrac

Sanchez Murillo Eduardo

Gameros Garcia Angel Geovanny

4B

Ficha del documento

Fecha	Revisión	Autor	Verificado dep. calidad.
20/01/2025		Meza Osuna Juan Manuel Alvares Salazar Sidrac Gameros Garcia Angel Geovanny Sanchez Murillo Eduardo	

Documento validado por las partes en fecha:

Por el cliente	Por la empresa suministradora
Fdo. D./ Dña	Fdo. D./Dña

Contenido

1 Introducción

Este documento presentará la Especificación de Requisitos de Software (SRS) correspondiente al proyecto en desarrollo, detallando de manera estructurada los requisitos funcionales y no funcionales relacionados con el sistema de seguridad destinado a entornos industriales.

Propósito

Aportar una solución de seguridad para controlar el acceso en instalaciones industriales sensibles.

El sistema debe garantizar que solo el personal autorizado pueda ingresar a áreas restringidas, mejorando la seguridad y reduciendo los riesgos de acceso no autorizado. El sistema debe permitir la administración eficiente de permisos y la auditoría en tiempo real de las actividades dentro de la instalación.

Alcance

El sistema proporcionará una solución de seguridad, utilizando dispositivos IoT, una interfaz para la administración de permisos, monitoreo en tiempo real de los accesos, auditoría de los eventos de seguridad, así como monitorear y gestionar las rondas realizadas por el personal de seguridad

Control de Acceso con IoT:

- Implementación de dispositivos IoT (lectores RFID, sensores biométricos) para autenticar y registrar los accesos a las áreas restringidas.
- Autorización de accesos en tiempo real a través de credenciales válidas (tarjetas RFID).

Gestión de Permisos:

- Creación y gestión de perfiles de usuarios con permisos de acceso configurables.
- Asignación de permisos a usuarios específicos para acceder a áreas especificas dentro de la instalación, en determinados horarios.

Monitoreo en Tiempo Real:

- Visualización de los accesos en tiempo real con detalles como el usuario, el área y la hora.
- Alertas en tiempo real para intentos de acceso no autorizado o violaciones de seguridad.

Sistema de Rondines para Seguridad:

- **Puntos de Control:** Se definirán puntos estratégicos donde el personal de seguridad deberá registrar su presencia durante las rondas nocturnas.

- **Registro de Rondas**: El personal escaneará su tarjeta RFID o usará una app móvil para registrar su presencia en los puntos de control. Esto permitirá asegurar que las rondas se estén cumpliendo de acuerdo con el plan.

- **Alertas:** Si el guardia no llega a un punto de control a tiempo, el sistema enviará

alertas a los administradores, al igual que con los accesos no autorizados.

- **Informes de Rondas:** El sistema generará informes sobre las rondas realizadas, detallando los tiempos y cualquier incidente registrado.

Interfaz Administrativa:

- Desarrollo de una interfaz web y móvil para los administradores del sistema, que permita gestionar dispositivos IoT, permisos y visualizar estadísticas de acceso.
- Funcionalidad para configurar y actualizar dispositivos de acceso.

Funcionalidades fuera del alcance

Integración con Sistemas Externos:

El sistema no incluye la integración con otros sistemas de seguridad de terceros, aunque se puede considerar como una extensión futura del sistema.

Control de Acceso en Entornos Externos:

El sistema está diseñado específicamente para áreas restringidas dentro de instalaciones industriales. No se incluye control de acceso para entornos externos o fuera de las instalaciones industriales.

Gestión de Personal o Recursos Humanos:

El sistema no gestionará datos personales completos de los empleados, como información de salario, empleo o historial médico. Solo gestionará información relevante para el control de acceso.

1.1 Personal involucrado

Nombre	Persona 1
Rol	Lider de proyecto
Categoría profesional	Desarrollador de software

Responsabilidades	• Supervisar el progreso general del proyecto y asegurar que se cumplan los plazos y objetivos. • Coordinar las tareas del equipo y asignar responsabilidades. • Asegurar que la integración de hardware (dispositivos IoT) y software (aplicación web/móvil) se lleve a cabo correctamente. • Revisión y aprobación de las fases de diseño y desarrollo del sistema.
Información de contacto	+1234567891
Aprobación	Si

Nombre	Persona 2
Rol	Desarrollador Backend
Categoría profesional	Desarrollador de software
Responsabilidades	• Desarrollar la lógica de servidor y la integración con dispositivos IoT (lectores RFID, sensores biométricos). • Crear la base de datos para gestionar accesos, rondas y generación de informes. • Implementar las APIs necesarias para la comunicación entre la aplicación móvil y la base de datos.
Información de contacto	+1234567891
Aprobación	Si

Nombre	Persona 3
Rol	Desarrollador Frontend
Categoría profesional	Desarrollador de software
Responsabilidades	• Diseñar y desarrollar la interfaz de usuario para la aplicación web y móvil. • Asegurar que la interfaz sea fácil de usar y adecuada para la gestión de rondas, accesos y generación de informes. • Trabajar en la integración con la lógica de backend, mostrando los datos de forma clara y eficiente. • Implementar las funcionalidades para el monitoreo en tiempo real y generación de alertas.
Información de contacto	+1234567891
Aprobación	Si

Nombre	Persona 4
Rol	Especialista en Seguridad IoT
Categoría profesional	Desarrollador de software
Responsabilidades	• Supervisar la instalación y configuración de los dispositivos IoT (lectores RFID). • Asegurar la comunicación adecuada entre los dispositivos IoT y la plataforma de backend. • Implementar y mantener las funciones de seguridad física para la protección de datos y dispositivos. • Evaluar y probar el sistema de rondines para asegurar que los puntos de control funcionen correctamente.
Información de contacto	+1234567891
Aprobación	Si

1.2 Definiciones, acrónimos y abreviaturas

Sistema de Control de Acceso (SCA): Conjunto de tecnologías y procesos que permiten gestionar y controlar el acceso de personas a áreas restringidas dentro de una instalación.

IoT (Internet of Things - Internet de las Cosas): Conjunto de dispositivos físicos conectados a Internet, capaces de recolectar, enviar y recibir datos de manera autónoma.

RFID (Radio Frequency Identification - Identificación por Radiofrecuencia): Tecnología que utiliza ondas de radio para identificar de manera única objetos o personas mediante etiquetas o tarjetas.

Punto de Control: Ubicación física dentro de la instalación donde el personal de seguridad debe registrar su presencia durante las rondas de vigilancia.

Rondines de Seguridad: Actividad en la que el personal de seguridad recorre puntos específicos dentro de la instalación para garantizar que las áreas estén protegidas.

Auditoría de Seguridad: Proceso de revisión y análisis de los registros de acceso y rondas para evaluar el cumplimiento de las políticas de seguridad.

UI: User Interface (Interfaz de Usuario)

UX: User Experience (Experiencia de Usuario)

1.3 Referencias

Referencia	Titulo	Fecha	Autor
#1	Manual de lectores RFID	Enero 2025	Juan Perez
#2	Estándares de Seguridad para IoT	Marzo 2024	Carlos Rodríguez
#3	Normas de Encriptación y Cifrado AES	Diciembre 2024	Laura Garcia

1.4 Resumen

Este documento describe el desarrollo de un Sistema de Control de Acceso con IoT para Áreas Restringidas, diseñado para mejorar la seguridad en instalaciones industriales. El sistema incluye dispositivos IoT para autenticar y controlar el acceso, así como un sistema de rondines para gestionar las rondas de seguridad.

2 Descripción general

2.1 Perspectiva del producto

El Sistema de Control de Acceso con IoT para Áreas Restringidas es un producto independiente que se integra en instalaciones industriales para gestionar el acceso y monitorear las rondas de seguridad. Este sistema está diseñado para ser autónomo en su funcionamiento, pero puede ser ampliado o integrado en un sistema mayor de gestión de seguridad industrial si fuera necesario.

El sistema no requiere una infraestructura de seguridad previamente existente, pero puede interactuar con otros sistemas si es necesario, como alarmas de seguridad, cámaras de vigilancia, o sistemas de gestión de personal. El producto se compone de:

1. **Dispositivos IoT**: Lectores RFID, sensores biométricos y otros dispositivos que se instalan en puntos estratégicos de la instalación para verificar la identidad de los usuarios y registrar su acceso.
2. **Aplicación Web/Móvil**: Plataforma para gestionar los accesos y monitorear las rondas de seguridad, así como generar informes detallados sobre las actividades realizadas por el personal de seguridad.
3. **Sistema de Rondas de Seguridad**: Un módulo que permite al personal de seguridad realizar rondas controladas, con puntos de control definidos donde deben registrar su presencia mediante dispositivos IoT.

2.2 Funcionalidad del producto

El Sistema de Control de Acceso con IoT para Áreas Restringidas ofrece una serie de funcionalidades clave para gestionar de manera segura y eficiente el acceso a áreas sensibles dentro de instalaciones industriales. A continuación, se resumen las principales funcionalidades del producto:

Autenticación de Acceso:

El sistema permite autenticar a los usuarios mediante dispositivos IoT, como lectores RFID y sensores biométricos.

El acceso a las áreas restringidas está controlado mediante permisos configurados para cada usuario, garantizando que solo personal autorizado pueda ingresar.

Gestión de Permisos:

Los administradores del sistema pueden configurar permisos de acceso para cada usuario desde la aplicación web/móvil.

Los permisos pueden ser asignados por área, horario y nivel de acceso, permitiendo un control granular de quién accede y cuándo.

Monitoreo de Accesos en Tiempo Real:

El sistema permite monitorear los accesos en tiempo real a través de la interfaz de gestión.

Cualquier intento de acceso no autorizado es registrado y genera alertas automáticas para los supervisores.

Registro de Rondas de Seguridad:

El personal de seguridad debe realizar rondas en puntos de control establecidos dentro de la instalación.

Durante las rondas, el personal registra su presencia utilizando los dispositivos IoT (lectores RFID o la app móvil), lo que asegura que se cumplan los procedimientos de seguridad.

Generación de Informes:

El sistema genera informes detallados sobre los accesos, incluyendo fecha, hora, lugar y persona involucrada.

También se generan informes sobre las rondas de seguridad realizadas, mostrando el cumplimiento de los puntos de control establecidos y cualquier incidente registrado.

Alertas de Seguridad:

El sistema envía notificaciones en tiempo real sobre accesos no autorizados, intentos de violación de seguridad o si el personal de seguridad no cumple con las rondas en el tiempo establecido.

Las alertas se envían a los administradores o supervisores para tomar medidas inmediatas.

Seguridad de Datos:

Los datos de acceso y las rondas de seguridad son almacenados de manera segura en la base de datos del sistema, con cifrado de alta seguridad.

Se implementan mecanismos de autenticación y autorización para proteger la información sensible.

Interfaz de Usuario Amigable:

El sistema ofrece una interfaz web/móvil intuitiva, accesible para administradores y personal de seguridad.

Los usuarios pueden gestionar permisos, visualizar registros y recibir alertas desde la plataforma de manera fácil y eficiente.

2.3 Características de los usuarios

Tipo de usuario	Administrador del sistema
Formación	Titulo en Ingenieria en sistemas, seguridad informatica, o areas relacionadas. Experiencia minima de 2 años
Habilidades	• Conocimiento en gestion de sistemas de control de acceso y seguridad. • Habilidad para configurar permisos de acceso, gestionar la base de datos y supervisar el funcionamiento general del sistema. • Conocimiento básico de redes, bases de datos y seguridad informática.
Actividades	• Configuración y gestión de permisos de acceso de los usuarios. • Monitoreo en tiempo real de los accesos y rondas de seguridad. • Generación de informes sobre accesos y actividades. • Gestión de alertas de seguridad y resolución de incidentes. • Supervisión y mantenimiento del sistema de seguridad.

2.4 Restricciones

Compatibilidad de Dispositivos IoT: El sistema debe ser compatible con lectores RFID y sensores biométricos específicos que se usarán en los puntos de control. Estos dispositivos deben cumplir con los requisitos de conectividad, como el soporte para Wi-Fi o Bluetooth**.**

Requerimientos de Energía: Algunos dispositivos IoT pueden tener restricciones de consumo energético, por lo que se debe planificar el uso de baterías de larga duración o integrar fuentes de energía adecuadas.

Restricciones de Software

Normas y Protocolos de IoT: Los dispositivos IoT deben ser compatibles con protocolos estándar como MQTT o HTTP para garantizar una comunicación eficiente y segura con la aplicación web y móvil.

Sistema Operativo: La aplicación web deberá ser accesible a través de navegadores modernos en sistemas operativos como Windows, Linux y macOS. Las aplicaciones móviles deben ser compatibles con iOS y Android.

Integración con Sistemas Externos: Aunque el sistema está diseñado como una solución independiente, en algunos casos podría ser necesario integrar el sistema con otros sistemas de seguridad existentes en la instalación, como cámaras de vigilancia o alarmas. Esta integración puede estar limitada por la compatibilidad de los protocolos de comunicación.

2.5 Suposiciones y dependencias

Suposiciones

Disponibilidad de Infraestructura de Red:

Se asume que las instalaciones industriales cuentan con una red Wi-Fi estable y de amplio alcance que permita la conexión constante de los dispositivos IoT.

Se dispone de acceso a internet para el funcionamiento de las aplicaciones web y móviles, en caso de ser necesario.

Compatibilidad de Hardware:

Se asume que los dispositivos IoT seleccionados (lectores RFID, sensores biométricos, etc.) son compatibles con los protocolos y estándares utilizados en el sistema. Los dispositivos IoT serán instalados y configurados adecuadamente según las especificaciones del diseño.

Disponibilidad de Recursos Humanos:

Los usuarios finales, incluidos los administradores y el personal de seguridad, recibirán capacitación adecuada para utilizar el sistema.

Entorno Operativo:

Se asume que el sistema será implementado en un entorno operativo con condiciones adecuadas (temperatura, energía eléctrica constante, etc.) para garantizar el correcto funcionamiento de los dispositivos IoT.

Cumplimiento de Plazos:

Se asume que no habrá interrupciones significativas en el cronograma del proyecto y que todos los componentes estarán disponibles según lo previsto.

Dependencias

Dependencia de Proveedores de Hardware:

El funcionamiento del sistema depende de la adquisición e instalación de dispositivos IoT específicos, como lectores RFID y sensores biométricos. Retrasos o problemas con los proveedores podrían afectar los plazos del proyecto.

Dependencia de Plataformas de Desarrollo:

La implementación de las aplicaciones web y móviles depende de plataformas de desarrollo como React nativa, y su compatibilidad con las versiones actuales de los sistemas operativos Android, iOS y navegadores webs modernos.

Dependencia de Normativas y Regulaciones:

El sistema depende del cumplimiento de regulaciones locales e internacionales de seguridad y privacidad de datos, como la GDPR o normativas locales relacionadas con la protección de datos personales.

Dependencia de Infraestructura Tecnológica:

El sistema depende de la disponibilidad de servidores para alojar la base de datos y las aplicaciones web. Cualquier problema con la infraestructura tecnológica podría afectar la funcionalidad del sistema.

Dependencia de Software de Terceros:

El uso de bibliotecas, frameworks y herramientas de terceros para la implementación del sistema puede generar dependencias. Cualquier cambio o discontinuidad en estas herramientas podría requerir ajustes en el desarrollo.

2.6 Evolución previsible del sistema

1. Expansión de Funcionalidades

- **Integración con Sistemas de Vigilancia**: Incorporación de cámaras de seguridad para asociar imágenes o videos en tiempo real con los eventos de acceso registrados.
- **Gestión Avanzada de Rondines**: Implementación de algoritmos de análisis predictivo para optimizar las rutas de los rondines de seguridad en función de eventos históricos y patrones de riesgo.
- **Soporte para Múltiples Ubicaciones**: Escalabilidad del sistema para gestionar accesos y rondas de seguridad en diferentes instalaciones desde una única plataforma centralizada.

2. Mejoras en la Seguridad

- **Autenticación Multifactor (MFA)**: Implementación de métodos adicionales de autenticación para garantizar un acceso más seguro, como contraseñas dinámicas o tokens físicos.
- **Cifrado Avanzado**: Actualización de protocolos de cifrado a estándares más avanzados para proteger los datos a medida que evolucionan las amenazas de seguridad cibernética.

3. Optimización del Rendimiento

- **Almacenamiento en la Nube**: Migración parcial o total de los datos de acceso y registros de rondines a soluciones en la nube para mejorar la accesibilidad y la escalabilidad.
- **Soporte para Dispositivos IoT de Nueva Generación**: Actualización de la compatibilidad con dispositivos IoT emergentes que ofrezcan mejores capacidades y menor consumo energético.

4. Análisis de Datos e Informes

- **Inteligencia Artificial y Machine Learning**: Incorporación de análisis de datos basados en IA para identificar patrones de acceso anómalos o predecir posibles incidentes de seguridad.
- **Informes Personalizables**: Desarrollo de herramientas para generar informes más detallados y adaptados a las necesidades específicas de cada instalación.

5. Experiencia de Usuario

- **Interfaz Multilenguaje**: Ampliación del soporte a múltiples idiomas para adaptarse a usuarios internacionales.
- **Acceso Offline**: Desarrollo de funcionalidades que permitan a los dispositivos IoT y aplicaciones móviles operar sin conexión temporal y sincronizar los datos una vez que se restablezca la conectividad.

6. Regulaciones y Cumplimiento

- **Actualizaciones para Cumplir Nuevas Normativas**: Adaptación continua a las regulaciones emergentes relacionadas con la privacidad y seguridad de datos en diferentes regiones.

3 Requisitos específicos

Requisitos funcionales

Número de requisito	**RF1**
Nombre de requisito	Gestion de usuarios
Descripcion	El sistema debe permitir a los administradores gestionar la información de los usuarios registrados, proporcionando opciones para: • Crear un nuevo usuario. • Modificar los datos existentes de un usuario, como nombre, rol, y permisos de acceso. • Eliminar un usuario del sistema. • Consultar la lista de usuarios registrados y sus detalles asociados.
Tipo	☐ Requisito ☐ Restricción
Fuente del requisito	Equipo de diseno y cliente industrial
Prioridad del requisito	☐ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional

Número de requisito	**RF6**
Nombre de requisito	Gestión de Rondines
Descripcion	El sistema debe gestionar las actividades de rondines asignadas al personal de seguridad, permitiendo: 1. Asignar rondines con puntos de control predefinidos a los guardias de seguridad. 2. Registrar automáticamente la validación de cada punto de control a través de dispositivos IoT (como lectores RFID). 3. Generar alertas si algún punto de control no es completado dentro del tiempo asignado. 4. Proporcionar un resumen del estado de cumplimiento de los rondines, incluyendo fechas, horas, y cualquier anomalía detectada.
Tipo	☐ Requisito ☐ Restricción
Fuente del requisito	Equipo de diseno y cliente industrial
Prioridad del requisito	☐ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional

Requisitos no funcionales

Número de requisito	**RF2**
Nombre de requisito	Gestión de Rondines
Descripcion	El sistema debe gestionar las actividades de rondines asignadas al personal de seguridad, permitiendo: 1. Asignar rondines con puntos de control predefinidos a los guardias de seguridad. 2. Registrar automáticamente la validación de cada punto de control a través de dispositivos IoT (como lectores RFID). 3. Generar alertas si algún punto de control no es completado dentro del tiempo asignado. 4. Proporcionar un resumen del estado de cumplimiento de los rondines, incluyendo fechas, horas, y cualquier anomalía detectada.
Tipo	☐ Requisito ☐ Restricción
Fuente del requisito	Equipo de diseno y cliente industrial
Prioridad del requisito	☐ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional

3.1 Requisitos comunes de los interfaces

- La interfaz web deberá ser intuitiva, con un diseño basado en material design, usando colores neutros y elementos visuales mínimos para evitar distracciones.
- Pantalla principal: Panel de administración con accesos rápidos a las opciones de usuarios, rondines y dispositivos IoT.
- La aplicación móvil deberá ofrecer un diseño adaptable, con acceso rápido a las rondas asignadas y alertas activas.
- Requisitos de accesibilidad: Soporte para lectores de pantalla y textos ampliables para usuarios con discapacidades visuales.

3.1.1 Interfaces de usuario

La interfaz web deberá ser intuitiva, con un diseño basado en material design, usando colores neutros y elementos visuales mínimos para evitar distracciones.

Pantalla principal: Panel de administración con accesos rápidos a las opciones de usuarios, rondines y dispositivos IoT.

La aplicación móvil deberá ofrecer un diseño adaptable, con acceso rápido a las rondas asignadas y alertas activas.

Requisitos de accesibilidad: Soporte para lectores de pantalla y textos ampliables para usuarios con discapacidades visuales.

3.1.2 Interfaces de hardware

Dispositivos IoT: Lectores RFID y sensores biométricos deben conectarse a la red principal mediante protocolos estándar como MQTT o HTTP.

Configuración: Los dispositivos deben ser inicializados desde la interfaz web con opciones de configuración como IP estática o DHCP.

Señales: Indicadores LED y alertas sonoras para accesos aprobados (verde) o denegados (rojo).

3.1.3 Interfaces de software

- **Sistema Operativo**: Windows y linux
- **Base de Datos**: Uso de MySql para almacenamiento de datos de acceso y rondines.
- Integración con API REST: Permite la interoperabilidad con otros sistemas industriales.

3.1.4 Interfaces de comunicación

- **La interfaz web deberá ser intuitiva**, con un diseño basado en material design, usando colores neutros y elementos visuales mínimos para evitar distracciones.
- **Pantalla principal**: Panel de administración con accesos rápidos a las opciones de usuarios, rondines y dispositivos IoT.
- **La aplicación móvil deberá ofrecer un diseño adaptable**, con acceso rápido a las rondas asignadas y alertas activas.
- **Requisitos de accesibilidad:** Soporte para lectores de pantalla y textos ampliables para usuarios con discapacidades visuales.

3.2 Requisitos funcionales

3.2.1 Requisito Funcional 1: Gestión de Usuarios

El sistema debe permitir registrar, actualizar y eliminar usuarios desde la interfaz web.

Validaciones: Evitar duplicados en registros mediante identificación única.

Relación: Cada usuario está vinculado a roles y permisos específicos.

3.2.2 Requisito Funcional 2: Control de Acceso

Validar credenciales en tiempo real utilizando dispositivos IoT.

Generar registros de acceso que incluyan fecha, hora, usuario y área.

3.2.3 Requisito Funcional 3: Gestión de Rondines

Asignar puntos de control a los guardias y registrar su cumplimiento automático.

Alertar sobre puntos incompletos o fuera del tiempo estimado.

3.2.4 Requisito Funcional 4: Generación de Reportes

Generar reportes detallados de accesos y rondines en formato PDF.

Permitir filtrar reportes por usuario, área o rango de fechas.

3.3 Requisitos no funcionales

3.3.1 Requisitos de rendimiento

- El sistema debe procesar solicitudes en menos de 2 segundos para garantizar fluidez en accesos.
- Soportar hasta 200 usuarios simultáneos en la interfaz web sin degradación del rendimiento.

3.3.2 Seguridad

- Cifrado de datos sensible con AES-256.
- Requerir autenticación de dos factores (2FA) para administradores.

3.3.3 Fiabilidad

- Garantizar un tiempo medio entre fallo de al menos 100 días en dispositivos IoT.

3.3.4 Disponibilidad

- 24 horas del dia.

3.3.5 Mantenibilidad

- Ofrecer herramientas de diagnóstico en la interfaz para verificar el estado de dispositivos IoT.
- Actualizaciones automáticas del software mediante un servidor central.

3.3.6 Portabilidad

- Compatible con navegadores modernos y aplicaciones móviles en Android e iOS.
- Uso de Docker para facilitar la migración entre servidores.

3.4 Otros requisitos

- **Requisitos Legales**: Cumplir con la normativa de protección de datos personales.
- **Requisitos Políticos**: El sistema debe permitir la exportación de datos para auditorías gubernamentales o internas.

4 Apéndices

- **Diagramas del Sistema:** // DER, MR, DICCIONARIO DATOS, CASOS DE USO, ETC.
- **Referencias Técnicas:** // Documentacion utilizada

5 Desarrollo del código:

En el desarrollo del backend del proyecto, por medio de PHP se accedió a la información de la base de datos por medio de métodos GET y SET para mandar consultar y actualizar la información. Estos métodos fueron separados en diferentes archivos para controlar el cómo se accesa a la información dependiendo del contexto que requiera la aplicación. Por ejemplo, el archivo “accesos.php” tiene los métodos para actualizar y subir nueva información para que desde el frontend solo pueda acceder a los respectivos métodos de la tabla “accesos” en la BD. De esta manera evitamos filtrado de información y el modificar información de otras tablas por error.

Estos métodos llaman a la información asignada y los codifican en formato JSON para posteriormente ser llamados desde React e iterarlos como una API.

Al realizar un cambio en la información de la BD, React envía la solicitud por medio de un formulario a PHP con los datos en formato JSON, en el backend, PHP se encarga de decodificar esos datos, procesar la solicitud y enviarlos a la BD.

Al usar esta estructura para manejar la información podemos crear un frontend más limpio usando en su mayoría componentes.

Métodos:

Accesos.php:

```
$method = $_SERVER['REQUEST_METHOD'];

switch ($method) {
    case 'GET':
        getAccesos();
        break;
    case 'POST':
        createAcceso();
        break;
    case 'PUT':
        updateAcceso();
        break;
    case 'DELETE':
        deleteAcceso();
        break;
    default:
        echo json_encode(["message" => "Método no permitido"]);
        break;
}

function getAccesos() {
    global $pdo;
    $stmt = $pdo->query("SELECT a.ID, a.estado, a.hora_acceso, a.id_registro, a.codigo_area,
                            la.tipo AS nombre_registro, ar.nombre
                        FROM acceso a
                        JOIN log_acceso la ON a.id_registro = la.id_registro
                        JOIN area ar ON a.codigo_area = ar.codigo_area");
    $accesos = $stmt->fetchAll(PDO::FETCH_ASSOC);
    echo json_encode($accesos);
}

function createAcceso() {
    global $pdo;
    $data = json_decode(file_get_contents("php://input"));
    $estado = $data->estado;
    $hora_acceso = $data->hora_acceso;
    $id_registro = $data->id_registro;
    $codigo_area = $data->codigo_area;
```

```
    $stmt = $pdo->prepare("INSERT INTO acceso (estado, hora_acceso, id_registro, codigo_area)
                           VALUES (?, ?, ?, ?)");
    $stmt->execute([$estado, $hora_acceso, $id_registro, $codigo_area]);
    echo json_encode(["message" => "Acceso creado"]);
}

function updateAcceso() {
    global $pdo;
    $data = json_decode(file_get_contents("php://input"));
    $id = $data->id;
    $estado = $data->estado;
    $hora_acceso = $data->hora_acceso;
    $id_registro = $data->id_registro;
    $codigo_area = $data->codigo_area;

    $stmt = $pdo->prepare("UPDATE acceso
                           SET estado = ?, hora_acceso = ?, id_registro = ?, codigo_area = ?
                           WHERE ID = ?");
    $stmt->execute([$estado, $hora_acceso, $id_registro, $codigo_area, $id]);
    echo json_encode(["message" => "Acceso actualizado"]);
}

function deleteAcceso() {
    global $pdo;
    $data = json_decode(file_get_contents("php://input"));
    $id = $data->id;

    $stmt = $pdo->prepare("DELETE FROM acceso WHERE ID = ?");
    $stmt->execute([$id]);
    echo json_encode(["message" => "Acceso eliminado"]);
}
?>
```

alertas.php:

```
$method = $_SERVER['REQUEST_METHOD'];

switch($method) {
    case 'GET':
        getAlertas();
        break;
    case 'POST':
        createAlerta();
        break;
    case 'PUT':
        updateAlerta();
        break;
    case 'DELETE':
        deleteAlerta();
        break;
    default:
        echo json_encode(["message" => "Método no permitido"]);
        break;
}

function getAlertas() {
    global $pdo;
    $stmt = $pdo->query("SELECT * FROM alerta");
    $alertas = $stmt->fetchAll(PDO::FETCH_ASSOC);
    echo json_encode($alertas);
}

function createAlerta() {
    global $pdo;
    $data = json_decode(file_get_contents("php://input"));
    $descripcion = $data->descripcion;
    $id_registro = $data->id_registro;

    $stmt = $pdo->prepare("INSERT INTO alerta (descripcion, id_registro) VALUES (?, ?)");
    $stmt->execute([$descripcion, $id_registro]);
    echo json_encode(["message" => "Alerta creada"]);
}

function updateAlerta() {
    global $pdo;
```

```
    $data = json_decode(file_get_contents("php://input"));
    $id = $data->id;
    $descripcion = $data->descripcion;
    $id_registro = $data->id_registro;

    $stmt = $pdo->prepare("UPDATE alerta SET descripcion = ?, id_registro = ? WHERE id = ?");
    $stmt->execute([$descripcion, $id_registro, $id]);
    echo json_encode(["message" => "Alerta actualizada"]);
}

function deleteAlerta() {
    global $pdo;
    $data = json_decode(file_get_contents("php://input"));
    $id = $data->id;

    $stmt = $pdo->prepare("DELETE FROM alerta WHERE id = ?");
    $stmt->execute([$id]);
    echo json_encode(["message" => "Alerta eliminada"]);
}
```

areas.php

```
$method = $_SERVER['REQUEST_METHOD'];

switch($method) {
    case 'GET':
        getAreas();
        break;
    case 'POST':
        createArea();
        break;
    case 'PUT':
        updateArea();
        break;
    case 'DELETE':
        deleteArea();
        break;
    default:
        echo json_encode(["message" => "Método no permitido"]);
        break;
}

function getAreas() {
    global $pdo;
    $stmt = $pdo->query("SELECT * FROM area");
    $areas = $stmt->fetchAll(PDO::FETCH_ASSOC);
    echo json_encode($areas);
}

function createArea() {
    global $pdo;
    $data = json_decode(file_get_contents("php://input"));
    $num_accesos = $data->num_accesos;
    $nombre = $data->nombre;
    $estado = $data->estado;
    $codigo_ruta = $data->codigo_ruta;

    $stmt = $pdo->prepare("INSERT INTO area (num_accesos, nombre, estado,
codigo_ruta) VALUES (?, ?, ?, ?)");
    $stmt->execute([$num_accesos, $nombre, $estado, $codigo_ruta]);
    echo json_encode(["message" => "Área creada"]);
}
```

```
function updateArea() {
    global $pdo;
    $data = json_decode(file_get_contents("php://input"));
    $codigo_area = $data->codigo_area;
    $num_accesos = $data->num_accesos;
    $nombre = $data->nombre;
    $estado = $data->estado;
    $codigo_ruta = $data->codigo_ruta;

    $stmt = $pdo->prepare("UPDATE area SET num_accesos = ?, nombre = ?, estado = ?,
codigo_ruta = ? WHERE codigo_area = ?");
    $stmt->execute([$num_accesos, $nombre, $estado, $codigo_ruta, $codigo_area]);
    echo json_encode(["message" => "Área actualizada"]);
}

function deleteArea() {
    global $pdo;
    $data = json_decode(file_get_contents("php://input"));
    $codigo_area = $data->codigo_area;

    $stmt = $pdo->prepare("DELETE FROM area WHERE codigo_area = ?");
    $stmt->execute([$codigo_area]);
    echo json_encode(["message" => "Área eliminada"]);
}
```

dispositivos.php

```
$method = $_SERVER['REQUEST_METHOD'];

switch($method) {
    case 'GET':
        getDispositivos();
        break;
    case 'POST':
        createDispositivo();
        break;
    case 'PUT':
        updateDispositivo();
        break;
    case 'DELETE':
        deleteDispositivo();
        break;
    default:
        echo json_encode(["message" => "Método no permitido"]);
        break;
}

function getDispositivos() {
    global $pdo;
    $stmt = $pdo->query("SELECT * FROM dispositivo");
    $dispositivos = $stmt->fetchAll(PDO::FETCH_ASSOC);
    echo json_encode($dispositivos);
}

function createDispositivo() {
    global $pdo;
    $data = json_decode(file_get_contents("php://input"));
    $tipo = $data->tipo;
    $estado = $data->estado;
    $ubicacion = $data->ubicacion;
    $codigo_area = $data->codigo_area;

    $stmt = $pdo->prepare("INSERT INTO dispositivo (tipo, estado, ubicacion, codigo_area)
VALUES (?, ?, ?, ?)");
    $stmt->execute([$tipo, $estado, $ubicacion, $codigo_area]);
    echo json_encode(["message" => "Dispositivo creado"]);
}
```

```
function updateDispositivo() {
    global $pdo;
    $data = json_decode(file_get_contents("php://input"));
    $codigo = $data->codigo;
    $tipo = $data->tipo;
    $estado = $data->estado;
    $ubicacion = $data->ubicacion;
    $codigo_area = $data->codigo_area;

    $stmt = $pdo->prepare("UPDATE dispositivo SET tipo = ?, estado = ?, ubicacion = ?,
codigo_area = ? WHERE codigo = ?");
    $stmt->execute([$tipo, $estado, $ubicacion, $codigo_area, $codigo]);
    echo json_encode(["message" => "Dispositivo actualizado"]);
}

function deleteDispositivo() {
    global $pdo;
    $data = json_decode(file_get_contents("php://input"));
    $codigo = $data->codigo;

    $stmt = $pdo->prepare("DELETE FROM dispositivo WHERE codigo = ?");
    $stmt->execute([$codigo]);
    echo json_encode(["message" => "Dispositivo eliminado"]);
}
```

empleados_accesos.php

```
$method = $_SERVER['REQUEST_METHOD'];

switch($method) {
    case 'GET':
        getEmpleadoAccesos();
        break;
    case 'POST':
        createEmpleadoAcceso();
        break;
    case 'PUT':
        updateEmpleadoAcceso();
        break;
    case 'DELETE':
        deleteEmpleadoAcceso();
        break;
    default:
        echo json_encode(["message" => "Método no permitido"]);
        break;
}

function getEmpleadoAccesos() {
    global $pdo;
    $stmt = $pdo->query("SELECT * FROM empleado_acceso");
    $empleadoAccesos = $stmt->fetchAll(PDO::FETCH_ASSOC);
    echo json_encode($empleadoAccesos);
}

function createEmpleadoAcceso() {
    global $pdo;
    $data = json_decode(file_get_contents("php://input"));
    $id_empleado = $data->id_empleado;
    $id_acceso = $data->id_acceso;

    $stmt = $pdo->prepare("INSERT INTO empleado_acceso (id_empleado, id_acceso)
VALUES (?, ?)");
    $stmt->execute([$id_empleado, $id_acceso]);
    echo json_encode(["message" => "Empleado acceso creado"]);
}

function updateEmpleadoAcceso() {
```

```
    global $pdo;
    $data = json_decode(file_get_contents("php://input"));
    $numero = $data->numero;
    $id_empleado = $data->id_empleado;
    $id_acceso = $data->id_acceso;

    $stmt = $pdo->prepare("UPDATE empleado_acceso SET id_empleado = ?, id_acceso =
? WHERE numero = ?");
    $stmt->execute([$id_empleado, $id_acceso, $numero]);
    echo json_encode(["message" => "Empleado acceso actualizado"]);
}

function deleteEmpleadoAcceso() {
    global $pdo;
    $data = json_decode(file_get_contents("php://input"));
    $numero = $data->numero;

    $stmt = $pdo->prepare("DELETE FROM empleado_acceso WHERE numero = ?");
    $stmt->execute([$numero]);
    echo json_encode(["message" => "Empleado acceso eliminado"]);
}
```

empleados.php

```
$method = $_SERVER['REQUEST_METHOD'];

switch ($method) {
    case 'GET':
        getEmpleados();
        break;
    case 'POST':
        createEmpleado();
        break;
    case 'PUT':
        updateEmpleado();
        break;
    case 'DELETE':
        deleteEmpleado();
        break;
    default:
        echo json_encode(["message" => "Método no permitido"]);
        break;
}

function getEmpleados() {
    global $pdo;
    $stmt = $pdo->query("SELECT e.ID, e.nombre, e.apellido_paterno, e.apellido_materno,
e.codigo_puesto, p.nombre_puesto AS puesto FROM empleado e LEFT JOIN puesto p ON
e.codigo_puesto = p.codigo");
    $empleados = $stmt->fetchAll(PDO::FETCH_ASSOC);
    echo json_encode($empleados);
}

function createEmpleado() {
    global $pdo;
    $data = json_decode(file_get_contents("php://input"));
    $nombre = $data->nombre;
    $apellido_paterno = $data->apellido_paterno;
    $apellido_materno = $data->apellido_materno;
    $codigo_puesto = $data->codigo_puesto;

    $stmt = $pdo->prepare("INSERT INTO empleado (nombre, apellido_paterno,
apellido_materno, codigo_puesto) VALUES (?, ?, ?, ?)");
    $stmt->execute([$nombre, $apellido_paterno, $apellido_materno, $codigo_puesto]);
    echo json_encode(["message" => "Empleado creado"]);
```

```
}

function updateEmpleado() {
    global $pdo;
    $data = json_decode(file_get_contents("php://input"));
    $id = $data->id;
    $nombre = $data->nombre;
    $apellido_paterno = $data->apellido_paterno;
    $apellido_materno = $data->apellido_materno;
    $codigo_puesto = $data->codigo_puesto;

    $stmt = $pdo->prepare("UPDATE empleado SET nombre = ?, apellido_paterno = ?,
apellido_materno = ?, codigo_puesto = ? WHERE ID = ?");
    $stmt->execute([$nombre, $apellido_paterno, $apellido_materno, $codigo_puesto, $id]);
    echo json_encode(["message" => "Empleado actualizado"]);
}

function deleteEmpleado() {
    global $pdo;
    $data = json_decode(file_get_contents("php://input"));
    $id = $data->id;

    $stmt = $pdo->prepare("DELETE FROM empleado WHERE ID = ?");
    $stmt->execute([$id]);
    echo json_encode(["message" => "Empleado eliminado"]);
}
```

Guardias_seguridad.php

```
$method = $_SERVER['REQUEST_METHOD'];

switch($method) {
    case 'GET':
        getGuardias();
        break;
    case 'POST':
        createGuardia();
        break;
    case 'PUT':
        updateGuardia();
        break;
    case 'DELETE':
        deleteGuardia();
        break;
    default:
        echo json_encode(["message" => "Método no permitido"]);
        break;
}

function getGuardias() {
    global $pdo;
    $stmt = $pdo->query("SELECT * FROM guardia_seguridad");
    $guardias = $stmt->fetchAll(PDO::FETCH_ASSOC);
    echo json_encode($guardias);
}

function createGuardia() {
    global $pdo;
    $data = json_decode(file_get_contents("php://input"));
    $nombre = $data->nombre;
    $apellido_paterno = $data->apellido_paterno;
    $apellido_materno = $data->apellido_materno;

    $stmt = $pdo->prepare("INSERT INTO guardia_seguridad (nombre, apellido_paterno,
apellido_materno) VALUES (?, ?, ?)");
    $stmt->execute([$nombre, $apellido_paterno, $apellido_materno]);
    echo json_encode(["message" => "Guardia creado"]);
}
```

```
function updateGuardia() {
    global $pdo;
    $data = json_decode(file_get_contents("php://input"));
    $id = $data->id;
    $nombre = $data->nombre;
    $apellido_paterno = $data->apellido_paterno;
    $apellido_materno = $data->apellido_materno;

    $stmt = $pdo->prepare("UPDATE guardia_seguridad SET nombre = ?, apellido_paterno
= ?, apellido_materno = ? WHERE ID = ?");
    $stmt->execute([$nombre, $apellido_paterno, $apellido_materno, $id]);
    echo json_encode(["message" => "Guardia actualizado"]);
}

function deleteGuardia() {
    global $pdo;
    $data = json_decode(file_get_contents("php://input"));
    $id = $data->id;

    $stmt = $pdo->prepare("DELETE FROM guardia_seguridad WHERE ID = ?");
    $stmt->execute([$id]);
    echo json_encode(["message" => "Guardia eliminado"]);
}
```

Log_accesos.php:

```
$method = $_SERVER['REQUEST_METHOD'];

switch($method) {
    case 'GET':
        getLogs();
        break;
    case 'POST':
        createLog();
        break;
    case 'PUT':
        updateLog();
        break;
    case 'DELETE':
        deleteLog();
        break;
    default:
        echo json_encode(["message" => "Método no permitido"]);
        break;
}

function getLogs() {
    global $pdo;
    $stmt = $pdo->query("SELECT * FROM log_acceso");
    $logs = $stmt->fetchAll(PDO::FETCH_ASSOC);
    echo json_encode($logs);
}

function createLog() {
    global $pdo;
    $data = json_decode(file_get_contents("php://input"));
    $tipo = $data->tipo;
    $fecha = $data->fecha;
    $hora = $data->hora;
    $detalles = $data->detalles;

    $stmt = $pdo->prepare("INSERT INTO log_acceso (tipo, fecha, hora, detalles) VALUES
(?, ?, ?, ?)");
    $stmt->execute([$tipo, $fecha, $hora, $detalles]);
    echo json_encode(["message" => "Log creado"]);
}
```

```
function updateLog() {
    global $pdo;
    $data = json_decode(file_get_contents("php://input"));
    $id = $data->id;
    $tipo = $data->tipo;
    $fecha = $data->fecha;
    $hora = $data->hora;
    $detalles = $data->detalles;

    $stmt = $pdo->prepare("UPDATE log_acceso SET tipo = ?, fecha = ?, hora = ?, detalles
= ? WHERE id_registro = ?");
    $stmt->execute([$tipo, $fecha, $hora, $detalles, $id]);
    echo json_encode(["message" => "Log actualizado"]);
}

function deleteLog() {
    global $pdo;
    $data = json_decode(file_get_contents("php://input"));
    $id = $data->id;

    $stmt = $pdo->prepare("DELETE FROM log_acceso WHERE id_registro = ?");
    $stmt->execute([$id]);
    echo json_encode(["message" => "Log eliminado"]);
}
```

Puesto_acceso.php:

```
$method = $_SERVER['REQUEST_METHOD'];

switch($method) {
    case 'GET':
        getPuestosAcceso();
        break;
    case 'POST':
        createPuestoAcceso();
        break;
    case 'PUT':
        updatePuestoAcceso();
        break;
    case 'DELETE':
        deletePuestoAcceso();
        break;
    default:
        echo json_encode(["message" => "Método no permitido"]);
        break;
}

function getPuestosAcceso() {
    global $pdo;
    $stmt = $pdo->query("SELECT * FROM puesto_acceso");
    $puestos_acceso = $stmt->fetchAll(PDO::FETCH_ASSOC);
    echo json_encode($puestos_acceso);
}

function createPuestoAcceso() {
    global $pdo;
    $data = json_decode(file_get_contents("php://input"));
    $nivel_acceso = $data->nivel_acceso;
    $puesto_codigo = $data->Puesto_codigo;
    $acceso_id = $data->Acceso_id;

    $stmt = $pdo->prepare("INSERT INTO puesto_acceso (nivel_acceso, Puesto_codigo,
Acceso_id) VALUES (?, ?, ?)");
    $stmt->execute([$nivel_acceso, $puesto_codigo, $acceso_id]);
    echo json_encode(["message" => "Puesto de acceso creado"]);
}
```

```
function updatePuestoAcceso() {
    global $pdo;
    $data = json_decode(file_get_contents("php://input"));
    $codigo = $data->codigo;
    $nivel_acceso = $data->nivel_acceso;
    $puesto_codigo = $data->Puesto_codigo;
    $acceso_id = $data->Acceso_id;

    $stmt = $pdo->prepare("UPDATE puesto_acceso SET nivel_acceso = ?, Puesto_codigo
= ?, Acceso_id = ? WHERE codigo = ?");
    $stmt->execute([$nivel_acceso, $puesto_codigo, $acceso_id, $codigo]);
    echo json_encode(["message" => "Puesto de acceso actualizado"]);
}

function deletePuestoAcceso() {
    global $pdo;
    $data = json_decode(file_get_contents("php://input"));
    $codigo = $data->codigo;

    $stmt = $pdo->prepare("DELETE FROM puesto_acceso WHERE codigo = ?");
    $stmt->execute([$codigo]);
    echo json_encode(["message" => "Puesto de acceso eliminado"]);
}
```

Puestos.php:

```
$method = $_SERVER['REQUEST_METHOD'];

switch ($method) {
    case 'GET':
        getPuestos();
        break;
    case 'POST':
        createPuesto();
        break;
    case 'PUT':
        updatePuesto();
        break;
    case 'DELETE':
        deletePuesto();
        break;
    default:
        echo json_encode(["message" => "Método no permitido"]);
        break;
}

function getPuestos() {
    global $pdo;
    $stmt = $pdo->query("SELECT * FROM puesto");
    $puestos = $stmt->fetchAll(PDO::FETCH_ASSOC);
    echo json_encode($puestos);
}

function createPuesto() {
    global $pdo;
    $data = json_decode(file_get_contents("php://input"));
    $descripcion = $data->descripcion;
    $nombre_puesto = $data->nombre_puesto;

    $stmt = $pdo->prepare("INSERT INTO puesto (descripcion, nombre_puesto) VALUES
(?, ?)");
    $stmt->execute([$descripcion, $nombre_puesto]);
    echo json_encode(["message" => "Puesto creado"]);
}

function updatePuesto() {
```

```
    global $pdo;
    $data = json_decode(file_get_contents("php://input"));
    $codigo = $data->codigo;
    $descripcion = $data->descripcion;
    $nombre_puesto = $data->nombre_puesto;

    $stmt = $pdo->prepare("UPDATE puesto SET descripcion = ?, nombre_puesto = ?
WHERE codigo = ?");
    $stmt->execute([$descripcion, $nombre_puesto, $codigo]);
    echo json_encode(["message" => "Puesto actualizado"]);
}

function deletePuesto() {
    global $pdo;
    $data = json_decode(file_get_contents("php://input"));
    $codigo = $data->codigo;

    $stmt = $pdo->prepare("DELETE FROM puesto WHERE codigo = ?");
    $stmt->execute([$codigo]);
    echo json_encode(["message" => "Puesto eliminado"]);
}
```

Reporte_accesos.php:

```
$method = $_SERVER['REQUEST_METHOD'];

switch($method) {
    case 'GET':
        getReportesAcceso();
        break;
    case 'POST':
        createReporteAcceso();
        break;
    case 'PUT':
        updateReporteAcceso();
        break;
    case 'DELETE':
        deleteReporteAcceso();
        break;
    default:
        echo json_encode(["message" => "Método no permitido"]);
        break;
}

function getReportesAcceso() {
    global $pdo;
    $stmt = $pdo->query("SELECT * FROM reporte_acceso");
    $reportesAcceso = $stmt->fetchAll(PDO::FETCH_ASSOC);
    echo json_encode($reportesAcceso);
}

function createReporteAcceso() {
    global $pdo;
    $data = json_decode(file_get_contents("php://input"));
    $id_reporte = $data->id_reporte;
    $id_registro = $data->id_registro;

    $stmt = $pdo->prepare("INSERT INTO reporte_acceso (id_reporte, id_registro) VALUES
(?, ?)");
    $stmt->execute([$id_reporte, $id_registro]);
    echo json_encode(["message" => "Reporte de acceso creado"]);
}

function updateReporteAcceso() {
```

```
    global $pdo;
    $data = json_decode(file_get_contents("php://input"));
    $numero = $data->numero;
    $id_reporte = $data->id_reporte;
    $id_registro = $data->id_registro;

    $stmt = $pdo->prepare("UPDATE reporte_acceso SET id_reporte = ?, id_registro = ?
WHERE numero = ?");
    $stmt->execute([$id_reporte, $id_registro, $numero]);
    echo json_encode(["message" => "Reporte de acceso actualizado"]);
}

function deleteReporteAcceso() {
    global $pdo;
    $data = json_decode(file_get_contents("php://input"));
    $numero = $data->numero;

    $stmt = $pdo->prepare("DELETE FROM reporte_acceso WHERE numero = ?");
    $stmt->execute([$numero]);
    echo json_encode(["message" => "Reporte de acceso eliminado"]);
}
```

Reportes.php

```
$method = $_SERVER['REQUEST_METHOD'];

switch($method) {
    case 'GET':
        getReportes();
        break;
    case 'POST':
        createReporte();
        break;
    case 'PUT':
        updateReporte();
        break;
    case 'DELETE':
        deleteReporte();
        break;
    default:
        echo json_encode(["message" => "Método no permitido"]);
        break;
}

function getReportes() {
    global $pdo;
    $stmt = $pdo->query("SELECT * FROM reporte");
    $reportes = $stmt->fetchAll(PDO::FETCH_ASSOC);
    echo json_encode($reportes);
}

function createReporte() {
    global $pdo;
    $data = json_decode(file_get_contents("php://input"));
    $tipo = $data->tipo;
    $fecha = $data->fecha;
    $codigo_ronda = $data->codigo_ronda;

    $stmt = $pdo->prepare("INSERT INTO reporte (tipo, fecha, codigo_ronda) VALUES (?,
?, ?)");
    $stmt->execute([$tipo, $fecha, $codigo_ronda]);
    echo json_encode(["message" => "Reporte creado"]);
}
```

```
function updateReporte() {
    global $pdo;
    $data = json_decode(file_get_contents("php://input"));
    $id = $data->id;
    $tipo = $data->tipo;
    $fecha = $data->fecha;
    $codigo_ronda = $data->codigo_ronda;

    $stmt = $pdo->prepare("UPDATE reporte SET tipo = ?, fecha = ?, codigo_ronda = ?
WHERE id = ?");
    $stmt->execute([$tipo, $fecha, $codigo_ronda, $id]);
    echo json_encode(["message" => "Reporte actualizado"]);
}

function deleteReporte() {
    global $pdo;
    $data = json_decode(file_get_contents("php://input"));
    $id = $data->id;

    $stmt = $pdo->prepare("DELETE FROM reporte WHERE id = ?");
    $stmt->execute([$id]);
    echo json_encode(["message" => "Reporte eliminado"]);
}
```

Rfid.php

```
$method = $_SERVER['REQUEST_METHOD'];

switch($method) {
    case 'GET':
        getRFID();
        break;
    case 'POST':
        createRFID();
        break;
    case 'PUT':
        updateRFID();
        break;
    case 'DELETE':
        deleteRFID();
        break;
    default:
        echo json_encode(["message" => "Método no permitido"]);
        break;
}

function getRFID() {
    global $pdo;
    $stmt = $pdo->query("SELECT * FROM rfid");
    $rfid = $stmt->fetchAll(PDO::FETCH_ASSOC);
    echo json_encode($rfid);
}

function createRFID() {
    global $pdo;
    $data = json_decode(file_get_contents("php://input"));
    $estado = $data->estado;
    $codigo_rfid = $data->codigo_rfid;
    $fecha_registro = $data->fecha_registro;
    $tipo = $data->tipo;
    $id_empleado = $data->id_empleado;

    $stmt = $pdo->prepare("INSERT INTO rfid (estado, codigo_rfid, fecha_registro, tipo,
id_empleado) VALUES (?, ?, ?, ?, ?)");
    $stmt->execute([$estado, $codigo_rfid, $fecha_registro, $tipo, $id_empleado]);
    echo json_encode(["message" => "RFID creado"]);
```

```
}

function updateRFID() {
    global $pdo;
    $data = json_decode(file_get_contents("php://input"));
    $id_RFID = $data->id_RFID;
    $estado = $data->estado;
    $codigo_rfid = $data->codigo_rfid;
    $fecha_registro = $data->fecha_registro;
    $tipo = $data->tipo;
    $id_empleado = $data->id_empleado;

    $stmt = $pdo->prepare("UPDATE rfid SET estado = ?, codigo_rfid = ?, fecha_registro = ?, tipo = ?, id_empleado = ? WHERE id_RFID = ?");
    $stmt->execute([$estado, $codigo_rfid, $fecha_registro, $tipo, $id_empleado, $id_RFID]);
    echo json_encode(["message" => "RFID actualizado"]);
}

function deleteRFID() {
    global $pdo;
    $data = json_decode(file_get_contents("php://input"));
    $id_RFID = $data->id_RFID;

    $stmt = $pdo->prepare("DELETE FROM rfid WHERE id_RFID = ?");
    $stmt->execute([$id_RFID]);
    echo json_encode(["message" => "RFID eliminado"]);
}
```

Ronda_guardia.php:

```
$method = $_SERVER['REQUEST_METHOD'];

switch($method) {
    case 'GET':
        getRondaGuardia();
        break;
    case 'POST':
        createRondaGuardia();
        break;
    case 'PUT':
        updateRondaGuardia();
        break;
    case 'DELETE':
        deleteRondaGuardia();
        break;
    default:
        echo json_encode(["message" => "Método no permitido"]);
        break;
}

function getRondaGuardia() {
    global $pdo;
    $stmt = $pdo->query("SELECT * FROM ronda_guardia");
    $rondasGuardias = $stmt->fetchAll(PDO::FETCH_ASSOC);
    echo json_encode($rondasGuardias);
}

function createRondaGuardia() {
    global $pdo;
    $data = json_decode(file_get_contents("php://input"));
    $codigo_ronda = $data->codigo_ronda;
    $id_guardia = $data->id_guardia;

    $stmt = $pdo->prepare("INSERT INTO ronda_guardia (codigo_ronda, id_guardia)
VALUES (?, ?)");
    $stmt->execute([$codigo_ronda, $id_guardia]);
    echo json_encode(["message" => "Ronda-Guardia asignada"]);
}
```

```
function updateRondaGuardia() {
    global $pdo;
    $data = json_decode(file_get_contents("php://input"));
    $numero = $data->numero;
    $codigo_ronda = $data->codigo_ronda;
    $id_guardia = $data->id_guardia;

    $stmt = $pdo->prepare("UPDATE ronda_guardia SET codigo_ronda = ?, id_guardia = ?
WHERE numero = ?");
    $stmt->execute([$codigo_ronda, $id_guardia, $numero]);
    echo json_encode(["message" => "Ronda-Guardia actualizada"]);
}

function deleteRondaGuardia() {
    global $pdo;
    $data = json_decode(file_get_contents("php://input"));
    $numero = $data->numero;

    $stmt = $pdo->prepare("DELETE FROM ronda_guardia WHERE numero = ?");
    $stmt->execute([$numero]);
    echo json_encode(["message" => "Ronda-Guardia eliminada"]);
}
```

Rondas.php

```
$method = $_SERVER['REQUEST_METHOD'];

switch($method) {
    case 'GET':
        getRonda();
        break;
    case 'POST':
        createRonda();
        break;
    case 'PUT':
        updateRonda();
        break;
    case 'DELETE':
        deleteRonda();
        break;
    default:
        echo json_encode(["message" => "Método no permitido"]);
        break;
}

function getRonda() {
    global $pdo;
    $stmt = $pdo->query("SELECT * FROM ronda");
    $rondas = $stmt->fetchAll(PDO::FETCH_ASSOC);
    echo json_encode($rondas);
}

function createRonda() {
    global $pdo;
    $data = json_decode(file_get_contents("php://input"));
    $hora_inicio = $data->hora_inicio;
    $hora_fin = $data->hora_fin;
    $intervalos = $data->intervalos;
    $codigo_ruta = $data->codigo_ruta;

    $stmt = $pdo->prepare("INSERT INTO ronda (hora_inicio, hora_fin, intervalos,
codigo_ruta) VALUES (?, ?, ?, ?)");
    $stmt->execute([$hora_inicio, $hora_fin, $intervalos, $codigo_ruta]);
    echo json_encode(["message" => "Ronda creada"]);
}
```

```
function updateRonda() {
    global $pdo;
    $data = json_decode(file_get_contents("php://input"));
    $codigo = $data->codigo;
    $hora_inicio = $data->hora_inicio;
    $hora_fin = $data->hora_fin;
    $intervalos = $data->intervalos;
    $codigo_ruta = $data->codigo_ruta;

    $stmt = $pdo->prepare("UPDATE ronda SET hora_inicio = ?, hora_fin = ?, intervalos =
?, codigo_ruta = ? WHERE codigo = ?");
    $stmt->execute([$hora_inicio, $hora_fin, $intervalos, $codigo_ruta, $codigo]);
    echo json_encode(["message" => "Ronda actualizada"]);
}

function deleteRonda() {
    global $pdo;
    $data = json_decode(file_get_contents("php://input"));
    $codigo = $data->codigo;

    $stmt = $pdo->prepare("DELETE FROM ronda WHERE codigo = ?");
    $stmt->execute([$codigo]);
    echo json_encode(["message" => "Ronda eliminada"]);
}
```

Rondinero.php

```
$method = $_SERVER['REQUEST_METHOD'];

switch($method) {
    case 'GET':
        getRondinero();
        break;
    case 'POST':
        createRondinero();
        break;
    case 'PUT':
        updateRondinero();
        break;
    case 'DELETE':
        deleteRondinero();
        break;
    default:
        echo json_encode(["message" => "Método no permitido"]);
        break;
}

function getRondinero() {
    global $pdo;
    $stmt = $pdo->query("SELECT * FROM rondinero");
    $rondineros = $stmt->fetchAll(PDO::FETCH_ASSOC);
    echo json_encode($rondineros);
}

function createRondinero() {
    global $pdo;
    $data = json_decode(file_get_contents("php://input"));
    $posicion = $data->posicion;
    $estado = $data->estado;
    $numero = $data->numero;
    $codigo_ruta = $data->codigo_ruta;

    $stmt = $pdo->prepare("INSERT INTO rondinero (posicion, estado, numero,
codigo_ruta) VALUES (?, ?, ?, ?)");
    $stmt->execute([$posicion, $estado, $numero, $codigo_ruta]);
    echo json_encode(["message" => "Rondinero creado"]);
}
```

```
function updateRondinero() {
    global $pdo;
    $data = json_decode(file_get_contents("php://input"));
    $ID = $data->ID;
    $posicion = $data->posicion;
    $estado = $data->estado;
    $numero = $data->numero;
    $codigo_ruta = $data->codigo_ruta;

    $stmt = $pdo->prepare("UPDATE rondinero SET posicion = ?, estado = ?, numero = ?,
codigo_ruta = ? WHERE ID = ?");
    $stmt->execute([$posicion, $estado, $numero, $codigo_ruta, $ID]);
    echo json_encode(["message" => "Rondinero actualizado"]);
}

function deleteRondinero() {
    global $pdo;
    $data = json_decode(file_get_contents("php://input"));
    $ID = $data->ID;

    $stmt = $pdo->prepare("DELETE FROM rondinero WHERE ID = ?");
    $stmt->execute([$ID]);
    echo json_encode(["message" => "Rondinero eliminado"]);
}
```

Rutas.php

```
$method = $_SERVER['REQUEST_METHOD'];

switch($method) {
    case 'GET':
        getRutas();
        break;
    case 'POST':
        createRuta();
        break;
    case 'PUT':
        updateRuta();
        break;
    case 'DELETE':
        deleteRuta();
        break;
    default:
        echo json_encode(["message" => "Método no permitido"]);
        break;
}

function getRutas() {
    global $pdo;
    $stmt = $pdo->query("SELECT * FROM ruta");
    $rutas = $stmt->fetchAll(PDO::FETCH_ASSOC);
    echo json_encode($rutas);
}

function createRuta() {
    global $pdo;
    $data = json_decode(file_get_contents("php://input"));
    $nombre = $data->nombre;

    $stmt = $pdo->prepare("INSERT INTO ruta (nombre) VALUES (?)");
    $stmt->execute([$nombre]);
    echo json_encode(["message" => "Ruta creada"]);
}

function updateRuta() {
    global $pdo;
    $data = json_decode(file_get_contents("php://input"));
```

```
    $id = $data->id;
    $nombre = $data->nombre;

    $stmt = $pdo->prepare("UPDATE ruta SET nombre = ? WHERE codigo = ?");
    $stmt->execute([$nombre, $id]);
    echo json_encode(["message" => "Ruta actualizada"]);
}

function deleteRuta() {
    global $pdo;
    $data = json_decode(file_get_contents("php://input"));
    $id = $data->id;

    $stmt = $pdo->prepare("DELETE FROM ruta WHERE codigo = ?");
    $stmt->execute([$id]);
    echo json_encode(["message" => "Ruta eliminada"]);
}
```

Conexión a la base de datos:

```
error_reporting(E_ALL);
ini_set('display_errors', 1);

$host = 'localhost';
$dbname = 'artemisdb';
$username = 'root';
$password = '';

try {
    $pdo = new PDO("mysql:host=$host;dbname=$dbname", $username, $password);
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
} catch (PDOException $e) {
    die("Conexión fallida: " . $e->getMessage());
}
```