DOCUMENTO DE ARQUITECTURA DEL PROYECTO SAFETRACK

# 1. Introducción

Este documento describe la arquitectura del sistema de la plataforma web SafeTrack, diseñada para la gestión de prevención de riesgos en una empresa. Se detallan los componentes principales, tecnologías utilizadas y la estructura general del sistema.

# 2. Visión General de la Arquitectura

SafeTrack sigue una arquitectura de **tres capas (Three-Tier Architecture)** para mejorar la modularidad, escalabilidad y mantenimiento del sistema:

### **Capa de Presentación (Frontend)**: Interfaz de usuario accesible desde navegadores web.

### **Capa de Lógica de Negocio (Backend)**: Procesamiento de reglas de negocio y conexión con la base de datos.

### **Capa de Datos (Database)**: Almacenamiento de información estructurada.

# 3. Tecnologías Utilizadas

| Componente | Tecnología |
| --- | --- |
| Frontend | Vue.js (Frameworks de JavaScript) |
| Backend | Php, Javascript |
| Base de Datos | MySQL |
| Autenticación | JWT (JSON Web Token) / OAuth 2.0 |
| Despliegue | Docker (opcional) |
| Control de Versiones | Git y GitHub/GitLab |
| Gestión de Proyectos | Jira |

## 4. Funcionalidad de Tecnologías según la Arquitectura de SafeTrack

A continuación, se describen las funciones específicas de los lenguajes y tecnologías utilizados en la arquitectura de la plataforma web SafeTrack, desarrollada para la gestión de prevención de riesgos en entornos empresariales.

### 4.1 Capa de Presentación (Frontend)

Lenguaje y tecnologías utilizadas:

* JavaScript
* HTML5 / CSS3 / SCSS

Función principal:  
 Permite la creación de una interfaz gráfica intuitiva que facilite el acceso y la interacción con el sistema por parte de los usuarios (prevencionistas y jefaturas).

Responsabilidades:

* Visualización de tareas asignadas mensualmente.
* Consulta de documentación asociada a los trabajadores.
* Recepción de notificaciones sobre vencimientos de documentos o exámenes.
* Revisión de reportes de cumplimiento por proyecto.
* Aplicación de filtros y búsquedas avanzadas.
* Comunicación con el backend mediante llamadas HTTP utilizando API REST.

### 4.2 Capa de Aplicación (Backend)

Lenguaje utilizado:

* PHP

Función principal:  
 Gestionar la lógica de negocio del sistema, procesando solicitudes del frontend, verificando permisos y realizando operaciones sobre la base de datos.

Responsabilidades:

* Registro y autenticación de usuarios.
* Validación de permisos de acceso por rol.
* Asignación y seguimiento de tareas.
* Registro de documentación y vencimientos.
* Cálculo del porcentaje de cumplimiento por proyecto.
* Generación de reportes e historiales.

Autenticación:

* JWT (JSON Web Token):  
   Se utiliza para la autenticación y autorización de usuarios. Al iniciar sesión, el backend emite un token que el frontend utiliza para cada solicitud protegida, validando la identidad y permisos del usuario.

Comunicación:

* API REST:  
   El backend expone endpoints RESTful que permiten la interacción con los distintos módulos del sistema. Algunos ejemplos:  
  + GET /api/tareas → Obtiene tareas asignadas al usuario.
  + POST /api/documentos → Envía o actualiza documentación de trabajadores.
  + GET /api/cumplimiento/proyecto/{id} → Retorna el porcentaje de cumplimiento de un proyecto específico.

Contenerización (opcional):

* Docker:  
   Se considera el uso de Docker para empaquetar el backend, facilitando su despliegue, pruebas y mantenimiento en diferentes entornos.

#### **4.4 Modelo de Base de Datos Relacional**

El sistema utiliza un modelo de base de datos **relacional**, adecuado para estructurar y vincular eficientemente la información. Las tablas están conectadas mediante claves primarias y foráneas, lo que permite mantener integridad referencial y facilitar la trazabilidad de la información.

**Tablas principales:**

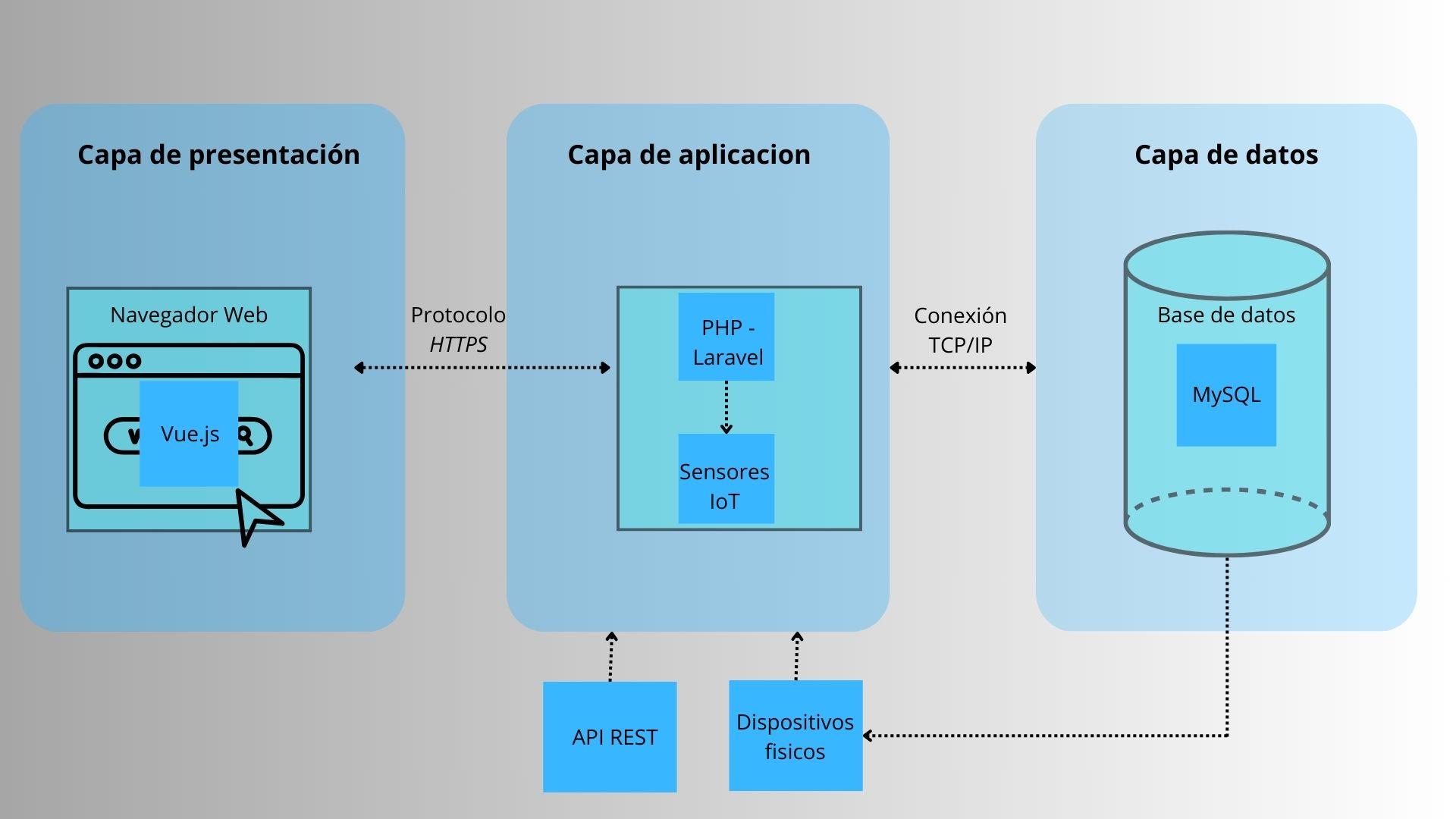
| **Tabla** | **Descripción** |
| --- | --- |
| **usuarios** | **Datos de login, rol y perfil del usuario (prevencionista, jefatura, admin).** |
| **trabajadores** | **Información de trabajadores por proyecto.** |
| **proyectos** | **Detalles de obras o zonas de trabajo.** |
| **documentos** | **Archivos asociados a cada trabajador (exámenes, certificados).** |
| **tareas** | **Actividades asignadas por fecha y proyecto.** |
| **vencimientos** | **Control de expiración de documentos.** |
| **cumplimientos** | **Porcentaje de cumplimiento por trabajador o proyecto.** |
| **roles** | **Tipos de usuario.** |
| **permisos** | **Accesos y restricciones por rol.** |

Los archivos como certificados o informes se almacenan en el servidor de archivos, mientras que en la base de datos se registra su ubicación, nombre, tipo y relación con el trabajador correspondiente.

#### **4.5 Flujo General de la Plataforma**

1. El usuario accede al sistema SafeTrack desde su navegador.
2. Inicia sesión y, si las credenciales son correctas, recibe un token JWT.
3. El frontend utiliza ese token para realizar solicitudes seguras hacia el backend.
4. El backend procesa las solicitudes, accede a la base de datos si es necesario y responde con la información solicitada.
5. La interfaz muestra los datos relevantes al usuario final.

## Diagrama de Arquitectura



# 6. Seguridad y Escalabilidad

* Implementación de **HTTPS** para la seguridad de la comunicación.
* Uso de **JWT** para autenticación de usuarios.
* Encriptación de contraseñas mediante **bcrypt**.
* Estrategia de escalabilidad con contenedores Docker.

# 7. Conclusión

La arquitectura de la plataforma **SafeTrack** está basada en un modelo de tres capas que permite una gestión eficiente, escalable y segura de los procesos relacionados con la prevención de riesgos. Cada capa —presentación, lógica de negocio y datos— cumple funciones específicas que se complementan entre sí, utilizando tecnologías modernas como JavaScript, PHP, API REST y JWT para garantizar una experiencia fluida y segura.

El sistema emplea una **base de datos relacional en MySQL**, adecuada para almacenar información estructurada de usuarios, trabajadores, documentos y proyectos. La seguridad está reforzada mediante el uso de HTTPS, cifrado de contraseñas y control de acceso por roles. Además, el uso opcional de **contenedores Docker** facilita la escalabilidad y portabilidad del sistema.

En conjunto, la arquitectura propuesta asegura un desarrollo modular y sostenible, adaptable a las necesidades del área de prevención de riesgos, con una estrategia de implementación progresiva que prioriza la validación continua de los módulos.