Diseño de Arquitectura del Sistema - SISTEC

**Fecha:** 17/09/2025

**Autor: Raurau Quispe, Juan Boris; Munares Chamorro, Juan Carlos**

**Versión:** 1.0

# 1. Introducción

Este documento describe de manera detallada la arquitectura del sistema para la aplicación web SISTEC; su propósito es permitir la gestión de equipos y alertas (y puede extenderse a tareas personales) de forma práctica y segura. El documento define la estructura general del sistema, los componentes reales, sus interacciones y las tecnologías empleadas..

# 2. Visón General de la Arquitectura

## 2.1. Estilo Arquitectónico

La aplicación implementada es un **sistema monolítico** basado en **cliente–servidor** tradicional, organizado en capas lógicas:

* **Capa de Presentación (Frontend):** Páginas HTML/PHP con CSS y JavaScript (no usa frameworks SPA).
* **Capa de Lógica de Negocio (Backend):** Código PHP que procesa formularios, consultas y operaciones CRUD.
* **Capa de Persistencia de Datos (Base de Datos):** MariaDB/MySQL (volcado incluido bd/basededatossistec.sql).

## 2.2. Diagrama General de Arquitectura:

Diagrama

El contenido generado por IA puede ser incorrecto.

# 3. Descripción de Componentes Principales

## 3.1. Frontend (Presentación)

* **Framework:** HTML + PHP (renderizado en servidor)
* **Estilo:** CSS (archivo css/style.css)
* **Responsabilidades:**
  + Renderizar las páginas principales del sistema (views/dashboard.php, views/login.php, views/equipos.php, views/alertas.php, views/usuarios.php).
  + Capturar entradas del usuario mediante formularios (login, registro, CRUD de equipos, usuarios y alertas).
  + Formulario para crear, editar y eliminar entidades.
  + Interacciones dinámicas básicas con **JavaScript** (archivo js/scipt.js).
  + Inclusión de cabeceras y pies mediante includes/header.php y includes/footer.php.

## 3.2. Backend (Lógica de negocio)

* **Lenguaje y Framework:** PHP (estructura monolítica con carpetas php/, includes/, views/).
* **Responsabilidades:**
  + Gestionar la lógica del sistema (altas, bajas, modificaciones y consultas de equipos, usuarios y alertas).
  + Validar datos y gestionar errores.
  + Aplicar reglas de negocio (control de accesos, gestión de alertas de equipos, roles de usuario).
  + Control de acceso mediante **sesiones PHP** y verificación de roles (includes/verificar\_sesion.php, includes/roles.php).

### 3.2.1. Módulos del Backend

* php/usuarios/ → scripts CRUD para usuarios (agregar.php, listar.php, actualizar.php, eliminar.php).
* php/equipos/ → scripts CRUD para equipos (alta, baja, modificación).
* php/alertas/ → gestión de alertas relacionadas con movimientos o tarjetas RFID.
* includes/verificar\_sesion.php → verificación de sesión y permisos de acceso.
* includes/roles.php → control de acceso basado en roles (admin, tecnico, invitado).

## 3.3. Base de Datos (Persistencia)

* Sistema Gestor: MySQL/MariaDB (archivo bd/basededatossistec.sql).
* ORM: No se utiliza ORM, las consultas están implementadas directamente en PHP (MySQLi/PDO).
* Esquema de Datos:
* Tabla usuarios: id, usuario, contraseña (hash bcrypt), rol (admin, tecnico, invitado).
* Tabla equipos: id, nombre, ubicación, estado (Autorizado, No autorizado), número de tarjeta RFID, fecha de registro.
* Tabla alertas: id, mensaje, ubicación, nivel (peligro, cuidado), fecha, número de tarjeta RFID (FK hacia equipos).

# 4. Integraciones Externas (Opcionales)

* Actualmente el sistema no cuenta con integraciones externas.
* Opcionales a implementar en el futuro:
  + Servicios de notificación por correo (PHPMailer, SendGrid).
  + Autenticación social con OAuth2 (Google, Microsoft).
  + Monitorización de errores con herramientas como Sentry.

# 5. Seguridad

* Contraseñas almacenadas con bcrypt (implementado en el dump de la BD).
* Control de acceso mediante sesiones PHP y roles.
* Validación de entradas de usuario a nivel de formulario y en backend.
* Se recomienda mejorar: uso de prepared statements (PDO/Mysqli) para evitar inyecciones SQL.
* Uso de HTTPS en entornos de producción.

# 6. Escalabilidad y Despliegue

Posibles problemas durante el desarrollo y escalabilidad:

* **Frontend:** Desplegado en servidores web compatibles con PHP (Apache/Nginx).
* **Backend:** Deploy en hosting LAMP/XAMPP local o servidores en la nube (DigitalOcean, Render, Railway, Heroku con buildpack PHP).
* **Base de Datos:** MySQL/MariaDB en servicios gestionados (RDS, Cloud SQL, Supabase).
* **Escalabilidad:**
  + Escalado vertical (mejores recursos en servidor).
  + Uso de balanceadores de carga para distribuir peticiones en múltiples instancias.
  + Almacenamiento de sesiones en Redis o base de datos para despliegues en clúster.

# 7. Conclusiones

La arquitectura del sistema SISTEC corresponde a un monolito en PHP con frontend HTML+CSS+JS y base de datos MySQL. Es adecuada para proyectos académicos o de pequeña a mediana escala. Para evolucionar:

* Integrar mejores prácticas de seguridad (prepared statements, CSRF tokens).
* Separar frontend y backend si se requiere escalabilidad (ej. migrar a API REST).
* Incorporar herramientas de notificación y monitorización para un entorno productivo.