# Método propuesto

## CASCADA

La metodología cascada cuenta con 5 faces que son fundamentales para el desarrollo del sistema web dividiendo el trabajo en fases secuenciales, en la que cada fase debe de completarse en su totalidad antes de que se pueda avanzar a lo siguiente.

## Fase 1

## Análisis

* Herramienta de recolección de información.
* Generar la propuesta.
* Definir los requisitos funcionales del sistema.
* Analizar el lector de códigos como protocolos soportados, configuración, documentación técnica.
* Establecer el flujo de trabajo en la línea de producción como el escaneo, Validación, Alarma (si hay error), Registro, Visualización.
* Determinar roles y permisos (administrador, operador).
* Analizar la infraestructura de red y servidor.

## Fase 2

## Diseño

* Crear diagramas de caso de uso
* Crear diagramas secuenciales
* Arquitectura cliente-servidor.
* Diagrama E-R
* Diagrama Relacional
* Backend en PHP (Diseño de la base de datos)
* Frontend en JavaScript para el panel de administración.
* Diseño de la ventana principal del lector de códigos
* Diseñar la arquitectura del sistema

## Fase 3

## Implementación

* Elegir el SGBD a utilizar
* Codificación de la base de datos
* Configurar el lector Keyence para enviar datos al servidor.
* Programar el backend para recibir, validar y almacenar los datos.
* Programar el frontend para mostrar información al administrador.
* Configurar la base de datos en XAMPP.
* Integrar la lógica de activación de alarmas desde el lector.

## Fase 4

## Verificación

* Pruebas de la base de datos
* Pruebas unitarias de cada módulo (escaneo, validación, almacenamiento).
* Pruebas de integración del sistema completo
* Pruebas de rendimiento de la velocidad de respuesta, carga de datos.
* Pruebas de error
* Validación con usuarios

## Fase 5

## Mantenimiento

* Corrección de errores
* Hacer actualizaciones
* Manual de usuario
* Manual técnico

**Desarrollo**

## Fase 1

## Análisis

### Herramienta de recolección de información.

* Levantamiento de información

Para poder llevar a cabo el desarrollo del sistema web se necesitó hacer un levantamiento de información con el operador de la línea de producción de las palomitas SLIMPOP, de tal manera en la que nos explicó su situación y brindo información que es fundamental para poder llevar a cabo el sistema, revisando a si los manuales técnicos del lector que se tienen a opción a usar, verificando así también sus protocolos soportados.

### Generar la propuesta

Se propone el desarrollo de un sistema web que automatice el proceso de escaneo de productos en la línea de producción, el sistema recibirá los datos del lector **X**, validará los códigos escaneados, emitirá una alerta sonora en caso de error, y registrará la información en una base de datos, el administrador podrá acceder a un panel de control donde visualizará reportes detallados por fecha, hora, código del producto, descripción y estado, esta solución busca reducir errores, mejorar y optimizar el tiempo de respuesta ante fallos en el escaneo.

### Definir los requisitos funcionales del sistema.

El sistema estará contando con requisitos específicos que son fundamentales para el funcionamiento.

* Recibir datos del lector mediante protocolo HTTP.
* Validar el código escaneado contra una lista de códigos válidos almacenados en la base de datos.
* Emitir una alerta sonora si el código es inválido.
* Registrar cada escaneo con fecha, hora, código y estado (válido/inválido).
* Permitir al administrador visualizar reportes filtrados por fecha y producto.
* Gestionar roles de usuario (operador y administrador) con diferentes niveles de acceso.
* Garantizar la seguridad de los datos mediante autenticación y conexión HTTPS.

### Analizar el lector de códigos como protocolos soportados, configuración.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Nombre del lector** | **Lenguaje** | **Función** | **Compatibilidad** | **Componentes** | **Librerías** | **Protocolos soportados** |
| Keyence SR-X100W | JavaScript+Node.js | El lector de códigos SR-X100W y el programa web trabajan en conjunto, el lector captura los códigos rápidamente y el sistema web se encarga de validarlos, notificar al operador y almacenar la información de forma organizada. | Si, el lector funciona como un dispositivo HID (Human Interface Device), es decir, como si fuera un teclado. | Frontend (HTML, JavaScript, CSS)  Backend (Node.js).  Base de datos (MySQL). | 1. En JavaScript (Navegador) 2. Visual: SweetAlert 3. Sonidos:   Audio() en JS   1. Autenticación: Passport.js | TCP/IP, SNTP, FTP, SFTP, HTTP, HTTPS, BOOTP, EtherNet/IP™, PROFINET, KV STUDIO, MC protocol, OMRON PLC Link, OPC UA |
| Keyence SR-X300W | JavaScript+Node.js+express.js | El lector de códigos Keyence SR-X300W se integra fácilmente con el sistema web desarrollado en JavaScript mediante Node.js esto es gracias a su conectividad Ethernet, puede enviar los datos directamente a un servidor web construido con Node.js, donde recibe los códigos escaneados y los valida contra una base de datos MySQL alojada en XAMPP. y Express.js. | Sí, el lector de códigos Keyence SR-X300W compatible con el programa desarrollado en JavaScript, siempre y cuando se utilice Node.js en el backend por lo que estos están diseñados para integrarse en sistemas mediante protocolos estándar como HTTP, TCP/IP, RS-232C y USB, lo que permite que se comuniquen con servidores web o aplicaciones locales. | Frontend web (HTML, CSS, JS).  Base de datos MySQL (XAMPP).  Node.js | 1. npm install mysql 2. npm install axios o npm install node-fetch | TCP/IP, SNTP, FTP, SFTP, HTTP, HTTPS, BOOTP, EtherNet/IP™, PROFINET, KV STUDIO, MC protocol, OMRON PLC Link, OPC UA |
| Keyence SR-2000W | Javascript + Node.js | El lector Keyence SR-2000W puede escanear los códigos en tiempo real y enviar los datos por TCP/IP al servidor web hecho en JavaScript (Node.js), el servidor valida el código, guarda el resultado en MySQL (XAMPP). | Es totalmente compatible ya que el lector soporta Ethernet TCP/IP, lo que permite conectarlo a tu servidor web para enviar datos. | Configuración vía KV STUDIO o interfaz web  Procesamiento de datos: JavaScript (Node.js), Express.js  Almacenamiento: MySQL (usando XAMPP) | Node.js: Para el servidor web.  Librería: mysql | TCP/IP, SNTP, FTP, BOOTP, EtherNet/IP™, PROFINET, KV STUDIO, MC Protocol, OMRON PLC Link |
| Keyence SR-1000W | HTML, CSS, JavaScript  PHP, Node.js o Python. | El lector captura el código para identificar el producto, posteriormente compara el código con los registrados para identificar anomalías. | Este escáner no se conecta directamente con los lenguajes como HTML, CSS o JavaScript, debido a que estos son lenguajes de interfaz pero si se pueden integrar perfectamente con PHP, Node.js o Python dentro del backend. | HTML+CSS, JavaScript, Audio API (FrontEnd)  Node.js, PHP, Python, Serial (RS-232C), MySQL, API REST (BackEnd). | Express.js (Enrutamiento web), socket.io (Comunicación con frontend), mysql (Conexión con base de datos), dotenv (Manejo de variables de entorno como puertos ip o claves). | TCP/IP, SNTP, FTP, BOOTP, protocolo MC, Omron PLC link, KV STUDIO, EtherNet/IP™, PROFINET. |

### Establecer el flujo de trabajo en la línea de producción como el escaneo, Validación, Alarma (si hay error), Registro, Visualización.

El flujo de trabajo del sistema será el siguiente:

* El producto pasa frente al lector.
* El lector escanea el código de barras.
* El lector envía el dato al servidor web mediante HTTP POST.
* El backend del sistema valida el código contra la base de datos.
* Si el código es válido, se registra como exitoso si no, se activa una salida del lector para emitir una alerta sonora.
* El sistema guarda el registro con fecha, hora, código y estado.
* El administrador accede al panel web para consultar los registros y generar reportes.

Este flujo garantiza una operación continua, validación inmediata y trazabilidad completa del proceso.

### Determinar roles y permisos (administrador, operador).

El sistema contará con un login que permitirá el acceso a dos tipos de usuarios uno será el operador y el otro el administrador

Operador: Usuario encargado de supervisar el escaneo en la línea de producción, tendrá acceso limitado a la interfaz web, donde podrá visualizar el estado de los escaneos en tiempo real, pero no podrá modificar ni consultar reportes históricos.

Administrador: Usuario con acceso completo al sistema, podrá consultar todos los registros, generar reportes por fecha y producto, configurar parámetros del sistema, y gestionar usuarios.

La autenticación se realizará mediante credenciales únicas, y se implementará control de acceso basado en roles para garantizar la seguridad de la información.

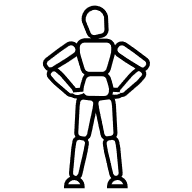
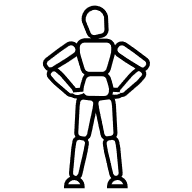
### Analizar la infraestructura de red y servidor.

El lector de códigos se conectará a la red Ethernet existente en la planta, utilizando una IP fija para facilitar la comunicación con el servidor, el servidor web estará alojado en la nube por lo que se requerirá abrir puertos específicos para recibir las solicitudes HTTP del lector, además, se implementará cifrado HTTPS para proteger la transmisión de datos y la red debe garantizar baja latencia y alta disponibilidad para evitar interrupciones en el proceso de escaneo.

## Fase 2

## Diseño

## Crear diagramas de caso de uso



Administrador

Operador

Escaneo de producto

Registro de productos

Acceso a listado de productos

## Crear diagramas secuenciales.

## PlantUML diagramArquitectura cliente-servidor.

## Elementos principales de la arquitectura

* **Cliente (navegador web)**
* Actores: Operador y Administrador
* Interfaz en HTML/CSS/JavaScript
* Envía solicitudes HTTP (al escanear o registrar productos).
* **Servidor Web (PHP + Servidor HTTP)**
* Contiene la lógica del sistema:
* Módulo de escaneo (consulta y validación de productos).
* Módulo de registro (inserción de productos).
* Módulo de reportes (consultas de escaneos).
* Conecta al cliente con la base de datos.
* **Base de Datos**
* Tabla productos: código, descripción, imagen.
* Tabla escaneos: fecha, hora, código, descripción, estado.
* Responde a las consultas y actualizaciones que realiza el servidor.

## Backend en PHP (Diagrama E-R)

Usuario

Producto

Escaneo

Realiza

Verifica

1:N

1:N

1:N

1:N

## Diagrama Relacional

## Diseño del login

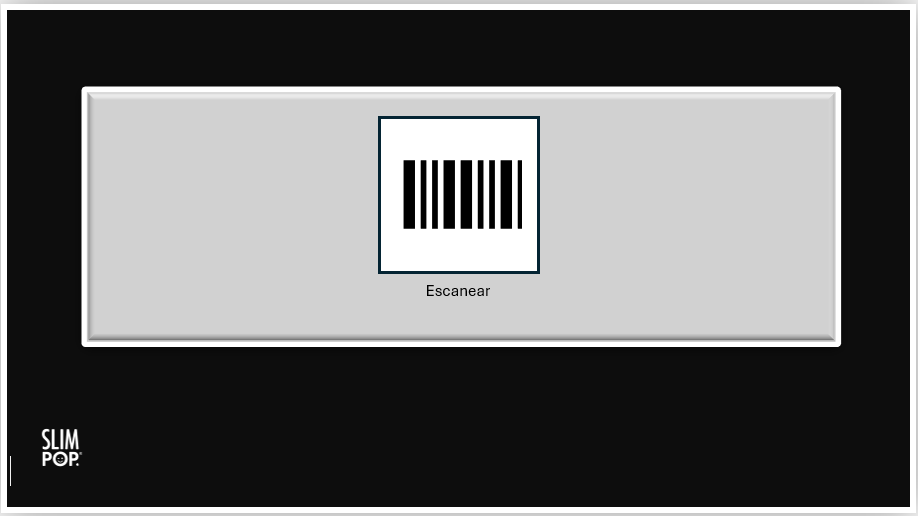
### Componentes de la ventana:

* Campo de texto “Usuario”: Ingreso de usuario para definir la ventana posterior.
* Campo de texto “Contraseña”: Ingreso de la contraseña que corresponde al usuario para su ingreso.
* Boton “Iniciar sesión”: Boton para ingreso pasando por proceso dependiendo del rol del usuario para su ventana principal.

## Frontend en JavaScript para el panel de administración. Ventana Principal “Administrador”

***Componentes de la ventana:***

* Botón “Escanear”: dirige a ventana de escaneo de códigos de barras.
* Botón “Registrar”: dirige a ventana de registro de productos.
* Botón “Productos registrados”: dirige a la ventana de los productos registrados.

1. **Diseño de ventana Principal “Operador”**

### Componentes de la ventana:

* Botón “Escanear”: dirige a ventana de escaneo de códigos de barras.

## Diseño de la Ventana Escaneo de productos

### Componentes de la ventana:

* Campo de texto “Código de barras”: Campo para ingresar el código de barras e iniciar la consulta de este dentro de la base de datos.
* Campo de texto “Descripción”: Campo consultado de la base de datos a través del código de barras.
* Campo de texto “Imagen”: En este campo se incluirá la imagen del producto que se escaneo y consulto.
* Botón “Registros”: Acceso a los registros de todos los productos que fueron escaneados, siendo registrados con fecha, hora, código de barras, descripción y el estado si fue valido o invalido.
* Botón “Regresar”: Retorno a la ventana anterior “Principal”.

1. **Diseño de la Ventana Registros**

### Componentes de la ventana:

* Tabla de “Registros”: Tabla donde se almacenarán los datos del escaneo para su próxima generación de reporte contando con los datos de Id de escaneo, fecha del escaneo, hora del escaneo, código de barras del producto escaneado, descripción que tiene el producto escaneado y el estado que tuvo el escaneo ya sea válido o invalido.
* Botón “Generar reporte”: Botón para generar un reporte de los productos que fueron escaneados con los datos incorporados en la tabla de “Registros”.
* Botón “Regresar”: Retorno a la ventana anterior “Escaneo de productos”.

1. **Diseño de la ventana Registrar Productos**

### Componentes de la ventana:

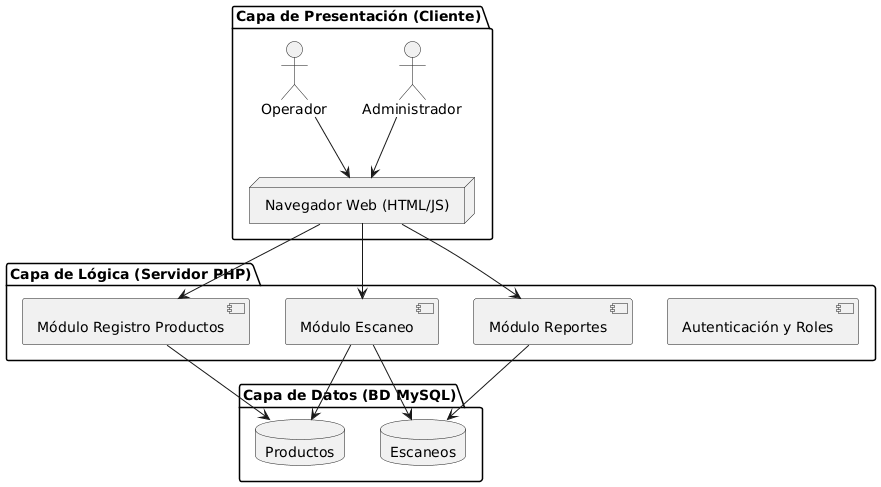
* Campo de texto “Código de barras”: Campo para ingresar el código de barras del producto que se desea registrar.
* Campo de texto “Descripción”: Campo para ingresar la descripción que tendrá el producto como el nombre del producto y algunas características del mismo.
* Campo de texto “Imagen”: Campo para ingresar una imagen que haga de referencia al producto.
* Botón “Registrar”: Botón para guardar la información que se introdujo dentro de la base de datos.

1. **Diseño de la ventana Productos Registrados**



* Componentes de la ventana:
* Tabla “Productos registrados”: Listado de los productos que se encuentran en la base de datos.
* Botón “Regresar”: Retorno a la ventana anterior “Principal”.
* Boton “Ver imagen”: Visualizacion de la imagen que tiene el producto.

## Diseñar la arquitectura del sistema



## Capa de Presentación (Cliente)

* Interfaz web en HTML/CSS/JavaScript.
* Se accede desde el navegador de Operador y Administrador.
* Funcionalidades según rol:

Operador: Escaneo y visualización.

Administrador: Escaneo, registro de productos y reportes.

## Capa de Lógica (Servidor PHP)

* Procesa las solicitudes y reglas del sistema.
* Módulos principales:

1. Módulo de Autenticación y Roles (gestiona permisos).
2. Módulo de Escaneo (valida códigos de barras contra la BD).
3. Módulo de Registro de Productos (solo para Admin).
4. Módulo de Reportes (consulta registros de escaneos).

## Capa de Datos (Base de Datos MySQL)

* Tabla productos: codigo\_barras, descripción, imagen.
* Tabla escaneos: id, fecha, hora, codigo\_barras, descripción, estado.
* Se encarga de almacenar y responder consultas SQL.

## Flujo General

* Usuario accede al navegador web.
* Solicitud HTTP → llega al servidor PHP.
* El servidor valida rol y acción → consulta o modifica la BD MySQL.
* La BD responde.
* El servidor genera la respuesta en HTML.
* El navegador del usuario muestra los resultados.

**Fase 3**

**Implementación**

* Elegir el SGBD a utilizar
* Codificación de la base de datos
* Configurar el lector **X** para enviar datos al servidor.
* Programar el backend para recibir, validar y almacenar los datos.
* Programar el frontend para mostrar información al administrador.
* Configurar la base de datos en XAMPP.
* Integrar la lógica de activación de alarmas desde el lector.

**Fase 4**

**Verificación**

* Pruebas de la base de datos
* Pruebas unitarias de cada módulo (escaneo, validación, almacenamiento).
* Pruebas de integración del sistema completo
* Pruebas de rendimiento de la velocidad de respuesta, carga de datos.
* Pruebas de error
* Validación con usuarios

**Fase 5**

**Mantenimiento**

* Corrección de errores
* Hacer actualizaciones
* Manual de usuario
* Manual técnico

Cronograma de actividades