

# **CORPORACIÓN UNIVERSITARIA MINUTO DE DIOS – UNIMINUTO**

## **ACTIVIDAD 1**

**ANDRES CAMILO OSPINA PRIETO**

**JUAN DAVID TELLEZ MONROY**

**LUIS ALIRIO TARAZONA PALACIO**

**DESARROLLO DE SOFTWARE SEGURO**

**TUTOR: EDWIN ALBEIRO RAMOS VILLAMIL**

**BOGOTÁ D.C 09 DE SEPTIEMBRE 2025**

## Introducción

El desarrollo de aplicaciones seguras y organizadas se ha convertido en una necesidad fundamental en el ámbito tecnológico actual. El presente documento expone el diseño e implementación del proyecto Software Seguro, una herramienta enfocada en la gestión de inventario con altos estándares de seguridad y estructura bajo el patrón arquitectónico Modelo-Vista-Controlador (MVC).

Este proyecto busca garantizar la protección de la información, la eficiencia en los procesos de registro y autenticación de usuarios, y el control adecuado de los recursos mediante funciones CRUD. Asimismo, se presentan las características técnicas, la composición modular del sistema y los diagramas que evidencian su arquitectura y funcionamiento.

### **Enlace repositorio GitHub**

[https://github.com/Adrw2019/software\\_seguro.git](https://github.com/Adrw2019/software_seguro.git)

## Características del Software

El sistema desarrollado en el proyecto Software Seguro incorpora un conjunto de características que garantizan seguridad, eficiencia y escalabilidad en la gestión de inventarios. Entre las más relevantes se destacan:

- **Autenticación y autorización de usuarios:** incluye registro de cuentas, inicio de sesión y recuperación de contraseñas mediante credenciales seguras.
- **Gestión integral de inventario (CRUD):** permite crear, consultar, actualizar y eliminar elementos del inventario de manera ágil y confiable.
- **Control de roles de usuario:** distingue entre perfiles de administrador y usuario estándar, asignando permisos y restricciones según el rol.
- **Interfaz dinámica en PHP:** ofrece vistas interactivas que facilitan la experiencia de uso y la navegación dentro del sistema.
- **Persistencia de datos en base de datos relacional:** respaldada en una estructura SQL (inventario.sql) que asegura integridad y consistencia de la información.
- **Arquitectura bajo el patrón MVC (Modelo-Vista-Controlador):** garantiza una clara separación de responsabilidades, favoreciendo la organización del código, su mantenibilidad y escalabilidad futura.

## Funciones del Sistema

El software Software Seguro integra un conjunto de funciones principales orientadas a la seguridad de la información y a la eficiencia en la gestión de inventarios. Estas funciones permiten al usuario interactuar con el sistema de manera práctica y confiable:

- **Registro de usuarios:** posibilita la creación de nuevas cuentas con credenciales seguras, almacenadas de forma protegida en la base de datos.



The 'Registro de Usuario' form features a title with a user icon, followed by input fields for 'Nombre', 'Email', and 'Contraseña'. A blue 'Registrarse' button is positioned below the fields, and a link for existing users is at the bottom.

**+ Registro de Usuario**

Nombre:

Email:

Contraseña:

**+ Registrarse**

[¿Ya tienes cuenta? Inicia sesión](#)

- **Inicio de sesión:** permite el acceso al sistema mediante autenticación con correo electrónico y contraseña, validando la identidad del usuario.



The 'Iniciar Sesión' form includes a title with a login icon, input fields for 'Email' and 'Contraseña', and a blue 'Ingresar' button. Links for registration and password recovery are located at the bottom.

**→ Iniciar Sesión**

Email:

Contraseña:

**→ Ingresar**

[¿No tienes una cuenta? Regístrate aquí](#)  
[¿Olvidaste tu contraseña?](#)

- **Recuperación de contraseña:** ofrece un mecanismo de restablecimiento seguro en caso de olvido o pérdida de credenciales.



**Recuperar Contraseña**

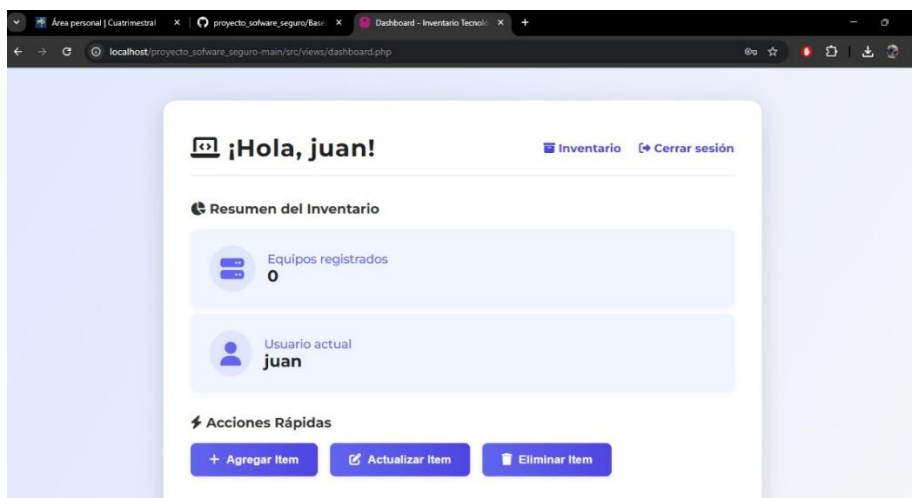
Ingresa tu correo electrónico y te enviaremos instrucciones para restablecer tu contraseña.

✉ Correo electrónico:

[Enviar instrucciones](#)

[Volver al login](#)

- **Gestión de inventario:** facilita la creación, edición, eliminación y consulta de ítems, manteniendo actualizado el control de los recursos.



¡Hola, juan!

[Inventario](#) [Cerrar sesión](#)

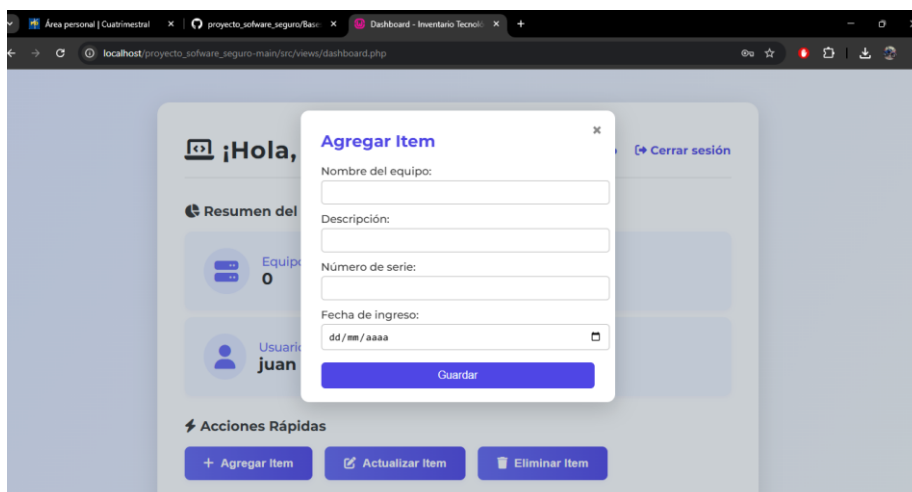
**Resumen del Inventario**

Equipos registrados: 0

Usuario actual: juan

**Acciones Rápidas**

[+ Agregar Item](#) [Actualizar Item](#) [Eliminar Item](#)



**Agregar Item**

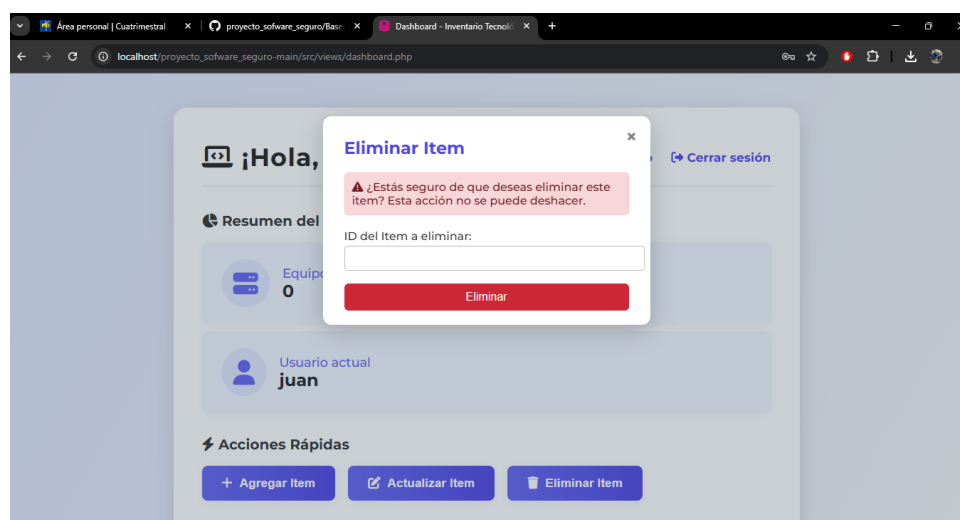
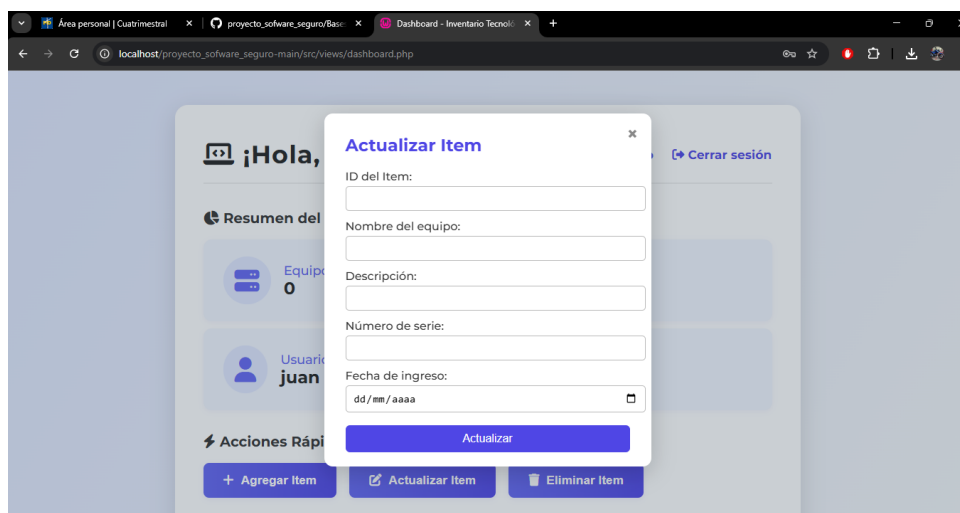
Nombre del equipo:

Descripción:

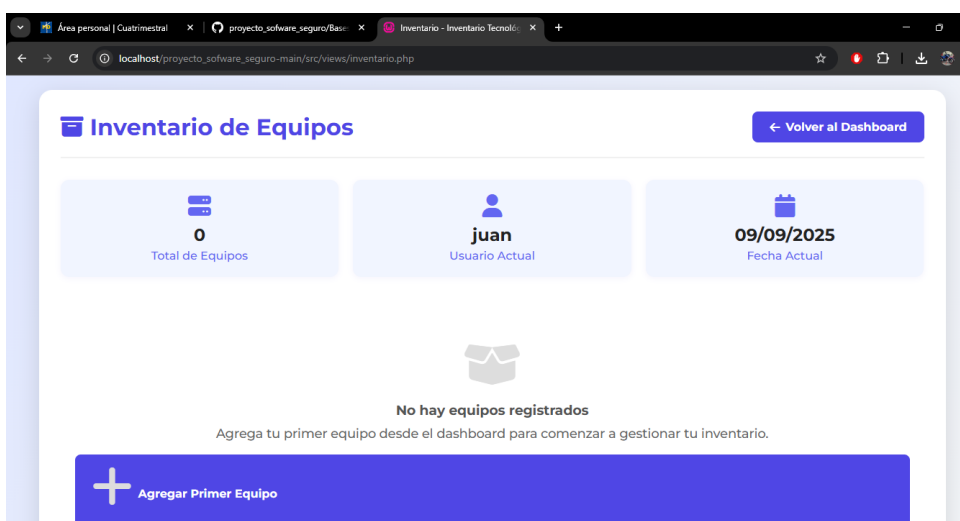
Número de serie:

Fecha de ingreso: dd/mm/aaaa

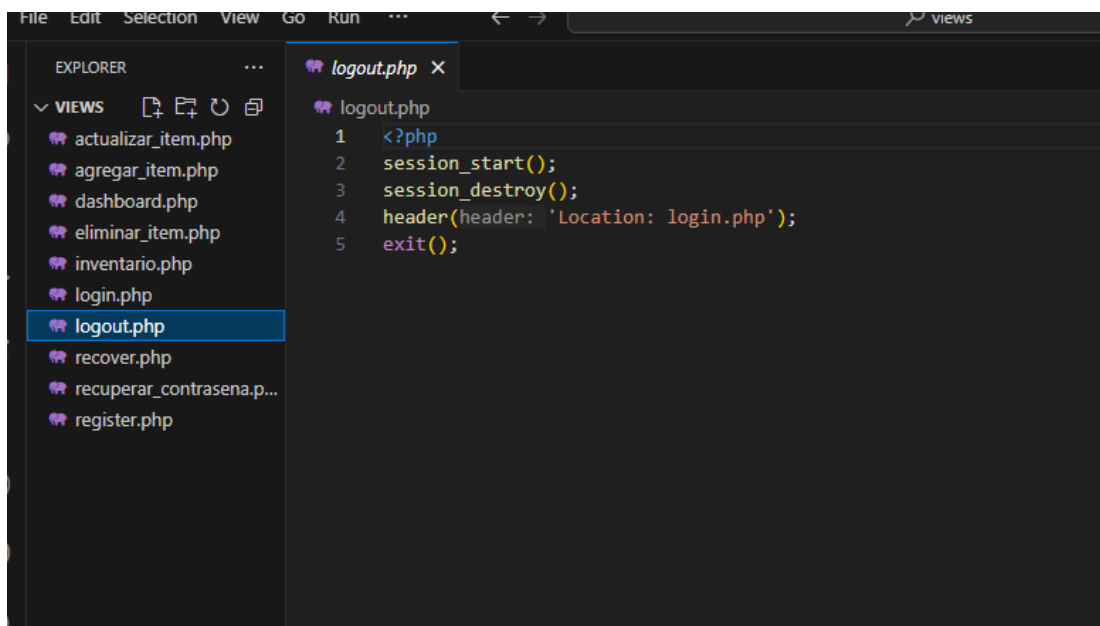
[Guardar](#)



- **Visualización de dashboard:** presenta un panel de control con información resumida y relevante sobre el estado del inventario.



- **Mecanismos de seguridad:** incluyen el uso de variables de entorno (.env), conexiones seguras a la base de datos y cifrado de contraseñas mediante algoritmos de hash.



```

1 <?php
2 session_start();
3 session_destroy();
4 header(header: 'Location: login.php');
5 exit();
  
```

## Composición del Sistema

El sistema Software Seguro está organizado de manera modular, lo que facilita su comprensión, mantenimiento y escalabilidad. Cada componente cumple una función específica dentro de la arquitectura y el sistema se organiza en los siguientes módulos:

- **config/:** Configuración de base de datos (db.js, db.php).
- **public/:** Archivos estáticos (CSS, JavaScript).
- **src/app.js:** Archivo principal de arranque del servidor Node/Express.
- **src/controllers/:** Contiene los controladores de la lógica de negocio.
- **src/models/:** Define las entidades de datos (Usuario, Inventario).
- **src/routes/:** Define las rutas de la aplicación que conectan controladores.
- **src/views/:** Plantillas PHP que constituyen la interfaz de usuario.
- **Bases de datos sql/:** Script SQL para crear y poblar la base de datos.

## Patrón Modelo-Vista-Controlador (MVC)

El proyecto Software Seguro implementa el patrón arquitectónico Modelo-Vista-Controlador (MVC), el cual proporciona una separación clara de responsabilidades entre los componentes del sistema. Los tres elementos principales del patrón se aplican de la siguiente manera:

- **Modelo (Model):**

- ✓ Representa la lógica de negocio y la estructura de los datos.
- ✓ Define las entidades principales del sistema (*Usuario e Inventario*).
- ✓ Gestiona la conexión con la base de datos y la manipulación de la información.

- **Vista (View):**

- ✓ Se encarga de la presentación y la interacción con el usuario.
- ✓ Está conformada por las plantillas en PHP (login.php, dashboard.php, inventario.php) junto con los recursos estáticos en la carpeta public/.
- ✓ Muestra la información de forma clara y facilita la usabilidad del sistema.

- **Controlador (Controller):**

- ✓ Actúa como intermediario entre el modelo y la vista.
- ✓ Procesa las solicitudes del usuario a través de las rutas HTTP.
- ✓ Ejecuta la lógica necesaria y devuelve la vista correspondiente.
- ✓ Ejemplos dentro del proyecto: authController.js, inventarioController.js y loginController.php.

- **Rutas (Routes):**

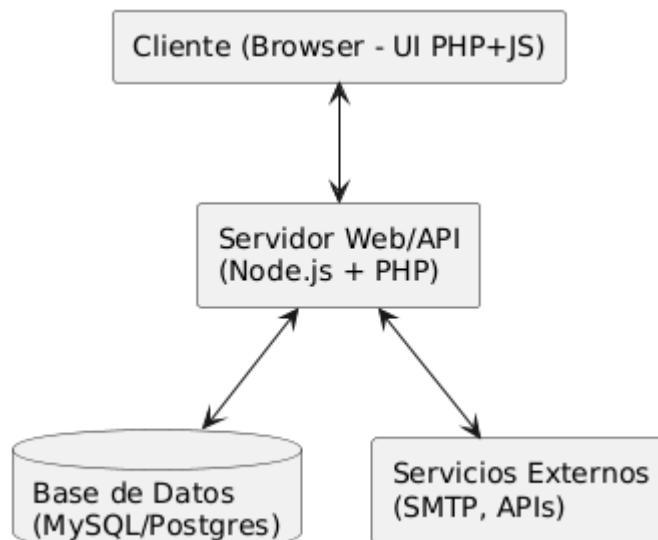
- ✓ Definen los caminos que permiten que las peticiones del cliente lleguen a los controladores adecuados.
- ✓ Facilitan la correcta comunicación entre los componentes del sistema.

En conjunto, el patrón MVC garantiza un flujo de trabajo ordenado y modular, permitiendo un desarrollo más ágil, seguro y fácil de mantener.

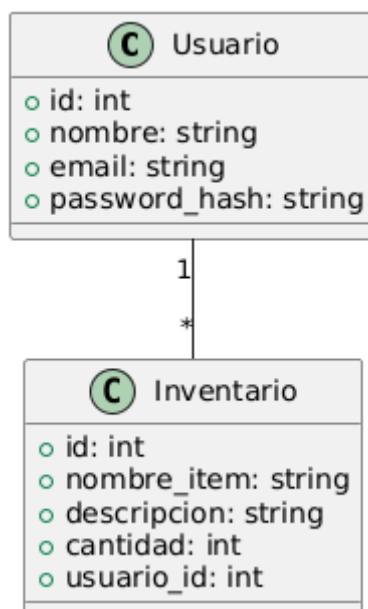


## Diagramas del Sistema

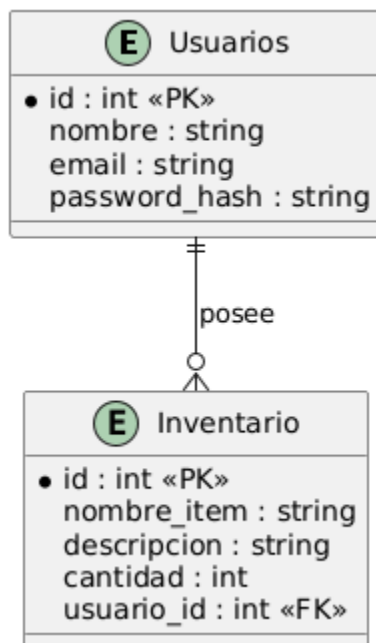
### 1. Diagrama de Componentes



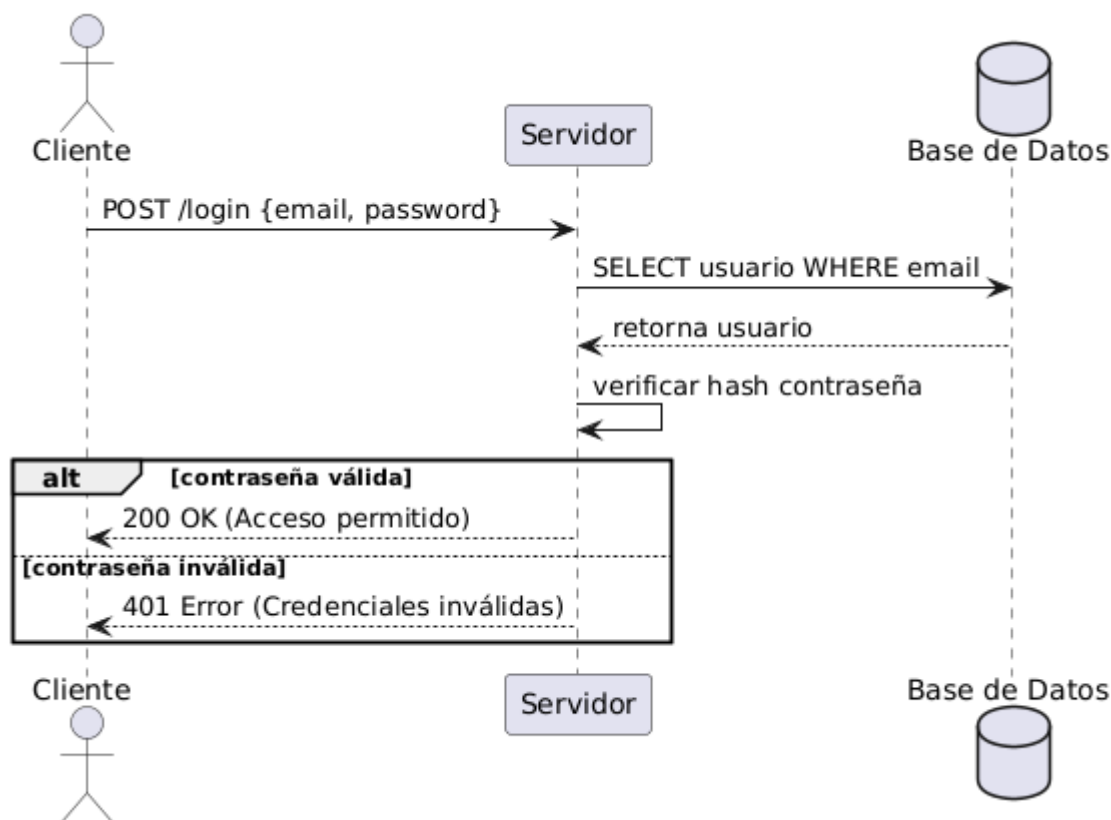
### 2. Diagrama de Clases



### 3. Diagrama Entidad-Relación



### 4. Diagrama de Flujo (Login)



## Conclusión

El proyecto Software Seguro constituye una solución práctica y robusta para la gestión de inventario, al integrar buenas prácticas de programación, seguridad de datos y un diseño estructurado mediante el patrón MVC. Su implementación permite a los usuarios acceder a un sistema confiable que asegura la integridad de la información y facilita la interacción a través de vistas dinámicas.

El enfoque modular garantiza escalabilidad y mantenibilidad del software. En conclusión, este desarrollo no solo cumple con los requerimientos técnicos planteados, sino que también sienta las bases para futuras mejoras y adaptaciones que puedan responder a nuevas necesidades.

## Referencias

- Fernández Romero, Y., & Díaz González, Y. (2012). *Patrón Modelo-Vista-Controlador*. <https://biblat.unam.mx/en/revista/telemtica-la-habana/articulo/patron-modelo-vista-controlador>
- Hernández Otálora, C. V. (2018, 07 de mayo). *Patrón arquitectónico MVC (Modelo Vista Controlador)* [Publicación institucional]. <http://repositorio.konradlorenz.edu.co/handle/001/138>
- ESIC Formación Profesional Superior. (s. f.). *Modelo-Vista-Controlador: qué es, cómo funciona y ventajas*. ESIC. Recuperado de ESIC Formación Profesional Superior. <https://www.esic.edu/rethink/tecnologia/modelo-vista-controlador-que-es-como-funciona-y-ventajas-c>