

**APLICACIÓN MÓVIL CON PRINCIPIOS DE DISEÑO**

**Desarrollo de Aplicaciones Móviles**

*Patiño Daniel, Capt Alex Paguay, Capt. Santiago Sañay*

7 de diciembre del 2024

Ing. Doris Chicaiza Ms.C

**Universidad de Fuerzas Armadas “ESPE”**

Contenido

[1. Resumen 3](#_Toc184500874)

[2. Introducción 3](#_Toc184500875)

[3. Objetivo General 4](#_Toc184500876)

[4. Objetivos Específicos 4](#_Toc184500877)

[5. Desarrollo 5](#_Toc184500878)

[Conceptos base 5](#_Toc184500879)

[Herramientas de desarrollo: 6](#_Toc184500880)

[Base de Datos 6](#_Toc184500881)

[Backend 7](#_Toc184500882)

[Crear un nuevo proyecto Node.js 7](#_Toc184500883)

[Instalar las dependencias necesarias: 8](#_Toc184500884)

[Configurar la base de datos MySQL: 8](#_Toc184500885)

[Frontend 10](#_Toc184500886)

[Pantalla de Login 10](#_Toc184500887)

[Pantalla de Registro 10](#_Toc184500888)

[Pantalla de Usuario 11](#_Toc184500889)

[6 Link de GitHub 12](#_Toc184500890)

[7 Conclusión 13](#_Toc184500891)

# Resumen

El sistema de gestión de usuarios implementado en este proyecto sigue una arquitectura de microservicios basada en una API RESTful que permite registrar, autenticar y gestionar usuarios de forma segura. El backend, desarrollado en PHP, utiliza el paquete JWT para manejar la autenticación y bcryptjs para cifrar las contraseñas, protegiendo así la información sensible de los usuarios. En el frontend, se emplea Flutter para crear una aplicación multiplataforma que interactúa con la API para proporcionar funcionalidades como el registro de nuevos usuarios, inicio de sesión, y la visualización de usuarios registrados mediante una pantalla dinámica. La base de datos MySQL almacena de forma eficiente los datos de los usuarios, mientras que las credenciales y tokens de acceso se mantienen seguros mediante SharedPreferences en el dispositivo móvil. El sistema está diseñado para ser escalable, asegurando la integridad y confidencialidad de los datos, mientras proporciona una interfaz amigable que facilita la gestión y visualización de la información. Este enfoque permite un acceso seguro y eficiente a los recursos del sistema, mejorando la experiencia del usuario sin comprometer la seguridad ni el rendimiento

# Introducción

El desarrollo de sistemas web y aplicaciones móviles requiere una integración eficaz de tecnologías para garantizar la seguridad, eficiencia y accesibilidad de los datos. En este proyecto, se ha diseñado un sistema de gestión de usuarios que utiliza una API RESTful para el registro, inicio de sesión y operaciones CRUD. (Hagos, 2018) El backend está implementado en PHP, utilizando autenticación JWT para gestionar sesiones seguras y bcryptjs para el cifrado de contraseñas. El frontend se desarrolla en Flutter, garantizando una experiencia de usuario moderna y multiplataforma. La base de datos está configurada con MySQL para almacenar de manera estructurada los datos de los usuarios.(*Flutter in Action - Eric Windmill - Google Libros*, s. f.)

# Objetivo General

Diseñar e implementar un sistema de gestión de usuarios mediante una API REST que permita realizar el registro, inicio de sesión (login) y operaciones CRUD, integrando un backend en PHP con autenticación basada en JWT, un frontend desarrollado en Flutter y el cifrado seguro de contraseñas utilizando bcryptjs, garantizando la seguridad y accesibilidad de la información para los usuarios.

# Objetivos Específicos

* Implementar el registro y autenticación de usuarios mediante una API REST
* Desarrollar el frontend en Flutter para interactuar con la API
* Implementar el cifrado seguro de contraseñas utilizando bcryptjs
* Establecer la conexión con la base de datos MySQL para operaciones CRUD

# 5. Desarrollo

## Conceptos base

Las APIs REST (Representational State Transfer) son un conjunto de principios arquitectónicos diseñados para la creación de servicios web escalables. Utilizan métodos HTTP como GET, POST, PUT y DELETE para interactuar con recursos, que suelen representarse en formato JSON o XML. Estas características permiten una comunicación eficiente entre sistemas independientes, integrando de manera efectiva el backend y el frontend.(*API Crash Course: Broken Object Level Authorisation Found in Coursera - Global Security Mag Online*, s. f.)

Node.js es un entorno de ejecución de JavaScript del lado del servidor, basado en el motor V8 de Google Chrome, que se utiliza para construir aplicaciones rápidas y escalables. En este proyecto, Node.js se emplea para desarrollar el backend, proporcionando una plataforma eficiente para manejar las operaciones asincrónicas y en tiempo real. Utilizando el framework Express, Node.js facilita la creación de una API RESTful para gestionar operaciones como el registr(Cepeda Enríquez & Loachamin Armas, 2012)

Flutter, un framework de código abierto creado por Google, se utiliza para el desarrollo del frontend. Este permite crear aplicaciones multiplataforma con un solo código base, ofreciendo interfaces modernas y personalizables. Además, Flutter interactúa con la API REST para gestionar datos y mejorar la experiencia del usuario final, gracias a su capacidad para compilar widgets en código nativo.

La biblioteca bcryptjs, que cifra las contraseñas generando hashes difíciles de revertir. Esta herramienta es fundamental para proteger datos sensibles y utiliza un algoritmo adaptativo que incrementa la complejidad del cifrado en función de los avances tecnológicos.

Jsonwebtoken se utiliza para implementar la autenticación basada en JWT (JSON Web Tokens). Estos tokens son objetos JSON compactos y autocontenidos que se generan en el backend tras un inicio de sesión exitoso. Los tokens permiten transmitir información de forma segura entre partes y su validez es verificada en endpoints restringidos para controlar el acceso a recursos sensibles dentro del sistema.(Gomes, 2014)

## Herramientas de desarrollo:

1. **Backend:** Utiliza Node.js con Express para gestionar la lógica del servidor y crear una API REST eficiente.
2. **Autenticación:** Implementada mediante JSON Web Token (JWT) para garantizar sesiones seguras y autorizadas.
3. **Cifrado de Contraseñas:** Realizado con bcryptjs para proteger la información sensible de los usuarios.
4. **Base de Datos:** MySQL se utiliza como sistema de gestión relacional para almacenar y organizar los datos.
5. **Frontend:** Desarrollado en Flutter (pendiente de integración) para proporcionar una interfaz de usuario moderna y multiplataforma.
6. **Entorno de Desarrollo:** Configurado en Visual Studio Code, utilizando Postman para la prueba y depuración de la API.

## Base de Datos

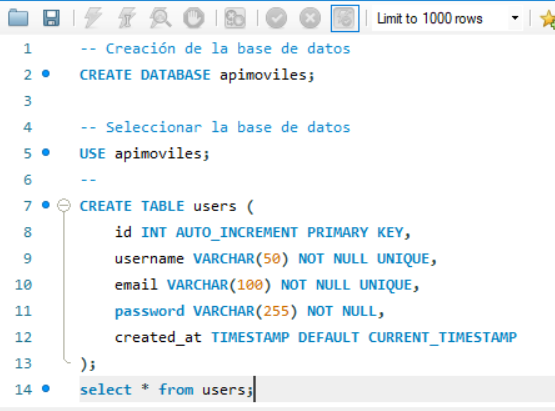
Implementación de la Base de Datos en MySQL

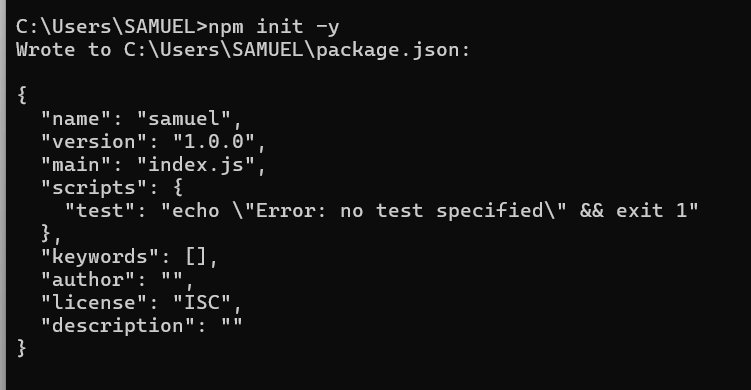
Figura 1 Creación de la Base de Datos

## Backend

## Crear un nuevo proyecto Node.js

Abre una terminal y dirígete al directorio donde deseas crear tu proyecto. Crea un nuevo directorio llamado "backend" y accede a él. Luego inicializa el proyecto con npm init -y, lo que generará un archivo package.json con la configuración básica del proyecto. Este archivo es crucial para gestionar las dependencias y scripts del proyecto.

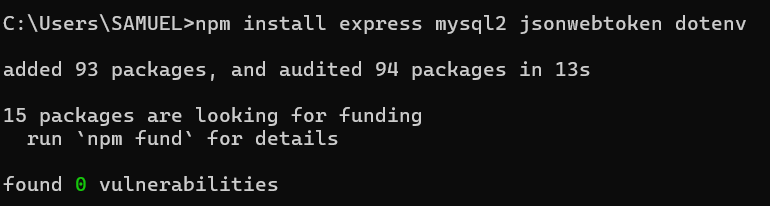
**Figura 2** Inicialización del Proyecto.



## Instalar las dependencias necesarias:

Para configurar el backend, instala las dependencias principales utilizando npm. Entre ellas, express se usará para crear el servidor, mysql2 para la conexión con la base de datos MySQL, jsonwebtoken para la autenticación mediante JWT y dotenv para manejar las variables de entorno. Estas dependencias se instalan con el comando npm install express mysql2 jsonwebtoken dotenv, asegurando así una configuración robusta para la API.

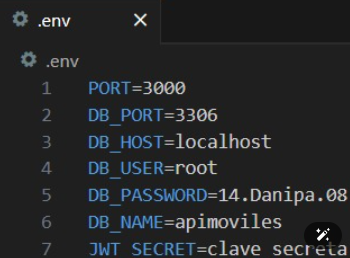
**Figura 3** Instalar Dependencias



## Configurar la base de datos MySQL:

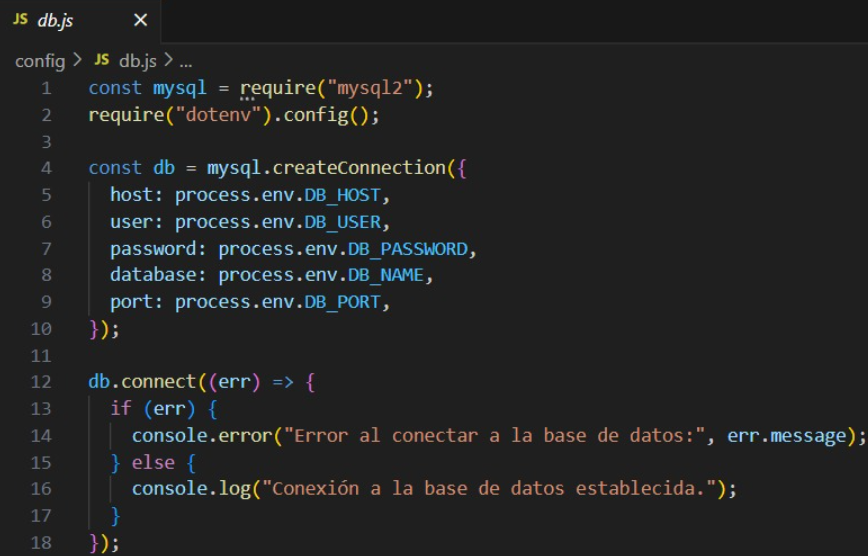
Asegúrate de tener instalado MySQL y haber creado un esquema de base de datos, como usuariosdb, para almacenar los datos de los usuarios. La configuración de la conexión se realiza mediante un archivo .env, donde se definen variables de entorno como DB\_HOST, DB\_USER, DB\_PASSWORD, DB\_NAME y SECRET\_KEY. Este archivo asegura que información sensible como las credenciales de la base de datos permanezca privada y accesible únicamente desde el servidor.

**Figura 4** Archivo NV



**Conectar a MySQL**  
Usa la biblioteca mysql2 para establecer la conexión entre el backend y la base de datos MySQL. El código de conexión incluye parámetros como el host, usuario, contraseña y nombre de la base de datos y verificar en la consola para confirmar si se ha conectado con éxito o si hubo algún error.

**Figura 5** Conexión MySQL

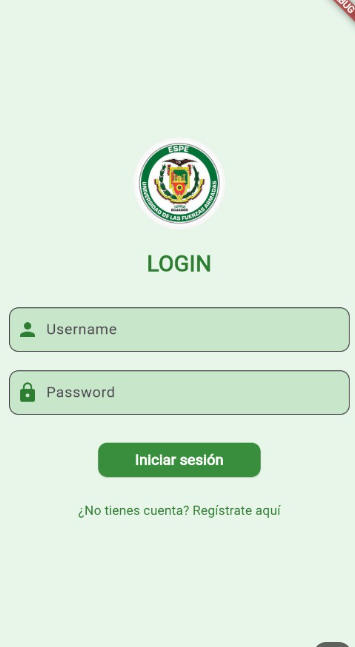


## Frontend

## Pantalla de Login

Se define una pantalla de inicio de sesión (LoginScreen) en una aplicación Flutter, donde el usuario puede ingresar su nombre de usuario y contraseña. Al hacer clic en el botón de "Iniciar sesión", se realiza una solicitud HTTP POST al backend para autenticar al usuario. Si las credenciales son correctas, se recibe un token JWT que se almacena en SharedPreferences para mantener la sesión del usuario. Si el inicio de sesión es exitoso, se redirige al usuario a la pantalla UsersScreen; de lo contrario, se muestra un mensaje de error.

**Figura 6** Login



## Pantalla de Registro

Se define una pantalla de registro (RegisterScreen) en una aplicación Flutter, donde los usuarios pueden ingresar su nombre completo, correo electrónico, nombre de usuario, contraseña y confirmar la contraseña. Si las contraseñas no coinciden, se muestra un mensaje de error. Al hacer clic en el botón de "Registrar", se realiza una solicitud HTTP POST al backend para registrar al usuario. Si el registro es exitoso (código de estado 201), se muestra un mensaje de éxito y los campos del formulario se limpian. En caso de error, se muestra un mensaje indicando que hubo un problema en el registro.

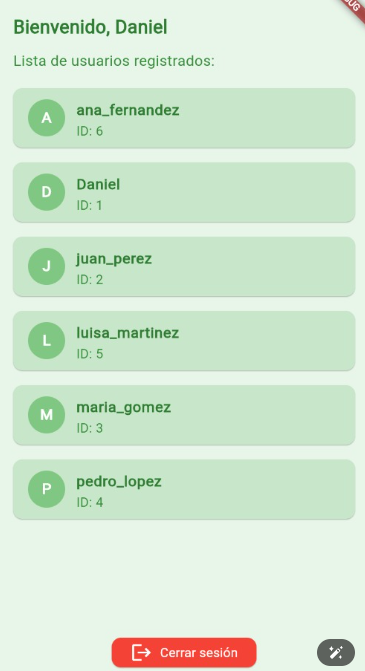
Figura 7 Pantalla Registro



## Pantalla de Usuario

Se define una pantalla de usuarios (UsersScreen) en una aplicación Flutter, donde se muestra una lista de usuarios registrados que se obtiene desde el backend. En la pantalla, el nombre del usuario autenticado se muestra en la parte superior, y debajo se presenta una lista de los usuarios obtenidos, mostrando su nombre de usuario y un identificador (ID). Si no hay usuarios, se muestra un mensaje indicando que no se encontraron. Además, hay un botón para cerrar sesión, que elimina los datos de autenticación almacenados en SharedPreferences y redirige a la pantalla de inicio de sesión. La solicitud a la API incluye un token JWT para autenticar al usuario.

**Figura 8** Pantalla Usuario



# 6 Link de GitHub

<https://github.com/alexpaguay1991/moviles2024/tree/main/aplicacionloginjwt>

# Conclusión

El sistema desarrollado permite gestionar usuarios de forma segura mediante una API REST con backend en PHP, autenticación JWT y almacenamiento cifrado de contraseñas con bcryptjs. El frontend, creado con Flutter, ofrece una interfaz moderna y multiplataforma, permitiendo el registro, inicio de sesión y operaciones CRUD. La integración entre el backend y frontend se realiza de manera eficiente, asegurando la protección de los datos sensibles y proporcionando una experiencia de usuario fluida y segura. El uso de JWT para la autenticación y SharedPreferences para el almacenamiento seguro refuerzan la seguridad del sistema

**8 . Referencias**

*API Crash Course: Broken Object Level Authorisation Found in Coursera—Global Security Mag Online*. (s. f.). Recuperado 6 de junio de 2024, de https://www.globalsecuritymag.com/API-Crash-Course-Broken-Object,20210709,113771

Cepeda Enríquez, M. B., & Loachamin Armas, C. A. (2012). *Análisis comparativo de los frameworks WCF (Windows Communication Foundation) y SCA (Services Component Architect) utilizados en el desarrollo de aplicaciones orientadas a servicios* [bachelorThesis, QUITO/EPN/2012]. http://bibdigital.epn.edu.ec/handle/15000/5151

*Flutter in Action—Eric Windmill—Google Libros*. (s. f.). Recuperado 18 de noviembre de 2024, de https://books.google.com.ec/books?hl=es&lr=&id=EzgzEAAAQBAJ&oi=fnd&pg=PT19&dq=flutter&ots=7zqBy37uBW&sig=3SitS-tZoZos0ecJhyVV6mVxxLw&redir\_esc=y#v=onepage&q=flutter&f=false

Gomes, D. A. (2014). *Web Services SOAP em Java - 2a Edição: Guia prático para o desenvolvimento de web services em Java*. Novatec Editora.

Hagos, T. (2018). Android Studio. En T. Hagos (Ed.), *Learn Android Studio 3: Efficient Android App Development* (pp. 5-17). Apress. https://doi.org/10.1007/978-1-4842-3156-2\_2