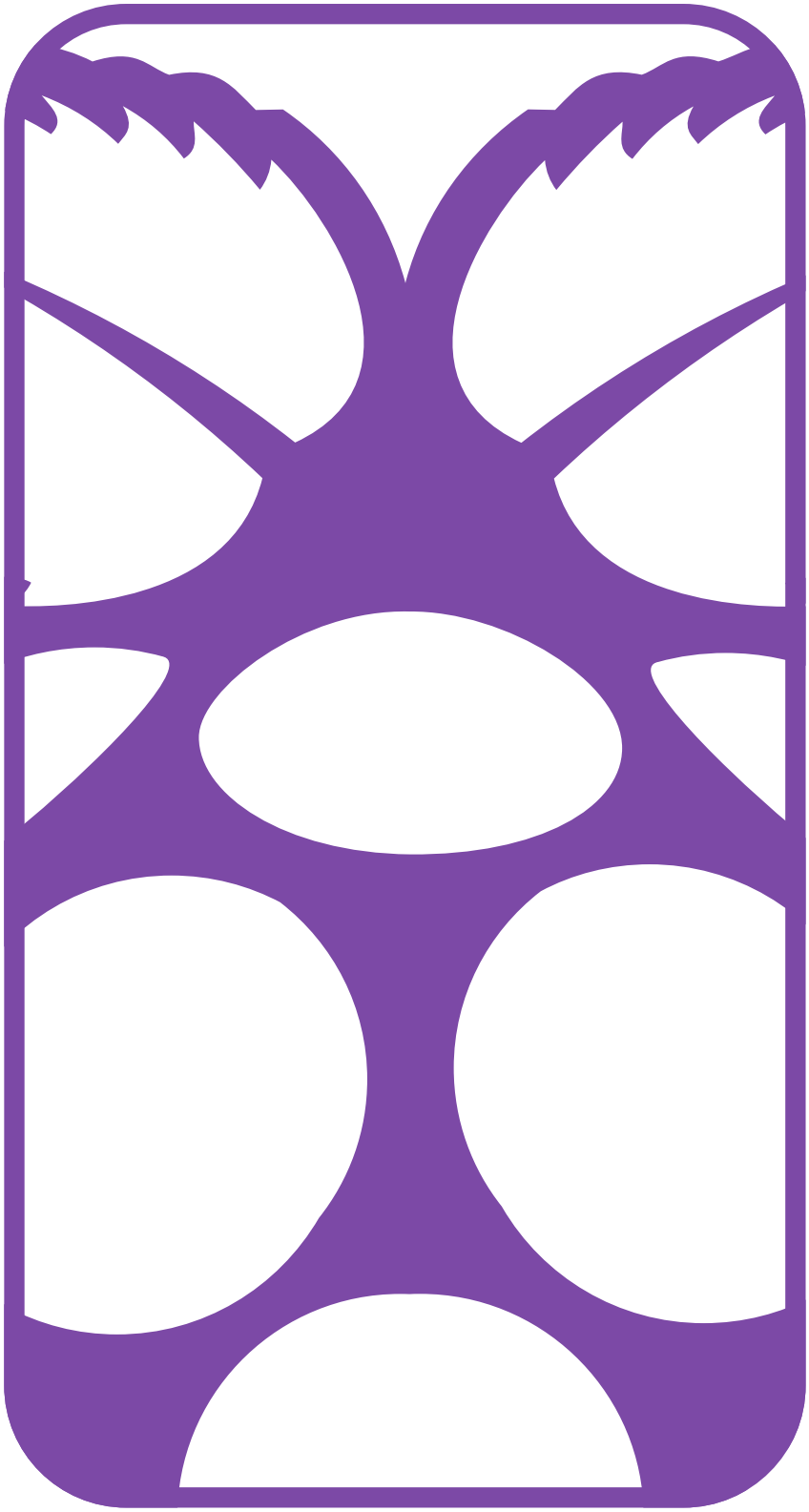


Sistema de Videovigilancia Inteligente con Flask y Raspberry Pi

Marcos López
Jhonny Vicmar Claure
Lúcia Abellán
Adrià Martínez

1. introducción

El presente documento describe el desarrollo de un sistema de videovigilancia basado en una Raspberry Pi, diseñado como parte de la asignatura de proyectos. Este trabajo fue realizado por un equipo de cuatro estudiantes con el objetivo de integrar diferentes áreas del desarrollo de software: programación en Python, arquitectura de sistemas, trabajo con bases de datos, gestión de entornos y comunicación en equipo.

El proyecto se fundamenta en la creación de una aplicación web mediante el framework Flask, conectada a una base de datos en la nube mediante MongoDB Atlas, y apoyada por un sistema físico de detección de movimiento gestionado por una Raspberry Pi. Además, se incorporó un sistema de internacionalización (i18n) y una gestión segura de variables mediante Infisical, garantizando así la seguridad de las credenciales y la escalabilidad del sistema.

2. Objetivos del Proyecto

El propósito principal del proyecto fue construir un sistema funcional que permitiera capturar imágenes de forma automática cuando la cámara detecta movimiento, almacenarlas en una base de datos y ofrecer una interfaz web donde los usuarios puedan visualizar el contenido capturado en tiempo real.

Entre los objetivos específicos destacan:

- Diseñar un backend modular y ligero con Flask.
- Implementar un sistema de detección de movimiento en la Raspberry Pi usando OpenCV.
- Almacenar datos e imágenes de forma segura en MongoDB Atlas.
- Integrar un sistema de notificaciones mediante Discord Webhooks.
- Implementar internacionalización en catalán y castellano.
- Asegurar la protección de credenciales y datos sensibles mediante Infisical.
- Desarrollar un entorno estable para pruebas, mantenimiento y ampliaciones futuras.

3. Descripción General del Sistema

El sistema combina hardware y software en un entorno distribuido. La Raspberry Pi actúa como punto de captura y procesamiento inicial, mientras que el backend Flask centraliza la gestión de usuarios, archivos, rutas y la comunicación con la base de datos.

La arquitectura básica sigue un modelo cliente-servidor, apoyado en tres componentes principales:

1. Raspberry Pi: dispositivo que ejecuta un script en Python con OpenCV. Se encarga de detectar movimiento y enviar las imágenes capturadas al servidor Flask.

2. Servidor Flask: recibe las imágenes, las valida y las almacena en la base de datos. Además, ofrece un panel web para visualización, autenticación de usuarios y streaming en tiempo real.

3. MongoDB Atlas: base de datos en la nube que almacena los registros de usuarios y las referencias a las fotos capturadas.

El sistema también cuenta con un servicio adicional que envía notificaciones automáticas al servidor de Discord del equipo, informando de cada nueva captura registrada. Este servicio mejora la supervisión y permite una comunicación inmediata sobre la actividad del sistema.

4. Arquitectura Técnica

El backend sigue una estructura modular con capas bien diferenciadas: rutas, controladores, servicios y esquemas de validación.

1. Rutas (Routes): definen los endpoints disponibles para la interacción con la API REST. Incluyen módulos para login, dashboard y gestión de fotos.

2. Controladores (Controllers): procesan la lógica de negocio y se comunican directamente con MongoDB para realizar operaciones CRUD (crear, leer, actualizar, eliminar).

3. Servicios (Services): agrupan funciones especializadas, como la transmisión de vídeo en tiempo real y el envío de mensajes a Discord.

3. Modelos y esquemas (Models/Schemes): utilizan la librería Marshmallow para validar los datos antes de ser almacenados.

Este patrón modular permite mantener el código organizado, facilitar la depuración y mejorar la escalabilidad. La comunicación entre módulos se realiza mediante Blueprints, una característica de Flask que permite dividir el proyecto en submódulos fácilmente reutilizables.

5. Tecnologías Utilizadas

Flask (Python)	Framework web principal para construir la API y las vistas del panel de control.
MongoDB Atlas:	Base de datos NoSQL alojada en la nube, ideal por su flexibilidad y velocidad.
OpenCV	Librería utilizada en la Raspberry Pi para la detección de movimiento mediante análisis de diferencias entre frames.
Flask-Babel y PyBabel:	herramientas para la internacionalización (i18n) que permiten cambiar entre catalán y castellano.
Infisical:	Gestor de variables de entorno que garantiza la seguridad de las credenciales.
Trello (Atlassian):	Herramienta de planificación utilizada por el equipo para organizar tareas, aunque se adoptó en etapas posteriores del proyecto.

6. Configuración del Entorno

Para la configuración del entorno de desarrollo se utilizó Python 3.11 con un entorno virtual (.venv) dentro del directorio backend.
La instalación de dependencias se realizó a través del archivo requirements.txt, que incluye librerías como Flask, Flask-PyMongo, Flask-Babel, OpenCV, Marshmallow y Requests.

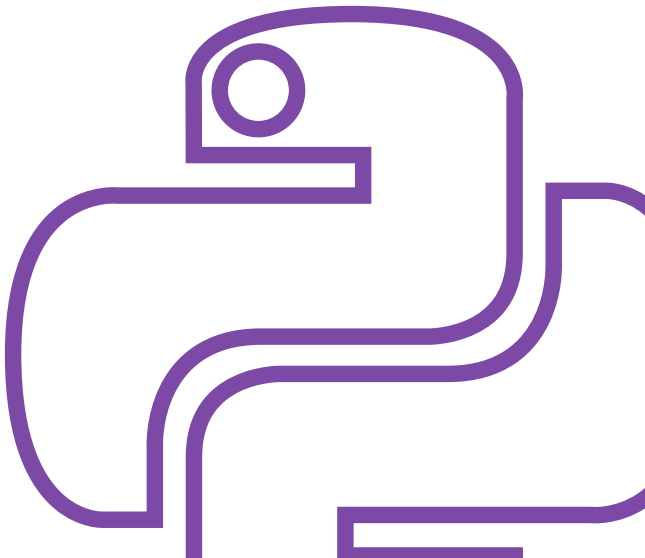
El sistema de variables de entorno se gestiona con Infisical, evitando el uso directo de archivos .env con credenciales. Cada miembro del equipo puede ejecutar el proyecto con los secretos cargados de forma segura mediante el comando:

```
infisical run --env=dev python app.py
```

En la Raspberry Pi, el script principal se ejecuta con:

```
python main.py
```

Este script inicia el bucle de detección, toma fotografías y las envía al endpoint del backend. El servidor Flask procesa la imagen, la guarda en el directorio media/screenshots/ y registra la información en la base de datos.



7. Internacionalización (i18n)

La aplicación implementa internacionalización utilizando PyBabel. Todas las cadenas de texto se marcan con la función _() tanto en los archivos Python como en las plantillas HTML.

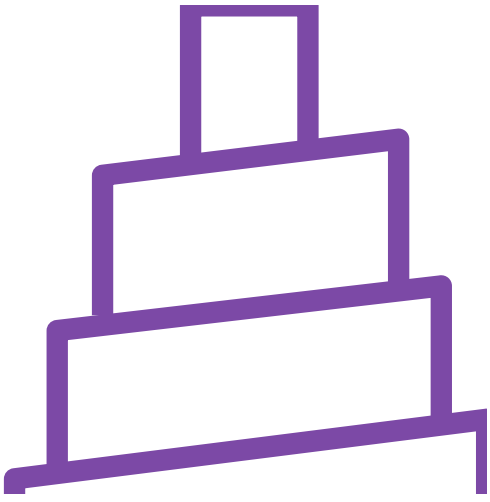
El sistema soporta actualmente dos idiomas: catalán y castellano. El cambio de idioma se realiza desde la interfaz web mediante parámetros en la URL, por ejemplo: ?lang=ca o ?lang=es.

Los archivos de traducción se encuentran en el directorio locales/, con las rutas:

```
locales/es/LC_MESSAGES/messages.po
locales/ca/LC_MESSAGES/messages.po
```

La compilación de traducciones para producción se realiza mediante el comando:

```
pybabel compile -d locales
```



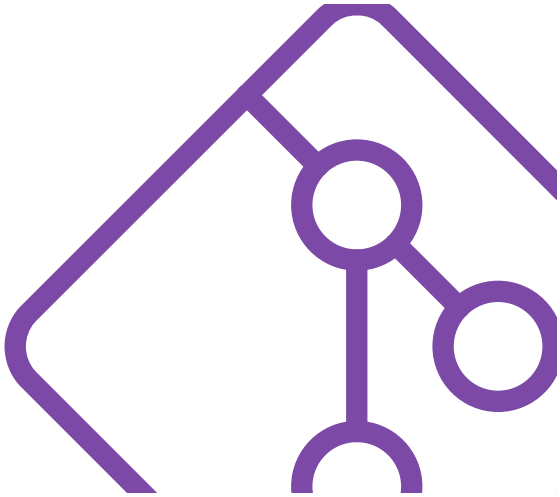
8. Seguridad y Gestión de Entorno

El proyecto prioriza la protección de credenciales y la separación de entornos de desarrollo y producción. Gracias a Infisical, las claves y URLs sensibles no se encuentran en el código fuente, sino que se almacenan cifradas y se cargan dinámicamente al ejecutar la aplicación.

Entre las variables más importantes se incluyen:

URL_MONGO	conexión a MongoDB Atlas.
WEBHOOK_DISCORD	URL para el envío de notificaciones.
SECRET_KEY	clave utilizada por Flask para gestionar sesiones de usuario.

Además, el archivo .gitignore se configuró para evitar subir archivos de entorno o logs al repositorio, reduciendo riesgos de exposición.



9. Flujo de Funcionamiento

El proyecto prioriza la protección de credenciales y la separación de entornos de desarrollo y producción. Gracias a Infisical, las claves y URLs sensibles no se encuentran en el código fuente, sino que se almacenan cifradas y se cargan dinámicamente al ejecutar la aplicación.

La Raspberry Pi inicia el proceso de detección utilizando OpenCV.

Cuando detecta movimiento, captura una imagen y la envía al backend Flask mediante una petición POST.

Flask valida los datos, guarda la imagen en el servidor y registra su información en MongoDB.

Se envía una notificación al canal de Discord del proyecto, con la fecha, la hora y el nombre del archivo.

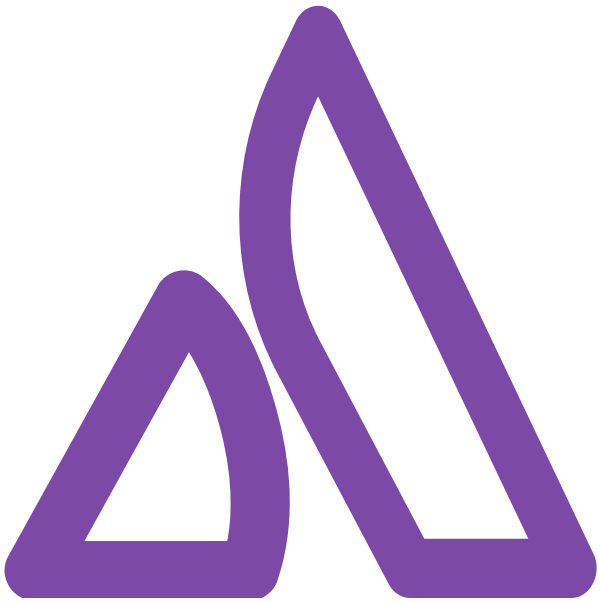
Los usuarios pueden acceder al panel web y visualizar tanto las fotos capturadas como la transmisión en tiempo real.

El sistema permite autenticación de usuarios y gestión diferenciada entre panel de usuario normal y panel de administrador.

10. Gestión del Proyecto

Durante el desarrollo, el equipo trabajó con un enfoque colaborativo, aunque inicialmente sin una estructura clara de tareas. Esta falta de planificación generó algunos retrasos y confusiones en la integración de los componentes. Como conclusión interna, el equipo reconoció la necesidad de un planteamiento más ordenado desde las etapas iniciales.

En las últimas fases del proyecto se decidió utilizar Trello (Atlassian) como herramienta de organización para mejorar la comunicación y el seguimiento de tareas. Esta decisión permitió coordinar las responsabilidades y mantener una visión global del avance del proyecto.



11. Resultados Obtenidos

El sistema final logró cumplir con los objetivos planteados.
Entre los principales logros se destacan:

- **Funcionamiento completo de la detección automática de movimiento.**
- **Comunicación estable entre la Raspberry Pi y el servidor Flask.**
- **Integración correcta con MongoDB Atlas.**
- **Notificaciones automáticas en Discord tras cada captura.**
- **Interfaz web funcional con cambio dinámico de idioma.**
- **Configuración segura de credenciales mediante Infisical.**

Aunque no se integraron elementos visuales avanzados ni algoritmos de inteligencia artificial, el sistema demostró estabilidad y escalabilidad, lo que lo convierte en una base sólida para futuras ampliaciones.

12. Conclusiones

El desarrollo de este proyecto permitió aplicar de forma práctica múltiples conceptos técnicos: programación en Python, arquitectura de servidores, bases de datos NoSQL, gestión de entornos seguros y control de versiones.
El trabajo en equipo fue fundamental, aunque se identificaron debilidades en la planificación y la distribución inicial de tareas. Aun así, el proceso fue una experiencia formativa muy valiosa.

Entre las lecciones más importantes se destacan:

> **La necesidad de definir una arquitectura antes de comenzar a programar.**

> **La utilidad de herramientas colaborativas como Trello para gestionar el flujo de trabajo.**

> **La importancia de la modularidad del código y de la seguridad en el manejo de credenciales.**

> **La relevancia de la documentación técnica y el control de versiones para garantizar la mantenibilidad del proyecto.**

En conclusión, el sistema de videovigilancia con Raspberry Pi y Flask constituye un proyecto completo, funcional y técnicamente coherente, que demuestra la integración efectiva entre hardware y software, y sienta las bases para desarrollos más avanzados en el ámbito del IoT y la automatización.

13. Bibliografía

Flask. Official Documentation.

PyBabel. Documentation.

Infisical. Secure Environment Management.

MongoDB Atlas. Documentation.

OpenCV. Python Tutorials.

Trello (Atlassian). Project Management Tool.