CameraGuard: Aplicación para gestionar cámaras domóticas

Carlos Enrique Reyes Euceda

Resumen— El propósito fundamental de este proyecto radica en la creación de una aplicación web con una interfaz gráfica simple e intuitiva, destinada a facilitar la gestión de cámaras de video en el ámbito de la seguridad del hogar. La conectividad entre estas cámaras de seguridad y la aplicación web se establecerá a través de un servidor en la nube, permitiendo al usuario visualizar en tiempo real las imágenes captadas por las cámaras y recibir alertas instantáneas en caso de detección de movimiento. Más allá de la funcionalidad pura, se pondrá un énfasis especial en la implementación de rigurosos protocolos de seguridad, garantizando así una experiencia fluida, privada y segura para los usuarios.

Palabras clave— Aplicación web, cámaras de seguridad, gestión remota, interfaz gráfica, servidor en la nube, alertas de movimiento, privacidad, seguridad en el hogar, experiencia del usuario, protocolos de seguridad.

Abstract— The primary objective of this project is to develop a web application featuring a straightforward and user-friendly graphical interface, specifically designed for the efficient management of video cameras within the realm of home security. The seamless communication between these security cameras and the web application will be facilitated through a cloud server. Through this interface, users will have the capability to observe real-time footage from the security cameras and promptly receive alerts in the event of motion detection. In addition to delivering a fully functional application, utmost attention will be given to ensuring the implementation of stringent security protocols, ensuring users a smooth, private, and secure experience.

Index Terms— Web Application, video-cameras, remote management, graphical interface, cloud server, motion alerts, privacy, home security, user experience, security protocols.

1 Introducción - Contexto del trabajo

Imaginemos por un momento la preocupación constante por la seguridad de nuestros hogares o negocios. Las noticias sobre robos, intrusiones o incidentes inesperados pueden generar una sensación de vulnerabilidad y estrés. Ante esta realidad, surge la necesidad de encontrar una solución que nos proporcione tranquilidad y control sin necesidad de recurrir a costosos sistemas de seguridad.

Este proyecto se centra en abordar la preocupación por la seguridad de manera efectiva y accesible para todos. La problemática que enfrentamos es la falta de una herramienta que nos permita supervisar y gestionar nuestros sistemas de seguridad de manera sencilla, eficiente y económica, sin importar dónde nos encontremos.

La solución que ofrece este proyecto es una aplicación web que brinda a las personas la capacidad de controlar sus cámaras de seguridad desde cualquier dispositivo con conexión a Internet. Este enfoque elimina la necesidad de estar físicamente presente en el lugar, lo que a menudo es costoso y poco práctico. En su lugar, la aplicación proporciona la comodidad de la supervisión remota, permitiéndonos tomar medidas inmediatas en caso de una situación inesperada.

Este proyecto se llevará a cabo mediante el desarrollo de una aplicación web que empleará tecnologías avanzadas, como Angular para el *frontend*, Java para el *backend* y una base de datos MySQL. La aplicación desarrollada con estas tecnologías estará alojada en la plataforma de hosting Heroku, ya que permite desplegar aplicaciones con costos muy razonables. Además, existirá la posibilidad de integrar los dispositivos IoT para una vigilancia más completa. La seguridad y privacidad de los datos de los usuarios serán fundamentales, y se implementarán rigurosos mecanismos de protección.

En un mundo en constante evolución, la seguridad y la tranquilidad son invaluables. Este proyecto no busca solo ofrecer una solución tecnológica, sino proporcionar a las personas la capacidad de cuidar lo que más les importa. Al abordar la preocupación compartida por la seguridad de manera accesible y eficaz, se busca brindar a todos la paz mental que merecen en un entorno cada vez más complejo. La seguridad no es un lujo; es un derecho fundamental, y este proyecto lo coloca al alcance de todos.

En este documento se podrán encontrar secciones donde se explica detalladamente los objetivos que se han definido y que se esperan conseguir al final de este proyecto, también se puede encontrar la metodología utilizada para conseguir dichos objetivos y la planificación que se ha utilizado para garantizar las entregas en los tiempos especificados para una división de la carga de trabajo adecuada.

Todos los detalles sobre cómo se han desarrollado las distintas partes del proyecto y las tecnologías y herramientas utilizadas se encuentran en la sección de desarrollo. Además, los resultados y conclusiones obtenidos a partir del desarrollo de este proyecto se podrán encontrar en sus respectivas secciones.

2 OBJETIVOS

2.1 Estudio de dispositivos *IoT*

Estudiar e investigar el funcionamiento y protocolos que utilizan los dispositivos *IoT* que podrían integrarse en este proyecto.

2.2 Interfaz gráfica simple e intuitiva

Estudiar e investigar sobre las herramientas y tecnologías que existen para diseñar una interfaz gráfica que sea amigable al usuario, fácil de utilizar para todo tipo de consumidores y brindar una experiencia fluida.

2.3 Estudio de datos

Investigar sobre los protocolos que se utilizan para enviar datos a través de internet, así como las distintas formas en las que los datos se pueden almacenar, comprimir y transferir.

2.4 API REST

Desarrollar una aplicación de servicios REST al que se le puedan hacer peticiones y que lleve toda la parte de la lógica de una aplicación.

2.5 Seguridad y privacidad

Utilizar mecanismos de seguridad para asegurar la privacidad del usuario y de los datos que se manipulan al utilizar la aplicación.

2.6 Accesibilidad y disponibilidad

Alojar la aplicación en un servidor para que sea posible acceder a través de internet y en cualquier momento a la aplicación web con las credenciales del usuario para poder ver a través de las cámaras de seguridad en tiempo real, así como recibir alertas de movimiento.

3 REQUISITOS DEL SISTEMA

3.1 Requisitos funcionales

1. **Inicio de Sesión:**
   - Permitir al usuario iniciar sesión con sus credenciales.
   - Redireccionar al usuario a la pantalla principal después del inicio de sesión.

2. **Reproductor de Video:**
   - Mostrar un reproductor de video en la pantalla principal.
   - Reproducir el video de las diferentes salas registradas.
   - Transmitir video en tiempo real mediante Web Sockets.

3. **Listado de Movimientos:**
   - Paginar el listado de movimientos para mostrar todos los registros almacenados.
   - Permitir hacer clic en un movimiento para visualizar una imagen correspondiente.

4. **Seguridad:**
   - Proporcionar servicios de autenticación y autorización a los usuarios.
   - Generar tokens para acceder a los servicios protegidos.
   - Implementar encriptación de datos.

5. **Registro de Usuarios:**
   - Permitir el registro de nuevos usuarios.
   - Almacenar credenciales de usuarios en una tabla de la base de datos.

6. **Notificación de Movimientos:**
   - Enviar notificaciones por correo electrónico al detectar un movimiento.

7. **Detección de Movimiento:**
   - Utilizar la librería OpenCV para la detección de movimiento.
   - Aplicar filtros para reducir falsos positivos.

3.2 Requisitos no funcionales

1. **Interfaz Gráfica:**
   - Diseñar la interfaz gráfica utilizando la herramienta Figma.
   - Implementar la interfaz en Angular.

2. ***Backend*:**
   - Desarrollar el *backend* con el *framework* Spring.
   - Configurar una base de datos local en MySql mediante MySql Workbench.
   - Utilizar Swagger para documentar servicios REST.

3. **Transmisión de Video en Tiempo Real:**
   - Utilizar Web Sockets para la transmisión fluida de fotogramas.
   - Simular la transmisión de video mediante el envío de fotogramas en intervalos.

4. **Despliegue:**
   - Desplegar la aplicación en Heroku para alojar aplicaciones Angular y Spring.
   - Configurar conexiones con JawsDB para la base de datos MySQL.

- Utilizar buildpacks para la instalación de librerías, incluyendo OpenCV.

4 METODOLOGÍA

4.1 Desarrollo

Como metodología de desarrollo he decido implementar la metodología Cascada (*Waterfall*). En este enfoque, las etapas del proyecto se desarrollan secuencialmente, una tras otra y cada fase se debe completar antes de pasar a la siguiente [1]. Dado que este proyecto tiene un objetivo final bien definido y no hay muchas restricciones de tiempo ni de presupuesto. Esta metodología es la más adecuada a las características del proyecto.

4.2 Herramientas de Software

Se utilizará Notion para planificar las tareas y fases del proyecto para garantizar entregas y obtención de objetivos a tiempo. Para el desarrollo de la aplicación se utilizarán Visual Studio Code y IntelliJ IDEA para escribir y ejecutar código y MySql Workbench para crear la base de datos. Los progresos que se vayan consiguiendo y las versiones del sistema que se vayan desarrollando se almacenaran en un sistema de control de versiones Git en local y GitHub en remoto.

4.3 Paradigma

Para este proyecto he decidido seguir el paradigma *frontend-backend* que consiste en la separación de una aplicación en una cliente o parte gráfica y otra parte de lógica o servidor que se comunican entre si mediante peticiones HTTP. El *backend* estará desarrollado con el lenguaje de programación Java y el *framework* Spring ya que permite desarrollar aplicaciones de servicios REST para comunicarse con una base de datos MySql mientras que el *frontend* estará desarrollado con el *framework* Angular.

4.4 Mecanismos de Seguridad

Para proteger y garantizar la seguridad de los datos y los usuarios, se implementarán mecanismos de seguridad como la encriptación de datos y la autenticación por medio de JWT y Spring Security.

5 PLANIFICACIÓN

La carga de trabajo se dividió en semanas, se implementó una metodología de desarrollo en cascada (*Waterfall*) y se utilizará Notion para la creación de tareas y subtareas con fechas límite para garantizar entregas y promover un desarrollo continuo.

Semana 1-3: Diseño y Desarrollo del *frontend*

- Semana 1: Diseño de la interfaz de usuario y creación de prototipos en Angular. Configuración del entorno de desarrollo.
- Semana 2: Continuación del diseño de la interfaz de usuario y prototipos. Inicio del desarrollo del *frontend*, implementando la estructura inicial.
- Semana 3: Desarrollo continuo del *frontend*, implementando las funcionalidades según los diseños. Pruebas de unidad en el *frontend*.

Semana 4-6: Desarrollo del *backend*

- Semana 4: Inicio del desarrollo del *backend* en Java. Configuración del entorno de desarrollo para el *backend*.
- Semana 5: Continuación del desarrollo del *backend*, implementando la lógica del sistema. Realización de pruebas de unidad en el *backend*.
- Semana 6: Desarrollo continuo del *backend*. Ajustes y correcciones en el *frontend* y *backend* según los resultados de las pruebas.

Semana 7-10: Integración *frontend* y *backend*

- Semana 7: Integración inicial entre el *frontend* y el *backend*. Pruebas de integración para garantizar la comunicación adecuada.
- Semana 8: Continuación de la integración entre el *frontend* y el *backend*. Ajustes y correcciones basados en los resultados de las pruebas de integración.
- Semana 9: Desarrollo de un sistema de autenticación utilizando JWT con Spring Security en el *backend* integrado con el formulario de inicio de sesión del *frontend.*
- Semana 10: Investigar opciones de alojamiento para *frontend* y *backend* y también maneras de hacer un *streaming* de video en la aplicación.

Semana 11-13: Desplegar aplicación y alertas

- Semana 11: Desplegar ambas aplicaciones en un servidor.
- Semana 12: Configurar el envío de alertas mediante correo electrónico.
- Semana 13: Pruebas finales del proyecto.

Semana 14-16: Redacción del informe final

- Semana 14 y 15: Redacción de la propuesta de la última versión del informe.

- Semana 16: Revisión de errores gramaticales y formato

Semana 17-18: Dossier

- Semana 17: Corrección de errores y redacción de la versión definitiva del informe.
- Semana 18: Crear dossier e incluir toda la documentación final.

6 DESARROLLO

6.1 Diseño

Siguiendo la planificación establecida, empecé con el diseño de la interfaz gráfica de la mano de la herramienta Figma. Elegí esta herramienta ya que después de investigar sobre herramientas de diseño descubrí que esta es muy completa y fácil de utilizar [2].

Apoyándome en los requisitos del sistema, decidí hacer un diseño que me permita realizar todas las funcionalidades descritas y también algunos componentes que podrían servir para agregar nuevas funcionalidades en un futuro. Los resultados del primer diseño fueron satisfactorios y por eso me quede con este.

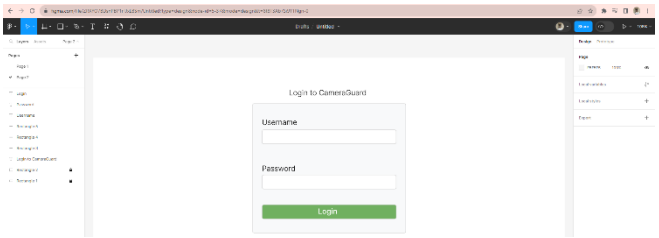

Fig. 1. Diseño de pantalla de inicio de sesión en Figma

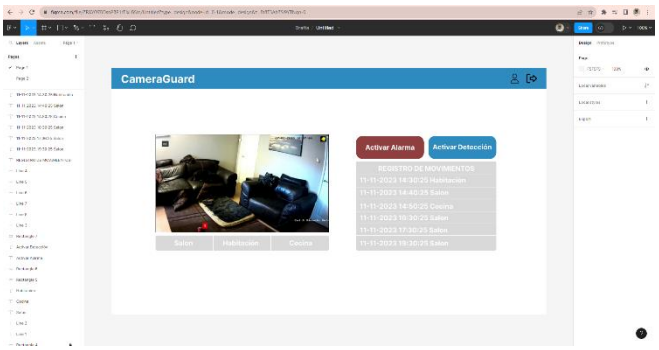

Fig. 2. Diseño de pantalla principal en Figma

6.2 Implementación

Investigue diferentes *frameworks* populares que se utilizan para desarrollar interfaces en un entorno web, entre estas se encuentran React, Vue.js y Angular. Si bien cada una tiene sus méritos, Angular destaca por su enfoque integral y su arquitectura robusta. Angular ofrece un conjunto de herramientas coherente y bien estructurado que facilita la creación de aplicaciones complejas y escalables. Su sistema de dos vías de enlace de datos, su amplia gama de módulos y su integración fluida con TypeScript brindan un entorno de desarrollo estructurado y fácil de mantener. Además, la comunidad activa y el respaldo de Google garantizan actualizaciones regulares y un soporte sólido [3].

Una vez diseñada la interfaz solo me hizo falta implementar las pantallas en un proyecto de Angular, creando componentes para la barra de navegación, video, página principal e inicio de sesión. Para facilitar el desarrollo de la interfaz y brindar una apariencia más atractiva de esta utilicé las librerías Bootstrap [4] y Angular Material [5]. Los resultados de la implementación en Angular fueron los siguientes:

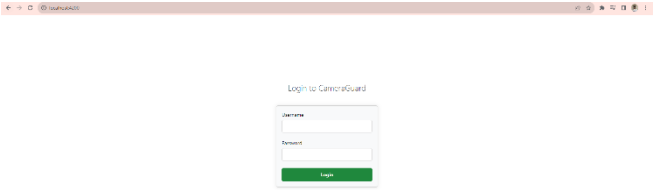

Fig. 3. Diseño de pantalla de inicio de sesión con Angular

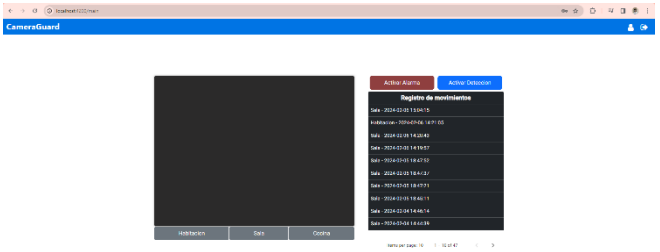

Fig. 4. Diseño de pantalla principal con Angular

El recuadro donde se va a mostrar el video en realidad es una imagen que irá cambiando con cada fotograma que se reciba para finalmente formar un video con estas secuencias.

Este diseño de interfaz va a permitir al usuario iniciar sesión con sus credenciales y después se le redirigirá a la pantalla principal donde hay un reproductor de video capaz de reproducir el video de las diferentes salas registradas, así como visualizar un listado de movimientos registrados que posteriormente se les podrá hacer clic para visualizar una imagen del movimiento. Este listado esta paginado para poder mostrar todos los registros almacenados en la misma pantalla. También existe un botón de cerrar sesión y uno de activar alarmas que podrá ser útil en un futuro si se llega a integrar la aplicación con una alarma real.

6.3 *Backend*

Tras finalizar el desarrollo del *frontend*, lo siguiente a desarrollar en la planificación fue el *backend*. Desde un principio la idea fue utilizar el *framework* Spring y después de investigar otras opciones, decidí seguir adelante con Spring ya que muchos destacan su compatibilidad con Angular [6] y basado en mi experiencia puedo decir que se adecua muy bien al proyecto que estoy desarrollando.

Empecé creando una base de datos local en MySql mediante la interfaz de MySql Workbench. Después cree una tabla para almacenar los movimientos detectados por las cámaras de seguridad, esta contiene campos que posteriormente se mostraran por pantalla en la aplicación como pueden ser fecha y hora, un identificador del movimiento y otro del usuario y una imagen en formato "BLOB" [7] que por sus siglas en ingles significa objeto binario grande. También creé otra tabla para almacenar credenciales de usuarios para posteriormente configurar un sistema de autenticación y una tabla para almacenar los espacios que le corresponden a cada usuario.

El siguiente punto en la planificación requería tener montada ya la base de datos con algunos datos de prueba para poder configurar el proyecto de Spring y crear una API para poder comunicar el *frontend* con el *backend* mediante servicios REST [8]. Para facilitar la fase de pruebas a todos los servicios que iba desarrollando configuré un entorno de documentación de servicios REST con Swagger [9]. Es similar al muy conocido Postman [10] pero con la diferencia de que Swagger documenta los servicios de forma automática y es muy fácil de configurar en un proyecto de Spring.

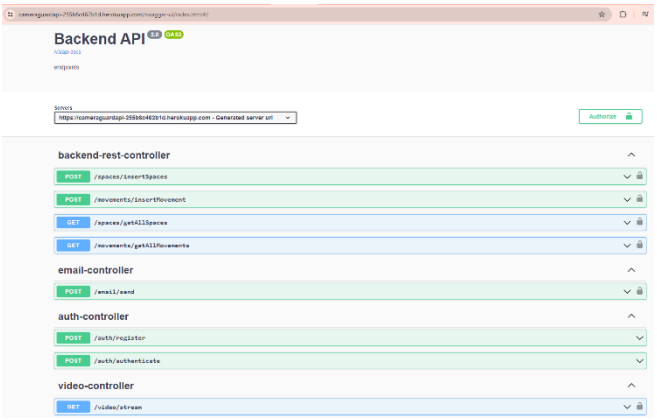

Fig. 5. Servicios del *backend* en Swagger

En esta interfaz existen servicios de autenticación desarrollados con Spring Security y JWT que permiten el acceso a usuarios mediante credenciales.

Los JWT (JSON Web Tokens) son un formato compacto y autenticado para la transmisión segura de información entre partes. Funcionan como tokens que contienen información en formato JSON, firmadas digitalmente para verificar su integridad [11]. Consisten en tres partes: encabezado, carga útil y firma. El encabezado especifica el tipo de token y el algoritmo de firma, la carga útil contiene información sobre el usuario o entidad, y la firma garantiza que el token no ha sido alterado. Estos tokens son utilizados comúnmente en sistemas de autenticación y autorización en aplicaciones web y API, permitiendo la verificación eficiente de la identidad y la validez de los datos transmitidos entre el cliente y el servidor.

El servicio de registro permite dar de alta a un usuario en la base de datos, guardando el nombre de usuario, correo electrónico y contraseña codificada con la herramienta PasswordEncoder de Spring Security. Con estas credenciales el usuario podrá autenticarse con el servicio de autenticación AuthenticationManager que nos brinda también Spring Security. Después de que un usuario se autentica se genera un token que posteriormente se necesitará para llamar a cualquiera de los servicios disponibles a excepción del propio servicio de autenticación. Si alguna llamada a un servicio de las que necesita autenticación previa no lleva token de autenticación se devuelve un error.

También podemos encontrar servicios para manipular los datos de las tablas de la base de datos. Estos servicios incluyen funcionalidades que podemos encontrar en cualquier CRUD básico como insertar y leer.

6.4 Transmisión de video en tiempo real

Una de las funcionalidades más críticas que define este proyecto es la transmisión en vivo de video, para esto no era una buena idea utilizar un servicio REST ya que se debe enviar un flujo continuo de fotogramas desde el *backend* hasta el *frontend* sin necesidad de que el *frontend* haga una petición por cada uno de los fotogramas. Por esta razón decidí investigar alternativas que me permitieran establecer un canal de comunicación abierto entre ambas partes del proyecto.

La alternativa que más se ajustaba a mis necesidades era la del uso de Web Sockets [12] ya que me permite enviar información desde el *backend* sin necesidad de hacer ninguna petición desde el *frontend* mediante una librería nativa de Spring. La manera en la que funciona es mediante una conexión persistente, bidireccional y de baja latencia entre el servidor y el cliente, permitiendo una comunicación instantánea y eficiente. El cliente hace una petición que contiene como parámetro el nombre del espacio que se desea visualizar y el servidor procesa la solicitud. Una vez recibido el nombre del espacio el *backend* empieza a enviar uno a uno los fotogramas que serán representados en forma de video en el *frontend*.

Cabe destacar que al no tener acceso a el video retransmitido por una cámara de vigilancia he tenido que simular la transmisión de un video grabado previamente en formato mp4 a 30 fotogramas por segundo. Este video esta almacenado en el directorio de recursos del proyecto y se fragmenta en fotogramas mediante el uso de la librería que se utiliza posteriormente para la detección de movimiento OpenCV [13]. Cada fotograma se convierte a un formato de base 64 y después se envía en un intervalo de 33.34 milisegundos para simular una transmisión de una cámara de seguridad que graba a 30 fotogramas por segundo y que

después pueda ser reconvertido a imagen y retransmitido como un video en tiempo real en el *frontend*.

6.5 Detección de movimiento

A pesar de haber desarrollado la transmisión de video alrededor de la premisa de un video en formato mp4 fragmentado en fotogramas, solo bastaría con cambiar la ruta del video por una dirección IP y un par de líneas más para que la aplicación funcione correctamente con una cámara de seguridad IP.

La detección de video se ha desarrollado, como se menciona anteriormente, mediante el uso de la librería de visión por computador OpenCV ya que es la más utilizada gracias a su política de código abierto. Esta librería se ha de instalar en el sistema que ejecuta la aplicación de Spring ya que desde el código solo es posible importar una dependencia que se comunica con la ya previamente instalada librería de OpenCV. Esto es algo importante a tener en cuenta a la hora de desplegar la aplicación en un entorno no local.

Para detectar movimiento entre fotogramas he utilizado la una estrategia donde se primero se selecciona el fotograma actual y anterior y se convierten a una escala de grises para reducir la cantidad de información a procesar ya que el color no es necesario a la hora de aplicar la detección de movimiento y solo agrega complejidad innecesaria al procesamiento de las imágenes. Después se les aplica un desenfoque gaussiano para suavizar la imagen y eliminar el ruido. Gracias a esto evitamos los movimientos falsos positivos que se pueden dar por pequeñas variaciones de pixeles que puede haber en el entorno, por ejemplo, los cambios de iluminación. El siguiente paso es calcular la diferencia entre el fotograma actual y anterior. Por último, se aplica un umbral adaptativo para resaltar las diferencias significativas entre los fotogramas al establecer un límite adaptativo según las condiciones locales de la imagen.

Esta imagen con todos los filtros aplicados pasa después por una función que determina todos los contornos de la imagen y resalta en color rojo estos contornos para una más fácil identificación del movimiento.

6.6 Alertas

La detección de movimiento se realiza siempre antes de enviar cada fotograma y si se detecta un movimiento, de forma asíncrona para no retrasar el envío del fotograma, se insertan los datos del movimiento en la tabla de movimientos de la base de datos e inmediatamente después se envía un aviso por correo con los datos del movimiento y una imagen. Este envío de correo se hace mediante el uso de la plataforma Mailgun [14] ya que me permite enviar hasta 10,000 correos con el plan gratuito, además se puede integrar muy bien con muchos de las plataformas de alojamiento de aplicaciones.

Fig. 6. Correo recibido tras haber detectado un movimiento

6.7 Integración

Después de haber desarrollado ambas partes del proyecto por separado llego el momento crucial en el desarrollo de integrar *frontend* y *backend* para tener un sistema funcional.

Para lograr la comunicación entre el *frontend* y el *backend*, se implementaron solicitudes HTTP [15] utilizando el protocolo REST. Angular facilita la realización de estas solicitudes mediante servicios, lo que permitió consumir los servicios proporcionados por el *backend* de manera sencilla. Se gestionaron aspectos como la autenticación, almacenando el token JWT generado durante el proceso de inicio de sesión para autorizar las solicitudes al *backend*. Las peticiones se autorizaban con el uso de la librería HTTP Interceptor de Angular que permite interceptar todas las peticiones HTTP que salen del cliente para agregar en la cabecera el token generado previamente.

Se estableció un filtro CORS (Cross-Origin Resource Sharing) [16] para facilitar la comunicación entre el *frontend* en Angular y el *backend* en Spring. Este filtro configura que dominios pueden hacer peticiones y que tipo de peticiones están permitidas. Esto posibilita la interacción entre las partes del sistema y añade una capa más de seguridad.

Para llevar a cabo la integración de la parte del Web Socket utilice las librerías SockJS para crear un *socket* y StompJS para iniciar la conexión con la parte del *backend*. Mediante una suscripción al servicio que envía los fotogramas, la interfaz actualiza el video con cada fotograma que recibe.

Se llevaron a cabo pruebas exhaustivas para garantizar que la integración entre el *frontend* y el *backend* funcionara correctamente. Se verificaron aspectos como la correcta transmisión de datos entre las capas, la gestión de errores y la sincronización de las operaciones. Estas pruebas incluyeron pruebas para asegurar la estabilidad y seguridad del sistema.

Utilizando la DevTools se puede inspeccionar la aplicación Angular para revisar que efectivamente se establece la conexión con el web socket y que se envían los fotogramas de uno en uno.

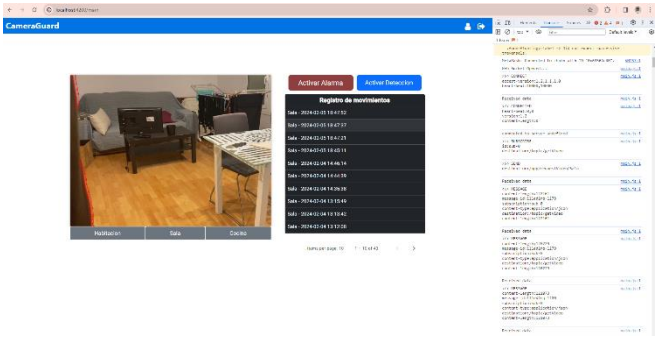

Fig. 7. Paquetes enviados por servidor

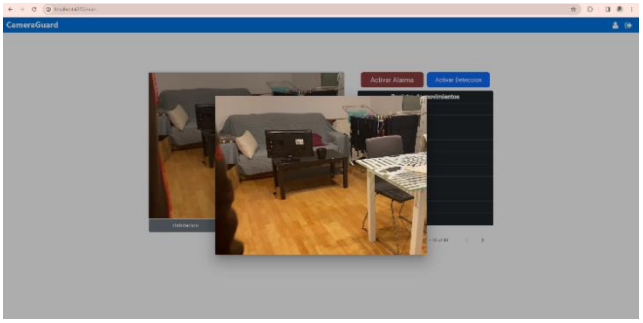

Fig. 8. Registro de imágenes

También se puede comprobar que la funcionalidad de hacer clic en un registro de la tabla de movimientos para visualizar una imagen del movimiento.

6.8 Alojamiento del servidor

Existían muchas opciones populares en el mercado para alojar aplicaciones Angular y Spring, como AWS (Amazon Web Services), Google Cloud Platform, y Microsoft Azure, entre otros. Sin embargo, Heroku [17] destaca como una opción especialmente atractiva, especialmente para estudiantes, debido a su enfoque centrado en la simplicidad, la facilidad de uso mediante sus "buildpacks" y su generosa política de créditos gratuitos.

Heroku es una plataforma en la nube que utiliza el concepto de buildpacks para simplificar y automatizar el proceso de implementación de aplicaciones. Un buildpack es un conjunto de scripts que instruyen a Heroku sobre cómo construir y configurar el entorno de ejecución para una aplicación específica. En este caso, se ha utilizado un buildpack de Java para la aplicación Spring y un buildpack de Node.js para la aplicación Angular. Cuando despliegas una aplicación en Heroku, los buildpacks correspondientes son automáticamente detectados y aplicados. El buildpack de Java se encargará de instalar las dependencias de Java y configurar el entorno de ejecución necesario para la aplicación Spring, mientras que el buildpack de Node.js hará lo mismo para la aplicación Angular. Esto simplifica el proceso de implementación, ya que no es necesario preocuparse por la configuración del entorno y permite centrarse en el desarrollo de las aplicaciones.

Para el *backend*, se configuró la aplicación Spring y se provisionó una base de datos MySQL en JawsDB [18]. Una vez obtenidas las credenciales de JawsDB establecí la conexión desde el *backend*. Utilizando MySQL Workbench, exporté la base de datos local y posteriormente la importé en la base de datos remota de JawsDB.

Las credenciales de la aplicación *backend* que estaban configuradas para la base de datos local fueron actualizadas con las nuevas credenciales de JawsDB. Además, ajusté las configuraciones del CORS para aceptar peticiones del nuevo dominio del *frontend* alojado en Heroku.

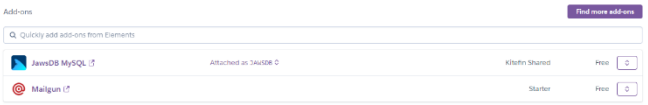

Fig. 9. Provisionamiento del servidor *backend* en Heroku

A parte del provisionamiento de la base de datos y del servicio de correos electrónicos hacia falta de alguna forma instalar la librería de OpenCV en el servidor, para esto existe una solución en la que mediante los buildpacks de Heroku se pueden instalar librerías y herramientas. Lamentablemente no existe ningún buildpack ya existente para instalar OpenCV pero sí que existe un uno llamado "heroku-buildpack-apt" que permite instalar una gran variedad de librerías en el entorno basado en Linux donde se despliegan los servidores mediante un fichero llamado "Aptfile". Para este proyecto la librería que se incluyó en este fichero fue "libopencv-dev".

Para el *frontend* configuré un archivo de servidor donde se definen las rutas para que Heroku pueda desplegar la aplicación correctamente.

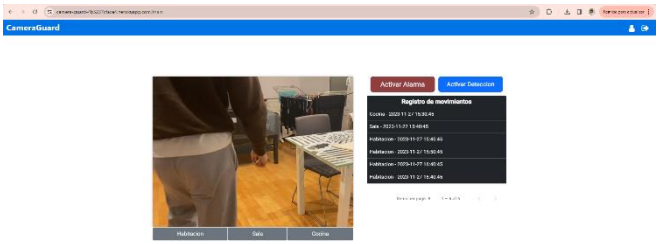

Fig. 10. Reproducción de un video de prueba en el servidor

Ambas partes tanto *frontend* como *backend* se configuraron para desplegarse directamente desde sus respectivos repositorios en GitHub. Heroku se conecta a los repositorios y los despliega cada vez que se sube un cambio a GitHub.

6.9 Planificación

Desde el inicio de este proyecto he ido siguiendo la planificación inicial definida en Notion, lo cual me ha ayudado a dosificar la carga de trabajo siguiendo la metodología Cascada. Sin embargo, dados los imprevistos que surgieron la planificación fue adaptada a la situación y al no tener los mismos objetivos también cambiaron los planes. En general la estimación de las tareas fue adecuada sin embargo la carga de trabajo en las fases de desarrollo del *backend* y

de integración fue mayor a lo esperado y menos de lo esperado en la fase de desarrollo y diseño del *frontend*.

7 *TESTS*

Se llevaron a cabo pruebas exhaustivas de usabilidad y funcionamiento mediante el *User Testing* para garantizar la eficacia de la aplicación. Los usuarios realizaron tareas específicas, proporcionando datos cuantitativos y cualitativos.

7.1 Escenario de Pruebas:

Se diseñó un escenario detallado que simula situaciones comunes, guiando a los participantes en la realización de tareas clave.

7.2 Resultados:

Los usuarios expresaron alta satisfacción, destacando la navegación intuitiva y la eficiencia en la gestión de cámaras. Las pruebas de funcionamiento confirmaron la estabilidad y respuesta adecuada del sistema.

7.3 Mejoras Iterativas:

Basándonos en los comentarios, se implementaron mejoras en el diseño y la funcionalidad de manera iterativa para adaptarse a las necesidades de los usuarios.

8 CAMBIOS

8.1 Objetivos

Dado que no fui capaz de adquirir los componentes de hardware a tiempo para seguir con la planificación he decidido cambiar el objetivo de seguridad para brindar seguridad al usuario desde otra perspectiva, enfocándome en la autenticación. También decidí cambiar el objetivo de montar un servidor que se encargue de comunicar los dispositivos IoT con la aplicación por la opción de alojar la aplicación con un servicio en línea.

8.2 Planificación

Por la misma razón que en los cambios de los objetivos he ajustado la planificación de acorde a los avances conseguidos y a los nuevos objetivos.

9 RESULTADOS

En el transcurso de este ambicioso proyecto, se han logrado avances sobresalientes que han consolidado el éxito de la aplicación CameraGuard. La cuidadosa concepción de la interfaz gráfica, meticulosamente diseñada con Figma y posteriormente implementada en Angular, ha culminado en una experiencia de usuario no solo funcional sino también agradable y accesible. En paralelo, el *backend*, construido con Spring y MySQL, ha demostrado su robustez a través de la implementación de servicios REST que facilitan la comunicación efectiva con la base de datos. Aspectos cruciales como la encriptación de datos, autenticación con JWT y la integración de Spring Security han fortalecido la seguridad del sistema.

La transmisión de video en tiempo real, un componente esencial de la aplicación se ha llevado a cabo con maestría mediante el uso de Web Sockets. Esta elección tecnológica ha permitido una transmisión fluida e ininterrumpida de los fotogramas, contribuyendo significativamente a la calidad general de la aplicación.

La integración sin fisuras entre el *frontend* y el *backend* se logró mediante solicitudes HTTP y el protocolo REST. La implementación de un filtro CORS ha facilitado la comunicación entre las diversas partes del sistema, y las pruebas exhaustivas realizadas abarcaron desde la transmisión precisa de datos hasta la gestión efectiva de errores y la sincronización de operaciones. Este enfoque meticuloso ha garantizado la estabilidad y seguridad del sistema.

El despliegue exitoso en Heroku ha permitido que la aplicación esté disponible de manera gratuita, brindando accesibilidad a un público más amplio. La configuración cuidadosa de conexiones con JawsDB y la actualización regular de credenciales han asegurado un funcionamiento óptimo en el entorno en línea.

Aunque se han cumplido con los objetivos propuestos, se reconoce que el sistema tiene margen para mejoras continuas, con la mira puesta en su evolución hacia un producto comercializable y rentable en el futuro.

10 CONCLUSIONES

La elección de la metodología de desarrollo Cascada ha demostrado su eficacia en el marco de este proyecto, aunque no exenta de desafíos. La interrelación estrecha entre diferentes partes del sistema ha generado la necesidad ocasional de ajustes retrospectivos, señalando la importancia de una planificación flexible y adaptativa. La distribución de tareas, aunque realizada de forma secuencial, se ha visto afectada por contratiempos y fallos de integración, subrayando la necesidad de una aproximación más flexible en futuros proyectos.

Las herramientas y tecnologías seleccionadas, desde Notion hasta Visual Studio Code, IntelliJ IDEA y MySQL Workbench, han demostrado su eficacia en todas las fases del proyecto. La elección cuidadosa de estas herramientas ha contribuido al éxito general del desarrollo y la implementación.

Las adaptaciones motivadas por la falta de hardware han llevado a un enfoque estratégico en la autenticación y el alojamiento en línea. A pesar de estos ajustes, el proyecto mantiene su enfoque primordial en proporcionar seguridad y tranquilidad a los usuarios, ofreciendo un control efectivo y accesible para la supervisión remota de cámaras.

La clave del éxito continuo de este proyecto reside en su capacidad de adaptación y monitoreo constante para responder a las cambiantes necesidades de los usuarios y explorar futuras oportunidades, como la integración de

dispositivos IoT. Más allá de ser simplemente una solución tecnológica, este proyecto busca dotar a las personas con la capacidad de proteger lo que más les importa en un entorno cada vez más complejo.

BIBLIOGRAFÍA

[1] Anastasia Stsepanets, "Modelo de cascada (Waterfall): qué es y cuándo conviene usarlo" https://blog.ganttpro.com/es/metodologia-de-cascada/ , 2023

[2] Ben Kopf, "The Power of Figma as a Design Tool", https://www.toptal.com/designers/ui/figma-design-tool.

[3] Udit Handa, "7 razones para usar Angular para sus aplicaciones web en 2021", https://cynoteck.com/es/blog-post/reasons-to-use-angular-for-your-web-app/

[4] Bootstrap, "Build fast, responsive sites with Bootstrap", https://getbootstrap.com/

[5] Angular Material, "Getting Started with Angular Material", https://material.angular.io/guide/getting-started

[6] Alejandro Ugarte, "Building a Web Application with Spring Boot and Angular", https://www.baeldung.com/spring-boot-angular-web, 2022

[7] European Knowledge Center for Information Technology (Ed.), "¿Qué es un BLOB en el mundo de las TIC?", https://www.ticportal.es/glosario-tic/blob-binario, 2018

[8] RedHat, "API REST", https://www.redhat.com/es/topics/api/what-is-a-rest-api, 2023

[9] Swagger, "About Swagger", https://swagger.io/about

[10] Yanina Muradas, "Qué es Postman y primeros pasos", https://openwebinars.net/blog/que-es-postman, 2019

[11] Spring, "OAuth 2.0 Resource Server JWT", https://docs.spring.io/spring-security/reference/servlet/oauth2/resource-server/jwt.html

[12] Spring, "Using WebSocket to build an interactive web application", https://spring.io/guides/gs/messaging-stomp-websocket

[13] Euroinnova International Online Education, "¿Qué es Open CV?", https://www.euroinnova.edu.es/nuevas-tecnologias/articulos/que-es-open-cv

[14] Mailgun, "Servicios de envío masivo de emails", https://www.mailgun.com/es/soluciones/casos-de-uso/email-marketing/servicios-envios-masivos-email/#:~:text=Mailgun%20ofrece%20una%20interfaz%20de,API%20y%20herramientas%20para%20desarrolladores

[15] MDN Contributors, "Generalidades del protocolo HTTP", https://developer.mozilla.org/es/docs/Web/HTTP/Overview, 2022

[16] Amazon, "Habilitación de CORS para un recurso de la API de REST", https://docs.aws.amazon.com/es_es/apigateway/latest/developerguide/how-to-cors.html

[17] Ricardo, "¿Qué es Heroku? Cómo funciona la plataforma y para qué sirve", https://platzi.com/blog/que-es-heroku/, 2018

[18] Heroku, "JawsDB MySQL", https://elements.heroku.com/addons/jawsdb

- *E-mail de contacte: 1605425@uab.cat*

- *Menció realitzada: Enginyeria del Software*
- *Treball tutoritzat per: Jordi Serra Ruiz (Àrea de Ciències de la Computació i Intel·ligéncia Artificial)*
- *Curs 2023/24*