CAMPUS
DE EXCELENCIA
INTERNACIONAL

Graduado en Ingeniería Informática

Universidad Politécnica de Madrid

Escuela Técnica Superior de
Ingenieros Informáticos

TRABAJO FIN DE GRADO

Desarrollo de la gestión de imágenes del observatorio astronómico Francisco Sánchez

Autor: Oscar Nicolás López

Director: Francisco Rosales García

MADRID, JULIO 2019

Resumen

Este proyecto consiste en el desarrollo de un nuevo módulo para la web del Observatorio Francisco Manuel Sánchez. El desarrollo de este nuevo módulo ha consistido en la creación de una nueva vista dedicada a la gestión de la visualización y captura de secuencias de imágenes a través de un reproductor, pudiendo almacenar y visualizar estas secuencias a través de una galería.

El nuevo módulo se ha desarrollado utilizando la API creada para el control de la cámara DMK del Observatorio, así como de la API para el Gateway de la web. La primera utilizada para la reproducción de las imágenes en directo, su almacenamiento en secuencias etiquetadas en palabras claves o “tags” y su visualización en la galería, mientras la segunda ha servido de apoyo para el control de acceso a la vista de visualización en directo y la muestra de la galería de cada usuario de la web en exclusiva.

Un usuario registrado en la web será capaz de controlar la emisión y recepción de las imágenes tomadas por la cámara acoplada al telescopio de forma que pueda ajustar los parámetros de esta a su gusto y, una vez encontrada la imagen deseada, guardarla en forma de secuencia que posteriormente podrá visualizar en su galería y compartir con el resto de usuarios a través de la búsqueda por tags.

Abstract

This Project will consist on the development of a new module for the Francisco Manuel Sanchez Observatory´s. This module will add two new views to the actual webpage, one dedicated to the streaming and capturing the output of the camera connected to the observatory´s telescope by making use of the API´s for the control of the camera previously developed, meanwhile the other one will be focused on the distribution and display of the images taken by this camera on dynamic gallery.

The user will be able to completely control the camera´s input and output of images and will be able to change the parameters of the camera freely and without restrictions, and, when found, save the image o images in a secuence that will later see displayed on his personal gallery, having the chance of seeing the other users pictures by searching them using tags.

Índice

Índice de figuras

1. INTRODUCCIÓN

El observatorio astronómico Francisco Manuel Sánchez se encuentra ubicado en la Escuela Técnica Superior de Ingenieros Informáticos (ETSIINF) de la Universidad Politécnica de Madrid. Este es el primer observatorio astronómico del mundo de acceso libre a través de internet. Inaugurado en el año 2009, el observatorio fue creado por el Grupo Cíclope [1].

El observatorio dispone de una página web donde, a través de la cual podemos acceder y controlar las distintas cámaras, la cúpula y la estación meteorológica, además del telescopio, ubicado en la azotea del bloque 6 de la ETSIINF.

Este proyecto surge debido a la necesidad de los usuarios de la web de almacenar y compartir las imágenes tomadas por ellos mismos a través de las distintas cámaras del observatorio, a las que se tiene acceso vía la web del OFS, partiendo de una serie de tres Trabajos de Fin de Grado realizados en los años anteriores a este, en los cuales, se desarrolló gran parte de la web ya descrita, centrándose, principalmente, en la gestión del registro de usuarios a la plataforma, una app para el control de los elementos del observatorio y una pasarela que da acceso a los usuarios al control de los dichos elementos.

El objetivo principal de este trabajo será, partiendo de todos estos avances, el poder generar una galería de imágenes donde cualquier usuario registrado en la página, pueda compartir las imágenes del firmamento de las cuales se sienta más orgulloso.

La captura de estas imágenes se llevara a cabo a través de la API creada en el último Trabajo de Fin de Grado realizado para el observatorio, el cual se ha enfocado en el control de las cámaras del observatorio, la captura de imágenes mediante estas y su almacenamiento en una base de datos. Además, se deberá desarrollar un sistema de almacenamiento de estas imágenes en un repositorio en la nube el cual sea propiedad del propio usuario, el diseño de toda la parte frontal de la web que esté enfocada a la galería y al control de estas cámaras y un sistema de comunicación entre la API ya existente en la web del observatorio y la nueva API creada para el control de las cámaras.

1.1 Objetivos

La finalidad de este trabajo es la de desarrollar un módulo de la aplicación web ya existente del Observatorio Astronómico Francisco Manuel Sánchez que consistirá en una galería de imágenes, una base de datos de tipo SQL donde almacenar la información de estas, un sistema de comunicación entre las distintas API´s que componen la web actual y el almacenamiento de estas imágenes en carpetas vía Google Drive.

A continuación se detalla la lista de objetivos a llevar a cabo:

- Manejo de herramientas para el desarrollo de aplicaciones web de tipo responsive.
- Diseño de interfaces web utilizando la tecnología AngularJS.
- Manejo de herramientas de control de versiones, Github en este caso.
- Integración y despliegue de los servicios y módulos en la web ya existente de gestión de experimentos.
- Desarrollo de una interfaz web responsive para la visualización y búsqueda por tags de las imágenes guardadas por los usuarios.
- Creación de un sistema de comunicación entre la API actual de la web y la API de control de las cámaras.
- Creación de una base de datos de tipo SQL donde almacenar la información de las imágenes capturadas.
- Creación de una interfaz de visualización en directo de la salida de la cámara conectada al telescopio.
- Creación de una interfaz intuitiva para el guardado de secuencias de imágenes tomadas por el telescopio.

2. ESTADO DEL ARTE

2.1 Introducción

En este capítulo, se hablará de la evolución de los observatorios astronómicos a lo largo de la historia, así como de los instrumentos de medición y sistemas de cómputo que se han utilizado en relación a la astronomía. Se abarcará desde los sistemas manuales de uso de los telescopios, a sistemas automatizados de control de estos a través de internet gracias a aplicaciones similares a la ofrecida por el OFS.

2.2 Historia de los observatorios astronómicos

Se define astronomía como la ciencia que se ocupa del estudio de los cuerpos celestes del universo, incluidos los planetas y sus satélites, los cometas y meteoroides, las estrellas y la materia interestelar, los sistemas de materia oscura, gas y polvo llamados galaxias y los cúmulos de galaxias; por lo que estudia sus movimientos y los fenómenos ligados a ellos.

Su registro y la investigación de su origen viene a partir de la información que llega de ellos a través de la radiación electromagnética o de cualquier otro medio. La mayoría de la información usada por los astrónomos es recogida por la observación remota [2].

La astronomía ha estado ligada al ser humano desde sus orígenes y es por esto, que es uno de los campos científicos más antiguos que se conocen, todas las civilizaciones antiguas han investigado en este campo científico.

Un observatorio astronómico es una institución desde la cual se investigan eventos y situaciones de tipo astronómico. Los primeros observatorios que se crearon en la historia fueron de este tipo y estaban destinados a observar fenómenos atmosféricos y astronómicos [3].

Los observatorios astronómicos más antiguos conocidos fueron construidos por los chinos y los babilonios sobre el año 2300 a.C. Estos observatorios eran, probablemente, grandes plataformas que permitían una visión del cielo sin obstáculos.

Sobre el 300 a.C. se construyó el más famoso observatorio de la antigüedad en Alejandría, en Egipto. Es probable que estuviera equipado con instrumentos tales como el astrolabio, con el que se podía medir la posición de las estrellas o planetas. Después de iniciarse la era cristiana, los árabes construyeron varios observatorios en Damasco y Bagdad, y en Mokatta, cerca de El Cairo.

El primer observatorio europeo se instaló en Nuremberg, Alemania, en 1471. Un siglo después el astrónomo danés Tycho Brahe construyó en la isla de Ven el gran observatorio Uraniborg. Este observatorio, en el que Brahe vivió y trabajó desde 1576 a 1596, se equipó con instrumentos utilizados para mediciones precisas de las posiciones de los cuerpos celestes. Las observaciones que hizo Brahe las utilizó el astrónomo alemán Johannes Kepler para desarrollar su teoría del Sistema Solar [4].

2.3 Historia de los observatorios astronómicos autónomos

Desde mediados del siglo XX, se empezaron a automatizar los primeros telescopios, estando su desarrollo, estrechamente relacionado con los telescopios fotoeléctricos. Podemos definir telescopio automatizado como un tipo de telescopio que realiza acciones, en este caso observaciones, de forma preprogramada y de manera autónoma sin ayuda humana.

En 1968, en la Universidad de Wisconsin, es inaugurado el primer telescopio automatizado. Este, controlado por un ordenador de 4 Kilobytes de memoria RAM, fue utilizado para realizar mediciones sobre una secuencia fija de estrellas durante la noche.

Ocho años más tarde, en 1976, el telescopio BTA-6, figura 2.1, es inaugurado en Rusia, con 6 metros de altura[5]. Este telescopio, ostento el título de ser el telescopio automatizado más grande del mundo hasta 1993 y sentó las bases que luego se usarían para la construcción de la montura del resto de telescopios.

Figura 2.1: Telescopio BTA-6.

Al BTA-6, le siguió por un año de diferencia el telescopio WIRO (Wyoming Infrared Observatory), construido por la Universidad de Wyoming.

En España, no es hasta 1984, año en el que el Carlsberg Meridian Telescope (figura 2.2) es trasladado a la isla de La Palma, que tuvimos el primer ejemplo de telescopio automatizado. Este telescopio, automatizado de forma conjunta entre el Instituto de Astronomía de Cambridge, la Universidad de Copenhague y el Real Instituto y Observatorio de la Armada, está situado en el Roque de los Muchachos, donde fue trasladado desde Holanda gracias a las condiciones atmosféricas que se dan en esta región [6].

Figura 2.2: Telescopio Carlsberg Meridian.

A finales de los 80 y hasta el inicio del siglo XXI, el incremento de telescopios automatizados fue subiendo de forma sustancial, dedicándose la mayoría de ellos a la observación de explosiones de rayos gamma. Estos se dedicaban a la realización de tareas de forma repetitiva.

Entre 1987 y 1988 se logró controlar un telescopio con comandos enviados a través de internet, estos observatorios autónomos podían controlar el posicionamiento los sistemas de guiado del telescopio. También se empezaron a introducir estaciones meteorológicas en los observatorios astronómicos, para poder predecir situaciones con condiciones atmosféricas adversas.

En la primera década del siglo XXI y gracias a la evolución de internet a escala global y la rapidez con la que avanza la tecnología, la cantidad de telescopios de tipo autónomo fue cada vez mayor, y con ellos, llegaron las redes de observatorios robóticos autónomos a escala global.

El Observatorio Francisco Manuel Sánchez, situado en la ETSIINF de la Universidad Politécnica de Madrid, debe su nombre en honor al profesor Francisco Manuel Sánchez, fundador del grupo Cíclope. Es el primer observatorio astronómico del mundo de acceso libre a través de internet.

El grupo Cíclope comenzó la idea de la construcción de un observatorio astronómico de acceso gratuito en el año 2002, durante la participación en el proyecto europeo. Sin embargo, no sería hasta 2004, con el proyecto LEARN-WEB (TSI-2004-04032), cuando se conseguiría financiación por parte del Gobierno de España, y hasta 2006, cuando se consiguió toda la financiación necesaria con la llegada del grupo Cíclope [8].

Las obras se comenzaron en 2006, y, tras varios años en construcción, fue inaugurado de forma oficial en 2009.

2.4 Estado actual de la web del Observatorio Francisco Sánchez

En la actualidad, la web del Observatorio Francisco Manuel Sánchez está compuesta por una parte frontal desarrollada en el año 2017 en AngularJS, que se apoya sobre dos APIs distintas, una, desarrollada en el mismo año que la parte frontal, dedicada al control de las reservas de los experimentos, el registro y control de sesión de los usuarios y el estado en tiempo real de la situación atmosférica del observatorio y otra, desarrollada entre el último trimestre de 2018 y el primer trimestre de 2019 dedicada a la visualización en tiempo real de las imágenes que envía la cámara conectada al telescopio, así como del guardado de secuencias de imágenes por usuario para almacenarlas en la base de datos.

A continuación, se explicará con detalle cada una de las partes que componen la web del observatorio.

2.4.1 Parte frontal de la web

La parte frontal de la página, a la que podemos acceder a través del enlace https://ofs.fi.upm.es, fue desarrollada a través del framework AngularJS, esta, usando el modelo vista-controlador que define al framework anteriormente citado, está compuesta de los siguientes módulos o vistas:

- **Inicio:** En esta vista se hace una introducción al Observatorio Francisco Manuel Sánchez. Justo a continuación de esta pequeña introducción, encontramos un desplegable desde el cual podemos seleccionar las cámaras que dan a la parte interior y exterior del Observatorio. También se muestra el tiempo meteorológico que hace en el observatorio y el estado en el que se encuentran los dispositivos instalados en este. Todo ello en tiempo real.

- **Equipamiento:** Esta vista nos muestra el equipamiento técnico del que dispone el Observatorio, así como una pequeña descripción de cada uno de los componentes que se nos muestran.

- **Acerca de:** En esta página se da información acerca del Observatorio, como por ejemplo, el año en el que fue inaugurado, su ubicación y una pequeña descripción. Además dispone de un carrusel de imágenes del Observatorio y sus creadores.

- **Contacto:** En esta vista, dividida en dos partes, podemos encontrar, situada a la izquierda, la dirección del Observatorio, enlaces a las redes sociales de las que dispone el grupo Ciclope y a su aplicación web. A la derecha podemos encontrar un mapa que se ha integrado en Google Maps, con vista en modo satélite, en el que se muestra la situación exacta del Observatorio.

- **Registrar:** A través de un botón situado en la parte superior derecha, en color rojo, bajo el nombre de "Registrar", podemos acceder a la vista de registro de usuarios a través de un formulario. Esta contiene 4 campos, Nombre de usuario, Email, Contraseña y Repetir contraseña. En caso de que un usuario nuevo quiera registrar y rellene los campos anteriormente citados de forma satisfactoria, se mostrará un mensaje comunicándole que recibirá un email para la confirmación de su registro.

- **Iniciar sesión:** Para acceder a esta vista, tendremos que clickar en un botón con la frase "Iniciar sesión" situado directamente a la izquierda del botón "Registrar". Este nos llevará a una pantalla en la que tendremos que introducir nuestro nombre de usuario y contraseña para poder realizar el inicio de sesión en la página.

- **Experimento:** En esta página el usuario puede realizar la solicitud de una reserva para realizar un experimento solar o nocturno. Una vez el usuario ha seleccionado el tipo de experimento del que quiere realizar la reserva, debe elegir una fecha concreta de entre una lista de fechas disponibles para realizar el experimento elegido.

- **Perfil:** Esta página muestra la información del usuario como el nombre de usuario y el email con los que se ha registrado, así como una tabla con las reservas de experimentos que tenga pendientes a partir de ese día, teniendo la posibilidad de eliminar una reserva o de activarla, en caso de activarla, será dirigido a la página de "observación" para realizar el experimento reservado.

- **Observación:** Esta página solo es accesible en la franja de tiempo en la que el usuario tenga hecha una reserva. Desde aquí se pueden llevar a cabo experimentos solares o nocturnos, con la posibilidad de controlar el telescopio del observatorio y apuntar a cualquier posición u objetivo. Al inicio de la página se puede ver el interior y exterior del observatorio, así como la vista de la cámara CCD acoplada al telescopio, con la opción de configurar los parámetros de la misma para poder sacar fotografías. Dependiendo del experimento que esté realizando, el usuario dispondrá de un botón para realizar un seguimiento, del sol en el caso del experimento solar o un seguimiento de la luna en caso del experimento nocturno.

2.4.2 API REST del gateway

Este API, de tipo Restfull, define una pasarela para el control remoto de los distintos dispositivos de los que dispone el Observatorio, así como de la consulta de su estado. Además, define todas las funcionalidades y llamadas que se deben realizar a la API para el control de registro de usuarios, la creación y gestión de las reservas para los experimentos solares y nocturnos. La documentación de estas funcionalidades está escrita siguiendo el estándar OpenAPI.

La definición de la API está escrita en un fichero en formato YAML y también está disponible de forma online en la URL https://ofs.fi.upm.es/swagger/.

El despliegue del servidor y de la pasarela se realiza a través de la tecnología de contenedores de tipo Docker, estos dos contenedores están ubicados en el ordenador que hace las veces de servidor situado en el laboratorio del departamento.

Todo el control de las funcionalidades del Gateway se lleva a cabo a través de llamadas de tipo Rest, siendo principalmente PUT, POST y GET a las distintas URI definidas en la API. A continuación, se muestran las llamadas más significativas que se pueden realizar, los cuerpos que deben llevar estas llamadas y sus respuestas:

- Control y registro de usuarios:
    - **Login de un usuario.** Se realiza a través de una petición POST a la url */login*, en el que se manda un objeto de tipo JSON con el nombre de usuario y contraseña. Si todo va bien, recibiremos el token necesario para realizar las peticiones autenticadas.
    - **Registro de un usuario.** Se realiza a través de una petición de tipo POST a la url */register* en la que se manda un objeto de tipo JSON que contendrá los campos username, password y email. En caso de que la petición haya ido de manera satisfactoria se registrará al usuario en la base de datos.
    - **Activación de un usuario.** La activación de usuario una vez ha realizado el registro se realiza a través de una petición POST a la url */users/{id}/actívate*. El cuerpo de la petición contiene el id del usuario que queremos activar.
    - **Información del usuario.** Realizaremos una petición de tipo GET a la url */users/loged*. Dado que es una petición de tipo GET, no contiene cuerpo, esta nos devolverá la información del usuario logeado.
    - **Información de todos los usuarios.** Si queremos obtener la información de todos los usuarios registrados en nuestra base de datos, realizaremos una petición GET a la url */users*, esta nos devolverá un JSON con los campos username, password, email y enabled de todos los usuarios de nuestro sistema.
- Gestión de las reservas:
    - **Listado de reservas.** Si se realiza una petición de tipo GET a la url */reservations*, obtendremos un listado con todas las reservas que haya entre las fechas que hayamos indicado.
    - **Creación de una reserva.** Para la creación de una nueva reserva, se realizará una petición PUT a */reservations* con los campos startDate y endDate, correspondientes al inicio y fin de las fechas de la reserva que queramos crear.
    - **Reservas de un usuario.** Petición GET a la url */reservations/own* que nos devolverá un listado con las reservas de las que es propietario el usuario en concreto.

- **Cancelación de una reserva.** Peticion de tipo PUT a la url */reservations/{id}/cancel*, donde el campo id es el identificador único de la reserva que queremos cancelar.

- **Compleción de una reserva.** Para marcar una reserva como completada, además de hacerse automáticamente una vez haya pasado la fecha fin de la reserva, se podrá realizar una petición de tipo PUT a */reservations/{id}/complete*, la cual pasará la reserva con el identificador {id} a completada.

- **Obtención de la reserva actual.** Si queremos consultar si existe una reserva actual, realizaremos una petición GET a */reservations/actual*, la cual nos devolverá, si existe, la reserva actual.

Control de estado de la estación meteorológica:

- **Estado de la estación.** Para comprobar el estado de la estación meteorológica se accederá desde la url */weatherstation/status* con una petición GET.

- Control de la cúpula del Observatorio:

  - **Estado de la cúpula.** La obtención de la información del estado de la cúpula se realiza a través de una petición de tipo GET a la url */dome/status.*

  - **Apertura de la cúpula.** Para la apertura de la cúpula se realiza una petición PUT a la url */dome/open* la cual realizará la apertura de la cúpula del Observatorio.

  - **Cierre de la cúpula.** Para realizar el cierre de la cúpula del Observatorio se realizará una petición PUT a */dome/close* la cual realizará el cierre de la cúpula del Observatorio.

- Control de la montura del telescopio:

  - **Estado de la montura.** El estado de la montura del telescopio puede ser consultado a través de una petición GET a la url */mount/status.*

  - **Movimiento de la montura a una posición concreta.** Para realizar movimientos de la montura a una posición en concreto, se realiza una petición de tipo PUT a la url */mount/move* la cual moverá la montura a las coordenadas que hayamos marcado en el cuerpo de dicha petición.

  - **Movimiento de la montura.** Si lo que se desea es mover la montura en una dirección en concreto, siendo las opciones arriba, abajo, izquierda o derecha, se realizará una petición POST a la url */mount/step* con la dirección hacia la que queremos mover la montura incluida en el cuerpo de la petición en un objeto de tipo JSON.

- Control de la cámara DMK:

    - **Estado de la cámara.** El estado de la cámara se consultará a través de una petición de tipo GET a la url */camera/status*.

    - **Configuración de la cámara.** La configuración de los parámetros de la cámara DMK se realiza a través de una petición de tipo PUT a la misma url utilizada para la comprobación del estado de la cámara en la que incluiremos un objeto JSON con los parámetros deseados.

    - **Captura de una fotografía.** Para la captura de imágenes a través de la cámara, se realiza una petición POST con el cuerpo vacío a la url */camera/takePhoto*, esta nos devolverá el id de la fotografía tomada.

    - **Obtención de una fotografía.** Una vez hayamos realizado la captura de una fotografía con la cámara y esta nos haya devuelto su id único, podremos obtener la fotografía a través de una petición GET a la url */camera/photo/{id}*, siendo el campo *{id}* el identificador único de la fotografía anteriormente capturada.

2.4.3 API REST para el control de la cámara DMK

Aunque ya se realizó a finales de 2017 un API para el control de la cámara DMK, esta resulto ser insuficiente si se quería llevar a cabo vídeos en directo y captura de imágenes a través de esta, por lo que, a finales del año 2018 y comienzos del año 2019 se realizó una nueva API Rest dedicada a la recepción y emisión de imágenes tomadas por esta cámara acoplada al telescopio del Observatorio y su almacenamiento en una base datos alojada en el servidor ubicado en el laboratorio del Grupo Cíclope.

El objetivo de este nuevo proyecto se centró exclusivamente en la obtención y control de los distintos parámetros de los que dispone la cámara DMK, como son el brillo, la gamma, la ganancia o el tiempo de exposición, así como la obtención y guardado en una base datos alojada en una Raspberry Pi de secuencias de imágenes. Estas secuencias serían guardadas en la base de datos con los campos usuario, tags (5 como máximo), identificador único y el nombre de estas. La base de datos es accesible a través de peticiones http, las cuales permiten el acceso a los recursos almacenados en esta base de datos.

Para el almacenamiento y creación de las tablas de la base de datos se utilizó el sistema de gestión de bases de datos SQLite y el desarrollo del lenguaje para la creación de las funcionalidades de la nueva API se realizó en Python.

Se realizaron los siguientes métodos a través de los siguientes tipos de llamadas:

- **GET.** Los métodos GET para este proyecto se usaron para acceder a los valores de cada uno de los parámetros de la cámara, para ver los estados de cada una de las actividades que se han mandado desde el cliente a la cámara, y, si la actividad en cuestión ha terminado, se mandan cada una de las URIs de las imágenes para que puedan ser visualizadas. Las funcionalidades a las que podremos acceder a través de métodos GET son:

    - **Parámetros:** Accede al listado de todos los parámetros de la cámara, se accede a este listado a través de la llamada */GetParameters.*

    - **Brillo:** Para acceder a leer el parámetro del brillo a través de la API, procederemos a realizar la llamada a esta a través de la url */GetParameters/Brightness.*

    - **Gamma:** Para acceder a leer el parámetro del brillo a través de la API, accederemos a través de la dirección */GetParameters/Gamma*.

    - **Ganancia:** Para acceder a leer la ganancia, accederemos a través de la dirección */GetParameters/Gain.*

    - **Exposición:** Para recibir al tiempo de exposición o shutter que esté usando la cámara, accederemos a través de la dirección */GetParameters/Exposure*.

    - **Identificador de secuencia:** Las secuencias de imágenes que se están guardando o que ya han sido terminadas, pueden ser consultadas en cualquier momento. Para poder visualizar el estado de este tipo de tarea, accederemos a través de la llamada */Task/{id},* donde {id} es el identificador único de la secuencia de fotos que se quiere consultar.

    - **Foto:** La llamada a la API a través de la url */Photo/{nombre}* devuelve archivo .jpg con nombre {nombre} desde la base de datos.

    - **Inicio del streaming de video:** La llamada a la API */VideoStreaming*, iniciará el streaming de video de la salida de la cámara, que en realidad no resulta ser más que un archivo que se va refrescando cada intervalo de tiempo.

    - **Parada del streaming de video:** Para terminar con el streaming de video se hará la llamada */VideoStreamingOff*.

- **POST.** Los métodos POST son utilizados para la adición de recursos a la base de datos. Para este proyecto las llamadas son utilizadas para hacer dos tipos de peticiones concretas: tomar una imagen o realizar una secuencia de imágenes concreta.

    - **Guardar captura**: Para tomar una imagen en singular, la petición a la API será a través de la siguiente llamada: */TakePicture*. Esta petición recibe un JSON con el nombre que se desea asignársele a la captura.

    - **Guardar una secuencia de imágenes:** Para tomar una secuencia de imágenes, hay que realizar una petición a la API a esta dirección: */CreateTask*. Esta petición recibe un JSON con el número de fotos que se desea asignársele, un array de tags para cuando se desee visualizar fotos mediante un criterio de búsqueda, y el autor que está tomando la secuencia de fotos además de un identificador único.

- **PUT.** Los métodos PUT en este proyecto son para poder modificar los parámetros característicos de la cámara para ajustar mejor la visualización del streaming. Al igual que los métodos POST, reciben un JSON con el valor que se desea asignarles. Para asignarle un parámetro, basta con poner la palabra SetParameters y el parámetro que se desea modificar. Estas son las opciones que da la API para la modificación de los parámetros:

    - **Brillo:** Para poder modificar el brillo, la petición que se debe de hacer a la API es la siguiente: */SetParameters/Brightness*.

    - **Gamma:** Para poder modificar el gamma, la petición que se debe hacer a la API es la siguiente: */SetParameters/Gamma*.

    - **Ganancia:** Para poder modificar la ganancia, la petición para la API es la siguiente:*/SetParameters/Gain*.

    - **Exposición:** Para poder modificar el tiempo de exposición, la petición para la API es la siguiente: */SetParameters/Exposure*.

- **DELETE**. En este proyecto solo hay un método DELETE, que servirá para borrar las fotos que se deseen.

    - **Borrar foto**. El borrado de una foto de la base de datos se realizá a través de la llamada a la API: */DeletePhoto/{name},* donde name es el nombre de la foto que se desea borrar. Lógicamente, dicho nombre ha de residir en la base de datos.

3.SITUACIÓN ACTUAL DEL OBSERVATORIO FRANCISCO MANUEL SANCHEZ

El observatorio Francisco Manuel Sánchez es inaugurado en el año 2009, por un equipo de investigadores de la Facultad de Informática de la Universidad Politécnica de Madrid, dirigido por el profesor que le da nombre. Situado en la Facultad de Informática de la Universidad Politécnica de Madrid e integrado en la red ASTROCAM de la Comunidad de Madrid [8].

Este es el primer observatorio astronómico del mundo accesible vía web de forma gratuita, en un primer momento, el observatorio era controlado de forma remota a través mediante el software denominado Cíclope Astro. Este software daba la oportunidad a los usuarios de controlar de manera remota las cámaras y la cúpula, además de proporcionar herramientas para el desarrollo de experimentos astronómicos, lo que permitía, por primera vez en la historia, el desarrollo de actividades relacionadas con la astronomía desde cualquier lugar del mundo. En el año 2017, se desarrolló una API Restfull para el control remoto de los distintos elementos del observatorio, además de un nuevo diseño para la página web del observatorio, siendo estos dos proyectos realizados en dos Trabajos de Fin de Grado diferentes por alumnos de la ETSIINF y, en el año 2018, se desarrolló una nueva API para la visualización en directo de manera online de las imágenes tomadas desde la cámara conectada al telescopio del observatorio y que, a la vez, permitía el guardado de imágenes que el usuario ve en directo.

El objetivo principal del observatorio es poder controlar hasta el más mínimo detalle de distintos tipos de proyectos astronómicos, todo de manera automatizada y accesible a través de internet.

3.1 Proyectos en desarrollo en la actualidad

3.1.1 Proyecto Cazadores de Asteroides

Este proyecto se inició en el año 2016 y continúa en la actualidad. Este está siendo desarrollado de manera conjunta por investigadores de la Universidad Politécnica de Madrid (UPM) y el Instituto de Astrofísica de Canarias (IAC).

El proyecto consiste en el desarrollo de una aplicación (app) para smartphones de tipo Android o IOs, el cual permitirá desarrollar una serie de experimentos de tipo científico para la búsqueda y detección de asteroides cercanos a la Tierra para poder, así, contrastar de manera científica los datos recogidos enviando estos datos al “Minor Planet Center” de la International Astronomy Union (IAU).

La app de tipo gratuita fue nombrada bajo el mismo nombre que el proyecto. Está diseñada bajo una interfaz gráfica parecida a la de un videojuego, de modo que sea más atractiva al público. El usuario ve en su móvil una secuencia de imágenes del cielo que proporciona la red de telescopios "Gloria" y este podrá marcar aquellos objetos que crea que puedan ser asteroides. Una vez marcadas las zonas sospechosas, se pasa un primer filtro, que realizan el resto de usuarios a través de un sistema de votación y, una vez votadas, pasan a ser verificadas por un equipo de astrónomos profesionales, los cuales tienen la última palabra a la hora de decidir si el objeto detectado es un asteroide o no.

La aplicación tiene como objetivo lograr la colaboración ciudadana para poder llevar el control de asteroides potencialmente peligrosos para la Tierra. La primera versión estable fue lanzada a principios de 2017, aunque ya en Noviembre de 2016 existía un prototipo listo para su lanzamiento.

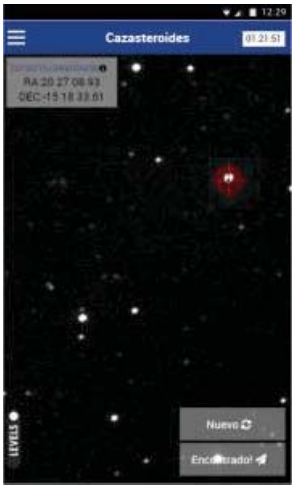

Figura 3.1: Interfaz Web de la aplicación Cazasteroides.

3.1.2 Proyecto Megara

El proyecto Megara, está desarrollado en colaboración por investigadores del IAC, investigadores de la Universidad Complutense de Madrid (UCM) e investigadores de la UPM, junto con otras universidades españolas y mexicanas.

MEGARA (Multi-Espectrógrafo en GTC de Alta Resolución para Astronomía), es un instrumento óptico instalado a finales de 2016 en el Gran Telescopio CANARIAS (GTC) [9]. Megara fue el primer instrumento capaz de analizar por primera vez la medusa cósmica, es decir, la luz proveniente de filamentos de gas que conectan las galaxias y que brillaron con intensidad únicamente durante las primeras épocas del Universo.

Es capaz de operar en espectroscopia 3D con alta resolución, capaz de obtener espectros ópticos de un gran número de objetos que se encuentren en un mismo campo de visión.

El objetivo de Megara es el de la cartografía de las propiedades y movimientos de las estrellas de otras galaxias cercanas a la nuestra y poder analizar con gran detalle nebulosas de gas de la Vía Láctea.

3.2 Proyectos anteriores

3.2.1 Proyecto ASTROMADRID

Este proyecto, llevado a cabo entre los años 2010 a 2013 tuvo como objetivo la coordinación de las actividades para el desarrollo de instrumentación astronómica por los diferentes grupos de la Comunidad de Madrid. El objetivo principal consistió en la constitución de un grupo multidisciplinar que se beneficiara de las sinergias que se originen entre cada grupo, de tal forma que se optimice el desarrollo de instrumentos astronómicos para beneficio de la comunidad nacional e incluso internacional [10].

Las líneas principales del proyecto son:

- Desarrollo de un nuevo instrumento para el Gran Telescopio de Canarias.
- Desarrollo de instrumentación espacial.
- Instrumentos astronómicos terrestres.
- Desarrollo del Observatorio Virtual Español y de herramienta de minería de datos.
- Formación de nuevos científicos e ingenieros.

3.2.2 Proyecto COLDEX

Desde el año 2002 y hasta el 2005, el proyecto COLDEX se desarrolló con el objetivo del acercamiento de las Tecnologías de la Información, así como del desarrollo de herramientas computacionales para el fomento de la experimentación científica, todo dentro de un marco colaborativo y distribuido en una comunidad intercultural de principiantes [11].

3.3 Equipamiento del Observatorio Francisco Sánchez

El Observatorio está dotado con el siguiente equipamiento [12]:

• Observatorio de 3,5 metros de diámetro.

• Telescopio 10" Meade LX200GPS.

• Cámara CCD SBIG Modelo ST-237A + Rueda de filtros CFW-5C.

• Cámara CCD DMK 41AU02 monocromática.

• Cuatro webcams Philips ToUcam Pro I y II, una de ellas acoplada al telescopio y otra al buscador. Ambas han sido modificadas para realizar fotografías de larga exposición. Las otras dos dan una vista parcial del observatorio.

• Estación Meteorológica Vantage Pro 2 Plus con Fan-Aspirated Radiation Shield, que ofrece datos sobre el estado del tiempo en tiempo real, que son imprescindibles para el adecuado uso del observatorio y cuyos datos se publican de forma gratuita.

• Diversos equipos que sirven tanto como servidor de las aplicaciones web, como de conexión y difusión de las imágenes y vídeos que captan las webcams dispuestas por la cúpula. Algunos corren con sistemas GNU/Linux, mientras que los encargados de correr las APIs son dos Raspberrys PI.

4. DESARROLLO DEL MÓDULO WEB

4.1 Introducción

La finalidad de este nuevo módulo para la página web del Grupo Cíclope es la de poner a disposición de los usuarios todas las herramientas de las que dispone el Observatorio Francisco Manuel Sánchez de una manera intuitiva y atractiva al usuario, de manera que tenga ganas de volver a hacer uso de estas y disfrute de ellas cada vez que vuelva a conectarse a la web.

El sistema de vídeo en directo dará la oportunidad a los usuarios de hacer uso de herramientas, como es el telescopio del observatorio, normalmente muy costosas y no al alcance de cualquiera, de manera gratuita. Podrán decidir los distintos atributos que tiene la cámara que está conectada al telescopio, como el tiempo de exposición, el brillo, el balance de blancos o el gamma, para así, a través del video, ser capaces de guardar secuencias de fotos de lo que están viendo en directo las cuales podremos visitar en nuestra galería personal, y organizarlas a través de un etiquetado por tags y poder sentirse protagonistas de los experimentos que lleven a cabo en la web.

La galería dará la oportunidad a los usuarios de ver los resultados de los experimentos que hayan llevado a cabo con anterioridad. Además, gracias al sistema de etiquetado por tags, podrán ver las secuencias tomadas por otros usuarios y comparar los resultados de sus experimentos con los del resto de usuarios de la web.

Todas las funcionalidades de este nuevo módulo para la web son posibles gracias a una API desarrollada en el anterior semestre y que continúa ampliándose actualmente, además de la API para el Gateway creada en otro proyecto anterior del Observatorio.

4.2 Definición de conceptos

En este capítulo se harán referencias a diferentes conceptos que se van a definir a continuación:

- **HTML:** Lenguaje de marcas de hipertexto (HyperText Markup Language), es el elemento más básico para la construcción y visualización de una página web, convirtiéndose así en el lenguaje estándar del World Wide Web. Determina el contenido de la página web, pero no su apariencia y funcionalidad, de esto se encargan otras tecnologías como CSS y JavaScript, respectivamente.

- **CSS:** Hojas de estilo en cascada (Cascading Style Sheet), es el lenguaje utilizado para la presentación de documentos HTML. Se utiliza para dar estilo a documentos HTML. Este describe cómo se va a mostrar un documento HTML en la pantalla, en papel o cómo se va a visualizar a través de un dispositivo de lectura.

- **JavaScript:** Es un lenguaje de programación interpretado. Se define como orientado a objetos, basado en prototipos, imperativo y dinámico.

  Se utiliza principalmente en su forma del lado del cliente (client-side), implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas, aunque existe una forma de JavaScript del lado del servidor (Server-side JavaScript). Su uso en aplicaciones externas a la web, por ejemplo en documentos PDF, aplicaciones de escritorio (mayoritariamente widgets) es también significativo [13].

- **DOM:** Modelo de Objetos del Documento o Modelo en Objetos para la representación de Documentos (Document Object Model) [14]. Es una interfaz de plataforma que promociona un conjunto estándar de objetos para representar documentos HTML, XHTML y XML. El DOM permite el acceso dinámico a través de programación, lenguajes como JavaScript, para acceder y modificar el contenido, estructura y estilo de los documentos HTML, XML.

- **HTTP:** Del inglés HyperText Transfer Protocol (Protocolo de Transferencia de Hipertexto), es el protocolo utilizado para el intercambio de información en la web. Define la semántica y sintaxis que utilizan distintos softwares web, tanto clientes, como servidores y proxy para interactuar entre sí. Está basado en el modelo cliente/servidor y es un protocolo sin estado, es decir, no guarda información alguna, por lo que se suelen utilizar las llamadas "cookies" para almacenar información de sesión. Una variante segura es el protocolo HTTPS (HTTPSSecure), cifrada por SSL/TLS, que permite la transferencia segura de información sensible entre el cliente y el servidor. Para que un servidor acepte conexiones HTTPS, el servidor debe de disponer de un certificado firmado por una autoridad de certificación para que el navegador lo acepte.

- **AJAX:** JavaScript asíncrono y XML (Asynchronous JavaScript And XML), es un término que describe las posibilidades que tiene el navegador de intercambiar información con el servidor en segundo plano, evitando así las recargas constantes de la página, lo que permite mejorar la interacción del usuario con la aplicación, así como, tiene la posibilidad de analizar y trabajar con documentos XML y JSON.

- **JSON:** Notación de objetos de JavaScript (JavaScript Object Notation), es un formato ligero de intercambio de datos que está basado en un subconjunto de la notación literal de objetos de JavaScript. Su uso es muy fácil en JavaScript, así como en otra multitud de lenguajes de programación. El uso y la presencia de JavaScript en la web ha sido fundamental para que JSON haya sido aceptado por la comunidad de desarrolladores AJAX.JSON está constituido por una colección de pares de nombre/valor y/o una lista ordenada de valores.

- **API:** Interfaz de programación de aplicaciones (Application Programming Interface), es una especificación formal de un conjunto de funciones y procedimientos que ofrece una biblioteca para ser utilizado por otro software proporcionando una capa de abstracción, ya que las funciones no hay que programarlas desde cero.

- **REST:** Transferencia de estado representacional (Representational StateTransfer), es un estilo de arquitectura software que se apoya totalmente en el estándar HTTP. Se utiliza para describir cualquier interfaz entre sistemas que utilicen HTTP para obtener datos o generar operaciones entre esos datos en cualquier formato como XML y JSON. Está basado en el modelo cliente/servidor sin estado, cada petición HTTP contiene toda la información necesaria para ejecutarla. Las operaciones más importantes que se aplican a los recursos de información son GET (leer y consultar), POST (crear), PUT (modificar) y DELETE (eliminar). Los recursos se manipulan a través de la URI o identificador de recursos uniforme.

4.3 Tecnologías utilizadas

Para el desarrollo de los nuevos módulos de la página web del observatorio, se ha utilizado como principal framework AngularJS, Bootstrap para dar funcionalidad a la aplicación y poder hacerla web-responsive, los lenguajes de programación JavaScript, CSS y HTML para el formato de la web y, para la conexión con las distintas APIs, peticiones HTTP a través de objetos JSON.

A continuación, se desarrollará en detalle cada una de las tecnologías utilizadas para el desarrollo de estos módulos y se justificará el uso de estos.

4.3.1 AngularJS

Angular JS es un framework de JavaScript de código abierto desarrollado por Google. Lanzado en octubre del año 2010, es utilizado para crear aplicaciones web de tipo Single Page Application (SPA) o, en español, aplicación de página única.

Una SPA (Single Page Application) es una aplicación web que se carga como si fuera un programa de escritorio, es un sitio web que cabe en una sola página con el propósito de dar una experiencia más fluida a los usuarios ya que todos los elementos de la página se cargan al inicio y puedes navegar por las diferentes vistas de la aplicación sin tener que recargar la página. Esta es una gran ventaja ya que no necesitas esperar recargas de página, que consuman datos y velocidad de internet en el transcurso de éstas. En un SPA todos los códigos de HTML, JavaScript, y CSS se cargan de una vez o los recursos necesarios se cargan dinámicamente como lo requiera la página y se van agregando, normalmente como respuesta de las acciones del usuario. Además de esto, la comunicación entre cliente y servidor se realiza de forma totalmente transparente al usuario, por lo que se logra que éste tenga la sensación de no abandonar nunca la página principal de la aplicación, logrando una mayor fluidez en experiencia de usuario.

AngularJS está construido en torno a la creencia de que la programación declarativa es la que debe utilizarse para generar interfaces de usuario y enlazar componentes de software. Este framework, adapta y amplía el HTML tradicional para servir mejor contenido dinámico a través de un data binding bidireccional que permite la sincronización automática de modelos y vistas [15].

Objetivos de diseño:

- Disociar la manipulación del DOM de la lógica de la aplicación. Esto mejora la capacidad de prueba del código.
- Considerar a las pruebas de la aplicación como iguales en importancia a la escritura de la aplicación. La dificultad de las pruebas se ve reducida drásticamente por la forma en que el código está estructurado.
- Disociar el lado del cliente de una aplicación del lado del servidor. Esto permite que el trabajo de desarrollo avance en paralelo, y permite la reutilización de ambos lados.
- Guiar a los desarrolladores a través de todo el proceso del desarrollo de una aplicación: desde el diseño de la interfaz de usuario, a través de la escritura de la lógica del negocio, hasta las pruebas.

Angular sigue el patrón MVC (Modelo Vista Controlador) de ingeniería de software y alienta la articulación flexible entre la presentación, datos y componentes lógicos. Con el uso de la inyección de dependencias, Angular lleva servicios tradicionales del lado del servidor, tales como controladores dependientes de la vista, a las aplicaciones web del lado del cliente. En consecuencia, gran parte de la carga en el backend se reduce, lo que conlleva a aplicaciones web mucho más ligeras.

La elección de AngularJS como framework para el desarrollo de los nuevos módulos de la web, se debe a que este fue el framework usado para el desarrollo en 2017 de toda la parte frontal de la web, que está en uso actualmente y gracias a las ventajas descritas en los anteriores párrafos.

4.3.2 GitHub

GitHub es una plataforma de desarrollo colaborativo de software para alojar proyectos utilizando el sistema de control de versiones Git [16]. El software, que opera desde enero de 2010, fue escrito en Ruby on Rails y es utilizado principalmente para la creación de código fuente de programas informáticos y su almacenamiento en la nube.

GitHub aloja tu repositorio de código y te brinda herramientas muy útiles para el trabajo en equipo, dentro de un proyecto. Además de eso, puedes contribuir a mejorar el software de los demás. Para poder alcanzar esta meta, GitHub provee de funcionalidades para hacer un fork y solicitar pulls.

Realizar un fork es simplemente clonar un repositorio ajeno (generar una copia en tu cuenta), para eliminar algún bug o modificar cosas de él. Una vez realizadas tus modificaciones, puedes enviar un pull al dueño del proyecto. Éste podrá analizar los cambios que has realizado fácilmente, y si considera interesante tu contribución, adjuntarlo con el repositorio original.

Entre las principales características y herramientas que proporciona este gestor de versiones caben destacar:

- Una wiki para el mantenimiento de las distintas versiones de las páginas.
- Un sistema de seguimiento de problemas que permiten a los miembros de tu equipo detallar un problema con tu software o una sugerencia que deseen hacer.
- Funcionalidades como si se tratase de una red social, por ejemplo, seguidores.
- Una página web por cada proyecto.
- Una herramienta de revisión de código, donde se pueden añadir anotaciones en cualquier punto de un fichero y debatir sobre determinados cambios realizados en un commit específico.
- Un visor de ramas donde se pueden comparar los progresos realizados en las distintas ramas de nuestro repositorio.

Dado que el código correspondiente a la web actual y que se debía ampliar para la creación de la galería y la pestaña de vídeo en directo estaba ya almacenado en el repositorio de GitHub del Grupo Cíclope, se creó una nueva rama de tipo feature, en el que se desarrollaron todas las nuevas funcionalidades de la página web, de tal manera que la evolución y desarrollo de la misma pudiera ser controlado por todos los miembros de este departamento.

4.3.3 GitKraken

GitKraken es una potente y elegante interfaz gráfica multiplataforma para git desarrollada con Electron. La cual nos permite, de forma muy sencilla, llevar el completo seguimiento de nuestros repositorios, ver y crear nuevas ramas, crear nuevos repositorios, todo el historial de nuestro trabajo, como los commits realizados por el ti y el resto de usuarios con los que compartas el espacio de trabajo.

Aunque los commits, push y pulls los realizáramos a través de la herramienta que nos brindaba Visual Studio Code, GitKraken nos ofrecía una manera muy visual e intuitiva de todo el historial de cambios ocurridos en el repositorio del Observatorio, por lo que la elección fue clara a la hora de elegir esta herramienta. En la imagen 4.1 podemos ver la parte correspondiente al desarrollo de la web del Observatorio en este interfaz.

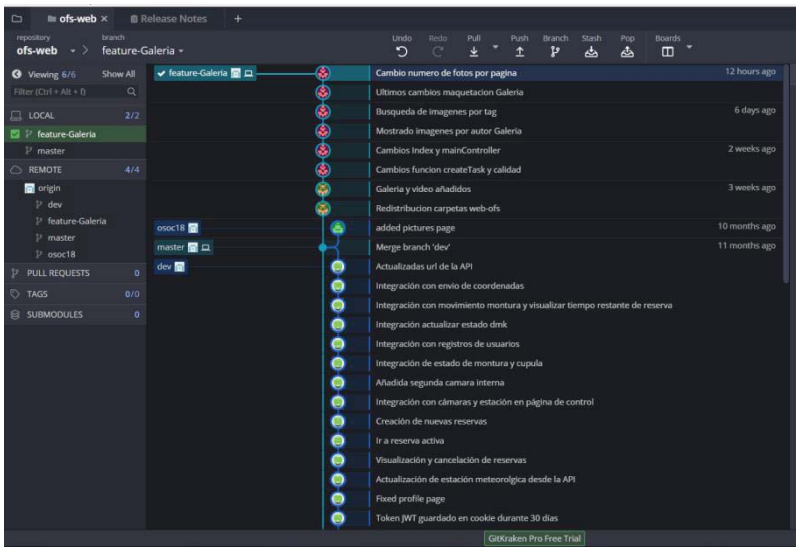

Figura 4.1: Repositorio de la web del OFS a través de la herramienta GitKraken.

4.3.4 Visual Studio Code

Visual Studio Code es un editor de código fuente desarrollado por Microsoft, el cual incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código.

Fue creado y diseñado para que funcione en los tres sistemas operativos mayormente utilizados: Windows, Linux y Mac OS. El 18 de noviembre de 2015, Visual Studio Code fue lanzado bajo la licencia MIT y su código fuente fue publicado en GitHub. También fue anunciada una nueva capacidad para agregar extensiones [17].

El 14 de abril de 2016, Visual Studio Code graduó la etapa de vista previa pública y se lanzó a la web.

Este IDE (Integrated Drive Electronics) viene con soporte incorporado para JavaScript, TypeScript y Node.js y tiene un rico ecosistema de extensiones para otros lenguajes, tales como C++, Java, Python, Php o Go, y tiempos de ejecución como .NET o Unity. Estas extensiones hacen de Visual Studio Code una elección muy recomendable a la hora de elegir un entorno de desarrollo con el que desarrollar programas informáticos de cualquier tipo.

Debido a su control integrado de Git, el usuario es capaz de realizar acciones conectadas directamente a su repositorio de GitHub de forma gráfica y descriptiva sin necesidad de utilizar la terminal de comandos de su sistema operativo.

En un principio, se usó el entorno de desarrollo WebStorm para la codificación de este proyecto, pero, una vez se indagó en profundidad sobre el resto de entornos disponibles en la web, se decidió usar el Visual Studio Code por las siguientes razones:

- Control integrado de Git, el cual facilitaba de gran manera el trabajo y subida de las versiones que se iban realizando al repositorio del Grupo Cíclope.
- Sistema de resaltado de sintaxis y finalización inteligente de código, el cual ahorraba en gran medida la carga de trabajo a la hora de codificar las nuevas funcionalidades.
- Sistema de extensiones, el cual permitió añadir ciertas extensiones que ayudaron al fácil desarrollo de la aplicación.

4.3.5 Servidor HTTP Apache

El servidor HTTP Apache es un servidor web HTTP de código abierto, para plataformas Unix, Microsoft Windows, Macintosh y otras, que implementa el protocolo HTTP/1.1 y la noción de sitio virtual según la normativa RFC 2616. Su nombre se debe a que alguien quería que tuviese la connotación de algo que es firme y enérgico pero no agresivo, y la tribu Apache fue la última en rendirse al que pronto se convertiría en el gobierno de Estados Unidos, y en esos momentos la preocupación de su grupo era que llegasen las empresas y "civilizasen" el paisaje que habían creado los primeros ingenieros de internet. En un principio, Apache consistía solamente en un conjunto de parches a aplicar al servidor de NCSA. En inglés, a patchy server (un servidor "parcheado") suena igual que Apache Server [18].

El servidor Apache es desarrollado y mantenido por una comunidad de usuarios bajo la supervisión de la Apache Software Foundation dentro del proyecto HTTP Server (httpd) y presenta, entre otras características altamente configurables, bases de datos de autenticación y negociado de contenido.

Apache es el servidor HTTP más usado y jugó un papel fundamental en el desarrollo de la World Wide Web y alcanzando su máxima cuota de mercado en 2005, siendo el servidor empleado en el 70% de los sitios web en el mundo. Sin embargo, ha sufrido un descenso en su cuota de mercado en los últimos años (estadísticas históricas y de uso diario proporcionadas por Netcraft2). En 2009, se convirtió en el primer servidor web que alojó más de 100 millones de sitios web [19].

Entre las ventajas más destacables de los servidores Apache, se encuentran:

- Modular. El servidor consta de una sección core y diversos módulos que aportan mucha de la funcionalidad que podría considerarse básica para un servidor web.

- Código abierto. El servidor apache es desarrollado y mantenido por una comunidad de usuarios bajo la supervisión de la Apache Software Foundation.

- Multi-plataforma. Puede ser utilizado en cualquier tipo de plataforma.

- Extensible.

Apache es usado principalmente para enviar páginas web estáticas y dinámicas en la World Wide Web, además, es usado para muchas otras tareas donde el contenido necesita ser puesto a disposición en una forma segura y confiable. Un ejemplo es al momento de compartir archivos desde una computadora personal hacia Internet [18].

El servidor Apache se utilizó para crear una versión en local de la parte frontal de la página web con el fin de previsualizar y probar código mientras éste era desarrollado, de tal manera que la página web en funcionamiento no se viese afectada por el desarrollo de los nuevos módulos.

4.3.6 BootStrap

Bootstrap es un framework originalmente creado por Twitter como una solución interna, que permite crear interfaces web con CSS y JavaScript, además de ser compatible con preprocesadores como Less y Saas, su principal particularidad es la de adaptar la interfaz del sitio web al tamaño del dispositivo en que se visualice. Es decir, el sitio web se adapta automáticamente al tamaño de un PC, una Tablet o cualquier otro dispositivo, ahorrándonos así el trabajo de tener que rediseñar un sitio web.

Esta técnica de diseño y desarrollo se conoce como “responsive design” o diseño adaptativo. El beneficio de usar responsive design en un sitio web es, principalmente, que el sitio web se adapta automáticamente al dispositivo desde donde se acceda [20].

Dado que uno de los principales objetivos de este proyecto era el de crear los nuevos módulos de manera que fueran del tipo responsive, la elección fue clara a la hora de hacer uso de bootstrap. Además, en la versión anterior de la web, el desarrollador ya había hecho uso de este framework para la creación del resto de pantallas, por lo que cambiar de framework solo complicaría el diseño de las nuevas pantallas para la página web.

4.3.7 SQLite

SQLite es una librería escrita en lenguaje C, que implementa una pequeña base de datos de tipo SQL, auto contenida y de alta fidelidad [21].

Este sistema de gestión de bases de datos de tipo relacional orientadas a objetos es de tipo código abierto y, a diferencia de los sistemas de gestión de bases de datos cliente-servidor, el motor de SQLite no es un proceso independiente con el que el programa principal se comunica. En lugar de eso, la biblioteca SQLite se enlaza con el programa pasando a ser parte integral del mismo utilizando la funcionalidad de SQLite a través de llamadas simples a subrutinas y funciones. Esto reduce la latencia en el acceso a la base de datos, debido a que las llamadas a funciones son más eficientes que la comunicación entre procesos. El conjunto de la base de datos (definiciones, tablas, índices, y los propios datos), son guardados como un solo fichero estándar en la máquina host. Este diseño simple se logra bloqueando todo el fichero de base de datos al principio de cada transacción [22].

SQLite es utilizado en una gran variedad de aplicaciones, destacando las siguientes:

- Adobe Photoshop Elements utiliza SQLite como motor de base de datos en su última versión del producto (la 6.0) en sustitución del Microsoft Access, utilizado en las versiones anteriores.

- Mozilla Firefox usa SQLite para almacenar, entre otros, las cookies, los favoritos, el historial y las direcciones de red válidas.

- Varias aplicaciones de Apple utilizan SQLite, incluyendo Apple Mail y el gestor de RSS que se distribuye con Mac OS X. El software Aperture de Apple guarda la información de las imágenes en una base de datos SQLite, utilizando la API Core Data.

- El navegador web Opera usa SQLite para la gestión de bases de datos WebSQL.

- Skype es otra aplicación de gran despliegue que utiliza SQLite.

SQLite se utilizó para crear las tablas de la base de datos que almacenan la información de las imágenes guardadas por los usuarios. Su uso en el proyecto se debe a que los desarrolladores de las APIs sobre las que se apoya el frontal de la web eligieron este sistema de gestión de bases de datos.

4.4 Desarrollo del proyecto

En un principio y tras una reunión con el departamento del Observatorio, el proyecto se focalizó en el desarrollo de una galería de imágenes de tipo web-responsive que ampliara la experiencia del usuario dentro de la web del observatorio, dando la oportunidad a los usuarios de compartir imágenes tomadas por ellos con el resto de usuarios de la plataforma.

Esta galería tendría un carácter de acceso global, de tal manera que en todo momento cualquier usuario viera, no solo las imágenes subidas por ellos mismos, sino también las del resto de usuarios de la página. Las imágenes se almacenarían en una base de datos ajena al observatorio. Para ello se decidió el uso de la plataforma Google Drive, a través de los servicios de la API de Google.

Paralelamente a este proyecto se estaba desarrollando un API para el control de la cámara DMK, que realizaba la captura y emisión de imágenes a través de la cámara acoplada a un telescopio. Debido al desconocimiento de esta nuevo API, y dada la necesidad de un sistema para la subida y almacenamiento de las imágenes en una base de datos, se decide crear una nueva vista encargada de esta gestión.

Tras varios meses de investigación en las herramientas a utilizar y de trabajo en la funcionalidad definida, hubo una nueva reunión con el equipo de coordinación del Observatorio para poner en común los avances del proyecto. En esa reunión, el proyecto sufrió un giro inesperado, debido a que salió a la luz la existencia del nuevo API, lo que traía consigo que la vista del sistema de gestión de imágenes ya no tuviera sentido.

Este malentendido provocó asumir una serie de cambios en el desarrollo del proyecto, resumiéndose en los siguientes puntos:

- Desaparición del sistema de gestión de subida de ficheros, dado que a partir de ese momento las imágenes serían tomada a través de la cámara asociada al telescopio del Observatorio únicamente, haciendo uso de la API de control de la cámara DMK, recién sacada a producción.

- Implementación de una nueva vista para el streaming en directo, la cual permitiría a los usuarios controlar las opciones de la cámara para el streaming.

- Implementación de una nueva funcionalidad que permitiría a los usuarios guardar secuencias de imágenes en función de lo que están viendo en el streaming.

- Implementación de una nueva funcionalidad de guardado y búsqueda por tags a través de la API de la cámara DMK.

- Desaparición de una galería global para todos los usuarios, la galería sería algo exclusivo de cada usuario, solo siendo posible ver las fotos del resto de usuarios una vez realizada una búsqueda por tag.

Además de la profundidad de los cambios debido al uso del nuevo API, se ha añadido la dificultad de trabajar en paralelo a la vez que se finalizaba el desarrollo del API y carecíamos de algunas funcionalidades (sistema de control de errores o la búsqueda de imágenes por tags y autor). Además se no podía probar los avances en el desarrollo contra el API de manera local, teniendo la limitación de hacerlo tan solo una vez por semana.

Debido a la gran cantidad de cambios nuevos a introducir y a que a la API de la cámara le faltaban aún ciertas funcionalidades por implementar (un sistema de control de errores, búsqueda de imágenes por tags y autor,…), el desarrollo se ralentizó. También tuvo un fuerte impacto la imposibilidad de probar la API de manera local, que provocaba que solo se pudieran probar los avances una vez por semana.

Todo esto ha provocado una ralentización en la ejecución y pruebas del desarrollo que, finalmente, hizo que se desestimara dentro de los objetivos del proyecto la conexión con Google Drive a través de la API de Google.

Puede decirse, por tanto, que el proyecto se ha desarrollado en dos líneas diferentes:

1. Una fase inicial en la que los objetivos marcados eran crear una nueva vista en la web del observatorio con un módulo de subida y almacenamiento de imágenes propias de usuarios a un repositorio común con la posibilidad de acceso universal y construido bajo la plataforma Google Drive, y los servicios de almacenamiento del API de Google

2. Una segunda fase y definitiva, tras la reunión a mediados del proyecto, en la que el módulo va dirigido a permitir a los usuarios el control de los atributos de la cámara DMK (cuya API estaba en la fase final de su desarrollo), el almacenamiento de las imágenes mientras realizan streaming y su posterior acceso al repositorio.

La primera acción que se realizó en el proyecto, previo a la codificación de las nuevas vistas y sus correspondientes controladores fue la inclusión de las nuevas rutas a sus correspondientes vistas en el archivo de control de la aplicación “app.js”, tal como se muestra en la Figura 4.2, y la inclusión de dos nuevos elementos en la vista “index.html” para el acceso a estas vistas desde la barra de navegación.

```
60
61          .when('/galeria', {
62              templateUrl: 'views/galeria.html',
63              controller: 'galeriaController',
64          })
65          .when('/video', {
66              templateUrl: 'views/video.html',
67              controller: 'videoController'
68          })
69
```

Figura 4.2: Definición de las nuevas rutas en el “app.js”.

4.4.1 Sistema gestor de subida de ficheros

Correspondiente a la fase inicial de proyecto, se desarrolló una nueva vista para la página web que consistiría en un sistema encargado de la gestión de la subida de imágenes así como la creación y uso de una nueva base de datos destinada al almacenamiento y acceso a las mismas.

Dentro de esta vista se permitía a los usuarios de la web que hubieran iniciado sesión:

- Subir archivos de imagen propiedad del usuario a una carpeta de Google Drive. Esta carpeta pertenecería al usuario, de modo que no ocupe espacio físico en los discos del observatorio.

- Asignar tags a las imágenes subidas que faciliten la categorización y búsqueda posterior de las mismas.

Esta nueva vista, la cual podemos apreciar en las figuras 4.3 y 4.4, se dividió en los siguientes elementos:

- Un título de la pantalla centrado en la parte superior, siguiendo el estilo de las otras vistas ya existentes, con la frase “Subir imágenes”.

- Inmediatamente debajo del título de la vista se situó una barra de progreso, la cual estaría encargada de mostrar el porcentaje de subida de las imágenes que llevaría el usuario.

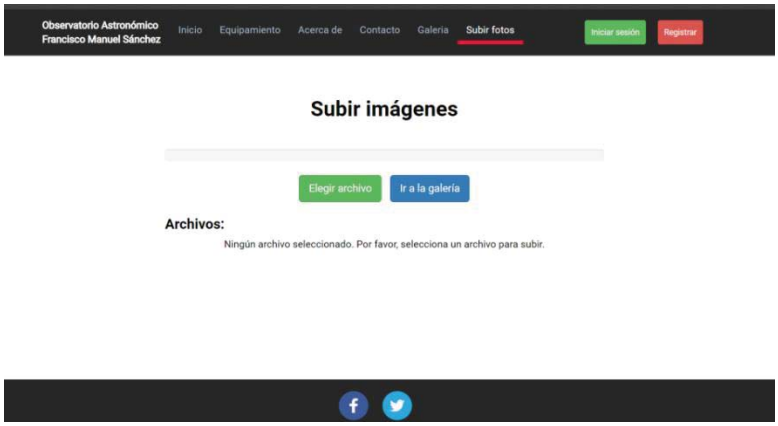

Figura 4.3: Vista simple del gestor de subida.

- Centrado justo a continuación de la barra de progreso empezando desde la izquierda estaría situado el botón “Elegir archivo”, el cual nos abriría una ventana modal de acceso a nuestros ficheros, a través de la cual podríamos elegir la imagen o imágenes que deseáramos subir a nuestra galería. Una vez pulsado este botón y elegido los archivos a subir, se desplegaría una parte oculta de la vista (Figura 4.4) en la que se muestran los siguientes elementos:

    - Dos nuevos botones, uno encargado de realizar la función de subida a la base de datos y el otro encargado de borrar archivos en caso de error en la elección.

- Una barra de escritura para los tags que se le quisieran asignar a esa imagen.

- Una previsualización de los detalles del archivo, con su nombre, peso y miniatura.

- A la izquierda del botón de despliegue del gestor de archivos se encontraría un botón de acceso directo a la galería para poder ver los resultados de nuestra subida, con el nombre “Ir a la galería”.

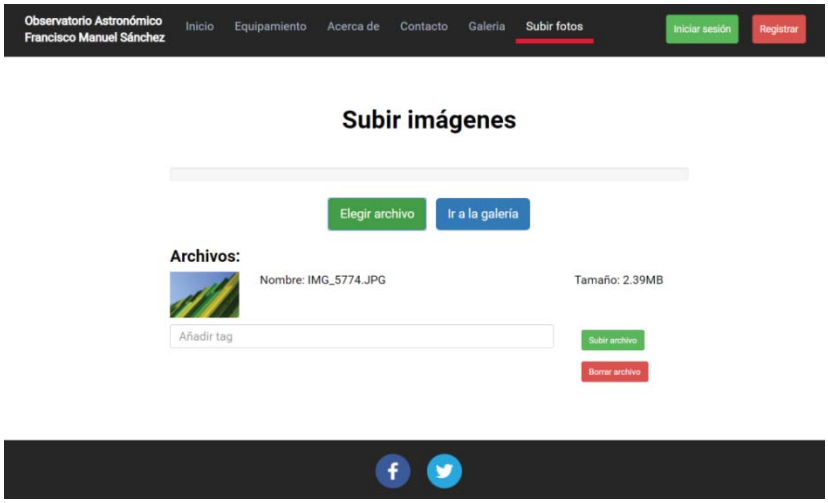

Figura 4.4: Vista del gestor una vez seleccionado un fichero a subir.

4.4.2 Vídeo en directo

Correspondiente a la segunda fase de desarrollo del proyecto, esta nueva vista se encarga de la emisión de las imágenes que se reciben de la cámara DMK acoplada al telescopio a través de un reproductor y la modificación de estas gracias a las distintas opciones de modificación de los parámetros de la cámara.

A través de una ventana modal, los usuarios podrán guardar secuencias de imágenes, hasta un número máximo de 1000, las cuales podrán etiquetar haciendo uso de palabras claves o “tags”, hasta un máximo de cinco tags por secuencia.

Para poder realizar todo estas funcionalidades, se hizo uso de la API para la cámara DMK y de la API para la gestión de reservas y usuarios.

De la API para el control de la cámara DMK se utilizaron las siguientes llamadas:

- /GetParameters/{Parámetro}. Esta llamada de tipo GET obtendrá el valor de los parámetros de la cámara y se los mostrará al usuario por pantalla. Un ejemplo de uso de esta llamada se encuentra en la figura 4.5.

```
246
247     $scope.getSaturation = function()
248     {
249             $http({
250             method: 'GET',
251             url: $scope.url + '/GetParameters/Saturation',
252         }).then(function successCallback(response){
253             console.log(response.data);
254             $scope.saturationSlider = response.data.Value.CurrentValue;
255             $scope.saturationNumber = response.data.Value.CurrentValue;
256             $scope.saturationMin = response.data.Value.MinValue;
257             $scope.saturationMax = response.data.Value.MaxValue;
258             $scope.saturationAuto = response.data.Value.DefaultValue;
259
260             $scope.unableSaturation = "";
261         }, function errorCallback(response){
262             console.log(response.statusText);
263         });
264     }
265
266
```

Figura 4.5: Ejemplo de uso de llamada /GetParameters/ en una función para la obtención del parámetro "Saturación".

- /VideoStreaming. Esta llamada de tipo GET se usa para la inicialización de la recepción de imágenes en directo a través de la cámara.

- /VideoStreamingOff. Esta llamada de tipo GET es utilizada para la finalización de la recepción de imágenes en directo a través de la cámara.

- /SetParameters/{Parámetro}. Estas llamadas de tipo PUT son usadas para la asignación de valores a la cámara a través de un elemento de tipo slider y otro de tipo numérico. Podemos ver un ejemplo de uso de esta llamada en la figura 4.6.

```
623
624     $scope.putWhitebalanceRed = function(position)
625     {
626         $scope.whitebalanceredNumber = position;
627
628         var dat = {Value: position};
629         var json = JSON.stringify(dat);
630
631         $http({
632             method: 'PUT',
633             url: $scope.url + '/SetParameters/WhiteBalanceRed',
634             data: json,
635         }).then(function successCallback(response){
636             console.log(response.data);
637             $scope.whitebalancered = response.data.Message;
638         }, function errorCallback(response){
639             console.log(response.statusText);
640         });
641     }
642
```

Figura 4.6: Ejemplo de uso de llamada /SetParameters/ en una función para la obtención de un atributo.

- /CreateTask. Esta llamada de tipo POST será la encargada de hacer la llamada a la API encargada del guardado de las secuencias de imágenes. Podemos ver el uso de esta llamada en la segunda parte de la función SetElements en la figura 4.7.

De la API del Gateway se utiliza únicamente la siguiente llamada:

- /users/loged. Esta llamada es usada para la obtención del nombre del usuario que esté realizando el guardado de la secuencia de imágenes, de tal manera que el nombre de autor que se cree en la base de datos de las imágenes sea el mismo que el del usuario conectado realizando el guardado de esa secuencia. De esta manera quedan conectadas las dos bases de datos. Podemos ver el uso de esta llamada en la primera parte de la función SetElements en la figura 4.7.

```
938
939     $scope.setElements = function(){
940         var urlGetInfoUser = "https://ofs.fi.upm.es/api/users/loged";
941         $http.get(urlGetInfoUser).then(function(userInfo){
942             var dat = {
943                 Amount: $scope.amountNum,
944                 Tags: $scope.listTags,
945                 Author: userInfo.data.username
946             };
947 
948             $http.post($scope.url + '/CreateTask', dat)
949             .then(function(response){
950                 console.log(response.data);
951                 $scope.ID = response.data.ID;
952                 $scope.text = "Success! " + response.data.Message + ". Your ID is " + $scope.ID;
953                 console.log($scope.text);
954                 $scope.cerrarModal('modalVideo');
955                 $scope.abrirModal('modalExito');
956             }).catch(function(err){
957                 console.log(err);
958                 $scope.cerrarModal('modalVideo');
959                 $scope.abrirModal('modalError');
960             });
961         })
962 
963     }
964
```

Figura 4.7: Función setElements() para el guardado de una secuencia de imágenes.

Para el diseño de la vista, se decidió dividir esta en dos partes. La parte desde el margen izquierdo hasta la mitad correspondiendo a la parte del reproductor del video en directo y el botonaje para realizar las acciones de inicio/parado del directo, así como de las acciones de guardado de secuencias de imágenes. La segunda mitad de la vista es usada para el listado de los atributos de la cámara y sus opciones de edición a través de un slider y una entrada de tipo numérica. Los atributos se verán en gris si no están disponibles, o en negro si están listos para su modificación (Figura 4.8).

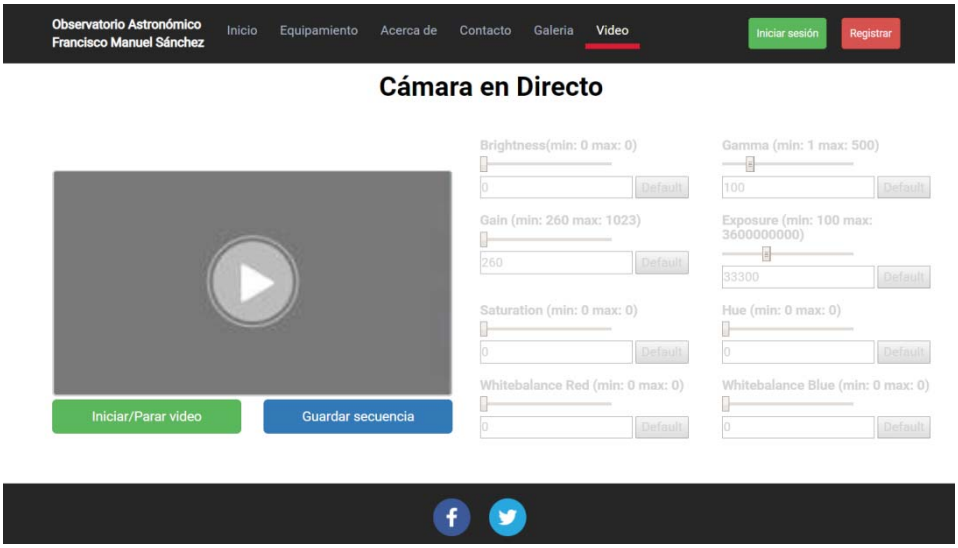

Figura 4.8: Vista del video en directo en una pantalla de 1920x1080

Para el guardado de las secuencias de imágenes, en vez de crear una nueva vista a la que se nos redireccionara una vez pulsado el botón “Guardar secuencia”, se optó por la creación de una ventana modal, en la que se introducirían las opciones de guardado y realizaría todo el flujo correspondiente a esta acción. Se desarrollaron tres vistas distintas de esta modal, una para la introducción de las opciones de guardado (Figura 4.8), otra modal en caso de error y otra en caso de éxito, la cual nos muestra el número de identificador de la secuencia creada (Figura 4.10).

Figura 4.9: Ventana modal para el guardado de secuencias.

Figura 4.10: Ventana modal de éxito

Además de la creación de la vista y el controlador asociada al módulo del streaming, también se modificó el controlador de la clase principal, llamado mainController, en el cual se añadió una función para la ocultación o muestra en la barra de navegación de la vista “Video”.

Esta nueva función, correspondiente a la mostrada en la figura 4.11, tiene como objetivo el control de acceso de usuarios al vídeo en directo, de tal manera que únicamente podrá acceder un usuario a esta pestaña, y solo si ha realizado la reserva de un experimento, ya sea solar o nocturno. Esto se realiza a través la llamada a la API del Gateway */reservations/own*, esta nos devuelve las reservas de las que es propietario el usuario conectado en formato de fecha, una vez recibida la respuesta de la API, comprobamos si la fecha actual reside en ese intervalo, si lo hace, podremos acceder al vídeo en directo, en caso contrario, esa vista no aparecerá en la barra de navegación.

De esta manera, debido a que el streaming se realiza refrescando la misma imagen de manera constante 3 veces por segundo a través de peticiones a la API, evitamos su sobrecarga y posible colapso.

```
$scope.enReserva =  function () {
    $http({
        method: 'GET',
        url: 'https://ofs.fi.upm.es/api/reservations/own',
    }).then(function successCallback(response){
        var reservasPropias = response.data;
        var objetoFecha = new Date();
        angular.foreach(reservasPropias, function(reserva){
            var ahora = new Date(objetoFecha.getYear()+objetoFecha.getMonth()+objetoFecha.getDate() + "T" + objetoFecha.getHours()+":"+objetoFecha.getMinutes());
            var inicio = new Date(reserva.startDate);
            var fin = new Date(reserva.endDate);
            if(ahora > inicio && ahora < fin){
                return true;
            }
        });
        return false;

    }), function errorCallback(response){
        console.log(response.statusText);
    }
}
```

Figura 4.11: Función enReserva() encargada del control de acceso de usuarios a "Vídeo".

Todo el diseño de está vista se realizó de tal manera que fuera web responsive, pudiendo así utilizar esta funcionalidad en cualquier tipo de dispositivo, distribuyéndose los elementos de esta vista de manera vertical en lugar de horizontal como lo mostrado en las figuras 4.8 y 4.12.

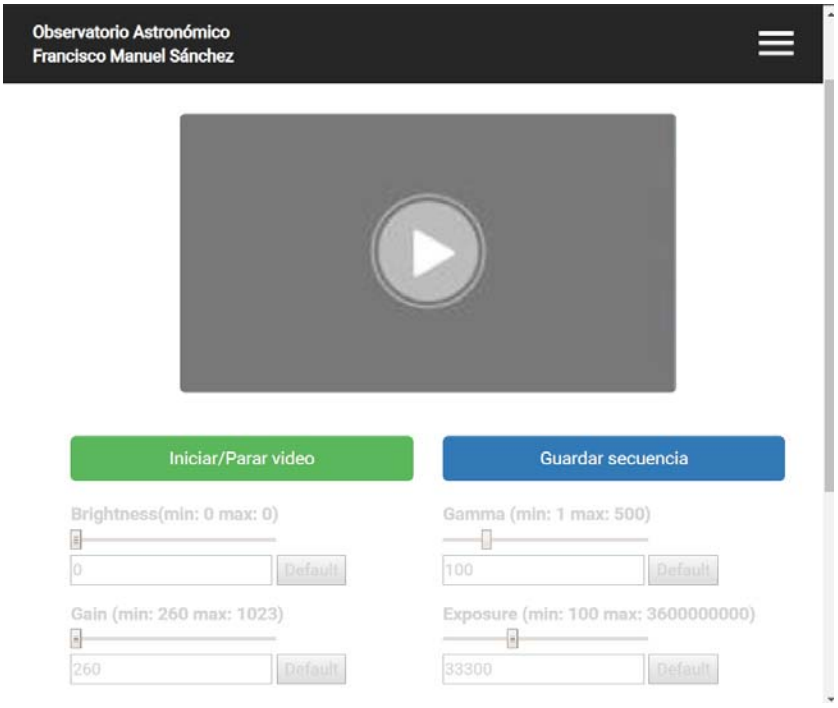

Figura 4.12: Vista del vídeo en directo en una pantalla de 864x635

4.4.2 Galería de imágenes

La vista para la galería de imágenes se desarrolló durante las dos fases del proyecto a partes iguales y está encargada de mostrar a cada usuario de la web las secuencias de imágenes que haya guardado durante los distintos experimentos que haya realizado usando el telescopio y la búsqueda y visualización de las imágenes que hayan sido etiquetadas haciendo uso de los "tags".

Cada usuario podrá ver únicamente sus secuencias guardadas desplegadas en grupos e 9 fotos por página, en caso de querer ver secuencias que hayan sido tomadas por otros usuarios, el usuario deberá hacer uso del sistema de búsqueda por tags.

El diseño de la galería de imágenes (Figura 4.13) se distribuyó de la siguiente forma, comenzando desde la parte superior de la pantalla hasta llegar a la parte inferior de esta:

- Un título centrado en la parte superior de la vista, una barra, esta vez correspondiente a la de búsqueda por tags.

- Un botón justo a continuación de la barra de búsqueda bajo el nombre "Descargar archivos", el cual nos permitirá descargar todo el contenido de nuestra galería en un archivo zip.

- Las secuencias de imágenes que el usuario haya guardado ordenadas en orden cronológico o, en caso de que haya realizado una búsqueda por tags, las imágenes de todos los usuarios que tengan ese tag asignado ordenadas por orden cronológico de subida. Se mostrarán únicamente nueve imágenes por página.

- Botones de paginado para la navegación entre las distintas páginas de imágenes del usuario.

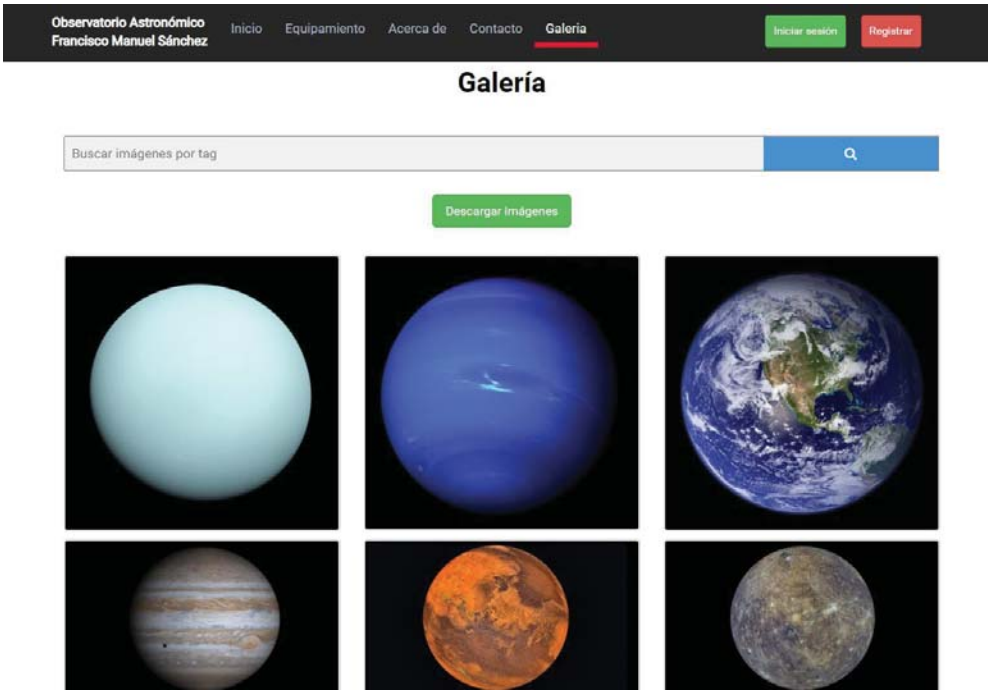

Figura 4.13: Vista de la galería.

Para poder realizar estas funcionalidades, se volvió a hacer uso de las distintas APIs disponibles del Observatorio. Mientras que la maquetación se realizó con Bootstrap y css para hacerla de tipo web-responsive.

De la API del Gateway se utilizó la llamada:

- users/loged. Esta llamada nos permitió crear la función init() en el controlador de la galería que, en conjunción con la llamada a la API de la cámara */TaskAuthor/{user}*, nos permitiría mostrar únicamente las imágenes a cada usuario que hayan sido creadas por él. De tal manera que se usa el nombre del usuario que recibimos de la llamada a *users/loged* para hacer acto seguido la llamada a */TaskAuthor/{user}* devolviéndonos las imágenes de las que ese usuario es propietario. El código de la función se encuentra de manera visual en la Figura 4.14.

```
13
14      function init() {
15          var urlGetInfoUser = "https://ofs.fi.upm.es/api/users/loged";
16          $http.get(urlGetInfoUser).then(function(userInfo){
17              $http({
18                  method: 'GET',
19                  url: $scope.url + '/TaskAuthor/' + userInfo.data.username,
20              }).then(function successCallback(response){
21                  $scope.photos = response.data;
22                  $scope.numPages = function () {
23                      return Math.ceil($scope.photos.Amount / $scope.numPerPage);
24                  };
25
26                  $scope.$watch('currentPage + numPerPage', function() {
27                      totalImagenes();
28                      var begin = (($scope.currentPage - 1) * $scope.numPerPage)
29                      , end = begin + $scope.numPerPage;
30
31                      $scope.filteredTodos = $scope.listPhotos.slice(begin, end);
32                  });
33              }), function errorCallback(response){
34                  console.log(response.statusText);
35              }
36          })
37      }
38
```

Figura 4.14: Función init() del controlador de la galería.

La función init(), mostrada en la figura 4.14, recibe las imágenes propiedad de un determinado usuario a través de la concatenación de las dos llamadas nombradas en el anterior párrafo. La respuesta que recibe de la última llamada llega en forma de un array con los nombres de las fotos en la base de datos. Una vez recibido este array, se llama a la función totalImagenes()(Figura 4.15). Esta función está encargada de realizar la concatenación de los nombres que se reciben como respuesta de la llamada a /TaskAuthor/{user}, a través de un bucle for, en un nuevo array de nombres, que contendrá la ruta completa para realizar el acceso a la base de datos, siendo esta ruta la formada por la llamada a la API, más el nombre recibido por la llamada a /TaskAuthor/{user}, un signo de interrogación y una marca de tiempo.

Una vez concatenadas las imágenes en nuevo array, dividimos el número total de fotos obtenidas y las separamos en porciones de nueve en nueve para poder crear asi la paginación de las imágenes en la vista.

De la API para el control de la cámara DMK se utilizaron las siguientes llamadas:

- /Photo/. Esta llamada es utilizada en la función totalImágenes()(Figura 4.15) para la recepción de las imágenes de la base de datos y su concatenación en un array.

```
38
39      function totalImagenes()
40      {
41          var i;
42          var totalFotos = $scope.photos.Amount;
43          for(i=0; i<totalFotos; i++)
44          {
45              var timestamp = Date.now();
46              var element = $scope.photos.members[i];
47              var photo = $scope.url + '/Photo/' + element + '?' + timestamp;
48              $scope.listPhotos.push(photo);
49          }
50      }
51
```

Figura 4.15: Función totalImagenes() del controlador de la galería.

- /CreateZip/. Esta llamada es la utilizada para realizar el almacenado de todas las imágenes de la galería en un único archivo zip para su descarga por parte del usuario.

Debido a que la API que controla la cámara DMK sigue en desarrollo, cuando se inició el desarrollo de la vista de la galería no existían dos funcionalidades básicas para el funcionamiento de esta. Por lo tanto, se decidió realizar una reunión entre la persona responsable del desarrollo de la API y la persona responsable del desarrollo del nuevo módulo de la web. Tras esta reunión, las siguientes funcionalidades fueron añadidas:

- /TaskAuthor/{user}. A través de esta llamada recibiremos un listado de las imágenes de las que sea propietario, es decir, haya guardado como secuencia, el usuario {user}. Gracias al desarrollo de esta llamada se pudo crear la función init() (Figura 4.13), a través de la cual somos capaces de mostrar las imágenes de la galería de manera personalizada por usuario.

- /TaskTags/{tag}. Esta llamada, de funcionamiento muy similar a la llamada */TaskAuthor/*, realizará una búsqueda en la base de datos de todas las imágenes almacenadas con el campo {tag}. Esta devolverá un array con los nombres de las imágenes en la base de datos que tendrá que ser procesado posteriormente por la función totalImagenes()(Figura 4.15). Esta llamada es utilizada en la función getByTag()(Figura 4.16).

```
    $scope.getByTag = function()
    {
        $http({
            method: 'GET',
            url: $scope.url + '/TaskTags/' + $scope.tag,
        }).then(function successCallback(response){
            $scope.photos = response.data;
            $scope.numPages = function () {
                return Math.ceil($scope.photos.Amount / $scope.numPerPage);
            };

            $scope.$watch('currentPage + numPerPage', function() {
                var i;
                var totalFotos = $scope.photos.members.length;
                $scope.listPhotos = [];
                for(i=0; i<totalFotos; i++)
                {
                    var timestamp = Date.now();
                    var element = $scope.photos.members[i];
                    var photo = $scope.url + '/Photo/' + element + '?' + timestamp;
                    $scope.listPhotos.push(photo);
                }
                var begin = (($scope.currentPage - 1) * $scope.numPerPage)
                , end = begin + $scope.numPerPage;

                $scope.filteredTodos = $scope.listPhotos.slice(begin, end);
            });
        }), function errorCallback(response){
            console.log(response.statusText);
        }
    }

}]);
```

Figura 4.16: Función getByTag().

Además del desarrollo por completo del código de la vista y el controlador para la Galería, hubo que añadir código adicional en otras vistas, como en la vista Index.html, en el que se añadió nuevos campos a la barra de navegación para poder acceder a la Galería. También se hizo uso de funciones ya existentes previo al desarrollo del nuevo módulo, como la función isLogged(), que comprueba si el usuario de la web ha iniciado sesión o no, mostrando así las nuevas partes de la web a los usuarios que hayan realizado el inicio de sesión de forma satisfactoria.

5. CONCLUSIONES Y LÍNEAS FUTURAS

Una vez finalizado el proyecto y revisando el resultado final, es muy reconfortante ver el modo en que éste ha contribuido a la mejora de la web del Observatorio, aportando nuevas funcionalidades a los usuarios que la visitan.

Gracias a este desarrollo y al API en el que se apoya, los usuarios pasan a ser parte interactiva de la web al permitirles utilizar la cámara DMK del observatorio. Pueden modificar los atributos de las imágenes del reproductor de video en directo y grabarlas en su librería, además de asociarle tags para su categorización y facilite su posterior búsqueda a través de la galería.

La creación de la Galería de imágenes favorece el trabajo colaborativo entre los usuarios al darles acceso, no solo a las imágenes almacenadas por ellos, sino también a las del resto de usuarios.

Todo esto ha supuesto un importante avance en la operativa con las imágenes y una considerable mejora de la experiencia de nuestros usuarios con la web.

Satisface ver que se han conseguido prácticamente todos los objetivos marcados para el proyecto, a pesar del giro que sufrió a mitad del periodo y que ha supuesto un esfuerzo adicional y un recorte en el tiempo de desarrollo. A la vez, entristece que la limitación del tiempo no haya permitido obtener un resultado final más completo con el producto ya integrado con la API de Google.

Como líneas futuras para este trabajo, se detallan a continuación las más importantes:

- Creación de un nuevo end-point para la obtención de las secuencias de imágenes, de tal manera que la carga de llamadas no sea tan grande para la API de control de la cámara DMK y esta pueda dedicarse exclusivamente al sistema de streaming.

- Integración de la API de Google para el control de la aplicación de Drive, de tal manera que el peso de las imágenes guardadas por los usuarios sea gestionada por esta aplicación y evitando el almacenamiento y gestión por parte del Observatorio.

- Mejora en el modo en el que se realiza el video en “directo”, dejando de ser una imagen refrescándose de manera continua tres veces por segundo y siendo una emisión real en directo.

- Mejora del estilo de diseño de las distintas vistas que componen la página web.

6. BIBLIOGRAFÍA

[1] «Grupo Cíclope,» https://ofs.fi.upm.es/acerca

[2] «Wikipedia,» https://es.wikipedia.org/wiki/Astronom%C3%ADa

[3] «Wikipedia,» https://es.wikipedia.org/wiki/Observatorio

[4] «Laboratorio de procesado de imagen,» http://www.lpi.tel.uva.es/~nacho/docencia/EMC/trabajos_02_03/RADIOASTRONOMIA/web/Indice/Ins/I_Ins/Obs/6_1_2/His.htm

[5] «Big Telescope Alt-azimuthal,» http://goldmine.mib.infn.it/educational/galleries/PhotoGallery4/pages/h5m_BTA_6m.html

[6] «Wikipedia,» https://es.wikipedia.org/wiki/Telescopio_Carlsberg_Meridian

[7] «Wikipedia,» https://es.wikipedia.org/wiki/Observatorio_Montegancedo

[8] «ETS de Ingenieros Informáticos UPM,» https://www.fi.upm.es/?pagina=827

[9] «MEGARA,» https://guaix.fis.ucm.es/megara

[10] «Astromadrid,» http://astromadrid.es/

[11] «Grupo Cíclope,» http://mercurio.datsi.fi.upm.es/index.php/es/proyectos/

[12] «ETS de Ingenieros Informáticos,» https://www.fi.upm.es/?id=om

[13] «Wikipedia,» https://es.wikipedia.org/wiki/JavaScript

[14] «Wikipedia,» https://es.wikipedia.org/wiki/Document_Object_Model

[15] «Wikipedia,» https://es.wikipedia.org/wiki/AngularJS

[16] «Conociendo GitHub,» https://conociendogithub.readthedocs.io/en/latest/data/introduccion/

[17] «Wikipedia,» https://es.wikipedia.org/wiki/Visual_Studio_Code

[18] «Wikipedia,» https://es.wikipedia.org/wiki/Servidor_HTTP_Apache

[19] «About the Apache HTTP Server Project,» http://httpd.apache.org/ABOUT_APACHE.html

[20] «Arweb,» https://www.arweb.com/chucherias/%C2%BFque-es-bootstrap-y-como-funciona-en-el-diseno-web/

[21] «SQLite,» https://www.sqlite.org/index.html

[22] «Wikipedia,» https://es.wikipedia.org/wiki/SQLite

Este documento esta firmado por

Firmante	CN=tfgm.fi.upm.es, OU=CCFI, O=Facultad de Informatica - UPM, C=ES
Fecha/Hora	Mon Jul 01 17:40:34 CEST 2019
Emisor del Certificado	EMAILADDRESS=camanager@fi.upm.es, CN=CA Facultad de Informatica, O=Facultad de Informatica - UPM, C=ES
Numero de Serie	630
Metodo	urn:adobe.com:Adobe.PPKLite:adbe.pkcs7.sha1 (Adobe Signature)