Aplicación web para la reproducción personalizada de videos en tiempo real desde una red de cámaras

Grado en Ingeniería Informática

AUTOR: Daniel Martínez Muñoz

TUTORIZADO POR:
Javier Sánchez Pérez y Nelson Monzón López

26 de julio de 2021

Agradecimientos

Me gustaría dar las gracias a mi familia, a mis amigos y especialmente a mi pareja, por apoyarme diariamente y por ayudarme a alcanzar mis objetivos.

También me gustaría agradecer a mis tutores Nelson Monzón López y Javier Sánchez Pérez por la ayuda que me han ofrecido durante el proyecto.

Agradecer a todas las personas del equipo QAISC, por aceptarme en su equipo. Especialmente a Alejandro Cabrera por el tiempo que dedicó a ayudarme en el desarrollo del back-end y Andrés Alonso-Martirena por ejercer de Product Owner aportando ideas y feedback durante el desarrollo de las funcionalidades.

Finalmente, agradecer a la Plataforma Oceánica de Canarias [1] (PLOCAN), y su red de observación de cámaras costeras, y a la Autoridad Portuaria de Las Palmas de Gran Canaria [2] por ofrecernos el entorno de trabajo necesario para la realización de este trabajo. Agradecemos a los grupos empresariales Anfi del Mar [3] y Radisson Blue [4] por brindarnos su apoyo en el marco de la iniciativa "Smart Coast AI Lab for tourism"liderada por QAISC y en colaboración con el Centro de Tecnologías de la Imagen [5] (CTIM).

[1] https://www.plocan.eu/
[2] http://www.palmasport.es/es/
[3] https://anfi.com/es/
[4] https://www.radissonhotels.com/es-es/hoteles/radisson-blu-resort-gran-canaria
[5] https://ctim.ulpgc.es/site/

Resumen

El objetivo principal de este Trabajo Fin de Grado (TFG) es desarrollar una aplicación web que permitirá visualizar en directo la transmisión de una red de cámaras ubicadas en diferentes zonas de Gran Canaria, ofreciendo una serie de funcionalidades básicas como: visualizar el vídeo en directo, mover el enfoque de las cámaras PTZ[6], mostrar la información de las cámaras y visualizar las últimas capturas del vídeo en directo (*time-lapse*). Además, la aplicación cuenta con un sistema de roles que permiten el acceso controlado a las diferentes funciones de las cámaras.

Palabras claves

Cámaras PTZ, WebRTC, Mapa interactivo, Aplicación web, Angular, SCRUM.

[6]PTZ es un acrónimo de Pan, Tilt, Zoom [1]. Las cámaras PTZ tienen la capacidad de rotar en el eje horizontal y vertical, así como acercar o alejar el enfoque de un área (zoom). Estas cámaras pueden ser controladas a distancia, se pueden programar para que sigan un patrón o se pueden poner en modo inteligente, que detecta automáticamente el movimiento de un objeto o una persona.

Abstract

The subject of my final degree project consists in the development of a web application that will allow users to access to a network system of several cameras located in different zones of Gran Canaria island, offering a series of basic functionalities like as view streaming, adjust the focus of the PTZ[7] cameras, show information of the cameras and view a time-lapse. In addition, the web application has a system of roles that allow controlled access to the different functions of the cameras.

Keywords

PTZ cameras, WebRTC, Interactive Map, Web Application, Angular, SCRUM.

[7]PTZ cameras have the ability to rotate Panning, Tilt and as well as zooming in or out on an area. These cameras can be controlled remotely, can be programmed to follow a pattern or can be put into intelligent mode that automatically detects the movement of an object or person.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

La empresa Qualitas Artificial Intelligence and Science (QAISC) [1] está realizando el despliegue de una red de cámaras PTZ, las cuales son capaces de apuntar en distintas direcciones y acercarse o alejarse a grandes distancias gracias a su capacidad de zoom, en varias localizaciones de costa, playa y litoral en la isla de Gran Canaria. Tiene la intención de desarrollar una interfaz interactiva, similar al servicio que ofrece la empresa Windy[2] . En este caso, el objetivo sería ofrecer un servicio similar pero integrado en el sistema PORTUS[3] desarrollado por el grupo QUALITAS.

En este sentido, este Trabajo Fin de Grado (TFG) persigue el desarrollo de un prototipo que permitirá a distintos usuarios ver en directo la transmisión de las cámaras ubicadas en diferentes zonas de Gran Canaria de manera interactiva en un mapa. Los usuarios podrán pulsar en las distintas cámaras y podrán visualizar la transmisión en tiempo real capturada por las cámaras.

Primeramente, se analiza un breve documento de especificaciones y *mock-ups* que han sido la guía inicial del trabajo. Además, se ha analizado la configuración, los datos de acceso para la transmisión de la información y en general el funcionamiento del sistema PORTUS.

En una segunda fase, se implementará un mapa interactivo para poder navegar en él, y poder ver todas las cámaras que se encuentren disponibles en el sistema.

En una tercera fase, se desarrollará una interfaz web que permite visualizar la retransmisión en directo (o *streaming*) de cada cámara, ofreciendo una serie de funcionalidades básicas como mover el enfoque de las cámaras, visualizar las últimas capturas tomadas del *streaming* (*time-lapse*),la información de la cámara como: nombre, ubicación, propietario, etc. y cuestiones de índole similares.

Además de desarrollar la citada interfaz web, se ha participado en algunas tareas de *back-end* como el acceso a servicios mediante una API, que se encuentran ya desarrollados por la empresa.

[1] www.qaisc.com
[2] https://www.windy.com/
[3] https://portus.puertos.es/

1.1. Motivación

Como motivación principal de este trabajo, se quieren aplicar los conocimientos adquiridos en Ingeniería Informática. A su vez, se ha decidido realizar un proyecto web, no solo por el interés en el aprendizaje en tecnologías de esta índole, sino también porque ha permitido trabajar dentro de un entorno empresarial, centrado en el I+D, colaborando en el desarrollo de un proyecto de software real mediante metodologías ágiles haciendo uso de los distintos artefactos, herramientas y roles propios de SCRUM. Respecto a estos últimos, el CEO del grupo QUALITAS ha ejercido de Product Owner, los tutores del trabajo como Scrum Master y el estudiante ha colaborado estrechamente con el equipo de desarrollo de la empresa para conseguir los distintos objetivos de este TFG.

1.2. Objetivos del trabajo

Este proyecto plantea el desarrollo de los siguientes objetivos:

- Desarrollo de una interfaz web integrada en un mapa, que permita visualizar una red de cámaras.
- Diseñar y desarrollar una interfaz gráfica de acceso y manejo de vídeos capturados en tiempo real, que contemplará parámetros de usabilidad y experiencia de usuario.
- Aprender nuevas tecnologías, extender conocimientos de ingeniería del software y aplicar técnicas de desarrollo ágiles.
- Conocer herramientas como OpenStack Swift que faciliten el almacenamiento y acceso de imágenes y vídeos.
- Aplicación de un flujo de trabajo que aplique técnicas de integración y entrega continua.
- Familiarización con metodologías ágiles en un contexto empresarial.

1.3. Competencias específicas

Seguidamente se mencionan las competencias del Grado en Ingeniería Informática de la Universidad de Las Palmas de Gran Canaria que han sido cubiertas en este trabajo:

TI01: Capacidad para comprender el entorno de una organización y sus necesidades en el ámbito de las tecnologías de la información y las comunicaciones.

TI02: Capacidad para seleccionar, diseñar, desplegar, integrar, evaluar, construir,gestionar, explotar y mantener las tecnologías de hardware, software y redes,dentro de los parámetros de coste y calidad adecuados.

TI03: Capacidad para emplear metodologías centradas en el usuario y la organización para el desarrollo, evaluación y gestión de aplicaciones y sistemas basados en tecnologías de la información que aseguren la accesibilidad, ergonomía y usabilidad de los sistemas.

TI05: Capacidad para seleccionar, desplegar, integrar y gestionar sistemas de información que satisfagan las necesidades de la organización, con los criterios de coste y calidad identificados.

1.4. Estado actual del proyecto

En este momento, la empresa contaba con tres cámaras operativas ubicadas en: el puerto Nelson Mandela, Anfi del Mar y el hotel Radisson Blu. Por otra parte, tenían una aplicación web para ver la lista de las cámaras, editarlas, eliminarlas y ver su *streaming* correspondiente. Además, contaba con una funcionalidad para cambiar el enfoque de las cámaras mediante un sistema de coordenadas X, Y y Z.

Por otro lado, esta aplicación también hace el papel de API, que está a medio desarrollar. Sin embargo, para el desarrollo de del proyecto se tuvo que crear las funciones en la API respectivas a las cámaras, al *streaming*, guardar en OpenStack Swift [2] y para el sistema de autenticación; además de sus respectivos *endpoints*.

Por último, el Product Owner había redactado un PDF con los requisitos del sistema, que se utilizará para desarrollar las funcionalidades requeridas.

1.5. Presupuesto

En este sección mostraremos el presupuesto necesario para la realización de este proyecto de fin de carrera. Primero lo desglosaremos en dos partes y, por último, calcularemos el coste total que conlleva realizar este proyecto.

1.5.1. Coste del hardware

Hardware	Coste
Ordenador portátil	800 €
Total	**800 €**

Cuadro 1.1: Coste hardware del proyecto

1.5.2. Coste del personal

Personal	Nº Horas	Coste/Hora	Coste
Alumno	300	13 €	3900 €
Tutores	20	25 €	500 €
Total			**4400 €**

Cuadro 1.2: Coste hardware del personal

1.5.3. Presupuesto final

Concepto	Coste
Coste del hardware	800 €
Coste del personal	4400 €
Presupuesto total	**5200 €**

Cuadro 1.3: Presupuesto total del proyecto

Capítulo 2

Estado del arte

2.1. Windy

Windy [3] es una aplicación que permite visualizar distintas capas de información meteorológicas sobre un mapa. Esto ayuda a monitorizar el estado del viento, las lluvias, las temperaturas, el oleaje, la calidad del aire, etc., a partir de datos a nivel mundial y en tiempo real, así como poder consultar los datos en formato histórico o hacer predicciones de estos. Además, Windy destaca por la calidad y la buena animación de la información sobre su mapa. Sin embargo, un punto negativo es la cantidad de opciones que tiene disponibles, que hacen que el usuario se vea saturado de tanta información.

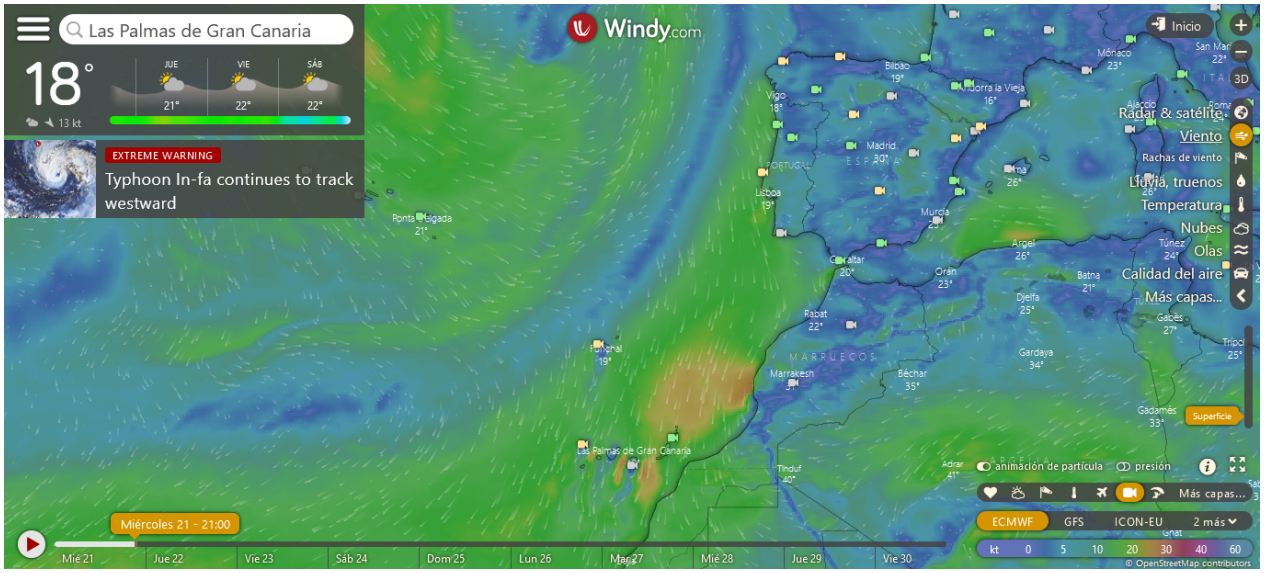


Ilustración 2.1: Página principal de la web de Windy.com

Ahora bien, centrándonos en el tema de las cámaras, consigue representar el icono de una cámara sobre el mapa, ubicándolas en sus respectivas posiciones. No obstante, al pulsar en

una de las cámaras esta no nos muestra la transmisión en tiempo real, si no que nos enseña un vídeo resumen del directo de las ultimas 24 horas, aunque también se puede cambiar a 30 días, 12 meses o de por vida. Además de esto, al pasar el ratón por encima de una cámara se muestra en un Pop-up la última imagen capturada, que tiene una serie de opciones como ver el sitio web de proveedor de la cámara, añadirla a favoritos, editarla, compartir, etc.

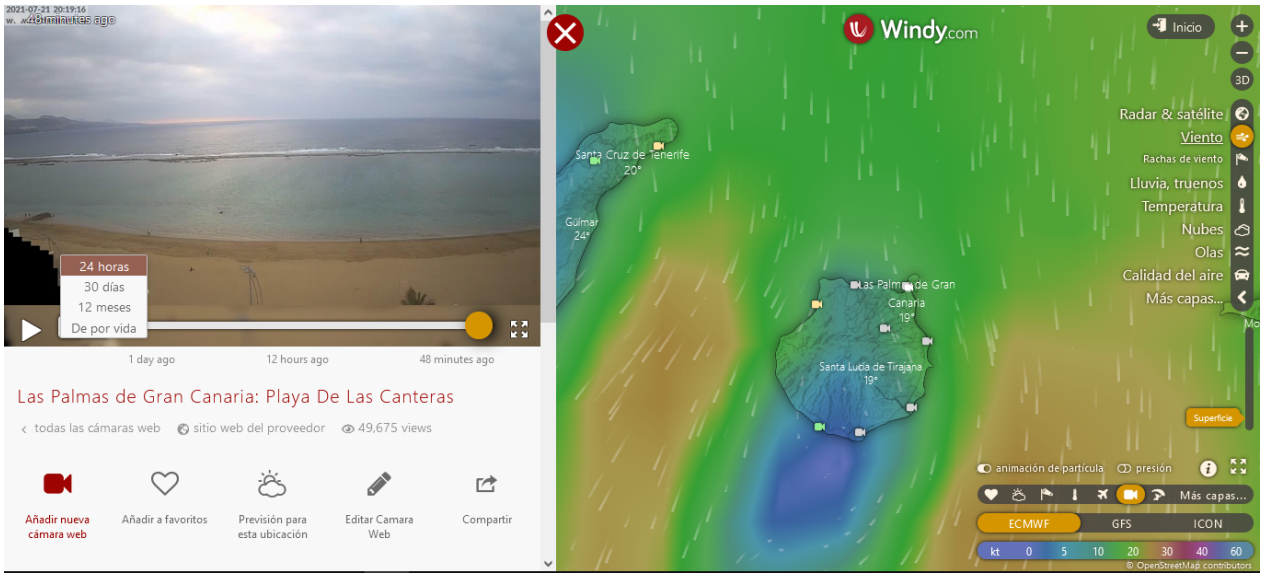


Ilustración 2.2: Panel lateral al abrir una cámara en Windy.com

Por último, Windy cuenta con un sistema para añadir y editar cámaras, basado en fases que se van desbloqueando a medida que se va completando el formulario, lo cual resulta llamativo pero es menos práctico.

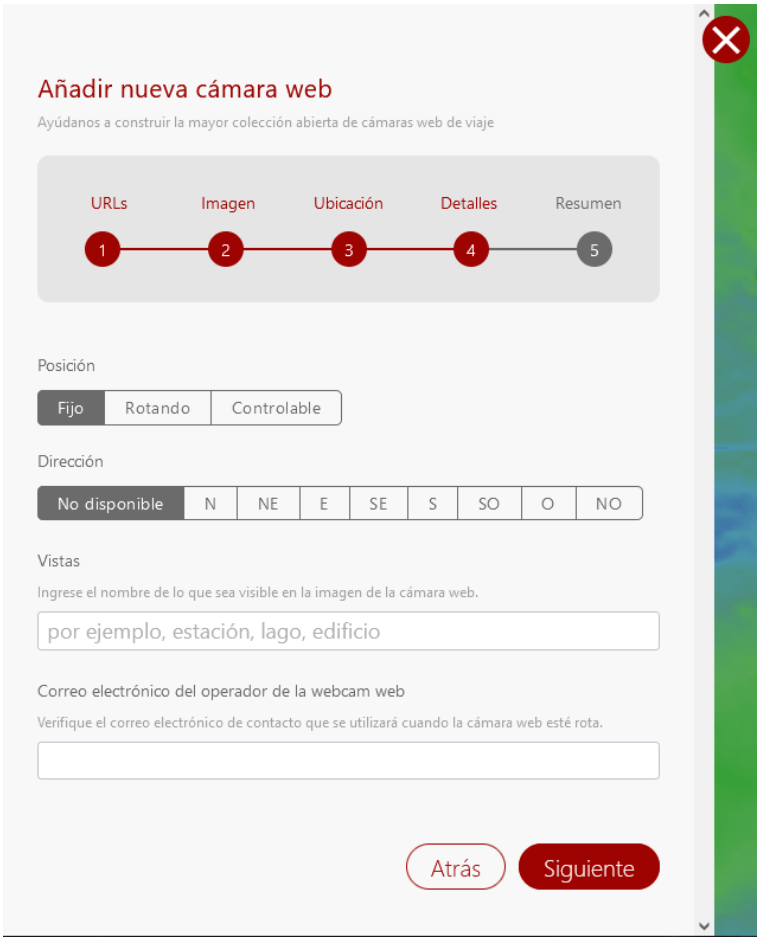


Ilustración 2.3: Vista para añadir nueva cámara

2.2. CanariasLife

CanariasLife [4] es un portal turístico que ofrece la posibilidad de ver en directo la transmisión de una red de cámaras situadas en los puntos más emblemáticos y atractivos de Canarias, con el objetivo de promover el turismo facilitando a los usuarios la posibilidad de ver en directo los sitios que quiera visitar. En esta ocasión las cámaras no se muestran sobre un mapa sino que se visualizan como una lista de miniaturas que se pueden filtrar por isla.

Ilustración 2.4: Portal web de Canarias Life

Al pulsar en una de las miniaturas se abre una nueva ventana que muestra el streaming de la cámara, información sobre el sitio de la transmisión y una recomendación de cámaras en directos que están ordenadas por cercanía a la cámara seleccionada. Una opción a destacar es que la página cuenta con una funcionalidad para ver un *time lapse* del streaming correspondiente a las últimas 24 horas.

Ilustración 2.5: Vista del streaming desde canariaslife.com

2.3. Frameworks para la creación de mapas web

A continuación, mostramos la diferencia de concepto y diseño entre la API de Leaflet y OpenLayers. De esta forma, tenemos un punto de partida para decidir cuál de estas dos API es más adecuada para nuestro proyecto. Para comprender cuál es la diferencia básica entre estas dos API, expondremos la definición de cada librería.

2.3.1. Openlayers

OpenLayers[1] es una librería que nos permite elaborar mapas interactivo, tiene puede mostrar distintos tipos de mapas, datos vectoriales, marcadores, capas de todo tipo. etc. Esta librería cuenta con un gran rendimiento, contiene numerosas funcionalidades para cualquier cosa, además es totalmente gratuito y *OpenSource*.

El propósito fundamental de OpenLayers es impulsar la utilización de información geográfica de todo tipo. Esto le permite incorporar funcionalidades avanzadas que otro librerías no tiene la capacidad de ejecutar, pero a consecuencia de esto, aumenta su peso y la dificultad de uso.

[1]https://openlayers.org/

2.3.2. Leaflet

Leaflet[2] es otra librería JavaScript de código abierto para realizar mapas interactivos con un entorno amigable en cualquier dispositivo. Tiene todas las características que la mayoría de los desarrolladores necesitan, a pesar de su pequeño tamaño. Leaflet está diseñado teniendo en cuenta la simplicidad, el rendimiento y la facilidad de uso. Funciona de manera eficiente en todas las principales plataformas móviles y de escritorio.

Tiene una API bien documentada y un código fuente simple y legible por el que se puede contribuir su facilidad de uso. Además Leaflet cuenta con una cantidad considerable de *plug-ins*. La mayoría de de los mapas disponibles en la web carecen de funcionalidades complejas en las que Leaflet puede cumplir todas las necesidades, por eso se ha convertido en la librería más extendida en la red.

2.3.3. Comparativa

De lo comentado anteriormente, podemos deducir que las principales diferencias entre ambas API surgen de la facilidad de uso y del diseño de su funcionalidad. Veamos ahora, algunas consecuencias de esas dos diferencias:

2.3.3.1. Facilidad de uso

Si comparamos en cada librería las líneas de código que se necesitan solo para crear un mapa con una sola capa de información y sin ninguna funcionalidad adicional, como podemos observar a continuación la figura 2.7 corresponde con Openlayers y la figura 2.6 corresponde con Leaflet:

```
private initMap(): void {

  // centrar mapa en Gran Canaria
  this.map = L.map('map').setView([27.95, -15.60], 10);
  L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
    minZoom: 2,
    maxZoom: 14,
    id: 'mapbox/streets-v11',
  }).addTo(this.map);
```

Ilustración 2.6: Creación de mapa con Leaflet

[2]https://leafletjs.com/

```
ngAfterViewInit(): void {

  this.layer = new TileLayer({
    source: new OSM(),
  });
  this.view = new OlView({
    center: Proj.fromLonLat([-15.60, 27.95]),
    zoom: 10,
    minZoom: 2,
    maxZoom: 14,
  });
  this.map = new OlMap({
    target: 'map',
    layers: [this.layer],
    view: this.view,
  });
```

Ilustración 2.7: Creación de mapa con Openlayers

Vemos que en Leaflet necesitamos menos líneas para crear un mapa con una configuración por defecto. Incluso para crear un mapa sencillo existen en OpenLayers se necesitan muchas más de líneas de código, pero la principal diferencia, está en el número de objetos distintos que se utilizan en cada caso.

En la figura 2.6, Leaflet utiliza solo dos tipos de objetos (L.Map y L.tileLayer), mientras que OpenLayers utiliza 5 objetos distintos (ol.Map, ol.layer.Tile, ol.source.OSM, ol.View y ol.proj.fromLonLat). Más tipos de objetos significa que es más difícil comenzar a usar la librería, pero a cambio, tenemos una mayor modularización lo cual nos ayuda a la reutilización del código y mejora capacidad para desarrollar mapas más personalizados.

Otro ejemplo de simplicidad lo encontramos en la utilización de los marcadores, el *binding* de los eventos y de los Pop-ups. En Leaflet para añadir un el icono de una cámara al mapa tan solo tenemos que crear un marcador, añadiendo las coordenadas donde se va a ubicar y el icono que queremos representar, en nuestro caso una cámara (figura 2.8). Para ello creamos un objeto icono al que le pasamos una imagen y el tamaño de esta, finalmente lo añadimos al mapa.

```
let cam_green = L.icon({
  iconUrl: '/assets/imgs/camera_green.png',
  iconSize: [24, 24],
});

let cam_Nelson_Mandela = L.marker([28.1622092, -15.4014788], {
  icon: cam_green,
}).addTo(this.map)
.bindTooltip('Puerto Nelson Mandela', { direction: 'top' })
.on('click', function (e) {
  $('.close-panel').removeClass('none');
  $('.camara-panel').removeClass('none');
});
```

Ilustración 2.8: Creación de Pop-ups en Leaflet

En cambio, en OpenLayers primero tenemos que crear un punto en la posición determinada y después crear un icono al que le asignaremos una imagen y el tamaño de esta, le asignamos el estilo del icono al punto en el mapa, después crear una capa y asignarle el icono y finalmente añadir esta capa al mapa (figura 2.9. En total hemos tenido que crear cinco objetos a diferencia de los dos necesario en Leaflet.

```
addLayerCameras(cameras: Camera[]) {
  cameras.forEach((camera: Camera) => {
    let iconFeature = new Feature({
      geometry: new Point(
        Proj.fromLonLat([camera.dd_longitude, camera.dd_latitude])
      ),
      name: 'camera/' + camera.id,
    });
    let iconStyle = new Style({
      image: new Icon({
        src: 'assets/imgs/cam_green.png',
        size: [24, 24],
      }),
    });
    iconFeature.setStyle(iconStyle);
    var vectorSource = new VectorSource({
      features: [iconFeature],
    });
    this.vectorLayer = new VectorLayer({
      source: vectorSource,
    });
    this.map.addLayer(this.vectorLayer);
  });
}
```

Ilustración 2.9: Creación de Pop-ups en Openlayers

```
  this.map.on('pointermove', (event: any) => this.hoverMouse(event));
}

hoverMouse(event: any) {
  var hit = this.map.forEachFeatureAtPixel(
    event.pixel,
    (feature) => {
      this.selectedFeature = feature;
      return true;
    },
    { hitTolerance: 5 }
  );
  hit ? this.openCameraPopup(event) : this.closePopup();
}
```

Ilustración 2.10: Unir un Pop-up con un evento hover en Openlayers

Además de esto en Leaflet, para *bindear* un Pop-up tan solo tenemos que llamar al método **bindTooltip()** y pasarle como parámetro lo que queremos mostrar en el Pop-up como un simple nombre o incluso un cualquier código en HTML, y para bindear un evento simplemente llamamos al método **on()** del objeto *market* y ya estaría unido el evento con el icono. En cuanto a Openlayers, si queremos *bindear* un Pop-up y un evento a una cierta Feature vamos a tener que disparar un evento cada vez que se mueva el ratón para saber si se ha posicionado encima de un Feature y saber si podemos abrir el Pop-up (figura 2.10).

2.3.3.2. Diseño de funcionalidad

- **Openlayers:** La API de OpenLayers tiene el comportamiento contrario. Por una parte, dispone de un núcleo más extenso y con menos desarrolladores, lo cual, conlleva que esos desarrolladores formen parte del núcleo de la API y por lo tanto sea algo más difícil que abandonen el proyecto.

  Por otra parte, al desarrollar con OpenLayers en un entorno como Node.js, podemos utilizar solo aquellas clases necesarias para nuestras aplicaciones tal y como se muestra en esta página. Así, podemos excluir de nuestro código todos aquellos tipos de objeto que no vayamos a utilizar. Dicho de otro modo, podemos mantener nuestras aplicaciones lo más ligeras posibles en función de las funcionalidades utilizadas. Una funcionalidad más avanzada, lógicamente, conlleva una aplicación menos ligera.

- **Leaflet:** La simplicidad de Leaflet se combina muy bien con el uso de plugins. Tal como hemos visto en la definición de Leaflet, esta API puede extender sus funcionalidades en base al uso de plugins. De este modo, se mantiene la simplicidad y ligereza del núcleo sin renunciar al uso y desarrollo de nuevas funcionalidades.

  Cualquier programador experimentado puede desarrollar nuevos plugins, lo cual, es tanto una ventaja como un inconveniente. Lógicamente es una ventaja porque ello facilita el desarrollo de nuevos plugins, pero también es un inconveniente debido a que la funcionalidad de esos plugins (con su adaptación a nuevas versiones de la API, soporte al usuario, etc.) recae en manos de terceros. Si esos terceros dejan de dar soporte, el plugin puede terminar por desaparecer dejándonos colgados durante el proceso de actualización de nuestras aplicaciones.

2.3.4. Conclusión

Si el objetivo fuera desarrollar una aplicación que solo muestre iconos o marcadores y no requiera de ningún tipo de funcionalidad compleja, sin duda eligiera Leaflet ya que su manejo es mucho más simple y me permitiría ahorrar bastante tiempo.

Por otro lado, si quisiera desarrollar una aplicación más compleja y que fuera crucial el rendimiento, como es el caso del proyecto del TFG, lo más razonable sería es usar Openlayers ya que esta librería nos proporciona mejor rendimiento y nos permite desarrollar funcionalidades más complejas.

Capítulo 3

Recursos utilizados

3.1. Tecnologías

3.1.1. TypeScript

Para conocer TypeScript, es fundamental hablar primero acerca de JavaScript. Actualmente, JavaScript es uno de los lenguajes más conocidos y populares, pues debido a su rápida evolución, ha sido posible la aparición de herramientas de desarrollo o lenguajes que hagan más sencillo el día a día de los desarrolladores. Por tanto, el progreso de JavaScript ha ocasionado que aparezca un nuevo lenguaje: TypeScript, para dar soluciones a muchos de los problemas presentes en JavaScript.

En definitiva, podemos decir que TypeScript [5] es como si fuera un superset de JavaScript, cuyo resultado final es un código de JavaScript.

TypeScript encapsula varios elementos:

- JavaScript 5 se considera actualmente como un estándar, ya que es comprendido por todos los navegadores, incluido Node.js a nivel de servidor.
- En vez de usar el tipado dinámico de JavaScript, se incorpora la utilización de un tipado fuerte con el fin de procurar resolver una serie de bugs que pueden aparecer al no controlar el tipo de dato que se maneja.
- Aparece la posibilidad de utilizar interfaces con el objetivo de poder aplicar programación orientada a objetos o determinar nuestros propios tipos.
- Para que los navegadores pueden interpretar TypeScript, este debe ser compilado a JavaScript.
- Typescript es un superset de JavaScript, esto es, que en todos los casos en los que se puede emplear JavaScript, también puede utilizarse Typescript.

- Recomendación de qué argumentos recibe una función, autocompletado de código, recomendación de qué tipo retorna una función.

TypeScript fue la herramienta principal para escribir la lógica de los componentes, además su fuerte tipado y su autocompletado fueron de gran ayuda a lo largo del desarrollo.

3.1.2. HTML

El Lenguaje de Marcado de Hipertexto, conocido como HTML [6], es el código empleado para desplegar y estructurar tanto de una página web como sus contenidos. Ejemplos de sus contenidos podrían ser una lista con viñetas, párrafos, o tablas de datos e imágenes. Sin embargo, cabe destacar que HTML no constituye un lenguaje de programación, sino un lenguaje de marcado para determinar la estructura del contenido.

Por tanto, HTML se basa en una serie de elementos empleados para encerrar diversas partes del contenido con el objetivo de que se vean o comporten de cierta manera. Las etiquetas de encierre pueden hacer que las palabras cambien a cursiva, agrandar o achicar la letra; hacer de una imagen o una palabra un hipervínculo a otro sitio, etc.

1. **La etiqueta de apertura:** consiste en el nombre del elemento (en este caso, p), encerrado por paréntesis angulares de apertura y cierre. Establece dónde comienza o empieza a tener efecto el elemento, en este caso, dónde es el comienzo del párrafo.
2. **La etiqueta de cierre:** es igual que la etiqueta de apertura, excepto que incluye una barra de cierre (/) antes del nombre de la etiqueta. Establece dónde termina el elemento, en este caso dónde termina el párrafo.
3. **El contenido:** este es el contenido del elemento, que en este caso es sólo texto.
4. **El elemento:** la etiqueta de apertura, más la etiqueta de cierre, más el contenido equivale al elemento.

Como no puede ser de otro manera, HTML sirvió para representar todo la estructura de los componentes del sitio web.

3.1.3. CSS

CSS (hojas de estilo en cascada) [7] consiste en un lenguaje declarativo que se ocupa la apariencia de las páginas web en el navegador, el cual navegador lleva a cabo declaraciones de estilo CSS a los elementos seleccionados y así mostrarlos de forma adecuada. Dicha declaración de estilo incluye las propiedades y sus respectivos valores, que establecen la apariencia de una página web. El término en cascada alude a las reglas que gobiernan la priorización de los selectores para cambiar el aspecto de una página, siendo esto un factor muy importante, pues un sitio web complejo puede poseer inmensas cantidades de reglas CSS, siendo una regla CSS una serie de propiedades vinculadas con un selector .

Cabe destacar que CSS, junto con HTML y JavaScript CSS, constituye una de las tres tecnologías web fundamentales. Por lo general, CSS diseña elementos HTML , aunque también puede utilizarse con otros lenguajes de marcado como XML o SVG.

Una vez tenemos la estructura de todos los componentes y la lógica de estos, podemos aplicar CSS para darle estilos y un sentido visual a nuestros componentes.

3.1.4. SQL

SQL es un lenguaje [8] de consulta empleado como interfaz en la comunicación con bases de datos, la ejecución de operaciones de acceso y el manejo de la información almacenada. Este lenguaje es útil para obtener el acceso a la información almacenada en las bases de datos.

Se trata un lenguaje sencillo de consulta el cual otorga la posibilidad de llevar a cabo operaciones de selección, inserción, actualización y borrado de datos, así como operaciones administrativas sobre las bases de datos.

Por otro lado, SQL es un estándar mantenido por ANSI. En consecuencia, las bases del lenguaje son idénticas en la mayoría de los motores de bases de datos relacionales. Una persona que posea conocimientos sobre las bases del lenguaje y que tenga la posibilidad de acceder a una referencia básica del motor particular que esté manejando estará en condiciones de escribir consultas para cualquier base de datos. A pesar de que hay insignificantes diferencias a la hora de planificar los bloques de las sentencias y precisar operaciones o funciones de administración, la estructura de las consultas es prácticamente igual. Asimismo, en función del motor de base de datos, tiene la posibilidad de soportar algunas operaciones extra cómo transacciones o procedimientos, así como un juego de funciones específico.

PostgreSQL fue el gestor de base de datos utilizado en este proyecto, ya que buscamos mantener los datos estructurados y relacionados entre ellos. Ahora bien, SQL es el lenguaje con el que podemos hacer las consultas a nuestra base de datos y traer los datos que necesitemos.

3.1.5. WebRTC

WebRTC, *Web Real-Time Communications* o comunicaciones web en tiempo real, es una tecnología que posibilita que los sitios web y aplicaciones puedan capturar y eventualmente retransmitir audio y/o vídeo, así como realizar el intercambio de datos arbitrarios entre navegadores sin que sea imprescindible un intermediario. El conjunto de estándares que comprende WebRTC permite que se compartan datos y se lleven a cabo videoconferencias de igual-a-igual (*peer-to-peer*), sin la obligatoriedad de que el usuario tenga que instalar complementos (*plug-ins*) o cualquier otro software de terceros. Asimismo, WebRTC contiene diversas API y protocolos interrelacionados que operan conjuntamente para lograr lo explicado anteriormente.

WebRTC es el protocolo utilizado para ver la transmisión de la red de cámaras. Aportamos más información en el apéndice A.

3.1.6. OpenLayers

OpenLayers[1] es una librería que nos permite elaborar mapas interactivo, tiene puede mostrar distintos tipos de mapas, datos vectoriales, marcadores, capas de todo tipo. etc. Esta librería cuenta con un gran rendimiento, contiene numerosas funcionalidades para cualquier cosa, además es totalmente gratuito y *OpenSource*.

OpenLayers es la librería que nos permite representar un mapa y poder interactuar con él, además, tiene funciones necesarias para este proyecto como la representación de marcadores y el uso de Pop-ups

3.1.7. Laravel

Laravel es un framework PHP [9], el cual cuenta con la mayor comunidad en el mundo de Internet, siendo uno de los más usados. Cabe destacar que Laravel aporta numerosas y potentes utilidades que hacen posible que los desarrolladores puedan acelerar el desarrollo de las aplicaciones. Además, Laravel es muy moderno como framework y hace hincapié en la calidad del código, la facilidad de mantenimiento y escalabilidad, permitiendo esto llevar a cabo proyectos, desde los más pequeños a los más grandes. Asimismo, fomenta las mejores prácticas y hace posible que el trabajo en equipo sea más fácil.

El framework Laravel maneja una arquitectura de carpetas avanzada, con lo que impulsa que los archivos se separen con un orden determinado y adecuado. Esto servirá de guía a todos los miembros del equipo de trabajo y será un estándar durante el transcurso de los diferentes proyectos. Claro está que también cuenta con una arquitectura de clases muy correcta, la permite la separación del código por responsabilidades. Su estilo arquitectónico es MVC[2].

Es el framework utilizado para la construcción del back-end de la aplicación ya que Angular es un framework centrado en el front-end.

3.1.8. Bootstrap

Bootstrap es un framework CSS [10] cuya función es estandarizar las herramientas de compañía, y fue desarrollado por Twitter en 2010. En sus inicios este framework se denominaba Twitter Blueprint y, en 2011, se convirtió en código abierto y su nombre pasó a ser Bootstrap. Fue actualizado varias veces desde ese entonces, por lo que ya se encuentra en la versión 4.4.

[1] https://openlayers.org/

[2] Modelo-vista-controlador es un patrón de arquitectura de software, que separa los datos y la lógica de una aplicación de su representación y del módulo encargado de gestionar los eventos y las comunicaciones.

La combinación que realiza Bootstrap con JavaScript y CSS permite estilizar los elementos de una página HTML, por lo que sus utilidades van mucho más allá de cambiar el color de los enlaces y los botones. Esta herramienta aporta interactividad en la página, por lo que proporciona un conjunto de componentes que favorecen la comunicación con el usuario, tales controles de página, como menús de navegación, barras de progreso y más.

Además de todos las características brindadas por Bootstrap, su objetivo más importante es hacer posible la construcción de sitios web responsive para dispositivos móviles, por lo que las páginas están preparadas para funcionar en smartphones, desktop y tablets, de una forma muy organizada y sencilla.

Boostrap ha sido de una gran ayuda ya que permite maquetar nuestra aplicación web de una forma mucho más rapida.

3.1.9. Angular

Angular [11] es un framework Javascript potente, óptimo para la creación de aplicaciones frontend modernas, de dificultad media o elevada. El tipo de aplicación Javascript desarrollado con Angular es del estilo SPA o también las denominadas PWA. El framework Angular proporciona una base para la producción de aplicaciones escalables, optimizadas y robustas. Asimismo, dicha base fomenta las buenas prácticas y un estilo de codificación de gran modularidad y codificación homogéneo.

Además de aportar una base para el desarrollo de la parte frontal, la programación Javascript del lado del cliente, también aborda técnicas de desarrollo de la parte del *back-end*, para la implementación del *Server Side Rendering*. A esta parte se le llama Angular Universal.

Angular es el framework de desarrollo utilizado en este proyecto ya que sus característica de desarrollo SPA se adapta perfectamente al proyecto.

3.2. Herramientas

3.2.1. Visual Studio Code

Visual Studio Code [12] es un editor de texto plano elaborado por Microsoft, que aporta a los clientes una herramienta de programación avanzada de forma alternativa al bloc de notas, por lo que es de código abierto y, además, gratuito.

Este editor está escrito completamente en Electron[3], un framework que se emplea para unir Chromium[4] y Node.js en forma de aplicación de escritorio. Resulta muy fácil de

[3]es el nombre de un framework de JavaScript preparado para crear aplicaciones de escritorio nativas y compatibles con Windows, Mac y Linux

[4]Chromium es una versión de código abierto de Google Chrome que se puede utilizar para crear distintos navegadores.

programar, flexible y muy potente. Asimismo, dispone de una gran cantidad de extensiones desarrolladas por la propia comunidad. Una de las características más destacable de este editor es IntelliSense, una función posibilita destacar la sintaxis de todo el código fuente a medida que se va escribiendo y, además, proporciona funciones como la de auto-completar, sustentándose en definiciones, variables y módulos.

Es el editor de texto utilizado por su ligereza, flexibilidad, su autocompletado y su gran cantidad de extensiones muy útiles para incrementar la velocidad de desarrollo.

3.2.2. Git

Se puede decir que un Git[13] consiste en un sistema de control de versiones con capacidad de rastrear cada cambio en un archivo, por lo que su propósito más importante es administrar todas las variaciones realizadas en un proyecto en un período de tiempo determinado. En definitiva, Git almacena estas modificaciones y la información asociada en una estructura de datos o repositorio. Este repositorio incorpora la confirmación de objetos con sus referencias, siendo un centro en el que los desarrolladores tienen la posibilidad de almacenar, probar, colaborar y compartir proyectos. Por tanto, Git tiene la capacidad de trabajar con un amplio abanico de proyectos de diversos tamaños, además de promover un flujo de trabajo fluido.

Git constituye un apoyo a la hora de trabajar en equipo, pues ofrece la posibilidad de llevar a cabo una supervisión del progreso y sirve de ayuda a los profesionales no tecnológicos y a los programadores a vigilar sus archivos.

Git ayuda en la colaboración en equipo, le permite realizar un seguimiento del progreso y ayuda a los programadores y a los profesionales no tecnológicos a supervisar sus archivos.

3.2.3. Overleaf

Overleaf [14] es un editor colaborativo de LaTeX basado en la nube que se utiliza para escribir, editar y publicar documentos científicos. Se asocia con una amplia gama de editoriales científicas para proporcionar plantillas oficiales de LaTeX para revistas y artículos científicos.

Overleaf fue lanzado originalmente en 2012 como WriteLaTeX por la compañía WriteLaTeX Limited, cofundada por John Hammersley y John Lees-Miller. Ambos son matemáticos y se inspiraron en sus propias experiencias académicas para crear una mejor solución para la escritura científica colaborativa.

Overleaf fue la herramienta utilizada para el desarrollo de la memoria de este proyecto.

Capítulo 4

Metodología de trabajo

En este trabajo se ha puesto en práctica la metodología de desarrollo ágil Scrum, complementada a su vez por técnicas de representación visual del flujo de trabajo denominadas Kanban. A pesar de ser un proyecto individual en cuanto a la implementación del código, se ha decidido seguir este modelo de ramificación para ponerlo en práctica y seguir un orden de versionado de la aplicación. A continuación, describiré estos conceptos detalladamente y cómo se han utilizado a lo largo del trabajo.

4.1. Scrum

Scrum [15] es un marco de trabajo con el que es posible utilizar un conjunto de diversas técnicas y procesos. Asimismo, consiste en un proceso de metodología ágil el cual se fundamenta en entregas parciales y regulares del producto final, primadas por el beneficio que otorgan a los usuarios. Scrum brinda valor y agilidad en cada nueva iteración, lo que lo convierte en una excelente elección para aquellos proyectos con entornos complejos en los que las condiciones pueden cambiar de manera constante o, en su defecto, no están bien definidos.

Scrum no se apoya en el seguimiento exhaustivo de un plan, sino que nos ajustamos y evolucionamos continuamente a las condiciones que vayan aconteciendo. Dicho de otra manera, debido a que partimos de la hipótesis de que trabajamos en entornos con exigencias que no son estables, por lo que es necesario flexibilidad y rapidez.

Las ventajas de Scrum están directamente relacionadas con los 12 principios del manifiesto ágil, de los cuales destacamos los siguientes:

- Al adecuarse bien a las dificultades funciona favorablemente en servicios o productos muy complicados.
- En vez de utilizar una gestión predictiva del producto, se prefiere una gestión evolutiva, en la que el objetivo no es anunciar lo que va a suceder, sino ajustarse a los cambios que van acaeciendo a lo largo del tiempo para desarrollar el producto.

- Se emplea un procedimiento de fases superpuestas en lugar de efectuar las fases de desarrollo de forma secuencial o en cascada.
- Aumenta el trabajo en equipo y la productividad.
- Aplicar una táctica de desarrollo incremental apoyándose de iteraciones o sprints.
- Otorgar más relevancia a la calidad del resultado fundamentado en el conocimiento implícito de las personas, en vez del conocimiento determinado por los procesos y la tecnología empleada.

El marco de trabajo Scrum está compuesto por los equipos Scrum, sus roles, eventos, artefactos y reglas asociadas. Cada componente dentro del marco de trabajo es útil para un determinado propósito, así como primordial para el éxito de Scrum y para su utilización.

4.1.1. Roles

4.1.1.1. El equipo de desarrollo

Se trata del conjunto de personas que aportan en cada incremento de valor añadido al producto. El equipo es responsable de la entrega del producto. Se aconseja un pequeño equipo que esté formado por un mínimo de tres personas y por un máximo de 9, con las habilidades transversales que se requieren para llevar a cabo el trabajo. En el cómputo del número de miembros del equipo de desarrollo no se tienen en cuenta ni el propietario del producto ni el Scrum Master.

El equipo de desarrollo está con compuesto por dos desarrolladores junior (el autor de este TFG y el estudiante Jesús Artiles Cidoncha que ha colaborado en otras funcionalidades de este producto en su TFG) y un desarrollador senior (Alejandro Cabrera Cárdenes) que nos ha apoyado y guiado en el desarrollo del producto.

4.1.1.2. El propietario del producto (Product Owner)

Es la persona que persigue que el producto adquiera el máximo valor posible para los clientes. Además, es quien se encarga de la gestión de la pila del producto (Product Backlog), la prioriza, elimina o agrega historias de usuarios, etc. También garantiza que el equipo de desarrollo trabaje de manera correcta desde la perspectiva del negocio.

El CEO de la empresa, Andrés Alonso Martirena, que ejerce de Product Owner y es la persona que ha ayudado a definir las funcionalidades desarrolladas acorde a su valor de negocio.

4.1.1.3. El Scrum Master

Es la persona que tiene la responsabilidad de quitar los inconvenientes que dificultan que el equipo logre el objetivo del sprint. No se trata de que el Scrum Master sea el líder del equipo, sino que hace de intermediario entre el equipo de desarrollo y el propietario del producto.

En este caso, el profesor Nelson Monzón López ha ejercido este rol guiándome y dirigiendo el desarrollo para alcanzar los objetivos propuestos.

4.1.2. Eventos

- **Sprint:** es un periodo de tiempo (time-box) entre dos semanas y un máximo de dos meses, en los cuales se lleva a cabo un incremento iterativo del producto, utilizable y potencialmente desplegable.

  Para este proyecto se planificó cuatro sprints de una duración aproximada de un mes

- **Planificación del sprint:** en esta reunión se procura escoger el plan que se seguirá para culminar el siguiente sprint. Su duración tiene un máximo de ocho horas para un Sprint de un mes; en cambio, para Sprints más cortos el evento es generalmente más breve. Para ello se eligen las historias de usuario que se necesiten para lograr el objetivo.

  Antes de comenzar cada unos de los sprint primero se dedico un tiempo en la planificación de este.

- **Revisión del sprint:** es una reunión en la cual se estudia el resultado que se ha alcanzado y el trabajo, tanto completado como no completado. Si es necesario, puede adaptarse la pila de producto. Dicha reunión tendrá una duración máxima de cuatro horas para Sprints de un mes.

- **Scrum diario:** se trata de una reunión en la que el equipo pone al día el esfuerzo de lo que queda por terminar y del trabajo ya terminado. La reunión tiene un bloque de tiempo de 15 minutos para el equipo de desarrollo y se realiza cada día.

  El equipo se desarrollo trabaja durante la semana dos días de forma remota y tres días (lunes, miércoles y viernes) de forma presencial, en estos días el Scrum Master realiza una reunión para conocer la situación actual y los objetivos del día.

- **Retrospectiva del sprint:** en esta reunión se verifican los puntos positivos y negativos del último sprint para mejorar, de esta manera, en próximas iteraciones. El propósito de la retrospectiva es desempeñar una mejora constante del proceso, contando con la participación de los miembros del equipo, quienes aportan sus impresiones sobre el sprint recién superado. Tiene una duración máxima de cuatro horas.

4.1.3. Artefactos

- **Pila del producto:** se trata de una lista metódica requerida por el usuario, que va variando y desarrollándose en función de la evolución del producto y el entorno en el que se usará. Además, varía de manera constante para distinguir lo que requiere el producto para ser útil, competitivo y adecuado
- **Pila del sprint:** se trata del conjunto de historias de usuario que el equipo debe elegir para el siguiente Sprint, con el objetivo de entregar el incremento del producto y conseguir el propósito del Sprint.
- **Incremento:** es la suma de todos los elementos de la Pila del Sprint concluidos a lo largo de una iteración dada y el valor de los incrementos de todos los Sprints anteriores. El incremento es un paso hacia una meta y tiene que estar en condiciones de usarse.

4.2. Kanban

Kanban [16] es un sistema de gestión visual cuyo propósito es manejar el flujo de trabajo en estados diferenciados. Con esto se logra identificar los diferentes ciclos por los que va pasando una historia de usuario. Además, dicha herramienta persigue prevenir los tiempos muertos y los cuellos de botella. En las figuras 4.1 y 4.2 se observa un ejemplo del tablero que se ha utilizado durante este TFG, en el cual se pueden diferenciar varios estados:

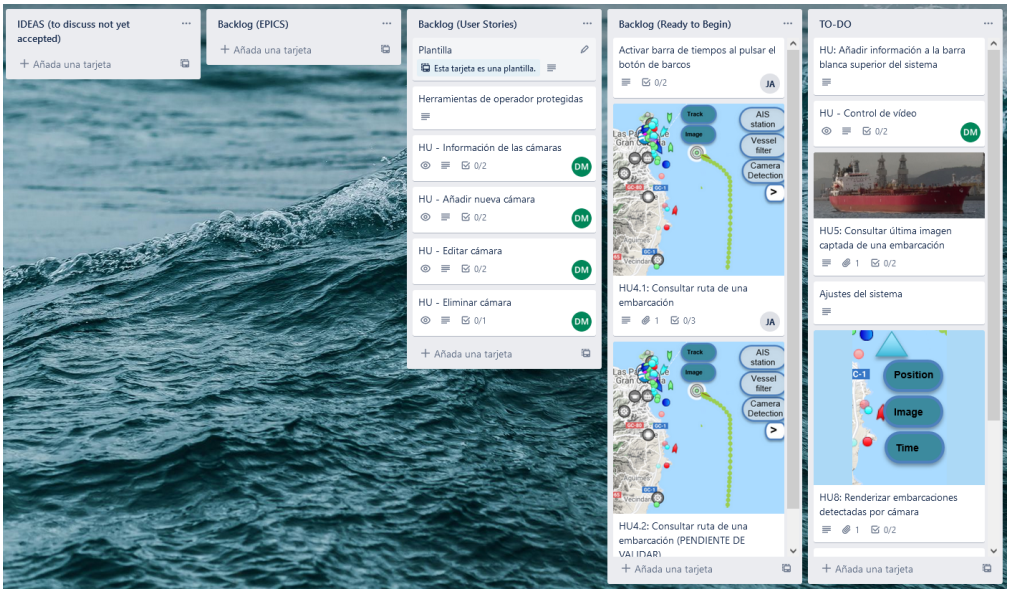


Ilustración 4.1: Captura del trello parte 1

- **Backlog (IDEAS):** describen posibles funcionalidades a desarrollar que aún no han sido validadas y que incluso quizás no lleguen nunca a desarrollarse.

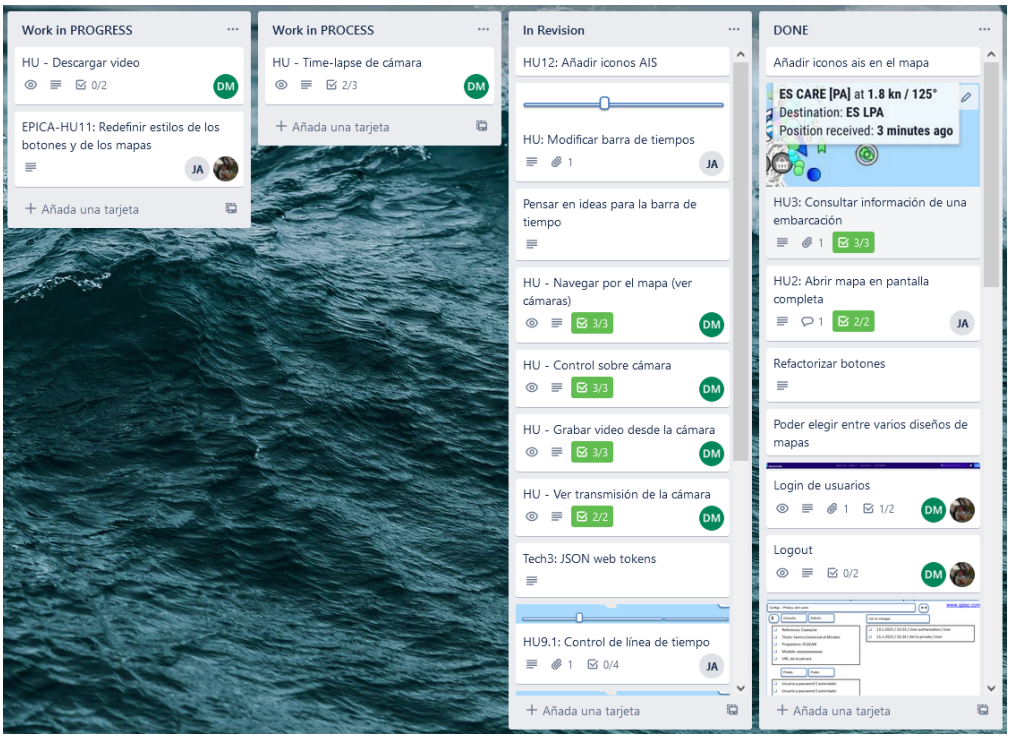


Ilustración 4.2: Captura del trello parte 2

- **Backlog (EPICS):** son grandes cantidades de trabajo que pueden desglosarse en un número de tareas más cortas.
- **Backlog (User Stories):** en este estado se encuentran las historias de usuario en evaluación. Aún se puede determinar si se incluirán al producto o no.
- **Backlog (Ready to Begin):** es el estado al que pasan las historias de usuario aprobadas que se incorporarán al producto. Estas historias ya están priorizadas y valoradas.
- **Work in PROGRESS:** se trata del estado en el que se mantienen las historias de usuario que se encuentran en desarrollo, sin haberse finalizado aún.
- **Work in PROCESS:** estado en el que perduran las historias de usuarios que están desarrollándose.
- **In Revision:** las historias alcanzan dicho estado para atravesar un ciclo de pruebas y valoraciones antes de manifestar que están terminadas.
- **DONE:** son las historias que ya se han finalizado y están validadas de manera adecuada.

Capítulo 5

Desarrollo del software

Tras haber estudiado las características de las aplicaciones nombradas en el capítulo 2 el objetivo de este TFG nos ha permitido fusionar las dos alternativas estudiadas con anterioridad, además de añadir nuevas funcionalidades que serán utilices para los clientes de este proyecto.

En primer lugar, se pretende construir una aplicación que permita ver sobre un mapa interactivos las distintas cámaras del sistema, para el diseño de este nos vamos a basar en la web de Windy, así como de su panel al abrir alguna de las cámaras.

Por otro lado, se quiere conseguir ver el *streaming* correspondiente de cada cámara tal como en CanariasLife pero además incluir nuevas funcionalidades como mover el enfoque de la cámara, ver el *time lapse*, grabar vídeos, etc.

Por último, se ha de diseñar un sistema para la administración de las cámaras como dar de alta o baja una cámara, editar la información como el nombre o ubicación, ver el listado de las cámaras, etc. Para esta función también se podrá mejorar el sistema de Windy para añadir nuevas cámaras y editarlas.

En resumen, lo que queremos ofrecer es que, mediante una aplicación web, el usuario pueda ver las cámaras ubicadas en un mapa interactivo, que pueda ver el *streaming* correspondiente, que pueda realizar ciertas funciones en la cámara y **todo esto desde una sola aplicación web**.

5.1. Análisis del documento de requisitos

Antes de empezar con el desarrollo del sprint 0, se dedicó tiempo a realizar la pila del producto, que consistió principalmente en extraer los requisitos necesarios para el proyecto del documento de requisitos.

En la fase inicial del proyecto, el CEO de la empresa QAISC, quien ejerce como propietario del producto, presentó un documento de requisitos como guía inicial para las especificaciones

que deben existir en la aplicación web. Así mismo, ofreció una plantilla del diseño de como quería que se mostrara la aplicación web. Este diseño lo podemos apreciar en la siguiente figura.

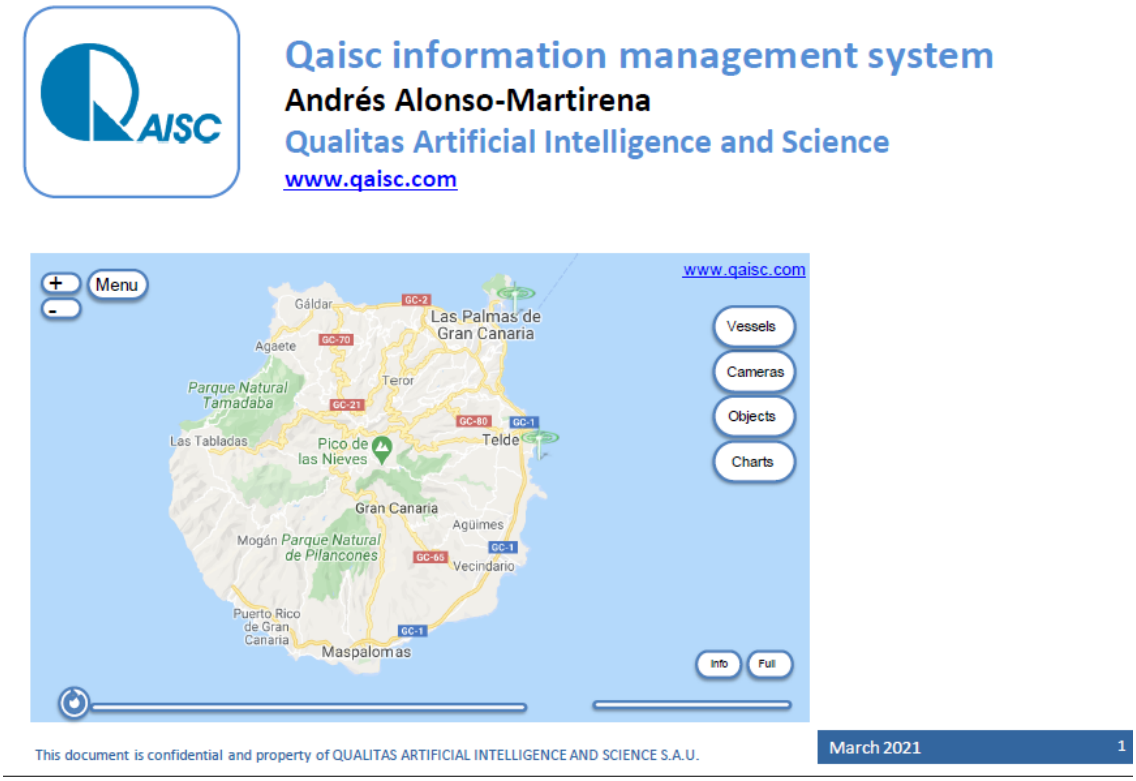


Ilustración 5.1: Maqueta provista por el Product Owner

Posteriormente, este trabajo se desglosó en las diferentes historias de usuario necesarias para desarrollar correctamente el proyecto en colaboración con los tutores del TFG. En este apartado, presentamos brevemente los principales requisitos obtenidos tras analizar el documento.

Primero que todo se espera que la página principal sea lo más sencilla posible, ofreciendo una interfaz basada en botones, en la izquierda pondremos los botones para hacer zoom in/zoom out y un menú que da entrada a los ajustes del sistema (idioma, sistema de hora, ajustes del sistema, logos, etc.); y la derecha pondremos un menú para elegir entre dos tecnologías de observación, las cámaras y el sistema AIS.

Al pulsar en el botón de cámaras, se mostrará sobre el mapa las cámaras, con un dos encima las térmicas, si están disponibles el símbolo será verde, si son privadas gris, si la información de más de una hora en color amarillo.

Al pulsar en una de las cámaras, aparecerá una escena asociada a la transmisión en tiempo real, además en la parte inferior se mostrará un serie de opciones (Config, Time-lapse, Gallery, Video, Follow, Compare, etc.)

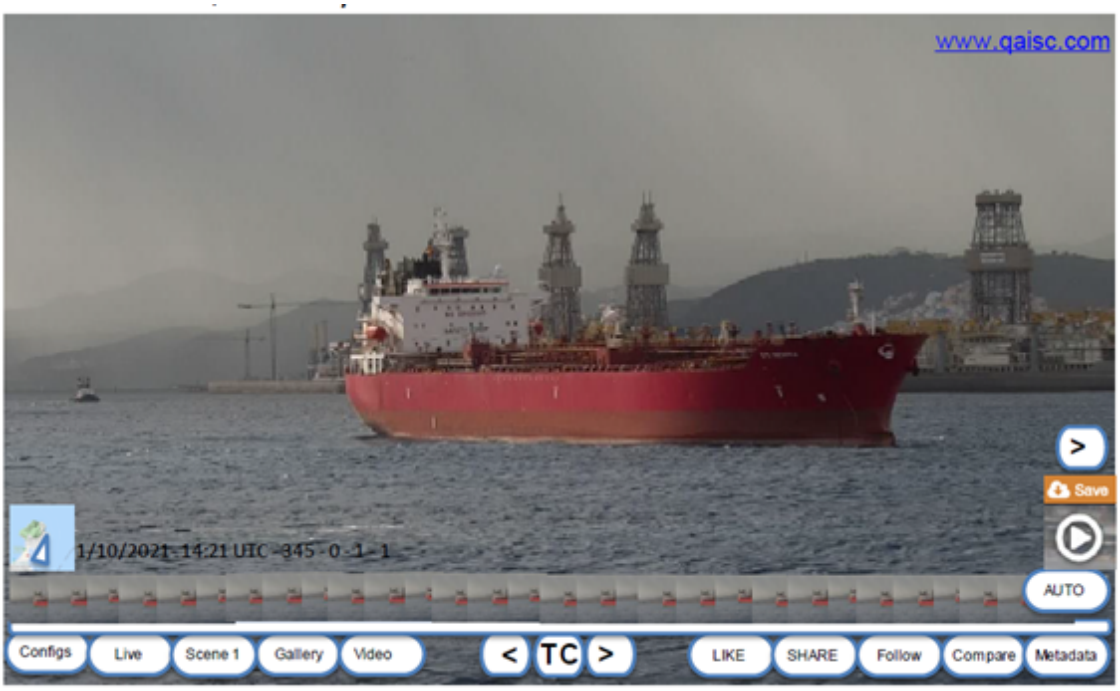


Ilustración 5.2: Maqueta para ver el *streaming* de una cámara

A continuación, vamos a listar las principales funcionalidades que se pedían para el siguiente proyecto.

- Al pulsar el botón Time lapse aparecerán las últimas 30 imágenes capturadas por el sistema.
- El botón Scene permite acceder a las direcciones de visión de la cámara, la preferida y las hasta nueve singulares
- En el botón Gallery está un carrusel con las cincuenta imágenes con like.
- El botón Video tiene asociada una galería de 10 videos
- Al pulsar el botón SHARE se abrirá una ventana con un menú para compartir la escena, por medios sociales.
- Al pulsar el botón Metadata superpone un pequeño mapa y la posición, dirección, elevación, fecha y hora de la cámara.
- La funcionalidad SAVE graba la escena en el dispositivo de consulta.
- Al elegir el botón Configs de una cámara aparecerá el menú de acceso a la configuración de la misma protegido mediante usuario y contraseña.
- Superado esto aparece el menú de identificación de cámara en el que se podrá consultar la información de la misma ó editarla.
- La consulta de una cámara que está en modo Private precisará usuario y contraseña en el momento de acceder a la misma.
- Una vez autenticado el usuario y contraseña para una cámara estará habilitado para consultar todas las cámaras asociadas al mismo
- La sesión caducará después de un tiempo predefinido.
- En modo MANUAL el visor se controlará con los botones derecha, izquierda, arriba, abajo, más zoom, menos zoom
- Existirá un grabador de vídeo con botones REC / STOP generará un vídeo asociado a la ID de la cámara y a la hora de inicio con una duración máxima de 10 minutos

A apartir de esto, se extrajeron las historias de usuarios que se van a desarrollar en el producto. Asimismo, incluí otras historias de usuarios que consideraba necesarias. Dichas historias de usuarios fueron supervisadas y revisadas por mis tutores del TFG, quienes realizaban el papel de Scrum Master.

La pila de producto mencionada anteriormente la podemos encontrar en el apéndice A.

5.2. Planificación

En este capítulo se describen las fases de la planificación de los sprints que se han seguido a lo largo del desarrollo del proyecto, el cual se ha dividido inicialmente en los siguientes sprints, que serán descritos con más detalle a lo largo del capítulo:

- Sprint 0: Familiarización con las herramientas y configuración del entorno de trabajo.
- Sprint 1: Integración de las cámaras con el mapa.
- Sprint 2: Funcionalidades de las cámaras.
- Sprint 3: Panel administrativo para la gestión de las cámaras.

5.3. Sprint Zero

5.3.1. Familiarización con las tecnologías

Para llevar a cabo este sprint se realizó un par de cursos con el objetivo de familiarizarme con las tecnologías y herramientas que vamos a utilizar a lo largo del proyecto.

Para ello, vamos a describir la ruta de aprendizaje seguida en Angular. Lo primero que se ha de aprender a usar es la consola de comando (CLI), la cual permite crear el sistema de ficheros para una aplicación, así como la creación y el manejo de los distintos esquemas. Los esquemas más utilizados son las clases, componentes, interfaces, módulos y los servicios.

El siguiente paso es entender cómo funciona un componente en Angular. Este se divide principalmente de tres archivos: el archivo HTML, que se encarga de darle cuerpo al componente; el archivo CSS, que le da los estilos al componente y el archivo TS, que se encarga de ejecutar la lógica del componente; y sirve para estructurar la aplicación y encapsular las funcionalidades de forma ordenada.

Otro punto importante son las Directivas que nos permite ejecutar cierta lógica para controlar los elementos del DOM. Las Directivas se dividen en dos tipos diferentes:

- **Directivas de atributo:** sirven para alterar la apariencia o comportamiento de un elemento del DOM y son usados como atributos de los elementos HTML utilizando para ello las directivas ngModel, ngClass, y ngStyle.
- **Directivas estructurales:** se aplican a los elementos HTML y actúan sobre el elemento padre y sus elementos hijos, esto les permite añadir, eliminar o manipular los elementos directamente en el DOM. Las más utilizadas son ngIf, ngFor, ngSwitch y ngTemplate

Una vez estudiadas las nociones básicas de Angular, vamos a aprender conceptos un poco más avanzados, como el uso de servicios para traer datos desde una base de datos. Los servicios en Angular son una forma de trabajar de manera síncrona con datos y llamadas, es

decir, si vamos a consumir un webservices nuestro usuario no tendría la necesidad de esperar como sucedía habitualmente, ya que ahora los servicios se encargan de traer los datos y los componentes para presentarlos de manera correcta.

Para comunicarnos con una base de datos vamos a utilizar Firestore Database, de Firebase, que nos proporciona una API para poder añadir, actualizar y/o eliminar registros de una colección. Además, existe otra manera de emplear una base de datos, la cual consiste en crear una aplicación back-end con NodeJS o algún framework como Laravel o Ruby On Rails para que actúe como API y poder atacar la base de datos desde nuestra aplicación de Angular. Una vez traídos los datos desde alguna API, tenemos la posibilidad de introducir el concepto de data binding, que se basa en la comunicación entre nuestro código HTML y la lógica de nuestro componente.

En este sentido, podemos distinguir cuatro tipos de databinding:

- **String interpolation:** que se utiliza para mostrar alguna propiedad de forma dinámica, consiguiéndose esto mediante el uso de una doble llave string.
- **Property binding:** que permite actualizar de forma dinámica alguna propiedad HTML, lo que podemos lograr recurriendo al empleo de corchetes [atrubuto HTML]=expresion;
- **Event binding:** que es la forma de comunicación que utilizamos cuando queremos reaccionar a algún evento provocado por el usuario. De esta manera, si el usuario hace clic en algún botón podemos vincularlo con un método. Para el uso de este databanding podemos seguir la siguiente sintaxis: (tipo evento)="método que se dispara".
- **Two-way binding:** que es una manera sencilla y breve de combinar los dos tipos de data binding que he explicado anteriormente (string interpolation y event binding). Con esto conseguimos que, por ejemplo, una string HTML se vaya actualizando cada vez que escribimos. La sintaxis para este tipo de binding es la siguiente: [(Directiva ngModel)]="propiedad a vincular"

Para culminar con este aprendizaje se realizó una to-do list, la cual me permitió poner en práctica todos estos conceptos. Además, esta aplicación también se desarrolló como un trabajo en la asignatura DAW2.

5.3.2. Configuración del entorno

Al trabajar en colaboración con la empresa QAISC tuvimos que tomar una serie de medidas para la configuración del entorno. La empresa cuenta un servidor principal y un servidor secundario, en el servidor principal tienen desplegadas varias aplicaciones como el back-end para la aplicación del proyecto, servidor de Gitlab, servidores de base de datos, etc. EL código de la aplicación se encuentre en un servidor de Gitlab propio de la empresa, esto se debe a que el código fuente de Gitlab es público y cualquiera puede copiarlo y montar su propio servidor privado.

Para poder trabajar, primeramente se desplegó una aplicación básica de Angular, es decir, solo contaba con el sistema de fichero que viene por defecto. Esta aplicación se encuentra

en un servidor principal de la empresa y para poder acceder a ella, me crearon un perfil de usuario con privilegios restringidos para prevenir que cometa cualquier imprudencia. Otra medida de seguridad fue crear una copia de la base de datos la cual podemos utilizar a nuestro antojo sin comprometer la base de datos original.

Para poder acceder a los recursos de la empresa (las aplicaciones, las bases de datos, servidor de Gitlab), me crearon un certificado para poder conectarme a través de VPN a la red de la empresa y poder trabajar de forma local desde mi casa.

Ahora para trabajar con el código de forma local, instalé un plugin de VS Code, Remote - SSH, el cual me permite conectarme mediante SSH al servidor de la empresa (siempre y cuando tenga activado la conexión VPN) y poder abrir el proyecto con VS Code. El siguiente paso para la configuración fue crear una rama en Git para trabajar de forma conjunta con el equipo, cada miembro del equipo trabaja en su propia rama y al terminar cada sprint se sube los cambios a la rama principal.

Una vez terminada la configuración del entorno ya puedo empezar a desarrollar el producto, para ello vamos a empezar con la planificación del sprint 1.

5.3.3. Planificación del sprint 1

Antes de comenzar con el primer sprint del proyecto se debe calcular el factor de foco. El factor de foco (FF) es el porcentaje del día en que las personas de un equipo de desarrollo son plenamente productivas y comprometidas al 100 % con la producción de Software, sin tener ningún tipo de interrupciones durante el día como pueden ser impedimentos, reuniones o tareas auxiliares. Se suele tomar como partida entre el 60 % y el 70 %, pero se volverá a calcular tras cada iteración.

Durante este sprint estuve trabajando 12 horas semanales, por lo que para el segundo sprint supuse que iba a trabajar el mismo número de horas a la semana, sumando un total de 48 horas ideales para un sprint de un mes. Si un punto de trabajo equivale a 4 horas ideales, a partir de esto podemos calcular la velocidad de trabajo ideal. Asumiendo que el factor de foco será del 70 %, la velocidad real será la velocidad ideal que, en este caso, serán:

48 horas / 4 puntos-horas = 12 puntos de trabajo.

Si ahora aplicamos el factor de foco, en total serán:

12 * 0,7 = 8,4 puntos de trabajo estimados para el sprint 1.

Esto significa que para el sprint 1 tenemos que meter en la pila del sprint un máximo de 8,4 puntos de historias.

Id	Nombre	Estimación		Prioridad
HU-16	Página de inicio	8 horas	2 puntos	10
HU-5	Navegar por el mapa	6 horas	1,5 puntos	9
HU-19	Abrir panel del *streaming*	4 horas	1 puntos	8
HU-17	Color de las cámaras	2 horas	0,5 puntos	6
HU-20	Ver *streaming* desde el mapa	8 horas	2 puntos	6
HU-21	Cabecera de la página principal	4 horas	1 puntos	5
Estimación total		**32 horas**	**8 puntos**	

Cuadro 5.1: Pila del Sprint 1

5.4. Sprint 1

Tomando como referencia la planificación de la pila del sprint que se realizó anteriormente, se comenzó a elaborar las historias de usuarios por orden de prioridad.

La función de la página de inicio consiste en mostrar el mapa, el cual ocupará la pantalla completa. Para lograr que la página de inicio muestre el mapa, lo primero que tenemos que hacer es instalar Openlayers en nuestro proyecto; y añadir posteriormente añadirlo en nuestro proyecto de la manera que se muestra en la figura 5.3

```css
.map-container {
  position: absolute;
  top: 0;
  left: 0;
  right: 0;
  bottom: 0;
}
#map {
  width: 100%;
  height: 100%;
}
```

(a) Estilos para el mapa

```
ngAfterViewInit(): void {
  this.layer = new TileLayer({
    source: new OSM(),
  });
  this.view = new OlView({
    center: Proj.fromLonLat([-15.6, 27.95]),
    zoom: 10,
    minZoom: 2,
    maxZoom: 14,
  });
  this.map = new OlMap({
    target: 'map',
    layers: [this.layer],
    view: this.view,
  });
}
```

(b) Crear objeto mapa

```
<div class="map-container">
  <div id="map"></div>
</div>
```

(c) Contenedor del mapa

Ilustración 5.3: Creación del mapa

Una vez integrado el mapa, el siguiente paso es colocar las cámaras sobre el mapa, para eso primero se tuvo que desarrollar una API y así poder traer los datos de las cámaras que se encuentran en la base de datos de la empresa, en formato JSON. Una vez obtenido el JSON solo hay que recorrer todo el *array* e ir creando una Feature que represente el icono de la cámara y ubicarlas en el punto que se especifique como se muestra en la figura 5.4.

```
getCamerasJson() {
  this.cameraService.getCameras().subscribe(
    (cameras: Camera[]) => {
      this.loadCameras(cameras);
    },
    (error) => console.log(error)
  );
}
```

(a) JSON de las cámaras

```
loadCameras(cameras: Camera[]) {
  cameras.forEach((camera: Camera) => {
    this.cameras.push(this.setCamera(camera));
  });
  this.addLayerCameras(this.cameras);
}
```

(b) Cargando cámaras desde el JSON

```
addLayerCameras(cameras: Camera[]) {
  cameras.forEach((camera) => {
    var iconFeature = new Feature({
      geometry: new Point(
        Proj.fromLonLat([camera.dd_longitude, camera.dd_latitude])
      ),
      name: 'streaming/' + camera.id,
    });
    iconFeature.setStyle(new Style({
      image: new Icon({
        src: 'assets/imgs/camera_on.png',
        scale: [0.045, 0.045],
      }),
    }));
    this.vectorLayer = new VectorLayer({
      source: new VectorSource({
        features: [iconFeature],
      })
    });
    this.map.addLayer(this.vectorLayer);
  });
}
```

(c) Añadir las cámaras al mapa

Ilustración 5.4: Añadir las cámaras al mapa

Cabe recalcar que, inicialmente, la llamada a la API se realizaba desde el propio componente. No obstante, posteriormente se creará un servicio para realizar este tipo de peticiones. Además, debe tenerse en cuenta que la mejor manera de utilizar los datos traídos desde la API es creando una interfaz que sirva como modelo de las cámaras.

Al principio, los datos traídos desde la API no eran de ningún tipo. Sin embargo, gracias al fuerte tipado que nos proporciona TS, podemos hacer que los datos sean del tipo Camera y así tener la capacidad de acceder y controlar sus atributos fácilmente, mediante el modelo Camera.

Para ver el *streaming* de un cámara queremos abrir un panel que muestre el *streaming* con un tamaño considerable. Para conseguir esto se tuvo que *bindear* cada cámara con un evento de click (figura 5.5) y conseguir, de esta manera, que al pulsar en una de las cámaras se abriera el panel del *streaming*, el cual por el momento no va a mostrar nada, solo queremos abrir el panel donde irá el *streaming* y los controles de la cámara.

```
this.map.on('click', (event: any) =>
  this.map.forEachFeatureAtPixel(
    event.pixel,
    (feature) => this.openCameraPanel(feature, event),
    { hitTolerance: 5 }
  )
);
```

Ilustración 5.5: Bindeando evento clic al pulsar una cámara

Una de las funcionalidades a desarrollar era ver el *streaming* desde el mapa, abriendo un Pop-up al pasar el ratón por encima de una de las cámaras. Si nuestro propósito es *bindear* un Pop-up a una Feature, tendremos que disparar un evento cada vez que se mueva el ratón, dado que así podremos saber si se ha posicionado encima de un Feature. Esto se consigue gracia al método **forEachFeatureAtPixel()** que nos devuelve la Feature que esta siendo seleccionada. Para el desarrollo de esta tarea se utilizó un objeto básico de Openlayers, Overlay, el cual no permite crear una capa por encima del mapa y es la que hará de base para cargar el *streaming*.

Por el momento, al abrir cualquiera de los Pop-ups se muestra un vídeo de prueba (figura 5.7. Es en el siguiente sprint cuando conseguimos visualizar el *streaming* podemos factorizar el Pop-up y cargar, de esta manera, el *streaming* correspondiente.

```
hoverCamera(event: any) {
  var hover = this.map.forEachFeatureAtPixel(
    event.pixel,
    (feature) => {
      this.selectedFeature = feature;
      return true;
    },
    { hitTolerance: 5 }
  );
  hover ? this.openCameraPopup(event) : this.closePopup();
}
```

(a) JSON de las cámaras

```
openCameraPopup(event: any) {
  this.content = document.getElementById('popup-content');
  if (this.overlay.getPosition() == undefined) {
    this.content.innerHTML =
      '<video autoplay src="assets/imgs/video.mp4" width="300"></video>';
    this.overlay.setPosition(event.coordinate);
  }
}
```

(b) Cargando cámaras desde el JSON

Ilustración 5.6: Creando Pop-up para el *streaming*

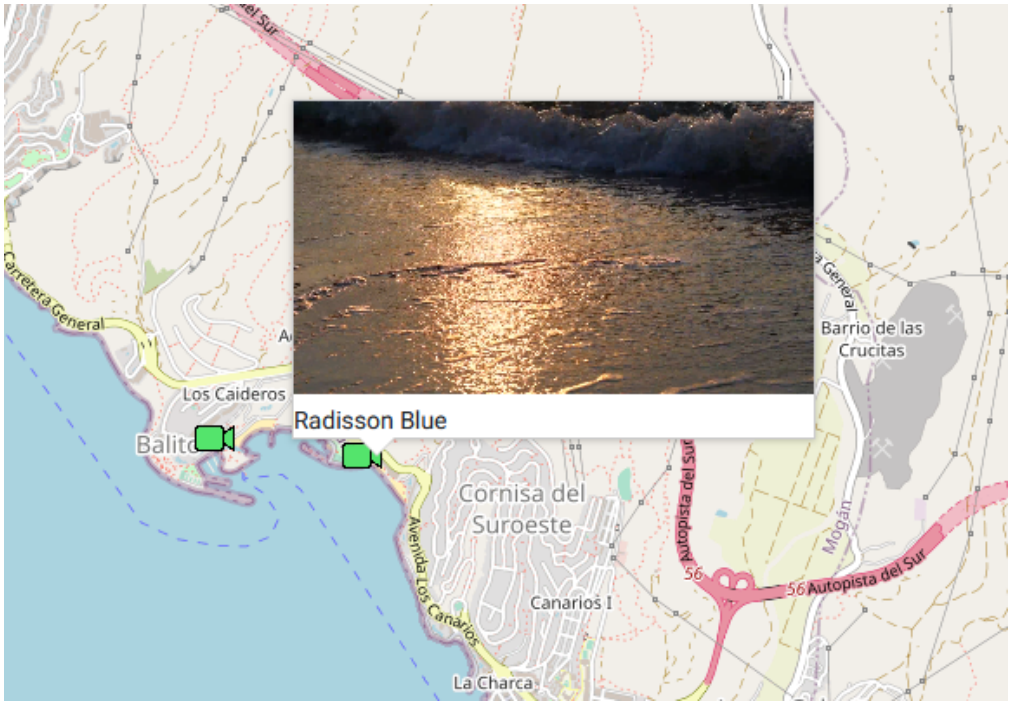


Ilustración 5.7: Pop-up cámara de prueba

5.4.1. Planificación del sprint 2

Al terminar el sprint aún quedaba una historia de usuario por terminar, la **HU-21: Cabecera de la página principal**, por lo que no se ha logrado completar todos los puntos que se pretendían entregar en este sprint. Esto quiere decir que la capacidad estaba sobre estimada y, debido a esto, se debe volver a calcular la velocidad para el siguiente sprint. Como la velocidad real fue de 6,5 puntos, ahora el nuevo factor de foco será:

7 (velocidad real) / 8,4 (velocidad ideal), lo que nos da un factor de foco del 78,30 %.

Para el segundo sprint tuvimos que crear la nueva pila del sprint en base al nuevo factor de foco. En este sprint se estimó que se realizaría un trabajo de 10 horas semanales para un sprint de dos meses (mayo y junio), ya que mayo es el último mes en el que se imparten clases y en el que hay que entregar todos los proyectos, trabajos, exámenes, etc. por lo que ese mes no iba a poder dedicarle mucho tiempo al TFG. Esto nos da un total de 80 horas ideales para el sprint, tomando como referencia que un punto de trabajo equivale a 4 horas ideales. A partir de esto podemos calcular la velocidad de trabajo ideal, asumiendo que el factor de foco será del 78,30 %, por lo que la velocidad real será la velocidad ideal que, en este caso, será:

80 horas / 4 puntos-horas = 20 puntos de trabajo.

Aplicando el factor de foco, en total serán:

20 * 0,783 = 15,66 puntos de trabajo estimados para el sprint 2.

Lo que significa que para dicho sprint tenemos que incluir en la pila del sprint un máximo de 15,66 puntos de historias.

5.5. Sprint 2

Se decidió meter las siguientes historias de usuarios en la pila del sprint, según su prioridad:

Id	**Nombre**	**Estimación**		**Prioridad**
HU-8	Ver transmisión del *streaming*	16 horas	4 puntos	10
HU-4	Controles sobre cámara	12 horas	3 puntos	9
HU-9	Time-lapse del *streaming*	16 horas	4 puntos	8
HU-18	Almacenamiento en OpenStack Swift	8 horas	2 puntos	8
HU-7	Grabar vídeo	4 horas	1 puntos	6
HU-10	Reproducir vídeo	4 horas	1 puntos	5
HU-14	Descargar vídeo	2 horas	0,5 puntos	5
Estimación total		**62 horas**	**15,5 puntos**	

Cuadro 5.2: Pila del Sprint 2

Para este sprint, la tarea más difícil y que más tiempo requirió fue poder ver la transmisión en directo de las cámaras. Para realizar esta tarea se tuvo que investigar bastante acerca del protocolo WebRTC, el cual me gustaría explicar con más detenimiento en el apéndice A.

Para poder visualizar la transmisión en directo de una de las cámaras se utilizó un Script JS el cual podíamos utilizar de la siguiente manera (figura 5.8):

```
window.onload = function () {
  var webRtcServer = null;
  webRtcServer = new WebRtcStreamer("video", 'http://192.168.22.51:8000/');
  webRtcServer.connect('rtsp://vps-smartcoast-radisson.qaisc.local:5540/channel-1');
}

window.onbeforeunload = function () {
  webRtcServer.disconnect();
};
```

Ilustración 5.8: Script para establecer conexión WebRTC

Una vez establecida la conexión WebRCT, el resultado obtenido se muestra en la figura 5.9:

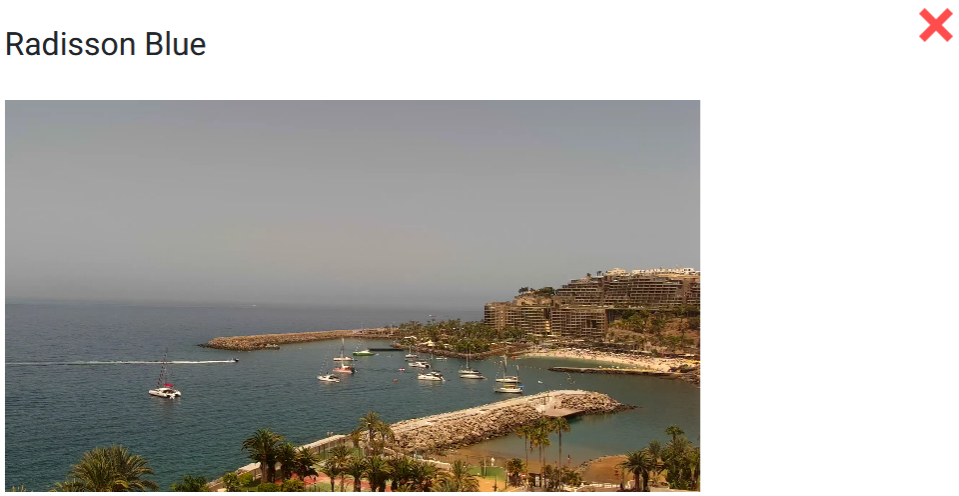


Ilustración 5.9: Visualizar el *streaming* de una cámara

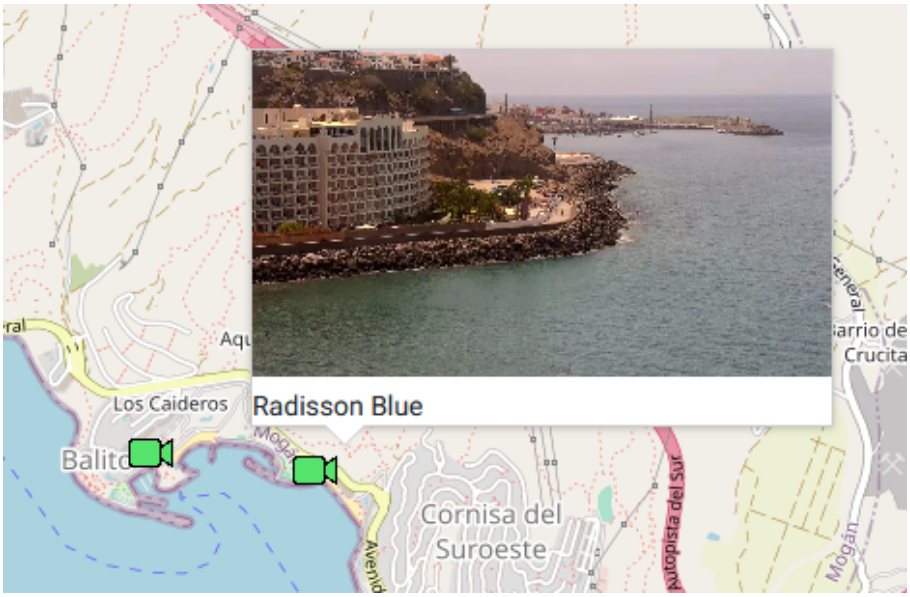
Una vez logrado el objetivo de poder visualizar el *streaming* de una de las cámaras, el siguiente paso consistió en poder cargar el *streaming* de cada cámara de forma dinámica, es decir, que cuando pulsemos en una de las cámaras se cargue en el panel el *streaming* correspondiente. Por tanto, se tuvo que cambiar el Script JS y refactorizarlo a una función a la que se le pasa la URL del *streaming* y la URL del canal RTSP.

```
var webRtcServer = null;

function connectWebRtc(url, video_url) {
    webRtcServer = new WebRtcStreamer("video", url);
    webRtcServer.connect(video_url);
}

function disconnectWebRtc() {
  webRtcServer.disconnect();
}
```

Ilustración 5.10: Script WebRTC refactorizado

Ahora que ya visualizamos el *streaming* correspondiente con cada cámara podemos dar por concluida la **HU-20: Ver *streaming* desde el mapa**, pues al abrir el Pop-up de cada cámara se muestra su respectivo *streaming* (figura 5.11).

(a) *streaming* desde Anfi del Mar



(b) streaming desde Radisson Blu

Ilustración 5.11: Visualizando el streaming correspondiente

El siguiente paso consistió en crear una interfaz para poder mover el enfoque de las cámaras PTZ. Para ello, primero se indagó en cuál sería la mejor forma de conseguirlo y se logró encontrar una librería denominada nippleJS que permitía utilizar un joystick virtual.

NippleJS cuenta con tres tipos de joysticks:

- **Joysticks dynamic:** que dan la posibilidad de crear un joystick en cualquier punto dentro de un cierto parámetro. El joystick desaparece visualmente al dejar de presionarlo y, además, permite tener varios a la vez.
- **Joysticks static:** que poseen una posición fija, ubicados en el centro del parámetro que lo contiene. Sin embargo, también podemos especificar en donde deseamos ubicarlo.
- **Joystick semi:** que actúa como los dos anteriores ya que primero se puede crear el joystick en cualquier posición y una vez creado actúa como el semi. No obstante, si pulsamos más allá de una cierta distancia se destruye el anterior joystick, volviéndose a originar un joystick en la nueva posición.

En este caso, el que más nos interesa para la página web sería el **joystick static**, pues lo que queremos es tener un joystick fijo en cierta posición.

Ilustración 5.12: Utilizando el joystick de nippleJS

Una vez se ha implementado el joystick en nuestro panel (figura 5.12), hay que relacionar los eventos y los datos que nos proporciona la librería con las funciones que queremos desarrollar, que son: mover hacia arriba, abajo, derecha e izquierda. Para ello vamos a utilizar el ángulo, la dirección X e Y, y la fuerza que nos puede servir para calcular la velocidad a la que vamos a mover la cámara (figura 5.13).

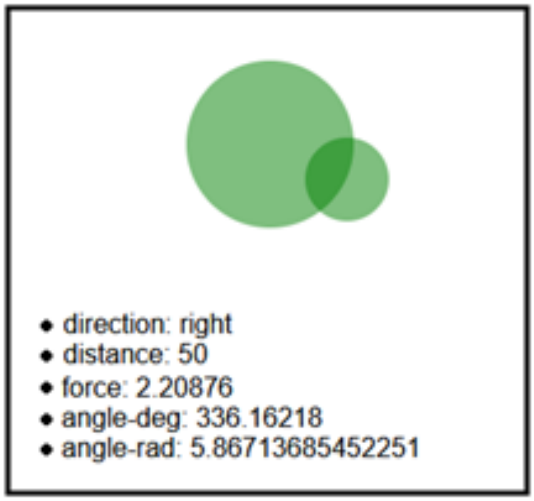


Ilustración 5.13: Datos proporcionados por el joystick

Ahora, podemos utilizar el estado de la variable *direction*, para que la cámara se mueva en dichas direcciones atacando a una API desarrollada por otro miembro del equipo. Una vez comprobado que funcionaba correctamente, se decidió que también se podían añadir movimientos diagonales, es decir, que la cámara se moviera al sureste, suroeste, noreste, y noroeste.

Por este motivo, se tuvo que empezar a utilizar el ángulo en el que se encontraba el joystick, además de segmentar el círculo para saber que ángulo correspondía con cada dirección. Esto lo podemos apreciar en la figura 5.14:

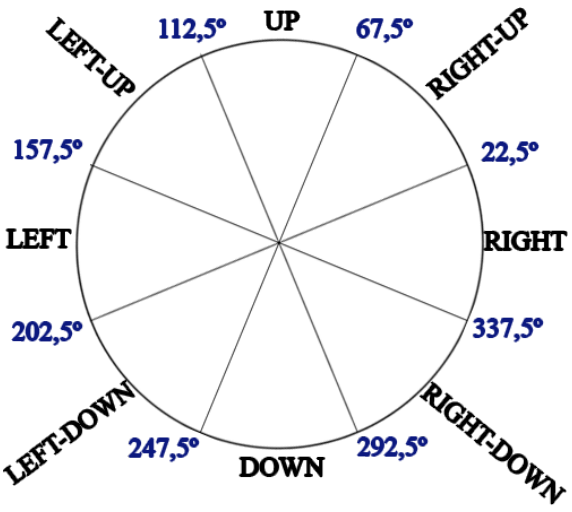


Ilustración 5.14: Segmentación de un círculo

A continuación, se debía relacionar los ángulos obtenidos con la dirección a la que se tenía que mover el enfoque de la cámara. Cabe destacar que se agregó una variable para que no se enviaran múltiples peticiones cuando se mantuviera el enfoque de la cámara en una dirección, esto es, que por ejemplo mantengamos el movimiento a la izquierda y solo se envíe una petición de mover la cámara a la izquierda, por ejemplo.

```
public sendData() {
  if (this.interactingStatic) {
    if ((this.angle >= 22.5 && this.angle < 67.5) && this.previousDirection != 'right-up') {
      this.previousDirection = 'right-up'
      this.moveCamera('continuous_move', this.v, this.v, 0);
    } else if ((this.angle >= 67.5 && this.angle < 112.5) && this.previousDirection != 'up') {
      this.previousDirection = 'up'
      this.stopCamera('stop_move')
      this.moveCamera('continuous_move', 0, this.v, 0);
    } else if ((this.angle >= 112.5 && this.angle < 157.5) && this.previousDirection != 'left-up') {
      this.previousDirection = 'left-up'
      this.moveCamera('continuous_move', -this.v, this.v, 0);
    } else if ((this.angle >= 157.5 && this.angle < 202.5) && this.previousDirection != 'left') {
      this.previousDirection = 'left'
      this.stopCamera('stop_move')
      this.moveCamera('continuous_move', -this.v, 0, 0);
    } else if ((this.angle >= 202.5 && this.angle < 247.5) && this.previousDirection != 'left-down') {
      this.previousDirection = 'left-down'
      this.moveCamera('continuous_move', -this.v, -this.v, 0);
    } else if ((this.angle >= 247.5 && this.angle < 292.5) && this.previousDirection != 'down') {
      this.previousDirection = 'down'
      this.stopCamera('stop_move')
      this.moveCamera('continuous_move', 0, -this.v, 0);
    } else if ((this.angle >= 292.5 && this.angle < 337.5) && this.previousDirection != 'right-down') {
      this.previousDirection = 'right-down'
      this.moveCamera('continuous_move' this: this his.v, 0);
    } else if ((this.angle >= 337.5 || this.angle < 22.5) && this.previousDirection != 'right') {
      this.previousDirection = 'right'
      this.stopCamera('stop_move')
      this.moveCamera('continuous_move', this.v, 0, 0);
    }
  }
}
```

Ilustración 5.15: Función para mover la cámara

Una vez se terminó de elaborar el sistema del movimiento de la cámara, nos percatamos de un fallo y es que, al cambiar de una dirección diagonal a una dirección horizontal o vertical, la cámara seguía moviéndose diagonalmente. Se intento buscar alguna solución leyendo la documentación del protocolo ONVIF, que es el que se emplea en la red de cámaras. A pesar de esto, la única forma de corregir este error fue parar la cámara antes de cambiar de una dirección diagonal a una dirección horizontal o vertical. Esto originó otro problema y es que, al parar el movimiento de la cámara cada vez que cambiamos el enfoque a una nueva dirección, resulta molesto a la hora de trabajar con el joystick.

Tras buscar durante unos días cómo se podría solucionar el problema, finalmente se optó por cambiar el sistema del joystick por un sistema basado en botones. Para este fin utilizamos un sistema de grid de CSS y se fue colocando cada botón en su sitio, podemos observar el resultado obtenido en la figura 5.16.

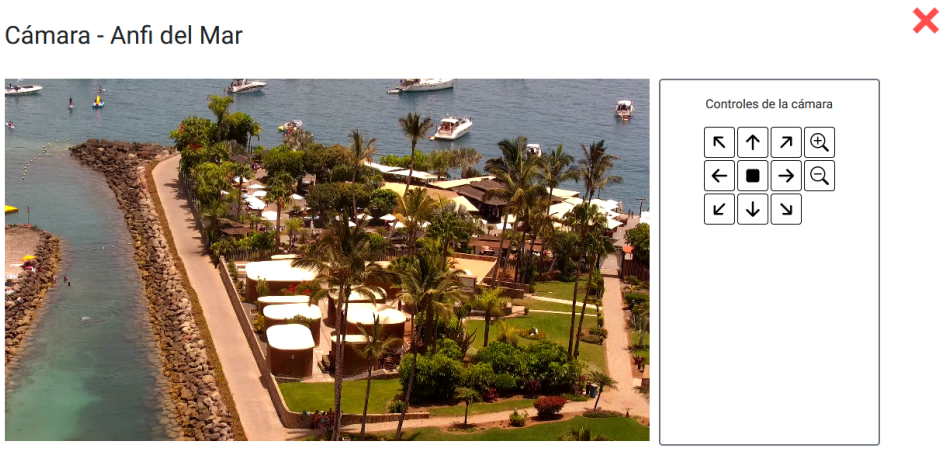


Ilustración 5.16: Sistema de movimiento basado en botones

Para darle un diseño llamativo a los botones se decidió utilizar una imagen SVG ya que esto me permite cambiar el fondo al botón cada vez que pasaba el ratón por encima.

Posteriormente, se desarrolló un *time lapse*, que consistía en tomar capturas del *streaming* cada cierto tiempo. A tal efecto, primero diseñamos el *time lapse* según las especificaciones del documento del Product Owner (figura 5.17).

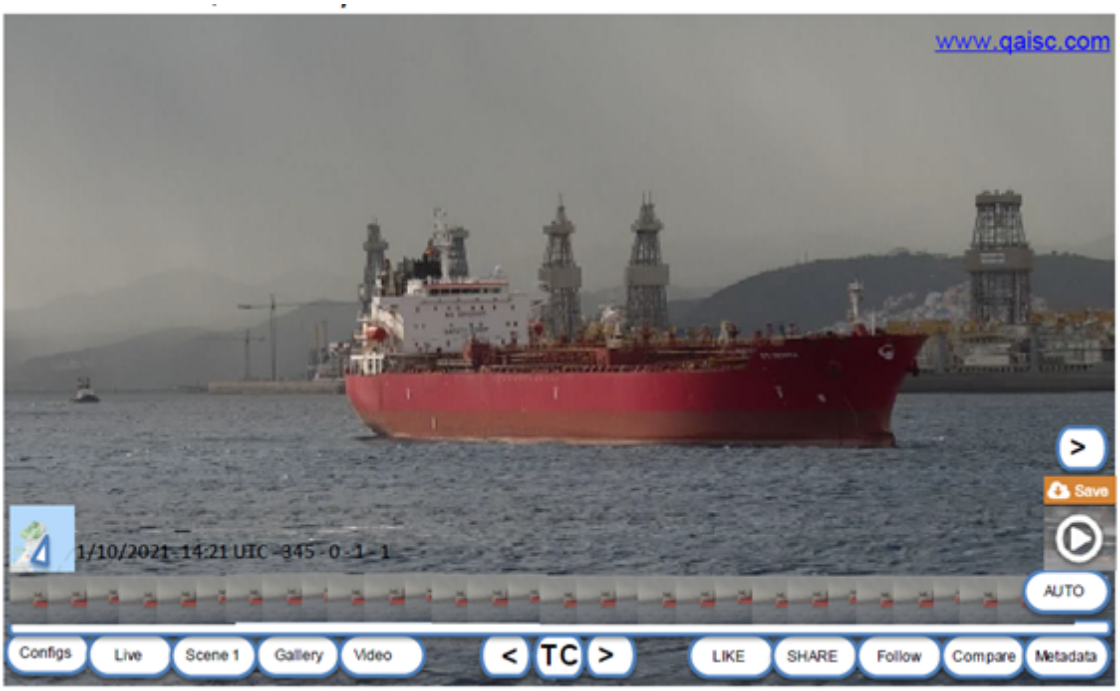


Ilustración 5.17: Requisito del Time-lapse

Para realizar esta funcionalidad se investigó como podía capturar una imagen desde un vídeo y se encontró este ejemplo [17].

Dicho ejemplo consiste básicamente en utilizar la etiqueta de HTML, canvas, y su método **getContext() y drawImage()**, el cual me permite dibujar una imagen pasándole como parámetro una fuente, como puede ser una imagen o un vídeo, y las coordenadas donde se quiere poner la imagen. A partir de esto, el siguiente paso fue desarrollar una función (figura 5.18) que se disparará cada cierto intervalo de tiempo, lo que se consigue mediante la función *setInterval()* de JavaScript. En ese momento interesaba tener sólo 10 capturas a la vez, para lo que se ha tenido que borrar la primera captura y añadir la nueva una vez alcanzada la capacidad máxima.

Tras realizar dicha función el resultado obtenido se aprecia en la figura 5.19.

```
public drawTimeLapse() {
  var myNodelist = document.querySelectorAll('#time-lapse-image');
  var parent = document.getElementById('time-lapse-images');
  if (myNodelist.length >= 10) {
    var child = document.getElementById('time-lapse-image');
    child?.remove();
  }
  var canvas = document.createElement('canvas');
  var ctx = canvas.getContext('2d');
  var video = <HTMLVideoElement>document.getElementById('video');
  var videoWidth = video.width;
  canvas.setAttribute('width', video.videoWidth.toString());
  canvas.setAttribute('height', video.videoHeight.toString());
  ctx?.drawImage(video, 0, 0, video.videoWidth, video.videoHeight);
  var data = canvas.toDataURL('image/jpeg', 1); //jpeg
  var img = document.createElement('img');
  img.setAttribute('src', data);
  img.setAttribute('id', 'time-lapse-image');
  img.setAttribute('width', '10%');
  parent?.appendChild(img);
}
```

Ilustración 5.18: Función para realizar el time lapse

Cámara - Anfi del Mar

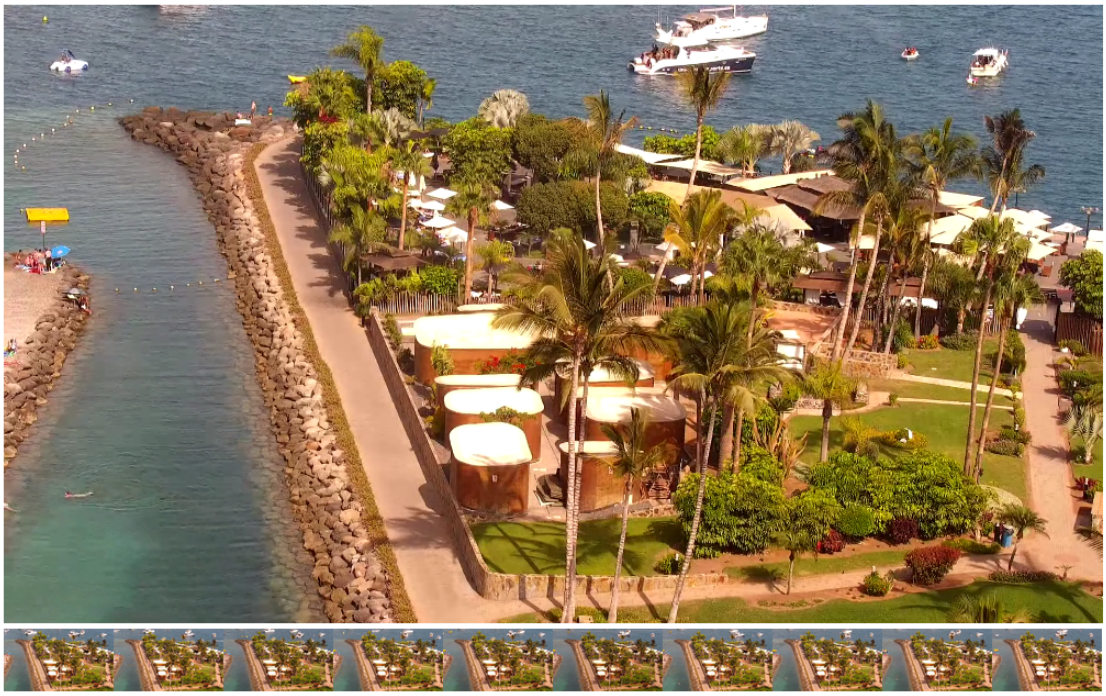


Ilustración 5.19: Primera versión del Time-lapse

Sin embargo, este método tenía un problema, ¿Qué sucedería si se quisiera tener más de 10 capturas y, a su vez, poder ver capturas más antiguas? se tendrían que hacer las imágenes aún más pequeñas, y ya con 10 capturas a la vez resulta un tanto complicado distinguir las diferencias entre cada una de ellas. Para solucionar esta problemática se realizó una función que al pulsar una imagen, esta adquiriera un tamaño lo suficientemente considerable como para poder distinguir bien las diferencias entre cada captura.

No obstante, esta solución no terminaba de ser factible, así que se decidió usar un carrusel de imágenes (figura 5.20) que mostrará por defecto una determinada cantidad de imágenes y, en el caso de querer visualizar más, se pudieran avanzar o retroceder las imágenes. De esta manera se podía decidir si quería ver el *time-lapse* de uno, de dos o de siete días. Además, cada imagen tenía asociada la fecha en la que la captura fue tomada.

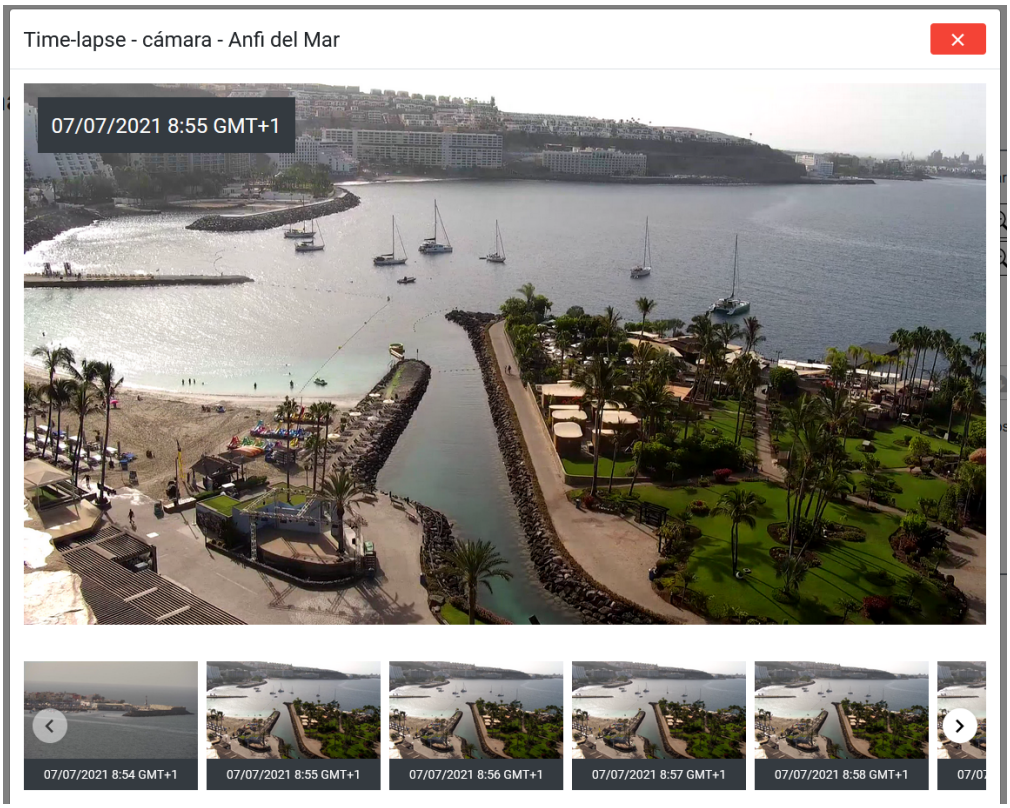


Ilustración 5.20: Segunda versión del Time-lapse

Para que la lógica del carrusel funcione bien, se necesita traer las imágenes desde el *back-end* para lo que se requiere, en primer lugar, almacenar las imágenes en la nube. Para conseguir esto vamos a utilizar OpenStack Swift, que no ofrece alta seguridad y confiabilidad de los datos, además de una gran escalabilidad de volumen de almacenamiento y potencia. En este caso, un miembro del equipo de desarrollo ya había montado el servidor de OpenStack Swift y había elaborado una API para comunicarse con al servidor. Finalmente, se incluyó un método para atacar a la API cada vez que se hiciera una captura del *streaming* para almacenar esa imagen en la nube.

5.5.1. Planificación del Sprint 3

Al finalizar este sprint todavía quedaron tres historias de usuario por terminar: la HU-7, HU-10 y HU-14, por lo que no se logró completar todos los puntos que se pretendían entregar en este sprint, lo que quiere decir que la capacidad estaba sobre estimada. Por lo tanto, se debe volver a calcular la velocidad para el siguiente sprint. Como la velocidad real fue de 13 puntos, ahora el nuevo factor de foco será:

13 (velocidad real) / 15,5 (velocidad ideal), lo que nos da un factor de foco del 83,87 %.

Para el segundo sprint debemos crear la nueva pila del sprint en base al nuevo factor de foco. En este sprint se estimó un trabajo de 16 horas semanales para un mes (julio) y se decidió que incrementara las horas de trabajo considerablemente ya que no tenía clases, por lo que le podía dedicar más tiempo al TFG. Esto nos da un total de 64 horas ideales para el sprint, tomando como referencia que un punto de trabajo equivale a 4 horas ideales.

A partir de esto podemos calcular la velocidad de trabajo ideal, asumiendo que el factor de foco será del 83,87 %. Por tanto, la velocidad real será la velocidad ideal que, en este caso, será: 64 horas / 4 puntos-horas = 16 puntos de trabajo. Si ahora aplicamos el factor de foco, en total serán 16 * 0,8387 = 13,42 puntos de trabajo estimados para el sprint 3.

5.6. Sprint 3

Como en el anterior sprint quedaron por realizas las siguientes historias de usuarios: HU-7, HU-10 y HU-14, las cuales están muy relacionadas entre sí ya que consisten en grabar, reproducir y descargar un vídeo. Por esta razón se les dió la mayor prioridad.

Se decidió meter las siguientes historias de usuarios en la pila del sprint, según su prioridad:

Id	Nombre	Estimación		Prioridad
HU-7	Grabar vídeo	4 horas	1 puntos	10
HU-10	Reproducir vídeo	4 horas	1 puntos	10
HU-14	Descargar vídeo	2 horas	0,5 puntos	10
HU-8	Cabecera página principal	8 horas	2 puntos	9
HU-1	Login usuario	6 horas	1,5 puntos	8
HU-2	Logout	0,5 horas	0.25 puntos	8
HU-3	Registro de usuarios	4 horas	1 puntos	8
HU-11	Información de las cámaras	8 horas	3 puntos	7
HU-12	Añadir nueva cámara	6 horas	1 puntos	6
HU-13	Editar cámara	6 horas	1 puntos	6
HU-15	Eliminar cámara	2 horas	0,5 puntos	6
Estimación total		**50,5 horas**	**12,625 puntos**	

Cuadro 5.3: Pila del Sprint 3

Para la realización de estas funcionalidades se crearon tres métodos (figura 5.21), el primer método que sirve para grabar un vídeo hace uso el objeto **MedriaRecorder**, al cual se le pasa como parámetro un *stream* y las opciones como el *mimeType* y el *bitsPerSecond*. Al terminar la grabación se almacena en una variable los datos en formato binario.

Si queremos reproducir este vídeo tenemos que utilizar objeto **Blob**, que sirve para representar una sucesión de datos binarios con el fin de convertirlos en una imagen, vídeo o *stream*. Además, vamos a utilizar el método **createObjectURL()** convertir los datos obtenidos en una URL que pueda ser interpretado como vídeo.

Si queremos descargar el vídeo el proceso será parecido, pero esta vez vamos a crear un elemento tipo *address* al que le asignamos la URL y el nombre del archivo que se va a descargar.

```
startRecording() {
  var video: any = <HTMLVideoElement>document.getElementById('video');
    var stream = video.captureStream();
  try {
    this.mediaRecorder = new MediaRecorder(stream, {
      mimeType: 'video/webm',
    });
  } catch (e) {
    console.error('Exception while creating MediaRecorder:', e);
    return;
  }
  this.mediaRecorder.ondataavailable = (event: any) => {
    if (event.data && event.data.size > 0) {
      this.recordedBlobs.push(event.data);
    }
  };
  this.mediaRecorder.start();
  this.isRecording = true;
  this.isFinishRecord = false;
}
```

(a) Grabación de un vídeo

```
downloadRecording() {
  this.isFinishRecord = false;
  const blob = new Blob(this.recordedBlobs, { type: 'video/mp4' });
  const url = window.URL.createObjectURL(blob);
  const a = document.createElement('a');
  a.style.display = 'none';
  a.href = url;
  a.download =
    this.currentCamera.name + '_' + new Date().toISOString() + '.mp4';
  document.body.appendChild(a);
  a.click();
  setTimeout(() => {
    document.body.removeChild(a);
    window.URL.revokeObjectURL(url);
  }, 100);
}
```

(b) Descargar el vídeo grabado

```
playRecording() {
  const videoBuffer = new Blob(this.recordedBlobs, { type: 'video/mp4' });
  this.downloadUrl = window.URL.createObjectURL(videoBuffer);
  this.recorded.nativeElement.src = this.downloadUrl;
  this.recorded.nativeElement.play();
  this.recorded.nativeElement.controls = true;
}
```

(c) Reproducir el vídeo grabado

Ilustración 5.21: Sistema para grabar, reproducir y descargar un vídeo

Ahora necesitamos una cabecera para poner los botones que permitan iniciar/cerrar sesión, un botón para los ajustes del sistema, un botón para mostrar el panel administrativo de las cámaras (este panel sólo será accesible para los administradores) y en un futuro, cuando se vayan desarrollando más funcionalidades, se podrá crear un botón para acceder fácilmente desde ahí. De esta manera el usuario podrá acceder fácilmente a estas funcionalidades. Para crear la cabecera de la aplicación web primero se creó un componente nuevo que hiciera la función de NavBar y para el diseño de este, sé utilizó el componente ToolBar de Angular Material. El resultado obtenido lo podemos observar en la figura 5.22.

Ilustración 5.22: Vista de la página principal usando un NavBar

A partir de esto se pueden vincular los botones mencionados anteriormente con las funciones a desarrollar en el registro, login y logout de usuarios. Para el registro de usuario se utilizaron los componentes de Angular Material, ya que estos componentes tienen un acabado bastante atractivo y elegante. Además, Angular ofrece una serie de directivas que se pueden usar para validar los formularios, como el uso de los Reactive Forms. Para esta tarea primero diseñamos la vista del formulario, como se puede apreciar en la figura 5.23 .

Una vez diseñada la vista del formulario, lo siguiente fue hacer la validación del formulario (figura 5.24) utilizando los módulos de **FormBuilder y Validators**. El registro de los usuarios en el sistema consiste en asignarles un token que les permita utilizar las funciones de la aplicación. Al iniciar sesión se comprueba que las credenciales (nombre y contraseña) son correctas y se devuelve el token, el cual se va a almacenar en el **localStorage** del navegador. Al cerrar sesión se eliminará este token del **localStorage**.

El token se utiliza para fortificar la API y para que sólo los usuarios que tengan un token válido puedan atacar a la API. Además, en un futuro se podrán diseñar políticas de seguridad de los tokens como el tiempo de validez, poder revocar los tokens, distinguir funcionalidades

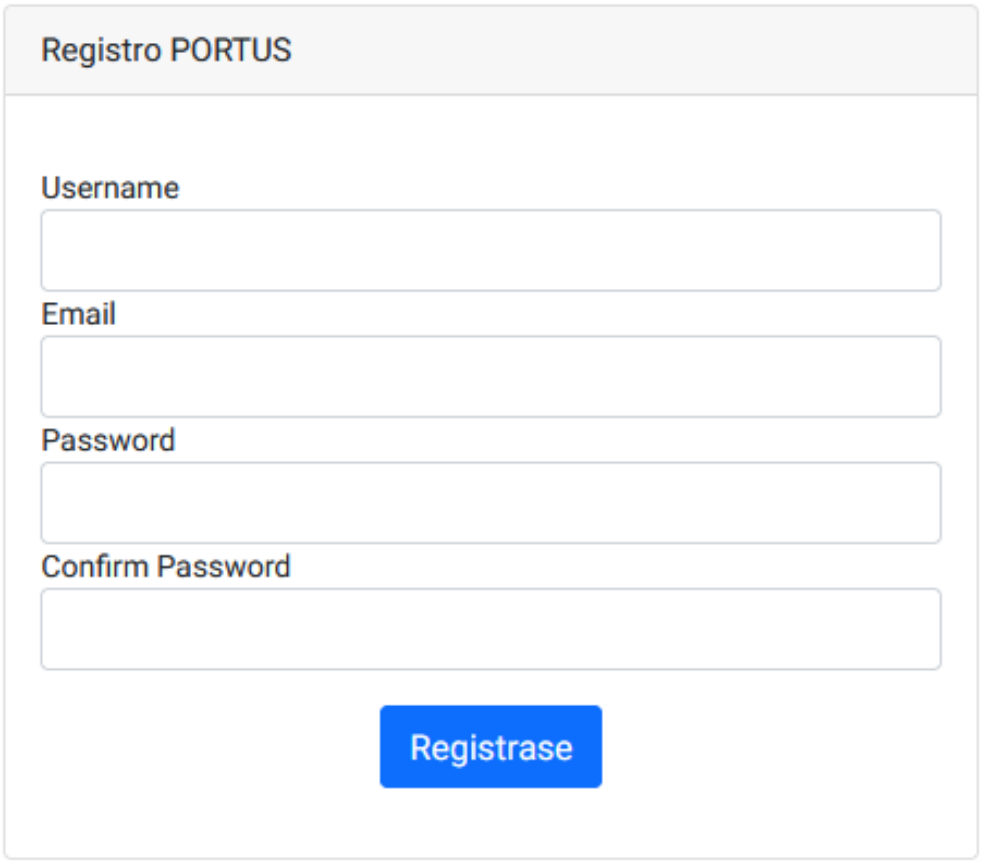


Ilustración 5.23: Vista del formulario para registrarse.

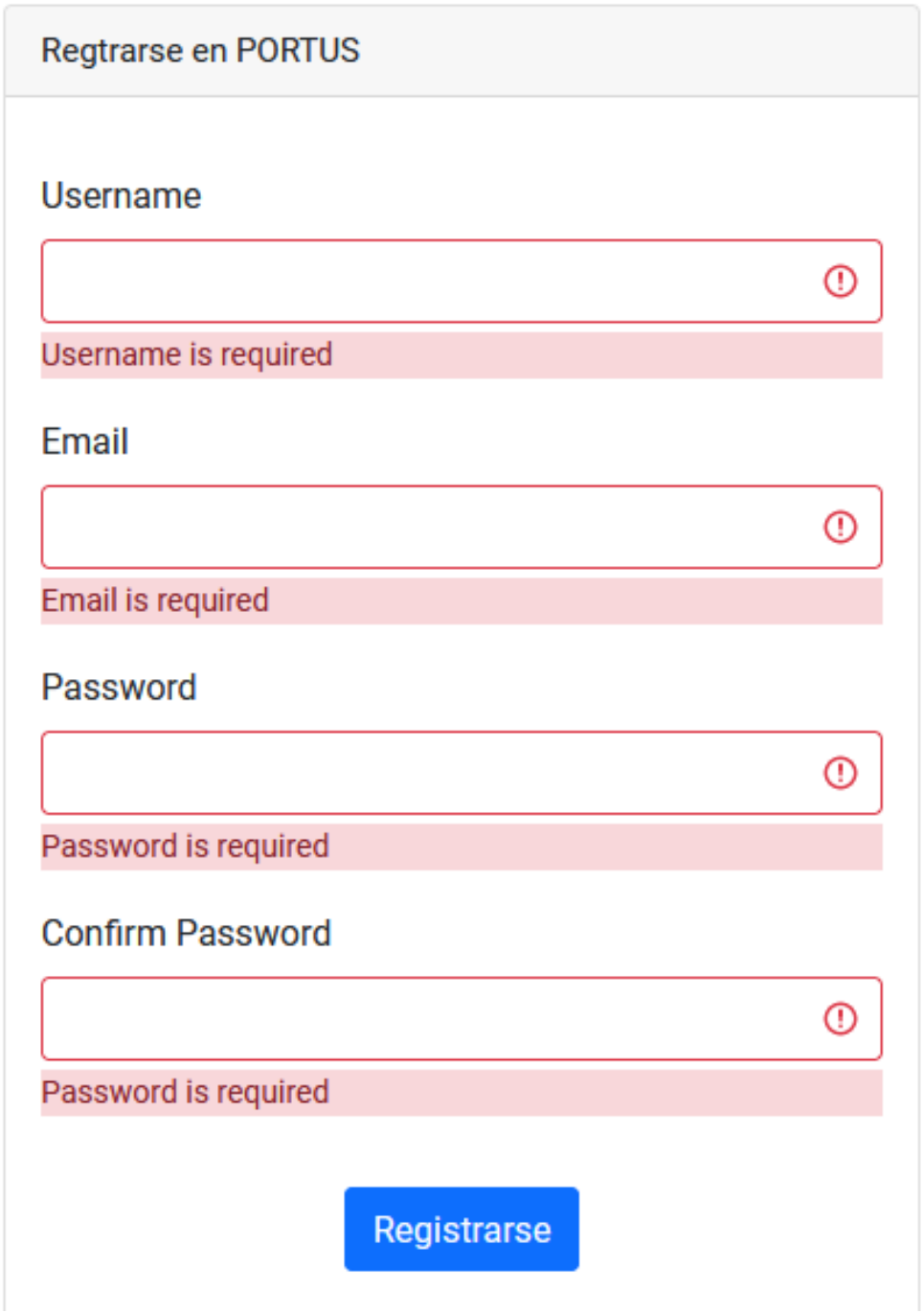


Ilustración 5.24: Vista del formulario con validación.

según el token provisto, etc.

Para poder ver una lista de las cámaras disponibles en el sistema, se decidió utilizar el componente **Table** de Angular Material (figura 5.25), ya que este nos provee de una serie de funciones bastantes útiles como: poder filtrar por cualquier campo de la tabla, poder ordenar los campos de mayor a menor alfabéticamente. Asimismo, cuenta con un paginador cuya utilidad es evitar que la tabla crezca infinitamente.

Ilustración 5.25: Vista del listado de cámaras registrados en el sistema

Posteriormente se añadió en la tabla unos botones para poder agregar, editar y eliminar una cámara, además de un botón para ver el *streaming* correspondiente a cada cámara. Para añadir y editar un cámara se creó un formulario (figura 5.26) con las clases de Bootstrap, lo que nos ayuda a ahorrar tiempo además de permitir crear formularios que se adapten al tamaño de la pantalla. A continuación, se relacionó el modelo Camera con los valores de los *inputs* mediante la directiva **[(ngModel)]**. También se desarrolló una validación (figura 5.27), como en el formulario de registrar usuarios. Por último, se llevó a cabo una función para parte del *back-end* con el fin de poder guardar o actualizar la información en la base de datos.

Para finalizar con el sprint se completó la última funcionalidad, la cual consistía en poder eliminar las cámaras, para ello se creó un método en la API que permitía realizar dicha acción. Una vez se realizo dicho método de la API lo único que se tiene que hacer es atacar a ese *endpoints* desde nuestra aplicación.

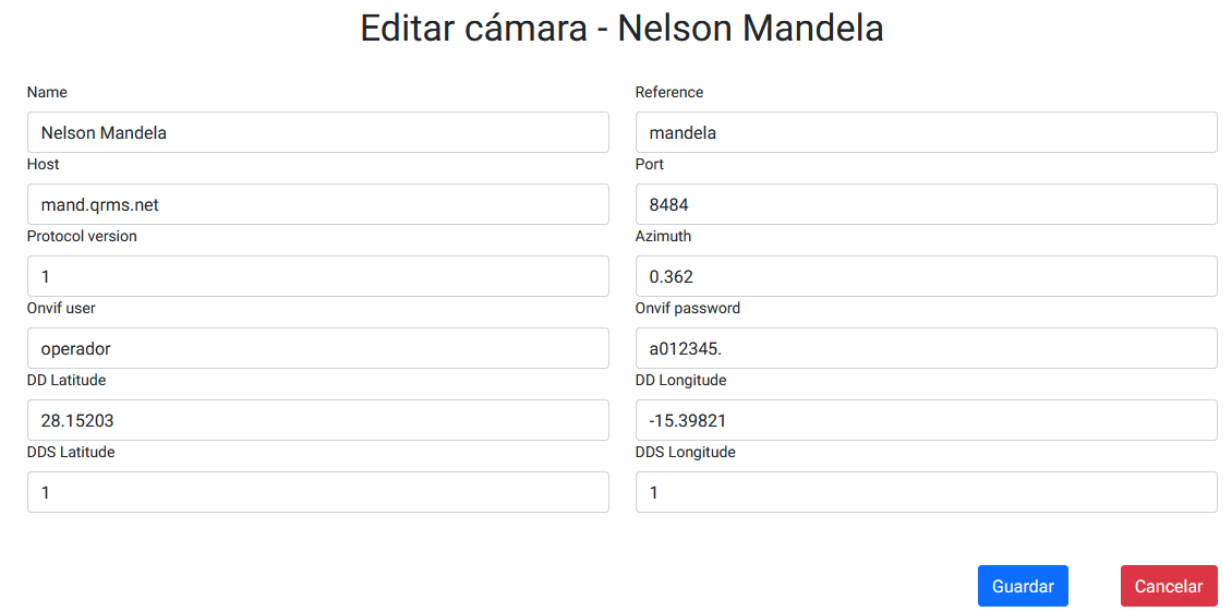


Ilustración 5.26: Formulario para añadir o editar una cámara

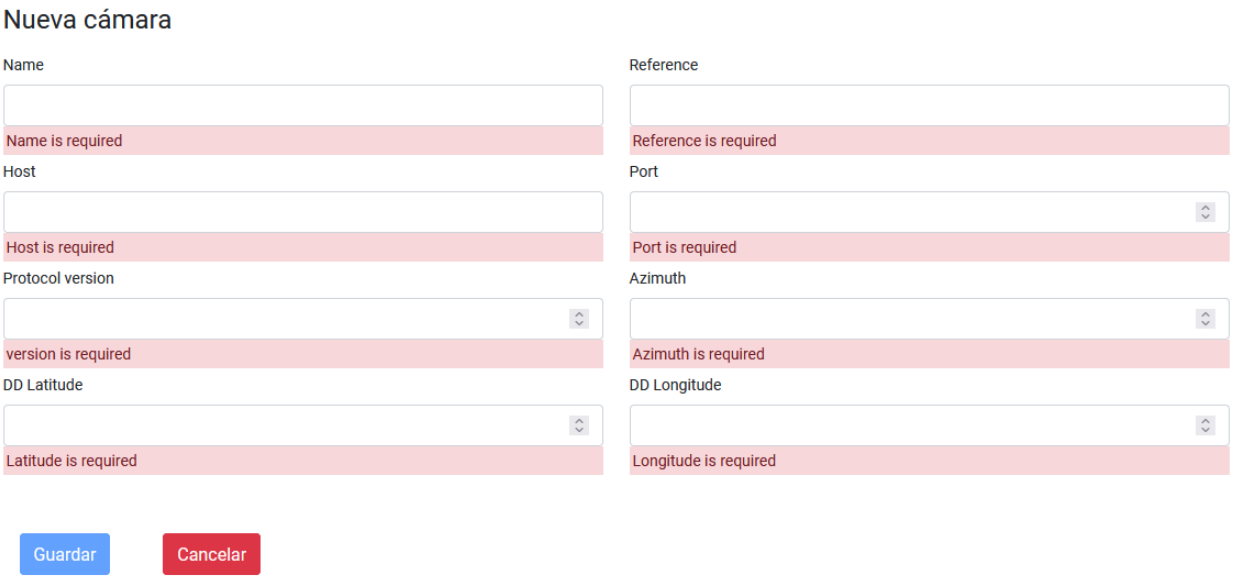


Ilustración 5.27: Formulario con validación para añadir o editar una cámara

Capítulo 6

Conclusiones y trabajo futuro

6.1. Conclusiones

En el desarrollo de este proyecto, se ha realizado una aplicación que ofrece mediante un mapa interactivo la ubicación de una red de cámaras ubicadas en zonas costeras de Gran Canaria. Además, podemos visualizar la transmisión en directo de cada una de las cámaras, así como ofrecer una serie de funcionalidades que cumple los principales requisitos que se pedían.

Para ello, el primer objetivo consistió en desglosar en historias de usuarios los requisitos especificados por el CEO de la empresa y trasladar esa visión a un producto que ha ido mejorando y añadiendo funcionalidades a lo largo de las distintas iteraciones.

Por otro lado, destacamos que la tecnología WebRTC es uno de los sistemas más importantes en el campo de las comunicaciones en tiempo real, tiene múltiples ventajas en comparación con los sistemas actuales que realizan las mismas funciones, pero la principal ventaja es que no requiere ningún software adicional ni complementos, lo que transmite confianza a los usuarios ya que solamente debes acceder al navegador web para utilizar el servicio.

Este proyecto me ha dado la oportunidad de trabajar en un ecosistema de trabajo real, siendo consciente de la importancia de la relación con todos los miembros del equipo durante las distintas fases de desarrollo siguiendo las metodologías de desarrollo ágil.

La metodología utilizada ha hecho que el desarrollo haya sido llevado con éxito y se haya logrado alcanzar los objetivos propuestos, al ser un proyecto conjunto es muy importante planificar bien lo que se quiere hacer en cada sprint, adaptarse bien a los continuos cambios del sistema, saber gestionar el tiempo para compaginarlo con las demás asignaturas y buscar las herramientas adecuadas que ayuden a cumplir los objetivos.

6.2. Trabajo futuro

Aunque se ha logrado alcanzar un resultado final que cumple con las expectativas planteadas en e proyecto, se ha dejado al sistema listo para futuras mejoras o implementaciones. En este apartado, voy a plantear las propuestas con las que me gustaría continuar desarrollando este proyecto.

- Poder comparar dos imágenes al mismo tiempo con alguna opción para hacer zoom.
- Optimizar la carga de las imágenes del time-lapse.
- Implementar un sistema de roles de usuarios.
- Poder avanzar, retroceder, pausar y reproducir el streaming como un falso directo.

Apéndice A

Protocolo WebRTC

A.1. ¿Qué es WebRTC?

El protocolo WebRTC (*Real Time Communication*) es una tecnología que posibilita que los sitios web y las aplicaciones puedan capturar y retransmitir audio y/o vídeo, así como realizar el intercambio de datos arbitrarios entre navegadores sin que sea imprescindible un intermediario. El conjunto de estándares que comprende WebRTC permite que se compartan datos y se lleven a cabo videoconferencias de igual-a-igual (*peer-to-peer*), sin la obligatoriedad de que el usuario tenga que instalar complementos (*plug-ins*) o cualquier otro software de terceros. Asimismo, WebRTC contiene diversas API y protocolos interrelacionados que operan conjuntamente para lograr lo explicado anteriormente.

Después de años de desarrollo, se considera que ha alcanzado el soporte y la madurez suficientes para convertirlo en un estándar. Desde su lanzamiento, no ha cambiado mucho, porque tal y como se lanzó ya funcionaba adecuadamente. Al principio, Google abogó firmemente por el estándar WebRTC, y el 26 de enero de 2021 el consorcio internacional W3C lo recomendó como estándar [18].

En los editores del estándar se encuentran Cisco, Google y Mozilla. Los dos primeros navegadores con soporte WebRTC fueron Google Chrome y Firefox. Posteriormente también se incorporó Edge cuando comenzó a estar implementado por el núcleo de Google Chrome. Cabe destacar que el navegador Edge había realizado un intento de implementación de WebRTC, pero debido a este cambio originó que se descartara por completo dicho desarrollo. El último en implementar WebRTC ha sido Safari, que se incorporó recientemente por el soporte del codec VP8.

Hoy día se puede utilizar WebRTC en cualquier navegador moderno, o cualquier otro navegador que este basado en Chromium. Un resumen de esto lo podemos apreciar en la figura A.1.

Ilustración A.1: Navegadores con soporte WebRTC

A.2. Usos del protocolo WebRTC

Una latencia reducida es excelente para la emisión de audio y vídeo en tiempo real, para compartir el audio del ordenador, compartir pantalla, etc. Al WebRTC también se le dan otros usos como puede ser la transferencia directa. Un ejemplo de este caso es la transferencia de ficheros desde mi propio navegador a otro navegador directamente, aunque en vista de que está basada en una técnica de *peer-to-peer* ya no hablamos del prototipo cliente-servidor en el cual los datos están centralizados en un servidor, sino que ambos clientes adquieren el mismo grado de importancia.

La tecnología WebRTC ha ganado una increíble popularidad en estos tiempos de pandemia en virtud de la posibilidad de tener aplicaciones de videoconferencia en nuestro navegador sin la necesidad de tener que instalar clientes nativos. Dicha instalación ha supuesto una problemática desde el momento en el que se cuenta con numerosos sistemas operativos y dispositivos para acceder a la red. Cuando más del 90 % de los sistemas eran Windows esta problemática no era tan perceptible, pero ahora que nos podemos conectar a Internet a través del móvil con Android o iOS, o desde el ordenador con Linux, Windows o Mac el instalar elementos nativos ha constituido un obstáculo, pues lógicamente si por cada plataforma nos tuviéramos que instalar un cliente nativo sería un proceso realmente dificultoso y trabajoso.

Usualmente empleamos aplicaciones de videoconferencia como Microsoft Teams o Google Meet, entre muchas otras como: Jitsi Meet, BigBlueButton, Webex, Around, Whereby, GoToMeeting. Todas estas aplicaciones usan WebRTC, sin embargo, algunos de estos recursos han tenido herramienta nativa la cual se instalaba por un tiempo, pero actualmente todas ellas se han acogido al protocolo WebRTC. Cabe destacar que Zoom no utiliza WebRTC, sino que emplea una tecnología propietaria con capacidad para gestionar audio y vídeo, intercambiándose estos a través de WebSockets.

Otro detalle importante a resaltar es el *Streaming*, protocolo basado también en el *peer-to-peer*. De manera habitual, el concepto *Streaming* que hace referencia a la emisión en directo de un vídeo como los que se pueden visualizar en Twitch[1] para una importante cantidad de usuarios. El *Streaming* en Internet se ha encontrado disponible desde mucho antes de la aparición del WebRTC, antes de esto, lo usual es usar técnicas como HLS, esta tecnología se utilizada para transmitir contenido en vivo y bajo demanda mediante el protocolo HTTP. HLS adapta el tamaño del contenido en función del ancho de banda del cliente, esto lo podemos entender fácilmente observando la figura[2] A.2. No obstante, con HLS se consigue una latencia de cinco a decenas se segundos. Hoy en día existen productos que posibilitan hacer *Streaming* de ultrabaja latencia empleando WebRTC como el caso de Milicast.

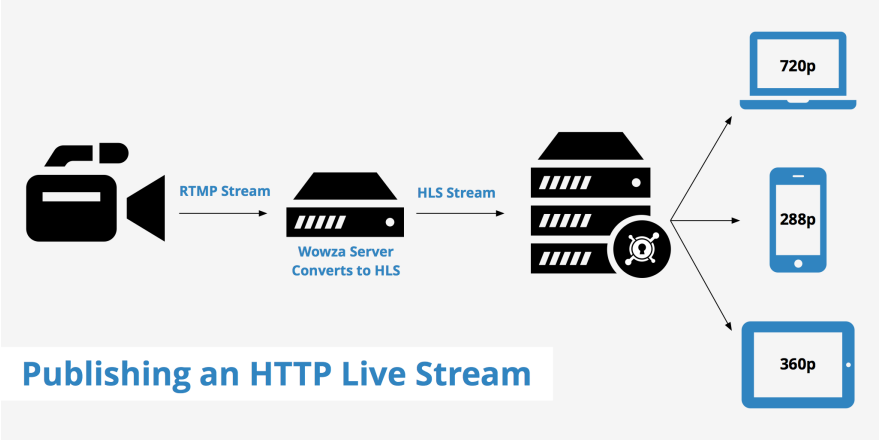


Ilustración A.2: Publicando un Http Live Streaming

A.3. ¿Cómo funciona el protocolo?

Una de las principales ventajas que ofrece es que permite comunicar un navegador directamente con otro, *peer-to-peer*, sin necesidad de que los datos deban pasar previamente por el servidor. Esto quiere decir que el servidor se utiliza únicamente para intercambiar información de la IP, intercambiar información sobre cuáles son los datos de las personas que van a conectarse, etc. Esto supone un cambio de paradigma relevante debido a que hasta ahora los navegadores entraban en contacto con un servidor; y para comunicarnos con otro usuario que tuviera su propio navegador había que pasar siempre mediante el servidor, como ocurre por ejemplo con WebSockets, que también puede implementarse para enviar vídeo y audio, aunque solo está diseñada para enviar información. Entonces una forma inmejorable de obtener intercambio de audio, vídeo y datos un poco más pesados o que necesiten una alta calidad de transmisión podría ser implementado por WebRTC. La necesidad de conectar directamente un navegador con otro nace de la necesidad de intentar rebajar la latencia lo

[1]Twitch es una plataforma que permite realizar transmisiones en vivo, la cual es propiedad de Amazon, Inc., esta plataforma tiene como función principal la retransmisión de videojuegos en directo.

[2]Imagen disponible en https://www.keycdn.com/support/publishing-an-http-live-stream

máximo posible, ya que si se logra comunicar los navegadores directamente se ahorraría el salto de ida y vuelta hasta el servidor.

¿Cómo es posible conseguir que dos navegadores se comuniquen de manera directa?

Primeramente, los navegadores se conectan a una página web, es decir, acceden a una URL y al acceder a dicha URL se conectan a un servidor web y allí se realiza un intercambio de información, de comunicación, etc. entre los distintos navegadores. Esto permite que los navegadores se conecten directamente entre sí, pues lo que se logra es que se elimine la latencia que se necesita para pasar por el servidor y volver, lo que es ideal para establecer una videollamada o una comunicación.

Cuando nos conectamos a Internet no estamos en una red global, sino que realmente estamos de la red de nuestra casa, una red local (LAN) que se comunica con el router, y este obtiene la IP pública, accediendo de esta manera a Internet. Entonces, cuando queremos conectarnos con otro navegador, en realidad nos conectamos a la IP pública de su router, después atraviesa su router y una vez hecho esto llegamos directamente al navegador o a los navegadores, pues puede haber varios. Para conseguir esto, se una el framework ICE [19] que nos permite traspasar los NATs y permite establecer una comunicación entre dos navegadores. Sin embargo, no siempre es factible la comunicación directa entre un navegador y otro. Cuando esto sucede se emplean unos servidores disponibles en internet que se pueden desplegar, ejerciendo estos una función de relay, es decir, actúan de intermediarios cuya única función es recibir los datos de un navegador y enviárselos al otro. Este software estándar se conoce como TURN [20] y lo podemos apreciar en la siguiente figura[3] A.3.

[3]Imagen sacada de https://www.computerlanguage.com/results.php?definition=TURN+server

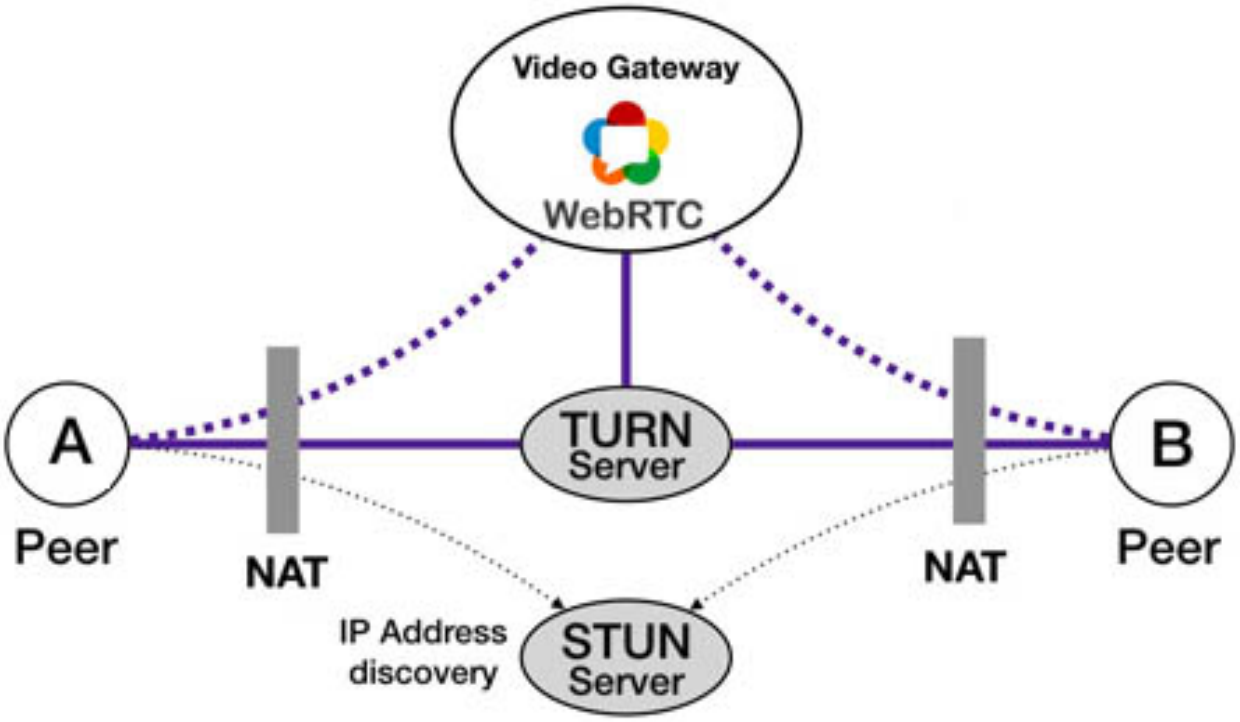


Ilustración A.3: Comunicación de dos navegadores mediando TURN

Apéndice B

Pila de producto

Historia de usuario 1	Login de usuario
Como	Usuario
Quiero	Poder logearme en el sitio web desde un botón de inicio de sesión que estará en la parte superior derecha de la aplicación (ver imagen ejemplo) y que abrirá un pequeño desplegable dónde insertar los datos de usuario (user y password).
Para poder	Acceder a mi cuenta a partir de mi usuario y disponer funcionalidades adicionales.
Prioridad	
Estimación	6 horas
Validación	El botón mostrará el nombre del usuario una confirma que se ha logueado. El botón mostrará el mensaje log in cuando no se esté logueado.

Cuadro B.1: Historia de usuario 1

Historia de usuario 2	Logout
Como	Usuario logueado
Quiero	Cerrar sesión.
Para poder	Impedir el acceso a mi cuenta.
Prioridad	
Estimación	0,5 h
Validación	La sesión se cierra y retorna a la vista en la que se está salvo que esta necesitase estar logueado. En caso de que ocurra la salvedad anterior, retornar al mapa principal.

Cuadro B.2: Historia de usuario 2

Historia de usuario 3	Registro de usuarios
Como	Usuario
Quiero	Poder registrarme con mi correo y una contraseña.
Para poder	Interactuar con el sitio web en modo privado y acceder a las cámaras asociadas a su usuario (normalmente es un organismo.
Prioridad	
Estimación	4 h
Validación	Cada mapa y pantalla de visualización de cámara tendrá en la parte superior un botón de inicio sesión que al pulsar mostrará un desplegable para el user (un correo) y el password (Alfanumérico). Al entrar en la sesión permitirá seguir en la ventana en la que se está añadiendo cosas propias de ese usuario (determinar que cosas) o sino será tal cual lo que ya estaba viendo. Al loguearse, dónde aparecía el botón de iniciar sesión aparecerá el nombre del usuario (ventana superior del marco).

Cuadro B.3: Historia de usuario 3

Historia de usuario 4	Control sobre cámara (botones)
Como	Usuario registrado
Quiero	Poder controlar el movimiento de la cámara (mover arriba, abajo, izquierda, derecha, más zoom, menos zoom).
Para poder	Tener un control manual sobre la cámara.
Prioridad	
Estimación	12 h
Validación	Comprobaremos la latencia que tiene el sistema para la respuesta (intentar que sea suave). La cámara se desplaza en todas las direcciones (controlando también valores en diagonal).

Cuadro B.4: Historia de usuario 4

Historia de usuario 5	Navegar por el mapa.
Como	Usuario
Quiero	Moverme a través del mapa con opciones de zoom y pantalla completa.
Para poder	Tener un control preciso de donde se encuentra la cámara.
Prioridad	
Estimación	6 horas
Validación	Cuando el usuario web pulse el botón derecho entonces el sistema actuará respecto a los movimientos del ratón. Cuando el usuario web pulse el botón de +/- zoom entonces el sistema podrá llegar a los limites que tiene actualmente. Comprobar que se puede poner el mapa a pantalla completa.

Cuadro B.5: Historia de usuario 5

Historia de usuario 6	Cabecera de la página principal
Como	Usuario
Quiero	Tener una cabecera en la cual pueda acceder a varias funcionalidades directamente.
Para poder	Iniciar sesión y cambiar ajustes del sistema cómodamente.
Prioridad	
Estimación	8 horas
Validación	Debe mostrar el icono de la empresa y los botones de ajustes e iniciar sesión Al iniciar sesión debe aparecer el nombre del usuario y el botón para cerrar sesión.

Cuadro B.6: Historia de usuario 6

Historia de usuario 7	Grabar video desde la cámara
Como	Usuario registrado
Quiero	Grabar un video osociado a la transmisión de la cámara en directo.
Para poder	Guardar las partes del directo que resulten de mayor interés.
Prioridad	
Estimación	4 h
Validación	Comprobar que el video se graba correctamente con una buena resolución. Comprobar que solo los usuarios logeados pueden acceder a esta función. El video solo prodrá tener una duracción máxima de 10 minutos.

Cuadro B.7: Historia de usuario 7

Historia de usuario 8	Ver transmisión de la cámara
Como	Usuario
Quiero	Ver la transmisión de la cámara.
Para poder	Visualizar (en directo) los barcos que se acercan a las zonas de las cámaras.
Prioridad	
Estimación	16 h
Validación	Comprobar que la transmisión de la camará se carga según la camará relacionada. Verificar que la transmisión se muestra correctamente con el protocolo webrtc.

Cuadro B.8: Historia de usuario 8

Historia de usuario 9	Time-lapse del streaming
Como	Usuario registrado
Quiero	Tener un time-lapse de la transmisión de las cámaras en directo.
Para poder	Tener un resumen del directo ya que los sucesos ocurren de forma muy lenta.
Prioridad	
Estimación	16 h
Validación	Verificar que esta funcionalidad solo esta disponible para los usuarios logeados. Comprobar que se puede elegir el rango del time-lase para un dia, dos dias y una semana. Bajar la calidad de las minuaturas para que no tarde tanto en cargar las imagenes.

Cuadro B.9: Historia de usuario 9

Historia de usuario 10	Reproducción de vídeo
Como	Usuario registrado
Quiero	Poder reproducir un vídeo grabado del directo.
Para poder	Interactuar con el vídeo pausando, reproduciendo de nuevo, reproduciendo hacia atrás, hacia delante y ver en pantalla completa.
Prioridad	
Estimación	4 horas
Validación	Debemos comprobar la interacción de distintos botones que señalen cada una de las acciones descritas. El vídeo se debe mostrar en un modal con un tamaño considerable.

Cuadro B.10: Historia de usuario 10

Historia de usuario 11	Información de las cámaras
Como	Usuario registrado
Quiero	Ver la información de las cámaras.
Para poder	Ver y/o editar el nombre, ubicación, propietario, url, etc. de las cámaras.
Prioridad	
Estimación	8 h
Validación	Comprobar que la información de las cámaras es la correcta. Verificar que se puede cambiar la información de las cámaras.

Cuadro B.11: Historia de usuario 11

Historia de usuario 12	Añadir nueva cámara
Como	Usuario administrador
Como	Usuario administrador.
Quiero	Añadir una nueva cámara desde un panel de administración.
Para poder	Tener más visibilidad y control sobre las costas de Gran Canaria.
Prioridad	
Estimación	6 h
Validación	Comprobar que se puede añadir una cámara desde el formulario correspondiente. Comprobar los datos rellenados en el formulario.

Cuadro B.12: Historia de usuario 12

Historia de usuario 13	Editar cámara
Como	Usuario administrador
Como	Usuario administrador.
Quiero	Editar los datos de una cámara desde un panel administrativo.
Para poder	Consultar y/o modificar la información de dicha cámara.
Prioridad	
Estimación	6 h
Validación	Verificar que se rellenan todos los datos del formulario. Comprobar que los datos se actualiza correctamente.

Cuadro B.13: Historia de usuario 13

Historia de usuario 14	Descargar vídeo
Como	Usuario registrado
Quiero	Descargar el vídeo de un determinado momento.
Para poder	Tener el vídeo almacenado locamente y visualizarlo en otro momento.
Prioridad	
Estimación	2 h
Validación	El vídeo se tiene que descargar en formato mp4. Al descargar el vídeo también se va aguardar el vídeo en OpenStack Swift.

Cuadro B.14: Historia de usuario 14

Historia de usuario 15	Eliminar cámara
Como	Usuario administrador
Como	Usuario administrador.
Quiero	Eliminar una cámara desde un panel administrativo.
Para poder	Dar de baja o suspender temporalmente a una cámara.
Prioridad	
Estimación	2 horas
Validación	Comprobar que se puede eliminar la cámara de la base de datos

Cuadro B.15: Historia de usuario 15

Historia de usuario 16	Página de inicio
Como	Usuario
Quiero	Visualizar la ubicación de las cámaras sobre un mapa.
Para poder	Visualizar y controlar la red de cámaras, de forma gráfica.
Prioridad	
Estimación	2 horas
Validación	Comprobar que el mapa se visualiza correctamente. Comprobar que visualiza correctamente la cabecera del página, el mapa y las opciones del mapa (ver las cámaras, ver los barcos). Las cámaras se tienen que cargar de forma dinámica desde la base de datos.

Cuadro B.16: Historia de usuario 16

Historia de usuario 17	Color de las cámaras
Como	Usuario
Quiero	Pintar las cámara de diferente colores.
Para poder	Distinguir el estado de las cámaras visualmente.
Prioridad	
Estimación	2 h
Validación	Comprobar que el color de las cámaras se actualiza dinámicamente dependiendo de su estado. Comprobar que la cámara se pinta según su estado.

Cuadro B.17: Historia de usuario 17

Historia de usuario 18	Almacenar datos en OpenStack Swift
Como	Usuario registrado
Quiero	Almacenar en la nube las capturas del time-lapse y los vídeos tomados.
Para poder	Descargar o visualizar las capturas desde cualquier lugar.
Prioridad	
Estimación	8 horas
Validación	Comprobar que se ha almacenado los datos en OpenStack Swift. Comprobar que se puede traer los datos desde OpenStack Swift.

Cuadro B.18: Historia de usuario 18

Historia de usuario 19	Abrir panel del streaming
Como	Usuario
Quiero	Ver el streaming desde un panel sobrepuesto al mapa.
Para poder	Visualizar el streaming de forma más óptima.
Prioridad	
Estimación	6 horas
Validación	Comprobar que se carga el streaming correspondiente a cada cámara. Comprobar que solo los usuarios registrados pueden abrir el panel.

Cuadro B.19: Historia de usuario 19

Historia de usuario 20	Ver streaming desde el mapa
Como	Usuario
Quiero	Ver un pequeño streaming en un pop-up desde el mapa.
Para poder	Visualizar el streaming sin tener que abrir el panel del streaming.
Prioridad	
Estimación	12 horas
Validación	Verificar la se establece y se cierra la conexión del streaming correctamente. Al poner el ratón encima de una cámara se abrirá un pop-up con el streaming correspondiente.

Cuadro B.20: Historia de usuario 20

Apéndice C

Glosario de acrónimos

- **CLI:** Command Line Interface, interfaz de línea de comandos.
- **CSS:** Cascading Style Sheets, hoja de estilos en cascada.
- **HTML:** HyperText Markup Language, lenguaje de etiquetas de hipertexto.
- **TS:** TypeScript.
- **DOM:** Document Object Model, modelo de objeto del documento.
- **URL:** Uniform Resource Locator, localizador de recursos uniforme.
- **API:** Application Programming Interfaces, interfaz de programación de aplicaciones.
- **HU:** Historia de Usuario.
- **HTTP:** Hypertext Transfer Protocol, protocolo de transferencia de hipertexto.
- **TFG:** Trabajo de Fin de Grado.
- **FF:** Factor de Foco.
- **JSON:** JavaScript Object Notation, notación de objeto de JavaScript.
- **SCTP:** Stream Control Transmission Protocol, protocolo de control de la transmisión en directo.
- **TURN:** Traversal Using Relays around NAT, Recorrido usando relay alrededor de NAT.
- **RTSP:** Real Time Streaming Protocol, protocolo de transmisión en tiempo real.
- **ONVIF:** Open Network Video Interface Forum, interface abierta de red de video.
- **LAN:** Local Network Area, red de area local.
- **NAT:** Network Address Translation, traducción de direcciones de red.
- **SVG:** Scalable Vector Graphics, gráficos vectoriales escalables.

- **PWA:** Progressive Web App, aplicación web progresiva.
- **SPA:** Single Page Application, aplicación de página única.
- **CEO:** Chief Executive Officer, director ejecutivo.
- **HLS:** Http Live Streaming, director ejecutivo.
- **ANSI:** American National Standards Institute, Instituto Nacional Estadounidense de Estándares.

Bibliografía

[1] Redatel.net, "Funciones, beneficios y usos de una cámara ptz," https://www.redatel.net/html/funciones-beneficios-y-usos-de-una-ptz.html, 2021, [Online; accessed: 15-July-2021].

[2] SwiftStack, "Openstackswift," https://www.swiftstack.com/product/open-source/openstack-swift, 2021, [Online; accessed: 2-July-2021].

[3] colaboradores de Wikipedia, "Windy (weather service)," https://en.wikipedia.org/wiki/Windy_(weather_service), 2021, [Online; accessed: 9-June-2021].

[4] P. Agostiniani, "Canariaslife, sobre nosotros," https://canariaslife.com/info/, 2021, [Online; accessed: 20-July-2021].

[5] typescriptlang.org, "What is typescript?" https://www.typescriptlang.org/, 2021, [Online; accessed: 20-July-2021].

[6] M. W. Docs, "Conceptos básicos de html," https://developer.mozilla.org/es/docs/Learn/Getting_started_with_the_web/HTML_basics, 2021, [Online; accessed: 19-July-2021].

[7] MDN Web Docs, "Css (cascading style sheets)," https://developer.mozilla.org/en-US/docs/Glossary/CSS, 2021, [Online; accessed: 19-July-2021].

[8] DesarrolloWeb.com, "Lenguaje SQL," https://desarrolloweb.com/home/lenguaje-sql, 2021, [Online; accessed: 19-July-2021].

[9] DesarrolloWeb.com, "Laravel," https://desarrolloweb.com/home/laravel, 2019, [Online; accessed: 19-July-2021].

[10] tockcontent, "Bootstrap: guía para principiantes," https://rockcontent.com/es/blog/bootstrap/, 2020, [Online; accessed: 19-July-2021].

[11] DesarrolloWeb.com, "Angular," https://desarrolloweb.com/home/angular, 2020, [Online; accessed: 19-July-2021].

[12] R. Velasco, "Visual studio code," https://www.softzone.es/programas/utilidades/visual-studio-code/, 2020, [Online; accessed: 19-July-2021].

[13] atlanssian bitbucket, "Qué es git?" https://www.atlassian.com/es/git/tutorials/what-is-git, 2020, [Online; accessed: 19-July-2021].

[14] colaboradores de Wikipedia, "Overleaf," https://en.wikipedia.org/wiki/Overleaf, 2021, [Online; accessed: 19-July-2021].

[15] K. S. y Jeff Sutherland, "La guía de scrum. la guía definitiva de scrum: las reglas del juego." https://scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-Spanish-European.pdf, 2017, [Online; accessed: 6-July-2021].

[16] Ken Schwaber y Jeff Sutherland, "The kanban guide for scrum teams." https://scrumorg-website-prod.s3.amazonaws.com/drupal/2018-04/2018%20Kanban%20Guide%20for%20Scrum%20Teams_0.pdf, 2018, [Online; accessed: 6-July-2021].

[17] M. W. Docs, "Capturar fotografías con la cámara web," https://developer.mozilla.org/es/docs/Web/API/WebRTC_API/Taking_still_photos, 2021, [Online; accessed: 6-June-2021].

[18] w3.org, "Webrtc 1.0: Real-time communication between browsers," https://www.w3.org/TR/webrtc/, 2021, [Online; accessed: 13-june-2021].

[19] WebRTC, "Getting started with peer connections," https://webrtc.org/getting-started/peer-connections, 2021, [Online; accessed: 11-june-2021].

[20] WebRTC, "Turn server," https://webrtc.org/getting-started/turn-server, 2021, [Online; accessed: 11-june-2021].

[21] WebRTC, "What is WebRTC," https://webrtc.org/, 2021, [Online; accessed: 11-June-2021].