UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

TRABAJO FIN DE MÁSTER

Plataforma de visualización y alerta de datos en tiempo real

Máster Universitario en Ingeniería Informática

Autor: MONTIEL CANO, Juan José

Tutor: MORA RINCON, Miguel Angel
Departamento de Ingeniería Informática

FECHA: Septiembre, 2020

Agradecimientos

Especial agradecimiento a mi tutor que me ha guiado y acompañado en la realización de este proyecto, así como a mi familia que me ha apoyado durante toda la realización del mismo

Índice de contenidos

Abstract

In this project, the different platforms that we can currently find on the market to display data in temporary graphs have been searched and analyzed. It has been observed that these platforms specialize in a specific type of data, for example economic data. This has given rise to the other main part of the project, which was to create a platform as generic as possible, which would allow users to define data downloads from different sources and display that data in a single web portal. These data could be of different types, since the platform seeks to generalize the visualization of data. Another of the key points that this project seeks is to allow users to create alerts about this data so that they are notified.

Resumen

En este trabajo se han buscado y analizado las distintas plataformas que podemos encontrar actualmente en el mercado para mostrar datos en gráficos temporales. Se ha observado que estas plataformas se especializan en un tipo de dato concreto, por ejemplo tados económicos. Esto ha dado pie a la otra parte principal del proyecto, que era crear una plataforma lo más genérica posible, que permitiese a los usuarios definir descargas de datos de distintas fuentes y mostrar esos datos en un único portal web. Estos datos podrían ser de distinto tipo, ya que la plataforma busca generalizar la visualización de datos. Otro de los puntos claves que busca este proyecto es permitir a los usuarios crear alertas sobre estos datos para que sean avisados.

Índice de figuras

Índice de tablas

1 - Introducción

En el presente documento de memoria de trabajo de fin de máster, se mostrarán las motivaciones, los desarrollo y las conclusiones a las que se ha llegado durante la realización de este proyecto. Se ha estudiado y desarrollado, en parte, la creación de una plataforma que permita mostrar datos obtenidos de diversas fuentes online, en un entorno lo más amigable posible dentro de una página web.

Este proyecto viene motivado por la búsqueda de una plataforma lo más genérica posible, que permita a los usuarios añadir fuentes de datos de diversos tipos, de tal forma que puedan especificar el lugar y el dato que desean recuperar y el sistema los mostrará en un gráfico temporal. Se ha buscado también la posibilidad de que los propios usuarios puedan crear o suscribirse a alertas de datos que sean de su interés, para que el sistema les informe cuando se produzcan.

Teniendo en cuenta la diversidad de apis y sistemas de datos desde los que recoger datos, este sistema debe ser lo más flexible posible, para poder adaptarse a las diferentes peculiaridades de las fuentes desde las que se quieren recuperar los datos. Otro de los aspectos importantes a resolver, es el hecho de que en este tipo de aplicaciones, el tiempo de actualización de datos y aviso de alertas, debe ser lo más rápido posible; es por ello, que el portal web debe actualizarse en tiempo real cuando un nuevo dato entre en el sistema y lo mismo pasa con las alertas, estas deben anunciarse en cuanto que se produzcan.

1.1 - Alcance

El alcance de este proyecto ha sido intentar desarrollar una plataforma que pueda satisfacer las motivaciones anteriormente descritas. Para ello el sistema puede recuperar datos de fuentes de datos externas y mostrar estos datos, en una interfaz web amigable, en forma de gráfico.

Otro de los propósitos alcanzados es la posibilidad de que los usuarios puedan crear alertas asociadas a estos datos, de tal forma que el sistema notifique cuando una de estas alertas se cumple. Estas alertas pueden ser creadas por los usuarios o pueden decidir suscribirse a alertas creadas por otros usuarios previamente, de tal forma que se notificará a todos los usuarios suscritos a una alerta.

Como parte más interna del sistema, este sistema debe realizar las acciones creadas por el usuario para la descarga de los datos. Estas acciones debe realizarlas cada cierto tiempo, para descargar los datos y volcar los resultados a un almacenamiento desde el que se leerán los datos descargados. Por lo tanto, se debe matizar que esta parte del sistema de descarga de datos no se comporta como un sistema real time, ya que recoge los datos cada cierto tiempo y no cuando el dato cambia.

1.2 - Estructura del proyecto

La estructura de la memoria expuesta, será la siguiente. La memoria empieza con una introducción y un alcance, en la que se explican las motivaciones y desarrollos realizados a un nivel muy abstracto. A continuación se encuentra el del estado del arte que existe actualmente, en el que se detallan algunas de las herramientas, de análisis de datos, que existen en la actualidad. La memoria continúa con una explicación más detallada del diseño del sistema que se ha planteado para solucionar las motivaciones descritas.

Después de mostrar el diseño planteado, se entrará en detalle de cómo se ha desarrollado la solución, pero esta vez de una forma más técnica y explicando el porqué de cada uno de los desarrollos creados.

Llegando a la parte final, se mostrarán los resultados conseguidos tras la realización del proyecto. Mostrando capturas de cómo vería el usuario final el sistema y cómo interaccionan con el mismo.

Y por último, se detallarán las conclusiones a las que se ha llegado durante la realización del proyecto, tanto de sistemas encontrados, como de opiniones personales.

2 - Estado del arte

Para entender un poco más la motivación a la hora de realizar este proyecto, debemos entender que en el momento actual, los datos cada vez están tomando más importancia y la información es un arma muy poderosa. Por todo esto, las empresas cada vez utilizan más datos a la hora de realizar ciertas operaciones estratégicas de la empresa, como se muestra en el artículo titulado “Monetización de los datos: la importancia de los datos para las empresas”. [6]

Actualmente, la capacidad de saber leer los datos y anticiparse a lo que pueda suceder, puede suponer una gran diferencia frente a competidores y por ello cada vez más empresas contratan los servicios de ciertas herramientas que les permiten ver los datos, que a ellos les interesa, de una forma rápida y sencilla.

Sin embargo, muchas de estas herramientas se especializan en ciertos datos y no abarcan todos los datos. Esto hace que las empresas tengan que tener diferentes fuentes de datos que observar para tomar decisiones, ya que la mayor parte de los datos están conectados de forma directa o indirecta y en un grado mayor o menor.

Un ejemplo de necesidad de acceso a datos de distinto ámbito, puede ser el siguiente. Una empresa de seguros quiere abrir una nueva sucursal en una nueva ciudad, para ellos analiza los datos financieros de sus posibles competidores, así como de sus clientes. No obstante, también le interesa saber el grado de siniestralidad en la zona y un punto muy importante, son los datos meteorológicos de la zona, ya que una mayor incidencia de desastres climatológicos del área determinada, repercutirá directamente en un mayor número de siniestros. Por ello la empresa va a necesitar analizar tanto datos financieros como datos meteorológicos de diferentes herramientas, cada una especializada en su campo.

En el mercado existen numerosas herramientas/webs en la que poder visualizar datos de diversos tipos, a continuación mostraremos ejemplos de herramientas financieras y web de datos meteorológicos, que suelen ser las más comunes.[7][9][11][12]

2.1 - Herramientas Financieras

Uno de los casos más comunes de herramientas de datos, son las herramientas financieras. Estas muestran datos históricos de los diferentes valores en gráficas, las cuales permiten a los analistas observar tendencias en el mercado y comprar o vender en momentos concretos. A parte de mostrar datos, estas herramientas también permiten crear alertas para que avisan si la condición marcada se cumple o no.

Existen numerosas herramientas financieras, a continuación se hablará de dos de las más importantes.

2.1.1 - Bloomberg

Bloomberg es una de las mayores empresas de asesoría financiera que existen en el mercado, provee de una herramienta Bloomberg Terminal, en la que podemos encontrar datos financieros, así como noticias bursátiles, las cuales son importantes para los analistas financieros. Esta herramienta nos muestra los valores bursátiles de monedas, acciones y otros dividendos en gráficos temporales, nos permite programar alarmas para que los analistas estén avisados cuando los valores de ciertos mercados suban o bajen, dependiendo de la alerta. Por otra lado, esta herramienta también permite comprar y vender acciones desde la propia herramienta bursátil. [7][8]

Figura 2.1 - Bloomberg Terminal Dashboard

Esta herramienta es una de las más usadas en el mundo bursátil, la mayoría de empresas utilizan este software para trabajar en las bolsas de valores de los distintos países. El gran inconveniente que tiene esta herramienta es su alto valor, las empresas pagan 24.000$ al año por la suscripción de 1 terminal. A cambio reciben la herramienta más completa del mercado en cuanto a información bursátil.

Todos los datos mostrados en esta memoria sobre Bloomberg han sido recogidos de la wikipedia y de otro blog muy conocido de herramientas financieras.

2.1.2 - TradingView

Esta aplicación está cogiendo mucha importancia en los últimos años desde que fue creada en 2011. Su principal cometido es proveer de las herramientas necesarias para realizar análisis financieros. Al igual que Bloomber Terminal, esta aplicación nos permite observar distintos valores financieros y cómo han ido cambiando estos a lo largo del tiempo, también nos permite programar alertas muy detalladas, así como diferentes herramientas que facilitan la vida al analista y que le permiten analizar mejor los datos. [9][10]

Figura 2.2 - TradingView dashboard

Su principal peculiaridad, por la que los usuarios han dado mucho reconocimiento a esta herramienta, es que te permite compartir tus análisis con el resto de la comunidad y de la misma forma, los usuarios pueden suscribirse a los análisis que tengas publicados, además pueden ver cómo han ido los análisis que publicastes anteriormente. Otro de los grandes factores que hacen que los usuarios, no profesionales, escojan esta herramienta, es que la herramienta con su configuración más básica, es gratuita y solo se paga si necesitas algunos aspectos extra opcionales.

Todos los datos mostrados en esta memoria sobre TradingView han sido recogidos de su propia página web y de la página llamada “Escueladetraders”.

2.2 - Páginas web meteorológicas

Otra de las fuentes de información más consultadas del mundo es algo tan simple como el tiempo que hace o el tiempo que hará mañana. Existen miles de herramientas que nos muestran información meteorológica de distinto tipo, desde la temperatura, hasta la presión en atmósferas, pasando por las precipitaciones. La mayor parte de estas herramientas proveen de datos a futuro, predicciones, y solo en algunas dan históricos de datos.
A continuación mostraré dos ejemplos de fuentes de datos meteorológicos.

2.2.1 - Aemet

Este organismo provee de los datos oficiales de meteorología a nivel estatal. Se trata de la Agencia Española de Meteorología, es un organismo oficial que provee de los datos de las diferentes estaciones que tiene repartidas por España. Estas estaciones ofrecen más o menos datos en función del número y variedad de instrumentos con los que cuenten.[11]

Como se muestra en su portal web, para poder mostrar todos estos datos, la AEMET ha creado una API REST semipública, ya que necesitas solicitar una clave para utilizar la API, desde la cual los usuarios pueden consultar diversos datos meteorológicos, así como una gran variedad de información relacionada con el tiempo. Esta API se llama OpenData y ha sido creada por la AEMET.

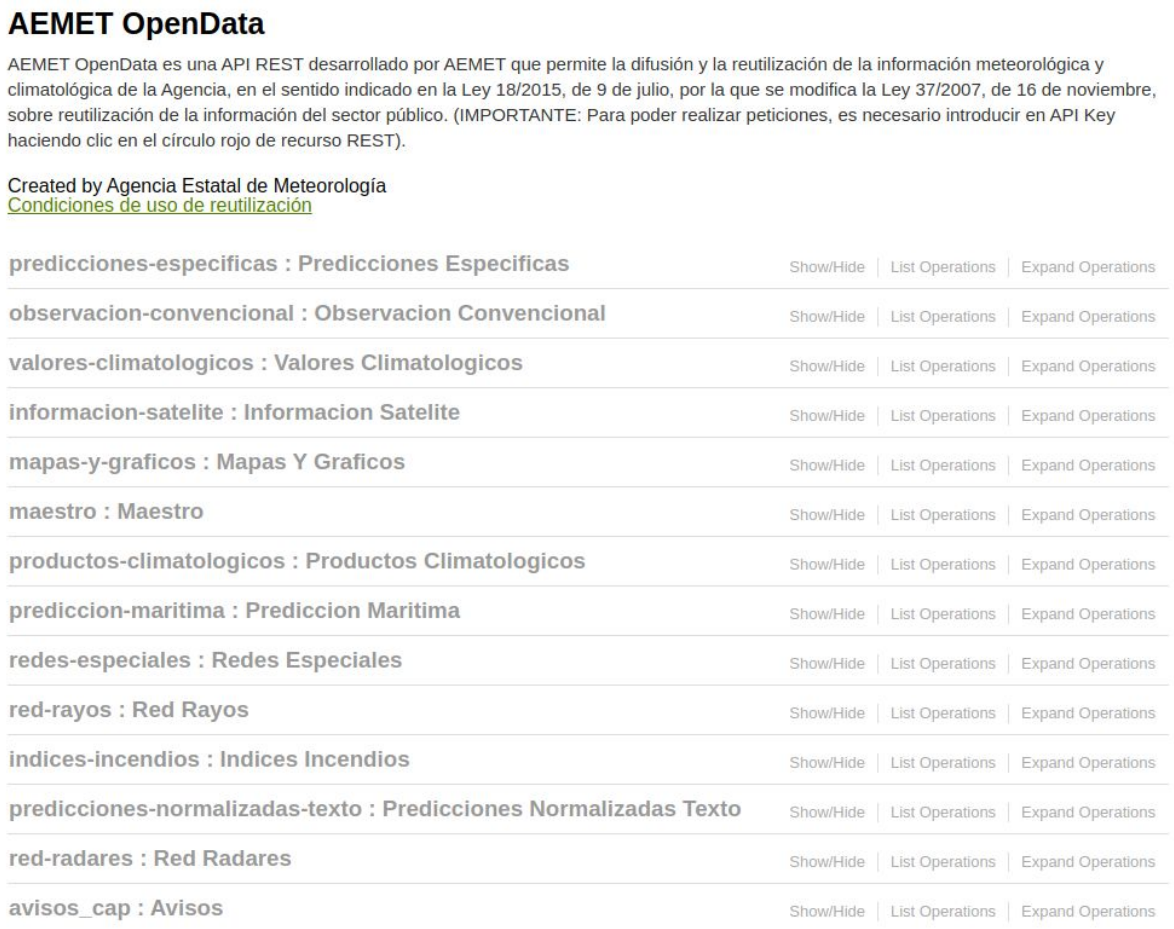

Figura 2.3 - AEMET API REST

En la imagen previa se pueden observar la cantidad de datos que AEMET publica a través de su API Rest, con ella se pueden realizar estudios muy diversos de catástrofes naturales, incendios, y un largo etc...

2.2.2 - ElTiempo

Otra de las páginas de datos meteorológicos más consultadas en España, según algunas publicaciones basadas en datos de visitas a nivel estatal., es la página eltiempo.es, esta página ofrece datos meteorológicos muy detallados, así como bien estructurados, de los próximos 14 días. Su interfaz gráfica es muy amigable y permite ver de forma rápida y sencilla el tiempo en diversos lugares o si se prefiere, los usuarios de la página pueden ver el detalle de los valores climatológicos de una zona concreta. [12][14]

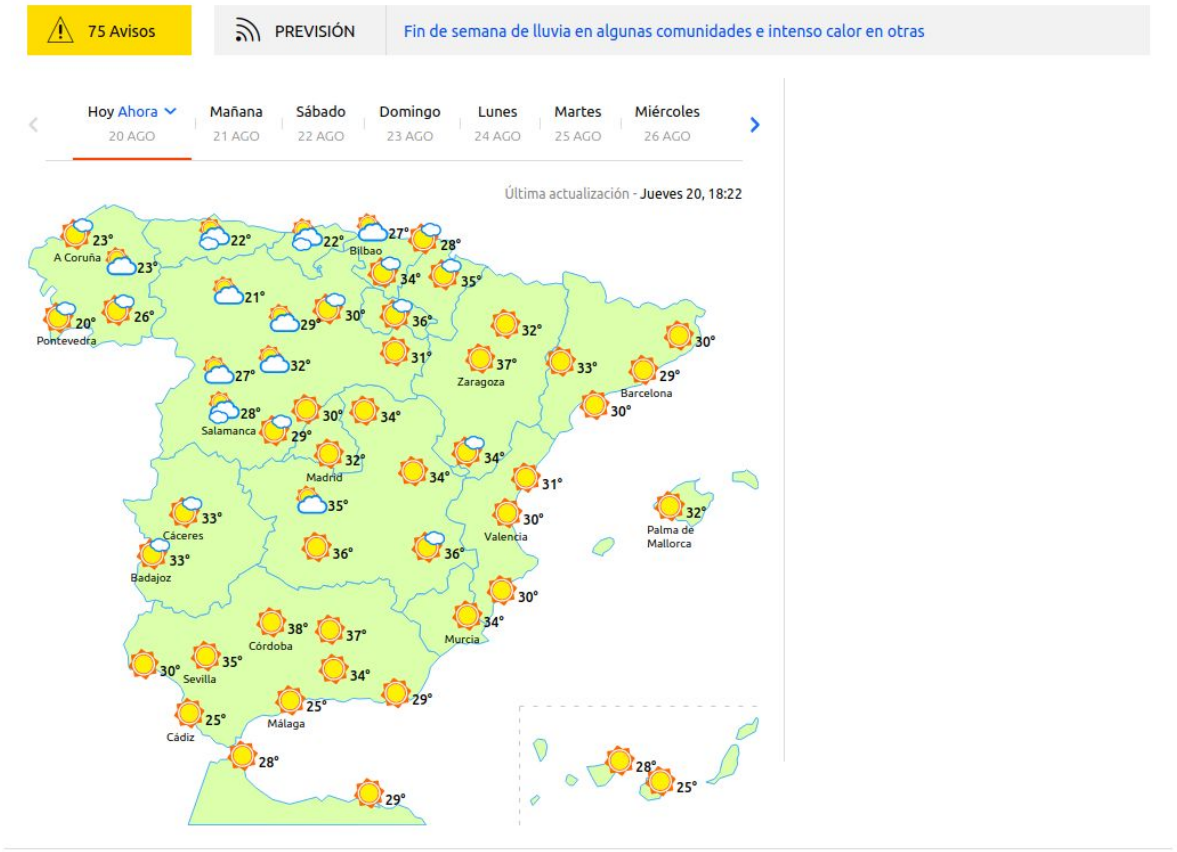

Figura 2.4 - ElTiempo.es

Todos estos datos son recogidos de fuentes oficiales como AEMET y plasmados de forma amigable en una interfaz web que hace que el usuario encuentre de forma fácil y sencilla la información.

3 - Análisis

En el capítulo que se encuentra a continuación, se plasmarán el conjunto de funcionalidades necesarias en el sistema, para que este cumpla con las necesidades descritas anteriormente.

Para ello se procederá a describir el alcance del sistema, mostrando los requisitos necesarios así como los casos de uso del mismo.

3.1 - Alcance del sistema

Teniendo en cuenta el número de horas que se pueden utilizar para el desarrollo del proyecto, y las limitaciones de personas que toman parte en el desarrollo del proyecto, el proyecto se centrará en las siguientes funcionalidades.

- Creación de un portal web donde accede el usuario:
    - Login: para poder acceder a ciertas funcionalidades del sistema, el usuario deberá esta logueado. Podrá loguearse con usuario y contraseña o a través de su cuenta de Google.
    - Registro: si el usuario no está registrado, podrá hacerlo a través del portal web introduciendo su email y contraseña o utilizando su cuenta de Google.
    - Añadir gráficos: el usuario podrá definir una nueva descarga de datos introduciendo los datos necesarios.
    - Gestión de alertas: el usuario podrá crear alertas, suscribirse a alertas ya creadas o eliminar la suscripción a una alerta.
    - Gestión de alertas activadas: el usuario podrá visualizar las alertas que se han activado, así como eliminar las alertas de las que ya tiene constancia.
    - Visualización de los datos: El usuario podrá acceder a cualquier gráfico para ver sus datos en un gráfico temporal. También podrá filtrar por tipo de gráfico.
- Creación de un sistema backend en la nube:
    - Almacenamiento: este sistema debe proveer de un sistema de almacenamiento de datos no estructurados, rápido y fiable. Donde se almacenarán los datos a los que accede tanto un servidor backend. como el portal web.
    - Análisis de alertas: el sistema debe estudiar cada nuevo dato entrante en el sistema de almacenamiento para ver si debe alertar a los usuarios.
    - Eliminación de datos por parte de administradores: los administradores podrán eliminar tanto usuarios, como cualquier tipo de dato que se incluya en el sistema.
    - Eliminación de datos por parte de usuarios: Los usuarios no podrán eliminar ningún documento almacenado, sólo podrán eliminar sus alertas activadas o desuscribirse de una alerta.

- Creación de un servidor backend de descarga de datos:
    - Acceso a sistema de almacenamiento: el servidor debe poder conectarse al almacenamiento definido en la nube y debe reaccionar cuando un una definición de descarga es insertada, modificada o borrada.
    - Descarga de datos: el servidor debe tener la capacidad de descargar datos de api. basándose en la definición de descarga definida por el usuario.

A parte de las funcionalidades básicas del sistema a nivel funcional, el sistema también debe satisfacer ciertas necesidades relacionadas con la escalabilidad, la seguridad y tiempo de respuesta. Estos requisitos son los siguientes:

- Se debe proporcionar seguridad en el almacenamiento de datos de usuario.
- El sistema debe ser escalable en función de la demanda producida.
- El tiempo de respuesta de ciertas partes del sistema, como la comprobación de alertas, debe ser lo menor posible.

3.2 - Casos de uso

A continuación se detalla el caso de uso de un usuario dentro del sistema.

Figura 2.1: Diagrama caso de uso de la plataforma

Tabla 3.1: Caso de uso - Login

Nombre	**Login**
Actores	Usuario registrado
Objetivo	Identificar al usuario dentro de la aplicación
Pre-condiciones	El usuario debe estar registrado o tener una cuenta de Google
Post-condiciones	El usuario accederá a funcionalidades disponibles solo para usuarios registrados
Escenario básico	El usuario rellena los datos de email y contraseña o selecciona la posibilidad de loguearse con su cuenta de Google.

Tabla 3.2: Caso de uso - Registro

Nombre	**Registro**
Actores	Usuario no registrado
Objetivo	Registrar al usuario dentro de la aplicación
Pre-condiciones	
Post-condiciones	El usuario podrá loguearse dentro de la aplicación
Escenario básico	El usuario rellena los datos de email y contraseña y envía estos datos para su registro dentro del sistema.

Tabla 3.3: Caso de uso - Visualización de datos

Nombre	**Visualización de datos**
Actores	Usuario
Objetivo	Visualizar datos dentro de una gráfica temporal
Pre-condiciones	
Post-condiciones	El usuario verá los datos a los que ha accedido, dentro de un gráfico temporal
Escenario básico	El usuario selecciona el gráfico que quiere ver y pulsa el botón para acceder a él.

Tabla 3.4: Caso de uso - Creación de descarga de datos

Nombre	**Creación de descarga de datos**
Actores	Usuario logueado
Objetivo	Añadir una nueva definición de descarga al sistema
Pre-condiciones	El usuario debe estar logueado dentro de la aplicación
Post-condiciones	El usuario podrá registrar una nueva definición de descarga de datos.
Escenario básico	El usuario rellena todos los datos necesarios para crear una nueva definición de descarga de datos y envía este formulario para que los datos se guarden en el sistema

Tabla 3.5: Caso de uso - creación de alertas

Nombre	**Creación de alertas**
Actores	Usuario logueado
Objetivo	Añadir una nueva alerta o suscribirse a una existente.
Pre-condiciones	El usuario debe estar logueado dentro de la aplicación
Post-condiciones	El usuario podrá crear una nueva alerta dentro del sistema
Escenario básico	El usuario rellena todos los datos necesarios para crear una nueva alerta y envía este formulario para que los datos se guarden en el sistema.

Tabla 3.6: Caso de uso - visualización de alertas

Nombre	**Visualización de alertas**
Actores	Usuario logueado
Objetivo	Ver las alertas a las que está suscrito el usuario.
Pre-condiciones	El usuario debe estar logueado dentro de la aplicación y debe estar suscrito al menos a una alerta. También debe estar visualizando el gráfico sobre el que tiene creada la alerta.
Post-condiciones	El usuario visualizará las alertas asociadas al gráfico que está visualizando.
Escenario básico	El usuario accede a visualizar un gráfico determinado y si tiene alguna alerta asociada a ese gráfico, se le mostrarán sus alertas.

Tabla 3.7: Caso de uso - eliminación de alertas activadas

Nombre	**Eliminación de alertas activadas**
Actores	Usuario logueado
Objetivo	Eliminar las alertas que se le han activado al usuario
Pre-condiciones	El usuario debe estar logueado dentro de la aplicación y debe tener al menos una alerta activada.
Post-condiciones	El usuario dejará de visualizar esa alerta como activada.
Escenario básico	El usuario accede a visualizar sus alertas activadas, pulsa el botón que le permite eliminarla y el sistema elimina esta alerta activada.

Tabla 3.8: Caso de uso - eliminación de alertas

Nombre	**Eliminación de alertas**
Actores	Usuario logueado
Objetivo	Eliminar suscripción de una alerta.
Pre-condiciones	El usuario debe estar logueado dentro de la aplicación y debe tener al menos una alerta a la que esté suscrito.También debe estar visualizando el gráfico sobre el que tiene creada la alerta.
Post-condiciones	El usuario dejará de estar suscrito a esa alerta.
Escenario básico	El usuario accede a visualizar sus alertas dentro de un gráfico concreto, pulsa el botón que le permite eliminarla y el sistema elimina esta alerta activada.

3.3 Análisis de requisitos

Tras mostrar el alcance del sistema así como los casos de uso que nos podemos encontrar por parte del usuario. Se puede extrapolar una serie de requisitos funcionales y no funcionales.

1. Requisitos funcionales
    1.1. portal web.
        1.1.1. [usuario] Login dentro de la aplicación.
        1.1.2. [usuario] Registro dentro de la aplicación.
        1.1.3. [usuario] Visualización de datos.
        1.1.4. [usuario] Creación de descarga de datos.
        1.1.5. [usuario] Cierre de sesión.
        1.1.6. [usuario] Visualización de alertas.

1.1.7. [usuario] Creación de alertas.
1.1.8. [usuario] Eliminación de alertas.
1.1.9. [usuario] Eliminación de alertas activadas.
1.2. Backend de la aplicación web.
1.2.1. [sistema] Almacenamiento de datos de la aplicación.
1.2.2. [sistema] sistema de autenticación de usuarios.
1.2.3. [sistema] Comprobación de situación de alertas.
1.2.4. [administrador] Eliminación de documentos y usuarios.
1.2.5. [sistema] Seguridad en almacenamiento de datos de usuario.
1.3. Servidor de descarga de datos.
1.3.1. [sistema] Acceso a los datos almacenados.
1.3.2. [sistema] Descarga de datos en base a lo definido por el usuario.
1.3.3. [sistema] Planificación de momento de la descarga.
1.3.4. [sistema] Almacenamiento de los datos descargados.
2. Requisitos no funcionales.
2.1. Portal web.
2.1.1. Portal web usable e intuitivo.
2.1.2. Control de acciones del usuario.
2.2. Backend de la aplicación web.
2.2.1. Escalable.
2.2.2. Tolerante a fallos.
2.2.3. Se debe garantizar el servicio.
2.2.4. Seguro.
2.2.5. Comunicación en tiempo real entre el backend y el portal web.
2.3. Servidor de descarga de datos.
2.3.1. Escalable.
2.3.2. Se debe poder acoplar con sencillez al backend del portal web.

4 - Diseño

4.1 - Diseño del sistema

Tras las motivaciones de este proyecto y la muestra del análisis de requisitos realizados en el apartado anterior que han llevado a desarrollar este proyecto, se comenzará a detallar el proyecto, comenzando con el diseño del sistema, acompañado de las explicaciones pertinentes.

Antes de mostrar el diseño de los distintos componentes del sistema, se ha creído conveniente mostrar cómo cada uno de estos componentes interaccionan entre sí. De esta forma se visualizará de una forma más sencilla cuales son los distintos componentes del sistema y en qué consiste cada uno de ellos, antes de entrar en el detalle de diseño de cada uno.

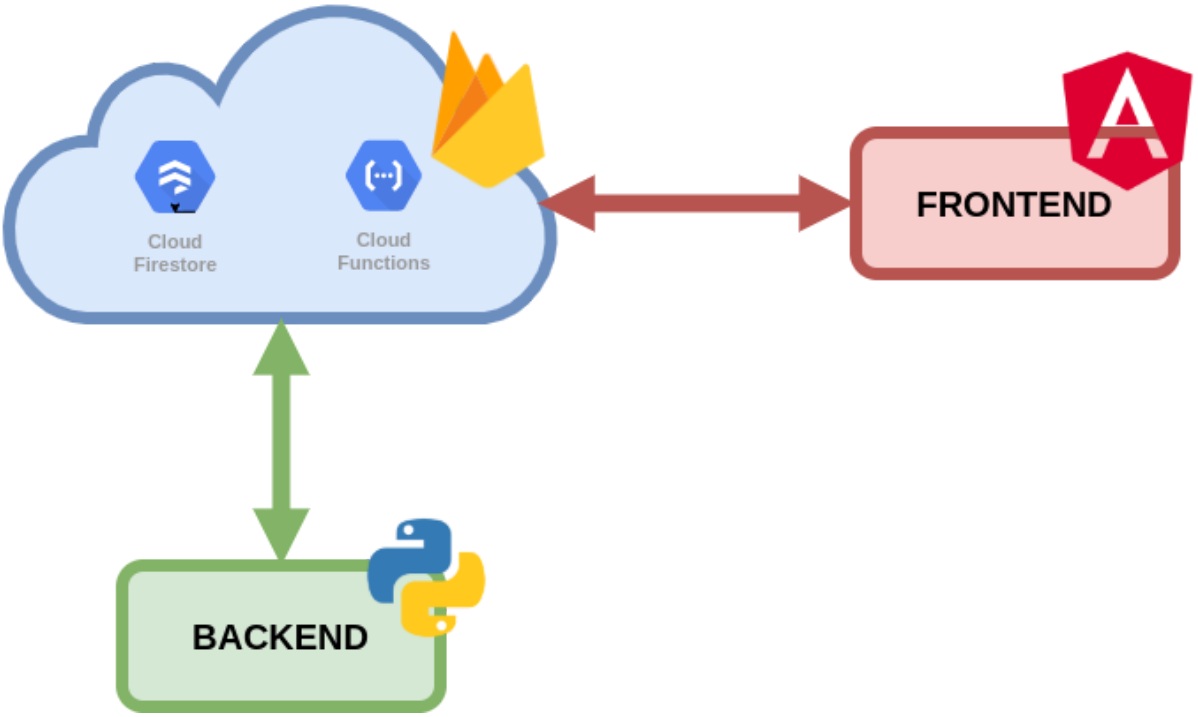

Figura 4.1 - Diagrama de módulos

El sistema se compone de tres componentes principales y completamente independientes. Un primer componente en el que se encuentra la interfaz web montada con Angular y que puede ser alojada en cualquier sistema de hosting. Un segundo componente intermedio que hace la función de backend del componente web y que se compone de una serie de funciones que comprueban si las alertas definidas por los usuarios se cumplen, y de un sistema de base de datos noSQL en el que se almacenan los datos de los gráficos y las alertas.

El Tercer componente principal del sistema es un servidor montado en Python cuya función principal es escuchar los cambios producidos en el sistema backend montado en la nube, y en base a los datos de los gráficos, descargar los datos definidos por el usuario y cargar los datos resultantes en el sistema de base de datos instalado en la nube.

Se comenzará hablando sobre el diseño montado en el segundo componente del sistema. En este componente, backend montado en la nube, se encuentra la base de datos montada sobre Cloud Firestore y las funciones que comprueban las alertas, montado sobre Cloud Functions de Firebase.

4.2 - Cloud Firestore

Se ha decidido utilizar este sistema de almacenamiento de datos no estructurados, debido a su facilidad de uso y su flexibilidad, sin olvidar la escalabilidad y continuidad del servicio que ofrece Firebase, lo cual es un requisito indispensable del sistema. Otro de los puntos importantes a tener en cuenta dentro de este servicio, es la gran flexibilidad en materia de

seguridad, permitiendo crear reglas tan específicas como el administrador quiera, lo cual ofrece una gran capa de seguridad para el sistema. Dentro de este sistema de base de datos NoSQL montado en la nube, se han definido los distintos tipos de datos que almacenará el sistema. En esta base de datos están almacenadas las definiciones de descarga de datos montadas por los usuarios, la descargas realizadas por el sistema de los datos buscados y las alertas definidas por el usuario. Cada uno de estos datos están almacenados en distintas colecciones dentro de Cloud Firestore: [15]

- **graphs-definition**: definiciones de descarga de datos
- **graphs**: almacena los datos ya descargados de los gráficos
- **alerts**: almacena las alertas definidas por los usuarios.
- **user-alerts**: almacena los datos de los usuarios y sus alertas que han sido activadas.

4.2.1 - Graphs-definition

En esta colección se encuentran las definiciones de descarga de datos. Estas están pensadas para que se pueda definir la descarga de cualquier tipo de dato, con esto se quiere decir que la definición de descarga de datos es lo suficientemente flexible como para que sirva para la mayoría de sistemas de descarga de datos. Cuando un usuario quiere definir una nueva descarga de datos, debe definir los siguientes campos:

- **name**: nombre del gráfico definido por el usuario
- **initial_url**: url inicial a la que el sistema irá a buscar los datos que se deben descargar
- **origin**: actualmente soporta el valor API y con este campo indicamos que los datos están almacenados en una API o en algún otro sistema
- **params**: se definen los parámetros extra, necesarios, en formato JSON con los que se va a realizar la petición a la API
- **header**: se definen los headers en formato JSON con los que se va a realizar la petición a la API
- **tree_access**: este campo almacena la secuencia de accesos que debe realizar el sistema para encontrar los datos buscados, en un array. La definición de cada acceso tiene el siguiente formato:
    - **type**: sus valores posibles son url o data. Se define si el tipo de dato se va a encontrar en los datos descargados o si se va a tener que realizar otra petición
    - **params**: se definen los parámetros extra que va a necesitar la nueva petición si type = url
    - **value**: este valor es de tipo array y en él encontraremos las palabras o posiciones clave a los que debe acceder el sistema para encontrar los datos o la url a la que realizar la petición
- **type**: en este literal se almacenará que tipos de datos son los que se van a descargar, para facilitar el filtrado, por tipos de datos, en la interfaz web.

A continuación se mostrará un ejemplo de un json de definición de una descarga, en este caso se define la descarga de la temperatura de un municipio de barcelona


```
{
  "origin":"API",
  "headers":{
     "cache-control":"no-cache"
  },
  "tree_access":[
     {
        "type":"url",
        "params":{

"api_key":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJqdWFubW9udEB1Y20uZXMiLCJqdGk
iOiJhNGUxZjQ5NS0xMWFlLTRlMjQtYjUyMi1kZDczNDg1YjNmYTUiLCJpc3MiOiJBRU1FVC
IsImlhdCI6MTU4MjIzNDg2MSwidXNlcklkIjoiYTRlMWY0OTUtMTFhZS00ZTI0LWI1MjItZ
GQ3MzQ4NWIzZmE1Iiwicm9sZSI6IiJ9.oC4GxhmXPj6j-DMfwB0Rrv_9OvCwKtckel4EyQH
tdAA"
        },
        "value":[
           "datos"
        ]
     },
     {
        "type":"data",
        "params":"",
        "value":[
           0,
           "prediccion",
           "temperatura",
           "dato",
           0,
           "value"
        ]
     }
  ],
  "params":{
```

```
"api_key":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJqdWFubW9udEB1Y20uZXMiLCJqdGkiOiJhNGUxZjQ5NS0xMWFlLTRlMjQtYjUyMi1kZDczNDg1YjNmYTUiLCJpc3MiOiJBRU1FVCIsImlhdCI6MTU4MjIzNDg2MSwidXNlcklkIjoiYTRlMWY0OTUtMTFhZS00ZTI0LWI1MjItZGQ3MzQ4NWIzZmE1Iiwicm9sZSI6IiJ9.oC4GxhmXPj6j-DMfwB0Rrv_9OvCwKtckel4EyQHtdAA"
  },
  "name":"Predicción Sant Martí de Centelles - 12am",
  "timeFormat":"",

"initial_url":"https://opendata.aemet.es/opendata/api/prediccion/especifica/municipio/diaria/08224"
}
```


Estas definiciones de descarga son las que el segundo componente principal del sistema está escuchando y en base a lo definido descarga los datos.

4.2.2 - Graphs

Cuando el sistema descarga los datos, almacena los resultados, en otra colección de datos dentro de Cloud Firestore, con el formato de datos que se mostrará a continuación, y con el mismo identificador que la definición creada por el usuario:

- **data**: almacena el resto de campos del grafico
    - **name**: nombre del gráfico definido por el usuario
    - **series**: array de los datos descargados. Cada dato descargado tiene la siguiente estructura:
        - **name**: marca temporal donde se define la hora en la que se ha descargado el dato
        - **value**: valor descargado
    - **type**: en este literal se almacenará que tipos de datos son los que se van a descargar, para facilitar el filtrado, por tipos de datos, en la interfaz web.

Este es un ejemplo de las descargas producidas por la definición del ejemplo anteriormente descrito:


```
{
  "data": {
     "name": "Prediccion Sant Marti de Centelles - 12am",
     "series": [
        {
           "name": "2020-08-10 16:53:26",
           "value": 10
```

```
        },
        {
            "name": "2020-08-10 17:03:25",
            "value": 23
        },
        {
            "name":"2020-08-10 17:04:26",
            "value": 30
        }
    ]
  }
}
```

4.2.3 - Alerts y user-alerts

Otra de las funcionalidades principales de la aplicación, es la posibilidad de definir o suscribirse a alertas. Estas alertas están relacionadas con el gráfico pero no son únicas para el usuario que la crea, simplemente el usuario que la crea es un mero suscriptor mas. Para manejar el sistema de alertas, en Firestore podemos encontrar dos colecciones independientes, una para guardar la definición de las alertas y otra para informar a un usuario de que una alerta se ha cumplido.

En la definición de alertas podemos encontrar los siguientes campos:

- **graph_id**: identificador del gráfico al que se va a asociar la alerta
- **name**: nombre identificativo de la alerta
- **type**: tipo de alerta. Actualmente se soportan Max y Min
- **users**: lista de identificadores de usuario que se han suscrito a esa alerta
- **value**: valor con el que se debe comparar para saber si se cumple la alerta

Aquí podemos ver un ejemplo de definición de alerta.

```
{
  "graph_id": "G5Qy3kSxVVZx4EZ6uohj",
  "name": "test_2",
  "type": "Max",
  "users": [
     "1kmLC46VkXTZEwizxH9Bv3ET4353"
  ],
  "value":100
}
```

Por otro lado, cuando el sistema comprueba que una alerta se cumple, se crea una activación de alerta dentro de una estructura definida para cada usuario. Los campos de esta estructura son:

- **_id**: identificador de usuario
- **alerts_activated**: array que contiene los campos de la alerta que se ha cumplido
    - graph_id, name, type, value

Este es un ejemplo de una activación de alerta para un usuario.

```
{
  "_id": "1kmLC46VkXTZEwizxH9Bv3ET4353",
  "alerts_activated": [
     {
        "graph_id": "G5Qy3kSxVVZx4EZ6uohj",
        "name": "test_2",
        "type": "Max",
        "value":100
     }
  ]
}
```

Expuestas cada una de las estructuras de datos que podemos encontrar en el sistema, se explicará el porqué de estas esta disposición. En primer lugar, el sistema de creación de gráficos se ha intentado que sea lo más flexible posible, el campo **tree_access** le da la posibilidad al usuario de realizar todas las iteraciones necesarias hasta encontrar el dato buscado. Este sistema tan abierto permite realizar búsquedas incluso dentro de un json anidado. Por ejemplo, si se quiere realizar una búsqueda sobre el siguiente json:

```
{
  "Att1": {
     "Att2": {
        "Array1": [
           {
              "Att3": "valor1",
              "Att4": "valor2"
           }
        ]
     }
  }
}
```

Dentro del campo **tree_access**, el usuario tendrá que introducir el siguiente json para poder recuperar al valor almacenado dentro de la clave Att3:

```
"tree_access":[
     {
        "type":"data",
        "params":"",
        "value":[
           "Att1",
           "Att2",
           "Array1",
           0,
           "Att3"
        ]
     }
  ],
```

Cuando los datos son descargados, se introducen en otra estructura independiente, de tal forma que la estructura de definición de componentes se quede lo más ligera posible para que sea menos pesada a la hora de que el segundo componente tenga que descargarse las modificaciones o creaciones de nuevos gráficos. Además esta separación permite que la parte front-end del sistema simplemente se suscriba a los datos descargados y no descargue datos innecesarios que no le sirvan para mostrar los datos.

Por otro lado, la definición de alertas también se encuentra separada de los datos del usuario para que la definición de alertas sea completamente independiente del usuario que la crea, lo cual hace que la estructura sea mucho más ligera ya que solo tiene que almacenar una lista de identificadores de usuario. Esto también hace que si un usuario no se ha suscrito a ninguna alerta o no se ha activado ninguna de sus alertas, el sistema no creará el documento donde se almacenan sus alertas activadas, de esta forma la estructura de alertas activadas solo se crea cuando es necesario y además permite al usuario borrar la notificación de alerta activada, dentro de su estructura de alertas.

Para concluir con este apartado, cabe mencionar que toda esta flexibilidad y fiabilidad mostrada anteriormente, no deja de lado la parte de seguridad dentro del sistema. En este caso, firestore nos proporciona un sistema de reglas con las que podemos implementar un seguridad a nivel de usuario o incluso documento.

4.3 - Firebase Functions y Google Authentication

Esta parte del sistema se encuentra una parte pequeña, pero muy importante, escalable y potente. Se trata de las funciones que realizan la comprobación de si una alerta se cumple o no. Debido a que existe un sistema backend que es capaz de subir a la aplicación miles de datos a la vez, el sistema backend montado sobre la nube debe ser capaz de poder hacer frente a esta demanda y debe poder analizar cada uno de los nuevos datos, para poder ver si una alerta se cumple o no. Por todo ello, se ha decidido montar esta parte del sistema sobre el sistema de **Functions** de Google Cloud, este sistema es capaz de lanzar una función distinta por cada nuevo dato que entre en la plataforma. Cuando una función es lanzada para comprobar el nuevo dato que ha entrado, recupera las alertas asociadas al gráfico al que está asociado el nuevo dato, y comprueba si alguna de las alertas se cumple. Si se cumple alguna condición de alerta, la función creará la correspondiente estructura de alerta activada en cada uno de los usuarios que están suscritos a esta alerta.

Como podemos ver en su web oficial, Functions ofrece ciertas limitaciones que se deben tener en cuenta si el sistema crece demasiado, la principal limitación es que Functions es el número de funciones que se pueden ejecutar en 1 segundo, 10000 invocaciones y cada una de las funciones no puede tardar más de 540 segundos en terminar.[13]

Además del servicio de Functions que proporciona Firebase, también se ha implementado el servicio de Google Authentication, el cual permite a los usuarios registrarse y loguearse en la aplicación de forma segura, de tal forma que Google Authentication mantiene seguros todos los datos sensibles del usuario, delegando esta seguridad de datos sensibles en Google Cloud. Este servicio de autenticación también permite que, dentro de la aplicación web que se describe a continuación, ciertas funciones están sólo disponibles para usuarios logueados.

4.4 - Portal web

Uno de los principales componentes de este tipo de sistemas, es su portal web. Para el desarrollo del portal web de esta aplicación, se ha decido utilizar Angular, debido a su gran compatibilidad con los servicios de Firebase. Este lenguaje modular nos permite crear componentes reutilizables dentro de la aplicación, lo cual reduce mucho el código y facilita la implementación de nuevos componentes más grandes.

A la hora de crear una aplicación en este framework, se debe tener en cuenta que se debe diseñar la aplicación separandola en componentes lo más pequeños posibles, y que los componentes más grandes, serán la composición de varios componentes más pequeños. También se debe tener en cuenta que no solo existen componentes visuales que nos permiten mostrar información, sino que también existen componentes de servicio, los cuales no tienen parte web, sino que se crean para realizar una serie de tareas backend.

Para el portal web se ha decidido realizar la siguiente estructura de componentes y servicios:

- **admin**: en esta carpeta encontraremos los componentes de administración de usuarios.
    - **login**: componente web que permite hacer login dentro de la aplicación.
    - **new-graph**: componente web que permite al usuario crear la definición de una nueva descarga de datos.
    - **register**: componente web que permite a un usuario registrarse
- **dashboard**: componente principal donde se encuentra el dashboard de la aplicación
- **user-alerts-dashboard**: componente donde se mostrarán las alertas activadas del usuario
- **reusable-components**: carpeta que contiene los componentes que van a ser reutilizados a lo largo de toda la aplicación
    - **alert-activated**:componente web que muestra una tarjeta con la información de la alerta que ha sido activada.
    - **chart**: componente web que muestra el gráfico.
    - **dashboard-card**: componente web que muestra una tarjeta con la información básica del gráfico.
    - **new-alert**: componente web que permite la creación de una nueva alerta.
    - **alert-defined**: componente que muestra la información de una alerta definida.
- **services**: esta carpeta contienen los servicios que han sido definidos dentro de la aplicación
    - **auth**: servicio que realiza la autentificación del usuario contra Firebase y guarda los datos temporales en un almacenamiento local común para todos los componentes.
    - **db**: dentro de esta carpeta se encuentran los servicios que realizan la conexión con la base de datos, en este caso solo tenemos un servicio que realiza la conexión con Cloud Firestore.

Cada uno de los componentes web definidos en Angular, tienen la siguiente estructura de ficheros:

- **<nombre>-component.html**: incluye el código HTML del componente
- **<nombre>-component.scss**: incluye los estilos particulares del componente
- **<nombre>-component.ts**: incluye el código TypeScript del componente.
- **<nombre>-component.spec.ts**: incluye las especificaciones de definición del componente.

4.5 - Backend

A continuación se hablará del diseño que se ha decido para el backend de la aplicación. Esta parte del sistema es completamente independiente de todos los componentes anteriormente descritos. Su principal tarea es la de descargar los datos definidos por el

usuario y la carga los datos obtenidos, en Firestore. Se ha decidido que sea este componente backend el que se conecte al otro componente backend montado sobre Google Cloud, para que sea lo más independiente posible, este sistema de conexión permitiría tener varios backend escuchando sobre Firestore, de tal forma que si uno se cae, otro pueda ocupar su lugar.

Otro de las partes importantes que hace que el backend sea lo más independiente posible del resto de componentes, es que cuando el backend observa que algún documento de firestore es creado o se cambia, el sistema almacena o modifica este documento en un sistema de base de datos NoSQL, local, montado con Mongodb. Esta cualidad también nos permite reducir lo máximo posible el número de accesos a Firestore.

Actualmente el sistema lanza cada hora un thread que recorre todas las definiciones de gráficos almacenadas en MongoDB, realiza la descarga de todos los datos asociados a estos gráficos y almacena los resultados en una copia local en Mongo y otra en Firestore.

5 - Desarrollo

En el capítulo que se le muestra a continuación se hablará en detalle de cada uno de los componentes expuestos en el capítulo anterior. Para que la explicación sea lo más estructurada posible, se van a separar las explicaciones en tres grandes bloques, que son los mismos que los expuestos en el diagrama, es decir, se empezará hablando del módulo backend, se continuará hablando de la infraestructura cloud y por último se hablará del módulo frontend

5.1 - Backend

El servidor backend por el que se comenzará está montado completamente en Python, se ha escogido este lenguaje debido a su facilidad a la hora de programar y al gran número de librerías de las que dispone, concretamente, la librería de firebase es perfecta para las necesidades del proyecto. Se recuerda que este sistema está montado para que puedan existir tantos backend de respaldo como se deseen, además de para facilitar la posibilidad de que cada backend se encargue de un tipo de dato.

5.1.1 - Servicios

El módulo de servicios está montado de tal forma que existe un servicio conectado a Firestore escuchando sobre la colección de definición de gráficos, que contiene la definición de las descargas de datos definidas por los usuarios. Cuando un documento es creado, modificado o eliminado en esta colección, el sistema detecta el cambio y procede a realizar la acción pertinente sobre la base de datos local montada sobre MongoDB. Dentro de estos dos servicios, se encuentran las siguientes funciones:

- Firestore
    - **__init__**: inicializa las conexiones con las colecciones graphs-definition y graphs, donde se guardan las definiciones de descarga y los datos descargados respectivamente.
    - **get_document_by_id**: recoge los datos de un documento a partir de su id.
    - **update_data**: añade los datos descargados dentro del documento almacenado en la colección graphs.
- MongoDB
    - **__init__**: Inicializa la conexión sobre la colección data_objects dentro de MongoDB.
    - **save_object**: almacena un nuevo documento dentro de la colección.
    - **update_object**: modifica un documento determinado.
    - **remove_object**: elimina un documento concreto.
    - **find_object_by_name**: busca un documento a partir del nombre.
    - **get_id_by_name**: devuelve el id en base al nombre.

    - **find_object_by_id**: busca un documento a partir del identificador del documento.
    - **get_all_objects**: devuelve todos los documentos de la colección
- **callback**: esta función es la que se llama cuando un documento es modificado, eliminado o creado en firestore, y en función de la acción, llama a una de las funciones del servicio de MongoDB .

5.1.2 - Planificador

Por otro lado, se ha desarrollado un planificador dentro del módulo que lanza un proceso cada hora, este proceso recorre todos los documentos almacenados en MongoDB y crea un objeto, llamado DataObject, por cada documento. Este objeto contiene toda la información para descargar los datos y es capaz de realizar la descarga por si mismo. Por cada objeto creado se lanza la descarga de sus datos y se guardan sus datos de nuevo en otra colección dentro de Firestore. Dentro de la clase DataObject, podemos encontrar las siguientes funciones:

- DataObject
    - **__init__**: inicializa el objeto con los valores de descarga.
    - **get_data**: realiza la descarga de los datos.
    - **compose_url_with_dates**: crea una url a partir de unas fechas de inicio y de fin.
    - **__get_internal_data_from_dict**: recoge los datos desde el diccionario descargado.
    - **get_value**: devuelve el valor del parámetro solicitado por parámetro.
    - **change_value**: cambia el valor de un parámetro concreto y llama a la función correspondiente.
    - **change_params**: cambia el valor de params.
    - **change_headers**: cambia el valor de headers.
    - **change_initial_url**: cambia el valor de initial_url.
    - **change_tree_access**: cambia el valor de tree_access.
    - **change_origin**: cambia el valor de origen.
    - **change_name**: cambia el valor de name.
    - **change_id**: cambia el identificador.
    - **get_last_value**: devuelve el último valor descargado.
    - **get_dict_information**: devuelve la información del objeto en forma de diccionario.

5.1.3 - Controlador

Como último componente del servidor backend, se ha desarrollado un controlador, que simplemente contiene funciones estáticas de apoyo para crear los objetos a partir de jsons y viceversa.

- Controller:

- **generate_data_from_file**: genera un objeto DataObject a partir de un fichero yaml pasado por parámetro.
- **write_objects_to_file**: escribe los datos de un objeto DataObject en un fichero yaml.
- **generate_dict_from_data_objects**: genera una lista de Diccionarios a partir de una lista de DataObjects enviados por parámetro.
- **generate_objects_from_dict**: genera una lista de DataObjects a partir de una lista de diccionarios enviados por parámetro.
- **generate_dict_from_data_database**: genera un array de diccionarios a partir de un cursor de MongoDB enviado por parámetro.

Para mostrar de una forma más clara las conexiones que existen entre los distintos componentes de este módulo, a continuación verán un diagrama explicativo.

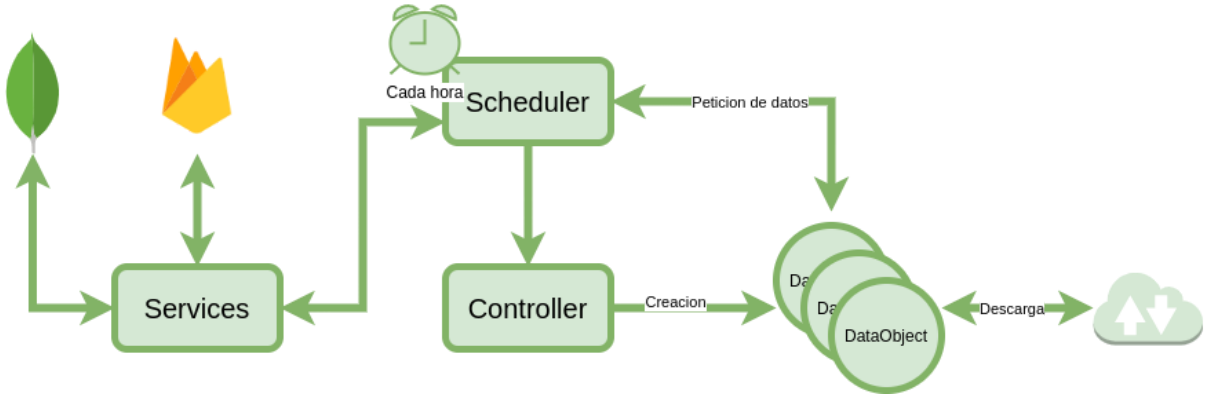

Figura 5.1 - Diagrama componente backend

5.2 - Firebase

Después de detallar cómo funciona el módulo backend del sistema, se continuará describiendo el siguiente módulo del sistema. En este caso el sistema backend montado sobre Firebase que funciona sobre Google Cloud. En este módulo, a parte de los datos almacenados en Firestore que se han detallado en el capítulo de diseño, se encuentran las funciones que comprueban si se debe producir una alerta o no. Estas funciones están montadas sobre el servicio de Functions de Firebase y están escritas en Javascript. Su funcionamiento es el siguiente. Existe una función llamada **onUpdatedData,** que es llamada cuando un nuevo dato es añadido a un gráfico ya existente. Cuando la función es activada, realiza una búsqueda de las alertas definidas a través del identificador del gráfico al cual se le están añadiendo nuevos datos. Si existen alertas para ese gráfico, comprueba si se cumple alguna alerta y en caso de que se cumpla alguna, crea la correspondiente alerta activada dentro del documento de alertas del usuario.

5.3 - Frontend

Por último, se detalla el módulo en el que se encuentra la aplicación web al que accede el usuario para consultar los datos descargados y mostrados en un gráfico, e informarse si una alerta se ha producido. Este módulo está desarrollado en el framework de Google, llamado Angular. Durante esta sección se detallarán los diferentes componentes y servicios que se encuentran dentro de este módulo, cuál es su funcionamiento y que funciones tienen dentro de la aplicación.

5.3.1 - Servicios

Se comenzará hablando de los servicios que realizan la conexión entre el portal web y los servicios montados en la nube, Firebase Authentication y Firestore. El primer servicio, de autenticación, permite a los usuarios entrar dentro del portal con un usuario y contraseña que los identifique o a través de su cuenta de Google. Si el usuario no está logueado, no podrá realizar ciertas acciones dentro de la aplicación. Su funciones internas son las siguientes:

- **isLoggedIn**: devuelve true o false dependiendo de si el usuario está logueado o no.
- **doLogout**: desloguea al usuario de la aplicación.
- **login**: realiza el login dentro de la aplicación a través de usuario y contraseña.
- **register**: registra un usuario a través de un usuario y una contraseña.
- **sendEmailVerification**: envía un email de verificación para comprobar el email puesto por el usuario.
- **sendPasswordResetEmail**: envía un email para resetear la contraseña del usuario.
- **loginWithGoogle**: realiza el login a través de una cuenta de Google.

El otro servicio definido en el módulo, es el servicio que realiza la conexión con las distintas colecciones dentro del servicio de Cloud Firestore. Este servicio permite recuperar documentos de colecciones, cambiar valores de otros documentos e incluso borrar documentos de colecciones. Sus funciones internas son las siguientes:

- **createGraph**: crea una nueva definición de descarga en la colección graph-definition, con los datos proporcionados por parámetro.
- **createAlert**: crea una nueva alerta o suscribe al usuario en una alerta si esta ya estaba definida en la colección alerts.
- **deleteAlert**: elimina la suscripción de una alerta de un usuario en la colección alerts.
- **deleteActivatedAlert**: elimina una alerta de las alertas activadas del usuario, de la colección user-alerts.
- **createUser**: crea la información básica de un usuario (identificador e email), dentro de la colección user-alerts.
- **getUserAlertsDefined**: devuelve las Alertas a las que está suscrito un usuario, de la colección alerts.
- **getUserAlertsActivated**: devuelve las alertas activadas del usuario, de la colección user-alerts.

- **getGraphDefinition**: devuelve la definición de un gráfico cuyo identificador es pasado como parámetro.
- **getGraphData**: devuelve la información de datos descargados de un gráfico cuyo id es pasado por parámetro, de la colección graphs.
- **getAllGraphData**: devuelve todos los documentos almacenados en graphs.
- **getGraphDefinitionTypes**: devuelve los distintos tipos de gráficos almacenados en la colección graph-definition.
- **getGraphsByType**: devuelve las definiciones de gráficos de un tipo pasado como parámetro.
- **getAllGraphsDefinition**: devuelve todas los documentos de la colección graphs-definition.
- **updateGraphDefinition**: modifica una definición de gráfico.

5.3.2 - Componentes

Después de detallar las funciones desarrolladas en los dos servicios principales del portal web, se hablará de los distintos componentes web que muestran la información del portal web, no si antes recordar que la programación en Angular, es una programación modular, por lo que los componentes más grandes están compuestos de componentes más pequeños.

Dentro de los componentes del módulo, se pueden encontrar componentes más pequeños y componentes más grandes, se comenzará hablando de los componentes más pequeños, como son los componentes reutilizables y los componentes de administración de usuarios. Dentro de los componentes reutilizables, se pueden encontrar los siguientes, con sus respectivas funciones principales:

- **alerts-activated**: este componente muestra, en forma de tarjeta, una alerta pasada como entrada. Permite abrir el gráfico directamente desde la tarjeta o eliminar la alerta activada.
- **chart**: muestra la información de una gráfico determinado en un gráfico temporal, este gráfico está soportado por la librería **ngx-charts**. Aunque este es un componente reutilizable, dentro de su ventana, también incluye el componente que permite crear una nueva alerta.
- **dashboard-card**: muestra en forma de tarjeta los datos básicos del gráfico y ofrece un botón con el que poder entrar a ver los datos en el gráfico.
- **new-alert**: permite a un usuario poder crear una alerta, este componente se coloca al lado de cada gráfico. Recoge los siguientes datos:
    - **name**: nombre de la alerta.
    - **type**: tipo de alerta (Min o Max).
    - **value**: valor con el que comparar si se ha producido la alerta.
- **alert-defined**: muestra en forma de tarjeta la información asociada a una alerta creada por el usuario, y permite borrar esa alerta.

Continuando con los componentes que permiten al usuario administrar su autenticación y la creación de nuevos gráficos, se tienen los siguientes componentes, cuyas funcionalidades son las siguientes:

- **login**: este componente es uno de los principales, permite al usuario loguearse en la aplicación de dos formas, puede introducir su usuario y contraseña o puede pulsar el botón que le permite loguearse a través de su cuenta de Google. Este componente llama al servicio de autenticación montado sobre Google Authentication.
- **new-graph**: provee al usuario de un portal web donde puede introducir todos los campos necesarios para crear una nueva descarga. Para los campos más complejos como el campo SequentialAccess, el usuario debe introducir los valores en formato json. Tras introducirse un nuevo gráfico, el componente utilizará el servicio de firestore para crear el documento en la colección correcta.
- **register**: permite al usuario introducir los campos necesarios para registrarse, y se conecta con el servicio de autenticación para realizar el registro.

Dejando a un lado los componentes atómicos del sistema, se pasará ahora a describir los componentes más grandes del módulo web. Estos componentes son componentes compuestos por otros componentes más pequeños, son los dos dashboard del portal. Uno de ellos permite ver en tarjetas las distintas definiciones de gráficos y el otro permite ver las tarjetas de todas las alertas activadas del usuario.

Estos componentes realizan una búsqueda a través del servicio de firestore, de los gráficos y las alertas activadas y los muestran en una pantalla que permite al usuario ver los gráficos y las alertas de una forma rápida y sencilla. Además, el dashboard donde se muestran los gráficos, permite ver sólo las alertas de un tipo determinado.

Después de explicar en detalle el desarrollo de cada uno de los componentes y servicios del módulo frontend, se mostrarán una serie de mockups donde se verá la como esta compuesto cada uno de los componentes más grandes del portal.

5.3.3 - Mockups

5.3.3.1 - Dashboard

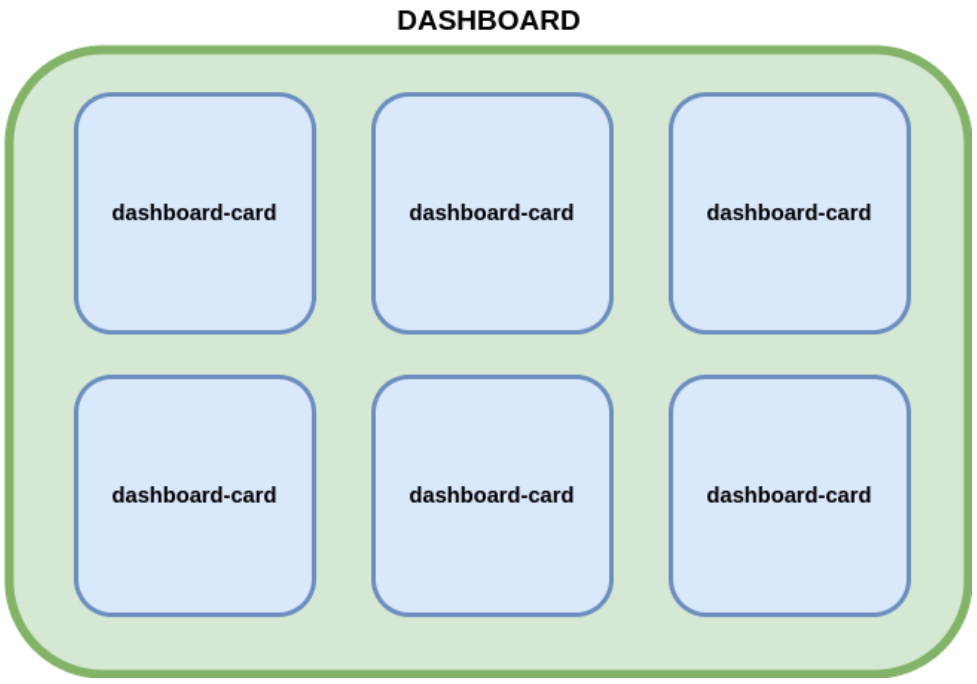

Figura 5.2 - Mockup dashboard portal web

5.3.3.2 - user-alerts-dashboard

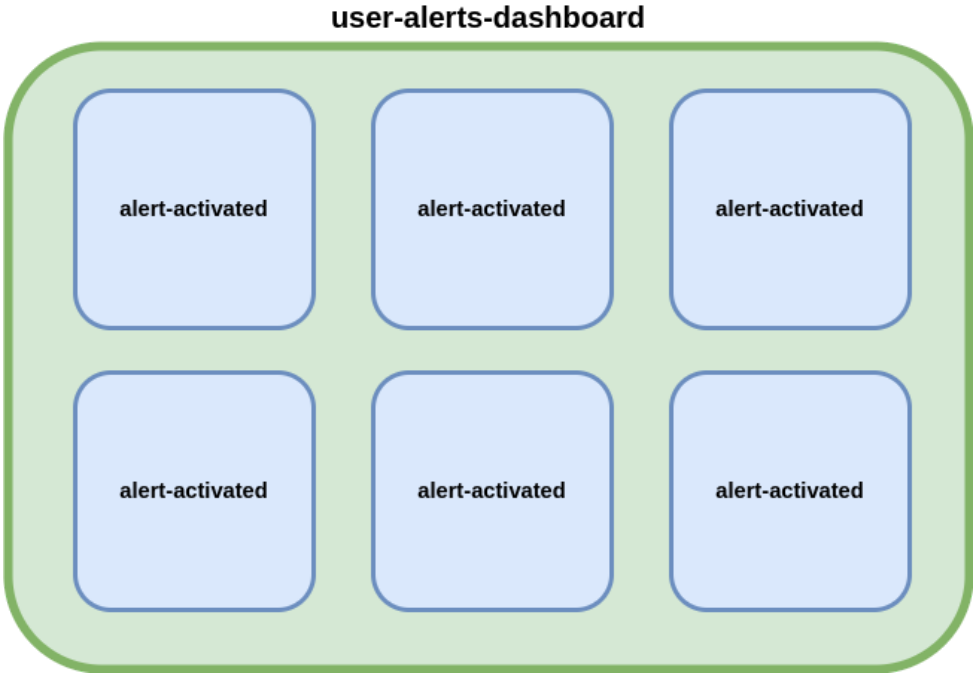

Figura 5.3 - Mockup dashboard alertas activadas

5.3.3.3 - chart

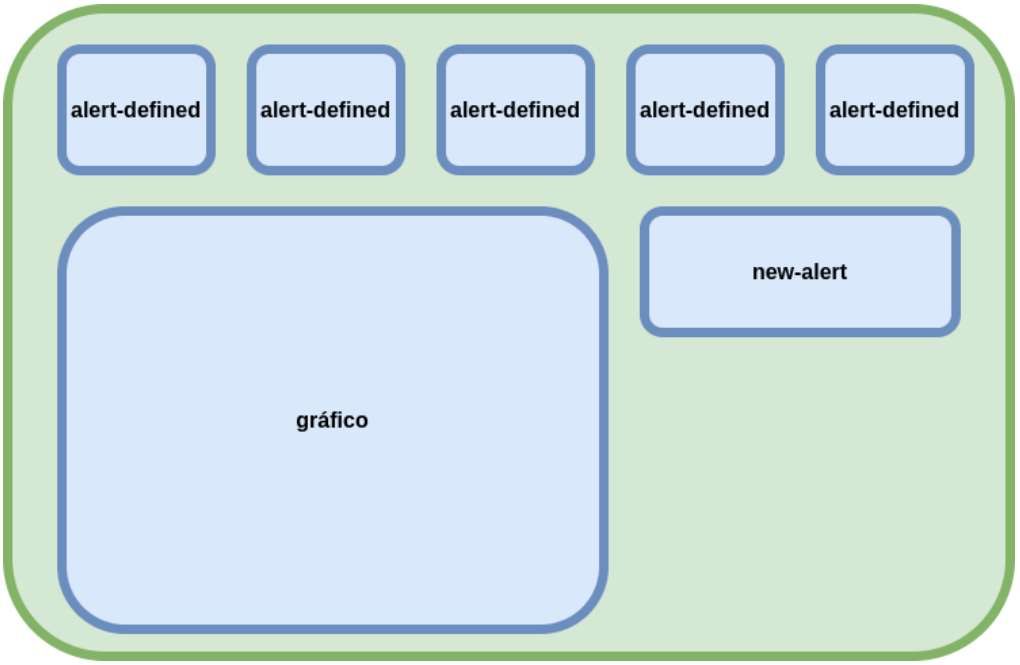

Figura 5.4 - Mockup pantalla gráfico

6 - Resultados

Llegados a este punto, después de detallar con la mayor profundidad y facilidad posible el proyecto y explicado como funcionan cada una de las piezas que componen la aplicación, Se mostrará con capturas de pantalla que el sistema funciona y cómo es su portal web. Los resultados que se van a mostrar en las imágenes, son recogidas de datos reales. Uno de los gráficos es la temperatura que pronostica la AEMET todos los días a las 12. AM en Sant Martí de Centelles y el otro es el valor del DÓLAR frente al EURO. Los dos datos son recogidos de APIs distintas.

6.1 - Backend

Se comenzará mostrando una captura de cómo se almacenan los datos en el MongoDB local del sistema.

Figura 6.1 - Objetos MongoDB almacenados

6.2 - Firestore

Los dos objetos de mongo mostrados en la imagen anterior son, a su vez, copias locales de las definiciones de gráficos almacenadas en Firestore, como se muestra en la siguiente imagen.

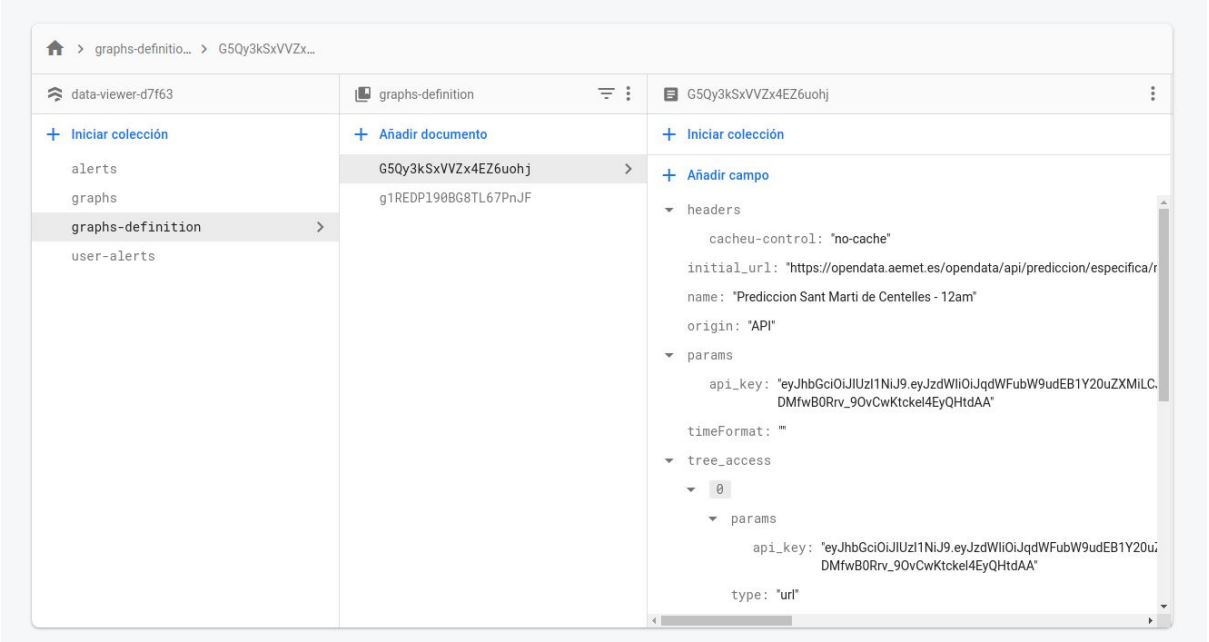

Figura 6.2 - Ejemplo documento graph-definition, Firestore

En este caso, la imagen muestra los datos relativos a la descarga de la predicción de temperatura. Tras ver cómo se almacenan las definiciones de descarga, se muestra cómo se almacenan los datos cuando el backend realiza la descarga de los mismo.

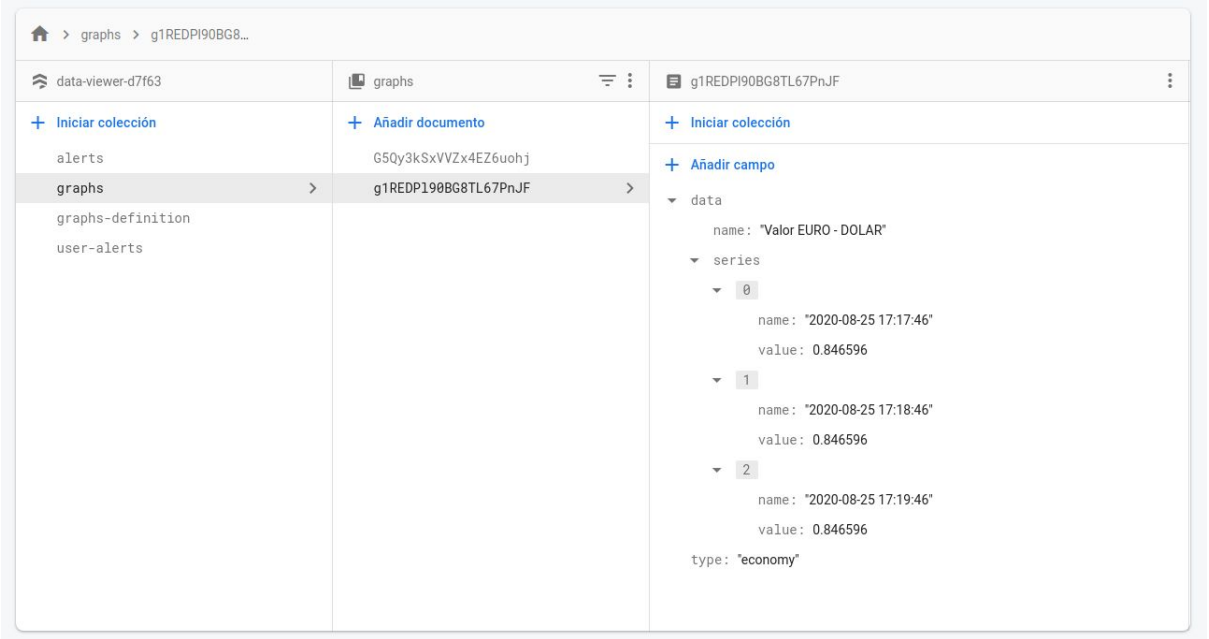

Figura 6.3 - Ejemplo documento graphs, Firestore

Estos son los datos descargados del valor del DOLAR frente al EURO. Por otro lado, cuando un usuario define una alerta, la alerta es almacenada en Firestore de la siguiente forma.

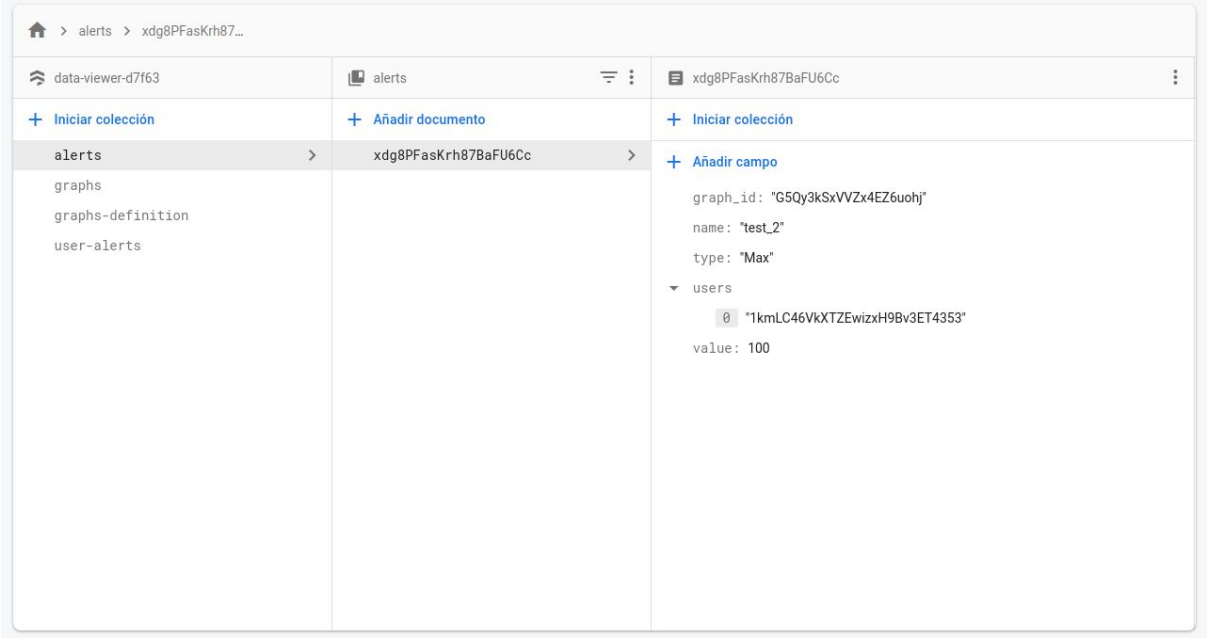

Figura 6.4 - Ejemplo documento alerts, Firestore

Como se puede ver, cuando el usuario se suscribe, el sistema añade su identificador a la lista de usuarios de la alerta. Cuando una de estas alertas se produce, el sistema crea una copia de la alerta en la estructura de alertas del usuario.

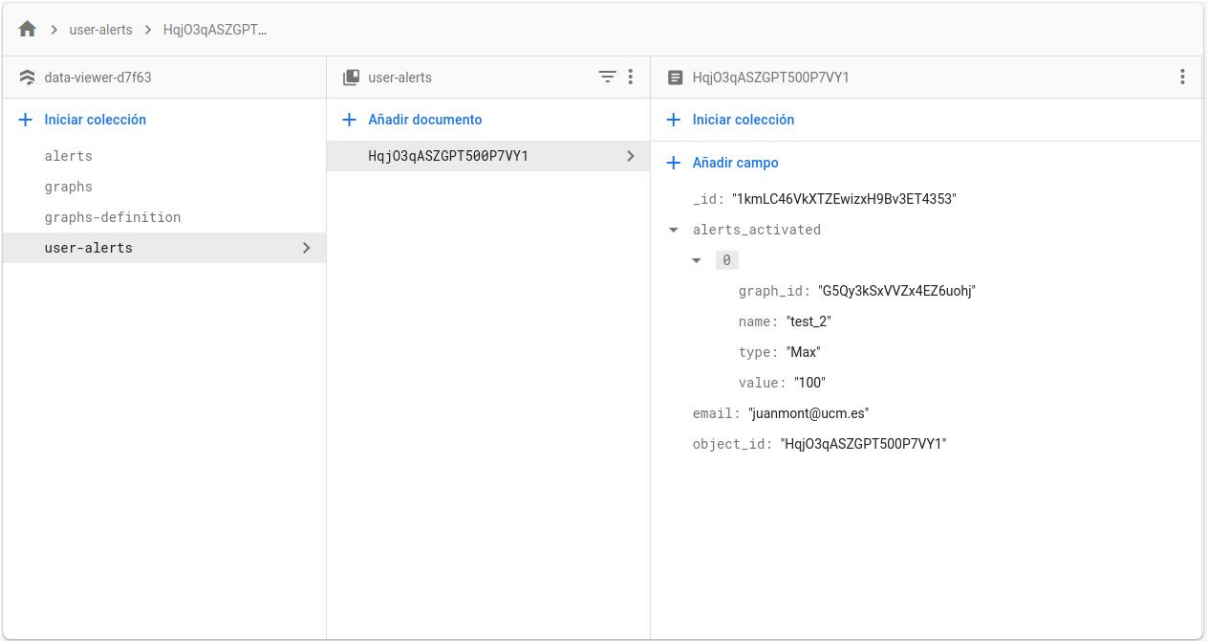

Figura 6.5 - Ejemplo documento user-alerts, Firestore

6.3 - Frontend

Para terminar con los resultados obtenidos tras el desarrollo, se mostrará con imágenes reales como se ve la interfaz web con la que el usuario interacciona con el sistema. Esta interfaz web es muy simple, dedicada a mostrar los datos y las alertas, y con un gran potencial de mejora.

Cuando un usuario entra por primera vez en el portal web, se le ofrece registrarse o loguearse para poder hacer ciertas de las funciones en las que se necesita tener un usarios, como crear un nuevo gráfico o crear una alerta. El sistema de registro y de login es muy sencillo, ya que no busca recoger datos de los usuarios, sino simplemente identificarlos.

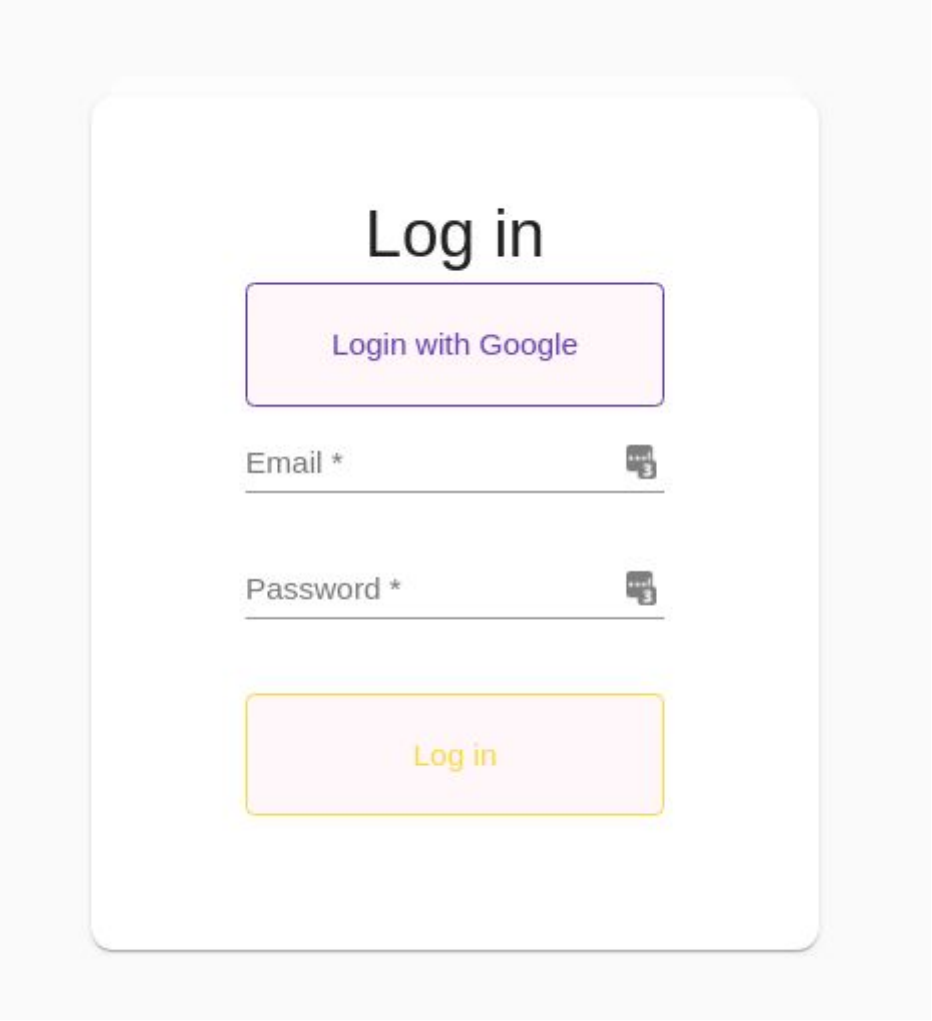

Figura 6.6 - Login, Portal web

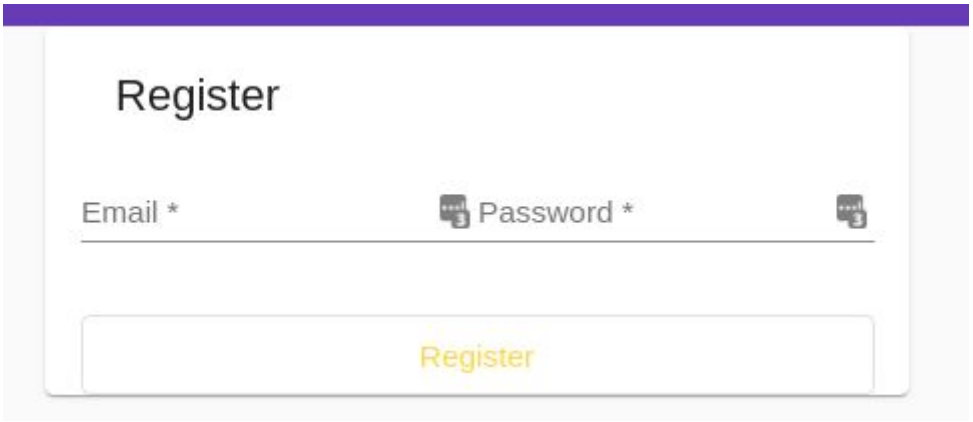

Figura 6.7 - Registro, Portal web

Cuando el usuario ya se ha logueado dentro de la aplicación, la primera pantalla que verá, será la siguiente.

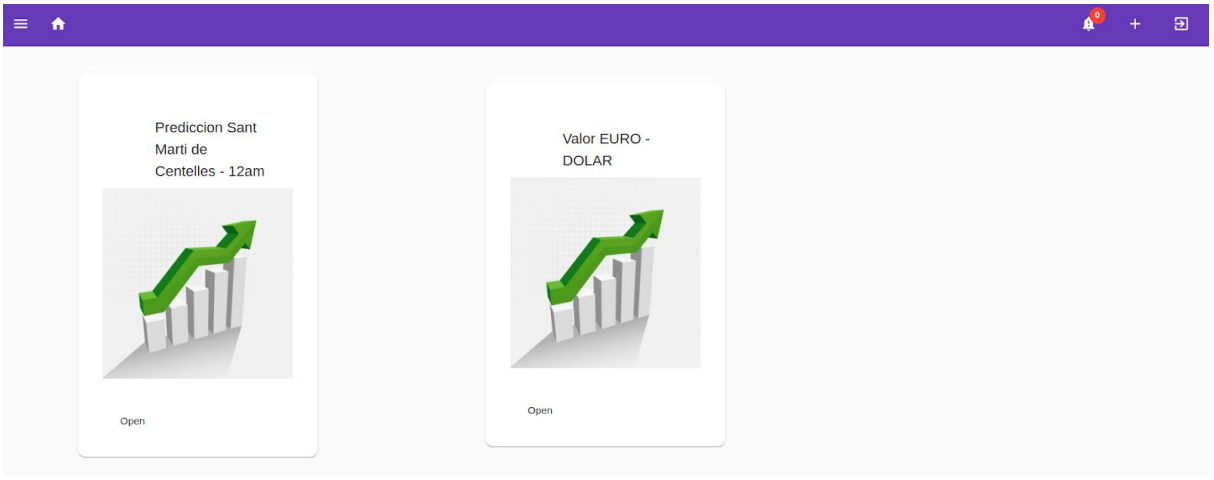

Figura 6.8 - Dashboard, Portal web

Desde esta pantalla podrá filtrar por tipo de gráfico utilizando la hamburguesa de arriba a la izquierda.

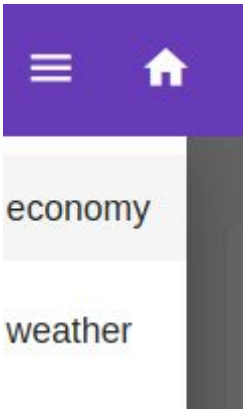

Figura 6.9 - Filtro, Portal web

También podrá ver si tiene alertas activadas o no. Pulsando el icono donde se muestran las alertas, en la campana que se encuentra en la barra superior, podrá acceder al listado de alertas que se han activado.

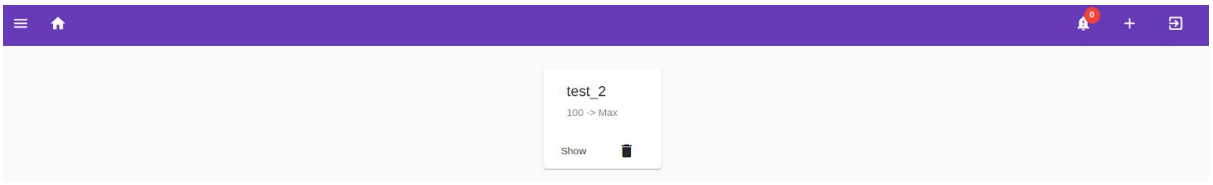

Figura 6.10 - Alertas, Portal web

Desde esta ventana podrá escoger ir a ver el gráfico donde ha saltado la alerta o eliminar la alerta que ha sido activada. Si el usuario escoge ir a ver el gráfico, se le abrirá una página como esta.

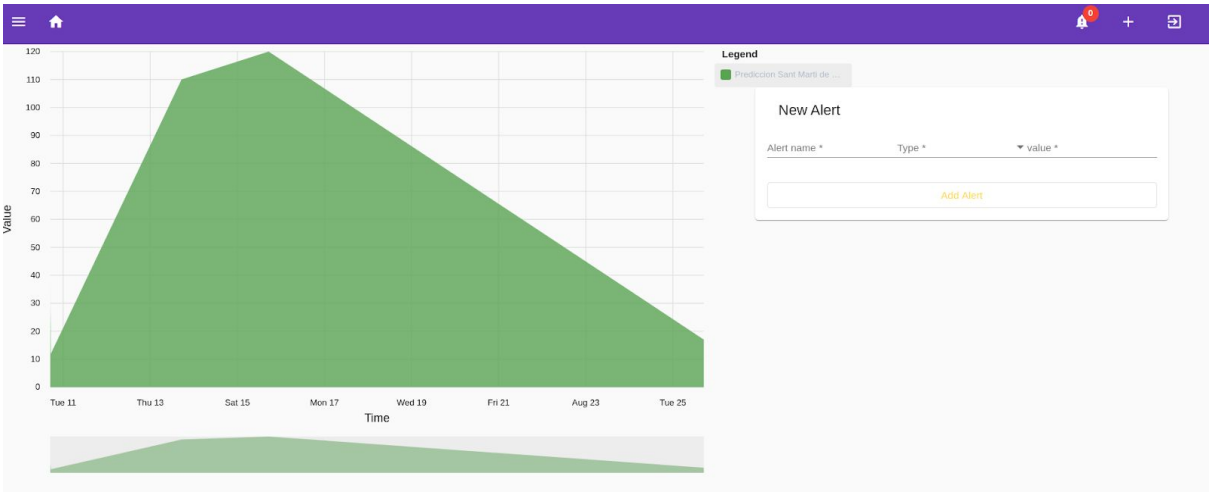

Figura 6.11 - Gráfico, Portal web

Desde esta ventana, puede crear nuevas alertas asociadas a ese gráfico. Si ya existe una alerta igual, el sistema añadirá el identificador del usuario a la alerta ya definida. Otra de las opciones que tiene el usuario desde cualquier pantalla, pulsando en el símbolo + de la barra superior, es ir a crear la descarga de un nuevo gráfico. Tras pulsar el botón se le mostrará la siguiente ventana.

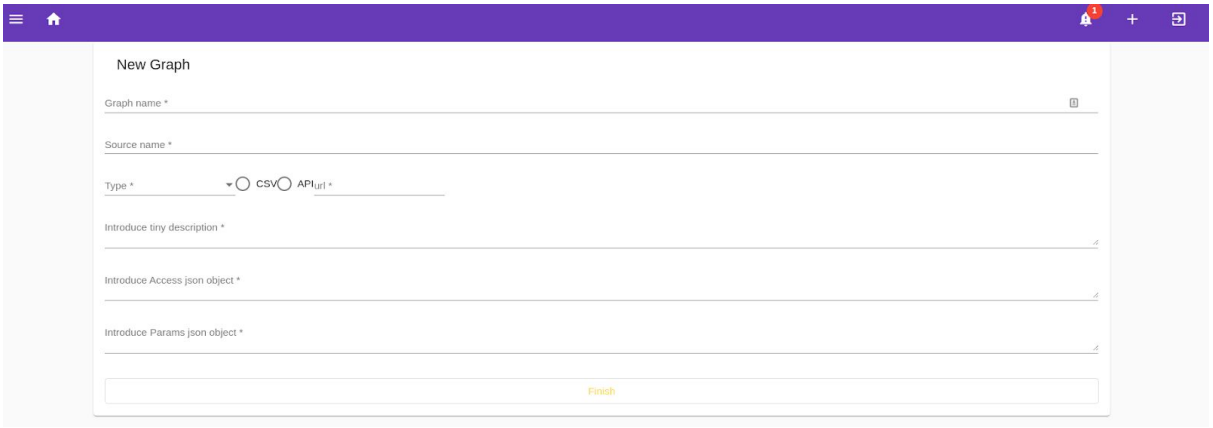

Figura 6.12 - nuevo gráfico, Portal web

6.4 - Pruebas realizadas

Para comprobar que se cumplen todos los requisitos propuestos en el capítulo de Análisis, se han realizado una serie de pruebas para comprobar el cumplimiento de estos requisitos. A continuación se listan las pruebas realizadas para comprobar el cumplimiento de lo propuesto en la parte de Análisis:

- Creación de varias definiciones de descarga:
  - Se ha comprobado que el sistema responde bien cuando se crean muchas definiciones de descarga.
- Comprobación de varias alertas simultáneamente
  - Se han realizado pruebas con varias alertas definidas sobre varios gráficos e introduciendo datos en varios de estos gráficos de forma concurrente. El sistema ha lanzado una función por cada dato nuevo y ha comprobado si se cumplen las alertas.
- Comprobación de si un usuario puede eliminar algún documento.
  - Debido a la imposibilidad de borrar definiciones de gráficos o alertas a través del portal web, un usuario no puede eliminar ningún documento almacenado en Firestore, únicamente puede eliminar su propia suscripción de una alerta o eliminar una alerta activada.

Estas son las principales pruebas realizadas sobre el sistema, se ha comprobado que se cumplen con todos los requisitos propuestos en el capítulo de análisis.

7 - Trabajo futuro

La realización de este proyecto ha solventado los requisitos que se proponía, pero estos requisitos solo son una pequeña parte de las funcionalidades que puede tener un herramienta profesional de análisis de datos. Es por esto, que se ha decidido incluir en esta memoria un apartado de trabajos o pasos futuros que se deben seguir para continuar evolucionando la aplicación. Las mejoras del sistema no están centradas en un único módulo del sistema, sino que se pueden realizar mejoras en cada uno de los componentes descritos anteriormente. A continuación se pasará a describir cada una de estas mejoras, junto con el problema que solventan.

Comenzando por el sistema backend montado sobre Cloud Firestore, como se detalla en el capítulo de diseño, el sistema podría tener problemas si se alcanza el límite de invocaciones en un segundo. Si se llegase a producir el límite de funciones que se lanzan 1 segundo, una de las soluciones sería crear una cola de mensajes para que almacene los mensajes que no se van a poder leer por límite de funciones. Para poder crear funciones a través de la cola de mensajes, se necesitará desarrollar un componente que lance más funciones cuando el número de funciones, ejecutándose concurrentemente, disminuya. Otra de las soluciones posibles, es crear un sistema de compartición de contadores, de tal forma que se reduzca el número de invocaciones. [16]

Otra de las mejoras posibles, pero esta vez el módulo backend montado sobre Python, sería lanzar un componente backend por cada tipo de gráfico que se tenga en el sistema, es decir, uno para los datos de tipo financiero, otro para los datos de tipo meteorológico, y así por cada tipo. A parte de esta mejora sobre el servidor backend, para solucionar la limitación del intervalo de ejecución fijo, se podría realizar un desarrollo para que el intervalo entre cada descarga puede variar dependiendo del tipo de gráfico o incluso dependa de un campo dentro de la definición del gráfico.

Para terminar con las mejoras en el sistema, se plantea una posible mejora sobre el módulo frontend, esta mejora consistiría *en* incluir una serie de plugins que el usuario se pudiese descargar y que solo funcionasen con los gráficos de un tipo determinado, por ejemplo un plugin que solo sirva con gráficos de tipo weather.

8 - Conclusiones

En este capítulo final, se relatan cuáles han sido los aprendizajes e impresiones personales obtenidos durante la realización del proyecto, así como las conclusiones tras acabar el proyecto.

El sistema que se ha creado durante la realización de este proyecto, permite a los usuarios introducir dentro de un mismo portal web, datos de distinto tipo y aunarlos dentro de un mismo lugar, lo cual facilita la visualización y el acceso a estos datos. Además al ser el usuario el que define la descarga de los datos, el sistema no se encuentra limitado a los datos definidos por los administradores, sino que son los propios usuarios los que deciden qué datos quieren que se muestren en las gráficas. Esto es una ventaja frente a las plataformas expuestas en el capítulo de estado del arte, las cuales se centraban en un tipo de dato. En este caso, la plataforma es tan genérica que permite incluir cualquier datos y crear alertas sobre el mismo.

A partir de esta parte dentro de la memoria se hablará en un lenguaje más informal y personal, ya que se habla de las conclusiones obtenidas por el realizador del proyecto.

Desde el comienzo del proyecto, mi principal busqueda fué encontrar un sistema que aunase en un único lugar, datos de distintas fuentes y categorías, ya que por lo que había investigado, la mayor parte de los sistema de muestra de información, se centran en mostrar datos de un tipo concreto, ya sean datos económicos, datos meteorológicos o incluso datos culturales. Mi intención era crear las bases de una plataforma que permitiese a los usuarios tener gráficos de distinto tipo en un único lugar. Esto puede ser un arma de doble filo, porque la generalización de las aplicaciones es buena para tener en un mismo lugar cosas comunes de todos los sistemas, pero también tiene el inconveniente de no tener las herramientas específicas de cada uno de los gráficos. Por ello las herramientas de muestra de datos económicos están muy centradas en este tipo de datos, porque crean extras dentro de la aplicación que solo sirven para utilizarse con este tipo de datos.

Por lo tanto una de las conclusiones a las que he llegado durante la realización de este proyecto es que su uso más común debe ser para crear alertas sobre datos, no sirve para hacer cosas específicas por cada tipo de dato.

Este proyecto no solo me ha servido para comprobar la gran variedad de aplicaciones de muestra de datos que podemos encontrar en internet, o para comprobar que la mayor parte de sistemas de obtención de datos proveen de un API que permite acceder a ellos. Mi principal enseñanza que me llevo de este proyecto, es saber crear y gestionar una aplicación en Angular, con acceso a un servicio montado en la nube, en este caso Firebase. Antes de la realización de este proyecto, solo había leído algo por encima cosas sobre Angular y había realizado una aplicación muy sencilla con Firebase. Pero este proyecto me

ha hecho comprender mucho más la infraestructura que proporciona Angular, así como montar un proyecto con una envergadura más o menos media en este Framework.

De este proyecto me llevo muchas enseñanzas y aprendizajes, sobre todo a la hora de montar un sistema escalable, con distintos módulos que permitan desacoplarse e incorporar nuevos módulos.

9 - Bibliografía

[1]A. Freeman, *Pro Angular 9*. [United States]: Apress, 2020.

[2]"Angular", *Angular.io*, 2020. [Online]. Available: https://angular.io/. [Accessed: 26- Aug- 2020].

[3]*Positronx.io*, 2020. [Online]. Available: https://www.positronx.io/build-progressive-web-app-pwa-with-angular/. [Accessed: 26- Aug- 2020].

[4]H. Yahiaoui, *Firebase cookbook*. 2017.

[5]"Documentación | Firebase", *Firebase*, 2020. [Online]. Available: https://firebase.google.com/docs?hl=es. [Accessed: 26- Aug- 2020].

[6]A. Minuto et al., "Monetización de los datos: la importancia de los datos para las empresas", *La Vanguardia*, 2020. [Online]. Available: https://www.lavanguardia.com/economia/20180104/434058696189/monetizacion-datos-empresas-the-valley.html. [Accessed: 29- Aug- 2020].

[7]"Bloomberg L.P.", *Es.wikipedia.org*, 2020. [Online]. Available: https://es.wikipedia.org/wiki/Bloomberg_L.P. [Accessed: 29- Aug- 2020].

[8]V. Bank →, "Bloomberg: ¿qué es y para qué se utiliza? - El blog de SelfBank by Singular Bank", *El blog de SelfBank by Singular Bank*, 2020. [Online]. Available: https://blog.selfbank.es/bloomberg-se-utiliza/. [Accessed: 29- Aug- 2020].

[9]"Gráficos Acciones gratuitos, Cotizaciones de Acciones e Ideas de Trading", *TradingView*, 2020. [Online]. Available: https://es.tradingview.com/. [Accessed: 29- Aug- 2020].

[10]"¿Qué es Tradingview y cómo funciona?", *Escuela de Traders*, 2020. [Online]. Available: https://escueladetraders.es/que-es-tradingview-y-como-funciona/. [Accessed: 29- Aug- 2020].

[11]"AEMET OpenData", *Opendata.aemet.es*, 2020. [Online]. Available: https://opendata.aemet.es/centrodedescargas/inicio. [Accessed: 29- Aug- 2020].

[12]"El Tiempo", *Eltiempo.es*, 2020. [Online]. Available: https://www.eltiempo.es/. [Accessed: 29- Aug- 2020].

[13]"Quotas and limits | Firebase", *Firebase*, 2020. [Online]. Available: https://firebase.google.com/docs/functions/quotas. [Accessed: 29- Aug- 2020].

[14]A. Moreau, "Eltiempo: la web meteorológica más visitada de España - Revista Náutica", *es.NauticWebNEWs.com*, 2020. [Online]. Available: http://es.nauticwebnews.com/751/eltiempo-la-web-meteorologica-mas-visitada-de-

espana/#:~:text=Eltiempo%20es%20la%20web%20meteorol%C3%B3gica,innovadores%20servicios%20de%20predicci%C3%B3n%20meteorol%C3%B3gica. [Accessed: 30- Aug- 2020].

[15]"Service Level Agreement for Hosting and Realtime Database | Firebase", *Firebase*, 2020. [Online]. Available: https://firebase.google.com/terms/service-level-agreement. [Accessed: 30- Aug- 2020].

[16]"Distributed counters | Firebase", *Firebase*, 2020. [Online]. Available: https://firebase.google.com/docs/firestore/solutions/counters. [Accessed: 30- Aug- 2020].