FIREBASE

PARA CONCURSOS MUSICALES

Juan Casado Ballesteros

Host de Firebase: https://advancedfirebase-2e194.web.app

Vídeo explicativo: https://www.youtube.com/watch?v=p1wtn27bxJw

GitHub: https://github.com/JuanCasado/ADVANCED_DATABASES/tree/master/P2

ÍNDICE

INTRODUCCIÓN

LA APLICACIÓN DESARROLLADA

A lo largo de esta práctica se ha utilizado Firebase para crear una aplicación web que permita gestionar concursos musicales. La aplicación permite crear concursos que a su vez se componen de programas. Cada programa y cada concurso tiene una imagen, un nombre y una descripción.

En los concursos participan los concursantes que para cada programa reciben una puntuación dada por el administrador. Los concursantes pueden también formar duetos para participar en los programas. Una vez que el programa es creado los usuarios registrados pueden votar a los concursantes logrando una puntuación igual a la que el administrador otorgó al concursante.

Finalmente, los concursantes según el número de votos y los usuarios según la puntuación obtenida al votar serán ordenados en rankings. Los rankings, así como los concursos y programas serán visibles por todos, usuarios registrados y no registrados.

Los usuarios registrados pueden hacerlo con un email, mediante Google o mediante ambos. Estos tendrán un perfil del cual podrán controlar qué será visible para otros usuarios. El perfil estará formado por un nombre, un email, una descripción y una foto. La foto del perfil podrá ser la del perfil de Google, una que el usuario proporcione o una aleatoria extraída de internet.

Adicionalmente se ha creado un sistema de escalo de permisos por el que un administrador superior puede ascender a un usuario registrado al cargo de administrador. Los administradores no superiores no pueden ascender de permisos a otros usuarios, pero sí pueden crear concursos y programas.

FIREBASE

En esta práctica se ha utilizado la plataforma de desarrollo Firebase. Firebase es una parte de la plataforma de desarrollo cloud de Google que permite implementar aplicaciones web y móviles dinámicas, responsivas, escalables y fácilmente configurables. Esta plataforma pone a nuestra disposición gran cantidad de elementos en un formato Software as a Service (SaS).

Esto se traduce en que no tendremos que preocuparnos de mantenerlos ni de mejorarlos, Google se encargará de ello. Esto impone una relación de confianza entre el proveedor de los servicios y el usuario de estos pues el usuario no tendrá forma de responder o actuar ante fallos del sistema pues este no le pertenece, aunque de otro modo tiene garantizado que dichos problemas le serán resueltos en caso de ocurrir.

Dentro de esta plataforma se encuentran gran cantidad de sistemas que proporcionan funcionalidad como autenticación de usuarios, bases de datos documentales distribuidas, bases de datos documentales en tiempo real, sistemas de almacenamiento de blobs, servicios de computación cloud (functions), servicios de mensajearía, configuración remota de aplicaciones, sistemas de monitorización y de colección de datos analíticos…

Estos servicios no están disponibles para todas las plataformas. Las plataformas móviles tienen la mayoría, pero por el contrario la plataforma web carece de muchos como el servicio de machine learning. Principalmente hemos utilizado dos grupos de servicios, los relacionados con autenticación, hosting y computación cloud (functions) y los relacionados con bases de datos documentales, en tiempo real y de almacenamiento de blobs.

Como hemos dicho Firebase se integra dentro de la plataforma de Google Cloud pudiendo interoperar con otras partes de esta como Big Table o Compute Engine.

TRABAJO REALIZADO

Se ha cumplido todos los requisitos proporcionados en el documento de descripción de la práctica sin ninguna limitación. Los usuarios pueden poner cualquier foto como foto de perfil, la seguridad de acceso a los datos se realiza tanto en la propia web como en las reglas de la base de datos, las votaciones y los rankings se crean reduciendo la información que es necesario transmitir…

La aplicación creada es una aplicación web que utiliza la librería React para manipular los componentes del DOM. Esto permite reducir el número de actualizaciones de la interfaz aumentando por tanto su rendimiento. Para combinar React con Firebase se debe poner especial atención al grafo de propagación de la información. Debemos delimitar con antelación los elementos que se verían modificados cuando un valor en la base de datos cambie. De este modo podremos observar por subscripción estos valores actualizado la interfaz con ellos.

Se ha intentado utilizar la mayor cantidad posible de servicios proporcionados por Firebase. No obstante, algunos no han podido ser utilizados pues no estaban disponibles para la plataforma web. Este es el caso de las notificaciones remotas, la configuración remota de aplicaciones, los mensajes cloud y la plataforma de machine learning que están solo disponibles para móvil y no para web.

Se listan a continuación cada uno de los componentes utilizados incluyendo para cada uno de ellos una pequeña descripción de qué son, detallando cómo se han utilizado para resolver los problemas que la aplicación a desarrollar plantea y las peculiaridades en el uso de cada uno de ellos.

FIRESTORE

Actualmente la plataforma de Firebase proporciona tres métodos de almacenamiento de datos. Storage para almacenar blobs y dos bases de datos documentales: Firestore y Firebase en tiempo real. Aunque la base de estas dos bases de datos sea similar; ambas son bases de datos cloud en las que se accede en SaS y que permiten almacenar documentos en formato JSON sus características son muy diferentes.

Firestore es la última base de datos que se ha incorporado al ecosistema de Firebase. Esta base de datos documental utiliza dos estructuras fundamentales, los documentos y las colecciones.

DOCUMENTOS

Un documento es la unidad mínima de información que se puede obtener en una consulta. Siempre que accedamos a Firestore para obtener información que no sean metadatos de los documentos será un documento completo. Cunado borremos información se hará también a nivel de documento. Las actualizaciones de documentos no obstante se pueden realizar a nivel de atributo.

Dentro de un documento encontraremos datos en formato JSON, pares de clave valor donde cada valor puede ser un número, una cadena, un booleano, un vector de valores u otro objeto JSON.

COLECCIONES

Los documentos se organizan en colecciones. Las colecciones pueden verse como un vector de documentos pues podemos iterar sobre ellos, filtrarlos y ordenarlos por los valores que contienen. Pero también podemos verlos como un objeto JSON de pares clave valor que se entienden como relacionados con un documento, pero no pertenecientes a él.

Los esquemas de seguridad se aplican sobre todos los documentos pertenecientes a una misma colección en lugar de a documentos aislados.

En principio los valores almacenados dentro de los documentos de una colección pudieran no tener ninguna similitud en esquema. Nada nos obliga a ello. Pero es habitual que esto suceda pues así serán más fáciles de manejar.

OTRAS CARACTERÍSTICAS

Firestore, a diferencia de Firebase en tiempo real incorpora mecanismos de escalado automático. Esto quiere decir que podremos indicar que nuestra base de datos se ubique en varias regiones y los datos se repartirán de la forma más conveniente por ellas. Adicionalmente con una única instancia de la base de datos podremos almacenar más cantidad de información y soportar mayor número de peticiones y conexiones simultáneas.

Para utilizar Firestore no debemos de preocuparnos en gran medida sobre minimizar el ancho de banda como ocurren Firebase en tiempo real. En su lugar debemos de centrarnos en reducir el número de documentos accedidos. Debemos de procurar que la información se reparta en el menor número de documentos posible siempre y cuando toda la información almacenada en el documento comparta unas mismas reglas de seguridad.

ORGANIZACIÓN DE LOS DATOS

Firestore es la base de datos que ha sido usada más intensivamente en este proyecto. En ella se almacenan la mayor parte de los datos siguiendo el siguiente esquema:

USUARIOS

Debido a que la seguridad se aplica a nivel de colección para garantizar que los datos de los usuarios pudieran ser en parte privados y en parte públicos se han utilizado dos colecciones distintas. Una para contener los datos públicos (todos en

un mismo documento) y otra para contener los privados (también en un mismo documento). El esquema resultante será similar al siguiente.

/Usuarios
/Usuarios/{usuario}/Private/{data}
/Usuarios/{usuario}/Public/{data}

La clave por la que se identifica a los usuarios es el uid que se proporciona de forma única a cada uno de ellos en el token de acceso a la aplicación. Se detallará esto en la parte de autenticación y seguridad.

CONCURSOS

Para almacenar la información relativa a los concursos se utiliza un documento en el que se guarda su nombre, descripción y ubicación de la imagen. Este documento puede ser visto por todos, pero editado solo por los administradores. A su mismo nivel se ubican las colecciones de programa, concursantes, usuarios que han participado en las votaciones de ese concurso y los programas del concurso.

/Concursos/{concurso}
/Concursos/{concurso}/Programas/{programa}
/Concursos/{concurso}/Usuarios/{usuario}/Programa/{votos}
/Concursos/{concurso}/Concursantes/{concursante}

Como se puede ver cada concursante y programa tiene su propio documento. Los documentos de los usuarios almacenan la puntuación de estos y el número de votos que les quedan por poder realizar en el concurso. Los votos de los usuarios son un documento por programa. Se ha elegido esta organización frente a incluir los votos como documentos de los programas pues serán los usuarios los que más veces los consulten de modo que serán más fáciles de extraer junto al resto de sus datos.

MENSAJES

Los mensajes se incluyen dentro de cada programa como una estructura recursiva. Un mensaje es un documento con una colección de mensajes (las contestaciones) que puede estar vacía en caso de no existir.

/Concursos/{concurso}/Programas/{programa}/Mensajes/{mensaje}

PETICIONES

El esquema minimiza el número de documentos a acceder para realizar cada acción. Tal y como todo ha sido organizado en cada documento se encuentra toda y solo la información necesaria para realizarla.

Esto se cumple en todos los casos excepto en el de los mensajes. Cada mensaje ocupa un único documento lo cual a primeras impresiones es un gasto innecesario de peticiones. Una posible mejor organización hubiera sido que cada hilo de mensajes ocupara cada documento. No obstante, esto hubiera implicado mayor trabajo para implementar su almacenamiento por lo que se ha decidido hacer tal y como se ha explicado.

A la hora de realizar las peticiones se ha favorecido el mecanismo de subscripción frente al de extracción única. El mecanismo de subscripción nos permite recibir notificaciones cada vez que los datos almacenados cambien para poder actualizar la interfaz de modo apropiado.

FIREBASE EN TIEMPO REAL

Esta es la base de datos documental original de la plataforma Firebase. Comparte con Firestore muchos principios como ser una base de datos cloud a la que se accede por SaS o la posibilidad de recibir notificaciones cuando los datos cambian por un mecanismo de subscripción.

No obstante, su diseño y forma de uso son muy distintos. Firebase en tiempo real no es una plataforma escalable, al menos no tanto como Firestore. No puede ser desplegada en múltiples regiones como una única base de datos centralizada si no que deben desplegarse en varias instancias aisladas pudiendo estar cada una en una región distinta. Adicionalmente en el uso de Firebase en tiempo real debemos centrarnos en minimizar la cantidad de información transmitida por petición y no tanto el número de peticiones realizadas.

Se puede ver que esta base de datos fue diseñada antes que Firestore pues tiene algunos componentes que podrían verse como legacy. Es decir, fueron diseñados de una forma que luego ha sido mejorada (en Firestore) pero que debe mantenerse para que los clientes que la utilizan no pierdan el servicio. Esto se ve claramente en el formato de las reglas de acceso que no siguen el formato IAM como sí lo hacen Storage o Firestore.

ORGANIZACIÓN DE LOS DATOS

Los datos de Firebase en tiempo real se organizan en forma de un único árbol JSON en lugar de en documentos y colecciones como ocurre en Firestore. Esto en principio puede no parecer demasiado importante ya que como hemos dicho las colecciones pueden verse como pares clave valor al nivel del documento al que pertenecen. Sin embargo, sí es importante debido a las implicaciones que tiene en la asignación de permisos de acceso. Si en Firebase en tiempo real un usuario puede acceder a un nodo podrá también acceder a todos los nodos desde él a las hojas del árbol JSON. En Firestore esto no era así; un usuario pude acceder a un documento y no tener permisos para acceder a las colecciones de ese nivel.

Para nuestra aplicación se utiliza Firebase en tiempo real solo para almacenar si los usuarios son administradores o no. Mediante esto podemos de una forma rápida y ligera almacenar un booleano por usuario que será accesible en lectura solo por el usuario al que pertenece y en escritura por Functions. Será desde Functions con los permisos de administrador habilitados desde dónde gestionaremos esta base de datos.

Cada usuario se subscribirá al nodo correspondiente de la base de datos para saber si tiene permisos de administrador o no. No obstante, lo que esté escrito en la base de datos, es solo un reflejo de la información que el usuario porta en su token de identidad (explicaremos esto más en detalle en la parte de autenticación).

En definitiva, es la información de los tokens de identidad la que realmente garantiza permisos de administrador. Pero modificar un token de un usuario no le notifica en tiempo real que sus permisos hayan cambiado. Es por ello por lo que Firebase en tiempo real juga un papel fundamental permitiendo al usuario saber cuando sus permisos han cambiado. También desde el punto de vista de un administrador es más fácil controlar los permisos otorgados a otros usuarios por medio de functions y de Firebase en tiempo real que de la API de administración pues esta es mucho más lenta. Tanto es así que esta es la forma recomendada por Google en su documentación para gestionar el escalado de permisos.

PETICIONES

Las peticiones a Firebase en tiempo real se realizan indicando el path en el árbol JSON en el que se encuentran los datos a los que deseamos acceder. Del mismo modo que con Firestore podemos subscribirnos a los datos para ser notificados cuando cambian. Es justo ese el comportamiento implementado en la aplicación.

STORAGE

Storage es la última base de datos que actualmente forma parte del ecosistema Firestore. Esta es una base de datos que permite almacenar Blobs de información; es decir, archivos en formato binario fin un esquema concreto.

Utilizaremos esta base de datos para almacenar las imágenes de los usuarios. Nos aprovecharemos de que Storage almacena junto a cada blob un documento JSON de metadatos para implementar a partir de ellos las reglas de seguridad. Los archivos son organizados dentro de un árbol como si fuera un sistema de almacenamiento de archivos tradicional.

Esta base de datos expone la información que contiene mediante una API basada en peticiones https get (a diferencia de las otras dos que utilizan una combinación de peticiones https post y websockets) Esto nos permite extraer los datos por medio de su API o por medio de una API genérica. Debido a que por defecto los tag img de html realizan una petición get sobre la ruta proporcionada en el atributo src para extraer la imagen que mostrarán podremos enlazar nuestra aplicación con Stogare de esa forma. De este modo extraer las imágenes de Storage se hace trivial.

Para subir las imágenes a Storage utilizamos el mecanismo standard de html para la extracción de Blobs mediante un callback a JavaScript con el contenido binario del archivo. Realizamos algunas comprobaciones como que el archivo sea una imagen para rellenar los metadatos de forma adecuada y que puedan atravesar las restricciones impuestas en las reglas de Storage.

FUNCTIONS

Functions es el mecanismo que propone Firebase para realizar computación cloud. El código que sea subido a Functions será escalado de forma automática, actualizado por medio de las técnicas de Rolling update y supervisado por Google para reiniciarlo en caso de fallo. Functions nos proporciona también mecanismos de log para controlar el correcto funcionamiento de nuestro código.

Functions se implementa como un servidor express distribuido con el que podremos comunicarnos por medio de peticiones https post. Functions incorpora seguridad cors redirigida al servicio de hosting de Firebase. Esto implica que cuando el servicio de hosting esté activo solo podremos enviar peticiones desde él y no desde una instancia local de la página web. No obstante, esta seguridad puede ser burlada por medio de la modificación de las cabeceras de autenticación de los paquetes enviados al servidor.

La principal limitación de Functions es que en su versión gratuita ofrece alta latencia en las peticiones. Estas se procesan rápidamente, pero tardan en comenzar a procesarse. Se entiende que en otras opciones de pago esto no pasará pues de otro modo la latencia es realmente insatisfactoria para usar este servicio principalmente pues desde otros servicios como cloud engine no se observa un comportamiento similar.

Las principales fortalezas de Functions son la poca necesidad de mantenimiento, pues con subir el código todo lo demás es gestionado de forma automática y la gran integración con el resto del ecosistema de Firebase. Desde Functions podremos acceder a las bases de datos de Firestore y Firebase en tiempo real como un administrador; es decir, podremos escribir y leer cualquier nodo y documento. Esto combinado con podernos subscribir a los datos almacenados en las bases de datos nos permite gran control sobre ellos.

Utilizando Functions se han implementado las votaciones de la aplicación, los usuarios proporcionan a functions los id de los concursantes a los que desean votar y Functions se encargará de que comprobar que el usuario no haya votado antes y de que el voto sea válido. Posteriormente el voto será añadido al listado de votos del usuario en el concurso y programa concreto actualizando los rankings de usuarios y de concursantes acorde a los puntos que el administrador les proporcionó.

Debido a que excepto en las votaciones no se han encontrado más restricciones sobre el formato de los datos que no pudieran implementarse con las reglas de la propia base de datos el resto de los accesos a la base de datos se realizan por medio de las API de cada una de ellas desde el cliente como se recomienda en la documentación. No obstante, para este caso, debido a la complejidad de las comprobaciones necesarias se ha optado por esta solución.

Otras acciones implementadas en Functions son las relacionadas con el escalado de permisos. Debido a que Functions puede realizar acciones de administrador de Firebase desde allí podemos editar los metadatos que cada usuario tendrá en su token de identificación. En los tokens se los administradores incorporaremos un flag que les identifica como tales. Adicionalmente si nuestro correo coincide con uno que esté en un vector almacenado de forma segura en Functions tendremos permisos de administrador superior. Esto nos permite obtener listados de los usuarios de la aplicación y otorgarles o retirarles permisos de administrador.

AUTENTICACIÓN Y SEGURIDAD

La autenticación que proporciona Firebase adopta un esquema tradicional de autenticación en tres pasos. Pudiendo ser extensible a un esquema de cuatro pasos si añadimos doble factor de autenticación.

En primer lugar, los usuarios deberán registrase con uno de los proveedores de autenticación compatibles. Estos son los principales proveedores de autenticación Google, Facebook, GitHub, Apple… Aunque también podrán hacerlo mediante un email y una contraseña. Posteriormente su forma de acceso deberá ser validada por el proveedor lo cual se traduce en introducir el usuario, la contraseña y opcionalmente también en haber verificado el email.

El siguiente paso consiste en obtener un token de autorización el cual es almacenado el local storage dentro de la base de datos del navegador. Este token está cifrado de forma propietaria por Firebase de modo que no podemos inspeccionar su contenido y lo que nos permite es obtener tokens de identificación.

El último paso es utilizar los tokens de identificación que son token que siguen el standard JWT. Esto implica que su contenido es vulnerable (las comunicaciones con Firebase están cifradas con SSL de modo que esto no es relevante) pero a pesar de ello es trazable. Es decir, si se ha modificado podremos saber que lo ha sido de modo que podremos rechazar el token. Además, tienen in periodo de validad de modo que antes de que caduquen deberán actualizarse haciendo uso del token de autorización.

Dentro del token de identificación podemos introducir metadatos como en nuestro caso si un usuario es administrador o no. En estos tokens Firebase ya introduce otra información adicional como un identificador único para los usuarios por medio del cual los identificamos también las bases de datos, el nombre del usuario o su email. Los metadatos pueden usarse también en las reglas de seguridad de la base de datos.

HOSTING

Firebase nos permite subir los archivos que forman nuestra página web a sus servidores siendo estos accesibles desde ellos por medio de una dirección url. El servidor que nos permite acceder a los archivos consta con cifrado SSL de modo que podemos acceder a la página web por medio de https.

Investigando más a fondo en el servicio de hosting podemos ver que si enviamos una petición http al puerto 80 en lugar de https al 443 se nos redirige al contenido protegido por https. Si bloqueamos los servicios de ssl de nuestro ordenador y repetimos la petición no se nos proporciona contenido de vuelta de modo que tenemos la certeza de que solo se podrá acceder a nuestra página web por medio de un canal cifrado. (Esto mismo sucede con Functions).

Para poder hostear nuestra página web solo debemos indicar por medio de qué ruta se debe proporcionar cada archivo. En nuestro caso debido a que la página web creada consta con su propio enrutador solo debemos redirigir el tráfico a él siendo este el que se encargue de realizar todo el trabajo.

Debido a que está limitado el tamaño de los archivos a hostear es recomendable aplicar sobre estos un formato min, así como reducir el tamaño y cantidad de las imágenes que se suban. Si es posible es preferible subir estas a Storage y acceder a ellas por medio de peticiones https get como hemos explicado anteriormente.