EBook Gratis

APRENDIZAJE firebase

Free unaffiliated eBook created from
Stack Overflow contributors.

#firebase

Tabla de contenido

Acerca de

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: firebase

It is an unofficial and free firebase ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official firebase.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Capítulo 1: Empezando con base de fuego

Observaciones

Firebase es un Backend como servicio (Baas) muy útil para el desarrollo de aplicaciones móviles.

Proporciona muchas funciones como **autenticación y seguridad** , **bases de datos en tiempo real y almacenamiento de archivos** , **analíticas** , **notificaciones push** , **AdMod** y muchas otras.

Proporciona el SDK para **Android, iOS, Web, NodeJS, C ++ y Java Server.**

Versiones

Plataforma SDK	Versión	Fecha de lanzamiento
Firebase JavaScript SDK	3.7.0	2017-03-01
Firebase C ++ SDK	3.0.0	2107-02-27
Firebase Unity SDK	3.0.0	2107-02-27
Firebase iOS SDK	3.14.0	2017-02-23
Firebase Android SDK	10.2	2017-02-15
Firebase Admin Node.js SDK	4.1.1	2017-02-14
Firebase Admin Java SDK	4.1.2	2017-02-14

Examples

Agrega Firebase a tu proyecto de Android

Aquí se detallan los pasos necesarios para crear un proyecto Firebase y conectarse con una aplicación de Android.

Agrega Firebase a tu aplicación

1. Cree un proyecto de Firebase en la consola de Firebase y haga clic en **Crear nuevo proyecto** .
2. Haga clic en **Agregar Firebase a su aplicación de Android** y siga los pasos de configuración.

3. Cuando se le solicite, ingrese el **nombre del paquete de** su **aplicación** .
   Es importante ingresar el nombre del paquete que usa tu aplicación; esto solo se puede configurar cuando agrega una aplicación a su proyecto Firebase.

4. Para agregar el certificado de firma de depuración SHA1 que se **requiere para los enlaces dinámicos, las invitaciones y el soporte** de `Gradle` **sesión de Google en Auth,** vaya a su proyecto en Android Studio, haga clic en la pestaña `Gradle` en el lado derecho de su ventana, haga clic en el botón `Refresh` , vaya `project(root)` -> `Tasks` -> `android` -> `signingReport` . Esto generará **MD5** y **SHA1** en la pestaña `Run` . Copia pegar SHA1 en la consola de base de fuego.

5. Al final, descargará un archivo `google-services.json` . Puedes descargar este archivo de nuevo en cualquier momento.

6. Si aún no lo ha hecho, copie esto en la carpeta del módulo de su proyecto, normalmente app /.

El siguiente paso es agregar el SDK para integrar las bibliotecas Firebase en el proyecto.

Agrega el SDK

Para integrar las bibliotecas de Firebase en uno de sus propios proyectos, debe realizar algunas tareas básicas para preparar su proyecto de Android Studio. Es posible que ya hayas hecho esto como parte de agregar Firebase a tu aplicación.

1. Agregue reglas a su archivo `build.gradle` nivel `build.gradle` , para incluir el **complemento de servicios de google** :

```
buildscript {
    // ...
    dependencies {
        // ...
        classpath 'com.google.gms:google-services:3.0.0'
    }
}
```

Luego, en su módulo de archivo Gradle (generalmente `app/build.gradle`), agregue la línea de aplicación del complemento en la parte inferior del archivo para habilitar el complemento de Gradle:

```
apply plugin: 'com.android.application'

android {
  // ...
}

dependencies {
  // ...
  compile 'com.google.firebase:firebase-core:9.4.0'
}
```

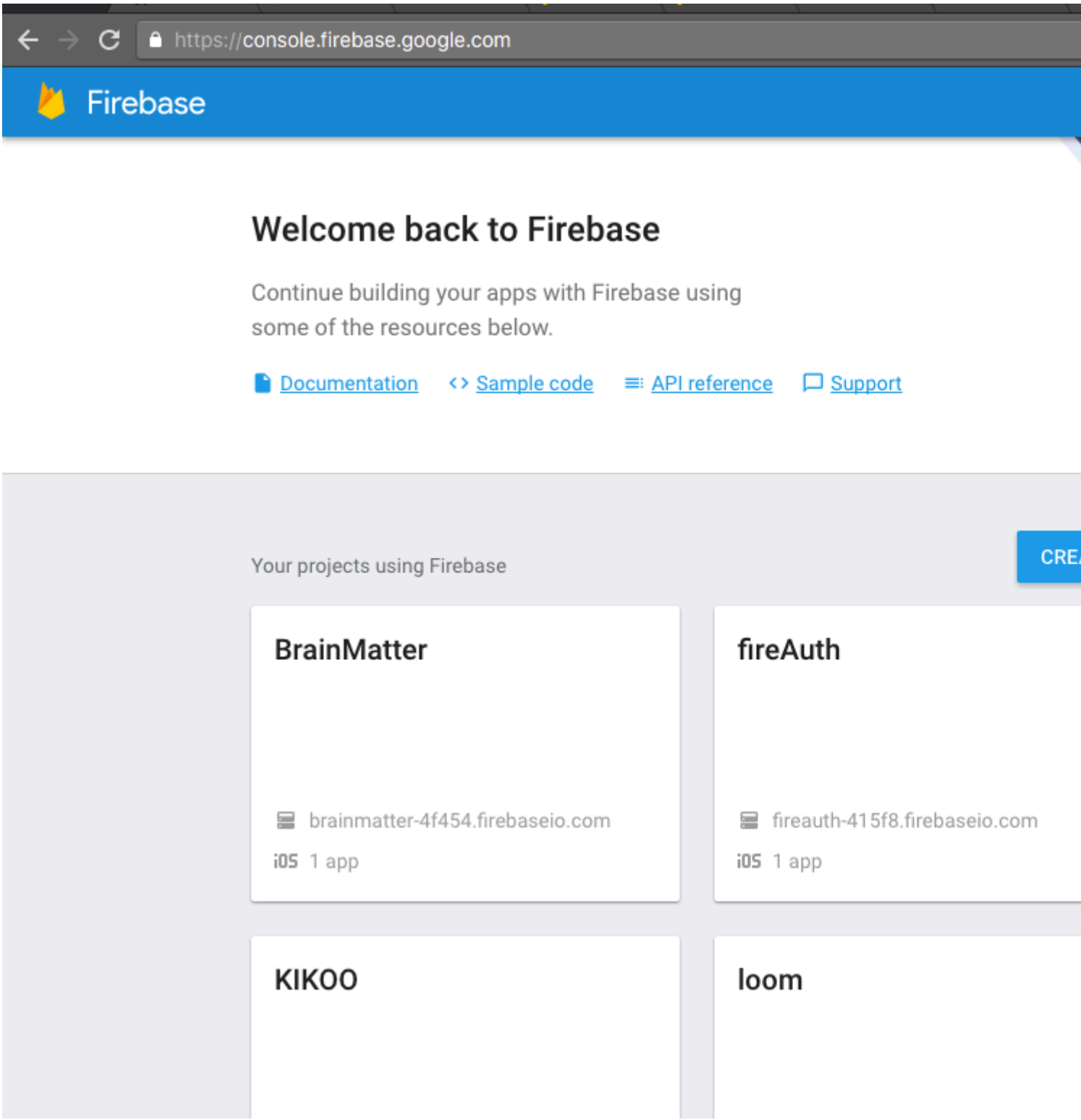
```
// ADD THIS AT THE BOTTOM
apply plugin: 'com.google.gms.google-services'
```

El último paso es agregar las dependencias para el SDK de Firebase usando una o más **bibliotecas disponibles** para las diferentes características de Firebase.

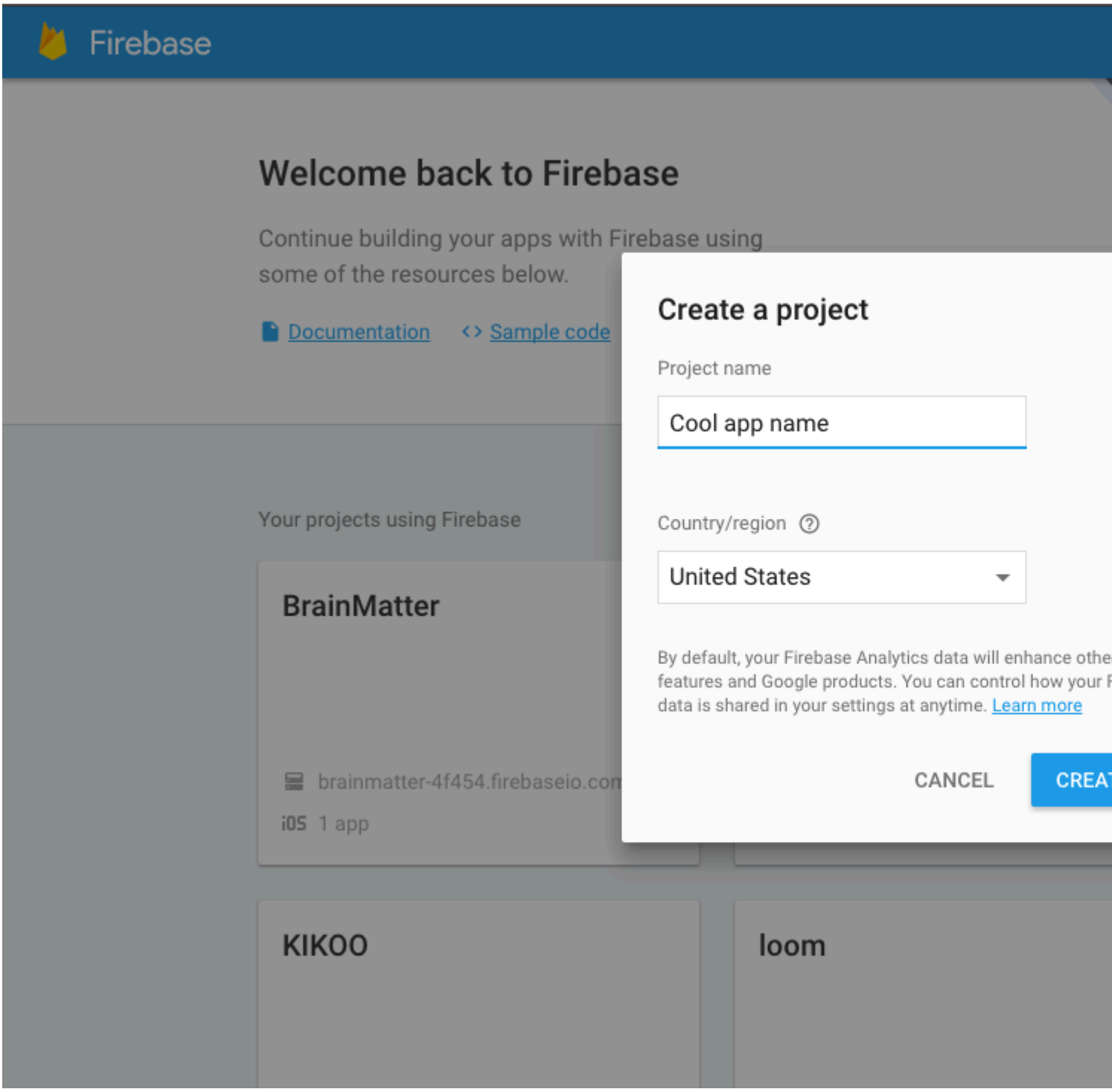
Línea de dependencia de Gradle	Servicio
com.google.firebase: firebase-core: 9.4.0	Analítica
com.google.firebase: firebase-database: 9.4.0	Base de datos en tiempo real
com.google.firebase: firebase-storage: 9.4.0	Almacenamiento
com.google.firebase: firebase-crash: 9.4.0	Reporte de Accidentes
com.google.firebase: firebase-auth: 9.4.0	Autenticación
com.google.firebase: firebase-messaging: 9.4.0	Mensajería en la nube / Notificaciones
com.google.firebase: firebase-config: 9.4.0	Configuración remota
com.google.firebase: firebase-invite: 9.4.0	Invitaciones / Enlaces Dinámicos
com.google.firebase: firebase-ads: 9.4.0	AdMob
com.google.android.gms: play-services-appindexing: 9.4.0	Indexación de aplicaciones

Configuración de Firebase para iOS

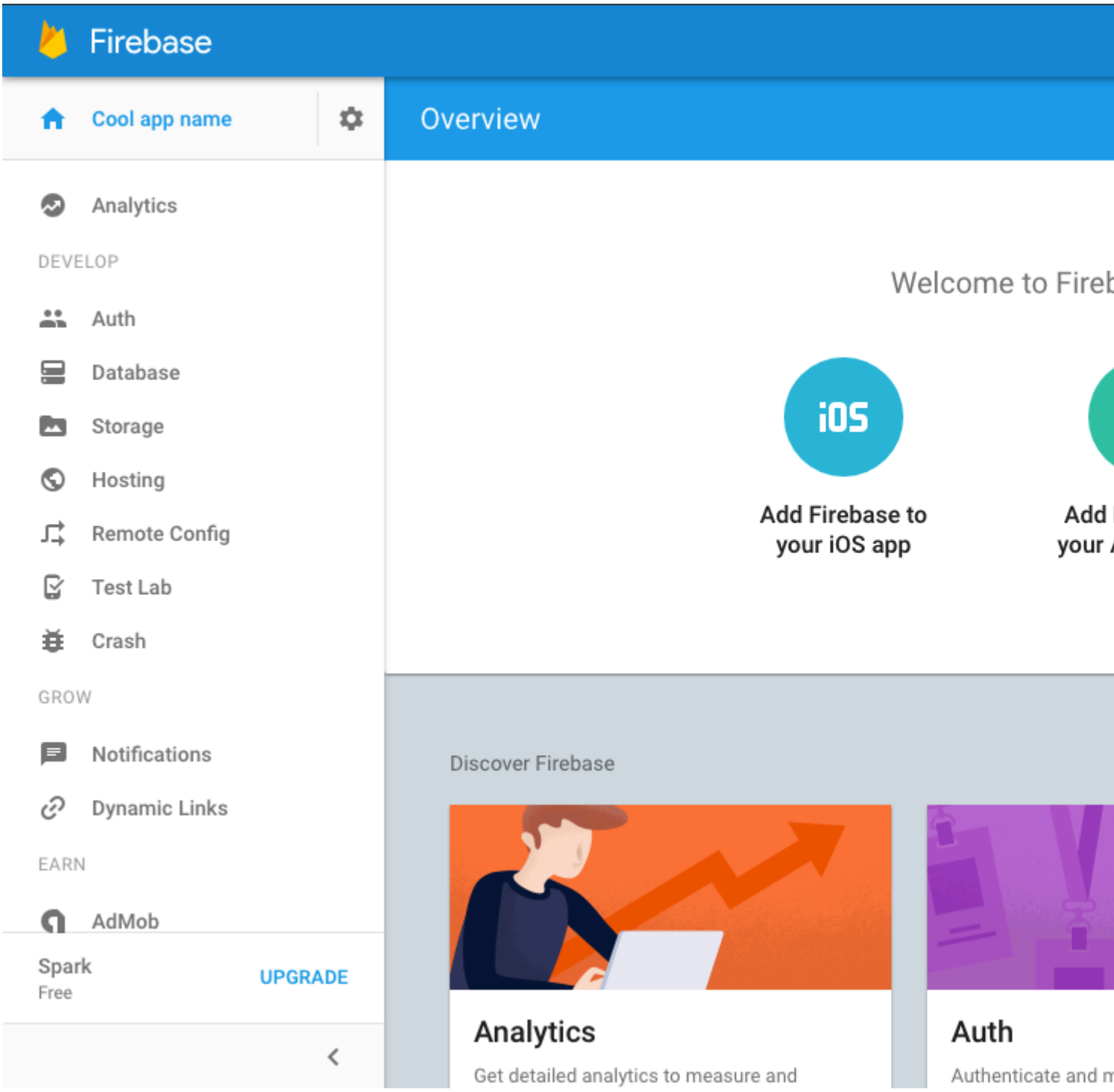
1. En primer lugar, desea ir al panel de base de fuego y crear un nuevo proyecto utilizando el botón 'Crear nuevo proyecto'.



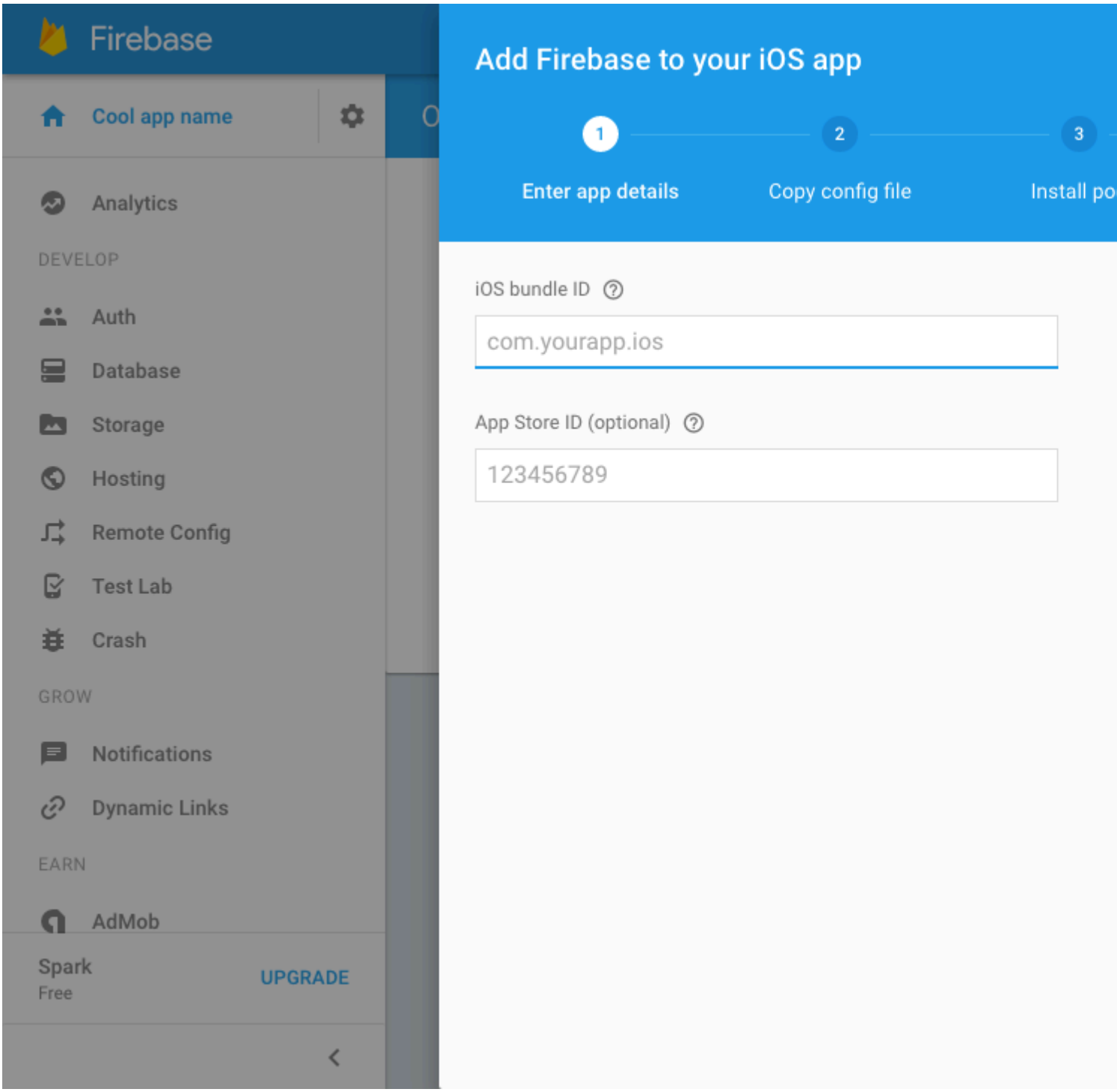
2. Desea crear un nuevo proyecto agregando el nombre de su aplicación, por ejemplo, pongo el mío como "Nombre de la aplicación", luego elija su región y presione "Crear proyecto"



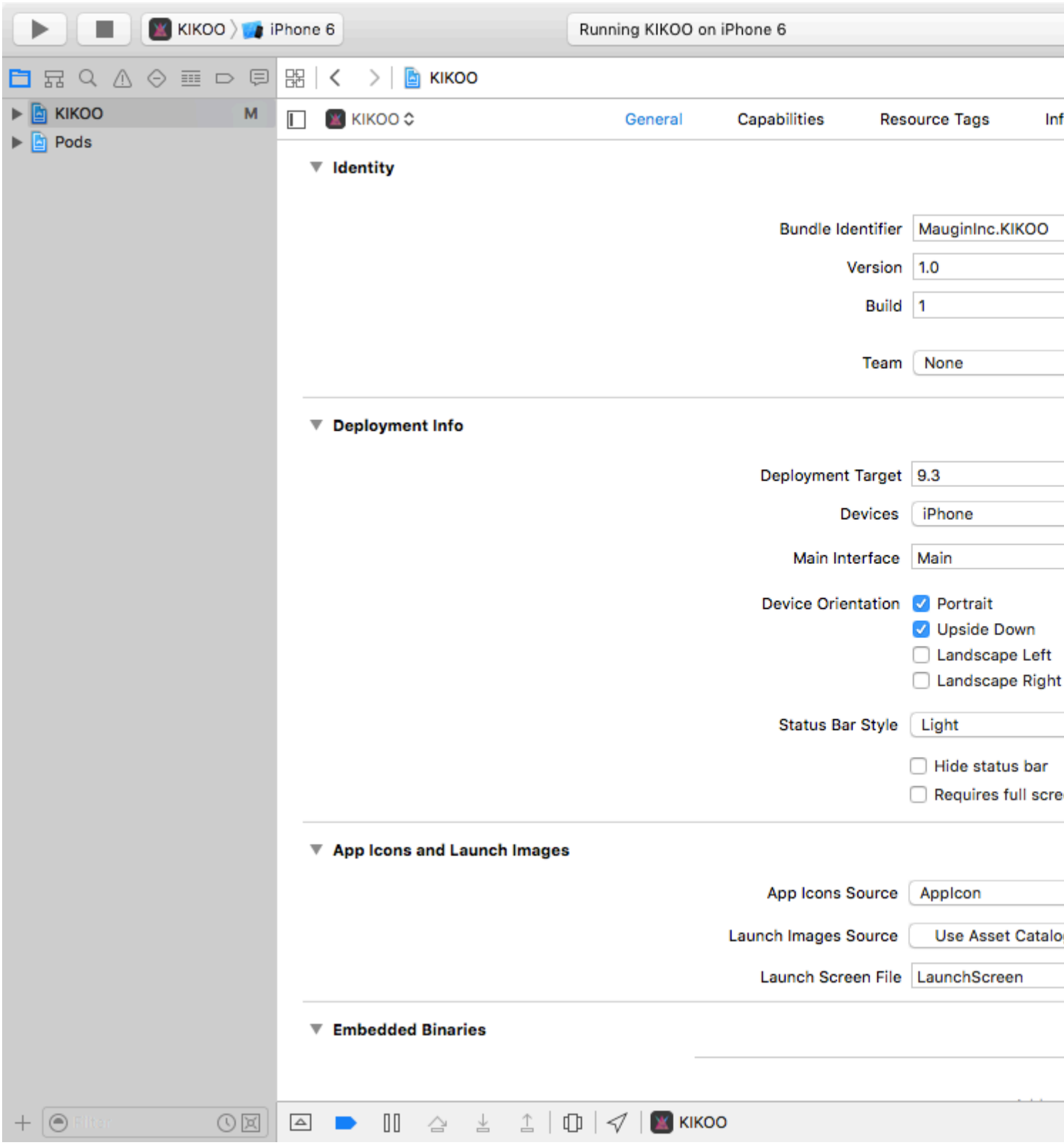
3. Después de crear el proyecto, se le dirigirá a esta página, que es el panel de control, y desde aquí debe elegir la plataforma en la que desea instalar Firebase. En este ejemplo, elegiremos IOS.



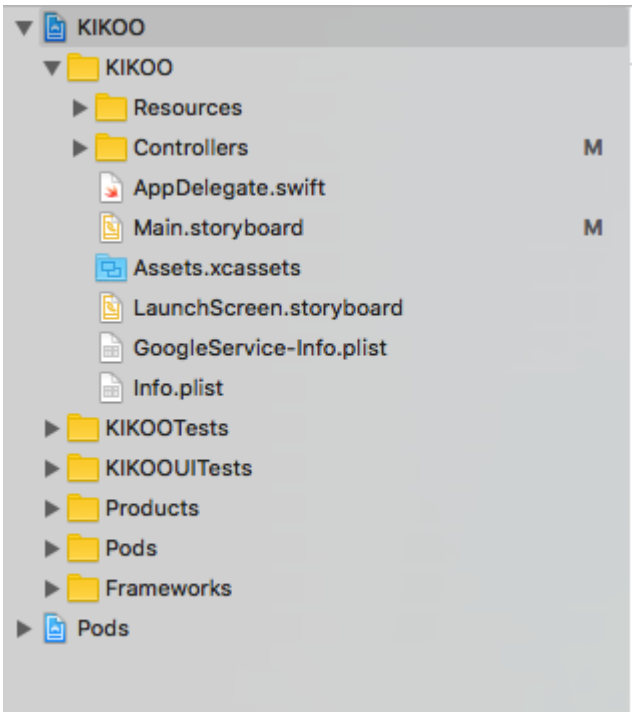
4. Después de seleccionar IOS, debería ver la misma ventana emergente que aparece en la imagen de abajo solicitando el paquete IOS y el ID de la tienda de aplicaciones. Solo deberá proporcionar el paquete IOS porque nuestra aplicación aún no está en la tienda de aplicaciones.



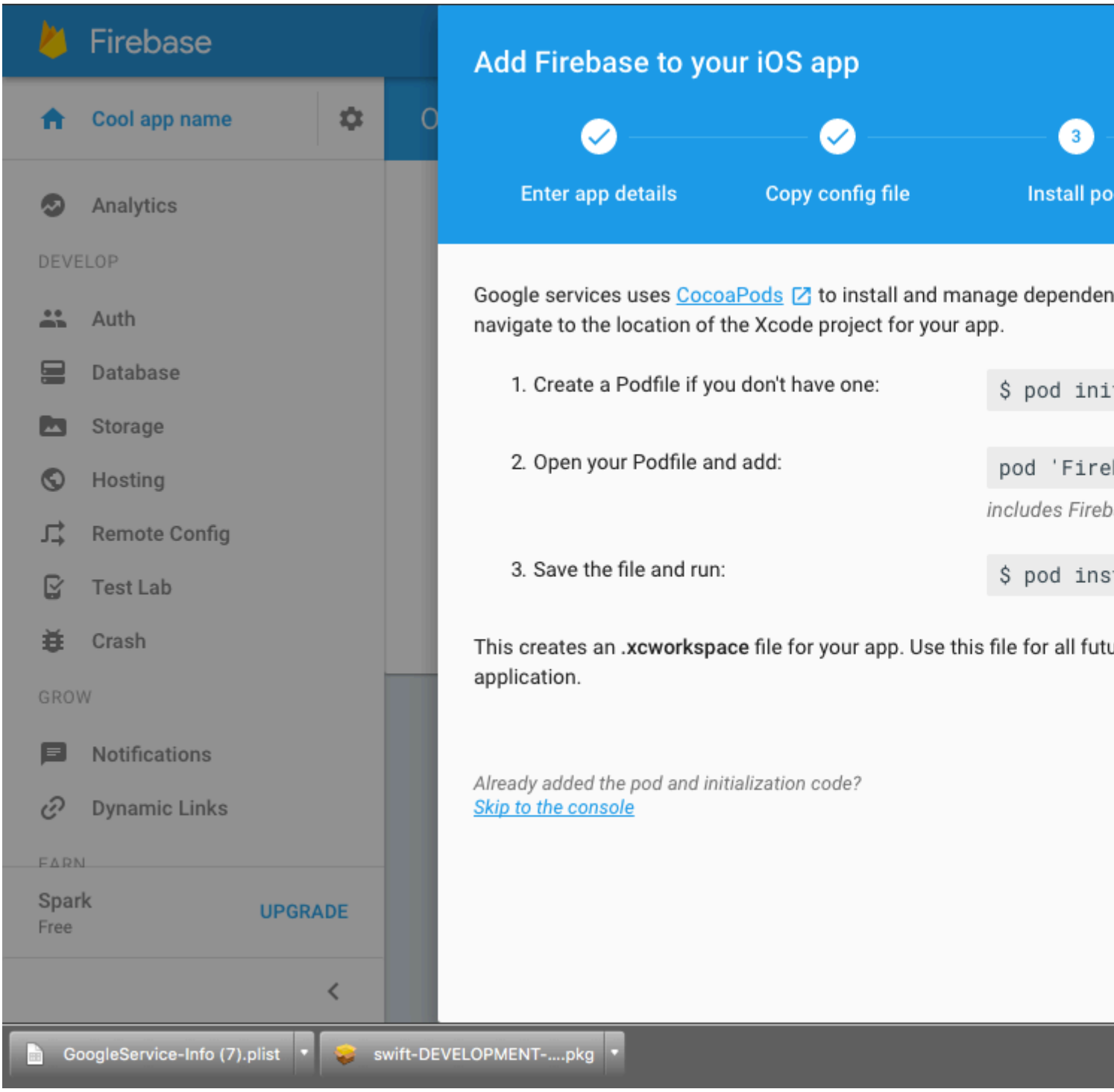
5. Obtenga el ID de paquete de xcode después de crear un proyecto de xcode de todos modos, por lo general, después de eso, podrá obtener el ID de paquete para su aplicación en la aplicación. La vista Genral en xcode será el primer campo en la parte superior y una vez que lo haya pegado en el campo Bundle en base de fuego, por ejemplo el mío, sería 'MauginInc.KIKOO'



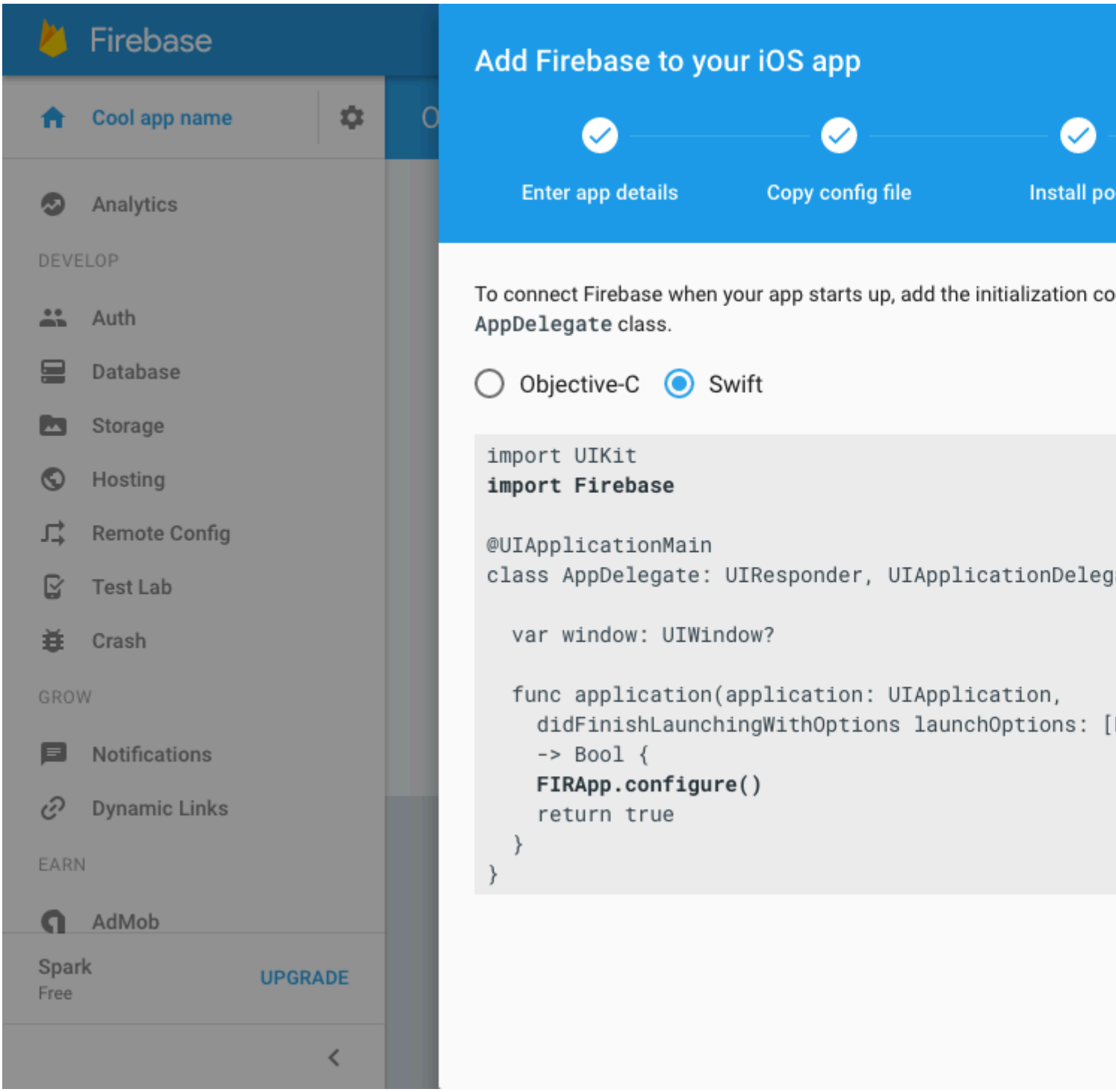
6. Después de hacer eso y presionar 'Siguiente', se descargará el archivo 'GoogleService-Info.plist' y lo que deberá hacer es moverlo a la carpeta raíz de su aplicación dentro de xcode



7. Usted querrá inicializar los pods e instalar los pods de base de fuego que necesita. Para ello, vaya a su terminal, navegue a su carpeta de proyectos de xcode y siga estas instrucciones proporcionadas por firebase.



8. Finalmente, desea configurar su aplicación para que Swift haga lo que hace mejor y eso hace que el desarrollo de la aplicación sea mucho más fácil y eficiente. Todo lo que necesita hacer es editar sus archivos AppDelegate.swift de la misma manera que la ventana emergente le muestra.



Eso es todo lo que ahora tiene Firebase instalado en su proyecto xcode para iOS

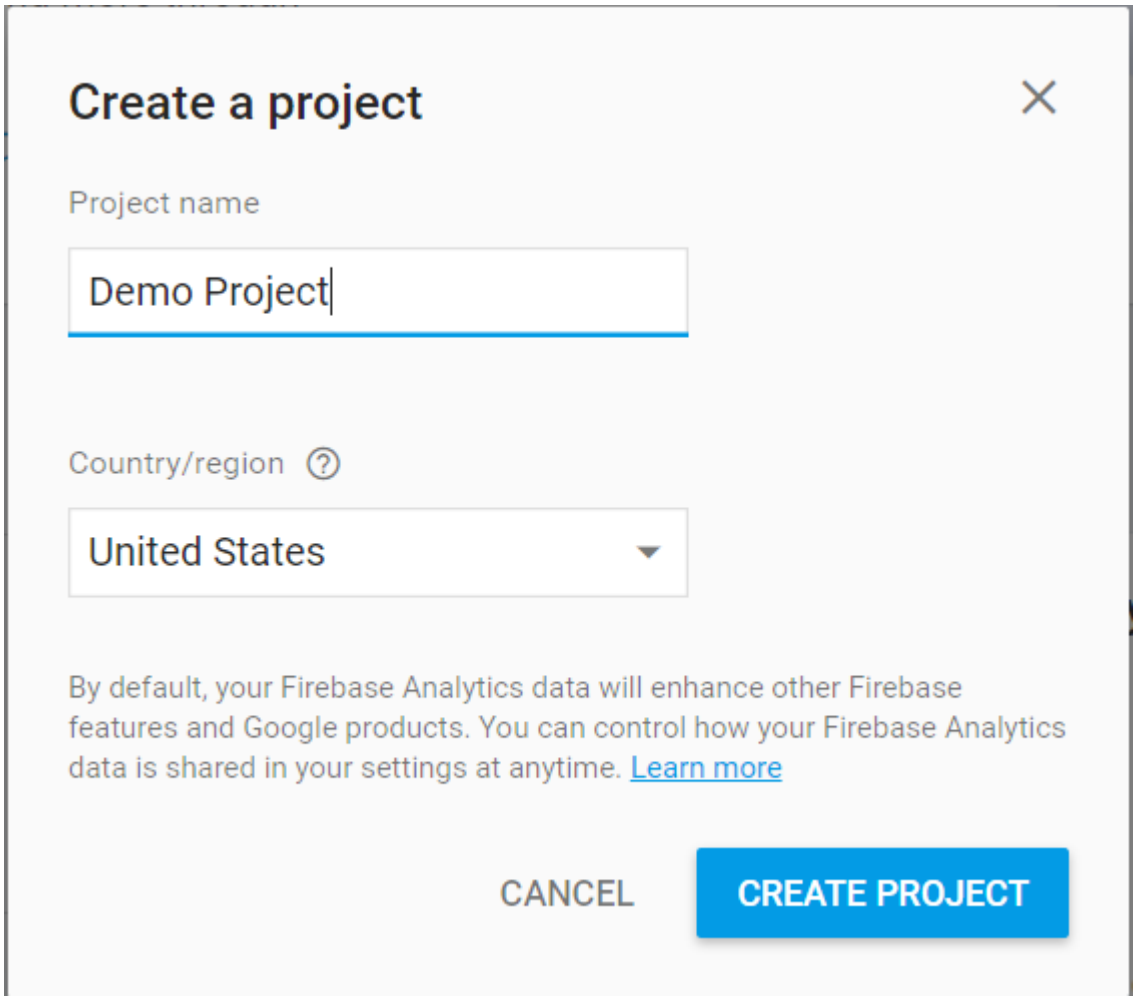
Comenzando en Firebase con una sencilla aplicación web Hello World en JavaScript

Este ejemplo mostrará cómo comenzar a usar Firebase en sus aplicaciones web con JavaScript.

Agregaremos un elemento **secundario de texto** a nuestra base de datos Firebase y lo mostraremos en tiempo real en nuestra aplicación web.

Empecemos.

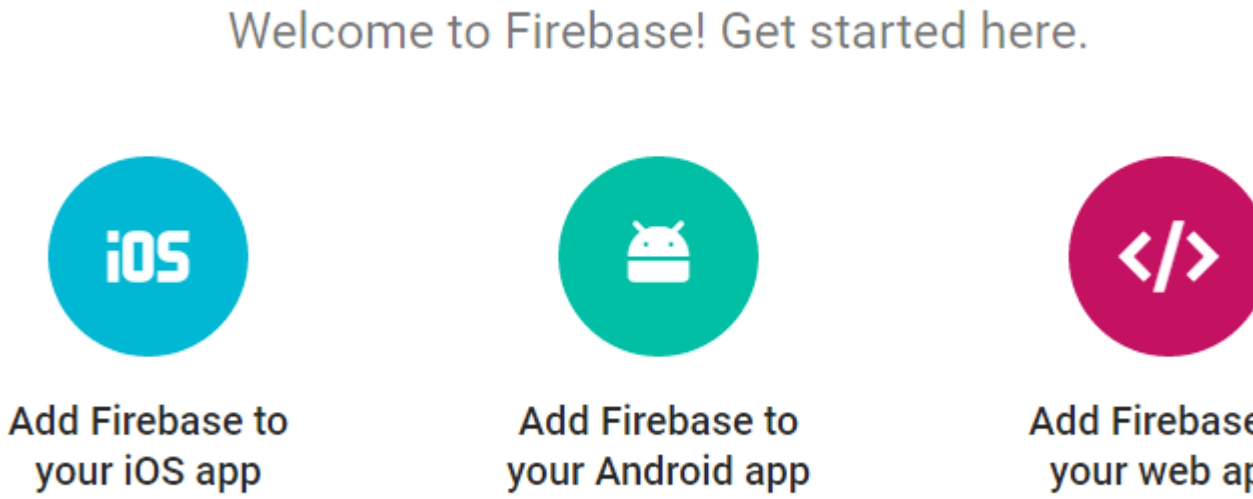
- Vaya a la Consola Firebase - https://console.firebase.google.com y cree un nuevo proyecto. Ingrese el nombre del proyecto, País / región y haga clic en **crear proyecto** .



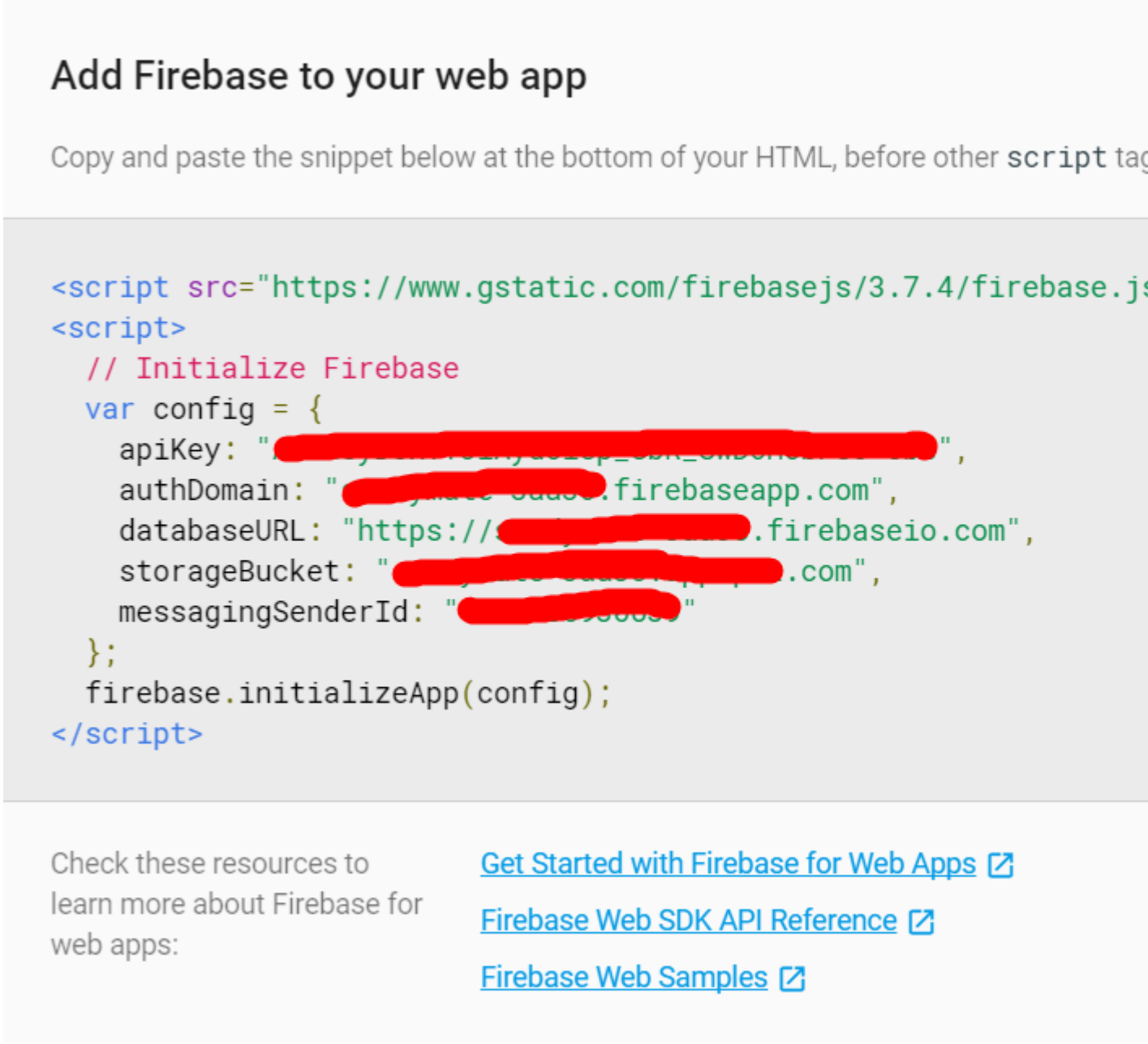
- Ahora crea un archivo **index.html** en tu computadora. Y añádele el siguiente código.

```
<body>
    <p>Getting started with Firebase</p>
    <h1 id="bigOne"></h1>
    <script>
          // your firebase JavaScript code here
    </script>

</body>
```

- Ahora ve a tu proyecto en la Consola Firebase y puedes ver esto



- Ahora haga clic en **Agregar Firebase a su aplicación web** . Aparecerá la siguiente ventana emergente, haga clic en el botón Copiar.



- Ahora vaya a su archivo **index.html** y agregue el fragmento a la sección del script de la siguiente manera

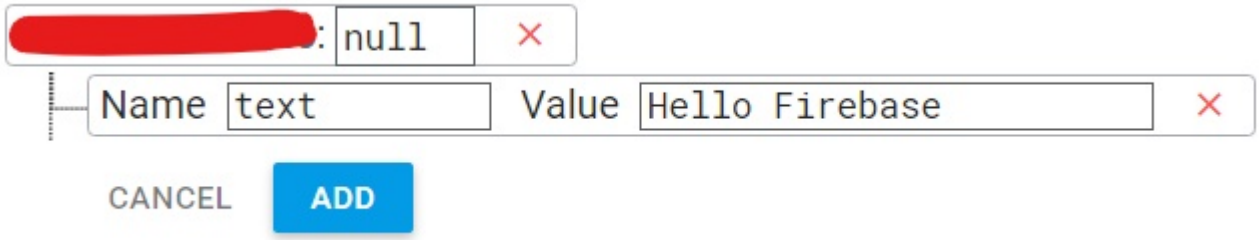
```
<body>

  <p>Getting started with Firebase</p>
  <h1 id="bigOne"></h1>

  <script src="https://www.gstatic.com/firebasejs/3.7.4/firebase.js"></script>
  <script>
    // Initialize Firebase
    var config = {
      apiKey: "apiKey",
      authDomain: "authDomain",
      databaseURL: "databaseURL",
      storageBucket: "storageBucket",
      messagingSenderId: "messagingSenderId"
    };
    firebase.initializeApp(config);
```

```
  </script>
</body>
```

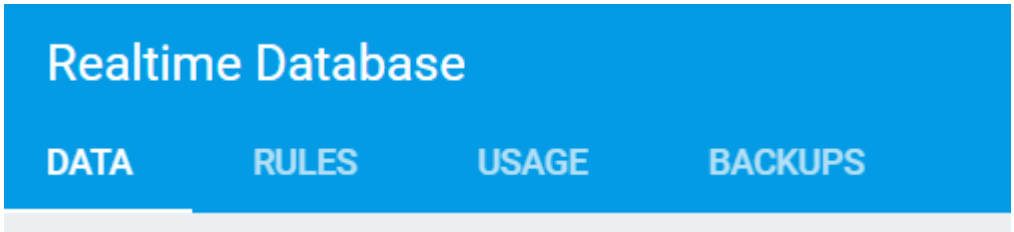
- Ahora ha terminado de agregar el código de inicialización de Firebase. Ahora necesitamos obtener nuestro valor de **texto** de la base de datos.
- Para hacerlo, agregue el siguiente código (Inicializar Firebase ya agregado en el último paso. No volver a agregar) dentro del script en **index.html**

```
<script>

    // Initialize Firebase
    var config = {
      apiKey: "apiKey",
      authDomain: "authDomain",
      databaseURL: "databaseURL",
      storageBucket: "storageBucket",
      messagingSenderId: "messagingSenderId"
    };
    firebase.initializeApp(config);

    // getting the text value from the database
    var bigOne = document.getElementById('bigOne');
    var dbRef = firebase.database().ref().child('text');
    dbRef.on('value', snap => bigOne.innerText = snap.val());

</script>
```

- Ahora hemos terminado con el archivo **index.html** y ahora vamos a la **base** de **datos** en la consola Firebase.
- Verás que está en blanco y vacío ahora mismo. Permite agregar un elemento **secundario de texto** en la base de datos y agregarle cualquier valor.



- Ahora haga clic en el botón **AÑADIR** .
- Ahora ve a la sección de **REGLAS** en la Base de Datos.



- Para fines de desarrollo en este momento, ahora mismo habilitaremos todas las consultas

de **lectura y escritura** .

```
{
  "rules": {
      ".read": "true",
      ".write": "true"
    }
}
```

```
{
  "rules": {
        ".read": "true",
        ".write": "true"
      }
}
```

- Ahora abre index.html en el navegador
- Verá el valor del texto en su página de la siguiente manera:

Getting started with Firebase

Hello Firebase

- Ahora, si regresa a su base de datos y cambia el valor **secundario** del **texto** a otra cosa, verá que el texto en el navegador también cambia sin ninguna actualización o recarga. Así es como funciona la **base de datos en tiempo real** en Firebase.

Lea Empezando con base de fuego en línea:
https://riptutorial.com/es/firebase/topic/816/empezando-con-base-de-fuego

Capítulo 2: ¿Cómo escucho los errores al acceder a la base de datos?

Introducción

Hay muchas razones por las que una operación de lectura o escritura puede fallar. Uno frecuente es porque sus reglas de seguridad rechazan la operación, por ejemplo, porque no está autenticado (de manera predeterminada, un usuario autenticado solo puede acceder a la base de datos) o porque está escribiendo / escuchando en una ubicación donde no lo hace. tener permiso.

Examples

Detecta errores al escribir un valor en Android

Hay muchas razones por las que una operación de escritura puede fallar. Uno frecuente es porque sus reglas de seguridad rechazan la operación, por ejemplo, porque no está autenticado (de manera predeterminada, un usuario autenticado solo puede acceder a una base de datos).

Puede ver estas violaciones de las reglas de seguridad en la salida logcat. Pero es fácil pasarlos por alto. También puede manejarlos en su propio código y hacerlos más visibles, lo que es especialmente útil durante el desarrollo (ya que su JSON, las reglas y el código cambian a menudo).

Para detectar una escritura fallida en Android, adjunte una devolución de llamada de finalización a `setValue` :

```
ref.setValue("My new value", new DatabaseReference.CompletionListener() {
    public void onComplete(DatabaseError databaseError, DatabaseReference databaseReference) {
        throw databaseError.toException();
    }
});
```

Lanzar una excepción como esta asegura que será muy difícil pasar por alto tal error la próxima vez.

Detecta errores al leer datos en Android

Una razón frecuente por la que su operación de lectura puede no funcionar es porque sus reglas de seguridad rechazan la operación, por ejemplo porque no está autenticado (de manera predeterminada, un usuario autenticado solo puede acceder a una base de datos).

Puede ver estas violaciones de las reglas de seguridad en la salida logcat. Pero es fácil pasarlos por alto. También puede manejarlos en su propio código y hacerlos más visibles, lo que es especialmente útil durante el desarrollo (ya que su JSON, las reglas y el código cambian a menudo).

Para detectar una lectura fallida en Android, debe implementar el método `onCancelled` de su `ChildEventListener` :

```
databaseRef.addChildEventListener(new ChildEventListener() {
    public void onChildAdded(DataSnapshot dataSnapshot, String s) { ... }
    public void onChildChanged(DataSnapshot dataSnapshot, String s) { ... }
    public void onChildRemoved(DataSnapshot dataSnapshot) { ... }
    public void onChildMoved(DataSnapshot dataSnapshot, String s) { ... }
    public void onCancelled(DatabaseError databaseError) {
        throw databaseError.toException();
    }
});
```

O si tienes un `ValueEventListener` :

```
databaseRef.addValueEventListener(new ValueEventListener() {
    public void onDataChange(DataSnapshot dataSnapshot, String s) { ... }
    public void onCancelled(DatabaseError databaseError) {
        throw databaseError.toException();
    }
});
```

Con este código en su lugar, será bastante difícil pasar por alto un error de seguridad al leer datos en Android.

Detecta errores al escribir un valor en iOS

Hay muchas razones por las que una operación de escritura puede fallar. Uno frecuente es porque sus reglas de seguridad rechazan la operación, por ejemplo, porque no está autenticado (de manera predeterminada, un usuario autenticado solo puede acceder a una base de datos).

Puede ver estas violaciones de las reglas de seguridad en la salida de su programa. Pero es fácil pasarlos por alto. También puede manejarlos en su propio código y hacerlos más visibles, lo que es especialmente útil durante el desarrollo (ya que su JSON, las reglas y el código cambian a menudo).

Para detectar una escritura fallida en iOS, adjunte un bloque de finalización a setValue :

```
let message = ["name": "puf", "text": "Hello from iOS"]
ref!.childByAutoId().setValue(message) { (error) in
    print("Error while writing message \(error)")
}
```

Lanzar una excepción como esta asegura que será muy difícil pasar por alto tal error la próxima vez.

Detectando errores al leer datos en JavaScript

Una razón frecuente por la que su operación de lectura puede no funcionar es porque sus reglas de seguridad rechazan la operación, por ejemplo porque no está autenticado (de manera predeterminada, un usuario autenticado solo puede acceder a una base de datos).

Puede ver estas violaciones de las reglas de seguridad en la consola de JavaScript de su navegador. Pero es fácil pasarlos por alto. También puede manejarlos en su propio código y hacerlos más visibles, lo que es especialmente útil durante el desarrollo (ya que su JSON, las reglas y el código cambian a menudo).

Para detectar una lectura fallida en JavaScript, debe implementar agregar una segunda devolución de llamada a su cláusula `on()` :

```
ref.on('value', function(snapshot) {
    console.log(snapshot.key, snapshot.val());
}, function(error) {
    alert(error);
})
```

Con este código en su lugar, será bastante difícil pasar por alto un error de seguridad al leer datos en JavaScript.

Detectando errores al escribir un valor en JavaScript

Hay muchas razones por las que una operación de escritura puede fallar. Uno frecuente es porque sus reglas de seguridad rechazan la operación, por ejemplo, porque no está autenticado (de manera predeterminada, un usuario autenticado solo puede acceder a una base de datos).

Puede ver estas violaciones de las reglas de seguridad en la salida de la consola. Pero es fácil pasarlos por alto. También puede manejarlos en su propio código y hacerlos más visibles, lo que es especialmente útil durante el desarrollo (ya que su JSON, las reglas y el código cambian a menudo).

Para detectar una escritura fallida en JavaScript, adjunte una devolución de llamada de finalización para `set` :

```
ref.set("My new value").catch(function(error)
    console.error(error);
    alert(error);
});
```

Mostrar una alerta como esta asegura que será muy difícil pasar por alto tal error la próxima vez.

Detecta errores al leer datos en iOS

Una razón frecuente por la que su operación de lectura puede no funcionar es porque sus reglas de seguridad rechazan la operación, por ejemplo porque no está autenticado (de manera predeterminada, un usuario autenticado solo puede acceder a una base de datos).

Puede ver estas violaciones de las reglas de seguridad en la salida de la consola. Pero es fácil pasarlos por alto. También puede manejarlos en su propio código y hacerlos más visibles, lo que es especialmente útil durante el desarrollo (ya que su JSON, las reglas y el código cambian a menudo).

Para detectar una lectura fallida en iOS, debe implementar el bloque `withCancel` de su observador:

```
    ref!.child("notAllowed").observe(.value, with: { (snapshot) in
        print("Got non-existing value: \(snapshot.key)")
    }, withCancel: { (error) in
        print(error)
    })
```

Lea ¿Cómo escucho los errores al acceder a la base de datos? en línea: https://riptutorial.com/es/firebase/topic/5548/-como-escucho-los-errores-al-acceder-a-la-base-de-datos-

Capítulo 3: ¿Cómo obtener el valor de la clave de inserción de la base de datos Firebase?

Introducción

En la base de datos Firebase, todo es un nodo, que sigue la clave del patrón: valor. Firebase Database nos proporciona una forma sencilla de generar claves únicas. Las claves únicas crean nuevos elementos mientras se actualizan los datos a una clave previamente almacenada.

Examples

Ejemplo de Android

Supongamos que tenemos una aplicación para perros, entonces nuestro modelo será una clase de perros.

```
DatabaseReference reference = FirebaseDatabase.getInstance().getReference().child("dogs");
```

Esta es la forma de enviar un perro a la base de datos, un nuevo perro único y configurar al perro con la clave.

```
String key = reference.push().getKey();
Dog dog = new Dog("Spike");
dog.setKey(key);
reference.child(key).setValue(dog);
```

El `reference.child(key).setValue(dog);` es equivalente de `reference.push().setValue(dog);` Y agrega el beneficio de obtener la clave dentro del objeto `Dog` .

Lea ¿Cómo obtener el valor de la clave de inserción de la base de datos Firebase? en línea: https://riptutorial.com/es/firebase/topic/10839/-como-obtener-el-valor-de-la-clave-de-insercion-de-la-base-de-datos-firebase-

Capítulo 4: ¿Cómo usar FirebaseRecyclerAdapter en lugar de RecyclerAdapter?

Examples

Aquí está el ejemplo para usar el componente FirebaseUi FirebaseRecyclerAdapter

Hola, amigos antes de comenzar el código, tenemos que declarar la dependencia para acceder al componente ui de firebase, por lo que aquí está la dependencia que puede poner en su gradel. De lo contrario, también puede agregar la dependencia como jar.

```
compile 'com.firebaseui:firebase-ui-database:0.4.0'
```

Luego, después de que estemos consultando en la base de datos de base de fuego para datos como sigue

```
DatabaseReference databaseReference = database.getReference().child("users");
Query query = databaseReference.limitToFirst(50);
```

Luego, después de que pasemos la consulta dentro de FirebaseRecyclerAdapter como sigue

```
private void setUpFirebaseAdapter(Query query) {

        mFirebaseAdapter = new FirebaseRecyclerAdapter<UserModel, FirebaseUserViewHolder>
                (UserModel.class, R.layout.row_user_list, FirebaseUserViewHolder.class, query)
{
            @Override
            protected void populateViewHolder(FirebaseUserViewHolder viewHolder, UserModel
model, int position) {
                customeLoaderDialog.hide();
                viewHolder.bindUser(model);
            }
        };

        my_recycler_view.setHasFixedSize(true);
        my_recycler_view.setLayoutManager(new LinearLayoutManager(this));
        my_recycler_view.setAdapter(mFirebaseAdapter);

    }
```

ChatUserModel.java (Clase de modelo)

```
public class ChatUserModel {
private long badge;
private String chat_id;
```

```
private String isDelete;
private String latestactivity;
private double timestamp;
private String user_id;
private String profilePic;
private String displayName;
private boolean isGroup;
String groupId;
private String creatorId;

public String getGroupId() {
    return groupId;
}

public void setGroupId(String groupId) {
    this.groupId = groupId;
}

public String getCreatorId() {
    return creatorId;
}

public void setCreatorId(String creatorId) {
    this.creatorId = creatorId;
}

public boolean isGroup() {
    return isGroup;
}

public void setGroup(boolean group) {
    isGroup = group;
}


public ChatUserModel() {

}

public long getBadge() {
    return badge;
}

public void setBadge(long badge) {
    this.badge = badge;
}

public String getChat_id() {
    return chat_id;
}

public void setChat_id(String chat_id) {
    this.chat_id = chat_id;
}

public String getIsDelete() {
    return isDelete;
}

public void setIsDelete(String isDelete) {
    this.isDelete = isDelete;
```

```
}

public String getLatestactivity() {
    return latestactivity;
}

public void setLatestactivity(String latestactivity) {
    this.latestactivity = latestactivity;
}

public double getTimestamp() {
    return timestamp;
}

public void setTimestamp(double timestamp) {
    this.timestamp = timestamp;
}

public String getUser_id() {
    return user_id;
}

public void setUser_id(String user_id) {
    this.user_id = user_id;
}

public String getProfilePic() {
    return profilePic;
}

public void setProfilePic(String profilePic) {
    this.profilePic = profilePic;
}

public String getDisplayName() {
    return displayName;
}

public void setDisplayName(String displayName) {
    this.displayName = displayName;
}}
```

FirebaseChatUserViewHolder.java (Recycler ViewHolder)

```
public class FirebaseChatUserViewHolder extends RecyclerView.ViewHolder implements
View.OnClickListener {

    private static final int MAX_WIDTH = 200;
    private static final int MAX_HEIGHT = 200;
    View mView;
    Context mContext;
    ChatUserModel userModel;

    public FirebaseChatUserViewHolder(View itemView) {
        super(itemView);
        mView = itemView;
        mContext = itemView.getContext();
        itemView.setOnClickListener(this);
    }
```

```
    public void bindUser(ChatUserModel userModel) {
        this.userModel = userModel;
        ImageView imgUser = (ImageView) mView.findViewById(R.id.imgUser);
        TextView tvName = (TextView) mView.findViewById(R.id.tvName);
        TextView tvStatus = (TextView) mView.findViewById(R.id.tvStatus);
        BadgeView badgeChat = (BadgeView) mView.findViewById(R.id.badgeChat);
        if (userModel.isGroup()) {
           //
imgUser.setImageDrawable(mContext.getResources().getDrawable(R.drawable.create_group));
        } else {
            Picasso.with(mContext)
                    .load(userModel.getProfilePic())
                    .resize(MAX_WIDTH, MAX_HEIGHT)
                    .centerCrop()
                    .into(imgUser);
        }

        tvName.setText(userModel.getDisplayName());
        tvStatus.setText(userModel.getLatestactivity());
        if (userModel.getBadge() > 0) {
            badgeChat.setVisibility(View.VISIBLE);
            badgeChat.setText("" + userModel.getBadge());
        } else {
            badgeChat.setVisibility(View.GONE);
        }

    }

    @Override
    public void onClick(View view) {
        if (!userModel.isGroup()) {
            Intent intent = new Intent(mContext, ChatConverstion.class);
            intent.putExtra("chat_id", "" + userModel.getChat_id());
            intent.putExtra("reciverUserName", "" + userModel.getDisplayName());
            intent.putExtra("reciverProfilePic", "" + userModel.getProfilePic());
            intent.putExtra("reciverUid", "" + userModel.getUser_id());
            mContext.startActivity(intent);
        }
    }
}
```

row_user_list.xml (diseño para la fila en la vista de reciclador)

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="@android:color/white"
    android:orientation="horizontal"
   >

    <LinearLayout
        android:gravity="center_vertical"
        android:layout_width="match_parent"
        android:id="@+id/llMainChat"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:paddingTop="@dimen/margin_small"
        android:paddingLeft="@dimen/margin_small"
        android:paddingBottom="@dimen/margin_small"
```

```
        android:paddingRight="@dimen/margin_small"
        >

        <com.tristate.firebasechat.custome_view.CircleImageView
            android:id="@+id/imgUser"
            android:layout_width="@dimen/tab_top_height"
            android:layout_height="@dimen/tab_top_height"

            app:civ_border_color="@color/dark_white"
            app:civ_border_width="2dp" />

        <RelativeLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="horizontal"
            android:layout_marginLeft="@dimen/margin_medium"
         >

            <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:orientation="vertical"
                android:layout_toLeftOf="@+id/badgeChat"
                android:layout_toStartOf="@+id/badgeChat"
                android:id="@+id/linearLayout">

                <TextView
                    android:id="@+id/tvName"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:ellipsize="marquee"
                    android:singleLine="true"
                    android:text="Dhaval Solanki"
                    android:textSize="@dimen/textsize_midle" />

                <TextView
                    android:id="@+id/tvStatus"
                    android:ems="3"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:gravity="center_vertical"
                    android:lines="1"
                    android:text="Online"
                    android:textColor="@color/greenStatusBar"
                    android:textSize="@dimen/textsize_small" />
            </LinearLayout>
            <com.tristate.firebasechat.custome_view.BadgeView
                android:id="@+id/badgeChat"
                android:layout_width="@dimen/margin_very_big"
                android:layout_height="@dimen/margin_very_big"
                android:layout_alignParentRight="true"
                android:layout_centerVertical="true"
                android:background="@drawable/badge_bg"
                android:gravity="center"
                android:padding="@dimen/corner_radius"
                android:text="999"
                android:textColor="@color/white"
                android:textSize="@dimen/textsize_verysmall"
                android:visibility="gone" />
```

```
        </RelativeLayout>


    </LinearLayout>
    <View
        android:layout_alignBottom="@id/llMainChat"
        android:layout_marginTop="@dimen/margin_small"
        android:layout_marginLeft="@dimen/margin_small"
        android:layout_marginRight="@dimen/margin_small"
        android:layout_width="match_parent"
        android:background="@color/avatar_back_color"
        android:layout_height="1dp"
        ></View>
</RelativeLayout>
```

Lea ¿Cómo usar FirebaseRecyclerAdapter en lugar de RecyclerAdapter? en línea: https://riptutorial.com/es/firebase/topic/8982/-como-usar-firebaserecycleradapter-en-lugar-de-recycleradapter-

Capítulo 5: Almacenamiento

Observaciones

Firebase Storage proporciona cargas y descargas seguras de archivos para sus aplicaciones Firebase, independientemente de la calidad de la red. Puede usarlo para almacenar imágenes, audio, video u otro contenido generado por el usuario. Firebase Storage está respaldado por Google Cloud Storage, un servicio de almacenamiento de objetos potente, simple y rentable.

Firebase Storage almacena sus archivos en un depósito de Google Cloud Storage compartido con la aplicación Google App Engine predeterminada, haciéndolos accesibles a través de las API de Firebase y Google Cloud. Esto le permite la flexibilidad de cargar y descargar archivos de clientes móviles a través de Firebase y realizar el procesamiento del lado del servidor, como el filtrado de imágenes o la transcodificación de videos con Google Cloud Platform. Firebase Storage se escala automáticamente, lo que significa que no es necesario migrar de Firebase Storage a Google Cloud Storage ni a ningún otro proveedor.

Esta integración hace que los archivos sean accesibles directamente desde las bibliotecas cliente de gcloud de Google Cloud Storage, para que pueda utilizar Firebase Storage con sus idiomas favoritos del servidor. Para tener más control, también puede usar las API de JS de XML y Google Cloud Storage.

Firebase Storage se integra a la perfección con Firebase Authentication para identificar a los usuarios y proporciona un lenguaje de seguridad declarativo que le permite establecer controles de acceso en archivos individuales o grupos de archivos, para que pueda hacer los archivos tan públicos o privados como desee.

Consulte la documentación pública de Firebase Storage para obtener las API, las muestras y las aplicaciones de ejemplo más actualizadas.

Examples

Empezando en iOS

Prerrequisitos

1. Cree un nuevo proyecto y agregue una aplicación iOS a ese proyecto en la Consola Firebase .
2. Descargue e incluya `GoogleServices-Info.plist` en su aplicación.

Agrega Firebase Storage a tu aplicación

Agregue la siguiente dependencia al `Podfile` de `Podfile` su proyecto:

```
pod 'Firebase/Storage'
```

Ejecute `pod install` y abra el archivo `.xcworkspace` creado.

Siga estas instrucciones para instalar Firebase sin CocoaPods

Configurar Firebase Storage

Debe inicializar Firebase antes de crear o utilizar cualquier referencia de la aplicación Firebase. Si ya ha hecho esto para otra característica de Firebase, puede omitir los siguientes dos pasos.

Importe el módulo Firebase:

```
// Obj-C
@import Firebase;
```

```
// Swift
import Firebase
```

Configure una instancia compartida de `FIRApp` , normalmente en la aplicación de su `application:didFinishLaunchingWithOptions:` method:

```
// Obj-C
[FIRApp configure];
```

```
// Swift
FIRApp.configure()
```

Obtenga una referencia al servicio de almacenamiento, usando la aplicación Firebase predeterminada:

```
// Obj-C
FIRStorage *storage = [FIRStorage storage];
```

```
// Swift
let storage = FIRStorage.storage()
```

Cree una referencia a un archivo en Firebase Storage:

```
// Obj-C
FIRStorageReference *reference = [[storage reference] child:@"path/to/file.txt"];
```

```
// Swift
let reference = storage.reference().child("path/to/file.txt")
```

Subir un archivo a Firebase Storage:

```
// Obj-C
```

```
NSData *data = ...
FIRStorageUploadTask *uploadTask = [riversRef putData:data metadata:nil
completion:^(FIRStorageMetadata *metadata, NSError *error) {
  if (error != nil) {
    // Uh-oh, an error occurred!
  } else {
    // Metadata contains file metadata such as size, content-type, and download URL.
    NSURL downloadURL = metadata.downloadURL;
  }
}];
```

```
// Swift
let data: NSData! = ...
let uploadTask = riversRef.putData(data, metadata: nil) { metadata, error in
  if (error != nil) {
    // Uh-oh, an error occurred!
  } else {
    // Metadata contains file metadata such as size, content-type, and download URL.
    let downloadURL = metadata!.downloadURL
  }
}
```

Lea Almacenamiento en línea: https://riptutorial.com/es/firebase/topic/4281/almacenamiento

Capítulo 6: Base de datos en tiempo real de Firbase con Android

Examples

Cómo conectar la base de datos en tiempo real con la aplicación de Android

Cómo implementar la base de datos FirebaseRealTime en la aplicación de Android. A continuación se indican los pasos para hacerlo.

1. Primero instale el sdk de firebase. Si no sabe cómo instalarlo, a continuación encontrará la URL para obtener ayuda. Instalar Firebase SDK

2. Después de registrar su proyecto en la consola firbase, la URL de la consola firbase es la URL de la consola Firebase

3. Después de completar con éxito el paso anterior, agregue la dependencia siguiente en su nivel de aplicación gradel. compile 'com.google.firebase: firebase-database: 9.2.1'

4. También una cosa más configurar sus reglas de base de datos de base de fuego. Si no sabe cómo configurarlo, a continuación encontrará la URL que lo ayudará. Configurar reglas de base de fuego

5. Ahora, después de todo, el código original es el inicio, primero recupere la instancia de su base de datos, lance FirebaseDatabase como la siguiente,

   Base de datos FirebaseDatabase = FirebaseDatabase.getInstance (); DatabaseReference myRef = database.getReference ("mensaje");

Ahora puede crear diferentes objetos diferentes de DatabaseReference para el acceso a un nodo diferente,

6. Ahora puede guardar o recuperar datos usando DataBaseReference como de la siguiente manera, Para guardar:

   myRef.setValue ("Demo para guardar");

Leer datos:

```
myRef.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(DataSnapshot dataSnapshot) {
        // This method is called once with the initial value and again
        // whenever data at this location is updated.
        String value = dataSnapshot.getValue(String.class);
        Log.d(TAG, "Value is: " + value);
    }
```

```
    @Override
    public void onCancelled(DatabaseError error) {
        // Failed to read value
        Log.w(TAG, "Failed to read value.", error.toException());
    }
});
```

Nota: Este es el único tema de introducción sobre cómo implementar la base de datos en la aplicación de Android, o más cosas disponibles en la base de datos FirebaseRealtime.

Lea Base de datos en tiempo real de Firbase con Android en línea:
https://riptutorial.com/es/firebase/topic/6482/base-de-datos-en-tiempo-real-de-firbase-con-android

Capítulo 7: Capacidades fuera de línea de Firebase

Introducción

En esta publicación encontrará las diferentes formas de implementar capacidades fuera de línea al usar `Firebase` , información sobre cuándo y por qué podría ser una buena idea habilitar capacidades fuera de línea y ejemplos de ello con la plataforma Android.

Observaciones

¿Qué debo usar? ¿La persistencia del disco o mantener llamadas sincronizadas?

Desde mi experiencia, puedo decir que siempre depende de lo que funcione su aplicación y de cómo administre las transacciones y la base de datos de su aplicación. Si, por ejemplo, tiene una aplicación en la que el usuario solo escribe y lee datos pero no puede eliminarlos o editarlos, usar `DiskPersistence` sería la opción correcta.

Además, `DiskPersistence` almacenará los datos en caché, lo que significa que su aplicación utilizará más espacio en los dispositivos del usuario, lo que tal vez no sea la mejor idea en su caso.

Por otra parte, si su aplicación gestiona muchas transacciones complejas y sus datos se actualizan muy a menudo, posiblemente deba evitar `DiskPersistence` y utilizar `keepSynced` en las referencias que desee mantener actualizadas.

¿Por qué?

`DiskPersistence` almacena los datos recuperados en local, lo que a veces puede causar gran cantidad de desincronización que muestra sus datos si no los usa junto con `ListenerValueEvents` continuo. Por ejemplo:

1. El usuario A escribe un mensaje "Hello World" en su aplicación, que se recibe para el usuario B
2. El usuario B descarga el mensaje del usuario A en su teléfono y ve el mensaje "Hola mundo"
3. El usuario A edita su mensaje a "Firebase is cool".
4. El usuario B seguirá viendo el mensaje "Hola mundo" incluso si actualiza los datos porque la referencia de la instantánea es la misma cuando Firebase lo filtra.

Para evitar esto, la mejor idea es mantener a los oyentes en las referencias que desea rastrear todo el tiempo.

¿Puedo usar los dos juntos?

Por supuesto que puede, y en la mayoría de las aplicaciones es posiblemente la mejor idea para evitar descargar muchos datos y ofrecer al usuario la posibilidad de trabajar con su aplicación, incluso si no tiene conexión.

Si no le importa usar el espacio de caché en el dispositivo del usuario, le recomiendo habilitar `diskPersistence` en su objeto `FirebaseDatabase` y también agregar un indicador `keepSync` a cada referencia que pueda tener muchas veces en un espacio de tiempo corto o que desee mantenerse fresco todo el tiempo.

Examples

Habilitar la persistencia del disco (solo Android / iOS)

Para habilitar la persistencia del disco, debe habilitar el indicador `persistenceEnabled` en el objeto `FirebaseDatabaseInstance` de su aplicación:

Androide

```
FirebaseDatabase.getInstance().setPersistenceEnabled(true);
```

iOS

```
Database.database().isPersistenceEnabled = true //Swift
[FIRDatabase database].persistenceEnabled = YES; //Objetive-C
```

Si desea deshabilitar la persistencia en algún momento del ciclo de vida de su aplicación, recuerde deshabilitarla de la misma manera:

Androide

```
FirebaseDatabase.getInstance().setPersistenceEnabled(false);
```

iOS

```
Database.database().isPersistenceEnabled = false //Swift
[FIRDatabase database].persistenceEnabled = NO; //Objetive-C
```

Mantener los datos actualizados (solo Android / iOs)

Firebase sincroniza y almacena una copia local de los datos para los oyentes activos cuando se utiliza en dispositivos móviles. Además, puede mantener ubicaciones específicas sincronizadas.

Android:

```
DatabaseReference workoutsRef = FirebaseDatabase.getInstance().getReference("workouts");
scoresRef.keepSynced(true);
```

iOs:

```
//Objetive-c
FIRDatabaseReference *scoresRef = [[FIRDatabase database] referenceWithPath:@"scores"];
[scoresRef keepSynced:YES];
//Swift
let scoresRef = Database.database().reference(withPath: "scores")
scoresRef.keepSynced(true)
```

El cliente Firebase descarga automáticamente los datos en estas ubicaciones y los mantiene actualizados, incluso si la referencia no tiene escuchas activas. Desactivas la sincronización con la siguiente línea de código.

Android:

```
scoresRef.keepSynced(false);
```

iOS:

```
[scoresRef keepSynced:NO]; //Objetive-C
scoresRef.keepSynced(false) //Swift
```

Lea Capacidades fuera de línea de Firebase en línea:
https://riptutorial.com/es/firebase/topic/10777/capacidades-fuera-de-linea-de-firebase

Capítulo 8: Cola de Firebase

Examples

Cómo usar la cola de base de fuego como un servidor para tu aplicación

Firebase proporciona backend como servicio, como desarrollador de aplicaciones, no tiene una opción para tener código de backend.

Este ejemplo muestra cómo usar la cola de base de fuego, crear un backend que operará en la parte superior de la base de datos de base de fuego y servir como un backend para su aplicación frontend.

Antes de entrar en el código, entendamos la arquitectura, cómo funcionará. Por brevedad, supongamos que estamos utilizando el sitio web como frontend y el servidor NodeJs como backend.

Prerrequisitos

1. Crea una aplicación Firebase usando tu cuenta de Google.
2. Agrega firebase a tu página web. Utilice la `bower install firebase --save`
3. Cree una cuenta de servicio usando su nueva cuenta de base de fuego creada (Configuración-> Permisos -> Cuentas de servicio -> CREAR CUENTA DE SERVICIO -> (especifique el nombre y marque esta casilla de verificación "Proporcionar una nueva clave privada") -> guardar el archivo json, necesitaremos que despues
4. Configure el servidor NodeJs que se puede alojar en su entorno preferido
5. Crear el siguiente punto final dentro de la `queue/specs`

   "solicitar respuesta":

   ```
   {
       "error_state": "request_error_processing",
       "finished_state": "finished_state",
       "in_progress_state": "request_in_progress",
       "start_state": "request_started"
   }
   ```

6. Dentro del servidor NodeJs, `npm install firebase --save` versión del lado del servidor firebase, `npm install firebase --save` , y intialice su cuenta de servicio usando el archivo json que obtuvimos en el paso 3, se ve así

   firebase.initializeApp ({serviceAccount: './your_file.json', databaseURL: 'get_from_firebase_account'});

Arquitectura

Aquí está todo el ciclo de cómo funciona.

En el lado frontal vas a hacer estos pasos.

1. Usando firebase web sdk, está escribiendo sus solicitudes directamente en la base de datos de firebase en el punto final 'cola / tareas', le permite llamar a su solicitud que está enviando al backend.
2. después de insertar su tarea, está registrando el oyente en la `queue/tasks/{taskKey}` del punto final, que se llamará cuando el servidor finalice el procesamiento de su solicitud, escribiendo la respuesta dentro de la tarea anterior

En el lado de atrás vas a hacer estos pasos.

1. Crear un servidor que escuche infinitamente el punto final 'cola / tareas'
2. Procesa sus tareas y escribe datos de respuesta dentro de la `queue/tasks/response`
3. Quitar la tarea

En primer lugar, cree esta función auxiliar, que proporciona una forma de gestionar las devoluciones de llamada y las promesas en conjunto.

```
function createPromiseCallback() {
    var cb;
    var promise = new Promise(function (resolve, reject) {
        cb = function (err, data) {
            if (err) return reject(err);
            return resolve(data);
        };
    });
    cb.promise = promise;
    return cb;
}
```

En la parte frontal vas a tener esta función.

```
function sendRequest(kind, params, cb) {

    cb = cb || createPromiseCallback();
    var requestObject = {
        kind: kind,
        params: params
    };
    var tasksRef = firebase.database().ref('queue/tasks');

    var requestKey = tasksRef.push().key;

    var requestRef = tasksRef.child(requestKey);

    function requestHandshake(snap) {
        if (snap && snap.exists() && (snap.val().response || snap.val()._state ===
config.firebase.task.finishState || snap.val()._error_details)) {
            var snapVal = snap.val();
            if (snapVal._error_details) {
```

```
                cb(snapVal._error_details.error);
            } else {
            cb(null, snapVal.response);
        }
        requestRef.off('value', requestHandshake);
    }
   }

   var bulkUpdate = {};
   bulkUpdate['queue/tasks/' + requestKey + '/request'] = requestObject;
   bulkUpdate['queue/tasks/' + requestKey + '/_state'] = config.firebase.task.startState;

   firebase.database().ref().update(bulkUpdate)
    .then(function (snap) {
        requestRef.on('value', requestHandshake);
    }).catch(function (err) {
        cb(err);
    });

   return cb.promise;
  }
```

puede usar esta función como `sendRequest('CreateHouseFacade', {houseName:'Test'})` .

El parámetro tipo es para el backend, para saber qué método llamar para procesar la solicitud. Parámetros es para pasar información de parámetros adicionales.

Y aquí está el código de fondo

```
const database = firebase.database();
const queueRef = database.ref('queue');

const queueOptions = {
    'specId': 'request_response',
    'sanitize': false,
    'suppressStack': false,
    'numWorkers': 3
};

function removeTask(task) {
    var taskRef = queueRef.child(`tasks/${task._id}`);
    return taskRef.remove();
}

function processTask(data, progress, resolve, reject) {
    try {
        requestHandler(data.request).then(response => {
            data.response = response || null;
            return resolve(data);
        }).catch(err => {
            return reject(err);
        }).then(snap => {
            removeTask(data);
        });
    } catch (err) {
        reject(err).then(snap => removeTask(data));
    }
}
```

```
function requestHandler(request) {
    if (!request || !request.kind) throw new Error('Absent Request or Kind');
    var deferredResponse = requestHandlerFactory(request.kind, request.params);
    return deferredResponse;
}

function requestHandlerFactory(kind, params) {
    // It includes mapping all your backend services
    switch (kind) {
        case 'CreateHouseFacade': return myService(params)
        default: throw new Error(`Invalid kind ${kind} was specified`);
    }
}
```

La función `myService` contiene su código de lógica empresarial que `CreateHouseFacade` solicitud `CreateHouseFacade` .

Lea Cola de Firebase en línea: https://riptutorial.com/es/firebase/topic/7619/cola-de-firebase

Capítulo 9: Cómo utilizar la base de datos Firebase para mantener una lista de usuarios de autenticación Firebase

Examples

Cómo guardar datos de perfil de usuario

Cada usuario autenticado tiene un `uid` Firebase que es único en todos los proveedores y se devuelve en el resultado de cada método de autenticación.

Una buena manera de almacenar los datos de su usuario es crear un nodo para mantener todos los datos de los usuarios y protegerlos utilizando sus reglas de seguridad.

- Base de datos

```
{
   "users": {
      "uid1" : {
         "name": "Steve",
         "surname": "Jobs"
      },
      "uid2" : {
         "name": "Bill",
         "surname": "Gates"
      }
   }
}
```

- seguridad

```
{
    "rules": {
        "users": {
            "$uid": {
                // If node's key matches the id of the auth user
                ".write": "$uid == auth.uid"
            }
        }
    }
}
```

El `$uid` en las reglas anteriores es una llamada "variable dólar", que garantiza que las reglas bajo este se apliquen a todos los nodos secundarios de `users` . Para obtener más información, consulte la documentación sobre el uso de $ Variables para capturar segmentos de ruta .

¿Por qué guardar datos de usuario en la base de datos?

La autenticación Firebase permite a los usuarios de su aplicación iniciar sesión con proveedores sociales o con su correo electrónico + contraseña. ¿Pero qué sucede si desea almacenar información adicional sobre un usuario, más allá de lo que la autenticación Firebase le permite especificar?

¿O qué sucede si desea mostrar una lista de los usuarios en su aplicación? Firebase Authentication no tiene una API para esto.

La mayoría de los desarrolladores resuelven este problema almacenando la información adicional en una base de datos separada. Este tema trata sobre cómo almacenar dicha información en la base de datos en tiempo real de Firebase .

Manejo de datos de cuenta de usuario en la base de datos en tiempo real

El sistema de autenticación Firebase es la fuente de un usuario `uid` , `displayName` , `photoURL` y tal vez `email` . Las cuentas basadas en contraseñas establecen estos valores *persistentes* en el sistema de autenticación a través del método `.updateProfile` . Al almacenar estos valores en la base de datos en tiempo real, rDB, el nodo de `users` plantea el problema de los datos obsoletos. Los nombres para mostrar, por ejemplo, pueden cambiar. Para mantener estos valores sincronizados, use el almacenamiento local en concierto con `.onAuthStateChange` .

en cada `.onAuthStateChange`

- `getItem('displayName')` y `getItem('photoURL')`
- comparar con `user.displayName` y `user.photoURL`
- si es diferente
    - `setItem('displayName')` y `setItem('photoURL')`
    - `db.ref.child('users').update` los valores de `displayName` y / o `photoURL`

`.onAuthStateChange` en cada carga o recarga de la página, así como en cada cambio de estado de autenticación. Potencialmente se dispara a menudo, por ejemplo, aplicaciones de varias páginas. Sin embargo, leer y escribir en el almacenamiento local es sincrónico y muy rápido, por lo que no habrá un impacto notable en el rendimiento de la aplicación.

Lea Cómo utilizar la base de datos Firebase para mantener una lista de usuarios de autenticación Firebase en línea: https://riptutorial.com/es/firebase/topic/1729/como-utilizar-la-base-de-datos-firebase-para-mantener-una-lista-de-usuarios-de-autenticacion-firebase

Capítulo 10: Consola Firebase

Sintaxis

1. Ejemplo de Firebase Analytics.
2. Firebase Console Explicación para cada componente.

Parámetros

Firebase Analytics	Firebase analytics y sus diferentes componentes.
Consola Firebase	¿Cómo funciona? & ¿Cómo se muestran los detalles en el panel?

Observaciones

Este documento es muy útil para aquellos que son los principiantes de las analíticas de base de fuego. Esto será de gran ayuda para entender cómo funciona la analítica de base de fuego en el escenario diferente.

Examples

Base de fuego todo en uno

Consola Firebase información en detalle

Android: Ejemplo de Firebase Analytics

Pasos para Android:

- Descargar código desde el enlace
- Compruebe FirebaseAnalyticsActivity
- Eso es todo lo que entenderá cómo funcionan los análisis de base de fuego para los diferentes escenarios.

Lea Consola Firebase en línea: https://riptutorial.com/es/firebase/topic/6660/consola-firebase

Capítulo 11: Estructurando datos

Introducción

La base de datos de Firebase es una base de datos NoSQL que almacena sus datos en forma de objetos JSON jerárquicos. No hay tablas ni registros de ninguna forma como normalmente tendría una base de datos SQL, solo nodos que conforman una estructura de clave-valor.

Normalización de datos

Para tener una estructura de base de datos adecuadamente diseñada, los requisitos de los datos deben ser detallados y previsibles. La estructura en este caso debe normalizarse; Cuanto más plano sea el árbol JSON, más rápido será el acceso a los datos.

Examples

Normas

La forma incorrecta

Considere la siguiente estructura

```
{
  "users": {

    // Uniquely generated IDs for children is common practice,
    // it's actually really useful for automating child creation.
    // Auto-incrementing an integer for a key can be problematic when a child is removed.

    "-KH3Cx0KFvSQELIYZezv": {
      "name": "Jon Snow",
      "aboutMe": "I know nothing...",
      "posts": {
        "post1": {
          "body": "Different roads sometimes leads to the same castle",
          "isHidden": false
        },
        "post2": { ... },
        // Possibly more posts
      }
    },
    "-KH3Dx2KFdSLErIYZcgk": { ... },     // Another user
    // A lot more users here
  }
}
```

Este es un gran ejemplo de lo que ***NO se*** debe hacer. Estructuras multi-anidadas como la anterior pueden ser muy problemáticas y podrían causar un enorme retroceso en el rendimiento.

La forma en que Firebase accede a un nodo es mediante la descarga de todos los datos de los niños, y luego iterando sobre todos los nodos del mismo nivel (los hijos de todos los padres).

Ahora, imagine una base de datos con varios *usuarios* , cada uno con cientos (o incluso miles) de *publicaciones* . El acceso a una *publicación* en este caso podría potencialmente cargar cientos de megabytes de datos no utilizados. En una aplicación más complicada, el anidamiento podría ser más profundo que solo 4 capas, lo que resultaría en descargas e iteraciones más inútiles.

La manera correcta

Aplanando la misma estructura se vería así

```
{
  // "users" should not contain any of the posts' data
  "users": {
    "-KH3Cx0KFvSQELIYZezv": {
      "name": "Jon Snow",
      "aboutMe": "I know nothing..."
    },
    "-KH3Dx2KFdSLErIYZcgk": { ... },
    // More users
  },

  // Posts can be accessed provided a user key
  "posts": {
    "-KH3Cx0KFvSQELIYZezv": {     // Jon Snow's posts
      "post1": {
        "body": "Different roads sometimes leads to the same castle",
        "isHidden": false
      },
      "post2": { ... },
      // Possibly more posts
    },
    "-KH3Dx2KFdSLErIYZcgk": { ... },
    // other users' posts
  }
}
```

Esto ahorra una gran cantidad de sobrecarga al iterar sobre menos nodos para acceder a un objeto objetivo. Todos los *usuarios* que no tienen ninguna publicación no existirían en la rama de *publicaciones* , por lo que la iteración sobre los usuarios en *la forma incorrecta* anterior es completamente inútil.

Relaciones bidireccionales

El siguiente es un ejemplo de una base de datos universitaria simple y mínima que utiliza relaciones bidireccionales

```
{
  "students": {
    "-SL3Cs0KFvDMQLIYZEzv": {
      "name": "Godric Gryffindor",
      "id": "900130309",
      "courses": {
         "potions": true,
         "charms": true,
         "transfiguration": true,
      }
```

```
    },
    "-SL3ws2KvZQLTYMqzSas": {
      "name": "Salazar Slytherin",
      "id": "900132319",
      "courses": {
         "potions": true,
         "herbs": true,
         "muggleStudies": true,
      }
    },
    "-SL3ns2OtARSTUMywqWt": { ... },
    // More students here
  },

  "courses": {
    "potions": {
      "code": "CHEM305",
      "enrolledStudents": {
        "-SL3Cs0KFvDMQLIYZEzv": true,     // Godric Gryffindor
        "-SL3ws2KvZQLTYMqzSas": true,     // Salazar Slytherin
        // More students
      }
    },
    "muggleStuddies": {
      "code": "SOC215",
      "enrolledStudents": {
        "-SL3ws2KvZQLTYMqzSas": true,     // Salazar Slytherin
        "-SL3ns2OtARSTUMywqWt": true,     // Some other student
        // More students
      }
    },
    // More courses
  }
}
```

Tenga en cuenta que cada estudiante tiene una lista de *cursos* y cada curso tiene una lista de *estudiantes* inscritos.

La redundancia no siempre es un mal enfoque. Es cierto que cuesta espacio de almacenamiento y tener que lidiar con la actualización de varias entradas al eliminar o editar un nodo duplicado; sin embargo, en algunos escenarios donde los datos no se actualizan con frecuencia, tener relaciones bidireccionales podría facilitar significativamente el proceso de búsqueda / escritura.

En la mayoría de los escenarios donde parece necesaria una consulta similar a SQL, la solución suele ser invertir los datos y crear relaciones bidireccionales.

Considere una aplicación que use la base de datos anterior que requiere la capacidad de:

1. Listar los cursos que un estudiante determinado está tomando **y** ...
2. Listar todos los alumnos en un determinado curso.

Si la estructura de la base de datos hubiera sido unidireccional, sería increíblemente más lento escanear o consultar uno de los dos requisitos anteriores. En algunos escenarios, la redundancia hace que las operaciones frecuentes sean más rápidas y mucho más eficientes, lo que, a largo plazo, hace que el costo de las duplicaciones sea insignificante.

Lea Estructurando datos en línea: https://riptutorial.com/es/firebase/topic/8912/estructurando-datos

Capítulo 12: FirebaseUI

Observaciones

Firebase es un conjunto de productos integrados diseñados para ayudarlo a desarrollar su aplicación, aumentar la base de usuarios comprometidos y ganar más dinero. Incluye herramientas que lo ayudan a construir su aplicación, como una base de datos en tiempo real, almacenamiento de archivos y autenticación de usuarios, así como herramientas para ayudarlo a crecer y monetizar su aplicación, como notificaciones push, análisis, informes de fallos y enlaces dinámicos.

Puedes pensar en Firebase como un conjunto de ladrillos Lego que puedes usar para construir tu obra maestra. Al igual que los ladrillos, Firebase es relativamente poco partidista, ya que hay un número infinito de formas de combinar las piezas y no te vamos a decir que ciertas formas son incorrectas :)

FirebaseUI se basa en Firebase y proporciona a los desarrolladores enlaces móviles nativos simples, personalizables y listos para la producción sobre las primitivas de Firebase para eliminar el código repetitivo y promover las mejores prácticas de Google

En la analogía de Lego, FirebaseUI es un conjunto de kits predefinidos con instrucciones que puede sacar del estante y modificar para satisfacer sus necesidades. Puede ver cómo usamos los componentes individuales de Firebase para construir FirebaseUI porque FirebaseUI es de código abierto. FirebaseUI debe ser valorado: le estamos diciendo cómo creemos que los ladrillos deben ir juntos, por lo que tomamos algunas decisiones. Pero como FirebaseUI es de código abierto, puede ingresar y cambiar lo que estamos haciendo para que se adapte mejor a sus necesidades individuales.

Si estás construyendo una ciudad de Lego, prefieres sacar un montón de casas de una colección precompilada y modificarlas para adaptarlas a tus necesidades que comenzar desde cero y diseñar cada edificio a mano, ¿no?

FirebaseUI le permite hacer exactamente esto, por lo que lo incluimos en nuestras aplicaciones de ejemplo y ejemplos. Los desarrolladores (incluidos nosotros mismos) son perezosos: queremos la mejor reutilización de nuestro código y los ejemplos más concisos, y FirebaseUI nos permite proporcionar ejemplos de muy alta calidad que se traducen en experiencias de usuario realmente buenas a una fracción del costo de desarrollo.

Examples

Comenzando con FirebaseUI

FirebaseUI ofrece clientes Android , iOS y web . Puedes comenzar con ellos así:

Androide:

```
// app/build.gradle

dependencies {
    // Single target that includes all FirebaseUI libraries
    compile 'com.firebaseui:firebase-ui:0.5.2'

    // FirebaseUI Database only
    compile 'com.firebaseui:firebase-ui-database:0.5.2'

    // FirebaseUI Auth only
    compile 'com.firebaseui:firebase-ui-auth:0.5.2'
}
```

iOS:

```
# Podfile

# Pull in all Firebase UI features
pod 'FirebaseUI', '~> 0.5'

# Only pull in the "Database" FirebaseUI features
pod 'FirebaseUI/Database', '~> 0.5'

# Only pull in the "Auth" FirebaseUI features (including Facebook and Google)
pod 'FirebaseUI/Auth', '~> 0.5'

# Only pull in the "Facebook" login features
pod 'FirebaseUI/Facebook', '~> 0.5'

# Only pull in the "Google" login features
pod 'FirebaseUI/Google', '~> 0.5'
```

Web:

```
<!--Include FirebaseUI sources in HTML-->

<script src="https://www.gstatic.com/firebasejs/ui/live/0.5/firebase-ui-auth.js"></script>
<link type="text/css" rel="stylesheet"
href="https://www.gstatic.com/firebasejs/ui/live/0.5/firebase-ui-auth.css" />
```

Lea FirebaseUI en línea: https://riptutorial.com/es/firebase/topic/6418/firebaseui

Capítulo 13: FirebaseUI (Android)

Examples

Añadiendo las dependencias.

FirebaseUI es solo una biblioteca de código abierto de Google que proporciona enlaces de IU sencillos para Firebase Auth y Firebase Database.

Para comenzar a agregar FirebaseUI a su aplicación, agregue estas dependencias en el archivo `build.gradle` su aplicación:

```
android {
    // ...
}

dependencies {
    // Required for FirebaseUI Database
    compile 'com.google.firebase:firebase-database:9.4.0'
    compile 'com.firebaseui:firebase-ui-database:0.5.1'

    // FirebaseUI Auth only
    compile 'com.google.firebase:firebase-auth:9.4.0'
    compile 'com.firebaseui:firebase-ui-auth:0.5.1'

    // Single dependency if you're using both
    compile 'com.firebaseui:firebase-ui:0.5.1'
}

apply plugin: 'com.google.gms.google-services'
```

Poblando un ListView

Suponiendo que ya ha configurado una aplicación en Android Studio, agregue un `ListView` a un diseño (o salte si ya está hecho):

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

<!-- Your toolbar, etc -->

<ListView
    android:id="@+id/list_view"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />

</android.support.design.widget.CoordinatorLayout>
```

Ahora vamos a crear un modelo para los datos con los que vamos a poblar nuestro `ListView` :

```
public class Person {

    private String name

    public Person() {
        // Constructor required for Firebase Database
    }

    public String getName() {
        return name;
    }

}
```

Asegúrese de que su `ListView` tenga un ID, luego cree una referencia a él en su `Activity` y configure su adaptador a un nuevo `FirebaseListAdapter` :

```
public class MainActivity extends AppCompatActivity {

    // ...

    private ListView mListView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // Find the ListView
        mListView = (ListView) findViewById(R.id.list_view);

        /*
         * Create a DatabaseReference to the data; works with standard DatabaseReference
methods
         * like limitToLast() and etc.
         */
        DatabaseReference peopleReference = FirebaseDatabase.getInstance().getReference()
            .child("people");

        // Now set the adapter with a given layout
        mListView.setAdapter(new FirebaseListAdapter<Person>(this, Person.class,
                android.R.layout.one_line_list_item, peopleReference) {

            // Populate view as needed
            @Override
            protected void populateView(View view, Person person, int position) {
                ((TextView) view.findViewById(android.R.id.text1)).setText(person.getName());
            }
        });
    }
}
```

Una vez que haya hecho eso, agregue algunos datos a su base de datos y observe cómo se llena el `ListView` .

Lea FirebaseUI (Android) en línea: https://riptutorial.com/es/firebase/topic/6610/firebaseui--

android-

Capítulo 14: Funciones en la nube para Firebase

Introducción

Firebase lanzó su versión beta de Cloud Functions for Firebase, que es similar al uso de Cloud Functions en Google Cloud Platform.

Cloud Functions es un entorno alojado, privado y escalable de Node.js donde puede ejecutar código JavaScript. Firebase SDK for Cloud Functions integra la plataforma Firebase al permitirle escribir código que responde a eventos e invoca funcionalidades expuestas por otras características de Firebase.

Examples

Enviar correos electrónicos de bienvenida a los usuarios para suscribirse.

Use el repositorio de GitHub para obtener el código completo:
https://github.com/firebase/functions-samples/blob/master/quickstarts/email-users

- Copia o clona el repositorio en tu computadora.

Ahora ve a tu Consola Firebase

- Cree un Proyecto Firebase utilizando la Consola Firebase.
- Habilitar el proveedor de **Google** en la sección de **autenticación** .
- Pegue el fragmento de **inicialización web** desde: **Firebase Console> Descripción general> Agregue Firebase** a su aplicación web en **public / index.html** donde se encuentra el **TODO** .

```
 * TODO(DEVELOPER): Paste the initialization snippet from: Firebase Console > Overview > Add
Firebase to your web app. *

**********************************************************************************************
-->
 <script src="https://www.gstatic.com/firebasejs/3.7.3/firebase.js"></script>
 <script>
   // Initialize Firebase
   var config = {
     apiKey: "your apiKey",
     authDomain: "authDomain.firebaseapp.com",
     databaseURL: "https://databaseURL.firebaseio.com",
     storageBucket: "storageBucket.appspot.com",
     messagingSenderId: "messagingID"
   };
   firebase.initializeApp(config);
```

```
</script>
```

Instale Firebase CLI en su computadora

- Si aún no tiene instalado **NodeJS** , instálelo desde https://nodejs.org/en/ (Asegúrese de tener la versión actualizada de NodeJS instalada en su computadora).
- Abra el símbolo del sistema / terminal e instálelo con **npm install -g firebase-tools** y luego configúrelo con el **inicio de sesión de firebase**
- Para elegir su proyecto que creó ahora ==> Configure la CLI localmente usando **firebase use --add** y seleccione su proyecto en la lista.
- Instalar dependencias localmente ejecutando: **funciones de cd; npm instalar; discos compactos -**

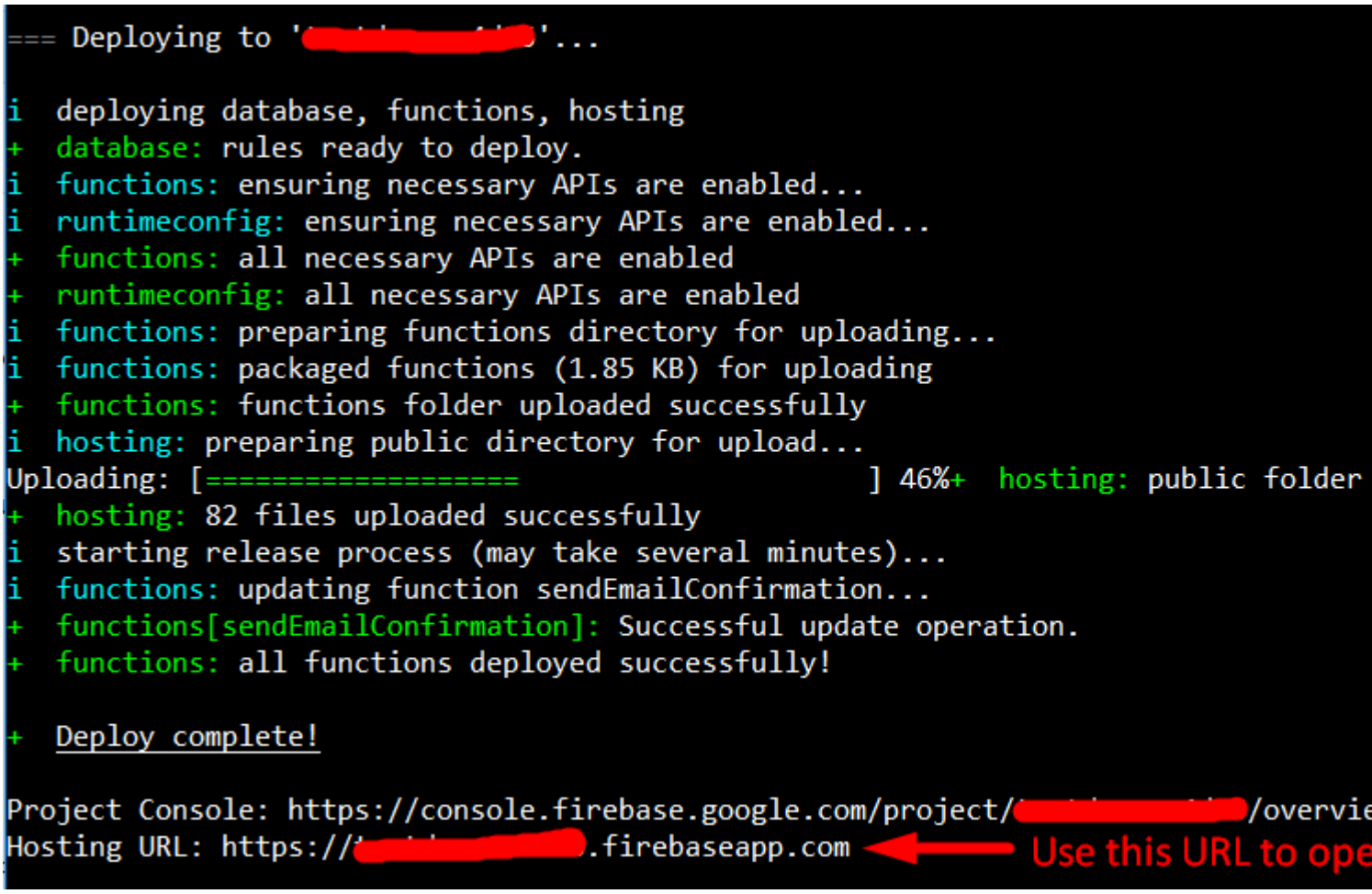
Establecer las variables de entorno de Google Cloud

- Configure las variables de entorno **gmail.email** y **gmail.password de** Google Cloud para que coincidan con el correo electrónico y la contraseña de la cuenta de Gmail utilizada para enviar correos electrónicos. Para esto, **abra el símbolo del sistema o la terminal** y escriba el siguiente comando CLI de Firebase:

  funciones de la base de fuego: config: set gmail.email = "myusername@gmail.com" gmail.password = "secretpassword"

Despliega el proyecto y prueba

- Para implementar el proyecto, abra **cmd / terminal** y use el comando **firebase deploy** para iniciar la implementación.



- Una vez hecho esto, use el comando para abrir el sitio en el **servidor de firebase open hosting: site** o hágalo manualmente desde la URL que se muestra.

Lea Funciones en la nube para Firebase en línea:
https://riptutorial.com/es/firebase/topic/9580/funciones-en-la-nube-para-firebase

Capítulo 15: Notificaciones push desde servidor personalizado

Introducción

Esto se puede hacer usando 2 métodos con **solicitud HTTP Post** , con **Firebase admin SDK** ejecutándose en su servidor. Aquí voy a discutir los dos.

Examples

Firebase Cloud Messaging HTTP Protocol

Desde su solicitud de servidor al enlace a continuación para enviar la notificación con algunos parámetros de solicitud

```
https://fcm.googleapis.com/fcm/send
```

Al solicitar agregar encabezados de la siguiente manera

```
Authorization    key=<Your_key_from_the_console>
Content-Type     application/json
```

El cuerpo de la solicitud varía

```
{
  "to" : <tokens or the topic>,
  "notification" : {
    "title":"This is a test title",
    "body":"This is the body"
  },
  "data": {
      //whatever key value payer you need to send
  }
}
```

Los parámetros a tomas Array of tokens like

```
["token1","token2",..........]
```

o una sola ficha como

```
"token"
```

o un nombre de tema que comience con **/ tema /** como

```
"/topic_name/"
```

Para condiciones de uso de múltiples temas usando || y operadores de && como

```
"/topic_name/ && /topic2/"
```

Utilizando Admin SDK (Nodo js)

Al principio, inicie el SDK de FireBase y el SDK de administración.

```
const functions = require('firebase-functions');
const admin = require('firebase-admin');

admin.initializeApp({
  credential: admin.credential.cert({
        //your admin credential certificate generated from the console. Follow this [link][1].
    }),
  databaseURL: "https:///<PROJECT_NAME>.firebaseio.com"
});
```

Cree una cadena JSON de carga útil como en el primer ejemplo.

```
var payload = {
              notification: {
                title: "Title of the notification,
                body: "Body of the notification",
              },
              data:{
                //required key value pair
              }
            };
```

Luego llame a diferentes métodos de envío para enviar la notificación.

Para el tema

```
admin.messaging().sendToTopic("/topic/", payload)
              .then(function(response) {
                console.log("Successfully sent message:", response);
              })
              .catch(function(error) {
                console.log("Error sending message:", error);
              });
            });
```

Para dispositivo

```
admin.messaging().sendToDevice(token, payload).then(response=>{
                       response.results.forEach((result, index) => {
                        const error = result.error;
                        if (error) {
                          console.error('Failure sending notification to', tokens, error);
                        } else{
```

```
                        console.log('Sucessfully sent to '+tokens);
                    }
                });
```

Lea Notificaciones push desde servidor personalizado en línea:
https://riptutorial.com/es/firebase/topic/10548/notificaciones-push-desde-servidor-personalizado

Capítulo 16: Reglas de la base de datos

Introducción

Con Firebase Realtime Database, las reglas de su base de datos son la seguridad del lado del servidor. Debe ser muy cuidadoso y consciente de quién tiene acceso a su base de datos. Es importante que nadie obtenga acceso a sus datos que no deberían.

De manera predeterminada, las reglas de la base de datos en tiempo real de Firebase permiten que cualquier usuario autenticado lea y escriba todos los datos, probablemente esto no sea lo que usted quiere que haga su aplicación.

Eche un vistazo a los ejemplos a continuación para diferentes escenarios.

Observaciones

La base de datos en tiempo real de Firebase proporciona un lenguaje de reglas flexible y basado en expresiones con una sintaxis similar a JavaScript para definir fácilmente cómo deben estructurarse sus datos, cómo deben indexarse y cuándo se pueden leer y escribir sus datos. Combinado con nuestros servicios de autenticación, puede definir quién tiene acceso a qué datos y proteger la información personal de sus usuarios del acceso no autorizado.

De forma predeterminada, las reglas de su base de datos requieren la autenticación Firebase y otorgan permisos completos de lectura y escritura a los usuarios autenticados. Las reglas predeterminadas aseguran que nadie pueda acceder a su base de datos antes de que tenga la oportunidad de configurar i

Documentacion oficial

https://firebase.google.com/docs/database/security/quickstart

Examples

Cómo configurar reglas

1. Entra en la consola de Firebase.
2. Elige tu proyecto
3. Haga clic en la sección Base de datos a la izquierda y luego seleccione la pestaña Reglas.

Si desea probar sus reglas de seguridad antes de ponerlas en producción, puede simular operaciones en la consola utilizando el botón Simular en la parte superior derecha del editor de reglas.

Las reglas por defecto

Las reglas predeterminadas requieren autenticación.
Permiten el acceso completo de lectura y escritura a los usuarios autenticados de su aplicación.
Son útiles si desea que los datos estén abiertos para todos los usuarios de su aplicación, pero no lo desean para el mundo.

```
// These rules require authentication
{
  "rules": {
    ".read": "auth != null",
    ".write": "auth != null"
  }
}
```

Cómo configurar tus archivos públicamente legibles y grabables

Solo define:

```
// These rules give anyone, even people who are not users of your app,
// read and write access to your database
{
  "rules": {
    ".read": true,
    ".write": true
  }
}
```

Puede ser útil durante el desarrollo, pero preste atención porque este nivel de acceso significa que **cualquiera puede leer o escribir en su base de datos** .

Cómo deshabilitar el acceso de lectura y escritura

Puede definir reglas privadas para deshabilitar el acceso de lectura y escritura a su base de datos por parte de los usuarios. Con estas reglas, **solo puede acceder a la base de datos cuando tiene privilegios administrativos (que puede obtener al acceder a la base de datos a través de la consola Firebase o al iniciar sesión desde un servidor)** .

```
// These rules don't allow anyone read or write access to your database
{
  "rules": {
    ".read": false,
    ".write": false
  }
}
```

Cómo conceder acceso solo a usuarios autenticados.

Este es un ejemplo de una regla que le otorga a cada usuario autenticado un nodo personal en `/users/$user_id` donde $ user_id es el ID del usuario obtenido a través de la **autenticación** .

```
// These rules grant access to a node matching the authenticated
// user's ID from the Firebase auth token
```

```
{
  "rules": {
    "users": {
      "$user_id": {
        ".read": "$user_id === auth.uid",
        ".write": "$user_id === auth.uid"
      }
    }
  }
}
```

Cómo permitir la lectura de un elemento específico del grupo, pero evitar la inclusión de miembros del grupo

Es una práctica común crear grupos de elementos mediante la creación de nodos de valor simple con ID de elemento como clave. Por ejemplo, podemos agregar un usuario al grupo "administradores" creando un nodo en **`/administrators/$user_id`** con un valor `true` . No queremos que nadie sepa quiénes son los administradores, por razones de seguridad, pero aún así queremos comprobar si un usuario autenticado es **administrador** . Con estas reglas podemos hacer precisamente eso:

```
{
  "rules": {
    "administrators": {
      // No one can list administrators
      ".read": "false",
      "$uid": {
        // Authenticated user can check if they are in this group
        ".read": "$uid === auth.uid",
        // Administrators can write
        ".write": "data.parent().child(auth.uid).val() === true",
        // Allow only add or delete, no duplicates
        ".validate": "!data.exists() || !newData.exists() || newData.isBoolean()",
      }
    }
  }
}
```

Lea Reglas de la base de datos en línea: https://riptutorial.com/es/firebase/topic/3352/reglas-de-la-base-de-datos

Capítulo 17: Reporte de Accidentes

Observaciones

Crash Reporting crea informes detallados de los errores en su aplicación.
Los errores se agrupan en grupos de trazas de pila similares y se clasifican según la gravedad del impacto en sus usuarios. Además de los informes automáticos, puede registrar eventos personalizados para ayudar a capturar los pasos que conducen a un bloqueo.

Crash Reporting se encuentra actualmente en versión beta, mientras que resolvemos algunos problemas conocidos en Android e iOS.

Documetantion oficial

https://firebase.google.com/docs/crash/

Examples

Configuración de informes de bloqueo en Android

1. Complete la parte de Instalación y configuración para conectar su aplicación a Firebase. Esto creará el proyecto en Firebase.

2. Agregue la dependencia de Firebase CrashReporting a su archivo `build.gradle` nivel de `build.gradle` :

```
compile 'com.google.firebase:firebase-crash:9.4.0'
```

Reportar el error en Android

Firebase Crash Reporting genera automáticamente informes de errores fatales (o excepciones no detectadas).

Puede crear su informe personalizado utilizando:

```
FirebaseCrash.report(new Exception("My first Android non-fatal error"));
```

Puede verificar el registro cuando FirebaseCrash inicializó el módulo:

> 07–20 08: 57: 24,442 D / FirebaseCrashApiImpl: **API de informes de FirebaseCrash inicializada** 07–20 08: 57: 24,442 I / FirebaseCrash: **Informes de FirebaseCrash inicializada** d com.google.firebase.crash.internal.zzg@3333d325 07–20 08: 57: 24.442 D / FirebaseApp: **Clase inicializada com.google.firebase.crash.FirebaseCrash.**

Y luego, cuando envió la excepción:

> 07–20 08: 57: 47.052 D / FirebaseCrashApiImpl: **java.lang. ThrowableException: Mi primer error no fatal de Android** 07–20 08: 58: 18.822 D / FirebaseCrashSenderServiceImpl: **Código de respuesta: 200** 07–20 08: 58: 18.822 D / FirebaseCrashSenderServiceImpl: **informe enviado**

Puede agregar registros personalizados a su informe con

```
FirebaseCrash.log("Activity created");
```

Lea Reporte de Accidentes en línea: https://riptutorial.com/es/firebase/topic/4669/reporte-de-accidentes

Capítulo 18: Usando Firebase con Nodo

Examples

Hello World Firebase Realtime Database en Node

Requisitos del sistema:

- Nodo JS

Empezando

1. Primero ve a la consola Firebase y crea un nuevo proyecto.
2. Después de crear el proyecto, en el proyecto, haga clic en el ícono de configuración junto al Nombre del proyecto en la barra lateral izquierda y seleccione Permisos.
3. En la página de permisos Haga clic en Cuentas de servicio en la barra lateral izquierda y luego haga clic en Crear cuenta de servicio
4. En la ventana emergente, ingrese el nombre de su cuenta de servicio y elija Rol de cuenta y seleccione Proporcionar una nueva clave privada y, a continuación, seleccione JSON y haga clic en Crear (Deje la opción Habilitar la delegación de dominios de la aplicación de Google sin marcar).
5. Cuando haga clic en crear, se descargará un archivo JSON con las credenciales de su cuenta, simplemente guarde el archivo en cualquier lugar de su sistema.
6. El siguiente paso es crear una base de datos en su consola Firebase para la cual vaya a la consola Firebase y haga clic en Base de datos en la barra lateral izquierda. Después de eso, simplemente cree un nuevo objeto de base de datos con nombre user_data con algún valor ficticio.
7. Ahora su proyecto de base de datos Firebase está configurado ahora simplemente copie el siguiente código en el directorio de su proyecto.

```
//Loading Firebase Package
var firebase = require("firebase");

/**
* Update your Firebase Project
* Credentials and Firebase Database
* URL
*/
firebase.initializeApp({
  serviceAccount: "<path to Firebase Credentials Json File>",
  databaseURL: "<Firebase Database URL>"
});  //by adding your credentials, you get authorized to read and write from the database


/**
* Loading Firebase Database and refering
* to user_data Object from the Database
*/
var db = firebase.database();
var ref = db.ref("/user_data");  //Set the current directory you are working in
```

```
/**
* Setting Data Object Value
*/
ref.set([
{
    id:20,
    name:"Jane Doe",
    email:"jane@doe.com",
    website:"https://jane.foo.bar"
},
{
    id:21,
    name:"John doe",
    email:"john@doe.com",
    website:"https://foo.bar"
}
]);

/**
* Pushing New Value
* in the Database Object
*/
ref.push({
    id:22,
    name:"Jane Doe",
    email:"jane@doe.com",
    website:"https://jane.foo.bar"
});

/**
* Reading Value from
* Firebase Data Object
*/
ref.once("value", function(snapshot) {
  var data = snapshot.val();   //Data is in JSON format.
  console.log(data);
});
```

8. Simplemente cambie con la URL del archivo de credenciales JSON (para empezar, copie el archivo de credenciales en la misma carpeta y en el archivo index.js solo agregue el nombre del archivo de credenciales).
9. El siguiente paso es cambiar el archivo index.js con la URL real de la base de datos de Firebase, podrá encontrar esta URL en la consola de Firebase en la pestaña Base de datos. La URL será como *https: //.firebaseio.com/* .
10. El paso final es hacer.

```
npm install firebase
```

11. Después de ejecutar el comando anterior, NPM instalará los paquetes necesarios necesarios para Firebase. Finalmente ejecutar y probar proyecto ejecutar.

```
node index.js
```

¿Qué hace realmente el proyecto?

El proyecto carga los datos de la base de datos Firebase basada en la nube. El proyecto también muestra cómo escribir y leer datos de un objeto de datos de Firebase.

Para ver cómo se actualizan sus datos en tiempo real, vaya a la consola, haga clic en el proyecto que realizó y, a la izquierda, presione Base de datos. Allí puede ver cómo se actualizan sus datos en tiempo real, junto con sus valores.

Firebase-cola y trabajador

Puede enviar tareas o datos a la base de datos en tiempo real de FireBase y ejecutar un trabajador que escucha la cola de FireBase para ejecutar un proceso en segundo plano.

Configurar base de fuego

1. Cree un proyecto Firebase en la consola Firebase, si aún no tiene uno. Si ya tiene un proyecto de Google existente asociado con su aplicación, haga clic en Importar Google Project. De lo contrario, haga clic en Crear nuevo proyecto.

2. Haga clic en el icono de configuración y seleccione Permisos.

3. Seleccione Cuentas de servicio del menú de la izquierda.

4. Haga clic en Crear cuenta de servicio.

   Ingrese un nombre para su cuenta de servicio.

   Opcionalmente, puede personalizar el ID del que se genera automáticamente a partir del nombre.

   Elija Proyecto> Editor de la lista desplegable de roles.

   Seleccione Proporcionar una nueva clave privada y deje el tipo de clave como JSON.

   Deje la opción Activar delegación de dominio de Google Apps sin seleccionar.

   Haga clic en crear

5. Cuando crea la cuenta de servicio, se descarga un archivo JSON que contiene las credenciales de su cuenta de servicio. Necesitará esto para inicializar el SDK en el servidor.

Servidor de configuración

Instale firebase-queue usando npm en su aplicación nodejs

```
npm install firebase firebase-queue --save
```

Una vez que haya instalado firebase y firebase-queue, puede comenzar creando una nueva Cola y pasándole su referencia de Firebase y una función de procesamiento.

Ahora creamos una tarea de cola de base de fuego desde la aplicación cuando se crea un nuevo

usuario y configuramos al trabajador para que escuche la tarea de cola de base de fuego y envíe un correo electrónico al correo de los usuarios creados.

* **server.js**

```
var app=express();
var Queue = require('firebase-queue'),
    Firebase = require('firebase');
```

Actualice sus credenciales del proyecto Firebase y la URL de la base de datos de Firebase

```
var firebase = Firebase.initializeApp({
    serviceAccount: "path/to/serviceAccountCredentials.json",
  databaseURL: "https://databaseName.firebaseio.com"
});
```

o puede ingresar las credenciales de base de fuego directamente como se muestra a continuación

```
var firebase = Firebase.initializeApp({
    serviceAccount: {
        projectId: "projectId",
        clientEmail: "foo@projectId.iam.gserviceaccount.com",
        privateKey: "-----BEGIN PRIVATE KEY-----\nkey\n-----END PRIVATE KEY-----\n"
    },
    databaseURL: "https://databaseName.firebaseio.com"
});


var refQueue = firebase.database().ref("queue/tasks");

createUser =  funtion(email, password){
    var user = {
        username: email,
        password: password
    };
    user = new db.users(user);
    user.save(function(err, user){
        if(!err){
            refQueue.push({case: "NEW_USER", data: user});
        }
    })
}

createUser("abc@abc.com", "password");
```

* **worker.js**

```
var Queue = require('firebase-queue'),
    Firebase = require('firebase');

//Update your Firebase Project Credentials and Firebase Database URL by one of the way
specified in server.js
var firebase = Firebase.initializeApp({
    serviceAccount: "path/to/serviceAccountCredentials.json",
  databaseURL: "https://databaseName.firebaseio.com"
```

```
});

var refQueue = firebase.database().ref("queue");

var queue = new Queue(refQueue, function(data, progress, resolve, reject) {
    switch(data.case){
        case "NEW_USER":
            sendMail(data.data.email);
            console.log("user created");
            //sendMail function is not an inbuilt function and will not work unless you define
and implement the function
            break;

    // Finish the task asynchronously
    setTimeout(function() {
        resolve();
    }, 1000);
});
```

ejecuta el servidor y el trabajador por separado y prueba alrededor con la cola de firebase

```
node server.js

node worker.js
```

Lea Usando Firebase con Nodo en línea: https://riptutorial.com/es/firebase/topic/6443/usando-firebase-con-nodo

Capítulo 19: Verificación de correo electrónico después de registrarse

Sintaxis

- Enviar verificación de correo electrónico a la dirección de correo electrónico del usuario registrado en el archivo. Firebase te permite personalizar lo que implica tu correo electrónico
- Cuando el correo electrónico llega a la cuenta de correo electrónico del usuario, el usuario hace clic en
- Usando el enrutador de su elección (se usa angulo-ui-enrutador en el ejemplo anterior), intercepte los parámetros en la URL.
- Mastique los parámetros utilizando la función `applyCode` en Firebase.
- Vea a continuación las funciones involucradas en el proceso anterior.

Parámetros

La función...	Hace
sendEmailVerification ()	Envía un correo electrónico de verificación a un usuario.
applyActionCode ()	Aplica el código de acción que cambia `emailVerified` de `false` a `true`

Observaciones

Lo anterior resume bastante bien cómo utilizar el esquema de verificación de correo electrónico con Firebase. Hasta ahora, es una de las formas más sencillas de verificar el correo electrónico que he visto.

Hay una pequeña explicación del ejemplo anterior disponible en Verificación de correo electrónico con Firebase 3.0 SDK.

Examples

Código de acción de verificación de proceso de envío - AngularJS

```
// thecontroller.js
$scope.sendVerifyEmail = function() {
    console.log('Email sent, whaaaaam!');
    currentAuth.sendEmailVerification();
  }

// where currentAuth came from something like this:
```

```
// routerconfig

....
templateUrl: 'bla.html',
resolve: {
    currentAuth:['Auth', function(Auth) {
      return Auth.$requireSignIn() // this throws an AUTH_REQUIRED broadcast
    }]
  }
...

// intercept the broadcast like so if you want:

....

$rootScope.$on("$stateChangeError", function(event, toState, toParams, fromState, fromParams,
error) {
      if (error === "AUTH_REQUIRED") {
        $state.go('login', { toWhere: toState });
       }
    });
....

// So user receives the email. How do you process the `oobCode` that returns?
// You may do something like this:

// catch the url with its mode and oobCode
.state('emailVerify', {
  url: '/verify-email?mode&oobCode',
  templateUrl: 'auth/verify-email.html',
  controller: 'emailVerifyController',
  resolve: {
    currentAuth:['Auth', function(Auth) {
      return Auth.$requireSignIn()
    }]
  }
})

// Then digest like so where each term is what they sound like:

.controller('emailVerifyController', ['$scope', '$stateParams', 'currentAuth', 'DatabaseRef',
  function($scope, $stateParams, currentAuth, DatabaseRef) {
    console.log(currentAuth);
    $scope.doVerify = function() {
      firebase.auth()
        .applyActionCode($stateParams.oobCode)
        .then(function(data) {
          // change emailVerified for logged in User
          toastr.success('Verification happened', 'Success!');
        })
        .catch(function(error) {
          $scope.error = error.message;
          toastr.error(error.message, error.reason, { timeOut: 0 });
        })
    };
  }
])
```

Lea Verificación de correo electrónico después de registrarse en línea: https://riptutorial.com/es/firebase/topic/3380/verificacion-de-correo-electronico-despues-de-

registrarse

Creditos

S. No	Capítulos	Contributors
1	Empezando con base de fuego	Ami Hollander, Community, Dan Levy, Devid Farinelli, ErstwhileIII, Gabriele Mariotti, RyanM, Sneh Pandya, TwiterZX, Vishal Vishwakarma
2	¿Cómo escucho los errores al acceder a la base de datos?	Frank van Puffelen, ThunderStruct
3	¿Cómo obtener el valor de la clave de inserción de la base de datos Firebase?	cutiko
4	¿Cómo usar FirebaseRecyclerAdapter en lugar de RecyclerAdapter?	Dhaval Solanki
5	Almacenamiento	Mike McDonald
6	Base de datos en tiempo real de Firbase con Android	Dhaval Solanki, Frank van Puffelen
7	Capacidades fuera de línea de Firebase	Francisco Durdin Garcia
8	Cola de Firebase	Vladimir Gabrielyan
9	Cómo utilizar la base de datos Firebase para mantener una lista de usuarios de autenticación Firebase	Devid Farinelli, eikooc, Frank van Puffelen, Ron Royston
10	Consola Firebase	Priyank Bhojak
11	Estructurando datos	ThunderStruct
12	FirebaseUI	Mike McDonald
13	FirebaseUI (Android)	Willie Chalmers III

14	Funciones en la nube para Firebase	Vishal Vishwakarma
15	Notificaciones push desde servidor personalizado	Aawaz Gyawali
16	Reglas de la base de datos	Frank van Puffelen, Gabriele Mariotti, riggaroo, Sasxa, Velko Ivanov
17	Reporte de Accidentes	Gabriele Mariotti
18	Usando Firebase con Nodo	Akshay Khale, Laurel, Noushad PP, Shiven
19	Verificación de correo electrónico después de registrarse	Rexford