EBook Gratis

APRENDIZAJE firebase-database

Free unaffiliated eBook created from
Stack Overflow contributors.

#firebase-database

Tabla de contenido

Acerca de

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

It is an unofficial and free firebase-database 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-database.

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: Comenzando con la base de datos firebase

Observaciones

Esta sección proporciona una descripción general de qué es la base de datos de base de fuego, y por qué un desarrollador puede querer usarla.

También debe mencionar cualquier tema grande dentro de la base de datos de base de fuego, y vincular a los temas relacionados. Dado que la Documentación para la base de datos de base de fuego es nueva, es posible que deba crear versiones iniciales de esos temas relacionados.

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 conectarlo 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. Al final, descargará un archivo `google-services.json` . Puedes descargar este archivo de nuevo en cualquier momento. (este archivo se encuentra en la configuración del proyecto en la consola Firebase).

5. Cambie Android studio View to Project y pegue el archivo google-service.json en la carpeta de aplicaciones

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:3.1.1'
    }
}
```

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'//THIS IS FOR ANALYTICS
  compile "com.google.firebase:firebase-database:11.0.2"
}

// BELOW STATEMENT MUST BE WRITTEN IN BOTTOM
apply plugin: 'com.google.gms.google-services'
```

Notas:

- Los datos no se pueden leer / escribir sin autenticar. Si lo quieres sin autenticación. Cambia las reglas en la base de datos de la consola de base de fuego.

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

- Añadir permiso de internet en Manifiesto
- Actualiza Google Play Services y Google Repository

Escribir valor simple en la base de datos

Primero, complete la instalación y configuración para conectar su aplicación a Firebase. Luego, desde cualquier lugar de tu clase, puedes escribir:

```
// Write a message to the database
FirebaseDatabase database = FirebaseDatabase.getInstance();
DatabaseReference myRef = database.getReference("message");

myRef.setValue("Hello, World!");
```

Se escribirá `Hello, Wold!` en el nodo de `message` , como se ve a continuación:

```
"your-project-parent" : {
    "message" : "Hello, World!"
}
```

Explicación

```
FirebaseDatabase database = FirebaseDatabase.getInstance();
```

El código anterior asignará la instancia de `FirebaseDatabase` al objeto de la `database` para su uso posterior.

```
DatabaseReference myRef = database.getReference("message");
```

El código anterior hará referencia `myRef` objeto `myRef` en el elemento secundario del `"message"` del elemento principal de su proyecto (en este ejemplo, es `"your-project-parent"`). Así que es `"your-project-parent/message"`

```
myRef.setValue("Hello, World!");
```

El código de arriba establecerá `"Hello, World!"` en camino referenciado por `myRef`

Asignar automáticamente el modelo personalizado a la estructura de datos

Después de configurar algunos datos en la base de datos y obtener una estructura que consta de varios nodos como este;

```
"user" : {
    "-KdbKcU2ptfYF2xKb5aO" : {
      "firstName" : "Arthur",
      "lastName" : "Schopenhauer",
      "userName" : "AphorismMan",
      "phone" : "+9022-02-1778",
      "gender": "M",
      "age" : 25
    },
    "-KdbQFjs9BDviuqQVHjY" : {
      "firstName" : "Werner",
      "lastName" : "Heisenberg",
      "userName" : "whereAmI",
      "phone" : "+9005-12-1901",
      "gender": "M",
      "age" : 75
    }
  }
```

Puedes categorizar estructuras de datos.

Creando clase

Crear una clase modelo para establecer en la base de datos.

```
@IgnoreExtraProperties
public class User {
    public String firstName;
    public String lastName;
    public String userName;
    public String phone;
    public String gender;
    public int age;

    public User() {
    }

    public User(String firstName, String lastName, String userName, String phone, String
gender, int age) {
        this.firstName = firstName;
        this.lastName = lastName;
        this.userName = userName;
        this.phone = phone;
        this.gender = gender;
        this.age = age;
    }
}
```

Algunas cosas para recordar al crear una clase modelo que desea asignar a sus datos:

1. Tienes que tener un constructor vacío.
2. El alcance de las Variables / Campos debe ser público, para que el DataSnapshot que regresa de la base de fuego pueda acceder a estos campos. Si no lo hace, cuando desea obtener datos, DataSnapshot no puede acceder a su modelo en la devolución de llamada y eso causará una excepción.
3. Los nombres de variables / campos deben coincidir con los de su estructura de datos.

Enviando a Firebase

Crear un objeto de usuario

```
User user = new User ( "Arthur","Schopenhauer","AphorismMan","+9022-02-1778","M",25)
```

y referencia

```
DatabaseReference databaseReference = FirebaseDatabase.getInstance().getReference();
```

Ahora tienes la referencia de tu base de datos. Cree un nodo de `user` con `databaseReference.child("user")` . Si lo hace `.push()` sus modelos se `.push()` bajo identificadores únicos creados al azar como el anterior, `"-KdbKcU2ptfYF2xKb5aO", "-KdbQFjs9BDviuqQVHjY"` .

```
databaseReference.child("user").push().setValue(user, new
DatabaseReference.CompletionListener() {
            @Override
            public void onComplete(DatabaseError databaseError, DatabaseReference
databaseReference) {
                Toast.makeText(getActivity(), "User added.", Toast.LENGTH_SHORT).show();

            }
        });
```

Si desea establecer sus datos bajo su clave específica, hágalo con `.child("yourSpecificKey")` lugar de `.push()` .

```
databaseReference.child("user").child("yourSpecificKey").setValue(user,...
```

Lea Comenzando con la base de datos firebase en línea: https://riptutorial.com/es/firebase-database/topic/3044/comenzando-con-la-base-de-datos-firebase

Capítulo 2: Base de datos en tiempo real de Firebase con Android

Examples

Integre la base de datos en tiempo real de Firebase con una aplicación de Android

Cómo implementar la base de datos Firebase Real-Time en aplicaciones Android.

Instalación / Instalación:

1. Primero, instale el SDK de Firebase (guía)
2. Registra tu proyecto utilizando la consola Firebase.
3. Después de completar con éxito los dos pasos anteriores, agregue la siguiente dependencia en su nivel de aplicación gradel.

```
compile 'com.google.firebase:firebase-database:9.2.1'
```

4. [Opcional] Configure las reglas de seguridad de su base de datos (referencia).

Ejemplo de implementación:

1. Declarar e inicializar la referencia de la base de datos.

```
FirebaseDatabase database = FirebaseDatabase.getInstance();
DatabaseReference myRef = database.getReference("message");
```

Más tarde puede crear diferentes referencias para acceder a diferentes nodos.

6. Escribir nuevos datos en la base de datos.

```
myRef.setValue("Writing Demo");
```

7. Leer datos de la base de 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());
 }
});
```

Lea Base de datos en tiempo real de Firebase con Android en línea: https://riptutorial.com/es/firebase-database/topic/6341/base-de-datos-en-tiempo-real-de-firebase-con-android

Capítulo 3: Base de datos FirebaseRealtime con Android

Examples

Añadir la base de datos en tiempo real en Android

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

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

```
compile 'com.google.firebase:firebase-database:9.2.1'
```

3. Configurar las reglas de la base de datos de Firebase

Ahora está listo para trabajar con la base de datos en tiempo real en Android.

Por ejemplo, escribe un mensaje de `Hello World` en la base de datos debajo de la clave de `message` .

```
// Write a message to the database
FirebaseDatabase database = FirebaseDatabase.getInstance();
DatabaseReference myRef = database.getReference("message");

myRef.setValue("Hello, World!");
```

Usando setValue para guardar datos

El método `setValue()` sobrescribe los datos en la ubicación especificada, incluidos los nodos secundarios.

Puede utilizar este método para:

1. Pasa los tipos que corresponden a los tipos JSON disponibles de la siguiente manera:

- Cuerda
- Largo
- Doble
- Booleano
- Mapa <Cadena, Objeto>
- Lista

2. Pase un objeto Java personalizado, si la clase que lo define tiene un constructor predeterminado que no toma argumentos y tiene captadores públicos para que se asignen

las propiedades.

Este es un ejemplo con un CustomObject.
Primero define el objeto.

```
@IgnoreExtraProperties
public class User {

    public String username;
    public String email;

    public User() {
        // Default constructor required for calls to DataSnapshot.getValue(User.class)
    }

    public User(String username, String email) {
        this.username = username;
        this.email = email;
    }
```

A continuación, obtenga la referencia de la base de datos y establezca el valor:

```
   User user = new User(name, email);
   DatabaseReference mDatabase mDatabase = FirebaseDatabase.getInstance().getReference();
   mDatabase.child("users").child(userId).setValue(user);
```

Ejemplo para inserción de datos o recuperación de datos de Firebase

Antes de comprender, es necesario seguir alguna configuración para la integración del proyecto con firebase.

1. Cree su proyecto en la consola Firebase y descargue el archivo google-service.json desde la consola y colóquelo en el módulo de nivel de aplicación de su proyecto. Siga el enlace para Crear proyecto en la consola.

2. Después de esto necesitamos agregar alguna dependencia en nuestro proyecto,

- Primero agregue la ruta de clase en nuestro nivel de proyecto gradle,

  ```
  classpath 'com.google.gms:google-services:3.0.0'
  ```

- Y después de eso aplique el complemento en el nivel de aplicación gradel, escríbalo a continuación de la sección de dependencia,

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

- Hay más dependencia que se requiere agregar en el nivel de aplicación gradle en la sección de dependencia

  ```
  compile 'com.google.firebase:firebase-core:9.0.2'

  compile 'com.google.firebase:firebase-database:9.0.2'
  ```

- Ahora comience a insertar datos en la base de datos de base de fuego. Primero se requiere

crear una instancia de

```
FirebaseDatabase database = FirebaseDatabase.getInstance();
```

después de la creación del objeto FirebaseDatabase, vamos a crear nuestra DatabaseReference para insertar datos en la base de datos,

```
DatabaseReference databaseReference = database.getReference().child("student");
```

Aquí el `student` es el nombre de la tabla, si la tabla existe en la base de datos, luego inserte los datos en la tabla; de lo contrario, cree uno nuevo con el nombre del estudiante. Después de esto, puede insertar datos usando `databaseReference.setValue();` funciona como siguiente,

```
HashMap<String,String> student=new HashMap<>();

student.put("RollNo","1");

student.put("Name","Jayesh");

databaseReference.setValue(student);
```

Aquí estoy insertando datos como hasmap. Pero también puede establecer como clase modelo,

- Comience a recuperar datos de base de fuego. Estamos usando addListenerForSingleValueEvent para leer el valor de la base de datos,

```
  senderRefrence.addListenerForSingleValueEvent(new ValueEventListener() {
          @Override
          public void onDataChange(DataSnapshot dataSnapshot) {
              if(dataSnapshot!=null && dataSnapshot.exists()){
                  HashMap<String,String>
studentData=dataSnapshot.getValue(HashMap.class);
                  Log.d("Student Roll Num "," : "+studentData.get("RollNo"));
                  Log.d("Student Name "," : "+studentData.get("Name"));
              }
          }

          @Override
          public void onCancelled(DatabaseError databaseError) {

          }
      });
```

Obtener valor / s de base de fuego

1. Crea una clase y agrega importaciones para analizar información:

```
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.IgnoreExtraProperties;

//Declaration of firebase references
private DatabaseReference mDatabase;

//Declaration of firebase atributtes
```

```
public String uID;
public String username;
public String email;

@IgnoreExtraProperties
public class User {

    //Default constructor
    public User() {

        //Default constructor required for calls to DataSnapshot.getValue(User.class)
        mDatabase = FirebaseDatabase.getInstance().getReference();

        //...
    }

    //...
}
```

2. Agregue `addListenerForSingleValueEvent()` a nuestra base de datos de referencia:

```
//Add new imports
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.ValueEventListener;

//...

public void getUser(String uID){

    //The uID it's unique id generated by firebase database
    mDatabase.child("users").child(uID).addListenerForSingleValueEvent(
            new ValueEventListener () {
            @Override
            public void onDataChange(DataSnapshot dataSnapshot) {
                // ...
            }

            @Override
            public void onCancelled(DatabaseError databaseError) {
                // Getting Post failed, log a message
            }
    });
}
```

3. Infle nuestra clase con información de base de `onDataChange()` en el evento `onDataChange()` :

```
        @Override
        public void onDataChange(DataSnapshot dataSnapshot) {

            //Inflate class with dataSnapShot
            Users user = dataSnapshot.getValue(Users.class);

            //...
        }
```

4. Finalmente podemos obtener diferentes atributos de la clase firebase como normalmente:

```
//User inflated
Users user = dataSnapshot.getValue(Users.class);

//Get information
this.uID = user.uID;
this.username = user.username;
this.email = user.email;
```

Mejores prácticas

1. La base de fuego admite 32 niveles secundarios diferentes, por lo que es sencillo escribir incorrectamente las referencias, para evadir esto, se crean las referencias privadas finales:

```
//Declaration of firebase references
//...
final private DatabaseReference userRef =
mDatabase.child("users").child("premium").child("normal").getRef();

//...

public void getUser(String uID){

    //Call our reference
    userRef.child(uID).addListenerForSingleValueEvent(
        new ValueEventListener () {
            @Override
            public void onDataChange(DataSnapshot dataSnapshot) {
                // ...
            }

            @Override
            public void onCancelled(DatabaseError databaseError) {
                // Getting Post failed, log a message
            }
    });
}
```

2. El evento `onCancelled()` se llama cuando el usuario no tiene acceso a esta referencia por las reglas de la base de datos. Agregue el código pertinente para controlar esta excepción si lo necesita.

Para más información visite la documentación oficial.

Lea Base de datos FirebaseRealtime con Android en línea: https://riptutorial.com/es/firebase-database/topic/6220/base-de-datos-firebaserealtime-con-android

Capítulo 4: Firebase Query

Introducción

Firebase Query puede usarse para ordenar una colección de datos basada en algunos atributos, así como restringida a la gran lista de elementos (para datos de chat similares) hasta un número adecuado para sincronizar con el cliente.

Al igual que con una referencia, puede recibir datos de una consulta utilizando el método on (). Solo recibirá eventos y DataSnapshots para el subconjunto de los datos que coinciden con su consulta.

Examples

Ejemplo de consulta de Firebase

```
private void loadData(){
        DatabaseReference dbRef = FirebaseDatabase.getInstance().getReference();

        Query dataQuery = dbRef.child("chat").orderByChild("id").equalTo("user1");
        dataQuery.addListenerForSingleValueEvent(new ValueEventListener() {
            @Override
            public void onDataChange(DataSnapshot dataSnapshot) {
                if (dataSnapshot.exists()) {
                    // dataSnapshot is the "issue" node with all children with id 0
                    for (DataSnapshot issue : dataSnapshot.getChildren()) {
                        // do something with the individual "issues"
                    }
                }
            }

            @Override
            public void onCancelled(DatabaseError databaseError) {

            }
        });
    }
```

Lea Firebase Query en línea: https://riptutorial.com/es/firebase-database/topic/10100/firebase-query

Capítulo 5: Hola Mundo!

Examples

Hola mundo en android

1. Completa la parte de instalación y configuración . Esto creará el proyecto en la consola Firebase y también instalará el SDK base en su aplicación de Android.
2. Agregue la dependencia de Firebase Realtime Database a su archivo `build.gradle` nivel de `build.gradle` :

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

3. Ahora escriba un mensaje de `Hello World` en la base de datos debajo de la clave de `message` .

```
// Write a message to the database
FirebaseDatabase database = FirebaseDatabase.getInstance();
DatabaseReference myRef = database.getReference("message");

myRef.setValue("Hello, World!");
```

4. Compruebe si el mensaje apareció en la `Realtime Database` en `Realtime Database` .

Hola mundo en iOS

1. Una vez que haya configurado Firebase para su Proyecto IOS siguiendo la documentación para configurar Firebase para IOS en otra documentación relacionada con FireBase, puede comenzar a usar Firebase.

2. Si aún no ha agregado el pod de la base de datos a su Podfile, hágalo ahora agregando el `pod Firebase/Database` para obtener la funcionalidad de la base de datos para Firebase

3. Inserte 'Hello World' en la base de datos, pero primero debe ir y editar las Reglas para su base de datos en el Tablero de su aplicación en firebase y cambiarlo a

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

   NOTA: Asegúrese de cambiar esto antes de comenzar la producción, ya que esto significa que cualquier persona puede leer y escribir en su base de datos, lo cual es un defecto de seguridad importante.

4. Ahora todo lo que tiene que hacer es importar Firebase en el archivo en el que está usando

firbase al usar `import Firebase` escriba la siguiente línea y debería ver un nodo 'Prueba' en la base de datos, expandirlo y debería tener un hijo de "Hola mundo" !!!!!!

```
FIRDatabase.database().reference().child("Test").setValue("Hello World")
```

FELICITA EN SU PRIMER BASE DE DATOS

Lea Hola Mundo! en línea: https://riptutorial.com/es/firebase-database/topic/8885/hola-mundo-

Capítulo 6: Lectura de datos

Examples

Comprender a qué datos hace referencia getReference ()

En este ejemplo, usamos esta base de datos:

```
"your-project-name" : {
    "users" : {
        "randomUserId1" : {
            "display-name" : "John Doe",
            "gender" : "male"
        }
        "randomUserId2" : {
            "display-name" : "Jane Dae",
            "gender" : "female"
        }
    },
    "books" {
        "bookId1" : {
            "title" : "Adventure of Someone"
        },
        "bookId1" : {
            "title" : "Harry Potter"
        },
        "bookId1" : {
            "title" : "Game of Throne"
        }
    }
}
```

Si usa la base de datos anterior, entonces:

- `FirebaseDatabase.getInstance().getReference()`

  apuntará a los datos principales de su proyecto, `"your-project-name"` . Por lo tanto, el `dataSnapshot` que adquirió contendrá todos los datos que contiene, incluidos todos los datos de `"users"` y datos de `"books"` .

- `FirebaseDatabase.getInstance().getReference("users")` y `FirebaseDatabase.getInstance().getReference().child("users")`

  tendrá el mismo resultado, apuntando a `"your-project-name/users"`

- `FirebaseDatabase.getInstance().getReference("users/randomUserId1")` y `FirebaseDatabase.getInstance().getReference().child("users/randomUserId1")` y `FirebaseDatabase.getInstance().getReference().child("users").child("randomUserId1")`

  tendrá el mismo resultado, apuntando a `"your-project-name/users/randomUserId1"`

  Nota: este ejemplo es necesario para comprender completamente qué datos están dentro del objeto dataSnapshot

Entender qué datos están dentro del objeto dataSnapshot

> Nota: primero debe saber a qué datos hace referencia getReference () antes de poder comprender completamente este ejemplo.

Existen tres métodos comunes para obtener sus datos de Firebase Realtime Database:

- `addValueEventListener()`
- `addListenerForSingleValueEvent()`
- `addChildEventListener()`

Cuando hablamos sobre qué datos están dentro del objeto `dataSnapshot` , entonces `addValueEventListener()` y `addListenerForSingleValueEvent()` son básicamente lo mismo. La única diferencia es que `addValueEventListener()` escucha los cambios realizados en los datos a los que se hace referencia, mientras que `addListenerForSingleValueEvent()` no lo es.

Así que consideremos que tenemos esta base de datos:

```
"your-project-name" : {
    "users" : {
        "randomUserId1" : {
            "display-name" : "John Doe",
            "gender" : "male"
        }
        "randomUserId2" : {
            "display-name" : "Jane Dae",
            "gender" : "female"
        }
    },
    "books" {
        "bookId1" : {
            "title" : "Adventure of Someone"
        },
        "bookId1" : {
            "title" : "Harry Potter"
        },
        "bookId1" : {
            "title" : "Game of Throne"
        }
    }
}
```

DataSnapshot producido por addValueEventListener y addListenerForSingleValueEvent

`dataSnapshot` producido por `addValueEventListener()` y `addListenerForSingleValueEvent()` contendrá valores de los datos exactos a los que se hace referencia. Al igual que cuando `ref` es punto a `"your-project-name"` entonces `dataSnapshot` debería ser:

```
... onDataChange(DataSnapshot dataSnapshot) {
    dataSnapshot.getKey(); // will have value of String: "your-project-name"
    for (DataSnapshot snapshot : dataSnapshot) {
        snapshot.getKey(); // will have value of String: "users", then "books"
        for (DataSnapshot deeperSnapshot : dataSnapshot) {
```

```
            snapshot.getKey();
            // if snapshot.getKey() is "users", this will have value of String:
"randomUserId1", then "randomUserId2"
            // If snapshot.getKey() is "books", this will have value of String: "bookId1",
then "bookId2"
        }
    }
}
```

DataSnapshot producido por addChildEventListener

`dataSnapshot` producido por `addChildEventListener()` contendrá valores de datos un nivel más profundo dentro de los datos a los que se hace referencia. Como en estos casos:

Cuando `ref` es punto a `"your-project-name"` entonces `dataSnapshot` debería ser:

```
... onChildAdded(DataSnapshot dataSnapshot, String s) {
    dataSnapshot.getKey(); // will have value of String: "users", then "books"
    for (DataSnapshot snapshot : dataSnapshot) {
        snapshot.getKey();
        // if dataSnapshot.getKey() is "users", this will have value of String:
"randomUserId1", then "randomUserId2"
        // If dataSnapshot.getKey() is "books", this will have value of String: "bookId1",
then "bookId2"
        for (DataSnapshot deeperSnapshot : dataSnapshot) {
            snapshot.getKey();
            // if snapshot.getKey() is "randomUserId1" or "randomUserId1", this will have
value of String: "display-name", then "gender"
            // But the value will be different based on key
            // If snapshot.getKey() is "books", this will have value of String: "title", but
the value will be different based on key
        }
    }
}
// dataSnapshot inside onChildChanged, onChildMoved, and onChildRemoved will have the same
data as onChildAdded
```

Lo más probable es que queramos usar `.getValue()` lugar de `getKey()` . Pero aquí utilizamos `getKey` porque siempre contendrá una cadena y no será necesario convertirlo en un objeto personalizado, un mapa u otro. Básicamente, cuando sabes a qué clave está apuntando `dataSnapshot` , puedes saber fácilmente qué valor contiene y analizarlo en tu propio objeto personalizado (o cualquier cosa)

Lea Lectura de datos en línea: https://riptutorial.com/es/firebase-database/topic/9242/lectura-de-datos

Capítulo 7: Reglas de base de datos en tiempo real de Firebase

Observaciones

Las reglas de la base de datos en tiempo real de Firebase determinan quién tiene acceso de lectura y escritura a su base de datos, cómo se estructuran sus datos y qué índices existen. Estas reglas se encuentran en los servidores Firebase y se aplican automáticamente en todo momento. Cada solicitud de lectura y escritura solo se completará si sus reglas lo permiten. De forma predeterminada, sus reglas están configuradas para que solo los usuarios autenticados tengan acceso completo de lectura y escritura a su base de datos. Esto es para proteger su base de datos contra el abuso hasta que tenga tiempo para personalizar sus reglas o configurar la autenticación.

Las reglas de la base de datos de Firebase tienen una sintaxis similar a JavaScript y vienen en cuatro tipos:

Rule Types	
.read	Describes if and when data is allowed to be read by users.
.write	Describes if and when data is allowed to be written.
.validate	Defines what a correctly formatted value will look like, whether i and the data type.
.indexOn	Specifies a child to index to support ordering and querying.

Examples

Autorización

Identificar a su usuario es solo una parte de la seguridad. Una vez que sepa quiénes son, necesita una forma de controlar su acceso a los datos en su base de datos. Las reglas de la base de datos de Firebase le permiten controlar el acceso de cada usuario. Por ejemplo, aquí hay un conjunto de reglas de seguridad que le permite a cualquiera leer la ruta `/foo/` , pero nadie le escribe:

```
{
  "rules": {
    "foo": {
      ".read": true,
```

```
      ".write": false
    }
  }
}
```

`.read` y `.write` reglas de cascada, por lo que otorga esta conjuntos de reglas acceso de lectura a los datos en ruta / foo /, así como cualquier ruta más profundas, como `/foo/bar/baz` . Tenga en cuenta que `.read` y `.write` reglas que permitan el acceso prevalecerá sobre otras reglas en la base de datos que no permiten el acceso; en otras palabras, todos aplicables, `.read` y `.write` reglas se ORed juntos). Por lo tanto, el acceso de lectura a `/foo/bar/baz` aún se concederá en este ejemplo, incluso si una regla en la ruta `/foo/bar/baz` evalúe como falsa.

Las Reglas de la base de datos de Firebase incluyen variables y funciones integradas que le permiten referirse a otras rutas, marcas de tiempo del lado del servidor, información de autenticación y más. Este es un ejemplo de una regla que otorga acceso de escritura para usuarios autenticados a `/users/<uid>/` , donde se obtiene el ID del usuario a través de la autenticación Firebase.

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

Validación de datos

La base de datos en tiempo real de Firebase no tiene esquemas. Esto facilita el cambio de cosas a medida que se desarrolla, pero una vez que su aplicación está lista para distribuir, es importante que los datos se mantengan consistentes. El lenguaje de reglas incluye un `.validate` regla que le permite aplicar la lógica de validación usando las mismas expresiones usadas para `.read` y `.write` reglas. La única diferencia es que todas las reglas de validación relevantes deben evaluarse como verdaderas para que se permita la escritura (en otras palabras, todas las reglas de `.validate` aplicables se `.validate` juntas para permitir una escritura de base de datos).

Esta regla impone que los datos escritos en `/foo/` deben ser una cadena de menos de 100 caracteres:

```
{
  "rules": {
    "foo": {
      ".validate": "newData.isString() && newData.val().length < 100"
    }
  }
}
```

Las reglas de validación tienen acceso a todas las mismas funciones y variables incorporadas como `.read` y `.write` reglas. Puede usarlos para crear reglas de validación que conozcan datos de

otras partes de su base de datos, la identidad de su usuario, la hora del servidor y mucho más.

Definir índices de base de datos.

La base de datos en tiempo real de Firebase permite ordenar y consultar datos. Para tamaños de datos pequeños, la base de datos admite consultas ad hoc, por lo que generalmente no se requieren índices durante el desarrollo. Sin embargo, antes de iniciar su aplicación, es importante especificar índices para cualquier consulta que tenga para asegurarse de que continúen funcionando a medida que su aplicación crezca.

Los índices se especifican utilizando la regla `.indexOn` . Aquí hay un ejemplo de declaración de índice que indexaría los campos de altura y longitud para una lista de dinosaurios:

```
{
  "rules": {
    "dinosaurs": {
      ".indexOn": ["height", "length"]
    }
  }
}
```

Lea Reglas de base de datos en tiempo real de Firebase en línea: https://riptutorial.com/es/firebase-database/topic/3348/reglas-de-base-de-datos-en-tiempo-real-de-firebase

Capítulo 8: Transacciones de base de datos en tiempo real de Firebase

Introducción

Las transacciones proporcionan un mecanismo para coordinar entre varias partes que podrían tener acceso a los mismos datos al mismo tiempo. Estas "partes" pueden ser diferentes instancias del mismo código como diferentes usuarios que ejecutan la misma aplicación o nodos en un clúster de servidores, partes del mismo programa o eventos diferentes programas como una aplicación de administración, una aplicación de "usuario final" y / o " backend "lógica del servidor.

Examples

Un contador distribuido

Imagine a muchos usuarios que ejecutan una aplicación web que intenta incrementar un contador en la base de datos. Cada usuario debe leer el conteo actual, agregar uno y escribir el valor actualizado. Para asegurarnos de que nadie lea el contador mientras alguien más está agregando uno, utilizamos una transacción:

```
ref.transaction(function(value){
  if (value === null) {
    // the counter doesn't exist yet, start at one
    return 1;
  } else if (typeof value === 'number') {
    // increment - the normal case
    return value + 1;
  } else {
    // we can't increment non-numeric values
    console.log('The counter has a non-numeric value: ' + value)
    // letting the callback return undefined cancels the transaction
  }
});
```

Lea Transacciones de base de datos en tiempo real de Firebase en línea: https://riptutorial.com/es/firebase-database/topic/9612/transacciones-de-base-de-datos-en-tiempo-real-de-firebase

Creditos

S. No	Capítulos	Contributors
1	Comenzando con la base de datos firebase	Abdul Wasae, Adarsh Madrecha, AtaerCaner , Community, ErstwhileIII, Gabriele Mariotti, koceeng, Nepster, Shiven, TwiterZX, Veeresh Charantimath
2	Base de datos en tiempo real de Firebase con Android	Dhaval Solanki, Krishna Kumar, ThunderStruct
3	Base de datos FirebaseRealtime con Android	Dhaval Solanki, Gabriele Mariotti, Merlí Escarpenter Pérez, ThunderStruct
4	Firebase Query	Dhaval Solanki
5	Hola Mundo!	noob, RyanM, ThunderStruct
6	Lectura de datos	koceeng
7	Reglas de base de datos en tiempo real de Firebase	Adarsh Madrecha, mckoss
8	Transacciones de base de datos en tiempo real de Firebase	Mike