

# Mobile Application Development

---

Produced  
by

David Drohan ([ddrohan@wit.ie](mailto:ddrohan@wit.ie))

Department of Computing & Mathematics  
Waterford Institute of Technology  
<http://www.wit.ie>



Waterford Institute *of* Technology  
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE



# Firebase Database Setup & Usage

---





# 1. Create your Firebase Project

The screenshot shows the Firebase console interface. On the left, a sidebar menu is visible with several sections: **Develop** (Authentication, Database, Storage, Hosting, Functions, ML Kit), **Quality** (Crashlytics, Performance, Test Lab...), **Analytics** (Dashboard, Events, Conversions, A...), **Grow**, and **Extensions**. The **Spark** section indicates a free \$0/month plan with an **Upgrade** button. The main content area is titled "HDip-Donation" and displays the message "Get started by adding Firebase to your app". It features icons for iOS, TV, Web, and Cloud Functions, along with a call-to-action "Add an app to get started". A large blue banner at the bottom states "Store and sync app data in milliseconds". The URL in the browser bar is [console.firebaseio.google.com/project/hdip-donation/overview](https://console.firebaseio.google.com/project/hdip-donation/overview).



# 1. Create your Firebase Project

The screenshot shows the Firebase console interface. On the left, a sidebar menu is open under the 'Develop' section, with 'Database' highlighted and a red box drawn around it. The main content area is titled 'Or choose Realtime Database' and features a yellow background with a circuit board graphic. It contains a section for 'Realtime Database' which describes it as 'Firebase's original database. Like Cloud Firestore, it supports real-time data synchronisation.' Below this is a blue 'Create database' button. At the bottom, there's a section titled 'More features for developers' with two small images: one of a circuit board and another of a banknote.

console.firebaseio.google.com/project/hdip-donation/database

Hdip-Donation - Overview - Firebase console

Go to docs

Firebase

Project Overview

Develop

Authentication

**Database**

Storage

Hosting

Functions

ML Kit

Quality

Crashlytics, Performance, Test La...

Analytics

Dashboard, Events, Conversions, A...

Grow

Extensions

Spark Free \$0/month

Upgrade

Or choose Realtime Database

Realtime Database

Firebase's original database. Like Cloud Firestore, it supports real-time data synchronisation.

[View the docs](#) [Learn more](#)

Create database

More features for developers



# 1. Create your Firebase Project

The screenshot shows the Firebase console interface. On the left, the navigation sidebar is visible with various project management sections like Project Overview, Develop, Quality, Analytics, Grow, and Extensions. The 'Database' section is highlighted with a red box. The main content area shows the 'Or choose Realtime Database' screen. A modal window titled 'Security rules for Realtime Database' is open. It contains instructions about defining data structure and securing data. Two options are presented: 'Start in locked mode' (radio button) and 'Start in test mode' (radio button, which is selected). Below the radio buttons is a code block showing the default security rules:

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

A warning message in an orange box states: 'Anyone with your database reference will be able to read or write to your database'. At the bottom of the modal are 'Cancel' and 'Enable' buttons.



# 1. Create your Firebase Project

The screenshot shows the Firebase console interface. On the left, a sidebar lists various services: Authentication, Database (which is highlighted with a red box), Storage, Hosting, Functions, ML Kit, Quality, Analytics, Grow, Extensions, and Spark. The main area is titled "Or choose Realtime Database". A modal window is open, titled "Security rules for Realtime Database". It contains instructions: "Once you have defined your data structure you will have to write rules to secure your data." Below this is a link "Learn more". Two radio button options are shown: "Start in locked mode" (selected) and "Start in test mode". The "locked mode" option includes a code snippet:

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

A note below the code says: "All third party reads and writes will be denied". At the bottom of the modal are "Cancel" and "Enable" buttons.



# 1. Create your Firebase Project

A screenshot of a web browser displaying the Firebase Database console for a project named "HDip-Donation". The left sidebar shows various services: Authentication, Database (which is highlighted with a red box), Storage, Hosting, Functions, ML Kit, Quality, Analytics, Grow, and Extensions. The main area is titled "Database" and "Realtime Database". It shows a single entry: "hdip-donation: null".

The screenshot shows the Firebase Database console interface. The left sidebar has a red box around the "Database" option under the "Develop" section. The main area displays the Realtime Database with one item: "hdip-donation: null".



# 1. Create your Firebase Project

The screenshot shows the Firebase Database Rules editor. A red box highlights the 'Rules' tab in the navigation bar. A prominent red warning message at the top states: "⚠️ Your security rules are defined as public, so anyone can steal, modify or delete data in your database". Below the warning, there is a code editor displaying the following security rules:

```
1  {
2    "rules": {
3      "donations": {
4        ".write": "auth.uid != null",
5        ".read": "auth.uid != null"
6      },
7      "user-donations": {
8        "$uid": {
9          ".write": "$uid === auth.uid",
10         ".read": "$uid === auth.uid"
11       }
12     }
13   }
14 }
```



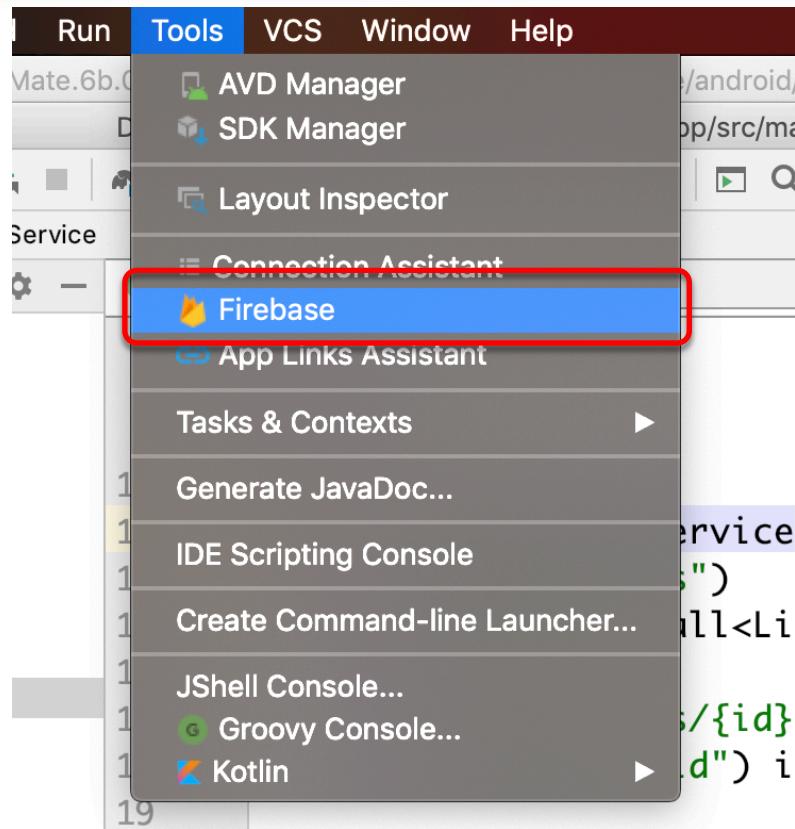
# 1. Create your Firebase Project

The screenshot shows the Firebase Realtime Database Rules editor. The left sidebar has sections for Project Overview, Develop (Authentication, Database, Storage, Hosting, Functions, ML Kit), Quality (Crashlytics, Performance, Test Lab...), Analytics (Dashboard, Events, Conversions, A...), Grow, and Extensions. The main area is titled 'Database' with a 'Realtime Database' dropdown. Below it are tabs for Data, Rules (which is selected and highlighted with a red box), Backups, and Usage. The central part is a code editor with a JSON-based rules configuration:

```
1  {
2    "rules": {
3      "donations": {
4        ".write": "auth.uid != null",
5        ".read": "auth.uid != null"
6      },
7      "user-donations": {
8        "$uid": {
9          ".write": "$uid === auth.uid",
10         ".read": "$uid === auth.uid"
11       }
12     }
13   }
14 }
```



## 2. Setup Firebase in your App





## 2. Setup Firebase in your App

The screenshot shows the Firebase Assistant tool window in Android Studio. The window has tabs for Assistant and Firebase, with the Firebase tab selected. It displays several service components:

- Analytics**: Measure user activity and engagement with free, easy, and unlimited analytics. [More info](#)
- Cloud Messaging**: Deliver and receive messages and notifications reliably across cloud and device. [More info](#)
- Authentication**: Sign in and manage users with ease, accepting emails, Google Sign-in, Facebook and other login providers. [More info](#)
- Realtime Database** (highlighted with a red box): Store and sync data in realtime across all connected clients. [More info](#)
- Storage**: Store and retrieve large files like images, audio, and video without writing server-side code. [More info](#)
- Remote Config**: Customize and experiment with app behavior using cloud-based configuration parameters. [More info](#)
- Test Lab**: Test your apps against a wide range of physical devices hosted in Google's cloud. [More info](#)
- App Indexing**: Get your app content into Google Search. [More info](#)
- Dynamic Links**: Create web URLs that can be shared to drive app installs and deep-linked into relevant content of your app. [More info](#)



## 2. Setup Firebase in your App

The screenshot shows the Firebase console interface. At the top, there's a navigation bar with tabs for 'Assistant' and 'Firebase'. Below the navigation bar is the Firebase logo and a brief introduction: 'Firebase gives you the tools and infrastructure from Google to help you develop, grow and earn money from your app.' There are links to 'Learn more' and 'More info'.

The main content area lists several services:

- Analytics**: Measure user activity and engagement with free, easy, and unlimited analytics. ([More info](#))
- Cloud Messaging**: Deliver and receive messages and notifications reliably across cloud and device. ([More info](#))
- Authentication**: Sign in and manage users with ease, accepting emails, Google Sign-in, Facebook and other login providers. ([More info](#))
- Realtime Database** (highlighted with a red box): Store and sync data in realtime across all connected clients. ([More info](#))
  - [Save and retrieve data](#)
- Storage**: Store and retrieve large files like images, audio, and video without writing server-side code. ([More info](#))
- Remote Config**: Customize and experiment with app behavior using cloud-based configuration parameters. ([More info](#))
- Test Lab**: Test your apps against a wide range of physical devices hosted in Google's cloud. ([More info](#))
- App Indexing**: Get your app content into Google Search. ([More info](#))
- Dynamic Links**: Create web URLs that can be shared to drive app installs and deep-linked into relevant content of your app. ([More info](#))
- Cloud Functions for Firestore**



## 2. Setup Firebase in your App

Assistant    Firebase

Firebase > Realtime Database

### Save and retrieve data

Our cloud database stays synced to all connected clients in realtime and remains available when your app goes offline. Data is stored in a JSON tree structure rather than a table, eliminating the need for complex SQL queries.

[Launch in browser](#)

- ① Connect your app to Firebase  
✓ Connected
- ② Add the Realtime Database to your app  
[Add the Realtime Database to your app](#)
- ③ Configure Firebase Database Rules
- ④ Write to your database

The Realtime Database provides a declarative rules language that allows you to define how your data should be structured, how it should be indexed, and when your data can be read from and written to. By default, read and write access to your database is restricted so only authenticated users can read or write data. To get started without setting up [Authentication](#), you can [configure your rules for public access](#). This does make your database open to anyone, even people not using your app, so be sure to restrict your database again when you set up authentication.

Retrieve an instance of your database using `getInstance()` and reference the location you want to write to.

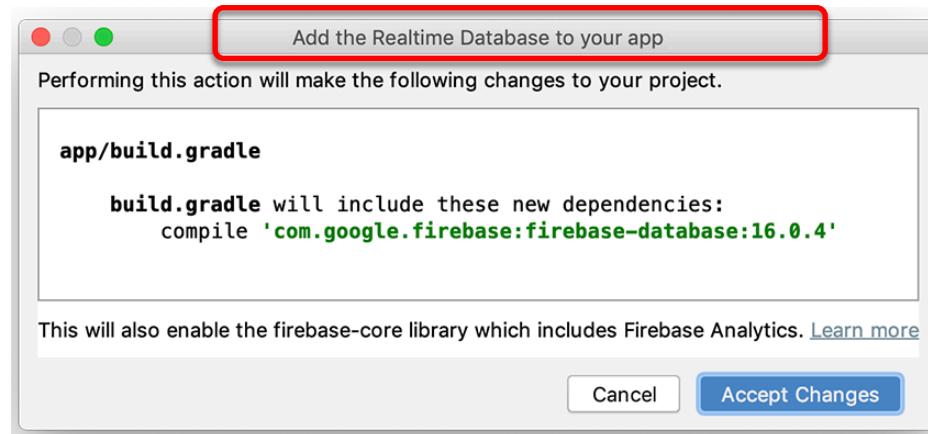
```
// Write a message to the database
 FirebaseDatabase database = FirebaseDatabase.getInstance();
 DatabaseReference myRef = database.getReference("message");

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

You can save a range of data types to the database this way, including Java objects. When you save an object the responses from any getters will be saved as



## 2. Setup Firebase in your App





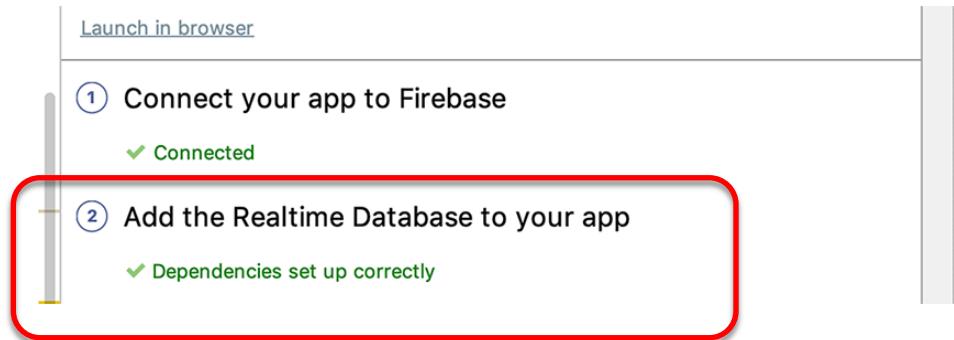
## 2. Setup Firebase in your App

[Launch in browser](#)

---

① Connect your app to Firebase  
✓ Connected

② Add the Realtime Database to your app  
✓ Dependencies set up correctly





### 3. Introduce Access Flow

---

<https://github.com/firebase/quickstart-android>



### 3. Introduce Access Flow – Write Basic Data

Retrieve an instance of your database using `getInstance()` and reference the location you want to write to.

Java  
Android

Kotlin  
Android

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

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

MainActivity.kt [

You can save a range of data types to the database this way, including Java objects. When you save an object the responses from any getters will be saved as children of this location.



### 3. Introduce Access Flow – Read Basic Data

To make your app data update in realtime, you should add a `ValueEventListener` to the reference you just created.

The `onDataChange()` method in this class is triggered once when the listener is attached and again every time the data changes, including the children.

Java      **Kotlin**  
Android    Android

```
// Read from the database
myRef.addValueEventListener(object : ValueEventListener {
    override fun onDataChange(dataSnapshot: DataSnapshot) {
        // This method is called once with the initial value and again
        // whenever data at this location is updated.
        val value = dataSnapshot.getValue(String::class.java)
        Log.d(TAG, "Value is: $value")
    }

    override fun onCancelled(error: DatabaseError) {
        // Failed to read value
        Log.w(TAG, "Failed to read value.", error.toException())
    }
})
```



### 3. Introduce Access Flow – Write Complex Data

To read or write data from the database, you need an instance of `DatabaseReference`:

Java	Kotlin
Android	<u>Android</u>

```
private lateinit var database: DatabaseReference
// ...
database = FirebaseDatabase.getInstance().reference
```



### 3. Introduce Access Flow – Write Complex Data

If you use a Java object, the contents of your object are automatically mapped to child locations in a nested fashion. Using a Java object also typically makes your code more readable and easier to maintain. For example, if you have an app with a basic user profile, your `User` object might look as follows:

Java      **Kotlin**  
Android    Android

```
@IgnoreExtraProperties  
data class User(  
    var username: String? = "",  
    var email: String? = ""  
)
```



User.kt

You can add a user with `setValue()` as follows:



### 3. Introduce Access Flow – Write Complex Data

```
private fun writeNewUser(userId: String, name: String, email: String?) {  
    val user = User(name, email)  
    database.child("users").child(userId).setValue(user)  
}
```

SignInActivity.kt

Using `setValue()` in this way overwrites data at the specified location, including any child nodes. However, you can still update a child without rewriting the entire object. If you want to allow users to update their profiles you could update the username as follows:

Java      [Kotlin](#)  
Android    [Android](#)

```
database.child("users").child(userId).child("username").setValue(name)
```





### 3. Introduce Access Flow – Read Complex Data

The following example demonstrates a social blogging application retrieving the details of a ‘post’

```
val postListener = object : ValueEventListener {
    override fun onDataChange(dataSnapshot: DataSnapshot) {
        // Get Post object and use the values to update the UI
        val post = dataSnapshot.getValue(Post::class.java)
        // ...
    }

    override fun onCancelled(databaseError: DatabaseError) {
        // Getting Post failed, log a message
        Log.w(TAG, "loadPost:onCancelled", databaseError.toException())
        // ...
    }
}
postReference.addValueEventListener(postListener)
```



### 3. Introduce Access Flow – Read & Write Lists

```
private fun writeNewPost(userId: String, username: String,  
                      title: String, body: String) {  
    // Create new post at /user-posts/$userid/$postid and at  
    // /posts/$postid simultaneously  
    val key = database.child("posts").push().key  
    if (key == null) {  
        Log.w(TAG, "Couldn't get push key for posts")  
        return  
    }
```

Firebase takes care of  
the Key

```
    val post = Post(userId, username, title, body)  
    val postValues = post.toMap()  
  
    val childUpdates = HashMap<String, Any>()  
    childUpdates["/posts/$key"] = postValues  
    childUpdates["/user-posts/$userId/$key"] = postValues  
  
    database.updateChildren(childUpdates)
```



### 3. Introduce Access Flow – Read & Write Lists

```
fun getAllPosts(userId: String) {  
    val postsList = ArrayList<Post>()  
    database.child("user-posts").child(userId)  
        .addValueEventListener(object : ValueEventListener {  
            override fun onCancelled(error: DatabaseError) {  
                println("Firebase Donation error : ${error.message}")  
            }  
  
            override fun onDataChange(snapshot: DataSnapshot) {  
                val children = snapshot.children  
                children.forEach {  
                    val donation = it.getValue<Post>(Post::class.java)  
                    postsList.add(donation!!)  
  
                    database.child("user-posts").child(userId)  
                        .removeEventListener(this)  
                }  
            }  
        })  
}
```

Triggered whenever list "user-posts" changes



# Listening for Child Events

---

- ❑ When working with lists, your application should listen for child events rather than the value events used for single objects
- ❑ Child events are triggered in response to specific operations that happen to the children of a node from an operation such as a new child added through the **push ()** method or a child being updated through the **updateChildren ()** method
- ❑ Each of these together can be useful for listening to changes to a specific node in a database
- ❑ In order to listen for child events on **DatabaseReference**, attach a **ChildEventListener**:



# Listening for Child Events

Listener	Event callback	Typical usage
ChildEventListener	onChildAdded()	Retrieve lists of items or listen for additions to a list of items. This callback is triggered once for each existing child and then again every time a new child is added to the specified path. The <b>DataSnapshot</b> passed to the listener contains the the new child's data.
	onChildChanged()	Listen for changes to the items in a list. This event fired any time a child node is modified, including any modifications to descendants of the child node. The <b>DataSnapshot</b> passed to the event listener contains the updated data for the child.
	onChildRemoved()	Listen for items being removed from a list. The <b>DataSnapshot</b> passed to the event callback contains the data for the removed child.
	onChildMoved()	Listen for changes to the order of items in an ordered list. This event is triggered whenever the <code>onChildChanged()</code> callback is triggered by an update that causes reordering of the child. It is used with data that is ordered with <code>orderByChild</code> or <code>orderByValue</code> .

# Viewing Your Data

The screenshot shows the Firebase Database console interface. The left sidebar contains navigation links for Project Overview, Authentication, Database (selected), Storage, Hosting, Functions, ML Kit, Quality, Analytics, Grow, and Extensions. The main area is titled "Database" and "Realtime Database". It features tabs for Data, Rules, Backups, and Usage. Below these tabs is a URL field with the value "https://hdip-donation.firebaseio.com/" and a copy icon. The main content area displays the database structure under the "hdip-donation" node:

```
hdip-donation
  +-- donations
  +-- user-donations
```

# Viewing Your Data

The screenshot shows the Firebase Realtime Database interface within the Firebase console. The left sidebar navigation includes 'Project Overview', 'Develop' (selected), 'Authentication', 'Database' (selected), 'Storage', 'Hosting', 'Functions', 'ML Kit', 'Quality' (Crashlytics, Performance, Test Lab...), 'Analytics' (Dashboard, Events, Conversions, A...), 'Grow', 'Extensions', and 'Spark' (Free \$0/month, Upgrade). The main area displays the 'Database' tab under 'Realtime Database'. The database structure is shown as:

```
hdip-donation
  - donations
    - Lsm0Esvo0x-e1ESpAhg
    - Lsm0J2An2KyOWbod6P_
    - Lsm0bu3jeoquDUEdF8G
    - LsnPobsXhodG3CZaYHM
    - LsnSVBknCuAPsvsPXx4
    - LsnS_OoZKDbhK0lzdpt
    - Lsom07dl3_CH0nHOJjc
    - Lsom3B7xFX1Ljs8ttbs
  - user-donations
    - MHdhscFcxNMelL1z1z3sA4EJZy92
    - Q4LKYEyrvP7WpNZ7ew6bCLaOJh1
```

# Viewing Your Data

The screenshot shows the Firebase Realtime Database interface within the Firebase console. The left sidebar contains navigation links for Project Overview, Authentication, Database (selected), Storage, Hosting, Functions, ML Kit, Quality, Analytics, Grow, Extensions, and Spark (Free \$0/month). The main area displays the database structure under the 'donations' node. A specific entry is expanded, showing fields: amount (1002), email (bob@wit.ie), message (a 1002 message), paymenttype (Paypal), uid (-Lsm0Esvo0x-e1ESpAhg), and upvotes (3). Other entries are listed below it.

```
hdipl-donation
  donations
    -Lsm0Esvo0x-e1ESpAhg
      amount: 1002
      email: "bob@wit.ie"
      message: "a 1002 message"
      paymenttype: "Paypal"
      uid: "-Lsm0Esvo0x-e1ESpAhg"
      upvotes: 3
    -Lsm0J2An2Ky0Wbod6P_
    -Lsm0bu3jeoquDUEdF8G
    -LsnPobsXhodG3CZaYHM
    -LsnSVBknCuAPsvsPXx4
    -LsnS_0oZKDBhK0lzDpT
    -Lsom07dl3_CH0nHOJjc
    -Lsom3B7xFX1Ljs8ttbs
```

# Viewing Your Data

The screenshot shows the Firebase Database console interface. The left sidebar contains navigation links for Project Overview, Authentication, Database, Storage, Hosting, Functions, and ML Kit. Below these are sections for Quality (Crashlytics, Performance, Test Lab), Analytics (Dashboard, Events, Conversions, A...), and Grow (Extensions). At the bottom, there's a Spark plan offer for \$0/month with an Upgrade button.

The main area displays the database structure for the project "HDip-Donation". The root node "hdip-donation" contains two main branches: "donations" and "user-donations".

- donations:** This branch contains several child nodes, each representing a donation record. The keys for these nodes are random strings starting with "-Lsm".
- user-donations:** This branch contains a single child node with the key "MHdhscFcxNMel1z1z3sA4EJZy92". This node has a child node "-LsnSVBknCuAPsvsPXX4" which contains the following data:
  - amount:** 1001
  - email:** "dave@dave.com"
  - message:** "a message"
  - paymenttype:** "Paypal"
  - uid:** "-LsnSVBknCuAPsvsPXX4"
  - upvotes:** 0



---

We'll look at how we store and retrieve  
‘Donations’ to Firebase in the Code  
walkthrough and the Lab



## References

---

<https://console.firebaseio.google.com/>

<https://firebase.google.com/>

<https://firebase.google.com/docs/database/android/start>

<https://github.com/firebase/quickstart-android>

Thanks.

