Mobile Application Development

Produced by

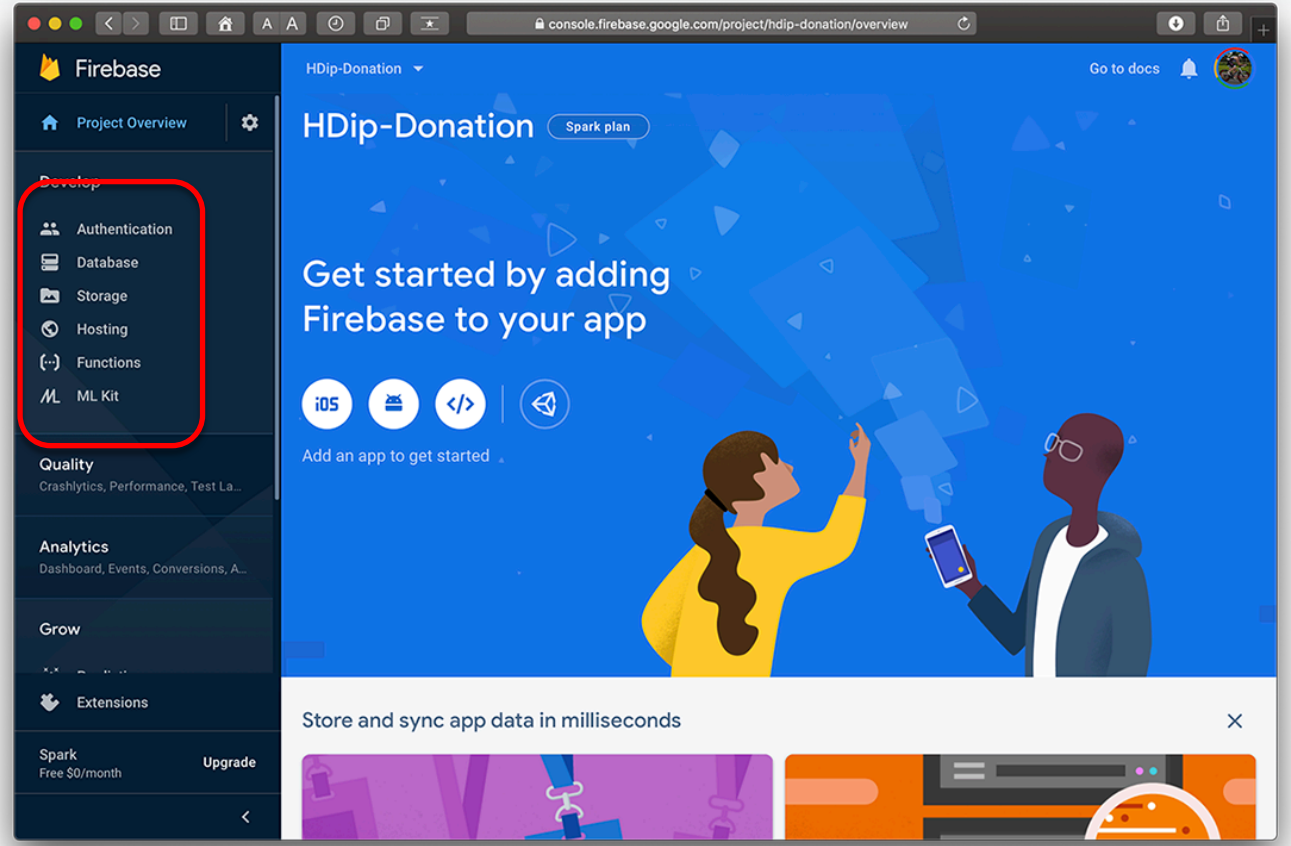
David Drohan (ddrohan@wit.ie)

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

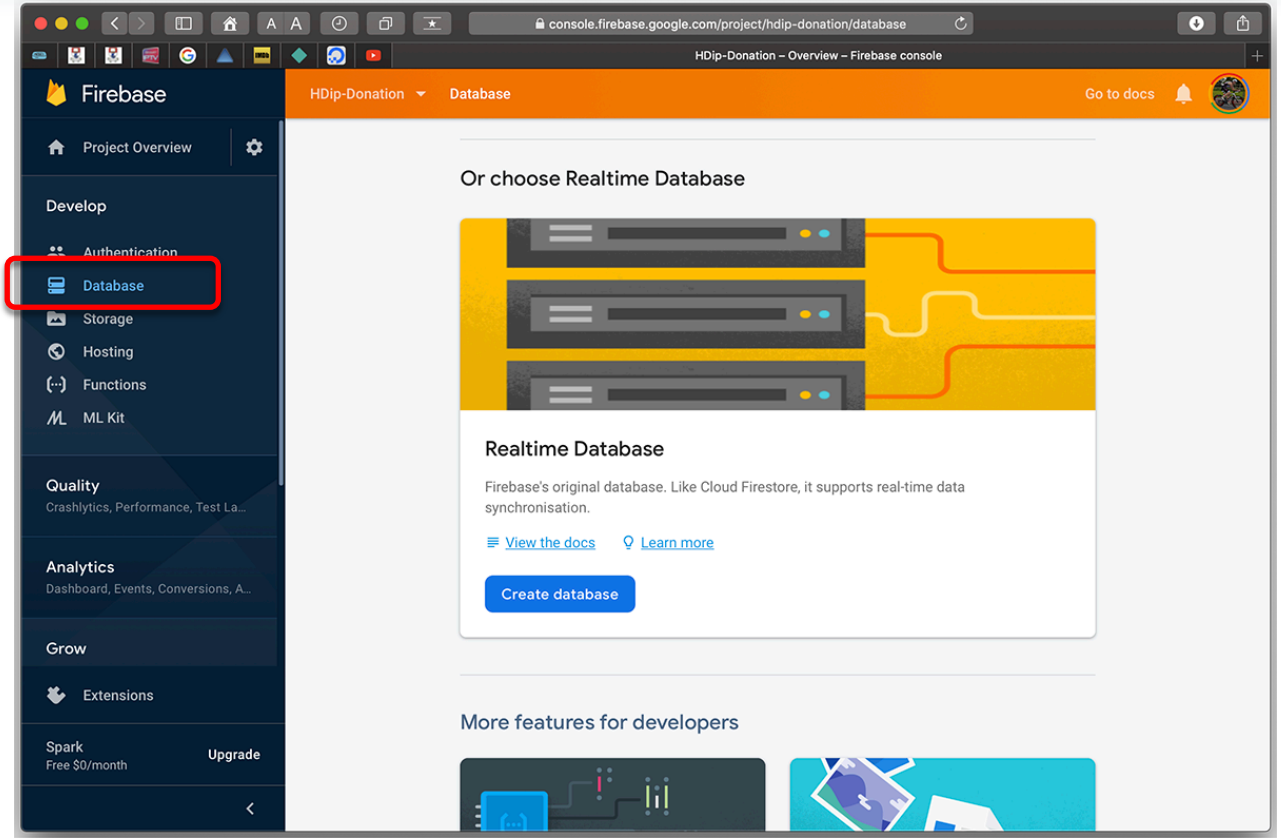
Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

Firebase Database Setup & Usage

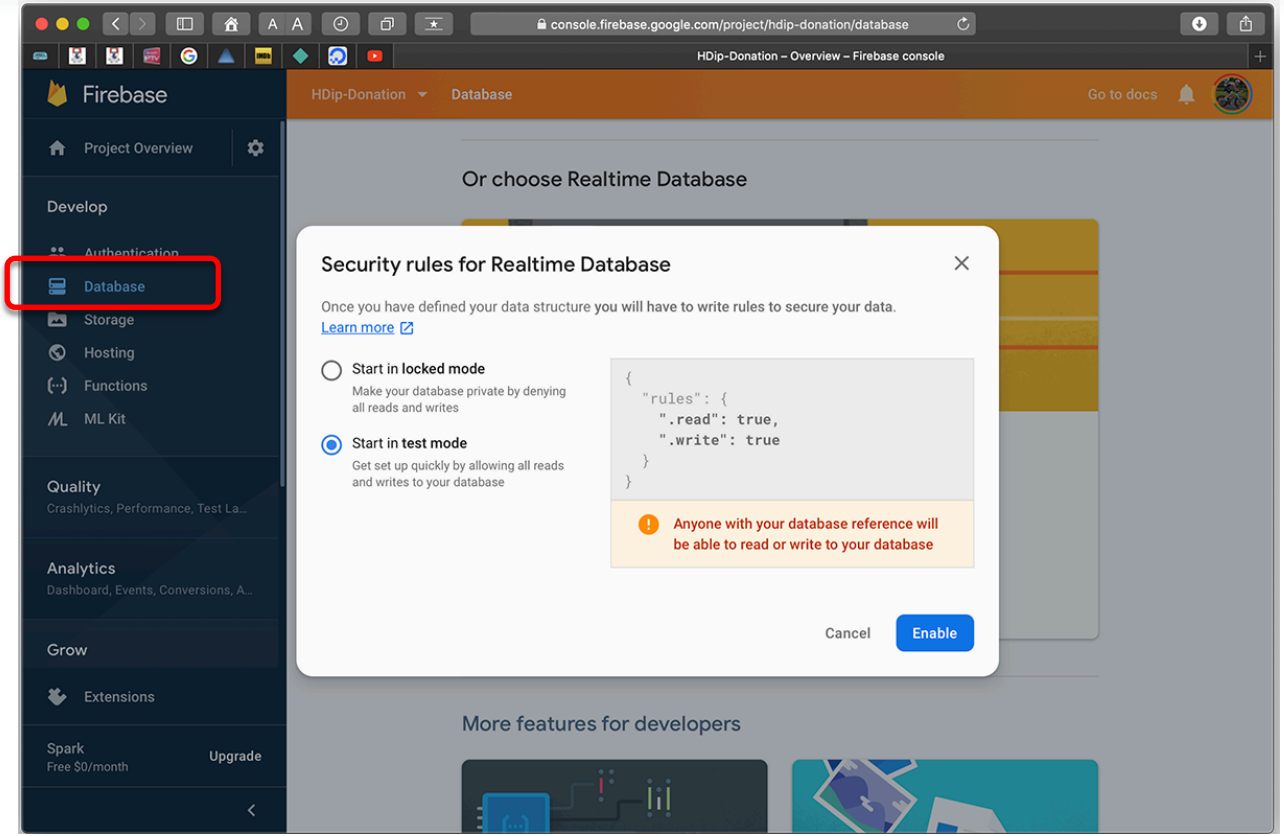
1. Create your Firebase Project



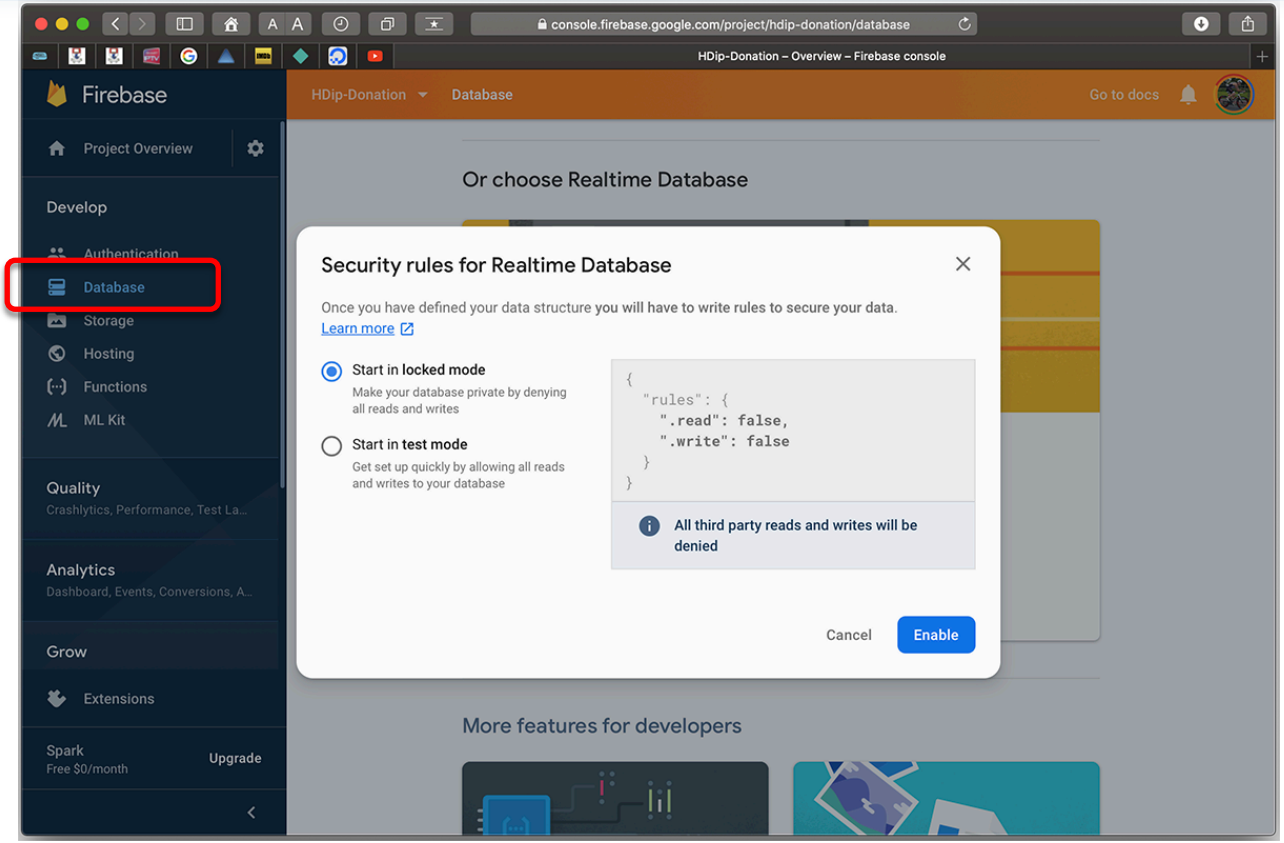
1. Create your Firebase Project



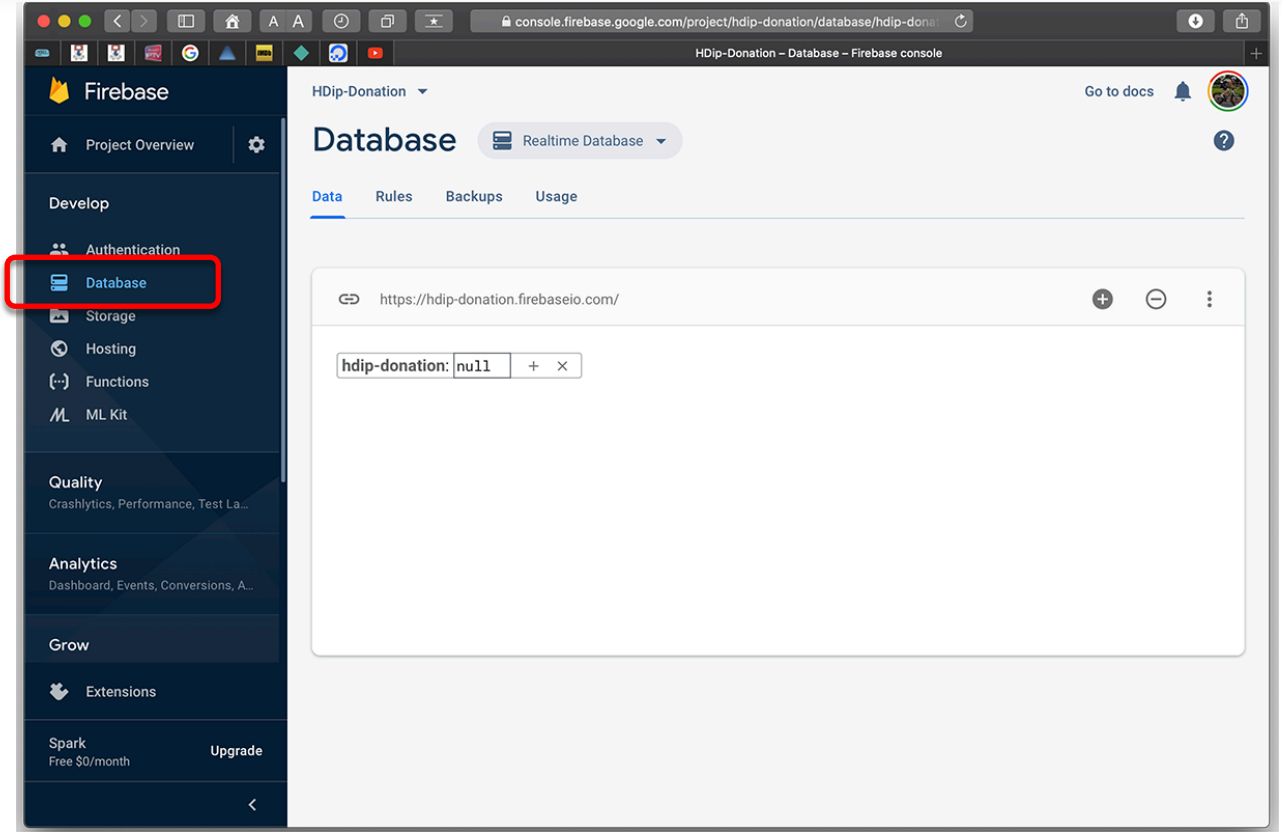
1. Create your Firebase Project



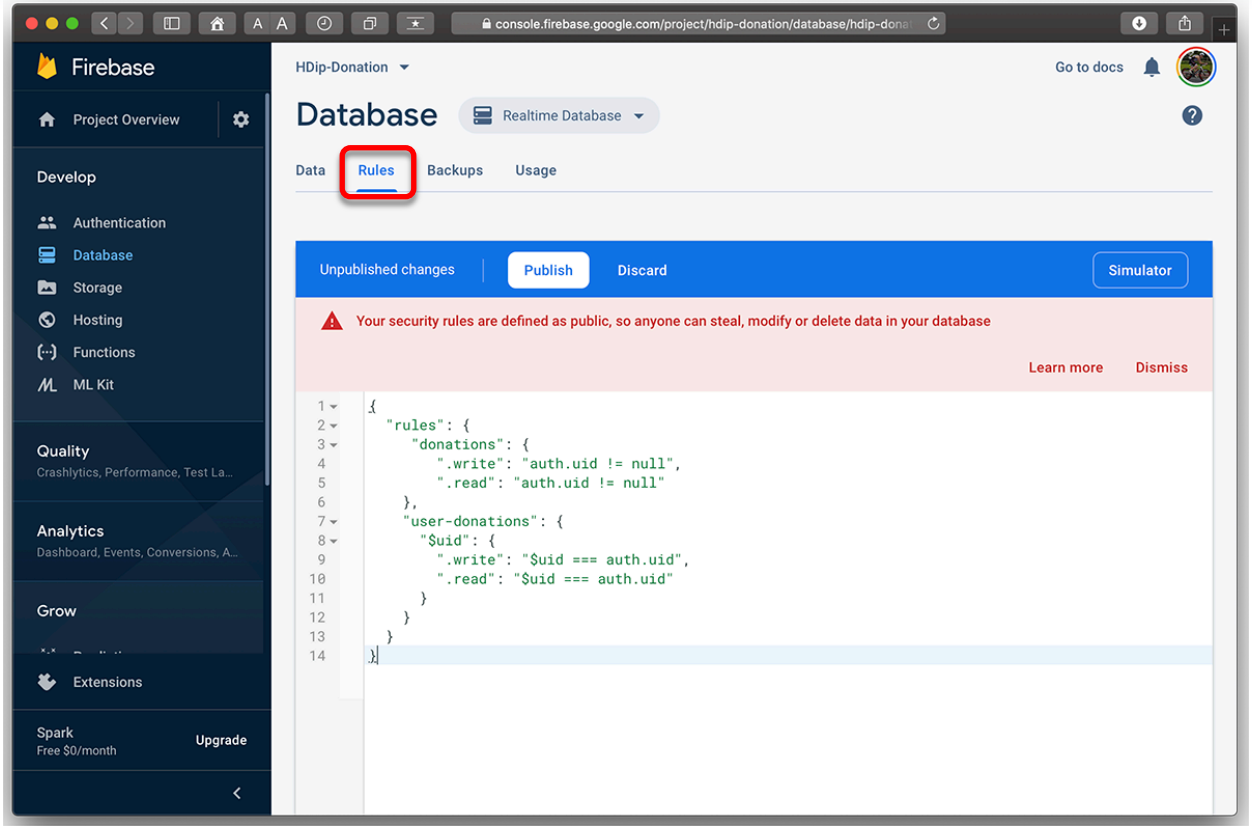
1. Create your Firebase Project



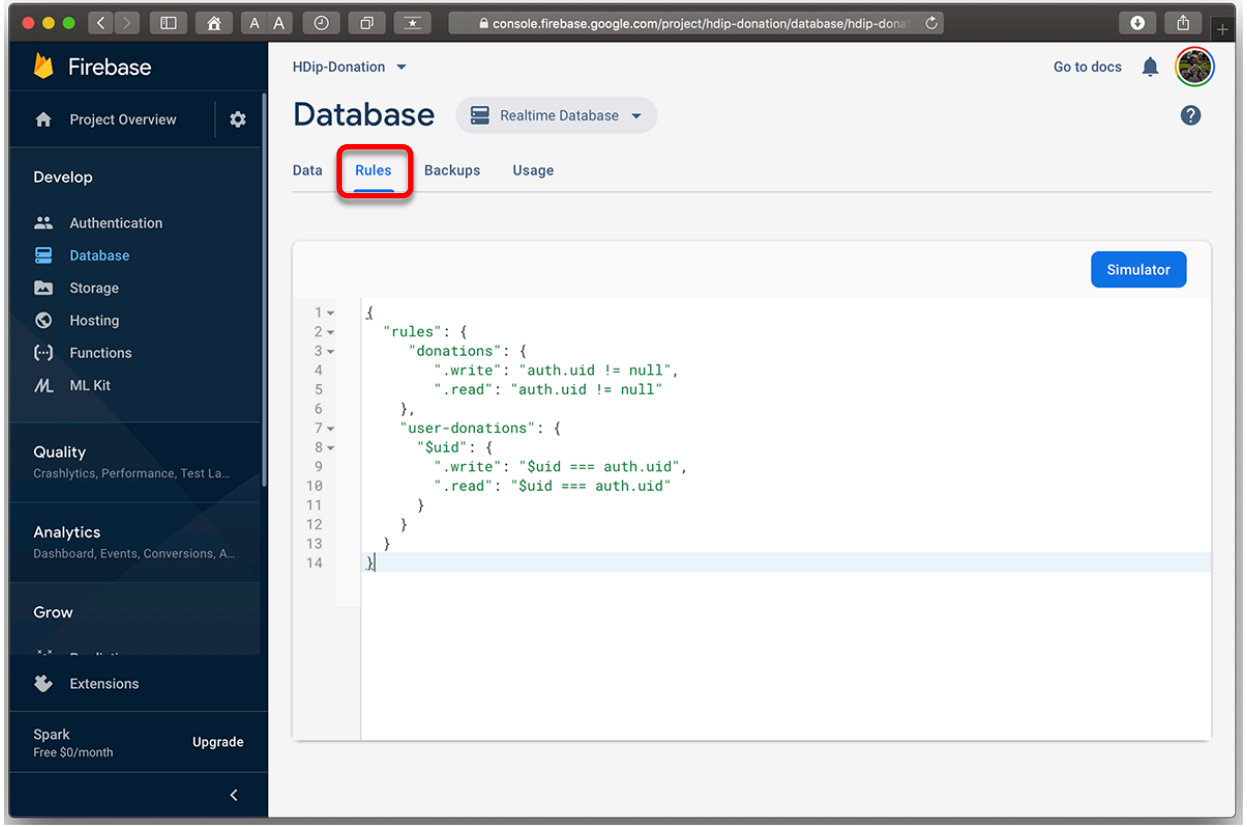
1. Create your Firebase Project



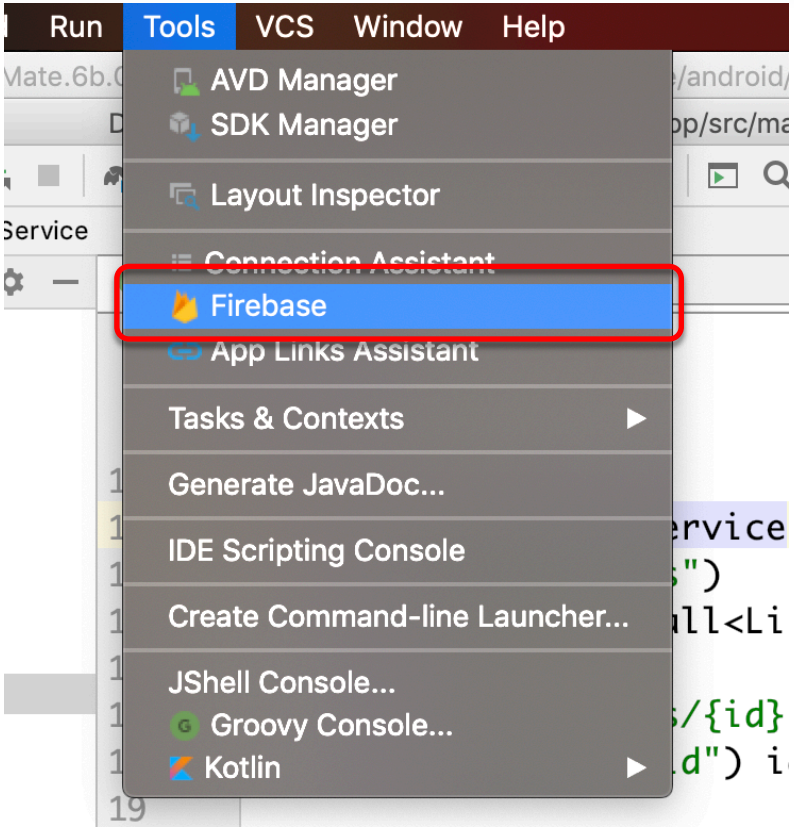
1. Create your Firebase Project



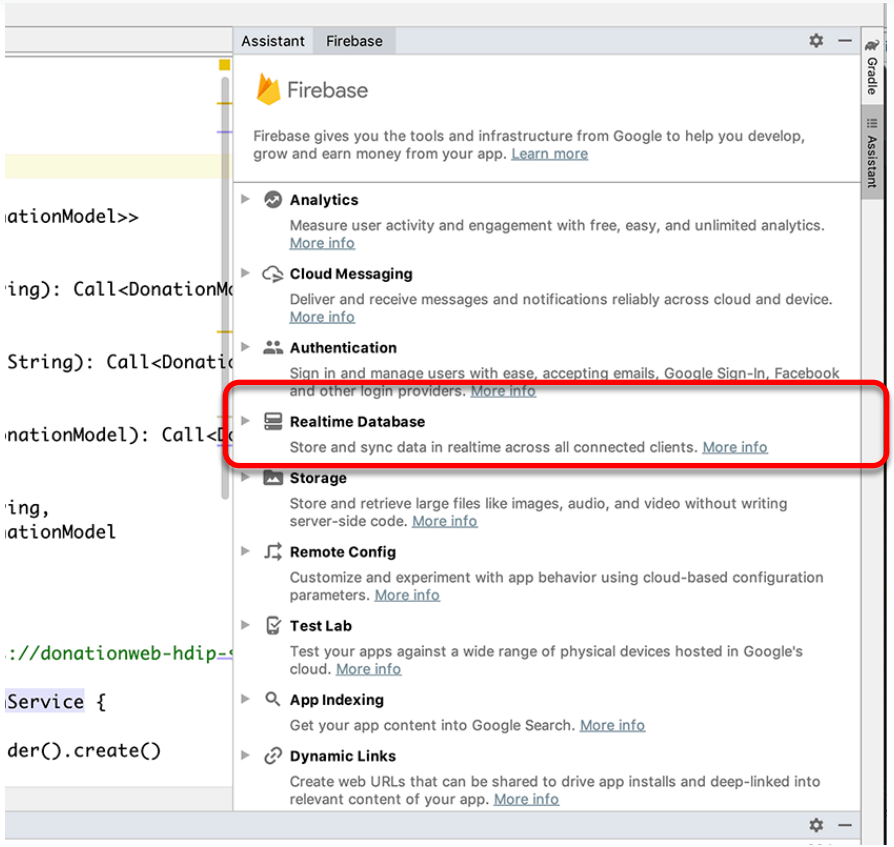
1. Create your Firebase Project



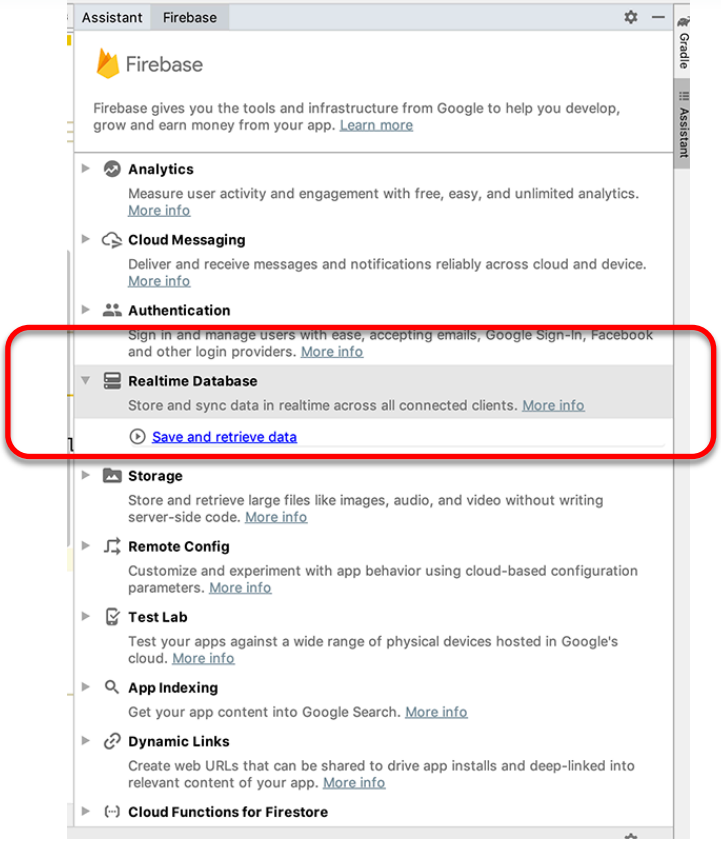
2. Setup Firebase in your App



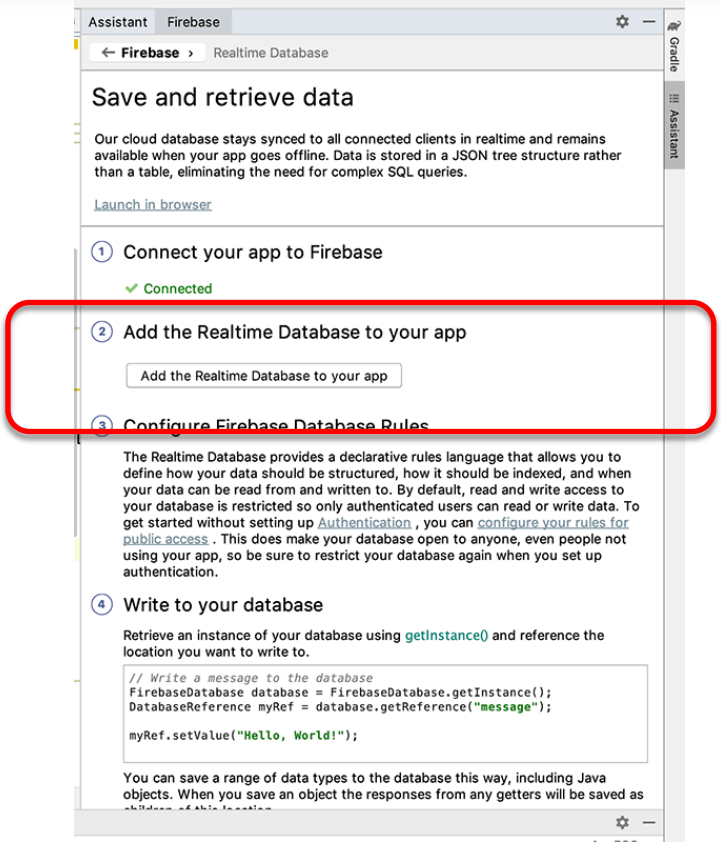
2. Setup Firebase in your App



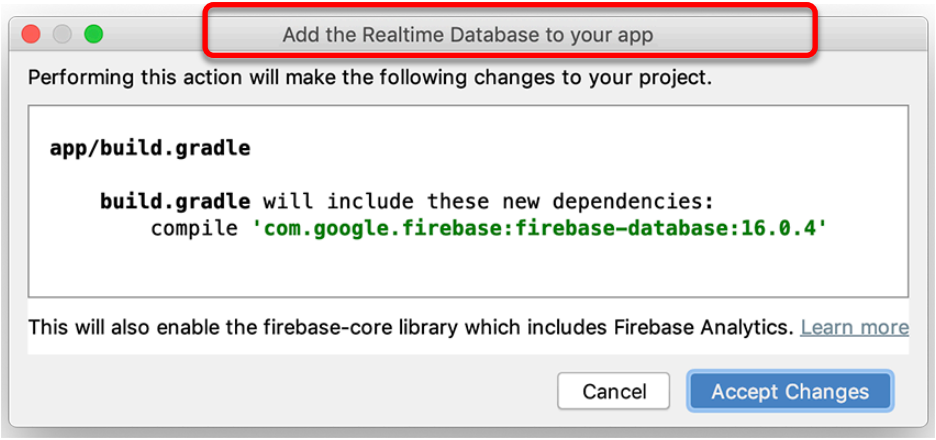
2. Setup Firebase in your App



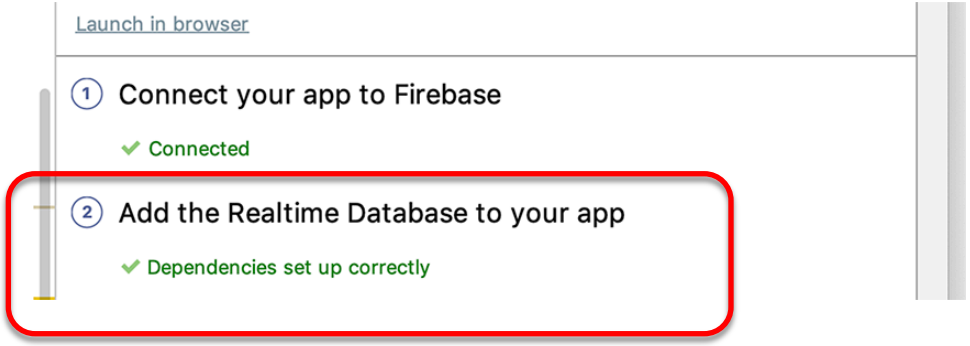
2. Setup Firebase in your App



2. Setup Firebase in your App



2. Setup Firebase in your App



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

```kotlin
// 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 Android | Kotlin Android

```kotlin
// 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 Android | Kotlin Android

```
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 Android | Kotlin Android

```kotlin
@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

```kotlin
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 Android | Kotlin Android

```kotlin
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
    }

    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)
}
```

Firebase takes care of the Key

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

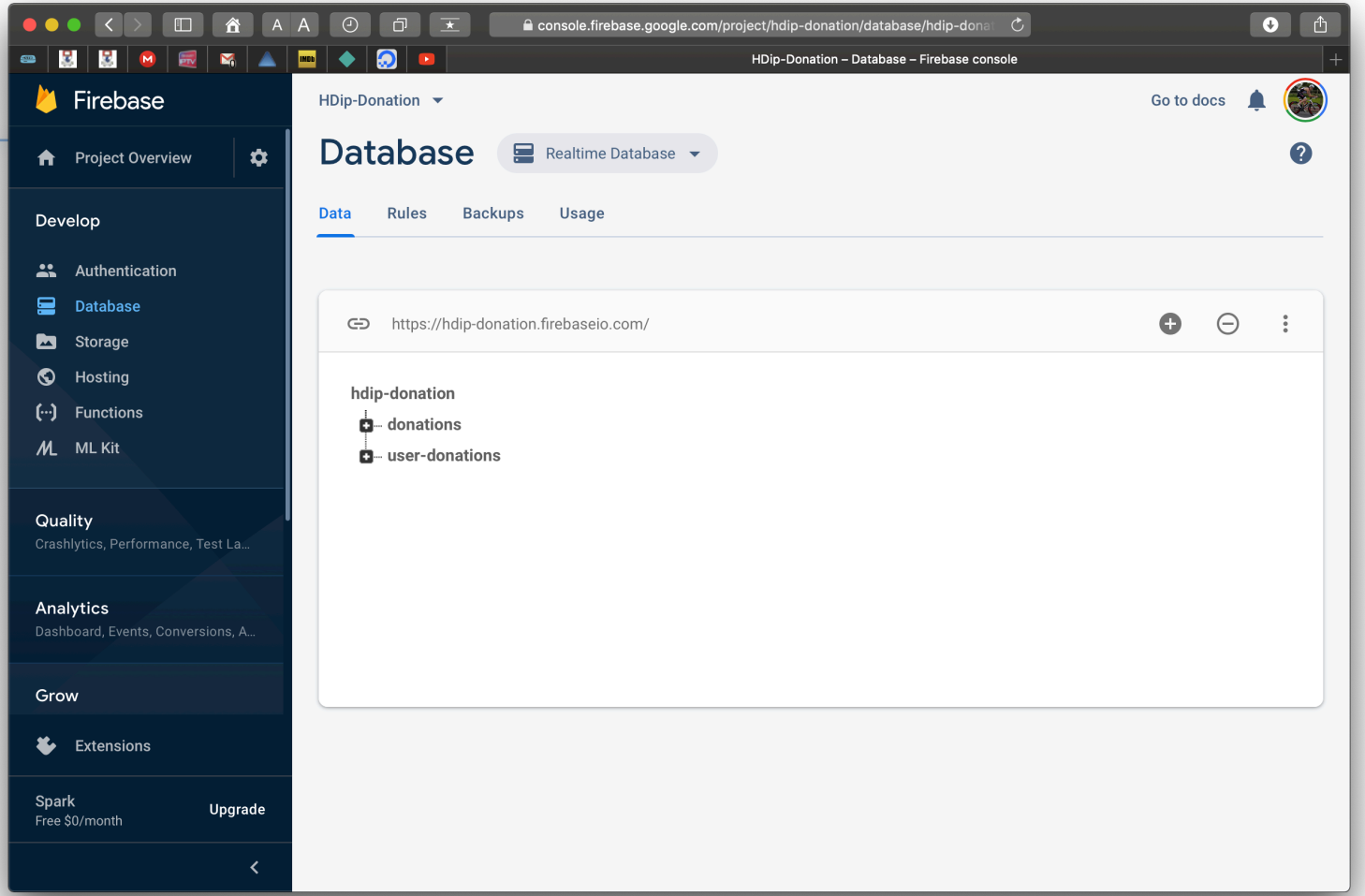
Listening for Child Events

- 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`**:

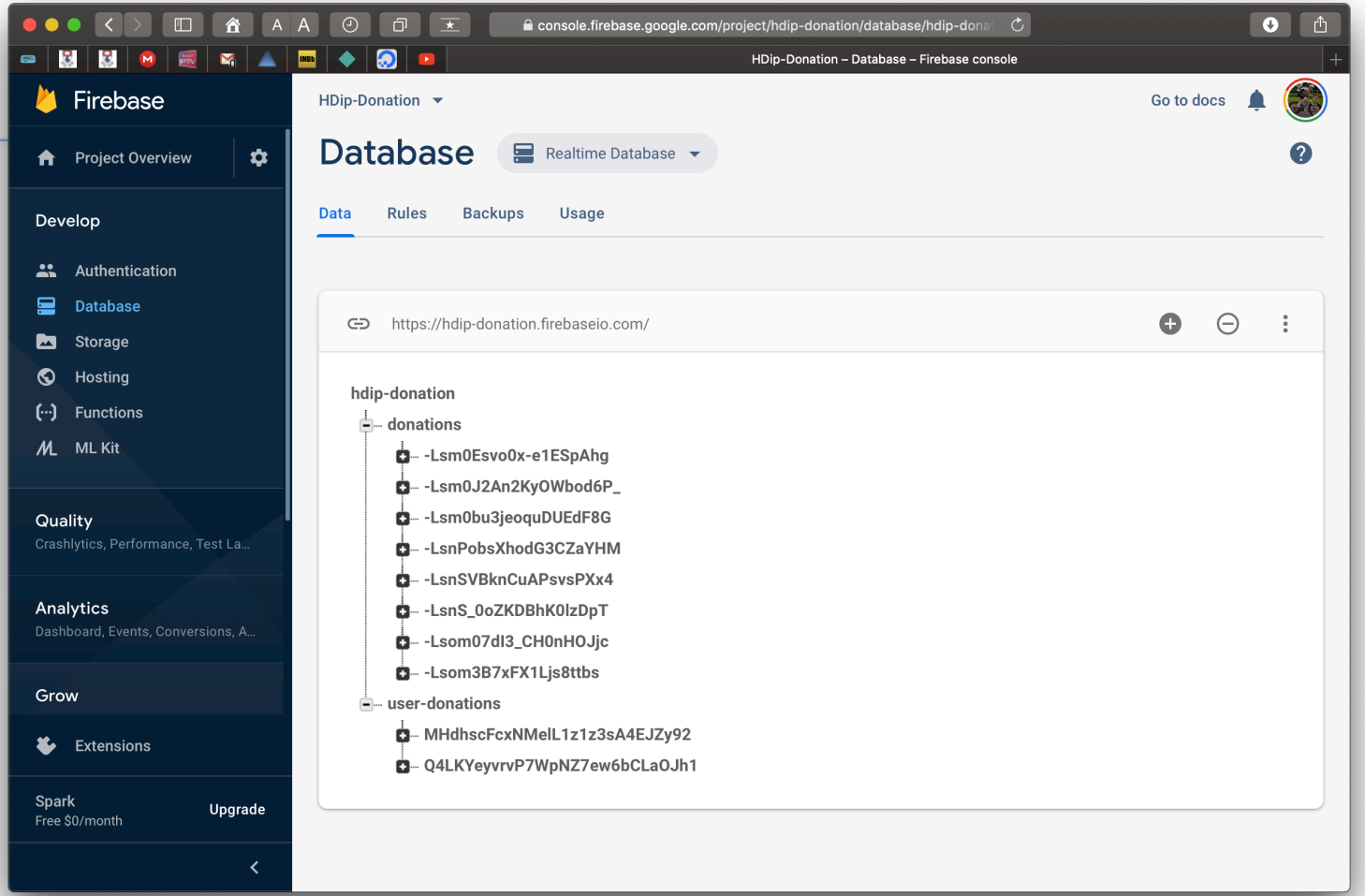
Listening for Child Events

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 `DataSnapshot` 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 `DataSnapshot` passed to the event listener contains the updated data for the child.
	`onChildRemoved()`	Listen for items being removed from a list. The `DataSnapshot` 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 `onChildChanged()` callback is triggered by an update that causes reordering of the child. It is used with data that is ordered with `orderByChild` or `orderByValue`.

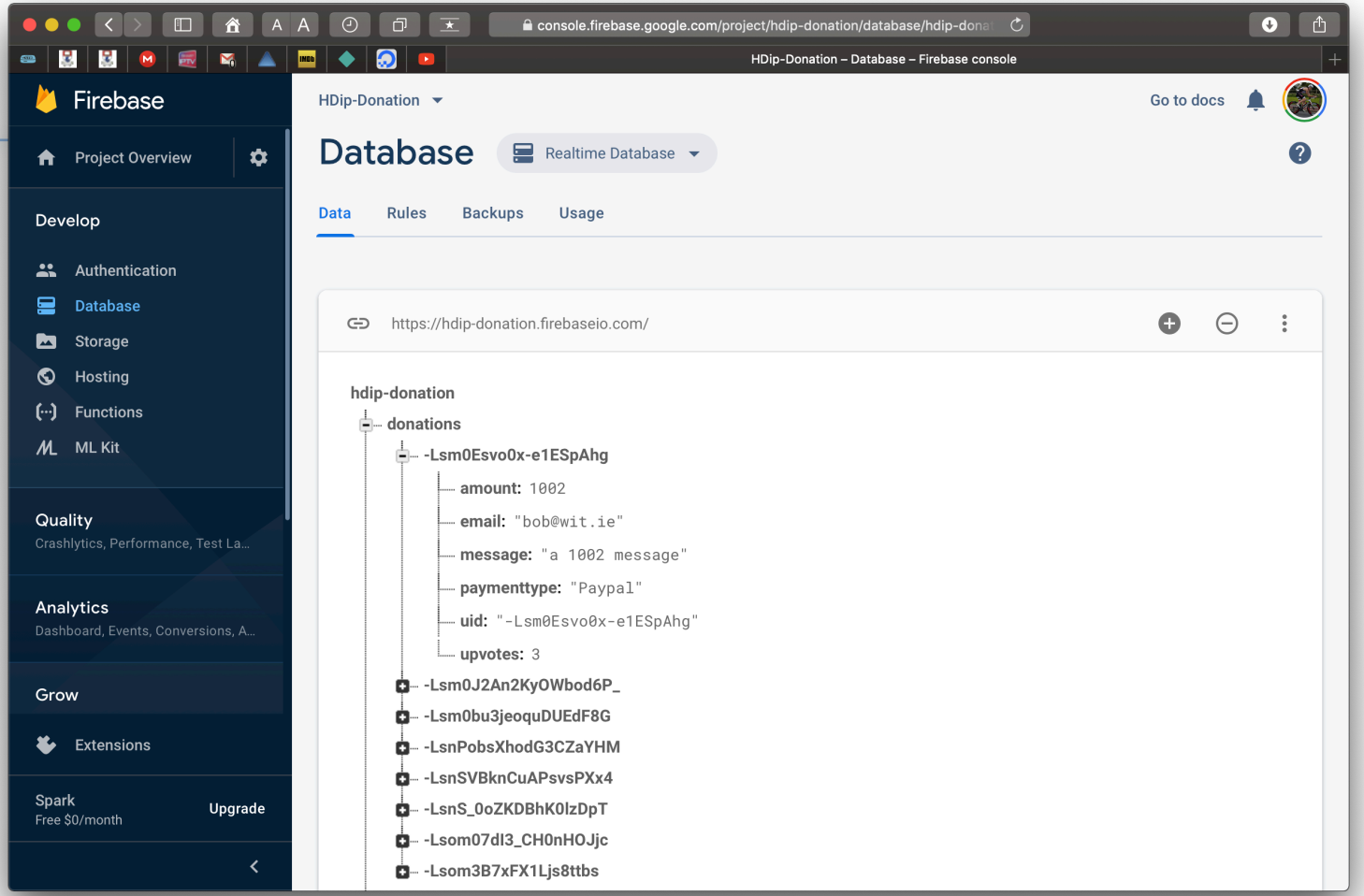
Viewing Your Data



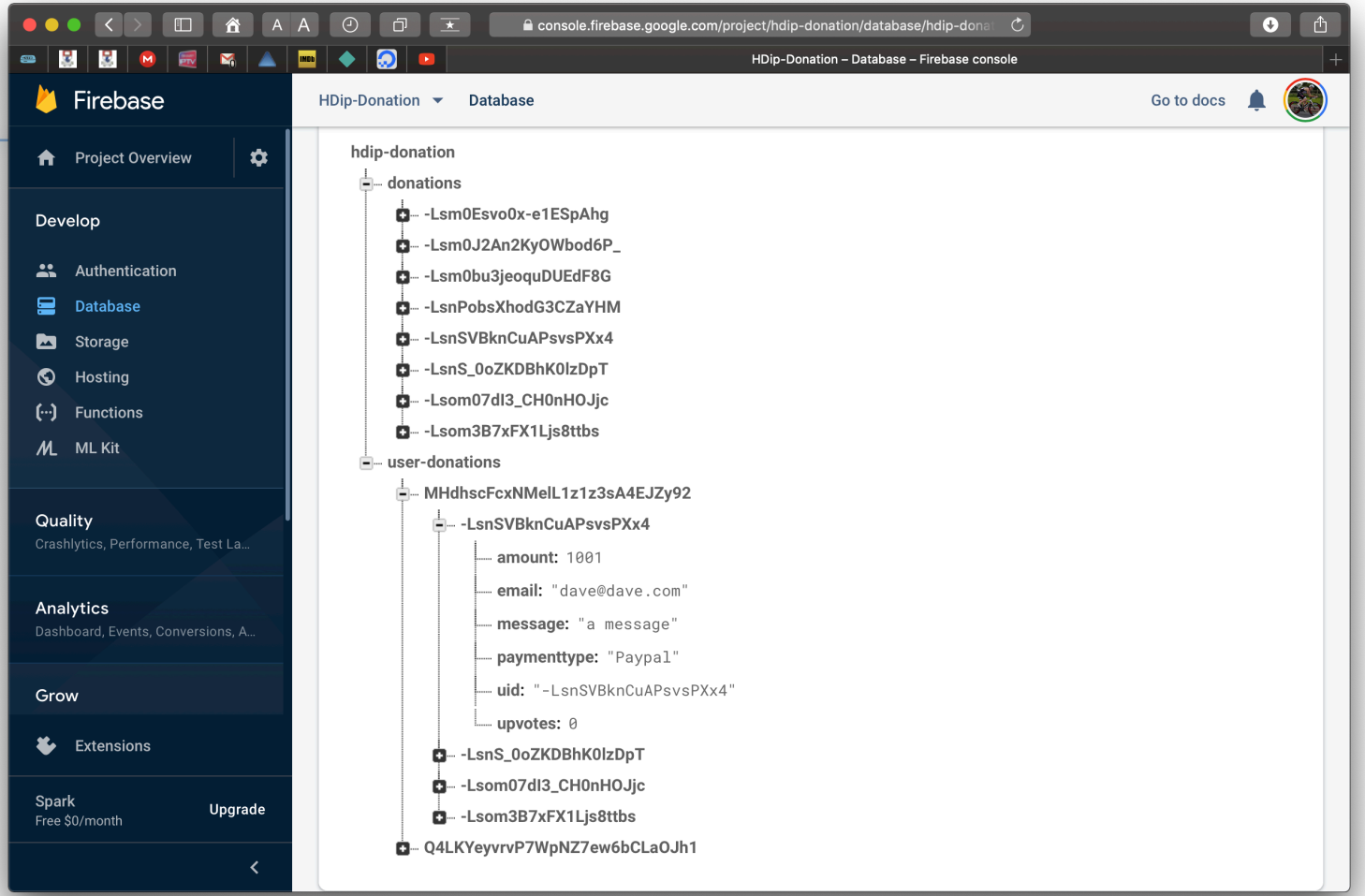
Viewing Your Data



Viewing Your Data



Viewing Your Data



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

References

https://console.firebase.google.com/
https://firebase.google.com/
https://firebase.google.com/docs/database/android/start
https://github.com/firebase/quickstart-android


Thanks.