



EBook Gratuito

APPENDIMENTO firebase

Free unaffiliated eBook created from
Stack Overflow contributors.

#firebase

Sommario

Di.....	1
Capitolo 1: Iniziare con Firebase.....	2
Osservazioni.....	2
Versioni.....	2
Examples.....	2
Aggiungi Firebase al tuo progetto Android.....	2
Aggiungi Firebase alla tua app.....	2
Aggiungi l'SDK.....	3
Configurazione di Firebase per IOS.....	4
Iniziare in Firebase con una semplice app Web Hello World in JavaScript.....	12
Iniziamo.....	12
Capitolo 2: Coda Firebase.....	18
Examples.....	18
Come utilizzare la coda Firebase come back-end per l'applicazione.....	18
Prerequisiti.....	18
Architettura.....	18
Capitolo 3: Come ascolto gli errori durante l'accesso al database?.....	22
introduzione.....	22
Examples.....	22
Rileva errori durante la scrittura di un valore su Android.....	22
Rileva errori durante la lettura dei dati su Android.....	22
Rileva errori durante la scrittura di un valore su iOS.....	23
Rilevamento di errori durante la lettura dei dati in JavaScript.....	23
Rilevare errori durante la scrittura di un valore in JavaScript.....	24
Rileva errori durante la lettura dei dati su iOS.....	24
Capitolo 4: Come ottenere il valore chiave di push dal database Firebase?.....	26
introduzione.....	26
Examples.....	26
Esempio di Android.....	26
Capitolo 5: Come usare FirebaseRecyclerAdapter invece di RecyclerView?.....	27

Examples.....	27
Ecco l'esempio per l'utilizzo del componente FirebaseAuth FirebaseAuthAdapter.....	27
Capitolo 6: Come utilizzare il database Firebase per mantenere un elenco di utenti di aute.....	33
Examples.....	33
Come salvare i dati del profilo utente.....	33
Perché salvare i dati dell'utente nel database.....	33
Gestione dei dati dell'account utente nel database in tempo reale.....	34
Capitolo 7: Conservazione.....	35
Osservazioni.....	35
Examples.....	35
Iniziare su iOS.....	35
Prerequisiti.....	35
Aggiungi Firebase Storage alla tua app.....	35
Configura Firebase Storage.....	36
Capitolo 8: Database in tempo reale di Firbase con Android.....	38
Examples.....	38
Come collegare il database in tempo reale con l'applicazione Android.....	38
Capitolo 9: Firebase Console.....	40
Sintassi.....	40
Parametri.....	40
Osservazioni.....	40
Examples.....	40
Firebase All In One.....	40
Capitolo 10: Firebase Funzionalità offline.....	41
introduzione.....	41
Osservazioni.....	41
Examples.....	42
Abilita la persistenza del disco (solo per Android / iOS).....	42
Mantenere i dati freschi (solo Android / iOS).....	42
Capitolo 11: FirebaseAuth.....	44
Osservazioni.....	44

Examples.....	44
Iniziare con FirebaseUI.....	44
Capitolo 12: FirebaseUI (Android).....	46
Examples.....	46
Aggiungere le dipendenze.....	46
Compilare un ListView.....	46
Capitolo 13: Funzioni cloud per Firebase.....	48
introduzione.....	48
Examples.....	48
Invia email di notifica di benvenuto agli utenti per la sottoscrizione.....	48
Ora vai alla tua console Firebase.....	48
Installa CLI Firebase sul tuo computer.....	48
Imposta le variabili di ambiente di Google Cloud.....	49
Distribuire il progetto e testare.....	49
Capitolo 14: Invia notifica dal server personalizzato.....	51
introduzione.....	51
Examples.....	51
Protocollo HTTP Firebase Cloud Messaging.....	51
Utilizzo di Admin SDK (nodo js).....	52
Capitolo 15: Regole del database.....	54
introduzione.....	54
Osservazioni.....	54
Documentazione ufficiale.....	54
Examples.....	54
Come configurare le regole.....	54
Le regole predefinite.....	54
Come impostare i tuoi file leggibili e scrivibili.....	55
Come disabilitare l'accesso in lettura e scrittura.....	55
Come concedere l'accesso solo agli utenti autenticati.....	55
Come consentire la lettura di un elemento specifico dal gruppo, ma evitare di elencare i m.....	56
Capitolo 16: Segnalazione di crash.....	57

Osservazioni.....	57
Documento ufficiale.....	57
Examples.....	57
Configura crash di installazione in Android.....	57
Segnala l'errore in Android.....	57
Capitolo 17: Strutturazione dei dati.....	59
introduzione.....	59
Examples.....	59
Cosa fare e cosa non fare.....	59
Relazioni a due vie.....	60
Capitolo 18: Utilizzando Firebase con Node.....	63
Examples.....	63
Hello World Firebase Realtime Database in Node.....	63
Firebase-code e worker.....	65
Capitolo 19: Verifica e-mail dopo l'iscrizione.....	68
Sintassi.....	68
Parametri.....	68
Osservazioni.....	68
Examples.....	68
Codice d'azione per la verifica dell'invio del processo di invio - AngularJS.....	68
Titoli di coda.....	70

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.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

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

Capitolo 1: Iniziare con Firebase

Osservazioni

[Firebase](#) è un backend as a service (Baas) molto utile per lo sviluppo di app per dispositivi mobili.

Offre molte funzionalità come **autenticazione e sicurezza**, **database in tempo reale e archiviazione file**, **analisi**, **notifiche push**, **AdMod** e molti [altri](#)

Fornisce l' [SDK](#) per **Android**, **iOS**, **Web**, **NodeJS**, **C ++** e **Java Server**

Versioni

SDK della piattaforma	Versione	Data di rilascio
Firekase 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
SDK Java di Firebase Admin	4.1.2	2017/02/14

Examples

Aggiungi Firebase al tuo progetto Android

Ecco i passaggi necessari per creare un progetto Firebase e connettersi con un'app Android.

Aggiungi Firebase alla tua app

1. Crea un progetto Firebase nella [console Firebase](#) e fai clic su **Crea nuovo progetto**.
2. Fai clic su **Aggiungi Firebase all'app per Android** e segui i passaggi di installazione.
3. Quando richiesto, inserisci il **nome del pacchetto dell'app**.
È importante inserire il nome del pacchetto utilizzato dalla tua app; questo può essere impostato solo quando aggiungi un'app al tuo progetto Firebase.

4. Per aggiungere il certificato di firma di debug SHA1 **richiesto per i link dinamici, gli inviti e il supporto per l'accesso a Google in Auth**, vai al tuo progetto in Android Studio, fai clic sulla scheda `Gradle` sul lato destro della finestra, fai clic sul pulsante `Refresh`, vai per `project (root) -> Tasks -> android -> signingReport`. Questo genererà **MD5** e **SHA1** entrambi nella scheda `Run`. Copia incolla SHA1 nella console di Firebase.
5. Alla fine, scaricherai un file `google-services.json`. Puoi scaricare nuovamente questo file in qualsiasi momento.
6. Se non lo hai già fatto, copia questo nella cartella del modulo del progetto, in genere `app /`.

Il prossimo passo è aggiungere l'SDK per integrare le librerie Firebase nel progetto.

Aggiungi l'SDK

Per integrare le librerie Firebase in uno dei tuoi progetti, devi eseguire alcune attività di base per preparare il tuo progetto Android Studio. Potresti averlo già fatto come parte dell'aggiunta di Firebase alla tua app.

1. Aggiungi le regole al tuo file `build.gradle` livello `build.gradle`, per includere il **plugin di google-services**:

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

Quindi, nel tuo file Gradle del modulo (in genere l' `app/build.gradle`), aggiungi la riga del plug-in in fondo al file per abilitare il plugin 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'
```

Il passaggio finale consiste nell'aggiungere le dipendenze per l'SDK di Firebase utilizzando una o più **librerie disponibili** per le diverse funzionalità di Firebase.

Gradle Dependency Line	Servizio
com.google.firebase: Firebase-core: 9.4.0	analitica
com.google.firebase: Firebase-database: 9.4.0	Database in tempo reale
com.google.firebase: Firebase-storage: 9.4.0	Conservazione
com.google.firebase: Firebase-incidente: 9.4.0	Segnalazione di crash
com.google.firebase: Firebase-auth: 9.4.0	Autenticazione
com.google.firebase: Firebase-messaging: 9.4.0	Cloud Messaging / Notifiche
com.google.firebase: Firebase-config: 9.4.0	Configurazione remota
com.google.firebase: Firebase-invita: 9.4.0	Invita / collegamenti dinamici
com.google.firebase: Firebase-annunci: 9.4.0	AdMob
com.google.android.gms: play-servizi-appindexing: 9.4.0	Indicizzazione delle app

Configurazione di Firebase per IOS

1. In primo luogo, vuoi andare alla dashboard di Firebase e creare un nuovo progetto usando il pulsante "Crea nuovo progetto".

Welcome back to Firebase

Continue building your apps with Firebase using some of the resources below.

 [Documentation](#)  [Sample code](#)  [API reference](#)  [Support](#)

Your projects using Firebase CREA

BrainMatter  brainmatter-4f454.firebaseio.com iOS 1 app	fireAuth  fireauth-415f8.firebaseio.com iOS 1 app
KIKOO	loom

2. Vuoi creare un nuovo progetto aggiungendo il nome della tua app, ad esempio inserisco il mio come 'Cool app name', quindi scegli la tua regione e premi 'Crea progetto'

Welcome back to Firebase

Continue building your apps with Firebase using some of the resources below.

 [Documentation](#) <> [Sample code](#)

Your projects using Firebase

BrainMatter

 brainmatter-4f454.firebaseio.com

iOS 1 app

KIKOO

loom

Create a project

Project name

Country/region 

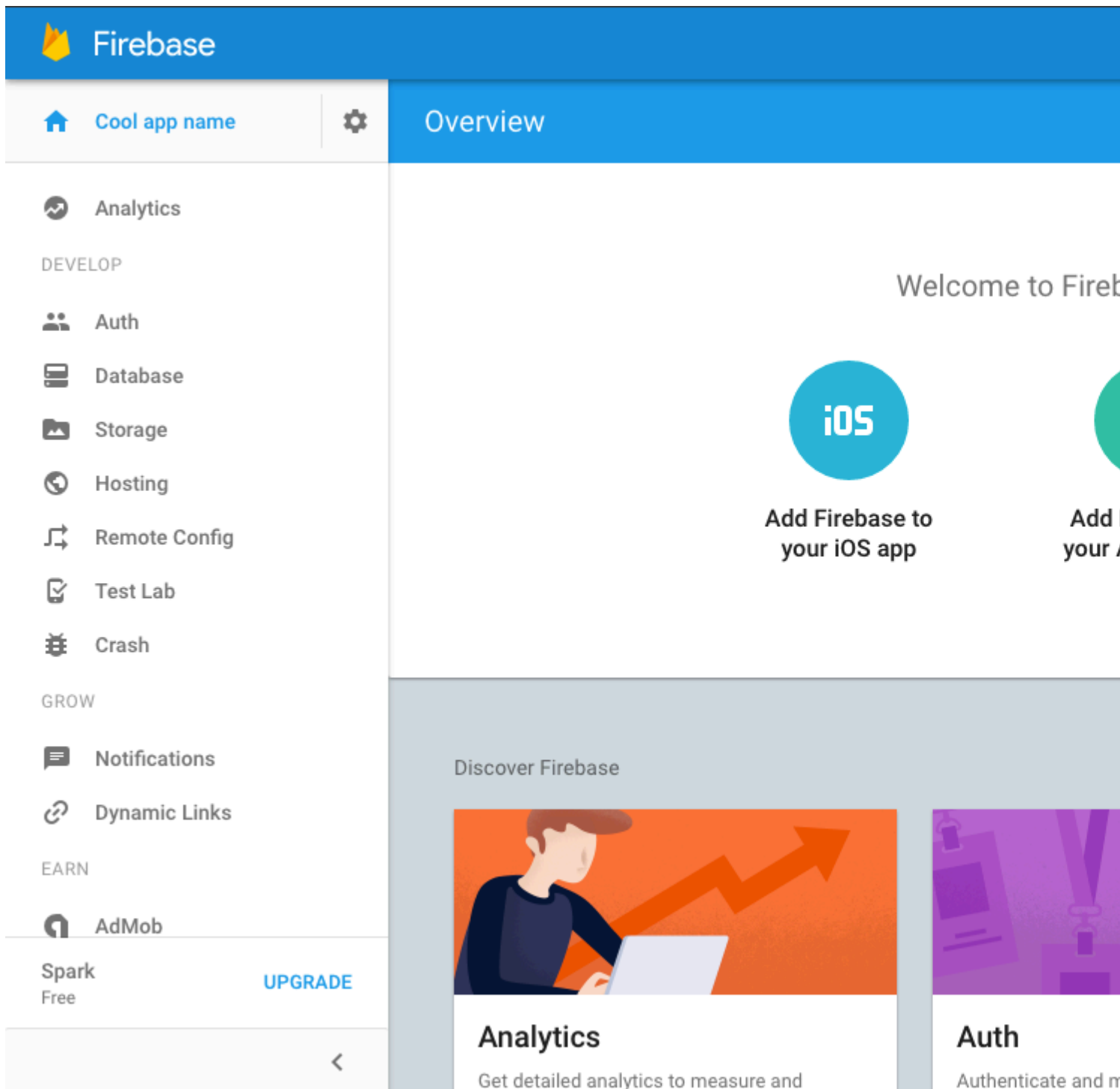
United States 

By default, your Firebase Analytics data will enhance other features and Google products. You can control how your data is shared in your settings at anytime. [Learn more](#)

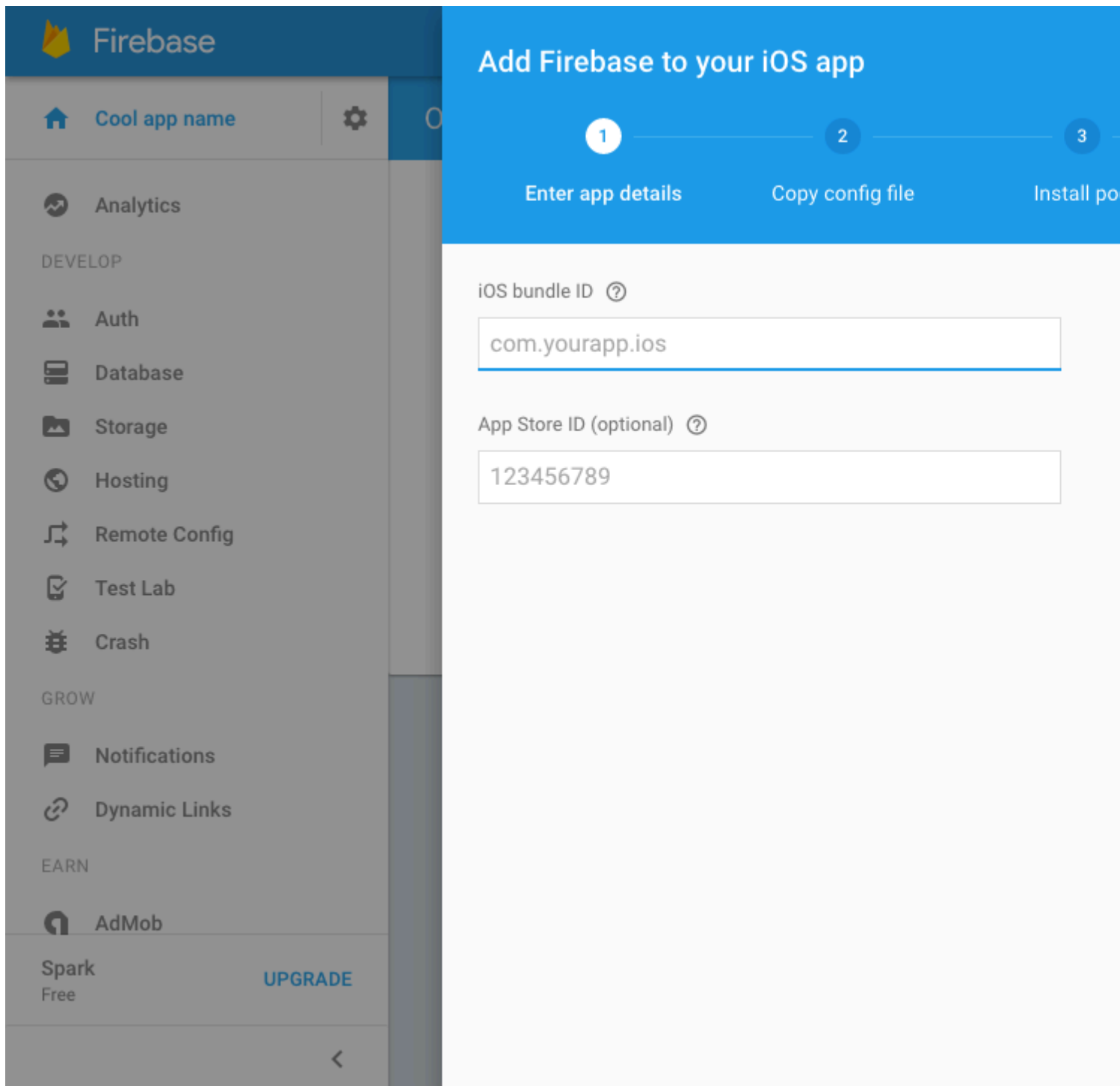
CANCEL

CREATE

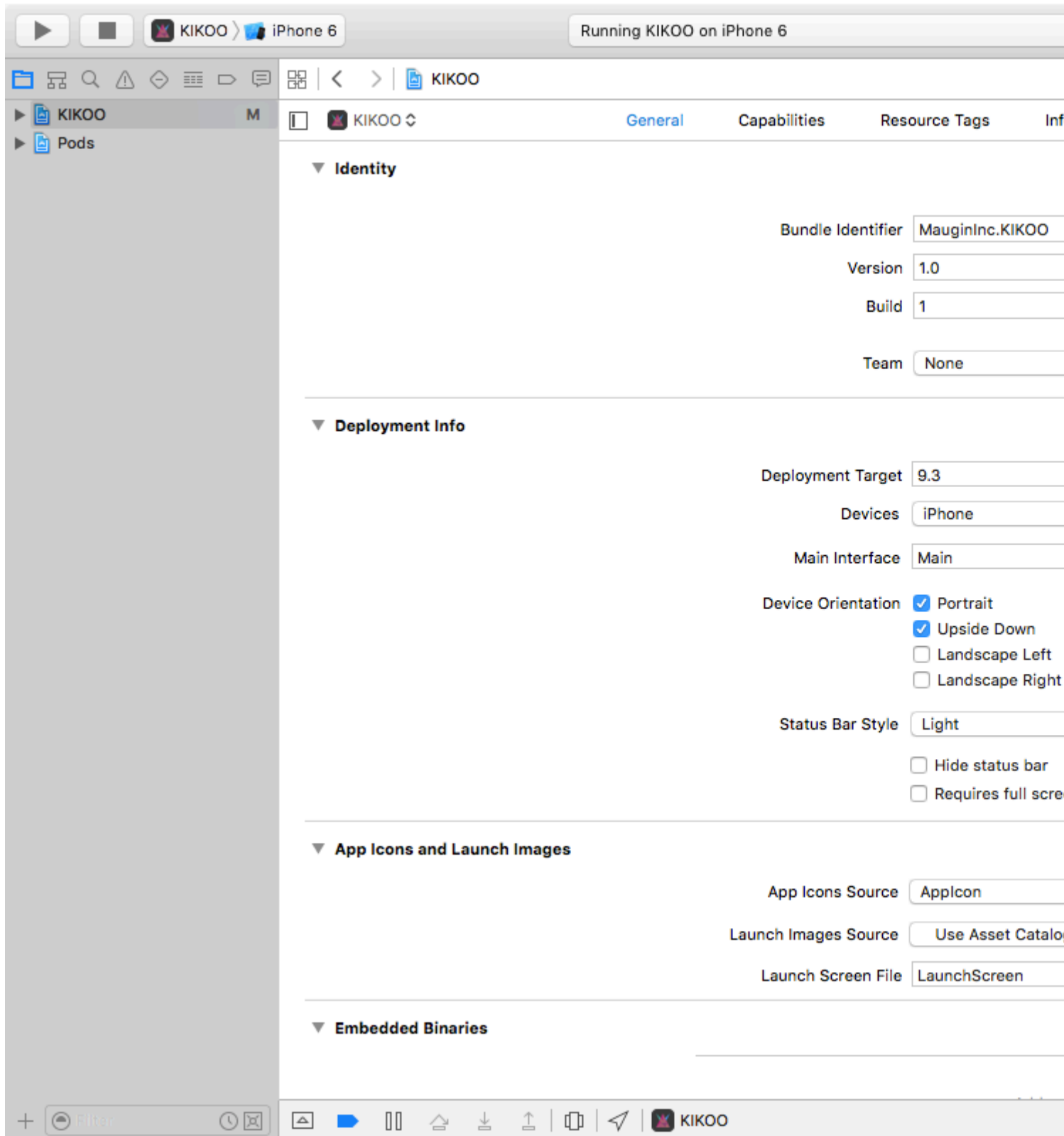
3. Dopo aver creato il progetto verrai indirizzato a questa pagina che è la dashboard e da qui devi scegliere una piattaforma che vuoi installare firebase per questo esempio sceglieremo IOS.



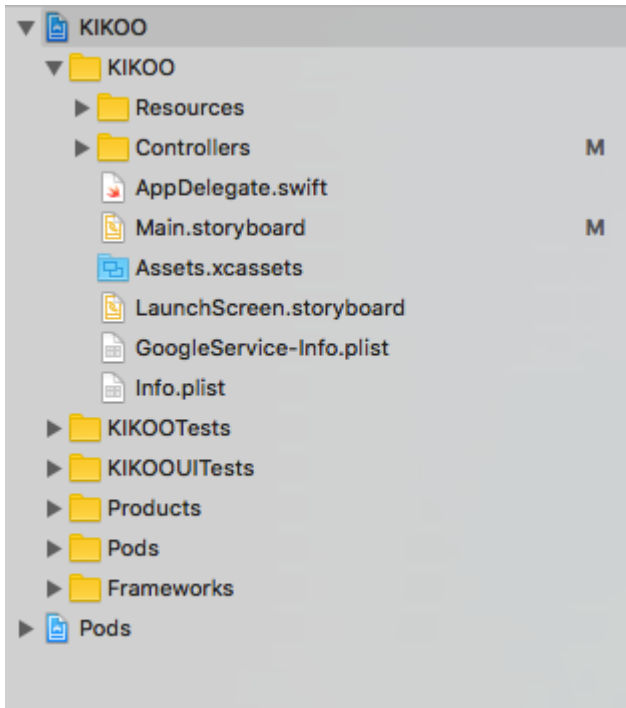
4. Dopo aver selezionato iOS dovresti vedere lo stesso pop-up di quello dall'immagine sottostante che richiede il pacchetto iOS e l'id dell'app store. Dovrai solo fornire il pacchetto iOS perché la nostra app non si trova ancora nell'app store.



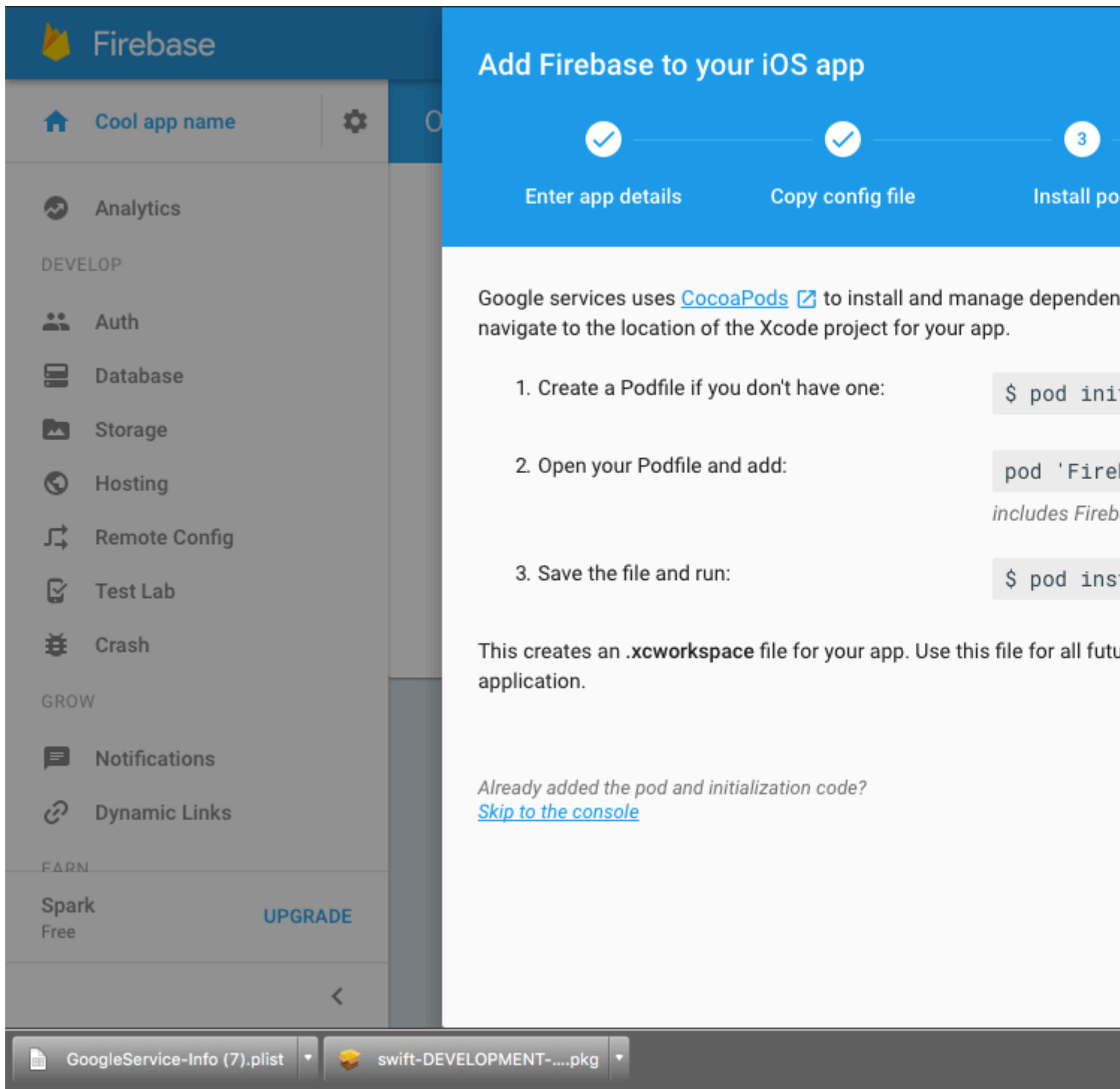
5. Ottieni l'ID del bundle da xcode dopo aver creato un progetto xcode, in ogni caso, in genere, dopo di che puoi ottenere l'ID del bundle per la tua applicazione nell'applicazione General view in xcode sarà il primo campo in alto e una volta che lo avrai incollato in il campo Bundle in Firebase per esempio il mio sarebbe 'MauginInc.KIKOO'



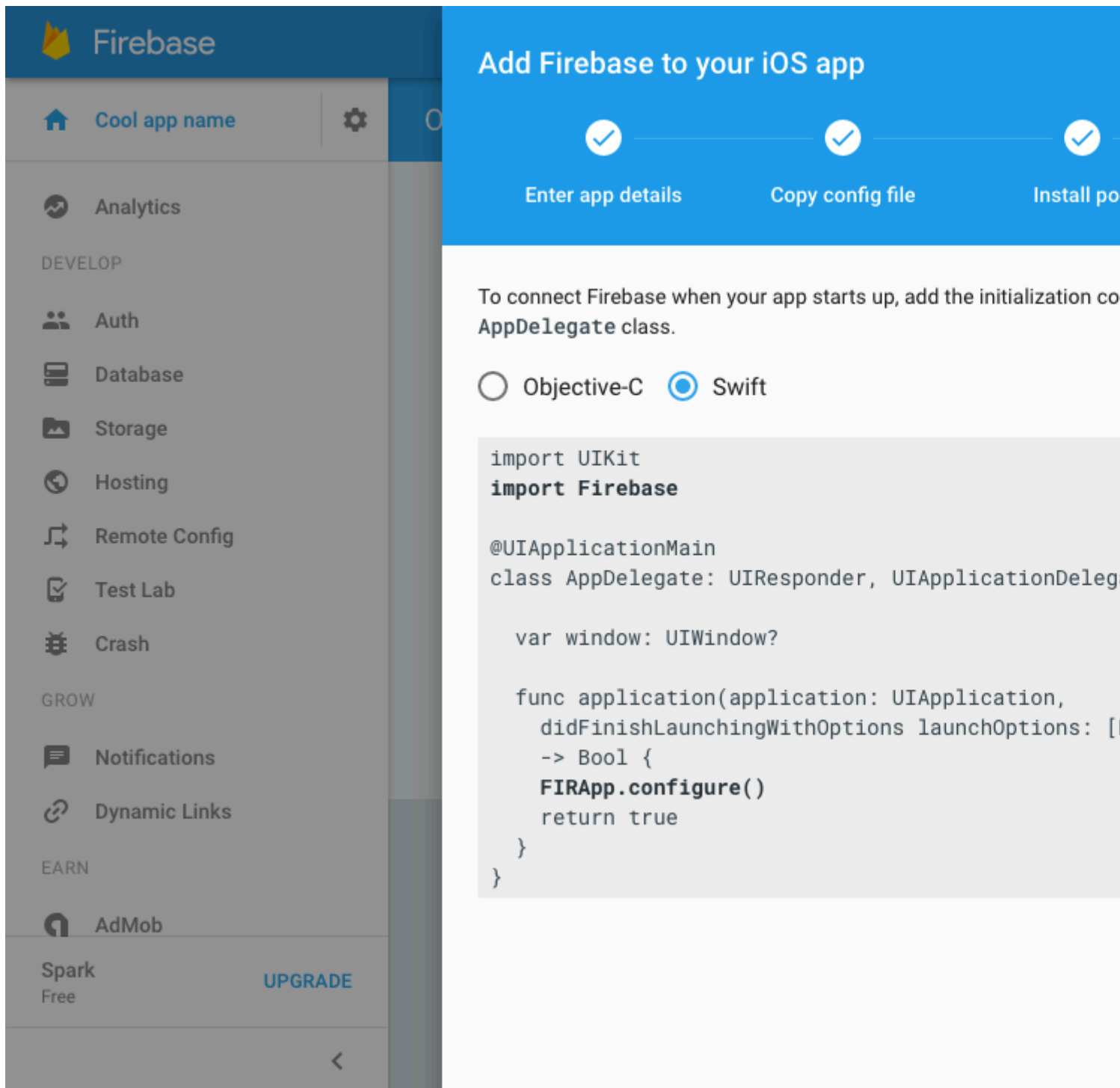
6. Dopo averlo fatto e premuto 'Avanti' verrà scaricato un file 'GoogleService-Info.plist' e sarà necessario spostarlo nella cartella principale della tua app all'interno di xcode



7. Dovrai inizializzare i pod e installare i pod firebase di cui hai bisogno esegui questo accedendo al tuo terminale e vai alla cartella del tuo progetto xcode e segui queste istruzioni fornite da firebase.



8. Infine, vuoi configurare l'app per consentire a Swift di fare ciò che sa fare meglio e questo rende lo sviluppo di app molto più semplice ed efficiente tutto ciò che devi fare è modificare i file AppDelegate.swift come mostrato dal pop-up.



Questo è tutto ciò che hai installato firebase nel tuo progetto xcode per IOS

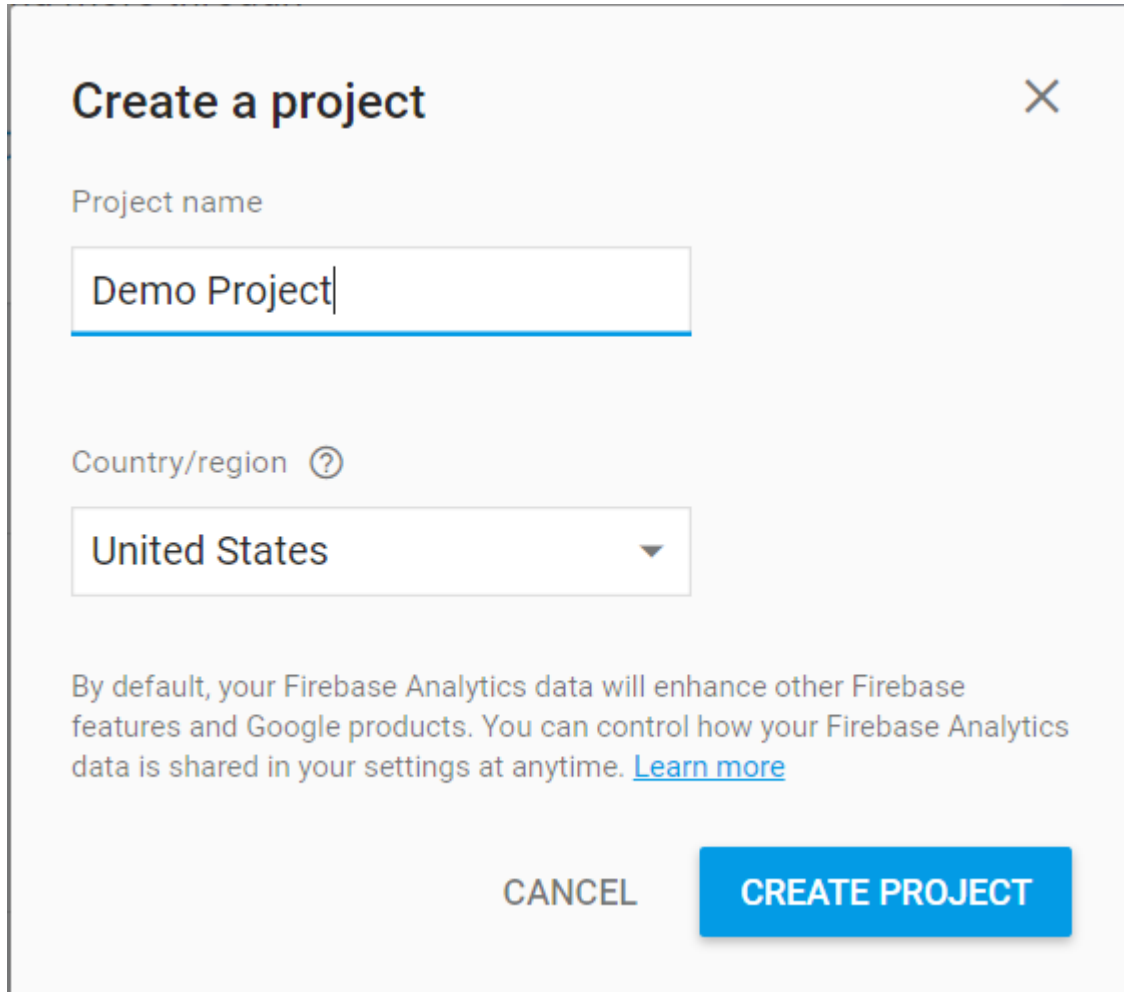
Iniziare in Firebase con una semplice app Web Hello World in JavaScript

Questo esempio mostra come iniziare con Firebase nelle tue app Web con JavaScript.

Aggiungeremo un **testo di testo** nel nostro database Firebase e lo mostreremo in tempo reale sulla nostra app Web.

Iniziamo.

- Vai alla Firebase Console - <https://console.firebase.google.com> e crea un nuovo progetto. Inserire il nome del progetto, Paese / regione e fare clic su **Crea progetto** .



Create a project ×

Project name

Demo Project

Country/region ?

United States ▼

By default, your Firebase Analytics data will enhance other Firebase features and Google products. You can control how your Firebase Analytics data is shared in your settings at anytime. [Learn more](#)

CANCEL CREATE PROJECT

- Ora crea un file **index.html** sul tuo computer. E aggiungere il seguente codice ad esso.

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

- Ora vai al tuo progetto su Firebase Console e puoi vederlo

Welcome to Firebase! Get started here.



Add Firebase to
your iOS app



Add Firebase to
your Android app



Add Firebase
your web app

- Ora fai clic su **Aggiungi Firebase alla tua app Web** . Apparirà il seguente pop-up, fare clic sul pulsante Copia

Add Firebase to your web app

Copy and paste the snippet below at the bottom of your HTML, before other script tags

```
<script src="https://www.gstatic.com/firebasejs/3.7.4/firebase.js"></script>
<script>
  // Initialize Firebase
  var config = {
    apiKey: "AIzaSyBjz9nUjWz0p_uBN_uNbn0u0u0u0u0",
    authDomain: "your-app-id.firebaseio.com",
    databaseURL: "https://your-app-id.firebaseio.com",
    storageBucket: "your-app-id.firebaseio.com",
    messagingSenderId: "your-app-id"
  };
  firebase.initializeApp(config);
</script>
```

Check these resources to learn more about Firebase for web apps:

[Get Started with Firebase for Web Apps](#) 

[Firebase Web SDK API Reference](#) 

[Firebase Web Samples](#) 

- Ora vai al tuo file **index.html** e aggiungi lo snippet alla sezione script come segue

```
<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>
```

- Ora hai completato l'aggiunta del codice di inizializzazione di Firebase. Ora dobbiamo ottenere il nostro valore di **testo** dal database.
- Per fare ciò aggiungi il seguente codice (Inizializza Firebase già aggiunto nell'ultimo passaggio. Non aggiungere nuovamente) all'interno dello script in **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());

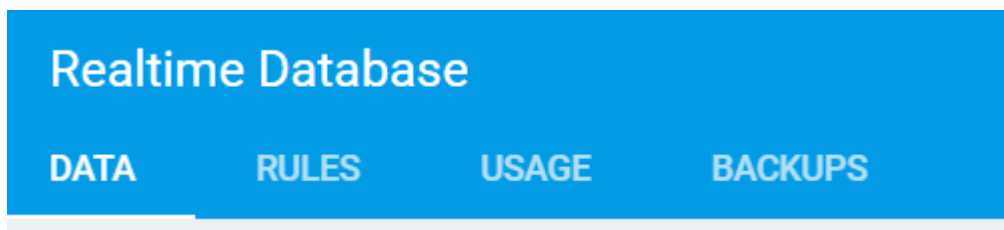
</script>
```

- Ora abbiamo finito con il file **index.html** e ora passiamo al **Database** nella Firebase Console.
- Vedrai che è vuoto e vuoto adesso. Consente di aggiungere un **testo figlio** nel database e aggiungere qualsiasi valore ad esso.



The screenshot shows a data entry form in the Firebase Database console. At the top, there is a redacted path followed by a field containing 'null' and a red 'X' button. Below this is a form with two main sections: 'Name' with a text input containing 'text', and 'Value' with a text input containing 'Hello Firebase'. To the right of the 'Value' input is another red 'X' button. At the bottom of the form are two buttons: 'CANCEL' and 'ADD'.

- Ora fai clic sul pulsante **AGGIUNGI**.
- Ora vai alla sezione **REGOLE** nel Database.



- Per lo scopo di sviluppo in questo momento, in questo momento abiliteremo tutte le query di **lettura e scrittura**.

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

```
1  {
2    "rules": {
3      ".read": "true",
4      ".write": "true"
5    }
6  }
```

- Ora apri index.html nel browser
- Vedrai il valore del testo sulla tua pagina come segue:

Getting started with Firebase

Hello Firebase

- Ora se torni al tuo database e cambi il valore **figlio** del **testo** in qualcos'altro, vedrai che anche il testo nel browser cambia senza alcun aggiornamento o ricarica. Questo è il modo in cui il **database in tempo reale** funziona su Firebase.

Leggi Iniziare con Firebase online: <https://riptutorial.com/it/firebase/topic/816/iniziare-con-firebase>

Capitolo 2: Coda Firebase

Examples

Come utilizzare la coda Firebase come back-end per l'applicazione

Firebase fornisce backend come servizio, come sviluppatore di applicazione non hai la possibilità di avere il codice di back-end.

Questo esempio mostra come utilizzare la coda firebase, creare backend che funzioneranno nella parte superiore del database Firebase e servire da backend per l'applicazione frontend.

Prima di entrare nel codice, capiamo l'architettura, come funzionerà. Per brevità, supponiamo di utilizzare il sito Web come frontend e il server NodeJs come back-end

Prerequisiti

1. Crea un'applicazione Firebase usando il tuo account google
2. Aggiungi Firebase alla tua pagina web. Usa `bower install firebase --save`
3. Crea un account di servizio utilizzando il tuo nuovo account Firebase creato (Impostazioni-> Autorizzazioni -> Account servizio -> CREA ACCOUNT SERVIZIO -> (specifica il nome e spunta questa casella di controllo "Fornisci una nuova chiave privata") -> salva il file json, avremo bisogno quello dopo.
4. Configura il server NodeJs che può essere ospitato nel tuo ambiente preferito
5. Crea il seguente endpoint all'interno della `queue/specs`

"richiedere risposta":

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

6. All'interno del server NodeJs installare la versione lato server firebase, `npm install firebase --save` e `npm install firebase --save` tuo account di servizio usando il file json che abbiamo ottenuto dal passaggio 3, assomiglia a questo

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

Architettura

Ecco l'intero ciclo come funziona.

Dal lato frontale farai questi passi

1. Utilizzando sdk di Firebase Web stai scrivendo le tue richieste direttamente nel database di Firebase nell'endpoint 'coda / attività', chiamiamo la tua richiesta che stai inviando al back-end.
2. dopo aver inserito l'attività si sta registrando il listener sulla `queue/tasks/{taskKey}` endpoint `queue/tasks/{taskKey}` che verrebbe chiamato quando il backend termina l'elaborazione della richiesta, scrivendo la risposta all'interno `queue/tasks/{taskKey}` precedente

Nel backend farai questi passi

1. Crea un server che ascolti infinitamente 'coda / attività'
2. Elabora i tuoi compiti e scrive i dati di risposta all'interno di `queue/tasks/response`
3. Rimuovi l'attività

Prima di tutto crea questa funzione di aiuto, che fornisce un modo per gestire insieme callback e promesse

```
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;
}
```

Nel frontend avrai questa funzione

```
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;
}

```

puoi usare questa funzione come `sendRequest('CreateHouseFacade', {houseName:'Test'})` .

Il parametro gentile è per il backend, per sapere quale metodo chiamare per la richiesta di elaborazione. Params serve per passare informazioni aggiuntive sui parametri.

Ed ecco il codice back-end

```

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 funzione `myService` contiene il codice logico aziendale che `CreateHouseFacade` richiesta `CreateHouseFacade`.

Leggi Coda Firebase online: <https://riptutorial.com/it/firebase/topic/7619/coda-firebase>

Capitolo 3: Come ascolto gli errori durante l'accesso al database?

introduzione

Ci sono molte ragioni per cui un'operazione di lettura o scrittura potrebbe fallire. Uno frequente è perché le regole di sicurezza rifiutano l'operazione, ad esempio perché non sei autenticato (per impostazione predefinita un database è accessibile solo da un utente autenticato) o perché stai scrivendo / ascoltando in una posizione in cui non lo fai avere il permesso.

Examples

Rileva errori durante la scrittura di un valore su Android

Ci sono molte ragioni per cui un'operazione di scrittura può fallire. Uno frequente è perché le regole di sicurezza rifiutano l'operazione, ad esempio perché non sei autenticato (per impostazione predefinita un database può essere visitato solo da un utente autenticato).

È possibile vedere queste violazioni delle regole di sicurezza nell'output del logcat. Ma è facile trascurare questi. Puoi anche gestirli nel tuo codice e renderli più visibili, il che è particolarmente utile durante lo sviluppo (dal momento che JSON, regole e codice cambiano spesso).

Per rilevare una scrittura non riuscita su Android, [allegare una callback di completamento a setValue](#) :

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

Lanciare un'eccezione come questa assicura che sarà molto difficile trascurare un errore simile la prossima volta.

Rileva errori durante la lettura dei dati su Android

Un motivo frequente per cui l'operazione di lettura potrebbe non funzionare è perché le regole di sicurezza rifiutano l'operazione, ad esempio perché non sei autenticato (per impostazione predefinita un database può essere visitato solo da un utente autenticato).

È possibile vedere queste violazioni delle regole di sicurezza nell'output del logcat. Ma è facile trascurare questi. Puoi anche gestirli nel tuo codice e renderli più visibili, il che è particolarmente utile durante lo sviluppo (dal momento che JSON, regole e codice cambiano spesso).

Per rilevare una lettura non riuscita su Android è necessario implementare il metodo `onCancelled` di

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();  
    }  
});
```

Oppure se hai un ValueEventListener :

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

Con questo codice in atto sarà piuttosto difficile trascurare un errore di sicurezza durante la lettura dei dati su Android.

Rileva errori durante la scrittura di un valore su iOS

Ci sono molte ragioni per cui un'operazione di scrittura può fallire. Uno frequente è perché le regole di sicurezza rifiutano l'operazione, ad esempio perché non sei autenticato (per impostazione predefinita un database può essere visitato solo da un utente autenticato).

Puoi vedere queste violazioni delle regole di sicurezza nell'output del tuo programma. Ma è facile trascurare questi. Puoi anche gestirli nel tuo codice e renderli più visibili, il che è particolarmente utile durante lo sviluppo (dal momento che JSON, regole e codice cambiano spesso).

Per rilevare una scrittura non riuscita su iOS, [allegare un blocco di completamento a setValue](#) :

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

Lanciare un'eccezione come questa assicura che sarà molto difficile trascurare un errore simile la prossima volta.

Rilevamento di errori durante la lettura dei dati in JavaScript

Un motivo frequente per cui l'operazione di lettura potrebbe non funzionare è perché le regole di sicurezza rifiutano l'operazione, ad esempio perché non sei autenticato (per impostazione predefinita un database può essere visitato solo da un utente autenticato).

Puoi vedere queste violazioni delle regole di sicurezza nella console JavaScript del tuo browser.

Ma è facile trascurare questi. Puoi anche gestirli nel tuo codice e renderli più visibili, il che è particolarmente utile durante lo sviluppo (dal momento che JSON, regole e codice cambiano spesso).

Per rilevare una lettura non riuscita in JavaScript devi implementare aggiungere una seconda richiamata alla tua clausola `on()` :

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

Con questo codice in atto sarà piuttosto difficile trascurare un errore di sicurezza durante la lettura dei dati in JavaScript.

Rilevare errori durante la scrittura di un valore in JavaScript

Ci sono molte ragioni per cui un'operazione di scrittura può fallire. Uno frequente è perché le regole di sicurezza rifiutano l'operazione, ad esempio perché non sei autenticato (per impostazione predefinita un database può essere visitato solo da un utente autenticato).

È possibile vedere queste violazioni delle regole di sicurezza nell'output della console. Ma è facile trascurare questi. Puoi anche gestirli nel tuo codice e renderli più visibili, il che è particolarmente utile durante lo sviluppo (dal momento che JSON, regole e codice cambiano spesso).

Per rilevare una scrittura non riuscita in JavaScript, allegare una callback di completamento per `set` :

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

Mostrare un avviso come questo assicura che sarà molto difficile trascurare un errore simile la prossima volta.

Rileva errori durante la lettura dei dati su iOS

Un motivo frequente per cui l'operazione di lettura potrebbe non funzionare è perché le regole di sicurezza rifiutano l'operazione, ad esempio perché non sei autenticato (per impostazione predefinita un database può essere visitato solo da un utente autenticato).

È possibile vedere queste violazioni delle regole di sicurezza nell'output della console. Ma è facile trascurare questi. Puoi anche gestirli nel tuo codice e renderli più visibili, il che è particolarmente utile durante lo sviluppo (dal momento che JSON, regole e codice cambiano spesso).

Per rilevare una lettura non riuscita su iOS, devi implementare il blocco `withCancel` del tuo osservatore:

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

Leggi Come ascolto gli errori durante l'accesso al database? online:

<https://riptutorial.com/it/firebase/topic/5548/come-ascolto-gli-errori-durante-l-accesso-al-database->

Capitolo 4: Come ottenere il valore chiave di push dal database Firebase?

introduzione

Nel database Firebase tutto è un nodo, che segue la chiave pattern: value. Firebase Database ci fornisce un modo semplice per generare chiavi univoche. Le chiavi univoche creano nuovi elementi mentre il caricamento dei dati su una chiave memorizzata in precedenza viene aggiornato.

Examples

Esempio di Android

Supponiamo di avere un'applicazione Dogs, quindi il nostro modello sarà una classe Dog.

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

Ecco come inviare un cane al database, un nuovo cane unico e impostare il cane con la chiave.

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

The `reference.child(key).setValue(dog);` è equivalente a `reference.push().setValue(dog);` E aggiungi il vantaggio di ottenere la chiave all'interno dell'oggetto `Dog`.

Leggi [Come ottenere il valore chiave di push dal database Firebase?](https://riptutorial.com/it/firebase/topic/10839/come-ottenere-il-valore-chiave-di-push-dal-database-firebase-) online:

<https://riptutorial.com/it/firebase/topic/10839/come-ottenere-il-valore-chiave-di-push-dal-database-firebase->

Capitolo 5: Come usare FirebaseRecyclerAdapter invece di RecyclerAdapter?

Examples

Ecco l'esempio per l'utilizzo del componente FirebaseUi FirebaseRecyclerAdapter

Ciao amici prima del codice di avvio, abbiamo bisogno di dichiarare la dipendenza per il componente ui firebase di accesso, quindi ecco la dipendenza che puoi mettere nel tuo gradle altrimenti puoi aggiungere dipendenza come jar.

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

Quindi, dopo aver interrogato il database di Firebase per i dati come segue

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

Quindi, dopo aver passato la query all'interno di FirebaseRecyclerAdapter come seguendo la strada

```
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 (Model Class)

```
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;
}
}}

```

FirestoreChatUserViewHolder.java (RecyclerView.ViewHolder)

```

public class FirestoreChatUserViewHolder 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 FirestoreChatUserViewHolder(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 (layout per riga nella vista recycler)

```

<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_middle" />

                <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>
```

Leggi Come usare **FirestoreRecyclerAdapter** invece di **RecyclerViewAdapter**? online:
<https://riptutorial.com/it/firebase/topic/8982/come-usare-firebaserecycleradapter-invece-di-recycleradapter->

Capitolo 6: Come utilizzare il database Firebase per mantenere un elenco di utenti di autenticazione Firebase

Examples

Come salvare i dati del profilo utente

Ogni utente autenticato ha un `uid` Firebase unico per tutti i provider e restituito nel risultato di ogni metodo di autenticazione.

Un buon modo per memorizzare i dati dell'utente è creare un nodo per conservare tutti i dati degli utenti e proteggerli usando le tue regole di sicurezza

- Banca dati

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

- Sicurezza

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

Il `$uid` nelle regole di cui sopra è una cosiddetta "variabile del dollaro", che garantisce che le regole sottostanti siano applicate a tutti i nodi figli degli `users`. Per ulteriori informazioni, consultare la documentazione [sull'utilizzo di \\$ Variabili per acquisire segmenti di percorso](#).

Perché salvare i dati dell'utente nel database

[Firebase Authentication](#) consente agli utenti della tua app di accedere con i social provider o la loro email + password. Ma cosa succede se si desidera memorizzare ulteriori informazioni su un utente, oltre a ciò che Firebase Authentication consente di specificare?

O se vuoi mostrare un elenco degli utenti nella tua app? Firebase Authentication non ha un'API per questo.

La maggior parte degli sviluppatori risolve questo problema memorizzando le informazioni aggiuntive in un database separato. Questo argomento spiega come memorizzare tali informazioni nel [Database in tempo reale di Firebase](#).

Gestione dei dati dell'account utente nel database in tempo reale

Il sistema di autenticazione Firebase è la fonte di un utente `uid`, `displayName`, `photoURL` e forse `email`. Gli account basati su password impostano questi valori *persistenti* nel sistema di autenticazione tramite il metodo `.updateProfile`. Memorizzando questi valori nel database Realtime, rDB, il nodo `users` pone il problema dei dati non aggiornati. I nomi visualizzati, ad esempio, possono cambiare. Per mantenere questi valori sincronizzati, utilizzare [l'archiviazione locale](#) in concerto con `.onAuthStateChanged`.

su ogni `.onAuthStateChanged`

- `getItem('displayName')` e `getItem('photoURL')`
- **confronta con** `user.displayName` e `user.photoURL`
- **se differente**
 - `setItem('displayName')` e `setItem('photoURL')`
 - `db.ref.child('users').update` i valori di `displayName` e / o `photoURL`

`.onAuthStateChanged` Si `.onAuthStateChanged` su ogni caricamento della pagina o ricarica, nonché su ogni modifica dello stato di autenticazione. Si spara potenzialmente spesso, ad esempio app multi pagina. Tuttavia, leggere e scrivere sull'archivio locale è sincrono e molto veloce, quindi non ci sarà alcun impatto notevole sulle prestazioni dell'app.

Leggi Come utilizzare il database Firebase per mantenere un elenco di utenti di autenticazione Firebase online: <https://riptutorial.com/it/firebase/topic/1729/come-utilizzare-il-database-firebase-per-mantenere-un-elenco-di-utenti-di-autenticazione-firebase>

Capitolo 7: Conservazione

Osservazioni

Firebase Storage fornisce caricamenti e download di file sicuri per le tue app Firebase, indipendentemente dalla qualità della rete. Puoi usarlo per memorizzare immagini, audio, video o altri contenuti generati dagli utenti. Firebase Storage è supportato da Google Cloud Storage, un servizio di archiviazione di oggetti potente, semplice ed economico.

Firebase Storage archivia i tuoi file in un bucket di Google Cloud Storage condiviso con l'app Google App Engine predefinita, rendendoli accessibili tramite Firebase e Google Cloud API. Ciò ti consente di caricare e scaricare file da client mobili tramite Firebase e di eseguire l'elaborazione sul lato server come il filtro delle immagini o la transcodifica video utilizzando Google Cloud Platform. Firebase Storage scala automaticamente, il che significa che non è necessario eseguire la migrazione da Firebase Storage a Google Cloud Storage o qualsiasi altro provider.

Questa integrazione rende i file accessibili direttamente dalle librerie client di Google Cloud Storage gcloud, quindi puoi utilizzare Firebase Storage con le tue lingue sul lato server preferite. Per maggiore controllo, puoi anche utilizzare le API XML e JSON di Google Cloud Storage.

Firebase Storage si integra perfettamente con Firebase Authentication per identificare gli utenti e fornisce un linguaggio di sicurezza dichiarativo che consente di impostare i controlli di accesso su singoli file o gruppi di file, in modo da rendere i file pubblici o privati come si desidera.

Consulta i [documenti pubblici per Firebase Storage](#) per le API, gli esempi e le app di esempio più aggiornate.

Examples

Iniziare su iOS

Prerequisiti

1. Crea un nuovo progetto e aggiungi un'app per iOS a quel progetto nella [console di Firebase](#).
2. Scarica e includi `GoogleServices-Info.plist` nella tua applicazione.

Aggiungi Firebase Storage alla tua app

Aggiungi la seguente dipendenza al file `Podfile` del tuo progetto:

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

Esegui `pod install` e apri il file `.xcworkspace` creato.

Segui queste istruzioni per installare Firebase senza CocoaPods

Configura Firebase Storage

È necessario inizializzare Firebase prima che venga creato o utilizzato qualsiasi riferimento all'app Firebase. Se lo hai già fatto per un'altra funzione di Firebase, puoi saltare i seguenti due passaggi.

Importa il modulo Firebase:

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

```
// Swift
import Firebase
```

Configurare `FIRApp` condivisa `FIRApp`, in genere nell'applicazione

`application:didFinishLaunchingWithOptions:` metodo:

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

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

Ottieni un riferimento al servizio di archiviazione, utilizzando l'app Firebase predefinita:

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

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

Crea un riferimento a un file in Firebase Storage:

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

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

Carica un file su 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
    }
}
```

Leggi Conservazione online: <https://riptutorial.com/it/firebase/topic/4281/conservazione>

Capitolo 8: Database in tempo reale di Firbase con Android

Examples

Come collegare il database in tempo reale con l'applicazione Android

Come implementare il database FirebaseRealTime nell'applicazione Android. Di seguito è riportato i passaggi per farlo.

1. Prima installa sCk Firebase, se non sai come installare, segui l'URL di aiuto. [Installa l'SDK di Firebase](#)
2. Dopo aver registrato il progetto nella console di firbase, l'URL della console di firbase è [Url della console di Firebase](#)
3. Dopo aver completato con successo sopra per aggiungere aggiungere la seguente dipendenza nel livello di livello dell'applicazione. compila 'com.google.firebase: firebase-database: 9.2.1'
4. Un'altra cosa ancora configura le regole del tuo database Firebase. Se non si sa come configurare, seguire l'URL che ti aiuta. [Configura le regole di Firebase](#)
5. Ora, dopo aver fatto tutte le cose, inizia il codice originale, in primo luogo recupera l'istanza del database che lancia FirebaseDatabase come segue,

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

Ora puoi creare diversi oggetti di DatabaseReference per il nodo di accesso diverso,

6. Ora puoi salvare o recuperare i dati usando DataBaseReference come segue, Per il salvataggio:

```
myRef.setValue ("Demo per Save");
```

Leggi i dati:

```
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: questo è l'unico argomento di presentazione su come implementare il database nell'applicazione Android perso di più elementi disponibili nel database FirebaseRealtime,

Leggi Database in tempo reale di Firbase con Android online:

<https://riptutorial.com/it/firebase/topic/6482/database-in-tempo-reale-di-firbase-con-android>

Capitolo 9: Firebase Console

Sintassi

1. Esempio di analisi di Firebase.
2. Spiegazione della console Firebase per ciascun componente.

Parametri

Firestore Analytics	Analisi di Firestore e i suoi diversi componenti
Firestore Console	Come funziona? & Come vengono visualizzati i dettagli nella dashboard?

Osservazioni

Questo documento è molto utile per coloro che sono i principianti dell'analisi analitica di Firestore. Sarà molto utile capire come funziona l'analisi di Firestore nello scenario differente.

Examples

Firestore All In One

[Informazioni della console Firestore in dettaglio](#)

[Android: esempio di analisi di Firestore](#)

Passi per Android:

- Scarica il codice dal link
- Controlla FirestoreAnalyticsActivity
- Questo è tutto ciò che capirai come funziona l'analisi di Firestore per il diverso scenario.

Leggi Firestore Console online: <https://riptutorial.com/it/firebase/topic/6660/firebase-console>

Capitolo 10: Firebase Funzionalità offline

introduzione

In questo post troverai i diversi modi per implementare le funzionalità offline quando utilizzi `Firebase`, le informazioni su quando e perché potrebbe essere una buona idea abilitare funzionalità offline ed esempi di esso con la piattaforma Android.

Osservazioni

Cosa dovrei usare? Persistenza del disco o chiamate `KeepSynced`?

Dalla mia esperienza posso dire che dipende sempre da cosa sta funzionando la tua app e da come gestisci le transazioni e il database della tua applicazione. Se ad esempio si dispone di un'applicazione in cui l'utente sta solo scrivendo e leggendo i dati ma non è in grado di eliminarli o modificarli, utilizzare `DiskPersistence` sarebbe la scelta giusta.

Inoltre, `DiskPersistence` memorizzerà i dati nella cache, il che significa che la tua app utilizzerà più spazio nei dispositivi dell'utente, il che forse non è l'idea migliore nel tuo caso.

D'altra parte, se la tua applicazione gestisce molte transazioni complesse e i tuoi dati vengono aggiornati molto spesso, probabilmente dovresti evitare `DiskPersistence` e usare `keepSynced` nei riferimenti che vuoi mantenere aggiornato.

Perché?

`DiskPersistence` memorizza i dati recuperati in locale, che a volte possono causare molta desincronizzazione mostrando i tuoi dati se non li usi insieme al continuo `ListenerValueEvents`. Per esempio:

1. L'utente A scrive un messaggio "Hello World" sulla tua app, che viene ricevuto per l'utente B
2. Utente B scarica il messaggio dell'Utente A nel suo telefono e visualizza il messaggio "Hello World"
3. L'utente A edita il suo messaggio su "Firebase is cool".
4. L'utente B continuerà a visualizzare il messaggio "Hello World" anche se aggiorna i dati perché il ref istantanea è lo stesso quando il filtro Firebase ne fa uso.

Per evitare ciò, l'idea migliore è di mantenere gli ascoltatori in continuo nei riferimenti che si desidera tracciare sempre.

Posso usare entrambi insieme?

Certo che puoi, e nella maggior parte delle app è probabilmente l'idea migliore per evitare di scaricare molti dati e dare all'utente la possibilità di lavorare con la tua app anche se non ha

alcuna connessione.

Se non ti interessa utilizzare lo spazio della cache nel dispositivo utente, ti consiglio di abilitare `diskPersistence` nell'oggetto `FirebaseDatabase` e di aggiungere anche un flag `keepSync` a ogni riferimento che può avere un sacco di volte in un breve spazio o che vuoi mantieni fresco tutto il tempo.

Examples

Abilita la persistenza del disco (solo per Android / iOS)

Per abilitare la persistenza del disco, devi abilitare il flag `persistenceEnabled` nell'oggetto `FirebaseDatabaseInstance` della tua applicazione:

androide

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

iOS

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

Se vuoi disabilitare la persistenza in qualche momento del ciclo di vita della tua app dovresti ricordarti di disabilitarla allo stesso modo:

androide

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

iOS

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

Mantenere i dati freschi (solo Android / iOS)

Firebase sincronizza e archivia una copia locale dei dati per i listener attivi quando viene utilizzata su dispositivi mobili. Inoltre, puoi mantenere sincronizzati luoghi specifici.

Android:

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

iOS:

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

Il client Firebase scarica automaticamente i dati in queste posizioni e li mantiene aggiornati anche se il riferimento non ha ascoltatori attivi. Si disabilita la sincronizzazione con la seguente riga di codice.

Android:

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

iOS:

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

Leggi Firebase Funzionalità offline online: <https://riptutorial.com/it/firebase/topic/10777/firebase-funzionalita-offline>

Capitolo 11: FirebaseUI

Osservazioni

Firebase è una suite di prodotti integrati progettati per aiutarti a sviluppare la tua applicazione, far crescere una base di utenti coinvolti e guadagnare di più. Include strumenti che ti aiutano a creare la tua app, come un database in tempo reale, l'archiviazione di file e l'autenticazione dell'utente, nonché strumenti per aiutarti a far crescere e monetizzare la tua app, come notifiche push, analisi, rapporti sugli arresti anomali e collegamenti dinamici.

Puoi pensare a Firebase come un set di mattoncini Lego che puoi usare per costruire il tuo capolavoro. Proprio come i mattoni, Firebase è relativamente inopportuno, dal momento che ci sono un numero infinito di modi per combinare i pezzi e non abbiamo intenzione di dirti che alcuni modi sono sbagliati :)

FirebaseUI è basato su Firebase e offre agli sviluppatori legature mobili native semplici, personalizzabili e predisposte per la produzione, oltre alle primitive di Firebase per eliminare il codice boilerplate e promuovere le best practice di Google

Nell'analogia Lego, FirebaseUI è un set di kit precostruiti con istruzioni che è possibile rimuovere dallo scaffale e modificare per soddisfare le proprie esigenze. Puoi vedere come abbiamo usato i singoli componenti di Firebase per costruire FirebaseUI perché FirebaseUI è open source. FirebaseUI deve essere supponente - ti stiamo dicendo come pensiamo che i mattoni dovrebbero andare insieme, quindi facciamo delle scelte. Ma poiché FirebaseUI è open source, puoi entrare e modificare ciò che stiamo facendo per soddisfare meglio le tue esigenze individuali.

Se stai costruendo una città di Lego, preferiresti tirare fuori un gruppo di case da una collezione di pre-costruzione e modificarle leggermente in base alle tue esigenze piuttosto che ricominciare da zero e progettare ogni edificio a mano, giusto?

FirebaseUI facciamo esattamente questo, ed è per questo che lo includiamo nelle nostre app ed esempi di esempio. Gli sviluppatori (noi stessi inclusi) sono pigri: desideriamo il migliore riutilizzo del nostro codice e gli esempi più concisi, e FirebaseUI ci consente di fornire esempi di qualità veramente elevata che si traducono in esperienze utente ottimali ad una frazione dei costi di sviluppo.

Examples

Iniziare con FirebaseUI

FirebaseUI offre client [Android](#) , [iOS](#) e [Web](#) . Puoi iniziare con loro in questo modo:

Android:

```
// 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" />
```

Leggi FirebaseUI online: <https://riptutorial.com/it/firebase/topic/6418/firebaseui>

Capitolo 12: FirebaseUI (Android)

Examples

Aggiungere le dipendenze

FirebaseUI è solo una libreria open-source di Google che fornisce facili collegamenti UI per Firebase Auth e Firebase Database.

Per iniziare ad aggiungere FirebaseUI alla tua app, aggiungi queste dipendenze nel file `build.gradle` dell'app:

```
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'
```

Compilare un ListView

Supponendo che tu abbia già configurato un'app in Android Studio, aggiungi una `ListView` a un layout (o salta se ciò è già stato fatto):

```
<?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>
```

Ora creiamo un modello per i dati che andremo a popolare con il nostro `ListView` con:

```
public class Person {  
  
    private String name  
  
    public Person() {  
        // Constructor required for Firebase Database  
    }  
  
    public String getName() {  
        return name;  
    }  
  
}
```

Assicurati che il tuo `ListView` abbia un ID, quindi crea un riferimento ad esso nella tua `Activity` e imposta il suo adattatore su un nuovo `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());  
            }  
        });  
    }  
}
```

Dopo averlo fatto, aggiungi alcuni dati al tuo database e guarda il `ListView` popola da solo.

Leggi **FirestoreUI (Android)** online: <https://riptutorial.com/it/firebase/topic/6610/firebaseui--android->

Capitolo 13: Funzioni cloud per Firebase

introduzione

Firebase ha lanciato la sua versione beta di Cloud Functions for Firebase, simile all'utilizzo delle funzioni cloud su Google Cloud Platform.

Cloud Functions è un ambiente Node.js ospitato, privato e scalabile in cui è possibile eseguire il codice JavaScript. Firebase SDK per Cloud Functions integra la piattaforma Firebase consentendo di scrivere codice che risponde agli eventi e invoca funzionalità esposte da altre funzionalità di Firebase.

Examples

Invia email di notifica di benvenuto agli utenti per la sottoscrizione.

Usa il repository GitHub per ottenere l'intero codice: <https://github.com/firebase/functions-samples/blob/master/quickstarts/email-users>

- Copia o clona il repository nel tuo computer.

Ora vai alla tua console Firebase

- Crea un progetto Firebase usando la console Firebase.
- Abilita il provider **Google** nella sezione **Auth** .
- Incolla lo snippet di **inizializzazione Web** da: **Firebase Console > Panoramica > Aggiungi Firebase** alla tua app Web nel **public / index.html** dove si trova **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.firebaseio.com",
    databaseURL: "https://databaseURL.firebaseio.com",
    storageBucket: "storageBucket.appspot.com",
    messagingSenderId: "messagingID"
  };
  firebase.initializeApp(config);
</script>
```

Installa CLI Firebase sul tuo computer

- Se non hai già installato **NodeJS** , installalo da <https://nodejs.org/en/> (assicurati di avere la versione aggiornata di NodeJS installata sul tuo computer).
- Aprire prompt dei comandi / terminale e installarlo con **npm install -g firebase-tools** e quindi configurarlo con **login firebase**
- Per scegliere il progetto che hai creato ora ==> Configura la CLI localmente usando **firebase --add** e seleziona il tuo progetto nell'elenco.
- Installa le dipendenze localmente eseguendo: **funzioni cd; installazione di npm; cd -**

Imposta le variabili di ambiente di Google Cloud

- Imposta le variabili di ambiente di Google Cloud **gmail.email** e **gmail.password** in modo che corrispondano **all'email** e alla password dell'account Gmail utilizzato per inviare email. Per questo **apri il prompt dei comandi o il terminale** e digita il seguente comando CLI di Firebase:

funzioni di firebase: config: set gmail.email = "myusername@gmail.com" gmail.password = "secretpassword"

Distribuire il progetto e testare

- Per implementare il progetto, aprire il **cmd / terminale** e utilizzare il comando **firebase deploy** per avviare la distribuzione.

```
=== Deploying to '[REDACTED]'...

i  deploying database, functions, hosting
+  database: rules ready to deploy.
i  functions: ensuring necessary APIs are enabled...
i  runtimeconfig: ensuring necessary APIs are enabled...
+  functions: all necessary APIs are enabled
+  runtimeconfig: all necessary APIs are enabled
i  functions: preparing functions directory for uploading...
i  functions: packaged functions (1.85 KB) for uploading
+  functions: functions folder uploaded successfully
i  hosting: preparing public directory for upload...
Uploading: [=====] 46%+  hosting: public folder
+  hosting: 82 files uploaded successfully
i  starting release process (may take several minutes)...
i  functions: updating function sendEmailConfirmation...
+  functions[sendEmailConfirmation]: Successful update operation.
+  functions: all functions deployed successfully!

+  Deploy complete!

Project Console: https://console.firebase.google.com/project/[REDACTED]/overview
Hosting URL: https://[REDACTED].firebaseapp.com ← Use this URL to open
```

- Una volta fatto, usa il comando per aprire il sito nel **firebase** del browser **open hosting: sito** o fallo manualmente **dall'URL** visualizzato.

Leggi Funzioni cloud per Firebase online: <https://riptutorial.com/it/firebase/topic/9580/funzioni-cloud-per-firebase>

Capitolo 14: Invia notifica dal server personalizzato

introduzione

Questo può essere fatto usando 2 metodi con **richiesta HTTP Post** , con **SDK admin Firebase** in esecuzione sul tuo server. Qui discuterò entrambi.

Examples

Protocollo HTTP Firebase Cloud Messaging

Dalla richiesta del server al link sottostante per inviare la notifica con alcuni parametri di richiesta

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

Durante la richiesta di aggiungere intestazioni come segue

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

Il corpo della richiesta varia

```
{
  "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
  }
}
```

I parametri to accettano array di token come

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

o un singolo token come

```
"token"
```

o un nome di argomento che inizia con / **topic** / like

```
"/topic_name/"
```

Per più argomenti utilizzare le condizioni usando || e gli operatori di && come

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

Utilizzo di Admin SDK (nodo js)

Inizialmente inizializza l'SDK di Firebase e l'SDK dell'amministratore

```
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"
});
```

Creare una stringa JSON del payload come nel primo esempio.

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

Quindi chiama diversi metodi di invio per inviare la notifica.

Per argomento

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

Per 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);
    }
  });
});
```

Leggi [Invia notifica dal server personalizzato online](#):

<https://riptutorial.com/it/firebase/topic/10548/invia-notifica-dal-server-personalizzato>

Capitolo 15: Regole del database

introduzione

Con Firebase Realtime Database, le regole del tuo database rappresentano la tua sicurezza sul lato server. Devi essere molto attento e consapevole di chi ha accesso al tuo database. È importante che nessuno possa accedere ai tuoi dati che non dovrebbero.

Per impostazione predefinita, le regole del database in tempo reale di Firebase consentono a qualsiasi utente autenticato di leggere e scrivere tutti i dati, probabilmente non è ciò che si desidera fare.

Dai uno sguardo agli esempi qui sotto per diversi scenari.

Osservazioni

Il database in tempo reale di Firebase fornisce un linguaggio di regole flessibile basato su espressioni con sintassi simile a JavaScript per definire facilmente la modalità di strutturazione dei dati, come deve essere indicizzato e quando è possibile leggere e scrivere i dati. In combinazione con i nostri servizi di autenticazione, è possibile definire chi ha accesso a quali dati e proteggere le informazioni personali degli utenti da accessi non autorizzati.

Per impostazione predefinita, le regole del database richiedono l'autenticazione Firebase e garantiscono permessi di lettura e scrittura completi solo agli utenti autenticati. Le regole predefinite assicurano che il tuo database non sia accessibile da chiunque prima di avere la possibilità di configurare i

Documentazione ufficiale

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

Examples

Come configurare le regole

1. Vai nella console di Firebase.
2. Scegli il tuo progetto
3. Fare clic sulla sezione Database sulla sinistra, quindi selezionare la scheda Regole.

Se desideri testare le tue regole di sicurezza prima di metterle in produzione, puoi simulare operazioni nella console utilizzando il pulsante Simula nella parte superiore destra dell'editor delle regole.

Le regole predefinite

Le regole predefinite richiedono l'autenticazione.

Consentono il pieno accesso in lettura e scrittura agli utenti autenticati della tua app. Sono utili se vuoi che i dati siano aperti a tutti gli utenti della tua app ma non vuoi che si aprano al mondo.

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

Come impostare i tuoi file leggibili e scrivibili

Basta definire:

```
// 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
  }
}
```

Può essere utile durante lo sviluppo, ma fai attenzione perché questo livello di accesso significa che **chiunque può leggere o scrivere nel tuo database** .

Come disabilitare l'accesso in lettura e scrittura

È possibile definire regole private per disabilitare l'accesso in lettura e scrittura al proprio database da parte degli utenti. Con queste regole, **puoi accedere al database solo quando disponi di privilegi amministrativi (che puoi ottenere accedendo al database tramite la console Firebase o effettuando l'accesso [da un server](#))** .

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

Come concedere l'accesso solo agli utenti autenticati

Ecco un esempio di una regola che assegna a ciascun utente autenticato un nodo personale a `/users/$user_id` dove `$ user_id` è l'ID dell'utente ottenuto tramite l' **autenticazione** .

```
// 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"
    }
  }
}

```

Come consentire la lettura di un elemento specifico dal gruppo, ma evitare di elencare i membri del gruppo

È pratica comune creare gruppi di elementi creando nodi con valore semplice con ID elemento come chiave. Ad esempio, possiamo aggiungere un utente al gruppo "amministratori" creando un nodo su `/administrators/$user_id` con un valore `true`. Non vogliamo che nessuno sappia chi sono gli amministratori, per motivi di sicurezza, ma vogliamo comunque verificare se un utente autenticato è un **amministratore**. Con queste regole possiamo fare proprio questo:

```

{
  "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()",
      }
    }
  }
}

```

Leggi Regole del database online: <https://riptutorial.com/it/firebase/topic/3352/regole-del-database>

Capitolo 16: Segnalazione di crash

Osservazioni

Crash Reporting crea report dettagliati degli errori nella tua app.

Gli errori sono raggruppati in cluster di tracce di stack simili e analizzati dalla gravità dell'impatto sugli utenti. Oltre ai report automatici, è possibile registrare eventi personalizzati per aiutare a catturare i passaggi che portano a un arresto anomalo.

Crash Reporting è attualmente in versione beta mentre risolviamo alcuni problemi noti su Android e iOS.

Documento ufficiale

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

Examples

Configura crash di installazione in Android

1. Completa la parte [Installazione e configurazione](#) per collegare la tua app a Firebase. Questo creerà il progetto in Firebase.
2. Aggiungi la dipendenza per Firebase CrashReporting al tuo file `build.gradle` livello di `build.gradle`:

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

Segnala l'errore in Android

Firebase Crash Reporting genera automaticamente report per errori irreversibili (o eccezioni non rilevate).

Puoi creare il tuo rapporto personalizzato usando:

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

È possibile archiviare il registro quando FirebaseCrash ha inizializzato il modulo:

```
07-20 08: 57: 24.442 D / FirebaseCrashApilmpl: API di reporting FirebaseCrash  
inizializzata 07-20 08: 57: 24.442 I / FirebaseCrash: report FirebaseCrash  
inizializzare d com.google.firebase.crash.internal.zzg@3333d325 07-20 08: 57:  
24.442 D / FirebaseApp: classe inizializzata  
com.google.firebase.crash.FirebaseCrash.
```

E poi quando ha inviato l'eccezione:

07-20 08: 57: 47.052 D / FirebaseCrashApiImpl: **lanciabile java.lang.Exception: il mio primo errore Android non fatale** 07-20 08: 58: 18.822 D /
FirebaseCrashSenderServiceImpl: **codice risposta: 200** 07-20 08: 58: 18.822 D /
FirebaseCrashSenderServiceImpl: **rapporto inviato**

Puoi aggiungere registri personalizzati al tuo rapporto con

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

Leggi Segnalazione di crash online: <https://riptutorial.com/it/firebase/topic/4669/segnalazione-di-crash>

Capitolo 17: Strutturazione dei dati

introduzione

Il database Firebase è un database NoSQL che memorizza i suoi dati sotto forma di oggetti JSON gerarchici. Non ci sono tabelle o record di qualsiasi forma come normalmente dovrebbe avere un database SQL, solo nodi che costituiscono una struttura di valori-chiave.

Normalizzazione dei dati

Per avere una struttura di database correttamente progettata, i requisiti dei dati devono essere accuratamente delineati e preveggenti. La struttura in questo caso dovrebbe essere normalizzata; più piatto è l'albero JSON, più veloce è l'accesso ai dati.

Examples

Cosa fare e cosa non fare

La strada sbagliata

Considera la seguente struttura

```
{
  "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.

    "-KH3Cx0KFvSQELIYZev": {
      "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
  }
}
```

Questo è un ottimo esempio di cosa **NON** fare. Strutture multi-nidificate come quella sopra possono essere molto problematiche e causare un enorme calo delle prestazioni.

Il modo in cui Firebase accede a un nodo è il download di tutti i dati dei bambini, quindi l'iterazione su tutti i nodi dello stesso livello (i figli di tutti i genitori). Ora immagina un database con diversi

utenti , ognuno con centinaia (o anche migliaia) di *post* . L'accesso a un *post* in questo caso potrebbe potenzialmente caricare centinaia di megabyte di dati inutilizzati. In un'applicazione più complicata, il nesting potrebbe essere più profondo di solo 4 livelli, il che comporterebbe download e iterazioni più inutili.

Il modo giusto

Appiattare la stessa struttura sarebbe come questo

```
{
  // "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
  }
}
```

Ciò risparmia una quantità enorme di overhead iterando su meno nodi per accedere a un oggetto target. Tutti gli *utenti* che non hanno alcun post non esisterebbero nel ramo dei *post* e quindi iterare sopra quegli utenti nel *modo sbagliato* sopra è completamente inutile.

Relazioni a due vie

Quello che segue è un esempio di un database college semplice e minimale che utilizza relazioni a due vie

```
{
  "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,
      }
    },
    "-SL3ns2OtARSTUMywgWt": { ... },
    // 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
        "-SL3ns2OtARSTUMywgWt": true,      // Some other student
        // More students
      }
    },
    // More courses
  }
}

```

Nota che ogni studente ha un elenco di *corsi* e ogni corso ha un elenco di *studenti* iscritti.

La ridondanza non è sempre un approccio sbagliato. È vero che costa lo spazio di archiviazione e deve gestire l'aggiornamento di più voci durante l'eliminazione o la modifica di un nodo duplicato; tuttavia, in alcuni scenari in cui i dati non vengono aggiornati spesso, la presenza di relazioni a due vie potrebbe facilitare notevolmente il processo di recupero / scrittura.

Nella maggior parte degli scenari in cui sembra necessaria una query simile a SQL, l'inversione dei dati e la creazione di relazioni bidirezionali sono solitamente la soluzione.

Prendi in considerazione un'applicazione che utilizza il database precedente che richiede la capacità di:

1. Elenca i corsi che un certo studente sta prendendo e ...
2. Elenca tutti gli studenti in un certo corso

Se la struttura del database fosse stata unidirezionale, sarebbe stato incredibilmente più lento scansionare o interrogare per uno dei due requisiti precedenti. In alcuni scenari, la ridondanza rende le operazioni frequenti più veloci e molto più efficienti che, a lungo termine, riducono il costo delle duplicazioni.

Leggi **Strutturazione dei dati online**: <https://riptutorial.com/it/firebase/topic/8912/strutturazione-dei->

dati

Capitolo 18: Utilizzando Firebase con Node

Examples

Hello World Firebase Realtime Database in Node

Requisiti di sistema:

- [Nodo JS](#)

Iniziare

1. Prima vai su [Firebase Console](#) e crea un nuovo progetto.
2. Dopo aver creato il progetto, nel progetto fai clic sull'icona delle impostazioni accanto al nome del progetto nella barra laterale sinistra e seleziona Autorizzazioni.
3. Sulla pagina delle autorizzazioni Fare clic su Account di servizio nella barra laterale sinistra, quindi fare clic su Crea account di servizio
4. Nella finestra popup, inserisci il nome dell'account di servizio e scegli Ruolo account e seleziona Fornisci una nuova chiave privata, dopodiché seleziona JSON e fai clic su Crea (lascia deselezionata Abilita deviazione dominio dominio Google App).
5. Quando si fa clic su crea, verrà scaricato un file JSON con le credenziali dell'account, è sufficiente salvare il file ovunque nel sistema.
6. Il passaggio successivo è la creazione di un database nella console Firebase per cui accedere alla console Firebase e fare clic su Database nella barra laterale sinistra. Dopo di ciò basta creare un nuovo oggetto database con nome user_data con qualche valore fittizio.
7. Ora il tuo progetto Firebase Database è ora configurato semplicemente copi il seguente codice nella directory del tuo progetto.

```
//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. Basta cambiare con l'URL del file delle credenziali JSON (per gli utenti iniziali basta copiare il file delle credenziali nella stessa cartella e nel file index.js è sufficiente aggiungere il nome file delle credenziali).
9. Il passaggio successivo è modificare l'indice in index.js con l'effettivo URL del database Firebase, sarai in grado di trovare questo URL nella Console Firebase nella scheda Database, l'URL sarà come *https://.firebaseio.com/* .
10. Il passo finale è fare

```
npm install firebase
```

11. Dopo l'esecuzione del comando precedente, NPM installerà i pacchetti necessari richiesti per Firebase. Finalmente per eseguire e testare il progetto eseguito

```
node index.js
```

Cosa fa realmente il progetto?

Il progetto carica i dati dal database Firebase basato su cloud. Il progetto dimostra anche come scrivere e leggere i dati da un oggetto dati Firebase.

Per visualizzare i tuoi dati aggiornati in tempo reale, vai alla [tua console](#) fai clic sul progetto che hai realizzato e, a sinistra, premi Database. Lì, puoi vedere i tuoi dati aggiornati in tempo reale, insieme ai loro valori.

Firestore-code e worker

Puoi trasferire attività o dati al database in tempo reale di Firebase ed eseguire un worker che ascolta la coda di Firebase per eseguire alcuni processi in background

Imposta firebase

1. Crea un progetto Firebase nella console Firebase, se non ne hai già uno. Se hai già un progetto Google esistente associato alla tua app, fai clic su Importa Google Project. Altrimenti, fai clic su Crea nuovo progetto ..
2. Fai clic sull'icona delle impostazioni e seleziona Autorizzazioni.
3. Seleziona Account di servizio dal menu a sinistra.
4. Fai clic su Crea account di servizio.

Inserisci un nome per il tuo account di servizio.

Puoi opzionalmente personalizzare l'ID da quello generato automaticamente dal nome.

Scegli Progetto> Editor dal menu a discesa Ruolo.

Seleziona Fornisci una nuova chiave privata e lascia il tipo di chiave come JSON.

Lascia vuoto Abilita la delega a tutto il dominio di Google Apps.

Fai clic su Crea

5. Quando si crea l'account di servizio, viene scaricato per te un file JSON contenente le credenziali dell'account del servizio. Avrai bisogno di questo per inizializzare l'SDK nel server.

Server di installazione

Installa firebase-queue usando npm nella tua app nodejs

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

Dopo aver installato Firebase e Firebase-Queue, puoi iniziare creando una nuova coda e passandogli il riferimento Firebase e una funzione di elaborazione.

Ora consente di creare un'attività di coda Firebase dall'app quando viene creato un nuovo utente e impostare worker per l'ascolto dell'attività di Firebase-Queue e inviare un messaggio di posta elettronica agli utenti creati.

* server.js

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

Aggiorna le tue credenziali del progetto Firebase e l'URL del database Firebase

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

oppure puoi inserire le credenziali di Firebase direttamente come di seguito

```
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 = function(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);
});
```

eseguire server e worker separatamente e testare con la coda di Firebase

```
node server.js

node worker.js
```

Leggi Utilizzando Firebase con Node online:

<https://riptutorial.com/it/firebase/topic/6443/utilizzando-firebase-con-node>

Capitolo 19: Verifica e-mail dopo l'iscrizione

Sintassi

- Invia e-mail di verifica per l'indirizzo e-mail dell'utente registrato su file. Firebase ti consente di [personalizzare ciò che la tua email comporta](#)
- Quando l'email raggiunge l'account e-mail dell'utente, l'utente fa clic
- Utilizzando il router scelto (utilizzato angular-ui-router nell'esempio sopra), intercetta i parametri nell'URL.
- Masticare i parametri usando la funzione `applyCode` in Firebase.
- Vedi sotto per le funzioni coinvolte nel processo sopra.

Parametri

La funzione...	fa
<code>sendEmailVerification ()</code>	Invia una email di verifica a un utente.
<code>applyActionCode ()</code>	Applica il codice azione che cambia <code>emailVerified</code> da <code>false</code> a <code>true</code>

Osservazioni

Quanto sopra riassume piuttosto come utilizzare lo schema di verifica e-mail con Firebase. Finora, si presenta come uno dei modi più semplici per verificare l'e-mail che ho visto.

C'è un po' di una spiegazione estesa dell'esempio sopra disponibile su [Email Verification con Firebase 3.0 SDK](#).

Examples

Codice d'azione per la verifica dell'invio del processo di invio - 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:

....

$scope.$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 });
      })
  };
}
])

```

Leggi Verifica e-mail dopo l'iscrizione online: <https://riptutorial.com/it/firebase/topic/3380/verifica-e-mail-dopo-l-iscrizione>

Titoli di coda

S. No	Capitoli	Contributors
1	Iniziare con Firebase	Ami Hollander , Community , Dan Levy , Devid Farinelli , ErstwhileIII , Gabriele Mariotti , RyanM , Sneh Pandya , TwiterZX , Vishal Vishwakarma
2	Coda Firebase	Vladimir Gabrielyan
3	Come ascolto gli errori durante l'accesso al database?	Frank van Puffelen , ThunderStruct
4	Come ottenere il valore chiave di push dal database Firebase?	cutiko
5	Come usare FirebaseRecyclerAdapter invece di RecyclerView?	Dhaval Solanki
6	Come utilizzare il database Firebase per mantenere un elenco di utenti di autenticazione Firebase	Devid Farinelli , eikooc , Frank van Puffelen , Ron Royston
7	Conservazione	Mike McDonald
8	Database in tempo reale di Firbase con Android	Dhaval Solanki , Frank van Puffelen
9	Firebase Console	Priyank Bhojak
10	Firebase Funzionalità offline	Francisco Durdin Garcia
11	FirebaseUI	Mike McDonald
12	FirebaseUI (Android)	Willie Chalmers III
13	Funzioni cloud per Firebase	Vishal Vishwakarma
14	Invia notifica dal server	Aawaz Gyawali

personalizzato		
15	Regole del database	Frank van Puffelen , Gabriele Mariotti , riggaroo , Sasxa , Velko Ivanov
16	Segnalazione di crash	Gabriele Mariotti
17	Strutturazione dei dati	ThunderStruct
18	Utilizzando Firebase con Node	Akshay Khale , Laurel , Noushad PP , Shiven
19	Verifica e-mail dopo l'iscrizione	Rexford