

 eBook Gratuit

APPRENEZ firebase

eBook gratuit non affilié créé à partir des
contributeurs de Stack Overflow.

#firebase

Table des matières

À propos.....	1
Chapitre 1: Démarrer avec Firebase.....	2
Remarques.....	2
Versions.....	2
Exemples.....	2
Ajouter Firebase à votre projet Android.....	2
Ajouter Firebase à votre application.....	2
Ajouter le SDK.....	3
Configuration de Firebase pour IOS.....	4
Mise en route de Firebase avec une simple application Web Hello World en JavaScript.....	12
Commençons.....	12
Chapitre 2: Capacités hors ligne de Firebase.....	18
Introduction.....	18
Remarques.....	18
Exemples.....	19
Activer la persistance du disque (Android / iOS uniquement).....	19
Garder les données à jour (Android / iOS uniquement).....	19
Chapitre 3: Comment écouter les erreurs lors de l'accès à la base de données?.....	21
Introduction.....	21
Exemples.....	21
Détecer les erreurs lors de l'écriture d'une valeur sur Android.....	21
Détecer les erreurs lors de la lecture de données sur Android.....	21
Détecer les erreurs lors de l'écriture d'une valeur sur iOS.....	22
Détection des erreurs lors de la lecture de données en JavaScript.....	22
Détection des erreurs lors de l'écriture d'une valeur en JavaScript.....	23
Détecer les erreurs lors de la lecture de données sur iOS.....	23
Chapitre 4: Comment obtenir la valeur de la clé push de la base de données Firebase?.....	25
Introduction.....	25
Exemples.....	25
Exemple Android.....	25

Chapitre 5: Comment utiliser FirebaseRecyclerAdapter au lieu de RecyclerViewAdapter?	26
Exemples	26
Voici l'exemple d'utilisation du composant FirebaseUi FirebaseRecyclerAdapter	26
Chapitre 6: Comment utiliser la base de données Firebase pour conserver une liste d'utilis	32
Exemples	32
Comment enregistrer les données de profil utilisateur	32
Pourquoi enregistrer les données utilisateur dans la base de données	32
Gestion des données de compte d'utilisateur dans la base de données en temps réel	33
Chapitre 7: Console Firebase	34
Syntaxe	34
Paramètres	34
Remarques	34
Exemples	34
Firebase All In One	34
Chapitre 8: Espace de rangement	35
Remarques	35
Exemples	35
Démarrer sur iOS	35
Conditions préalables	35
Ajouter Firebase Storage à votre application	35
Configurer le stockage Firebase	36
Chapitre 9: File d'attente Firebase	38
Exemples	38
Comment utiliser la file d'attente Firebase en tant que backend pour votre application	38
Conditions préalables	38
Architecture	39
Chapitre 10: Firbase Realtime Database avec Android	42
Exemples	42
Comment connecter une base de données en temps réel avec une application Android	42
Chapitre 11: FirebaseUI	44
Remarques	44

Exemples.....	44
Premiers pas avec FirebaseUI.....	44
Chapitre 12: FirebaseUI (Android).....	46
Exemples.....	46
Ajouter les dépendances.....	46
Remplir un ListView.....	46
Chapitre 13: Fonctions Cloud pour Firebase.....	48
Introduction.....	48
Exemples.....	48
Envoyer des emails de notification de bienvenue aux utilisateurs pour s'abonner.....	48
Allez maintenant sur votre console Firebase.....	48
Installez Firewall CLI sur votre ordinateur.....	48
Définir des variables d'environnement Google Cloud.....	49
Déployer le projet et tester.....	49
Chapitre 14: Notification push du serveur personnalisé.....	51
Introduction.....	51
Exemples.....	51
Protocole HTTP Firebase Cloud Messaging.....	51
Utilisation du SDK Admin (Nœud js).....	52
Chapitre 15: Rapport de collision.....	54
Remarques.....	54
Documentation officielle.....	54
Exemples.....	54
Configurer Crash Reporting dans Android.....	54
Signaler l'erreur dans Android.....	54
Chapitre 16: Règles de base de données.....	56
Introduction.....	56
Remarques.....	56
Documentation officielle.....	56
Exemples.....	56
Comment configurer les règles.....	56

Les règles par défaut.....	56
Comment définir vos fichiers lisibles et accessibles en écriture.....	57
Comment désactiver l'accès en lecture et en écriture.....	57
Comment accorder l'accès uniquement aux utilisateurs authentifiés.....	57
Comment autoriser la lecture d'un élément spécifique d'un groupe, mais empêcher la liste d.....	58
Chapitre 17: Structuration des données.....	59
Introduction.....	59
Exemples.....	59
À faire et à ne pas faire.....	59
Relations bidirectionnelles.....	60
Chapitre 18: Utiliser Firebase avec Node.....	63
Exemples.....	63
Hello World Firebase Base de données en temps réel dans le nœud.....	63
Firebase-queue et worker.....	65
Chapitre 19: Vérification de l'e-mail après inscription.....	68
Syntaxe.....	68
Paramètres.....	68
Remarques.....	68
Exemples.....	68
Code d'action de vérification d'envoi et de traitement - AngularJS.....	68
Crédits.....	70

À propos

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

Chapitre 1: Démarrer avec Firebase

Remarques

[Firebase](#) est un backend en tant que service (Baas) très utile pour le développement d'applications mobiles.

Il offre de nombreuses fonctionnalités telles que l' **authentification et la sécurité** , la **base de données en temps réel et le stockage de fichiers** , les **analyses** , les **notifications push** , **AdMod** et bien d' [autres](#).

Il fournit le [SDK](#) pour **Android, iOS, Web, NodeJS, C ++ et Java Server**

Versions

Platform SDK	Version	Date de sortie
Firebase JavaScript SDK	3.7.0	2017-03-01
Kit de développement logiciel Firebase C ++	3.0.0	2107-02-27
SDK Firebase Unity	3.0.0	2107-02-27
Firebase iOS SDK	3.14.0	2017-02-23
Firebase Android SDK	10.2	2017-02-15
SDK Admin Node.js de Firebase	4.1.1	2017-02-14
Firebase Admin Java SDK	4.1.2	2017-02-14

Exemples

Ajouter Firebase à votre projet Android

Voici les étapes requises pour créer un projet Firebase et pour vous connecter à une application Android.

Ajouter Firebase à votre application

1. Créez un projet Firebase dans la [console Firebase](#) et cliquez sur **Créer un nouveau projet** .
2. Cliquez sur **Ajouter Firebase à votre application Android** et suivez les étapes de configuration.

3. Lorsque vous y êtes invité, entrez **le nom du package** de votre **application** .
Il est important de saisir le nom du package utilisé par votre application. Cela ne peut être défini que lorsque vous ajoutez une application à votre projet Firebase.
4. Pour ajouter le certificat de signature de débogage SHA1 **requis pour les liens dynamiques, les invitations et la prise en charge de Google Sign-In dans Auth**, accédez à votre projet dans Android Studio, cliquez sur l'onglet `Gradle` à droite de la fenêtre, cliquez sur le bouton `Refresh` . pour `project (root) -> Tasks -> android -> signingReport` . Cela va générer **MD5** et **SHA1** dans l'onglet `Run` . Copiez collez SHA1 dans la console firebase.
5. À la fin, vous allez télécharger un fichier `google-services.json` . Vous pouvez télécharger ce fichier à tout moment.
6. Si vous ne l'avez pas déjà fait, copiez-le dans le dossier du module de votre projet, généralement `app /`.

L'étape suivante consiste à ajouter le SDK pour intégrer les bibliothèques Firebase dans le projet.

Ajouter le SDK

Pour intégrer les bibliothèques Firebase dans un de vos propres projets, vous devez effectuer quelques tâches de base pour préparer votre projet Android Studio. Vous l'avez peut-être déjà fait dans le cadre de l'ajout de Firebase à votre application.

1. Ajoutez des règles à votre fichier `build.gradle` niveau `build.gradle` pour inclure le **plug-in google-services** :

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

Ensuite, dans le fichier Gradle de votre module (généralement `app/build.gradle`), ajoutez la ligne de plug-in apply au bas du fichier pour activer le plug-in 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'
```

La dernière étape consiste à ajouter les dépendances du kit SDK Firebase en utilisant une ou plusieurs **bibliothèques disponibles** pour les différentes fonctionnalités Firebase.

Ligne de dépendance Gradle	Un service
com.google.firebase: firebase-core: 9.4.0	Analytique
com.google.firebase: base de données firebase: 9.4.0	Base de données en temps réel
com.google.firebase: firebase-storage: 9.4.0	Espace de rangement
com.google.firebase: crash de base de données: 9.4.0	Rapport de collision
com.google.firebase: firebase-auth: 9.4.0	Authentification
com.google.firebase: messagerie-base: 9.4.0	Cloud Messaging / Notifications
com.google.firebase: firebase-config: 9.4.0	Configuration à distance
com.google.firebase: invite-firebase: 9.4.0	Invites / Liens dynamiques
com.google.firebase: annonces firebase: 9.4.0	AdMob
com.google.android.gms: play-services-appindexing: 9.4.0	Indexation des applications

Configuration de Firebase pour IOS

1. Tout d'abord, vous souhaitez accéder au tableau de bord de Firebase et créer un nouveau projet à l'aide du bouton "Créer un nouveau projet".

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 CRE

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

- Vous souhaitez créer un nouveau projet en ajoutant le nom de votre application, par exemple, je mets le mien en tant que «Cool app name», puis choisissez votre région et appuyez sur «Create Project».

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

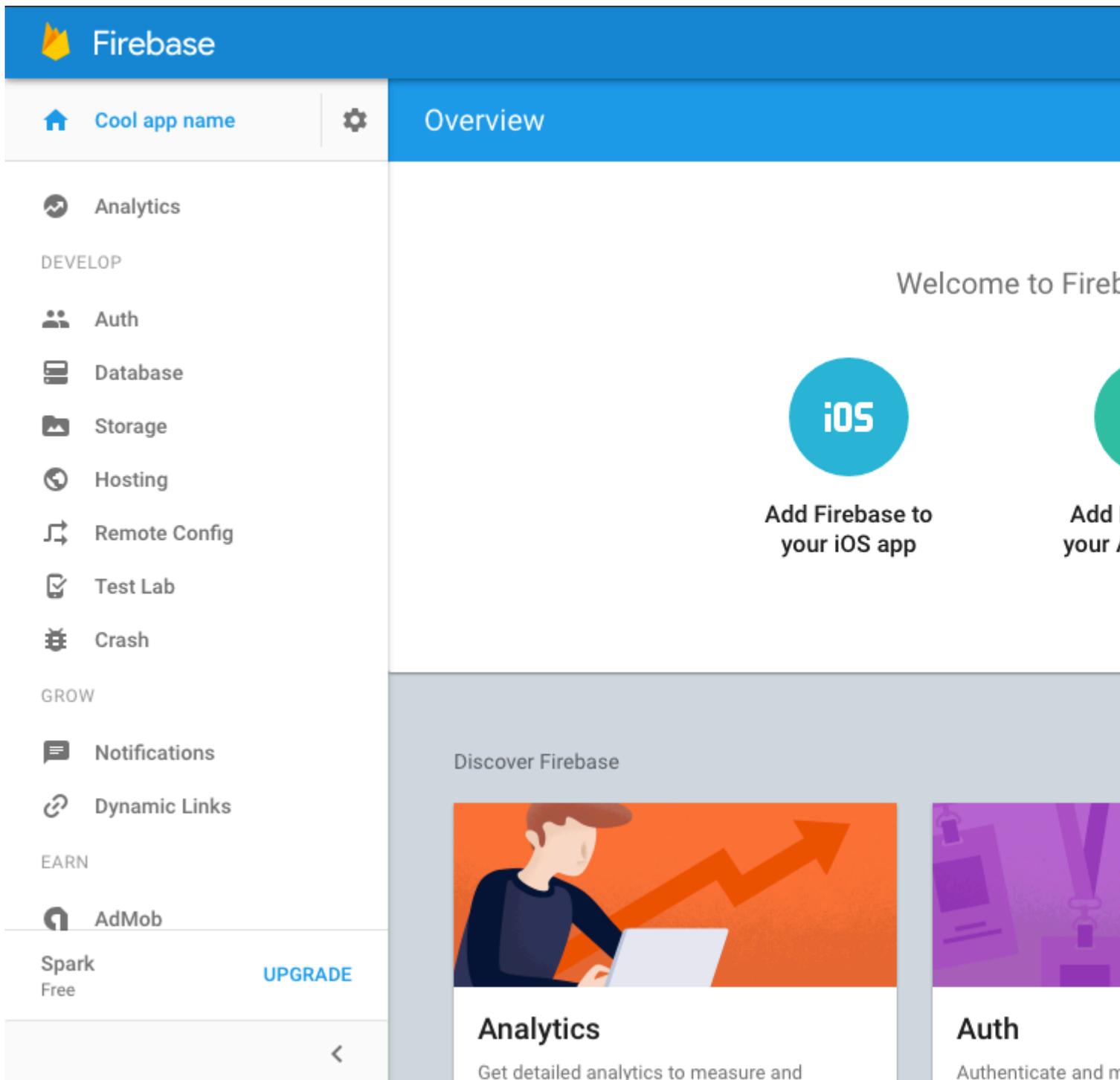
Country/region 

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](#)

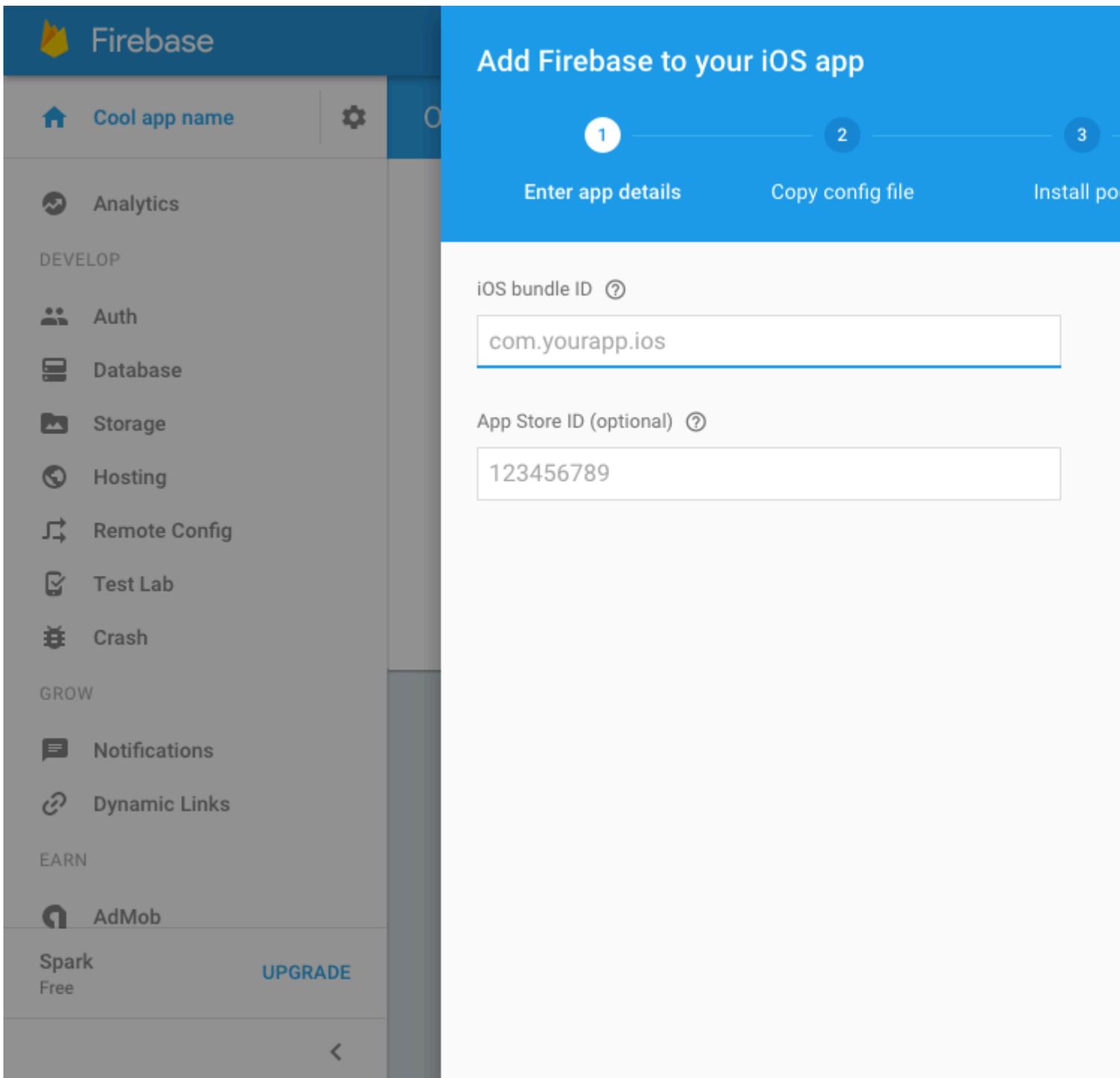
CANCEL

CREATE

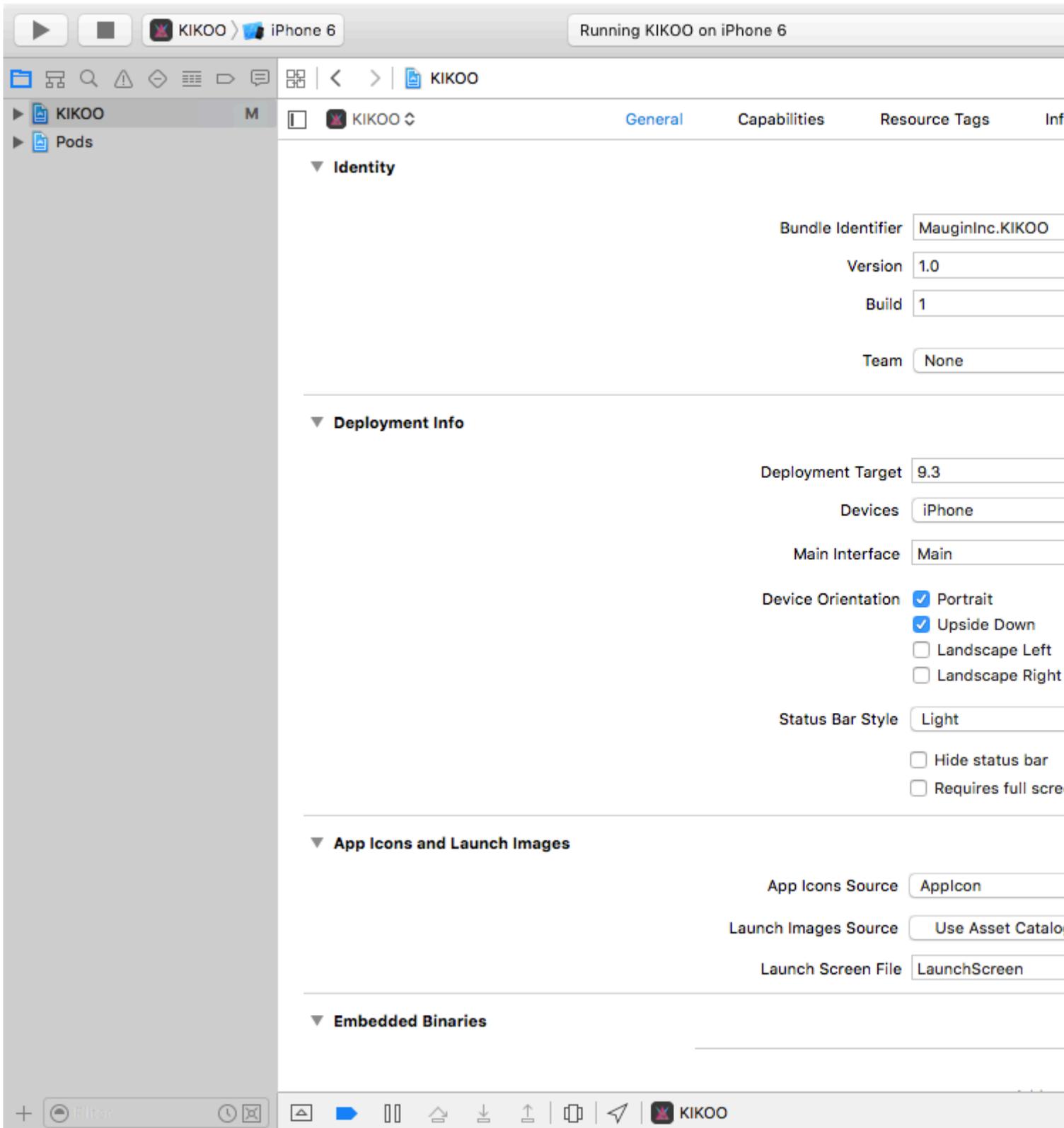
- Après avoir créé le projet, vous serez dirigé vers cette page qui est le tableau de bord et à partir de là, vous devez choisir une plate-forme sur laquelle vous souhaitez installer Firebase pour cet exemple, nous choisirons IOS.



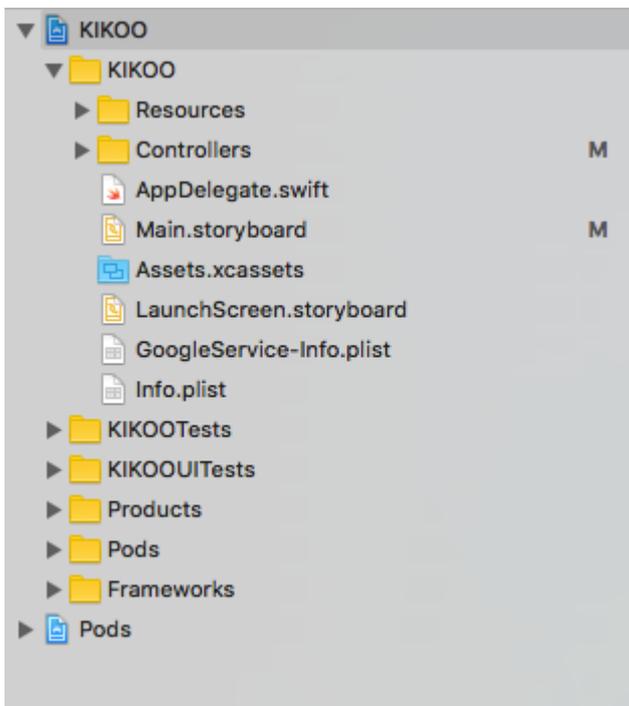
4. Après avoir sélectionné iOS, vous devriez voir la même fenêtre que celle de l'image ci-dessus demandant le paquet iOS et l'ID du magasin d'applications. Vous aurez seulement besoin de fournir le paquet iOS car notre application ne figure pas encore sur l'App Store.



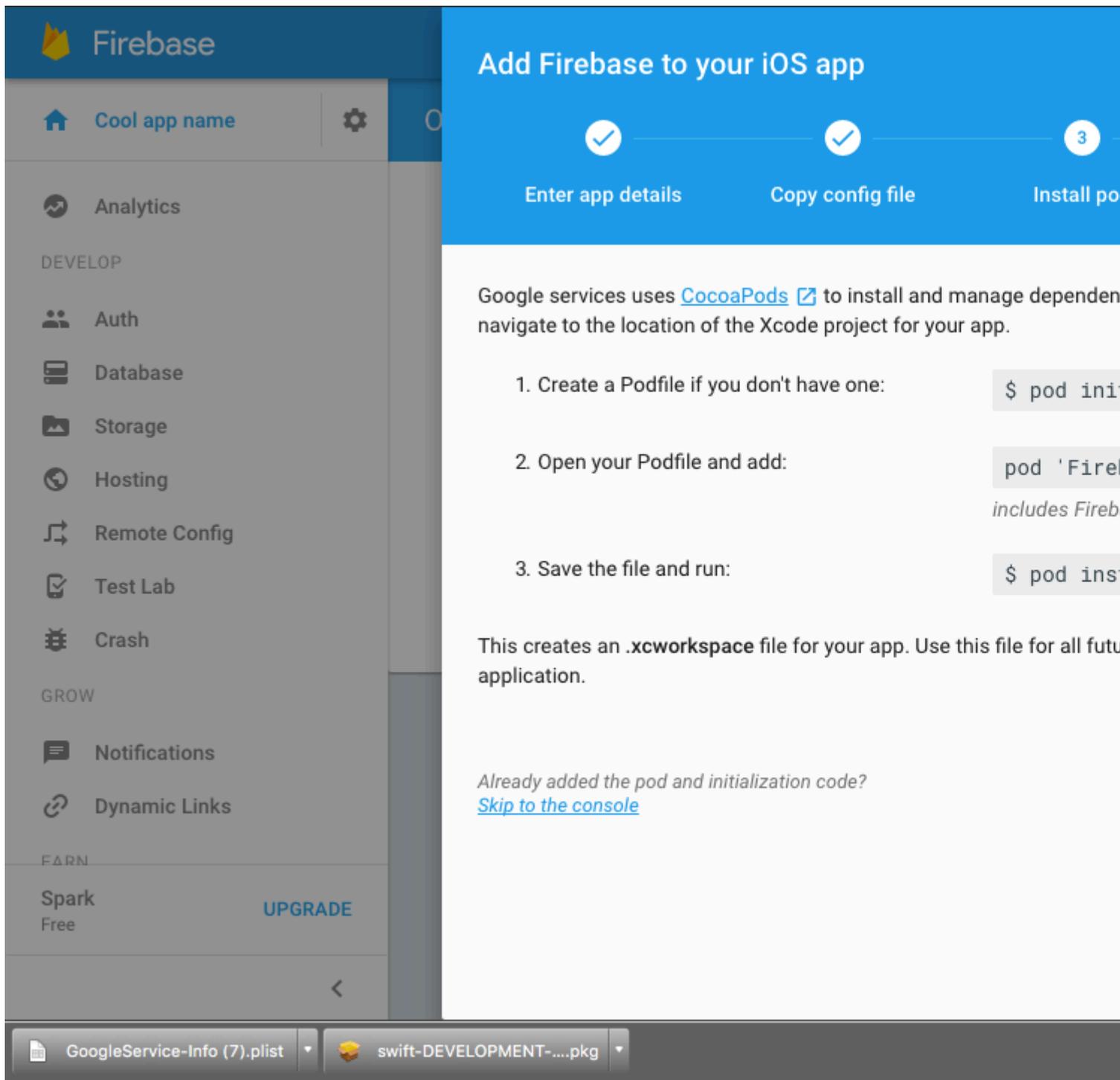
5. Obtenez l'ID de bundle de xcode après avoir créé un projet xcode de toute façon vous devriez normalement obtenir l'identifiant de bundle pour votre application sur la vue Général de l'application dans xcode, il sera le premier champ en haut et une fois le champ Bundle dans firebase par exemple le mien serait 'MauginInc.KIKOO'



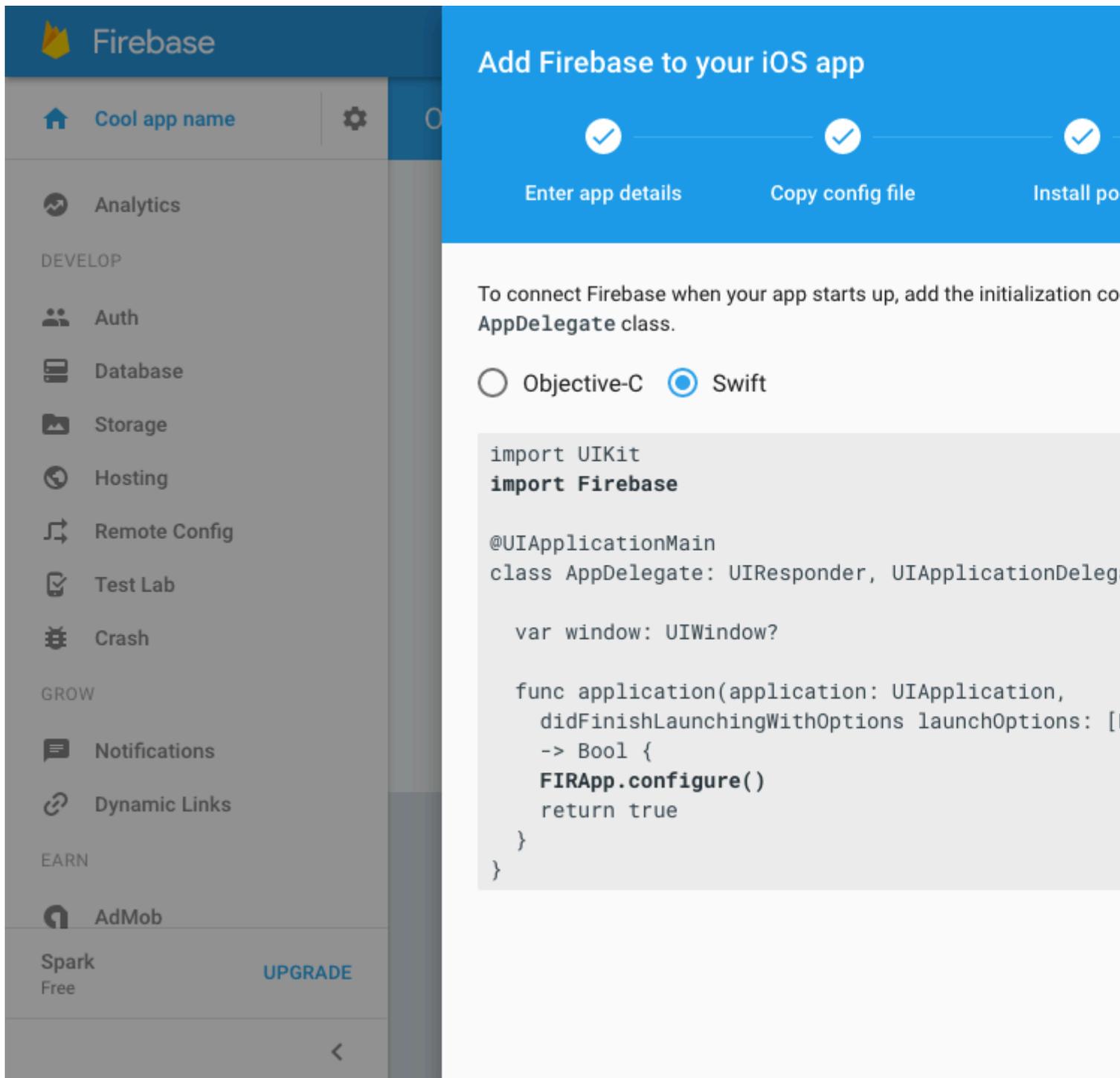
6. Après avoir fait cela et appuyé sur "Suivant", un fichier "GoogleService-Info.plist" sera téléchargé et ce que vous devez faire est de le déplacer dans le dossier racine de votre application dans xcode.



7. Vous voudrez initialiser les modules et installer les modules de base de données dont vous avez besoin pour effectuer cette opération en accédant à votre terminal et en accédant à votre dossier de projet xcode et suivez les instructions données par firebase.



8. Enfin, vous voulez configurer votre application pour que swift fasse ce qu'elle fait de mieux et que cela rende le développement d'applications beaucoup plus facile et efficace.



C'est tout ce que vous avez maintenant FireBase installé dans votre projet xcode pour IOS

Mise en route de Firebase avec une simple application Web Hello World en JavaScript

Cet exemple montre comment démarrer avec Firebase dans vos applications Web avec JavaScript.

Nous allons ajouter un **enfant de texte** dans notre base de données Firebase et l'afficher en temps réel sur notre application Web.

Commençons.

- Accédez à la console Firebase - <https://console.firebase.google.com> et créez un nouveau projet. Entrez le nom du projet, pays / région et cliquez sur **créer un projet** .

Learn more'. At the bottom, there are two buttons: 'CANCEL' and 'CREATE PROJECT'." data-bbox="107 157 814 598"/>

Create a project ✕

Project name

Country/region ?

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**

- Maintenant, créez un fichier **index.html** sur votre ordinateur. Et ajoutez-y le code suivant.

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

- Maintenant, allez à votre projet sur Firebase Console et vous pouvez le voir

Welcome to Firebase! Get started here.



Add Firebase to
your iOS app



Add Firebase to
your Android app



Add Firebase
your web app

- Cliquez maintenant sur **Ajouter Firebase à votre application Web** . Vous verrez apparaître la fenêtre suivante, cliquez sur le bouton de copie


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

- Vous avez maintenant terminé d'ajouter le code d'initialisation Firebase. Maintenant, nous devons obtenir notre valeur de **texte** de la base de données.
- Pour ce faire, ajoutez le code suivant (Initialize Firebase déjà ajouté à la dernière étape. Ne pas rajouter) dans le script dans **index.html**

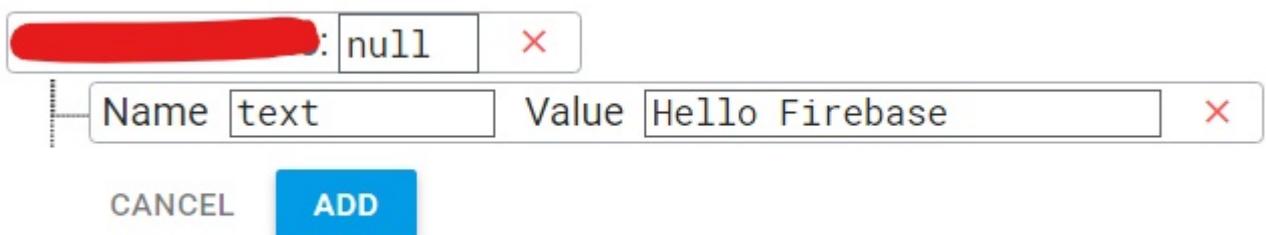
```
<script>

// 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.innerHTML = snap.val());

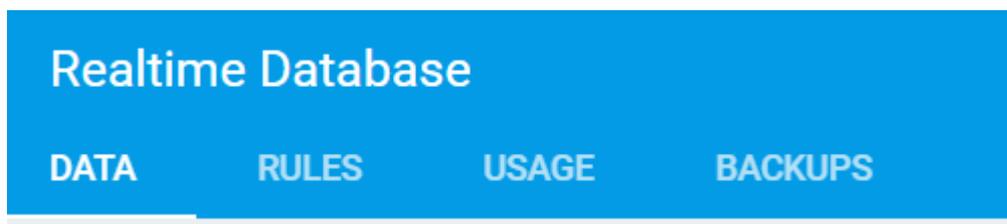
</script>
```

- Maintenant, nous avons tous terminé avec le fichier **index.html** et allons maintenant dans la **base de données** dans Firebase Console.
- Vous verrez que c'est vide et vide maintenant. Permet d'ajouter un **enfant texte** dans la base de données et d'y ajouter une valeur.



The screenshot shows a data entry form in the Firebase Realtime Database console. The form has two main fields: 'Name' and 'Value'. The 'Name' field contains the text 'text' and has a dropdown arrow on its left. The 'Value' field contains the text 'Hello Firebase'. There are two buttons at the bottom: a blue 'ADD' button and a grey 'CANCEL' button. A red bar is visible above the 'Name' field, and a red 'X' icon is in the top right corner of the form area.

- Cliquez maintenant sur le bouton **AJOUTER** .
- Maintenant, allez à la section **RÈGLES** de la base de données.



- À des fins de développement, nous allons maintenant activer toutes les requêtes de **lecture**

et d'écriture .

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

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

- Maintenant, ouvrez index.html dans le navigateur
- Vous verrez la valeur de texte sur votre page comme suit -

Getting started with Firebase

Hello Firebase

- Maintenant, si vous revenez à votre base de données et que vous modifiez la valeur du **texte pour enfant** , vous verrez que le texte dans le navigateur change également sans aucune actualisation ou rechargement. Voici comment la **base de données en temps réel** fonctionne sur Firebase.

Lire Démarrer avec Firebase en ligne: <https://riptutorial.com/fr/firebase/topic/816/demarrer-avec-firebase>

Chapitre 2: Capacités hors ligne de Firebase

Introduction

Dans cet article, vous trouverez les différentes manières d'implémenter des fonctionnalités hors ligne lors de l'utilisation de `Firestore`, des informations sur quand et pourquoi pourraient être une bonne idée pour activer des fonctionnalités hors ligne et des exemples avec la plate-forme Android.

Remarques

Que dois-je utiliser? Persistence du disque ou appels persistants?

D'après mon expérience, je peux dire que cela dépend toujours du fonctionnement de votre application et de la manière dont vous gérez les transactions et la base de données de votre application. Si, par exemple, vous avez une application où l'utilisateur écrit et lit simplement des données mais ne peut pas les supprimer ou les éditer, utilisez `DiskPersistence` serait le bon choix.

De plus, `DiskPersistence` stockera les données dans le cache, ce qui signifie que votre application utilisera plus d'espace dans les périphériques de l'utilisateur, ce qui n'est peut-être pas la meilleure idée dans votre cas.

D'autre part, si votre application gère beaucoup de transactions complexes et que vos données sont mises à jour très souvent, vous devriez peut-être éviter `DiskPersistence` et utiliser `keepSynced` dans les références que vous souhaitez conserver.

Pourquoi?

`DiskPersistence` stocke les données extraites en local, ce qui peut parfois entraîner une désynchronisation importante en affichant vos données si vous ne les utilisez pas avec `ContinentsListenerValueEvents`. Par exemple:

1. L'utilisateur A écrit un message "Hello World" sur votre application, qui est reçu par l'utilisateur B
2. L'utilisateur B télécharge le message de l'utilisateur A sur son téléphone et voit le message "Hello World"
3. Le message de l'utilisateur A est édité "Firebase is cool".
4. L'utilisateur B surveillera toujours le message "Hello World" même s'il met à jour les données car la référence de l'instantané est la même lorsque Firebase le filtre.

Pour éviter cela, la meilleure idée est de garder les auditeurs continus dans les références que vous souhaitez suivre à tout moment.

Puis-je utiliser les deux ensemble?

Bien sûr, vous pouvez le faire, et dans la plupart des applications, la meilleure idée est d'éviter de télécharger beaucoup de données et de donner à l'utilisateur la possibilité de travailler avec votre application même s'il n'a pas de connexion.

Si vous ne vous souciez pas de l'utilisation de l'espace de cache dans la `diskPersistence` utilisateur, je vous recommande d'activer `diskPersistence` dans votre objet `FirebaseDatabase` et d'ajouter également des indicateurs `keepSync` à chaque référence qui peut avoir beaucoup de temps dans un espace réduit. garder au frais tout le temps.

Exemples

Activer la persistance du disque (Android / iOS uniquement)

Pour activer la persistance du disque, vous devez activer l'indicateur `persistenceEnabled` dans l'objet `FirebaseDatabaseInstance` de votre application:

Android

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

iOS

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

Si vous souhaitez désactiver la persistance à certains moments du cycle de vie de votre application, n'oubliez pas de la désactiver de la même manière:

Android

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

iOS

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

Garder les données à jour (Android / iOS uniquement)

Firebase synchronise et stocke une copie locale des données pour les écouteurs actifs lorsqu'ils sont utilisés sur des appareils mobiles. De plus, vous pouvez conserver des emplacements spécifiques synchronisés.

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)
```

Le client Firebase télécharge automatiquement les données à ces emplacements et les met à jour même si la référence ne contient aucun écouteur actif. Vous désactivez la synchronisation avec la ligne de code suivante.

Android :

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

iOS:

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

Lire Capacités hors ligne de Firebase en ligne:

<https://riptutorial.com/fr/firebase/topic/10777/capacites-hors-ligne-de-firebase>

Chapitre 3: Comment écouter les erreurs lors de l'accès à la base de données?

Introduction

Il y a de nombreuses raisons pour lesquelles une opération de lecture ou d'écriture peut échouer. Un problème fréquent est que vos règles de sécurité rejettent l'opération, par exemple parce que vous n'êtes pas authentifié (par défaut, une base de données n'est accessible que par un utilisateur authentifié) ou parce que vous écrivez / écoutez à un endroit où vous ne l'êtes pas. avoir la permission

Exemples

Détecter les erreurs lors de l'écriture d'une valeur sur Android

Il existe de nombreuses raisons pour lesquelles une opération d'écriture peut échouer. Un problème fréquent est que vos règles de sécurité rejettent l'opération, par exemple parce que vous n'êtes pas authentifié (par défaut, une base de données n'est accessible que par un utilisateur authentifié).

Vous pouvez voir ces violations de règles de sécurité dans la sortie logcat. Mais il est facile de les ignorer. Vous pouvez également les gérer dans votre propre code et les rendre plus visibles, ce qui est particulièrement utile pendant le développement (puisque votre JSON, vos règles et votre code changent souvent).

Pour détecter un échec d'écriture sur Android, vous devez [joindre un rappel d'achèvement à setValue](#) :

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

Lancer une exception comme celle-là garantit qu'il sera très difficile d'ignorer une telle erreur la prochaine fois.

Détecter les erreurs lors de la lecture de données sur Android

Une raison fréquente pour laquelle votre opération de lecture peut ne pas fonctionner est que vos règles de sécurité rejettent l'opération, par exemple parce que vous n'êtes pas authentifié (par défaut, une base de données n'est accessible que par un utilisateur authentifié).

Vous pouvez voir ces violations de règles de sécurité dans la sortie logcat. Mais il est facile de les ignorer. Vous pouvez également les gérer dans votre propre code et les rendre plus visibles, ce

qui est particulièrement utile pendant le développement (puisque votre JSON, vos règles et votre code changent souvent).

Pour détecter une lecture échouée sur Android, vous devez implémenter la méthode `onCancelled` de votre `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();
    }
});
```

Ou si vous avez un `ValueEventListener` :

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

Avec ce code en place, il sera difficile de négliger une erreur de sécurité lors de la lecture de données sur Android.

Détecter les erreurs lors de l'écriture d'une valeur sur iOS

Il existe de nombreuses raisons pour lesquelles une opération d'écriture peut échouer. Un problème fréquent est que vos règles de sécurité rejettent l'opération, par exemple parce que vous n'êtes pas authentifié (par défaut, une base de données n'est accessible que par un utilisateur authentifié).

Vous pouvez voir ces violations de règles de sécurité dans la sortie de votre programme. Mais il est facile de les ignorer. Vous pouvez également les gérer dans votre propre code et les rendre plus visibles, ce qui est particulièrement utile pendant le développement (puisque votre JSON, vos règles et votre code changent souvent).

Pour détecter un échec d'écriture sur iOS, vous devez [joindre un bloc d'achèvement à setValue](#) :

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

Lancer une exception comme celle-là garantit qu'il sera très difficile d'ignorer une telle erreur la prochaine fois.

Détection des erreurs lors de la lecture de données en JavaScript

Une raison fréquente pour laquelle votre opération de lecture peut ne pas fonctionner est que vos règles de sécurité rejettent l'opération, par exemple parce que vous n'êtes pas authentifié (par défaut, une base de données n'est accessible que par un utilisateur authentifié).

Vous pouvez voir ces violations de règles de sécurité dans la console JavaScript de votre navigateur. Mais il est facile de les ignorer. Vous pouvez également les gérer dans votre propre code et les rendre plus visibles, ce qui est particulièrement utile pendant le développement (puisque votre JSON, vos règles et votre code changent souvent).

Pour détecter un échec de lecture en JavaScript, vous devez implémenter un deuxième rappel à votre clause `on()` :

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

Avec ce code en place, il sera difficile de négliger une erreur de sécurité lors de la lecture de données en JavaScript.

Détection des erreurs lors de l'écriture d'une valeur en JavaScript

Il existe de nombreuses raisons pour lesquelles une opération d'écriture peut échouer. Un problème fréquent est que vos règles de sécurité rejettent l'opération, par exemple parce que vous n'êtes pas authentifié (par défaut, une base de données n'est accessible que par un utilisateur authentifié).

Vous pouvez voir ces violations de règles de sécurité dans la sortie de la console. Mais il est facile de les ignorer. Vous pouvez également les gérer dans votre propre code et les rendre plus visibles, ce qui est particulièrement utile pendant le développement (puisque votre JSON, vos règles et votre code changent souvent).

Pour détecter un échec d'écriture en JavaScript, vous devez joindre un rappel d'achèvement à `set` :

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

Afficher une alerte comme celle-ci garantit qu'il sera très difficile d'ignorer une telle erreur la prochaine fois.

Détecter les erreurs lors de la lecture de données sur iOS

Une raison fréquente pour laquelle votre opération de lecture peut ne pas fonctionner est que vos règles de sécurité rejettent l'opération, par exemple parce que vous n'êtes pas authentifié (par défaut, une base de données n'est accessible que par un utilisateur authentifié).

Vous pouvez voir ces violations de règles de sécurité dans la sortie de la console. Mais il est facile de les ignorer. Vous pouvez également les gérer dans votre propre code et les rendre plus visibles, ce qui est particulièrement utile pendant le développement (puisque votre JSON, vos règles et votre code changent souvent).

Pour détecter une lecture échouée sur iOS, vous devez implémenter le bloc `withCancel` de votre observateur:

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

Lire Comment écouter les erreurs lors de l'accès à la base de données? en ligne:

<https://riptutorial.com/fr/firebase/topic/5548/comment-ecouter-les-erreurs-lors-de-l-acces-a-la-base-de-donnees->

Chapitre 4: Comment obtenir la valeur de la clé push de la base de données Firebase?

Introduction

Dans Firebase Database, tout est un noeud qui suit la clé de motif: value. La base de données Firebase nous fournit un moyen simple de générer des clés uniques. Des clés uniques créent de nouveaux éléments pendant le téléchargement des données vers une clé précédemment stockée.

Exemples

Exemple Android

Supposons que nous avons une application Dogs, alors notre modèle sera une classe Dog.

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

C'est comment envoyer un chien à la base de données, un nouveau chien unique et mettre le chien avec la clé.

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

La `reference.child(key).setValue(dog)` ; est équivalent à `reference.push().setValue(dog)` ; Et ajoutez l'avantage d'obtenir la clé dans l'objet `Dog` .

Lire [Comment obtenir la valeur de la clé push de la base de données Firebase?](https://riptutorial.com/fr/firebase/topic/10839/comment-obtenir-la-valeur-de-la-cle-push-de-la-base-de-donnees-firebase-) en ligne:

<https://riptutorial.com/fr/firebase/topic/10839/comment-obtenir-la-valeur-de-la-cle-push-de-la-base-de-donnees-firebase->

Chapitre 5: Comment utiliser FirebaseRecyclerAdapter au lieu de RecyclerAdapter?

Exemples

Voici l'exemple d'utilisation du composant FirebaseUi FirebaseRecyclerAdapter

Bonjour les amis avant de commencer le code, nous avons besoin de déclarer une dépendance pour accéder au composant ui de firebase, alors voici la dépendance que vous pouvez mettre dans votre gradle autrement vous pouvez ajouter une dépendance en tant que jar également.

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

Ensuite, nous interrogeons dans la base de données firebase des données comme suit

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

Ensuite, après que nous passions la requête dans FirebaseRecyclerAdapter comme suit

```
private void setUpFirebaseAdapter(Query query) {  
  
    mFirebaseAdapter = new FirebaseRecyclerAdapter<UserModel, FirebaseViewHolder>  
        (UserModel.class, R.layout.row_user_list, FirebaseViewHolder.class, query)  
    {  
        @Override  
        protected void populateViewHolder(FirebaseViewHolder 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 (classe de modèle)

```
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 (mise en page pour la ligne dans la vue du recycleur)

```

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

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

```

```

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

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

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

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

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

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

</RelativeLayout>

```

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

Lire Comment utiliser [FirebaseRecyclerAdapter](https://riptutorial.com/fr/firebase/topic/8982/comment-utiliser-firebase-recycleradapter-au-lieu-de-recycleradapter-) au lieu de [RecyclerViewAdapter](https://riptutorial.com/fr/firebase/topic/8982/comment-utiliser-firebase-recycleradapter-au-lieu-de-recycleradapter-)? en ligne:
<https://riptutorial.com/fr/firebase/topic/8982/comment-utiliser-firebase-recycleradapter-au-lieu-de-recycleradapter->

Chapitre 6: Comment utiliser la base de données Firebase pour conserver une liste d'utilisateurs d'authentification Firebase

Exemples

Comment enregistrer les données de profil utilisateur

Chaque utilisateur authentifié a un Firebase `uid` qui est unique parmi tous les fournisseurs et est retourné dans le résultat de chaque méthode d'authentification.

Un bon moyen de stocker les données de votre utilisateur consiste à créer un nœud pour conserver toutes les données des utilisateurs et les protéger en utilisant vos règles de sécurité.

- Base de données

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

- Sécurité

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

Le `$uid` dans les règles ci-dessus est une "variable dollar", qui garantit que les règles sous-jacentes sont appliquées à tous les nœuds enfants des `users`. Pour plus d'informations, reportez-vous à la documentation sur l' [utilisation de variables \\$ pour capturer des segments de chemin](#).

Pourquoi enregistrer les données utilisateur dans la base de données

L'authentification Firebase permet aux utilisateurs de votre application de se connecter avec des fournisseurs de services sociaux ou leur courrier électronique + mot de passe. Mais que faire si vous souhaitez stocker des informations supplémentaires sur un utilisateur, au-delà de ce que l'authentification Firebase vous permet de spécifier?

Ou si vous souhaitez afficher une liste des utilisateurs dans votre application? L'authentification Firebase n'a pas d'API pour cela.

La plupart des développeurs résolvent ce problème en stockant les informations supplémentaires dans une base de données distincte. Cette rubrique explique comment stocker ces informations dans la [base de données Firebase Realtime](#) .

Gestion des données de compte d'utilisateur dans la base de données en temps réel

Le système auth Firebase est la source d'un utilisateur `uid` , `displayName` , `photoURL` , et peut - être `email` . Les comptes basés sur un mot de passe définissent ces valeurs *persistantes* dans le système d'authentification via la méthode `.updateProfile` . Stocker ces valeurs dans la base de données Realtime, rDB, le nœud `users` pose le problème des données obsolètes. Les noms d'affichage, par exemple, peuvent changer. Pour conserver ces valeurs en synchronisation, utilisez [le stockage local](#) de concert avec `.onAuthStateChange` .

sur chaque `.onAuthStateChange`

- `getItem('displayName')` **et** `getItem('photoURL')`
- **comparer à** `user.displayName` **et** `user.photoURL`
- **si différent**
 - `setItem('displayName')` **et** `setItem('photoURL')`
 - `db.ref.child('users').update` **les valeurs de** `displayName` **et / ou** `photoURL`

`.onAuthStateChange` déclenche à chaque chargement ou rechargement de page, ainsi qu'à chaque changement d'état d'authentification. Il se déclenche souvent, par exemple des applications multi-pages. Toutefois, la lecture et l'écriture sur le stockage local sont synchrones et très rapides. Il n'ya donc pas d'impact notable sur les performances des applications.

[Lire Comment utiliser la base de données Firebase pour conserver une liste d'utilisateurs d'authentification Firebase en ligne: https://riptutorial.com/fr/firebase/topic/1729/comment-utiliser-la-base-de-donnees-firebase-pour-conserver-une-liste-d-utilisateurs-d-authentification-firebase](#)

Chapitre 7: Console Firebase

Syntaxe

1. Exemple d'analyse Firebase.
2. Explication de la console Firebase pour chaque composant.

Paramètres

Firestore Analytics	Analyse de Firestore et ses différents composants
Console Firestore	Comment ça marche? & Comment les détails sont-ils affichés dans le tableau de bord?

Remarques

Ce document est très utile pour ceux qui débutent les analyses de la base de données. Cela sera très utile pour comprendre comment fonctionne l'analyse de la base de données dans les différents scénarios.

Exemples

Firestore All In One

[Informations sur la console Firestore en détail](#)

[Android: Exemple d'analyse Firestore](#)

Étapes pour Android:

- Télécharger le code du lien
- Vérifiez `FirestoreAnalyticsActivity`
- C'est tout ce que vous comprendrez comment fonctionne l'analyse de firestore pour les différents scénarios.

Lire Console Firestore en ligne: <https://riptutorial.com/fr/firebase/topic/6660/console-firebase>

Chapitre 8: Espace de rangement

Remarques

Firebase Storage fournit des téléchargements et des téléversements de fichiers sécurisés pour vos applications Firebase, quelle que soit la qualité du réseau. Vous pouvez l'utiliser pour stocker des images, du son, de la vidéo ou tout autre contenu généré par l'utilisateur. Firebase Storage est soutenu par Google Cloud Storage, un service de stockage d'objets puissant, simple et économique.

Firebase Storage stocke vos fichiers dans un compartiment Google Cloud Storage partagé avec l'application Google App Engine par défaut, les rendant ainsi accessibles via les API Firebase et Google Cloud. Cela vous permet de télécharger et de téléverser des fichiers à partir de clients mobiles via Firebase et d'effectuer un traitement côté serveur, tel que le filtrage d'images ou le transcodage vidéo, à l'aide de Google Cloud Platform. Firebase Storage s'adapte automatiquement, ce qui signifie qu'il n'est pas nécessaire de migrer de Firebase Storage vers Google Cloud Storage ou tout autre fournisseur.

Cette intégration rend les fichiers accessibles directement à partir des bibliothèques client Google Cloud Storage gcloud, de sorte que vous pouvez utiliser Firebase Storage avec vos langues côté serveur préférées. Pour plus de contrôle, vous pouvez également utiliser les API XML et JSON de Google Cloud Storage.

Firebase Storage s'intègre parfaitement à l'authentification Firebase pour identifier les utilisateurs et fournit un langage de sécurité déclaratif qui vous permet de définir des contrôles d'accès sur des fichiers individuels ou des groupes de fichiers afin de rendre vos fichiers publics ou privés.

Consultez la [documentation publique pour Firebase Storage](#) pour obtenir les API, les exemples et les exemples d'applications les plus récents.

Exemples

Démarrer sur iOS

Conditions préalables

1. Créez un nouveau projet et ajoutez une application iOS à ce projet dans la [console Firebase](#).
2. Téléchargez et incluez `GoogleServices-Info.plist` dans votre application.

Ajouter Firebase Storage à votre application

Ajoutez la dépendance suivante au `Podfile` votre projet:

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

Exécutez `pod install` et ouvrez le fichier `.xcworkspace` créé.

Suivez ces instructions pour installer Firebase sans CocoaPods

Configurer le stockage Firebase

Vous devez initialiser Firebase avant de créer ou d'utiliser une référence à une application Firebase. Si vous l'avez déjà fait pour une autre fonctionnalité Firebase, vous pouvez ignorer les deux étapes suivantes.

Importez le module Firebase:

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

```
// Swift
import Firebase
```

Configurez une `FIRApp` partagée `FIRApp`, généralement dans l'application de votre `application:didFinishLaunchingWithOptions:` `method:`

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

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

Obtenez une référence au service de stockage à l'aide de l'application Firebase par défaut:

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

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

Créez une référence à un fichier dans Firebase Storage:

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

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

Chargez un fichier dans 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
    }
}
```

Lire Espace de rangement en ligne: <https://riptutorial.com/fr/firebase/topic/4281/espace-de-rangement>

Chapitre 9: File d'attente Firebase

Exemples

Comment utiliser la file d'attente Firebase en tant que backend pour votre application

Firebase fournit un backend en tant que service, en tant que développeur d'applications, vous n'avez pas la possibilité d'avoir du code backend.

Cet exemple montre comment utiliser la file d'attente de firebase, créer un backend qui fonctionnera en haut de la base de données firebase et qui servira de backend pour votre application frontend.

Avant d'entrer dans le code, comprenons l'architecture, comment cela fonctionnera. Par souci de concision, supposons que nous utilisons un site Web en tant que serveur frontal et serveur NodeJ en tant que serveur principal.

Conditions préalables

1. Créez une application Firebase à l'aide de votre compte Google
2. Ajoutez une base de feu à votre page Web. Utilisez `bower install firebase --save`
3. Créez un compte de service à l'aide de votre nouveau compte Firebase créé (Paramètres-> Autorisations -> Comptes de service -> CRÉER UN COMPTE DE SERVICE -> (spécifiez le nom et cochez la case "Fournir une nouvelle clé privée") - cela plus tard.
4. Configurez le serveur NodeJs qui peut être hébergé dans votre environnement préféré
5. Créer le noeud final suivant dans la `queue/specs`

"demande de réponse":

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

6. Dans le serveur NodeJs, `npm install firebase --save` version côté serveur de Firebase, `npm install firebase --save`, et intialize votre compte de service en utilisant le fichier json que nous avons obtenu à l'étape 3, cela ressemble à ceci:

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

Architecture

Voici le cycle complet comment ça marche.

Du côté du frontend tu vas faire ces étapes

1. En utilisant WebSdk Firebase, vous écrivez vos requêtes directement dans la base de données firebase dans la file d'attente / tâches du noeud final, appelons cela votre requête que vous envoyez au backend.
2. Après avoir inséré votre tâche, vous enregistrez l'écouteur dans la `queue/tasks/{taskKey}` du noeud final `queue/tasks/{taskKey}` qui sera appelée lorsque le serveur finit de traiter votre demande, en écrivant la réponse dans la tâche ci-dessus.

Dans le côté backend tu vas faire ces étapes

1. Créer un serveur qui écoute à l'infini les "files d'attente / tâches" des points d'extrémité
2. Traite vos tâches et écrit les données de réponse dans la `queue/tasks/response`
3. Supprimer la tâche

Tout d'abord créer cette fonction d'assistance, qui permet de gérer les rappels et les promesses ensemble

```
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;
}
```

Dans le côté frontal, vous aurez cette fonction

```
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;
}

```

vous pouvez utiliser cette fonction comme `sendRequest('CreateHouseFacade', {houseName:'Test'})`.

Le paramètre Kind est pour le backend, pour savoir quelle méthode appeler pour la demande de traitement. Les paramètres permettent de transmettre des informations supplémentaires sur les paramètres.

Et voici le code backend

```

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 fonction `myService` contient votre code de logique métier qui va `CreateHouseFacade` requête `CreateHouseFacade` .

Lire File d'attente Firebase en ligne: <https://riptutorial.com/fr/firebase/topic/7619/file-d-attente-firebase>

Chapitre 10: Firebase Realtime Database avec Android

Exemples

Comment connecter une base de données en temps réel avec une application Android

Comment implémenter la base de données FirebaseRealTime dans une application Android. Voici les étapes à suivre pour le faire.

1. Installez d'abord sdk Firebase, si vous ne savez pas comment installer, voici l'URL d'aide. [Installez Firebase SDK](#)
2. Après avoir enregistré votre projet dans la console Firbase, l'URL de la console Firbase est l'URL de la console [Firebase](#).
3. Une fois que vous avez terminé avec succès, ajoutez les dépendances suivantes au niveau de votre application. compile 'com.google.firebase: base de données firebase: 9.2.1'
4. En outre, une autre chose permet de configurer les règles de votre base de données Firebase. Si vous ne savez pas comment configurer, voici l'URL qui vous aide. [Configurer les règles de la base de feu](#)
5. Maintenant, après tout, le code d'origine est démarré, récupérez d'abord votre instance de base de données, lancez FirebaseDatabase comme suit,

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

Vous pouvez maintenant créer différents objets différents de DatabaseReference pour le noeud d'accès différent,

6. Maintenant, vous pouvez enregistrer ou récupérer des données à l'aide de DataBaseReference comme suit, pour l'enregistrement:

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

Lire les données:

```
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());
    }
});
```

Note: Ceci est le seul sujet d'introduction pour implémenter une base de données dans une application Android perdue de plus de choses disponibles dans la base de données FirebaseRealtime,

Lire Firbase Realtime Database avec Android en ligne:

<https://riptutorial.com/fr/firebase/topic/6482/firebase-realtime-database-avec-android>

Chapitre 11: FirebaseUI

Remarques

Firebase est une suite de produits intégrés conçus pour vous aider à développer votre application, développer une base d'utilisateurs engagée et gagner plus d'argent. Il comprend des outils qui vous aident à créer votre application, comme une base de données en temps réel, un stockage de fichiers et une authentification utilisateur, ainsi que des outils pour développer et monétiser votre application, comme les notifications push, les analyses, les

Vous pouvez considérer Firebase comme un ensemble de briques Lego que vous pouvez utiliser pour créer votre chef-d'œuvre. Tout comme les briques, Firebase est relativement peu sollicité, car il existe un nombre infini de façons de combiner les pièces et nous ne vous dirons pas que certaines méthodes sont fausses :)

FirebaseUI est basé sur Firebase et fournit aux développeurs des liaisons mobiles natives simples, personnalisables et prêtes pour la production par-dessus les primitives Firebase afin d'éliminer le code passe-partout et de promouvoir les meilleures pratiques de Google.

Dans l'analogie avec Lego, FirebaseUI est un ensemble de kits prédéfinis avec des instructions que vous pouvez retirer de l'étagère et adapter à vos besoins. Vous pouvez voir comment nous avons utilisé les composants individuels de Firebase pour construire FirebaseUI car FirebaseUI est open source. FirebaseUI doit faire l'objet d'un avis - nous vous disons comment nous pensons que les briques doivent aller ensemble, alors nous faisons des choix. Mais comme FirebaseUI est open source, vous pouvez modifier ce que nous faisons pour mieux répondre à vos besoins.

Si vous construisez une ville de Lego, vous préférez tirer un tas de maisons d'une collection de pré-construction et les modifier légèrement pour répondre à vos besoins plutôt que de partir de zéro et concevoir chaque bâtiment à la main, non?

FirebaseUI vous permet de faire exactement cela, c'est pourquoi nous l'incluons dans nos exemples d'applications et d'exemples. Les développeurs (y compris nous-mêmes) sont paresseux - nous voulons la meilleure réutilisation de notre code et les exemples les plus concis, et FirebaseUI nous permet de fournir des exemples de très grande qualité qui se traduisent par de très bonnes expériences utilisateur à un coût de développement réduit.

Exemples

Premiers pas avec FirebaseUI

FirebaseUI propose [des clients Android](#), [iOS](#) et [Web](#). Vous pouvez commencer avec eux comme ça:

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" />
```

Lire FirebaseUI en ligne: <https://riptutorial.com/fr/firebase/topic/6418/firebaseui>

Chapitre 12: FirebaseUI (Android)

Exemples

Ajouter les dépendances

FirebaseUI est juste une bibliothèque open source de Google qui fournit des liaisons UI faciles pour Firebase Auth et Firebase Database.

Pour commencer à ajouter FirebaseUI à votre application, ajoutez ces dépendances dans le fichier `build.gradle` votre application:

```
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'
```

Remplir un ListView

En supposant que vous avez déjà configuré une application dans Android Studio, ajoutez un objet `ListView` à un modèle (ou ignorez si cela est déjà fait):

```
<?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>
```

Maintenant, créons un modèle pour les données que nous allons remplir dans notre `ListView` avec:

```
public class Person {

    private String name

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

    public String getName() {
        return name;
    }

}
```

Assurez-vous que votre `ListView` a un identifiant, puis créez une référence dans votre `Activity` et définissez son adaptateur sur un nouveau `FirestoreListAdapter` :

```
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
         * like limitToLast() and etc.
         */
        DatabaseReference peopleReference = FirebaseDatabase.getInstance().getReference()
            .child("people");

        // Now set the adapter with a given layout
        mListView.setAdapter(new FirestoreListAdapter<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());
            }
        });
    }
}
```

Une fois cela fait, ajoutez des données à votre base de données et regardez la `ListView` elle-même.

Lire [FirestoreUI \(Android\)](https://riptutorial.com/fr/firebase/topic/6610/firebaseui--android-) en ligne: <https://riptutorial.com/fr/firebase/topic/6610/firebaseui--android->

Chapitre 13: Fonctions Cloud pour Firebase

Introduction

Firebase a lancé sa version bêta de Cloud Functions for Firebase, similaire à l'utilisation des fonctions Cloud sur Google Cloud Platform.

Cloud Functions est un environnement Node.js hébergé, privé et évolutif dans lequel vous pouvez exécuter du code JavaScript. Firebase SDK for Cloud Functions intègre la plate-forme Firebase en vous permettant d'écrire du code qui répond aux événements et appelle des fonctionnalités exposées par d'autres fonctionnalités Firebase.

Exemples

Envoyer des emails de notification de bienvenue aux utilisateurs pour s'abonner.

Utilisez le dépôt GitHub pour obtenir le code complet: <https://github.com/firebase/functions-samples/blob/master/quickstarts/email-users>

- Copiez ou clonez le référentiel sur votre ordinateur.

Allez maintenant sur votre console Firebase

- Créez un projet Firebase à l'aide de la console Firebase.
- Activer le fournisseur **Google** dans la section **Auth**
- Collez l'extrait d' **initialisation Web** de: **Firestore Console > Overview > Ajoutez Firebase** à votre application Web dans le **fichier public / index.html** où se trouve le **TODO** .

```
* TODO(DEVELOPER): Paste the initialization snippet from: Firestore Console > Overview > Add  
Firestore 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>
```

Installez Firewall CLI sur votre ordinateur

- Si vous n'avez pas déjà installé **NodeJS** , installez-le depuis <https://nodejs.org/en/> (Assurez-vous d'avoir la version mise à jour de NodeJS installée sur votre ordinateur.)
- Ouvrez l'invite de commande / terminal et installez-le avec **npm install -g firebase-tools** , puis configurez-le avec la **connexion firebase**.
- Pour choisir votre projet que vous avez créé maintenant ==> Configurez la CLI localement en utilisant **firebase use --add** et sélectionnez votre projet dans la liste.
- Installez les dépendances localement en exécutant: les **fonctions cd; npm installer; cd -**

Définir des variables d'environnement Google Cloud

- Définissez les variables d'environnement Google Cloud **gmail.email** et **gmail.password** pour **qu'elles** correspondent à l'adresse e-mail et au mot de passe du compte Gmail utilisé pour envoyer des e-mails. Pour cela, **ouvrez l'invite de commande ou le terminal** et tapez la commande suivante de l'interface de ligne de commande Firebase:

```
fonctions firebase: config: set gmail.email = "myusername@gmail.com" gmail.password = "secretpassword"
```

Déployer le projet et tester

- Pour déployer le projet, ouvrez le **cmd / terminal** et utilisez la commande **firebase deploy** pour démarrer le déploiement.

```
=== 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
```

- Une fois cela fait, utilisez la commande pour ouvrir le site dans le navigateur **firebase open hosting:** ou faites-le manuellement à partir de l'URL affichée.

Lire Fonctions Cloud pour Firebase en ligne: <https://riptutorial.com/fr/firebase/topic/9580/fonctions-cloud-pour-firebase>

Chapitre 14: Notification push du serveur personnalisé

Introduction

Cela peut être fait en utilisant 2 méthodes avec la **requête HTTP Post** , avec le **SDK admin Firebase** exécuté sur votre serveur. Ici, je vais discuter des deux.

Exemples

Protocole HTTP Firebase Cloud Messaging

De votre demande de serveur au lien ci-dessous pour envoyer la notification avec certains paramètres de demande

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

Tout en demandant des en-têtes comme suit

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

Le corps de la demande varie

```
{
  "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
  }
}
```

Les paramètres prennent Array de jetons comme

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

ou un seul jeton comme

```
"token"
```

ou un nom de sujet commençant par / **topic** / like

```
"/topic_name/"
```

Pour les conditions d'utilisation de plusieurs sujets utilisant || et les opérateurs && aiment

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

Utilisation du SDK Admin (Nœud js)

Initialement initier le SDK Firebase et le SDK d'administration

```
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"
});
```

Créez une chaîne JSON de charge utile comme dans le premier exemple.

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

Appelez ensuite différentes méthodes d'envoi pour envoyer la notification.

Pour sujet

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

Pour appareil

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

Lire Notification push du serveur personnalisé en ligne:

<https://riptutorial.com/fr/firebase/topic/10548/notification-push-du-serveur-personnalise>

Chapitre 15: Rapport de collision

Remarques

Crash Reporting crée des rapports détaillés sur les erreurs de votre application.

Les erreurs sont regroupées en grappes de traces de pile similaires et triées en fonction de la gravité de l'impact sur vos utilisateurs. En plus des rapports automatiques, vous pouvez enregistrer des événements personnalisés pour vous aider à capturer les étapes précédant un plantage.

Crash Reporting est actuellement en version bêta tandis que nous résolvons certains problèmes connus sur Android et iOS.

Documentation officielle

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

Exemples

Configurer Crash Reporting dans Android

1. Complétez la partie [Installation et configuration](#) pour connecter votre application à Firebase. Cela créera le projet dans Firebase.
2. Ajoutez la dépendance de Firebase CrashReporting à votre fichier `build.gradle` niveau du `build.gradle` :

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

Signaler l'erreur dans Android

Firebase Crash Reporting génère automatiquement des rapports pour les erreurs fatales (ou les exceptions non interceptées).

Vous pouvez créer votre rapport personnalisé en utilisant:

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

Vous pouvez archiver le journal lorsque FirebaseCrash a initialisé le module:

```
07-20 08: 57: 24.442 D / FirebaseCrashApilImpl: API de génération de rapports  
FirebaseCrash initialisée 07-20 08: 57: 24.442 I / FirebaseCrash: initialisation des  
rapports FirebaseCrash d com.google.firebase.crash.internal.zzg@3333d325 07-20  
08: 57: 24.442 D / FirebaseApp: classe initialisée  
com.google.firebase.crash.FirebaseCrash.
```

Et puis quand il a envoyé l'exception:

```
07-20 08: 57: 47.052 D / FirebaseCrashApiImpl: Possibilité de lancer  
java.lang.Exception: Ma première erreur non fatale Android 07-20 08: 58: 18.822  
D / FirebaseCrashSenderServiceImpl: Code de réponse: 200 07-20 08: 58: 18.822 D  
/ FirebaseCrashSenderServiceImpl: Rapport envoyé
```

Vous pouvez ajouter des journaux personnalisés à votre rapport avec

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

Lire Rapport de collision en ligne: <https://riptutorial.com/fr/firebase/topic/4669/rapport-de-collision>

Chapitre 16: Règles de base de données

Introduction

Avec la base de données en temps réel Firebase, vos règles de base de données constituent la sécurité côté serveur. Vous devez faire très attention et savoir qui a accès à votre base de données. Il est important que personne ne puisse accéder à vos données.

Par défaut, les règles de base de données Firebase Realtime permettent à tout utilisateur authentifié de lire et d'écrire toutes les données, ce n'est probablement pas ce que vous souhaitez que votre application fasse.

Regardez les exemples ci-dessous pour différents scénarios.

Remarques

La base de données Firebase Realtime fournit un langage de règles flexible, basé sur des expressions, avec une syntaxe de type JavaScript pour définir facilement la structure de vos données, leur indexation et la lecture et l'écriture de vos données. Combiné à nos services d'authentification, vous pouvez définir qui a accès à quelles données et protéger les informations personnelles de vos utilisateurs contre les accès non autorisés.

Par défaut, vos règles de base de données nécessitent une authentification Firebase et n'accordent des autorisations de lecture et d'écriture complètes qu'aux utilisateurs authentifiés. Les règles par défaut garantissent que votre base de données n'est pas accessible par n'importe qui avant d'avoir la possibilité de configurer i

Documentation officielle

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

Exemples

Comment configurer les règles

1. Allez dans la console Firebase.
2. Choisissez votre projet
3. Cliquez sur la section Base de données à gauche, puis sélectionnez l'onglet Règles.

Si vous souhaitez tester vos règles de sécurité avant de les mettre en production, vous pouvez simuler des opérations dans la console à l'aide du bouton Simuler situé dans la partie supérieure droite de l'éditeur de règles.

Les règles par défaut

Les règles par défaut nécessitent une authentification.

Ils permettent un accès complet en lecture et en écriture aux utilisateurs authentifiés de votre application. Ils sont utiles si vous souhaitez que les données soient ouvertes à tous les utilisateurs de votre application, mais ne veulent pas qu'ils soient ouverts au monde.

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

Comment définir vos fichiers lisibles et accessibles en écriture

Il suffit de définir:

```
// 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
  }
}
```

Cela peut être utile pendant le développement, mais faites attention car ce niveau d'accès signifie que **tout le monde peut lire ou écrire dans votre base de données** .

Comment désactiver l'accès en lecture et en écriture

Vous pouvez définir des règles privées pour désactiver l'accès en lecture et en écriture à votre base de données par les utilisateurs. Avec ces règles, **vous pouvez uniquement accéder à la base de données lorsque vous disposez de privilèges administratifs (que vous pouvez obtenir en accédant à la base de données via la console Firebase ou en vous [connectant depuis un serveur](#))** .

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

Comment accorder l'accès uniquement aux utilisateurs authentifiés

Voici un exemple de règle qui attribue à chaque utilisateur authentifié un nœud personnel sur `/users/$user_id` où `$ user_id` est l'ID de l'utilisateur obtenu via **Authentication** .

```
// 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"
      }
    }
  }
}

```

Comment autoriser la lecture d'un élément spécifique d'un groupe, mais empêcher la liste des membres du groupe

Il est courant de créer des groupes d'éléments en créant des nœuds de valeur simples avec l'ID d'élément comme clé. Par exemple, nous pouvons ajouter un utilisateur au groupe "administrateurs" en créant un nœud à l'adresse `/administrators/$user_id` avec une valeur `true`. Nous ne voulons pas que quiconque sache qui sont les administrateurs, pour des raisons de sécurité, mais nous voulons toujours vérifier si un utilisateur authentifié est **administrateur**. Avec ces règles, nous pouvons faire exactement cela:

```

{
  "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()",
      }
    }
  }
}

```

Lire Règles de base de données en ligne: <https://riptutorial.com/fr/firebase/topic/3352/regles-de-base-de-donnees>

Chapitre 17: Structuration des données

Introduction

La base de données Firebase est une base de données NoSQL qui stocke ses données sous la forme d'objets JSON hiérarchiques. Il n'y a pas de tables ou d'enregistrements de forme quelconque, comme une base de données SQL aurait normalement, juste des noeuds qui constituent une structure clé-valeur.

Normalisation des données

Pour avoir une structure de base de données correctement conçue, les exigences en matière de données doivent être soigneusement définies et anticipées. La structure dans ce cas devrait être normalisée; plus l'arborescence JSON est plate, plus l'accès aux données est rapide.

Exemples

À faire et à ne pas faire

La mauvaise direction

Considérons la structure suivante

```
{
  "users": {

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

    "-KH3Cx0KFvSQELIYZezv": {
      "name": "Jon Snow",
      "aboutMe": "I know nothing...",
      "posts": {
        "post1": {
          "body": "Different roads sometimes leads to the same castle",
          "isHidden": false
        },
        "post2": { ... },
        // Possibly more posts
      }
    },
    "-KH3Dx2KFdSLerIYZcgk": { ... }, // Another user
    // A lot more users here
  }
}
```

C'est un excellent exemple de ce qu'il **ne** faut **pas** faire. Les structures multi-imbriquées, telles que celle ci-dessus, peuvent être très problématiques et entraîner un revers de performance considérable.

La manière dont Firebase accède à un nœud consiste à télécharger toutes les données des enfants, puis à les parcourir sur tous les nœuds de même niveau (tous les enfants des parents). Maintenant, imaginez une base de données avec plusieurs *utilisateurs*, chacun ayant des centaines (voire des milliers) de *messages*. L'accès à une *publication* dans ce cas pourrait potentiellement charger des centaines de mégaoctets de données inutilisées. Dans une application plus complexe, l'imbrication pourrait être plus profonde que 4 couches, ce qui entraînerait davantage de téléchargements et d'itérations inutiles.

Le droit chemin

Aplatir la même structure ressemblerait à ceci

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

Cela évite une énorme surcharge en itérant sur moins de nœuds pour accéder à un objet cible. Tous les *utilisateurs* qui n'ont pas de messages n'existeraient pas dans la branche des *publications* et, par conséquent, l'itération sur ces utilisateurs dans *le mauvais sens* ci-dessus est totalement inutile.

Relations bidirectionnelles

Voici un exemple de base de données collégiale simple et minimale qui utilise des relations bidirectionnelles

```
{
  "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,
  }
},
"-SL3ns2OtARSTUMyqwWt": { ... },
// 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
      "-SL3ns2OtARSTUMyqwWt": true,      // Some other student
      // More students
    }
  },
},
// More courses
}
}

```

Notez que chaque élève a une liste de *cours* et chaque cours a une liste d' *étudiants* inscrits.

La redondance n'est pas toujours une mauvaise approche. Il est vrai que cela coûte de l'espace de stockage et qu'il faut gérer la mise à jour de plusieurs entrées lors de la suppression ou de la modification d'un nœud dupliqué. Cependant, dans certains scénarios où les données ne sont pas souvent mises à jour, le fait d'avoir des relations bidirectionnelles pourrait faciliter considérablement le processus d'extraction / écriture.

Dans la plupart des scénarios où une requête SQL semble nécessaire, inverser les données et créer des relations bidirectionnelles est généralement la solution.

Considérez une application utilisant la base de données ci-dessus qui nécessite la possibilité de:

1. Énumérez les cours suivis par un élève **et** ...
2. Liste tous les étudiants dans un certain cours

Si la structure de la base de données avait été unidirectionnelle, il serait incroyablement plus lent d'analyser ou d'interroger l'une des deux exigences ci-dessus. Dans certains scénarios, la

redondance rend les opérations fréquentes plus rapides et beaucoup plus efficaces, ce qui, à long terme, rend les doublons négligeables.

Lire Structuration des données en ligne: <https://riptutorial.com/fr/firebase/topic/8912/structuration-des-donnees>

Chapitre 18: Utiliser Firebase avec Node

Exemples

Hello World Firebase Base de données en temps réel dans le nœud

Configuration requise:

- [Nœud JS](#)

Commencer

1. Allez d'abord dans la [console Firebase](#) et créez un nouveau projet.
2. Après avoir créé le projet, dans le projet, cliquez sur l'icône Paramètres en plus du nom du projet dans la barre latérale gauche et sélectionnez Autorisations.
3. Sur la page des autorisations Cliquez sur les comptes de service dans la barre latérale gauche, puis cliquez sur Créer un compte de service
4. Dans la fenêtre contextuelle, entrez le nom de votre compte de service et choisissez Rôle du compte, puis sélectionnez Fournir une nouvelle clé privée puis sélectionnez JSON et cliquez sur Créer (Laisser Activer la délégation à l'échelle du domaine Google App décochée).
5. Lorsque vous cliquez sur Créer, vous téléchargez un fichier JSON avec vos informations d'identification de compte, enregistrez simplement le fichier Anywhere dans votre système.
6. L'étape suivante consiste à créer une base de données dans votre console Firebase pour laquelle accéder à la console Firebase et à cliquer sur Database dans la barre latérale gauche. Ensuite, créez simplement un nouvel objet de base de données avec le nom `user_data` avec une valeur factice.
7. Maintenant, votre projet de base de données Firebase est configuré, copiez simplement le code suivant dans votre répertoire de projet.

```
//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. Changez simplement avec l'URL du fichier d'informations d'identification JSON (pour commencer, copiez simplement le fichier d'informations d'identification dans le même dossier et dans le fichier index.js, ajoutez simplement le nom du fichier).
9. La prochaine étape consiste à modifier le fichier index.js avec l'URL de la base de données Firebase, vous pourrez trouver cette URL dans Firebase Console dans l'onglet Base de données. L'URL sera similaire à *https://.firebaseio.com/*.
10. La dernière étape consiste à faire

```
npm install firebase
```

11. Après l'exécution de la commande ci-dessus, NPM installe les paquets nécessaires à Firebase. Enfin pour exécuter et tester le projet exécuter

```
node index.js
```

Que fait réellement le projet?

Le projet charge les données de la base de données Firebase basée sur le cloud. Le projet montre également comment écrire et lire des données à partir d'un objet de données Firebase.

Afin de voir vos données mises à jour en temps réel, allez sur [votre console](#), cliquez sur le projet que vous avez réalisé, et sur la gauche, cliquez sur Base de données. Vous pouvez y voir vos données mises à jour en temps réel, avec leurs valeurs.

Firestore-queue et worker

Vous pouvez envoyer des tâches ou des données à la base de données temps réel de Firebase et exécuter un agent qui écoute la file d'attente de la base de données pour exécuter des processus en arrière-plan.

Configuration de la base de feu

1. Créez un projet Firebase dans la console Firebase, si vous n'en avez pas déjà un. Si un projet Google existant est déjà associé à votre application, cliquez sur Importer un projet Google. Sinon, cliquez sur Créer un nouveau projet.
2. Cliquez sur l'icône Paramètres et sélectionnez Autorisations.
3. Sélectionnez Comptes de service dans le menu de gauche.
4. Cliquez sur Créer un compte de service.

Entrez un nom pour votre compte de service.

Vous pouvez éventuellement personnaliser l'ID de celui généré automatiquement à partir du nom.

Choisissez Projet> Editeur dans la liste déroulante Rôle.

Sélectionnez Fournir une nouvelle clé privée et laissez le type de clé JSON.

Ne pas activer la délégation de domaine à l'échelle du domaine Google Apps.

Cliquez sur Créer

5. Lorsque vous créez le compte de service, un fichier JSON contenant les informations d'identification de votre compte de service est téléchargé pour vous. Vous en aurez besoin pour initialiser le SDK sur le serveur.

Serveur d'installation

Installez Firestore-queue en utilisant npm dans votre application nodejs

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

Une fois que vous avez installé firebase et firebase-queue, vous pouvez commencer par créer une nouvelle file d'attente et lui transmettre votre référence Firestore et une fonction de traitement.

Maintenant, créons une tâche de file d'attente de base Firebase à partir de l'application lorsqu'un nouvel utilisateur est créé et définissez worker pour qu'il écoute la tâche de file d'attente de base de données et envoie un courrier électronique aux utilisateurs créés.

* server.js

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

Mettez à jour vos références de projet Firebase et votre URL de base de données Firebase

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

ou vous pouvez entrer les informations d'identification de la base de feu directement comme ci-dessous

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

exécuter le serveur et le serveur séparément et tester avec la file d'attente de la base de données

```
node server.js
node worker.js
```

Lire Utiliser Firebase avec Node en ligne: <https://riptutorial.com/fr/firebase/topic/6443/utiliser-firebase-avec-node>

Chapitre 19: Vérification de l'e-mail après inscription

Syntaxe

- Envoyer un courrier électronique de vérification à l'adresse e-mail de l'utilisateur connecté dans le fichier. Firebase vous permet de [personnaliser le contenu de votre courrier électronique](#)
- Lorsque le courrier électronique atteint le compte de messagerie de l'utilisateur, l'utilisateur clique sur
- En utilisant le routeur de votre choix (routeur angulaire utilisé dans l'exemple ci-dessus), interceptez les paramètres dans l'URL.
- Mâchez les paramètres en utilisant la fonction `applyCode` dans Firebase.
- Voir ci-dessous pour les fonctions impliquées dans le processus ci-dessus.

Paramètres

La fonction...	Est-ce que
<code>sendEmailVerification ()</code>	Envoie un email de vérification à un utilisateur.
<code>applyActionCode ()</code>	Applique le code d'action qui modifie l' <code>emailVerified</code> de <code>false</code> à <code>true</code>

Remarques

Ce qui précède résume à peu près comment utiliser le système de vérification des e-mails avec Firebase. Jusqu'à présent, c'est l'un des moyens les plus simples de vérifier le courrier électronique que j'ai vu.

Il existe une explication plus détaillée de l'exemple ci-dessus disponible sur [Email Verification with Firebase 3.0 SDK](#).

Exemples

Code d'action de vérification d'envoi et de traitement - 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 });
      })
  };
}
])

```

Lire Vérification de l'e-mail après inscription en ligne:

<https://riptutorial.com/fr/firebase/topic/3380/verification-de-l-e-mail-apres-inscription>

Crédits

S. No	Chapitres	Contributeurs
1	Démarrer avec Firebase	Ami Hollander , Community , Dan Levy , Devid Farinelli , ErstwhileIII , Gabriele Mariotti , RyanM , Sneh Pandya , TwiterZX , Vishal Vishwakarma
2	Capacités hors ligne de Firebase	Francisco Durdin Garcia
3	Comment écouter les erreurs lors de l'accès à la base de données?	Frank van Puffelen , ThunderStruct
4	Comment obtenir la valeur de la clé push de la base de données Firebase?	cutiko
5	Comment utiliser FirebaseRecyclerAdapter au lieu de RecyclerView?	Dhaval Solanki
6	Comment utiliser la base de données Firebase pour conserver une liste d'utilisateurs d'authentification Firebase	Devid Farinelli , eikooc , Frank van Puffelen , Ron Royston
7	Console Firebase	Priyank Bhojak
8	Espace de rangement	Mike McDonald
9	File d'attente Firebase	Vladimir Gabrielyan
10	Firebase Realtime Database avec Android	Dhaval Solanki , Frank van Puffelen
11	FirebaseUI	Mike McDonald
12	FirebaseUI (Android)	Willie Chalmers III
13	Fonctions Cloud pour	Vishal Vishwakarma

	Firestore	
14	Notification push du serveur personnalisé	Aawaz Gyawali
15	Rapport de collision	Gabriele Mariotti
16	Règles de base de données	Frank van Puffelen , Gabriele Mariotti , riggaroo , Sasxa , Velko Ivanov
17	Structuration des données	ThunderStruct
18	Utiliser Firestore avec Node	Akshay Khale , Laurel , Noushad PP , Shiven
19	Vérification de l'e-mail après inscription	Rexford