



THE UNIVERSITY OF
CHICAGO



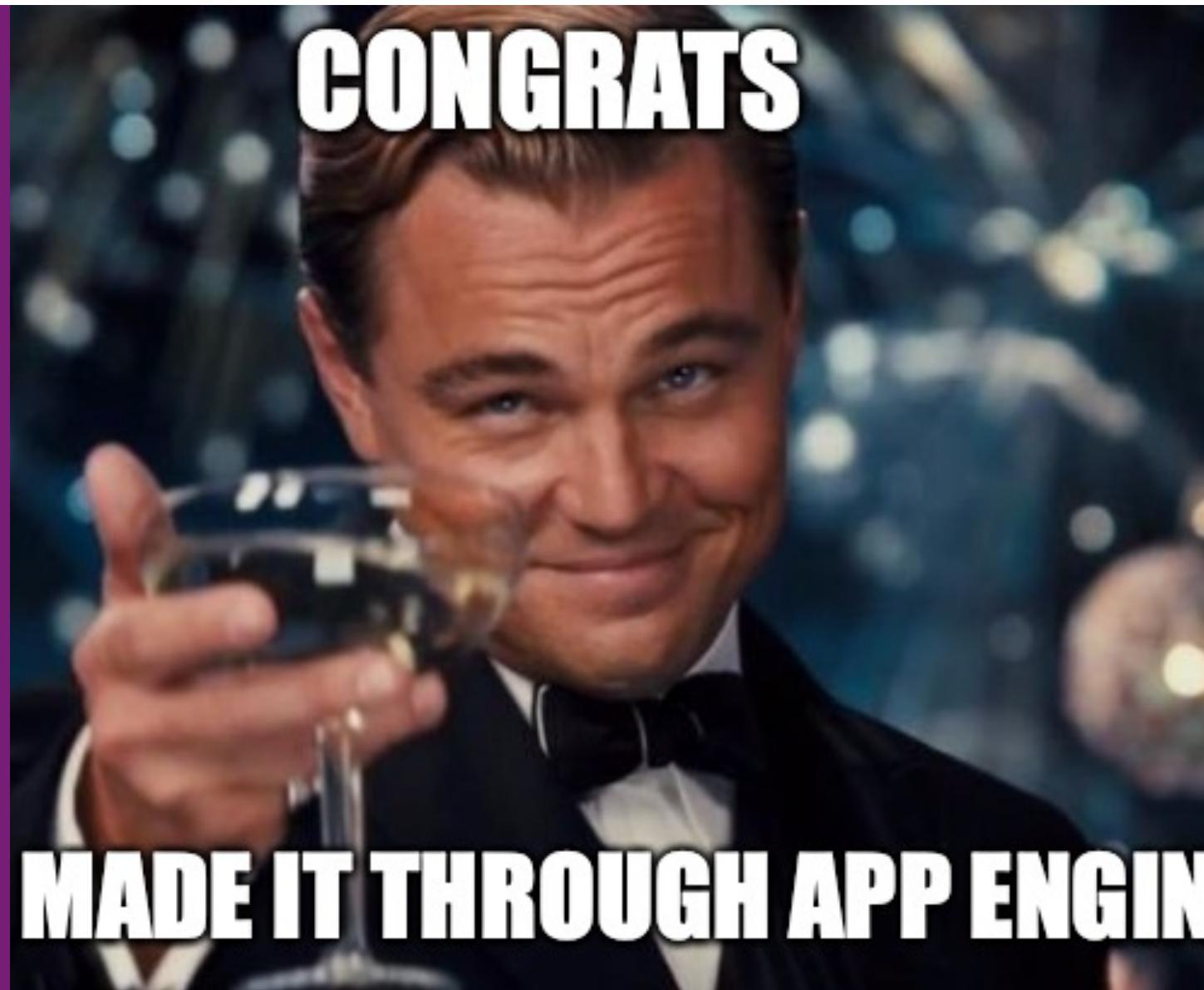
MPCS 51033 • AUTUMN 2019 • SESSION 4

BACKENDS FOR MOBILE APPLICATIONS

CLASS NEWS

CLASS NEWS

- Homework



FIREBASE



FIREBASE

- https://youtu.be/O17OWyx08Cg?list=PLI-K7zZEsYLmOF_07layrTntevxtbUxDL



FIREBASE

The diagram illustrates the Firebase ecosystem. On the left, a blue cylinder icon represents the database, connected by a line to a smartphone. The smartphone screen shows a login interface with a green padlock icon, the username 'Firecat83', and a password field with five asterisks. Below the screen is a virtual keyboard. In the center, a globe icon is surrounded by numerous user profile icons, with a callout bubble showing '25K'. To the right, a screenshot of a YouTube channel page is shown. The channel title is 'Introducing Firebase' with '17/17 videos'. The first video thumbnail is 'Introducing Firebase Authentication' by Firebase. The second is 'Introducing Firebase Cloud Messaging' by Firebase. The third is 'Introducing Firebase Realtime Database' by Firebase. The fourth is a yellow thumbnail for 'Introducing Firebase' by Firebase.

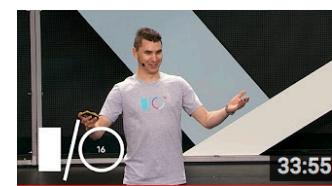
Introducing Firebase



Firebase

Subscribed 56,536

287,389 views



Zero to App: Develop with
Firebase - Google I/O 2016

Firebase
119,707 views
33:55

FIREBASE

- Firebase pricing structure

Products	Spark Plan Generous limits for hobbyists Free	Flame Plan Fixed pricing for growing apps \$25/month	Blaze Plan <u>Calculate pricing for apps at scale</u> Pay as you go <small>✓ Free usage from Spark plan included*</small>
A/B Testing		Free	
Analytics		Free	
App Indexing		Free	
Authentication			
Phone Auth - US, Canada, and India <small>?</small>	10k/month	10k/month	\$0.01/verification
Phone Auth - All other countries <small>?</small>	10k/month	10k/month	\$0.06/verification
Other Authentication services	✓	✓	✓
Cloud Firestore			
Stored data	1 GiB total	2.5 GiB total	\$0.18/GiB
Network egress	10GiB/month	20GiB/month	Google Cloud pricing
Document writes	20K/day	100K/day	\$0.18/100K
Document reads	50K/day	250K/day	\$0.06/100K
Document deletes	20K/day	100K/day	\$0.02/100K

FIREBASE

REDWHALE STARTUPS VENTURE CAPITAL TECHNOLOGY GROWTH ABOUT & EDITORS DESK SUB

Firebase
Don't Get Service Trapped!

Firebase Costs Increased by 7,000%

HomeAutomation [Follow] May 17, 2017 - 10 min read

Update (05/17/2017 8:50 AM):

I am happy to say that I received a phone call promptly this morning from the executive team at Firebase. They professed their sincere apology for the situation and explained what happened specifically with the account.

While we haven't reached a final conclusion on everything, it would appear the specifics of the article have been heard. Many of the complaints (namely the metrics, among other things) were outlined as things they would put a focus on to improve upon quickly.

I will make sure to keep it updated as any relevant information is provided.

Update (05/17/2017 1:00 PM):

Upon further discussions, credits have been taken care of and a plan has been set to address the concerns aggressively. Andrew (Founder of Firebase) and his



Image by Aphinya Dechaler, Titled: Read/Write Monster Feasting on Firebase Bill

How NOT to get a \$30k bill from Firebase

the secret is in the architecture

Aphinya Dechaler [Follow] May 27 - 5 min read *

July last year, a crowd funding campaign went viral in Colombia. It was all good and dandy in the first 48 hours. They managed to reach over 2 million sessions and over 20 million page views—with a website that stayed fully functional without a hitch.

Until they saw the bill.

HACKERNOON AI Crypto Coding Fortune Startups About Community Sponsors

Be seen in a new tech job.

How we spent 30k USD in Firebase in less than 72 hours

Nicolas Contreras V @n1500 October 14th, 2019

TWEET THIS

avacaPorDeLaCalle became the largest crowdfunding campaign in Colombia collecting 3 times more than the previous record so far in only two days! It became one of the biggest political crowdfunding campaign in history.

Crowdfunding, alternativa hasta para la financiación electoral

En Colombia existen 13 plataformas de financiación colectiva online Francisco Rincón - frincon@larepublica.com.co... www.larepublica.co

IG SUCCESS FOR VAKI

48 hours after the campaign was released, we had reached many records. The campaign collected 3 times more than the previous record in Colombia at time. We had reached more than 2 million sessions, more than 20 million pages visited and received more than 15 thousand supports. This averages thousand users active on the site in average and collecting more than 20 supports per minute.

Firebase Realtime Database

TL;DR

Last year at GreenLionSoft we decided to progressively ditch various of the goodies provided by AWS and go "all in" into the Google's Firebase suite.

In this article I want you to learn some critical tips to avoid the errors we committed while integrating Firebase Realtime Database that resulted in a unexpected monthly bill of more than 1.000€.

...

FIREBASE IN ACTION

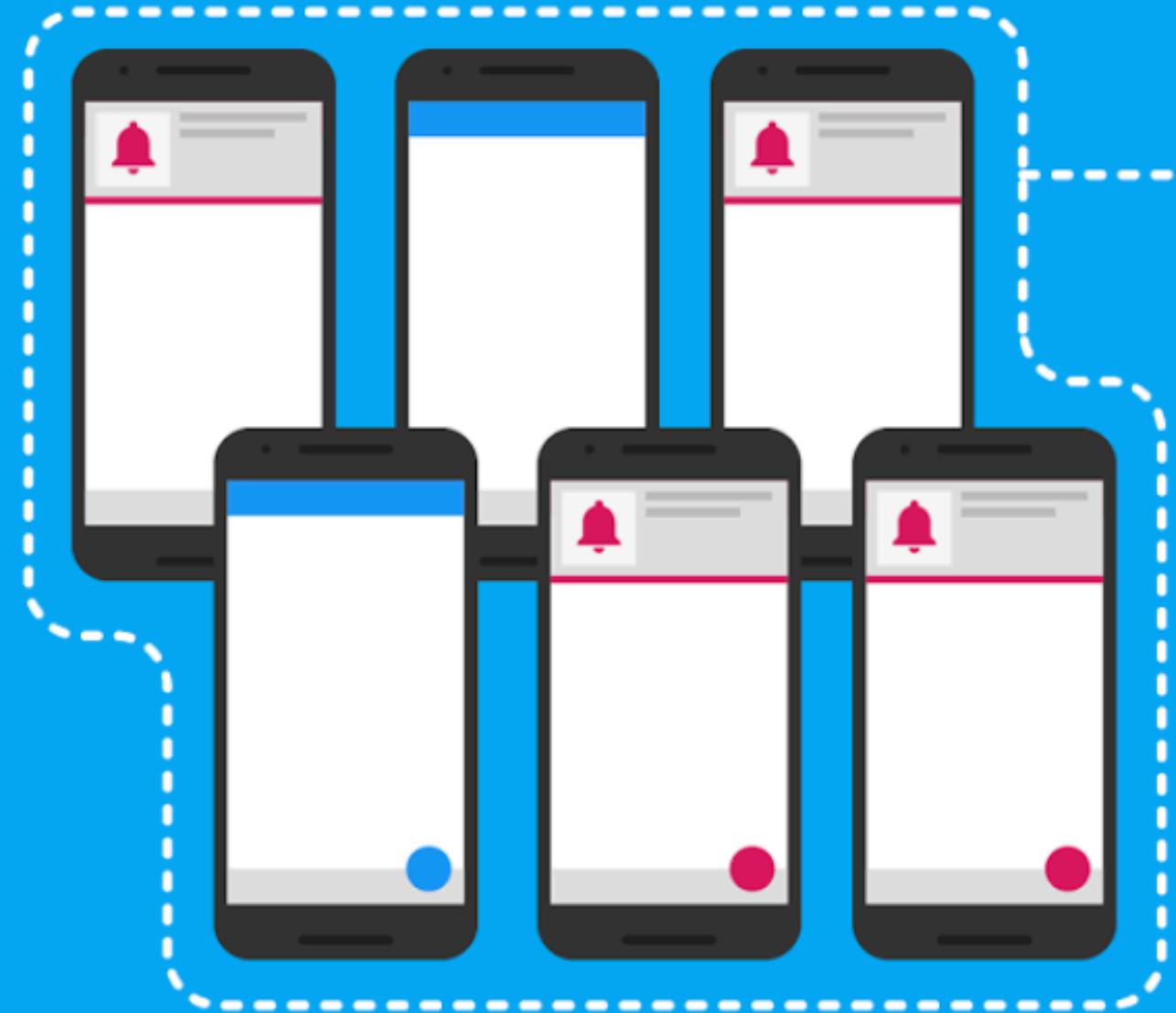
PREREQUISITES FOR FIREBASE

- Before you begin, you need a few things set up in your environment:
 - Xcode 7.0 or later
 - An Xcode project targeting iOS 7 or above
 - The bundle identifier of your app
 - CocoaPods 1.0.0 or later

FIREBASE IN ACTION

PREREQUISITES FOR FIREBASE

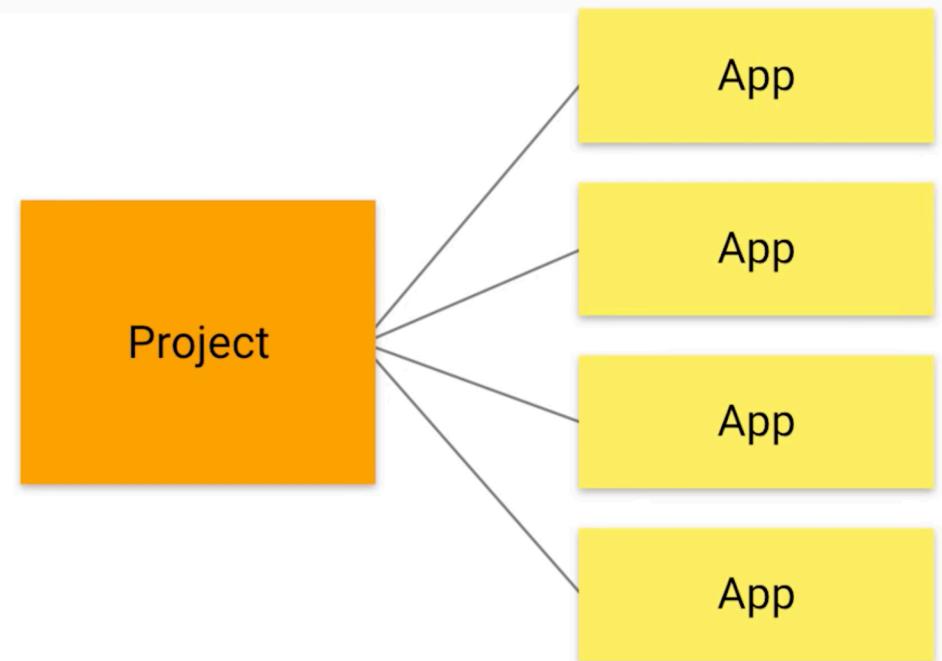
- For Cloud Messaging
 - A physical iOS device
 - APNs certificate with Push Notifications enabled
 - In Xcode, enable Push Notifications in App > Capabilities



FIREBASE IN ACTION

PROJECTS VS. APPS

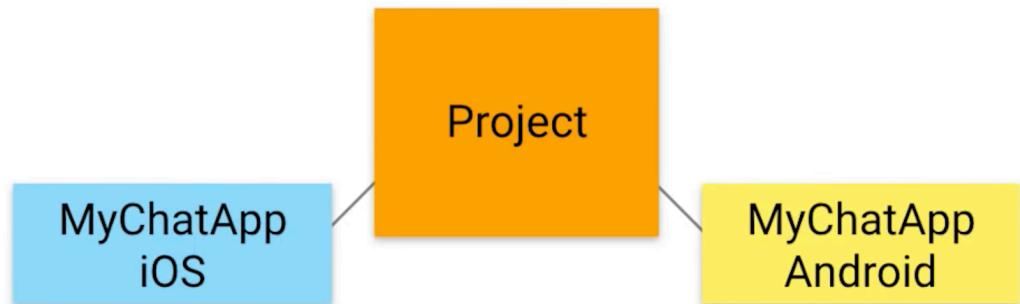
- Project consists of 1+ apps
- Apps all share the same backend



FIREBASE IN ACTION

PROJECTS VS. APPS

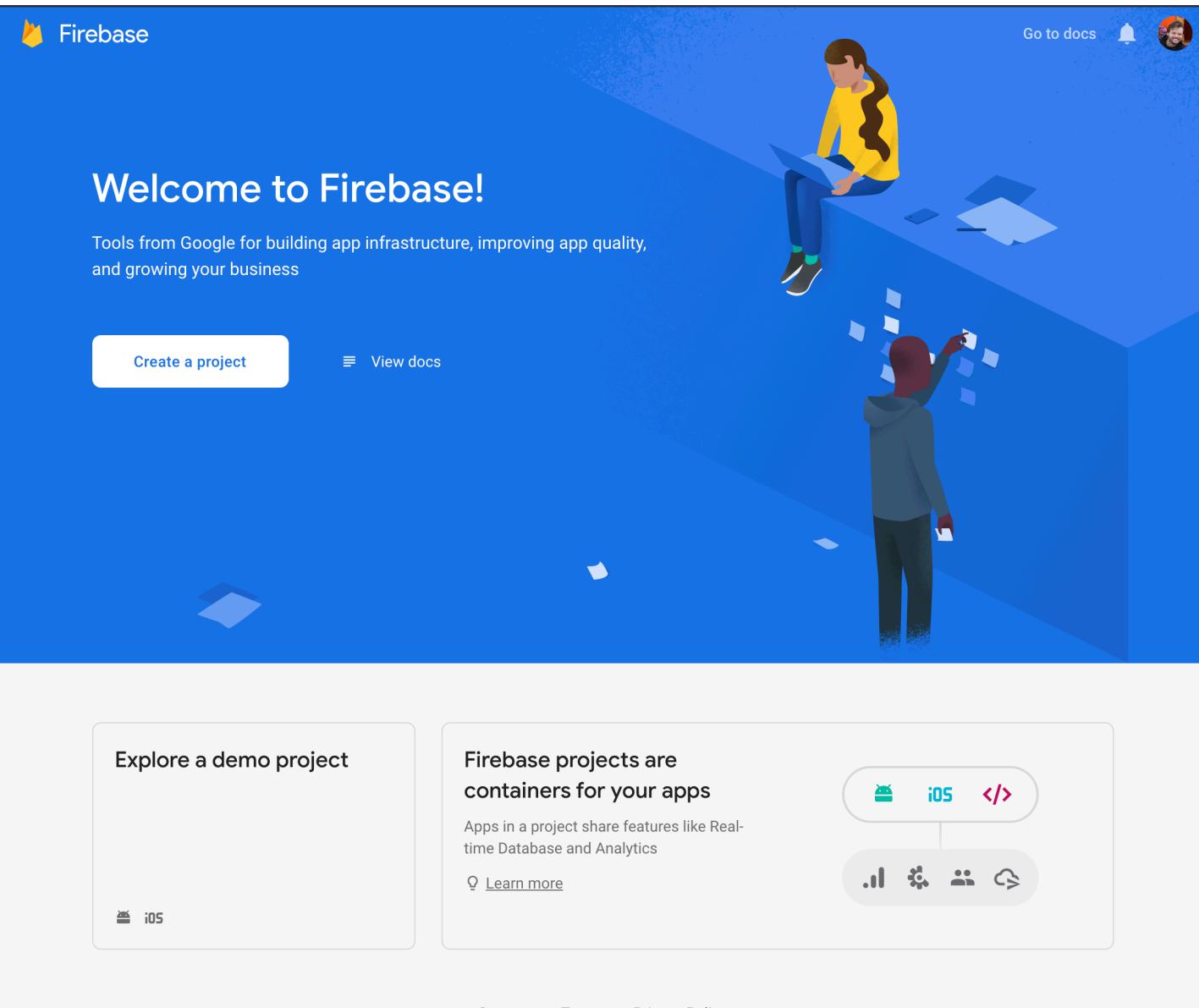
- Cross platform app
 - iOS and Android would be different apps in the same project
 - Access same data from different platforms
 - Send messages to all apps



FIREBASE IN ACTION

PROJECTS VS. APPS

- Create a Firebase project



The image shows the official Firebase website homepage. At the top right, there are links for "Go to docs" and a user profile icon. The main title "Welcome to Firebase!" is prominently displayed, followed by a subtitle: "Tools from Google for building app infrastructure, improving app quality, and growing your business". Below this are two buttons: "Create a project" and "View docs". The background features a blue gradient with abstract white shapes and two people: one sitting and working on a laptop, and another standing and interacting with floating square icons.

Welcome to Firebase!

Tools from Google for building app infrastructure, improving app quality, and growing your business

Create a project View docs

Explore a demo project

iOS

Firebase projects are containers for your apps

Apps in a project share features like Real-time Database and Analytics

Learn more

Android iOS </>

Wi-Fi Location People Cloud

Overview Terms Privacy Policy

FIREBASE IN ACTION

PROJECTS VS. APPS

- Create a Firebase project
 - Link to a GCP project

×

Create a project (Step 1 of 3)

**Let's start with a name for
your project**

Enter your project name

Add Firebase to one of your existing Google Cloud Platform projects

[Learn more](#)

-  photo-timeline-python3
-  session2
-  app-engine-quickstart
-  perfectappengine
-  mpcs51033-2017-autumn-photos
-  MyFirstProject

FIREBASE IN ACTION

PROJECTS VS. APPS

- Analytics

× Create a project (Step 2 of 3)

Google Analytics for your Firebase project

Google Analytics is a free and unlimited analytics solution that enables targeting, reporting, and more in Firebase Crashlytics, Cloud Messaging, In-App Messaging, Remote Config, A/B Testing, Predictions, and Cloud Functions.

Google Analytics enables:

-  A/B testing [?](#)
 User segmentation & targeting across Firebase products [?](#)
 Predicting user behavior [?](#)

-  Crash-free users [?](#)
 Event-based Cloud Functions triggers [?](#)
 Free unlimited reporting [?](#)

-  Enable Google Analytics for this project
Recommended

[Previous](#)

[Continue](#)

FIREBASE IN ACTION

PROJECTS VS. APPS

- Analytics

×

Create a project (Step 3 of 3)

Configure Google Analytics

Choose or create a Google Analytics account [?](#)

Select an account [▼](#)

Upon project creation, a new Google Analytics property will be created in your chosen Google Analytics account and linked to your Firebase project. This link will enable data flow between the products. Data exported from your Google Analytics property into Firebase is subject to the Firebase terms of service, while Firebase data imported into Google Analytics is subject to the Google Analytics terms of service. [Learn more](#).

[Previous](#)

[Create project](#)

FIREBASE IN ACTION

PROJECTS VS. APPS

● Analytics

X Create a project (Step 3 of 3)

Configure Google Analytics

Choose or create a Google Analytics account [?](#)

chat-cat

Analytics location [?](#)

United States

Data sharing settings and Google Analytics terms

- Use the default settings for sharing Google Analytics data. [Learn more](#)
 - ✓ Share your Analytics data with Google to improve Google Products and Services
 - ✓ Share your Analytics data with Google to enable Benchmarking
 - ✓ Share your Analytics data with Google to enable Technical Support
 - ✓ Share your Analytics data with Google Account Specialists
- I accept the [Measurement Controller-Controller Data Protection terms](#) and acknowledge I am subject to the [EU End User Consent Policy](#). This is required when sharing Google Analytics data to improve Google Products and Services. [Learn more](#)
- I accept the [Google Analytics terms](#)

Upon project creation, a new Google Analytics property will be created and linked to your Firebase project. This link will enable data flow between the products. Data exported from your Google Analytics property into Firebase is subject to the Firebase terms of service, while Firebase data imported into Google Analytics is subject to the Google Analytics terms of service. [Learn more](#).

[Previous](#)

[Create project](#)

FIREBASE IN ACTION

PROJECTS VS. APPS

- Console

The screenshot shows the Firebase console interface for the 'chat-cat' project. On the left, a sidebar lists 'Project Overview', 'Develop' (Authentication, Database, Storage, ...), 'Quality' (Crashlytics, Performance, Test Lab, ...), 'Analytics' (Dashboard, Events, Conversions, Au...), and 'Grow' (Predictions, A/B Testing, Cloud Mes...). A yellow callout box points to the 'iOS' icon in the 'Develop' section. The main area displays the project name 'chat-cat' and 'Spark plan'. It features a central message: 'Get started by adding Firebase to your app' with icons for iOS, Android, web, and Cloud Functions. Below this is a button 'Add an app to get started'. To the right, there's an illustration of two people interacting with a large smartphone screen. At the bottom, there are two cards: 'Authentication' (Authenticate and manage users) and 'Cloud Firestore' (Realtime updates, powerful queries, and automatic scaling). A magnifying glass icon highlights the 'Cloud Firestore' card.

FIREBASE IN ACTION

REGISTER APP

- Add Firebase to your app
- Bundle id needs to match you app bundle id
- App Store ID will be generated when you submit
 - Invites, Dynamic Links

x Add Firebase to your iOS app

1 Register app

iOS bundle ID ?

App nickname (optional) ?

App Store ID (optional) ?

[Register app](#)

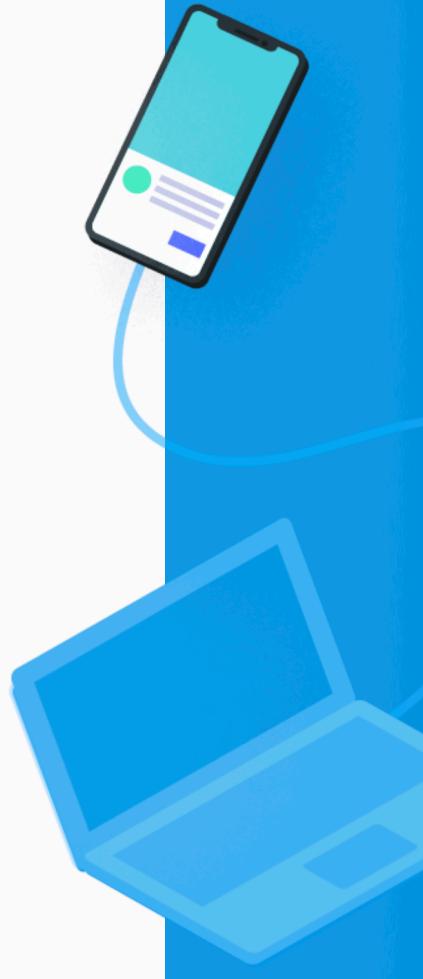
2 Download config file

3 Add Firebase SDK

4 Add initialization code

5 Run your app to verify installation

[Go to docs](#)



FIREBASE IN ACTION

CONFIG FILE

- Generates a .plist to include in your project

Add Firebase to your iOS app

1 Register app

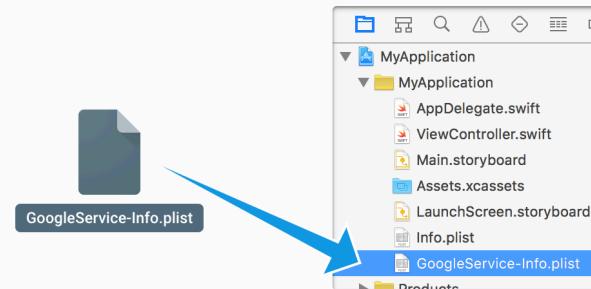
iOS bundle ID: mobi.uchicago.chat-cat, App nickname: Chat Cat

2 Download config file

[Download GoogleService-Info.plist](#)

Instructions for Xcode below | [C++](#)

Move the GoogleService-Info.plist file you just downloaded into the root of your Xcode project and add it to all targets.



Previous

Next

3 Add Firebase SDK

4 Add initialization code

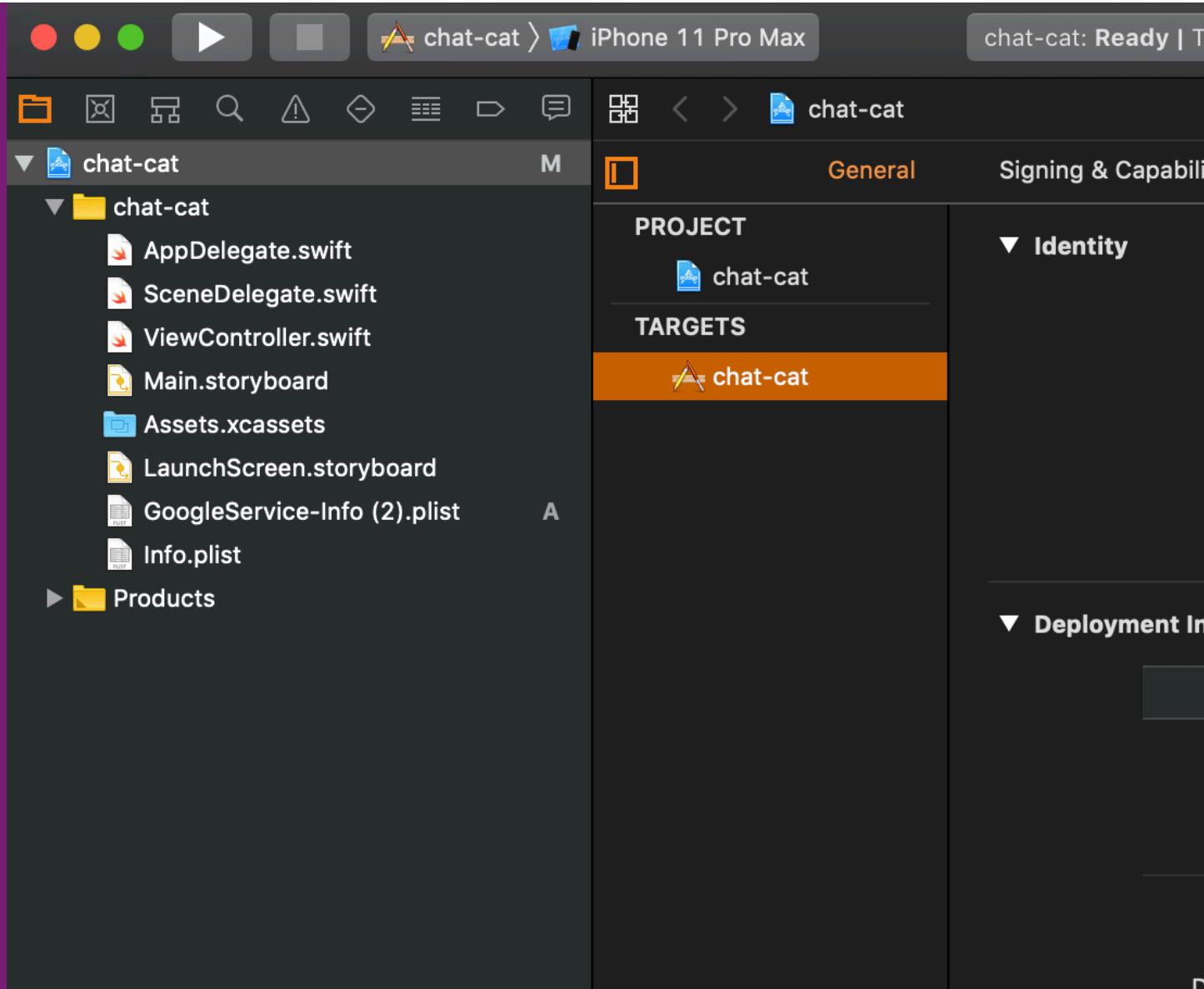
5 Run your app to verify installation

[Go to docs](#)

FIREBASE IN ACTION

CONFIG FILE

- Put in your project



FIREBASE IN ACTION

CONFIG FILE

Key	Type	Value
▼ Information Property List	Dictionary	(17 items)
AD_UNIT_ID_FOR_BANNER_TEST	String	ca-app-pub-3940256099942544/2934735716
AD_UNIT_ID_FOR_INTERSTITIAL_T...	String	ca-app-pub-3940256099942544/4411468910
CLIENT_ID	String	487781121319-ms6r6t0ninob0i8ri4mord3kd5roto9v.apps.googleusercontent.com
REVERSED_CLIENT_ID	String	com.googleusercontent.apps.487781121319-ms6r6t0ninob0i8ri4mord3kd5roto9v
API_KEY	String	AlzaSyDjXHxBu32Fwmh9urpLpB-B33av8hUPoNA
GCM_SENDER_ID	String	487781121319
PLIST_VERSION	String	1
BUNDLE_ID	String	mobi.uchicago.firechat
PROJECT_ID	String	firechat-66f87
STORAGE_BUCKET	String	firechat-66f87.appspot.com
IS_ADS_ENABLED	Boolean	YES
IS_ANALYTICS_ENABLED	Boolean	NO
IS_APPINVITE_ENABLED	Boolean	YES
IS_GCM_ENABLED	Boolean	YES
IS_SIGNIN_ENABLED	Boolean	YES
GOOGLE_APP_ID	String	1:487781121319:ios:78425b1380795a79
DATABASE_URL	String	https://firechat-66f87.firebaseio.com

FIREBASE IN ACTION

ADD SDK

- Firebase uses cocoapods for SDK distribution
- Changes xcodeproject to xcworkspace

(COCOAPODS)

APP ABOUT GUIDES BLOG

SEARCH*



* Type here to search by name, version, author, keywords, summary, and dependencies.

WHAT IS COCOAPODS

CocoaPods is a dependency manager for Swift and Objective-C Cocoa projects. It has over 30 thousand libraries and is used in over 2 million apps. CocoaPods can help you scale your projects elegantly.

INSTALL

GET STARTED

CREATE A POD

FIREBASE IN ACTION

ADD SDK

[HTTPS://
WWW.YOUTUBE.CO
M/WATCH?
V=IEAJVNRDZAO](https://www.youtube.com/watch?v=IEAJVNRDZAO)



FIREBASE IN ACTION

ADD SDK

- Pod install
Firebase SDK

✗ Add Firebase to your iOS app

✓ Register app

iOS bundle ID: mobi.uchicago.chat-cat, App nickname: Chat Cat

✓ Download config file

3 Add Firebase SDK

Instructions for CocoaPods | [Download ZIP](#) [C++](#)

Google services use [CocoaPods](#) to install and manage dependencies. Open a terminal window and navigate to the location of the Xcode project for your app.

Create a Podfile if you don't have one:

```
$ pod init
```



Open your Podfile and add:

```
# add the Firebase pod for Google Analytics
pod 'Firebase/Analytics'
# add pods for any other desired Firebase products
# https://firebase.google.com/docs/ios/setup#available-pods
```



Save the file and run:

```
$ pod install
```



This creates an .xcworkspace file for your app. Use this file for all future development on your application.

[Previous](#)

[Next](#)

4 Add initialization code

[Go to docs](#)

FIREBASE IN ACTION

ADD SDK

- Pod install
Firebase SDK

```
firechat-2019 — emacs Podfile — 80x24

# Uncomment the next line to define a global platform for your
# platform :ios, '9.0'

target 'chat-cat' do
  # Comment the next line if you don't want to use dynamic frameworks!
  use_frameworks!

  # Pods for chat-cat

  # add the Firebase pod for Google Analytics
  pod 'Firebase/Analytics'

  # add pods for any other desired Firebase products
  # https://firebase.google.com/docs/ios/setup#available-pods
end
```

-uu-:**-F1 **Podfile** All L15 (Fundamental)-----

FIREBASE IN ACTION

ADD SDK

- Firebase pods
- There are multiple Firebase pods
- Only use what you need

Available pods

The following pods are available for the various Firebase products.

- ★ You no longer need to add the iOS pod `Firebase/Core`. This SDK included the Firebase SDK for Cloud Analytics (or any of the Firebase products that require or recommend the use of Analytics), you now only need to add the pod: `Firebase/Analytics`.

Firebase product	Pods
AdMob	<code>pod 'Firebase/AdMob'</code> <i>(required) pod 'Firebase/Analytics'</i>
Analytics	<code>pod 'Firebase/Analytics'</code>
Authentication	<code>pod 'Firebase/Auth'</code>
Cloud Firestore	<code>pod 'Firebase/Firestore'</code>
Cloud Functions for Firebase Client SDK	<code>pod 'Firebase/Functions'</code>
Cloud Messaging	<code>pod 'Firebase/Messaging'</code> <i>(recommended) pod 'Firebase/Analytics'</i>
Cloud Storage	<code>pod 'Firebase/Storage'</code>

FIREBASE IN ACTION

ADD SDK

- Install the Firebase pods in your project

✗ Add Firebase to your iOS app

✓ Register app

iOS bundle ID: mobi.uchicago.chat-cat, App nickname: Chat Cat

✓ Download config file

3 Add Firebase SDK

Instructions for CocoaPods | [Download ZIP](#) [C++](#)

Google services use [CocoaPods](#) to install and manage dependencies. Open a terminal window and navigate to the location of the Xcode project for your app.

Create a Podfile if you don't have one:

```
$ pod init
```



Open your Podfile and add:

```
# add the Firebase pod for Google Analytics
pod 'Firebase/Analytics'
# add pods for any other desired Firebase products
# https://firebase.google.com/docs/ios/setup#available-pods
```



Save the file and run:

```
$ pod install
```



This creates an .xcworkspace file for your app. Use this file for all future development on your application.

[Previous](#)

[Next](#)

4 Add initialization code

[Go to docs](#)



FIREBASE IN ACTION

ADD SDK

```
Installing FirebaseAuthInterop (1.0.0)
Installing FirebaseCore (6.3.0)
Installing FirebaseCoreDiagnostics (1.1.0)
Installing FirebaseCoreDiagnosticsInterop (1.0.0)
Installing FirebaseDatabase (6.1.0)
Installing FirebaseInstanceID (4.2.5)
Installing GoogleAppMeasurement (6.1.2)
Installing GoogleDataTransport (2.0.0)
Installing GoogleDataTransportCCTSupport (1.1.0)
Installing GoogleUtilities (6.3.1)
Installing leveldb-library (1.22)
Installing nanopb (0.3.901)
Generating Pods project
Integrating client project
```

WORK IN WORKSPACE

```
[!] Please close any current Xcode sessions and use `chat-cat.xcworkspace` for this project from now on.
Pod installation complete! There are 2 dependencies from the Podfile and 14 total pods installed.
```

```
[!] Automatically assigning platform `iOS` with version `13.1` on target `chat-cat` because no platform was
specified. Please specify a platform for this target in your Podfile. See `https://guides.cocoapods.org/synt
ax/podfile.html#platform`.
```

```
tabinkowski:firebase/chat-cat (master)
```

```
530 %
```

FIREBASE IN ACTION

ADD SDK

- Add initialization code to AppDelegate
 - Initializes services
 - Reads the plist and sets global variables

✗ Add Firebase to your iOS app

✓ Register app

iOS bundle ID: mobi.uchicago.chat-cat, App nickname: Chat Cat

✓ Download config file

✓ Add Firebase SDK

4 Add initialization code

To connect Firebase when your app starts up, add the initialization code below to your main `AppDelegate` class.

Swift Objective-C

```
import UIKit
import Firebase

@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate {

    var window: UIWindow?

    func application(_ application: UIApplication,
                    didFinishLaunchingWithOptions launchOptions:
                        [UIApplicationLaunchOptionsKey: Any]?) -> Bool {
        FirebaseApp.configure()
        return true
    }
}
```

Previous

Next

[Go to docs](#)

FIREBASE IN ACTION

CONSOLE

× Add Firebase to your iOS app

- 1 Register app
iOS bundle ID: mobi.uchicago.chat-cat, App nickname: Chat Cat
- 2 Download config file
- 3 Add Firebase SDK
- 4 Add initialization code
- 5 Run your app to verify installation

Checking if the app has communicated with our servers. You may need to uninstall and reinstall your app.



FIREBASE IN ACTION

BUILD THE APP

```
chat-cat
```

2019-10-21 22:21:31.570916-0500 chat-cat[40730:16737715] - <AppMeasurement>[I-ACS036002] Analytics screen reporting is enabled. Call +[FIRAnalytics setScreenName:setScreenClass:] to set the screen name or override the default screen class name. To disable screen reporting, set the flag FirebaseScreenReportingEnabled to NO (boolean) in the Info.plist

2019-10-21 22:21:31.668387-0500 chat-cat[40730:16737714] 6.9.0 - [Firebase/Analytics][I-ACS023007] Analytics v.6.0.1 started

2019-10-21 22:21:31.674156-0500 chat-cat[40730:16737714] 6.9.0 - [Firebase/Analytics][I-ACS023008] To enable debug logging set the following application argument: -FIRAnalyticsDebugEnabled (see <http://goo.gl/RfcP7r>)

CONSOLE GETS PRETTY MESSY FROM FIREBASE LOGGING

All Output Filter

FIREBASE IN ACTION

CONSOLE

- ✓ Add Firebase SDK
- ✓ Add initialization code
- 5 Run your app to verify installation

✓ Congratulations, you've successfully added Firebase to your app!

Previous

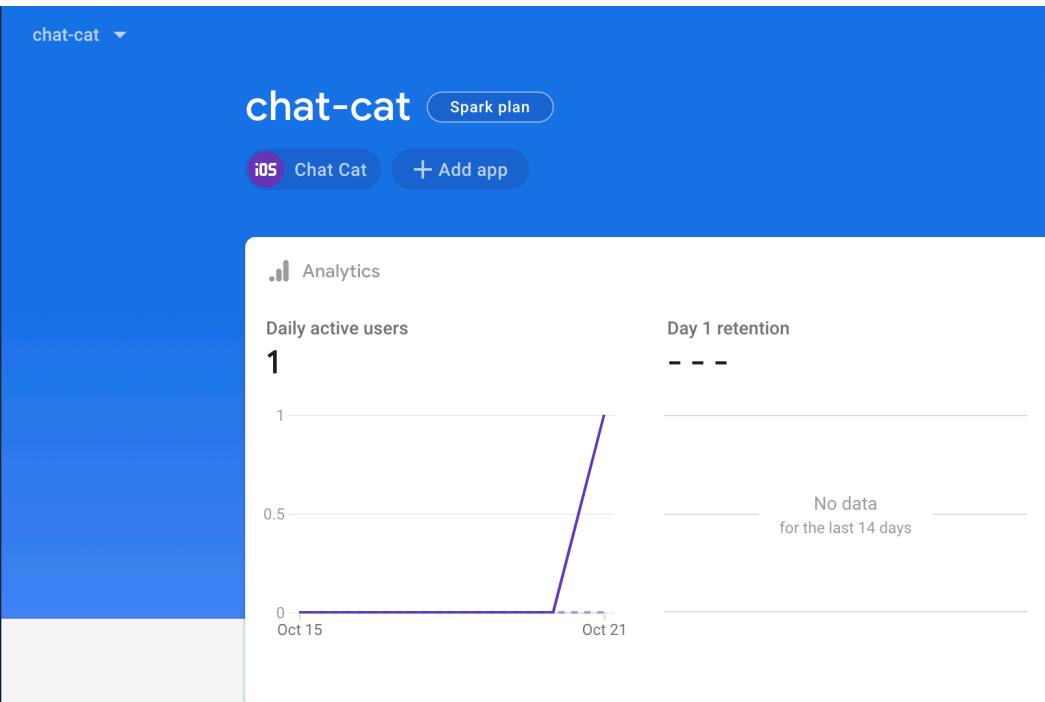
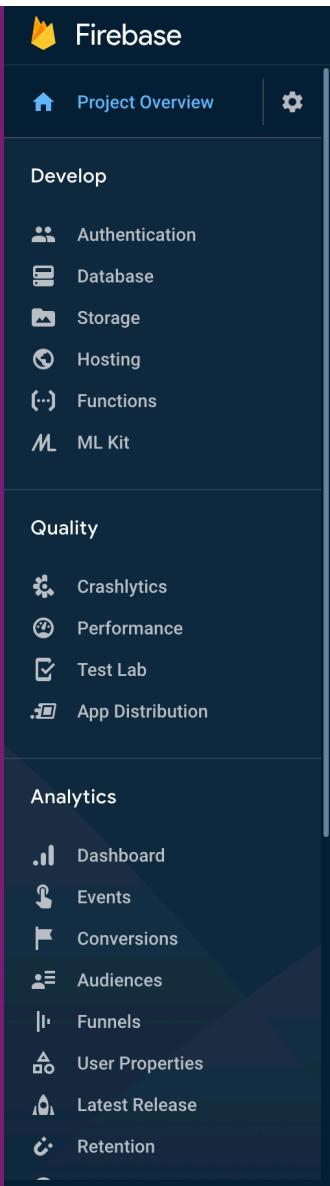
Continue to console



FIREBASE CONSOLE

FIREBASE CONSOLE

- Enable services
- Data
- Analytics
- Configure



Store and sync app data in milliseconds



FIREBASE CONSOLE

- Everything you wanted to know about your app
- Quick access to documentation

The screenshot shows the Firebase Authentication console under the 'chat-cat' project. The left sidebar includes sections for Project Overview, Develop (Authentication, Database, Storage, Hosting, Functions, ML Kit), Quality (Crashlytics, Performance, Test Lab, App Distribution), Analytics (Dashboard, Events, Conversions, Audiences, Funnels, User Properties, Latest Release, Retention, Extensions), and a Help section.

The main 'Authentication' screen displays a table of users. The columns are Identifier, Providers, Created, Signed In, and User UID. A single user entry is shown:

Identifier	Providers	Created	Signed In	User UID
abinkowski@uchicago.edu		Oct 21, 20...	Oct 22, 20...	KvbPdex7EgXN6ZXQVL3Cw...

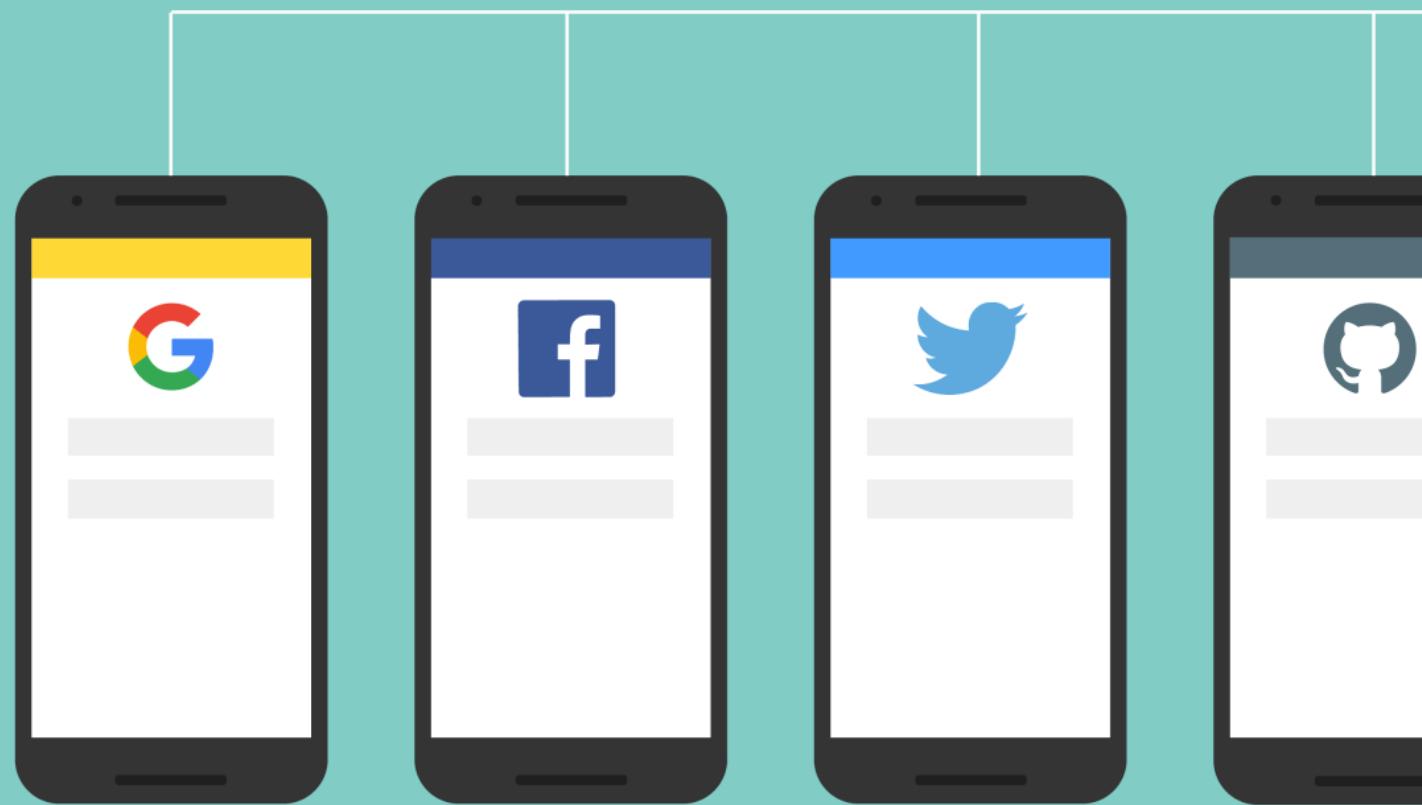
At the top right of the table, there is an 'Add user' button. A large yellow arrow points from the bottom right towards this button. The top right corner of the slide also features a yellow arrow pointing upwards.

FIREBASE AUTHENTICATION

FIREBASE AUTHENTICATION

- [https://
www.youtube.co
m/watch?
v=8sGY55yxicA&i
ndex=14&list=PLI
_=
K7zZEesYLmOF_07
IayrTntevxtbUxDL](https://www.youtube.com/watch?v=8sGY55yxicA&index=14&list=PLI_=_K7zZEesYLmOF_07IayrTntevxtbUxDL)

Firebase



FIREBASE AUTHENTICATION

- Provide a secure and customized experience for users
- Many options, including your own custom systems

Authentication

Users

Sign-in method

Templates

Usage



Search by email address, phone number, or user UID

Add u

Identifier

Providers

Created

Signed In

User UID ↑



Authenticate and manage users from a variety of providers without server-side code

[Learn more](#)

[View the docs](#)

[Set up sign-in method](#)

FIREBASE AUTHENTICATION

- Authentication flow
 - User logs in to service through interface
 - Credentials passed to Firebase Authentication SDK
 - Backend does authorization and passes back response from the client

Firebase



FIREBASE AUTHENTICATION

- Templates for interaction

Authentication

Users Sign-in method Templates Usage

Templates

Email

 Email address verification

 Password reset

 Email address change

 SMTP settings

SMS

 SMS verification

Template language
English

Email address verification

When a user signs up using an email address and password, you can send them a confirmation email to verify their registered email address. [Learn more](#)

Sender name
not provided

From
noreply@chat-cat-2f244.firebaseio.com

Reply to
not provided

Subject
Verify your email for Chat Cat

Message

Hello %DISPLAY_NAME%,

Follow this link to verify your email address.

https://chat-cat-2f244.firebaseio.com/_auth/action?mode=<action>&oobCode=<code>

If you didn't ask to verify this address, you can ignore this email.

Thanks,

Your Chat Cat team

FIREBASE AUTHENTICATION

- Using FirebaseUI Auth

Using FirebaseUI Auth

1

Set up sign-in methods

For email address and password sign-in and any federated identity providers you want to support, enable them in the Firebase console and complete any configuration required by the identity provider, such as setting your OAuth redirect URL.

2

Customize the sign-in UI

You can customize the sign-in UI by setting FirebaseUI options, or fork the code on GitHub to customize the sign-in experience further.

3

Use FirebaseUI to perform the sign-in flow

Import the FirebaseUI library, specify the sign-in methods you want to support, and initiate the FirebaseUI sign-in flow.

FIREBASE AUTHENTICATION

- Firebase Authentication SDK
- You create the UI

Using the Firebase Authentication SDK

1

Set up sign-in methods

For email address and password sign-in and any federated identity providers you want to support, enable them in the Firebase console and complete any configuration required by the identity provider, such as setting your OAuth redirect URL.

2

Implement UI flows for your sign-in methods

For email address and password sign-in, implement a flow that prompts users to type their email addresses and passwords. For federated sign-in, implement the flow required by each provider.

3

Pass the user's credentials to the Firebase Authentication SDK

Pass the user's email address and password or the OAuth token that was acquired from the federated identity provider to the Firebase Authentication SDK.

FIREBASE AUTHENTICATION

- Select method

Authentication

Users Sign-in method Templates Usage

Sign-in providers

Provider	Status
Email/Password	Disabled
Phone	Disabled
Google	Disabled
Play Games	Disabled
Game Center <small>Beta</small>	Disabled
Facebook	Disabled
Twitter	Disabled
Github	Disabled
Yahoo	Disabled
Microsoft	Disabled
Anonymous	Disabled

FIREBASE AUTHENTICATION

- Third-party authentication requires granting access
- Set up on each service

The screenshot shows the Firebase Authentication settings interface. At the top, there are three disabled social sign-in providers: Facebook, Twitter, and GitHub. The GitHub provider has its 'Enable' switch turned off. Below it, there are fields for 'Client ID' and 'Client secret', both of which are currently empty. A note below these fields instructs users to add the authorization callback URL to their GitHub app configuration, with a 'Learn more' link provided. The URL listed is https://firechat-66f87.firebaseio.com/_/auth/handler. At the bottom, there is another disabled provider for 'Anonymous' sign-in.

☰ Firebase FireChat Authentication Go to docs ⋮

Facebook Disabled

Twitter Disabled

Github

Enable

Client ID

Client secret

To complete set up, add this authorization callback URL to your GitHub app configuration.
[Learn more](#)

https://firechat-66f87.firebaseio.com/_/auth/handler

Anonymous Disabled

CANCEL

FIREBASE AUTHENTICATION

- Get Google for 'free'

The screenshot shows the Firebase Authentication interface under the 'Sign-in method' tab. It lists 'Sign-in providers' including 'Email/Password' and 'Phone', both of which are disabled. The 'Google' provider is selected, showing its configuration options. A note indicates that Google sign-in is automatically configured for iOS and web apps, but needs to be set up for Android apps by adding the SHA1 fingerprint. A modal window titled 'Update the project-level setting below to continue' is open, prompting for the project public-facing name ('project-323794022426') and project support email ('Not configured'). A red error message 'Please select an email address' is displayed next to the support email field. At the bottom right of the modal are 'Cancel' and 'Save' buttons.

Sign-in providers

Provider	Status
Email/Password	Disabled
Phone	Disabled

Google

Enable

Google sign-in is automatically configured on your connected iOS and web apps. To set up Google sign-in for your Android apps, you need to add the [SHA1 fingerprint](#) for each app on your [Project Settings](#).

Update the project-level setting below to continue

Project public-facing name [?](#)
project-323794022426

Project support email [?](#)
Not configured

Please select an email address !

Whitelist client IDs from external projects (optional) [?](#)

Web SDK configuration [?](#)

Cancel Save

FIREBASE AUTHENTICATION

```
# Uncomment the next line to define a global platform for your project
# platform :ios, '9.0'

target 'chat-cat' do
  # Comment the next line if you don't want to use dynamic frameworks
  use_frameworks!

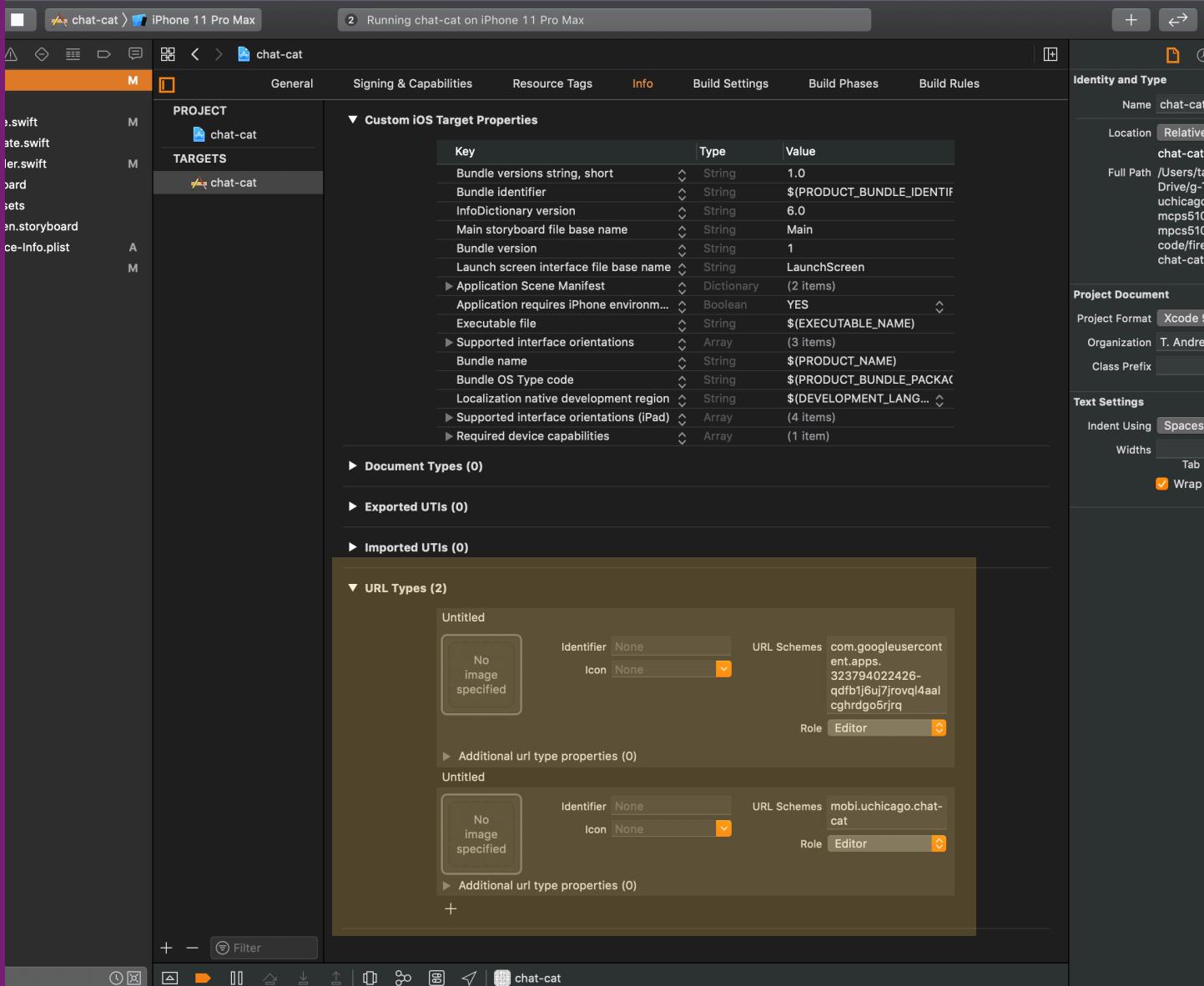
  # Pods for chat-cat

  # add the Firebase pod for Google Analytics
  pod 'Firebase/Analytics'

  # add pods for any other desired Firebase products
  pod 'Firebase/Database'
  pod 'Firebase/Auth'
  pod 'GoogleSignIn'
  # https://firebase.google.com/docs/ios/setup#available-pods
end
```

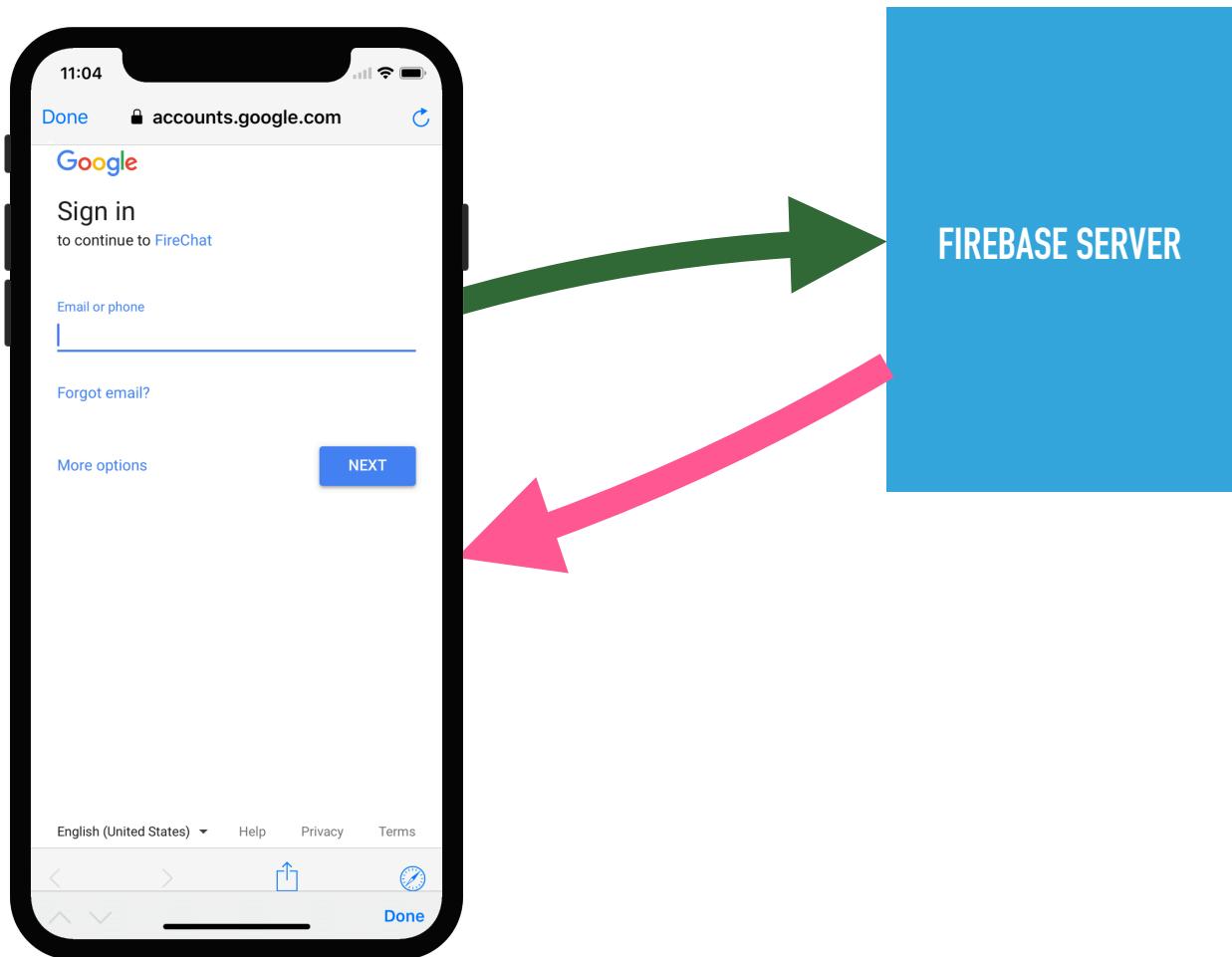
FIREBASE AUTHENTICATION

- Add URL types to the target info
- Provides way to launch app from login callback
- Info in Google-info.plist



FIREBASE AUTHENTICATION

- Flow
 - App launches browser
 - Browser logs in
 - Successful login redirects to URL
 - URL launches app



FIREBASE AUTHENTICATION

- Update the app delegate to handle authentication

```
// https://medium.com/@ibjects/google-sign-in-using-firebase-for-ios-straight-forward-step-by-step-guide-b2d2984fbf8e

import UIKit
import Firebase
import GoogleSignIn

@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate, GIDSignInDelegate {

    func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?) -> Bool {
        // Use Firebase library to configure APIs
        FirebaseApp.configure()

        GIDSignIn.sharedInstance().clientID = FirebaseApp.app()?.options.clientID
        return true
    }

    /// Open the app from a callback
    @available(iOS 9.0, *)
    func application(_ app: UIApplication, open url: URL, options: [UIApplication.OpenURLOptionsKey : Any] = [:]) -> Bool {
        let handled = GIDSignIn.sharedInstance().handle(url)
        return handled
    }

    func application(_ application: UIApplication, open url: URL, sourceApplication: String?, annotation: Any) -> Bool {
        return GIDSignIn.sharedInstance().handle(url)
    }

    /// Google Sign-in Sign-out Delegate Methods
    func sign(_ signIn: GIDSignIn!, didSignInFor user: GIDGoogleUser!, withError error: Error?) { ... }

    func sign(_ signIn: GIDSignIn!, didDisconnectWith user: GIDGoogleUser!, withError error: Error!) {
        // Perform any operations when the user disconnects from app here. ...
    }
}
```

FIREBASE AUTHENTICATION

```
// Open the app from URL callback during Google authentication
@available(iOS 9.0, *)
func application(_ application: UIApplication,
                 open url: URL,
                 options: [UIApplicationOpenURLOptionsKey : Any]) -> Bool {
    return GIDSignIn.sharedInstance().handle(url,
                                             sourceApplication:options[UIApplicationOpenURLOptionsKey.sourceApplication] as? String,
                                             annotation: [:])
}

// MARK: - Google Sign In/Out Delegate Methods
//
func sign(_ signIn: GIDSignIn!, didSignInFor user: GIDGoogleUser!, withError error: Error?) { ... }
```

- Allows the app to open from a callback
- Standard implementation (not custom to Firebase)

FIREBASE AUTHENTICATION

```
func sign(_ signIn: GIDSignIn!, didSignInFor user: GIDGoogleUser!, withError error: Error!) {
    //Sign in functionality will be handled here
    if let error = error {
        print(error.localizedDescription)
        return
    }
    guard let auth = user.authentication else { return }
    let credentials = GoogleAuthProvider.credential(withIDToken: auth.idToken, accessToken: auth.accessToken)

    // Pass to Firebase
    Auth.auth().signIn(with: credentials) { (authResult, error) in
        if let error = error {
            print(error.localizedDescription)
        } else {
            print("Login Successful.")

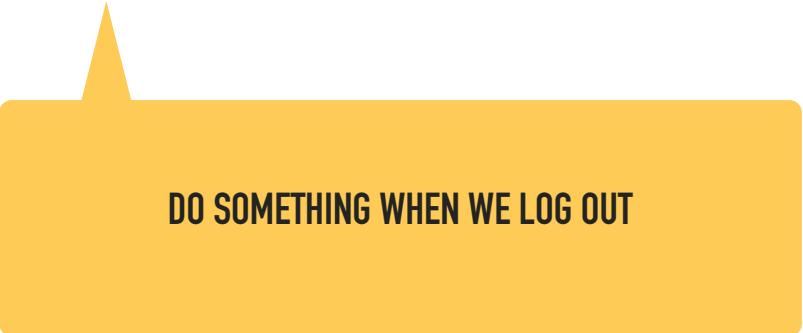
            //This is where you should add the functionality of successful login
            //i.e. dismissing this view or push the home view controller etc
            let user: GIDGoogleUser = GIDSignIn.sharedInstance()!.currentUser
            let fullName = user.profile.name
            let email = user.profile.email
            print(fullName!)
            print(email!)
        }
    }
}
```

AUTHENTICATE
FROM GOOGLE

SUCCESS FROM
GOOGLE NOW PASS
TO FIREBASE

FIREBASE AUTHENTICATION

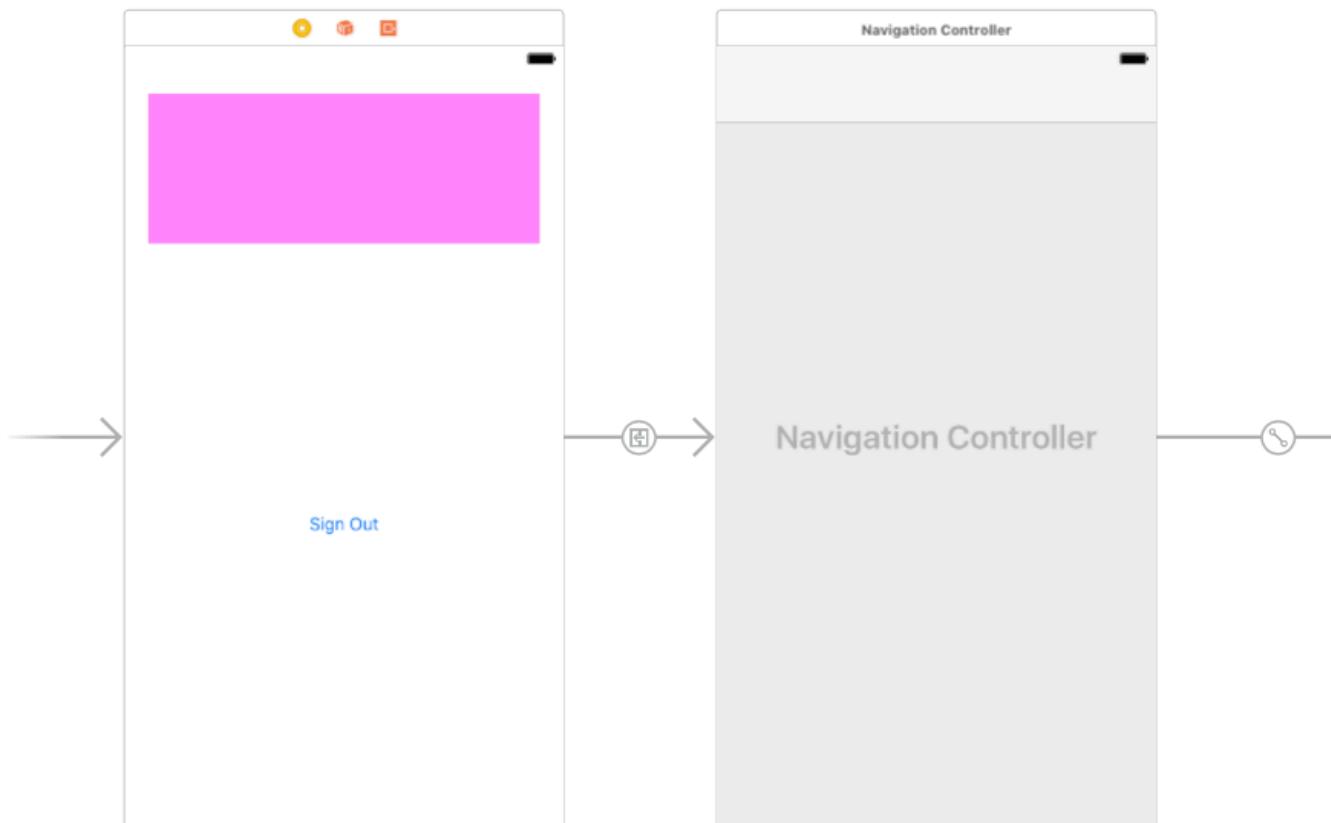
```
//  
// MARK: - Google Sign In/Out Delegate Methods  
//  
func sign(_ signIn: GIDSignIn!, didSignInFor user: GIDGoogleUser!, withError error: Error?) { ... }  
  
func sign(_ signIn: GIDSignIn!, didDisconnectWith user: GIDGoogleUser!, withError error: Error!) {  
    // Perform any operations when the user disconnects from app here.  
    print("Signed out...")  
}
```



DO SOMETHING WHEN WE LOG OUT

FIREBASE AUTHENTICATION

- Login view controller
 - Google button
 - Logout button
 - Authenticate
 - Segue when successful



FIREBASE AUTHENTICATION

- Login view controller
 - Google button
 - Logout button
 - Authenticate
 - Segue when successful

```
import UIKit
import Firebase
import FirebaseAuth
import GoogleSignIn

class ViewController: UIViewController, GIDSignInDelegate {

    // Google button
    @IBOutlet weak var LoginButton: GIDSignInButton!

    @IBAction func googleSignInPressed(_ sender: Any) {
        GIDSignIn.sharedInstance().signIn()
    }

    @IBAction func tapLogout(_ sender: Any) {
        //GIDSignIn.sharedInstance().signOut()
        let firebaseAuth = Auth.auth()
        do {
            try firebaseAuth.signOut()
        } catch let signOutError as NSError {
            print ("Error signing out: %@", signOutError)
        }
        print("signed out")
    }

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view.
        GIDSignIn.sharedInstance()?.presentingViewController = self
        GIDSignIn.sharedInstance().delegate = self
    }

    override func viewWillAppear(_ animated: Bool) {
        let user = Auth.auth().currentUser
        if user?.uid == nil {
            print("User not logged in")
        } else {
            print("User logged in:",Auth.auth().currentUser?.email ?? "not logged in")
        }

        // Setup listener
        Auth.auth().addStateDidChangeListener() { auth, user in

            // Automatically segue after a successfull login; users seem to
            // hang around while developing, you may need to clean and remove
            // app from simulator
            if user != nil {
                self.performSegue(withIdentifier: "segueToMessages", sender: nil)
            }
        }
    }
}
```

SEGUE IF
AUTHENTICATED
USER

FIREBASE AUTHENTICATION

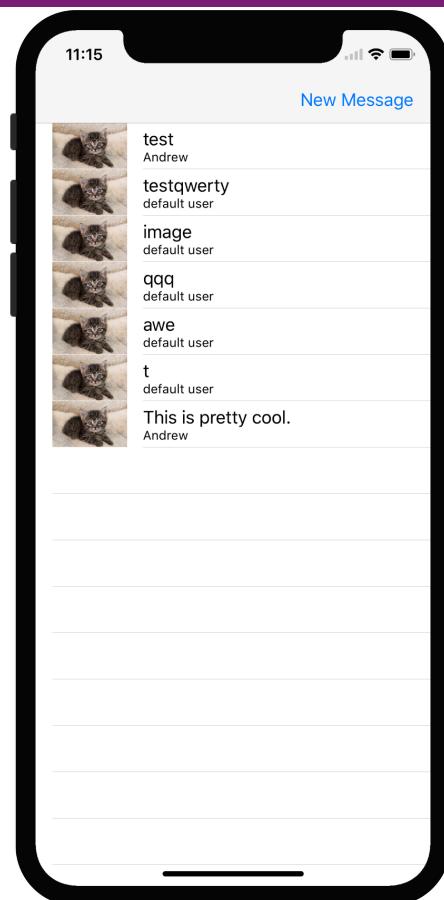
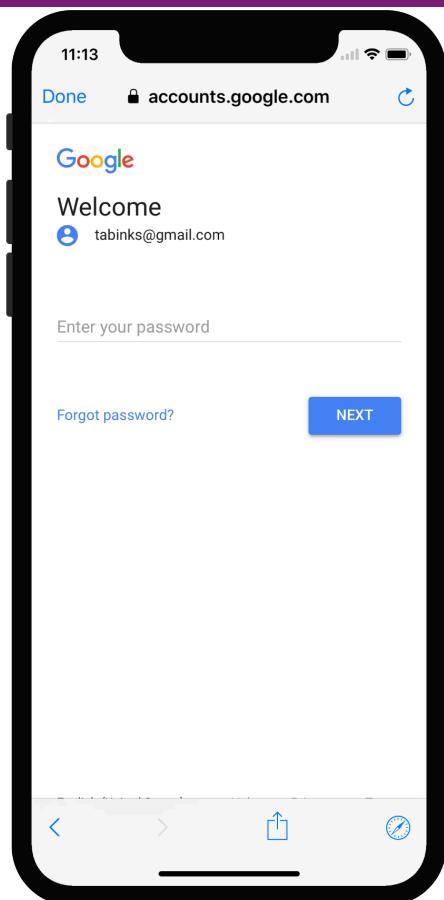
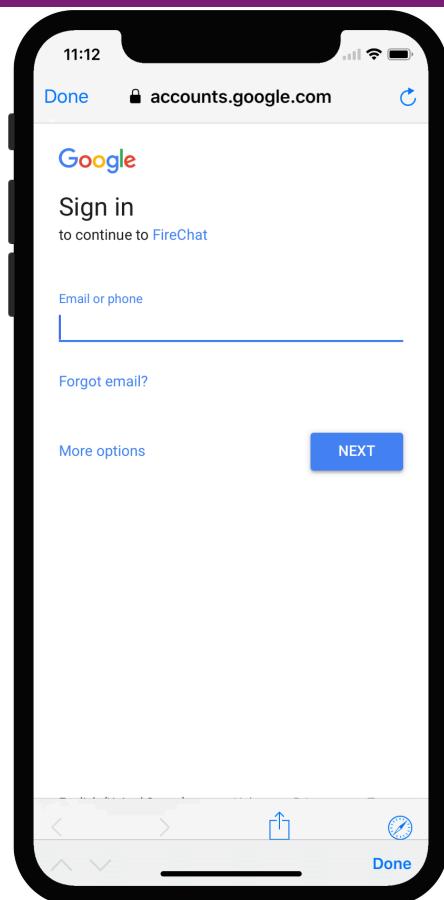
```
override func viewDidAppear(_ animated: Bool) {  
  
let user = Auth.auth().currentUser  
if user?.uid == nil {  
    print("User not logged in")  
} else {  
    print("User logged in:",Auth.auth().currentUser?.email ?? "not logged in")  
}  
  
// Setup listener  
Auth.auth().addStateDidChangeListener() { auth, user in  
    // Automatically segue after a successfull login; users seem to  
    // hang around while developing, you may need to clean and remove  
    // app from simulator  
    if user != nil {  
        self.performSegue(withIdentifier: "segueToMessages", sender: nil)  
    }  
}
```

ADD OBSERVER IN
CASE USER LOGS
OUT

FIREBASE AUTHENTICATION

```
@IBAction func tapLogout(_ sender: Any) {  
    //GIDSignIn.sharedInstance().signOut()  
    let firebaseAuth = Auth.auth()  
    do {  
        try firebaseAuth.signOut()  
    } catch let signOutError as NSError {  
        print ("Error signing out: %@", signOutError)  
    }  
    print("signed out")  
}
```

FIREBASE AUTHENTICATION



FIREBASE AUTHENTICATION

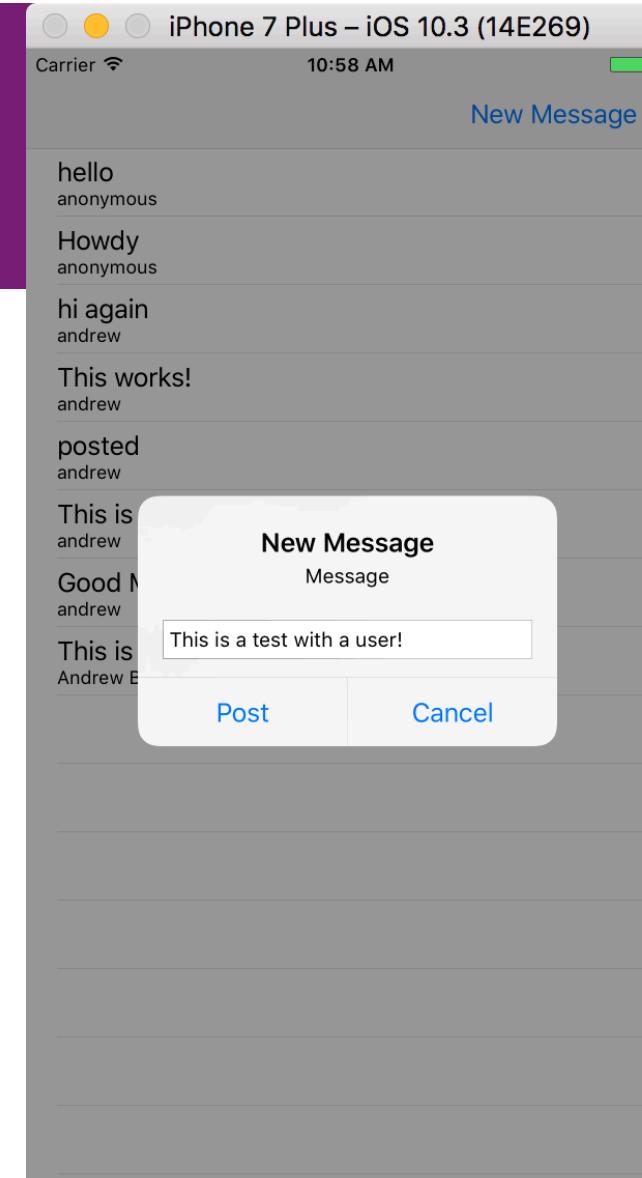
Authentication

Users Sign-in method Templates Usage

Search by email address, phone number, or user UID					Add user	C
Identifier	Providers	Created	Signed In	User UID ↑		
abinkowski@uchicago.edu		Oct 21, 2019	Oct 21, 2019	KvbPdex7EgXN6ZXQVL3CwHO8Q...		
					Rows per page:	50 ▾
					1-1 of 1	<

FIREBASE AUTHENTICATION

firechat-66f87



FIREBASE DATABASES

FIREBASE

- Options for storing data
 - Realtime Database (classic)
 - Cloud Firestore

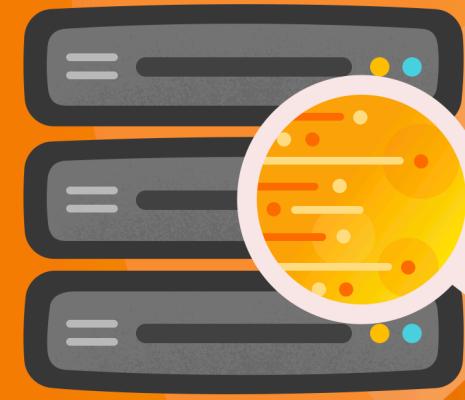
Cloud Firestore

Realtime updates, powerful queries, and automatic scaling

[Create database](#)

[Learn more](#)

-  Find out if Cloud Firestore is right for you
[Compare databases](#)
-  How do I get started?
[View the docs](#)
-  How much will Cloud Firestore cost?
[View pricing](#)
-  What can Cloud Firestore do for me?
[Learn more](#)



DATABASE

Choose a Database: Cloud Firestore or Realtime Database

Firebase offers two cloud-based, client-accessible database solutions that support realtime data syncing:

- **Cloud Firestore** is Firebase's newest database for mobile app development. It builds on the successes of the Realtime Database with a new, more intuitive data model. Cloud Firestore also features richer, faster queries and scales further than the Realtime Database.
- **Realtime Database** is Firebase's original database. It's an efficient, low-latency solution for mobile apps that require synced states across clients in realtime.

Which database is right for your project?

[HTTPS://FIREBASE.GOOGLE.COM/DOCS/DATABASE/RTDB-VS-FIRESTORE?AUTHUSER=0](https://firebase.google.com/docs/database/rtdb-vs-firebase?authuser=0)

We recommend Cloud Firestore for most developers starting a new project. Cloud Firestore offers additional functionality, performance, and scalability on an infrastructure designed to support more powerful features in the future.

DATABASE

PROS

- Realtime
 - Giant JSON tree
 - Simple queries
 - Automatic scaling to a point
 - Old way, but useful for more 'realtime' apps
- Firestore
 - Document based
 - More complex queries
 - Automatic scaling
 - New way, more robust

DATABASE

PROS

- Realtime
 - Giant JSON tree
 - Simple queries
 - Automatic scaling to handle traffic
 - Old way, but useful for more 'realtime' apps
- CAN USE BOTH IN THE SAME PROJECT (APP)
- Firestore
 - Document based
 - Complex queries
 - Automatic scaling
 - New way, more robust

REALTIME DATABASE

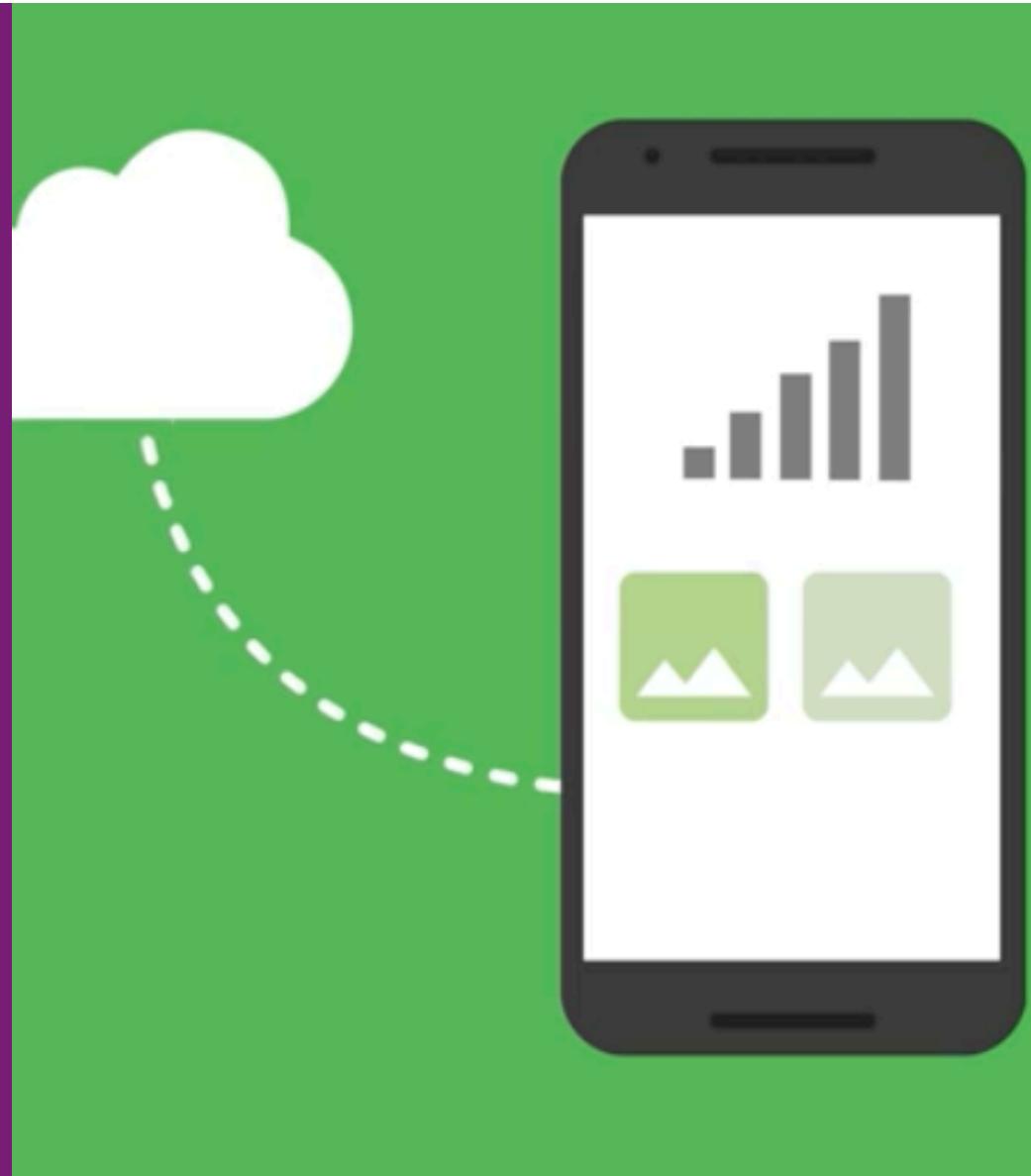
REALTIME DATABASE

- Store and sync data with NoSQL cloud database
- Data is synced across all clients in realtime, and remains available when your app goes offline
 - https://youtu.be/O17OWyx08Cg?list=PLIK7zZEsvYLmOF_07IayrTntevxtbUxDL



REALTIME DATABASE

- Data is stored as JSON
- Synchronized in realtime to all clients
 - Milliseconds
- All networking code is included as part of the SDK



REALTIME DATABASE



- Data is stored in a local cache while the user is offline
- Changes are automatically synced when the device comes back online

REALTIME DATABASE

- Implementation steps

1	Integrate the Firebase Realtime Database SDKs	Quickly include clients via Gradle, CocoaPods, or a script include.
2	Create Realtime Database References	Reference your JSON data, such as "users/user:1234/phone_number" to set data or subscribe to data changes.
3	Set Data and Listen for Changes	Use these references to write data or subscribe to changes.
4	Enable Offline Persistence	Allow data to be written to the device's local disk so it can be available while offline.
5	Secure your data	Use Firebase Realtime Database Security Rules to secure your data.

REALTIME DATABASE

```
target 'chat-cat' do
  # Comment the next line if you don't want to use dynamic frameworks!
  use_frameworks!

  # Pods for chat-cat

  # add the Firebase pod for Google Analytics
  pod 'Firebase/Analytics'

  # add pods for any other desired Firebase products
  pod 'Firebase/Database'
  # https://firebase.google.com/docs/ios/setup#available-pods
end
```

NEED TO POD INSTALL AGAIN

REALTIME DATABASE

Database  Realtime Database ▾

Data Rules Backups Usage

 <https://chat-cat-2f244.firebaseio.com/>  

 Your security rules are defined as public, so anyone can steal, modify, or delete data in your database [Learn more](#)

chat-cat-2f244:  

DATABASE SECURITY

REALTIME DATABASE

- Security is handled through "Firebase Realtime Database Security" rules
 - You set permissions on the database
 - Authentication is handled by "Firebase Authentication"



REALTIME DATABASE

- Rules are set through the console
- Command line

Or choose Realtime Database

Security rules for Realtime Database

Once you have defined your data structure you will have to write rules to secure your data.

[Learn more](#)

Start in locked mode

Make your database private by denying all reads and writes

Start in test mode

Get set up quickly by allowing all reads and writes to your database

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

i All third party reads and writes will be denied

Cancel

Enable



Functions

Extend and connect Firebase features



Storage

Store & retrieve user generated content



Hosting

Deploy web apps in seconds

[See all Develop features](#)

REALTIME DATABASE

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

- Default
 - Requires authentication

REALTIME DATABASE

```
// 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  
  }  
}
```

- Public
 - Anyone

REALTIME DATABASE

```
// These rules grant access to a node matching the authenticated
// user's ID from the Firebase auth token
{
  "rules": {
    "users": {
      "$uid": {
        ".read": "$uid === auth.uid",
        ".write": "$uid === auth.uid"
      }
    }
  }
}
```

- User
 - Each user has their own database

REALTIME DATABASE

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

- Private
 - No one can access outside of console

DATA STRUCTURE

DATA STRUCTURE

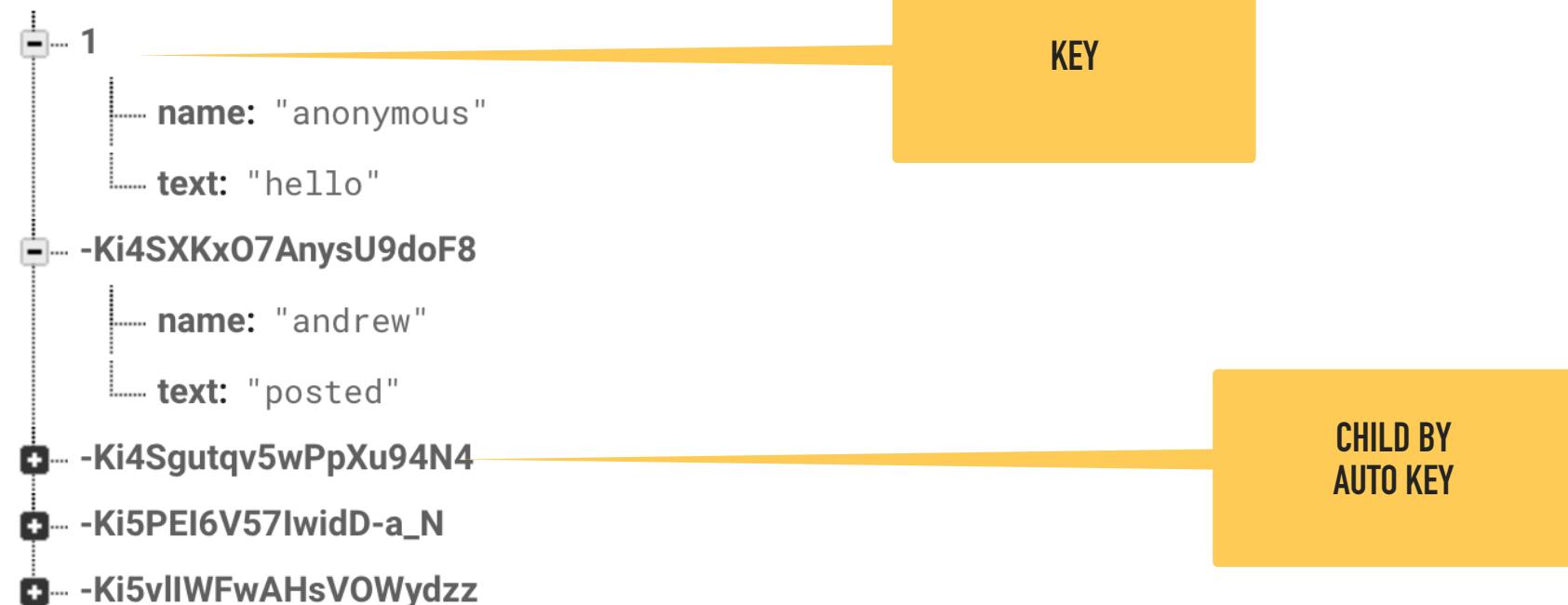
- JSON trees stored online
- Use multiple trees (not linked)
- Data is represented as a node accessible by a key
 - Use your own key (string)
 - `childByAutold`

```
{  
  "users": {  
    "alovelace": {  
      "name": "Ada Lovelace",  
      "contacts": { "ghopper": true },  
    },  
    "ghopper": { ... },  
    "eclarke": { ... }  
  }  
}
```

DATA STRUCTURE

[firechat-66f87](#) > [messages](#)

messages



DATA STRUCTURE

BEST PRACTICES FOR DATA

- Avoid nesting data
- Fetch returns all data
- Access control
- Parsing in Swift isn't fun

```
{  
    // This is a poorly nested data architecture,  
    // of the "chats" node to get a list of conversations  
    // potentially downloading hundreds of megabytes  
    "chats": {  
        "one": {  
            "title": "Historical Tech Pioneers",  
            "messages": {  
                "m1": { "sender": "ghopper", "message": "..."},  
                "m2": { ... },  
                // a very long list of messages  
            }  
        },  
        "two": { ... }  
    }  
}
```

DATA STRUCTURE

BEST PRACTICES FOR DATA

- Flatten data
- Isolate into different calls
- Use metadata for faster fetching
- Maintain unique references

```
{  
    // Chats contains only meta info about each conversation  
    // stored under the chats's unique ID  
    "chats": {  
        "one": {  
            "title": "Historical Tech Pioneers",  
            "lastMessage": "ghopper: Relay malfunction found. Cause: moth.",  
            "timestamp": 1459361875666  
        },  
        "two": { ... },  
        "three": { ... }  
    },  
  
    // Conversation members are easily accessible  
    // and stored by chat conversation ID  
    "members": {  
        // we'll talk about indices like this below  
        "one": {  
            "ghopper": true,  
            "alovelace": true,  
            "eclarke": true  
        },  
        "two": { ... },  
        "three": { ... }  
    },  
  
    // Messages are separate from data we may want to iterate quickly  
    // but still easily paginated and queried, and organized by chat  
    // conversation ID  
    "messages": {  
        "one": {  
            "m1": {  
                "name": "eclarke",  
                "message": "The relay seems to be malfunctioning.",  
                "timestamp": 1459361875666  
            }  
        }  
    }  
}
```

DATA STRUCTURE

BEST PRACTICES FOR DATA

- Flatten data
- Isolate into different calls
- Use metadata for faster fetching
- Maintain unique references

```
{  
    // Chats contains only meta info about each conversation  
    // stored under the chats's unique ID  
    "chats": {  
        "one": {  
            "title": "Historical Tech Pioneers",  
            "lastMessage": "ghopper: Relay malfunction found. Cause: moth.",  
            "timestamp": 1459361875666  
        },  
        "two": { ... },  
        "three": { ... }  
    },  
  
    // Conversation members are easily accessible  
    // and stored by chat conversation ID  
    "members": {  
        // we'll talk about indices like this below  
        "one": {  
            "ghopper": true,  
            "alovelace": true,  
            "eclarke": true  
        },  
        "two": { ... },  
        "three": { ... }  
    },  
  
    // Messages are separate from data we may want to iterate quickly  
    // but still easily paginated and queried, and organized by chat  
    // conversation ID  
    "messages": {  
        "one": {  
            "m1": {  
                "name": "eclarke",  
                "message": "The relay seems to be malfunctioning.",  
                "timestamp": 1459361875666  
            }  
        }  
    }  
}
```

THE KEY "ONE" IS
USED ACROSS
DIFFERENT
DICTIONARIES

ALL REFER TO THE
SAME INSTANCE OF
THE CHAT "ONE"

DATA STRUCTURE

BEST PRACTICES FOR DATA

- Don't be afraid of redundant information
- Think "ease of retrieval" instead of efficient storage

```
// An index to track Ada's memberships
{
  "users": {
    "alovelace": {
      "name": "Ada Lovelace",
      // Index Ada's groups in her profile
      "groups": {
        // the value here doesn't matter, just that the key exists
        "techpioneers": true,
        "womentechmakers": true
      },
      ...
    },
    "groups": {
      "techpioneers": {
        "name": "Historical Tech Pioneers",
        "members": {
          "alovelace": true,
          "ghopper": true,
          "eclarke": true
        }
      },
      ...
    }
  }
}
```



/USERS/\$UID/GROUPS/\$GROUP_ID

DATA STRUCTURE

BEST PRACTICES FOR DATA

- Don't be afraid of redundant information
- Think "ease of retrieval" instead of efficient storage

```
// An index to track Ada's memberships
{
  "users": {
    "alovelace": {
      "name": "Ada Lovelace",
      // Index Ada's groups in her profile
      "groups": {
        // the value here doesn't matter, just that the key exists
        "techpioneers": true,
        "womentechmakers": true
      }
    },
    ...
  },
  "groups": {
    "techpioneers": {
      "name": "Historical Tech Pioneers",
      "members": {
        "alovelace": true,
        "ghopper": true,
        "eclarke": true
      }
    },
    ...
  }
}
```

/USERS/\$UID/GROUPS/\$GROUP_ID

SELECT ALL FROM GROUPS WHERE MEMBERS CONTAINS GHOPPER

DATA STRUCTURE

- There is a CLI tool that can profile your database
 - Speed
 - Bandwidth
 - Query Indices

```
{  
  "scores": {  
    "bruhathkayosaurus" : 55,  
    "lambeosaurus" : 21,  
    "linhenykus" : 80,  
    "pterodactyl" : 93,  
    "stegosaurus" : 5,  
    "triceratops" : 22  
  }  
}
```

DATA STRUCTURE

- You can specify which keys to index on to improve performance

```
{  
  "scores": {  
    "bruhathkayosaurus" : 55,  
    "lambeosaurus" : 21,  
    "linhenykus" : 80,  
    "pterodactyl" : 93,  
    "stegosaurus" : 5,  
    "triceratops" : 22  
  }  
}  
  
{  
  "rules": {  
    "scores": {  
      ".indexOn": ".value"  
    }  
  }  
}
```

READ DATA

READ AND WRITE DATA

- Get a reference to the database

```
/// Database reference
let reference = Database.database().reference()
```

- Access nodes by key
- Set values by key

```
self.ref.child("users/(user.uid)/username").setValue(username)

self.ref.child("users").child(user.uid).setValue(["username": username])
```

READ AND WRITE DATA

- Observe changes to the database
- Take a snapshot when you need it

```
override func viewDidAppear(_ animated: Bool) {
    super.viewDidAppear(animated)

    // Set up observer for the messages data
    reference.observe(.value, with: { snapshot in
        print(snapshot.value!)

        // Create an array of the incoming data; convert it; then update the tab
        var freshData = [MessageItem]()
        for item in snapshot.children {
            let messageItem = MessageItem(snapshot: item as! DataSnapshot)
            freshData.append(messageItem)
        }

        self.messages = freshData
        print(self.messages)
        self.tableView.reloadData()

    })

    // Keep track of the user
    Auth.auth().addStateDidChangeListener { auth, user in
        guard let user = user else { return }
        self.user = user
    }

}
```

OBSERVE CHANGES TO DATA

DATA RETURNED AS SNAPSHOT

READ AND WRITE DATA

- The values are returned as native Swift types not JSON
 - Dictionary, Array, String, Int, Double



READ AND WRITE DATA

- Best practice to convert to custom struct

```
struct MessageItem {

    let key: String
    let name: String
    let text: String
    let ref: DatabaseReference?

    init(name: String, text: String, key: String = "") {
        self.key = key
        self.name = name
        self.text = text
        self.ref = nil
    }

    init(snapshot: DataSnapshot) {
        key = snapshot.key
        let snapshotValue = snapshot.value as! [String: AnyObject]
        name = snapshotValue["name"] as! String
        text = snapshotValue["text"] as! String
        ref = snapshot.ref
    }
}
```



```
chat-cat-2f244
  messages
    -Lroj25VC3rN_80YBuab
      name: "Thomas Binkowski"
      text: "This is my first post!"
    -LrojuJHjuMIL3JB7o23
      name: "Thomas Binkowski"
      text: "This is my second post."
```

READ AND WRITE DATA

- Best practice to convert to custom struct

```
for item in snapshot.children {  
    let messageItem = MessageItem(snapshot: item as! DataSnapshot)  
    freshData.append(messageItem)  
}
```

```
struct MessageItem {  
  
    let key: String  
    let name: String  
    let text: String  
    let ref: DatabaseReference?  
  
    init(name: String, text: String, key: String = "") {  
        self.key = key  
        self.name = name  
        self.text = text  
        self.ref = nil  
    }  
  
    init(snapshot: DataSnapshot) {  
        key = snapshot.key  
        let snapshotValue = snapshot.value as! [String: Any]  
        name = snapshotValue["name"] as! String  
        text = snapshotValue["text"] as! String  
        ref = snapshot.ref  
    }  
}
```



READ AND WRITE DATA

- Best practice to convert to custom struct

```
struct MessageItem {  
  
    let key: String  
    let name: String  
    let text: String  
    let ref: DatabaseReference?  
  
    init(name: String, text: String, key: String = "") {  
        self.key = key  
        self.name = name  
        self.text = text  
        self.ref = nil  
    }  
  
    init(snapshot: DataSnapshot) {  
        key = snapshot.key  
        let snapshotValue = snapshot.value as! [String: AnyObject]  
        name = snapshotValue["name"] as! String  
        text = snapshotValue["text"] as! String  
        ref = snapshot.ref  
    }  
}
```

STRUCT DOESN'T KNOW ABOUT FIREBASE...KEEP A DIRECT REFERENCE TO IT

READ AND WRITE DATA

```
let userID = Auth.auth()?.currentUser?.uid
ref.child("users").child(userID!).observeSingleEvent(of: .value, with: { (snapshot) in
    // Get user value
    let value = snapshot.value as? NSDictionary
    let username = value?["username"] as? String ?? ""
    let user = User.init(username: username)

    // ...
}) { (error) in
    print(error.localizedDescription)
}
```

- Single callback for data not expected to change (eg. game levels, locations, members, etc.)

WRITE DATA

WRITE DATA

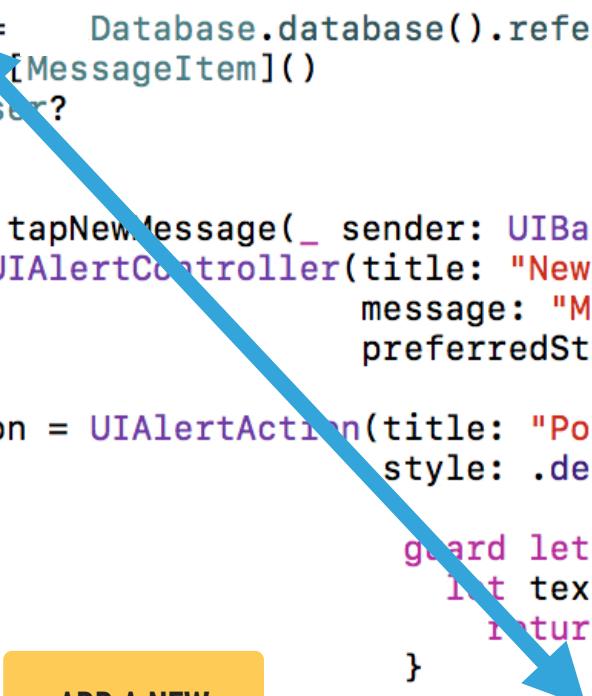
```
let reference = Database.database().reference().child("messages")
var messages: [MessageItem]()
var user: User?

@IBAction func tapNewMessage(_ sender: UIBarButtonItem) {
    let alert = UIAlertController(title: "New Message",
        message: "Message",
        preferredStyle: .alert)

    let saveAction = UIAlertAction(title: "Post",
        style: .default) { _ in
        guard let textField = alert.textFields?.first,
            let text = textField.text else {
            return
        }

        let data = ["name": self.user?.displayName, "text": text]
        self.reference.childByAutoId().setValue(data)
    }
}
```

ADD A NEW MESSAGE



WRITE DATA

```
let reference = Database.database().reference().child("messages")
var messages = [MessageItem]()
var user: User?

@IBAction func tapNewMessage(_ sender: UIBarButtonItem) {
    let alert = UIAlertController(title: "New Message",
                                 message: "Message",
                                 preferredStyle: .alert)

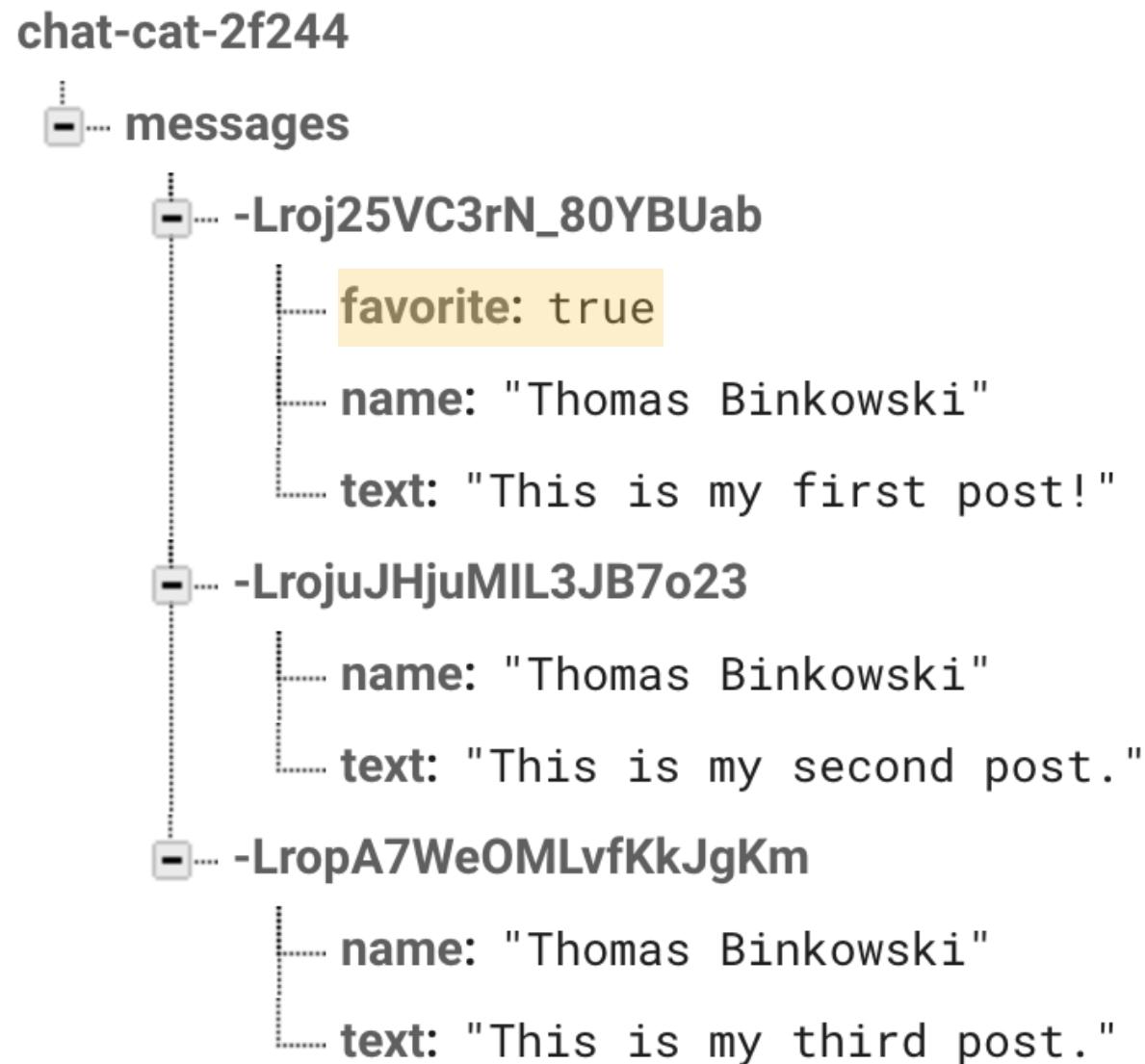
    let saveAction = UIAlertAction(title: "Post",
                                  style: .default) { _ in

        guard let textField = alert.textFields?.first,
              let text = textField.text else {
            return
        }

        let data = ["name":self.user?.displayName, "text": text]
        self.reference.childByAutoId().setValue(data)
    }
}
```

CAN LEAVE
FIELDS
EMPTY

WRITE DATA



- Data doesn't have to be homogenous

WRITE DATA

- Overwrite the entire node and children

```
self.ref.child("users").child(user.uid).setValue(["username": username])
```

- Updates a value

```
self.ref.child("users/(user.uid)/username").setValue(username)
```

WRITE DATA

```
let key = ref.child("posts").childByAutoId().key
let post = ["uid": userID,
            "author": username,
            "title": title,
            "body": body]
let childUpdates = ["/posts/\\(key)": post,
                    "/user-posts/\\(userID)/\\(key)": post]
ref.updateChildValues(childUpdates)
```

- Use `updateChildValues` to make changes to multiple locations

WRITE DATA

```
let key = ref.child("posts").childByAutoId().key
let post = ["uid": userID,
            "author": username,
            "title": title,
            "body": body]
let childUpdates = ["/posts/\\(key)": post,
                    "/user-posts/\\(userID)/\\(key)": post]
ref.updateChildValues(childUpdates)
```

- Delete data by call `deleteValue` on a node, set to `nil`
- Multiple children delete using `updateChildValues` to nil

WORKING WITH LISTS OF DATA

WORKING WITH LISTS OF DATA

- Using `childByAutoId` keeps data sorted by timestamp and unique
- Call `getKey` to find the unique key of a node
- Use the keys for flattening your data structure

```
let reference = FIRDatabase.database().reference().child("messages")
var messages = [MessageItem]()
var user: FIRUser?

@IBAction func tapNewMessage(_ sender: UIBarButtonItem) {
    let alert = UIAlertController(title: "New Message",
                                 message: "Message",
                                 preferredStyle: .alert)

    let saveAction = UIAlertAction(title: "Post",
                                  style: .default) { _ in

        guard let textField = alert.textFields?.first,
              let text = textField.text else {
            return
        }

        let data = ["name":self.user?.displayName,"text": text]
        self.reference.childByAutoId().setValue(data)
    }
}
```

WORKING WITH LISTS OF DATA

```
reference.observe(.value, with: { snapshot in
    print(snapshot.value!)

    var freshData = [MessageItem]()
    for item in snapshot.children {
```

- Returns everything
- Returns an array

WORKING WITH LISTS OF DATA

Event type	Typical usage
DataEventTypeChildAdded	Retrieve lists of items or listen for additions to a list of items. This event is triggered once for each existing child and then again every time a new child is added to the specified path. The listener is passed a snapshot containing the new child's data.
DataEventTypeChildChanged	Listen for changes to the items in a list. This event is triggered any time a child node is modified. This includes any modifications to descendants of the child node. The snapshot passed to the event listener contains the updated data for the child.
DataEventTypeChildRemoved	Listen for items being removed from a list. This event is triggered when an immediate child is removed. The snapshot passed to the callback block contains the data for the removed child.
DataEventTypeChildMoved	Listen for changes to the order of items in an ordered list. This event is triggered whenever an update causes reordering of the child. It is used with data that is ordered by <code>queryOrderedByChild</code> or <code>queryOrderedByValue</code> .

- Monitor for child events

WORKING WITH LISTS OF DATA

- Monitor for child events
- Think about how your data is structured

```
override func viewDidLoad() {
    super.viewDidLoad()

    // Load the data and observe
    //uploadFile()

    reference.observe(. , with: { snapshot in
        switch DataEventType.childAdded {
        case DataEventType.childChanged
        case DataEventType.childMoved
        case DataEventType.childRemoved
        case DataEventType.value
        }
        var freshData = [MessageItem]()
        for item in snapshot.children {
            if let messageItem = MessageItem(snapshot: item)
                self.messages = freshData
                print(self.messages)
                self.tableView.reloadData()
        }
    })
}
```

WORKING WITH LISTS OF DATA

```
// Listen for new comments in the Firebase database
commentsRef.observe(.childAdded, with: { (snapshot) -> Void in
    self.comments.append(snapshot)
    self.tableView.insertRows(at: [IndexPath(row: self.comments.count-1, section: self.kSectionComments)])
})
// Listen for deleted comments in the Firebase database
commentsRef.observe(.childRemoved, with: { (snapshot) -> Void in
    let index = self.indexOfMessage(snapshot)
    self.comments.remove(at: index)
    self.tableView.deleteRows(at: [IndexPath(row: index, section: self.kSectionComments)], with: UITableViewRowAnimation.fade)
})
```

RETURNS THE CHANGED NODE ONLY IN THE SNAPSHOT

WORKING WITH LISTS OF DATA

Method	Usage	
<code>queryOrderedByKey</code>	Order results by child keys.	KEYS HAVE INHERENT TIME STAMP
<code>queryOrderedByValue</code>	Order results by child values.	
<code>queryOrderedByChild</code>	Order results by the value of a specified child key.	

- Query for sorted data

```
let myTopPostsQuery = (ref.child("user-posts").child(getUid()).queryOrdered(byChild: "starCount"))
```

WORKING WITH LISTS OF DATA

- Filtering returned data

Method	Usage
queryLimitedToFirst	Sets the maximum number of items to return from the beginning of the ordered list of results.
queryLimitedToLast	Sets the maximum number of items to return from the end of the ordered list of results.
queryStartingAtValue	Return items greater than or equal to the specified key or value, depending on the order-by method chosen.
queryEndingAtValue	Return items less than or equal to the specified key or value, depending on the order-by method chosen.
queryEqualToString	Return items equal to the specified key or value, depending on the order-by method chosen.

```
// Last 100 posts, these are automatically the 100 most recent
// due to sorting by push() keys
let recentPostsQuery = (ref?.child("posts")?.queryLimited(toFirst: 100))!
```

STORE DATA OFFLINE

STORE DATA OFFLINE

- Firebase keeps a local cache that it uses when there is no network connectivity
- The app functions the same
 - Callbacks are made
 - Data can be added, deleted, edited
- Changes are stored in queue
- Once back online, all changes are applied to the datastore
- Stored for active listeners by default

STORE DATA OFFLINE

```
let scoresRef = Database.database().referenceWithPath("scores")
scoresRef.keepSynced(true)
```

- Stored for active listeners by default
- Request other paths be stored

STORE DATA OFFLINE

```
let scoresRef = Database.database().referenceWithPath("scores")
scoresRef.queryOrderedByValue().queryLimitedToLast(4).observeEventType(.ChildAdded, withBlock: { snapshot in
    print("The \(snapshot.key) dinosaur's score is \(snapshot.value)")
})
```

- Data queries use the cached version (if available)

STORE DATA OFFLINE

```
// Use Firebase library to configure APIs
FirebaseApp.configure()
Database.database().isPersistenceEnabled = true
```

- On by default 😊

STORE DATA OFFLINE

```
let presenceRef = Database.database().referenceWithPath("message");
// Write a string when this client loses connection
presenceRef.onDisconnectSetValue("I disconnected!")
```

```
let userLastOnlineRef = Database.database().referenceWithPath("users/joe/lastOnline")
userLastOnlineRef.onDisconnectSetValue(FIRServerValue.timestamp())
```

- Manage presence (ie who is connected)

MANAGE PRESENCE

MANAGE PRESENCE

- Identify who is online
 - Add a child to user
 - Update when they are online
 - Remove the child when they are off

```
// since I can connect from multiple devices, we store each connection
// any time that connectionsRef's value is null (i.e. has no children)
let myConnectionsRef = FIRDatabase.database().referenceWithPath("connections")

// stores the timestamp of my last disconnect (the last time I disconnected)
let lastOnlineRef = FIRDatabase.database().referenceWithPath("lastOnline")

let connectedRef = FIRDatabase.database().referenceWithPath("connected")

connectedRef.observeEventType(.Value, withBlock: { snapshot in
    // only handle connection established (or I've reconnected after disconnecting)
    guard let connected = snapshot.value as? Bool where connected == false else {
        return
    }
    // add this device to my connections list
    // this value could contain info about the device or a timestamp
    let con = myConnectionsRef.childByAutoId()
    con.setValue("YES")

    // when this device disconnects, remove it
    con.onDisconnectRemoveValue()

    // when I disconnect, update the last time I was seen online
    lastOnlineRef.onDisconnectSetValue(FIRServerValue.timestamp())
})|
```

MANAGE PRESENCE

```
// since I can connect from multiple devices, we store each connection instance separately
// any time that connectionsRef's value is null (i.e. has no children) I am offline
let myConnectionsRef = FIRDatabase.database().referenceWithPath("users/joe/connections")

// stores the timestamp of my last disconnect (the last time I was seen online)
let lastOnlineRef = FIRDatabase.database().referenceWithPath("users/joe/lastOnline")

let connectedRef = FIRDatabase.database().referenceWithPath(".info/connected")

connectedRef.observeEventType(.Value, withBlock: { snapshot in
    // only handle connection established (or I've reconnected after a loss of connection)
    guard let connected = snapshot.value as? Bool where connected else {
        return
    }
    // add this device to my connections list
    // this value could contain info about the device or a timestamp instead of just true
    let con = myConnectionsRef.childByAutoId()
    con.setValue("YES")

    // when this device disconnects, remove it
    con.onDisconnectRemoveValue()

    // when I disconnect, update the last time I was seen online
    lastOnlineRef.onDisconnectSetValue(FIRServerValue.timestamp())
})|
```

MANAGING USERS
ON/OFFLINE

3 DICTIONARIES

MANAGE PRESENCE

```
// since I can connect from multiple devices, we store each connection instance separately
// any time that connectionsRef's value is null (i.e. has no children) I am offline
let myConnectionsRef = FIRDatabase.database().referenceWithPath("users/joe/connections")

// stores the timestamp of my last disconnect (the last time I was seen online)
let lastOnlineRef = FIRDatabase.database().referenceWithPath("users/joe/lastOnline")

let connectedRef = FIRDatabase.database().referenceWithPath(".info/connected")

connectedRef.observeEventType(.Value, withBlock: { snapshot in
    // only handle connection established (or I've reconnected after a loss of connection)
    guard let connected = snapshot.value as? Bool where connected else {
        return
    }
    // add this device to my connections list
    // this value could contain info about the device or a timestamp instead of just true
    let con = myConnectionsRef.childByAutoId()
    con.setValue("YES")

    // when this device disconnects, remove it
    con.onDisconnectRemoveValue()

    // when I disconnect, update the last time I was seen online
    lastOnlineRef.onDisconnectSetValue(FIRServerValue.timestamp())
})|
```

MANAGE PRESENCE

```
// since I can connect from multiple devices, we store each connection instance separately
// any time that connectionsRef's value is null (i.e. has no children) I am offline
let myConnectionsRef = FIRDatabase.database().referenceWithPath("users/joe/connections")

// stores the timestamp of my last disconnect (the last time I was seen online)
let lastOnlineRef = FIRDatabase.database().referenceWithPath("users/joe/lastOnline")

let connectedRef = FIRDatabase.database().referenceWithPath(".info/connected")

connectedRef.observeEventType(.Value, withBlock: { snapshot in
    // only handle connection established (or I've reconnected after a loss of connection)
    guard let connected = snapshot.value as? Bool where connected else {
        return
    }
    // add this device to my connections list
    // this value could contain info about the device or a timestamp instead of just true
    let con = myConnectionsRef.childByAutoId()
    con.setValue("YES")

    // when this device disconnects, remove it
    con.onDisconnectRemoveValue()

    // when I disconnect, update the last time I was seen online
    lastOnlineRef.onDisconnectSetValue(FIRServerValue.timestamp())
})|
```

MANAGE PRESENCE

```
// since I can connect from multiple devices, we store each connection instance separately
// any time that connectionsRef's value is null (i.e. has no children) I am offline
let myConnectionsRef = FIRDatabase.database().referenceWithPath("users/joe/connections")

// stores the timestamp of my last disconnect (the last time I was seen online)
let lastOnlineRef = FIRDatabase.database().referenceWithPath("users/joe/lastOnline")

let connectedRef = FIRDatabase.database().referenceWithPath(".info/connected")

connectedRef.observeEventType(.Value, withBlock: { snapshot in
    // only handle connection established (or I've reconnected after a loss of connection)
    guard let connected = snapshot.value as? Bool where connected else {
        return
    }
    // add this device to my connections list
    // this value could contain info about the device or a timestamp instead of just true
    let con = myConnectionsRef.childByAutoId()
    con.setValue("YES")

    // when this device disconnects, remove it
    con.onDisconnectRemoveValue()

    // when I disconnect, update the last time I was seen online
    lastOnlineRef.onDisconnectSetValue(FIRServerValue.timestamp())
})|
```

MANAGE PRESENCE

```
if let name = user?.displayName {  
    let lastMessage = Database.database().reference(withPath: "/users/\\(name)/lastMessage/")  
    lastMessage.setValue(ServerValue.timestamp())  
}
```

- Simple way to do it

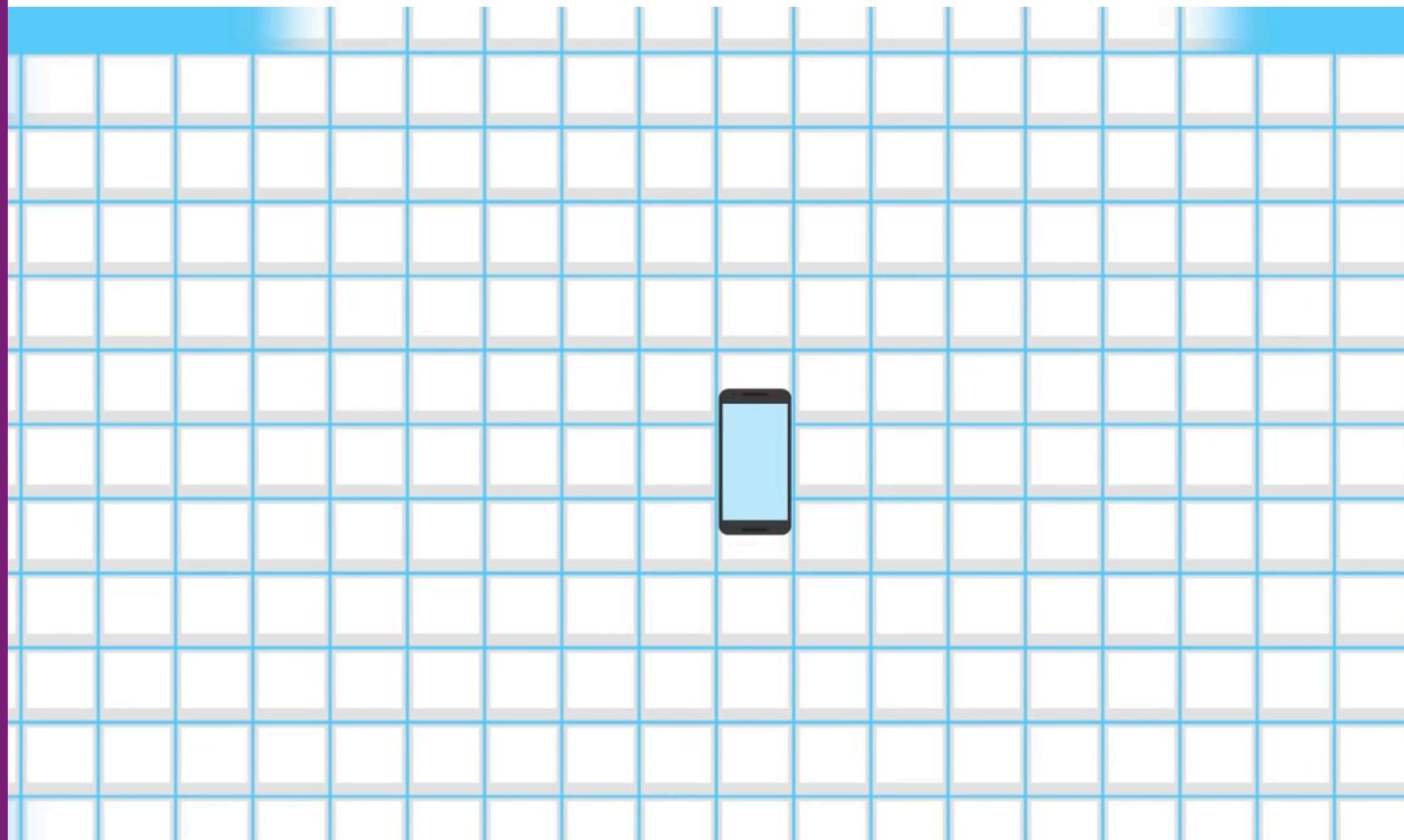
BREAK TIME



FIREBASE STORAGE

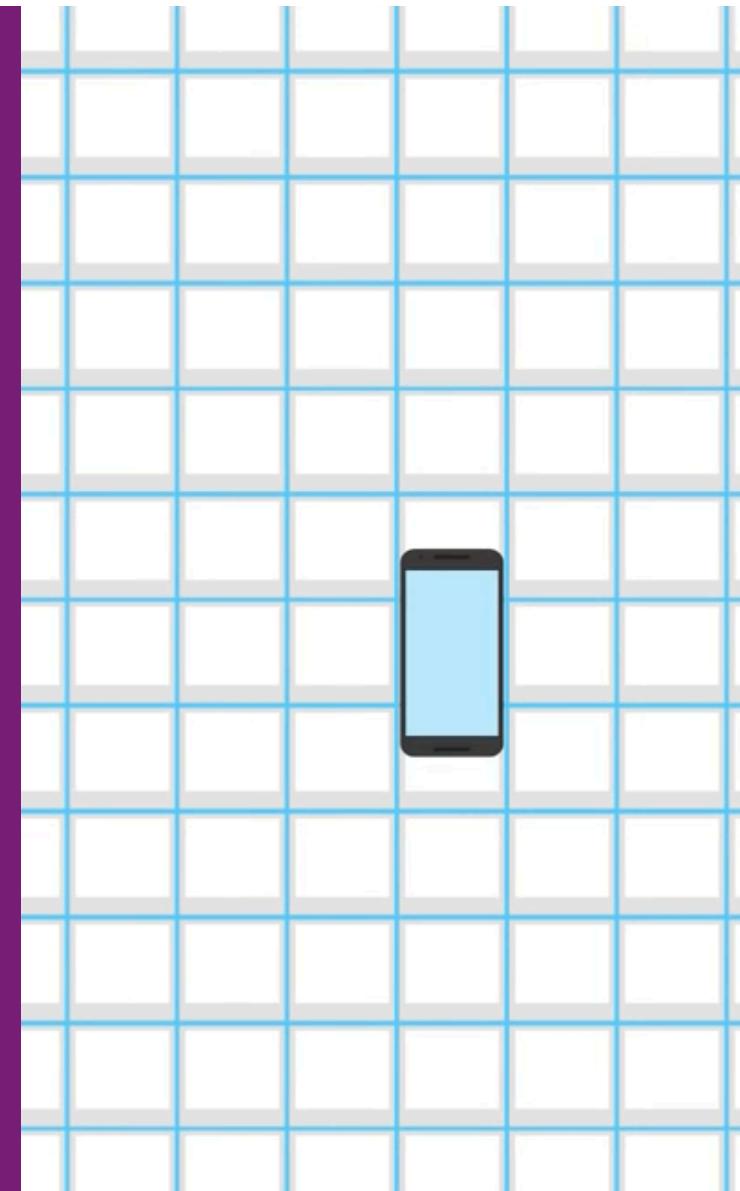
FIREBASE STORAGE

- [https://
www.youtube.co
m/watch?list=PLI-
K7zZEsvYLmOF_07
layrTntevxtbUxDL
&v=tyjqozrEPY](https://www.youtube.com/watch?list=PLIK7zZEsvYLmOF_07layrTntevxtbUxDL&v=tyjqozrEPY)



FIREBASE STORAGE

- Cloud Storage for Firebase stores files (blobs)
 - Images
 - Video
 - ...
- Storage is also accessible through Google Cloud Storage



FIREBASE STORAGE

1

Integrate the Firebase SDKs
for Cloud Storage.

Quickly include clients via Gradle, CocoaPods, or a script
include.

2

Create a Reference

Reference the path to a file, such as "images/mountains.png",
to upload, download, or delete it.

3

Upload or Download

Upload or download to native types in memory or on disk.

4

Secure your Files

Use [Firebase Security Rules for Cloud Storage](#) to secure your
files.

FIREBASE STORAGE

```
target 'chat-cat' do
  # Comment the next line if you don't want to use dynamic frameworks
  use_frameworks!

  # Pods for chat-cat

  # add the Firebase pod for Google Analytics
  pod 'Firebase/Analytics'

  # add pods for any other desired Firebase products
  pod 'Firebase/Database'
  pod 'Firebase/Auth'
  pod 'GoogleSignIn'
  pod 'FirebaseStorage'

# https://firebase.google.com/docs/ios/setup#available-pods
```



FIREBASE STORAGE

- Online console for storage

Storage



Store and retrieve user-generated images, audio, and video with side code

[Learn more](#)

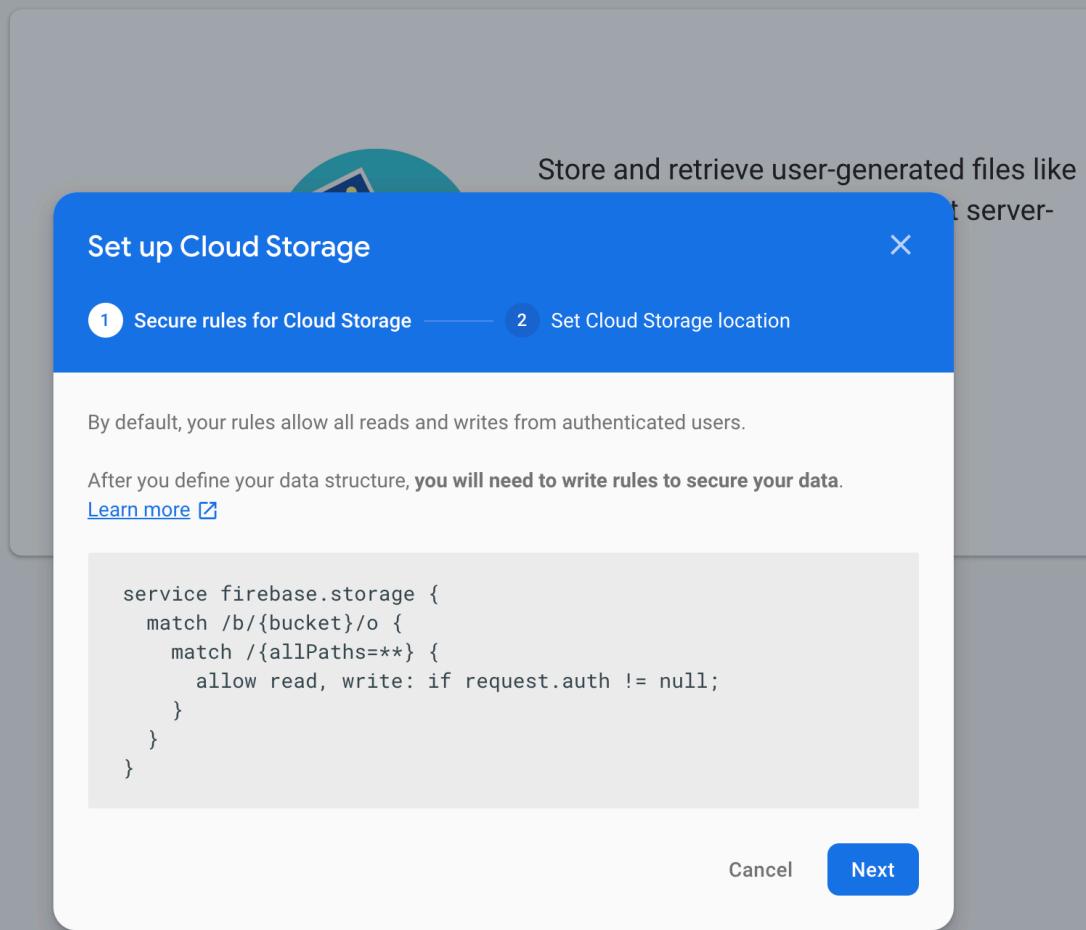
[View the docs](#)

[Get Started](#)

FIREBASE STORAGE

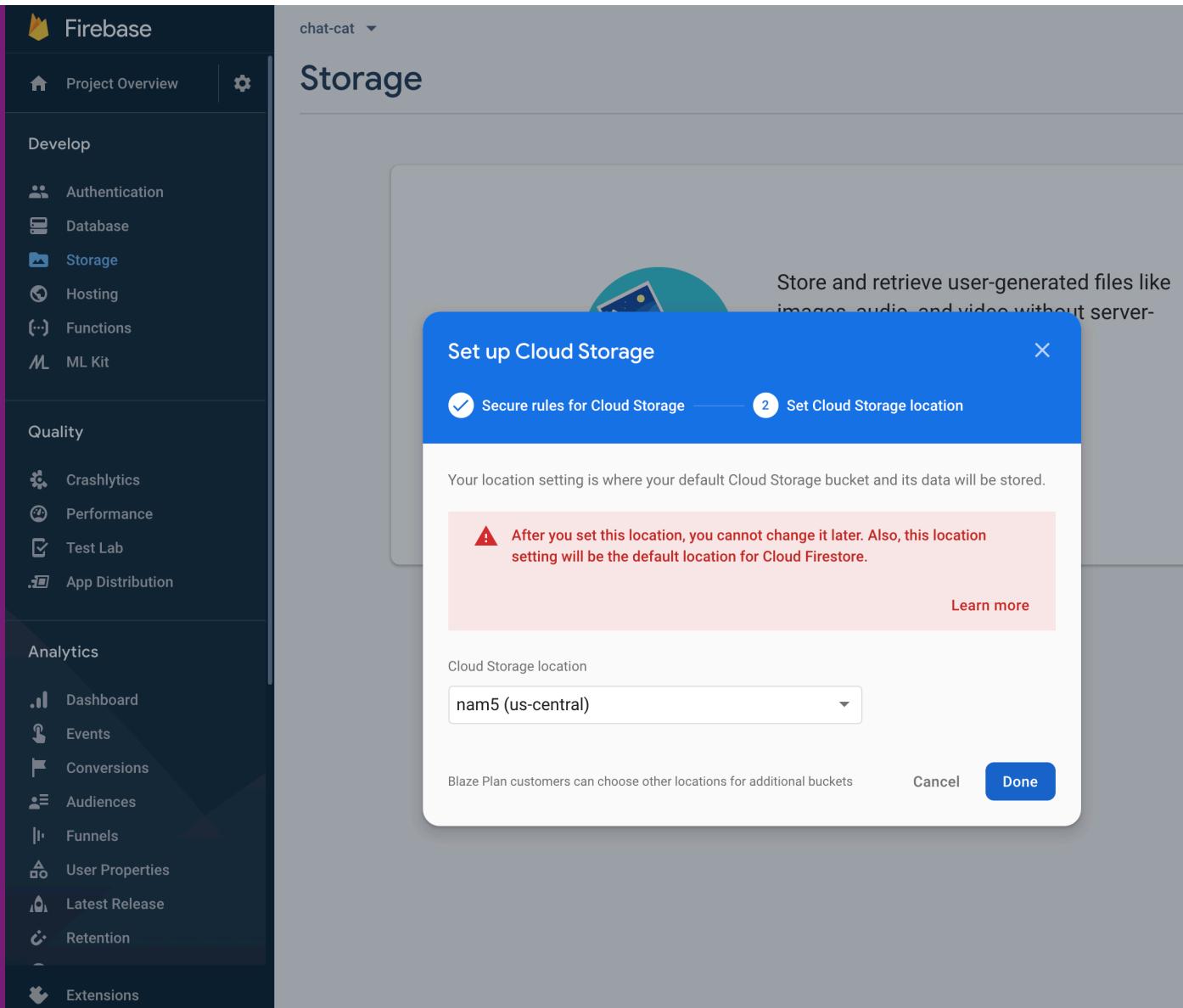
- Set permission rules

Storage



FIREBASE STORAGE

- Set location



The screenshot shows the Firebase console interface. On the left, a sidebar lists various services: Project Overview, Authentication, Database, Storage, Hosting, Functions, and ML Kit under 'Develop'; Crashlytics, Performance, Test Lab, and App Distribution under 'Quality'; and Dashboard, Events, Conversions, Audiences, Funnels, User Properties, Latest Release, Retention, and Extensions under 'Analytics'. The main area is titled 'Storage' and contains a large callout box for 'Set up Cloud Storage'. The box has two tabs at the top: 'Secure rules for Cloud Storage' (which is checked) and 'Set Cloud Storage location'. A note below says, 'Your location setting is where your default Cloud Storage bucket and its data will be stored.' A warning message in a red box states, 'After you set this location, you cannot change it later. Also, this location setting will be the default location for Cloud Firestore.' Below the note is a dropdown menu labeled 'Cloud Storage location' with 'nam5 (us-central)' selected. At the bottom right of the dialog are 'Cancel' and 'Done' buttons.

chat-cat ▾

Storage

Store and retrieve user-generated files like images, audio, and video without server-side code.

Set up Cloud Storage

Secure rules for Cloud Storage 2 Set Cloud Storage location

Your location setting is where your default Cloud Storage bucket and its data will be stored.

⚠ After you set this location, you cannot change it later. Also, this location setting will be the default location for Cloud Firestore.

Learn more

Cloud Storage location

nam5 (us-central)

Blaze Plan customers can choose other locations for additional buckets

Cancel Done

FIREBASE STORAGE

```
// Only authenticated users can read or write to the bucket
service firebase.storage {
  match /b/{bucket}/o {
    match /{allPaths=**} {
      allow read, write: if request.auth != null;
    }
  }
}
```

- Default - users only

FIREBASE STORAGE

```
// Anyone can read or write to the bucket, even non-users of your app.  
// Because it is shared with Google App Engine, this will also make  
// files uploaded via GAE public.  
service firebase.storage {  
  match /b/{bucket}/o {  
    match /{allPaths=**} {  
      allow read, write;  
    }  
  }  
}
```

- Public

FIREBASE STORAGE

```
// Grants a user access to a node matching their user ID
service firebase.storage {
  match /b/{bucket}/o {
    // Files look like: "user/<UID>/path/to/file.txt"
    match /user/{userId}/{allPaths=**} {
      allow read, write: if request.auth.uid == userId;
    }
  }
}
```

- User based storage

FIREBASE STORAGE

```
// Access to files through Firebase Storage is completely disallowed.  
// Files may still be accessible through Google App Engine or GCS APIs.  
service firebase.storage {  
    match /b/{bucket}/o {  
        match /{allPaths=**} {  
            allow read, write: if false;  
        }  
    }  
}
```

- Private
- Static images uploaded by developer

FIREBASE STORAGE

- References to our storage
- Data accessible by path

```
// Points to the root reference
let storageRef = Storage.storage().reference()

// Points to "images"
let imagesRef = storageRef.child("images")

// Points to "images/space.jpg"
// Note that you can use variables to create child values
let fileName = "space.jpg"
let spaceRef = imagesRef.child(fileName)

// File path is "images/space.jpg"
let path = spaceRef.fullPath;

// File name is "space.jpg"
let name = spaceRef.name;

// Points to "images"
let images = spaceRef.parent()
```

FIREBASE STORAGE

```
// Create a root reference
let storageRef = storage.reference()

// Create a reference to "mountains.jpg"
let mountainsRef = storageRef.child("mountains.jpg")

// Create a reference to 'images/mountains.jpg'
let mountainImagesRef = storageRef.child("images/mountains.jpg")

// While the file names are the same, the references point to different files
mountainsRef.name == mountainImagesRef.name;           // true
mountainsRef fullPath == mountainImagesRef fullPath; // false
```

- Create a reference node

UPLOAD

FIREBASE STORAGE

- Upload from image in memory

```
// Data in memory
let data = Data()

// Create a reference to the file you want to upload
let riversRef = storageRef.child("images/rivers.jpg")

// Upload the file to the path "images/rivers.jpg"
let uploadTask = riversRef.putData(data, metadata: nil) { (metadata, error) in
    guard let metadata = metadata else {
        // Uh-oh, an error occurred!
        return
    }
    // Metadata contains file metadata such as size, content-type.
    let size = metadata.size
    // You can also access to download URL after upload.
    riversRef.downloadURL { (url, error) in
        guard let downloadURL = url else {
            // Uh-oh, an error occurred!
            return
        }
    }
}
```

FIREBASE STORAGE

- Upload from image on disk

```
// File located on disk
let localFile = URL(string: "path/to/image")!

// Create a reference to the file you want to upload
let riversRef = storageRef.child("images/rivers.jpg")

// Upload the file to the path "images/rivers.jpg"
let uploadTask = riversRef.putFile(from: localFile, metadata: nil) { metadata, error in
    guard let metadata = metadata else {
        // Uh-oh, an error occurred!
        return
    }
    // Metadata contains file metadata such as size, content-type.
    let size = metadata.size
    // You can also access to download URL after upload.
    riversRef.downloadURL { (url, error) in
        guard let downloadURL = url else {
            // Uh-oh, an error occurred!
            return
        }
    }
}
```

FIREBASE STORAGE

```
func uploadFile() {  
    // Local file you want to upload  
    let kitten = UIImage(named:"Kitten")!  
    let data = kitten.pngData()!  
  
    let kittenRef = storageRef.child("images/kitten.jpg")  
  
    // Upload the file to the path "images/rivers.jpg"  
    let _ = kittenRef.putData(data, metadata: nil) { (metadata, error) in  
        guard metadata != nil else {  
            // Uh-oh, an error occurred!  
            return  
        }  
    }  
}
```

FIREBASE STORAGE

- Default metadata is taken from the image
- You can override it

```
// Create storage reference
let mountainsRef = storageRef.child("images/mountains.jpg")

// Create file metadata including the content type
let metadata = StorageMetadata()
metadata.contentType = "image/jpeg"

// Upload data and metadata
mountainsRef.putData(data, metadata)

// Upload file and metadata
mountainsRef.putFile(from: localFile, metadata: metadata)
```

FIREBASE STORAGE

```
// Add a progress observer to an upload task
let observer = uploadTask.observe(.progress) { snapshot in
    // A progress event occurred
}
```

- Observe progress on a file upload

FIREBASE STORAGE

```
// Start uploading a file
let uploadTask = storageRef.putFile(from: localFile)

// Pause the upload
uploadTask.pause()

// Resume the upload
uploadTask.resume()

// Cancel the upload
uploadTask.cancel()
```

- Manage uploads

FIREBASE STORAGE

```
// Local file you want to upload
let localFile = URL(string: "path/to/image")!

// Create the file metadata
let metadata = FIRStorageMetadata()
metadata.contentType = "image/jpeg"

// Upload file and metadata to the object 'images/mountains.jpg'
let uploadTask = storageRef.putFile(localFile, metadata: metadata)

// Listen for state changes, errors, and completion of the upload.
uploadTask.observe(.resume) { snapshot in
    // Upload resumed, also fires when the upload starts
}

uploadTask.observe(.pause) { snapshot in
    // Upload paused
}

uploadTask.observe(.progress) { snapshot in
    // Upload reported progress
    let percentComplete = 100.0 * Double(snapshot.progress!.completedUnitCount)
        / Double(snapshot.progress!.totalUnitCount)
}
```



UPLOAD AND OBSERVE

FIREBASE STORAGE

- Full example, with all the trimmings

```
// Create the file metadata
let metadata = StorageMetadata()
metadata.contentType = "image/jpeg"

// Upload file and metadata to the object 'images/mountains.jpg'
let uploadTask = storageRef.putFile(from: localFile, metadata: metadata)

// Listen for state changes, errors, and completion of the upload.
uploadTask.observe(.resume) { snapshot in
    // Upload resumed, also fires when the upload starts
}

uploadTask.observe(.pause) { snapshot in
    // Upload paused
}

uploadTask.observe(.progress) { snapshot in
    // Upload reported progress
    let percentComplete = 100.0 * Double(snapshot.progress!.completedUnitCount)
        / Double(snapshot.progress!.totalUnitCount)
}

uploadTask.observe(.success) { snapshot in
    // Upload completed successfully
}

uploadTask.observe(.failure) { snapshot in
    if let error = snapshot.error as? NSError {
        switch (StorageErrorCode(rawValue: error.code)!) {
        case .objectNotFound:
            // File doesn't exist
            break
        case .unauthorized:
            // User doesn't have permission to access file
            break
        case .cancelled:
            // User canceled the upload
            break

            /* ... */

        case .unknown:
            // Unknown error occurred, inspect the server response
            break
        default:
            // A separate error occurred. This is a good place to retry the upload.
            break
        }
    }
}
```

DOWNLOAD

FIREBASE STORAGE

- Once you have a reference, you can download files from Cloud Storage in three ways:
 - Download to Data in memory
 - Download to an URL representing a file on device
 - Generate an URL representing the file online

FIREBASE STORAGE

```
// Create a reference with an initial file path and name
let pathReference = storage.reference(withPath: "images/stars.jpg")

// Create a reference from a Google Cloud Storage URI
let gsReference = storage.reference(forURL: "gs://<your-firebase-storage-bucket>/images/stars.jpg")

// Create a reference from an HTTPS URL
// Note that in the URL, characters are URL escaped!
let httpsReference = storage.reference(forURL: "https://firebasestorage.googleapis.com/b/<your-firebase-storage-bucket>/o/?alt=media&token=<image-token>")
```

- Files are accessible through different APIs

FIREBASE STORAGE

- Files are accessible through different APIs and locations

The screenshot shows the Firebase Storage console interface. At the top, there's a header with the URL "gs://firechat-66f87.appspot.com" followed by a "images" folder indicator. To the right of the URL is a blue "UPLOAD FILE" button with a white arrow icon, and a small "+" icon. Below the header is a table with four columns: "Name", "Size", "Type", and "Last modified". A single file, "kitten.jpg", is listed. The details for "kitten.jpg" are shown in a large panel on the right side of the table row. The file information includes:

Name	Size	Type	Last modified
kitten.jpg	1,0...	application/octet-stream	Oct 16, 2017, 7:50:48 PM

kitten.jpg

Name
kitten.jpg

Size
1,023.14 KB

Type
application/octet-stream

Created
Oct 16, 2017, 7:50:48 PM

Updated
Oct 16, 2017, 7:50:48 PM

File location

Storage location
gs://firechat-66f87.appspot.com/images/kitten.jpg

Download URL 1 [revoke](#)
<https://firebasestorage.googleapis.com/storage/v1/b/firechat-66f87.appspot.com/o/images%2Fkitten.jpg?alt=media&token=9bf1-7603b7722680>

[Create new download URL](#)

Other metadata

No metadata found

FIREBASE STORAGE

- Files are accessible through different APIs
- Download to memory

```
// Create a reference to the file you want to download
let islandRef = storageRef.child("images/island.jpg")

// Download in memory with a maximum allowed size of 1MB (1 * 1024 * 1024 bytes)
islandRef.getData(maxSize: 1 * 1024 * 1024) { data, error in
    if let error = error {
        // Uh-oh, an error occurred!
    } else {
        // Data for "images/island.jpg" is returned
        let image = UIImage(data: data!)
    }
}
```

FIREBASE STORAGE

- Download to file on disk

```
// Create a reference to the file you want to download
let islandRef = storageRef.child("images/island.jpg")

// Create local filesystem URL
let localURL = URL(string: "path/to/image")!

// Download to the local filesystem
let downloadTask = islandRef.write(toFile: localURL) { url, error in
    if let error = error {
        // Uh-oh, an error occurred!
    } else {
        // Local file URL for "images/island.jpg" is returned
    }
}
```

FIREBASE STORAGE

- Why work so hard?
- FirebaseUI (part of Storage)
will download and cache
 - Based on SDWebImage



build passing pod v4.0.0 platform osx | ios | tvos | watchos license MIT dependencies up to date references codecov 76%

This library provides an async image downloader with cache support. For convenience, we added elements like `UIImageView`, `UIButton`, `MKAnnotationView`.

Features

- Categories for `UIImageView`, `UIButton`, `MKAnnotationView` adding web image and cache management
- An asynchronous image downloader
- An asynchronous memory + disk image caching with automatic cache expiration handling
- A background image decompression
- A guarantee that the same URL won't be downloaded several times
- A guarantee that bogus URLs won't be retried again and again
- A guarantee that main thread will never be blocked
- Performances!
- Use GCD and ARC

Supported Image Formats

- Image formats supported by `UIImage` (JPEG, PNG, ...), including GIF
- WebP format, including animated WebP (use the `WebP` subspec)

Requirements

FIREBASE STORAGE

```
# add the Firebase pod for Google Analytics
pod 'Firebase/Analytics'

# add pods for any other desired Firebase products
pod 'Firebase/Database'
pod 'Firebase/Auth'
pod 'GoogleSignIn'
pod 'FirebaseStorage'

pod 'FirebaseUI/Storage'
pod 'SDWebImage'

end
```

FIREBASE STORAGE

```
import UIKit
import Firebase
import FirebaseStorage
import FirebaseUI
```

```
override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
    let cell = tableView.dequeueReusableCell(withIdentifier: "Cell", for: indexPath)

    // Configure the cell...
    let item = messages[indexPath.row]
    cell.textLabel?.text = item.text
    cell.detailTextLabel?.text = item.name

    let reference = storageRef.child("images/kitten.jpg")

    // Placeholder image
    let placeholderImage = UIImage(named: "Placeholder")

    // Load the image using SDWebImage
    cell.imageView?.sd_setImage(with: reference, placeholderImage: placeholderImage)

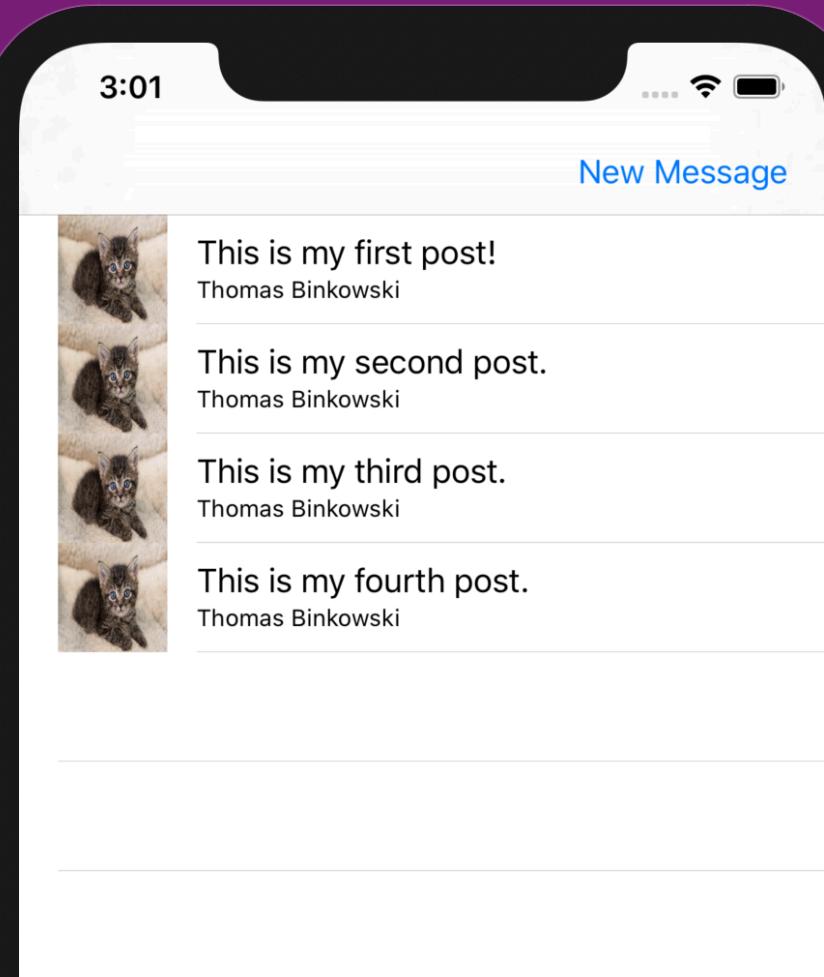
    return cell
}
```

- Download directly to UIImage; swap placeholder image out

FIREBASE STORAGE

```
override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {  
    let cell = tableView.dequeueReusableCell(withIdentifier: "MessageCell", for: indexPath)  
  
    // Configure the cell...  
    let item = messages[indexPath.item]  
    cell.textLabel?.text = item.message  
    cell.detailTextLabel?.text = item.sender  
  
    let reference = storageReference.child("kittens").child(item.id)  
  
    // Placeholder image  
    let placeholderImage = UIImage(named: "placeholder")  
  
    // Load the image using SDWebImage  
    cell.imageView?.sd_setImage(with: reference, placeholderImage: placeholderImage)  
  
    return cell  
}
```

- Download directly to the device



```
import UIKit  
import Firebase  
import FirebaseStorage  
import FirebaseUI
```

```
lexPath) -> UITableViewCell {  
    let cell = tableView.dequeueReusableCell(withIdentifier: "MessageCell", for: indexPath)
```

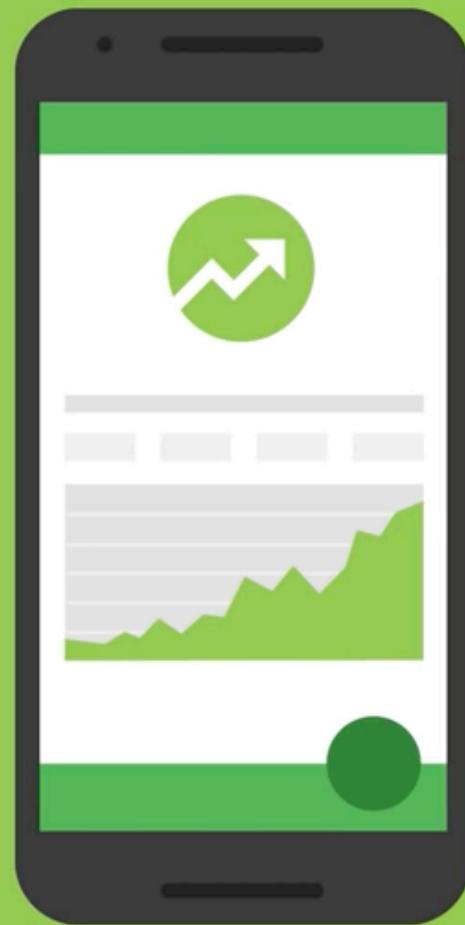
```
image)
```

```
out
```

FIREBASE ANALYTICS

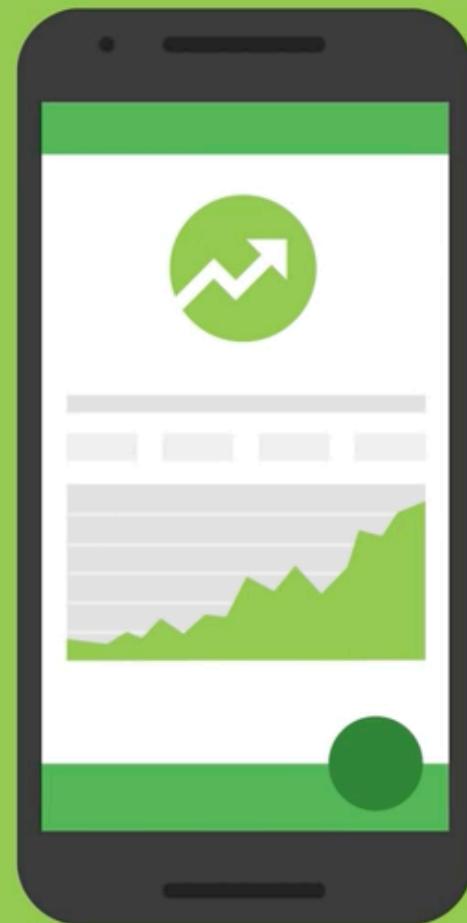
FIREBASE ANALYTICS

- [https://youtu.be/
iT6EalwtonY?
list=PLI-
K7zZEesYLmOF_07
layrTntevxtbUxDL](https://youtu.be/iT6EalwtonY?list=PLI-K7zZEesYLmOF_07layrTntevxtbUxDL)



FIREBASE ANALYTICS

- Firebase Analytics is a free app measurement solution that provides insight on app usage and user engagement
- Installed in Core



FIREBASE ANALYTICS

- Integrate with other services

BigQuery

Link your Firebase Analytics app to BigQuery where you can perform custom analysis on your entire Analytics dataset and import other data sources.

Firebase Crash Reporting

Firebase Analytics logs events for each crash so you can get a sense of the rate of crashes for different versions or regions, allowing you to gain insight into which users are impacted. You can also create audiences for users who have experienced multiple crashes and respond with Firebase Notifications directed at that audience.

Firebase Notifications

Firebase Analytics automatically logs events that correspond to your Firebase Notifications and supports reporting on the impact of each campaign.

Firebase Remote Config

Use Firebase Analytics audience definitions to change the behavior and appearance of your app for different audiences without distributing multiple versions of your app.

Google Tag Manager

Integrating [Google Tag Manager](#) alongside Firebase Analytics enables you to manage your Firebase Analytics implementation remotely from a web interface after your app has been distributed.

FIREBASE ANALYTICS

- Key data is pulled from authentication
- Consider what you want to know

- 1 Connect your app to Firebase

Getting started with Analytics is easy. Just add the Firebase SDK to your new or existing app, and data collection begins automatically. You can view analytics data in the Firebase console within hours.
- 2 Log custom data

You can use Analytics to log custom events that make sense for your app, like E-Commerce purchases or achievements.
- 3 Create audiences

You can define the audiences that matter to you in the Firebase console.
- 4 Target audiences

Use your custom audiences to target messages, promotions, or new app features using other Firebase features, such as Notifications, and Remote Config.

FIREBASE ANALYTICS

- Automatically tracked events

Event name	Triggered...
first_open	<p>the first time a user launches an app after installing or re-installing it.</p> <p>This event is not triggered when a user downloads the app onto a device, but instead when he or she first uses it. To see raw download numbers, look in Google Play Developer Console or in iTunesConnect.</p>
in_app_purchase	<p>when a user completes an in-app purchase that is processed by the App Store or iTunes or Google Play. The product ID, product name, currency, and quantity are passed as parameters.</p> <p>This event is triggered only by versions of your app that include the Firebase SDK. Also, subscription revenue, paid app-purchase revenue, and refunds are not automatically tracked, and so your reported revenue may differ from the values you see in the Google Play Developer Console. Events which are flagged as being invalid or sandbox (test) are ignored.</p>
user_engagement	periodically, while the app is in the foreground.
session_start	when a user engages the app for more than the minimum session duration after a period of inactivity that exceeds the session timeout duration .

FIREBASE ANALYTICS

- New Pod

```
target 'chat-cat' do
  # Comment the next line if you don't want to use dynamic frameworks!
  #use_frameworks!

  # Pods for chat-cat

  # add the Firebase pod for Google Analytics
  pod 'Firebase/Analytics'

  # add pods for any other desired Firebase products
  pod 'Firebase/Database'
  pod 'Firebase/Auth'
  pod 'GoogleSignIn'
  pod 'FirebaseStorage'

  pod 'FirebaseUI/Storage'
  pod 'SDWebImage'

end
```

FIREBASE ANALYTICS

- Predefined events

Dashboard

We have changed the way we track sessions, which may impact your session-related metrics. [Learn more](#)



Last
→ ← C

Events

Parameter Reporting

Recommendations [?](#)

sign_up [Recommended Event \(General\)](#)

login [Recommended Event \(General\)](#)

share [Recommended Event \(General\)](#)

Search...

Event name ↑

Count

↓↑

Users

↓↑

first_open

1

1

screen_view

4

1

session_start

3

1

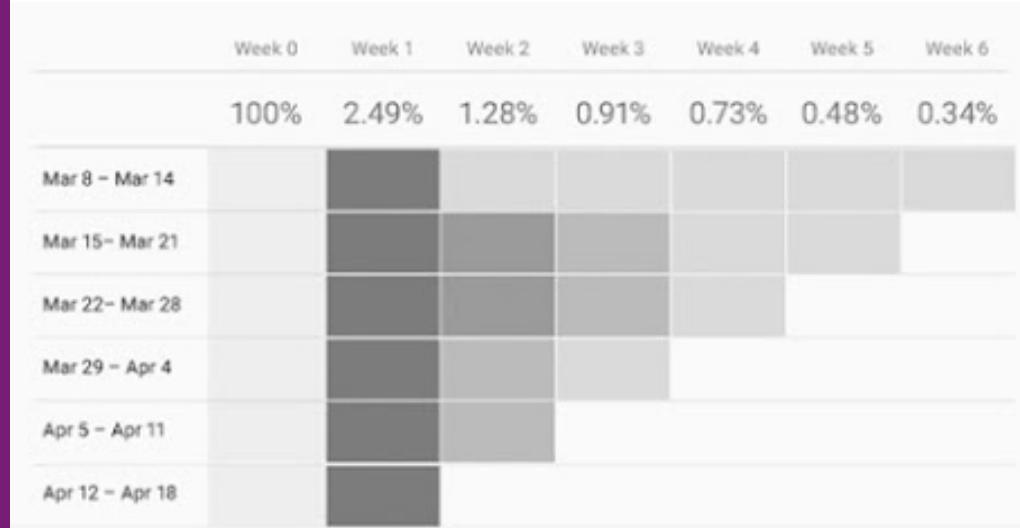


Send your raw events to BigQuery. [LEARN MORE](#)

[link to bigquery](#)

FIREBASE ANALYTICS

- Active users
- Average revenue, IAP
- first_open, first_open attribution
- Retention cohort
- User engagement (daily, per user, sessions, avg. session duration)
- Device and App info
- Location
- Demographics
- Screen views
- ...much much more



FIREBASE ANALYTICS

```
Analytics.logEvent(AnalyticsEventSelectContent, parameters: [  
    AnalyticsParameterItemID: "id-\\" + title! + "\",  
    AnalyticsParameterItemName: title!,  
    AnalyticsParameterContentType: "cont"  
])
```

- Custom events

FIREBASE ANALYTICS

```
analytics.logEvent('select_content', {  
  content_type: 'image',  
  content_id: 'P12453',  
  items: [{ name: 'Kittens' }]  
});
```

- Custom events

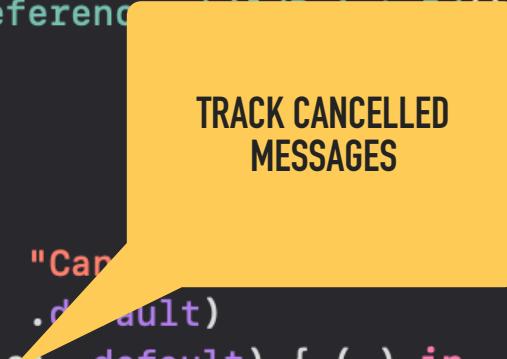
FIREBASE ANALYTICS

```
        guard let textField = alert.textFields?.first,
              let text = textField.text else {
            return
        }

        let data = ["name":self.user?.displayName,"text": text]
        self.reference.child("messages").setValue(data)

    }

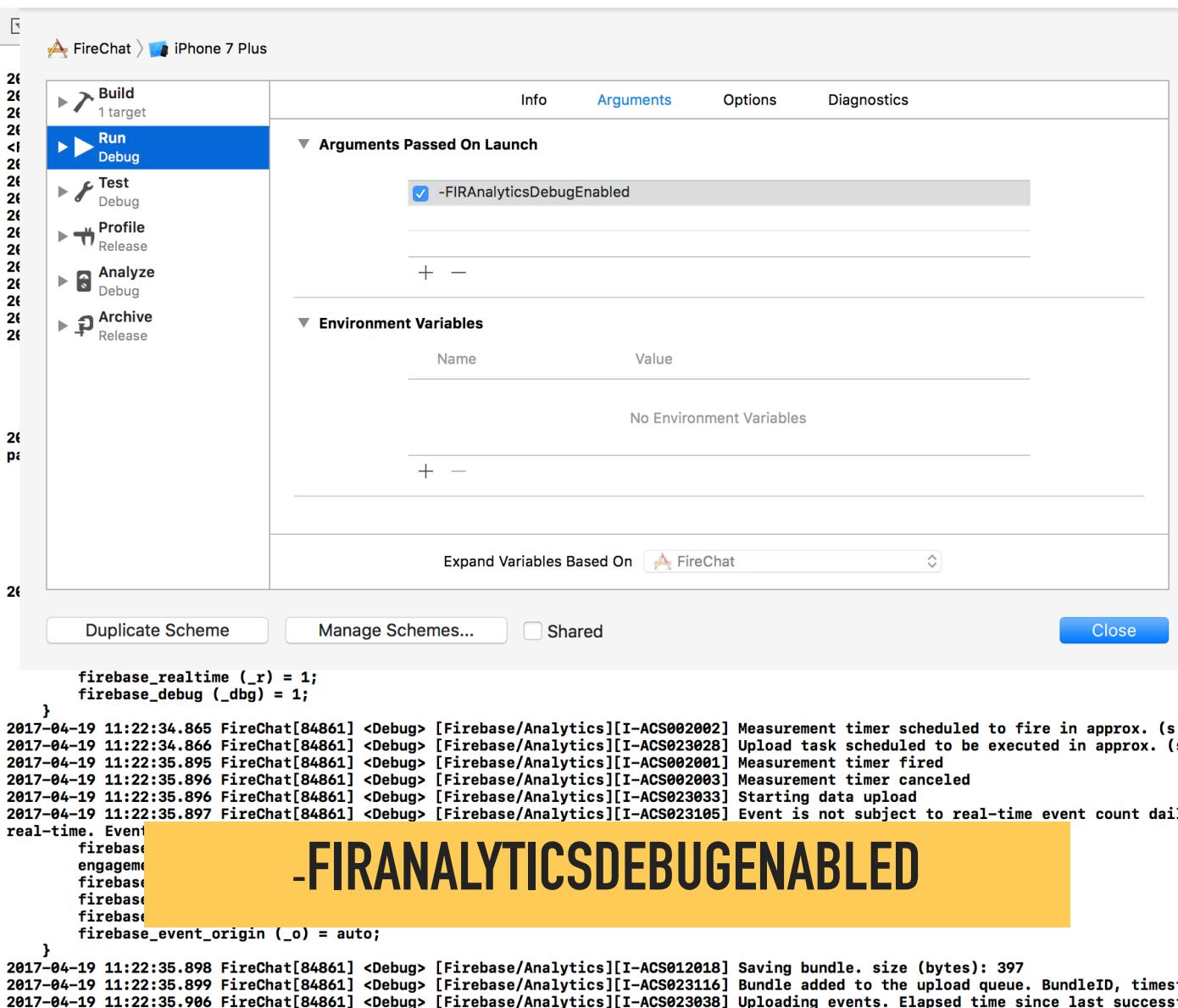
let cancelAction = UIAlertAction(title: "Cancel",
                                style: .default)
let _ = UIAlertAction(title: "Cancel", style: .default) { (_) in
    Analytics.logEvent("message-cancelled", parameters: nil)
}
alert.addTextField()
```



TRACK CANCELLED
MESSAGES

FIREBASE ANALYTICS

- View login in the console
 - Products > Scheme
 - Passed on launch



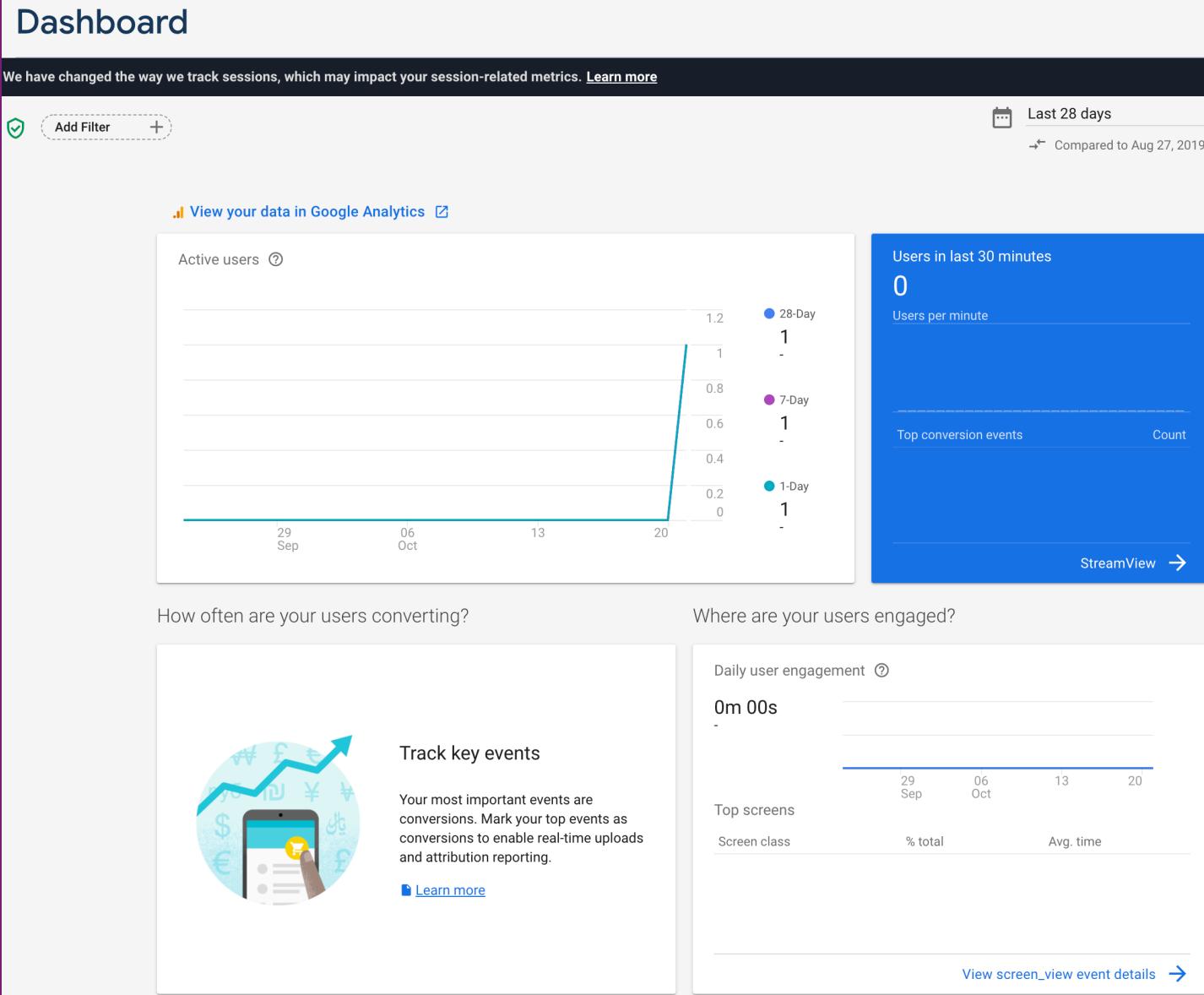
FIREBASE ANALYTICS

```
2017-04-19 11:22:20.771 FireChat[84861] <Debug> [Firebase/Analytics][I-ACS012018] Saving bundle. size (bytes): 392
2017-04-19 11:22:20.771 FireChat[84861] <Debug> [Firebase/Analytics][I-ACS023116] Bundle added to the upload queue. BundleID, timestamp (ms): 4, 1492618939686
2017-04-19 11:22:20.779 FireChat[84861] <Debug> [Firebase/Analytics][I-ACS023038] Uploading events. Elapsed time since last successful upload (s): 1.4290108680725
2017-04-19 11:22:20.780 FireChat[84861] <Debug> [Firebase/Analytics][I-ACS023039] Measurement data sent to network. Timestamp (ms), data: 1492618940779,
<FIRAPBMeasurementBatch: 0x6000000a2e0>
2017-04-19 11:22:20.782 FireChat[84861] <Debug> [Firebase/Analytics][I-ACS900000] Uploading data. Host: https://app-measurement.com/a
2017-04-19 11:22:20.908 FireChat[84861] <Debug> [Firebase/Analytics][I-ACS901006] Received SSL challenge for host. Host: https://app-measurement.com/a
2017-04-19 11:22:21.039 FireChat[84861] <Debug> [Firebase/Analytics][I-ACS023044] Successful upload. Got network response. Code, size: 204, -1
2017-04-19 11:22:21.041 FireChat[84861] <Debug> [Firebase/Analytics][I-ACS002002] Measurement timer scheduled to fire in approx. (s): -0.3548879027366638
2017-04-19 11:22:21.041 FireChat[84861] <Debug> [Firebase/Analytics][I-ACS023028] Upload task scheduled to be executed in approx. (s): -0.3548879027366638
2017-04-19 11:22:21.048 FireChat[84861] <Debug> [Firebase/Analytics][I-ACS023024] No data to upload. Upload task will not be scheduled
2017-04-19 11:22:21.048 FireChat[84861] <Debug> [Firebase/Analytics][I-ACS002003] Measurement timer canceled
2017-04-19 11:22:34.853 FireChat[84861] <Debug> [Firebase/Analytics][I-ACS033003] Scheduling user engagement timer
2017-04-19 11:22:34.853 FireChat[84861] <Debug> [Firebase/Analytics][I-ACS002003] Engagement timer canceled
2017-04-19 11:22:34.854 FireChat[84861] <Debug> [Firebase/Analytics][I-ACS002002] Engagement timer scheduled to fire in approx. (s): 3600
2017-04-19 11:22:34.854 FireChat[84861] <Debug> [Firebase/Analytics][I-ACS023051] Logging event: origin, name, params: auto, user_engagement (_e), {
    firebase_screen_id (_si) = 7128337720279460482;
    engagement_time_msec (_et) = 15878;
    firebase_screen_class (_sc) = FireChat.MessagesTableViewController;
    firebase_event_origin (_o) = auto;
}
2017-04-19 11:22:34.854 FireChat[84861] <Debug> [Firebase/Analytics][I-ACS023073] Debug mode is enabled. Marking event as debug and real-time. Event name
parameters: user_engagement (_e), {
    firebase_screen_id (_si) = 7128337720279460482;
    engagement_time_msec (_et) = 15878;
    firebase_screen_class (_sc) = FireChat.MessagesTableViewController;
    firebase_event_origin (_o) = auto;
    firebase_realtime (_r) = 1;
    firebase_debug (_dbg) = 1;
}
2017-04-19 11:22:34.864 FireChat[84861] <Debug> [Firebase/Analytics][I-ACS023072] Event logged. Event name, event params: user_en
```

IT'S A LITTLE CREEPY

FIREBASE ANALYTICS

- Data is accessed in the online console
- Time hour delay



FIREBASE ANALYTICS

```
firebase.analytics().setUserProperties({favorite_food: 'apples'});
```

- Add custom properties to the `user` for analytics analysis later

ASSIGNMENT 4

ASSIGNMENT 3

- Part 1
 - Firebase tutorials
- Part 2
 - Develop a group messaging application using Firebase

PART 1

ASSIGNMENT 3 - PART 1

- Ray Weinerlich Tutorial to build a grocery app
- <https://www.raywenderlich.com/139322/firebase-tutorial-getting-started-2>

 raywenderlich.com Tutorials ▾ Videos ▾ Podcast Forums Store LOGIN / SIGN UP

Firebase Tutorial: Getting Started

 Attila Hegedüs on September 19, 2016 [Tweet](#) [Like](#)

Update note: This tutorial has been updated for iOS 10 and Swift 3 by Attila Hegedüs. The original tutorial was written by David East.

Firebase is a mobile-backend-as-a-service that provides several features for building powerful mobile apps. Firebase has three core services: a realtime database, user authentication and hosting. With the [Firebase iOS SDK](#), you can use these services to build powerful apps without writing a single line of server code.

The realtime database is one of the most unique features of [Firebase](#).

Ever used pull-to-refresh to fetch new data? You can forget about it with Firebase.

When a Firebase database updates, all connected users receive updates in realtime. This means your app can stay up to date without user interaction.

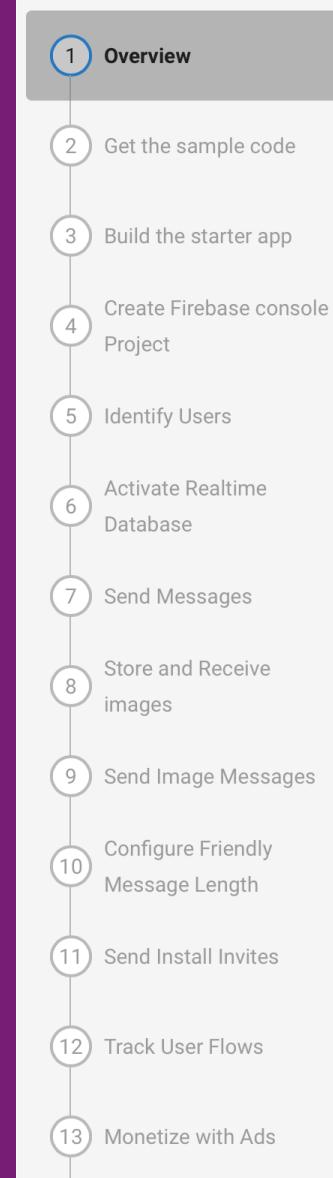
In this Firebase tutorial, you'll learn all the fundamentals of Firebase by making a collaborative grocery list app called [Grocr](#). When items get added to the list they'll appear instantly on any user's devices, you're not going to stop there. You'll tweak [Grocr](#) to work offline, so the list stays in sync even with a spotty grocery store data connection.



Learn the fundamentals in the [Firebase tutorial](#).

ASSIGNMENT 3 - PART 1

- Firebase codelab tutorial to build a (very sparse) chat application
- <https://codelabs.developers.google.com/codelabs/firebase-ios-swift/#0>



← Firebase iOS Codelab Swift

1. Overview

Friendly Chat Crash Invite Fresh Config SIGN OUT

Ibrahim Ulukaya: hey
sent by: Ibrahim Ulukaya

Ibrahim Ulukaya: nice
sent by: Ibrahim Ulukaya

To: Send

AdMob Test Smart Banner
developers.google.com/codelab
monetization-tutorial.html

Welcome to the Friendly Chat codelab. In this codelab, you'll learn to... applications. You will implement a chat client and monitor its performance.

This codelab is also available in [Objective-C](#).

What you'll learn

✓ Sync data using the Firebase Realtime Database.

PART 2

NEXT WEEK TOPICS

- Cloud Firestore
- Cloud functions
- Firebase Cloud Messaging
- Remote config
- App Indexing
- Dynamic links
- Admob
- Invites
- Ad words
- Notification

IF YOU WANT TO GET A PAID DEVELOPER ACCOUNT, THIS IS THE WEEK. THIS WILL ALLOW YOU TO SEND PUSH NOTIFICATIONS.

THIS IS OPTIONAL.

ASSIGNMENT 3 - PART 2

- Build a group messaging application using Firebase (yes the world needs another one)



ASSIGNMENT 3 - PART 2

- Authenticated users will be able to send messages to other authenticated users
- The messages can be text or images
- Users will invite other users to a chat using Firebase Invites (next week) and utilizing Dynamic links
- Notifications when someone joins a group



ASSIGNMENT 3 - PART 2

- The application interface can be "sparse" as long as it supports the functionality





THE UNIVERSITY OF
CHICAGO



MPCS 51033 • AUTUMN 2019 • SESSION 4

BACKENDS FOR MOBILE APPLICATIONS