**Advantages of Using Firebase For App Development**

Firebase the latest platform which has effective tools to develop Mobile and Web Applications. It is founded by Andrew Lee and James Tamplin in September 2011. It is the fully featured platform for App Development. It is build to help developers to share various features between cross-platform apps related Database, Config, and Notifications. You can perform various complex operation easily with firebase to build high-quality and bug-free apps for Android, iOS, and The Web. With Firebase, you can integrate authentication functionality in application, create storage functionality for an app to collect and store pictures, audio, and video without using server-side code.

Here are some advantages of using Firebase for App Development Project:

**Real-time Database:**

Firebase has real-time and cloud-based database where you can store data is JSON and synchronized continuously to all connected clients. If you want to create an Android, iOS, or Web app which provides real-time updates to users without creating Database or API then you should use firebase. It is capabilities to manage backend components of applications. The real-time database is the foremost advantage of the Firebase.

**Authentication:**

There are many apps that need identification of a user to save user data in the cloud to provide an enhanced experience on various devices. Firebase provides instant UI Libraries and SDKs for authenticating client across application using email id, password, or username. Here you can integrate various sign-in techniques to allow user login into your app.

**Hosting:**

Firebase provides fast, secure, static, and production-grade hosting for developers. It allows developers to efficiently deploy web apps and static content to a CDN(Content Delivery Network).

**Storage:**

It is another best advantage of Firebase. It is very useful when you want to create an app for storing and serving files of users such and images and videos.

**Notifications:**

Firebase notification is a free service which allows targeted user notifications for mobile app developers. It has the notification console GUI where you can create and send notifications to targeted users.

**App Indexing**

This feature is used to index application in Google search results. After app indexing, If a user search related to your app, it will start the app installed in user's device directly from the search result.

**AdMob:**

Admob is advertising facility of the Firebase which is used to generate profits from your app. You can also use the Firebase Analytics to measure app usage in order to create advertising strategies.

# **Why you should go with Firebase: Advantages and Disadvantages**

Firebase a cloud service provider now is under Google. Today we will discuss why you should go with Firebase for creating apps quickly.  
Firebase is not an ordinary database only. It is a realtime scalable back end and with it you can build apps that can easily serve millions of people across the globe.  
The services provided by firebase can be seen in the below image.

# **14 Benefits of using Firebase**

Building an application or website may seem to be a very exhaustive process at first sight involving investment of significant time and cost. However, there are many services available which can facilitate the development work in various ways. Firebase is one of them. This giant is ready to take away a significant part of the burden from the developer’s shoulders.

[Firebase](http://firebase.google.com/) was established by Andrew Lee and James Tamplin back in 2011 yet was launched formally in April 2012. Initially, the framework was expected to be a real-time database giving its APIs, enabling users to store and synchronize information across various customers. However, the plans changed a little bit when Firebase was taken over by Google two years after its release. Today, the service has various functions that a wise entrepreneur may find exceptionally valuable. Firebase has reached from 110,000 developers when it was initially obtained by Google in October of 2014 to a booming number of 450,000 plus developers at present.

Firebase is a framework which is help for building portable and web application for your business with real-time database which implies when one user updates a record in the database, that update would be conveyed to every single user, be those users on a website, iOS or Android device. It gives a basic and unified platform with so many Google features packed-in. You don’t need to configure your server when you use Firebase. Everything will be taken care of by Firebase automatically.

There are numerous elements that make working with Firebase marvelous from a developer’s point of view, that pertain to the core technology of development. This helps in maintaining the state of harmony between the developer & the client by causing minimal delay of work.

## Benefits of using Firebase

**Real-time Database**

Real-time Database is a cloud-hosted database. Data is stored as JSON and synchronized continuously to each associated client. When you build cross-platform applications with iOS, Android, and JavaScript SDKs, the greater part of your customers’ demand is based on one Real-time Database instance and consequently getting updates with the most current data. By utilising this feature of Firebase, there is no necessity to make your own database or own API, Firebase handles all the components that usually come along with creating a backend for applications. It gives an adaptable, expression-based rules language to define how your data should be organized and when information can be perused from or composed to.

**Hosting**

Hosting is production-grade web content that facilities the developers. With Hosting, you can rapidly and effectively send web applications and static content to a [Content Delivery Network](http://www.cmarix.com/8-Benefits-of-Content-Delivery-Network) (CDN) with a single command. It is very easy process in Firebase because it contains Custom domain support, Global CDN and Auto Provisioned SSL Certificate ideas for that. Whether you are sending a simple application landing page or a complex Progressive Web App. Hosting gives you the infrastructure, features, and tooling tailored to convey and manage static websites.

**Authentication**

Firebase Authentication gives backend services, simple to-use SDKs, and instant UI libraries to confirm clients over your application. It supports authentication using passwords, email id or username.

You can allow users to sign in to your Firebase app either by using FirebaseUI as a complete drop-in authentication solution or by using the Firebase Authentication SDK to manually integrate one or a few sign-in techniques into your application.

**Storage**

It is built for application developers who need to store and serve user-generated content, for example photos or videos. It gives secure document transfers and downloads for Firebase applications, regardless of network quality. You can utilize it to store pictures, sound, video, or other user-generated content. Firebase Storage is upheld by Google Cloud Storage, a capable, basic, and cost-effective object storage service.

**Cloud Messaging**

It is a cross-platform messaging solution that lets you dependably convey messages at zero expense. You can inform a customer that new email or other information is accessible to sync. You can send notification messages to drive user reengagement and maintenance.

**Remote Config**

It is a cloud service that gives you a chance to change the conduct and appearance of your application without requiring users to download an application update. Your application controls when updates are applied, and it can as often as possible check for updates and apply them with a negligible effect on execution.

**Test Lab**

Test Lab is utilized for testing your application on gadgets hosted in a Google data-center. It helps you to find issues that only happen on particular gadget configurations. A test result includes logs, videos, and screenshots which are available in your project in the Firebase console. Even if you haven’t composed any test code for your application, Test Lab can practice your application consequently, looking for crashes.

**Crash Reporting**

It helps to create detailed reports of the errors which are assembled into groups of comparative stack flow triaged by the severity of effect on users. In addition to automatic reports, you can register custom events to help capture the steps which leads to a crash.

**Notifications**

It is a free service of Firebase which empowers focused user notifications for mobile application developers. It gives a choice to developers and organizations looking for an adaptable notification platform which requires minimal coding effort to begin, and a graphical console for sending messages.

**App Indexing**

By using this component you easily get to index your application in Google Search. For an instance, if your application is already installed in user’s device when he searches for related content, it will live your app directly from the search results. If users have not installed your application yet, an install card shows up in search results.

**Dynamic Links**

They are the smart URLs that dynamically change behavior to provide the best experience across various platforms. You can use it in web, email, social media, referral or any promotions to increase user acquisition, maintenance, and value to Gain end-to-end understanding into all your development channels through analytics on the Firebase console.

**Invites**

Word-of-mouth is an outstanding and amongst the best methods for motivating users to install your application. It is a cross-platform solution for sending customized email and SMS invitations, on boarding of clients, and measuring the effect of invitation

**AdWords**

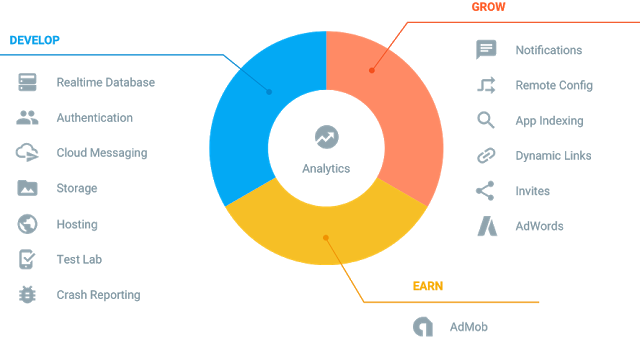
You can characterize custom audiences in the Firebase console based on device data, custom events, or user properties. By using this you can achieve potential clients with the help of online advertisements. You can gain deep insights into promotion conversions, and run targeted advertisement campaigns using Firebase Analytics to engage your audiences.

**AdMob**

It is a simple approach to adapt mobile apps with focused in-application promoting. It is mobile advertising platform which you can use to create revenue with the help of your application. Using it with Firebase Analytics gives you extra application usage information and analytics capabilities.

The majority of the above can now be executed when utilizing Firebase. One feature rich platform. However, they have their own limitations and trade-offs also. For example, unless your application is running over unified database which is being modified by a vast number of users simultaneously & the updates are required to be notified, the use of Firebase is not legitimized. Since essential review functions can be performed by MySQL database. It does not have any SQL features so if you want to send an email to a group of users, you’ll need to put the JSON into your favourite programming language and build the list manually or additionally write information to an SQL database for querying later.

If you have a business application or are considering to start one, [Firebase](http://www.cmarix.com/inquiry.html) is the one of the good options available for that. This interface actually has all that you could need to begin a fruitful business application or  to launch your current configuration forward in the right direction. Between analytics, development, growth, and money-making considerations, you will be set with this platform for a long time of continued success. It helps to save lot of time aiding the developer to concentrate on the core functionalities that make application or websites faster, extraordinary and user centric.

[](https://2.bp.blogspot.com/-riybXWDo1nE/V_pG3S83ksI/AAAAAAAADQM/ayLZ4LkjMt4LOx6PrMTavrFFUb5rmDmBACLcB/s1600/firebase.png)

Firebase is available for many platforms. You can build Apps for Android, iPhone, Web etc. Using firebase is also very easy as there are many official and unofficial [**firebase ios tutorial**](https://www.simplifiedios.net/firebase-ios-tutorial-user-registration/) as well as [**firebase android tutorial**](https://www.simplifiedcoding.net/firebase-realtime-database-example-android-application/) exist.

### **Lets see some advantages of using Firebase**

* If your app does run of a centralized DB, and is updated by a lot of users - then it's more than capable of handling the Real-Time data updates between devices.
* Stored in the cloud so readily available everywhere.
* Cross Platform API (If you are using this DB with an App)
* They Host the data. -Meaning if you are storing a lot of data, you don't have to worry about hardware!

But with advantages there exist some disadvantages as well.

### **Disadvantages of using Firebase**

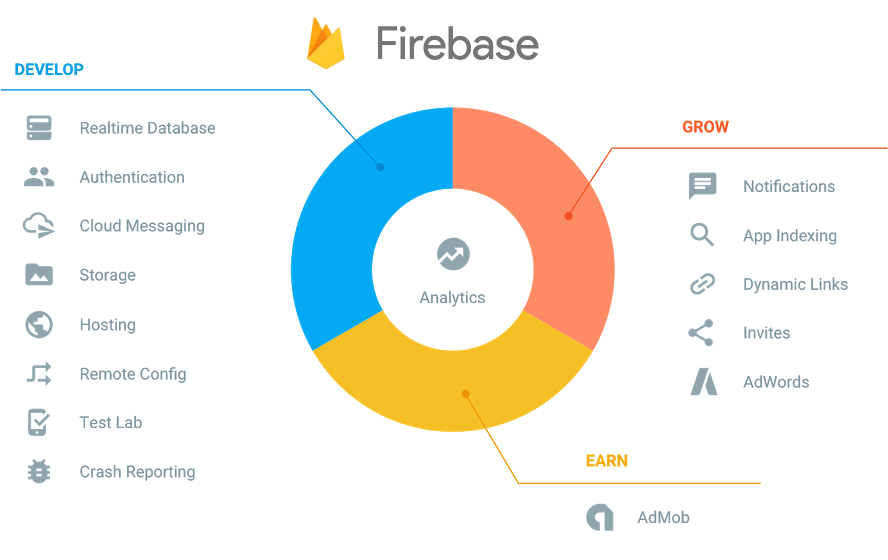
* Unless your app runs of one centralized database updated by a vast quantity of users, it's a major overkill.
* Storage format is entirely different to that of SQL, (Firebase uses JSON) so you wouldn't be able to migrate that easily.
* Reporting tools won't be anywhere near the ones of standard SQL.
* Costs! -Limited to 50 Connections and 100mb of Storage!
* You don't host the data, Firebase does. And depending on which server you get put on, viewing there up time there seems to be a lot of disruption lately.

But still some features of firebase which you have to use like Firebase Cloud Messaging. Though alternative for push notification exist but it is the service worth trying. As google will remove Google Cloud Messaging and will continue Firebase Cloud Messaging only.

# Why Choose Firebase as Cloud Backend for your Mobile Apps

The market of various apps is growing progressively. The competition in [mobile apps development](https://www.cleveroad.com/services/mobile-development)is too high to let yourself spend too much time on your product. Moreover, it’s very important to see the customer’s reaction and how your idea works in real life. That’s why it’s so important to release your application as soon as possible. And one of the best decisions to manage all that is to use [Firebase](https://firebase.google.com/) as cloud backend for your mobile apps.

Firebase is a [BaaS (Backend as a service),](https://en.wikipedia.org/wiki/Mobile_backend_as_a_service) which provides RTDB (real time database) services for mobile apps and web developers. It was established in 2011 and was purchased by Google in 2015.

Services offered by Firebase

## Why use Firebase?

Why is it useful and why you may need Firebase? Let’s make a brief of its capabilities for developer’s needs.

Firebase can be your only service to manage your web products and apps if:

* There is no need for integration with 3rd-party environments (log-in process via social nets, any kind of Maps or payment services)
* The aim is to create a lightweight app or upgrade an old one.

However, there may be situations which involve adding a server code:

* The app is complicated and cannot be managed by a front-end code only. A reliable server is needed to run the back-end code;
* There is a necessity for embedding 3rd-party APIs;
* Advanced level of safety is needed, and Simple Login service is not enough in this case.

If your app has all or several characteristics listed above and you develop a Firebase-powered app with a back-end code, your server becomes an agent between the client and Firebase. But Firebase itself continues dealing with real-time changes and scaling.

### Firebase is RTDB (Real Time Database)

Firebase NoSQL cloud provides a database for real-time applications as a service. This service provides an API for developers, lets you synchronize data and store it in the Firebase cloud. The RTDB syncing function provides the client with all the missing files after the connectivity recovering which makes it very valuable for stable work of your app. The company also took into account the possibility of integration with Android, iOS, JavaScript, Java, Objective-C and Node.js applications.

### Authentication process

Except traditional auth scheme (login/email-password), 100% Firebase powered app supports simplified login process through different networks. In this situation there is still no need for writing any server code.

### Firebase Storage Reliability

Using Firebase SDK to store and scale your data, you may be totally sure of its safety and reliability. The information can be sent directly from the client to Firebase Cloud, various privacy settings can be adjusted to limit access to defined groups or files. Being supported by Google Cloud, Firebase Storage can offer you almost endless storage space for keeping your data.

### Firebase Cloud Messaging

Google Cloud Messaging is incorporated into Firebase and works perfectly for web and mobile solutions. C2DM (cloud to device messaging) from Firebase is toll-free.

Speaking about mailout targeting, there are 3 ways how it can be done:

* To specified devices;
* To some groups of devices;
* To devices which are joined by some attributes.

The Firebase service can let you send up to 4 KB to the application but yet save the battery charge.

### Notifications

After messages to various users have been sent, it’s very desirable to let recipients know they have some info to check. For this purpose, there is Firebase Notification Console. The principle is similar to the messaging process. Firebase handles notifications without any additional coding. Very handy and time-saving decision to reach your clients.

### Firebase Remote Config

As it was mentioned at the beginning, Firebase is incredibly useful when you need to make an application faster. But this also means, in all likelihood, that a capability to upgrade the product without new release is very essential. Remote Config will be very helpful in such circumstances. With assistance of Firebase Console it’s easy to update your app’s design or its behaviour on the fly.

### Simplified debugging with Firebase

*Test Lab for Android*

If you have an Android application, you get an additional bonus from Firebase. Test Lab is great for testing a newly developed app or check it after adding a new feature. Robo test is great for performing tests and emulating user’s behaviour, exploring your app’s UI. Moreover, Test Lab cooperates with various services, like gCloud command line, Android Studio and has extended facilities for testing your product on devices located in Firebase data centers.

*Crash Reporting*

After testing your app, you can continue with one more tool to finish the testing process – Crash Reporting. With this instrument, all found errors are joined into groups, rated by severity and can be tracked by logs.

### Extending the capabilities with Firebase for built apps

If your application is being successfully used by your clients, there are still many spheres where Firebase can be helpful.

*Dynamic Links*

How to bring customers from the web right to your mobile app to download it? Dynamic Links can assist with that. Comparing to deep links, this service not only directs users to the app, but navigates to the desired feature after its installation or shows some additional info about the advantages of your product.

Dynamic Links provide the most important info without extra searching steps.

*Firebase Invites*

Your users can share some info from your app or promote it via invites. This function also works on the basis of Dynamic Links, so if one user shared some info or promo code, or anything else, the invite will lead exactly where it was supposed to (web, App Store or Google Play, etc).

### Exhaustive information and further growth of your product

But how to find out the statistics of your app’s downloads and where people found it, which banner was effective or how often a certain link was clicked?

All that and much more can be done by Firebase Analytics. You will be provided with almost unlimited reports on the following topics:

* Crashes and fixes;
* Efficiency of your advertising campaign;
* Customers’ behaviour.

The service is very precious because of its flexibility. You will not have to use different tools to analyze various attributes. Firebase Analytics gathers all the essential data by means of SDKs. All the information will be available for a survey on the Firebase Console. You will see a really wide range of info, starting from geography of downloads and demographic information up to the weak sides of your product.

FA is pretty similar to Google Analytics, just intended to improve your iOS or Android app and toll-free as its older friend as well. With this service, you will surely make your app shine.

*AdMob + AdWords*

After having analyzed the info about your product, you may want to continue promoting it. And for these needs there are AdMob and AdWords which are now integrated with Firebase. The next step is to make up your choice with the tools that you may want from a huge amount of Google advertisers, define the core users in the Console and enjoy watching the growth of customers.

*App Indexing*

If you have not only a website but an Android app too, you can associate them to make your users launch the mobile app more often. There is a possibility to connect them in such a way that you may bring the users to your app. Moreover, there is an instrument to define mistakes in your content indexing to make your products appear in search results more often.

All the services and advantages of Firebase, described above will pleasantly surprise with their reasonable prices:

* Android Test Lab, RTDB and Storage have very reasonable pricing grid;
* Analytics, Dynamic Links and crash Reporting are distributed without any limitations.

And again, Firebase is suitable for Android, iOS and web, which lets you create and manage products all together, using one multi functional service for all your needs you could ever imagine and get more tremendous results day by day.

# Reasons Not To Use Firebase

Building real-time applications is today standard. At [Crisp](https://crisp.im/), we used [Firebase](https://firebase.google.com/) in production over 9 month, starting from day one. From the dream to the nightmare, here is our experience.

Note that this article covers the production aspect of using Firebase Realtime Database, and we still think that this solution still great to build MVPs, Hackathon Projects, or if you have your own reasons.

**Edit: Firebase reached us and was very receptive to the issues we encountered. We are glad to see such an engaged team for their product. Our case is special as we have very specific needs for which we got to the limits of Firebase (which was by the way the pre-2016 Google I/O version). Nonetheless, this is an excellent service if all you need is storing user data and dealing with real-time issues.**

## The Project

### Our Initial (Good) Reasons To Use Firebase

[Crisp](https://crisp.im/) is a a dead-simple live chat focused on user-experience. We wanted to make a cost-effective solution so we decided to use a [Socket.IO](http://socket.io/) backend to deal with live-chat on websites and in the same time, using Firebase for our dashboard, all synced with [AMQP](https://en.wikipedia.org/wiki/Advanced_Message_Queuing_Protocol) on the backend-side.

**This hybrid architecture allowed us to build the MVP in only 3 weeks**, both the livechat and the dashboard, used by separate people in the same time, and get it to communicate with micro-services. This server-less MVP dashboard allowed us to focus on UI and UX rather than server code, and this was actually a good thing at that point.

We bootstrapped the product quickly and got to market early. Exactly what you'd want with an MVP.

### Then Came The Storm

First few months went great, and then boom. We got [featured on ProductHunt](https://www.producthunt.com/tech/crisp). We had a great growth, but with it came our first scale issues.

From a huge asset, Firebase became a nightmare, slowing down our execution. Building new features was deal-breaker and we had to think twice before releasing new things in production.

We discovered Firebase technical limits too, as we encountered performances issues when we scaled to 100GB+ Firebase traffic every month.

## 10 Reasons To Switch

#### 1. Spaghetti code

Server-less, doesn't mean code-less! Using Firebase means that **all your server logic is now running right in your web or mobile client.**

In most of apps, you have to send welcome emails, process images (avatars, etc), deal with payments, and build your business-core features. You really don't want all those to be done on the client, as this can range from impossible to dangerous for your business.

These things may still be "hacked" using Firebase, but it means that you will have to add even more code to your web-app, and it could be a nightmare to maintain if you have a mobile app too.

Think about distribution: any database logic change results in client app updates. How do you deal with clients who didn't update? Is it a correct thing to do for your users to deactivate older clients to force update?

#### 2. Integrating Firebase with micro-services is infernous

At Crisp, we use micro-services, and in many-cases you have to query a database, to get user information, IDs, etc.

Firebase can be both used from the client (eg: Web, mobile apps), but also from the backend (eg: NodeJS). You may query directly Firebase over the network, right from your backend, but you should avoid doing this because it is really slow at scale.

We used [Redis](http://redis.io/) to cache all these operations, meaning that we had to synchronize all the data back-and-forth. The micro-service doing this job (in other words - connecting to Firebase) had memory issues because Firebase caches data in memory and don't seem to release all unused references (we used the [Firebase NodeJS library](https://www.npmjs.com/package/firebase)).

#### 3. Pricing

Server-less doesn't mean cost-less. No, no, no: one day you will have to pay for your laziness. Indeed, we had to when we received this email from Firebase:

*Firebase paid tiers come with a usage quota, plus overage fees if you exceed the plan limit. Your Firebase, crisp-<..>, has been exceeding its plan’s limits, and, due to an error on our end, you have not been charged overage fees. We will be correcting this beginning next week, which will result in your account being charged for any excess usage.*

*[..]*

*Thanks so much for being a Firebase customer!*

Thank you for the mail, Firebase!

Paying $100 per month for something you can run on a $5 DigitalOcean droplet is something that get you to think twice when dealing with server-less code.

With your own server code, you will gain maintainability and productivity and you will have a cost-effective code-base.

#### 4. Firebase downloads all subtrees on load

Assuming that you are building a Slack-like app, you will have to download all channels data on app load.

Some people will say that it could be something improved with pagination, but with Firebase, you cannot paginate because you cannot get query array length, you cannot paginate ordered arrays, etc...

#### 5. You might have inconsistencies

Firebase supports offline operations. It's works like "git commits", but the main issue is that if you client goes offline and then online and you have concurrency on some input data (for instance, a shared notepad), you might have inconsistencies. Pretty much like [Git merge conflicts](https://help.github.com/articles/resolving-a-merge-conflict-from-the-command-line/).

#### 6. The problem of data migration

With Firebase, you can't deal easily with data-migration like you can do with a simple SQL database, an [ORM](https://en.wikipedia.org/wiki/Object-relational_mapping) or [ODM](https://en.wikipedia.org/wiki/Object_Data_Manager).

This means that you will have to do things like such:

if (user && user.new\_subdocument && user.new\_subdocument.new\_property) {

// Do stuff.

}

**Resulting in safety conditions, everywhere.**

#### 7. Relations are marvelous

Dealing with relations with NoSQL is hard, dealing with relations with Firebase is pain in the ass.

For instance: an user belongs to a team, and a team has users.

**User:**

{

name : "John Doe",

team\_ids : [...]

}

**Team:**

{

name : "Acme Inc",

user\_ids : [...]

}

It means that you user has to watch team\_ids, and then populate on your own teams and your team has to watch user\_ids and populate users. This example is simple. Picture the whole thing with more relations, and you get a spaghetti logic.

#### 8. Queues are buggy

To deal with server code and micro-services, Firebase introduced queues to share operations between server and prevent concurrency (eg: to avoid sending an email twice).

The NodeJS library implementing Firebase queues is named: [firebase-queues](https://www.npmjs.com/package/firebase-queue).

This feature was not well maintained by Firebase teams, and we had several bugs, including synchronization issues, locks, etc.

Besides, queues were a scale bottleneck as queue items can be inserted (stacked) quickly, but consumed (unstacked) very slowly. If your connected clients insert more than your backend is able to process (which is seems to be limited by the firebase-queue library timers, and not your CPU or network), this will result in huge processing delays that are not acceptable when dealing with real-time applications. Furthermore, I've just described a potential DOS attack vector: connect to a client and flood the queue with records, and the whole queue tasks will be greatly delayed, such that the service becomes unusable.

**So yeah, avoid Firebase queues.**

#### 9. You don't own your data

Beside the fact that your data is hosted on servers that you don't own, it's not possible to export your user data. You can't export emails, and user accounts are not recoverable (you cannot export user accounts with passwords).

Plus, it was not possible to export our data when we had hundreds megabytes hosted. We had to contact Firebase by email.

**Notice**: exporting data email/password data is possible by contacting Firebase Team, but not from Dashboard.

#### 10. Complex queries are impossible

It's still impossible to query your database to find fields with some properties.

For instance, you want to build a Slack-like app: you cannot count unread messages, even if you have the proper structure that would allow to do so (eg: a read = tag). The hack is to count all unread messages client-side, **after** having retrieved the whole dataset. It's a huge performance issue.

You can't perform operations to get active users, or doing batch operations to update documents having some fields.

#### Bonus. Want to build an API for your product? It's impossible.

Today, most apps expose a developer API. This is impossible to do with Firebase.

Of course, you could still remotely query your Firebase database upon receiving API requests from clients, but this would be extremely slow as the data is hosted remotely and the Firebase libraries are leaking memory (see above).

The second solution is to make an HTTP wrapper. All that said, in this case the best option is build your API with standard storage backend you host, using your own database systems (relational or non-relational). This is what we did.

## Migration

#### Architecture

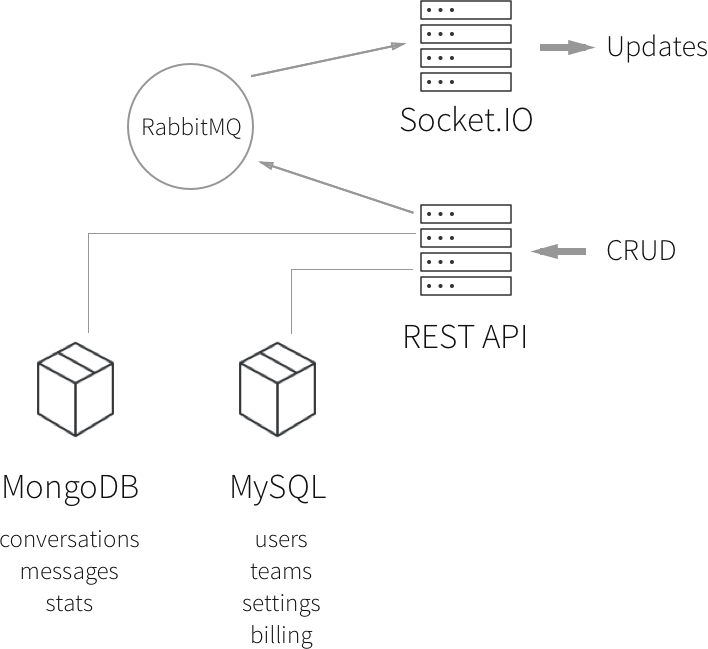
We decided to migrate to a REST API. To make things simple, we chose to use an old-fashioned SQL Database, which overs 90% of our data storage needs. It has relations, models, and it's perfect to make a bulletproof, simple, and maintainable API. You can read our API docs [here](https://docs.crisp.im/api/).

For the 10% part, we decided to use MongoDB to store messages and conversations at scale. We used UUIDv4 as primary keys, both in MongoDB (this is the de-facto standard in MongoDB) and into our SQL DB (this is not the standard in the SQL world) to make it transparent for everybody.

All CRUD (Create/Read/Update/Delete) operations are performed over the REST HTTP layer and some asynchronous replies (after some API requests) are send back over a Websocket layer (we call it our Realtime API).

Updates are internally forwarded over AMQP and then synced with a Socket.IO micro-service, and an authentication layer filters ressources that you are subscribed to.

Simple, effective, and sustainable. We are now **free to switch from Socket.IO to another engine**, or even **migrating from MongoDB to another data store**.



#### Let's Code

Started to build the API from [Paw](https://paw.cloud/) (a [Postman](https://www.getpostman.com/)-like app, for Mac), by mocking API routes and simulating things. Then, we made it working with real code. We used NodeJS but you could use PHP, Ruby or every language that you and your team are happy to work with.

At the same time, we removed all Firebase logic from our apps to remove all logic from the client and plugged all network operations over our API. The complexity should be implemented in the server.

These two parts took around three weeks, then came our migration scripts, migration tests, and production migration.

#### Let's Breathe Again

This migration allowed us to focus again on execution and several things were improved into our product:

* Web app loading time were reduced from **20 seconds** for some users with many conversations to **2.5 seconds** (constant time, regardless of how many chats you have).
* Features like search, pagination, and more are now possible because: MongoDB.
* The code of our apps became simple again.
* Making mobile apps became simple, too.
* Migrations are now easy thanks to schemas and ORM.
* We can query our data-bases to update so inputs when needed (applying a discount, etc).
* We now have a developer API.
* If you read this one, you'll have a -30% discount on all Crisp paid plans forever (use coupon code IHATEFIREBASE).

**Thank you for reading, and now, think twice before using Firebase!**