The Ultimate GUIDE to Creating a Mobile App

NEW FOR 2018

LIMITED EDITION

jetruby

Chapter 1

TAKING THE LEAP

The first and only thing on your mind is your Idea. It keeps you up at night...

"There's an app for that"

Ask yourself: "Is your application going to be a solution to a challenge that someone is currently facing?"

Identifying a gap in the market or a problem is one thing. The next thing you should stop and wonder about is if the solution already exists or if there is a reason that nobody has come up with it yet.

The easiest way to check if your idea is already out there is to spend a couple of hours on thorough online searching. This will save you from bitter disappointment also prevent you from being sued in the future. In the next chapter, we'll talk about both market and target audience research in more detail.

Chapter 2

KNOWING YOUR AUDIENCE

Who are my users?

Just because you've got a brilliant idea for your future mobile app doesn't guarantee success. If you want people to download your app, you have to make sure that they want what you're offering.
To get an overall view of your users, you need to do research before, during and even after the development process.

To illustrate how to approach finding your target audience, let's take the following example. Imagine you created an app that allows users to book car wash services. Consequently, your target users will be car owners who are interested in giving their cars a wash.

The next step is to find out more information on those car owners.
How often people use car wash services and how much time do they spend on it? Does gender play a role in how often people wash their cars? What about location or occupation? Finding out as much about your potential users will help you to better understand and solve their problem.

To find this information, you can start by looking at potential competitors and their users. Additionally, there are a bunch of tools that will be able to help you in gathering data about your potential markets: Google Trends/Adwors,Flurry,Quantcast, and App Annie.

Is it the right time for your app?

Another important factor of your app's success is choosing "the right time" to launch.

Looking at our car wash example, you can probably assume that more people wash their cars in summer as opposed to in other season when it might rain more often. Your app could also be very specific to a different season, or holiday like Christmas or Halloween.

Choosing a strong USP

Your app's Unique Selling Point will be what differentiates it from the competition. This means that the app should have one or more unique qualities. **What are the main advantages of your app?**

To determine this, create an overview of the following

- What does your app do?
- Who is it for, who can use it?
- What are the main differences between your app and the competitors' app?
- Why should people use your app instead of any of the other's available out there?

Our car wash app could perhaps offer 24/7 washes, whereas competitors don't.

Main takeaway: When identifying your target audience, find out demographics such as gender, age, location, occupation, education, and socioeconomic status. Make sure your app has a USP and you launch it at the right time.

Chapter 3

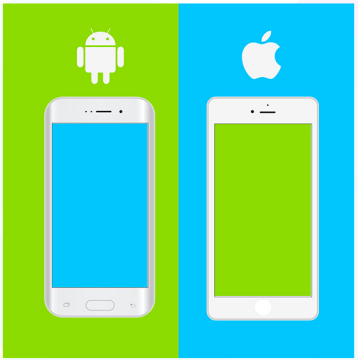
CHOOSING A PLATFORM

Another factor that plays a role in your app's success, is the mobile platform that you'll be focusing on. The choice usually boils down to Android and iOS operating systems. But what about Windows Phone?The offspring of Microsoft failed to compete with its main rivals and, as a result, announced to receive no further updates. While for the owners of Lumias and other Windows Phone-based devices this is a devastating piece of news, for businesses, it's a huge relief since you have only two mobile operating systems to maintain. Unless your user base are the three last remaining Blackberry users... in which case, you should consider whether you should be allowed out on your own.

How different are Android and iOS in terms of app development?

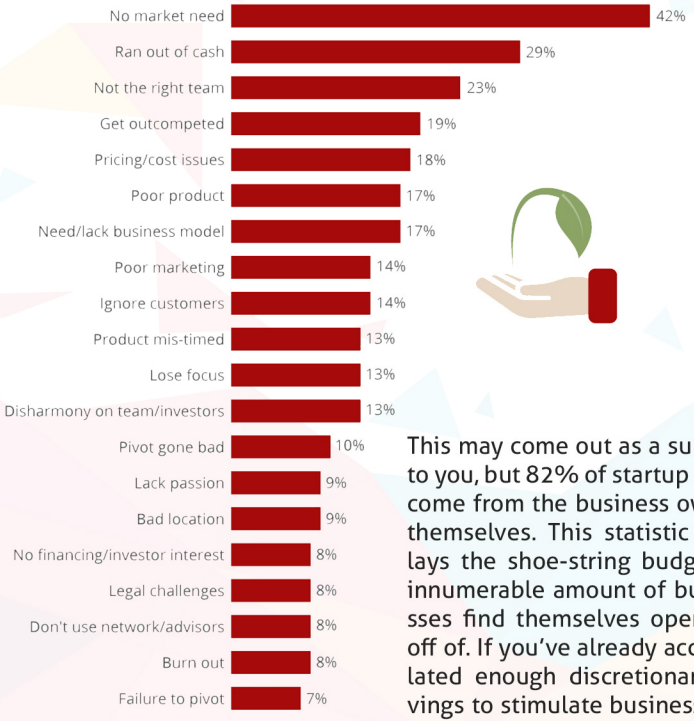
In a nutshell, if you want to develop a mobile application for a specific platform, you have to use a language that is natively supported by that platform. For example, iOS apps are based on Swift or Objective-C while Android apps are developed with Java or Kotlin (don't worry about the names of the programming languages - we'll elaborate on them further in the book). These apps are also called "native". At this point, you're probably wondering if it's possible to use only one language to develop a mobile application for both platforms. Yes, it's possible.

As technologies were evolving, the tech community realized that it'd be easier to create only one app which could be used across different platforms than create separate apps. As a result, cross-platform frameworks were born: Ionic, Xamarin, etc. They're called "hybrid".



The Top Reasons Startups Fail

Most frequently cited reasons for startup failure*



This may come out as a surprise to you, but 82% of startup funds come from the business owners themselves. This statistic displays the shoe-string budget an innumerable amount of businesses find themselves operating off of. If you've already accumulated enough discretionary savings to stimulate business, you may be okay for the short-term, although it's still important to track how much you're losing, how much you're earning, and what your make-or-break point is.

However, there's still a large portion of money (18%) that isn't coming from the business owner. If you're starting from scratch, additional funding is available from several sources. This funding will help you get your mobile app off the ground. This includes loans from both traditional banking institutions and private lenders.

If you're living in the US, the U.S. Small Business Administration (SBA) may also be able to offer

According to Jakob Nielsen's "Usability Engineering", users notice system delays that are as short as a single second and if an action (e.g. updating the newsfeed) takes longer, they can become distracted. Native applications fully utilize the processing power of mobile devices so their response time is minimized to the greatest extent possible.

Internet connection isn't a must. Since native applications are installed on the device itself, they allow you to access data in the offline mode. Combined with great performance, the ability to work offline makes native applications a perfect choice for developing mobile games.

Disadvantages:

Cost of development. Native applications are more expensive in development than cross-platform and web applications. This is because you need to develop a separate application for each platform that needs to be supported afterwards.

Intercompatibility with other mobile OS. This one is super easy: apps developed with Swift, Objective-C are only compatible with iOS, the same with Android.

Submission to the app stores is a tricky process. What's more, every update that you roll out after the first release also has to be authorized by the app store before it becomes available.

Chapter 3

Native applications

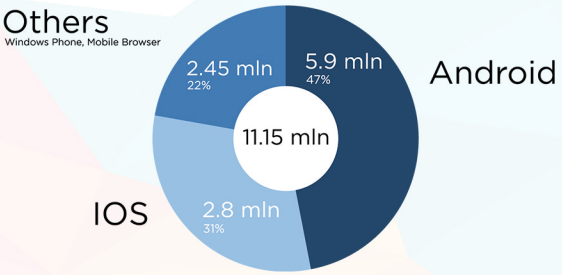
Advantages:

Reusable code base. Instead of writing several different applications, mobile developers can use the same code for all platforms. Moreover, you can just implement a feature for one platform and it will work on the other one. Such reusability is also an advantage when it comes to deployment.

Cost-effective. Developing an application that works across multiple platforms and uses the same codebase is very cost effective. To have your idea implemented as a cross platform application doesn't require heavy investments in terms of resources. Thanks to a short learning curve and a unified stack of technologies, all platforms can be maintained by a single team.

Wider audience reach and marketing opportunities. Cross platform applications target multiple platforms. As a result, businesses can grow their user base much faster compared to native apps. This is especially beneficial for small businesses. When you have an app released for both Android and iOS at the same time, it saves you time and money as opposed to creating only one app for either of the platforms.

Unified UI and UX across all platforms. How your application performs is undoubtedly important. However, no less important is how the app looks and feels. It may be challenging to keep the same UI and UX simply because there are two separate teams working on them. Using a single codebase allows cross platform applications to maintain the same look and user experience across all platforms.



Disadvantages:

Not as flexible as native apps. The fact that cross platform applications can be developed for both Android and iOS is an advantage and a disadvantage at the same time. Why? When you need to implement some cool feature that is expected to work and be supported on all devices equally, there is a chance that cross platform technologies won't be able to provide you with the same level of flexibility you can find in native programming languages.

Performance isn't their strongest aspect. Because cross platform applications have to give up a great deal of flexibility for the sake of wider support, they also lose in terms of performance compared to their native counterparts.

So, if you want to impress users with a lightning fast interface, rich functionality, and overall performance, a native app is what you need. In addition, you get better security and stability. The price for this is that you'll most likely need to hire two dedicated teams for each platform.

Cross platform apps, on the other hand, can be developed for both iOS and Android, wherein the whole thing is a one-team job. Plus, cross platform apps are much easier in terms of maintenance and deployment, so you can spend more time and money on marketing and attracting new customers. However, their biggest disadvantage is lower performance, which may be especially crucial if you're developing an application with features that require deep hardware integration.

Main takeaway: there is no magic bullet when it comes to choosing which platform to use: just check the advantages of each platform.

HOW IMPORTANT IS A DATABASE

Data synchronization and offline capabilities are key to successful mobile apps.

- It would be difficult for any online app to store huge amounts of data for millions of its customers without a database.
- In addition to storing data, a database allows you to easily update any specific, existing data or replace it with newer data.
- From a security viewpoint, the data, which is stored in the database of your app, is less likely to be tampered with or stolen compared to other ways of storing it.

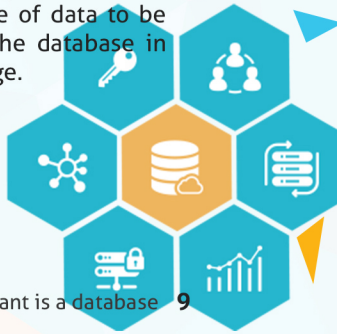
Having a database, you can easily check if there is a duplicate set of data.

If a user enters an incomplete set of data, the database will identify and avoid any errors.

Factors to be considered while choosing the right database for your application

Data structure: Your data structure basically depends on how you intend to store and retrieve it. As your application deals with data present in a variety of formats, selecting the right database should include picking the right data structures for storing and retrieving the data. Selecting the wrong data structures for working with your data will result in your application's data extraction time lengthening; and will require more development efforts to work around any data issues.

Size of data to be stored: This factor takes into consideration the quantity of data you need to store and retrieve as critical application data. The amount of such data may vary depending on a combination of the data structure selected, the ability of the database to differentiate data across multiple file systems and servers, and even vendor-specific optimisations. Thus, you need to choose the database taking into consideration the overall volume of data generated by the application at any specific time and also the size of data to be retrieved from the database in any given package.



Speed and scalability: This addresses the time taken to service all incoming reads and writes to your application. Some databases are actually designed to optimise read-heavy applications, while others are designed to support write-heavy solutions. Selecting a database that can handle your application's input/output needs can actually go a long way to building a scalable architecture.

Accessibility of data: The number of users concurrently accessing the database and the level of computation involved in accessing any specific data are also important factors to consider while choosing the right database. The processing speed of the application gets affected if the database chosen is not good enough to handle large loads.

Data modelling: This helps map your application's features into the data structure and you will need to implement the same. Starting with a conceptual model, you can identify the entities, their associated attributes, and the entity relationships that you will need. As you go through the process, the type of data structures you will need in order to implement the application will become more apparent. You can then use these structural considerations to select the right category of database that will serve your application the best.

Scope for multiple databases: During the modelling process, you may realize that you need to store your data in a specific data structure, this may mean certain queries cannot be optimised fully. This may be for reasons like complex search requirements, the need for robust reporting capabilities, or the requirement for a data pipeline to accept and analyze the incoming data. In all such situations, more than one type of database may be required for your application. When choosing more than one database, it's quite important to select one database that will own any specific set of data.

This database acts as the canonical database for those entities. Any additional data bases that work with this same set of data may have a copy, but will not be considered as the owner of this data.

Safety and security of data: You should also check the level of security that any database provides to the data stored in it. In scenarios where the data to be stored is highly confidential, you need to have a highly secured database. The safety measures implemented by the database in case of any system crash or failure is a significant factor to keep in mind here.

With that said, let's run through the options of storing data in your application.

SQLite

SQLite is the most used (and) open source database. It comes integrated in the latest Android versions by default. For interaction with the OS, SQLite uses the SQLiteDatabase and the SQLiteOpenHelper Java interfaces. The programming interface is fairly simple and user-friendly since it's written in pure ANSI-C. In terms of size, SQLite is relatively portable: you can store the whole database in one disk file. At the same time, implementing standard mechanisms to manage SQLite is hardly a trivial procedure as you need to manually write tons of code. Because of that, developers have to use various libraries and ORM add-ons.

Core Data

This is the second main database available in iOS by default. From a technical perspective, Core Data was designed to focus more on objects rather than the traditional table database methods. With Core Data, you are actually storing contents of an object which is represented by a class. There is some difference between SQLite and Core data in terms of memory:

- Uses more memory than SQLite
- Uses more storage space than SQLite
- Faster in fetching records than SQLite.
- Don't have Data Constrain mplement business logic.
- Operates on in memory (data needs to be loaded from disk to memory)
- Need to load entire data if we need to drop table or update

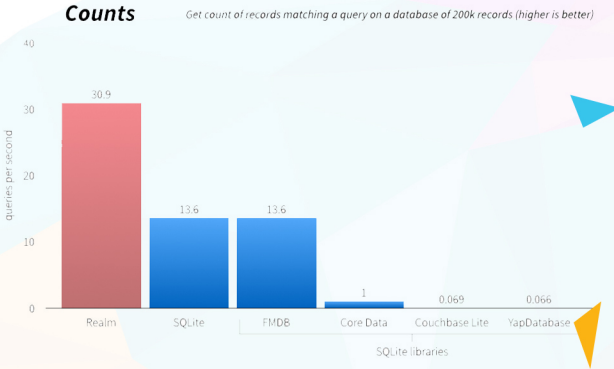
But Core data is Faster than SQLite

Realm

Realm was designed to be faster and more efficient than the previous database solutions. This new solution is a cross-platform mobile database. It's available in Objective-C and Swift, and it's designed for both iOS and Android. Being a fully-fledged cross-platform mobile database, Realm was created to become the ultimate solution for storing data: both performance and efficiency-wise. What really sets it apart is that you can handle all the work with a couple of lines of code. Realm is very easy to install and more responsive compared to SQLite alternatives like ORMLite or Greendao.

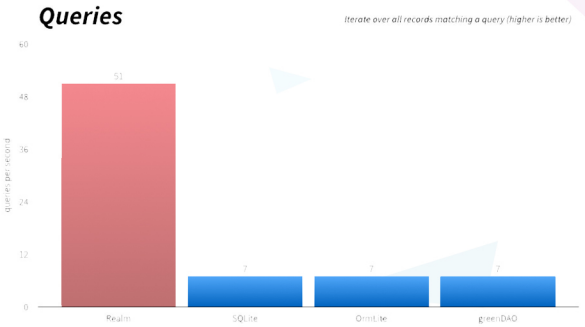
Realm is an incredibly fast library to work with. Realm is faster than SQLite and Core Data and the benchmarks above are the best evidence for that. But there are other benefits of using Realm:

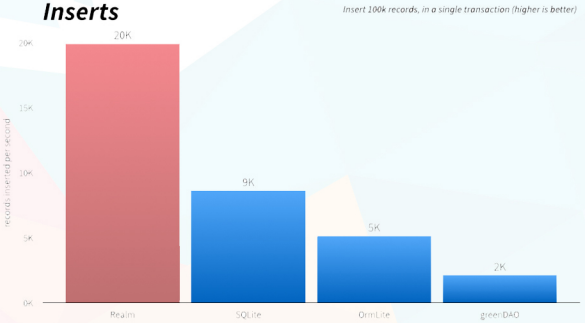
- The ease of integration and the ease of use. Realm comes with great documentation and requires almost no effort to get started.



- **Cross Platform:** Realm database files are cross platform and can be shared among iOS and Android. Regardless if you work with Java, Objective-C, or Swift, you will use your high-level models.
- **It's a trusted product.** Realm has proven itself as a reliable solution for banking and enterprise applications, healthcare providers and even top rated applications like Starbucks.
- **It's fully open-source,** which means no fee or paid subscription is required. It's FREE!
- **Scalable:** Realm is ready for scalability and works with large data volumes easily. You will bring speed and stability to your app by using Realm as your database provider.

Main takeaway: If you need a lightweight database, then SQLite may be your choice. If you're dealing with a complex object graph with many entities, attributes, and relationships, Core Data is a more preferable option.





Chapter 5

ACCEPTING PAYMENTS

In our modern world, online shopping and e-commerce have became an integral part of our lives. In order for your clients to be able to use and pay for your product or services, you'll need a payment gateway.
A Payment Gateway is any piece of software development that allows to make payments for e-commerce services within websites and applications.

Without Payment Gateways you wouldn't be able to make purchases online.
Going back to our car wash booking app, let's imagine a different function for it. Now, we are selling DIY car wash products. Our customer picks a car wash soap, polish, and a microfiber wiper and puts it in his cart. As soon as the products are in his cart, it gets linked up with the payment gateway.

After the request has been sent to the card processor, the payment gets approval. The next step is when the information is displayed in the merchant account after. Few days later, it's sent to the merchant's bank.

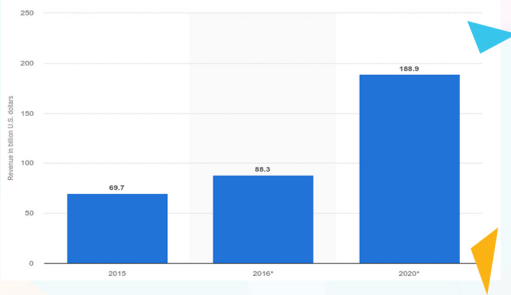
There are various account types:

The Dedicated Merchant account is the best choice for you if you're a merchant or business owner. It adjusts to customer requests and the money process transfer is pretty fast. Downside: it's expensive.

The Aggregate Merchant Account allows you to store all your funds together, as well as other people's money. This kind of account is more popular than a dedicated one. Additionally, the biggest payment processing players, such as Braintree, PayPal, and Stripe offer the aggregate merchant account.

Other things to consider:

Applicable security certificates. You'll also need to purchase PCI DSS compliance certificate for assuming the authority to handle customer banking data. It doesn't even matter whether you're using the most trusted gateway providers or not, you'll have to get this certificate.



Type of products that you're selling. One of the main factors that affect the electronic money system is the type of product being sold. This is because there are various methods for selling different categories of product (it can be physical goods or digital content). Moreover, don't forget that the Apple App Store as well as Google Play Store won't allow you to use third-party e-commerce services.

As a result, to prevent any misunderstandings you need to make sure that all the transactions within your application use specific user accounts - Gmail or Apple consequently.

After that, you'll have to connect a payment gateway in order to be able to carry out online transactions.

Choosing a Payment Gateway Provider

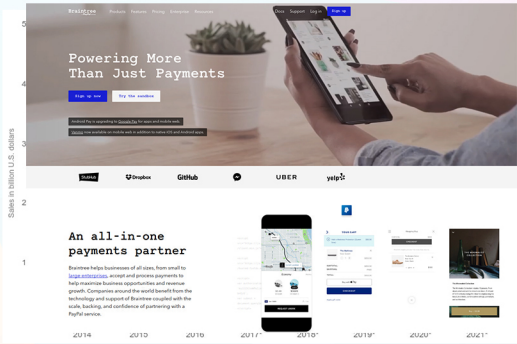
Braintree develops payment gateways that are considered to be some of the most reliable out there. It's relatively young as they launched in 2010 and are popular accepting a wide variety of payment types. They accept credit or debit cards as well as Apple/Android Pay payment, Venmo, and even Cryptocurrency (like Bitcoin). One of the biggest advantages of Braintree is that you won't need to sign a contract and you completely own your customer data.

Price List:

- Rate: 2.9% + 30 cents per successful charger
- No fees for the first $50,000 processed
- Monthly fees: none
- Setup fees: none
- Currency conversion fee: 2%

Features:

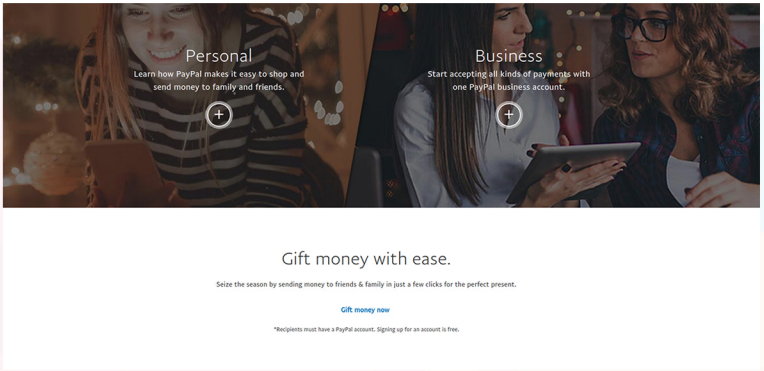
- Advanced attack protection and fast detection of swindlers
- Direct integration with PayPal
- Compatible with seven programming languages
- Uses The Payment Card Industry or PCI
- Supports about 130(!) currencies.



PayPal

PayPal is probably one of the most popular and even ubiquitous services for electronic payments from e-commerce apps. This payment system is popular partly because it's in the form of a digital wallet. The majority of people use PayPal for booking and buying tickets. Besides, PayPal is chosen by many customers as it allows to do shopping without providing merchants any information about credit card data.

Besides, using the PayPal mobile SDK you'll be able to accept PayPal, credit card as well as Bitcoin payments. Additionally, it's pretty simple to create simple Payment Buttons using a snippet or build an elegant custom payment form.



Payment Buttons using a snippet or build an elegant custom payment form.

Here are the grooviest features of PayPal:

- Invoicing is quite simple
- Host your own checkout page
- Comprehensive reporting tools for all your financial activity
- Reporting tools for all your financial activities
- Managing payouts to multiple recipients have never been so fast and simple

Price List:

- Rate: pay a competitive rate of 2.9% + $0.30 per U.S. transaction with no monthly or hidden fees

- Monthly fees: no fees or $30 in case you want to get the PayPal Payments Pro package
- Setup fees: none
- Currency conversion fees: It actually depends on your location and varies starting from 0.4% to 3.3%.

Stripe

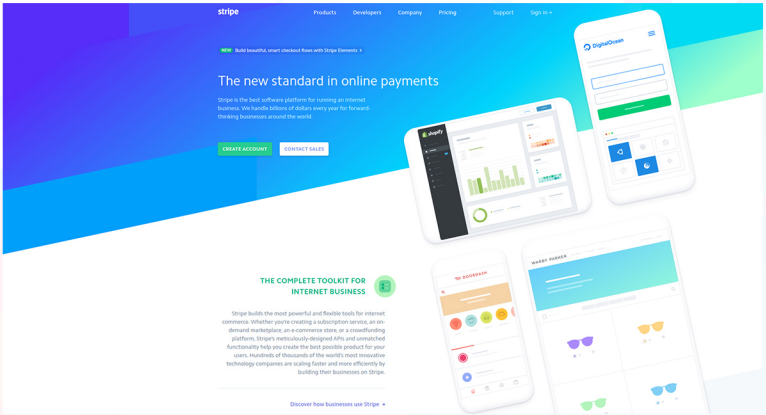
This is one of the most popular e-cash services. It has also become synonymous with "simplicity" in the payments world. This is because they make things simple for both developers, customers, and merchants. Stripe allows one to perform one-click purchases, subscriptions and even do shopping directly from apps. Besides, thanks to the new API it can allow purchases from Twitter feeds without exiting Twitter. Stripe is an extremely user-friendly service with clear documentation and smooth and simple integration with mobile applications.

Features:

- Supports Android and iOS platforms
- Integrate social media buttons for payment process.
- Customized payment forms.

Price List:

- Rate: 2.9% + 30 cents per successful charge
- Currency conversion fee: 2%
- Setup fees: none
- Monthly fees: none



Amazon Payments

Amazon is a giant in the online shopping market. It launched in 2007 in order to provide the same experience on other sites. But what does this actually mean? Literally, that means that if your customer has an Amazon account, all their payment and shipping details will be populated to your site. Even if it's their first time on your site, you can bring them that much closer to one-click checkout.

This feature will be especially useful for you and will allow you to increase conversions as well as reducing chargebacks. Their price list is pretty standard. Their charge a standard 2.9% + $0.30 per transaction.

Authorize.Net

Authorize.Net was launched in the far distant past: 1996. It's considered one of the oldest players in the e-commerce payment industry. Their market share is huge, specifically because of the fact that they've been on the market so long. They don't allow their customers complete access to their customer accounts. Whether it's a disadvantage or not is up to you. Anyway, if you decide to leave Authorize.Net, you'll need to contact customers to resubscribe to your services and recollect their payment information for simple checkouts. Additionally, Authorize.Net still charges a setup fee and a monthly gateway fee, though their price is not so different: 2.9% + $0.30 per transaction. Probably, the biggest advantage of Authorize.Net that they can offer is their serious relationship with security.

Main takeaway: Research and decide which payment gateway is best for your app

Chapter 6

SECURITY: WHY IT'S IMPORTANT AND HOW TO ENSURE IT

The amount of personal info kept on mobile devices these days is absolutely mind-boggling. From the exact location and favorite places, to bank accounts and passwords - our smartphones have become a true gold mine of information that many would like to get their hands on. In light of this, the issue of data protection has never been so urgent - for both users AND developers alike.

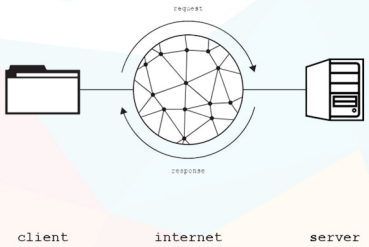
In spite of the fact that security has significantly improved in recent years, it is still an actively debated topic, with good reason. If you get it wrong and something goes wrong, you can watch yourself crash and burn right there. It's probably one of the top issues you need to have on your list.

How to Secure User Data on iOS Devices

In order to better understand why a typical client-server app so often comes with security problems we need to take a look at its implementation scheme:

The chain usually consists of 3 elements:

1. Client. (iOS app)
2. Communication channel (The Internet)
3. The app's server.



So what could go wrong? Well, from the security point of view - almost everything, because user data can be compromised on any of these components.

Data exposure happens when:

- Data storage is insecure;
- Data leakage (some OS components may save sensitive data without letting you know about it);
- Cryptography is bad;
- Weak protection of execution files, if any.

Let's elaborate on each of these points.

Insecure data storage

All apps on iOS run in a sandbox. This is also the place where the app data is stored. Say we have app 1 and app 2. The first one does not have access to the

sandbox of second one as well as sandboxes of other apps. If app 1 saves a file it can't be read by the other app. Someone new to security principles might think that everything is ok. When, in fact, it's not. Storing data in a sandbox does not mean it cannot be read from the OUTSIDE. And this won't even require a jailbreak.

My advice:
- Avoid storing sensitive data in CoreData/NSUserDefaults/ openly;
- Use data encryption;
- Forget about static encryption keys;
- Use KeyChain for storing user accounts or encryption keys;
- Use DataProtection;
- Select only the most secured class suitable for the app

We should probably expand on what security class does. It determines when a certain piece of information (a file or a KeyChain record) is available for the app. It also determines whether the record will be backed up or not.

Data leakage

User's personal information is usually stored in HTTP or cookies when they send a request. While the cache of the application is stored locally. You already know what could happen next.

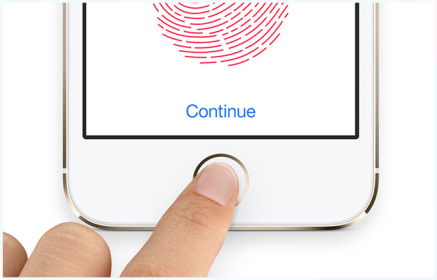
My advice: set query caching and cookie policy properly

A common problem is the exposure of sensitive data in the app switcher. When you exit or minimize the app, the system makes a screenshot of the app's status. The status is then shown in the app switcher, so user data becomes visually exposed.

My advice: modify the UI so that it won't expose certain information before exiting the app.

It's your responsibility to care about security on behalf of your users. Everything's fine until it's not. Then it's not. It's really not.

My advice: use a pin code or Touch ID to check on the login screen. This will provide additional security at the user interface point.



Protection of executable files

Protecting executable files in iOS is more of an etiquette matter rather than a real necessity. Though, following some basic principles has never done any harm, has it?

My advice:

- Use stack protection;
- ASLR;
- Delete all debugging info and logs from release builds.

TLS security (transport layer security) is based on the certificate validation. It's a protocol, which enables data encryption for socket-based communication, along with authentication of servers and (optionally) clients to prevent spoofing. If you do not check the validation, the security of server connections can't be guaranteed.

My advice: use Certificate Pinning. Each query to the server should also be signed.

Certificate Pinning will allow you to avoid spoofing the certificate from the trusted sources to conduct man-in-the-middle attacks. At the same time, signing each query won't let server requests repeat should TLS be intercepted and bypassed.

Server

The server is the last but not least component. The first rule here is that your app must not contain any information about the server.

My advice: erase any mention of test servers and/or test endpoints, because they are usually much less protected, if protected at all. Doing this will allow you to avoid server attacks, data structure exposure

And what about Android applications? Is it easier/more difficult to ensure security of user data on Android devices? We'll address this issue in our next chapter.

Chapter 6

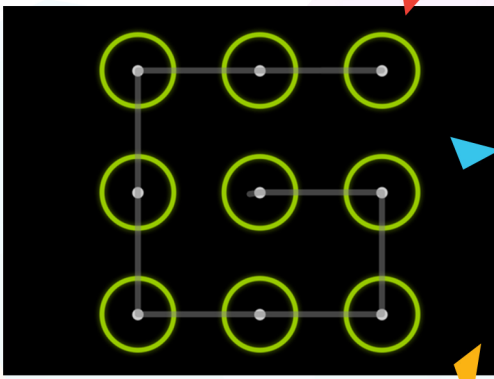
How To Protect Data On Android

In terms of storing user data in Android by standard means, the platform uses the following:
- SharedPreferences, a tool that keeps small amounts of data, e.g. app settings;
- Account Manager, a tool that keeps user account data, i.e. logins, emails, passwords;
- SQLite, the database where the rest of the app data is stored.

A fly in the ointment here, is that all of the three elements use files as a storage. Once the device is rooted, they become highly exposed. For this reason, keeping important things like passwords "in an open view" is considered to be a dangerous practice. A good one, however, is to use third-party encryption tools for storing and transmitting data between the client and the server.

Starting from version 5.0 (Lollipop), the Android security issue has been being actively worked on. Any user data on a device is now being encrypted before writing and decrypted before reading. However, given the openness of the operating system's source code and the ease of changing the firmware, calling such measures a good protection is an overstatement. The situation is also aggravated by the fact that only 50% of all Android devices are running version 5.0 or higher, According to Google. And there is no way of knowing for sure if the encryption is implemented in a specific device model as the number of smartphone and tablet manufacturers is huge and keeps growing.

In spite of that, things aren't completely hopeless. Android still does a relatively good job ensuring security in a running application. The standard means of data protection allow developers to:
- set a mandatory lock password to request before displaying sensitive information;
- disable in-app screenshotting;
- disable screenshotting the app in the background;
- set a mandatory password to enter before important actions.



For maximum protection, however, you will need to implement extra layers of security. We recommend starting with Android's database. There are several ways to encrypt SQLite. If you're going to deal with third-party databases, though, we have good news for you. Most of them, e.g. Realm, come with encryption mechanisms by default. The next step is to make sure attackers won't be able to access user data, e.g. by implementing a pre-set password or PIN code.

The approach above applies to any kinds of data. Wherein, the security level depends on the encryption method and user key.

To make brute-forcing a less trivial procedure, you can employ a more elaborate solution, e.g. an unlock pattern.

It is an excellent alternative to the conventional password and the best way to protect sensitive data on Android versions below 5.0. Even though newer versions have a lot more to offer in terms of data protection, I still think additional security measures won't hurt.

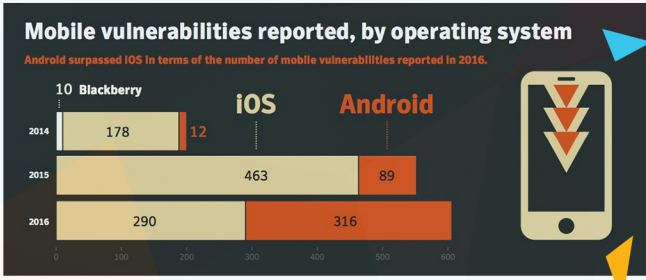
App data protection

Android applications are distributed as .apk files. Normally, users download them from official places like Google Play or Amazon. However, the same and many other applications are also available on third-party websites. And since the .apk format can be easily decompiled into pure Java code, attackers can gain access to various user data, e.g. authentication credentials, mechanisms of data transmission and protection, keys etc.

To prevent that, developers use ProGuard, a tool that allows for compressing data, removing unused parts of libraries, as well as renaming methods and variables.

Files created with ProGuard take up less space and, most importantly, are much harder to decompile.

Storing data in the native code is

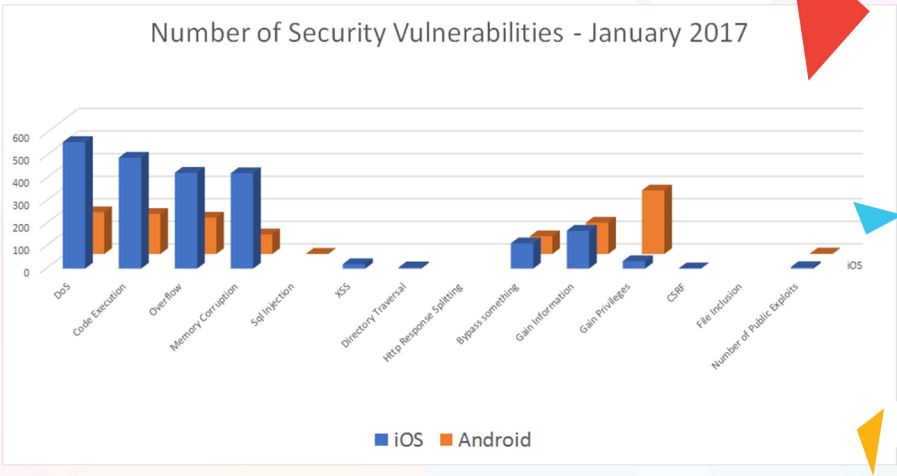


also a good way to protect the app from attackers. Why? The reason is simple: native code is one of the least likely places for them to look. Combined with database encryption, this approach offers a relatively high level of protection. On the other hand, it requires making compiled packages for each supported architecture. Basically, developers will have to increase the size of the installation file or create a separate one for each platform - which is a big disadvantage.

However discouraging this might sound, attackers still can easily gain full access to most of the applications on the market. In this regard, I always recommend users follow these rules:

- store as little sensitive data on your smartphone as feasibly possible;
- download information only when necessary;
- use the secure HTTPS protocol.

Yet there are a lot of applications that are simply built to store data in the smartphone's memory. Usually, it's either to be able to work offline or cache data for better performance. In this case, though, additional user data encryption improves security by a large margin. Everything we described above does not require high additional costs, because most of the protection mechanisms are implemented through standard documented methods. In the end, whatever it costs is worth it. The bad publicity and loss of users will cost you more in the long term.



Chapter 7

DESIGN

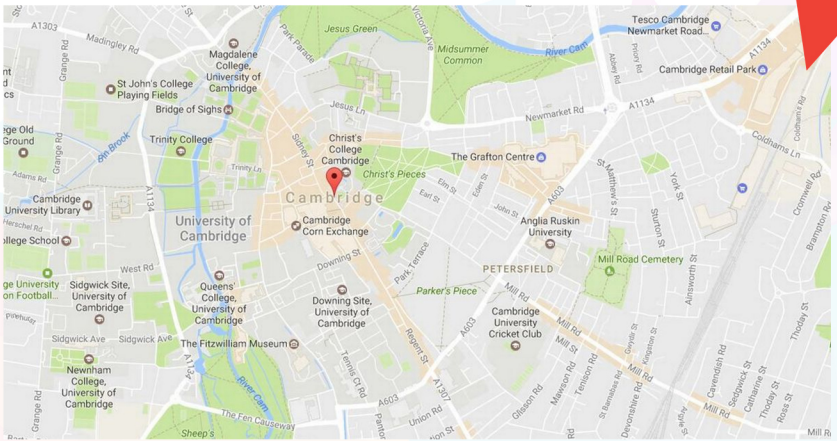
Designing a mobile app is a real challenge, especially because you want it to survive amongst millions of other existing apps. That's why it is highly important to understand what makes a good design and how a wrong color can make your whole investment go in vain. Here are a few key bits of advice that our company considers critical for good mobile design.

Content as the main part of value proposition

What is the most important part of the application, in terms of value provision? If you're thinking "content", you're right. This rule relates to any kind of application, whether it's a social feed, to-do list, or news updates. It's your content that entices people to download and start using your application. One of the most successful examples is Google Maps. Having decided to redesign their application, Google completely removed all the buttons and other unnecessary panels and made maps their interface.

Space

Whitespace (or "negative space") is an empty space between and around elements of a page. Although many may consider it a waste of valuable screen estate, whitespace is a



waste of valuable screen estate, whitespace is an essential element in design.The white space on a page can be every bit as important as the space occupied by UI elements because even empty space serves a purpose and supports the visual integrity of a layout. Whitespace can make both scanning and reading content easier and more predictable.
Also, it's better to group related elements together to create better content relationships.
Many apps and websites suffer from cramming too many elements and information together without enough breathing room. If cluttering your interface overloads your user with too much information, then reducing the clutter will improve comprehension: by removing distractions, you force users to focus only on what's immediately visible.

Let the products speak for themselves when you're working on the app design.

Create a conversation flow

To make your application as user-friendly as possible, the information flow of your application should give users an impression of having a conversation with their friend. This will contribute to creating a relationship between users and your app in a natural way. Nowadays, we can see the convergence between informal and work-related mobile use devices. The majority of people have grown accustomed to interacting with websites and apps in an informal way.

Here's a number of rules that can help you in building trust:

- Give your users the opportunity of two-way communication.
- Creating a "Call" or "Get in touch" button will allow mobile users to know that they can be a part of two-way communication. Address only one topic at a time.

- Always focus on the users' intentions and goals. Don't overwhelm users with a large amount of information. Remember that a few smaller tasks are much better than one complex task.

- Avoid login walls. Sometimes, forcing registration can scare away users instead of engaging them. This is particularly true for applications at an early stage.

Make your error messages more mobile-friendly

User-friendly mobile messages are another aspect of conversational forms. Their main goal is to help users, reduce their anxiety and extricate them from difficult situations. **Your error messages should be explanatory, concise, and not ambiguous as the user needs to be focused on solving his problem.**

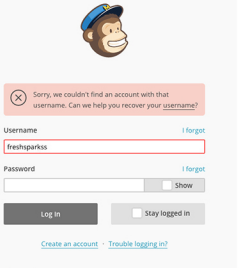
Here are some basic principles for error messages:

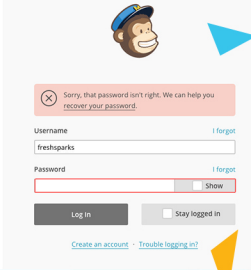
- To emphasize the meaning, use various colors.
- To allow users to correct as fast as possible, use real-time, inline validation.
- Ensure that your error messages are visible enough and are placed in the proximity to the field in error. It's not the best solution if error messages appear in the overlays.
- Find an appropriate image for your error. That will ensure better comprehension and problem-solving. Illustrations are your best friends as long as they convey the necessary information. Remember, a picture tells a thousand words!

Make navigation simple

Traditionally, websites have their menu bar at the top of the page. Unfortunately, on mobile versions it occupies too much space. To avoid it, try to make your menu a drop down or an icon on the top corner of your screen.

There's another desktop habit that doesn't relate to mobile usage. The main intention of every mobile user is to keep things accessible. If your UX makes your user tap through multiple levels of menus to find





something, they'll probably give up and leave your application. **Do your best to avoid multi level menus, keep the design minimal and focus on the key message that you want to send to your user.**

Design in Touch

Do you remember the good old days when we could interact with a computer using our mouse? Me neither. Anyway, touch design is more technical than you think; you have to take into consideration finger parameters of various shapes and sizes as well as various kinds of pressure applied to touch screens. You must also ensure that all the elements that require gesture or touch input, function correctly and are large enough to avoid overlap with near elements or misinterpretation of the touch event.

Moreover, don't forget that many devices that use styluses or even some older ones, still use directional keypads. So, don't fully rely on touch inputs. Besides, there are mobile devices and browsers that don't actually support javascript touch events. There are also some situations when a user has a combination of various input devices - when they're using a mouse connected to the tablet, for example. Though, such situations happen quite rarely. Evaluate the importance of such situations for your app and make sure you're able to handle them.

The golden rule with your push notifications

Have you ever been bombarded with huge amounts of useless notifications that annoy and totally distract you from your day?

Annoying notifications are one of the main reasons why people may delete your app.

One of the most common and simultaneously simplest mistakes that people make is that you send TOO MANY notifications. As a result, people get more notifications that they can actually handle. Don't overwhelm your users with push

for example. Though, such situations happen quite rarely. Evaluate the importance of such situations for your app and make sure you're able to handle them.

The golden rule with your push notifications

Have you ever been bombarded with huge amounts of useless notifications that annoy and totally distract you from your day? Annoying notifications are one of the main reasons why people may uninstall your app.

One of the most common and simultaneously simplest mistakes that people make is that you send TOO MANY notifications. As a result, people get more notifications that they can actually handle. Don't overwhelm your users with push messages, otherwise they might leave and delete your app.

As soon as people start using your app, they probably won't mind getting a few notifications per day. **The most important criteria is that the notifications should bring the value to your clients.**

Time your notifications

Don't send them at odd hours. Oddly timed notifications that are sent between 12am and 7am will probably wake up and annoy your users. Of course, users can always disable notifications if they want to, though it doesn't mean you should bombard them with your notifications at night. A good solution would be sending your notifications at times that would be most convenient for the users. Don't hesitate to do it only in a situation where you need to inform the user about something really important. In any other situation, the best time for push notifications is between 6pm and 10pm. And, of course, take into account your target audience's time zone!

The other important criteria for your design is that it should be both useful and intuitive. If your app is not useful, it has no practical value for users and there's no reason to use it. On the contrary, if your app is useful but its design is terrible and requires a lot of time and effort to figure out, no one will bother learning how to use it. Remember that you make your design for your users, not you. Finally, if in doubt (or even if not in doubt), **TEST, TEST, TEST!**

Chapter 8

FUTURE-PROOFING

When Pierre-Olivier Latour was travelling through Asia, he realized that the pile of photos he had taken was completely disorganized, making it inconvenient to look through them. As a result, Latour and his friend created Everpix. The essence of the application is that it sorts and arranges your photos and uploads them online for you. Everpix was used a special algorithm that sorted through photos and selected the best ones.

By 2012, Latour and his team had 55,000 application users and also received funding. However, just a year after, they were broke. So what happened?

The team became fixated on the idea of doing things perfectly. Everpix was a good app; it had great design and user experience, powerful functionality, and some very unique features (one of them was "Flashback" a feature, which shows you the photo, which you took the same day a year ago). But perfectionism doesn't come cheap. It took a year and a half for the team to develop the application. By the time Everpix was introduced to the market, Latour and his friends had neither money, nor time to raise more capital. Even though Everpix was a cool app, it simply wasn't able to attract the necessary number of users to ensure further growth.

Lesson learned

Developing an application is more than creating a great user experience. In addition to creating a product that your target audience wants, it's also about making sure that the application can bring you profit for further development. The earlier you'll be able to receive this profit, the faster you can grow the functionality. Especially today, when app marketing is dominated by players as Uber and Amazon. In the highly competitive world of apps, even the best product needs some serious marketing efforts behind it.

The safest way to avoid the mistake that the Everpix team made, is to choose the right tech stack from the very beginning. With the right technologies, you can create an MVP (Minimum Viable Product) of your app and launch it on the market in just a month! As we said, there are two main platforms that should be definitely covered by your application, such as Android and iOS.

Bear in mind that modern cross-platform app development frameworks (React Native, NativeScript, and PhoneGap) are the best cost-saving solution to proof of concept. With cross-platform development you'll be able to save money as well as allow for early market entry with an MVP. Likewise, choosing the right tech stack can also allow you to scale the app when it grows beyond the size of a startup. Speaking of scalability, it's a very important factor to be taken into consideration when creating your mobile app, especially if it's meant for a large number of users and managing large volumes of data.

What features will it need to have? How many can I lose without affecting the overall functionality of the app?

In this part we'll discuss the main features that every successful application should have. There can be an embarrassment of riches at the very beginning of an application's development. Sometimes people think EVERYTHING is needed straight away. It seems that the more features your app has, the better it is. Nevertheless, this perception is mistaken. As a consequence, it may lead to a waste of time and money. To avoid these misunderstandings, you need to make your app as "lean" as possible.

Do your best to create a great UX

It should provide utility and value to users. Do your best to persuade a user that it's your app that's worth his attention.

Allow users to customise your app

Flexible fonts, colors, settings, sizes etc.

Make sure your app is secure

Remember, data confidentiality is protected on the legal level.

Social media integration

If your users need to create an account to start using your mobile application, you should have the option for them to use their social media to make one. API integration is key.

Regular Updates

If you want to make your app popular, you need to have an ongoing software product development cycle including regular updates and evolution. Find a team to maintain the

the server. Make sure that your content consists of relevant and up-to-date information. In order to implement this point you need to update your app with all the fixes and new features, regularly. Pay attention on if your product keeps evolving and do your best to make each update as valuable as you can.

Interoperability

To ensure the required interoperability, ask your development team about the most current technologies, standards, and protocols.

Offline mode

The majority of apps require an Internet connection to function properly. Nevertheless, most of the people don't have constant internet connection or unlimited data, so allow your users to perform all the functions of your mobile app in offline mode.

Keep it simple

Despite the fact that a lot of features can make your app a masterpiece, too many features can also negatively influence on UX. This happens when features don't meet the user's requirements. Bear in mind that unnecessary features won't help your users but will make it difficult to navigate through. The best way not to make the experience slow and clunky is to provide only the most necessary features to start with.

Feedback system

As soon as you make your app available to the public, it's important to get an idea of what users actually think about your app and its features. Provide a decent feedback system and you'll be able to get information on the experience of your users. Moreover, it'll be much easier for users to report any potential bugs.

Main takeaway: Simplicity is key to success. You don't have to stuff your app with numerous unnecessary features. Determine the main goal of your app and focus on those features that will contribute to implementing this goal.

The next question is, how to actually implement your project. Which is more reliable? To learn coding and develop an app by yourself or to outsource it?

Chapter 9

BUILDING YOUR APP

Is learning to code a good idea?

You can find literature about mobile development or sign up for courses or tutorials. This is what a lot of beginners developer do at the start of their career. If you chose to do this, you should understand that coding, just like any other field, will take years to master. Of course, you can learn some tricks and programming basics in a few months, but this knowledge will hardly be sufficient to create a solid MVP.

"Why should I consider outsourcing?"

Outsourcing will allow you to focus on your core business processes. Not many people choose to learn coding and even fewer of them are both passionate about it and have talent. Find someone who is all that and give them the resources, space, and opportunity to create something truly remarkable.

The opportunity to have all the experts you need. You already know that developers are extremely important, but many other job roles are needed to bring your mobile project from page to screen. Let's not forget about QAs, who will ferret out every bug with their watchful eyes, PMs, whose job it is to keep in check your merry band of developers as well as UX/UI designers, and will take care of enhancing user satisfaction by improving the usability and accessibility.

Reduced business risks. Software Outsourcing and BPO companies face far higher liabilities than in-house staff. When you're outsourcing your digital product, you can be sure that if anything goes wrong your, they will fix it on the house.

Being ahead of the tech curve. Mobile and web technologies are constantly evolving, so for your business to be able to compete with the top players in your field, you're going to need access to the most advanced tools and resources. With the latest technologies and strategies, you're able to maintain a competitive edge in a fast-paced marketplace.

Chapter 9

Are there no downsides?

Nothing in this world is perfect. If you decide to outsource, and there possible challenges that you need to be aware of:

- **Lack of experience.** Not everyone who offers outsourcing services is as experienced as they say they are. What can you do about it? Go through their portfolio, read feedback from their current/previous clients, download and click through the apps that they've built.

- **Possible communication issues and cultural conflicts.** Even though modern technologies allow for keeping in touch 24/7 despite geographical differences, intercultural communication still poses a challenge for businesses.

- **You depend on the well-being of the outsourcing company.** There's always the risk that the guys on the other side of the sea will go bankrupt and leave you holding the bag. Choose someone established and growing.

- **Possible exposure of confidential information.** When you outsource an application that deals with sensitive data there is a risk of that data being exposed. That's why you should evaluate a outsourcing company carefully to make sure that your data is protected. Also, you need to make sure to include this aspect in the contract.

Freelancers

Freelancers are a common thing nowadays.In fact, over a third of the US workforce is now working freelance. If you're thinking of hiring a freelancer, here is what you need to know about them.

Good things first:

- **Freelancers are cheap.** Freelancers undoubtedly cost less than hiring a part-time or full-time employee. They are much more cost efficient, because you only pay them for the work they actually do.

- **They can get more work done.** Freelancers are able to set their own schedule, which means they can work whenever they choose. They have the ability to work more hours on projects, either long term or simply when necessary. Their schedules are entirely flexible, so work gets done no matter what the day brings.

- **They often specialize in one thing.** Freelancers typically specialize in one thing, and that is what you hire them for. The benefit is that they are really good at what they do. Even besides any training or certifications they may have, freelancers do the same job day in and day out.

Now, the downsides:

- **They aren't always trustworthy.** Freelancers don't have as much of a problem disappearing as your normal employees do. Deadlines are missed, work quality drops, and sometimes, people disappear.

- **You risk hiring an amateur.** At freelancers' portfolio doesn't always mean they know what they're doing.

Dedicated team

Outsourcing to a dedicated team has many similarities to hiring freelancers, but there are some differences as well.

How outsourcing to a dedicated team works

There are two main defining aspects of a dedicated team. A dedicated team is only going to be working for one client at a time, which means you will be getting their full undivided attention. Location doesn't really matter here, so you'll have the opportunity to find the best experts in the field that you want, anywhere in the world. Similar to hiring freelancers, a outsourced team usually works on a specific project for a certain amount of time.

IT is one of the biggest outsourcing fields for a couple of reasons. First of all, there's a huge market for IT services and

it's become a necessity for almost every company. Secondly, IT is often a closed system, separated from the rest of the company than other departments. Externalizing it to a certain degree is therefore easier.

The two most common mistakes to look out for

- **Hiring the wrong team.** Finding a team that suits the project you need is the most vital step of the outsourcing process. You might be tempted to go cheap, but that will often be a grave error. Base your choice on quality, not on price; a good outsourcing team will have cost-efficiency high on their priority list.

- **Bad communication.** Secondly, and this applies to all collaboration with external workforces, communication will make or break the business relationship.

The advantages of a dedicated team

- **Skilled and experienced professionals.** Growing and nurturing a team with the right skill set for the project you want is a task that can take months, if not years. This way, you won't have to.

- **Established teamwork and communication.** It's not only about knowing how to build websites, fix software or do bug testing. If the team members don't know each other, they will be inefficient. In a dedicated external IT team, people know the strengths and weaknesses of their colleagues and can act accordingly.

- **Cost-efficient and more flexible to eventualities.** External IT teams are cheaper than growing your own. Especially if you're a small to middle business that can't afford that kind of long-term investment. And you won't have to keep your external IT around all the time.

Main takeaway: There are many ways to get an app built and benefits and downsides to all. Consider whether you should build the app yourself, hire someone in-house, freelancers, or outsource it to a dedicated team.

Chapter 10

MONETIZATION AND BEYOND

App store and monetization

With open source being the dominant force in the software world, there is little wonder that online customers are used to having most of the things on the Internet for free. In that case, what are the options for monetizing your app if the customers aren't really happy about parting with their money?

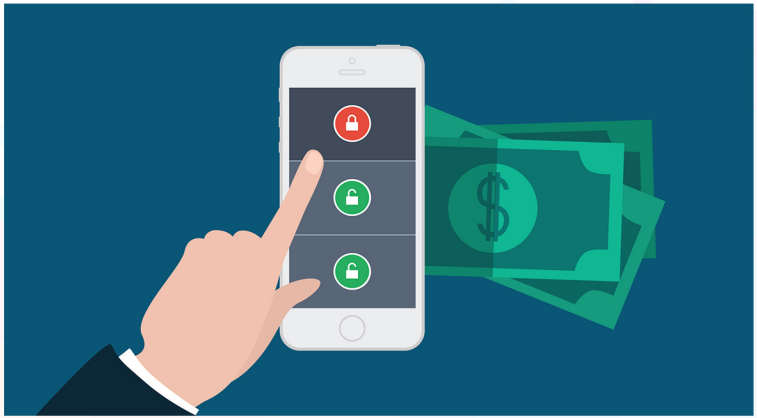
But before we elaborate on a few options for monetization, let's have a look at raw figures. According to Sensor Tower, the revenue from mobile apps on App Store and Google Play has increased by 35% in 2017, making it a 58.6 billion dollar market. Let's have a look at the most spread app monetization models.

In-App advertising

In-app ads are the most commonly used way of monetizing mobile apps. According to Statista, the revenue from in-app advertisement will reach $117.2 billion by 2020. For the record, it was only $40.5 billion just three years ago.

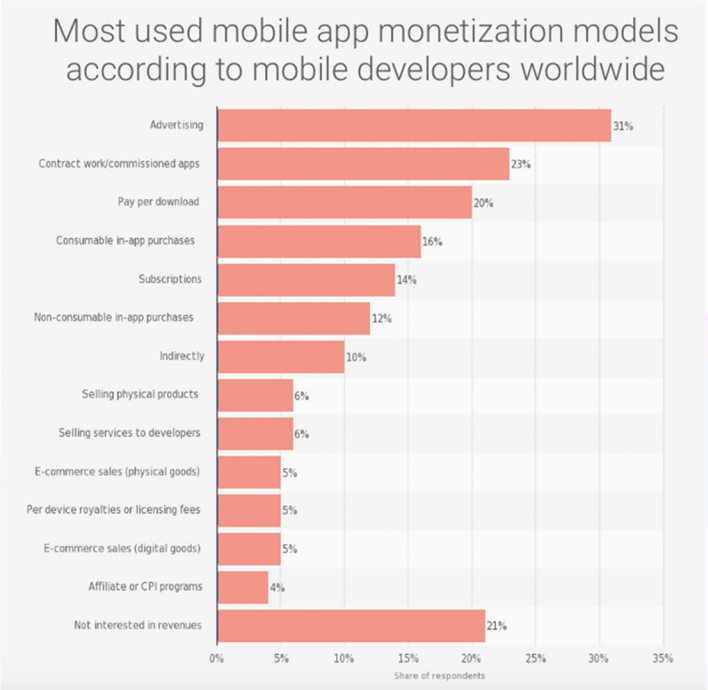
The benefit of such an approach is that your app can remain free-to-download, encouraging more people to use it. There are several types of in-app ads:

- **Banners** are usually placed either at the bottom or at the



top of your app's interface. However, because of that, users find them more distracting than other kinds of advertising, which makes banners a bit ineffective.

- **Transition ads** are typically inserted between certain pages within the app. You can think of them like tv commercials as they are usually short videos. This type of in-app ad works best for mobile games as well as other app genres, if implemented appropriately.
- **Reward ads** are launched after/during a certain event or a campaign in the mobile app. Users receive a notification, which offers them some sort of a discount, gift card, coupon, etc.
- **Notifications** appear in the status bar of the mobile device, letting the user know about the promoted content. The main downside of notifications is that they can be obtrusive, so you should be careful when implementing this type of in-app ad.
- **Native ads** are considered the new king in the world of content



based apps since they are mostly displayed in the news feeds. For example, you may have seen native ads in Facebook. The reason why they're called "native" is because the format of the advertisement matches the look of the app, which makes it a great choice in terms of user experience.

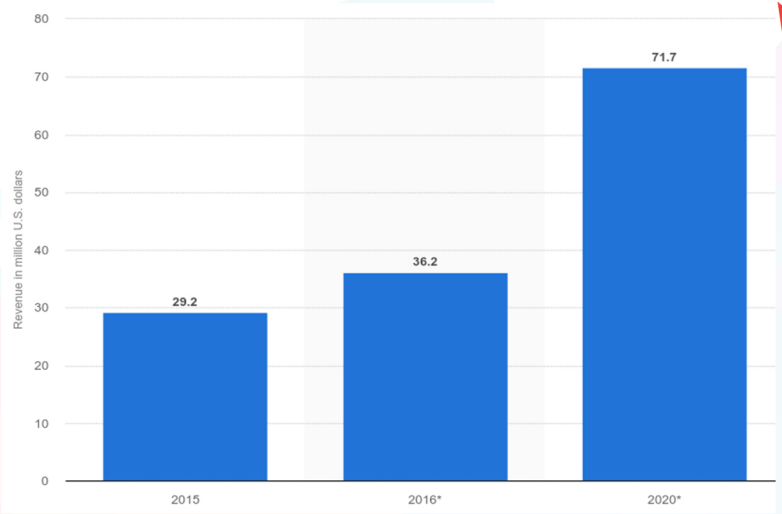
However, if your audience doesn't like the ads, you can still take advantage of the situation by allowing users to turn off the annoying messages after purchasing the ad-free version.

In-App purchases

In 2017, about 49% of mobile app owners implemented in-app purchases into their non-gaming apps. By that time, 79% of gaming apps had already been using the feature. In terms of revenue, this type of monetization is an indisputable winner. In-app purchases account for more than 50% of app revenue. The top grossing apps in Apple's App Store, the top 30 apps, are all free but offer in-app purchases. Last year, they generated $37 billion worldwide.

Typically, in-app purchases for a free app allow users to:

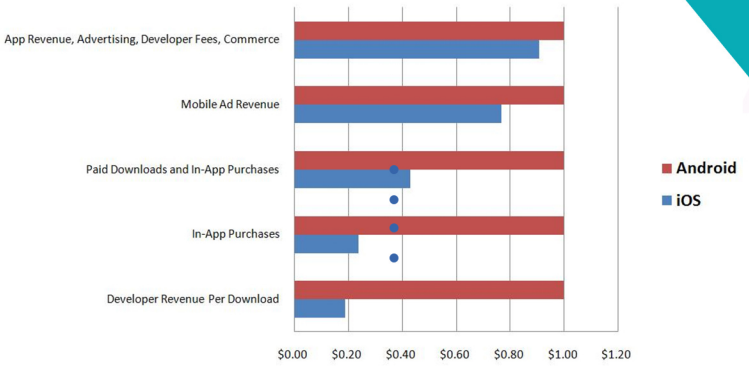
- Unlock new features
- Purchase a subscription
- Buy virtual goods
- Purchase additional content



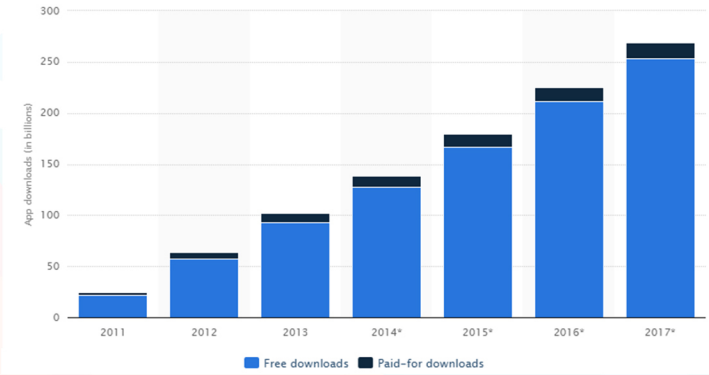
Users who download and engage with an app first tend to spend more on in-app purchases. In-app purchases require well-designed incentives to purchase that doesn't take away from the user experience. However, once you prove your app is worth it, you gain more users who are willing to pay for what they get.

One of the most popular ways of implementing in-app purchases today is freemium-based apps. This advertising strategy assumes that you offer users an app that can be downloaded for free, but requires paying money to unlock some of the advanced features. When using freemium, you should be careful about the features that you exclude from the basic version of the app.

It's very important that the users can freely access the core functionality. For example, if



Number of free and paid mobile app store downloads worldwide from 2011 to 2017 (in billions)



your app takes those photos, users have to be able to take black-and-white photos right after downloading the app. At the same time, things like cropping, resizing, etc. can be kept for the premium version.

Finally, there are subscription-based apps. Their advantage is that they bring stable, recurring revenue. The brightest examples of such apps are Netflix or Spotify. However, subscriptions are more demanding to monetize. They need proper marketing and high positions in app store rankings.

Paid app download

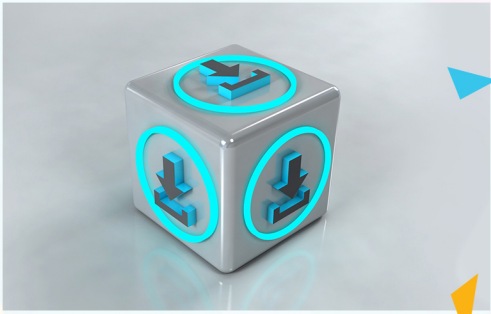
The most obvious way to generate revenue through your app is by selling it in the app store. With a paid download, users pay a one-time fee for downloading the app.

This model can be challenging because it's difficult to convince users to pay without having tried the app, especially with so many free options available in the app store. On the Apple App Store, the price can range from $.99 to $999.99 and Apple and Google both take 30%. Putting a price on your app is often a barrier for people to download your app. That's why, before making your app paid, make sure that:

- You have a strong marketing and social media presence
- The app can offer added value over free and similar counterparts
- The value corresponds to the price

Main takaway: There's plenty of forms of monetization. You just have to find out which will work in your case, and if it won't, just try another one.

Your focus, however, should be enlarging your customer base - the number of people who actually use your apps. The bigger that number, the bigger the potential for monetization. So focus on making the app great and the profits will follow.



Chapter 11

GOING WELL? SCALE AND MAINTAIN!

Congratulations! We came to the last point in mobile app development. Your app has been finally developed and you've joined the big leagues! If your app really is helpful, users will undoubtedly find out about it. However, you can significantly accelerate this process and scale your app much quicker, if you utilize some inorganic measures.

Let's be honest, you won't get a lot of users if you stand around and just wait for them to come to you. Unfortunately, life just isn't that kind. In this chapter, we'll talk about a few ways to successfully scale your app and reach your first million users.

Tell your customers you exist

To download your app, people just need to know about it, right? You can't launch it and expect millions of users to just fall into your lap. You'll have to work hard to add new users as well as keeping the ones you've already acquired; the question is how to enlarge your user base? The answer is quite simple, you need to help people to find you in the app stores, but that's actually not as simple as you'd think. You'd be amazed by the number of apps that are currently available in the App Store.

Impressive, isn't it? And here's a statistic that shows the number of available apps in the Google Play Store.

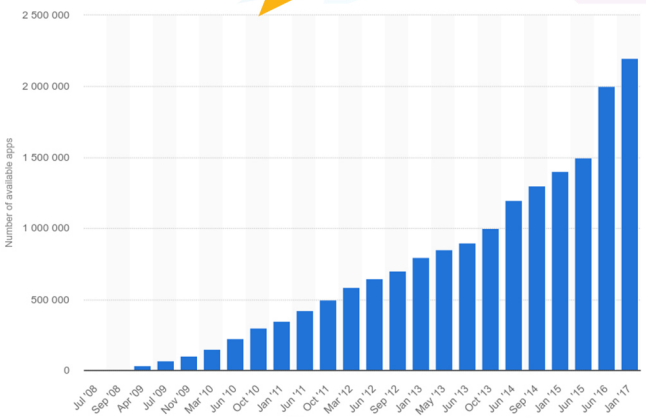
As you can see, there are a HUGE amount of apps, it'll be quite tough to get on the first page of the market. The first things users see are icons, screenshots, descriptions, etc. Make sure they're attractive enough. If not, no promotion will help you. Once you're clear on this, you can make your app easier to find using the app store optimisation features or search engine optimisation.

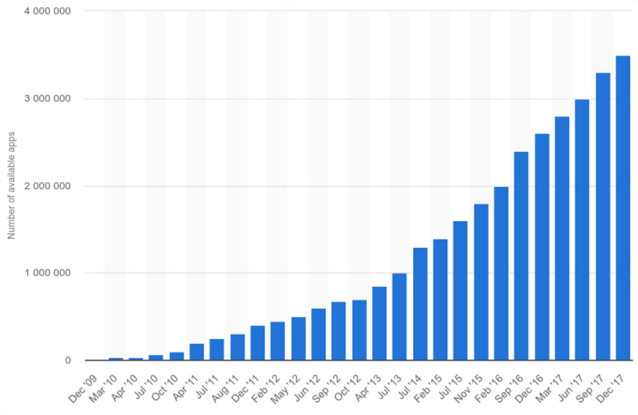
This will allow you to significantly increase the number of downloads.

The main point is that you make your scalability much easier by implementing various organic channels as part of your marketing strategy. Pay special attention to long-tail keywords that your target users are looking for, connect with influencers via social media, and pitch guest blog posts.

Ad copy is important

It's well known that paid advertising can contribute to attracting a great number of users to your app. Nevertheless, you'll need to rework your ad copy if you want to permanently scale an influencer marketing campaign. Here's a tip - an ideal ad copy should be a reflection of the visitor's end goal and offer a great deal.





While iterating your product in the process, you need to iterate your acquisition channel. To put it simply, if your ad copy doesn't work properly in one channel, you could try other channels such as Linkedin, Instagram, and Facebook.

Create your own army of influencers

When you've successfully deployed both organic and paid user acquisition channels for your app, you can feel proud, However, don't step back, this is only half the battle. The next step is to tap into the potential of influencers. The more influencers your app engages, the more chances it has to be downloaded. Nevertheless, that doesn't mean you can launch your app and immediately contact your influencers to convince them to promote your app. You must have a good strategy as well as proven research. Only after this step you can consider influencer marketing.

Stick to your development team

It's probably one of the most important factors that you need to take into account while scaling your app. Bear in mind that your dev team is your family. It's these guys that have been with you since the very beginning of your app development. It's these people who have gone with you through a lot of troubles and know every aspect of your app. Sometimes you just need to believe in people. Encourage them, show your dev team how you appreciate their work. And undoubtedly, they'll do their best to make a masterpiece out of your app.

Have a Clear Goal

It doesn't matter how much time you spent trying to reach your first million users. The most important part in scaling your app is that you need a clear goal as well as commitment to see this goal through to the end. You must understand who you're targeting, provide people with an incredible product, make your app searchable to your target audience, and finally learn who their influencers are. Having taken into account all of these factors, you may significantly boost your content reach and help convince people to install your app and keep it.

Maintenance

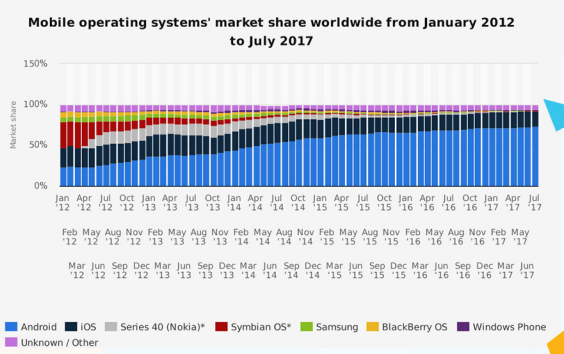
After spending hundreds of hours thinking about your app concept and even more hours on its implementation, you've finally developed your masterpiece. Nevertheless, your work is not finished. There's a popular misconception that a mobile app can be built once and deployed forever. Unfortunately, it's not how it works in practice. If your app stably works today, it doesn't mean that it will always be so. Times change and so do requirements for apps.

Various versions of operating system

Undoubtedly, your app perfectly works on the current operating system. However, every year people get a new version of iOS and Android. What happens then? Each time one of these updates hits your users' devices, there is a high chance that some features of your app will fail to work properly. That's why your maintenance plan should include proactive audits that take place as soon as developers get a beta version of a new operating system. That will allow you to get an idea of those changes that'll be necessary. You should make all these changes before your users get a new version of the OS.

Security updates and future continuous development

As far as mobile app maintenance goes, you will need to spend time on security updates. Such updates are crucially important as they will allow securing your app as well as your users' information/data. Remember that no one is safe from bugs and various security holes as they can pop up at any time. Make sure that your programming team is capable of fixing any issue as soon as they arise.



The user interface

Despite the fact that any user is always glad to see new features and improvements, s/he also expects the User Interface to stay relatively modern. A UI that's cool today, will be outdated within a year. Remember that all users want an application to look and be; sophisticated.

Another important aspect of maintaining the UI is taking into account every year manufactures are releasing new devices with various screen sizes (this is especially true for to Android users). Imagine how a UI that was designed for a small screen will look like on a larger one. Probably, that won't be the best experience.

New enhancements and features

Here we can see several options. In the case of you developing your app with a specific goal, and it perfectly achieves it, adding new features may be not so important. That's an exception to the rule: if your app is important to business and users, it'd be better to update it at least once a year with new features or some enhancements to previous features.

Software Libraries

If you're using software libraries in your application, make sure your development team keeps them up to date when any contained libraries are updated.

Performance Optimization

Bear in mind a good rule: "Your system can always perform better." There are major and minor changes that can be made to your server-side system and front-end integration that will be able to speed it up. You'll also be able to integrate technologies to speed up your server-side system without changing the infrastructure. Here are the most common integrations.

File Compression. Compression will be especially useful if your application uses large files. For example, it can be images that need to be loaded from the server. Imagine a situation when your friend takes a photo and uploads this photo to an application. In its turn, the application sends this photo to the server. After this, the server compresses the file and stores it. How can it actually be useful? This is pretty simple. You'll receive a photo that is smaller. Consequently, the loading will be much quicker.

Caching. Caching is integrated into the visible (front-end)

application. This feature allows for storing information locally on the device. Caching is very useful, as it removes the unnecessary loading of information from the server (that usually slows the system down).

Your users should see that you actually care

Users wait for new updates. It's the updates that keep them using your app. Having kept your app updated with new features, even if they're small, as well as embracing the design language of your target platform, you'll be letting your users know that you really care.

When you release a new update, you let your users know that you haven't forgotten them. You let them know that they can rely on your application no matter what. That any data stored in the app that they access on a regular basis will continue to be available to them. Your users, in turn, will thank you by greater installs, more market penetration, as well as a better reputation.

Here's a list of other areas that might actually change an app's release and thus, will need maintenance.

- **Programming language.** You need to update apps based on changes to various programming languages (such as Swift, Objective C, Java or HTML5)

- **Styles and design.** From time to time you need to change styles and designs. The same things relate to usage patterns.

- **Infrastructure.** Bear in mind that any time the infrastructure on which your app is hosted chan- ges, you need to update your app. This is especially when you switch from self-hosting to a larger hosting platform.

Main takeaway: As you can see, app maintenance is as equally important as its development. When we talk about app maintenance, it's most important to keep your app up-to-date as well as fighting off bugs. It requires constant vigilance and a good system in place.