Designing Mobile Apps

Javier Cuello & José Vittone

Designing Mobile Apps

Javier Cuello & José Vittone

Designing Mobile Apps
©Javier Cuello
©José Vittone

First Edition: June 2013
Version 1.1
ISBN: 978-84-616-6159-6

Editor: Catalina Duque Giraldo
Illustrations: ©TugaMOVIL
Photographs of the Authors: © Comitè Fotogràfic
English Translator: Constanza Mendoza
English Editor: Emily Markley Stewart

You may also find this book at www.appdesignbook.com

Contents

Introduction

Designing apps is no easy task, especially if you haven't worked with them before. And if you're coming from the world of web, you're stepping into a completely new context.

Though they may seem a recent phenomenon, the reality is that apps have been among us for quite some time. They've become increasingly more popular as of late, attracting now only users but also designers and developers, who are making the most of the opportunity the mobile screen offers. When it comes to apps, technological advancement means better experiences, supported by visual design.

For designers, making the switch to mobile phones can prove quite a challenge. However, it is a great opportunity to get into a market where clients are demanding more and better communication and promotional tools.

Of course, we haven't forgotten about programmers. The market demands high-quality apps that stand out from the rest, and producing a good-looking application is no longer enough. Apps have to function well and be easy to use.

Designers and programmers must work together to create apps that consider every single detail. First impressions are lasting impressions.

In the chapters of this book, we will develop several concepts that will help you to produce high-quality mobile app projects.

Why This Book?

Documentation and information on the subject of app development and design is scattered all over the web. It is highly fragmented and hard to find, an ocean of information.

Our beginnings in app design weren't easy. Everything was very different from how it is today, and there was no one we could ask or anywhere could we could look to find answers to our questions. A few but very valuable years have gone by since then, and what we've learned, by necessity and on the fly, has been very useful. Now, we want to share our knowledge with you.

That being said, we encourage you to be curious and investigate on your own. Consider this book to be the starting point of your journey that will take you to new places. It should be a reference point for you to forge your own path.

Do, try, and make mistakes! Start designing as soon as possible, even small personal projects. Pay attention to the work of others. If you don't have a smartphone, find one and explore it. Investigate and play with it until you get the hang of it, learn its secrets.

How is the Book Structured?

In the following chapters, we will guide you through the entire app design process, from conception to release. We'll delve into interaction and interface, also providing you with related links for you to further expand your knowledge.

This book is not only about visual design. Our goal is to dig deeper and discuss subjects like the market, user testing and promotion. We firmly believe that a designer must be present throughout the entire development process, that he or she should be involved in and committed to each and every stage.

Even though we talk about app design in a general way, on several occasions, we compare the most common mobile operating systems: Android, iOS and Windows Phone. Android and iOS are, currently, the leading operating systems, and Windows Phone, though not as significant yet, is on the rise. What's more, it has departed from the traditional style of design with a very different kind of proposal.

Additionally, you will find interviews with top design professionals, including Loren Brichter, Dustin Mierau, Irene Pereyra and Erik Spiekermann. They have shared with us their perspectives in the world of apps, enriching our content enormously.

At the end of the book, you will find a glossary of common terms for reference. We want to make sure you understand every word we write.

About the Authors

Javier “Simón” Cuello

Born in Mendoza, a small city in western Argentina known for its beautiful views, mountainous location and high-quality wines, Simón found his calling for graphic design at a young age. Beyond his visual education, he has also taken a specific interest in interactive and new technologies.

Now living in Barcelona, Simón has worked in mobile app design from the outset. He has created products for the very first versions of iOS and Android and worked for clients like Yahoo! and Zara.

Beyond design, Simón loves to travel. And when he’s not out and about, you can find him thinking up new ideas and taking up independent projects managed by small teams. Follow him on Twitter at @millonestarde.

José Vittone

Also from Argentina, José is from the province of Buenos Aires. He and Simón met in Barcelona while completing a Master's program in Elisava.

José's work at Usolab taught him what usability and user-centered design really mean. He has been designing interfaces for eight years now, focusing on conceptualization and production and juggling several mobile apps at a time.

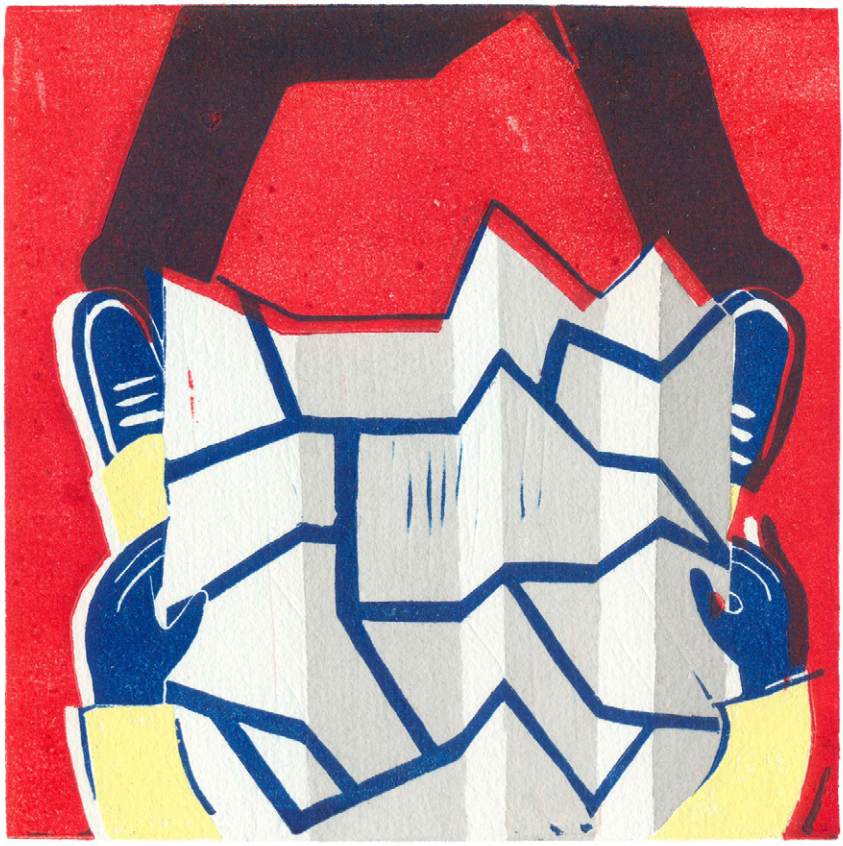
José is passionate about the small details that make a big difference. He is innately curious to find out how things work and is always on the forefront of technology. Follow him on Twitter at @josevittone.

About the Illustrations

Each chapter presents content with accompanying images. The drawings are the work of TugaMOVIL[1], a small Polish-Chilean studio based in Barcelona, the owners of which we have the good fortune to call our friends. Pay close attention to these illustrations, created by Maga and Seba, that will accompany you in this journey we are about to begin.

Let's go!

1 http://www.tugamovil.com/appdesignbook/



1

Applications

Applications have been among us for quite some time. Before delving into how to design apps, it's important to know a bit more about them. What are the different kinds of applications, and what are their characteristics? What's the difference between an app and a mobile website? Find the answers to these questions and more.

What is an Application?

Fig 1.1. There are thousands of apps available in the App Store.

Applications, also known as apps, have been around for a while. In fact, they were included in Nokia or BlackBerry operating systems years ago. Mobile phones from that period, what we now call feature phones, had smaller screens and generally weren't tactile.

So what is an application? Essentially, it's a piece of software. To provide an analogy, applications are to mobile phones what programs are to computers.

Nowadays, a plethora of applications are on the market, coming in every imaginable shape, size and color. Initially, however, apps were focused on improving personal productivity: alarms, calendars, calculators and email.

It was the iPhone that changed everything, generating new business models. It transformed applications into profit generators for developers and created a new opportunity for app marketplaces to emerge, such as the App Store, Google Play and the Windows Phone Store.

App developing tools have also become available for designers and programmers, making it easier to produce and launch an application now than ever before.

Apps vs. Mobile Websites

Applications and mobile websites share the same screen, but that is where the commonalities end. While apps have to be downloaded and installed before use, a website may be accessed through an internet browser. But because of the size of the mobile screen, many websites can't be properly visualized on such devices.

Mobile websites that adapt specifically to a mobile device are called responsive websites and are an example of liquid design. The content takes the shape of its container, displaying information as needed. Thus, whole columns, text blocks and graphics from a website can adapt to a certain space differently — or even disappear — according to whether they are being accessed from a mobile phone, tablet or desktop computer.

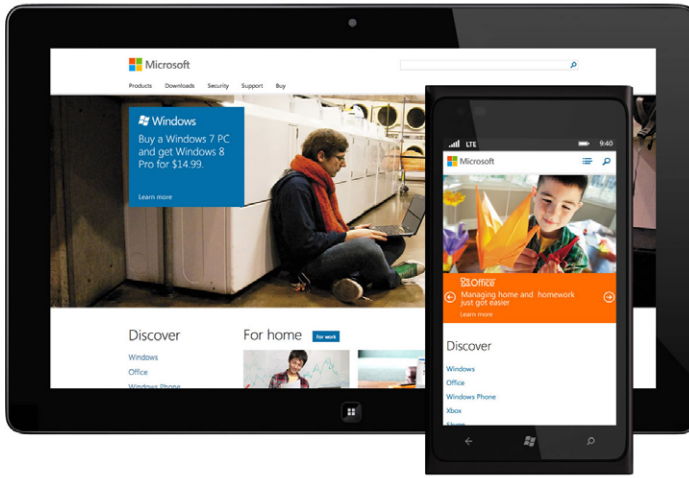


Fig 1.2. Responsive design adapts to the device where it's visualized.

Those who already have a responsive website may consider whether it's necessary to design an application, the conclusion depending on the goals of a business and the characteristics that differentiate applications from websites.

For example, apps can be utilized without an internet connection, and they can also access certain hardware features of a phone, such as sensors. On the whole, an application offers a better user experience, avoiding excessive wait times and rendering content navigation more fluid.

It's not always necessary to choose between one and the other. Websites and applications are not competitors, they're complementary. Websites are often useful as an information channel to motivate an application download.

Mobile First

When the time comes to design an application, a website sometimes exists as a precedent. In such cases, the app must draw from all of the features and content conceived for the web and adapt them so that they make sense for a mobile phone.

In other cases, design starts from scratch, with no website or application to start from, and a decision has to be made about which of them will be created first. It is here that we come upon the concept of *mobile first*, which implies articulating the design process for mobile from the outset.

When the mobile phone is the starting point, it is imperative to concentrate on the essentials of a product and focus only on what's important for the device.

Once the application has been designed, the question of adaptation for other devices arises. How does one extend and escalate content and rethink layout? All devices have different uses, and when it's time to adapt design, it is essential to keep in mind the specific features of each.

Mobile first is a new concept and emerging trend that has not yet consolidated. Currently, it is only one way of approaching

the design process, and as such, the convenience of working in this way must be evaluated before getting started[1].

App Design and the Development Process

An application's design and development process spans from the inception of an idea to analysis once it's hit the market. During the different stages, designers and developers work simultaneously and in close consultation.

Beyond the design and development perspective, it is also important to take into account the roles of coordination, client participation and company stockholders. These issues will be explained in detail in the following chapters.

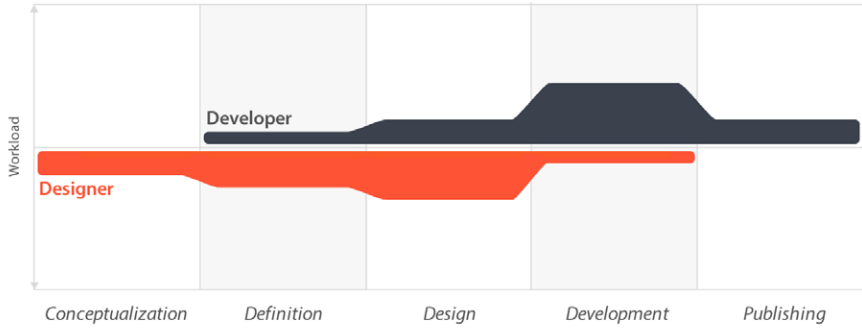


Fig 1.3.
The design process involves different stages, where designers and developers work simultaneously with varying workloads, depending on the moment.

1 Wroblewski, Luke. Mobile First. http://www.lukew.com/ff/entry.asp?933

1. Conceptualization

This stage entails the formation of an idea for an application, taking into consideration the user's needs and problems. The idea is a response to preliminary research and a posterior concept viability check.

- Ideation
- Research
- Formalization

2. Definition

In this step of the process, end users are defined through methodologies like *Personas* and *User Journey*. It is here that the foundations for functionality, which will determine the range of the project and the design and programming complexity of the app, are laid.

- User definition
- Functional definition

3. Design

In this stage of design, previously-discussed concepts and definitions become tangible, initially as wireframes that allow for the creation of the first prototypes for testing with users. Then, a finished visual design is provided to the developer as separate files and model screens for programming the code.

- Wireframes
- Prototypes
- User tests

- Visual design

4. Development

The programmer is in charge of bringing designs to life and creating the structure upon which the application's functions will be based. Once an initial version has been built, time must then be dedicated to correcting functional bugs and ensuring proper performance in preparation for market approval.

- Code programming
- Bug correction

5. Publishing

The application is finally made available to users in stores, after which it is monitored by means of analytics, statistics and user comments. These are utilized to evaluate its behavior and performance, correct mistakes, make improvements and update for future versions.

- Launch
- Follow-up
- Update

Types of Applications

At the programming level, there are several ways of developing an app. Each has different characteristics and limitations, especially from a technological standpoint. And though this may seem to be outside of the realm of the designer, the fact

is that the type of application chosen conditions visual design and interaction.

Native Applications

Native applications are those that have been developed with the software each operating system offers, generically referred to as Software Development Kits (SDKs). Android, iOS and Windows Phone have different SDKs and native applications designed and programmed specifically for each platform in their respective SDK languages.

These kinds of apps are downloaded and installed from application stores. In the case of Android, there are certain exceptions (see *Chapter 16*).

Native applications are updated frequently, meaning that the user has to download them again and again in order to

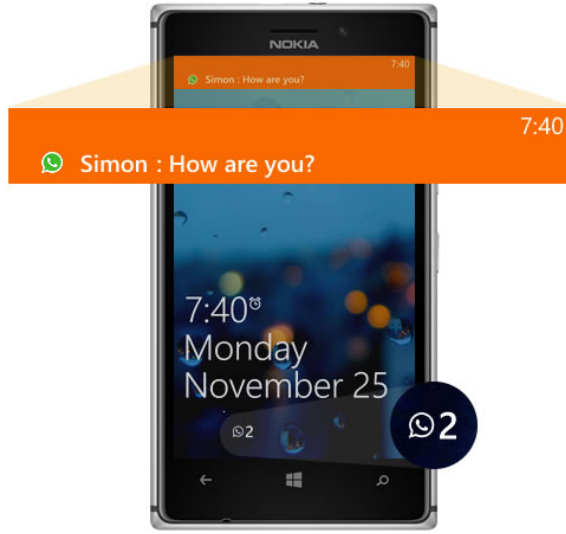


FIG 1.4.
Native apps allow the use of the notification system.

receive the most up-to-date versions that correct bugs or add new features.

One characteristic of native apps that is generally underestimated is the fact that they can use operating system notifications to alert the users even when an application is not being used. An example of this: WhatsApp messages.

And because native applications don't require an internet connection to work, they offer a more fluid user experience and true integration into the phone. Native apps use all of a device's hardware, including cameras and sensors (GPS, accelerometer, gyroscope, etc.).

At the design level, these types of applications have an interface that is based on each operating system's guides, achieving more cohesiveness and consistency with other applications and the OS itself. This favors usability and benefits users directly.

FIG 1.5.
Facebook has a web app and a native app.

Web Applications

Web app programming is based in HTML, JavaScript and CSS, tools already familiar to web programmers.

In such cases, SDKs are not used, making it possible to program independent from the operating system. This is why these applications can be used easily in different platforms, without major inconveniences or the need to develop a different code for each particular case.

It isn't necessary to install web apps, since they are visualized with on the smartphone browser as if they were a normal website. They're not distributed in an app store, meaning that they must be marketed and promoted independently.

FIG 1.6. Netflix has a hybrid app that looks practically the same in iOS and Android.

Because these apps work from the web, users are always ensured the latest version, meaning no updates required. But unlike native apps, web applications require an internet connection to work properly.

Additionally, web apps have some restrictions and inconveniences in relevant areas, such as memory management. What's more, it is impossible to maximize the power of all of the different components of a phone's hardware.

Web apps usually have more generic interfaces, independent of each operating system's appearance. User identification experience, as far as navigation and interaction elements are concerned, tends to be less significant than in the case of native applications.

Hybrid Applications

These applications are a sort of combination of the aforementioned. Development is similar to that of a web app (with HTML, CSS and JavaScript), and once an application is finished, it is compiled and packaged in such a way that the end result looks like a native application.

This allows developers to use an almost identical code to obtain different apps, for example, one for Android and another iOS, and distribute them in two different stores.

As opposed to web apps, hybrid applications, via libraries, are able to use phone capacities much in the same way native apps do[2].

Hybrid apps have a visual design that is not significantly related to an operating system. However, there are ways of using native controls and buttons from each platform to comply with a particular aesthetic.

2 Seven, Doug. What is a Hybrid Mobile App? http://icenium.com/community/blog/icenium-team-blog/2012/06/14/what-is-a-hybrid-mobile-app-

There are a handful of tools available to assist with the development of hybrid applications. Apache Cordova[3] is one of the most popular, but there are others out there as well, such as Icenium[4].

What Should You Use?

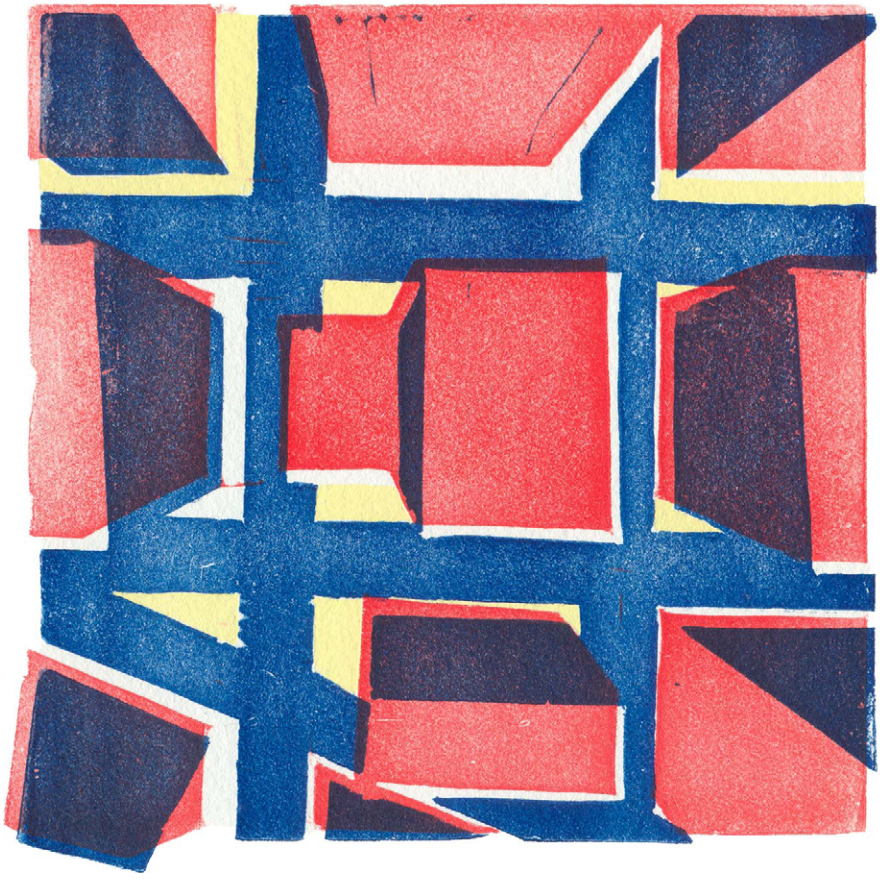
The selection of which development format to use is determined by a few fundamental factors and the way user experience will ultimately be affected. In cases where factors like app availability without internet connection, notifications and hardware resources are important, going the route of a native application will be the best option.

If none of these things are really important for the application, it may be easier to design a web app and draw upon previous knowledge of website development. This often implies lower development costs and a more agile way of working.

Native applications offer better user experience, and, above all, performance. That's why apps like Facebook and LinkedIn, which were originally hybrid, have become native. Additionally, they comply better with design guidelines for each operating system.

All of this in mind, we will be focusing on native applications from here on out. In the chapters to come, we will study native apps and explore what sets them apart.

3 http://cordova.apache.org/
4 http://www.icenium.com/



2

Grasping the Possibilities

Before developing the idea of an app, it is necessary to make some decisions that will set the stage for the project. In this chapter, we will explain the different options for making an economic profit off of your application and the equipment and resources you need to move forward.

App Categories

One way of grouping apps is according to the content they offer the user. The category of an app will condition, at the design level, how detailed the interface will be and the possibilities for monetization.

It may seem unfair to label apps, because they hardly ever belong to just one category. In fact, many are put in different categories at the same time. For all intents and purposes, we have divided them as follows:

Entertainment

This is the dwelling place of gaming apps and all other apps that help users to have fun. The graphics, animations and sound effects engage the user's constant and uninterrupted attention.

They generally have a design that is not strictly bound by platform guidelines. For example, Angry Birds[1] has similar graphics across different operating systems.

As for business models, they're generally quite flexible, because applications in these categories are sometimes down-

Fig 2.1. Angry Birds is one of the most popular mobile games on the market right now.

1 http://www.angrybirds.com/

loaded in full paid versions and in other instances offer other purchased possibilities (by item, level, etc.).

Social

Social apps are those oriented towards communication, contact networks and interaction among users. The best-known social app on the market today is, of course, Facebook. Path[2], Twitter[3] and Instagram[4] are also examples.

They are generally free of charge, and their business models depend on the personal data obtained from users or in-app purchases, such as stickers in the case of Path.

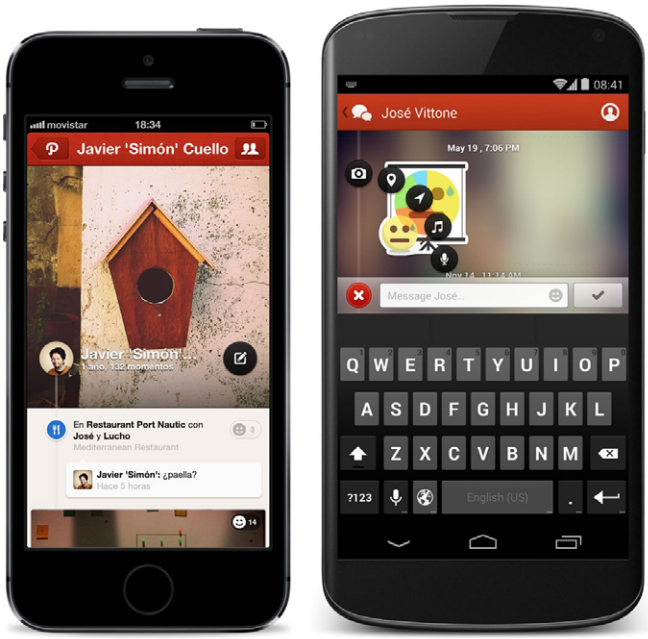


FIG 2.2. Path is a social network that allows a maximum of 150 contacts.

2 https://path.com/
3 https://twitter.com/
4 http://instagram.com/

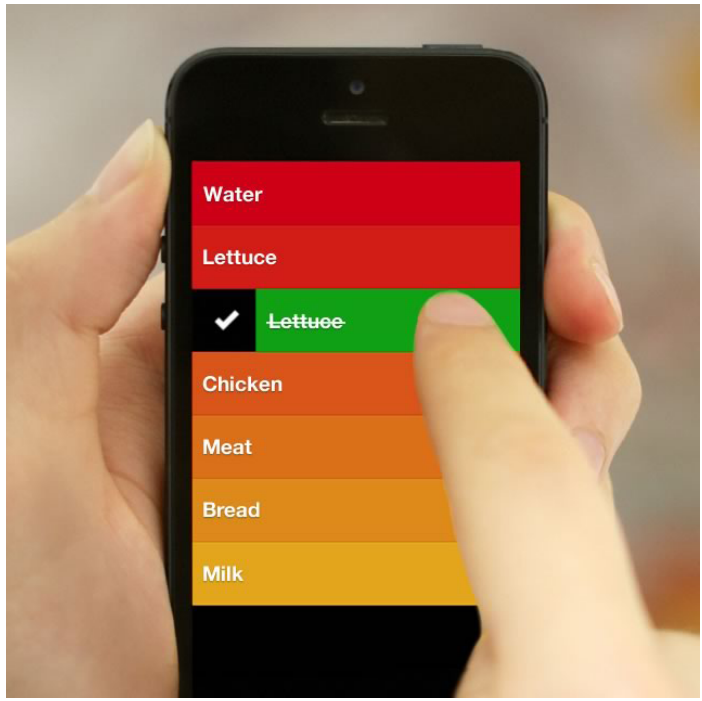


Fig 2.3. Clear revolutionized to-do apps with its particular style.

Utilities and Productivity

As apps in this category are more commonly associated with the business sector, they provide tools for solving problems that are quite specific and are based on the execution of concrete, quick tasks. Efficiency takes priority over all else.

Among these, we commonly find to-do lists (Clear[5] and Flow[6]) and apps for work teams (Basecamp[7] and Evernote[8]). Users find value in tools that allow them to simplify their daily lives.

5 http://www.realmacsoftware.com/clear
6 http://www.getflow.com/
7 http://basecamp.com/
8 https://evernote.com/

In this category, business models vary. If the app is only available for mobile devices, it's normal for the app to have a cost, as in the case of Clear. On the other hand, apps associated with a cloud service for which the user is already paying, such as Basecamp, can be downloaded for free.

Educational and Informative

Educational and informative apps are used as transmitters of knowledge and news. In these apps, access to content is the most important factor; therefore, legibility, ease of navigation and search tools are fundamental. Some, such as Articles[9], are paid, whereas others, like Wikipedia, are free.

FIG 2.4. Articles is a good alternative to the Wikipedia app, even though it's paid.

9 http://sophiestication.com/articles/

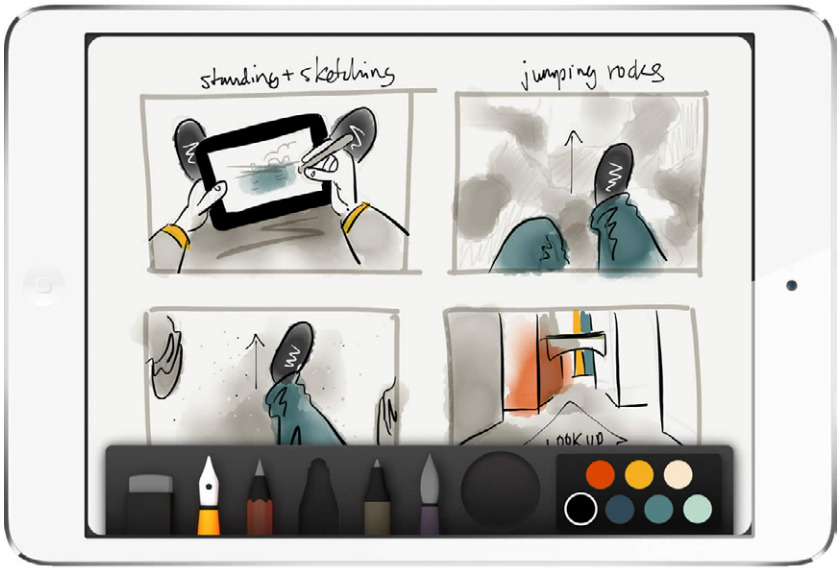


Fig 2.5.
Paper for iPad is a creation app based on the drawing block metaphor.

Creation

These apps are centered on user creativity, for example, apps that allow users to edit videos, alter photographs, produce sounds or write.

Even though they are usually paid, some offer less complete free versions or have additional components and functions that must be purchased individually. Besides basic tools, Paper[10] also offers additional drawing elements that can be bought within the app itself.

10 http://www.fiftythree.com/paper

Free, Paid or a Little Bit of Both

This is a subject that stirs quite a bit of a debate. Apps are a relatively new product, and even though they often follow the same business models already in use for other kinds of software, they are still surrounded by a cloud of uncertainty when it comes to how to make a profit.

That being said, an app's objective may not be to make money at all but instead to serve as a communications channel with users or to extend the scope of a brand. For independent developers, this is a personal choice, and for brands, it depends on global goals.

All models, both free and paid, have pros and cons. So what are they?

Free Apps

Without a doubt, the biggest benefit of a free app is the scope it can reach. These types of products have zero download barriers.

This first step is spreading the word and make sure it is clear to users that, in downloading, they've got nothing to lose. To a certain extent, this narrows down the expectations for a product. Nobody expects something that is free to be absolutely awesome.

This can be especially useful for products that still require development to fully mature. In such cases, their availability on the market makes it possible to take advantage of certain user behavioral indicators—how they use the app, frequency, etc.—to improve future versions. A free app can also serve as a window to promote a paid version of the same app or other apps from the same developer.

An inconvenience that is worth noting when it comes to free apps is that, because of fierce competition, it is much more difficult to gain visibility in app rankings and stores. Getting

to first place as a free app requires many more downloads than with a paid product.

Nevertheless, the fact than an app is free does not mean that it is impossible to make money from it, as we will see in a bit.

Paid Apps

Because they require a significant number of downloads to be profitable, it is harder to achieve success with a paid app. Moreover, putting a price on a download creates a barrier that many users are hesitant to cross, especially when they aren't already familiar with the product. Except in some very specific cases, such as Cut the Rope[11], paid apps rarely become big hits.

Whether or not a user is willing to pay for an app depends on many factors, one being the availability of free alternatives. If there are two similar apps, one paid and the other one free, the odds are, obviously, that the free app will have more downloads.

The store that sells the app also determines the possibilities for charging. For example, iPhone users have become more adept at paying for an app than those who use Android or Windows Phone. However, it also takes more quality to satisfy their expectations. This happened for quite some time with WhatsApp, which was free in one store and paid in another even though its functionality and features were practically identical.

Independent of the platform, the user pays for value, something that the app contributes that other apps don't have and justifies the price—a price that many times is conditioned by the market and competition.

One parameter users generally take into account before paying for an app is user ratings. The more and higher the reviews are, the better. Users are much more likely to pay for

11 http://www.cuttherope.ie/

FIG 2.6.
Cut the Rope and WhatsApp are two apps that, in spite of being paid, have been downloaded a lot.

an app with many reviews and an average rating of 4.5 stars than for an app with no reviews and negative ratings.

Lastly, it's important to consider that each of the stores—Google Play, App Store or Windows Phone Store—charges a 30% commission on the sale price of an app. This means the developer only keeps 70% of what the user pays for the download.

Freemium

The freemium model (a combination of the words free and premium) is a blend of the two kinds of models previously discussed. These apps may be downloaded for free with basic and limited use. More advanced features can then be unlocked with payment.

This model combines the best of both worlds: it can reach a wide audience because it's free, but it can also offer more advanced services for users to take advantage of and enjoy.

The hard part is determining which parts of the app should be free and what should be paid. Providing too many free features will result in few people opting for the paid version, since the free version will be enough. Likewise, if most of the features are paid, users may find the app impractical and stop using it.

A clear example of this kind of distribution comes in games. Players are allowed to get through certain levels, but in order to finish the game, they have to download the paid version.

Monetization

Monetization models are different ways of obtaining a profit through apps. They shouldn't be seen as individual and separate, since they usually depend on whether an app is free, paid or freemium, and also on its category.

In-App Purchase

Some apps allow for the purchase of small, separate items that improve basic features, an example being photography apps that sell advanced filters. This is also the case for apps that offer premium content or subscriptions and is generally associated with freemium distribution.

Full Version Payment

In such cases, two versions of an application are developed: a free version with basic features that let users try the app with some limitations or advertising, and a paid version that offers additional possibilities for someone looking for more.

This model is being used less and less because of the inconveniences of having two separate apps. Not only does this often annoy users who don't want to be bothered with downloading a new app, but at the development level, it also poses a challenge

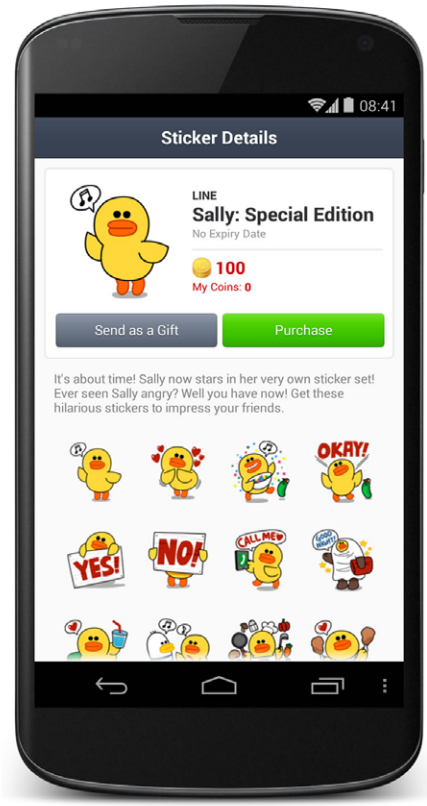


FIG 2.7. Line is an app that uses in-app purchases to sell stickers.

in transporting user-defined settings from one app to another. What's more, user reviews and positions in the store rankings are independent for each version.

These inconveniences can be avoided by offering the premium version as an in-app purchase.

Fig 2.8. Fruit Ninja is an app that has a free version and a paid version.

Advertising

Advertising can be used in free apps as a tool for gaining economic profit. It usually comes by way of a series of small ads that can be tapped by the user to access other websites or download other apps. In this model, earnings depend on the number of people that tap on the ads.

The main inconveniences of this model are user privacy intrusion and ad visualizations that affect user experience overall.

Each operating system has its own advertising system, and each sets different conditions. Google offers an advertising program for apps in Android called Google AdMob[12], whereas iOS has iAd[13] and Windows Phone has Microsoft Advertising[14].

When weighing the option of including advertising, it is essential to consider the platforms where the app will be distributed. Ads are quite commonplace for Android, where users

12 http://www.google.es/ads/admob/
13 https://developer.apple.com/iad/
14 http://advertising.microsoft.com/en-ca/windows-8-ads-in-apps

FIG 2.9.
Each operating system has different advertising programs for its apps.

are more likely to accept advertisements in lieu of paying. But ads can be a problem for iOS, because users simply have a different level of tolerance.

CHOOSING A PLATFORM

Before deciding to design for one or several platforms, you've got a lot to consider, from resources and complexity to the user you're targeting.

Currently, the two operating systems with the most significant market share are Android and iOS, Android coming first[15].

15 KOESTSIER, John. Android up 13%, iOS down 7%, BlackBerry down 81% ... and Windows Phone up a massive 52%. http://venturebeat.com/2013/04/01/android-up-13-ios-down-7-blackberry-down-81-and-windows-phone-up-a-massive-52/

Meanwhile, Windows Phone and BlackBerry are battling it out for third place.

Broad Scope vs. Exclusivity

Designing for an operating system as popular as Android means a potentially greater scope of users. However, the many different screen resolutions and operating system versions available for the platform have rendered designing for it a more complex experience. In other words, building for Android will cost you.

On the other hand, designing for iOS means focusing on a smaller and more exclusive market. Apple doesn't have the same quantity of users as Android mobile devices, but it has the advantage of being more consistent in screen resolutions and operating system versions (and is easier to update). Hence, these factors aren't a barrier for design.

Windows Phone is at a marked disadvantage when compared with the two giants, but the scenario is improving with the help of Nokia and HTC (note that Microsoft recently bought Nokia). Even though the number of Windows Phone users is not significant enough now to take on Android and iOS, there is a lot of potential for growth.

That being said, if you really want to design an app "for all," you have got to go with Android and iOS, at least.

User Personalities

Each operating system has users with specific characteristics—geographic, demographic, psychographic and behavioral—that differentiate them. And though this may seem secondary, gaining a deep understanding of user profiles and personalities provides a number of clues as to who will use the app and what they'll expect from it.

In general, iOS users place more value on the user experience. They are detail-oriented and have a higher socioeconomic profile than consumers from other platforms. Thus, they are more accustomed to paying for apps[16]. iOS users love consistency. They like to see everything in its place and prefer to avoid surprises. This is, in part, because Apple is a more closed and restrictive system when it comes to app approval. It has a specific set of design rules in place that ensure a certain quality and regularity in its apps.

Android is an open-source operating system that allows more freedom for users and companies to make their contributions, and as a result, has a more personal touch. Android users are more open to trying different ideas and apps that break away from the paradigm and present alternatives that set them apart. This also means coming across applications that are a bit chaotic, even after Google's efforts to establish more clarity in its general design guidelines. Moreover, Android has a wider variety of devices at different price ranges, and this makes it more massive.

Finally, Windows Phone is an operating system that is attracting users that love the simplicity of its plain, luxury-free interface. Prioritizing practicality over aesthetics, Windows Phone is focused on users who prefer a nice experience with simple navigation.

Going It Alone vs. Joining a Team

Embarking on a project can be a lengthy journey, depending on the development complexity and scope of an app. Building an app requires the participation of at least two individuals: a designer and a developer.

16 Jiménez Cano, Rosa. Google Play no puede con la App Store: http://tecnologia.elpais.com/tecnologia/2013/04/26/actualidad/1366971919_151325.html

The designer is in charge of defining the general structure of the screens and their interactive elements, interface design, and the preparation of files for the developer. The developer, in turn, takes on the task of transforming the app so that it is no longer just a set of images on a screen, programming its functionality.

These two participants can define general operation properties, the reach of the project and the user experience they want to achieve. In fact, working together allows them to complement each other's knowledge in their respective areas of expertise. For example, a designer can propose a certain interface, but the developer has to consider what the design infers in terms of development complexity. On the other hand, the developer has to propose functionality according to the usability advice provided by the designer. By working in tandem, they can obtain a high-quality app.

Small teams allow for an agile way of working—it's always easier and faster to agree on something with only one person than with two or three. To a certain extent, however, working in a small team may limit the quality and complexity of a project as well.

A two-person team is the absolute minimum, but the ideal, or what we could call the Dream Team, consists of a much larger unit: a project leader to take charge of general coordination, information architecture and usability specialists, visual designers, platform-specific developers, and even writers and QA (quality assurance) staff.

The inconvenience of a bigger team is that it implies greater coordination efforts from all parties involved and a lot of management, which multiplies with each participant. Bigger doesn't always mean better, because having more people in a group implies the need to define in detail the role of each individual in order to ensure that no one detracts from the project.

Products of excellent quality, such as iA Writer[17] from Information Architects, have been developed by companies with relatively few employees. In fact, in many cases, team members don't even share the same office. Projects in digital formats can be worked on by teams in different parts of the world with the help of project management tools like Basecamp and, of course, fluid communication.

Resources

Before getting started, it is vital to have a complete understanding of the path to getting an app published—especially the investment involved. This refers not only to people and knowledge but also to equipment.

If the designer is the one carrying out the project and needs a developer, he or she may not know exactly what to look for in a partner. Different platforms require different programming knowledge.

Android

Android apps are programmed in Java using Android's own libraries, so at the programming level, a developer with sound knowledge of standard Java shouldn't have many problems joining the Android world.

Programming apps for this operating system can be done with a Mac or a PC, Windows or Linux. The Android Studio software and all of the material needed to develop an app can be downloaded from Android's developer website[18].

17 http://www.iawriter.com/
18 http://developer.android.com/

When developing, Android Studio lets the programmer use several device simulators. They can also perform a real use trial by connecting a mobile device to a computer.

iOS

Any programmer willing to work with iPhone and iPad should have some knowledge of object-oriented programming before transitioning to Objective-C, the programming language used for iOS[19].

In terms of hardware and software requirements, a Mac computer is needed along with the Xcode, the official SDK for developing for iPhone and iPad. Xcode may be downloaded for free.

The code can be tested directly with a simulator (a representation of the phone that shows how the code behaves) on a computer. This is sufficient most of the time but does have certain limitations as an app may appear to behave much faster than it will on a phone.

Ideally, to perform a more real development trial, the code should be tested with an iPhone connected to a Mac. This requires the purchase of a developer license[20], something that will be necessary for publishing. The cost of the license is 99 dollars per year.

Windows Phone

The programming language used with Windows Phone is C#, so finding a developer with experience in that arena will be

19 Kumar, Mugunth & Napier, Rob. http://iosptl.com
20 https://developer.apple.com/programs/ios/

a good idea. To build an app, C# is combined with Windows Phone's own libraries.

Developers that don't love Windows will have to get used to it, because they'll have to have at least Windows 7. But don't despair—it can be installed on any PC and also as a virtual machine in Mac. And speaking of software, Microsoft Visual Studio is also required for building for Windows Phone. The free version is enough to develop an app, but it's clear that paying for a full version is much more convenient.

The Windows Phone simulator works relatively well, because the app can be tried directly on a computer. In order to perform a more real simulation, it is possible to connect a phone to a computer with Windows. But for that, you also have to buy a developer license that costs 99 dollars per year[21].

21 http://dev.windowsphone.com/en-us

3

Irene Pereyra: UX with Magic in Fi

We must confess that we are big fans of Fantasy Interactive. A place where more than a few designers would be happy to work[1], the agency gives free rein to creativity and boasts a world-class team.

We first heard about Fantasy Interactive when its portfolio landed at our feet, largely including interactive experiences built for clients like Microsoft[2], Google[3] and USA Today[4]. Many of its success cases are outlined on the Fi website and feature so much detail and definition that they seem to be the projects themselves. The agency's level of dedication to design can be seen in every single thing it does, even in the icons of its apps and websites[5].

We're all on a quest for perfection—perfection that doesn't exist and is impossible to attain. This, in turn, may leave us unsatisfied with our work. How do we know when to stop? When is a project finished? Irene Pereyra, Global Director for UX and Strategy for Fi, gave us her answer:

> *There is a moment in every single project when you finally get this "aha!" clarity. Everything leading up to that just somehow doesn't feel right and your brain keeps churning through various iterations and ways to improve, enhance and streamline. Doing UX is a bit like untangling a really big ball of yarn. You don't know how long it will take, but once you get to the end, the satisfaction is immense. Being able to stop iterating because it's "done" is my favorite moment in any project.*

The clarity Irene refers to is something achieved after one follows the stages of a design process and doesn't necessarily begin

1 http://blog.f-i.com/six-reasons-you-want-to-work-at-fi/
2 http://blog.f-i.com/explore-touch-with-internet-explorer/
3 http://www.f-i.com/google/ramayana/
4 http://blog.f-i.com/usatoday-com-redesigning-one-of-americas-most-popular-news-site/
5 http://www.f-i.com/fi/icons/

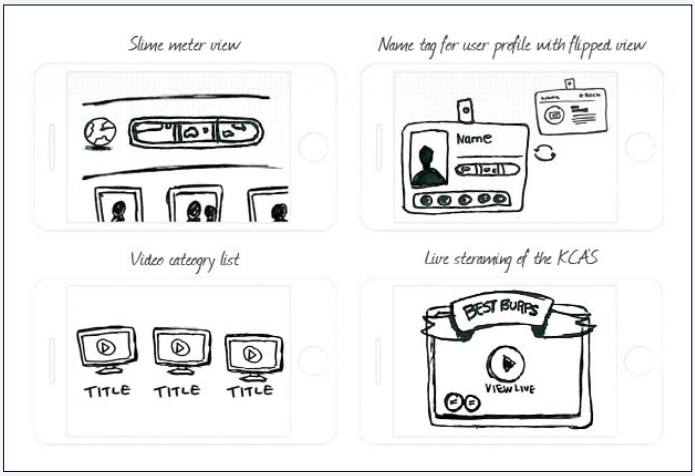


FIG 3.1. Wireframes, even when on paper, are fundamental before starting visual design.

with visual design. The idea of a solution should be translated to reality, for example, using wireframes, before interface detail design commences. Not many designers are used to doing this, so how important is it?

> *Sketching and concepting are hugely important. Whether you do that on paper, or in wireframe format on the computer doesn't really matter (actually, ideally you should do both!). Not jumping straight into design allows you to focus on things like functionality, the interaction of the overall framework, and the journey you want people to go on when they travel through your interactive experience. Jumping straight into design is extremely dangerous because that can lead to just focusing on the design details, which of course is also hugely important, but really should be considered at a later stage in the "design" process once the interaction framework has been determined.*
>
> *Iteration is best during this "black and white / paper napkin" stage because you are not yet married to any visual*

treatments, and it's much easier to quickly move through different concepts and ideas for micro-interactions.

Irene sees visual design as the result or consequence of the initial planning of user experience:

Ultimately, good UX is all about what is appropriate for what experience and what form-factor, and the design layer that you apply to that should enhance the overall interactive experience, regardless of whether that is skeuomorphic or flat.

These initial stages are vital for the project in ensuring continuity with the rest of the process and guaranteeing the same passion for quality. This isn't exactly easy to achieve in projects where large teams composed of professionals with a wide range of profiles and backgrounds are involved, not to mention the client.

How do you work as a team and keep the client involved in the process?

We have a very multi-disciplinary approach to projects, which really just means that everyone across all the different disciplines is involved from the very beginning at different levels of intensity. Since we tend to move quite fast through the different phases, it's actually incredibly helpful to have the opinions of developers and designers right at the start of a project, as that tends to alleviate potential problems that may occur along the way.

As far as involving the client, besides regular check-ins, meetings and scrums, we also post our daily progress on Basecamp in the form of PDFs and video walkthroughs of those PDFs. We love doing video walkthroughs of the

work so that the client can have a video presentation of the work at their own leisure. Clients are busy people, and they don't always have time to read through all the annotations, so hearing us present the work whenever is convenient for them has proven to be extremely effective in getting directional feedback.

Clients don't just choose Fi. Fi chooses its clients, leveraging its reputation to work only with the best of the best.

You have worked for some of the most important brands in the world. Have you ever said no to any of them? What motivates you to start a new project?

Yes, we have said 'no' to some of them, even big brands that most companies would dream to work on! The key to a successful project really depends on 3 main things, and from a creative perspective, these are all the things that we consider before taking on a new project:

1. *Is the client aligned with your thinking, and are they ready and able to be innovative?*

2. *Is it something we haven't really done before? If yes, that's great, because it allows you to start a project without any preconceived notions or baggage.*

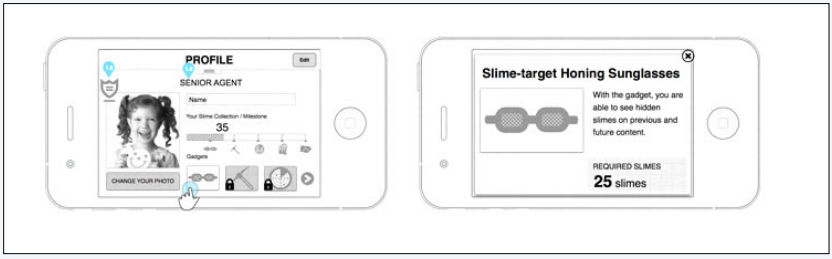
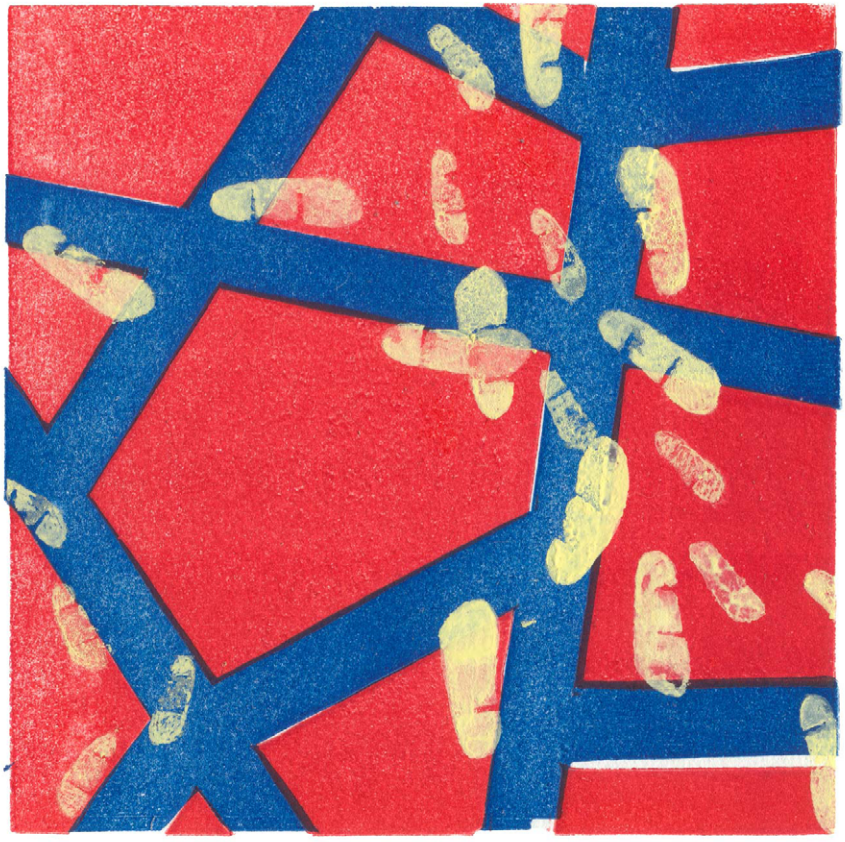


FIG 3.2. Irene participates actively in the early stages of her projects.

3. *Lastly, does the timeline and budget allow for experimentation and failure? Not once in my entire career did I hit the bullseye right away, and it is very important to know that you have the time to go off into the wrong direction for a little while before finding your way again.*

   What really motivates me to start a new project is the "great unknown." I love diving into things that make me nervous and unsure, and I love that deadlines sometimes force creativity out of you in ways that you never imagined.

Visit www.f-i.com to learn more about Fi and be dazzled.



4

Exploring Ideas

The high number of apps available in the market means there is a lot of competition, but it's also a challenge and an opportunity to stand out. Offer something that differentiates your app and adds value for the user. Testing your idea is the first step.

A Myriad of Apps

In daily life, we often come across situations that require an immediate solution. And when there's no solution in sight, we sometimes say to ourselves, "There should be an app for that." This way of thinking has saturated app stores with worthless products that aren't really useful. Nowadays, there's an app for pretty much everything.

This scenario is perhaps most evidenced in the Google store, which has fewer restrictions when it comes to approving similar apps. There is a plethora of calculator and flashlight apps with absolutely nothing new to offer.

Clearly, this isn't an encouraging prospect for the designer looking to launch a product. Moreover, it is intimidating to incorporate an app into this immense offer, especially if you want to make it stand out.

Fig 4.1. There are many flashlight apps with different proposals.

When an App Isn't the Answer

An app isn't always the answer for each and every situation. Sometimes, bringing in technology adds unnecessary agents to a process.

For example, an app that opens your car door is useless. Finding your phone in your pocket, searching for the app, and utilizing the adequate feature for the task takes a lot longer than just getting out your keys. Apps should make tasks easier and faster, not longer and harder. The use of an app as a response to a situation is valid when it simplifies a process and improves user experience.

Differentiated Value

Differentiation is key, and if you want your app to stand out, you've got to offer something significant that really adds value. The road to getting there is based on three fundamental considerations that can truly make a difference.

1. Have a Goal

The goal is the engine of an app, the driving force. Practically anything included in an app must, to some extent, have the goal in mind. This is directly related to the user's needs and the way in which the app addresses them.

Needs may be related to entertainment, information, social interaction or a specific problem, such as finding the best way home when you're lost. The more an app addresses needs, the more valuable it will be.

2. Think About the User

User-centered design puts people at the center of a process. It takes into account emotions, motivations and needs at the moment of proposing solutions.

Knowing who a user is renders decision-making easier. Designers can then make choices that create intuitive apps that are much simpler to use.

3. Determine Context

Where will the app be used? Context places users in a determined physical space that affects and conditions the way they interact with the screen. Context also takes into account factors

Fig 4.2. The context of use takes into consideration where users are when they interact with an app.

like general environment, the presence of others and related actions.

For example, it is one thing to design an app for someone at the gym and another for a person waiting for a bus or at home messaging with friends.

An Idea is Born

Ideas change the world. For a designer, an idea is a proposal for users based on his or her interpretation of how their needs can be solved. Ideas are not necessarily original or unique, but they should be better than the rest.

Starting out, you may not have a specific idea in mind. What should you do? Research, research, research. It is as simple as looking around, paying attention to your surroundings, to the people around you and the difficulties they encounter in their daily lives—or, even better, the problems you yourself face.

And If It Already Exists?

Once you come up with an idea, the next logical step is to check whether the idea already exists, if a related solution has already been developed by others and how. This is generally known as benchmarking.

Upon investigation online and in app stores, it's normal to find an app that, at least at first glance, is similar. But don't be disheartened! This doesn't mean you should abandon the idea and start over.

Analyze and assess existing alternatives to determine how they can be improved, complemented or supplemented to add more value. You can make a difference by specializing a certain feature, simplifying or improving other aspects that will ultimately lead to better user experience and usability.

FIG 4.3. Mailbox is an email product that generated a lot of expectation for its way of managing email as to-do tasks.

This much-needed investigation to improve an initial idea can be what marks the difference. Consider factors your competitor hasn't taken to account and implement that one thing they've missed.

This is what happened with Mailbox App, an email client[1]. When it was launched, there were other alternatives in the market, including Gmail. However, the team took advantage of the opportunity to offer a product that challenged the concept of email laid out by their competitors. With that, they gained recognition.

1 http://www.mailboxapp.com/

But Will It Work?

After landing on an idea, you've got to determine whether it has the potential to work.

Ask users for their opinions and get feedback for the concept you've proposed. The process doesn't have to be expensive or time consuming. Observe and pay attention to people's behaviors and difficulties, and ask yourself whether your idea can help. Talk about your idea with friends and acquaintances and ask what their impressions are.

We're not saying your research will clear up all doubts and provide definitive answers, but it is a good way to get a preliminary validation of your concept. Base your work on real information, not untested assumptions.

5

Defining the Proposal

It’s now time to design (at least wireframes). To get started, we first need to understand and study users a bit more. What are their needs, and how can design address them?

User Research

Getting to know users makes it possible to design an app that takes into account motivations, needs and problems as the starting point for building a proposal. This knowledge is not based on assumptions and theories; on the contrary, it's based on research that helps determine the user profile for the app. *Personas* and *User Journey* are some of the methodologies used to achieve this.

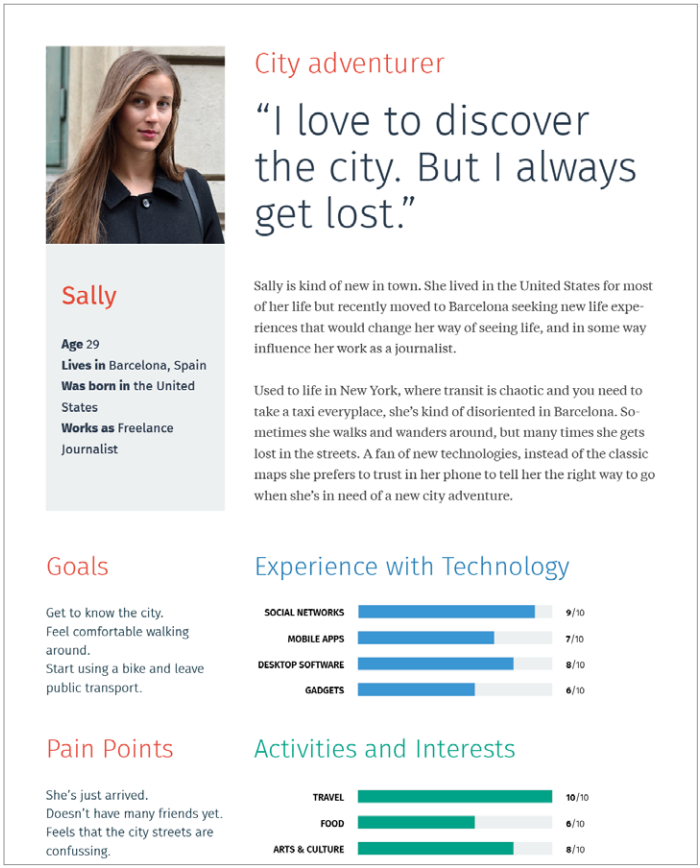


Fig 5.1. Personas allows extracting patterns that are common to many users.

Personas

The concept of Personas was coined by Cooper, a San Francisco-based strategy and design company[1]. It is a very useful tool that is utilized constantly in interaction design today. Its function is to define user models and archetypes for design.

In order to create Personas, real research is needed—you can't just rely on conjectures. Many possible users should be analyzed in order to determine the behavioral and thought patterns they share, avoiding their individual characteristics and focusing only on what they have in common.

The end result of this investigation is a visual representation in which the user is modeled after the data collected: the Persona will have a face, name, story, ambitions and objectives.

Many different types of Personas can be determined for an app, but for this exercise to be useful, there shouldn't be more than three. Ideally, the project should focus in one main Persona.

User Journey

Considered individually, Personas can help us to understand user models. But we also need to know how those models behave and feel when they have a certain goal in a specific context[2].

It is here that the User Journey intervenes, a way of visually comprehending, from beginning to end, the process that a Persona embarks on from the moment a need arises to the second that need is satisfied, using the app.

To make things clearer, here's an example. Someone who's lost can use a mobile phone to find the way home. To create a

1 http://www.cooper.com/journal/personas

2 Hobbs, Jason. An introduction to user journeys. http://boxesandarrows.com/an-introduction-to-user-journeys/

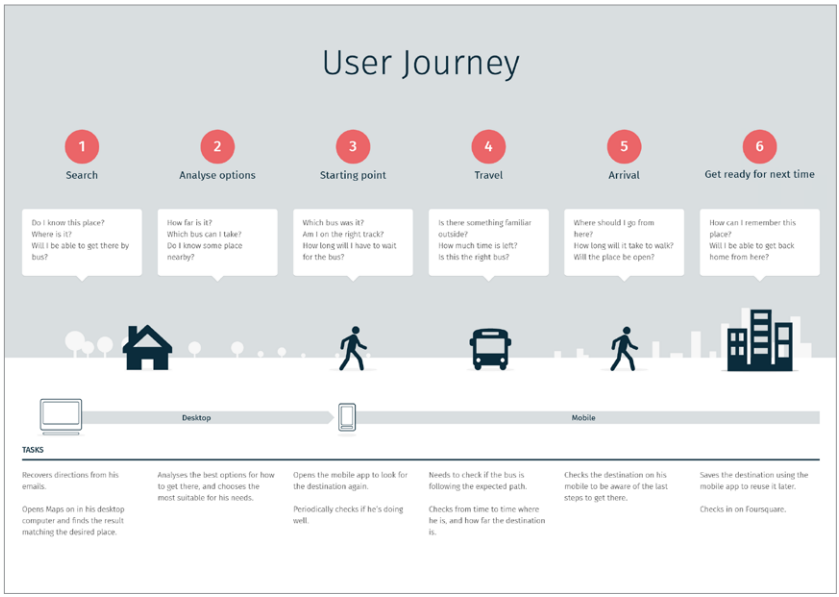


FIG 5.2. The User Journey is represented graphically to differentiate stages and the relationships between them.

User Journey based on this scenario, you must consider the entire situation, from the moment he or she gets lost, before opening the app, to the moment he or she achieves the goal of finding the way, going through each of the actions performed with the app.

This process can be visualized graphically in a linear representation. Stages may be delineated to detect user emotions as well as the difficulties encountered with each step and the actions required throughout (an example being search). Thus, it is possible to detect the moments when design should focus on solving interaction problems in order to obtain a more usable application.

User Journey is also useful for laying out the preliminary foundation of information organization and defining features without taking into account a rigid or hierarchical structure.

Functional Definition

All actions and interactions needed in order for the user to achieve his or her goal are translated into features that an app should have. By following the User Journey, it is possible to detect needs at each stage and the tools required to move on to what's next.

Using the same example of the directions-providing app, the fundamental features would be: determining current location, searching for destination address, and choosing transportation

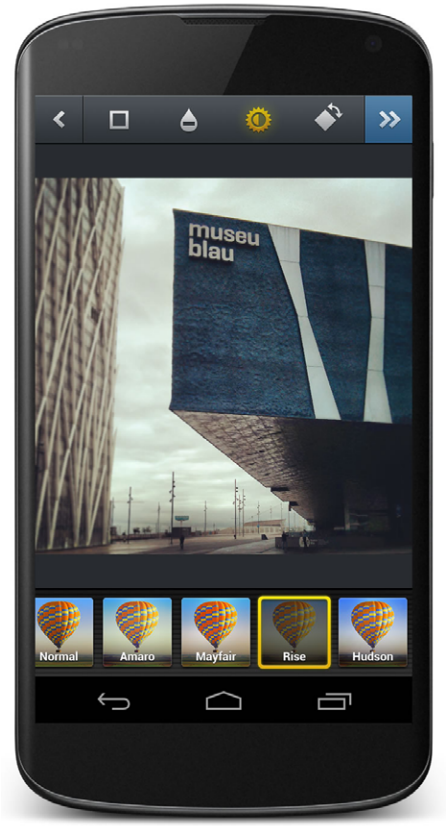


Fig 5.3. Currently, Instagram is one of the most popular photo and video apps.

options. Each action is equally important and serves the app's overall purpose.

An infinite number of complimentary features can be added, for example, saving a destination. Each time a new feature is added to complement the main goal, it is essential that the feature be truly useful, taking into account Persona and context of use. We should never forget what is expected of a mobile phone in each particular case, in this example, the capacity to solve a specific problem quickly.

Each feature that is added also represents more time for development and more complexity, which is why it's important to decide carefully, in each case, if it deserves to be included. You don't want to end up with a product that is so saturated with features that the experience is ruined. It's always preferable to do less well.

For example, popular photography app Instagram[3] became what it is today after paring down a series of unnecessary features. Now, its complementary features, such as filters, represent real value for users, allowing Instagram to differentiate its product from the rest.

Proposing a Vision

Users often ask for features they think should be included in a product that has already been built. This can be a double-edged sword, because what is useful for one user may not be useful for everyone[4]. In such cases, it's fundamental to strike a balance between user opinion and the proposition.

3 Systrom, Kevin. Instagram: What is the genesis of Instagram? http://www.quora.com/Instagram/What-is-the-genesis-of-Instagram

4 http://gettingreal.37signals.com/ch04_Make_Opinionated_Software.php

Information Architecture

Information architecture is a way of organizing the content and features of the entire app in such a way that they can be found quickly and easily by the user. In a global sense, information architecture considers the relation of content on different screens and, on a more particular level, the organization of content within the screen itself.

After having defined User Journey and app features, an architecture diagram determines the screens and features required at each stage. Business aspects and technological requirements may also be taken into account.

One way of visualizing architecture consists of representing each screen with a rectangle—in apps, generally more vertical than horizontal—where the connections between rectangles indicate a way of navigating from one screen to the next and the means of each action.

This diagram helps with the study of an app's complexity at the glance, allowing you to analyze different levels of depth and to visualize and understand the relationship among con-

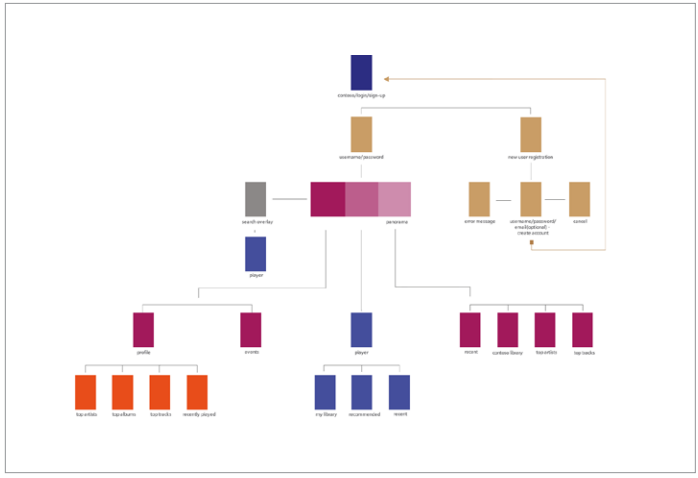


Fig 5.4.
An information architecture diagram makes it possible to view the links between content quickly.

tent in a much more organized way. Whatever is defined at this stage will have a direct impact on the kind of navigation chosen afterwards.

Wireframes

A wireframe is a very simplified representation of an individual screen that allows us to have an initial idea of the organization of elements, identifying and separating those which are informative with those that are interactive.

A wireframe is to an app what a blueprint is to a house. This blueprint shows spaces and functional elements in a clear and simplified way.

Like blueprints, wireframes are drawn lineally and in the same color, avoiding aesthetic temptation and focusing in the structure or skeleton of a screen. Elements like texture, shades and volumes are left aside in this stage.

What Do Wireframes Do?

Wireframes are not always made. In fact, it's normal for a designer to skip this step in order to go directly to the visual

Fig 5.5. Wireframes are fundamental for understanding how each app screen can be structured.

design of the interface. However, it is a practice that can have negative consequences in the long run. Wireframes may be used in a number of different ways:

1. A personal exploration tool. A wireframe allows the designer to evaluate different navigation and interaction alternatives quickly, without having to invest too much time in a finished design that may not work when tested.
2. A tool for communicating abstract ideas. In the early stages of a project, it's necessary to transmit the general idea of the app to other people, focusing on functionality, objective and rational, and avoiding the distractions contributed by the subjectivity of aesthetic elements.
3. A mechanism for performing initial interface evaluations. Before having designed or developed a functional app, you can detect interaction and usability problems by asking for feedback from users and people who are not familiar with the project.

Making wireframes in the early stages means saving money and time. It makes possible the evaluation of navigation and interaction and creates a solid foundation for visual design.

Wireframing Options

Paper
This is the most basic and accessible approach. It consists simply of taking out a piece of paper and sketching out screens and interaction components. Generally, all wireframes begin on paper and then develop through other tools prior to software design.

FIG 5.6. Wireframes can be done on paper quickly.

Stencils

This would be a next step. A bit more professional than a free-hand wireframe, it still retains the essence of work on paper. With the help of stencils, generally made of metal, it is possible to draw interaction elements directly on paper, with a much more formal, clean and uniform structure.

FIG 5.7. Metal templates can be used to maintain the freshness of paper while at the same time achieving more precise results than with freehand drawing.

Templates

This way of creating a prototype consists of using digital files of templates that have basic interaction and interface elements (buttons, lists and headers) to build in a computer, using design programs, such as Photoshop, and basic screens based on those components. Depending on the operating system, different

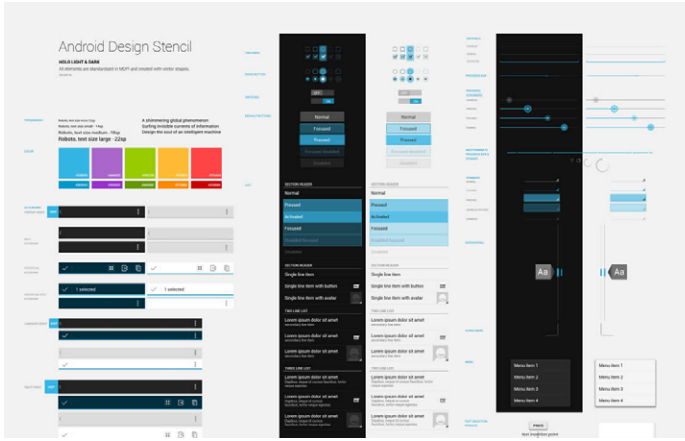


FIG 5.8. There are templates, in this case for Android, that can be used to design wireframes on a computer.

resources can be used; the ones that are more widespread are those from iOS, followed closely by Android. Currently, there are few templates available for Windows Phone.

Additionally, there is also desktop software on the market with template libraries for making wireframes. Some of the best known include Balsamiq[5], Omnigraffle[6] and Axure[7], but there are many other options. If the intention is to design wireframes using a browser and saving proposals in the cloud, UXPin[8] allows users to work collaboratively with other team members.

Choosing What's Best

In reality, there is no better or worse way of creating wireframes. The alternatives we offer are the best known. Generally, the choice is made according to the complexity of the task at hand. It also depends on personal elements, such as comfort with the tool.

5 http://www.balsamiq.com/
6 http://www.omnigroup.com/products/omnigraffle/
7 http://www.axure.com/
8 http://uxpin.com/

Certain designers can do a great job drawing on paper before working with a computer, and on the contrary, some people feel more at ease working directly with the computer, using templates and design programs.

It is advisable to try different options in order to find the one that offers comfort and agility and that adequately represents the ideas for the different screens of the app.

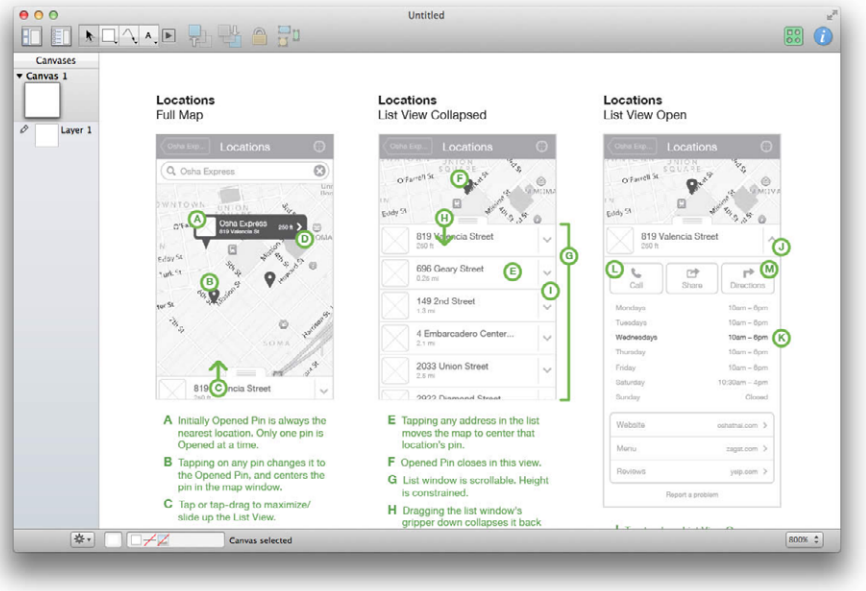


FIG 5.9.
Omnigraffle is desktop software that allows using imported templates as libraries.

Prototypes

Prototypes are representations for internal testing or for user testing. They help detect usability errors in the early stages of design. Generally, they are models with enough interaction to navigate different screens[9].

Prototypes can be based on wireframes or visual designs. Their levels of fidelity vary according to how similar they are to the expected final version of the app in terms of appearance and behavior.

Which Screens Should Be Prototyped?

There is no need to make a prototype with all possible screens of an app. Prototypes are destined to be trials; therefore, they should be created only for the screens required for completing the task being tested. For example, to test a user login, the prototype should include all screens needed, from data input to the last message of success.

Prototype Formats

There are different ways of creating an app prototype, from drawing on paper to using traditional design software and even programs created exclusively to develop prototypes. The choice depends on immediacy and the resemblance to the expected final result.

According to the output format that can be obtained from a prototype (to then share with other team members, clients

9 Armengol, Daniel. ¿Qué es un prototipo? http://interactionphilia.com/que-es-un-prototipo

and stockholders), the ways of making prototypes can be classified as:

Interactive Documents

Prototypes can be presented as the same documents that have always been used for commercial presentations and information distribution, adding the necessary interaction that allows browsing from one part of the document to another.

For example, a PDF based on design files or photographs of a sketch of the app can represent its basic functions. Balsamiq is a tool that does this well.

The same can happen with presentation files like those used in Microsoft PowerPoint, Apple Keynote or OpenOffice documents. Using templates, Keynotopia[10] creates documents that are compatible with these software products, and an interaction layer is added to simulate the behavior of the app.

In order to be opened, these kinds of documents have to be downloaded first. Transitions and animations that link pages can be somewhat basic.

Web Versions

Most prototype software, for both desktop and the cloud, creates representations of an app as websites based in HTML5 and CSS3 to show interactions and animations. Some of these programs even create a launch icon in the phone's home screen in order to develop a simulation experience that is as similar to reality as possible.

Some of the tools used to create prototypes in this way are Codiqa[11], FluidUI[12], Framer[13] and Flinto[14].

10 http://keynotopia.com/
11 http://www.codiqa.com/
12 https://www.fluidui.com/
13 http://www.framerjs.com/
14 https://www.flinto.com/

Other Formats

Alternatively, there is specific software available for developing prototypes. Briefs[15], for example, is a piece of software that can achieve results very similar to those of a native app. Using a desktop program, Briefs creates a simulation of the app that needs a special viewer (a free app) in the target phone in order to be opened.

If a very fast prototype is needed, built with the phone itself, a good tool is POP[16], an app that uses a mobile phone's camera to photograph wireframes on paper and provide interaction to navigate different screens. This is an option that can bail you out of a tough situation and help to perform quick checks.

15 http://giveabrief.com/
16 http://popapp.in/

6

Dustin Barker: Electronic Banking Made Simple

When it comes to apps that are very complicated to design and develop, one of the first things that come to mind is electronic banking. Apps of this nature have many possible features—payments, transfers, etc.—and security is an issue that cannot be overlooked.

Simple is a money management system based entirely on a digital platform. Not a physical bank, it solves what could be highly complex through an experience that is, in fact, simple. And because it's a service that is used entirely on the internet, its application is of the utmost importance.

Dustin Barker, Mobile Engineering Director at Simple, is in charge of developing an app that, besides having a spectacular user experience, fosters so much confidence that the platform's security is never in doubt.

How do you manage to make users feel safe and confident when using the Simple app?

Since the mechanics of how we secure our customers' data are not visible, the feeling of security has to come from the interaction. We go above and beyond in securing our mobile apps, but often it's the details unrelated to security that provide users with a sense of safety.

We look to establish trust through quality. A responsive, stable app is essential to a feeling of security since it reassures the user that the engineering team is mindful with details. A high quality exterior gives the user a basis to assume a high quality (and therefore) secure interior.

In addition to establishing trust through quality, we spend a lot of time designing the security mechanisms that the user can see. Features like multifactor authentication, which we use for sending payments, are essential to not only

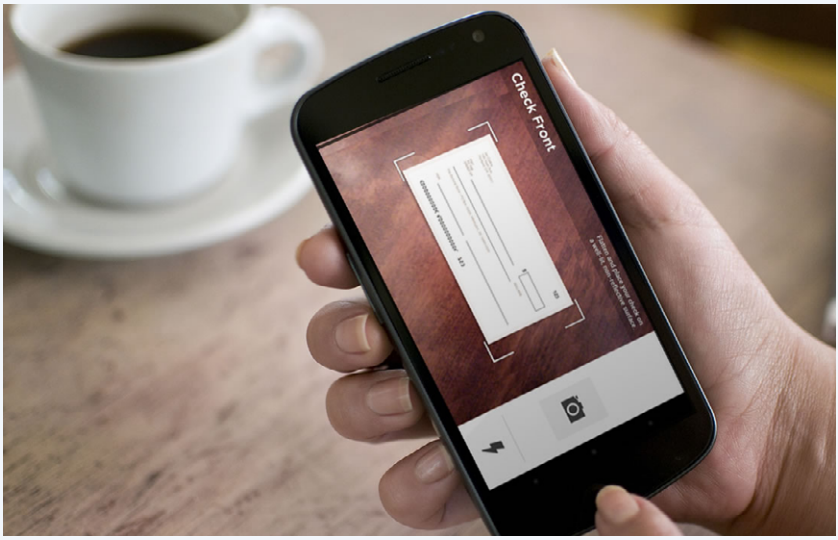


FIG 6.1.
At Simple, they design apps for Android and iPhone in parallel.

securing the interaction, but establishing trust and a sense of safety for our customers while they are using Simple.

Dustin was also the ideologist of the mobile first way of working, which privileges mobile design before a desktop website is created. Even though this methodology is booming today, it has still failed to demonstrate clear advantages in comparison to the traditional process.

When you look back, what do you think are the most important benefits of this way of working?

Our focus on Mobile gives us constraints to work within and helps us make decisions. Parameters like screen-size, battery life, and network connectivity force us to pare down features to only the essentials.

Mobile apps are used in distracting environments —like waiting in a line or during a conversation— so we use this

knowledge to make decisions about how we can make every interaction as efficient and informative as possible.

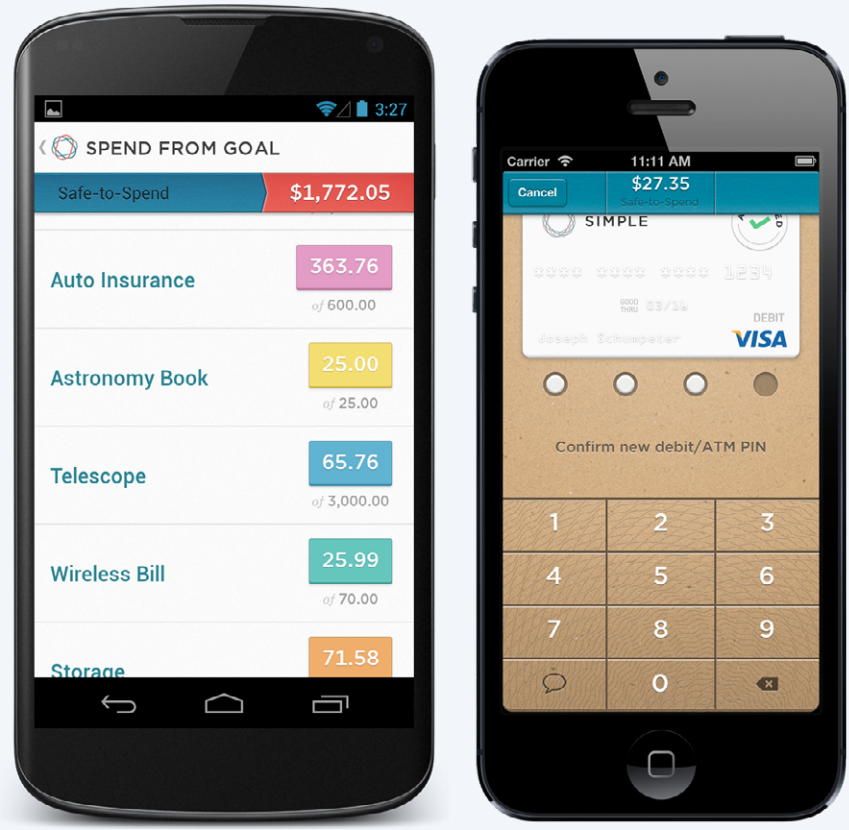
When we think about a feature, we don't limit our thinking to just what transpires between the screen and the user, but where the user might be located, what they might doing while they interact with the app, what their needs are in different contexts. These considerations help us discover details about a feature that might have otherwise been overlooked.

What's next after the iPhone app? An Android version, of course. Dustin told us what he learned from the experience of designing first for iOS and then for Android and targeting a different kind of user:

One of our primary goals in building Simple for iPhone was to make all interactions as intuitive as possible. In order to achieve that, we relied heavily on iOS idioms that we knew our users would be familiar with. In order to achieve that same goal on Android, we couldn't use those same iOS idioms. We had to start from scratch. We re-imagined the Simple app using only Android architecture and the result is an app that is consistent with our brand and offers a unique Android experience.

In the first releases of Simple for iPhone, we learned how to streamline and simplify each feature. We also learned which aspects of the app were particularly tricky or prone to error. While the design for the Android app was completely new, we were able to begin building exactly where Simple for iPhone left off.

Since the launch of Simple for Android, engineering decisions we've made in the Android have also been



carried back to our iOS app. For instance, we knew that the category list in earlier versions of our iPhone app was less than optimal. We debuted a new category list design in the first release of our Android app and once we saw that it was successful, we quickly ported it back to the iOS app. Our users have been thrilled with the result.

Now, we're in a position to let our experiences on both platforms cross-pollinate and we're able to try different things on each to explore what works best.

FIG 6.2.
In these kinds of apps, gaining a user's trust is fundamental.

Besides mobile apps, there is also the web. In the realm of money management services, some operations can be difficult to complete from a mobile screen and are best left for desktop computers.

How do you choose which features it makes sense to have in a mobile device?

> *Simple's features can be divided into functions for controlling (send a payment, transfer money, deposit a check, etc.) and analyzing (reports, statements) finances. We've focused our mobile app on the functions for controlling finances. These are the functions that our customers need on the go. You can read a bill, grab your phone, open Simple, and send a payment in seconds with just a few taps. Functions for analyzing finances might find their way into our mobile app eventually, but when they do we'll still look for ways to optimize these interactions.*

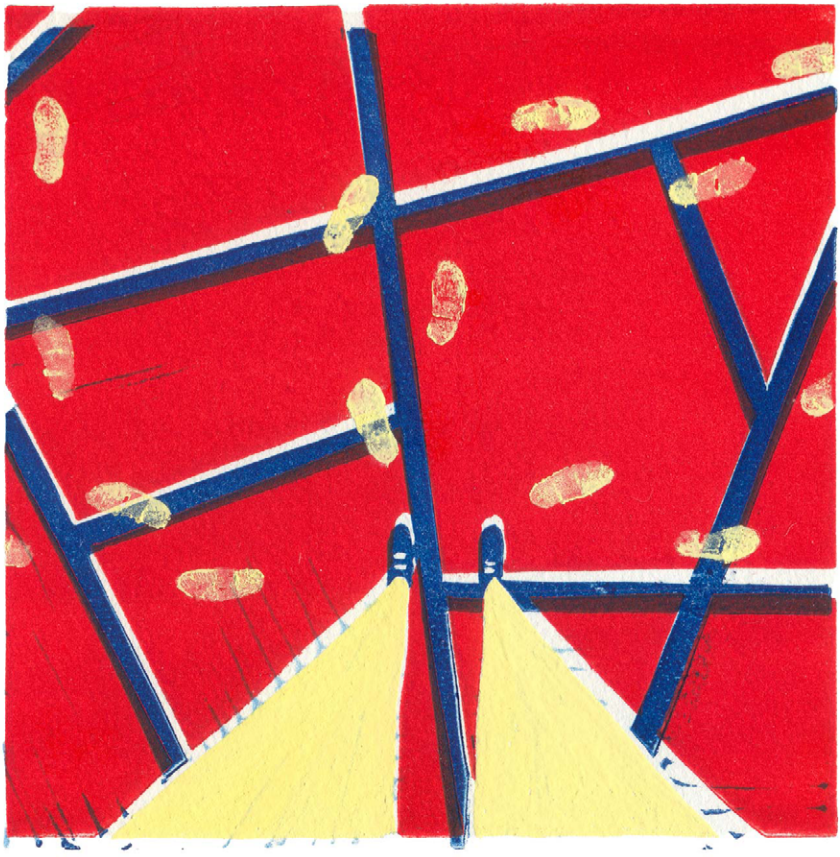
Simple apps serve as great inspiration for designers and developers who dream of a product with a clean visual appearance, without disregarding functional or business aspects. Contrary to how it may seem, achieving this is not impossible. Dustin revealed his work process:

> *The key for Simple has been to continuously evolve our design as each feature is built. We never stop designing the interface and we never hesitate to revisit these decisions, even very late in the process of building a feature.*
>
> *We begin each feature with a design phase, during which engineers provide both technical and aesthetic feedback as the designers iterate. Once we have a solid vision, we begin building, but we know the design phase is not yet complete. As we build, we discover how*

a design 'feels' on a touch screen and this leads to new ideas about how the feature should look and behave. In order to get constant feedback on our progress, we ship a new build to all employees every night.

We're always looking for ways to improve each feature and we never regard any one feature as 'complete'. This process is only possible because our engineers and designers work very closely and collaborate constantly.

Simple is only available in the United States. Visit the company website and learn more at www.simple.com



7

Interaction and Patterns

All operating systems propose different ways of interacting with the elements on a screen. Getting to know the differences between them and understanding how to best utilize the elements most familiar to users ensure that they will feel comfortable and confident using the app.

The Principles of User Experience

Each operating system has its own identity, which is reflected in the appearance and behavior of each of the elements within the interface. Such elements reveal the unique personality traits of each operating system, the very essence that sets the experiences they offer apart.

Not all interfaces share the same fundamentals, which are manifested in interface design. The following concepts are considered key elements of operating systems and the apps they contain.

Simplicity

Visual simplicity is directly related to usability. It implies a certain amount of minimalism and demands that the elements present in the interface have well-defined functions that contribute to an app's objectives and are of true utility for the user.

Mobile phones are not ideal devices for showing a lot of information on a screen. Thus, simplicity also means managing visual economy and using sound criteria to determine what to include in design. Too many elements can overwhelm the user, so whatever appears on screen should be necessary within a specific context.

Creating a simple design is, paradoxically, pretty complicated, but it yields enormous benefits for the app's user experience.

Consistency

An app is composed of many different screens. At the same time, it functions within an operating system with certain visual and interaction characteristics. The Android, iOS or Windows

Phone user is already familiar with the OS and expects apps to behave in a determined manner.

Consistency is about respecting the user's knowledge and habits not only inside the app, but also in relation to the rest of the OS. This favors the intuitive use of the app and allows the user to foresee behavior without much effort.

The existing relationship between appearance and behavior must also be consistent. The visual aspect of a certain interactive element, such as a button with an icon, can lead the user to expect a specific behavior based on how it looks.

For example, if a button is used for the delete action in the operating system, the user will expect to have the same effect inside the app. Fulfilling that expectation means having consistency.

Intuitive Navigation

Another aspect that should be carefully considered in app design is how content is navigated. Users should have a clear, easy understanding of how to get around in the app.

Intuitive navigation is also related to consistency. Each operating system has its own set of elements, such as buttons, tabs and panels. Using them correctly will enable the user to recognize them with ease, thus getting from one section to another.

Likewise, it is essential for the user to know and foresee what will happen or appear after tapping a button. Intuitively knowing where he or she is within the content of the app and knowing how to go back are important factors that benefit users significantly. Intuitive navigation enables a fluid and effortless use of the app.

Interaction and Ways of Holding a Mobile Device

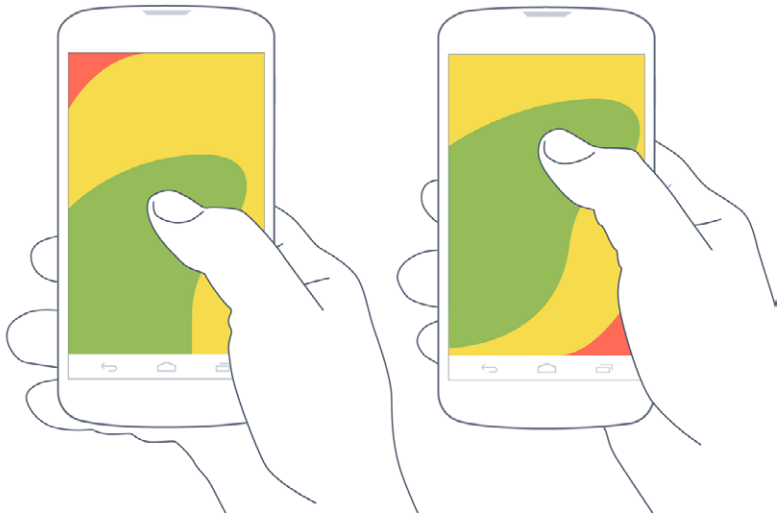


Fig 7.1. According to the way a smartphone is held, the thumb has more or less difficulty in accessing certain areas.

Mobile device app design should take into consideration the way in which users hold their phones. Similarly, which fingers interact and how they are used have an incidence in interface design and condition the location of interactive elements on the screen[1].

Even though the user can hold the phone in several different ways, one of the most common is holding it with only one hand, something that can be liberating (since it leaves the other hand free) but also conditioning, because it attaches a lot of responsibility to the thumb for interaction.

Hence, the anatomical characteristics of the hand determine which areas of the screen can be reached with the most comfort.

1 Saine, Jamie. How Mobile Users Hold Devices. http://blog.utest.com/how-mobile-users-hold-devices/2013/03/

The *Rule of Thumb*[2] refers to the screen surface the thumb is capable of reaching without too much trouble and gives us a few clues as to how to hierarchically organize the elements of an interface.

For example, the most frequently-used buttons should be positioned on the lower part of the screen, where they are easiest to reach. Meanwhile, controls that are generally avoided, such edit and delete, are located elsewhere, with more restricted access.

As Josh Clark[3] explains, the location of buttons and other controls on the lower part of the screen shouldn't be only a matter of convenience, because locating them in that area also avoids covering them with the hand when moving over the mobile phone. This concept was already contemplated in everyday items like iPods or calculators, where the movement of the hand generally did not interfere with the visualization of data.

However, each operating system uses screen space differently, and that use also conditions applications. For example, iOS locates tabs within the comfort zone of the thumb, allowing simple shifts between content. Meanwhile, Android uses tabs in the upper area that are harder to reach but avoid conflict with other buttons, such as the back arrow, located on the lower part of the screen. Windows Phone is a similar case.

With larger phones in which the screen exceeds four inches, or for performing more precise tasks like writing, it is common for users to hold the device with both hands. In such cases, one hand is dedicated entirely to holding the phone and the other, generally using the index finger, taps and gestures with more liberty and precision.

2 Hinman, Rachel. Mapping the Screen for Touch. The Mobile Frontier. Rosenfeld Media.
3 Clark, Josh. Designing for touch. The Mobile Book. Smashing Magazine.

Incidence in Device Orientation

Taking into account a device's orientation when an app is being used means making the most of each scenario. Generally, smartphones are held vertically, while in the case of tablets, it is common to go from portrait to landscape more frequently.

In smartphones, landscape orientation is used largely in situations that require making the most of the entire screen. Holding the phone horizontally provides space for a bigger keyboard and more surface to tap keys for more comfortable writing.

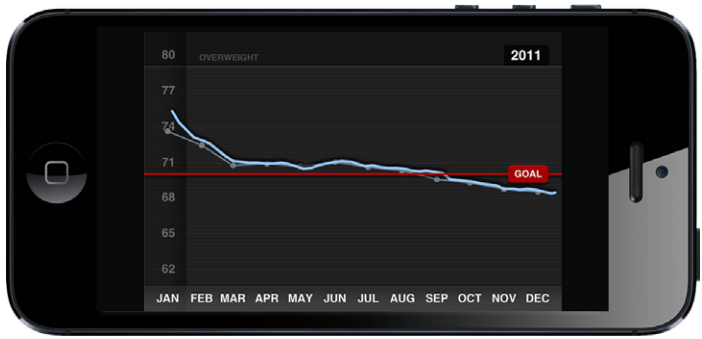


Fig 7.2. Different orientations are an opportunity to rethink the arrangement of the information that is most useful for each case.

It's advisable to design for both orientations so that users aren't forced into only one version. However, it is also necessary to evaluate whether it is really worthwhile for the app, taking into consideration that designing a landscape version doesn't mean directly transferring each element to the most similar position in the portrait orientation. It means taking full advantage of the space available in landscape mode, relocating and rearranging graphic and interactive elements to improve usability.

Patterns of Interaction

According to Martijn van Welie, «an interaction pattern is a short hand summary of a design solution that has proven to work more than once. Please be inspired: use them as a guide, not as a law.»[4]

Interaction patterns are tested solutions that answer common design problems. Using them as a guide can help expedite and simplify design work in an interface. Using interaction patterns also ensures that users will find familiar elements in the interface, making them feel more comfortable and confident when using the app.

Navigation

Simple and consistent navigation is an essential component of user experience in an app, involving numerous questions. How will the user move around the app? With menus, or with the content itself? What happens when there is a notification? How can the user go back?

Tabs

Tabs can be used to filter content or move among screens that, according to information architecture, have a hierarchy that indicates where the user is and what's next.

Best practices indicate that it's always necessary to indicate the tab selected. Order and location must be maintained and can't change from one screen to another, and they mustn't be used to include actions besides navigation.

Android uses tabs in the upper part of the screen (just below the action bar), which, unlike iOS, are generally used for the second navigation level. It has two different tabs: fixed (always

4 http://www.androidpatterns.com/

FIG 7.3. The tabs are located on the top area in Android and Windows Phone and on the lower part in iOS.

visible) and sliding (the user can see the current tab and two adjacent ones). In any case, Google suggests applying a golden rule of using no more than five to seven tabs.

iOS, on the other hand, always locates them in the lower part of the screen. In the iPhone, it is possible to visualize a maximum of five tabs. If more are needed, the last tab will read *more*, and house less relevant sections. According to Apple's official guides, tabs should always be visible and are best used for organizing the most important content within the hierarchy.

Windows Phone goes against the visual metaphor of tabs with something called *pivot control*, even though the function is the same. They are always located in the upper part of the screen and, because of their large size, play the part of title and tab at the same time. The recommendation is to use them in complement with and as subordinates of the Panorama menu, which navigates higher content hierarchies within the app.

FIG 7.4. Use of lists in Android, iOS and Windows Phone.

Lists

Depending on how you look at it, all vertically-arranged structured content can make up a list. This way of showing as many items as necessary allows the user to tap any of them to obtain complementary information.

Lists can show texts or images, but it's always important to arrange their content hierarchically. For example, in an email app, it's normal to give more importance to the sender than the subject, date received or first lines of the message. Arranging the elements inside a list helps visualize the name of the sender—in this case, the most relevant piece of information—at a glance.

When there are several elements, an index system can be added to complement navigation as you scroll through the content list.

In Android, the use of lists is widespread. Google's style guides recommend grouping related content, like in the settings page of the operating system, for better comprehension. As users, we are able to memorize only a few elements in the short term.

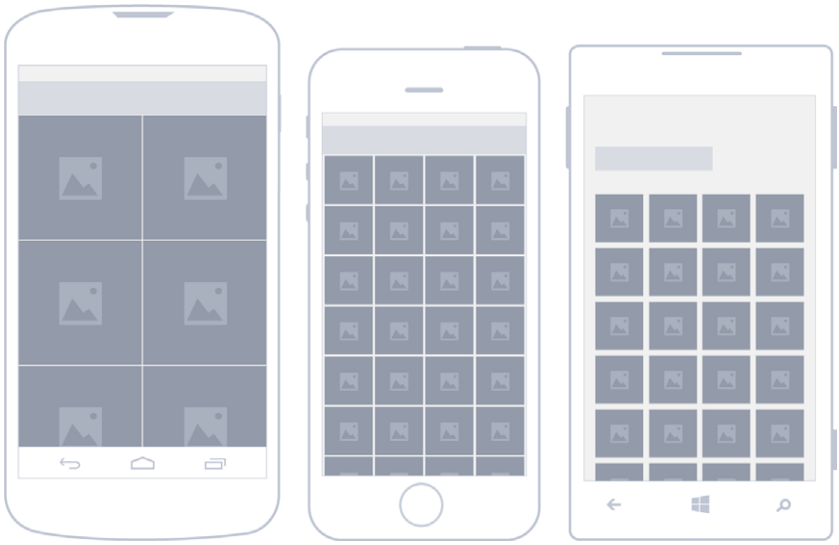


Fig 7.5. Image galleries use a reticular format in Android, iOS and Windows Phone.

In iOS, lists also occupy the entire screen width. While some pile one item over another, there are also grouped lists that allow for the division of content into blocks separated by vertical spaces. It is common for iOS lists to have an arrow on the right of each item.

In the case of Windows Phone, the List View shows a list of items that may contain images and texts, and it adds the possibility of showing data in a grid.

Image Gallery

Image arrangement is determined by the grid proposed by each operating system. When they exceed the space available, a crop is performed (generally square) of the images on display.

Android is a particular case. When showing images in the grid view, it's possible to use a horizontal scroll. When this happens, it's advisable to show a small slice of the following images.

Drawer Menu

This pattern, made popular by Facebook, allows users to switch rapidly between an app's screens. With the tap of a button, a side list with hidden content appears. Another way of getting to the list is sliding the finger from the left side of the screen.

Fig 7.6. The drawer menu has become very popular in Android, iOS and Windows Phone, even though, for now, only Android incorporates it into official guides.

The use advantages of this pattern are clear: better use of space and, once the list is displayed, a comfortable way of navigating content. But this format isn't without its drawbacks, as it forces users to tap the button and display the panel to get a full view of the options available.

Up to now, the only operating system that has standardized the use of drawer menus in official guides is Android. It has been recommended for the highest navigation levels of the app, or when menu options are not directly related.

There are many iOS apps that use the drawer menu as well. A nice example of this is Path, even though it's not recognized by design guidelines. The social network began implementing this type of navigation with its second version.

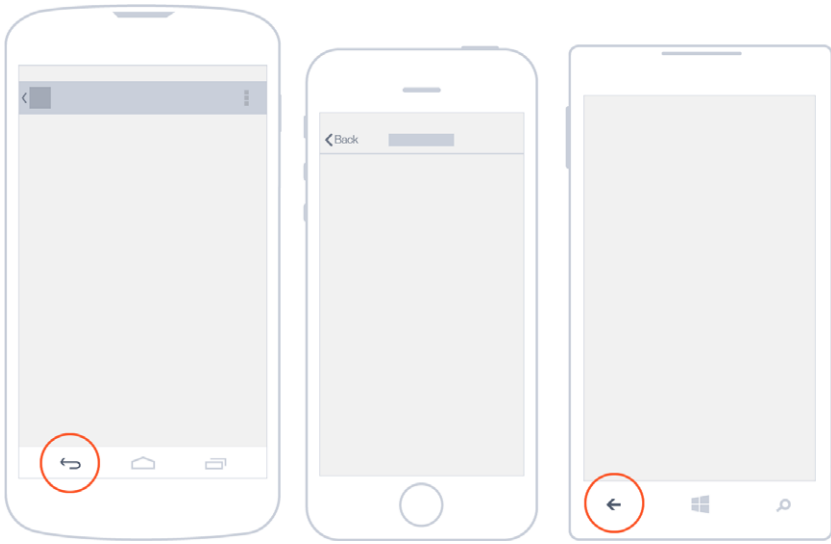


FIG 7.7. The back button in Android uses up and down; in iOS, always up; and in Windows Phone, it's the physical button of the device.

However, drawer menus are quite uncommon in Windows Phone. In fact, the Facebook app is one of the few that uses it to navigate content on such devices.

Back

Users who are accustomed to the web will find this to be a familiar way of navigating. As users progress in content, it becomes necessary to have a way of returning or going back to upper levels. In the mobile world, with screen-to-screen navigation, the use of the back button is very frequent.

In iOS, this button is inside the navigation bar, located on the upper-left corner with a label that displays the title of the previous page. Here, navigation among pages is hierarchical.

In the case of Windows Phone, the physical button of the telephone outside the app interface is in charge of managing this way of navigating content. In fact, the interface of the app should not contemplate the use of the button, leaving it entirely to the operating system.

Android Confusion: Up or Back?

From its fourth version and onwards, Android has proposed a new way of navigating an app's information structure based on the hierarchical relationship between screens. Thus, it has introduced the up button. On the app's home screen, it shouldn't appear, because there are no superior levels.

Fig 7.8. Currently, Android has two ways of navigating back, which can be confusing.

Meanwhile, the back button is physical in some smartphones and incorporated into the virtual navigation bar in the system's interface in others. It is always visible and used to navigate to the previous screens visited by the user, in chronological order. If the button is tapped at the top level of the app, the user will exit the application.

These two buttons work in a complementary manner, but they can also represent conflicts. Both should be kept in mind in order to ensure excellent user experience.

Windows Phone and Panorama Navigation

Panorama is an exclusive pattern that belongs to Windows Phone and offers the possibility of navigating content horizontally, as if each of the screens were next to one another. Moving through the screens gives a sensation of content continuity, supported partly by the use of background images.

Panorama is recommended for the superior level of navigation, as if it were a home page. From there, the main parts of the app can be displayed along with content previews.

This pattern is composed of a background image, title (generally, the app's name or logo), section titles and previews.

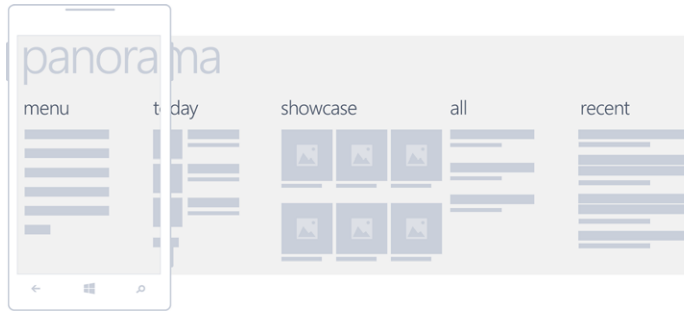


Fig 7.9. Windows Phone proposes a way of navigating first-level screens using Panorama by moving horizontally.

Actions

What actions are necessary right now? What actions would the user expect to find after accessing the screen? Which of the actions is more important? These are some of the questions that should be asked when defining the actions that will be found in each screen of the app.

Most actions can be performed only on the determined pages where they make sense. However, there may be exceptions, when it's necessary for an action to be visible all of the time. If this is the case, important actions should be highlighted in a very obvious way. This is what happens in Twitter with the action to compose a new tweet, which is present in the interface most of the time.

Actions may be located in different places according to their hierarchy and function, where the more important are visible and the less important are hidden. No matter where they are located, their position should be consistent across different screens and apps in the operating system.

Action Bar

In all systems, the summary of actions that can be performed is represented by means of icons. That's why the correct selection of graphic resources is fundamental.

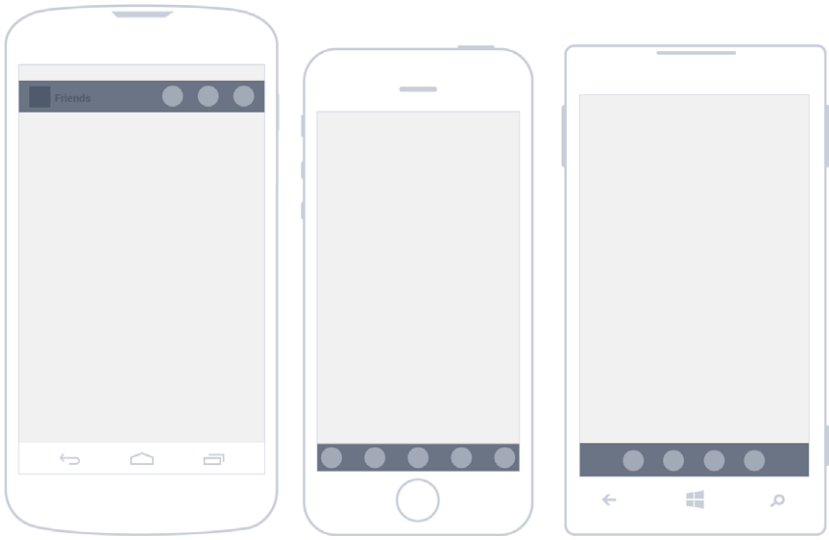


Fig 7.10. The action bar is located on the upper part in Android and in the lower part in iOS and Windows Phone.

In Android, action buttons are found in the upper right area of the interface. There may be a few exceptions in which they're located in the lower part of the screen, separated from navigation. It is essential to arrange actions according to their use frequency. The screen's width will determine how many items can be shown: from two in the smaller mobile devices to five in tablets.

In the case of iOS, the most common location is the lower area, even though in the iPad, action bars are located on the upper right.

Likewise, Windows Phone always uses the lower area of the screen to locate the actions that can be performed. In fact, only the most common actions can be displayed (a maximum of five). It may also be the case that there is nothing to highlight, in which case, it's convenient to use the minimized bar.

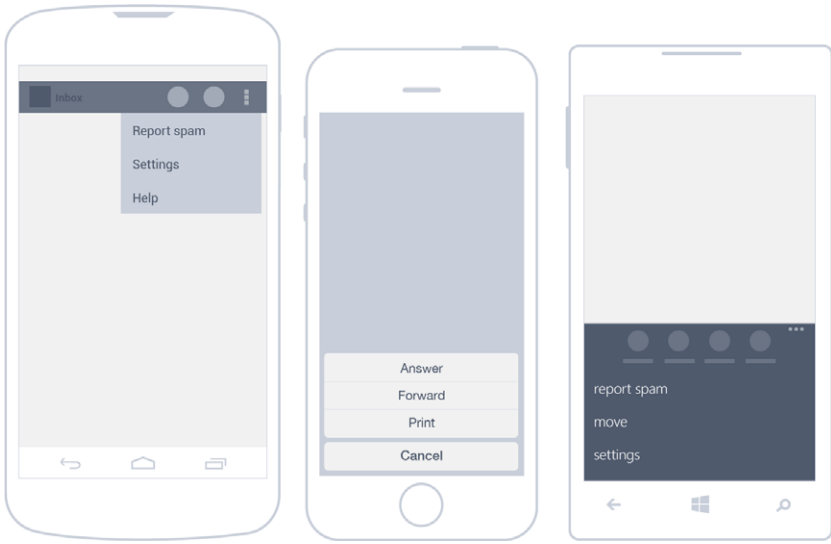


FIG 7.11. Android, iOS and Windows Phone offer different alternatives for solving action overflow.

Action Overflow

Additional and less frequent features are discovered by means of progressive disclosure. Basically, they are hidden most of the time until the user claims them.

In Android, the options that don't fit into the action bar are automatically displayed as overflow actions. They can be accessed through a button with an icon of three vertical squares that opens them in a list format.

Apple's proposal is to group related actions, initially hidden, and then show them in the button list format.

In Windows Phone, as well as in Android, additional actions are located below the actions bar and are indicated with dots that, when tapped, reveal hidden options in a list format.

Shortcuts

There are certain actions that should be readily available so that users can achieve their objectives quickly, for example, accessing actions associated to items in a list or grid without having to navigate deeply to find them.

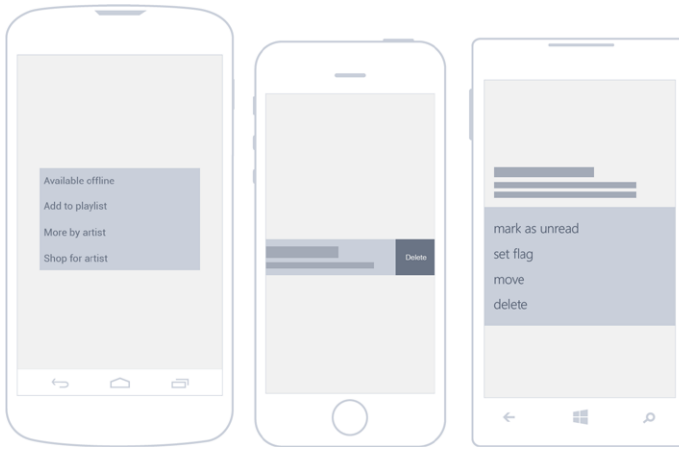


FIG 7.12. Ways of accessing actions quickly in Android, iOS and Windows Phone.

In the case of a music app, it is bothersome if adding a song to a playlist requires deep navigation. In such situations, it's advisable to use shortcuts to simplify repetitive actions.

In Android, before version four, it was possible to access shortcuts by tapping an item continuously, thus displaying a menu of possible actions. Currently, that gesture (long press) is

FIG 7.13. The share action displays downwards in Android, upwards in iOS and appears in full screen in Windows Phone.

recommended only to access the edit mode in a list. Meanwhile, it's possible to create a shortcut by means of a triangular icon located at the bottom of the elements that have these kinds of activities associated with them.

A pretty common example of a shortcut in iOS is when there is a need to eliminate an item from a list. A horizontal swipe is performed over the row to delete it. On the other hand, related actions are located in situ, on the item itself.

In Windows Phone, shortcuts are unfolded in a contextual menu format when holding the tap on an item.

Sharing

Sharing is probably one of the most popular actions today, via Facebook, Twitter, text message, etc. Operating systems have taken note and integrated sharing options accordingly.

Search

Taking into account that one of the main uses of mobile devices is content consumption, the search tool is an essential way of

FIG 7.14. Search is located in the upper part in Android and iOS, and Windows Phone uses a separate screen.

finding content. In the case of apps that display loads of data, search may even be the main feature.

A search can be performed by introducing text—the most common method—or by voice. Whenever possible, it's best to show results as the user writes out the inquiry in order to improve experience. Ideally, the wait time from data input to results should not exceed one to two seconds.

Android makes the search option available from the action bar. If search is an important feature for the app, as in the case of Dropbox[5], it should have the first position in the bar. When tapping search, the upper bar is modified and becomes the search bar.

In the iPhone, it's normal to find a search field over lists like Contacts and other apps. Within the text field, filters may appear for refining complex searches.

In Windows Phone, search is treated as a separate action, available in the action bar and carried out through a distinct page where text is introduced and results are listed.

List Editing

The user may need to modify several elements from a list simultaneously. The flow of accomplishing this is quite simple: the elements in question are selected, and the corresponding action is performed.

For example, to add a tag to three received email messages located in an email list, they must first be selected, and then, a desired action is chosen. So far, nothing revolutionary.

Now comes the interesting part: the way of selecting elements from a list varies considerably from one OS to another. Let us explain.

To make a multiple selection in Android, an element must be long pressed. Once the first item is selected, the action bar changes, indicating how many elements have been selected

5 https://www.dropbox.com/mobile

Fig 7.15. All three operating systems, Android, iOS and Windows Phone, present different alternatives for making multiple selections from a list.

and what actions are available. To leave this view, Android proposes the use of the check button.

In iOS, list edit is activated with the edit button, located in the navigation bar. As such, the selection is shown together with related actions. In order to exit the view without applying any changes, in the same place where access was granted, a cancel button appears.

In Windows Phone, there are two ways to access a selection. It can be done with the action bar, where the user finds the select option, or by using a shortcut, which is very practical once learned. The trick is to select the left side of the first item so that the list is shown in edit mode, allowing for the selection of several items at once.

Dialogue Boxes

There are particular cases in which users must be interrupted temporarily to prompt a decision or better explain something

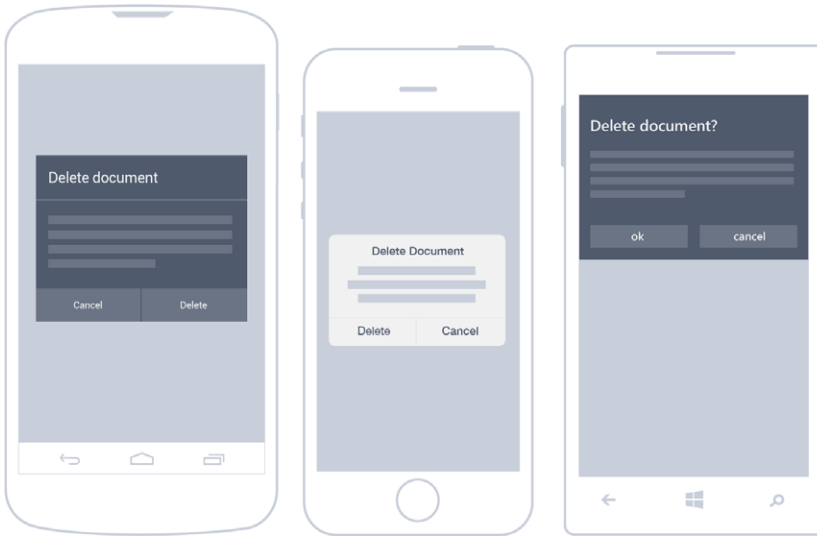


Fig 7.16. Dialogue boxes that require the user to make decisions in Android, iOS and Windows Phone.

that has happened before continuing on with a task. When dialogues are visible on screen, it's not possible to do anything else in the rest of the app.

When boxes only contain notices that don't require a decision, they are only informative and have one button to be closed. It's advisable to limit their use for serious or transcendental messages that absolutely cannot wait.

Otherwise, dialogue boxes are used to communicate to users the need to make a decision, with two or more options available.

Android uses dialogue boxes extensively. Requests can be as simple as *OK* or *Cancel* or complex designs with a form inside.

In iOS, dialogue boxes are located in the center of the screen, most often with one or two buttons in the lower area. If there are more than two buttons, they appear in piles. These dialogues are usually so simple that they only have a title with a short description.

In Windows Phone, dialogue boxes appear in the upper area. They generally take up only part of the screen but, in exceptional cases, occupy it entirely.

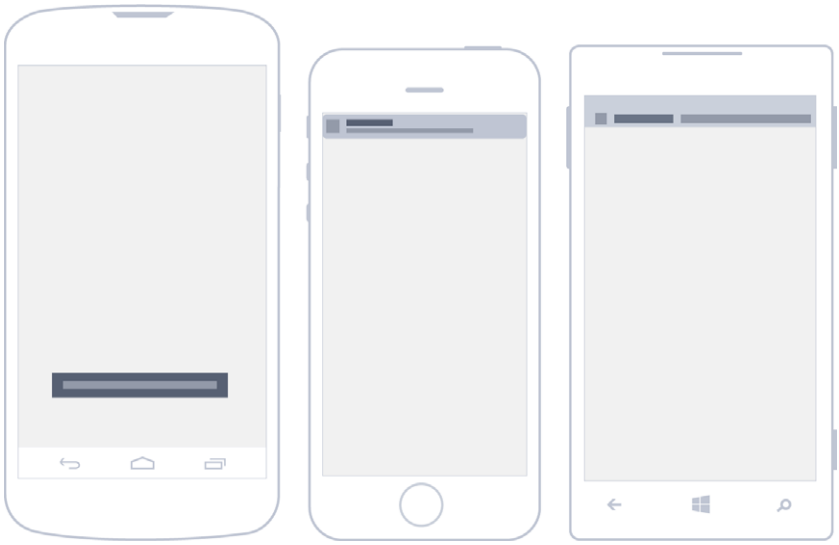


Fig 7.17. Only Android offers in-app native notifications, while in iOS and Windows Phone, they must be programmed.

In-App Notifications

What is the app doing? How do I know if the action has worked? Is it over? Do I have to do anything else? These are questions that plague users when there is no visual confirmation of their actions.

To mitigate uncertainty, it is advisable to display explicitly how things are progressing and what will happen next. These kinds of messages are presented in small announcements that appear after a few seconds.

Unlike dialogue boxes, notices don't require the user's intervention and don't interrupt workflow.

Android incorporates toasts. Visually, a toast consists of a small, rounded rectangle in black, located in the lower area of the screen and above any other element in the interface.

This notification appears for a short period of time, with text (generally only one line) that provides the user with feedback—an example being while an app is saving changes.

FIG 7.18. Different kinds of keyboards used for data entry in Android, iOS and Windows Phone.

Because it's a kind of warning that can go unnoticed, it is used to communicate messages that are not critical.

iOS and Windows Phone don't have a concrete solution similar to the Android proposal. Notices inside the app are left to the designer's criteria, for example, by means of external libraries.

Data Input

Data input on a mobile phone can be tedious when fields require the use of the keyboard, an element that occupies a large part of the screen and makes navigating difficult when introducing information.

All operating systems have developed different keyboards, depending on the kind of text to be entered.

If keyboards can be avoided, it is often better to use other alternatives, such as sliding menus, drop-downs and checks. Alternatively, there are also hardware components, such as

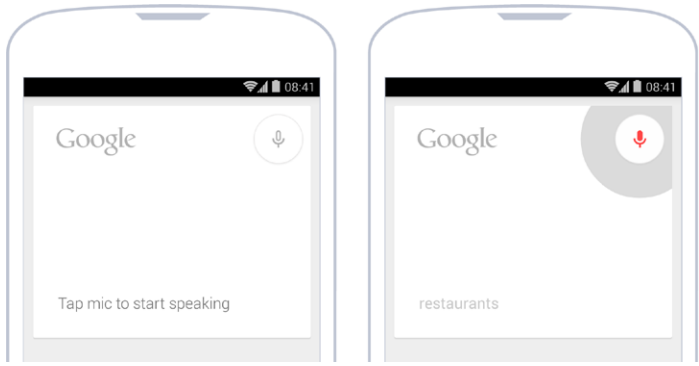


Fig 7.19. The telephone's microphone can be used for data entry without the need to use the keyboard, as in the case of Google.

location sensors, cameras and microphones, which can also be used to input data into an app.

An example is Google, which offers the option of voice searches and utilizes geolocalization.

Gestures

Tapping is the main input method of modern mobile phones. Everything depends on the user's hands, which manipulate the elements directly on the screen. Action and reaction happen in the same place, similar to the real world.

When screens with multiple gesture support were introduced, it seemed as though tactile interfaces were going to get considerably richer. The reality is that, a few years later, it can't be said that the most complex gestures have been extensively adopted by users. This is, in part, because the more complicated the gestures, the fewer people can perform them[6].

On the contrary, simple gestures such as tap, drag and swipe, which require only one or two fingers, have been assimilated very well. Users find them natural and familiar.

6 Fidalgo, Armando. Interfaces táctiles: el desafío de las tabletas. http://www.slideshare.net/Afidalgo/interfaces-tctiles-ux-spain

You can take advantage of the use of gestures when designing apps. They should be considered the means for all actions and content navigation. Here, it's also important to take advantage of the user's previous knowledge and to be consistent with the operating system.

It should be possible to perform the app's basic actions with simple gestures to ensure that users can perform them, leaving the most complex gestures as an alternative to interact with the app's interface.

Each OS has attempted to impose its own conventions[7], but luckily, there are some gestures that are shared by Android, iOS and Windows Phone. Below is a list of the most common gestures and their uses:

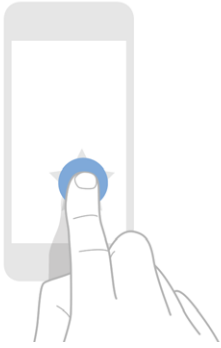
Tap

Tap the surface with the tip of the finger.

- *Selects the main action.*
- *Selects the main action.*
- *Selects the main action.*

Drag

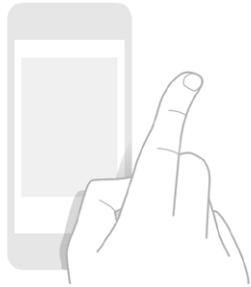
Move a finger over the surface without losing contact.



- *Delete by dragging horizontally in lists.*
- *Showing a delete button by dragging horizontally in lists. Move items in lists.*
- *Change to other tabs or sectors of a Panorama view.*

7 Wroblewski, Luke. Touch Gesture Reference Guide. http://lukew.com/touch

Slide
Slide quickly and without stopping the tip of a finger on the surface.

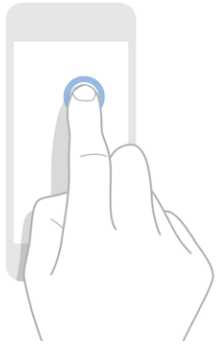


- *Browse content. Change to other tabs.*
- *Browse content.*
- *Browse content. Change to other tabs or sectors of a Panorama view.*

Long press
Tap the surface during a long period of time without moving the finger

- *Enter the list edit mode.*
- *Show a tooltip. Increase the visible content under the finger.*
- *Show a tooltip without selecting the element.*

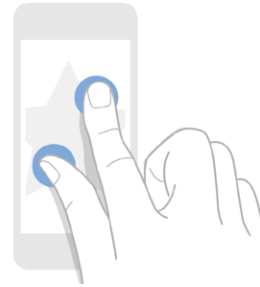
Double tap
Tap the surface rapidly twice with the tip of the finger.



- *Zoom in and zoom out. Select text.*
- *Zoom in and zoom out.*
- *Zoom in and zoom out.*

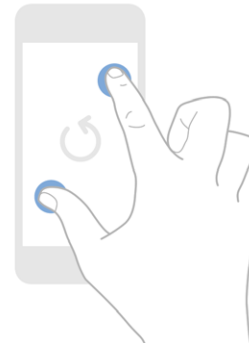
Pinch and spread
Tap the surface with two fingers, spread and separate the fingers.

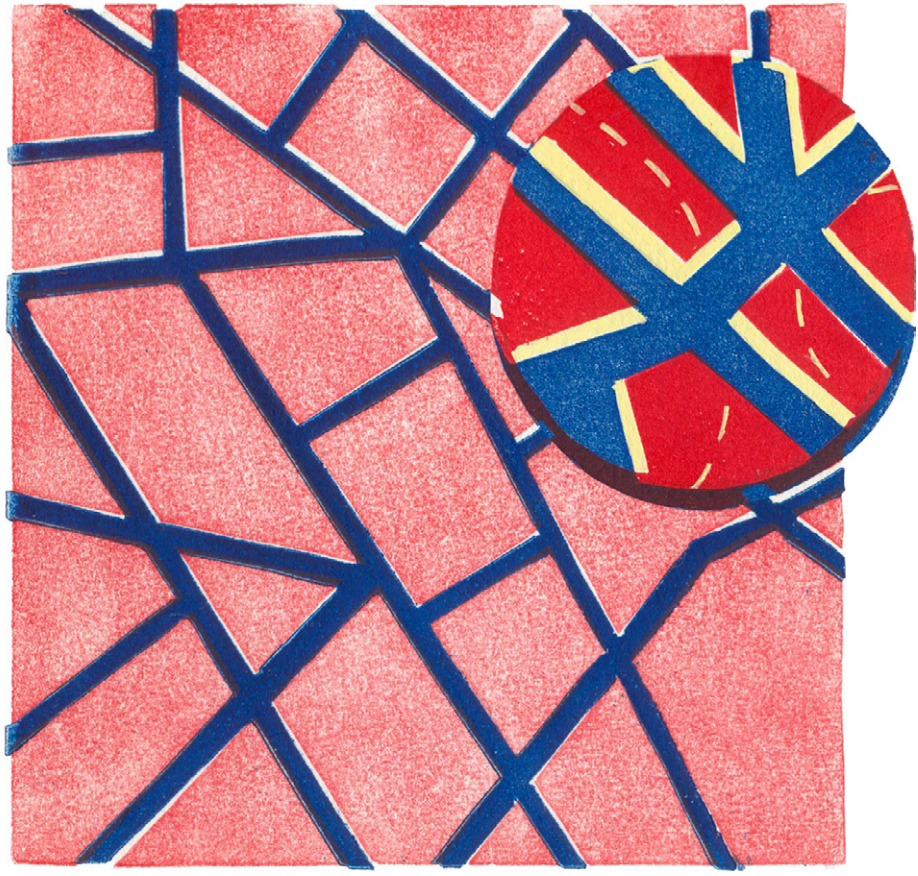
- *Zoom in or zoom out.*
- *Zoom in or zoom out.*
- *Zoom in or zoom out.*



Rotate
Tap the surface with two fingers, spread and separate the fingers while rotating the wrist.

- *Rotate an image or a map.*
- *Rotate an image or a map.*
- *Rotate an image or a map.*





8

Visual Design

Visual design is, for many designers, the most interesting stage of the process. Here, wireframes are brought to life with a style defined by the designer and the personality of each operating system.

Interface Styles

An app's interface is like the clothes a person wears on the street. It is the layer that separates the user from the functional core of the app, the place where interactions are born.

To a greater extent, it is made up of buttons, graphics, icons and backgrounds, with different visual appearances in each of the operating systems. Android, iOS and Windows Phone have their own ways of contemplating design.

The designer's job consists of interpreting the personality of each operating system and pouring in his or her own vision and design style in order to make apps that, besides being easy to use, are different from the rest and have visual coherence with the platform.

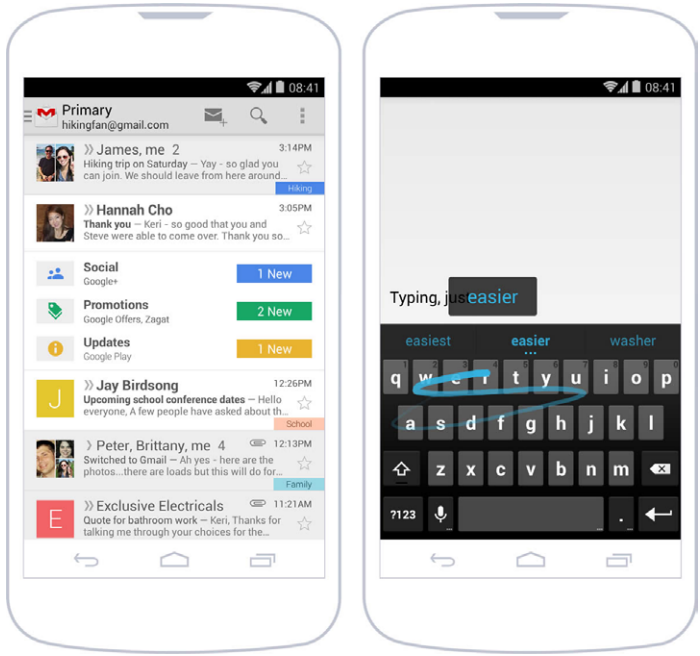


Fig 8.1. Android has a simple design style, with Roboto as the main font.

The Simple Beauty of Android

Android's design is based in a brilliant cleanliness with regards to the composition of the interface. Each graphic, button and text is accompanied by the idea of visual uncluttering, at the same time dazzling with the smaller details.

Roboto, the operating system's typeface, is what gives it a lot of its identity. It is combined with a well-defined style of colors and buttons. Android relies on simplicity that is controlled but not boring, and on certain occasions, it breaks or transcends its own formalities to captivate the user.

iPhone, Searching for Visual Lightness

Leaving behind the style that characterized it up to its sixth version, iOS currently advocates an ideology shared with other

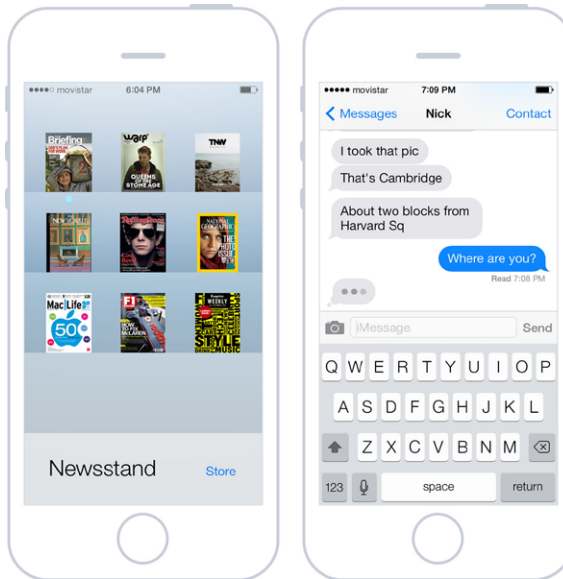


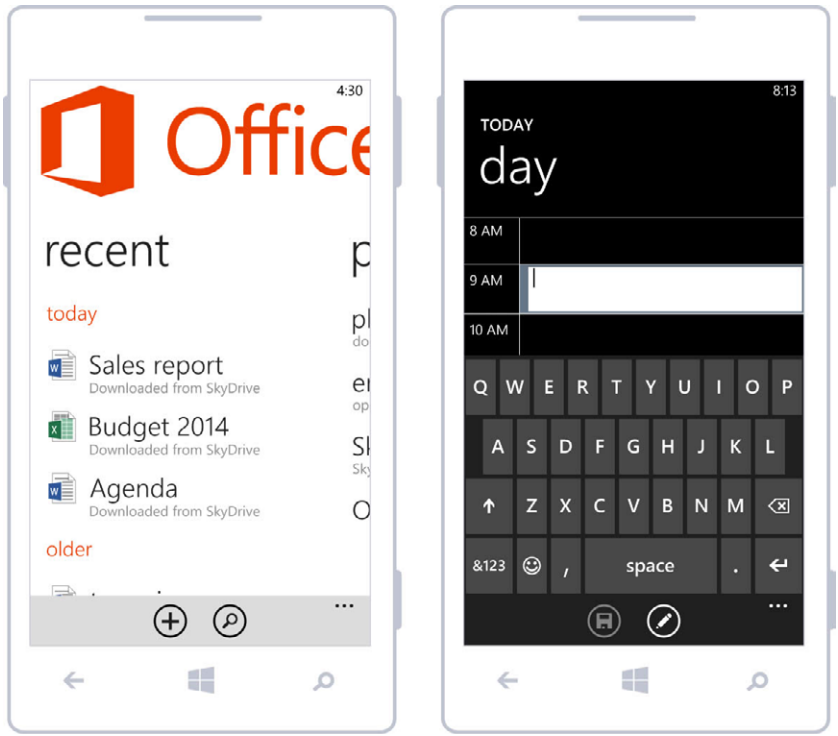
FIG 8.2. Visual clarity, as a result of the color choice and typography of iOS, attempts to order content hierarchically in backgrounds and secondary elements.

operating systems: strip away all unnecessary elements, privileging content over the container.

In order to achieve this, it has reduced controls and graphics to the very minimum, always with the idea of visually lightening the elements on screen. This concept is supported by typographic choice—Neue Helvetica, often in lighter versions—and colors—white backgrounds with stronger hues for icons and texts.

Each component of the interface is treated as a layer superimposed onto another, sometimes with a certain transparency and blurriness, which gives a sense of continuity and permanence in the context.

FIG 8.3. The Windows Phone's design has a flat style called Metro, based on its grid and typography.



Windows Phone's Flat Design

Windows Phone's design style is a flat one, without bevels, gradients or excessive aesthetic decorations. This visual cleanliness also applies to content. Only the most important content remains on screen, highlighting it within a specific context.

The interface consists of an infographic approximation for the icons, with a heavy use of the grid and typeface as one of the main resources to imbue the design with personality.

Native and Custom Interfaces

Native interfaces are based on elements like buttons, lists and headings that are preset in each platform. They have a defined aspect with respect to the basic characteristics of appearance, such as color, size and font, which can be adapted to a greater or lesser extent to attune them to the desired aesthetic.

When starting to design, it is often best to define the interface with native controls. This provides a solid base to start from and also eliminates the need to create all elements from scratch.

The inconvenience with native interfaces is that they limit the personality of the design. And sometimes, you really have to take the design to the next level. In such situations, all or some elements of the interface can be personalized and recreated as images. For example, a customized visual element could be a text field inside a form that is generated as an image to take advantage of the possibility of including textures, bevels or specific shadows that a native control may not offer.

If the idea is to design a custom interface, this should be planned out beforehand, because it incurs more complexity and development time. Likewise, not all designs for these kinds of interfaces appear accurately in the functional app, because their correct implementation depends on the developer's proficiency.

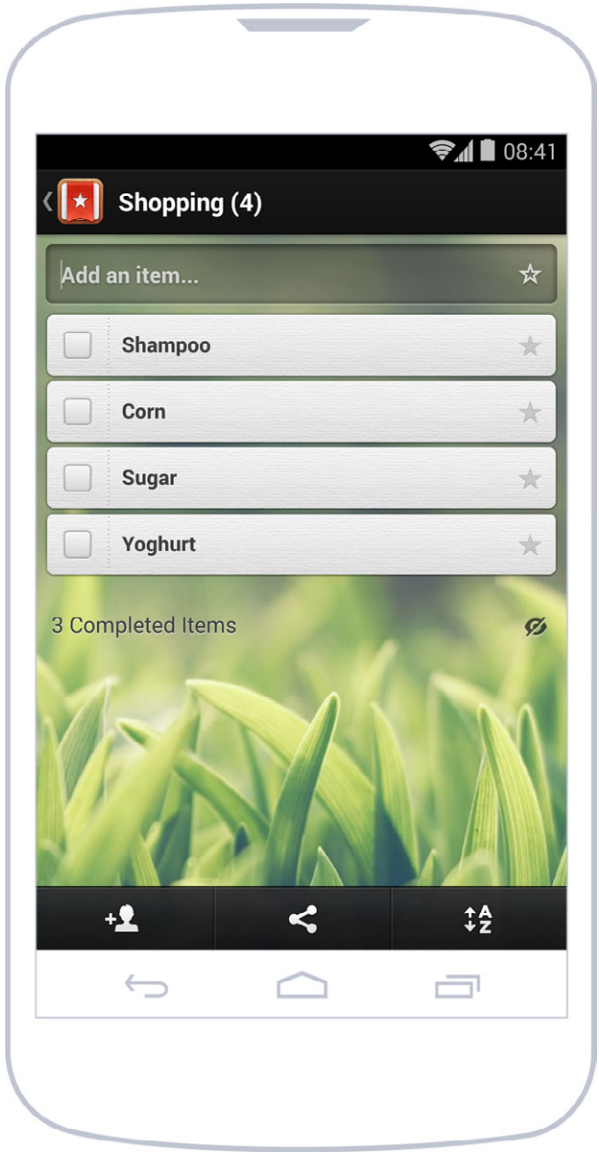


Fig 8.4.
The Wunderlist app incorporates custom controls to the interface, such as the input, that are well integrated.

Decisions, Decisions

But when should native and custom interfaces be used? In most cases, it's not about choosing one or the other, but about striking an adequate balance that combines both. It's most common to start from a native interface and customize only the elements considered necessary.

The kind of app also determines this matter. For example, apps that value visual details and experience in general tend to have more than one customized element. Such is the case of Path and its way of adding moments, something designed and developed especially for the specific nature of the app.

There are other apps that place greater value on completing tasks and benefit from a cleaner appearance that does not distract from the process. WhatsApp is visually based nearly entirely on the native elements of the operating system.

Apart from the objective and type of app, there are other variables that should be considered. Native interfaces have in their favor the fact that they are composed of elements with which the user is already familiar and therefore don't require any learning or adaptation. This can have a positive influence on an app's usability.

Customized interfaces, however, can offer a more finished look. But when working with them, it's necessary to consider their compatibility with multiple devices and performance in general (an excessive use of images can slow an app down).

Visual Identity

An app is, among other things, a communications tool. It is part of a system and an opportunity to extend a company or a product's identity. Through the different screens of the app, colors, fonts and backgrounds act as elements that reflect that identity.

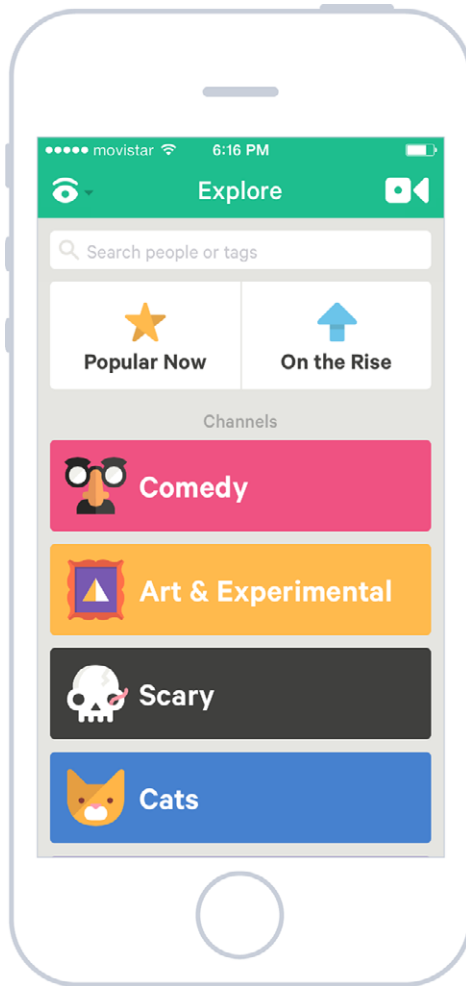


FIG 8.5.
Vine combines the corporate color in the header with other colors in hashtags.

Clearly, one of the components of identity is brand. Even though it may be tempting to make extensive and repetitive use of the brand, it's best to include it in places designed for that end, for example, on introductory and login screens, or in the about section. This ensures the correct exposure of identity without affecting navigation or user experience.

Icons and Launch Image

First impressions are lasting impressions. And in the world of apps, that first impression is limited to two visual components: the launch icon and the launch image—also called splash—that will appear when an app is opened.

These elements will be seen before anything else, even before the app is in use. You've got to appreciate their importance and pay attention to how they're dealt with to start off on the right foot.

Launch Icon

An app is a product on a shelf next to many others, and the launch icon is its packaging.

This icon will represent the app in different app stores, along with screenshots and promotional texts, and serve as a sales element to get users to download.

Once this step has been completed and the app is installed, it will exist next to the many other apps that are also in the user's possession. That's why the launch icon should be dis-

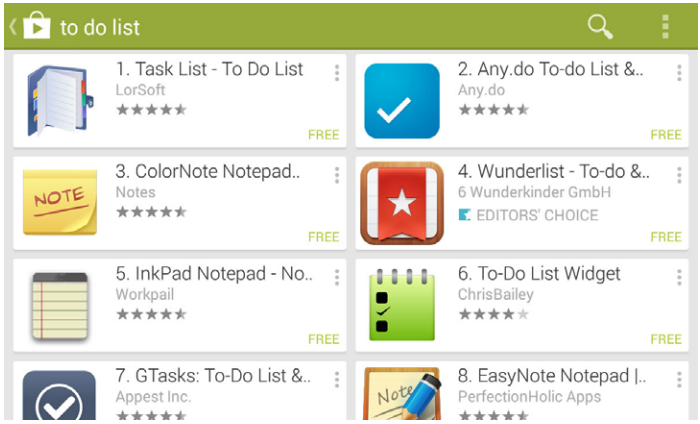


Fig 8.6. An app faces lots of competition when a user finds it in search results, in this case, Google Play.

FIG 8.7.
The launch icon of an app must have personality to set it apart from the rest, but it must also clearly represent what the app does. In the example, icons are shown corresponding to a few to-do apps and one social network. Which one do you think is the social network?

tinctive and representative of the app. Distinctive, because it must differentiate from the rest, even from those with similar functions; and representative, because the visual characteristics must communicate clearly the main objective of the app. Simple, uncluttered shapes with significant attention to detail are those that are most effective.

Size is also something to take into consideration. An icon may seem large in an app store but become much smaller once installed. When designing, all possibilities must be considered, and more or less detail should be added and subtracted according to dimensions.

To be more specific, each OS has different requisites that the launch icon must fulfill. Android[1] and iOS[2] have detailed visual style and technical requirements.

In Android, icons are objects represented from the front, with a slight perspective, as if they were being watched from above. They portray volume and depth, playing with transpar-

1 http://developer.android.com/design/style/iconography.html
2 http://developer.apple.com/library/ios/#documentation/userexperience/conceptual/mobilehig/IconsImages/IconsImages.html

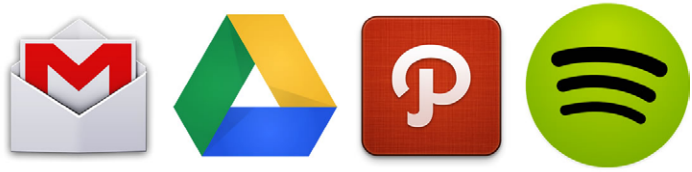


Fig 8.8. In Android, icons usually have shades and a slight perspective.

ency to integrate better into the screen. Shapes are distinctive, and realism is limited.

The situation is different in iOS. Generally, icons are highly simplified representations of real objects or abstractions of the

Fig 8.9. Detailed and realistic icons inside container shapes with rounded corners: the identity seal of iOS.

app's concept. The icon has only one main element, without many details, on an opaque background. To complete the visual look of the icon on the screen, iOS adds rounded borders to the image, which must be square and abide by size requirements.

Meanwhile, Windows Phone has a very characteristic style in which icons are pictographs. Shapes are extraordinarily

Fig 8.10. Even though there are exceptions, good practices for Windows Phone indicate that launch icons should be simple, white shapes.

simple and in plain colors (especially white), with almost no details, and they are perfectly integrated with their container. Transparency is fundamental, since images are located inside a

square shape on a background color that can change according to the user's chromatic preferences.

Interior Icons

Once inside the app, interior icons have less prominent, more functional roles than the launch icon. They can go unnoticed, but their job is worth consideration.

Their use is generally associated with three scenarios. First, they act as visual aids to reinforce information, for example, in a dialogue box with an alert. Second, interior icons complement interactive elements, when they are inside buttons or tabs. Finally, they improve the use of space. In such cases, the icon visually condenses something that expressed in text would be too long or difficult to understand.

Icons must transmit the actions they perform by themselves, depending on their context. For example, a delete icon may be related to only one element in particular or to several, depending on where it's located and the element of the interface visually associated with it.

When icons accompany certain actions—if they don't have text labels that help exemplify their function—it's more impor-

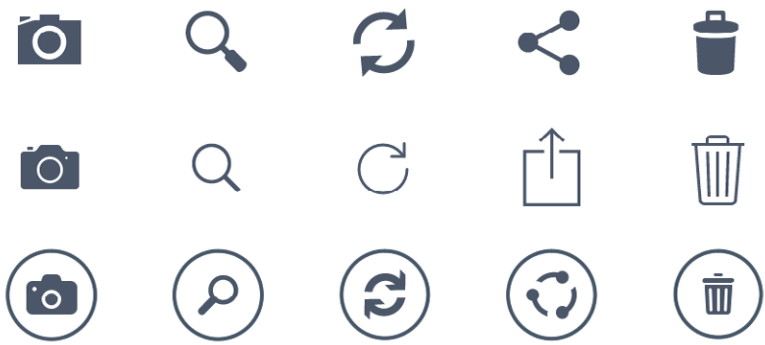


Fig 8.11. From top to bottom, interior icons for Android, iOS and Windows Phone.

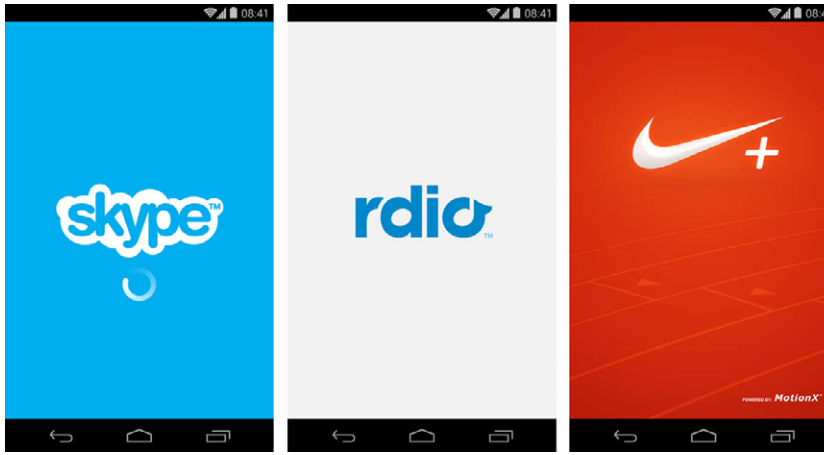


Fig 8.12. Launch screens in Android have little information and are only shown briefly during the initial load.

tant for them to be clear and representative. This happens when, because of space limitations, an icon and text can't be included at the same time. This is a trend that's becoming more and more common in applications.

The interpretation of icons assumes a certain amount of subjectivity that must be eliminated through correct use. For example, each platform has icons associated with actions like search, save and edit. The user already knows what they mean; therefore, using them as expected will help with consistency and improve usability.

Launch Image

Also known as a splash, the launch image is the first screen the user will see when the app is launched. Lately, the use of this screen has become more restricted or even avoided all together. It is generally shown quickly the first time the app is opened. This screen is the presentation of the content during the initial load, so it's normal to display a load indication with the rest of the graphic elements.

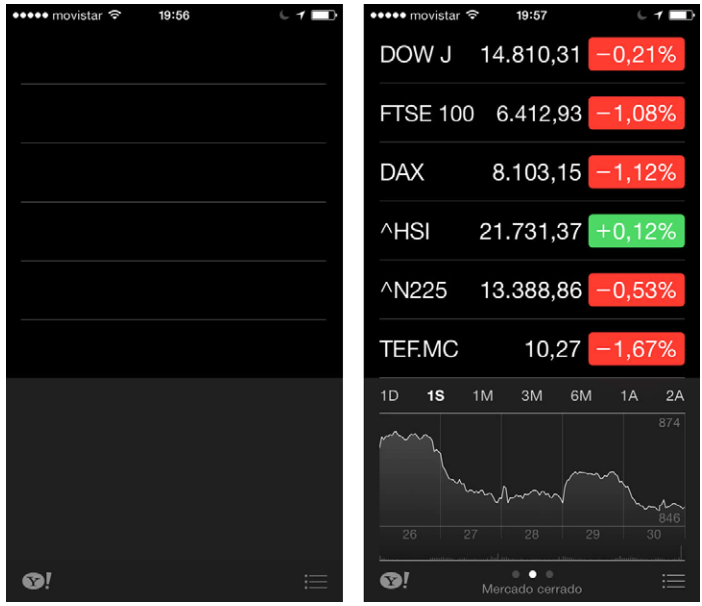


FIG 8.13. In iOS, it's advisable to use a launch screen that is a representation of the container of the information but lacks the actual information to gain a sensation of immediacy when loading.

The launch screen is so ephemeral that it is only visualized for a couple of seconds at most. Because of its short lifespan, the information shown should be limited to the name and version of the app.

In some cases, this splash is a representation of the app's container: an almost identical image to the one the user will see when the app is finished loading, but without the data that might change, such as text, button labels and status bar elements, seemingly making it more fluid to load[3].

In the case of Windows Phone, the operating system is in charge of managing the launch screen, consisting of an enlarged launch icon.

As for the orientation of this screen in smartphones, it's worth remembering that it is generally displayed vertically. With tablets, it's necessary to determine the orientation being

3 http://blog.manbolo.com/2012/06/27/of-splash-screens-and-ios-apps

used when the splash is shown, and according to that, use the corresponding version for portrait or landscape orientation.

The Grid

The grid is the invisible structure upon which all visual elements are propped. Their function is to separate each of the components of the interface into a neat space, organizing the areas that will be left blank and those that will have shapes. A well-defined grid transforms into a design aid that, by generating order and simplicity, improves the app's usability.

In its most basic shape, the grid consists of a simple module: a square of a certain size that is used as reference. In turn, this module can be divided in submultiples for smaller spacing.

When interface design is being developed, the grid is represented with guiding lines. Once the structure is finished, it can be perceived as a visual rhythm that places elements harmoniously in a space.

When designing interfaces for mobile devices, the grid helps determine margins and button locations, the separation of fonts and the inner and outer space of the containers. Of course, each operating system has a different grid, and therefore, a different module.

Android

In Android, the basic module is `48dp`, equivalent to nine millimeters, the minimum size recommended for interactive elements. Complying with this size and abiding by the button dimensions ensures they can be touched with the finger without problems, a fundamental consideration in mobile design.

On the other hand, for spacing and separation, an `8dp` module is used. For example, row content has a separation

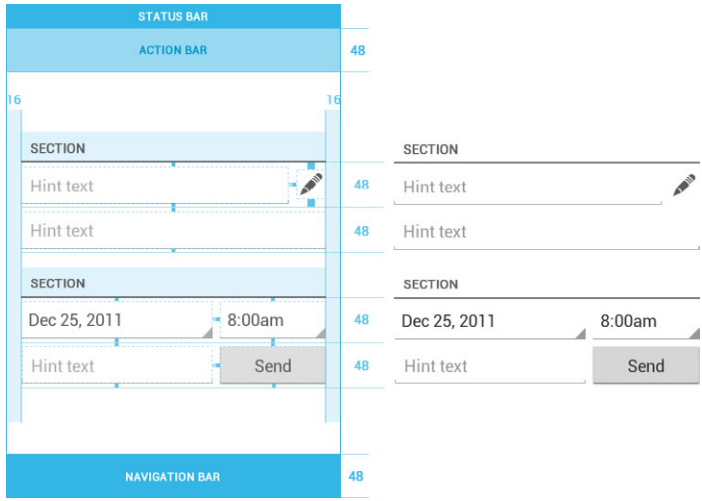


FIG 8.14. The image shows clearly how an interface in Android is made up of a basic module of 48dp.

(upper and lower) of `4dp`. As a result, when two rows are one on top of one another, they conform a total space of `8dp` between them. In the side margins, the designs are usually `16dp`, or two `8dp` modules together[4].

iPhone

Designs for iPhone are also based on a grid, but in this case, the base module is `44px` to ensure that buttons and list elements can be tapped easily. Many apps for iPhone follow this rule, and it is recommended by Apple.

This `44px` module is divided in others of `11px` that, when repeated as many times as necessary, create spaces and separation between tables, buttons and other elements of the interface, generating a vertical rhythm[5].

4 http://developer.android.com/design/style/metrics-grids.html
5 http://aentan.com/design/new-visual-proportions-for-the-ios-user-interface/

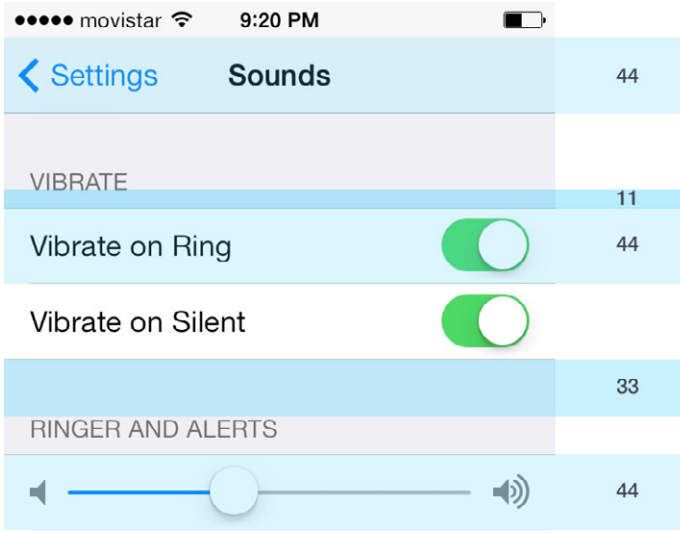


FIG 8.15. In iOS, the basic module has 44px.

Windows Phone

In Windows Phone, the grid is more evident because of the use of square shapes in the interface. The home screen of the operating system is based on tiles, which clearly demonstrate the distribution and size of the main elements.

The grid is based on a `25px` module with a `12px` separation. This formula, repeated on the whole screen, creates a visual structure where designs are created. Additionally, the rows and columns can group in different ways in order to accomplish personalized designs that make different uses of the space.

Lists, graphics, photo thumbnails and buttons use the grid as their base and, as a result, create a stable appearance that provides a sense of order, simplicity and visual cleanliness throughout all of the screens of an app in Windows Phone.

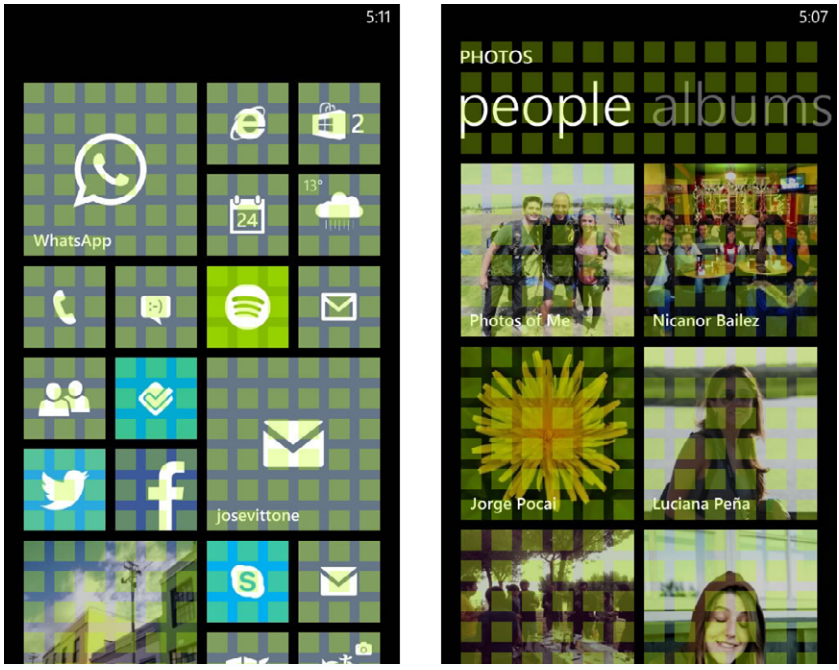


FIG 8.16. In Windows Phone, the grid is the cornerstone upon which all the elements on the screen are based.

TYPOGRAPHY

The goal of the typographic font is to make the text easily readable. This can be achieved not only with an adequate choice of font, but also by managing the size, line spacing, column width and visual contrast with the background.

On this last point, contrast is of the utmost importance. A mobile phone is a device often used outside of the home and on the street. The sun may shine directly on the screen, and if there isn't a good contrast of typeface and background, the text will not be readable.

Typography is a component that, like buttons and graphics, also rests on a grid that will define its location and position in the general context of the screen.

Serif or Sans-Serif

Generally, the debate on font choice is concentrated mostly on typefaces with serif and without serif. Admittedly, in small sizes and low resolutions, cleaner, more open and sans-serif fonts are better, but serif fonts can be considered for main titles, when size means that serifs will not be an obstacle to readability.

Even though it's not common to use mobile phones for prolonged reading, correct legibility is a fundamental part of design. As such, typefaces are as important as any other visual element of the interface and shouldn't be taken lightly.

Legibility and Resolution

Because mobile devices are a digital medium, they have their own characteristics and some limitations that are different from traditional print media. The screen has a lot of influence on typographical behavior and performance when taking into account that it is, in many cases, extremely small.

Currently, a number of devices have screens with good resolutions, eliminating a complexity factor and a need for concern when the time comes to decide on a font. These kinds of screens can be found in high-end mobile devices; however, there are other smartphones with less interesting characteristics.

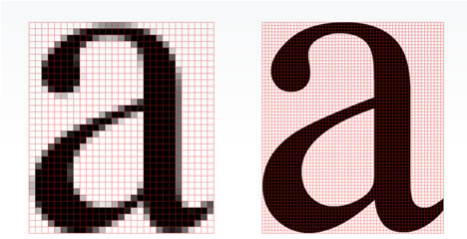


FIG 8.17. Characters are more legible in devices with high-quality screens. In low-density screens, typographic choice can be a big problem.

In the latter case, choosing typefaces is a quite difficult task that is not limited to the selection of family and size. The smaller the screen, the greater the chances that characters will be poorly displayed.

Minimum Sizes

In digital media, such as a smartphone or a tablet, something that conditions the choice of typeface size is the holding distance. Mobile phones are usually held in such a way that the screen is closer to the reader's eye than in the case of a tablet, allowing a smaller typeface.

At the same time, in smartphones, the screen space is much more constrained, forcing an adjustment of line spacing and character separation to make the most of the area available without affecting readability.

Minimum sizes can vary according to operating system, screen resolution and font. In any case, when using small sizes, it's advisable to choose simple and open fonts, with character, line and margin spacing to accomplish a visual separation that makes them easy to read.

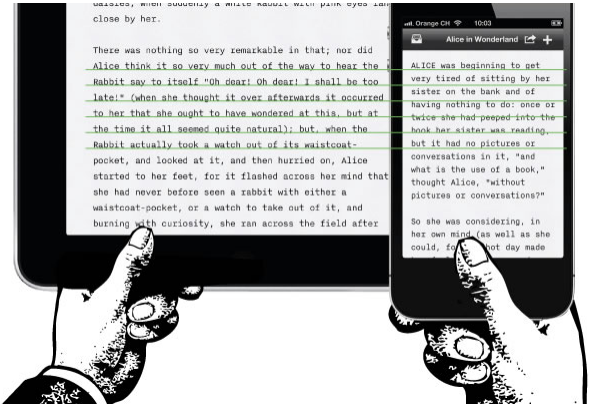


FIG 8.18. Screen size and reading distance condition the choice of font size, line spacing and letter spacing.

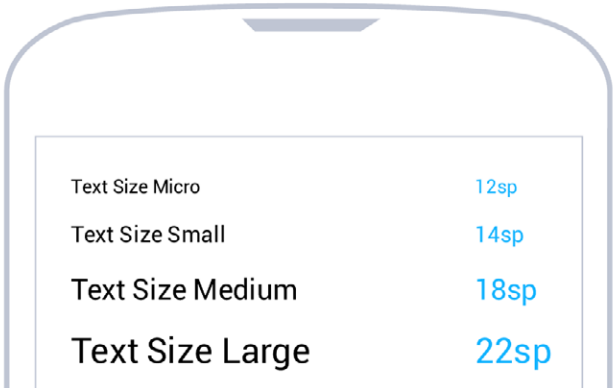


FIG 8.19. Different font sizes in Android according to the use and hierarchy of the elements.

The best way to ensure correct legibility is to test the font on the phone for which it's being designed. Computer screen designs tend to be deceptive, and testing designs in the most real environment possible enables adjustments and corrections until the adequate size is found.

In Android, font size is measured in sp (scaled pixels), a way of modifying a font scale according to screen size and user-defined preferences in the smartphone's configuration. The most common sizes are between `12sp` and `22sp`.

Size varies in iOS, depending on where the text is located. In the case of the retina density, main titles are around `34px`, and the size of labels inside important buttons is approximately `28px`. From there, it starts to get smaller in the different elements, nearing `14px`. However, it's advisable not to use dimensions smaller than `20px` in reading texts.

In Windows Phone, more than in other platforms, design depends heavily on typography. A bad choice of font size in an app that already has few visual elements can spell disaster. We recommend not using a size smaller than `20px` for smaller texts, and in the case of titles, considering sizes up to `70px`.

Hierarchies

As a visual element and component of an interface, typography is also subject to hierarchies. Its importance depends not only on function but also on the information that it contains and its position on the screen.

To define different levels of importance in a text, besides size, you must consider the other variables (bold, regular or light, etc.) and color.

A text with a higher hierarchy is that which is located as a main section title. On the other hand, the information contained inside a row in a list can have several hierarchies. In an email, for example, it could show the name of the sender, a summary of the content and the date sent. All of those elements are more or less relevant, and defining their importance is the

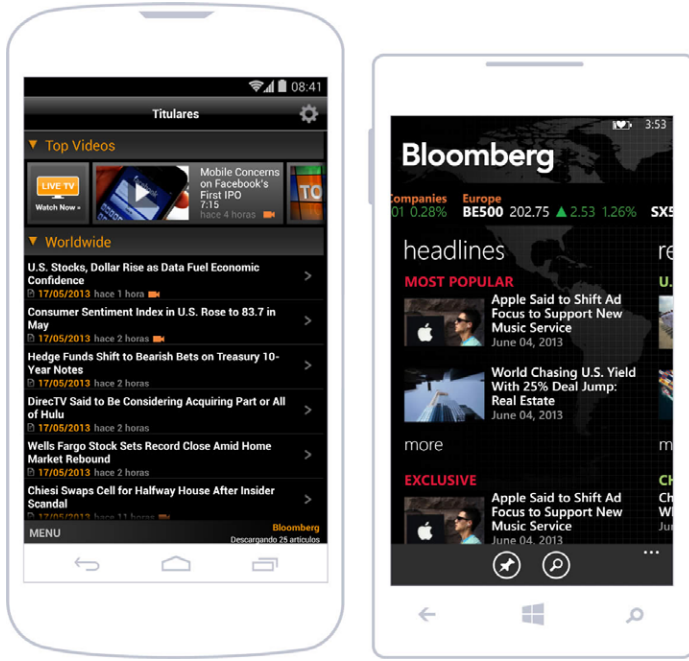


FIG 8.20. While in the screen on the left, it's not easy to distinguish a clear hierarchy. In the one on the right, the importance of each element is more evident, thanks to the use of type size.

first step in determining the typographic style that should be applied.

Operating System Fonts

Android, iOS and Windows Phone all have their typographic preferences and sets of system fonts. This doesn't mean the designer should necessarily follow them, but choosing one of the options available will help relate with each platform's identity.

In the case of Android, the Droid Sans family was historically the most often used and marked an age of the platform. However, in the latest versions of Android, it has been replaced by Roboto, which is especially designed for high-resolution mobile phones and boasts variants that go from thin to black[6].

Neue Helvetica is the most emblematic iOS font, and many apps use it. Over time, it has transformed into a brand identity for the operating system. However, there are more than 260 fonts available that can be used natively in iOS[7].

Windows Phone introduced Segoe UI as a hallmark that accompanies the clean, modern and geometrical style of its interface. Even so, it has a series of type families that complement it, particularly in cases where special characters are needed that are not found in Segoe UI, as in foreign languages[8].

All of these operating systems also allow the incorporation of additional fonts. But bear in mind that variety and quality are two very different things: many of the fonts available were not created for legibility on a screen.

6 http://developer.android.com/design/style/typography.html
7 iOS 7: Font List. http://support.apple.com/kb/HT5878
8 Text and fonts for Windows Phone. http://msdn.microsoft.com/en-us/library/windowsphone/develop/cc189010(v=vs.105).aspx

FIG 8.21. Neue Helvetica is a classic in iOS, in spite of the problems it has in reduced sizes.

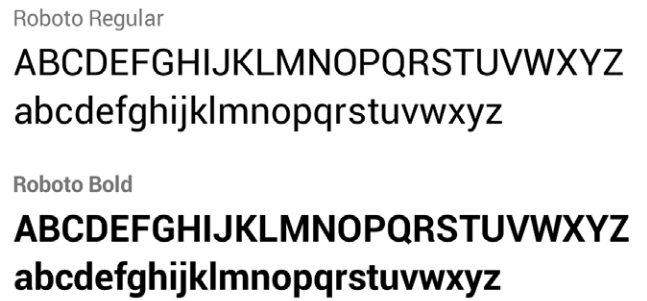


FIG 8.22. Roboto is Android's flag font, conceived with variants for mobile with high screen quality.

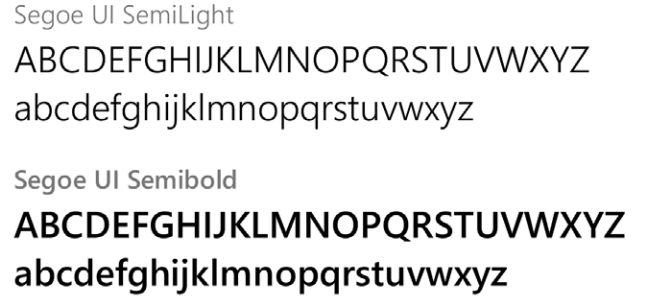


FIG 8.23. Segoe UI is the characteristic font of Windows Phone, the protagonist of the interface.

Color

Color is a vital resource in an app's design. The use of color comprises headings, texts, buttons, backgrounds and many other elements that make up the interface. In some instances, it's associated with identity (corporate color) and in others, it responds to aesthetic criteria and design decisions.

A color on its own, except in the case of reserved colors, doesn't mean much. As part of a chromatic system, it is consistent, conscious and context-related use that gives color meaning for the user.

Reserved Colors

Because they have connotations that can't be avoided, there are certain colors that must be used cautiously. They are referred to as reserved colors, and their use should be limited especially to those listed below:

- *Red:* For errors and important alerts. It's a color that naturally indicates danger and immediately catches the eye.
- *Yellow:* Warning. It indicates that the action about to be performed implies making a decision that causes a certain consequence; therefore, the user should be careful.
- *Green:* Success and confirmation messages that an action has been performed correctly.

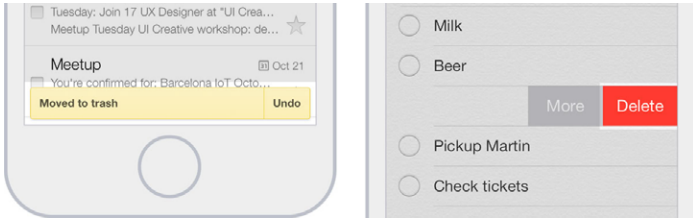


Fig 8.24. Colors like yellow and red are usually reserved to reinforce the meaning and importance of certain visual elements.

In Texts

Color can be used to highlight phrases and words that can be tapped, as in the case of links. It's important to maintain visual consistency so as to allow the user, intuitively and at a glance, to know what the interactive typographic elements are.

Another function of color in texts is to assign hierarchy to content. Complementary information can be highlighted or minimized depending on the color used. For example, content of a certain level of importance could be highlighted in the text by using a different color. In the same way, color can also be used to identify secondary information by using lighter tones that are less prominent.

In Backgrounds

Background color must be in line with the color chosen for the typeface, since a minimum contrast is necessary for reasons of legibility and accessibility.

Dark backgrounds can cause the eyes to strain more quickly, so if the app is frequently used or requires reading for a considerable amount of time, it's convenient to revise the chromatic choice and take the background color to a lighter alternative. However, dark background colors can be a good alternative when the content is highly visual, as in the case of photographs or videos, since it helps highlight these elements.

In Interactive Elements

It's possible to resort to color as the answer to or as feedback on concrete actions of the user, an option that many times is not taken into consideration. Selected or tapped elements like buttons and rows can be highlighted in a color that visually

indicates where the user has tapped, something that is particularly difficult to tell in a mobile device.

In the case of disabled elements, color is generally lighter than when the element is in its normal state, and it's even possible to resort to the use of transparencies. In any event, the objective is to indicate, in an obvious way, that the element will not produce an effect when tapped.

In Headers

Besides highlighting them as the main element, when color is used in headers, it must be in complete harmony with the background and the other elements on the screen. In the end, it's an important space that will be used, for example, to place the different section titles and other interactive elements that have an incidence on the content of the screen being watched.

Since headers are present in different sections of the app, chromatic consistency must be maintained when navigating different sections. However, some apps (an example being that of news publication *The Guardian*) make use of headers to play with variations of color that mark a visual difference between

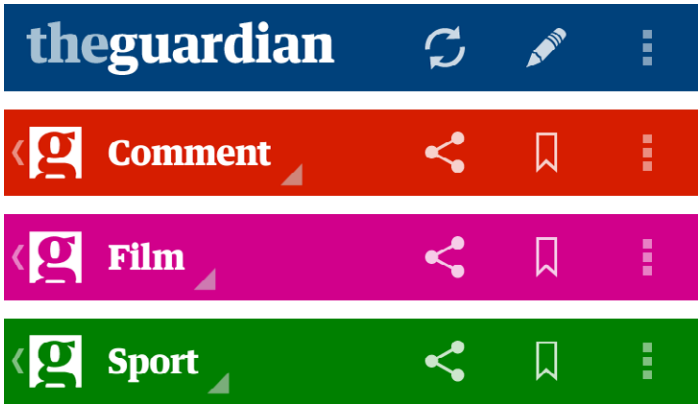


Fig 8.25. The app of the newspaper The Guardian uses color in the headers to differentiate each of the main sections.

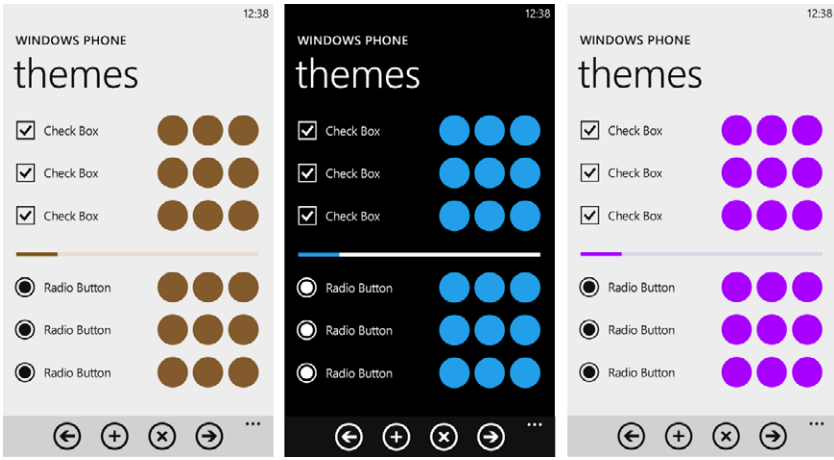


Fig 8.26. Windows Phone lets the user choose between light and dark themes and one color to highlight important elements.

each section. The header extends visual identity and at the same time gives the user a better understanding of the section he or she is visiting.

Default Colors

Android and Windows Phone use themes that affect the color of their apps and that can be light or dark. In Windows Phone, the choice of available themes is left to the user, who can modify the interface display of the entire OS.

All native elements of the apps are modified by this decision. Dialogues, buttons and list backgrounds will be visualized according to the chosen theme. However, in Windows Phone, respecting the user's choice is up to each app, because even when the user has chosen a color for his or her theme, the app can decide to use a different chromatic option. For example, even if the user chooses a dark color as the theme, the app can show elements in white.

Maintaining the user's theme selection has advantages and disadvantages that must be considered in design. For example,

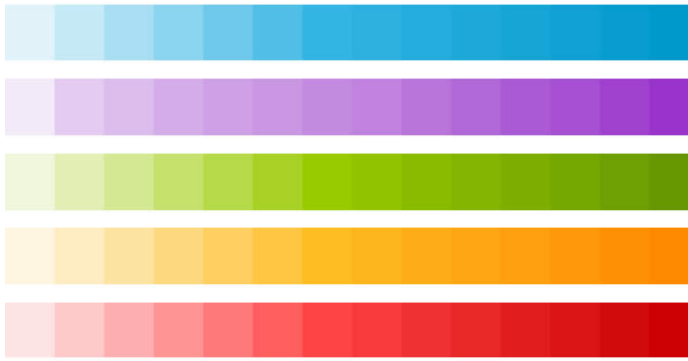
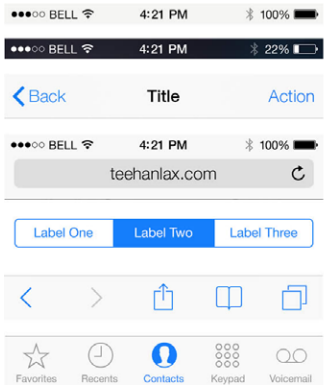


FIG 8.27. Android suggests a color palette, even though the main recommendation is blue.

using the color of the theme chosen provides visual consistency and comfort throughout the whole operating system. The user feels he or she is always in the same context. However, differentiation from other apps can be low, and it decreases the possibilities of incorporating corporate colors.

In Android, theme choice is not a user option. Designers are the ones in charge of deciding whether to apply dark or light themes to the whole app or only to certain screens.

In both cases, the colors of the themes are combined with other secondary colors, their function being to highlight elements that require attention. In Windows Phone, this secondary color can also be chosen by the user from a list of options in the



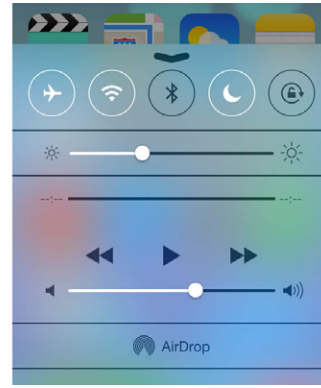


FIG 8.28. By default, iOS colors are based on black and blue hues.

preferences section, but again, the app doesn't have to abide. And even though the chromatic palette offers other options that can be considered when planning visual design, in Android, blue is recommended as the secondary color.

Because it has no themes, the iPhone presents a quite different case. Color proposal is native, established by default, and is based on clarity: white and grey for backgrounds, toolbars and section headings. These elements also have a certain transparency that displays what's behind. While black is used for informative elements, blue is used to highlight the font in buttons and form labels as well as other elements that should stand out (icons, selected tabs, controls). Even so, the colors of these elements can easily be changed.

Language: Mind Your Words

An app is full of places for text—in headers, titles, buttons, error messages and empty screen notices. In each case, words are as important as the graphic elements that contain or accompany them. Textual and visual languages go hand in hand to create a user experience that is consistent in all senses.

The Wrong Word

At the beginning, how a phrase is formulated is often overlooked, even though it has a direct influence on the way the user utilizes and relates to the app. For example, labeling a button incorrectly can lead to confusion as to the result of an indicated action, causing uncertainty and frustration for the user, who as a consequence makes a mistake.

Situations like this are frequent. Even though it may seem obvious, a message should be understood easily by the user

and require little to no interpretation. This can be achieved by using a simple and direct language.

The Receiver

As with any message, an app also has a receiver. Getting to know who the receiver is allows a deeper understanding of how to address him or her. This is essential in determining how to articulate phrases and the language used. It's not the same to talk to a child as it is talking to an adult, and you can't address a tech expert in the same way you can a novice. All of this influences word choice.

In the world of apps, you often come across products that use too much technical jargon. This results in messages full of technicalities or complex words that an unprepared user could find difficult to digest or that sound too strict. For example, the word "access" can be replaced by "enter," a more pleasant synonym.

Communicating Mistakes

The way of communicating mistakes deserves its own section. The scenario can prove stressful and unpleasant for users, so

Fig 8.29. Error messages in Letterpress are as fun as playing with the app, taking the drama out of the situation.

friendly messages are essential. Technicalities and accusations must be avoided. An example of this is the sense of humor that Letterpress has when communicating errors. Since it's an entertainment app, the team takes advantage of the user's disposition to have fun.

Texts in Other Languages

When an app is available in different languages, you've got to pay careful attention when incorporating translated text and ensure that it fits and displays correctly in the visual space assigned. Depending on the language, some words may be (a lot) longer than in the original language and would end up being cut off or hidden under other elements of the interface.

Visual Details

The difference is in the details. They can be what set an average app apart from a great one. At the start of the design and development process, attention to detail is minimal and often falls secondary to other aspects of design. But as the interface enters its final stages, it's important to consider all details and catch anything that could ruin all of the hard work performed.

Empty Screens

When designing, not only the ideal scenarios must be considered, like a list that is already completed or a full screen. It's also necessary to plan the design for when the app is starting to be used and there's still information missing. For example, contemplate how an empty screen will look without items and when there's still nothing to show.

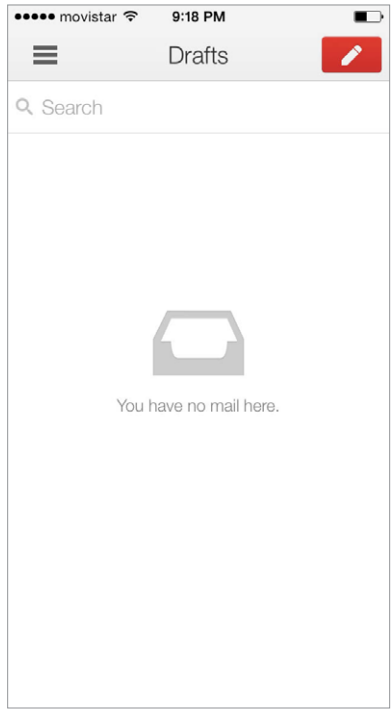


FIG 8.30. Empty screens also have to be designed for when no information is available. They are used to explain how a section functions.

Another situation that must be considered is what to do with the backgrounds that appear when lists don't cover the entire screen and with placeholders where the images would appear once loaded.

Ephemeral Graphics

The design of elements that appear on the screen for only a short amount of time should also be considered. For example, if the connection is slow, the user might have to spend more time than expected watching the load screen. This is where visual details experience their moment of glory.

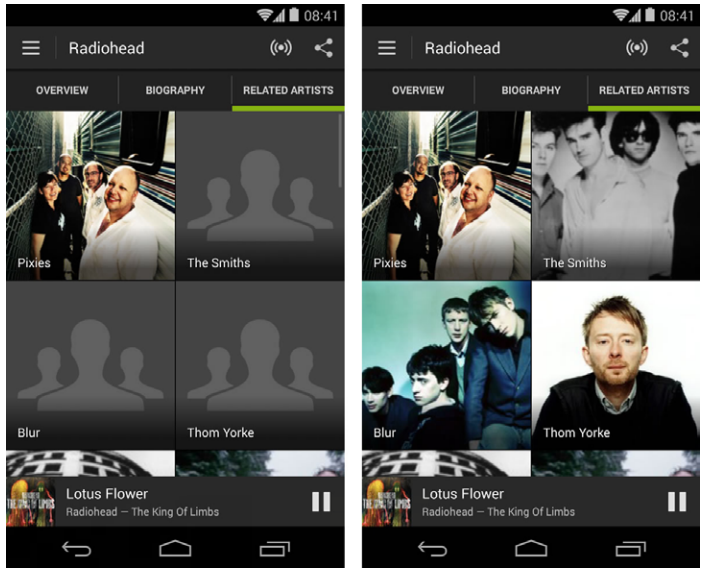


FIG 8.31. Elements may take longer to appear on screen in scenarios with low internet connection speeds – something that's important to consider.

Visual Secrets

If you want to get away with a few small luxuries, graphics that are hidden at first sight and are not easily accessed by users are ideal. Once discovered, they generate a special pleasure that is supported by complicity. For example, in the Google+ app[9], when swiping down to refresh content, a series of colored lines appears—a detail that's not visible until we perform this action.

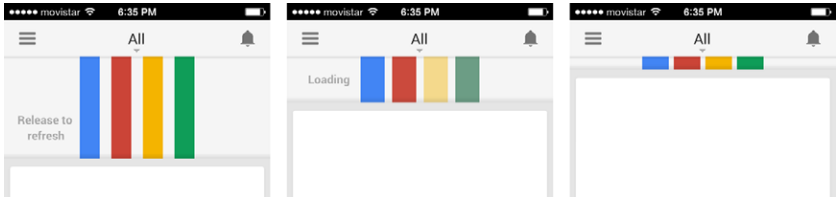


FIG 8.32. Google+ takes advantage of hidden spaces to add a touch of color.

9 http://www.google.com/mobile/+/

Animating the App

Animations are harder to design and render perceiving visual details more difficult. However, they also mark the difference in user experience[10], filling the app with life and making it more pleasant to use. At the same time, they accompany a main function or action. They are certainly not fundamental and could go without animation, but, without a doubt, they influence how a user feels when interacting.

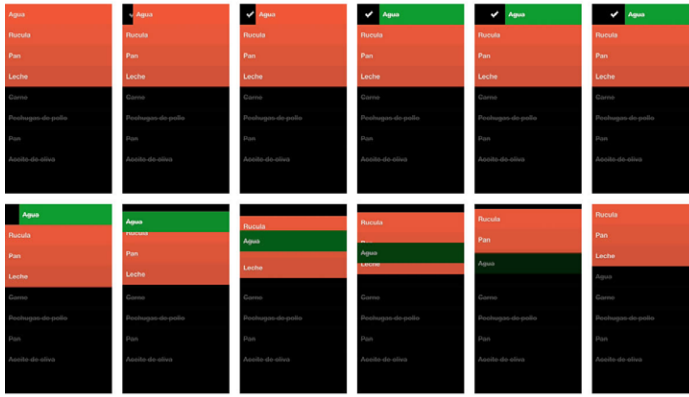


Fig 8.33. Clear uses animations as an indicator to let the user know what happens to tasks once completed.

According to their function and the way they are used, animations can serve a number of different purposes:

1. *Feedback*. On-screen graphics can be animated to show the result of an action. That would be the case of an element that disappears from the screen after an action is executed, showing where it goes and what happens to it.
2. *Transitions*. Animations can help make transitions between screens more fluid. This can represent a fluid navigation experience when going from one piece of content to the next.

10 Jonhson, Ben. Great animations make great apps. http://www.raizlabs.com/2012/09/great-animations-great-apps/

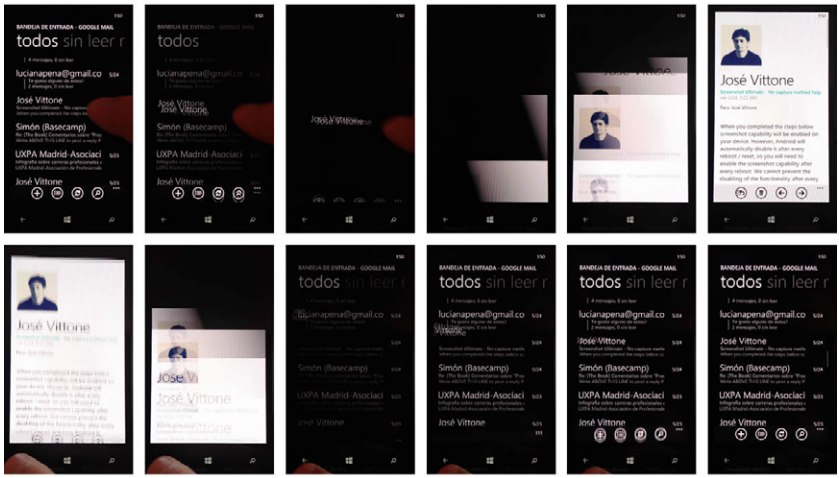


FIG 8.34. In Windows Phone, there are many transitions between screens that enrich the navigation experience.

3. *Informative tools.* Especially when the app is being used for the first time, it is important to outline how it is used and highlight the presence of a new component. In these cases, animation can help direct attention, which is, in fact, one of its main functions.
4. *Pure visual candy.* Sometimes, animation has no other function beyond catching the user's eye. It could be a simple detail that not everyone sees but that leaves a lasting impression when noted.

9

Dustin Mierau: Path and the Value of Details

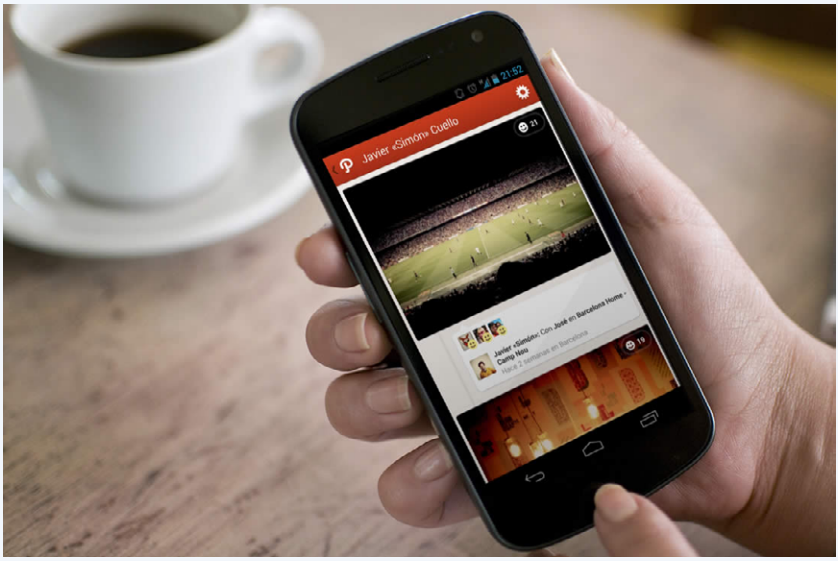


FIG 9.1.
The visual treatment of the interface of Path takes into account every inch of the elements that compose it.

Path is a mobile-focused social network with a personalized twist. It allows users to add a maximum of 150 people to their groups of friends and share moments, in many ways acting as a personal journal. And even though its popularity varies across markets, it has already crossed the 20-million-user barrier.

We've loved Path from the very start, especially its visual style and attention to detail in the interface—for example, the fact that the app's clock moves as you scroll the timeline, a notable improvement in user experience.

We interviewed Dustin Mierau, Co-founder and Chief of Design at Path, and asked him to explain a bit more about the importance and role of these elements.

How important do you think these details are to the general design of the interface?

> *It's definitely a balance. We have a lot of fun adding whimsy to our products, but we're careful to ensure*

that the product focuses on being useful first, intuitive second, and compelling third. The clock helped us remove clutter from the feed allowing us to show more stories on screen at the same time: useful. Less data-like text (e.g. timestamps) made what was on screen read more intuitively. And with an analog clock we were able to add the feeling of moving backwards and forwards through time, making Path's feed (hopefully) more compelling.

These kinds of ideas may seem difficult to transmit to teams, especially in early versions of an app, when the focus is more on functionality and less on visual elements that are often postponed.

How do you think the interaction between designers and programmers should be? Do you find it difficult to communicate concepts to developers?

Not difficult at all. There is trust between the teams at Path. More elaborate ideas need some convincing, and rightfully so, as they usually require a lot more effort and time to implement. Engineers and designers are at Path to build great products. So it's a collaboration really, and communication is frequent. How a product works is the result of a great collaboration between engineers and designers. The more respect there is between these teams the better the resulting product will work. You can usually sense a team's friction through their product.

Balancing the work of creating an interface with a well-finished appearance and developing a solid base that ensures an app's correct functioning can be tough, but they're not mutually exclusive.

Now that Path is becoming more complex, is it difficult to maintain functionality without sacrificing the user experience and vice versa?

It's true that Path has much more functionality now than it did two years ago. I think we're considerate of every piece of functionality we add to our products. There is much we haven't added because we haven't found a good way to incorporate it. A lot of functionality doesn't mean much if you can't figure out how to use it.

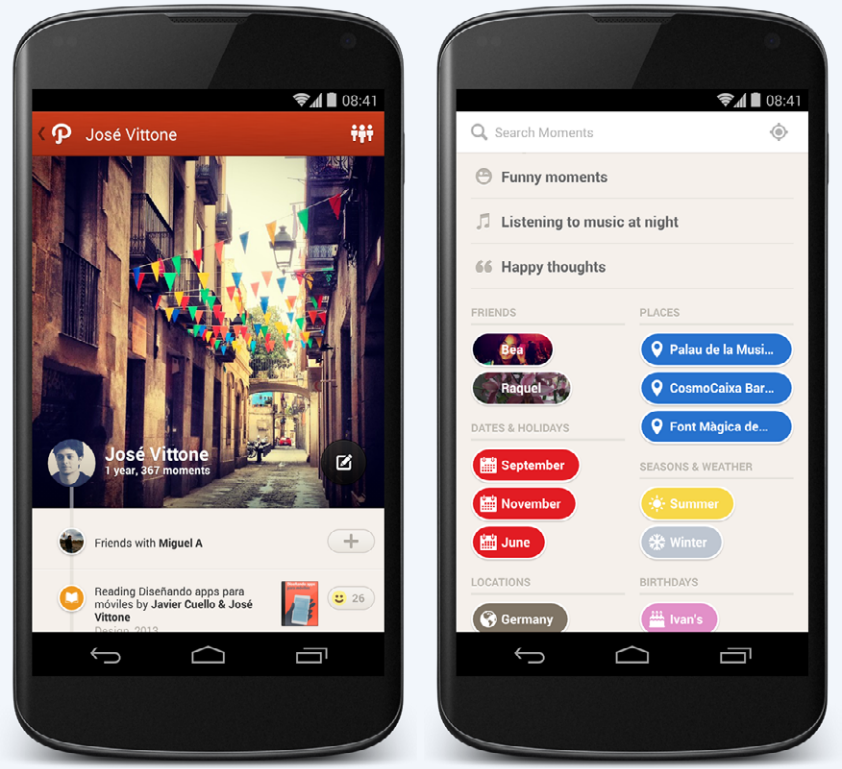


Fig 9.2. Each function added to the app has a reason for existing and is carefully evaluated to guarantee the best possible experience.

We put a great deal of effort into finding the right way to add features to a product. Features are added such that they work well on their own and, when combined with other product features, create an experience that is compelling, useful, and intuitive.

After its apps for iPhone and Android, Path launched an iPad version that, in comparison with the others, is relatively new and is on its way of defining its own personality.

The Path app for iPad has recycled some of the design patterns that were already present for iPhone, but others are new. What are the benefits of having so much space to play with?

It's not just more space we have to play with, it's an entirely different use case. When using Path on your iPhone or iPod Touch, you are most likely busy, at work, on a trip, outside, with friends, etc. When you're on your iPad, you're at home, at a cafe, at work, and probably sitting. Yes, you're in a better place to consume more and probably have time to interact more, but the type of content you're prepared to create is unique. With the iPad you're ready to share a collection of photos, write deeper thoughts, take time to edit videos, and simply be more creative.

We really haven't scratched the surface of what Path on iPad can be. I think the team is looking forward to taking advantage of the real estate the device offers to allow Path users to be much more creative with how they share their lives.

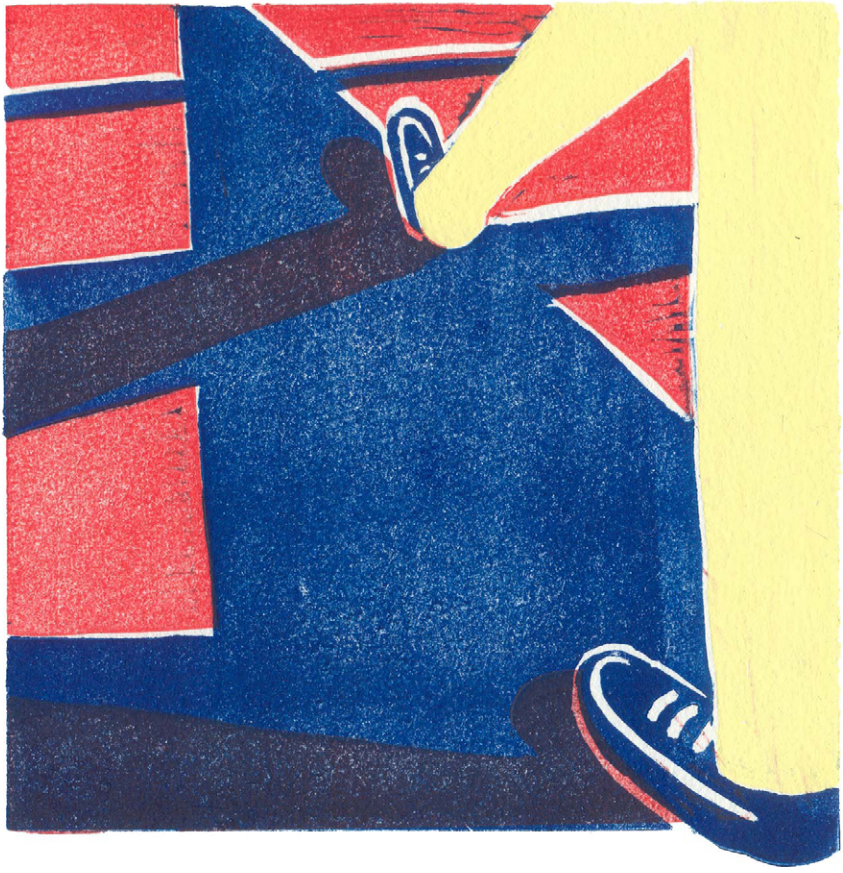
Without a doubt, this app is a good reference for designers and developers, who can use it as a solid example of how to make a high-quality product without neglecting function or design.

This is hard to achieve for many designers, especially when a project is just starting out and the road ahead is unclear.

We asked Dustin for some advice for new app designers who may be lost in the myriad of applications and operating systems out there:

> *Figure out a utility, strip away the nonsense, and make it yours. Absorb as much technical knowledge as you can. Be humble. Be curious. Respect engineers. Respect your users.*

Find more information on and download Path at www.path.com



10

Testing with Users

Usability tests are a fundamental tool for correcting and improving an app. They are carried out based on user observations: how they interact with the app, and how easily they find it to use.

What are Usability Tests?

As an app is being designed, teams must verify that it is easy to use and fulfills the purpose of the original idea—hence why it's essential to test with users.

Usability tests are performed to get valuable user feedback so that corrections and improvements may be made[1]. Tests are performed in labs or spaces designed specifically for this, where a user accomplishes tasks previously established by a moderator in charge of leading the test while a group of observers takes notes.

What These Tests are Not

Usability tests should not be confused with focus groups, which have a completely different objective in getting design opinions without taking into account utility.

When It's Time to Test

Ideally, tests should be performed before launching an app, and better yet, before programming the code. Testing in the early stages, and frequently, can help with new ideas before publishing, when there is still time to implement changes. The advantage of this is that it saves time and money and makes possible the inclusion of changes that can have a long-lasting impact on the project.

Although generally considered a one-time event, tests can be performed as often as necessary, even after an app has been launched. Tests allow us to continuously gather information and make adjustments accordingly.

1 ARMENGOL, Daniel. ¿Qué es un test de usabilidad? http://www.usolab.com/wl/2012/08/que-es-un-test-de-usabilidad.php

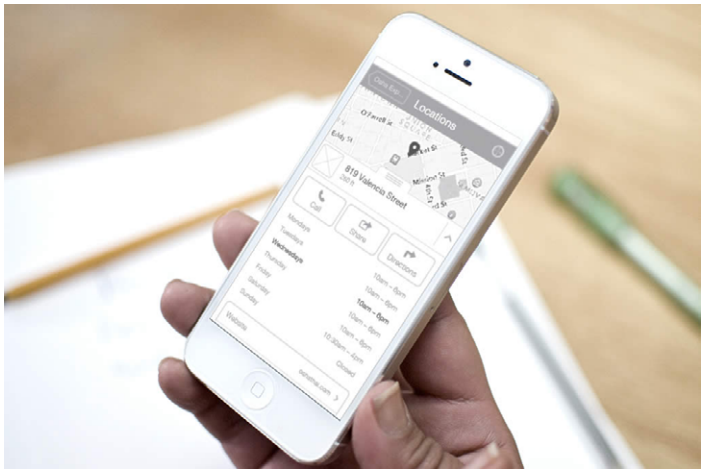


Fig 10.1. There's no need to have a finished app to perform tests with users. Ideally, tests should be carried out when there's a prototype.

Mobile Testing

When a usability test is performed with a mobile device, users don't necessarily have to test a completely finished app. As mentioned, it's advisable to do testing in the early stages of the design and development process, meaning that a prototype is often sufficient[2].

Mobile apps are related to a certain context of use, something difficult to replicate unless a field test is done. It's important to consider that using the app in a room is probably not the most realistic scenario, but it's clearly better than not performing a test at all. In such cases, use conditions should be simulated as closely as possible. For example, testing should be done with an average internet connection speed, not high-speed wireless[3].

The phone should also resemble reality as closely as possible. Ideally, that means a test should be performed directly on a user's mobile device. If that can't be arranged, a device

2 CASSANO, Jay. Secrets From Facebook's Mobile UX Testing Team. http://www.fastcolabs.com/3007979/open-company/secrets-facebooks-mobile-ux-testing-team

3 LANG, Tania. Eight Lessons in Mobile Usability Testing. http://uxmag.com/articles/eight-lessons-in-mobile-usability-testing

Fig 10.2. It's important to record gestures in tests. Mr. Tappy helps with this.

that is as similar as possible should be used. Before testing, the user should be provided a few guidelines as to how the device works so that the smartphone's characteristics don't interfere and the conclusions reached are related exclusively to the app. Likewise, if the app tested is, for example, for Android, the user should be a frequent user of that operating system.

When testing, you also need to pay attention to one element specific to mobile devices: touch gestures. For this, filming devices may be installed that, besides recording where and what the user taps, can also register what gestures are used to complete the actions[4].

Guerrilla Testing

Guerrilla tests are an agile and economic alternative to traditional usability tests, which can prove costly with the inclusion of specialized professionals, numerous users and expensive workspaces. With a more informal format and fewer users, gue-

4 http://www.mrtappy.com/

rilla tests can be quite efficient and are especially useful when a project's deadline is approaching[5].

A guerrilla test consists of gathering a determined number of users to try the app. The main objective is watching how they behave and getting information to correct mistakes[6].

How It Works

To perform the test, one member of the team should be designated as the leader responsible for the test. For this role, a great amount of patience is essential, as is the ability to listen.

Ideally, at least one other person also takes part in the guerrilla test to accompany the leader and observe. The observer will be in charge of taking notes on the usability problems that users encounter, without active participation.

The complete development of the test can be divided into three very distinct stages: planning, execution and analysis.

Planning

At this stage, the foundations are laid for the test before meeting with users. The goal is determined—such as testing the entire app or only some parts—and the guidelines needed to make the most of the test are defined.

At this stage of the process, participants are chosen. Ideally, five to eight participants should be selected—the amount of people needed to detect the most common usability issues of the app. Having more participants does not ensure that more problems will be found.

5 http://stamfordinteractive.com.au/conversations/benefits-of-guerrilla-testing/

6 Chisnell, Dana. Usability Testing Demystified. http://alistapart.com/article/usability-testing-demystified

Besides test users, the place where the test will be done has to be determined. Preferably, it should be a quiet place with no external distractions or interruptions.

Which of the two possible tests participants will carry out must be decided. The first alternative consists of simply showing the users the app and observing them to determine if they understand what it does and how it works. The second kind of test involves assigning them a specific task and observing them as they perform that task with the help of the app.

Based on what has been decided, a few questions can be defined to help obtain information about user interactions. These questions should be neutral and adapted to their levels of knowledge. It's best to break the ice with easy questions to avoid intimidating participants at the start.

Once all the decisions mentioned above have been made, a small pilot test can be performed with team members to get everything prepped.

Execution

Usability tests are performed with one participant at a time. Before beginning the test, do a brief introduction to explain the test and how long it will last. The volunteer should be invited to participate actively throughout the process and be encouraged to express all of his or her opinions, concerns and issues, even when they're not critical. It's important that the participant feels that his or her opinions are valuable and useful for the project, and that they will not affect the feelings of the team.

During the execution of the test, the moderator should be more aware of the user's behavior than of what he or she actually says. It's not uncommon for participants to unintentionally lie or try to please the moderator to avoid coming off poorly.

When the goal of the test is to perform a series of tasks with the app, the moderator should ask the user to perform them without any assistance. Behavior can't be biased or influenced. Orientation is only allowed in situations when it is absolutely

Fig 10.3. During test execution, the moderator and observers take notes on the behavior of volunteers.

necessary and as an opportunity to ask users what they think the expected result of an action may be.

The moderator might not be the only person that asks questions during the test. Participants may have doubts, and it should be clear to them that their questions will be answered once the test is over.

When this stage is finished, it's nice to give each participant a small reward for having completed the test. Even though it can be mentioned beforehand, it's recommended not to give them this final retribution before the test so as not to condition their answers.

Analysis

Once the tests are over, and shortly thereafter, the moderator and observers should meet to comment on the results of the test and review all notes taken.

This is the stage where problems are identified and solutions are proposed immediately. Once issues are rectified, another

test can be performed to determine how effective the solutions are and identify any potential new conflictive solutions that may have appeared.

Other Ways of Gathering Information

Guerrilla tests are not the only way of getting information from users. There are other ways that are faster—in fact, some require just a few seconds.

Dogfooding

There is an urban legend that says that because of a TV advertisement for dog food, a Microsoft executive sent an email to employees telling them to taste their own food—in other words, use their own products[7].

Since then, dogfooding has become a fast and easy way to test one's own software. It's as simple as using the app you're working on to gain a profound understanding of how it works.

Over time, dogfooding can become a standard practice for detecting technical and conceptual mistakes. These kinds of problems are often difficult to detect in design or code programming and easier to pinpoint when abstracted and experienced in a more relaxed environment.

Yes, dogfooding is easy and comfortable, but it doesn't replace user testing. More than anything, it's useful for gaining confidence in your own work.

7 El-Qudsi, Ismael. ¿Qué es el dogfooding? http://www.elqudsi.com/articulos/%C2%BFque-es-el-dogfooding/

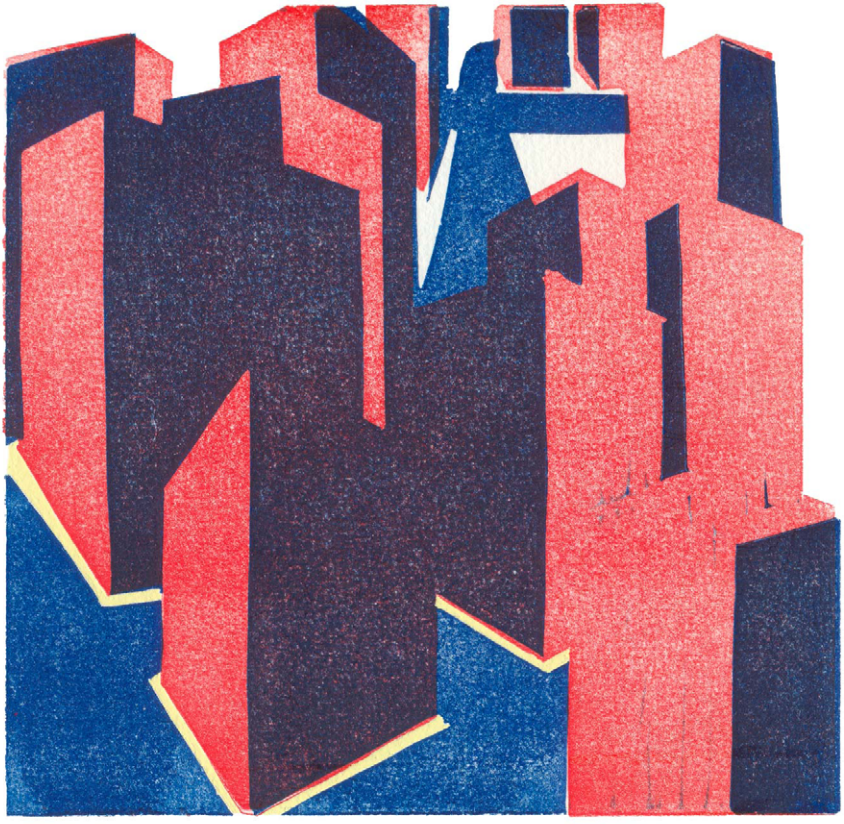
The Five-Second Test

The first impression of a design provides a lot of valuable information, especially with respect to content hierarchies. The five-second test is, essentially, explaining a concrete situation to a user and showing him or her the app's design for only five seconds immediately after. Next, the participant is asked to speak about everything he or she remembers seeing.

User reactions after a quick glance help to establish what is more eye-catching. It also helps one to determine whether users' impressions are in line with what the team has originally conceived.

There are a number of tools available online for these types of tests, such as the website FiveSecondTest[8].

8 http://www.fivesecondtest.com/



11

Preparing Files

Once the interface design has been defined, the next, and perhaps more tedious, step is to separate and prepare the files for the developer. Doing this correctly will ensure the accurate implementation of the design.

Understanding Technological Differences

There are countless mobile devices on the market, each with different hardware characteristics defined by the manufacturer. Those that most affect design are related to screen quality, since they will influence interface design planning and, later, the images that need to be prepared for the developer.

Screen Size

A determining factor in mobile device screens is their size: the distance, measured diagonally and in inches, from one end to the other.

In this respect, Android is a very fragmented operating system, because manufacturers offer a great number of models with screen sizes that change from one telephone to another without much coherence. This is a consequence of being an open operating system available in many devices. To bring some order to this chaos, Android has decided to group the different screen sizes in four main categories: small, normal, large and extra-large.

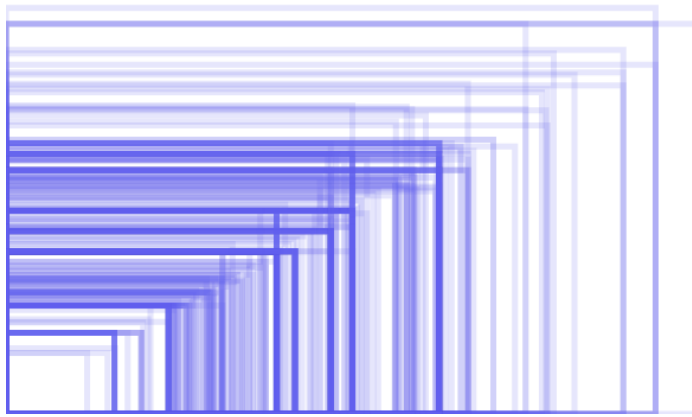


Fig 11.1. In Android, there is immense fragmentation because of the numerous screen sizes each smartphone manufacturer provides.

In the iOS world, things are a bit simpler. iPads, iPad Minis and iPhones have screen specifications that don't represent a problem, since they are all under Apple's control.

To date, there are few manufacturers that provide the Windows Phone operating system, and the screen sizes are limited to three types, with slight aspect ratio changes between them: WVGA (15:9), WXGA (15:9) and 720p (16:9).

Screen Density

Besides size, another factor to consider is screen density: the quantity of pixels in a certain physical space measured in dots per inch (DPI), the dot being equivalent to the pixel. Screen density can be understood along the same lines as population density—the relation of inhabitants per square kilometer. In the case of apps, the inhabitants would be the pixels, and the screen would be the surface. A higher screen density indicates that there are more pixels available, and, therefore, improved quality in image definition and other elements that integrate the interface.

Density influences the designer's work because it determines the characteristics of the document he or she will use to start work and the number of images that will have to be produced when the design is finished. The more densities supported by an OS, the more images needed.

In Android, there's an enormous variety of densities that cover mobile phones and tablets, offering a wide selection of visualization qualities. They are most often grouped as low, medium, high and extra-high[1].

iOS has only two densities: retina and non-retina. The first is the newest and is used in almost all recent devices, including

1 http://developer.android.com/guide/practices/screens_support.html

Fig 11.2. An image has to be designed for each density in which the app will work. An edit icon, for example, is designed for all Android densities.

the iPad Mini. Its visual characteristics are double the per-inch pixel quantity of non-retina options.

Lastly, Windows Phone separates densities in medium, high and HD, depending on the device.

Preparing Design Files

When designing an interface, the first step is a white canvas, and it is here that we encounter the first doubt. What is the ideal workspace size?

Android recommends starting design with a document prepared for a standard base—a normal screen size with medium density—and then, when the time comes, separating the images for the developer. They should be scaled for higher or lower densities.

For iOS, it is most convenient to design for retina, the highest density. In Windows Phone, it's best to design for the lowest density available, WVGA, and scale from there to the other two.

Luckily, designing from scratch is not always the norm. It's possible to find design templates online for Android[2], iOS[3] and Windows Phone[4], which contain interface elements that can be used to start with a base of visual components like buttons, bars

2 http://developer.android.com/design/downloads/index.html
3 http://www.teehanlax.com/tools/ios7/
4 http://msdn.microsoft.com/library/windowsphone/design/ff637515(v=vs.105).aspx

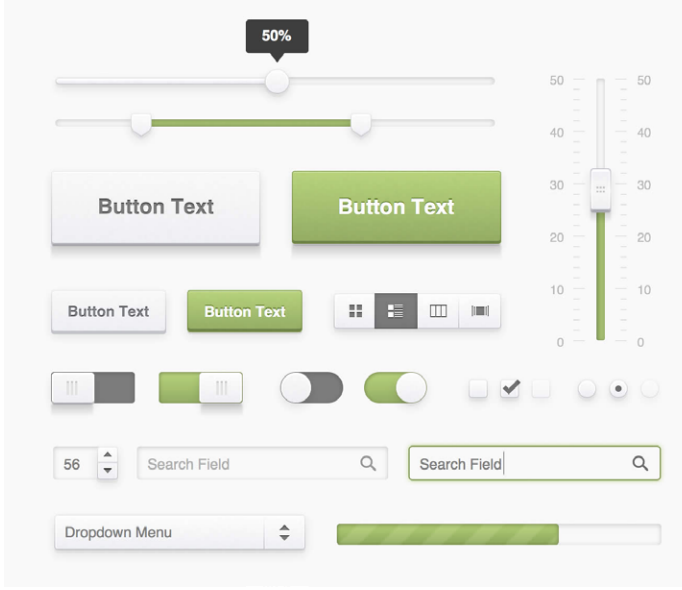


FIG 11.3. Some interface component templates can be used to avoid designing everything from scratch.

and backgrounds. Most of these are vector graphics, meaning they're easier to scale to different sizes.

Designing and Testing in a Mobile Device

An interface can look really great on a computer screen but become an entirely different beast on a mobile device, its ultimate destination. As work is being done, it's best to consistently test the interface on a smartphone to assess visualization. Don't wait until after the app is implemented and coded.

Getting used to performing tests and incorporating them into the design process will allow you to ensure that the size of the elements and fonts are correct and that sufficient contrast is present.

There is a wide array of tools that allow testing directly from a computer connected to a smartphone. For Android, there's

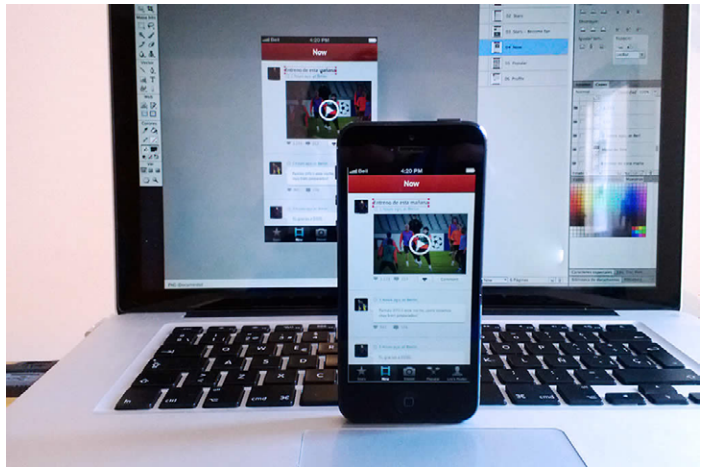


Fig 11.4. With LiveView, you can verify what the interface design looks like on a smartphone while you work.

Android Design Preview[5], and for iOS, you've got LiveView[6]. There aren't any comparable tools available for Windows Phone just yet.

Image Slicing

Once the design is ready, the images included in the interface must be separated. Save each element in a separate file so that the developer can access them easily and begin bringing the app to life.

An Image for Each Density

When images are separated, different available densities should be considered, and a version for each should be created.

5 https://code.google.com/p/android-ui-utils/
6 http://www.zambetti.com/projects/liveview/

If the most common densities are covered, four different images for each graphic element should be created for Android. If the design has been created on a standard base, it will be necessary to do a few calculations in which the width and height of the images are multiplied so that they look good across densities.

To facilitate an easier design process, Android works with density independent pixels, also known as DP. Each DP is the equivalent of `1px` in a medium-density screen and can be transferred to other densities by calculating the value for each case. For example, if we have an icon with a width and height of `20dp`, it will have `20px` in medium density, `15px` in low density (`20dp` multiplied by 0.75), `30px` in high density (`20dp` multiplied by 1.5) and, finally, `40px` in extra-high density (`20dp` multiplied by 2).

There's no need to get entangled in calculations, work with a giant calculator or go back to school to become a rocket scientist. There are tools available online that can perform these calculations for you[7].

In iOS, the math is much easier: the images designed in retina should be transformed to half their size. As a trade secret, it's important to consider that the width and height of the graphics designed for retina should be even numbers so that it's easier to divide them by two later on.

In Windows Phone, images are scaled as vectors with .svg format. Therefore, they can be scaled by code to the necessary size. This method also ensures the optimization of images for Windows 8 apps, guaranteeing visual consistency.

When scaling images, make sure they maintain their quality and aren't deformed or blurred. Sometimes, the imperfections found in a new size have to be adjusted until a sharp and well-defined image is achieved, in other words, pixel perfect.

7 http://coh.io/adpi

Buttons

When it comes to buttons, it's not a good idea to work with images that contain the complete shape of the button in its final size. Instead, it's preferable to have images that can be enlarged to cover different widths and lengths without becoming deformed. With this method, it's possible to use the same image for buttons of different sizes throughout the entire app. As a result, few images require redesign and export with every change.

To improve work and gain control over the areas of an image to be enlarged, Android has made available a tool called Draw 9-patch[8]. The final result is an image that, after implementa-

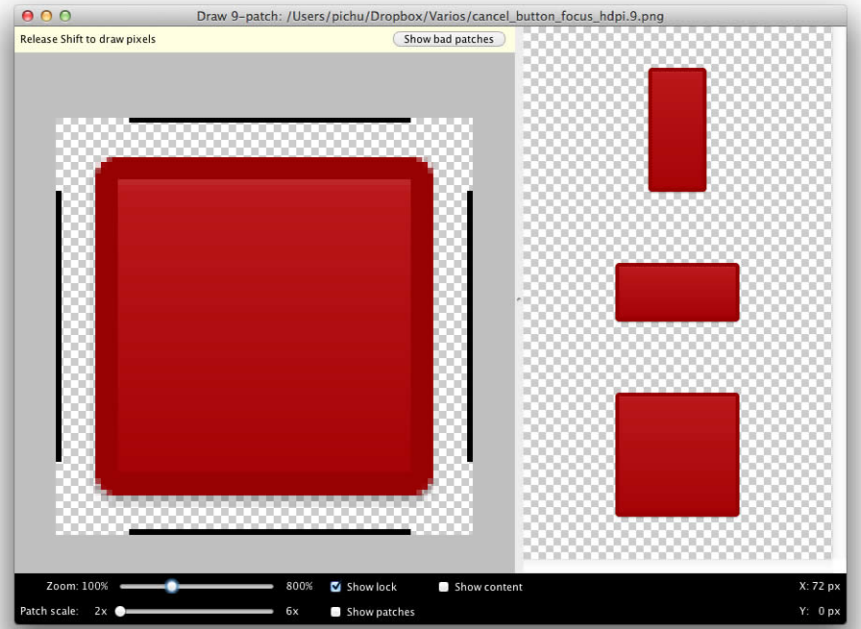


FIG 11.5. With Draw 9-Patch, it's possible to divide the image of the button in areas and determine how it will scale when stretched.

8 http://developer.android.com/tools/help/draw9patch.html

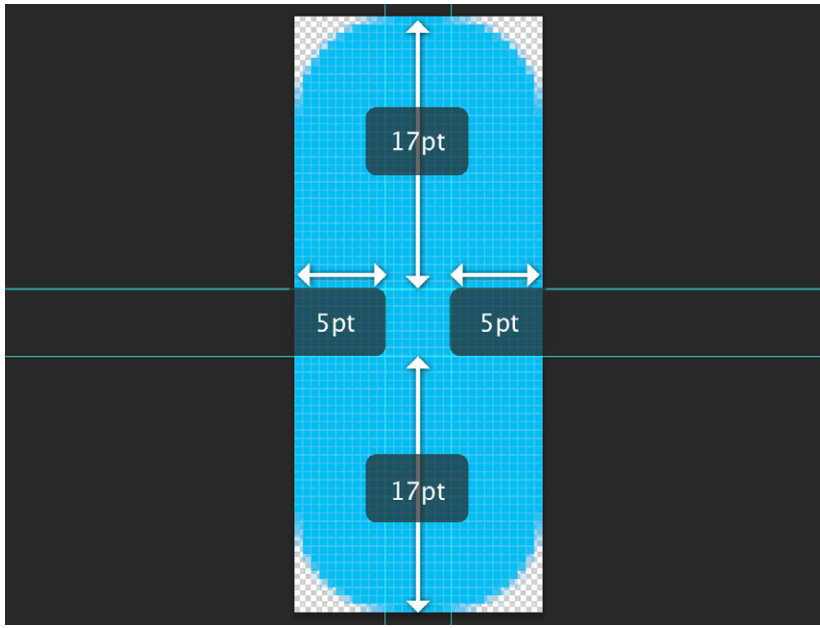


Fig 11.6. In iOS, button images can be designed anticipating the way the button will grow, so it's not necessary to generate images for each different size.

tion, will detect which areas can be enlarged and which should remain unchanged.

In iOS, the idea is similar, only there's no need for an external tool. In a button with rounded corners, a security area can be defined that won't enlarge, and the center of the button can be used to cover the height and width necessary. These values can be agreed upon with the developer and should be entered in the code when the app is being programmed[9].

It's worth keeping in mind that buttons don't have only one state. In general, design is performed in the normal state, but buttons can also be tapped or disabled, and there's no hover state, as is the case with web buttons. For each situation, a visual variation of the image is required to highlight any differences.

9 http://robots.thoughtbot.com/post/33427366406/designing-for-ios-taming-uibutton

Backgrounds

Backgrounds are not necessarily images with the same size of the screen for which they are designed. For example, a background with a texture or pattern can be the image of a module in the size needed that is then repeated by code as many times as necessary.

In the case of gradients, an image with the necessary height and width can be repeated several times. So, if there's a vertical gradient, an image with the height of the screen and 1px of width can be exported and repeated horizontally until it covers all the available space.

In Windows Phone, a particular situation occurs: designs that are based in Panorama have a format, especially horizontal, that includes several screens[10].

Tabs and Bar Icons

Before creating new ones, default icons that come with the operating system should be reviewed.

With iOS, the Apple Symbols font includes the icons that are most commonly used in the apps for iPhone or iPad. And

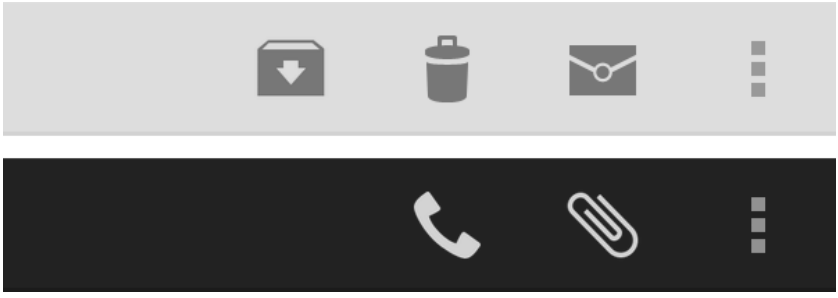


Fig 11.7. Depending on the theme of the app, in Android, icons have to be designed in light or dark colors so that they have enough contrast with the background.

10 Panorama control design guidelines for Windows Phone. http://msdn.microsoft.com/en-us/library/windowsphone/design/hh202912(v=vs.105).aspx

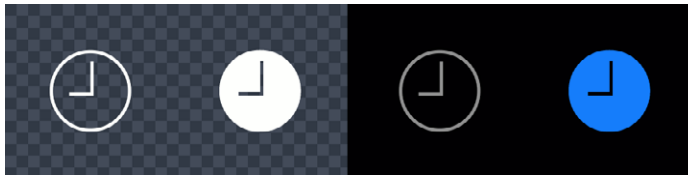


Fig 11.8. iOS adds predefined effects to the original image of the icon – white and with a transparent background – depending on whether or not it's selected.

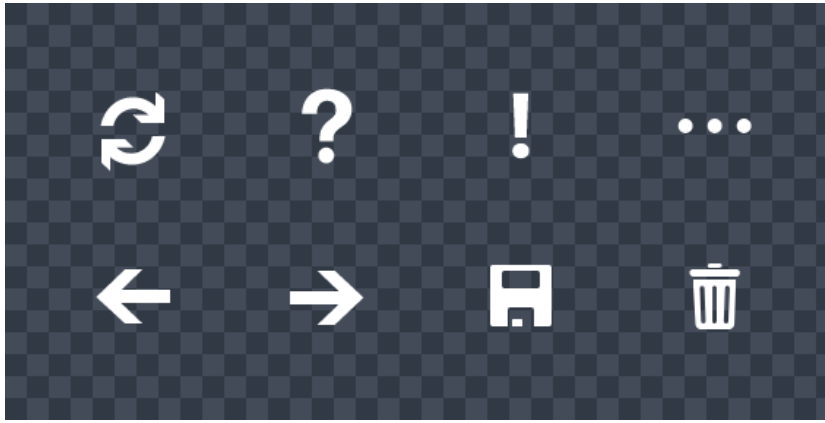
in the case of Android, you may download a pack of official icons[11]. Windows Phone also has a series of predesigned icons[12].

Whenever you need to create your own icons, you cannot lose sight of the visual style of the operating system. Maintaining consistency means the user won't realize which icons have been designed especially for the app and which have not.

In Android, images for menu icons, tabs and action bars must be designed with the necessary contrast according to the theme (dark or light) that was chosen to develop the app: `.png` images with a transparent background.

Fig 11.9. In Windows Phone, icons have simple, flat shapes.

Dialog icons must follow the graphic style of colors and shades recommended by Android in its guide[13], and tabs must



11 http://developer.android.com/downloads/design/Android_Design_Icons_20120814.zip
12 App bar icon buttons for Windows Phone. http://msdn.microsoft.com/en-us/library/windowsphone/develop/ff431806(v=vs.105).aspx
13 http://developer.android.com/design/style/iconography.html

include the two states—normal and selected. There is a template available from Android that offers a guide and imitates visual style in each case.

In iOS, icons have effects that are applied to the toolbar, navigation and tabs. All of those images are used as masks. Therefore, they should be white with anti-aliasing and a transparent background. It's also important not to include shadows, and if bevels are used, they must be at a 90º angle, as if the icon were being illuminated by a beam on the upper part.

Default visual effects of the operating systems can be overwritten if unwanted, as long as the images to replace them are provided.

In the case of Windows Phone, new images created for the icons usually follow the aesthetic the operating system proposes: minimalistic, flat shapes, without gradients or bevels. It's best to choose white, since they will have the color of the theme chosen by the user. In consequence, the resulting file will be an easily scalable `.svg`.

File Naming

When working with many different images, especially for different resolutions, it's important to be organized and consistent in naming exported files.

Working orderly is a huge advantage when you're searching for images inside a project. The name should be enough to identify the function and context of the specific image.

In Android, a different folder for each density is recommended. Images will be saved there, under the same name, for each case.

As an additional help to folders, the images can be named with prefixes. For example, the icons can begin with `ic_`. Combinations of prefixes can even be used for more clarity on the

image type. A graphic called `ic_launcher_calendar.png` will give you a pretty good idea about what it does and where it goes.

Exported images for iOS also follow a special naming convention, depending on whether they are for devices with or without retina. In the first case, `@2x` should be included after the name. For example, we would have the image `calendar@2x.png` for retina and calendar.png for non-retina. If the file is for iPad, the suffix `~ipad` can be added at the end to differentiate it from iPhone images.

In Windows Phone, if scalable .svg images are created, it's not necessary to have different versions. However, when working with `.png` images, a suffix should be added at the end to indicate the density of the device, such as `_VGA`, `_WXVGA` and `_720p`.

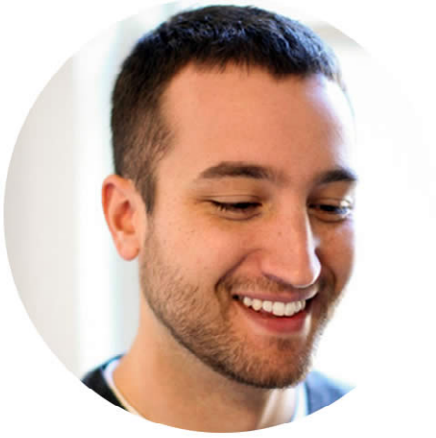
Communication

In order to ensure correct implementation, the designer and the developer must communicate constantly and fluently.

The developer may have doubts about the sizes of graphics and fonts, margins, the use of the grid, missing images or, simply, things the designer has overlooked or hasn't considered. To reduce uncertainties, especially when working remotely, the designer should prepare a series of visual documents that facilitate interpretation.

In the document, screens of the app can be displayed, preferably those that are most complex or that mark important differences and changes. By means of lines and markings, the different sizes of the fonts and spaces in between visual elements may then be indicated.

Documents and interpretations aside, questions and concerns are likely to arise. Designers and developers must work together to find the best solutions.



12

Loren Brichter: Thinking Outside the Box

In the world of app design and development, there are few pacesetters. Not many people have broken the rules to solve problems in new, unconventional ways and proposed really inventive concepts. Loren Brichter, called the "high priest of app design" by the Wall Street Journal[1], has transformed the industry.

Loren has built an enviable resume before even hitting the age of 30. A few years ago at Apple, he helped design the software that was installed in the first iPhone. He told us about his participation in the project:

> *My job was very deep in the software stack, working on low level graphics code that enabled higher levels UI mechanisms to function. It was an honor being part of the project.*

Apple isn't the only big name he's worked for. He also spent time at Twitter, having caught the microblogging network's attention with a personal project, Tweetie.

Currently, Brichter works out of Philadelphia with his own company, Atebits, away from technology epicenter Silicon Valley.

You have worked in big companies and also on your own, at Atebits. What have you learned from this experience?

> *I worked for two incredible companies (Apple and Twitter), but I think I enjoy working for myself the most. It requires a different kind of discipline, and I certainly miss some of the resources that big companies can provide, but it's exciting working on something that you have complete freedom with!*

1 Lessin, Jessica E. High Priest of App Design, at Home in Philly. http://online.wsj.com/article/SB10001424127887324392804578358730990873670.html

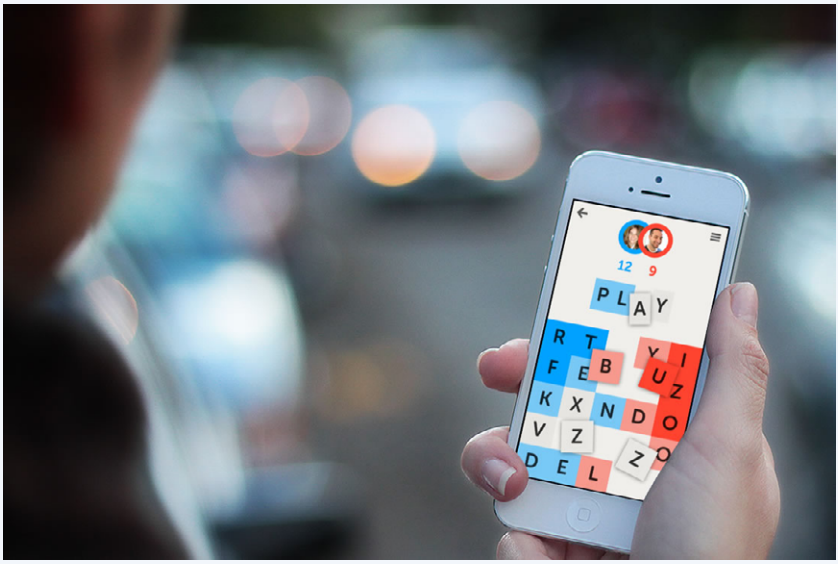


Fig 12.1. The visual style of Letterpress is characterized by its clean finish, letting the most important elements stand out.

The last app designed by Loren is Letterpress, an online game that enables players to use random letters to form words and challenge opponents everywhere in the world. The app has been so successful that Apple has used it in mini iPad advertisements.

This success is due in great part to user experience. Letterpress generates a healthy addiction in users—that famous amount of engagement that most apps aim to achieve. The interface is quite clean, with a white base that renders letter blocks the authentic stars of the game.

When Letterpress launched, its interface design strayed significantly from iOS, in which bevels, shades and textures prevailed. Could this new, more independent, abstract style be a trend?

> *The market is relatively young and we're starting to see some conventions emerge, but innovation is still popping up all over the place. I'm trying to distill good*

patterns and extrapolate… hopefully discovering new, interesting ways to interact with these devices.

We are certain that you have a lot of ideas for new apps in your head. How do you realize when one of them is mature enough to become a project?

Maybe it's like art versus porn, you just know it when you see it. There are degrees of maturity, I had a few things that had the potential to be viable products, but Letterpress seemed to have a good set of attributes that would make it fun to push out into the world at the time.

Two important considerations are: will other people find it fun and useful? Can it be finished? It's very hard to draw a line at "1.0" — it's important that an idea can make it to that point.

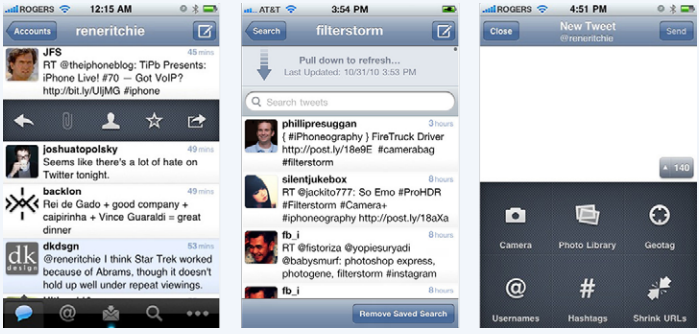


FIG 12.2. Tweetie incorporated several interaction mechanisms invented by Brichter that other apps began to imitate.

Loren Brichter is also the creator of *pull to refresh*, a feature that has expanded and is used in many apps, including Facebook, Path and Pinterest. The same happened with cell swipe, which displays more options and side panels.

There are a number of other interaction mechanisms out there that, while not one of Loren's creations, are consistently and constantly used, an example being the hamburger button for menu navigation.

> *From a high-level perspective it's fun to see ideas like this spread and evolve across the ecosystem. Frankly, I'm a fan of stacked panel navigation… it makes an abstract concept real and physical. I'm sure we'll see lots more experiments along those lines, designs that take advantage of our innate expectations of physical reality to make interfaces more intuitive.*

Loren is a restless individual. He's constantly brainstorming and coming up with new ideas that will continue to disrupt the industry.

Learn more about Letterpress at www.atebits.com

13

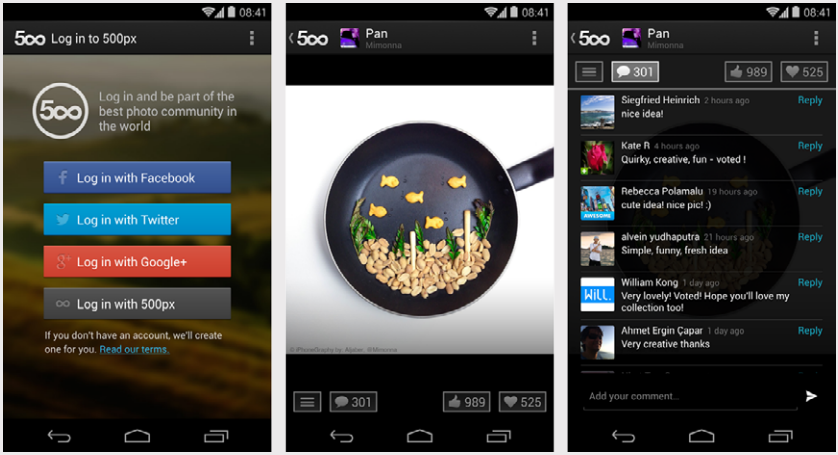
Design Best Practices

There’s a lot to learn from those who have already paved the way. We’ve selected three apps from each principal operating system that, for us, got it right.

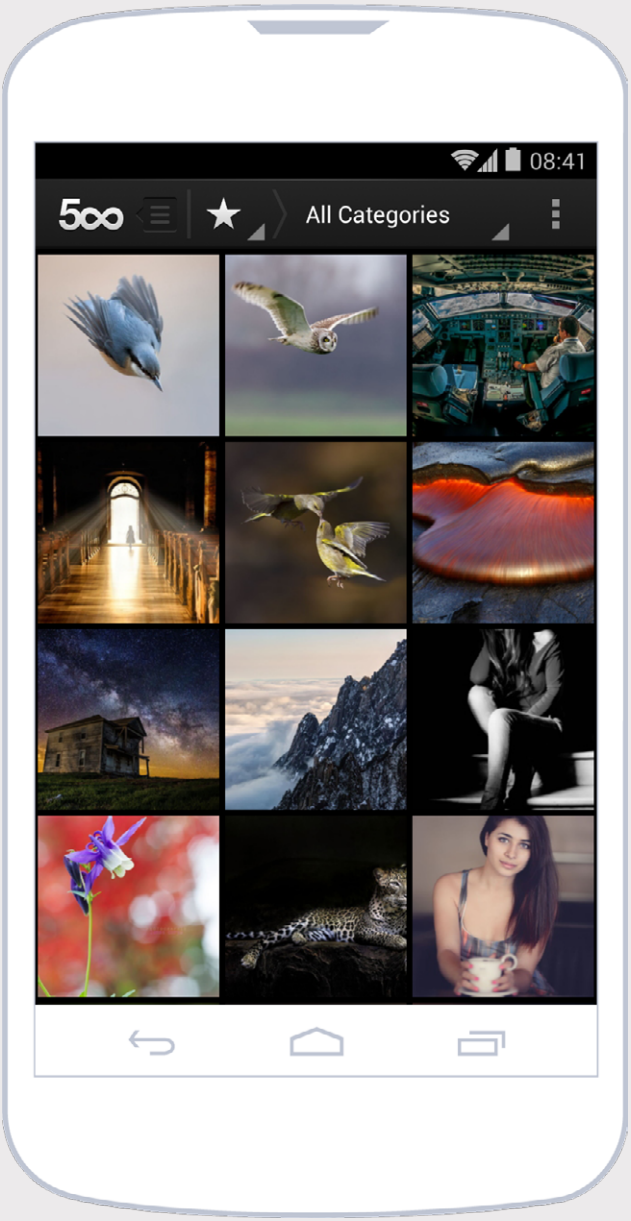
Android

500px[1]

This app for photographers nailed it. It offers multiple and clear login options and has successfully translated the platform's explorative quality. Visually exceptional, 500px uses dark colors for controls that help to enhance pictures.



1 https://play.google.com/store/apps/details?id=com.fivehundredpx.viewer

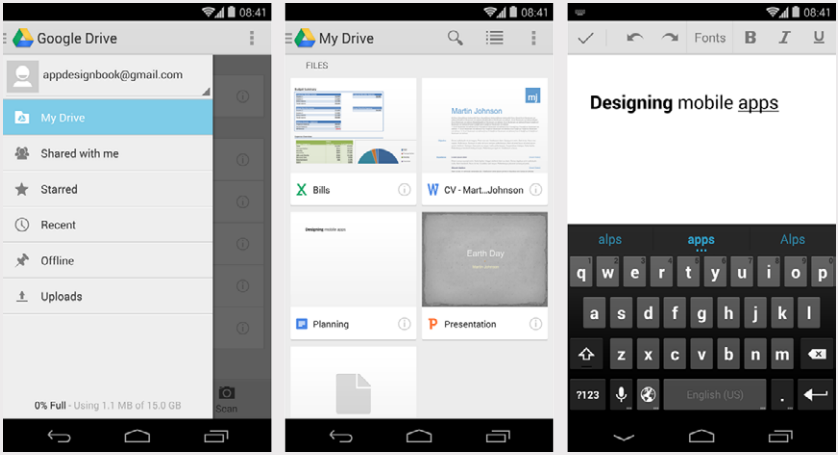


← *Fig 13.1. Login, picture details and comments screens.*

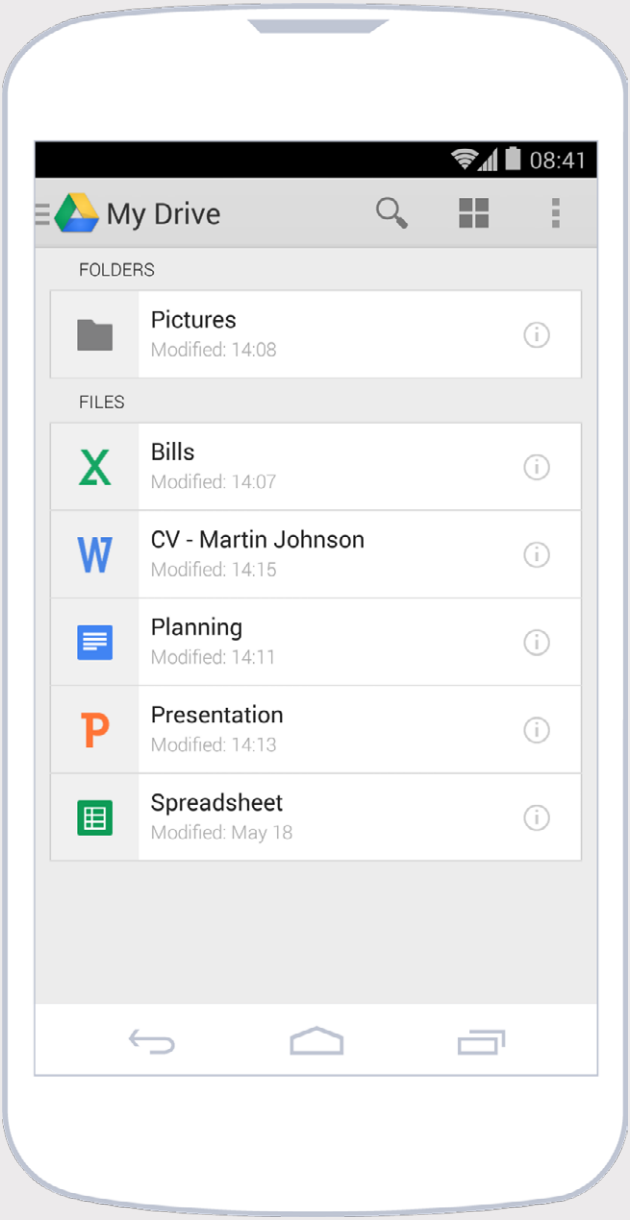
Fig 13.2. Categories screen.

Google Drive[2]

This cloud-based storage service and document collaboration tool keeps its main features at the center while at the same time including editing elements as well. It is ideal for users who change frequently between their desktop computers and mobile devices.



2 https://play.google.com/store/apps/details?id=com.google.android.apps.docs

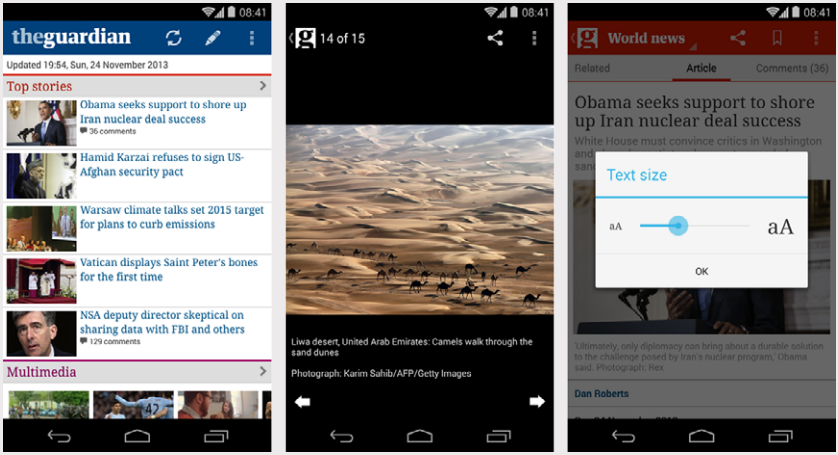


← *Fig 13.3. Drawer menu, file preview and edit.*

Fig 13.4. Documents list.

The Guardian[3]

The English newspaper makes an extensive use of native elements and, at the same time, keeps its own corporate identity, using headers as the strongest element of communication. Likewise, it establishes clear hierarchies for reading texts and contemplates accessibility features, such as zooming in on fonts.



3 https://play.google.com/store/apps/details?id=com.guardian

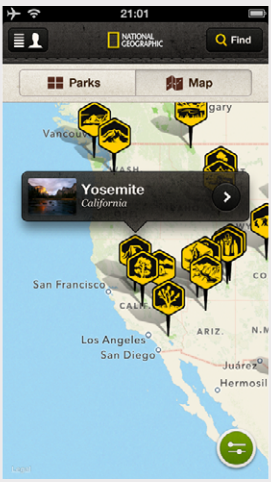
← *Fig 13.5. Main news screen, photo detail and text size configuration.*

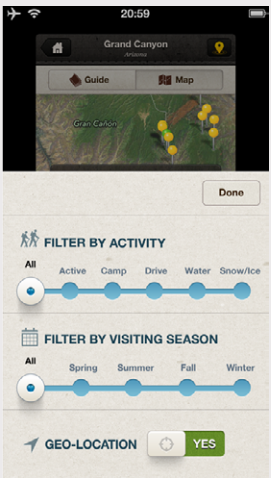
Fig 13.6. News detail.

iOS

National Parks[4]

On both the iPhone and the iPad, the attention to detail evident in this app from National Geographic is absolutely astonishing. The different screens enable the appreciation of the fair use of textures with transitions that add value to the concept. Layout hierarchies are very clear, thus favoring the interpretation of content.





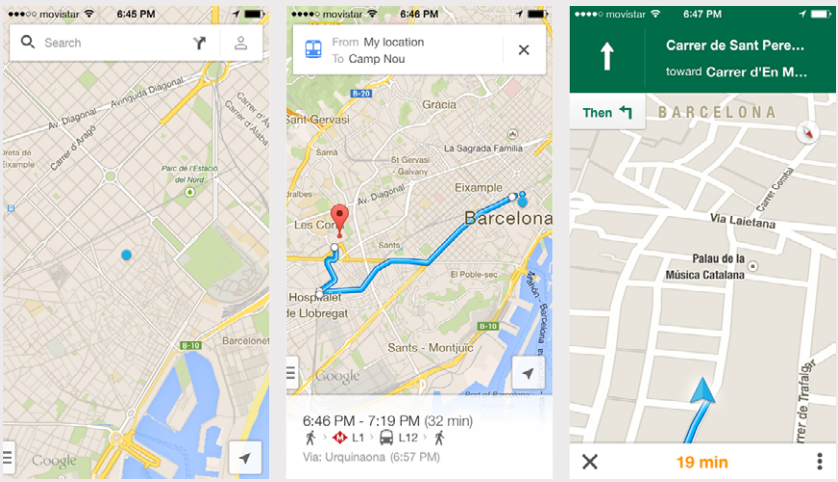
4 https://itunes.apple.com/us/app/national-parks-by-national/id518426085?mt=8&uo=4

← *FIG 13.7.*
Parks seen as photographs and as locations on a map, and filtering options.

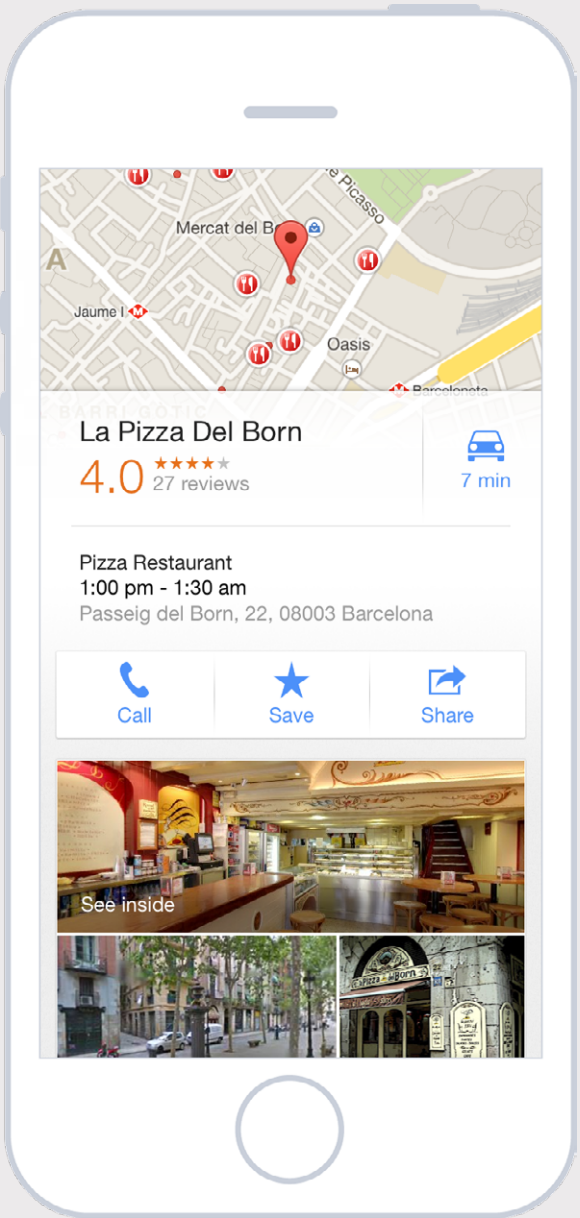
FIG 13.8.
Park detail.

Google Maps[5]

This highly anticipated iPhone app has not disappointed. Since its launch, Google has demonstrated that it sets its own rules wherever it goes. The app makes a conscious use of space; each screen displays elements in direct relation to current needs, hiding and highlighting only what's necessary. It is simple and robust at the same time.



5 https://itunes.apple.com/us/app/google-maps/id585027354

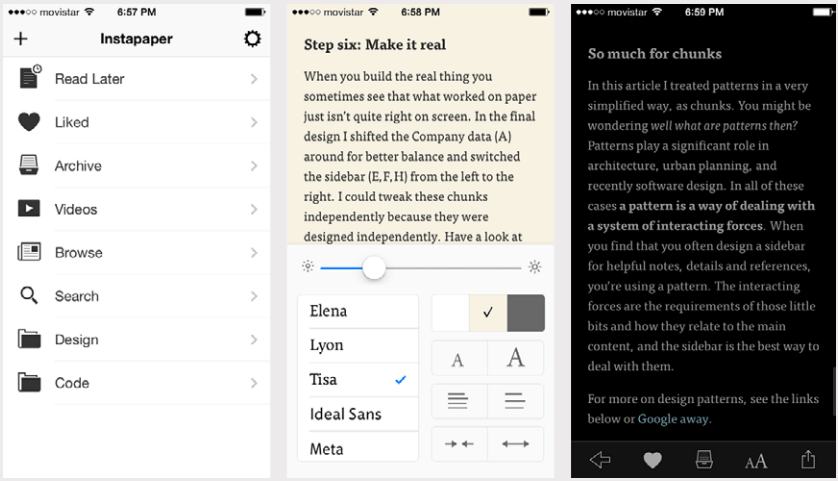


← *FIG 13.9.*
Main search screen, results with road and navigation screen.

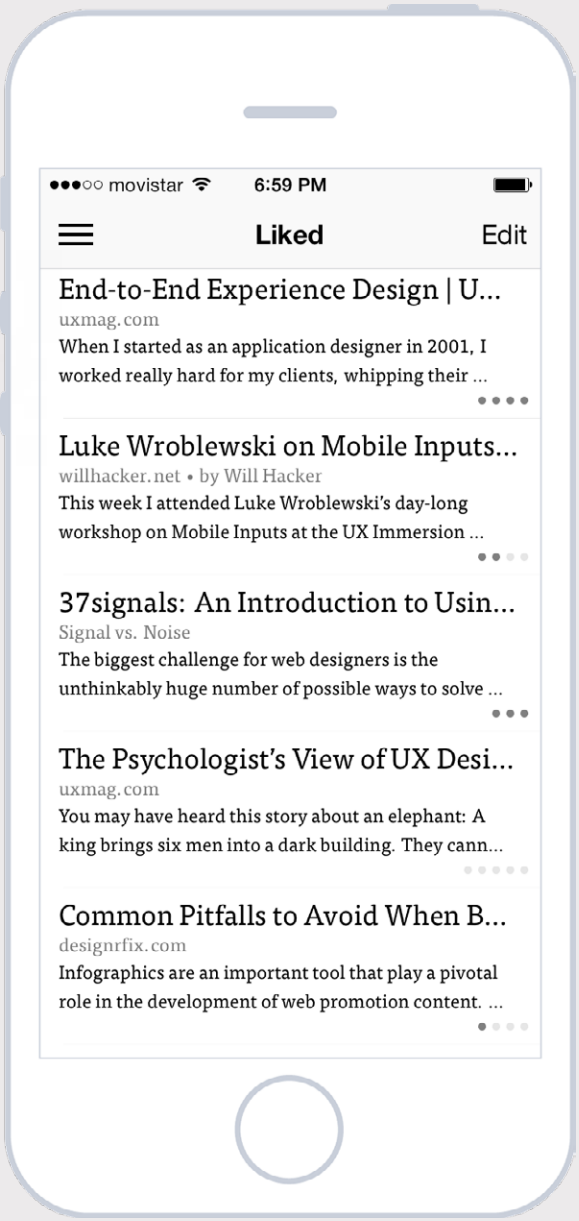
FIG 13.10.
Detail of the result with options.

Instapaper[6]

One of the pioneers of the read it later concept, Marco Arment's text reading app is super polished, and everything revolves around the main idea: making reading easy. Flexible at every moment, it is capable of hiding unnecessary controls and switching to night mode to adapt to low light conditions.



6 https://itunes.apple.com/us/app/instapaper/id288545208



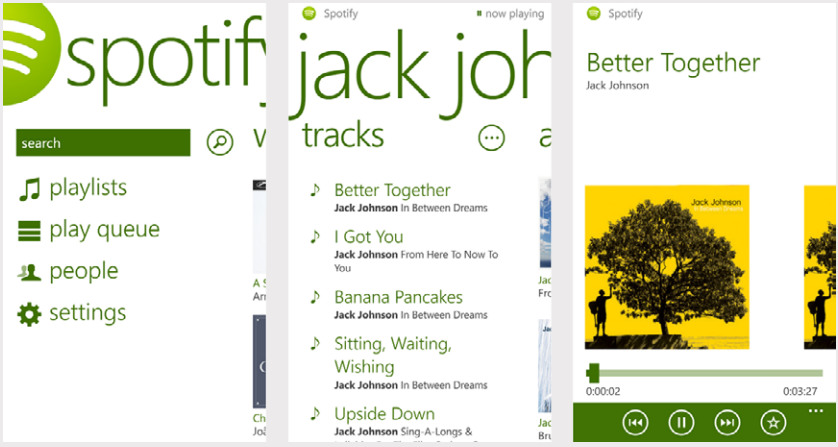
← *FIG 13.11. Main menu, screen with reading options and night reading mode.*

FIG 13.12. Items marked as favorites.

Windows Phone

Spotify[7]

The online music service has found a great way to adapt to the Windows Phone proposal. It uses branding on patterns like Panorama and Pivot Control in a way that only a handful other apps can. It has an excellent balance of features, branding and consistency with the operating system.



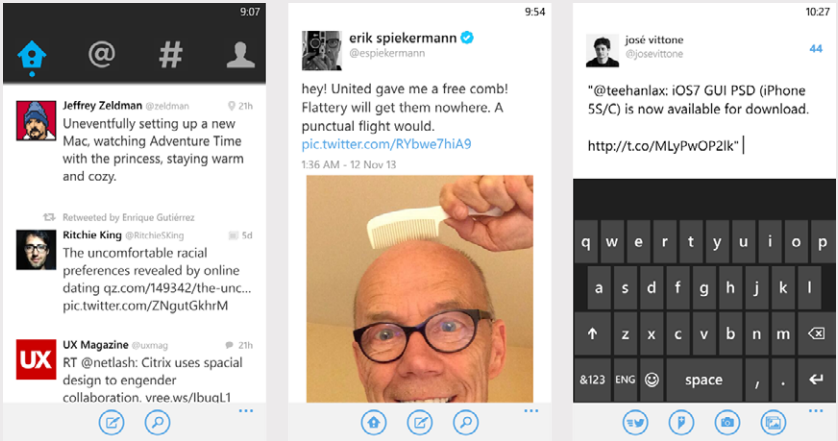
7 http://www.windowsphone.com/s?appid=10f2995d-1f82-4203-b7fa-46ddbd07a6e6

← *Fig 13.13. Main screen with Panorama, list of songs and playback options.*

Fig 13.14. Album thumbnails.

Twitter[8]

At a glance, we know it's Twitter's app, and that's saying a lot. They've found a way to strike a balance between native elements and visual identity without diverging from Microsoft's proposed guidelines. It is a clear example of a content-centered application.



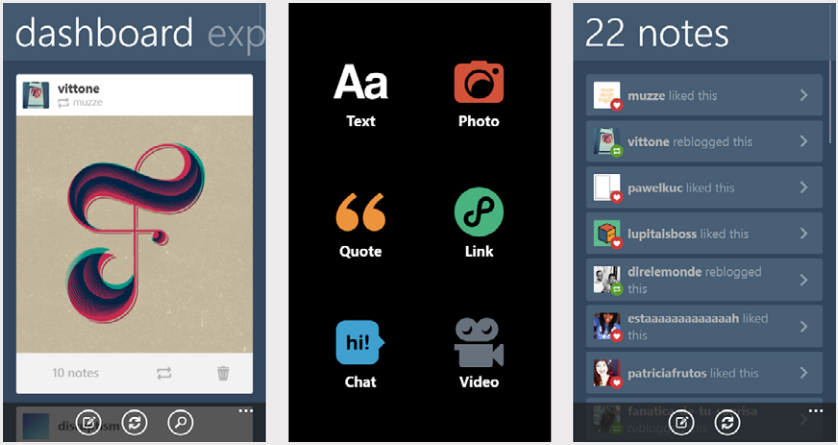
8 http://www.windowsphone.com/s?appid=0b792c7c-14dc-df11-a844-00237de2db9e

← *Fig 13.15. Main screen, tweet detail with picture and composition screens.*

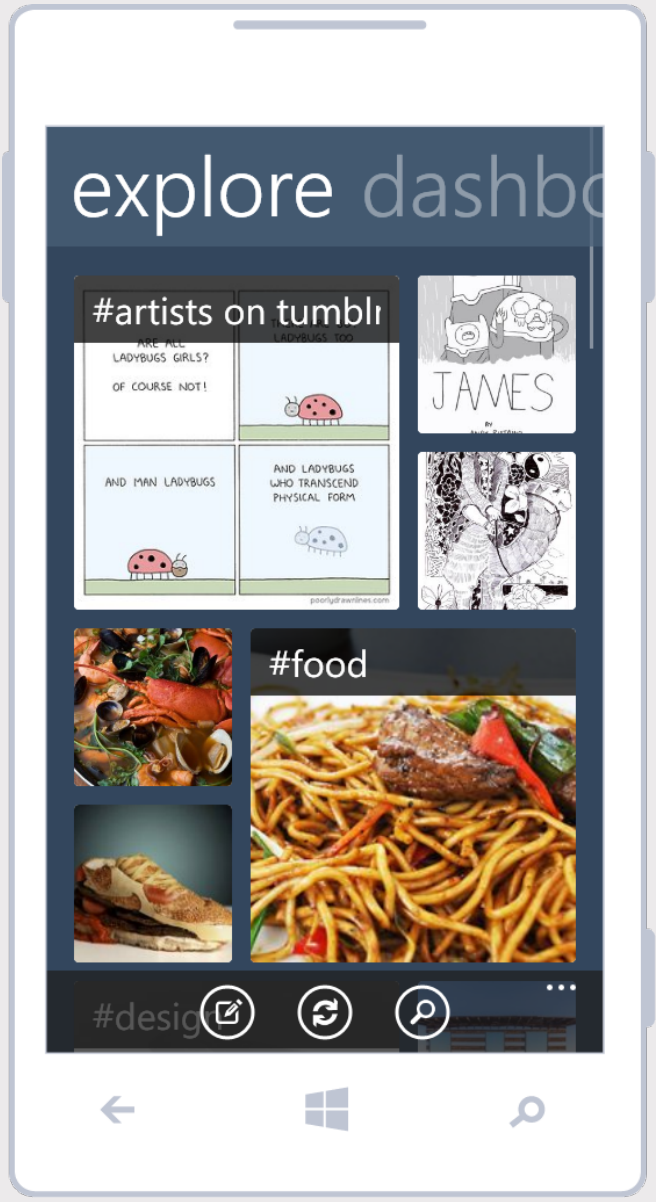
Fig 13.16. Tweet detail.

Tumblr[9]

The microblogging platform understood the mobile environment, and their answer was a simple and effective app. The features familiar to web users are maintained in an app that is focused on the exploration of visual material and the ability to publish content quickly.



9 http://www.windowsphone.com/s?appid=ffa2fb4f-61b2-4075-ac7b-488846998b72



← *Fig 13.17. Main screen, selecting the new entry type and notes of an entry.*

Fig 13.18. Exploring content.

14

The World of Tablets

Designing for tablets implies a larger load in terms of the space and circumstances present. Taking an app from smartphone to tablet isn't a literal translation and requires careful analysis.

Users are constantly alternating between devices. And much of the time, they're using multiple devices or engaging in other activities, such as watching television, simultaneously.

Designing a tablet version of an app is a quite natural step. In fact, in some senses, it is a necessary continuation of the experience. But it's important to remember that you are dealing with two very different devices, not only in terms of size but also in the ways in which they are used (how and where) and the relationships they foster with users.

Both visual design and interaction are particularly affected by the tablet's characteristics, which is why understanding tablets enables us to make the most of them.

Getting to the Big Screen

Having so much space available can, at first, be disorienting. But like with smartphones, the success of an app's design for tablets lies largely in the correct use of that space.

A tablet's screen should not be interpreted only as a bigger area where the mobile phone app's interface can be transferred directly. Here are a few things to consider:

Visual Strength

A tablet enables a better appreciation of graphic details and interface approach. The smartphone's screen limits space in such a way that all visual components are conditioned by the area around them. On the tablet, some icons and illustrations have more space to be visualized and displayed.

This directly affects the quality of the images and the level of detail of the work involved. Some details that haven't been previously considered find themselves in the spotlight on a tablet, thus making interface design more important.

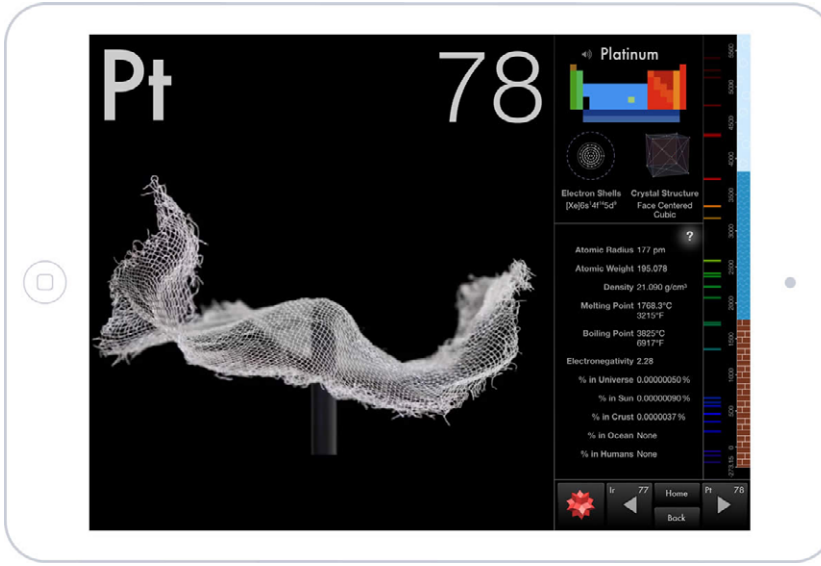


FIG 14.1. The space available in a tablet allows graphics to look better on screen.

Typography

In comparison with a mobile phone, tablets are held in such a way that the screen is farther away from the user's eyes. This makes it compelling to adapt the fonts used and their visual characteristics, thus ensuring the legibility of texts. The farther the screen is from the user, the bigger the font and line spacing should be to foster correct legibility.

Use of Space

The way in which a bigger space is managed not only has an influence on the visual but also on navigation and, in turn, on user interface interaction.

While the smartphone's reduced space forces users to follow more steps to access information, in a tablet, this sequence can

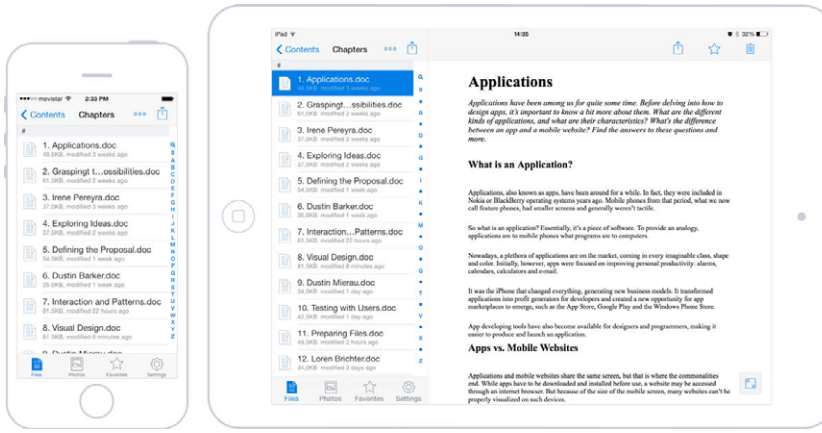


FIG 14.2.
Tablets use space differently, allowing viewing two screens in one. Dropbox shows a list of files alongside content, something that can't be done in an iPhone without changing screens.

be reduced. The tablet enables the inclusion of what would be separated into two screens on a smartphone in one single view.

To make the most of the situation, and as an example of use, the left side panel can be utilized to include a list of items, and the content can be displayed at the same time on the right side of the screen. Elements and content are related at a glance, without having to change context. In an email app, you can view inbox elements and message content simultaneously.

Having more space also makes navigation easier, since fewer levels are needed to find specific content. The screens previously separated by depth can now share the same level.

Operating systems offer different ways of making the most of space and managing panel division. Independently of the choice, it's important to relate the selected item to the one visualized and decrease the interruption caused by screen navigation. This can be done by moving the main content to a central area, with the help of dialogues and floating elements for secondary actions.

These divided views should be considered horizontally and vertically, anticipating how they will adapt to orientation changes. It's important to consider this, because with a tablet, orientation change is much more frequent than with a mobile phone. On the iPad, for example, the elements on the left column when in landscape format can be accessed from a drop-down menu when the tablet is in portrait orientation.

Gestures

Sometimes, a smartphone imposes restrictions because it is uncomfortable to make gestures with more than two fingers at the same time, or because of the lack of space and distance that separates them when making the gestures. In a tablet, gestures can be easily made.

There is much more freedom when it comes to manipulating directly the graphic elements of the interface, with less need for buttons and controls to perform many actions. For example, rotating a photograph, a task that used to be somewhat difficult, can be done with two or three fingers, and can become an easier action.

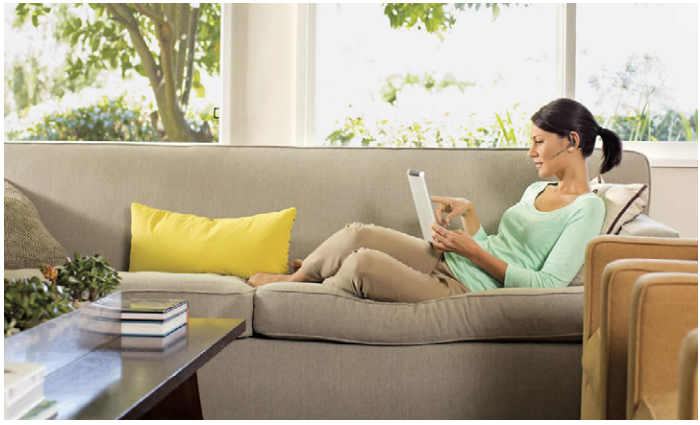


Fig 14.3. Tablets are more associated with a home use context, where users are more relaxed and have more time to interact.

Context of Use

In general, the device used is affected by its context—where we are, what we want to do, and how much time we have.

A tablet is usually more related to a home environment. It's common to use it laying down on a sofa or bed. The user is generally more relaxed and has more time to spare than with a smartphone.

Here's a statistic: tablet users spend almost twice as much time on the internet than mobile phone users, especially for news and entertainment[1].

This context establishes a very important difference, because it affects the way users relate to the tablet and how they interact with it. In consequence, it influences how apps are used.

Interaction

Tablets are harder and heavier to transport than smartphones. The user needs a supporting point that allows him or her to bear the weight of the tablet.

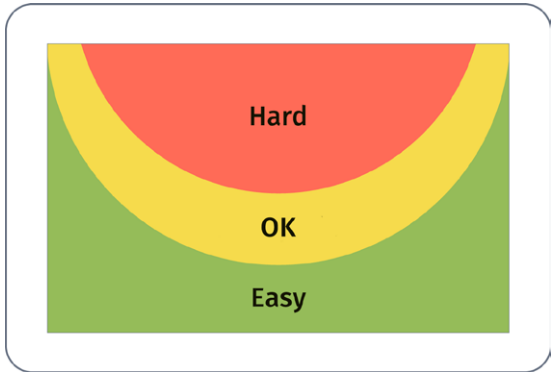


Fig 14.4. Because of the size and the way of holding a tablet, the lower area of the screen provides easier access, and the upper area presents more complexity.

1 http://services.google.com/fh/files/misc/multiscreenworld_final.pdf

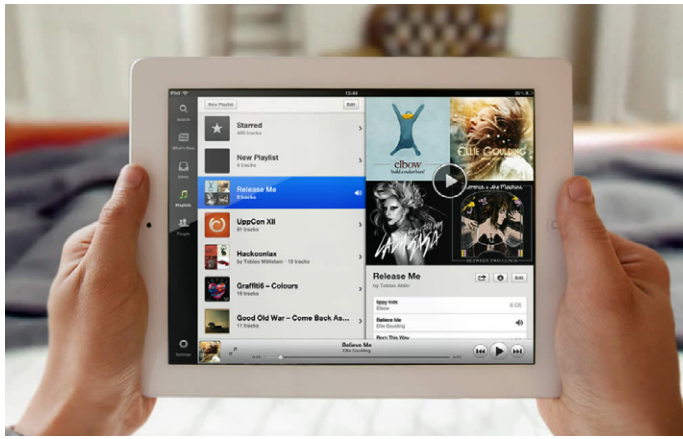


FIG 14.5. Spotify is an example of an app that is very well designed for the iPad, taking advantage of the characteristics of this media.

Depending on whether they're holding the device in portrait or landscape orientation, users have different levels of ease of access to varying parts of the screen (the lower corners). This is where the most frequent actions should be. Locate outside of this comfort zone should be actions that are more critical or have a bigger impact, thus reducing the risk of them being tapped by mistake.

Interactive elements located at the center and superior parts of the screen should be reduced. Placing them in these areas requires the user to cross his or her hand and arm over the screen, blocking content. This renders it impossible to see the element being tapped and the result of the action.

Functionality

Users most often use their tablets in the comfort of their own home, where they share the devices with others. These factors mean it's important to rethink the tasks performed with an app.

In a smartphone, for a user that many times is on the move and more focused on performing quick tasks, some features are pointless. In the case of a tablet, the situation is quite dif-

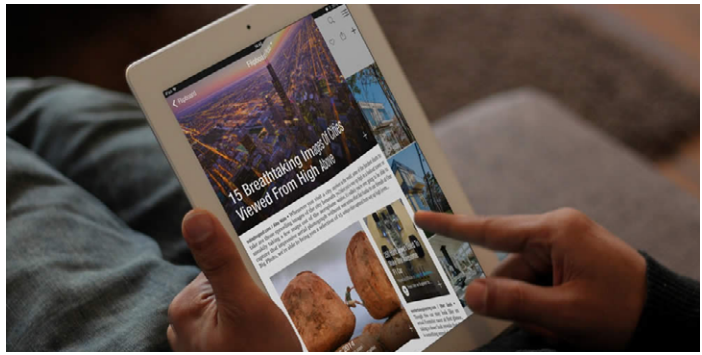


Fig 14.6.
In its iPad version, Flipboard is similar to a magazine and reinforces this concept with screen transitions.

ferent. That's why some of the functional characteristics that had been relegated now make sense.

You can broaden and improve user experience without losing focus on tasks that contribute to the app's goal. As we said before, it comes down to making sense—not adding in elements just because.

Consider an app used for visualizing documents that also allows for editing. While performing these tasks on a mobile phone can be quite hard, data and text input on a tablet is relatively easy.

App Content

Tablets are the preferred target of some apps that, because of their content, require a more relaxed use context.

This is the case of apps that require reading extensive texts, such as newspapers and magazines. An example of this is the iPad[2] version of Flipboard, an app that really makes the most of the experience by replicating a magazine and transferring the concept seamlessly through information quality, visual design and animations.

2 http://flipboard.com/

15

Erik Spiekermann: The Master of Typography

In the world of typography, this German figure is a star. Erik Spiekermann has designed numerous typefaces throughout his career, among them ITC Officina and FF Meta (the latter is often referred to by designers as 90s Helvetica).

Meta, based on more rounded shapes than Helvetica, has been used in an endless number of designs around the world. Firefox is no exception and, in fact, selected it for its corporate identity. Erik has worked with the company to create Fira[1], a Meta-based typeface. Simpler and wider, Fira is easier to read on a screen and is ideal for the recently-launched Firefox OS for mobile devices.

Spiekermann explained that digital environments don't mean a new condition for typography. Instead, they can be compared to specific scenarios that have existed for quite some time:

> *Bad resolution, small type, lots of information, readers in a hurry, changing environments —all those are not new conditions and information designers have been dealing with those forever. The screen is just bad paper, as far as type is concerned.*

However, choosing a font family for an app is no simple task. Searching for those with the best performance on poor-quality printing paper is not enough, and there are certain parameters that should be taken into account. Luckily, Spiekermann came to the rescue and told us what we should consider when choosing a typeface:

> *Use typefaces that were designed for information. If space is no issue, wider ones are better, if space is tight (and it normally is) use those that work for telephone books or forms (all my faces do, eg FF Info, FF Unit, FF Meta, ITC Officina) like Bell Gothic, Bell*

1 http://www.mozilla.org/en-US/styleguide/products/firefox-os/typeface/

FIG 15.1. Meta has become a modern classic and is used all over the world.

Centennial, Lucida (still Apple's best), plus a lot of robust new faces: FF Tisa, FF Yoga, FF Suhmo.

Low contrast for serif faces, open shapes (unlike Helvetica but like Frutiger) and some contrast for sans faces.

Considering the years of experience Spiekermann has in the area of typographic design, when he speaks, we listen.

Currently, Erik forms part of Edenspiekermann, a communications and design agency with offices in Amsterdam, Berlin, Stuttgart and San Francisco. Visit his website at www.edenspiekermann.com

16

Launching the App

The culmination of the design and development process is publication. In order to reach that point, certain requirements must be fulfilled. And, of course, the story goes on after launch—a new chapter of user feedback and improvements.

Publishing the App in Official Stores

An app is published after the testing stage, when there is no doubt about its correct functioning, stability and performance. At this point, the app should be free of usability and design errors.

The publishing process of each store is relatively easy and well documented. Before starting, it's better to be prepared and have all required design resources. Make sure that the app conforms to all publishing policies so as to avoid rejection or suspension.

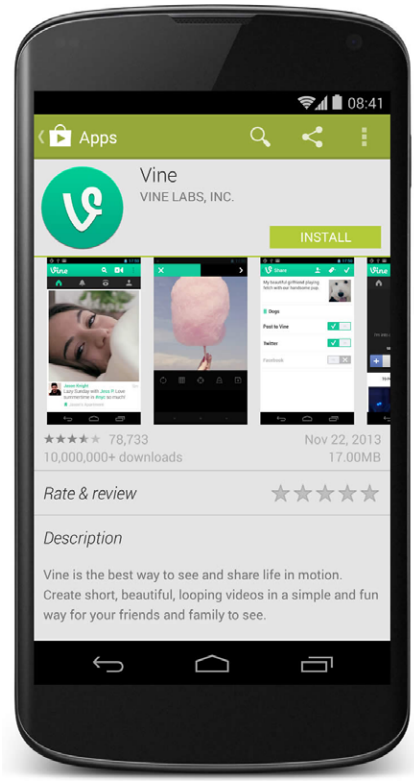


Fig 16.1. Visual elements, such as launch icons and screenshots, are essential to convince the user to download the app when he or she comes upon it in a store.

It can take a few days to get an app approved, especially when it's the first time it is being submitted. This timeframe should be considered, especially when the intention is to launch at a certain time of the year, such as Christmas.

It's also worth noting that publishing isn't free. Initiating the process (which varies according to the store) in Google Play, App Store and Windows Phone Store comes at a cost.

Google Play requires the payment of a one-time only 25-dollar fee when creating a developer account. In the case of the App Store, the cost is higher and renewable (99 dollars per year). The Windows Phone Store presents developers with an almost identical situation.

Creating a developer account and making the corresponding payment grants access to a series of management tools and statistics on app downloads and monetization.

Promotional Images and Elements

When an app is published, not only should the app file be uploaded, but also those elements that will accompany the app on the store's promotional page — namely, screenshots, descriptive texts, launch icons and images[1].

Prepare these resources well. They will be visible to the public and prove vital in promoting and communicating your product. Texts and images should be meticulously readied and comply with the store's guidelines regarding format, sizing and resolution[2].

In the particular case of screenshots, choose those that are most representative and attractive. If a screenshot doesn't show anything significant about the app, leave it out. Take the oppor-

1 http://msdn.microsoft.com/en-us/library/windowsphone/help/jj206715(v=vs.105).aspx
2 https://support.google.com/googleplay/android-developer/answer/1078870

tunity to add other graphics that explain how the app works and that reinforce the concept the app is meant to communicate.

When the content on the screens includes names of people and places, or if it has photographs or titles, replace that information with data that best simulates a real situation.

Even though it may seem like a minor detail, the status bar should be as clean as possible, with no distracting icons, even though it might be necessary to tweak the image in order to make it look more presentable.

Pay attention to the selection of these elements. Together with the app's icon, they are essential in getting downloads. Many users will make the decision of whether to download the app based on what can be seen here.

The Approval Process

Uploading an app does not guarantee that it will be published. Each store has a different approval process.

Google Play's process is much more open and tolerant[3]. Most apps are published unless they clearly breach the store's policies, in which case they may be removed. If the situation calls for it, the developer's account may be suspended. Of course, having such an easy publishing process also implies a plethora of apps that aren't exactly useful.

The App Store has an especially rigorous approval process. Its policy has been conceived to ensure a certain quality standard for app publishing. At times, however, it borders on exaggeration.

The complete guide of causes of rejection can be found in the developer's account. To name a few, there are those that make reference to non-original content, those that are similar to existing apps, apps that are not finished (for example, demos

3 https://support.google.com/googleplay/android-developer/?hl=en&rd=1#topic=2364761

or betas), and apps that are offensive to other people, religions or that have content deemed inappropriate for children.

In the case of Windows Phone, the app must also go through a certification process that verifies its compliance with all of the store's rules. The most important aspects to take into account for an app's approval are content quality and general appearance—there should be no doubt about the operating system to which the app belongs, no unnecessary decorations or elements, and a good use of system buttons[4].

Publishing and Distributing Elsewhere

The option of distributing apps by means of official stores is, without a doubt, the most recommendable. Such systems have mechanisms in charge of providing the necessary resources to manage download payments and in-app purchases, and they offer advertisement. However, this is not the only way to proceed.

Alternatives to Google Play

There are several options for Android when the idea is to avoid or complement publishing in Google Play. There are other stores that also offer app downloads for this operating system, such as the Amazon Appstore[5] and Samsung's store[6].

Alternatively, download can be offered directly from a website. With a link to the app file, anyone can download and install it in his or her phone without encountering problems. The only

4 App certification requirements for Windows Phone. http://msdn.microsoft.com/en-us/library/windowsphone/develop/hh184843(v=vs.105).aspx

5 http://www.amazon.com/mobile-apps/b?ie=UTF8&node=2350149011

6 http://apps.samsung.com/

prerequisite is that users enable the *install third party apps* option on their devices[7].

Another way of distributing an app for Android is via email. Because an app consists of only one file, it can easily be sent by email. If the receiver has enabled the install third party apps option, the *install now* button will appear in the message.

This option is great for trusted and trial users, and also when there is a desire to keep a certain independence from stores, with all the limitations doing so implies. Clearly, the biggest inconvenience is that it gives free rein to piracy, since anyone with the file can send it to someone else without appropriate consent.

iOS and Windows Phone, Only for Testing Purposes

In the case of iOS and Windows Phone[8], independent distribution is not as free and massive. Before being published in the App Store or Windows Phone Store, the app can be sent to individual users, but the accounts of those who will use the app on their smartphones must be authorized previously, one by one.

This way of distributing the app has no other purpose than making it available inside a company or giving it to a select and limited group of users for pre-launch testing. This practice is advisable especially to ensure good functioning in different devices and operating system versions.

7 http://developer.android.com/tools/publishing/publishing_overview.html

8 Company app distribution for Windows Phone. http://msdn.microsoft.com/en-us/library/windowsphone/develop/jj206943(v=vs.105).aspx

After Launch

Work on the app is not finished once it has been launched and published in the stores. In fact, the most exciting stage begins here, because the product is now in the hands of real users. The app is now dressed and roaming the streets[9].

This means those who use it will begin to share their experiences and impressions. This, combined with use and download statistics[10], will serve as a reference for improving the app and correcting design and functionality flaws that have been overlooked.

These kinds of improvements will be conducive to better quality, and this, in turn, will translate into a greater number of downloads and more positive comments. Users are the key to spreading the word about an app, and it's better to keep them happy. Remember that the opinions of the press and specialized media can attract more downloads, too.

User Comments

Opinions are wonderful to have as feedback—and not just from your circle of friends and family, but from those outside your social group as well.

Once uploaded to the store, each app receives ratings and reviews. Although not all of them are helpful, pay attention to all comments made and identify those that are most useful.

There's no better opinion than that of a user who's downloaded and used an app. Reports about functional failures and opinions about design and usability can be found in the user comments. Use them as guidelines for constant improvement.

9 Improving App Quality After Launch. http://developer.android.com/intl/es/distribute/googleplay/strategies/app-quality.html

10 http://appfigures.com/

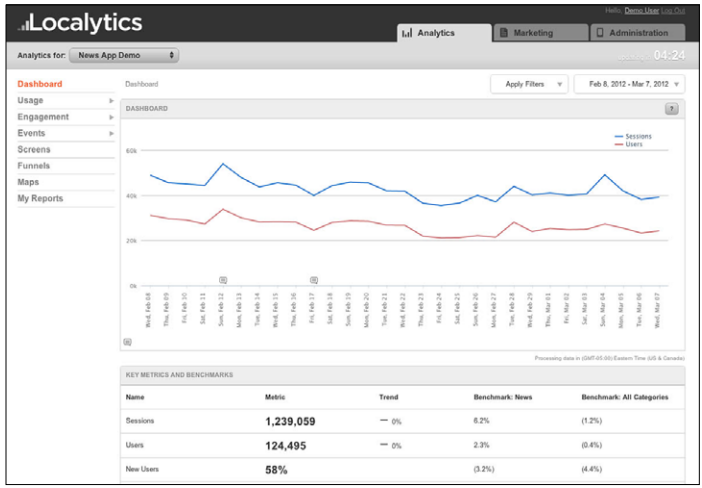


FIG 16.2. Using analytics allows us to obtain information about user behavior and app performance in order to make improvements and correct errors.

Usage Analytics

Just like on the web, you can collect statistical data about an app's use with the help of analytical tools designed for mobile devices.

Analytics enable the study of user behavior, such as navigation and interaction patterns. It's possible to discover the most visited screens and how they're found, as well as the buttons and elements most often used. Another piece of information that can be useful and can be collected from analytics is retention time, in other words, the amount of time the users spend with the app.

Focus analytics on key performance indicators (KPIs) that make sense for the kind of app developed. There are many different types of apps, and each requires different metrics. In a game, it's important to study retention time and the number of occasions the user opens the game again. But for social networking, it would be more interesting to analyze the quantity of new content published and how often that content is shared.

Getting the right information implies using data more efficiently. As a result of a conscious analytical study, a series of improvements can be made available in future versions or updates of the app, refining and improving it even more.

Once there is a somewhat clear prospect, you can start using analytics immediately, even long before the app is launched on the market. This is especially useful when alpha or beta versions are distributed. Likewise, integrating analytics beforehand is a good way of finding out whether they have been installed correctly to produce the desired results. When making the app public, it's best that analytics are already prepared so you can start gathering information from day one.

There are many tools on the market for analytics management. Some examples are Google Analytics[11], Mixpanel[12], Flurry[13] and Localytics[14]. Each has different characteristics, and choosing one or the other will depend on certain parameters, such as the information they provide, cost and user quantity.

Promotion

To get downloads, it is essential to promote your app and let users know it exists. Different stores address this task; however, not all promotional work can be left to them or to search engines. There are other ways of promoting your app—marketing campaigns, internet advertising and reviews on specialized blogs.

A very effective way of promoting an app is with a website that, unlike corporate and other product pages, is quite uncomplicated and has few levels of navigation.

Build a solid, minimal structure. Your landing page is the first thing users will see. It is here that you should provide the

11 http://www.google.com/analytics/features/mobile.html
12 https://mixpanel.com/
13 http://www.flurry.com/
14 http://www.localytics.com/

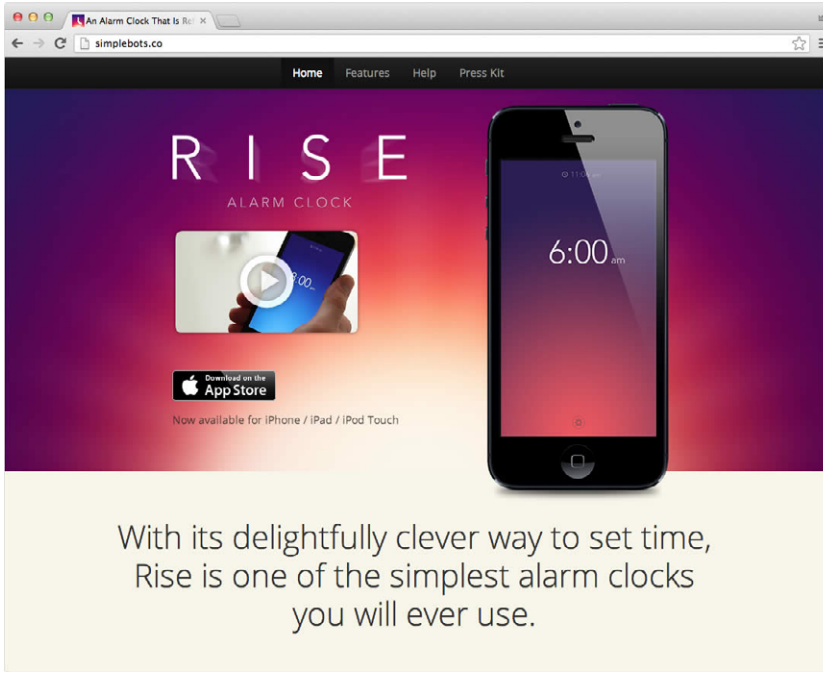


FIG 16.3. Rise has a landing page that does an excellent job. It presents the essential information, including a link to the store, and has an attractive visual appearance.

answers to some very basic questions. What does the app do? Is it free? What device models are compatible? Graphic design and message should help clear up any doubts as quickly as possible.

In the case of Android, downloads independent from the store are allowed, so the website will also serve as a distribution channel, providing links to obtain the app directly.

Nonetheless, the best option is to use the landing page as a gateway to the store. Each store offers varying promotional elements with their identities that can be included in a website, such as download buttons[15].

15 http://developer.android.com/intl/es/distribute/googleplay/promote/badges.html

Updates

Comments, observations and usage statistics are transformed into a series of improvements that can be implemented in the app. So, when users have one version of an app, there are updates available to replace it[16].

When corrections to the app are necessitated by very serious failures that hinder use, a forced update can be performed. As the name indicates, a forced update coerces users into downloading the latest version. The risk associated with this kind of app update is that some users will stop using the app all together because they don't want to update and therefore won't be able to use the version already installed. In other cases, updates can be optional and downloaded whenever the user decides.

Updates are generally the consequence of a development cycle that lasts for a certain number of days. Work is performed in testing and implementing future changes that will result in a new version of the app. The advantage of working in this way is that the apps are improved constantly and that the users know that they can periodically expect a better version.

Android and iOS apps are often updated automatically, but Windows Phone still requires user intervention as apps must be updated manually.

Note that uploading a new version of an app to the Apple or Microsoft stores implies submitting it again for policy compliance review. In the case of Google Play, the process is much more straightforward and direct.

16 Warren, Christina. When & How You Should Update Your Mobile App. http://mashable.com/2011/09/22/mobile-app-update/

17

Firefox OS and Ubuntu for Phones: The Newbies

While we've generally focused on Android, iOS and Windows Phone, there are other mobile operating systems on the market[1].

Such is the case of Firefox OS[2] and Ubuntu[3] for smartphones, two representatives of a new group of players, also including Tizen and Sailfish, that are trying to earn a place among the giants. Both Firefox OS and Ubuntu have an open code philosophy and are opting for HTML as a structure base.

This quite possibly represents a clear shift in paradigm, a new way of perceiving mobile apps that affects design, development and user interaction.

We interviewed the people responsible for the design of Firefox OS and Ubuntu in an attempt to gain a deeper understanding of what it is that sets them apart.

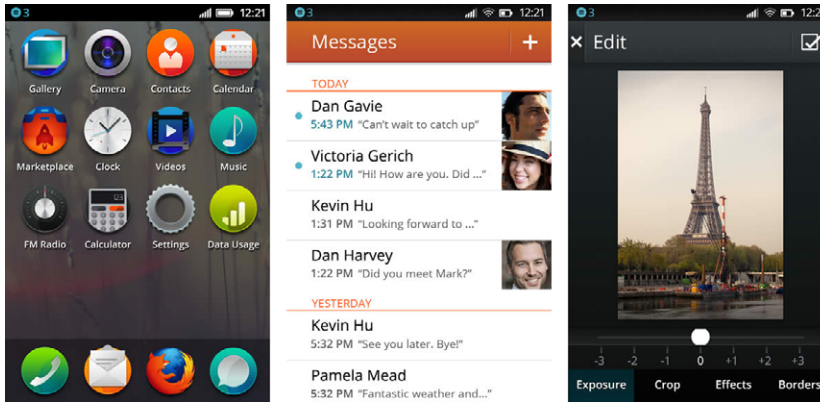
Firefox OS

Mozilla's operating system can already be found in some phones available on the market now, and it has the potential to reach many more in the months to come. It is the first bet the creators of the famous internet browser are making in the mobile realm.

Patryk Adamczyk is Chief of Design at Firefox OS. He filled us in on what's special about the system:

> *Mozilla's goal is to keep the web open and allow everyone to have access to the full web experience. Firefox OS not only ships with a fully functional browser but is build only using web technologies, allowing millions of web developers to write apps for the platform without needing to learn a new proprietary language.*

1 http://en.wikipedia.org/wiki/Mobile_operating_system
2 http://www.mozilla.org/es-ES/firefox/partners/
3 http://www.ubuntu.com/phone



In many ways Firefox OS functions in a familiar way to its competitors, as its positioned at users switching from outdated smart or feature phones. The aim was to make that transition smooth. For our version 1, device cost and the learning curve are low with a strong aim at the emerging markets where internet accessibility is untapped, most people access it through their home desktop or internet cafes.

We've designed most of the components using pure CSS for speed and scalability. As we progress through incremental upgrades in the coming months our plan is to evolve the UI to feel even more web-like.

Fig 17.1. Firefox OS is seeking out visual elements that build its identity but at the same time has clear influences from other operating systems.

Firefox OS's design principles[4] were the base upon which a visual style[5] was built. Even though we can expect the look of the interface to change and evolve over time, so far, Firefox OS is visually closest to Android, though it also demonstrates influences from iOS and Windows Phone.

4 Adamczyk, Patryk. MozCamp Warsaw: Design Principles Behind Firefox OS UX. https://blog.mozilla.org/ux/2012/09/mozcamp-warsaw-design-principles-behind-firefox-os-ux/

5 http://www.mozilla.org/en-US/styleguide/products/firefox-os/

The design of this operating system privileges content over container, attaching more weight to visual content, such as videos and photographs, on a grid that varies according to need, complemented by dark backgrounds that make graphics stand out.

The visual style of Firefox OS has textures and volumes, though they are very measured and controlled, especially in the case of apps like the clock, which seems to be made out of real materials.

For Firefox, typography has always been important, and this is also reflected in their operating system for smartphones. Erik Spiekermann's Meta typeface, used traditionally as the brand's identity seal, has evolved to become a mobile version called Fira, created by the same designer.

However, some of the more distinctive details of Firefox OS are the round icons on the home screen—something completely different from other operating systems.

It remains to be seen how Firefox OS will develop and evolve in the weeks and months to come. Will it manage to establish a visual identity that truly sets it apart?

Ubuntu for Phones

Ubuntu is a Linux distribution that has been around for a while now. Developed by Canonical, it is the preferred distribution of many Linux fans. It has focused all its efforts on its desktop computer operating system up to now.

However, things started to change with Ubuntu's first incursions in Android. As a logical consequence, it started to develop its own mobile proposal—largely an extension of its operating system.

Ivo Weevers, Chief of Design for Ubuntu, is the ideal person to answer the question of what makes Ubuntu different from other smartphone operating systems:

FIG 17.2. The landing of Ubuntu in smartphones is a promising alternative.

> *Ubuntu is the only platform that offers a true convergent design approach. The same core OS is running across all form factors, including phones, tablets, desktop and smart TVs.*
>
> *The user interface scales seamlessly and beautifully to adjust for different screen sizes and input methods. This enables unique solutions, such as running a desktop from the phone, or building an app once that will run everywhere.*

Clearly, for such an omnipresent operating system, it's a challenge to maintain design guidelines in each platform.

So, what will be the design principles of Ubuntu for smartphones?[6]

> *Ubuntu uses simple natural swiping gestures from the edges of the screen to make it easy to access your content and switch between apps.*

6 http://design.ubuntu.com/apps/get-started/design-vision

Every edge has a specific purpose (which takes the edge/gesture approach further than any other solution), making all your apps, content and controls instantly accessible, without the need to navigate back to the home screen every time.

Touch screens started to allow contextual keyboards: only show the keyboard when the user needs it. Ubuntu takes this a step further by showing UI controls only when the user needs it.

We've used 3 key design principles, which underlie this design direction:

1. *Focus on the content.*
2. *Fast and natural interactions.*
3. *Sophisticated style.*

Although some of these principles are mentioned by other platforms as well, Ubuntu is the only one, which make this truly reality through the edge interaction model.

Unfortunately, this operating system is currently only available for Nexus. Those with other smartphone models will have to wait a bit longer to try it out.

The anticipation builds surrounding how the market will transform as new software options develop simultaneously.

How will they influence the rest?

18

Final Thoughts

In the chapters of this book, we've taken you through the entire app development and design process, from conception to publication.

We've developed the concepts we consider most interesting at each stage, as well as those not often found in operating system manuals and guidelines. The latter are the result of our personal experiences, information that, without a doubt, would have made our lives easier when we began.

We hope you make the most out of our words and take them as a starting point for your own investigation. Each subject covered is immensely rich, and it is impossible to include them completely in one book. You will surely be more interested in some topics than others, and we recommend you to use the links provided to broaden your knowledge.

The Best is Yet to Come

Once you have finished this book, you'll be ready to put what you've learned into practice. There is no need to wait until a client requests a new project. You can start by making your own apps. Even if they don't turn into published and profitable products, they will serve as useful learning tools and help you to devise your own ways of solving problems.

Just one phone can be enough to get started in this world. Use it with awareness, and make critical observations of how others have built their apps. This way of interacting with your smartphone should become, from this moment on, standard practice.

Make the most of basic tools like a pen and paper to sketch out your ideas, write down concepts and evolve. You need nothing but your imagination. And who knows, those rough sketches may end up being the next big thing in mobile.

The Role of the Designer

We firmly believe that the designer's role goes beyond the traditional stereotype of an interface builder. Design is an incredibly interesting discipline that is full of possibilities and forking paths, especially because of its close relationship with technology. In an app, we can talk about information, interaction and visual design.

It's normal to identify more with and feel more attached to certain roles in a project. Performing the same role continuously over time is what makes you a specialist. But what really makes you a professional is having knowledge that goes beyond the borders of your particular field of interest. In our case, it's knowing how the other stages of design and development of the app are executed. As such, interaction and integration with the other members of a team are essential in generating the synergy needed to build better-quality apps.

A designer must be involved in the entire process, even the parts that are more technical and tedious. It can't be avoided. On the contrary, it should be maximized to improve the designer's own work.

Developers need to go above and beyond, too. They should understand design work and be conscious of the impact it has on interface. Besides making an app look good, design also affects usability.

Understanding this dynamic can help a programmer to incorporate an aesthetic and visual concept to aid decision-making in the building process. Programmers have to be guided by their own instincts and refrain from depending entirely on designers.

We no longer talk about professionals of the future. Today's professionals are those who can solve problems and propose solutions that go beyond their defined roles. It's time for you to become one of them.

The Future of Apps

It's an ideal time to immerse yourself in app design and development. Even though operating systems have their own proposals, they are constantly evolving and generally open to innovation proposed by designers. Patterns are partially open, leaving room for a designer's initiative—something pointed out by Loren Brichter in our interview.

Likewise, new operating systems, such as Firefox OS and Ubuntu, are emerging. We now have to wait and see how the panorama will change over the months to come. We intuitively know that the design philosophy of each platform will be affected by the rest, just like the influence of the plain Windows Phone style set a precedent that its competitors have embraced.

These are exciting times, and the best is yet to come.
Be prepared.

Get Involved

Now that you've finished the book, you may want to start designing your app.

If you still have questions, doubts and concerns regarding the subjects covered, feel free to send them our way. Contact us at via email **hola@appdesignbook.com** or on Twitter (reach Simón at **@millonestarde** and José at **@josevittone**). The advantage of writing a digital book is that it can be updated relatively easily. Obsolete content, incorrect data and broken links can be updated and improved in future versions of *Designing Mobile Apps*.

We encourage you to visit our website, **appdesignbook.com**. All of the content contained in this book is available there, free of charge, for everyone. It is an excellent opportunity to interact with us and other readers who, just like you, want to tackle the world of apps.

We look forward to hearing from you.

Javier "Simón" Cuello
José Vittone

Glossary

The following pages provide a more detailed explanation of some of the terms used in this book. Although, in some occurrences, the meaning of certain terms can go beyond the context of apps, we have decided to narrow down the scope of the descriptions to our area of concern.

Accessibility

The ability to access content, regardless of a person's physical capacities. At a visual level, accessibility is determined by the size of text and buttons and by the contrast of those elements against the background. An accessible app also refers to correct code programming that allows, for example, for content to be interpreted by accessories for people with visual disabilities.

App

The name most commonly used to refer to a certain kind of software; an abbreviation of the word application. An app is a piece of software that is executed in mobile phones and tablets and is our object of study.

Benchmarking

The systematic practice used to evaluate products, services and processes in a comparable manner. In our context, it's the comparative and analytical study of other applications to determine their quality and characteristics, using them as parameters for reference.

Compile

The action of packaging a code. As a result of compiling an application's code, the final file is ready to be uploaded to the store.

Context of Use

General environment that consists of the location and physical space around the user and the device. Context of use also determines the way in which these two components relate and interact with one other.

CSS

Acronym for Cascading Style Sheets. Whether in separate files or inside HTML code, this language determines the visual appearance of a web application and defines font color and size.

DP

Acronym for density-independent pixels; a unit of measure used by Android in relation to the screen's physical density. DPs are units relevant to 160 DPI screens, in which a DP is equal to one pixel.

Using DP instead of pixels is a solution suggested to correctly adapt an interface for use in screens with different densities.

Feedback

The answer from the interface to keep the user informed about the actions he or she has just performed. Generally immediate. Feedback can be a confirmation of success or an error message obtained when executing a certain task, and it can manifest itself by means of notifications or much more subtle visual elements.

Feedback can also refer to the observations and comments received from users, which can be used as parameters or indicators for improving an app.

HTML

Acronym for HyperText Markup Language. The language traditionally used to build web pages and web applications for mobile devices. It defines the structure of a web document based in a series of tags.

Interface or UI

The layer in between a device and a user that allows them to interact. In applications, interface makes reference to the graphic component containing elements that produce reactions when touched and that allow the user to perform tasks, and also to those static components that facilitate the interpretation of content.

JavaScript

Programming language used mainly for web projects like sites and applications. Often interacts with HTML and CSS to provide more functionality.

KPI

Acronym for Key Performance Indicator. KPI measures the variables that are being executed in a project with the aim of

obtaining relevant data to determine general performance and decipher whether set goals have been accomplished.

Library

In programming, this name is given to the set of external codes that can be used to cause certain behaviors. Directly related to the programming language chosen.

Mobile Device

Also called cell phone, an electronic device of variable size where applications function. You've probably got one with you now.

Monetize

The action of obtaining an economic profit. In an application, this is related to the different ways of obtaining income from an app is also directly tied to business model and commercial strategy.

Operating System or OS

The software contained in each telephone where applications are executed. The different versions of Android, iOS and Windows Phone are examples of operating systems.

Orientation

The way content is displayed on a screen according to how the user is holding a device (tablet or smartphone). Portrait or landscape.

Persona

Personification of users as the result of studies based on their behaviors and ethnographic characteristics. This investigation is based on the common patterns detected in users. The concept of Persona was created by Cooper and is a common tool for interaction design.

Pixel or PX

Physical units made up of colored dots that are spread across a screen surface. In design, pixels are used as units of measure for the graphic components of an interface in different editing programs.

Ranking

Ordered classification in which the stores sort applications according to factors like number of downloads and positive reviews.

Scenario

The combination of context of use and Persona. It determines how the user relates to the mobile device in a specific situation.

Screen Density

Quantity of pixels per physical space in a screen. Generally measured in DPI (dots per inch). Densities may vary from one device model to another and in general can be grouped by low, medium and high, depending on the operating system.

Screen Resolution

The number of pixels that can be shown on a device's screen. It consists of the width and height relation.

Screen Size

The physical size of the screen from one edge to the other, measured diagonally and in inches.

SDK

Acronym for Software Development Kit. An SDK provides programmers with the necessary tools for developing an application's code. Offered by Android, iOS and Windows Phone.

Simulator

A simulator makes it possible to test an application without a mobile device. Code may be executed in a computer, and results can be visualized on the screen to perform preliminary checks on an app's functioning.

SP

In Android, the acronym for scale-independent pixels used for texts. In design, they behave in the same way as DPs, the only difference being that the size measured in SP can also be affected by user preferences.

Store

The application distribution and marketing channel for free and paid downloads. Each mobile operating system mentioned in this book has an official store; however, in the case of Android, there are several other alternatives besides Google Play, for example, Amazon and Samsung app stores.

Theme

Combination of colors used by Android and Windows Phone by default. In Windows Phone, the user can choose from a selection of themes that affect background colors and standout elements in all screens of the operating system.

Usability

In a much broader sense, usability is related to the application interface's effectiveness and efficiency in allowing a determined user to perform a task or accomplish a goal. Usability cannot be analyzed in isolation and is tied to a particular context and specific user. It is directly associated with user experience.

User

The user is the individual who interacts by means of the interface. The user is the focus of user-centered design, which is based on the principles of meeting the user's needs. Its main goal is to come up with solutions that solve problems, taking into consideration emotions and expectations.

User Experience (UX)

Concentrates the emotions or perceptions of a person when using an interface or product. In the case of apps, UX is influenced by a group of factors that determine whether the experience is positive or negative, including accessibility, visual design, interaction design and usability.

Image credits

Introduction
Photo of the illustrations by PÉREZ, Catalina. enseguida.net/collections/tuga-prints/

1. Applications

1.1. By the authors. 2013.
1.2. WALTON, Trent. 2012. http://trentwalton.com/2012/10/03/a-new-microsoft-com.
1.3. By the authors. 2013.
1.4. By the authors. 2013.
1.5. By the authors. Screenshots of Facebook. 2013.
1.6. By the authors. Screenshots of Netflix. 2013.

2. Grasping the Possibilities

2.1. By the authors. Screenshot of Angry Birds. 2013.
2.2. By the authors. Screenshots of Path. 2013.
2.3. By the authors, based on image by Realmac Software. 2013. https://itunes.apple.com/es/app/clear/id493136154?mt=8.
2.4. Sophiestication Software. Press Kit. 2013.
2.5. By the authors. Screenshot of Paper. 2013.
2.6. By the authors. Screenshots of Cut the Rope and WhatsApp. 2013.
2.7. By the authors. Screenshot of Line. 2013.
2.8. Halfbrick. 2013.
2.9. By the authors. 2013.

3. Irene Pereyra: UX with Magic in Fi

3.1. Fi. http://www.f-i.com/nickelodeon/kids-choice-awards.

3.2. Fi. http://www.f-i.com/nickelodeon/kids-choice-awards.

4. Exploring Ideas

4.1. By the authors. Screenshots of iBulb, iLed FlashLight and Linterna. 2013.

4.2. DE SOUSA, Fernando. 2013.

4.3. Dropbox Inc. Press Kit. 2013.

5. Defining the Proposal

5.1. By the authors. 2013.

5.2. By the authors. 2013.

5.3. By the authors. Screenshot of Instagram. 2013.

5.4. By the authors, based on image byS UX Booth. 2012. http://www.uxbooth.com/articles/lessons-learned-designing-a-windows-8-app.

5.5. Estudio Chipsa. 2013. www.chipsa.ru.

5.6. STARK, James.

5.7. ARNALL, Timo.

5.8. Android. 2013. http://developer.android.com/intl/es/design/downloads/index.html.

5.9. By the authors. Screenshot of Omnigraffle. 2013.

6. Dustin Barker: Electronic Banking Made Simple

6.1. By the authors. Screenshot of Simple. 2013.

6.2. By the authors. Screenshot of Simple. 2013.

7. Interaction and Patterns

7.1. By the authors, based on image by HOOBER, Steven. 2013. http://blog.utest.com/how-mobile-users-hold-devices/2013/03/

7.2. By the authors. Screenshot of Weightbot. 2013.

7.3. By the authors. 2013.
7.4. By the authors. 2013.
7.5. By the authors. 2013.
7.6. By the authors. 2013.
7.7. By the authors. 2013.
7.8. By the authors. 2013.
7.9. By the authors. 2013.
7.10. By the authors. 2013.
7.11. By the authors. 2013.
7.12. By the authors. 2013.
7.13. By the authors. 2013.
7.14. By the authors. 2013.
7.15. By the authors. 2013.
7.16. By the authors. 2013.
7.17. By the authors. 2013.
7.18. By the authors. 2013.
7.19. By the authors. Screenshot of Google. 2013.

8. Visual Design

8.1. By the authors. Screenshots of Gmail. 2013.
8.2. By the authors. Screenshots of iOS. 2013.
8.3. By the authors. Screenshots of Windows Phone. 2013.
8.4. By the authors. Screenshot of Wunderlist. 2013.
8.5. By the authors. Screenshot of Vine. 2013.
8.6. By the authors. Screenshot of Google Play. 2013.
8.7. By the authors. 2013.
8.8. By the authors. 2013.
8.9. By the authors. 2013.
8.10. By the authors. 2013.
8.11. By the authors. 2013.
8.12. By the authors. Screenshots of Skype, rdio and Nike+ Running. 2013.
8.13. By the authors. Screenshot of iOS. 2013.
8.14. Android. 2013. http://developer.android.com/intl/es/design/style/metrics-grids.html.

8.15. By the authors. Screenshot of iOS. 2013.
8.16. By the authors. Screenshots of Windows Phone. 2013.
8.17. By the authors. 2013.
8.18. LÜSCHER, Christoph. 2012. http://ia.net/blog/responsive-typography-the-basics/
8.19. By the authors, based on image by Android. 2013. http://developer.android.com/intl/es/design/style/typography.html.
8.20. By the authors. Screenshots of Bloomberg. 2013.
8.21. By the authors. 2013.
8.22. By the authors. 2013.
8.23. By the authors. 2013.
8.24. By the authors. 2013.
8.25. By the authors. Screenshots of The Guardian. 2013.
8.26. By the authors, based on image by Microsoft. 2013. http://msdn.microsoft.com/en-us/library/windows-phone/design/hh202878(v=vs.105).aspx.
8.27. Android. 2013. http://developer.android.com/intl/es/design/style/color.html.
8.28. By the authors, using images by Teehan + Lax. 2013. http://www.teehanlax.com/tools/iphone/.
8.29. By the authors. Screenshot of Letterpress. 2013.
8.30. By the authors. Screenshot of Gmail. 2013.
8.31. By the authors. Screenshots of Spotify. 2013.
8.32. By the authors. Screenshots of Google+. 2013.
8.33. By the authors. Screenshots of Clear. 2013.
8.34. By the authors. Screenshots of Windows Phone. 2013.

9. Dustin Mierau: Path and the Value of Details

9.1. By the authors. Screenshot of Path. 2013.
9.2. By the authors. Screenshot of Path. 2013.

10. Testing with Users

10.1. By the authors, based on image by BULT, Mark. 2013.
10.2. Mr. Tappy. Press Kit. 2013.

10.3. MELBOURNE, Ben. 2013.

11. Preparing Files

11.1. OpenSignal. 2013. http://opensignal.com/reports/fragmentation.php.
11.2. Android. 2013. http://developer.android.com/design/style/devices-displays.html.
11.3. YERMOLAYEV, Nikita. 2013. www.komarov.mobi.
11.4. By the authors. 2013.
11.5. By the authors. 2013.
11.6. LEMEDEN, Reda. 2013.
11.7. By the authors. 2013.
11.8. By the authors. 2013.
11.9. By the authors. 2013.

12. Loren Brichter: Thinking Outside the Box

12.1. By the authors, based on image by Atebits. Press Kit. 2013.
12.2. iMore. 2013. http://www.imore.com/hall-fame-loren-brichter-and-tweetie.

13. Design Best Practices

13.1. By the authors. Screenshots of 500px. 2013.
13.2. By the authors. Screenshot of 500px. 2013.
13.3. By the authors. Screenshots of Google Drive. 2013.
13.4. By the authors. Screenshot of Google Drive. 2013.
13.5. By the authors. Screenshots of The Guardian. 2013.
13.6. By the authors. Screenshot of The Guardian. 2013.
13.7. By the authors. Screenshots of National Parks. 2013.
13.8. By the authors. Screenshot of National Parks. 2013.
13.9. By the authors. Screenshots of Google Maps. 2013.
13.10. By the authors. Screenshot of Google Maps. 2013.
13.11. By the authors. Screenshots of Instapaper. 2013.
13.12. By the authors. Screenshot of Instapaper. 2013.
13.13. By the authors. Screenshots of Spotify. 2013.

13.14. By the authors. Screenshot of Spotify. 2013.
13.15. By the authors. Screenshots of Twitter. 2013.
13.16. By the authors. Screenshot of Twitter. 2013.
13.17. By the authors. Screenshots of Tumblr. 2013.
13.18. By the authors. Screenshot of Tumblr. 2013.

14. The World of Tablets

14.1. Touch Press. Press Kit. 2013.
14.2. By the authors. Screenshots of Dropbox. 2013.
14.3. Plantronicsgermany. Press Kit. 2013.
14.4. By the authors, based on image by WROBLEWSKI, Luke. 2012. http://www.lukew.com/ff/entry.asp?1649.
14.5. Spotify. Press Kit. 2013.
14.6. By the authors. Screenshot of Flipboard. 2013.

15. Erik Spiekermann: The Master of Typography

15.1. By the authors, using resources provided by Erik Spiekermann. 2013. http://spiekermann.com/en/downloads/.

16. Launching the App

16.1. By the authors. Screenshot of Vine. 2013.
16.2. Localytics. Press Kit. 2013.
16.3. Simplebots. 2013. www.simplebots.co.

17. Firefox OS and Ubuntu for Phones: The Newbies

17.1. Firefox OS. Press Kit. 2013.
17.2. Ubuntu. Press Kit. 2013.

Aknowledgements

We want to acknowledge and thank everyone who, in one way or another, was involved in our project. Their time, motivation and ideas have rendered the results much better than we ever imagined.

By sections and in alphabetical order:

Interviews

- Patryk Adamczyk *(Mozilla)*
- Dustin Barker *(Simple)*
- Loren Brichter *(Atebits)*
- Dustin Mierau *(Path)*
- Irene Pereyra *(Fi)*
- Erik Spiekermann *(Edenspiekermann)*
- Ivo Weevers *(Ubuntu)*

Comments and Content Contributions

- Magalí Amalla
- Daniel Armengol

- Paloma Celaá
- Pierluigi Cifani
- Andrés Colmenares
- Ruymán Ferreyra
- Armando Fidalgo
- Victoria Gerchinhoren
- Xavi Pinyol
- Florencia Rosenfeld
- Marc Torrent
- Sergi Vélez

Other Contributors

- Jordi Aguilà
- Jonatan Castro
- Catalina Duque Giraldo
- Rodrigo Encinas
- Esteban Humet
- Paula Marques
- Catalina Pérez
- María Daniela Quirós
- Jure Sustercic
- Seba and Maga

We would also like to thank our families, Lu for her support, and everyone who has accompanied us over these past few months, supporting and spreading the word about the project.

Thank you very much.