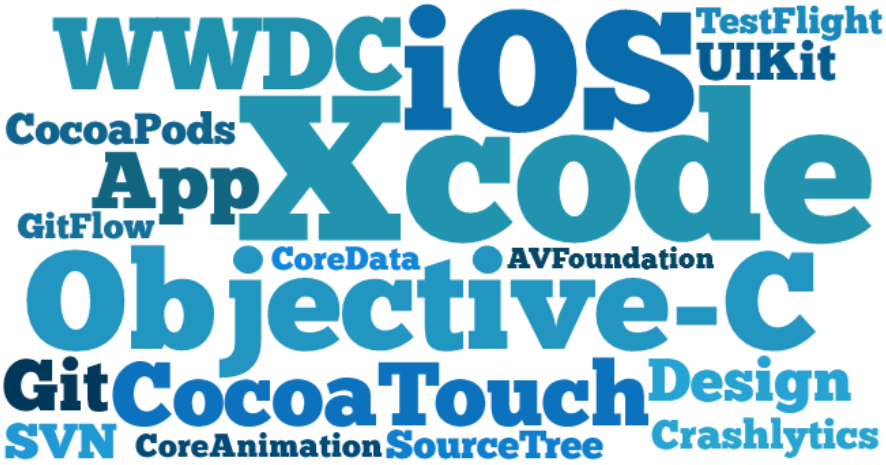
Matteo LALLONE

PROFILE Summary



Summary

This portfolio shows several iOS apps I worked on, while clarifying my experiences and skills in the iOS development field.

After getting a Master's Degree in Electronic Engineering, I worked for three years in a railway diagnostic company, as a system engineer dealing with computer vision systems for the detection of railway infrastructure defects.

During the last year in that company, I developed a strong interest for Apple products and the iPhone in particular. I started approaching the iOS development and found myself very passionate about that, more than I was passionate about being an happy user. I decided to start a career as an iOS developer, and I was thrilled about being able to work on topics I liked so much.

I had the chance to work on several projects for high profile clients, during my job as an employed iOS software engineer. Every team I joined could benefit of my proposals, solutions and great efforts towards the spreading of the iOS community best practices: I enjoy reading a lot about the latest updates, and I am always ready to experiment with the latest bit of news (the WWDC is by far my favorite event of the year!).

After working full-time in different companies, I decided to put my efforts in personal projects and freelance activities. I also had the opportunity to manage the activities surrounding the launch of an independent application on the App Store, and dealing with everything is needed in order to emerge from the App Store chaos.

PROFILE Core skills

My style

How I write code

I like to keep things simple, avoiding heavy pre-optimizations and cumbersome implementations. I really like working with UI/UX designers to build engaging products, and I have so much fun in experimenting with them new solutions.

iOS Development

What I use

Xcode: for the complete iOS development process.

Instruments: a series of Apple development tools to measure and diagnostic performances of my projects.

Git: for source control.

GitFlow: to maintain a clean workflow with Git, with a nice branch structures for features and main actions.

SourceTree: to manage Git repos with a solid IDE with GitFlow support.

Fabric: for an advanced crash reports management.

TestFlight: to deliver test builds to testers and customers, with an App Store-like behavior and very simple setup for final users.

Core skills matrix

Objective-C	2010-2017	Advanced
Xcode	2010-2017	Advanced
Cocoa Touch	2010-2017	Advanced
Git	2011-2017	Good
SVN	2011-2013	Good
Design Patterns	2008-2017	Advanced
C++	2008-2014	Good

Localizable Strings Merge: to handle string localization in my projects.

I really like relying on my own code to have full control on my projects, but for some tasks I use some open source tools.

CocoaPods: for a clean dependencies management.

AFNetworking: to handle network communications.

SSKeychain: when iOS keychain-related tasks are needed.

Google Analytics: when a fairly complex analytics structure is required.

Facebook SDK: when I need to deal with the Facebook API.

Learning

What I'm looking into

Speech Recognition API: Due to an upcoming update for a project I'm working on, I'm currently digging in the new (iOS 10+) Speech Recognition API, in order to provide dictation features with custom UX.

About me

What I like to do

I feel very lucky in being passionate about my day job. When I am not writing code for work or for fun, I read a lot of blogs and I follow constantly the work of a bunch of really great people, from the development, design and tech world in general.

I enjoy traveling whenever I can, and I've been to many countries so far. In my spare time, I like listening to a lot of music and watching TV series and movies.

Matteo Lallone

iOS software engineer

Email: alfgriever@gmail.com
GitHub: iGriever
Twitter: iGriever
LinkedIn: Matteo Lallone

PNGS e Monti della Laga 2011

Parco Nazionale del Gran Sasso e Monti della Laga

A guide for the hikers visiting the Gran Sasso National Park

Available on the App Store (Sincro Consulting S.p.A.)

My responsibilities

This was my first project as a professional iOS developer. It was really exciting having the possibility to be totally in charge of the development of the app, following a series of company guidelines. I took care of the whole development process, and later on helped with the Android porting of the application as a consultant for the general design of the app.

I also had the chance to suggest some solutions from the UI/UX point of view.

What I used

IDE: Xcode

Source control: SVN

Network: ASIHTTPRequest

Data persistence: SQLite with a custom Obj-C wrapper

Relevant frameworks: UIKit, Foundation, Core Location

Challenges

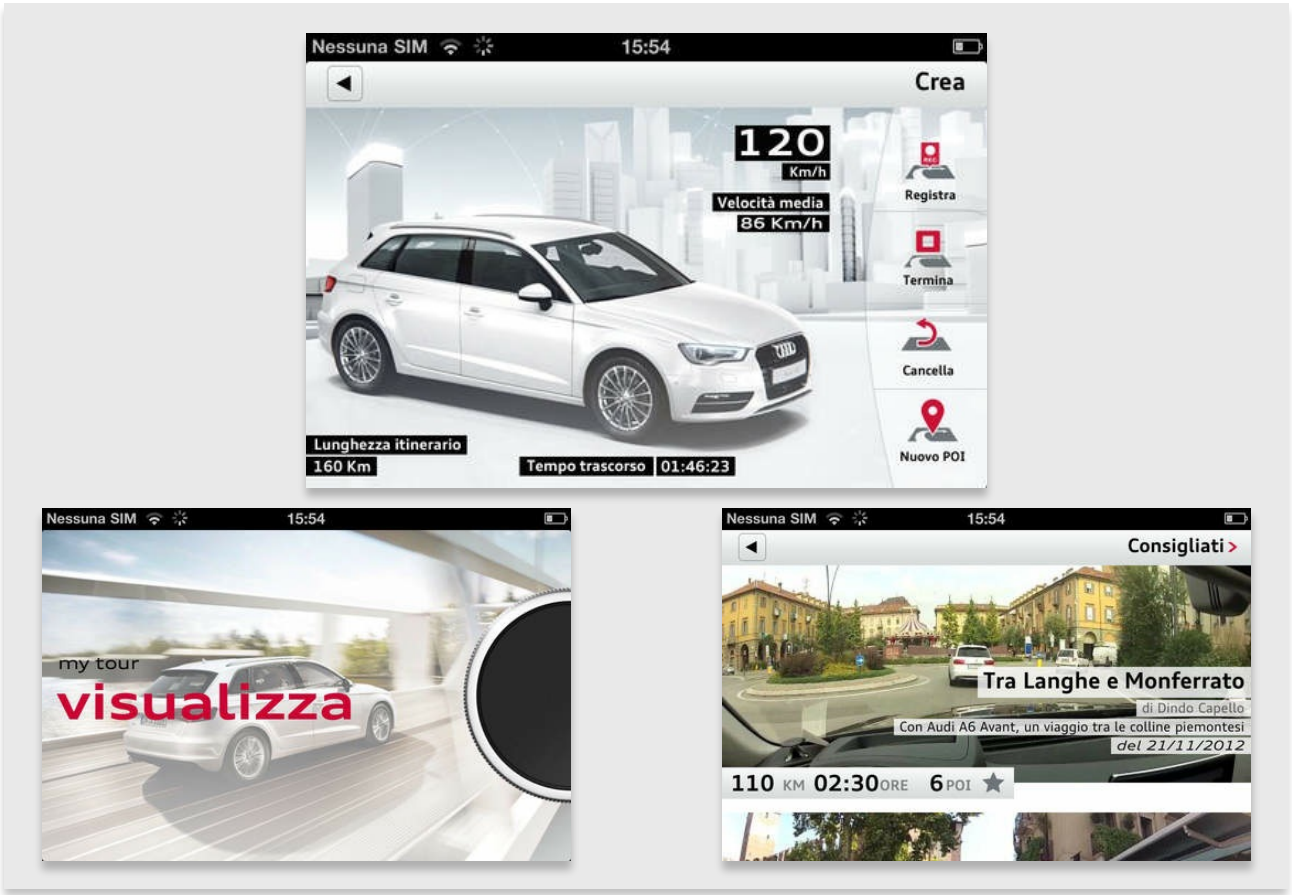
Of course, as this was my first iOS project, there were challenges on many aspects. As I was very passionate as a user, I shared some proposals to the UI/UX designer in charge for the project, in order to give a more pleasant experience to the final users (and to be able to play a little bit with some cool UI stuff). I tried to build and maintain a clean project, focusing on code and UI reuse. It was the first occasion in which I could start getting my hands dirty, always keeping an eye with the latest news in the field. At first it was overwhelming having to deal with a lot of Apple documentation, but soon I found myself happy in exploring WWDC sessions and sample code, and that was just the beginning of a very happy professional path.

Audi my tour 2012

Audi my tour

A travel app designed for Audi drivers

Available on the App Store (Volkswagen Group Italia S.p.A.)



My responsibilities

I was in charge for the app development for almost the whole building process. I worked together with our back end developers, in order to refine the API to be used in the iOS application. The final version, currently on the App Store, is quite different from the one I developed especially from the UI/UX point of view, due to some last-minute major changes in requirements coming from the customer. At that time, the app had been in testing phase for a while, and I had to move to another project waiting for the test results. The major rework was performed by a coworker of mine, but some of the core application logic remained the one I wrote.

What I used

IDE: Xcode

Source control: SVN

Network: RestKit (really a massive dependency, at the time it was a consolidated team practice, definitely overkill for our purposes)

Data persistence: Core Data

Relevant frameworks: UIKit, Foundation, Core Location, AVFoundation

Challenges

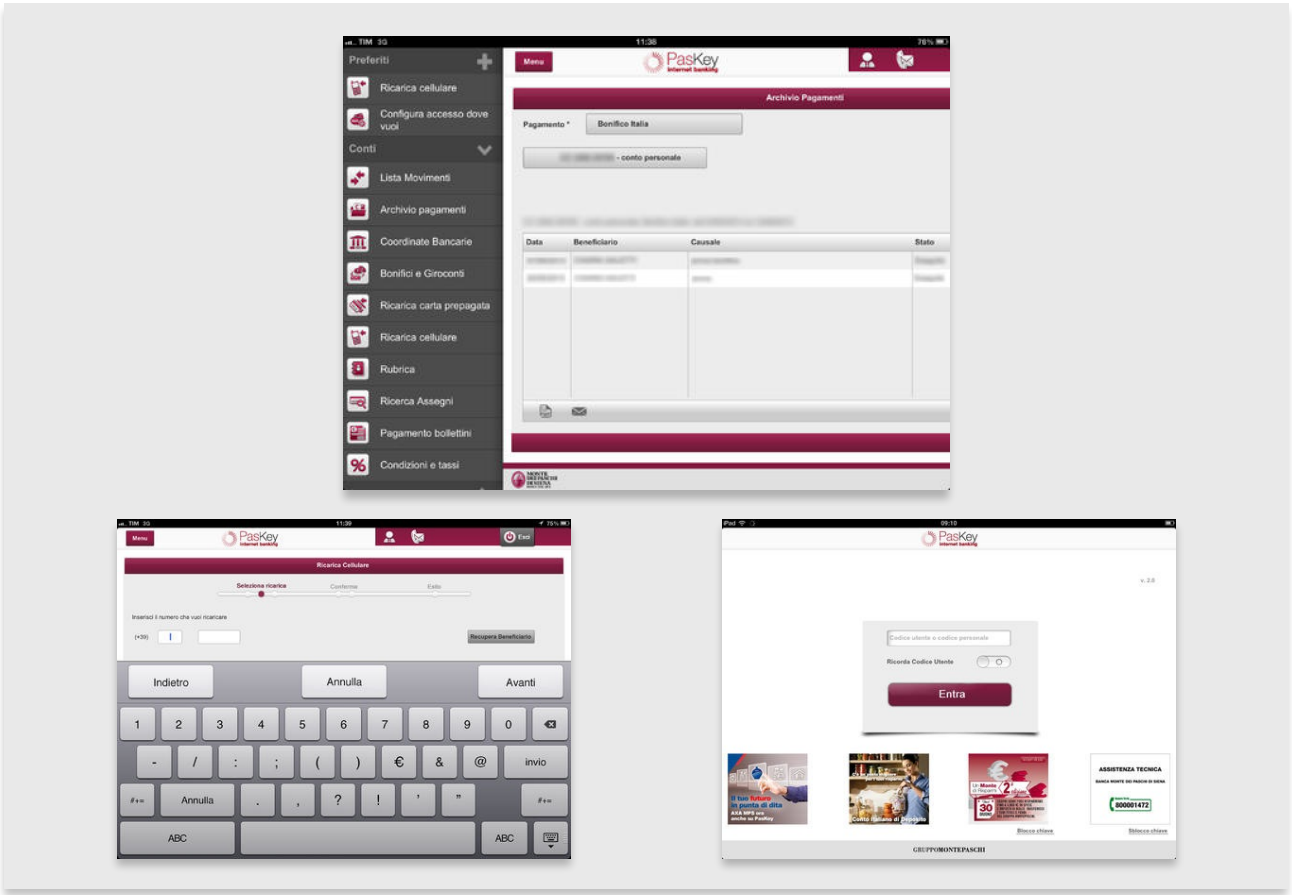
The goal of the app is giving users the ability to record and share travel routes on a dedicated Audi community. Each route is identified by its path and a series of media annotation, with text, photos and videos. Audi drivers get the chance to unlock some advanced in-app features. The main challenge here was dealing with the Core Location framework, in order to build the recording logic for the routes and implement all the required measurements on the current session with some filters in order to avoid spurious data.

MPS for iPad 2012-2013

Banca Monte dei Paschi di Siena per iPad

The official mobile banking app for MPS

Available on the App Store (Banca Monte dei Paschi di Siena S.p.A.)



My responsibilities

I took part of the development team when a major update was in progress, as the app was already on the App Store. The update involved the introduction of many new features for the mobile banking users, as well as the resolution of a series of pending issues.

What I used

IDE: Xcode

Source control: SVN

Network: ASIHTTPRequest (discontinued library, originally chosen for the project)

Relevant frameworks: UIKit, Foundation

Challenges

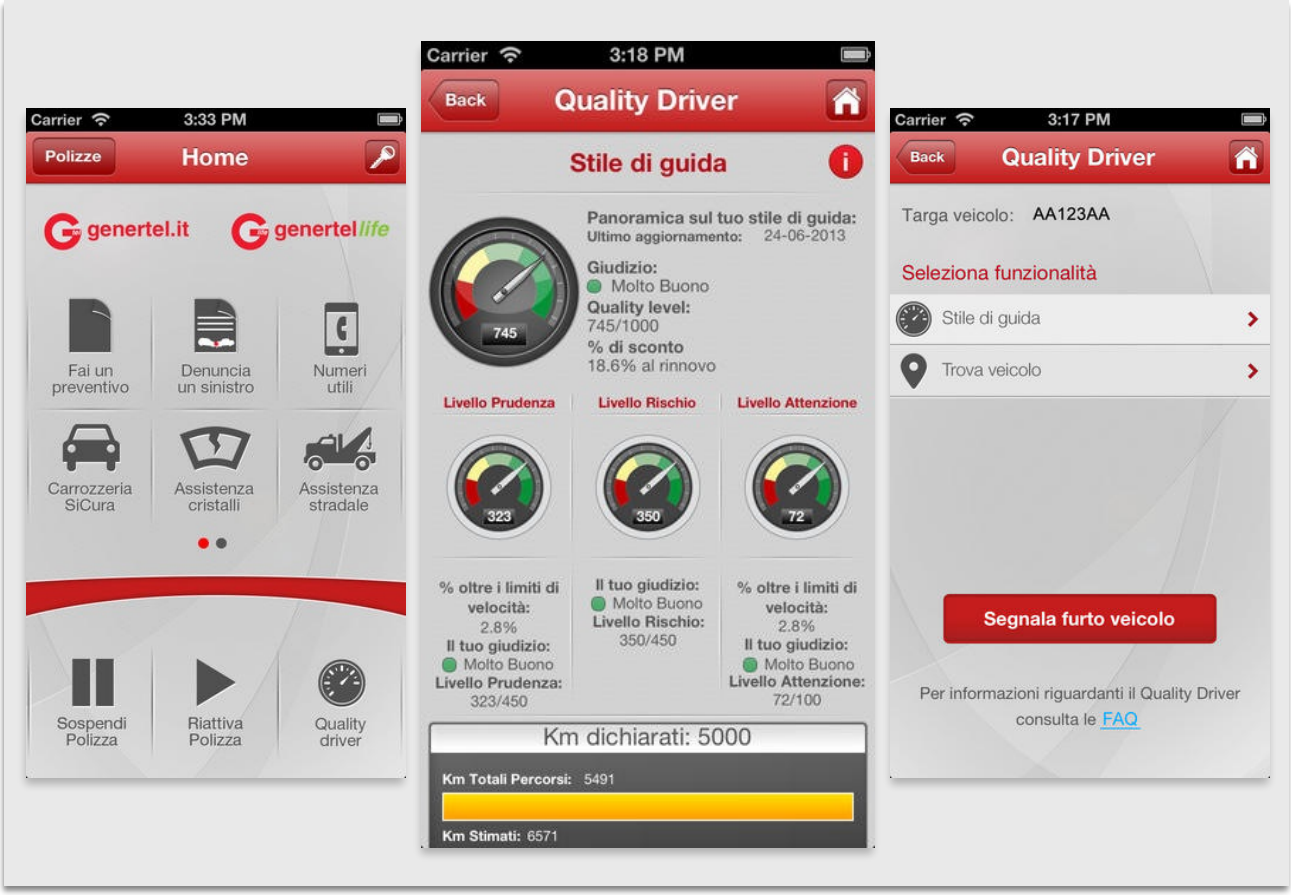
The challenges for this huge project came from two different point of views. The first one regarded the actual development on an existent application with a very complicate architecture, due to a heavily customized UI (with no iOS 5+ helpers, for compatibility reasons) and complex network and data layers. The second one concerned the team work, as the team was quite big, across three different companies, and we had to work with a very rigid workflow, using a ton of documentation and JIRA as an issue tracker for the whole development process. It was the first time I took part of such a large team on a heavy structured project from the documentation point of view, so it took a while to be comfortable with the whole process.

Genertel 2012-2013

Genertel

The official app for the insurance company customers

Available on the App Store (Genertel S.p.A.)



My responsibilities

I took part of the project in order to add two new sections to the app, for two corresponding new services granted by the insurance company. The first service basically gave the user the possibility to retrieve a list of driving style parameters, measured by a dedicated device provided by the insurance company. The device was able to communicate directly with the company servers, so the app was responsible of the communication with the company back end infrastructure. The second service used the data provided to the back end by the same device in order to find the vehicle in case of loss or theft, showing its position on a map, with route calculations. Moreover, I worked on the overall interface to add support for 4" displays.

What I used

IDE: Xcode

Source control: SVN

Network: ASIHTTPRequest (discontinued library, originally chosen for the project)

Relevant frameworks: UIKit, Foundation, Core Graphics, Core Animation, Core Location, MapKit

Challenges

Nothing too challenging here actually, besides working on a non-well optimized application, which required a lot more attention with the 4" display support introduction, in order to avoid missing some sections.

My Vodafone 2013

My Vodafone

The official Vodafone app for Italian users

Available on the App Store (Vodafone Omnitel N.V.)

My responsibilities

I took part to the development team for My Vodafone in two different occasions. The first one was to improve the application stability, getting rid of several issues and adopting Crashlytics for the crash report management. This allowed us to track and fix a lot of pending issues, improving the overall rating of the application. The second one was participating to the redesign of the old version of the application, which needed a revamp from the source code and the UI/UX design point of view. I worked on the transition to native components for one of the main sections of the application, which was formerly completed based on web views.

What I used

IDE: Xcode

Source control: SVN

Relevant frameworks: UIKit, Foundation, Core Graphics, Core Animation

Crash report management: Crashlytics

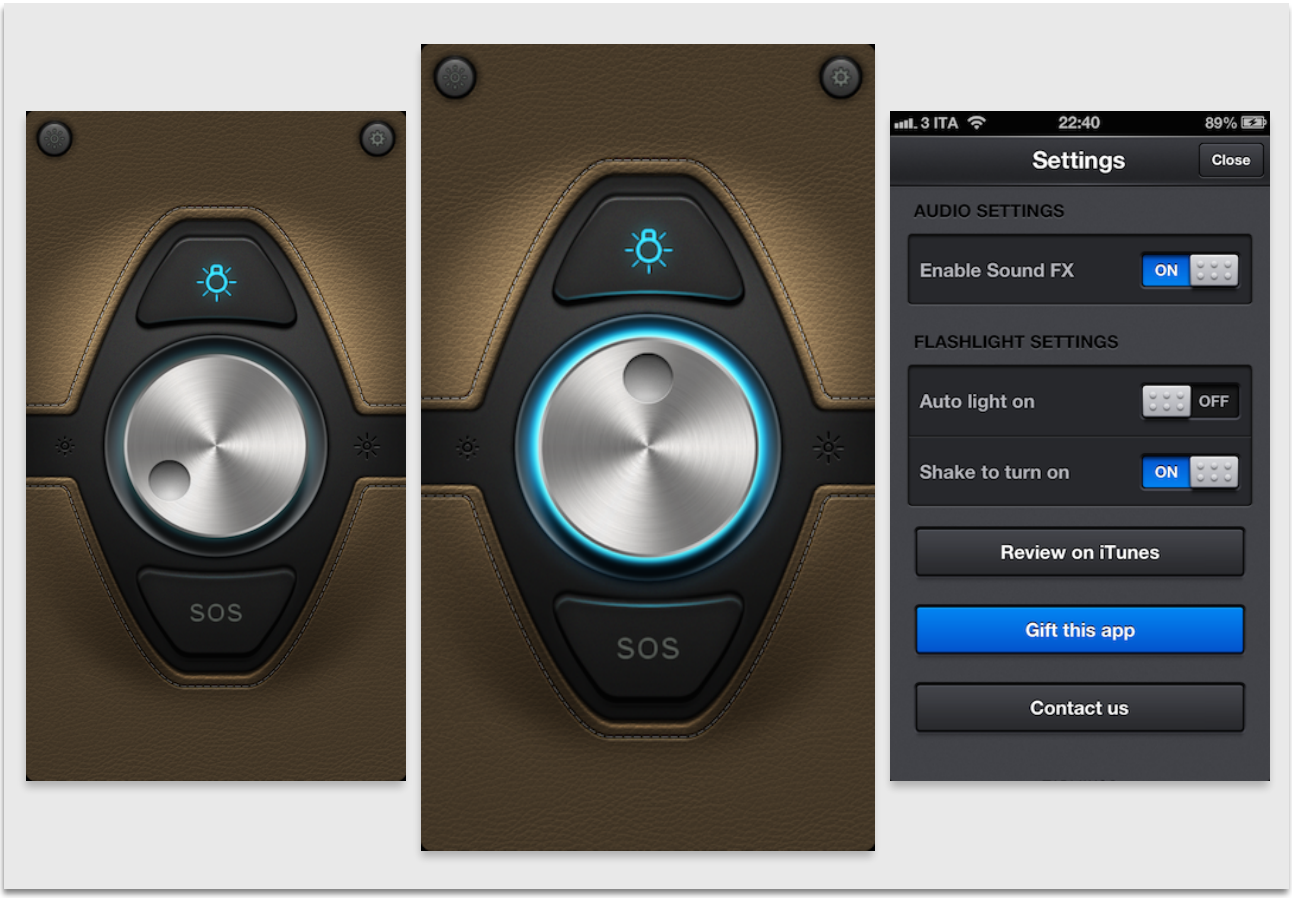
Challenges

The stability improvement tasks, working on the old version of the app, was pretty tough due to the average age of the source code, which involved manual memory management, discontinued open source libraries and so on. It was a nice chance to debug nasty memory-related issues, alongside having the chance to use and appreciate Crashlytics as a wonderful crash report management tool.
The redesign task involved the need to implement a nice demo application for the new user data section with native components. The hardest part was achieving smooth animations and overall consistent behavior with scrolling and fading in/out cells, after building a bunch of custom UI controls (circular and segmented slider, animated counter labels, etc.).

Fadelight

An elegant flashlight app for iPhone

Unpublished (Tapwings)



My responsibilities

I developed Fadelight in the same two-men team (Tapwings) which brought Boxie to the App Store. It started as a playground project, in order to test the team work with my fellow designer. Of course, we couldn't help but crafting every part of Fadelight with great care for details, even if it was "just" a flashlight application. This project gave me the occasion to use our custom UI library in a production project for the first time, and helped me refine the related API, and of course, polish the behavior and the appearance for many of the custom components I developed. Definitely a lot of fun, and our personal goodbye to the skeuomorphic design.

What I used

IDE: Xcode

Source control: Git

Relevant frameworks: UIKit, Foundation, AVFoundation, Core Motion, Core Graphics, Core Animation

Test management: TestFlight

Challenges

The basic flashlight logic was, as one can guess, pretty simple to implement. At the time, no App Store flashlights allowed users to set a variable intensity for the light (a new iOS 6 API allowed to do that).

The main challenges involved some particular UI bits, especially the variable intensity for the light reflections on the various components. The knob was definitely the most time consuming module of the app, since we decided to add some metal brush reflections effects according to the device motion sensor data, and adopted some cool visual tricks in order to do that right.

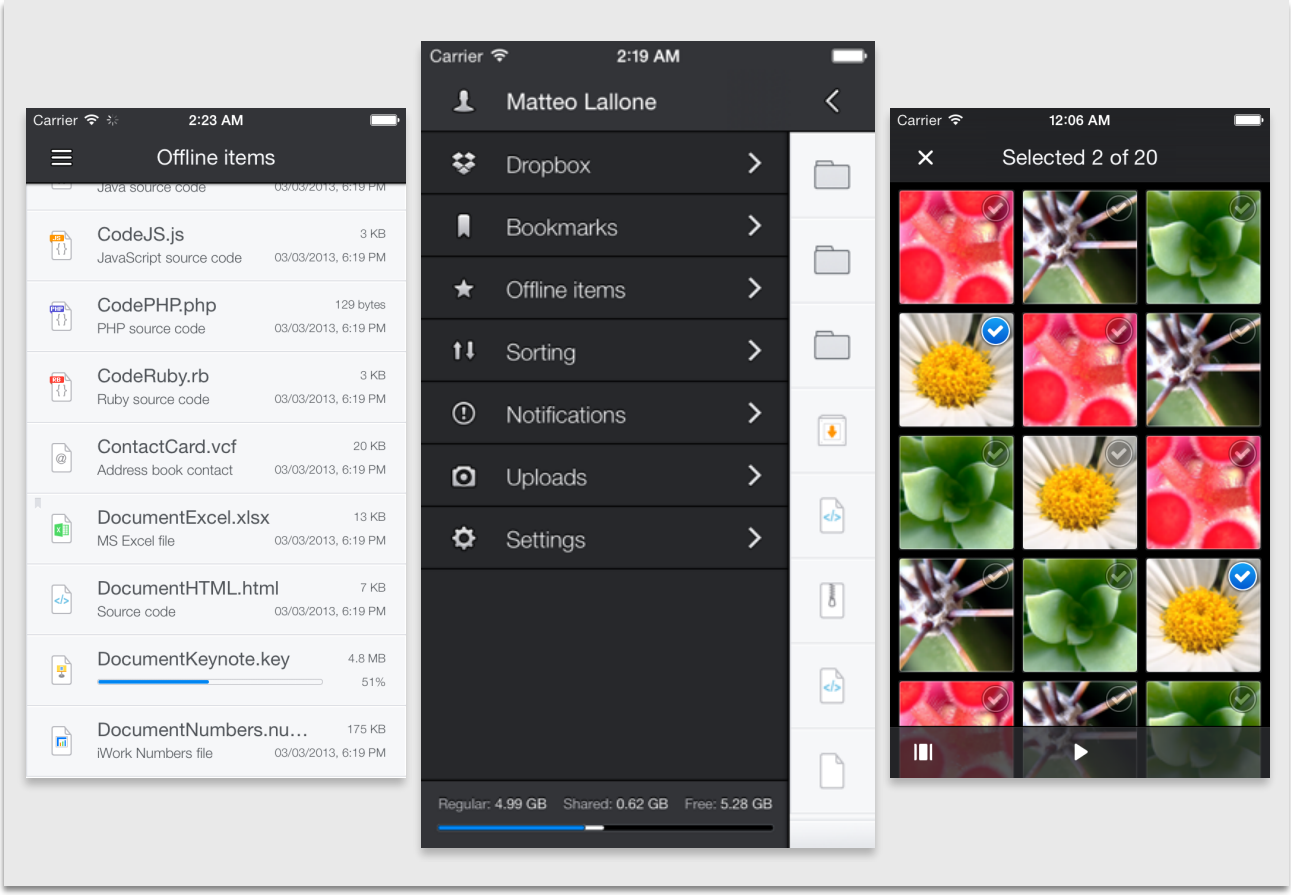
Fadelight couldn't reach the App Store eventually, as no more flashlight applications were allowed by Apple, at the time, even with the new variable intensity feature.

Boxie 2012-2014

Boxie - Prettify your Dropbox

A full-featured Dropbox client for iPhone

Previously Available on the App Store (Tapwings) - NOW DISCONTINUED



My responsibilities

I built Boxie from its foundations, taking care of the whole process, from the architecture design to the actual development. The work started on a wide library with custom UI components, which I used as a critical foundation for the project. Then I started working on the network layer, writing a block-based wrapper for the Dropbox Core SDK, and then on the local data persistence layer. I worked side-by-side with my fellow designer to polish the UI and UX of the app. Moreover, I had to deal with several other aspects concerning the development and the launch of an indie application, such as project management, test management, press relationship, promotion, analytics, customer support and, before and after the launch, dealing with the Dropbox team to start ongoing conversations about the product.

What I used

IDE: Xcode

Source control: Git

Network: Dropbox Core API

Libraries: SSKeychain, SSZipArchive, Google Analytics

Data persistence: Core Data

Relevant frameworks: UIKit, Foundation, AVFoundation, StoreKit, Core Graphics, Core Animation

Test management: TestFlight

Crash report management: Crashlytics

Challenges

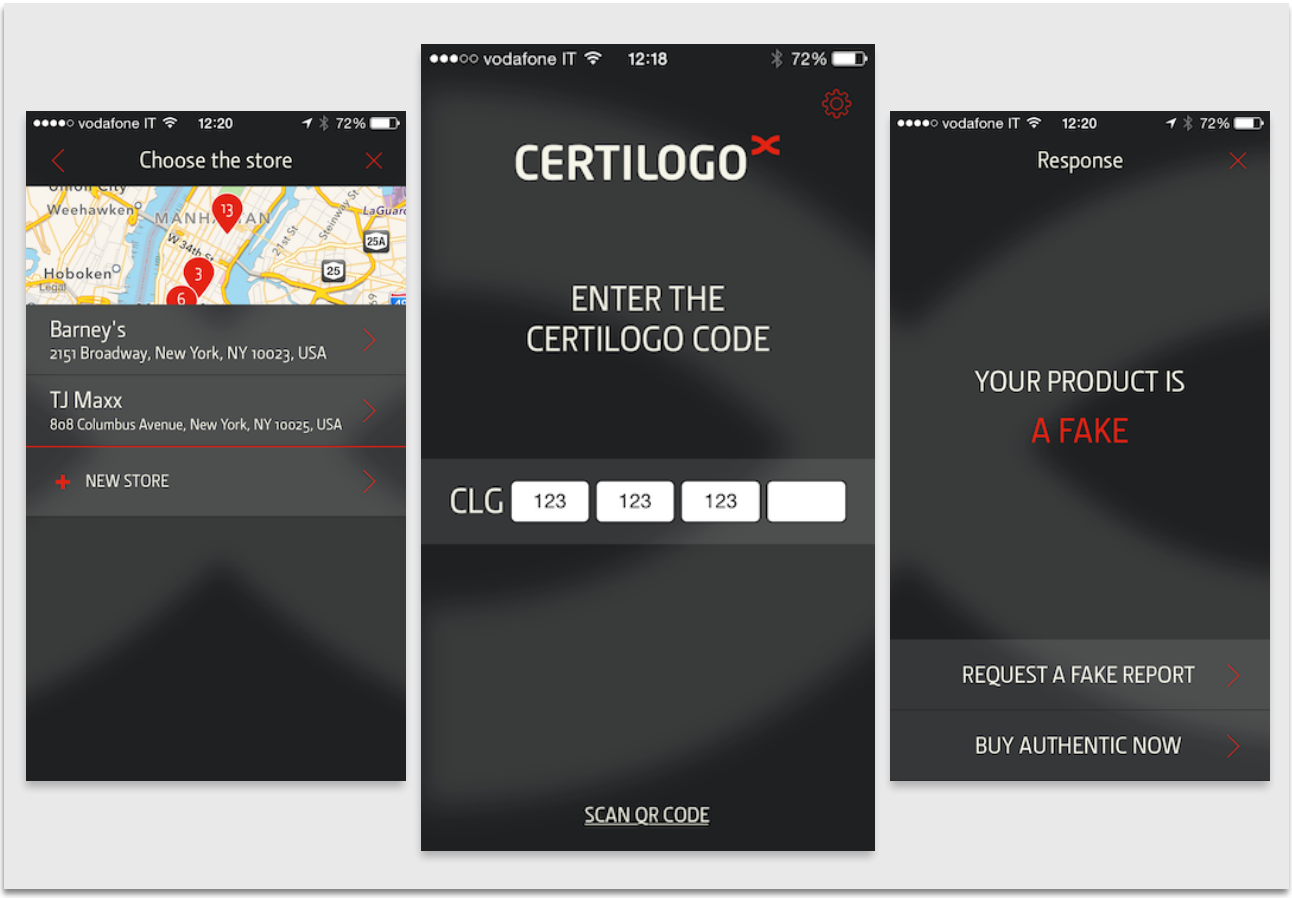
Boxie is my first full independent project. I had to design it in each part, together with a fellow designer. A great effort was put in building several custom UI components, in managing offline items handling and in polishing all the animations even with fairly complex view hierarchies. Definitely a lot of fun, with a lot of appreciation from users, press and Dropbox itself.

Certilogo Authenticator 2014

Certilogo Authenticator

A tool to verify the authenticity of commercial products

Available on the App Store (Certilogo S.p.A.)



My responsibilities

I was in charge of the whole iOS development process and ALC management. I also supported the in-house team in the analysis and design phase which lead to the API development, used by the iOS application. The application is now part of a platform, which previously served mobile devices only with a web app. The company who commissioned the project had no previous experience in native mobile development, so I provided my expertise in order to inform the client and make them comfortable with the best practices in native development.

My contribution constantly involved feedback about the UX/UI design of the application, too.

What I used

IDE: Xcode

Source control: Git

Network: AFNetworking

Libraries: Facebook SDK, Google+ SDK, CCHMapClusterController, SPGooglePlacesAutocomplete

Dependency Manager: CocoaPods

Relevant frameworks: UIKit, Foundation, CoreLocation, MapKit

Collaboration Tools: HipChat

Test management: TestFlight

Crash report management: Crashlytics

Challenges

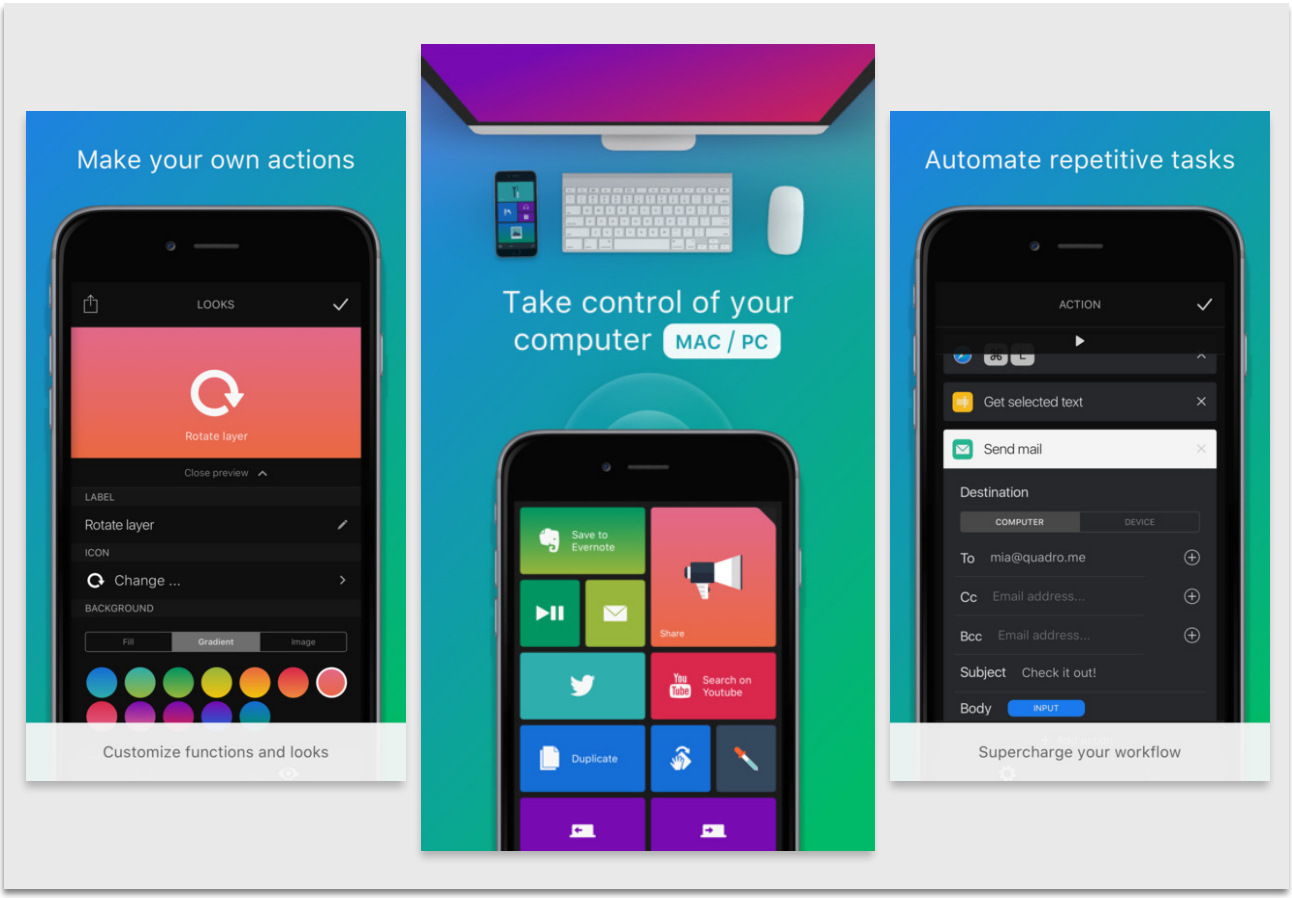
The main challenges here were connected to the application flow, which can be rather complicated for particular scenarios. I had the chance to play with custom view controller transitions, in order to re-implement the base navigation controller animations. This is the first application in which I decided to go 100% with Auto Layout (except for some minor views) and… once you're in, you never leave!

Quadro 2015

Quadro

A smart controller for your MAC and WINDOWS computer

Available on the App Store (Actions S.r.l.)



My responsibilities

My contributions to this project were mainly focused on the UI layer of the application. The project was already in a quite advanced development state when I joined the team working on it, but it needed a series of additions in terms of UI, especially on the main command interface of the app (the so-called "palette"), for which I had the chance to suggest and adopt PaintCode as a development tool: it was crucial in order to implement custom-shaped controls with the ability to change their size and shape dynamically. I also worked on the main controls animations, for tasks as selecting/triggering/sorting/resizing the palette action components. I also had to implement a vertical card notification system, with an API to be used across the app, for which I provided complete documentation for the whole team. Finally, I developed various tutorials for the application, both as an intro presentation and as contextual hints using coaching marks.

What I used

IDE: Xcode

Source control: Git

Relevant frameworks: UIKit, Foundation, CoreAnimation, CoreGraphics

Development Tools: PaintCode

Collaboration Tools: Slack

Challenges

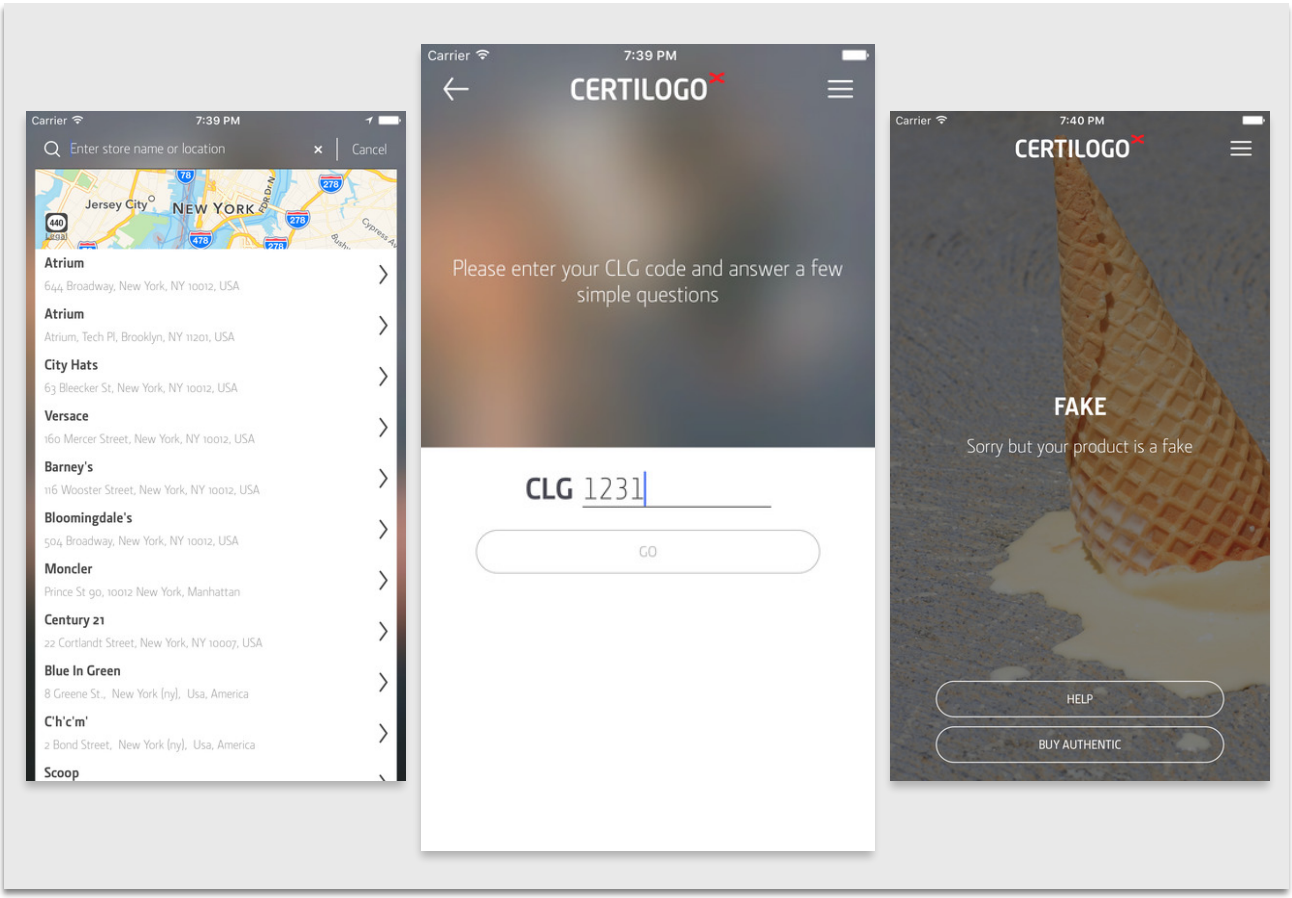
This was a huge project in terms of complexity and architecture, so the first challenge was being confident with the general infrastructure and the team. As for the development tasks, the main challenges concerned the heavy use of collection views and animations, and the development of a clean and simple API for the notification system, in order to allow the other developers to use it in an easy way. A ton of moving parts in this project, creating a solid dynamic UI without any bugs was a tough one.

Certilogo Auth 2.0 2015-2017

Certilogo Authenticator 2.0

A tool to verify the authenticity of commercial products

Available on the App Store (Certilogo S.p.A.)



My responsibilities

The owner of the project decided to re-design the whole platform, and this lead to the update of the original iOS application I previously worked on. Since there were major changes in the infrastructure (back-end, design guidelines, authentication flow specifications) we decided to start from scratch. As for version 1.0, I was in charge of the whole development process, and supported the design and back-end team during the re-design and implementation cycle.

What I used

IDE: Xcode

Source control: Git

Network: AFNetworking

Libraries: FBSDKLoginKit, GoogleSignin SDK, GoogleMAps CCHMapClusterController, CCHLinkTextView, CTAssetsPickerController

Dependency Manager: CocoaPods

Relevant frameworks: UIKit, Foundation, CoreLocation, MapKit

Collaboration Tools: Slack

Test management: TestFlight

Crash report management: Fabric

Challenges

The main challenges for this project concerned the implementation of a series of custom UI controls and general flow flexibility/theming solutions, in order to be able to adopt different application visual styles and new proudct authentication flows, if necessary. The whole application design process was focused on the possibility of creating new application targets in an easy way, to be able to develop custom-branded children applications for different customers, in the future.