

Ilya Belyavskiy

MAP-BASED IOS APPLICATION
DEVELOPMENT USING ARCGIS
RUNTIME SDK

Bachelor's Thesis
Information Technology

May 2015



DESCRIPTION

		Date of the bachelor's thesis 8 May 2015
Author(s) Ilya Belyavskiy	Degree programme and option Information Technology	
Name of the bachelor's thesis Map-based iOS application development using ArcGIS Runtime SDK		
Abstract The aim of this bachelor's thesis was to get familiar with the process of development of a map-based mobile phone application for iOS and to create a solution to a real life demand. Nowadays, modern gadgets play an important role in our everyday life. There are hundreds of applications available on Application Stores for different operating systems. This allows using these gadgets for varieties of different purposes. One the most interesting and rapidly developed trends in mobile application development is related to maps. Such features, as tracking location, navigation and location sharing combined with gadgets' portability and high productivity improve our lives. This study is about creating such application for a certain target audience – for Fishermen, who visit or live in Finland. Its main function is to show user's location on maps and to help him/her with catching fish by finding depth differences of the lakes' or rivers' bottoms. Such differences are places, where fish prefers to stay. This application was created using ArcGIS product line and iOS development tools. The result of this project is a functional demo of application called FishinGo, which performs its main task well. Currently, it still needs improvements. For example, it does not cover whole Finland, but only a part of it. This application will be used later as a platform for the future development of this idea and will be released to the App Store afterwards.		
Subject headings, (keywords) iOS, ArcGIS, software development, programming, GIS, application, map-based, depth data		
Pages 46p. + 5 p. appendices	Language English	URN
Remarks, notes on appendices		
Tutor Esa Hannus	Employer of the bachelor's thesis My Own Startup	

CONTENTS

1	INTRODUCTION.....	1
2	METHODS AND TOOLS	2
3	IOS DEVELOPMENT AND GIS	4
3.1	iOS Application Development	6
3.1.1	Objective-C	8
3.1.2	Xcode Project.....	9
3.2	ArcGIS.....	12
3.3	Map data.....	12
3.3.1	Data sources	13
3.3.2	Data licensing.....	13
3.3.3	Data format.....	14
4	FISHINGO’S CONCEPT	14
4.1	Purpose.....	15
4.2	Project requirements.....	15
4.3	Competitors	15
4.4	Market and Monetization	17
4.5	Target Audience	18
4.6	Name	18
5	FISHINGO DEVELOPMENT	18
5.1	Studying Objective-C	20
5.2	The first experience.....	21
5.3	Editing maps.....	24
5.4	Challenges	25
5.5	Design.....	27
5.6	Storyboard	28
5.7	Development	29
5.7.1	Adding ArcGIS Runtime SDK to Xcode	30
5.7.2	Configuring Views and ViewControllers.....	31
5.7.3	Configuring Navigation.....	31
5.7.4	Connecting ArcGIS WebMap	33
5.7.5	Displaying Current User’s Location.....	35

5.7.6 Adding Search.....	36
5.7.7 Adding the «My Location» button	37
5.7.8 Configuring a Pop-up	38
5.7.9 Configuring Application’s Settings.....	40
5.8 Testing.....	43
6 CONCLUSION.....	44
7 BIBLIOGRAPHY	47

1 INTRODUCTION

Nowadays, technology is a very important part of our lives. We can feel its impact everywhere, in many different spheres. It helps us a lot – we communicate, save our personal time or even get motivation, have fun, and much more. A great step forward was done with the rapid development of mobile gadgets during the last seven years. Nowadays we can do such things which were hard to believe several years ago: see which and when public transport is coming, where best spots or places to have a dinner are according to the social opinion or reviews, track and measure our personal physical activity and health conditions and many more. One of the latest trends in IT is the usage of GIS data for different, sometimes even unpredictable, purposes. One of the best application development studios in Saint-Petersburg made a statement that this is the most popular kind of applications among their customers' orders. My thesis is about one of them – map-based mobile phone application showing the depth of lakes and rivers in Finland.

This thesis combines two interesting spheres of IT, GIS and programming. The aim of this thesis is to get familiar with the development process of a map-based mobile phone application. This application is called FishinGo and its purpose is to show depth maps on the screen of a smartphone. The target device operating system is iOS by Apple, because of several reasons. Firstly, I had all the required equipment. Secondly, iOS users download much more applications and pay for them than users of any other operating systems, and the final reason is my personal preference.

This thesis is about the whole development process of a mobile device application – from an idea, or a solution, which came from a real life problem, to getting familiar with the necessary tools for its realization and almost completing its release to the App Store.

During this thesis I will study:

- learning Objective-C from, probably not 0, but something close to that.
- the ways of implementing the GIS data in mobile applications nowadays.
- collecting the required data from the Finnish language sources.
- learning the differences between GIS data.

- iOS application development.
- ArcGIS product line.

The way I decided to complete the case is, first of all, by collecting data from Finnish sources like Finnish transport agency, Liikennevirasto, LAPIO and others. I also needed to get used to the iOS mobile development as well. After choosing the proper tools the data had to be prepared to the required formats, uploaded to ArcGIS online, and finally, the results had to be connected to an iOS application using the ArcGIS Framework for iOS, upon which the whole application with different features is built.

Further in the thesis I am going to describe the tools to be used and my preparations such as concept creation and design development. Secondly I explain the most important theory points of different GIS data types and kinds of solutions. Then, I will introduce some basics of simple iOS application development. Of course, I will emphasize using the ArcGIS Framework as some kind of “core” of the project and describe what and how I did there. Finally, I will mention why I did not decide to rush and try to release it to the App Store at the moment of graduation. I will also explain why I would like to continue this project and how I am even planning to get profit from it.

2 METHODS AND TOOLS

To develop an iOS application for the first time, a study needs be done in order to find at least some information for the starting point. The topic needs to be studied well. Very many kinds of approaches of information gathering will be used in this thesis - from reading related articles, basic how-to guides, tutorials by Apple and video lessons, to reading books about programming and even joining the famous free web course CS 193P iPhone Application Development by Stanford University (Wenderlich 2010, Tekritisoftware 2013, Solt 2013, Stanford 2011).

It is extremely important to collect background information about the requirements for the hardware and software setup for such a project. The environment must be set up in the right way. First and most important of all, any modern computer by Apple running the latest OS X

version will be required. Comparing to other environments for development, like Android, the process of getting ready for the development is not as easy as in the current case with Apple for iOS.

This is the list of the basic software that is needed during this project:

- the latest Xcode version (currently 6.1) on a MacBook Pro with the latest version of OSX up to the moment of the development (OS X Yosemite version 10.10.3),
- additional code editor - Sublime Text 2 (optional, any other alternative may be used),
- Adobe Photoshop CS6 or any other image editor for prototyping and UI/UX planning and implementation,
- ArcGIS for Desktop (latest version - 10.2.2) and ArcGIS Runtime SDK for iOS (current - 10.2.4).

Secondly, it is required to learn the main principles of Objective-C and Cocoa Touch. Though my experience includes a .NET programming course at Mamk and C++ on additional programming courses at high school before, most of the important aspects had to be revised.

During the planning phase, Adobe Photoshop will be used for creating prototypes of the screenshots of the future application. This is usually done to create a storyboard of a future application, to understand possible user actions and to know what will be implemented. Storyboard is a map of an application. It is very important to understand all the interactions between the elements in an application for a developer. Also, the design elements are usually prepared for the developer as well, when custom, user interface elements are cut from the prototypes .psd files and saved as .png images to replace stock appearance.

The application is going to be created using Xcode. It is a fundamental and required toolset for building apps which combines almost everything needed for the development of an iOS or OS X application. It is impossible to test, debug or sync applications to a local developer's device like iPhone without Xcode. Interface Builder is an editor which allows designing user interface without using any code. It contains an Object Library which contains different UI elements, such as windows, buttons, text fields and many others. One of the greatest features of Xcode is the Assistant, which allows to make easy connection of UI elements to the code. To view and test the results, the iOS simulator is used. It comes with Xcode, so no additional

steps and actions are required for using it. Since the Xcode version 6.0, the Swift programming language was introduced. I also wanted to try Swift very much, mainly because it is a bit more simple and comfortable programming language to start with. However, it is still a currently under development, and it is much more problematic to find related tutorials, guides or books about it, mainly because it is very new. Also, Objective-C is much more universal. My opinion is that learning at least its basics will be better for getting familiar with Swift as well.

The ArcGIS product family will be used for everything related to the maps in this project. ArcGIS for Desktop package will be used for editing and manipulating the gathered data. The gathered data will be uploaded to ArcGIS Online and set up in a proper way there as well. ArcGIS Runtime for iOS SDK will be installed and integrated into the application. It will be used to display the prepared map from the ArcGIS online.

3 IOS DEVELOPMENT AND GIS

Mobile gadgets are the technology that was developed rapidly during the last ten years. Everyone knows that smartphones can easily replace computers for performing most of the tasks and can even do even more because of their compact sizes. This is also their huge advantage. They are comfortable for keeping and using them everywhere and anywhere. In my opinion, the most comfortable and rapidly developed operating system for smartphones and tablets is iOS. It was created by Apple and presented on June 2007, and was named as “iPhone OS” until June 2010 (Wikipedia). Nowadays, millions of iPhones and iPads are running this operating system. One of the main advantages is the number of high quality applications created for it. This happened, because Apple created all the comfortable conditions for the developers from the beginning by releasing the iOS SDK in 2008. Secondly, Apple has strict guidelines for releasing applications to the App Store, which makes their quality higher than in competitors’ application stores. GIS means Geographic Information System. Though maps existed for thousands of years, the term GIS was invented only in 1962, when Canada Land Inventory (CLI) invited Roger Tomlinson to define functional requirements of what would later be called the Canada Geographic Information System (ArcGIS 2012). In 1965 the first Laboratory was founded by Howard Fisher at Harvard University (Chrisman 2005). GIS is any information system which combines almost everything related to geographic data, such as

creating, displaying and editing maps, analysis and statistics. There are different kinds of GIS applications, based on their working principle. Because there are many of the, certain Open Geospatial Consortium standards were introduced since 1994 (Open Geospatial Consortium 2013). These standards are responsible for certification of different GIS products working principles. Communication scheme of communication between GIS tools certified by OGS standards is represented on FIGURE 1.

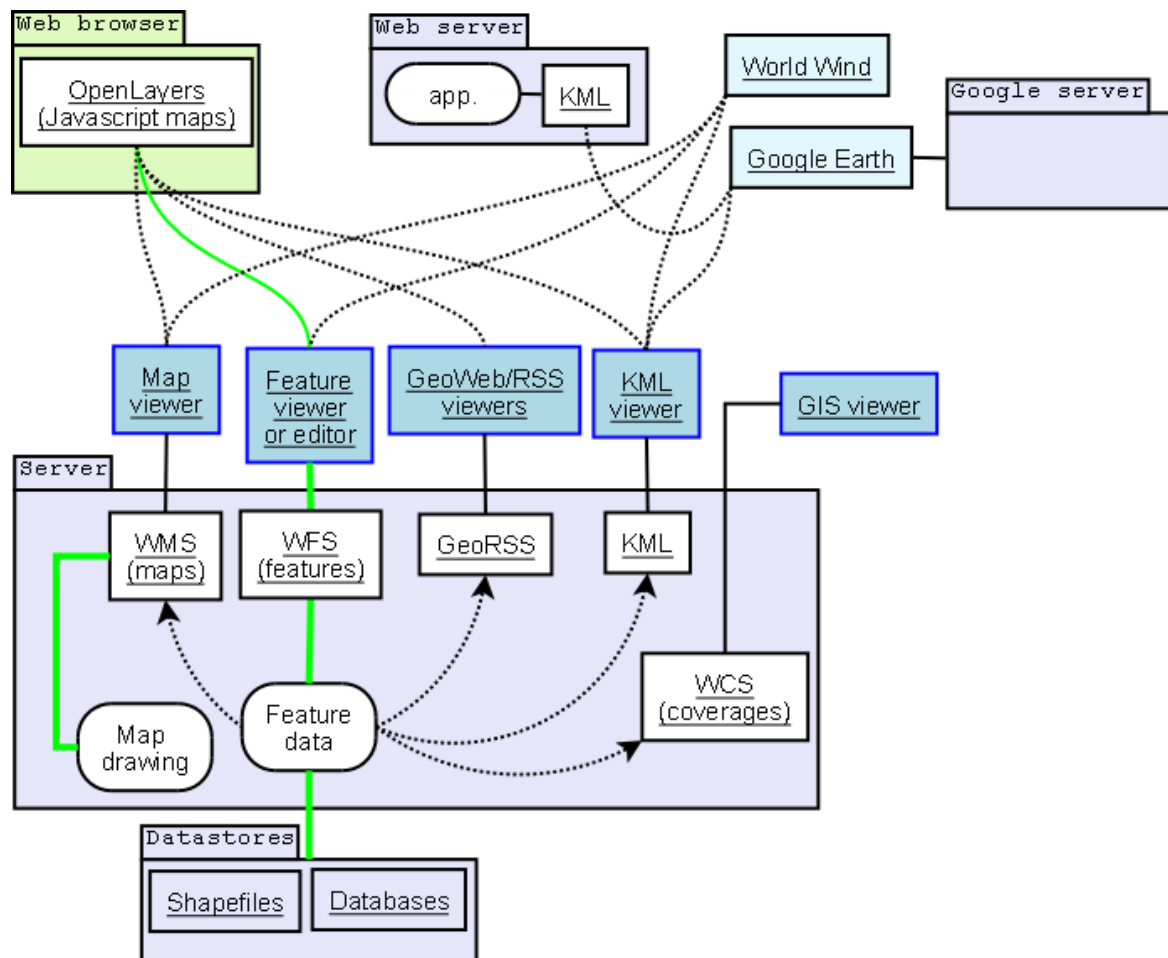


FIGURE 1. Different ways of communication and interaction between GIS services

Many unique and useful GIS related applications are created nowadays for our mobile gadgets. This thesis is about creating such application. That is why it is related to both of this different, but interesting spheres of Information Technology.

3.1 iOS Application Development

One of the greatest Apple's advantages against other platforms has always been its huge number of developers and applications created. There are more than 380 000 members of Apple's paid developer program nowadays, there were 1.2 million applications in the App Store and \$10 billion were earned by developers in 2014 (Apple 2014, Perez 2014). Another advantage is that though there are different iOS versions on the market, the fragmentation is very low. According to the latest statistics available (FIGURE 2), more than 81% of devices from Apple are running the latest operating system version - iOS 8.

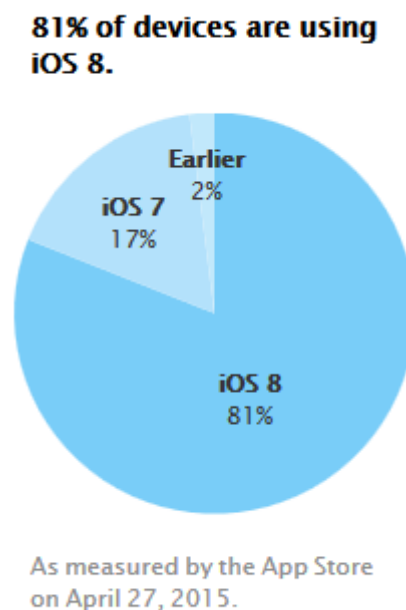


FIGURE 2. iOS fragmentation

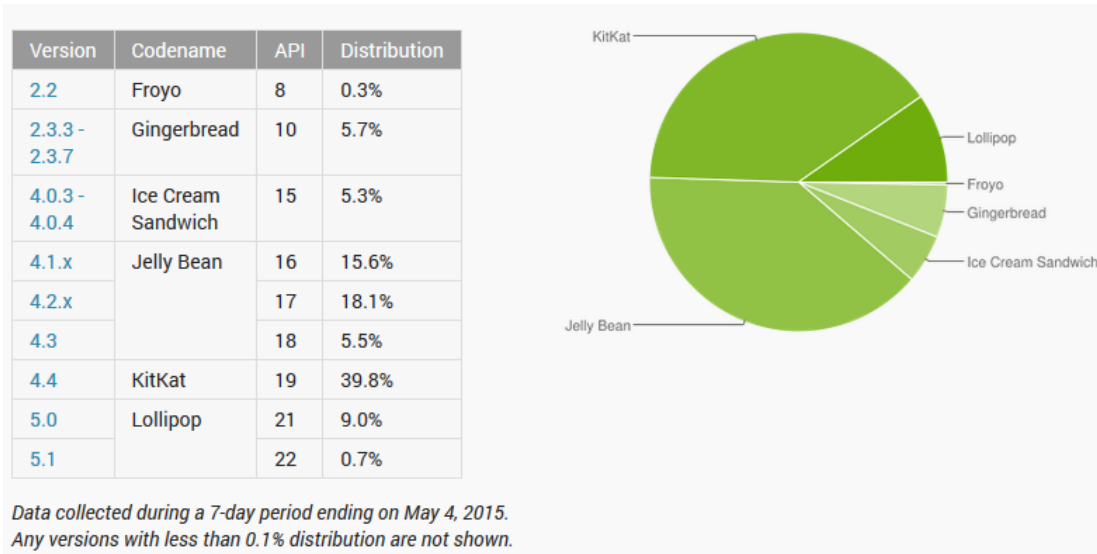


FIGURE 3. Android fragmentation

This is an incredible result for Apple, because there is no such competitor with results even close to that. Fragmentation may be a real problem, because newer operating systems support more features. Developers need to follow the guidelines of each mobile operating system version to make their applications work well and backwards compatible. Because Apple does not share its mobile operating system with third party manufacturers, there is no serious screen sizes fragmentation as well. According to the iOS Design Guidelines, there are only three resolution standards in iOS (Mynttinen 2014). This makes designers and developers lives much easier, comparing to the Android case. According to the research “The many faces of a little green robot”, the situation is totally opposite with Android (OpenSignal 2012). This can be easily noticed on FIGURE 4.

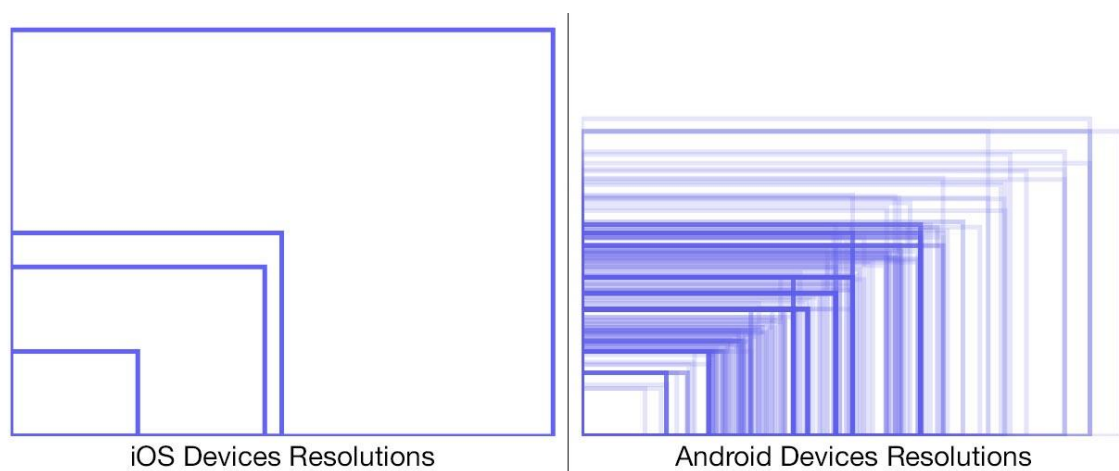


FIGURE 4. Comparison of iOS and Android devices resolutions fragmentation

Because iOS has many advantages like these mentioned above and because I have got all required hardware and software equipment, a decision was made to develop an application for Apple's mobile operating system first. I found out that there are two main programming languages available: the recently introduced by Apple called Swift, and the older, but much more popular and similar to other C languages – Objective C (Mac Developer Library 2014, 2015).

3.1.1 Objective-C

“Objective-C is a programming language that blends C's speed and ubiquity with an elegant object-oriented environment... It is the gateway drug for many of Apple's niftiest technologies, such as the Cocoa toolkit and the iPhone SDK. Once you've mastered the Objective-C language, you're well on your way to conquering the rest of the platform. And from there, you can try to take over the world”. (Dalrymple Mark, Knaster Scott, Preface). In other words, learning Objective-C is like obtaining a key from the world of programming for OS X and iOS. Actually, Objective-C is just a kind of a layer on top of C with several syntax and semantic features, based on Smalltalk which had been developed by NeXT, for object-oriented programming support (Wikipedia).

When speaking of developing iOS applications, it is impossible not to mention the Cocoa/Cocoa Touch toolkits. Written in Objective-C, they contain different elements of UI, gestures, basic forms and much more. If iOS is considered as a set of layers, then Cocoa is on the top (FIGURE 5). It is responsible for supporting developed applications and contains quite many important and useful features and frameworks, starting with view controllers, notifications and gesture recognizers and ending with MessageUI, Game Kit, Map Kit, iAd and UI Kit frameworks. Cocoa is also the primary application environment for the OS X and the only one for the iOS. It consists of a number of object-oriented libraries, runtime development environments. This is why many OS X apps and, of course, all the iOS apps are/were created using Cocoa. One of the reasons is the strict Apple's guidelines for releasing applications for its operating systems. The Media layer provides Cocoa Touch with multimedia abilities and includes Core Graphics, Core Text and OpenGL ES. The Core Services layer contains fundamental services, like settings, hardware features, like accelerometer and gyroscope, compass and GPS. Its frameworks include Core Location and Core Motion, which are very important

for this particular thesis. And the last layer, Core OS, is the core itself, the file system, network infrastructure, OS security and drivers support. (Mac Developer Library 2013)

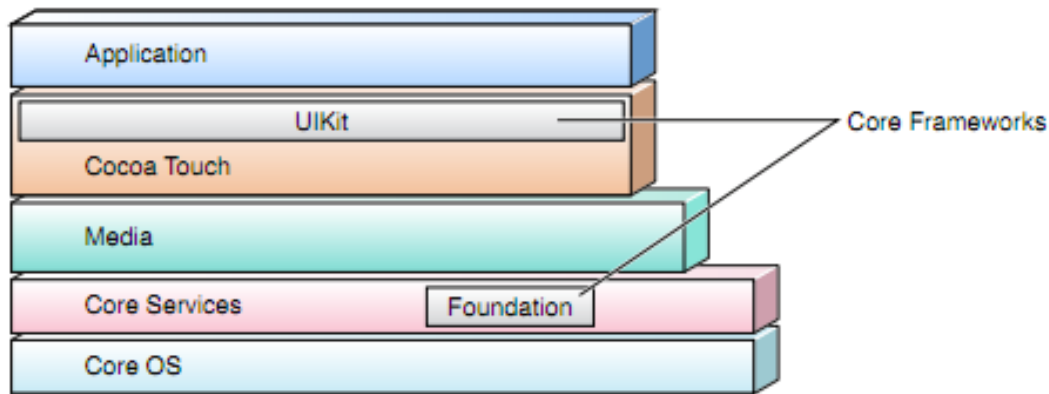


FIGURE 5. Cocoa in the architecture of iOS

3.1.2 Xcode Project

Xcode is a toolset used to create, develop, modify and test iOS and OS X applications. An Xcode project is a collection of files and settings needed to construct the application. To create a new project, after launching Xcode it is required to choose File -> New -> Project. There, it is possible to choose already existing and basic templates, give a project a proper name, set the company identifier, etc. It is important not to forget that the Devices pop-up menu is set to the iPhone. When created, it is already possible to run an empty application in a simulator. To run a project the created application has to be launched in the iOS Simulator or synced with a connected device, if there is a developer license. To build a project is to compile the code and assemble it with various resources into the actual app.

Each class contains two files: the header and the implementation (method). This is why the contents of any created iOS project contain the files shown on FIGURE 6.

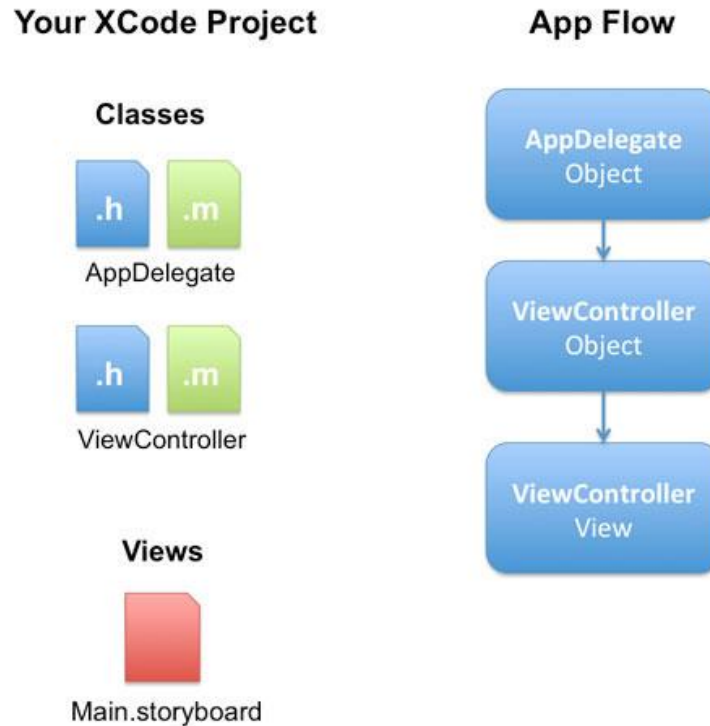


FIGURE 6. XCode Project Files

AppDelegate – a class which exists in each iOS application. It is responsible for reaction to various global events which are sent by the operating system to an application, for example low memory warning, when the user wants to quit the app, etc.

ViewController – a class which is responsible for controlling and managing the View.

Main.storyboard – view managed by the ViewController class. A View is a graphical representation of the iOS application. Actually, Views may be created in two different ways:

- Interface Builder – a visual application editor, which allows the arrangement of views into hierarchies, Views' setting configuration and connect different actions with the code. The Interface Builder is represented in the project by a .nib file and is the storyboard of the project.
- Programmatically - the default initialization method for views is the initWithFrame method which sets the initial size and position of the view relative to its parent view. (iOS Developer Library 2014)

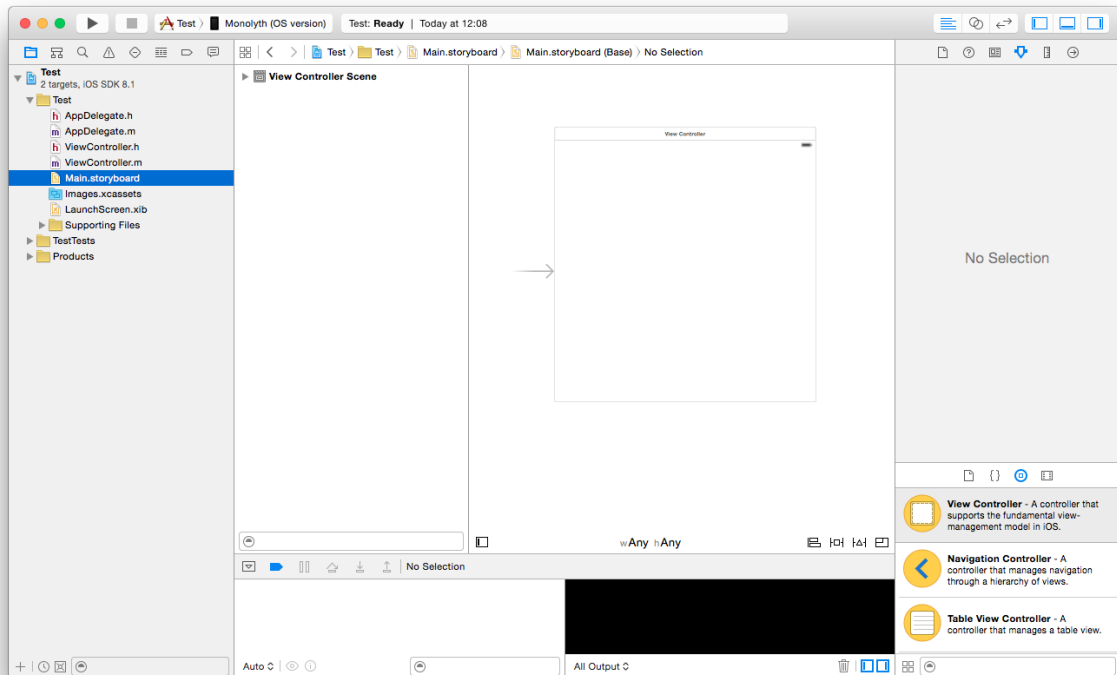


FIGURE 7. The project window

Xcode's interface, shown on FIGURE 7, consists of four main areas:

1. The Navigator pane is on the left. It contains eight tabs – the Project navigator, the Symbol navigator, the Find navigator, The Issue navigator, the Test navigator, the Debug navigator, the Breakpoint navigator and the Report navigator. Most of them are used for testing an application and fixing errors and bugs which occur during testing.
2. In the middle there is the editor where all possible interactions with the code can be implemented. It depends a lot on the file, chosen in the Project navigator. It may show the file's code, when a class file is selected, and the Interface Builder, when the storyboard is selected.
3. The Utilities pane is on the right. This is a very important pane which contains the following tabs – the File inspector, the Quick Help inspector, the Identity inspector, the Attributes inspector, the Size inspector and the Connections inspector.
4. The Debugger pane is at the bottom.

Because opening all panes at once leaves not much space to work with during coding, there are quite useful buttons on the top right of the project window, which allow hiding or showing unnecessary panes.

3.2 ArcGIS

There are quite many different ways and different map services on the mobile market today. The most popular map application on smartphones and tablets nowadays is Google Maps (Wikipedia 2015). Others may be popular in specific countries, like Yandex.Maps in Russia (Ghedin 2013). Some map services are native for some operating systems, like Apple Maps and created applications with Apple MapKit are for iOS. But there are such services, or ecosystems, which allow much more possibilities for companies or users, and which are more professionally oriented or experienced, shortly – more advanced. One of the providers of such advanced mapping services is the ArcGIS product line by ESRI, the market share of which was 40% in 2010 (Wikipedia 2015). Environmental Systems Research Institute (ESRI) was founded in 1969 by Jack Dangermond and his wife Laura (Esri). When in 1981 it held the first International User's Conference, there were only 16 attendees. In 2012 there were around 15 000 visitors. It is the world's largest event dedicated to geographic information system technology. Nowadays it has more than 300 000 customers from all around the world. (Howell 2009). There are two main product lines by ArcGIS, the desktop and the server-based. The desktop edition, which was used for editing maps during this thesis, includes such tools as ArcMap, ArcScene, ArcGlobe and ArcCatalog. Finally, the most important ArcGIS product during this thesis was a part of ArcGIS mobile, called ArcGIS Runtime SDK for iOS.

3.3 Map data

One of the most important tasks in this thesis was gathering the depth maps data. Without these maps achieving any results would not be possible. Map data can be represented in different ways. There are two basic types of GIS data: spatial and attribute. Spatial data describes the absolute and relative location of geographic features. Attribute data describes characteristics of the spatial features which can be quantitative or qualitative in nature. Such data is often referred to as tabular data (Buckey). GIS data may also be divided into two other types: raster (grid based) and vector (coordinate based). The type of data, suitable for this thesis was vector data. It represents point, lines and polygons. Data about Finnish lakes of rivers with depths lines had to be collected in shapefiles and added as layers. Vector data may also contain multipoints, annotations, dimensions, etc. Raster data is much bigger in size, because it is divided

into cells and identified by rows and columns. The size of the cells depends on the data accuracy and the resolution required. Its attributes are also limited, comparing to the vector data type (Reiser 2011). The more I was looking for the content, the more questions appeared about sources, licensing and formats.

3.3.1 Data sources

There are three companies which own practically all the depth-related data of Finland:

- Finnish Environment Institute (SYKE Latauspalvelu LAPIO)
- Finnish Transport Agency (Liikennevirasto)
- National Land Survey of Finland (National Land Survey of Finland)

These companies own different information about depths of lakes, rivers and even seas in or nearby Finland. This is the reason, why several sources of data will be used in this thesis simultaneously and why there will be some uncovered areas without any depth information at all. However, only two primary sources of information were used. The first one is the Finnish Environment Institute. It is a research institute and a centre for environmental expertise. It has a service, called Latauspalvelu LAPIO which allows downloading different kinds of data. The second data source is the Finnish Transport Agency. It is an expert organization specializing in transport, operated by jurisdiction of the Ministry of Transport and Communications of Finland (Finnish Transport Agency 2015). Its map data and file download service is called Liikennevirasto. Unfortunately, not all Finnish lakes are covered in the created application, but this is explained in details later in this thesis.

3.3.2 Data licensing

During my search of the Syvyyskartta data I was informed by Mr Mika Ahvenainen from the Finnish Transport Agency as following: “The depth information is available at Latauspalvelu, but it is licensed so that it is not allowed to be used on navigation applications” (Ahvenainen 2015). This is also mentioned in the right of use section (2.1) of the Finnish Transport Agency’s hydrographic data license. It states, that navigational purposes refer to all maritime operations, such as route planning, positioning, determining the best direction and ensuring the vessel’s safe passage. (Finnish Transport Agency 2014). In order to avoid any problems, related to

such question, the user has to be acknowledged about that forbiddance. The solution of this licensing moment is explained later in this thesis.

3.3.3 Data format

The ArcGIS Runtime SDK for iOS supports long lists of different Geographic coordinate systems and Projected coordinate systems (ArcGIS Runtime SDK for iOS 2015). Geographic Coordinate System uses a three-dimensional spherical surface to define locations on the Earth, which are defined by longitude and latitude values (Geonet 2011). Projected Coordinate System refers to data that is defined by a flat 2D surface and can be measured in meters and feet (ArcGIS Resource Center 2013). The coordinate system chosen to be used is the universal standard for the entire world – WGS84, or GCS_WGS_1984 (Wikipedia). However, the downloaded data from the Finnish providers was in a different format, so each layer had to be converted from ETRS89 to WGS84. The difference between these formats is only 42 cm (Institut Cartogràfic i Geològic de Catalunya). ETRS89 is the latest pan-European coordinate system, used in Finland, and is replacing the KKK coordinate system since 2005 (Uikkanen 2014).

4 FISHINGO'S CONCEPT

There is a variety of different ways of creating a map-based application nowadays. There are different frameworks, different methods and different ways of implementation. Firstly, maps can be either included into an application, making it huge and heavy, or they can be downloaded while using an application, and its size would be smaller, but such application would be totally useless without any internet connection. There is also a mixture of the two previous types of applications. The maps can be downloaded and stored in cache on the device's memory, but would still require the Internet access for such features as locations search. My map-based application is going to be of this third type. Unfortunately, this is not covered in this thesis, because it requires some serious changes to the framework itself. This is one of features I would like to postpone for the future development.

4.1 Purpose

The purpose of any application is to solve a real life problem and improve something. The problem in this case is time demand for finding a great fishing spot. FishinGo increases the chances of catching fish resulting in decreasing the probability of not catching anything resulting in decreasing the probability of getting unsatisfied. In other words, this iOS application may simply save precious time. Its core appeal is to show the depth of rivers and lakes in order to allow fishermen to choose better spots with higher chances of catching somethings. Also, FishinGo will improve the competitive spirit of fishing. Fishing as a hobby is a part of life which will be improved by decreasing the amount of time required to get satisfaction.

4.2 Project requirements

The technical specification of this iOS application was created, describing what it is going to do/show/etc., mentioning all the possible user actions with it. An application must combine then:

1. Show depth maps of lakes and rivers.
2. Show current location of a user on a basemap with depth maps as layers (using ArcGIS Runtime SDK for iOS) using A-GPS/GPS.
3. Store maps of depths locally on the device in order to provide usage without the Internet connection (kept for the future development).
4. Contain small encyclopedia about the kinds of fish, ways of fishing and fishing techniques (optional, for future development).
5. Contain useful information about nearby infrastructure related to fishing, such as fishing equipment stores, gas stations, places to purchase fishing licenses.

4.3 Competitors

Research for possible competitors and existing ways of implementation of similar ideas had to be done in order to define if there was some target audience at least somewhere in the world. Related and already released applications currently present on the market can be noticed in Table 1.

TABLE 1. Already existing applications (possible competitors)

Application name	Description
iFish USA (iTunes App Store 2013)	This application allows to search information about fishing in the United States of America, such as lakes, guides, weather conditions and forecasts. However, there is no such functionality as depth maps at all.
Fishidy (Fishidy 2015, iTunes App Store 2015)	Fishidy is a totally different story. It is a fishermen's social network which contains tons of useful information. Also, it allows to save your last fishing location and to browse selected by the community hot spot locations. It does not cover all the lakes in the United States of America, but 8 000 of them.
Fishing Hot Spots (iTunes App Store 2014)	This application is an attempt to create another fishermen community, by allowing its members to share the location of the "fishy" places.
Trimble – GPS Fish (Trimble GPS Fish 2015, iTunes App Store 2015)	Another great iOS and Android app which contains the database of around 6 900 lakes in the US. It allows to plan routes, record fishing tips, check weather forecasts and quite many features. Its maps also follow the WGS 84 standard. However, most features, including the depth maps, are for the "elite members" only.

To sum up, almost all currently available competitors' applications which involve fishing and depth maps as their primary topic are for the United States only. The demand exists and, lucky for me, this topic has not been developed well in Europe and Russia yet. Also, there are different Finnish services which provide depth maps and are somehow related to this. They are mentioned in Table 2.

TABLE 2. Already existing depth map online services in Finland (websites)

ProKalastus (PRO KALASTUS 2012)	This company is a fishing goods importer and retailer. It also specializes in organizing high-level fishing trips. There is a depth map on the company's website made with Paikkatietoikkuna.
---------------------------------	---

RetkiKartta (METSÄHALLITUS 2014)	This service is provided by Metsähallitus for hunting and fishing areas. It has great detailed depth maps, as demonstrated in FIGURE 8 below.
Lipas (LIPAS 2013)	This is the official website of Finnish sport facilities. It provides quite useful information about different services, like buying licenses and renting boats.



FIGURE 8. Depth maps (Syvyyskartta) by RetkiKartta

These website owners were contacted by me by email with questions about their services and data origin. Almost all of them mentioned Liikennevirasto as the primary source of the depth maps data and wished good luck with the project FishinGo. This helped me to focus on getting the depth related information from the most popular data provider among these services.

4.4 Market and Monetization

From the beginning I had decided that my app will be free of charge. The freemium model and in-app purchases are becoming much more profitable and popular nowadays (Jones 2013). On

one hand, I decided to follow the trend, but on the other, my primary interest is getting the experience.

However, there are other possible ways of monetization as well, like selling maps for different countries or regions as in-app purchases. Also, advertisements as a small thin banner at the bottom of the application or on the settings page could be shown to users, which would bring profit from showing ads or removing them by an in-app purchase function.

4.5 Target Audience

FishinGo can be used by people from different countries, professional fishermen and amateurs, tourists and local people. There are quite many resorts and places in Finland which either sell fishing licenses or even keep population of specific kinds of fish in some lakes and provide incredible number of services from teaching how to fish to renting a whole resort with lakes and houses or facilities on its territory. Many tourists enjoy spending time this way. For example, I personally know quite many businessmen, who travel to Finland on weekends or on holidays just to relax from everyday duties, routine, city mess and to have a rest and fun with Fishing. My father is one of them.

4.6 Name

As for the name, I chose FishinGo – depth map of lakes and rivers in Finland. The name consists of two parts: the actual name of the application and the additional keywords description which would make searching for this application much easier on the App Store. Obviously, FishinGo consists of two separate words: Fishing + Go.

5 FISHINGO DEVELOPMENT

Nowadays, mobile application development is, probably, the most developed and interesting topic in the IT world. Hundreds of applications are released almost every day, giving ordinary people not only passion, job or an interesting hobby, but a chance to change their lives completely and to become extremely rich and famous all over the world. Huge amounts of money

are involved, huge companies are buying smaller ones or startups for millions, sometimes even for billions of dollars in order not only to stay on the top, but just simply to stay alive, because sometimes these startups may become their biggest competitors and reasons of bankruptcy. Moreover, I completely agree with the idea that programming is the language of the future, as technologies develop rapidly and change our lives incredibly fast almost every day, and all modern technologies need to be controlled with our natural language, telling them what exactly we want.

In fact, the current thesis topic is not my first project related to application development for iOS. I have been participating in another project called Guesspoint since 2012. It was started together with Dmitrii Viktorov, a graduate of Business Management Degree Program of Mikkel University of Applied Sciences. My role in the project is kind of a combination of a product owner, team leader, idea creator, and user interface and user experience designer. I am also the person who divides the philosophy, the main idea to smaller pieces, a much more detailed technical described project, and explains the hired specialists (programmers and designers) what is expected from them and how it should be realized. I wanted to get more technical experience, like programming, in order to understand what we are paying for, what is done and what is not, and probably to join the development as a programmer one day minimizing project costs that way, rising the product's quality and giving me a chance to implement exactly what I had planned.

I had several ideas for my own small project and I chose the obvious one for me: a mobile application for Fishermen which shows their current location on the lakes and rivers on the depth maps. To tell the truth the idea came from my father. His hobby is fishing and this is one of the reasons, why my family has been visiting Finland several times per year since, probably, when I was 7 years old. He asked me once, if there is any app for his iPhone which could display the depth maps of fishing lakes. He already used a web version of such maps, called Retkikartta, but it is not very stable and not compatible with mobile phones. And most importantly, it is barely not accessible during fishing, because the mobile Internet connection in nature is far from being good and it is pricey, especially for tourists, because of the carrier roaming.

5.1 Studying Objective-C

I have gone a long way of studying Objective-C programming since I started watching video tutorials called Teach me Xcode in Russian on YouTube (AppleInsider.ru 2013). However, the teaching method appeared to be more and more boring from one video to another. Moreover, the video course assumed previous knowledge of Objective-C, so I understood that I need to learn the basics of the programming language first. I was advised to try a collection of books, and I have read the following ones:

- Mark, David & Bucanek, James 2012. Learn C on the Mac: For OS X and iOS.
- Dalrymple, Mark, Knaster, Scott. 2012. Learn Objective-C on the Mac: For OS X and iOS.
- David, Mark, Nutting, Jack, LaMarche, Jeff & Olsson Fredrik 2013. Beginning iOS 6 Development.

These books are great and have a lot of useful theory. However, they were incredibly time demanding and sometimes not quite useful for this case because of several reasons:

- 1) It was quite hard to learn the basics without understanding how these basics relate to the particular project, questions and issues. More on a specific task was needed than on learning everything related to programming for iOS in general.
- 2) The Xcode version installed on MacBook Pro was much newer than mentioned in the book. That is why most of the practical code instructions were incompatible when attempts to implement them in practice were made. And unfortunately, downgrading Xcode was definitely not an option, because it would require downgrading OS X operating system, too.

5.2 The first experience

Practice is a great way to learn anything. At first, a simple iOS application with a built-in Apple MapKit was created. Useful experience was received. The new iOS 8 had changes that prevented showing the user's location without any pop-up alert (even when simulated in Xcode) by default, because of new security reasons. This is the solution:

“Beginning with iOS 8, in addition to calling `startDataSource` on the map's location display to show the device's location, you also need to add the `NSLocationWhenInUseUsageDescription` key to your app's `Info.plist` file along with the text to be displayed to the user when asking for authorization to access the device's location service.” (ArcGIS Runtime SDK for iOS 2015)

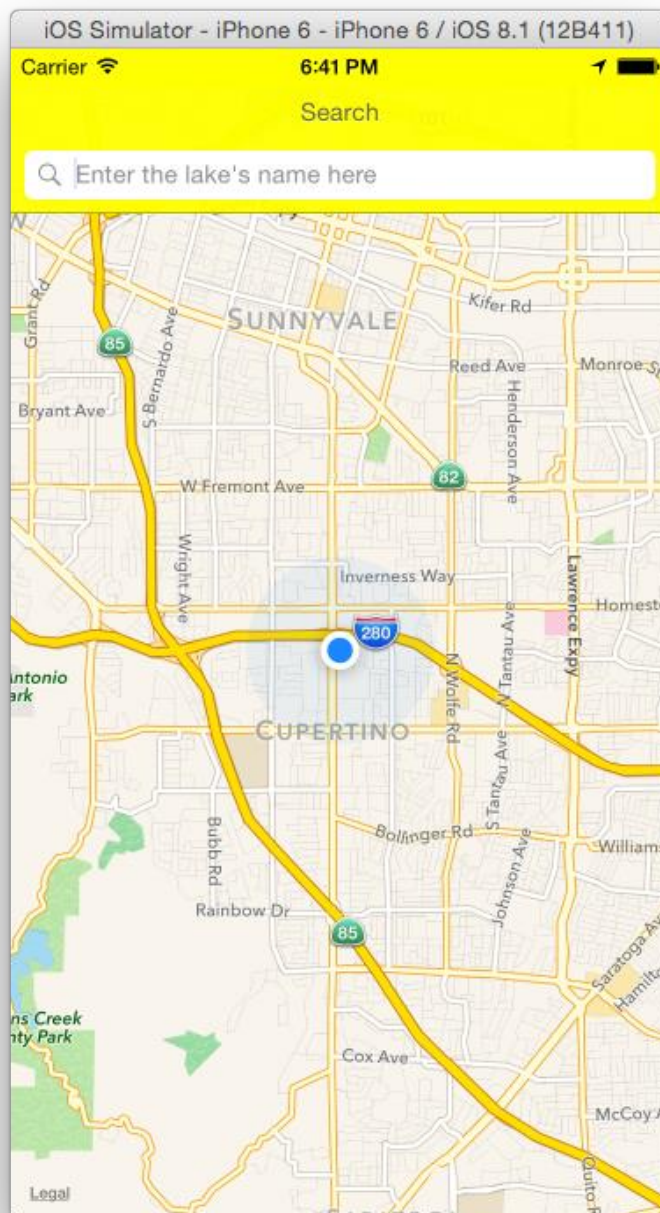


FIGURE 9. The first test iOS Application with Apple MapKit and working location

The next step was to add the basic map-related functionality, such as showing the user's current location by zooming into it when opening the application (Pigott 2013). Another self-study lesson was based on creating an automatically adjustable and universal user interface, depending on the portrait or landscape orientation of the device and on the device's screen

size as well. If the device is rotated, the squares' sizes are adjusted automatically and their location as well.

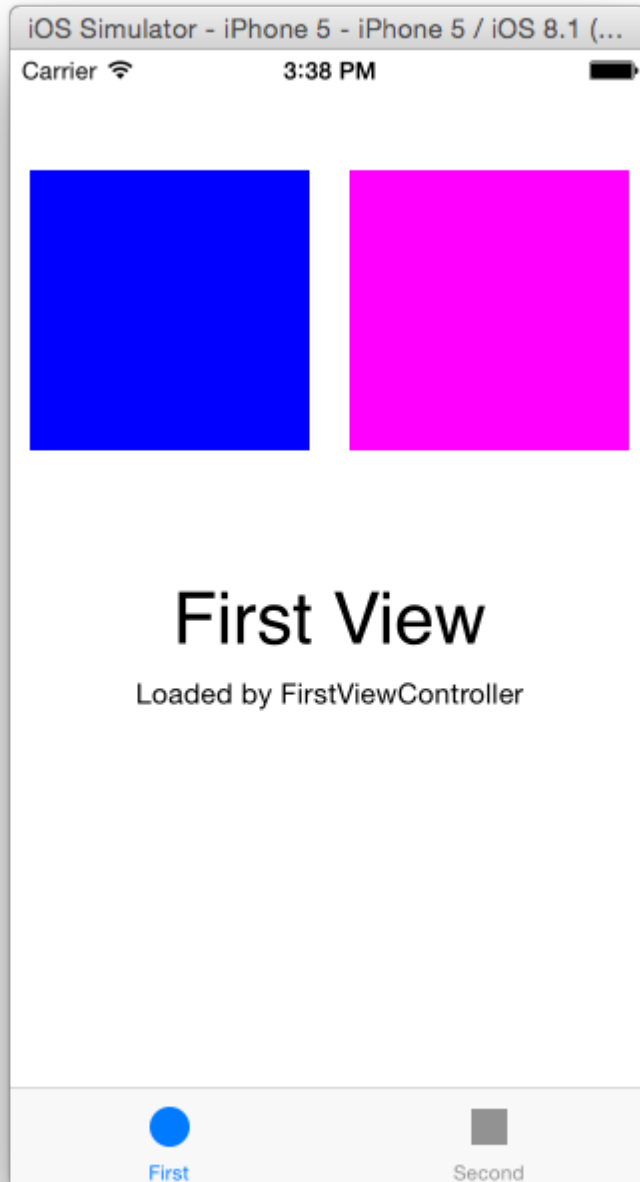


FIGURE 10. Test application with AutoLayout

One of the most important sources of information during the whole project was Apple Developer Library (iOS Developer Library 2015). It included a lot of simple explanations for im-

portant things, great examples, easy for implementation in practice. However, sometimes it was difficult to browse the website, especially when I needed to visit a more general article.

5.3 Editing maps

While trying to download the maps – which was a little difficult, because there is not any English version of the website at the moment of attempts – the following problem was faced: when the whole Finland was selected as a region for downloading, an empty .zip folder without any map-related data at all arrived to my inbox. Luckily, after contacting Mr Mika Ahvenainen, the Project Manager (Development of Information Services) at Finnish Transport Agency, I was advised to select a smaller region. That is one of the reasons why the application does not cover the whole country at the current stage. However, depth maps in FishinGo consist of two downloaded pieces of depth maps which cover different lakes from Latauspalvelu and OIVA. This means, that more areas of Finland can be added as layers at any moment.

During this project to view, edit and customize maps a GIS program ArcMap was used. It is the main component of the ArcGIS for Desktop products family and offers a wide range of functions for different tasks and can be used for different purposes. However, mostly the basic features were needed for the project's purposes. First of all, a basemap was required. It is a map, which shows background reference information such as landforms, roads, landmarks and political boundaries, onto which other thematic information is placed (ESRI GIS Dictionary). The most suitable for this project was, obviously, a topographic map. It is a type of map, characterized by large-scale detail and quantitative representation of terrain, usually using contour lines in modern mapping (GIS wiki 2011). To add it in ArcMap, going to Menu -> File -> Add Data -> Add Basemap is required and to select it from there. After that, it appears as the bottom layer in the project. To add the downloaded shapefile (.shp) elements from the Finnish GIS data providers to the project it was necessary to go to Menu -> File -> Add Data -> Add Data and to select a .shp file, one by one at the time. Immediately after that, a window suggesting data conversion appeared where the original ETRS89 format of the downloaded data was chosen to be converted into the universal GCS_WGS_1984.

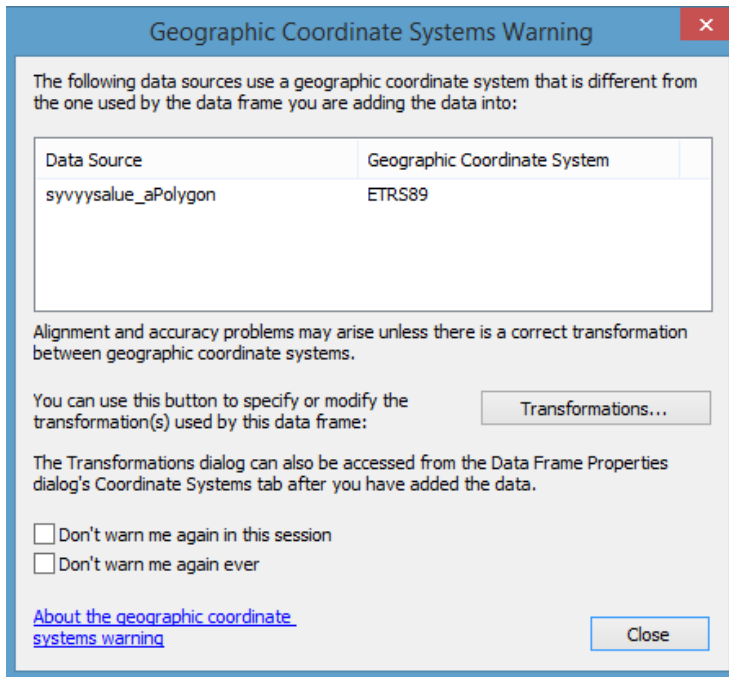


FIGURE 11. Converting GIS Data

Right after that, a new layer appeared on the table of contents on the left side in ArcMap. By clicking the layer with the right mouse button, all possible modifications could be done. To change the layer's appearance, a small coloured rectangle below layer's name had to be clicked twice.

5.4 Challenges

Unfortunately, during the editing processes I faced quite many limitations. One of them was quite low productivity of my Windows PC during editing maps. Sometimes, it took up to a minute to load all the layers and polygons. There were also challenges, related to ArcGIS Online, too. The first of them was maximum shapefile size limitation. It is demonstrated in FIGURE 12.

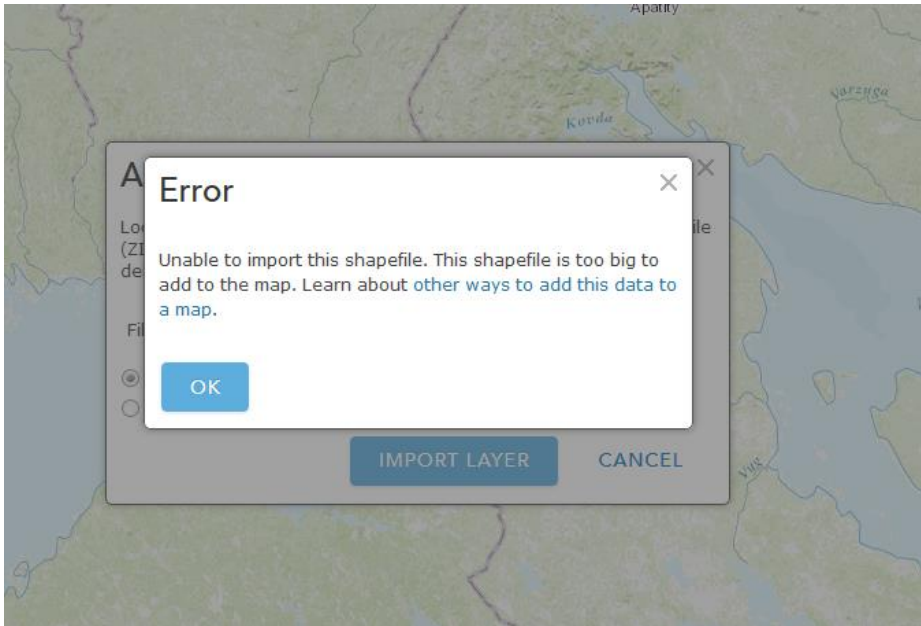


FIGURE 12. Shapefile size limitation

The second challenge – another limitation with the maximum number of features in a single shapefile is demonstrated in FIGURE 13.

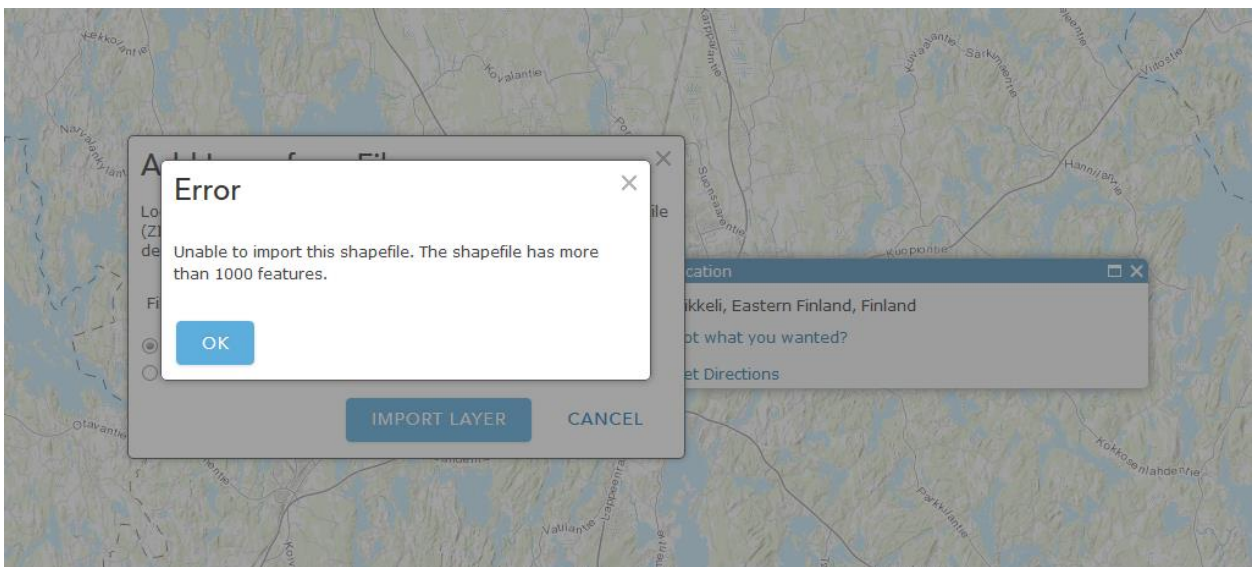


FIGURE 13. Shapefile features limitation

Because of these issues above, I decided to choose a smaller area of Finland to work with in this thesis and selected a small area around Mikkeli. Each layer was clipped, exported from ArcMap and uploaded to the ArcGIS Online where all design modifications to these layers were done. For example, different colours were assigned to different depths as follows:

- 0-1.5m - light blue
- 3-10m – bright blue
- 10-20m – darker blue
- below 20m – very dark blue

This did not work right in my first prototypes. That is why in some screenshots in my thesis the depth maps in Jyvaskyla area are in grey colours. This was an earlier version of my prototype. Fortunately, I figured out the proper and main attributes and everything worked fine.

During this thesis I used a free trial 60-day version. It ended just three days before this thesis seminar. Another profile to demonstrate my results was created and the prepared maps were uploaded there again.

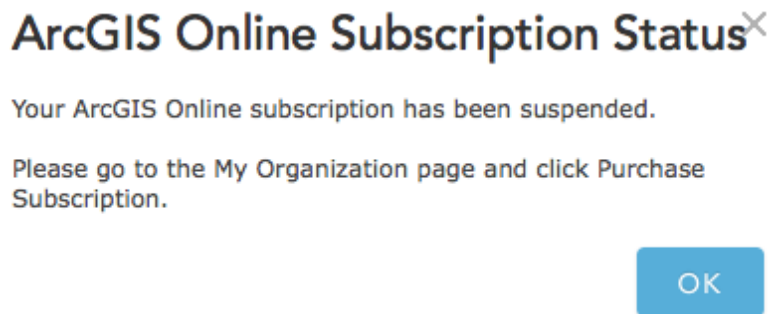


FIGURE 14. Notification about end of my trial period

5.5 Design

I decided not to spend too much attention on the design, but just to keep it according to the iOS 7/8 guidelines. These guidelines include smooth, thin default fonts, blurred tempered glass effect for the different elements and much more (iOS Developer Library 2014). For example, the navigation bar element is not grey – it is tempered glass styled. In other words, it is a bit transparent and blurred maps are displayed below.

The FishinGo’s icon was completely drawn by me using Photoshop. It represents a depth map in a form of fish. It was done according to the latest trend in flat design. The colour palette was taken from the depth maps in the application. There is also a version for the FishinGo’s

current stage of alpha development.



FIGURE 15. FishinGo Icon for the App Store release



FIGURE 16. Current icon with Alpha mark

5.6 Storyboard

After I made several so called “screenshots” of my prototype as images using Photoshop, I created a storyboard to demonstrate all the actions possible inside the application. A storyboard is a visual representation of the appearance of application. It helps to define navigation

and relations between ViewControllers, user experience and main classes in the project. In general, storyboards are used for planning and defining the conception of application. They can be done in differently from pen and paper sketches as the simplest to advanced mockups concepts with animation right on the target devices.

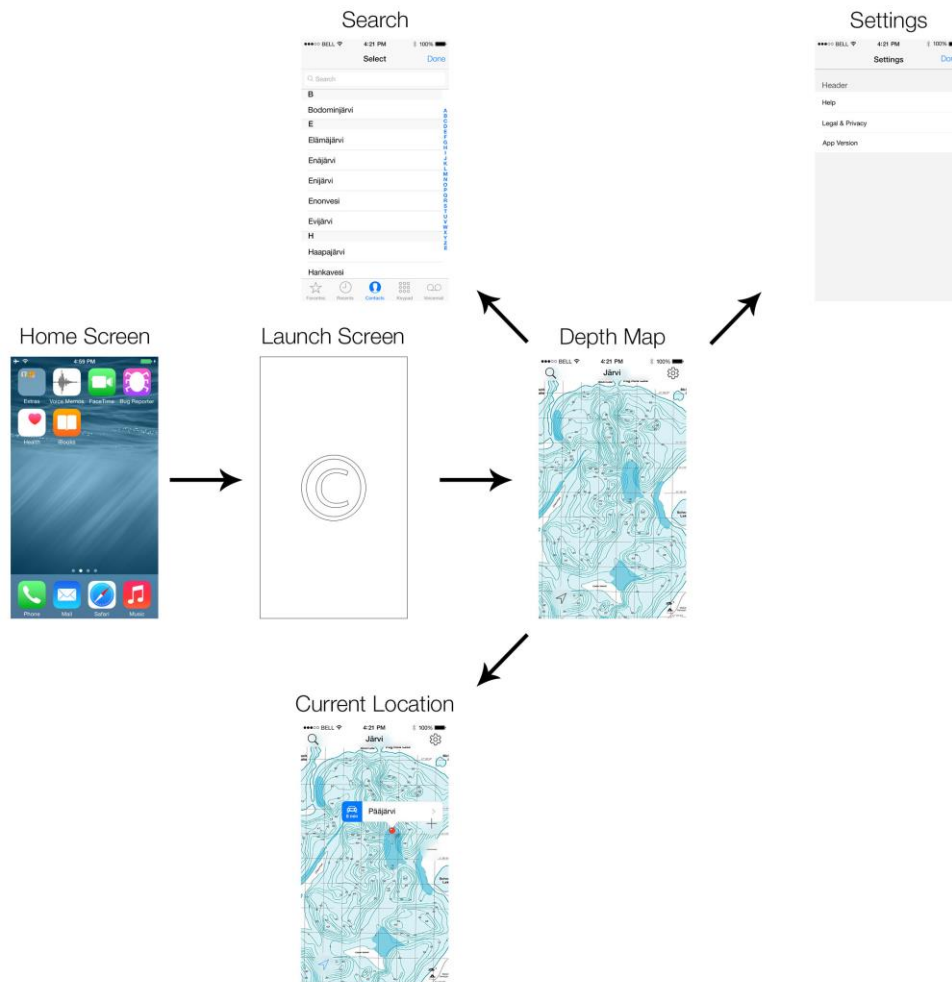


FIGURE 17. Early prototype of the Storyboard

5.7 Development

The main development process was the following: a new feature was tested in a separate project, and only after it worked the way it should, it was added to the main Xcode project.

The process of creating the application was the following:

1. Add the ArcGIS Runtime SDK to the Xcode and test it.
2. Configure ViewControllers and their hierarchy.

3. Add proper navigation between ViewControllers.
4. Connect the ArcGIS framework to my previously created ArcGIS map.
5. Customize map in the right way.
6. Add location functionality (displaying a user's current location).
7. Add search field functionality.
8. Add the required buttons.
9. Configure pop-ups.
10. Configure Settings TableView.
11. Add design elements – AppDelegate, icons, etc.

And of course, I was testing the application during the whole process, thanks to the iOS Simulator which is a part of Xcode.

5.7.1 Adding ArcGIS Runtime SDK to Xcode

The installation of the framework was quite easy. There is a perfectly clear instruction on its webpage tutorial which involves just several steps of adding the necessary links and dependencies in the project with the previously downloaded for free ArcGIS SDK frameworks (ArcGIS Runtime SDK for iOS 2015). The manual setup was done in several steps:

1. Adding framework search paths.
2. Adding build flags (-ObjC -framework ArcGIS -l c++).
3. Adding dependencies and enabling modules.
4. Optional – adding ArcGIS resource bundle, containing a huge set of default ArcGIS user interface elements.
5. Adding the line of code to an Xcode project's classes' headers - #import <ArcGIS/ArcGIS.h>

After this setup I could start testing the prepared maps. However, several steps were still required to continue.

5.7.2 Configuring Views and ViewControllers

The configuration of the View was relatively easy as well. This included creating two ViewControllers – a navigation controller for the navigation bar and the table controller for the settings section of my application. After adding the navigation controller, the navigation bar automatically appeared. By dragging the required button to the navigation bar on the first View it was added on it. The same way a new View is created, simply by dragging the View controller element from the Object Library to the Storyboard. However, a new class has to be created, for example `ViewController2`, by adding this name into the Class field of the Identity inspector and by creating basic class files: the header and the modifier, `ViewController2.h` and `ViewController2.m` as in my example.

5.7.3 Configuring Navigation

Because this was my first iOS application creation experience ever, I can't but mention how easy it is to work in Xcode. Interface Builder is a great feature which allows creating application's user interface by managing elements from the Object Library. Also, because FishinGO is quite a small application, the navigation between ViewControllers is very simple. When a user opens the application for the first time, he immediately get access to its primary function – maps. There is only one single button in its right corner which leads to settings.

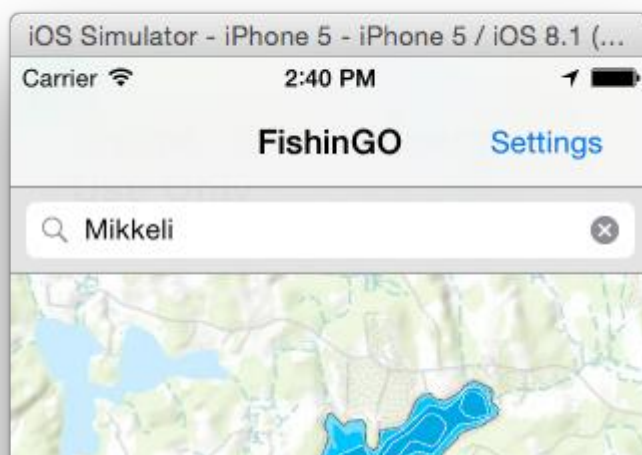


FIGURE 18. Navigation Bar with Search

In order to give the button navigation functionality, it is required to select it, to press CTRL and to start dragging the button, when a special arrow appears, onto the required View. After performing this action, a pop-up appears with several suggestions of action Segue: push, modal or custom. The difference between push and modal Segue is that a push Segue is adding a new ViewController into the same navigation stack where the original ViewController belongs to. A modal Segue is a better choice, when a new ViewController does not have to be a part of the UINavigationController, but simply is the next step from the previous ViewController. In my case, I used the push Segue, and the Back buttons were created automatically without any special actions. If I would have chosen a modal Segue, the NavigationBar would not be created automatically on a new ViewController, and there would be no Back button either.

When the application is opened, the Navigation controller is first one for obvious reasons: it is responsible for the UINavigationController. The button Settings leads to the Table View Controller, which has only one Section with a title Support and serves as an information point. There are three static Table View Cells – Acknowledgement, Feedback and About and Legal, which are buttons and lead to three simple View controllers with the same titles respectively. There is only one Table View Cell which is not a button. It is the version of the application and serves right now more for the future purposes. To make it a simple label it simply requires turning off the User interaction enabled tick from the Attributes inspector.

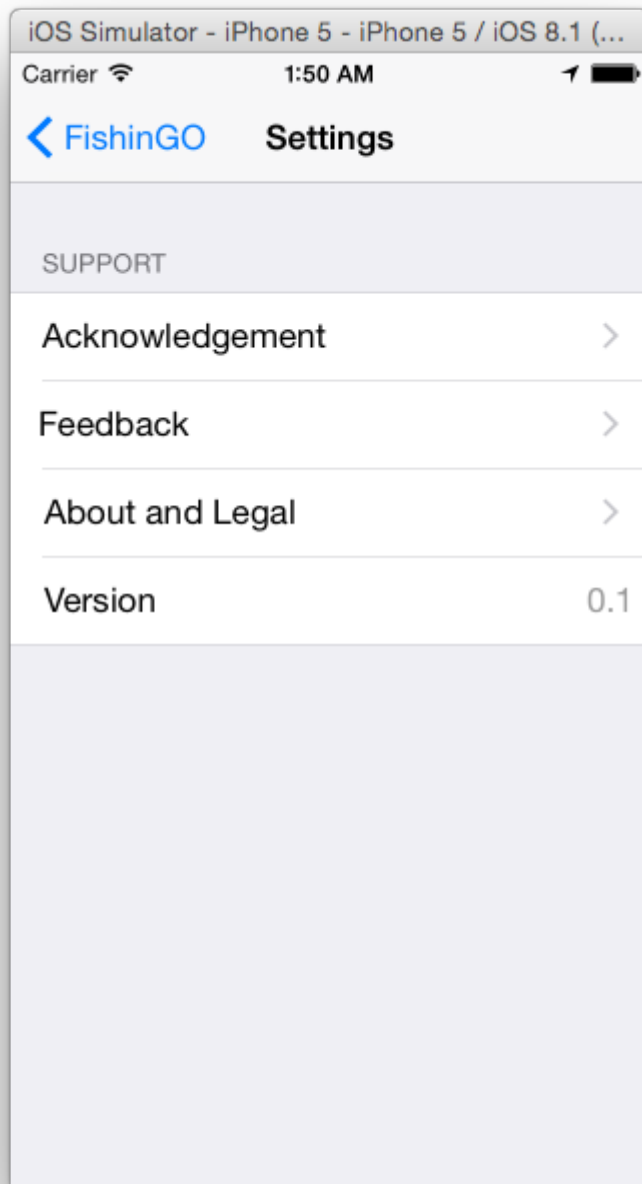


FIGURE 19. Settings page

5.7.4 Connecting ArcGIS WebMap

This was one of the most important parts of my thesis work, because if I did not handle this, there would be no primary functionality in my application. In other words, there would be no purpose in it. During this step I faced quite many difficulties:

In my first prototypes I was using code from the ArcGIS Runtime SDK for iOS examples, and the problem was that it contained only information about loading layers, but not about loading WebMaps. The code to load a layer was the following:

```
AGSTiledMapServiceLayer *tiledLayer =  
[AGSTiledMapServiceLayer tiledMapServiceLayerWithURL:[NSURL URLWithString:@"http://services.arcgisonline.com/ArcGIS/rest/services/World  
_Topo_Map/MapServer"]];  
[self.mapView addMapLayer:tiledLayer withName:@"Basemap Tiled Layer"];
```

Quite a lot of time was spent on trying to select the important code for my project, but finally, after the WebMap was added to the MapView everything worked without any problems.

```
//Setting up my credentials  
AGSCredential* credential = [[AGSCredential alloc] initWithUser:@"ilush93" password:@"mamk123456"];  
credential.authType = AGSAuthenticationTypeToken;  
//Setting up the right WebMap  
self.webMap = [[AGSWebMap alloc] initWithItemId:@"c7c59bfc69d14593a59953c900dc8818" credential:credential];  
self.webMap.delegate = self;  
//Setting the webMap onto the Mapview  
[self.webMap openIntoMapView:self.mapView];
```



FIGURE 20. FishinGo displaying a test WebMap with Jyväskylä area (old prototype)

5.7.5 Displaying Current User's Location

This part of code is responsible for showing user location immediately after the map is loaded:

```
#pragma mark - AGSMapViewLayerDelegate methods
- (void)mapViewDidLoad:(AGSMapView *) mapView {
    [mapView.locationDisplay startDataSource];
}
```

The method `startDataSource` is used for gathering the user's location. The following lines of code are responsible for moving/panning to the user's current location after learning it:

```
self.mapView.locationDisplay.autoPanMode = AGSLocationDisplayAutoPanModeDefault;  
self.mapView.locationDisplay.wanderExtentFactor = 0.75;
```

The `autoPanMode` is a factor which is used to determine when the map is centered on a location. It is used only with `AGSLocationDisplayAutoPanModeDefault`, a declaration, which makes the map recenter on the location symbol. The `wanderExtentFactor` value may be from 0 and 1. A value of 0.25 means that an extent 25% of the current extent is used as the "wander" extent. A value of 1 means that the entire extent in view is used as the "wander" extent.

5.7.6 Adding Search

To give a user a chance to view different locations easily, I had to add a search bar to the `MapView` the same way as other usual elements from the `Objects Library` and to customize its basic appearance. When the search field is clicked by user and something is typed, a layer with location results is added to the map. Then an image of a pin is assigned to this layer's locations in order to display each of the search results. If this was not the first search, the results would be cleared from the map. The code is described and explained in the Appendix 2 (2, 3) to this thesis.

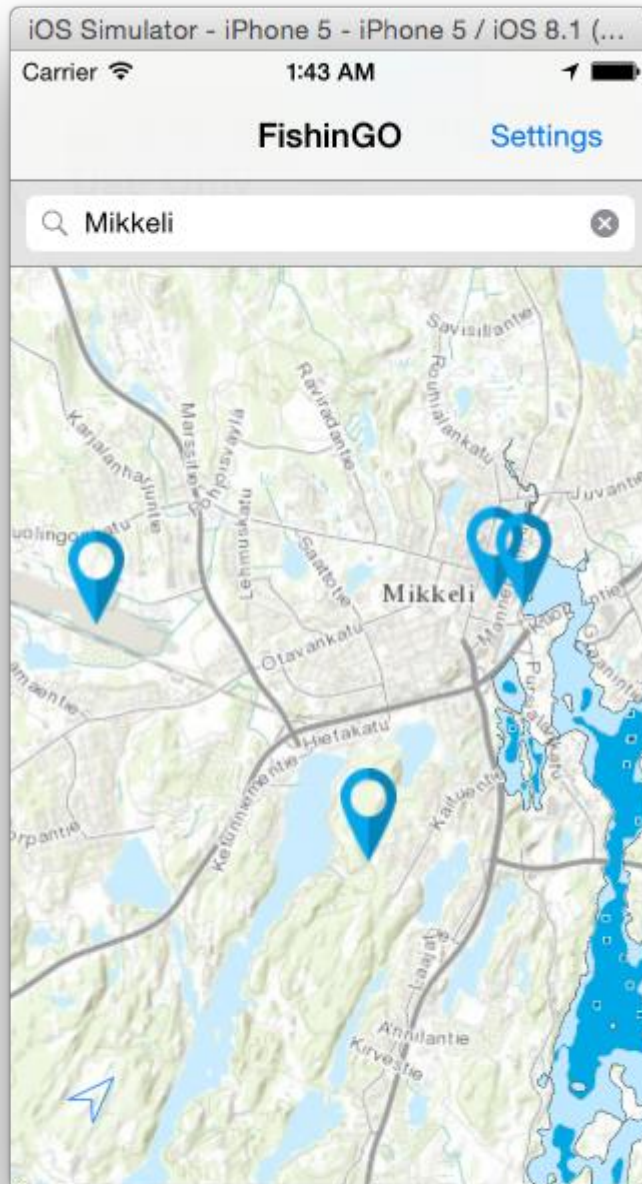


FIGURE 21. Displaying search results for a request “Mikkeli”

5.7.7 Adding the «My Location» button

To create a button which shows or returns to the user’s current location after browsing the map, I had to take several simple steps as follows:

1. Add a button to the MapView.

2. Go to the Connections inspector.
3. Drag the Touch Up Inside action to the class's header file and assign it a name myLoc.
4. Simply use the same code as I wrote for moving to the user's location when opening the application.

This is the only button on the MapView, though I created zoom in and zoom out buttons on the prototype. There are several reasons for that. Firstly, I wanted to save space for browsing and viewing the map. Secondly, modern smartphones like iPhone support the zoom actions using gestures by default.

5.7.8 Configuring a Pop-up

After consultations with my thesis supervisor, the decision was made to create a pop-up window. It will appear when the application is opened and tell the user that both the me, the developer, and Latauspalvelu, the data source, do not take any responsibility for any kind of navigational usage of this application. The acknowledgement in the application's settings also states that the only purpose of this application is to show the approximate current location of the user on the depth maps of some of the lakes and to give some kind of advice for which place to consider for fishing. The first plan was to implement two buttons on the pop-up window – “Disagree” which would simply quit the application when pressed, and “Agree” which will allow the user to continue and start using the application. However, when I started implementing the “Disagree” button, I found out, that Apple forbids creation of such elements in applications – they are simply not accepted by the App Store, because of the guidelines. Here is an example of Apple's reply to a developer, who had implemented such a button:

«The iOS Human Interface Guidelines specify,

"Always Be Prepared to Stop iOS applications stop when people press the Home button to open a different application or use a device feature, such as the phone. In particular, people don't tap an application close button or select Quit from a menu. To provide a good stopping experience, an iOS application should:

Save user data as soon as possible and as often as reasonable because an exit or terminate notification can arrive at any time.

Save the current state when stopping, at the finest level of detail possible so that people don't lose their context when they start the application again. For example, if your app displays scrolling data, save the current scroll position."

It would be appropriate to remove any mechanisms for quitting your app». (Stackoverflow 2013)

This was a strong reason why only a single button "I agree" was left. On one hand, this way does not leave any choice for the user, for example, to disagree, but on the other hand, it performs its main function well by acknowledging the user about this licensing and responsibility moment. The code can be noticed in Appendix 2(1) and the result is represented on FIGURE 22.

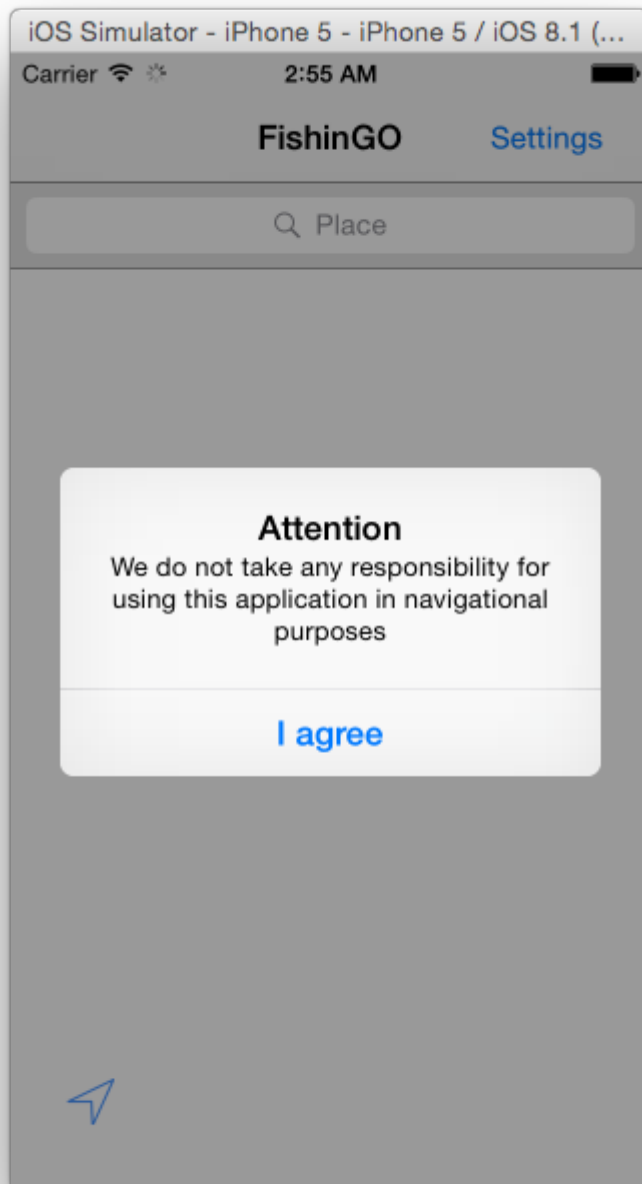


FIGURE 22. Displaying the pop-up acknowledgement

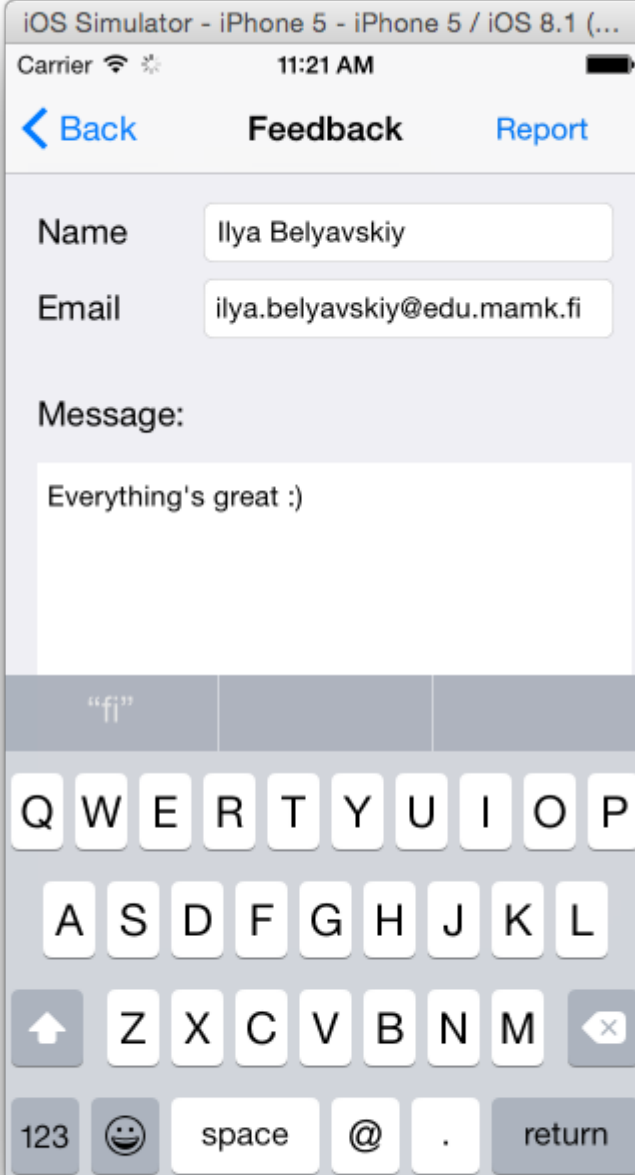
5.7.9 Configuring Application's Settings

Settings are configured using Table View Controller. That is why it is required to drag this element from the Objects library to the Storyboard. Next, on the Attributes Inspector tab it is required to change the content type from Dynamic Prototypes to Static Cells, because in our

case, the content in the cells will never change. Also, the number of sections was set to “1”. The style was changed from Plain to Grouped according to Apple’s iOS UI Guidelines (iOS Developer Library 2014). Giving the section a title SUPPORT is done in the section’s parameters. In order to add some text on the table view cells it is required simply to drag labels from the Object Library and name them properly. To add arrows on the right side of the buttons, or table view cells, the parameter of Accessory should be changed to a disclosure indicator. Creating a connection between a button and a ViewController is as simple as always – simply by pressing the CTRL button and pointing the appearing arrow to the required ViewController. To add lines of text to the Acknowledgement or About and Legal ViewControllers’ text views should be used. To prevent any changes the ticks on “Editable” should be turned off. “Selectable” was left turned on in order to allow the user to open web links. The results can be noticed on FIGURE 19, FIGURE 23 and FIGURE 24.



FIGURE 23. Acknowledgement Page



iOS Simulator - iPhone 5 - iPhone 5 / iOS 8.1 (...)

Carrier 11:21 AM

[Back](#) **Feedback** [Report](#)

Name

Email

Message:

“fi”

Q W E R T Y U I O P

A S D F G H J K L

Z X C V B N M

123 space @ . return

FIGURE 24. Feedback form

5.8 Testing

In order to test FishinGo the developer license was acquired from another project Guesspoint, mentioned in the beginning of the thesis. Unfortunately, such issues as Xcode version incompatibility with iPhone 5 current firmware occurred. To test the application on a real iPhone the

Xcode version had to be updated from 6.1 to 6.3 which supports iOS 8.3. However, in the App Store reviews section of Xcode 6.3 there are many warnings about unstable work and different problems, like glitches and problems with projects already created. Based on these recommendations the testing phase of this project will be carried out as future development in order not to corrupt the current results or harm the working environment.

6 CONCLUSION

This chapter concludes all the results which were developed during the thesis work. Its aim was to get familiar with iOS application development, with Objective-C in particular, with GIS data editing and management and to create the first working iOS application which combines both of the previously mentioned spheres of Information Technology.

Modern gadgets and smartphones are parts of our lives nowadays. Because of the combination of powerful resources in compact sizes they allow human beings to improve everyday life, get better and save precious time. As application development is rapidly expanding, there are quite many tools to learn it, sometimes even by self study methods, as was done in this thesis. Apple does its best to create the most comfortable conditions possible for beginning developers. The greatest advantages of iOS development are Apple's ecosystem – OS X, iOS and Xcode – and the community, with an extremely huge library of knowledge, experience and examples and support. It is quite pleasant to notice that GIS technologies also follow this trend. They can and will be used practically anywhere, sometimes for unpredictable and unusual purposes, like in the FishinGo case. It is extremely great that such a leader on the market of advanced software for professionals, as ArcGIS, understands these trends and supports developers with the required software, frameworks and guidelines in order to expand its technologies to a higher number of platforms.

During this thesis an application FishinGo, depth maps of lakes and rivers of Finland, was created from the idea stage to a fully working prototype. Unfortunately, not all the goals were achieved. For example, storing all the data in the application itself is still a lot more complicated topic. Also, the process of collecting data appeared to be not as easy as it was supposed to be. Different licensing and technical issues occurred. That is why, it is still early

to publish the application created somewhere. But, it is a great platform for the future development, and of course, a great experience for a person who wanted to try programming, really liked it and wishes to continue developing applications in the future.

I had been struggling for a long time with the question of selecting the proper topic for my bachelor's thesis. It changed several times, but with progress and experience I got more and more clear image of what I would like this to be. It was an extremely pleasant project of getting used to iOS development, figuring out how everything works there, and finally, understanding the code. In the beginning I had an idea – a solution to a real life problem. I have never developed any kind of applications like this before. I perfectly understand that for most of the modern and experienced developers this project would seem extremely easy, but for me, creating this small application is a real achievement, probably because this was my first experience in this field and I wanted to overcome my weakness – being a total zero at programming. I wanted to get better at programming, as I really believe that this is a very important, profitable and rapidly developing area of Information Technology, but very difficult. I learnt it much better during the case FishinGo from a different point of view and really liked this experience. I would like to continue developing the map based application created during this thesis and to make it a real alternative to its competitors from the United States. There is a certain demand on the market, and there is a certain target audience which wishes this application to appear on the global application store. I would also like to mention that I have noticed, how fast Finnish IT companies are developing their products. When I was researching this topic in the Russian App Store in 2014, I forgot about exploring the Finnish market of such applications. Fortunately, there are existing applications, which serve the same function. *Maastokartta Suomi* by Kehittäjä: Shingle Oy is one of them (iTunes App Store 2014).

I would also like to mention, that I have already found an investor into my project. He is one of the examples of the target audience, mentioned in chapter 4.5. He is a well known businessman in Saint-Petersburg (and North-Western part of Russia). Obviously, he is a fisherman enthusiast, who travels from Russia to Finland up to several times per month in order to have rest from the city bustle and everyday routine. He has quite many ideas for the future development and is also interested in adding more functionality, like social networking and sharing catch and places with his friends. The main reason for searching for investments was

that to make this project live and to release it to the App Store, it is required to purchase licenses in order to use ArcGIS map services and ArcGIS Runtime SDK.

My plans for the future development involve several main aspects:

- Purchasing all required licences from ArcGIS.
- Covering all possible lake or river surfaces of Finland with depth maps in FishinGo.
- Making the maps available for downloading and saving them to a smartphone's cache as an in-app purchase.
- Creating the iPad version of FishinGo.
- Giving more fishing related functionality, like forecasts, information about wind strength, sun and moon phases, small fishing encyclopedia and information about places where fishing equipment or licenses may be purchased.
- Adding a Russian localization of FishinGo.
- Looking for cooperation with professionals on the market of sonar equipment in order to create special products for use with smartphones. For example, cheaper sonars without any screens may be created and connected to smartphones by using Bluetooth.

Because of these reasons and plans above I decided to found my own application development startup. However, there are still many things to consider and a lot of job to be done, before FishinGo becomes a high quality product, popular among fishermen in Finland and Russia.

7 BIBLIOGRAPHY

Ahvenainen, Mika 2015. Syvyyskartta Thesis Project. Private email message 28.01.2015. Referenced on April 28, 2015.

Apple 2014. Creating jobs through innovation. WWW pages.

<https://www.apple.com/about/job-creation/>

No update information. Referenced on April 28, 2015.

AppleInsider.ru 2013. Teach me XCODE (НАУЧИ МЕНЯ XCODE). WWW pages.

http://www.youtube.com/watch?v=yj9qAUXb5vg&list=PLU_h3oBFtBgsp3zq9KiHkrgKPSMrySJAu&index=1

No update information. Referenced on April 28, 2015.

ArcGIS 2012. The 50th Anniversary of GIS. WWW pages.

<http://www.esri.com/news/arcnews/fall12articles/the-fiftieth-anniversary-of-gis.html>

No update information. Referenced on April 29, 2015.

ArcGIS Resource Center 2013. What are geographic coordinate systems? WWW pages.

http://help.arcgis.com/en/arcgisdesktop/10.0/help/index.html#/What_are_geographic_coordinate_systems/003r00000006000000/

Updated on 31.07.2013. Referenced on April 28, 2015.

ArcGIS Runtime SDK for iOS 2015. Add a map to your app. WWW pages.

<https://developers.arcgis.com/ios/objective-c/guide/develop-your-first-map-app.htm>

No update information. Referenced on April 28, 2015.

ArcGIS Runtime SDK for iOS 2015. Geographic coordinate systems. WWW pages.

<https://developers.arcgis.com/ios/objective-c/guide/geographic-coordinate-systems.htm>

No update information. Referenced on April 28, 2015.

ArcGIS Runtime SDK for iOS 2015. Install and set up. WWW pages.

<https://developers.arcgis.com/ios/objective-c/guide/install.htm>

No update information. Referenced on April 28, 2015.

ArcGIS Runtime SDK for iOS 2015. Projected coordinate systems. WWW pages.

<https://developers.arcgis.com/ios/objective-c/guide/projected-coordinate-systems.htm>

No update information. Referenced on April 28, 2015.

Buckey, David. The GIS Primer. An Introduction to Geographic Information Systems. WWW pages.

http://bgis.sanbi.org/gis-primer/page_14.htm

No update information. Referenced on April 29, 2015.

Chrisman, Nick 2005. Charting the Unknown: How computer mapping at Harvard became GIS. PDF file.

http://www.gsd.harvard.edu/gis/manual/lcgsa/HarvardBLAD_screen.pdf

No update information. Referenced on April 29, 2015.

Dalrymple Mark, Knaster Scott 2009. Learn Objective-C on the Mac: For OS X and iOS. New York: Apress.

ESRI GIS Dictionary. WWW pages.

<http://support.esri.com/en/knowledgebase/GISDictionary/term/basemap>

No update information. Referenced on April 28, 2015.

Esri. History Up Close. WWW pages.

<http://www.esri.com/about-esri/history/history-more>

No update information. Referenced on April 29, 2015.

Finnish Transport Agency 2015. Finnish Transport Agency. WWW pages.

<http://portal.liikennevirasto.fi/sivu/www/e/fta>

Updated on 29.04.2015. Referenced on April 29, 2015.

Finnish Transport Agency 2014. The Finnish Transport Agency's hydrographic data license. WWW pages.

http://portal.liikennevirasto.fi/sivu/www/f/aineistopalvelut/avoin_data/Merikartoitus_lisenssi/License_hydrographic#.VVEUZ5Np1_A

Updated on 01.10.2014. Referenced on April 29, 2015.

Fishidy 2015. WWW pages.

<http://www.fishidy.com/>

No update information. Referenced on April 28, 2015.

Geonet 2011. Projected Coordinate System vs. Geographic Coordinate System. WWW pages.

<https://geonet.esri.com/thread/23160>

No update information. Referenced on April 28, 2015.

Ghedin, Guido 2013. An eye on search engine Yandex, the king of the Runet. WWW pages.

<http://www.digitalintheround.com/yandex-russia-runet/>

No update information. Referenced on April 28, 2015.

GIS wiki 2011. Topographic map. WWW pages.

http://wiki.gis.com/wiki/index.php/Topographic_map

Updated on 26.05.2011. Referenced on April 28, 2015.

Howell, Donna 2009. Jack Dangermond's Digital Mapping Lays It All Out. WWW pages.

<http://news.investors.com/management-leaders-and-success/081409-503454-jack-dangermondand8217s-digital-mapping-lays-it-all-out.htm>

No update information. Referenced on April 29, 2015.

Institut Cartogràfic i Geològic de Catalunya. Relationship of ETRS89 with WGS84. WWW pages.

<http://www.icc.cat/eng/Home-ICC/Geodesy/ETRS89/Geodetic-aspects-of-the-ETRS89/Relationship-of-ETRS89-with-WGS84>

No update information. Referenced on April 29, 2015.

iOS Developer Library 2014. Programming Guide for iOS. WWW pages.

https://developer.apple.com/library/ios/documentation/WindowsViews/Conceptual/ViewPG_iPhoneOS/CreatingViews/CreatingViews.html

Updated on 17.09.2014. Referenced on April 28, 2015.

iOS Developer Library 2015. Tutorial: Basics. WWW pages.

https://developer.apple.com/library/ios/referencelibrary/GettingStarted/RoadMapiOS/FirstTutorial.html#//apple_ref/doc/uid/TP40011343-CH3-SW1

Updated on 09.03.2015. Referenced on April 28, 2015.

iTunes App Store 2013. iFish USA - The App for Fishing in America by the App Door.

WWW pages.

<https://itunes.apple.com/us/app/ifish-usa-app-for-fishing/id442793900?mt=8>

Updated on 20.06.2013. Referenced on April 28, 2015.

iTunes App Store 2014. Fishing Spots - Angling Map & Trip Weather. WWW pages.

<https://itunes.apple.com/us/app/fishing-spots-angling-map/id417922502?mt=8>

Updated on 27.04.2014. Referenced on April 28, 2015.

iOS Developer Library 2014. iOS Human Interface Guidelines. WWW pages.

<https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/>

Updated on 08.04.2015. Referenced on April 28, 2015.

iTunes App Store 2014. Maastokartta – Suomi. WWW pages.

<https://itunes.apple.com/fi/app/maastokartta-suomi/id727466441?l=fi&mt=8>

Updated on 28.04.2015. Referenced on April 28, 2015.

iTunes App Store 2015. Fishidy - Fishing Maps, Reports, Hot Spots & Local Fishing Tips.

WWW pages.

<https://itunes.apple.com/us/app/fishidy-fishing-maps-reports/id561498932?mt=8>

Updated on 01.04.2015. Referenced on April 28, 2015.

iTunes App Store 2015. Trimble GPS Fish Pro (formerly Cabela's Recon Fish). WWW pages.

<https://itunes.apple.com/us/app/cabelas-recon-fish/id494321820?mt=8>

Updated on 21.04.2015. Referenced on April 28, 2015.

Jones, Chuck 2013. Apps With In-App Purchase Generate the Highest Revenue. WWW pages.

<http://www.forbes.com/sites/chuckjones/2013/03/31/apps-with-in-app-purchase-generate-the-highest-revenue/>

No update information. Referenced on April 28, 2015.

Liikennevirasto. WWW pages.

<https://extranet.liikennevirasto.fi/extranet/web/public/latauspalvelu>

No update information. Referenced on April 28, 2015.

LIPAS 2013. Map of Sport facilities. WWW pages.

<http://www.liikuntapaikat.fi/lipas>

No update information. Referenced on April 28, 2015.

Mac Developer Library 2013. Cocoa Fundamentals Guide. WWW pages.

<https://developer.apple.com/legacy/library/documentation/Cocoa/Conceptual/CocoaFundamentals/WhatIsCocoa/WhatIsCocoa.html>

Updated on 18.09.2013. Referenced on April 28, 2015.

Mac Developer Library 2014. Programming with Objective-C. WWW pages.

<https://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/ProgrammingWithObjectiveC/Introduction/Introduction.html>

Updated on 17.09.2014. Referenced on April 28, 2015.

Mac Developer Library 2015. The Swift Programming Language: About Swift. WWW pages.

https://developer.apple.com/library/mac/documentation/Swift/Conceptual/Swift_Programming_Language/

Updated on 08.04.2015. Referenced on April 28, 2015.

METSAHALLITUS 2014. Metsähallitus hunting and fishing areas. WWW pages.

<http://www.retkikartta.fi>

Updated on 09.06.2014. Referenced on April 28, 2015.

Mynttinen, Ivo 2014. The iOS Design Guidelines. WWW pages.

<http://iosdesign.ivomynttinen.com/>

Updated on 11.11.2014. Referenced on April 29, 2015.

National Land Survey of Finland. WWW pages.

<http://www.maanmittauslaitos.fi/en/kartat>

No update information. Referenced on April 28, 2015.

Open Geospatial Consortium 2013. OGC History (detailed). WWW pages.

<http://www.opengeospatial.org/ogc/historylong>

No update information. Referenced on April 29, 2015.

OpenSignal 2012. The many faces of a little green robot. WWW pages.

<http://opensignal.com/reports/fragmentation.php>

No update information. Referenced on April 29, 2015.

Perez, Sarah 2014. iTunes App Store Now Has 1.2 Million Apps, Has Seen 75 Billion Downloads To Date. WWW pages.

<http://techcrunch.com/2014/06/02/itunes-app-store-now-has-1-2-million-apps-has-seen-75-billion-downloads-to-date/>

No update information. Referenced on April 28, 2015.

Piggott, Sam 2013. iOS MapKit and MKMapView - Pinpoint and zoom to the user's location.

WWW pages.

<https://www.youtube.com/watch?v=S-bmu8DCOhA>

No update information. Referenced on April 28, 2015.

PRO KALASTUS 2012. Syvyyskartta. WWW pages.

<http://www.prokalastus.fi/kalastus-info/syvyyskartta>

No update information. Referenced on April 28, 2015.

Reiser, John 2011. GIS Data Types. WWW pages.

<http://www.slideshare.net/johnjreiser/gis-data-types>

No update information. Referenced on April 29, 2015.

Solt, Paul 2013. Objective C Programming: Make iPhone Apps using Objective-C for iOS 7 and Xcode 5. WWW pages.

<http://www.youtube.com/watch?v=sgrVQBXu3QU>

No update information. Referenced on April 28, 2015.

Stackoverflow 2013. Exit application when click button - iOS. WWW pages.

<http://stackoverflow.com/questions/14335848/exit-application-when-click-button-ios>

No update information. Referenced on April 28, 2015.

Stanford 2011. CS 193P iPhone Application Development. WWW pages.

<http://web.stanford.edu/class/cs193p/cgi-bin/drupal/>

Updated on 28.04.2015. Referenced on April 28, 2015.

SYKE Latauspalvelu LAPIO. WWW pages.

http://paikkatieto.ymparisto.fi/lapio/lapio_flex.html#app=3d81&c3ad-selectedIndex=0

No update information. Referenced on April 28, 2015.

Tekritisoftware 2013. How to Create Map based Applications in iPhone. WWW pages.

<http://www.tekritisoftware.com/mapkit-framework-ios-application>

No update information. Referenced on April 28, 2015.

Trimble GPS Fish 2015. WWW pages.

<http://www.gpshuntfish.com/Products/TrimbleGPSFish/>

No update information. Referenced on April 28, 2015.

Uikkanen, Eino 2014. Finnish coordinate systems. WWW pages.

<http://www.kolumbus.fi/eino.uikkanen/geodocs/geo/ficoords.htm>

Updated on 17.01.2013. Referenced on April 29, 2015.

Wenderlich, Ray 2010. iOS Tutorial: How To Create A Simple iPhone App. WWW pages.

<http://www.raywenderlich.com/1797/ios-tutorial-how-to-create-a-simple-iphone-app-part-1>

Updated on 06.02.2014. Referenced on April 28, 2015.

Wikipedia 2015. ArcGIS: Sales. WWW pages.

<https://en.wikipedia.org/wiki/ArcGIS#Sales>

Updated on 20.04.2015. Referenced on April 28, 2015.

Wikipedia. iOS: History. WWW pages.

<https://en.wikipedia.org/wiki/IOS#History>

Updated on 27.04.2015. Referenced on April 28, 2015.

Wikipedia. Objective-C: Syntax. WWW pages.

<http://en.wikipedia.org/wiki/Objective-C#Syntax>

Updated on 28.04.2015. Referenced on April 28, 2015.

Wikipedia 2015. Web mapping: History of Web Mapping. WWW pages.

http://en.wikipedia.org/wiki/Web_mapping#History_of_web_mapping

Updated on 01.04.2015. Referenced on April 28, 2015.

Wikipedia. World Geodetic System. WWW pages.

http://en.wikipedia.org/wiki/World_Geodetic_System

Updated on 28.04.2015. Referenced on April 28, 2015.

APPENDIX 1
ViewController.h

```
#import <UIKit/UIKit.h>
#import <ArcGIS/ArcGIS.h>

@interface ViewController : UIViewController <AGSWebMapDelegate,
AGSMapViewLayerDelegate, UISearchBarDelegate, AGSLocatorDelegate,
AGSCalloutDelegate> {}

@property (strong, nonatomic) IBOutlet AGSMapView *mapView;
@property (nonatomic, strong) AGSGraphicsLayer *graphicsLayer;
@property (nonatomic, strong) AGSLocator *locator;
@property (nonatomic, strong) AGSCalloutTemplate *calloutTemplate;
@property (nonatomic, strong) AGSWebMap *webMap;
@property (nonatomic, strong) NSString* webmapId;

@property (strong, nonatomic) IBOutlet UIButton *myLoc;
- (IBAction)myLoc:(id)sender;

@end
```

APPENDIX 2(1)

ViewController.m

```

#import "ViewController.h"
#define CHOOSE_WEBMAP_TAG 0

@interface ViewController ()

@end

@implementation ViewController

- (void)viewDidLoad
{
    [super viewDidLoad];

    //Pop-up
    self.mapView.callout.delegate = self;
    UIAlertView* webmapPickerAlertView = [[UIAlertView alloc] initWithTitle:@"Attention" message:@"We do not take any responsibility for using this application in navigational purposes" delegate:self cancelButtonTitle:nil otherButtonTitles:nil, nil];
    [webmapPickerAlertView addButtonWithTitle:@"I agree"];
    [webmapPickerAlertView show];

    //Loading a webMap
    //Setting up my credentials
    AGSCredential* credential = [[AGSCredential alloc] initWithUser:@"ilush93" password:@"mamk123456"];
    credential.authType = AGSAuthenticationTypeToken;
    //Setting up the right WebMap
    self.webMap = [[AGSWebMap alloc] initWithItemId:@"c7c59bfc69d14593a59953c900dc8818" credential:credential];
    self.webMap.delegate = self;
    //Setting the webMap onto the Mapview
    [self.webMap openIntoMapView:self.mapView];

    /*Seting the map view's layerDelegate to self so that the ViewController is informed when map is loaded*/
    self.mapView.layerDelegate = self;

    //Zooming in to user's location
    self.mapView.locationDisplay.autoPanMode = AGSLocationDisplayAutoPanModeDefault ;
    self.mapView.locationDisplay.wanderExtentFactor = 0.75;
}

- (BOOL)prefersStatusBarHidden{
    return NO;
}

- (void)didReceiveMemoryWarning
{

```

APPENDIX 2(2)

ViewController.m

```

    [super didReceiveMemoryWarning];
}

//Displaying user's current location
#pragma mark - AGSMapViewLayerDelegate methods
- (void)mapViewDidLoad:(AGSMapView *) mapView {
    [mapView.locationDisplay startDataSource];
}

//Search
- (void)searchBarSearchButtonClicked:(UISearchBar *)searchBar {
    [searchBar resignFirstResponder];

    if(!self.graphicsLayer){
        //This layer holds geocoding results
        self.graphicsLayer = [AGSGraphicsLayer graphicsLayer];
        [self.mapView addMapLayer:self.graphicsLayer with-
        Name:@"Results"];

        //Assigning an image for search results a simple renderer to the
        layer to display results as pushpins
        AGSPictureMarkerSymbol* pushpin = [AGSPictureMarkerSymbol pic-
        tureMarkerSymbolWithImageNamed:@"BluePin.png"];
        pushpin.offset = CGPointMake(9,16);
        pushpin.leaderPoint = CGPointMake(-9, 11);
        AGSSimpleRenderer* renderer = [AGSSimpleRenderer simpleRender-
        erWithSymbol:pushpin];
        self.graphicsLayer.renderer = renderer;
    }else{
        //Reset search results
        [self.graphicsLayer removeAllGraphics];
    }

    //This is the connection for AGSLocator to the geocode service
    on ArcGIS Online, which processes the request
    if(!self.locator){
        self.locator = [AGSLocator locatorWithURL:[NSURL URLWith-
        String:@"http://geocode.arcgis.com/arcgis/rest/services/World/Ge-
        ocodeServer"]];
        self.locator.delegate = self; //Setting the delegate
    }
    //Setting the parameters
    AGSLocatorFindParameters* params = [[AGSLocatorFindParameters al-
    loc]init];
    params.text = searchBar.text;
    params.outFields = @"*";
    params.outSpatialReference = self.mapView.spatialReference;
}

```

APPENDIX 2(3)

ViewController.m

```

//Starting the geocoding operation
    [self.locator findWithParameters:params];
}

//Displaying the results:
- (void)locator:(AGSLocator *)locator operation:(NSOperation *)op
didFind:(NSArray *)results {
//if there are no results
    if (results == nil || [results count] == 0)
    {
        //then show a simple alert, that nothing was found
        UIAlertView *alert = [[UIAlertView alloc] initWithTitle:@"No Re-
            sults"
            message:@"No Results Found"
            delegate:nil
            cancelButtonTitle:@"OK"
            otherButtonTitles:nil];

        [alert show];
    }
    else //if there are some results
    {
        //then create a callout token-based template if there is not
any already
        if(!self.calloutTemplate){
            self.calloutTemplate = [[AGSCalloutTemplate alloc]init];
            self.calloutTemplate.titleTemplate = @"${Match_addr}";
            self.calloutTemplate.detailTemplate = [NSString
stringWithFormat:@"${DisplayY}%@ ${DisplayX}%@", @"\u00b0",
@"\u00b0"];
        }

//Assign the callout template to the layer so that all graphics
within this layer and display their information in the callout in
the same manner
        self.graphicsLayer.calloutDelegate =
self.calloutTemplate;
    }

    //Add an image (a pin) for each result
    for (AGSLocatorFindResult* result in results) {
        AGSGraphic* graphic = result.graphic;
        [self.graphicsLayer addGraphic:graphic];
    }
    //Zoom into results with animation
    AGSMutableEnvelope *extent =
[self.graphicsLayer.fullEnvelope mutableCopy];
    [extent expandByFactor:1.5];
    [self.mapView zoomToEnvelope:extent animated:YES];
}
}

```

APPENDIX 2(4)**ViewController.m**

```
//If the usage of location is restricted in this application, then
display an error
- (void)locator:(AGSLocator *)locator operation:(NSOperation *)op
didFailLocationsForAddress:(NSError *)error
{
    UIAlertView *alert = [[UIAlertView alloc] initWithTi-
        tle:@"Locator Failed"
        message:[NSString stringWithFormat:@"Error: %@", er-
            ror.description]
        delegate:nil
        cancelButtonTitle:@"OK"
        otherButtonTitles:nil];

    [alert show];
}

//My location button
- (IBAction)myLoc:(id)sender {
    self.mapView.locationDisplay.autoPanMode = AGSLocationDisplayAu-
toPanModeDefault ;
    self.mapView.locationDisplay.wanderExtentFactor = 0.75;
}

@end
```