Design of networking game in the iPhone iOS platform

Partonas Alexandros
University of Western Macedonia
Department of Informatics and Telecommunications Engineering
Kozani, Greece
axelpart@hotmail.com

Dr. Dasygenis Minas
University of Western Macedonia
Department of Informatics and Telecommunications Engineering
Kozani, Greece
mdasyg@ieee.org

Abstract—**In our days everyone has the possibility to use a mobile device more than a desktop computer because of its portability. In this way every user can have direct and easy access to any information. The last five years we have witnessed a new revolution in information technology: The revolution of the mobile applications, especially on smart phones and tablet devices. It is estimated that soon 80% people will use their mobile devices twice the time they work on desktop. Today, there are many different mobile platforms such as Android, iOS and Windows Mobile. Here, we present our design for the iOS platform, using as a test case a quiz application. Starting from the idea, we move to a hard copy design, coding, networking, design verification and testing. The main focus of this paper is to present the full design flow and to motivate other developers to select the iOS as their working platform.**

Application design; quiz game; iOS; iPhone; xCode; Application; Smartphones; SQLite Database; Bluetooth; WiFi

I. INTRODUCTION

As market demand surges for apps to run on iOS, Android and whatever operating system will power the next wave of smart devices, companies are facing a dearth of mobile development talent. For IT professionals with programming skills, that gap represents a fresh opportunity to embark on a career makeover.

The latest mobile devices and their applications are changing the way we communicate, do business, and access news and entertainment. Everyone has the ability to access their mobile device easier than their Desktop computer. Businesses, consumers and programmers have embraced this innovative content, making mobile application developer one of the most demanding and fastest growing IT careers.

Applications for mobile platforms have evolved to a point that offers the user a rich and fast experience. Applications are technically categorized based on the programming environment in which they are executed, such as iOS, Android, or Symbian OS. They are also divided according to their functions for mobile platforms as follows: communication applications such as email, production applications such as calculators, multimedia applications such as audio and video streaming, and game applications.

Mobile developers, develop inside a mobile development environment using the Objective C, C++, C# or Java programming languages. A mobile app developer chooses the mobile platform they will develop for, such as Google's Android or Apple's iOS. Then, they learn the programming languages and software development environment for that platform. As a mobile developer, must take into consideration that they have to deal with many different devices. They have to think about the available screen sizes, processor units and RAM, in order to execute an application on every different device. That is the main difference between developing for a Desktop computer and a mobile device. On Desktop computers we can choose the performance and RAM and have a powerful machine.

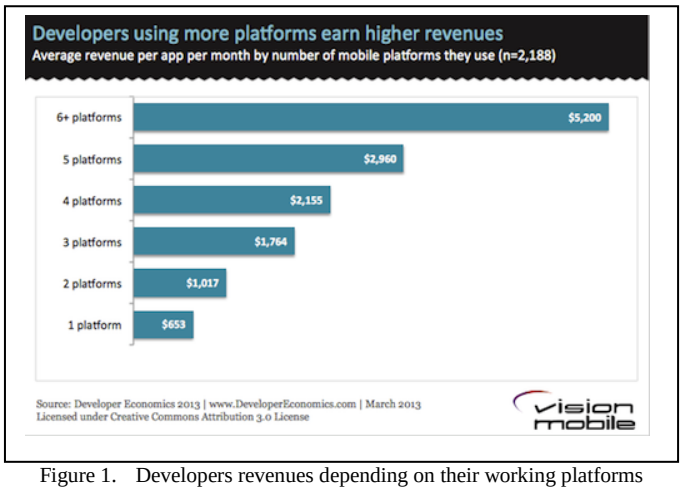

Figure 1. Developers revenues depending on their working platforms

Figure 1, saws that the developers employing just one platform are probably solo, amateur developers or have not yet had the success that warrants (and allows) an expansion onto more platforms. As developers become more successful, they will expand onto new platforms and generate more revenue. So while, expanding on more platforms is not sufficient to generate more revenue on its own, those that do find success are likely to invest in a multi-platform strategy.

We all know that it is hard for someone to be a developer. However, we are here to motivate you by presenting our own mobile design flow from the scratch, for a quiz application in

the iOS platform. In section II, we initiate with the idea, where we analyze the steps of design and development of the application. In Section III, we describe the operations connected with the database. In Section IV and V, we are dealing with the creation of the quiz project and its networking function, respectively. In Section VI, we present our testing procedure and analyze the results. Finally, give the concluding remarks in Section VII.

II. DESIGNING AND DEVELOPMENT OF APPLICATION

It is acceptable, that most of the students who try to become developers give up in the middle of the road, because of high demands. It is hard for a new developer to deal the many different things that are required in order to achieve a functional application.

When we decided to create our application, we thought that it would be useful for someone to be able to test their knowledge while enjoying a coffee with their friends. It is also very fruitful for a teacher to use a knowledge game to intrigue his students and to help them grasp and retain knowledge about his course. Researchers have shown that learning using games has a much stronger impact than the steer typical teaching of a lecture [7].

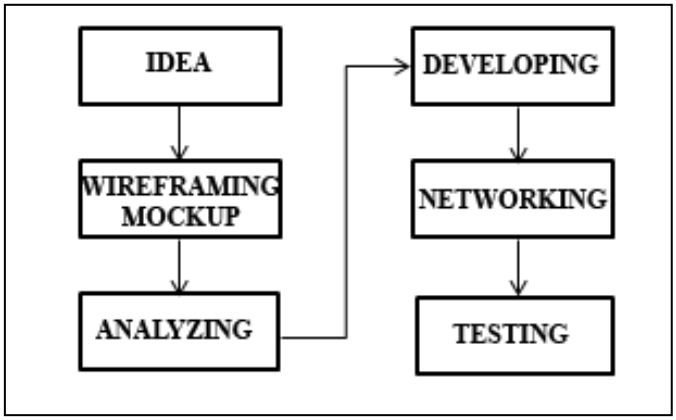

Figure 2. Steps for designing an application

To create a successful application, a developer has to be methodic, systematic and follow some important steps. We present our design flow in iOS platform, which is architecture independent and can be used from developers, in order to create their application from the beginning. Figure 2, shows the steps that we follow in order to create our application. First, we have an idea for an application. Then, we create a sketch of the application view. In the third step, we see how we can analyze the functions of our project. On steps four and five we begin developing the application and finally we start testing it.

A. *It all starts with an idea*

Before a developer initiates an application, it is suggested to perform a survey of similar applications, to investigate their drawbacks and to note down the best practices. We came to the conclusion that the most of the apps need social networks for playing multiplayer. Some multiplayer apps are exclusively developed for iPad, because of its large screen. So, we decided to create an application that does not require sharing the screen with other users, but allows every user to see the same information with the rest.

To take full advantage of the tools given to a developer, one must first describe as solid as possible the type of application. Our application belongs to the category of entertainment and educational applications - games [7] designed to test the knowledge of the users. For the development of a successful application in iOS a developer should know some basic things. These have to do with designing, proper memory management, responsiveness, energy consumption and security.

B. *Wireframing the idea (Sketching)*

To start with, the designer needs to create on a paper a sketch of how he imagines the interface. This process is very important because it helps him to arrive at the desired result with less effort. Also by drawing on paper, it is easier for the designer to show his ideas to others so that he receives comments and reviews about them. Then, the final result may be completely different from the original sketch.

Wireframing [1] is a very important step in the designing and development of an application. It is a low-level outline (draft, framework, and blueprint) of someone's design, which helps them and their customers to illustrate the structure and the sensation that leaves their designing. In wireframes, a designer replicates data throughout the application with their real dimensions and sizes, but nevertheless it does not contain graphic elements, but only the position that they will take in the final designing on screen. Through the wireframe, the designer can make structural changes, changes to buttons, text, titles, and application components in a very short time, and thus they can easily achieve a clearer picture of their application.

For our application we originally created every single view on paper. Then we used software to digitize our design and make a more accurate picture of our application. Figure 3, saws the digital design (mockup) of our multiplayer game view controller.

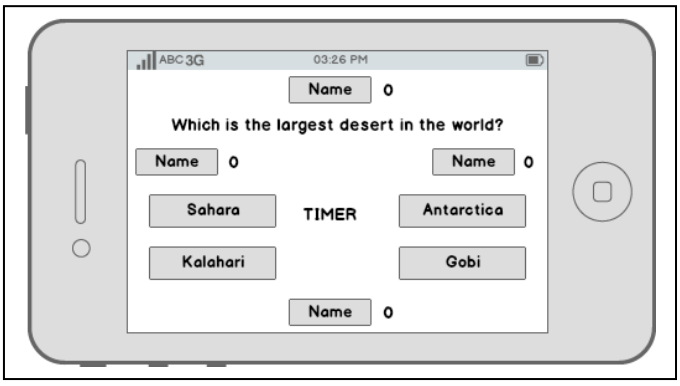

Figure 3. Digital wireframing of game view controller

C. *Analyzing the modes of execution*

Before a developer begins to develop his application programmatically, it is important to clarify how it operates. This is usually analyzed by the user's actions in relation to the application. It would be even easier for the developer to

analyze his app's modes of execution, by creating diagrams with the actions of users.

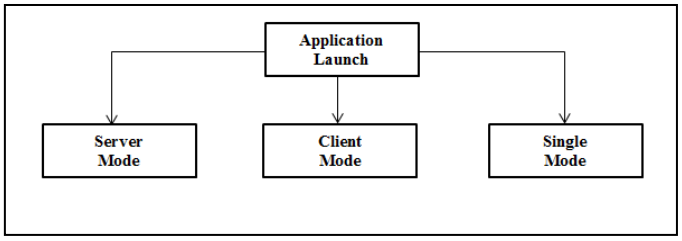

Figure 4. Flow diagram of user actions

Starting the quiz game, the user can choose the "behavior" of the device. Figure 4 shows that there are three options of execution. The first is about hosting an online-web game (server), and the second about a client in a network game. These two options use networking technologies (Bluetooth Wi-Fi). The third option is about the single use that does not require any networking technology.

After that, the choices the user has when connecting as an administrator are being analyzed. If he chooses to host a game, the device starts transmitting the beacon in order for the clients to discover his device. When networking technologies are disabled, the user is notified with an alert message to enable at least one of networking technologies. Then, he waits for other clients to connect, with the maximum number of three. Once the connection is completed, he chooses the time and the questions that he is going to play with the rest of the players.

As a client, the user sees the available servers and he connects to one of them. When all the clients are connected, the server sets the game, and the first question is appeared.

Finally, we have the function of single user. By selecting to play alone, the user selects the category and then sets the time and the questions. Networking technologies are not needed in this option.

III. ANALYSING AND DESIGNING THE DATABASE

Usually, mobile applications have to store data. This can either be fulfilled by means of a file, or by a database, or in a web location. Each one of these techniques has its advantages and disadvantages. For storing the data, we have decided to use a database created by SQLite [2], because of the size that it occupies. Its use is ideal for devices where memory and process power are limited.

Before beginning coding, developer should begin with the creation of the external files. These files usually consist of graphic features such as button images, or a database. All external files will be added to his total project when he starts coding. In case of a database, in order to achieve a functional and working database, the developer should determine the architecture of his application and the roles of its users.

This method helped us in our application and we achieved to optimize our database. As a result, we created only two tables for the quiz game instead of six.

A. *Architecture*

The developer has to determine the architecture of his application so that he can simplify his project and start implementing it step by step. Our system is based on two different categories of implementation. In the first one, we use the client-server model with bidirectional communication. That means that we communicate and share information between all devices on the creation of the game. In the second case, we only use the system in which a simple extraction of data to the user is being done.

In the first case, the system supports up to four users, one of them being the host of the game. For this function, a connection via wireless networks from the server is being created and starts sending information to the rest of the available clients. The entire game architecture is based on the message packets between server and clients.

In the second case, we have a different approach to the game. We have a single user who receives on-screen information about the game.

B. *Roles of the system*

In our application we analyzed all possible roles and operational flows, in order to construct the database. Due to lack of space we will present only one flow of execution for the administrator role. The user who chooses to use the application in communication protocols IEEE 802.11 (Wi-Fi) - Bluetooth, is defined as an administrator and becomes the local server that will host the remaining players.

Basic operational flow for administrator role

- In this use case, the administrator starts the host game by selecting the corresponding button.
- Then, he/she enters the name of his/her preference for the creation of the local server that will host the game.
- He/She waits for the other players who wish to connect to the administrator's network to do it.
- After the number of the available players is completed, he/she selects "Start".
- Then, the form for selecting the question categories is displayed.
- The administrator has the option to select one or more of them if he/she wishes.
- Finally, by pressing the button "Start", he/she also defines the desired duration time of the game, and the number of the questions on which he/she will compete with the rest connected users.

Alternative flow - Canceling creating the local server

Administrator has the option to cancel the impending start of the game that has been created whenever he/she wishes by clicking on the "X" at the bottom left of the game form.

C. *Database operation*

In our application, the database was created using SQLite by the terminal. It is suggested for a developer to use the terminal in order to begin familiarizing with writing code. After its creation, he can use the SQLite plugin for Firefox to add faster entries into tables.

In our quiz game the database is closed. This means that the three users that we defined above have not the ability to add or remove questions through the application. The only way to add data in the database is online, or through a new refreshed version of the entire application.

When the application starts, the database is initially set to be renewed after choosing the category. That means that it writes in the field «Used» the value 'NO' to all the questions that are saved in the database. This is the proper use of the database in the process of the game. Choosing the category of questions they want, either the server or the single user start the game.

The database starts exporting the category questions the user selected randomly and with selection criteria the fact that the field «Used» has the value 'NO'. For each question that is being displayed, the change of the variable of the field «Used» with the value 'YES' to this question is automatically being done. This way we can avoid the appearance of a double question.

Finally, so that there is no abuse-intemperance of the device memory, we have defined that the database closes each time it performs a question.

IV. CREATING THE PROJECT

All new mobile operating systems are coming with their own software application for development, in the case of anyone wants to occupy and develop apps for each operating system. So, if a developer decides to create an application, he has all the tools he needs to achieve that.

We chose iOS because we wanted to learn the secrets of Objective C language. It is said that it resembles to C++ but in reality it has a lot different syntax.

When programming for iOS we wrote our code in Xcode and used its simulation for testing. Xcode gives to developer the ability to choose between different options for his project. We suggest that the developers choose an “Empty Application” project on xCode, because with this option they can customize View Controllers for their applications.

A. *Analysing game view controller*

All view controllers in our application where created from “Empty view controllers” by adding only the elements we needed. By starting our application, the user joins the menu. Menu consists of the following three buttons: “Host Game”, “Join Game”, and “Single Player”.

When the user selects “Single Player”, the category controller is appeared. This controller consists of one table view which is filled with values of category table. Additionally, two buttons are available, one for starting the game, and the other for exiting to the main menu [4]. By choosing a category, the user has the ability to know the total number of its questions. For that purpose, an alert message that displays the necessary information was created.

Figure 5 represents the game view controller, which consists of seven labels. One of them is for the display of the time, one for the score of the correct answers, and the other five for the question and its’ possible answers. Also, there are four more buttons that cover the labels of the answers and will be used for the method that will check the right answer.

Figure 5. View of the game when loading the answers

B. *Analysing the game code*

In the process of programming, developers should use techniques to minimize the effort in building and deploying their application. The most important is to learn about errors, how to handle them, and how to fix bugs of their application. Many times, we had to deal with problems in our application. Bellow we analyze the game code and some techniques we used.

Starting the game, the first thing to do is to clear the screen so that the notification message of time input and number of questions are displayed. To achieve this in every controller, there is a method - (void)WillAppear:(Bool) animated, stating the methods that will be activated at the start of the controller. Here we created a method to achieve the screen clearing and disabling the use of the buttons in case the user accidentally touches the screen.

After clearing the screen, if we wish to start the game we must set the time and questions. To complete the notification message we create two variables of "Int" type. After the user has set the data on the time and questions by pressing the "OK" button on the notification message, the method of time display, timer start up, and questions are triggered. At the same time we get the values and we place them in their own labels. In order for this button to activate, certain conditions must be fulfilled.

For the game time, the user needs to enter more than two digits, while the number of questions should not exceed the listed questions. To do that, we need a variable of "NSInteger",

which we will import from the previous view controller (category view controller).

When a question is presented to the user, they must choose a possible answer. It would not be right to derive the answers in the order they are written to the database. So, we create a condition that randomly shuffles the answers "arc4random" and it saves them temporarily in a table of "NSMutableArray" type.

When the temporary table is filled with the four scrambled numbers of responses, we display the answers shuffled to the user, setting the text in every label.

In order to ensure the proper operation of the answer method, we set tags on each label, which receives the correct answer with numbers "999" and "0" for the wrong. We also equate the labels to the corresponding buttons to implement the method of the correct answer.

We create a method -(IBAction) which we connect with all the buttons. Depending on the user's choice, if the answer is correct, the method that increases the points for every correct answer and the method of the next question are running in sequence. If the answer is wrong, the method of the next question is executed.

Finally, to arouse the interest of the user, the moment he/she is choosing an answer, at the same time a hidden second timer of ten seconds with a countdown is "triggered". With this timer we created a method on the total user points. We have set the maximum number of points at one hundred. The time of ten seconds starts counting the moment that all the answers are displayed. This means that the faster the user answers the more points he/she gets.

V. CREATING NETWORKING PART

In our interconnected world, it is very important for an application to exploit the communication abilities of modern mobile phones, such as Wi-Fi, Bluetooth or near field communication (nfc). Each of these technologies has its advantages and disadvantages. For example if you create an RSS Feeder you have to use Wi-Fi. In that case neither Bluetooth nor NFC are functional. Depending on the category your application belongs, it is recommended to use the most suitable networking technologies.

In our application, we selected Wi-Fi and Bluetooth because of their diversity. In the first case, Wi-Fi technology gives us the ability to connect with other users in a local network, but also with the tethering technology, which allows the use of a mobile phone as an intermediate (access point) so that an internet access is provided to another connected device, either via cable or wirelessly. Bluetooth on the other hand, is ideal for use when the users are less than ten meters far from each other and on places where Wi-Fi is not available.

A. Networking architecture

For creating online-network games, developers require to know that they actually have a choice between two architectures, which are: client - server and peer users (peer-to-peer) [3]. For the implementation of the game, the architecture peer-to-peer was chosen. The user who hosts the game is the server, while all the other players are the customers.

In the client-server model, the server is responsible for everything and determines what the “truth” is. The clients send update messages to the server all the time and the server informs all the clients respectively. The clients do not have the capability to communicate with each other.

In the peer-to-peer model, all participants are equal and they all do the same job, but we must ensure that each participant sees the same things as the rest, since there is no central administrator.

B. Client-server state machines

When developing networking applications, it is handy for a developer to create state machines. State machines give us the opportunity to understand the logic of our application. Bellow we present a sample of our quiz application state machines.

Starting from the client, it is good to create a diagram which describes the possible states of the objects in the class. This makes the usage of the objects simpler.

Overall, we create four states. If the client chooses to join a game he enters the state "Idle", in which he remains idle. Then, he moves to the "Searching for Servers" state. When the user decides to connect to a specific server, the client goes into "Connecting to Server" state, where he tries to connect to the server and finally moves to state "Connected" once the connection is successfully established. If at any time during the last two states the server drops the connection, the client moves back to “Idle” state.

The state machine of the server in relation to the client has only three situations. By choosing to become a server, a user gets into the state "Idle" in which he waits without doing anything. Then, he automatically moves to state "Accepting connections". When the user presses the "Start" button to start the game between users, he is transferred to the "Ignoring new connections" state. This means that no new clients are allowed to connect.

C. Identifying error and disconnection reasons

In networking applications, the connection between devices is often sensible. It is suggested that the developer identifies networking errors and disconnections and demonstrate them to the users of his application. In our application in specific, we produced an alert message that warns users about the following errors:

a) Quit reason: No network.

b) Quit reason: Conection dropped.

c) Quit reason: User quit.

d) Quit reason: Server Quit.

D. Packet structure

For all messages the developer has to follow a specific format. Each package must be at least ten bytes. Different types

of packages have the same header but different payload. Figure 6 shows the consistence of our packages.

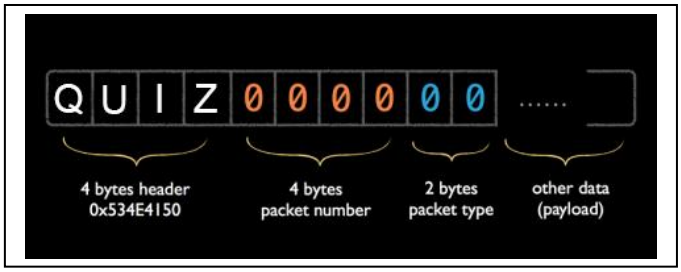

Figure 6. Quiz application data structure

The first four bytes, from the header with the name QUIZ verify that the packages are ours. The next four bytes are used to recognize when a packet arrives out of order, and consist of a 32-bit integer. The last two bytes represent the packet type. Due to the fact that we have many kinds of messages that are sent between the server and clients, this 16-bit integer determines what type of packet it is.

VI. TESTING

The application was initially created for the single user scenario. Then, it was based on the technologies of Bluetooth-Wi-Fi for multi-user scenarios. Decisive role in the functioning of the application played the SQLite, which was used to create the application database. Additionally, with the simple graphic elements that composed the application, we achieved a lightweight version.

For the implementation of the application, a laptop computer with operating system Mac OS X was used. The application was placed on the iPhone 3GS and iPhone 5S devices and was tested extensively. The result was considered satisfactory, based on the different users that tested the application. Trying the application practically arose various errors that have been corrected along the way.

A. *Usage performance*

By running the famous "Who wants to be a millionaire" application, we came to the conclusion that the RAM usage was approximately 55Mb. Our application with three players connected, consumes approximately 19Mb of RAM, which is less than 1/3 of the above mentioned application which was tested in single player mode.

A mobile developer should be careful about the resource usage, like memory. For this reason he should minimize all unnecessary RAM usage, by ignoring non-related packets. For example, while playing multiplayer, in case a player gives a wrong answer, there is no need for the server to send back verification for the wrong answer. Instead, the only sent packet is a packet for informing the next player that it's his turn.

VII. SYNOPSIS AND CONCLUSIONS

Programming for the iOS operating system is not only an interesting experience, but also a promising source of income, especially for young developers. It may be a rough and climbly road, but in the end rewards are awaiting for the systematic, careful and endured developer. Developing for the mobile platform is a slippery track, and for that reason we present our design flow and experience to better help all enthusiastic mobile developers and to motivate indecisive developers. We presented that a careful step by step approach alleviates most of the problems, and guides towards the finalization of a nature project, useful to the community.

A. *Future work*

The main future goal is to provide the application to the public via the App Store. For this process, we must be enrolled in Apple's iOS developers program, with the cost of $99 per year. This allows every device holder with iOS operating system to download the application, test and rate it.

It is also important to enrich the database with more questions, so that the user always discovers something new. Then, out of the users' reviews we will be able to achieve better results with an update of the application, such as more user-friendly environment.

Below are some of the most important extensions of the application:

- A Greek version of the application and a translation thereof in the major foreign languages, so that additional users would be able to use it.
- Addition of different sounds for the correct or the wrong answer, and of different colors on the buttons of responses.
- Addition of a connection to the most popular social networks and high scores among the friends of the user.
- Hospitality and database support through internet, so that every user refreshes the database before the start of the game.

VIII. REFERENCES

[1] «GreekTuts,» [Digital]. Available: http://www.greektuts.net/wireframe-before-design/, 2013.

[2] «SQLite,» [Digital]. Available: http://www.sqlite.org/docs.html, 2014.

[3] Peter Bakhirev, PJ Cabrera, Ian Marsh, Scott Penberthy, Ben Britten Smith and Eric Wing, "Beginning iPhone Games DEvelopment", Apress, 2010.

[4] Bill Dudney and Chris Adamson, "iPhone SDK Development, Building iPhone Applications", The Pragmatic Bookself, 2009.

[5] Erica Saudan, "The iPhone Developer's Cookbook - Building Applications with the iPhone 3.0 SDK", 2nd Edition – Addison Wesley 2010.

[6] «iOSTutorials,» [Digital]. Available: http://www.raywenderlich.com/tutorials, 2013.

[7] Ελένη Ρώσσιου, Σπύρος Παπαδάκης "Μαθαίνουμε Παίζοντας: Το Εκπαιδευτικό Παιχνίδι στη Διδασκαλία της Πληροφορικής στο Γυμνάσιο", Σύρος 2007.