TALLINN UNIVERSITY OF TECHNOLOGY
Faculty of Information Technology

Department of Informatics
Chair of Software Engineering

Grounded architectural decisions for the development of multi-platform mobile application. Case study of "Online Chess Tournaments" application

Bachelor's thesis

Student:	Elina Muhhamedjanova
Student code:	121101IAPB
Supervisor:	Martin Rebane

Tallinn
2015

Author’s Declaration

Herewith I declare that this thesis is based on my own work. All ideas, major views and data from different sources by other authors are used only with a reference to the source. The thesis has not been submitted for any degree or examination in any other university.

	Elina Muhhamedjanova
(*date*)	(*signature*)

Oleme teadlikud ja nõustume, et Elina Muhhamedjanova käsitleb oma TTÜ Informaatikainstituudis kaitstavas lõputöös “Online Chess Tournaments” tarkvara, mille ta on arendanud meie firmas töötades.

Remarc Systems OÜ
Lastekodu tn 48 Tallinn 10144, Eesti

---------- Aleksandr Voronkov ----------

(*date*) (*signature*)

Annotatsioon

Lõputöö eesmärgiks on mitmel platvormil töötava - “Online Chess Tournaments” mobiilirakenduse arendamiseks arhitektuurivalikute tegemine ja põhjendamine.

Lõputöö esimeses lõigus annan ülevaate erinevatest erinevad arhitektuuriprintsiipidest: mobiilirakenduse mudel, kasutajaliidese raamistik, andmesideprotokollid ja päringute meetod. Nimetatud arhitektuurilahenduste võimalikute variantide võrdlemisel on hinnatatud konkreetse lahenduse nõrkused ja tugevused.

Lõputöö teises jaos põhjendatakse eelneva analüüsi alusel mitmel platvormil töötava mobiilirakenduse arendamiseks tehtavaid konkreetseid arhitektuurivalikuid. Selles jaos on käsitletakse ka mobiilirakenduse arendamise põhilisi protsesse.

Töös käsitletud juhtumianalüüsis valiti mitmel platvormil töötava mobiilirakenduse arendamiseks on hübriidne mobiilirakenduse mudel kasutades Ionic raamistikku. Arhitektuuristiilina valiti REST – the Representational State Transfer. Päringute meetodite variantidest on valitud traditsiooniline päringute meetod.

Valitatud arhitektuurilahenduste alusel realiseeriti Chess Rating Agency (international) Ltd tellimusel “Online Chess Tournaments” mobiilirakendus.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 66 leheküljel, 6 peatükki, 28 joonist, 14 tabelit.

Abstract

The main objective of this thesis is to make a grounded architectural decisions for the development of multi-platform mobile application “Online Chess Tournaments”.

The first section of this work is devoted to establishing the notion of different architectural approaches, such as - mobile application model, UI framework, communication protocols and polling techniques. Different varieties of approach methods were compared.

The second part of the thesis focuses on the making grounded decisions on the basis of studies conducted in the first part of the work. Also, in this section the main parts of the “Online Chess Tournaments” mobile application implementation process are examined.

As a result, it was choosen for the development of multi-platform mobile application “Online Chess Tournaments” a hybrid mobile application model with using Ionic UI framework. REST – the Representational State Transfer - was choosen as an architectural style and as polling techniques it was choosen the traditional polling method.

After selecting the above architectural solutions, “Online Chess Tournaments” mobile application was implemented for Chess Rating Agency (International) Ltd company for FIDE.

The thesis is in English language and contains 66 pages of text, 6 chapters, 28 figures, 14 tables.

Abbreviations

FIDE	Fédération Internationale des Échecs (french); World Chess Federation(eng)
UI	User Interface
REST	Representational State Transfer
SOAP	Simple Object Access Protocol
UX	User eXperience
SDK	Software Development Kit
CSS	Cascading Style Sheets
HTML5	HyperText Markup Language, version 5
API	Application Programming Interface
OS	Operating System
DOM	Document Object Model
WebGL	Web-based Graphics Library
IDE	Integrated Development Environment
XML	eXtensible Markup Language
URL	Uniform Resource L?ocator
RDF	Resource Description Framework
HTTP	HyperText Transfer Protocol
CRUD	Create/Read/Update/Delete
HTML	HyperText Markup Language

JSON	JavaScript Object Notation
SMTP	Simple Mail Transfer Protocol
WS	Web Services
WSDL	Web Services Description Language
RCP	Rich Client Platform
FTP	File Transfer Protocol
CPU	Central Processing Unit
ASAP	As Soon As Possible
UTC	Coordinated Universal Time
AI	Artificial intelligence

List of Figures

List of tables

Table of Contents

1. INTRODUCTION

Nowadays popularity of mobile phones among people is growing rapidly. Today people use their phones not only for calling, but for a whole range of online activities: playing games, searching the web, listening to music, taking photos and filming, checking email, online shopping, GPS navigation. The only limitations to use are the apps available. In order to show the situation more clearly, I would like to give some statistics. In recent years, the mobile phones have taken a dominant part of the market compared to corporate and consumer' personal computers and tablets. The line graph bellow shows that mobile devices are replacing computers. (Figure 1)

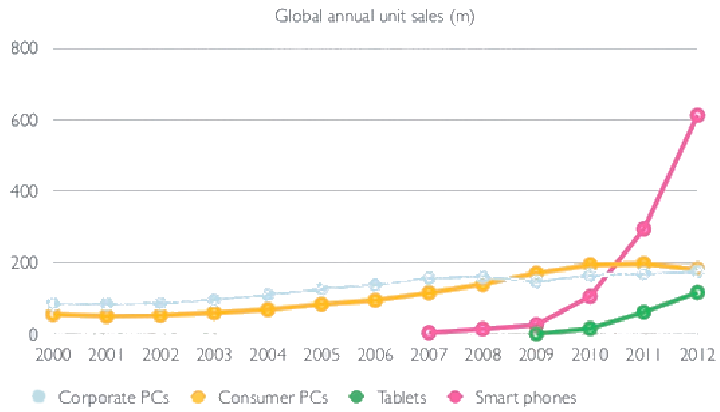


Figure 1 Smartphones sales statistic. [46]

With popularity of smartphones and with the growth of the technology, the usage of mobile applications increases significantly every year. The statistic (figure 2) shows a forecast for the number of mobile app downloads from 2009 to 2017.

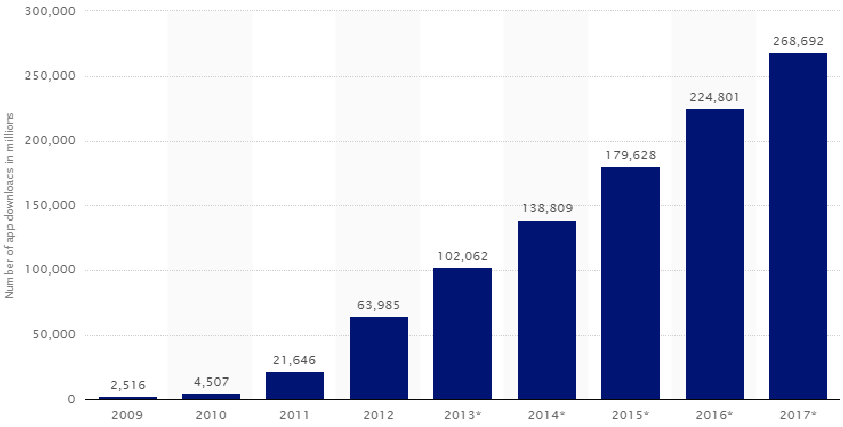


Figure 2 Number of mobile app downloads worldwide. [45]

This trend is the reason that more and more companies want to have both mobile and desktop versions of their software. The Chess Rating Agency (international) Ltd for FIDE is not the exception. The target of this work is to implement a mobile version of the product for the Chess Rating Agency. The idea is to provide users with a fast access to international tournament broadcasting system via their mobile phones. The main idea, why the application will be demand among users, is that a temporary lack of access to the Internet should not be a hindrance to the work of the whole application. At the same time the web application requires a permanent Internet connection.

1.1 RESEARCH PROBLEM AND OBJECTIVES

This thesis is envisioned as a starting point of addressing the problem of the making correct architectural decisions for the development of multi-platform “Online Chess Tournaments” mobile application. “Online Chess Tournaments” is mobile application for Chess Rating Agency (international) Ltd for FIDE. The Chess Rating Agency has already got a web application for broadcasting the international chess tournaments – “Online Chess Tournaments Broadcasting” system. Because of speedily increasing usage of mobile phones and mobile applications, as it has been said before, the aim is to implement the system in mobile version for Android and iOS platform. In this bachelor's thesis a lot of attention will be paid to the following aspects: the choice of the most suitable mobile application model, UI framework and selection of communication protocol between client and server and also the polling technique without increasing the load on the server. The target is to make grounded decisions for the selection of architecture and service types for the system.

1.2 RESEARCH SCOPE AND LIMITATIONS

The first target is to choose a suitable model and a UI framework for the implementation of mobile application for cross-platform mobile devices. Secondly, a communication protocol and polling technique are needed to be choosen.

The third goal that needs to be achieved is that the application will replay tournaments which are already finished in offline mode. So, the application has to know if there are any updates to download. Overall, the aim is to implement the “Online Chess Tournaments” mobile application with all the requirements that have been set by the customer. For this reason, we should analyse all approaches and make correct architectural decisions for the development of multi-platform mobile application.

1.3 RESEARCH METHODOLOGY

To achieve the solution for the problem, this bachelor's thesis was made through documentary analysis. Documentary analysis involves obtaining data from existing documents without having to question people through interview, questionnaires or to observe their behavior. [47] Also, The Case Study was used as a research method. Case Study research excels at bringing us to an understanding of a complex issue of object and can extend experience or add strength to what is already known through previous research. [48]

1.4 THESIS STRUCTURE

Literature review is divided into three sections: “Mobile applications models”, “UI frameworks”, “Protocols and standards: SOAP vs. REST” and “Polling techniques”. “Mobile applications models” chapter serves as an introduction of the notion of available mobile application models such as: native, hybrid and web. In this chapter the features of each model with the advantages and the disadvantages will be summarized. In “UI frameworks” chapter the UI frameworks for realization the hybrid mobile applications will be presented. “Protocols and standards” chapter presents SOAP and REST. This chapter explains the main characteristics of each method. After a short overview of each approach these two methods are compared to each other. In chapter “Polling techniques” are presented two polling methods: traditional polling and long-polling. The chapter discusses strong and weak points of each approach.

Case study section focuses on application implementation and making the grounded architectural decision for “Online Chess Tournaments” system. “Project description” section examines the process choices such as selection of the application model, UI framework and communication protocols between client and server. Also in this chapter are presented the client requirements for the system. “Project implementation” is devoted to a detailed examination of evolution of the project concept, its architecture, design and main processes of the implementation. In the chapter “Review of similar apps” a brief inquiry into the state of analogous applications is presented and a reason why this application is a unique product on the market is also explained. “Results and analysis” section describes the final result of developing the “Online Chess Tournaments” system.

Conclusion chapter summarizes all the goals achieved and presents prospects for further development.

2. MOBILE APPLICATIONS MODELS AND UI FRAMEWORKS

2.1 MOBILE APPLICATIONS MODELS

It is important to investigate different types of application models in order to find out which one fits present-day demands in the best possible way.
There are three main types of mobile applications: native, hybrid and web applications.

This chapter serves as a comparison between the three mobile applications types and presents possible technical advantages and disadvantages between different solutions that are currently available for developing mobile applications.

2.1.1 Web applications

Web apps are designed to look and behave like apps and in general are ideal when the purpose is simply to make content or functionality available on mobile. Web application use JavaScript, CSS, HTML5 or other languages. A developer would not have access to a standardized SDK. This approach creates cross-platform applications that work on multiple devices and platforms through the web browser.

Advantages:

- A single code, which can be accessed by any browser-enabled mobile device.
- No approval process needed, and updates to the application can happen instantaneously.

Disadvantages:

- Do not have full access to all the methods exposed by the device's operating system.
- Cannot be found on the applications stores.
- Need access to the Internet.

2.1.2 Hybrid applications

The main part of the applications is built using cross-compatible web technologies, such as HTML5, CSS and JavaScript that are later packed into a native container. Such application model is called hybrid.

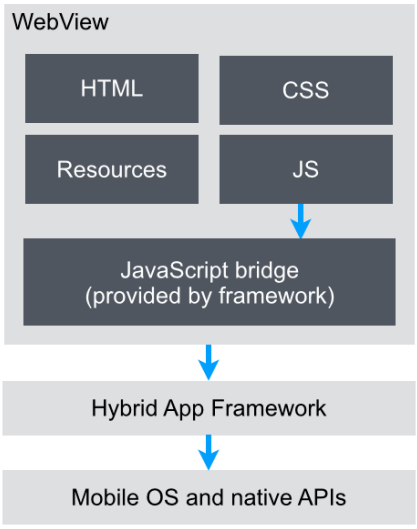


Figure 3 high level architecture of Apache Cordova. [10]

The figure 3 illustrates the high-level architecture of Apache Cordova. Cordova is a platform for web-based hybrid apps. It does the heavy lifting of supporting and interacting with various device-specific APIs and combining the essence of all native APIs in one JavaScript API that is accessible by the hybrid app. [10] The native application contains a hidden browser, which called WebView that is linked to HTML files. Using frameworks for developing hybrid applications, it is possible to wrap the HTML code with native code. Hybrid applications are available to users from the applications Stores and markets. [16]

Advantages:

- Faster and easier to develop.
- Cross-platform.
- Lower budget costs and a huge community of supporters and developers.
- Can be featured and searched for in the app store.

Disadvantages:

- Mobile phones are not fast enough to smoothly run a hybrid app.
- Not all native capabilities are supported.

2.1.3 Native applications

Native applications are specific to a given platform (OSX, Windows, Linux, iOS, Android) using the development tools and languages that the respective platform supports.

Each mobile platform offers developers their own development tools, interface elements and standardized SDK. [5] Native applications are downloaded and installed on the device through applications stores and markets.

The most common mobile development environments and their programming languages are illustrated below. (Table 1)

	IOS	Android	BlackBerry	Windows Phone
Languages	Obj-C, C, C++, Swift	Java (Some C, C++)	Java	C#, VB.NET
Tools	Xcode	Android SDK	BB java Eclipse plugin	VisualStudio, WindowsPhone, Dev Tools
Executable Files	.app	.apk	.cod	.xap
Application Stores	Apple Store	Google Play Market	BlackBerry App world	Windows Phone Market

Table 1 The most common mobile development environments and their programming languages

Advantages:

- They offer the fastest, most reliable and most responsive experience to users.
- Full access to the device's hardware and OS features.
- Very fast and polished, making them great for high-performance apps or games.
- Can be featured and searched for in the application store.

Disadvantages:

- Native apps often cost more to develop and distribute because of the distinct language and tooling ecosystems, which require more investment in developer skills if you need to develop for more than one platform. [3]

- It might take a longer period of time to develop.

2.1.4 Summary

Each way of designing and building apps comes with its own set of benefits and drawbacks. To summarize all features there is presented the comparison table for each mobile development method. (Table 2)

	Native	Hybrid	Mobile web
Languages/tools /intstuments needed for cross-platform apps	Objective-C Java , C , C++ , C# VB.net, Swift	HTML, CSS, Javascript, Mobile development framework	HTML, CSS, Javascript
Distribution	App Store/Market	App Store/Market	Web-browser
Development Speed	Slow	Medium	Fast
Number of applications needed to reach major smartphone platforms	4	1 *Notice:* Some frameworks Maintain only android and iOS platforms)	1
Ongoing application maintenance	Difficult	Moderate	Low
Device access	Full access	Full access, excepting some features. *Notice:* nowadays almost all frameworks maintain full access to OS.	Partial access
Offline access	Yes	Yes	No
Advantages	Opportunity to create applications with rich and clear user interfaces and/or heavy graphics.	Combines the development speed of mobile web apps with the device access and app store distribution of native apps.	Offers fast development, simple maintenance, and full application portability. One mobile web app works on any platform.
Disadvantages	Development time, development cost, no portability (apps cannot be used on other platforms).	Cannot handle heavy graphics. Requires familiarity with a mobile framework.	Cannot handle heavy graphics. Cannot access camera or microphone. The internet connection is required.

Table 2 The overview of mobile applications models [4]

It is important to evaluate the technical and non-technical merits of each mobile application model - especially as it relates to mobile application's requirements. There are several aspects, which are worth paying attention before embarking upon development of a mobile application.

Some of them are presented here:

- Which mobile platforms are to be targeted?
- Is distribution of application via app stores required?
- What are the technical abilities of development team?
- Does application require offline support?
- What technical requirements does the application have?

If the aim is to target more than one platform the fastest and cheapest way is to build a hybrid or web mobile application.

If optimal performance is required, native apps are the best choice because of their ability to take full advantage of the device's software and hardware. [7] On the other hand, it takes more time and resourses to develop high quality products, because for each platform you have to write new a code and use different tools and instruments.

The chart bellow illustrates relation between budget and perfomance among three mobile applications models. (Figure 4)

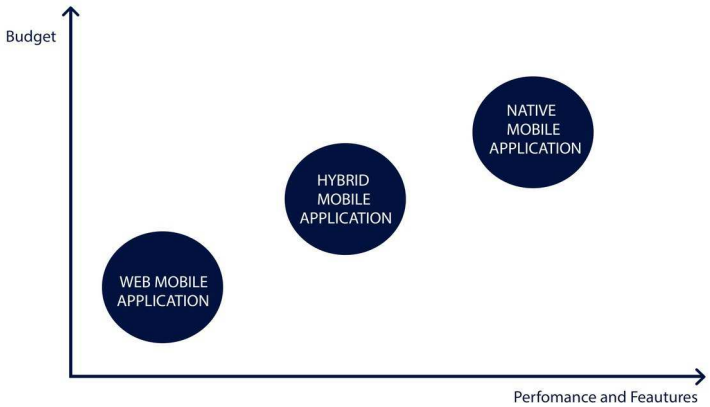


Figure 4 Relation between budget and perfomance [7].

2.2 UI FRAMEWORKS FOR HYBRID MOBILE APPLICATIONS

User Interface Frameworks for hybrid mobile applications are libraries that will enable developers to create a professional grade user experience for their mobile application easily and fast. A hybrid mobile application needs a correct framework choice. Here we present some of the most popular UI framework for hybrid mobile applications.

2.2.1 Ionic Framework

Ionic is an open source front-end SDK for developing hybrid mobile apps with HTML5, CSS3 and Javascript. [12]
Ionic is a front-end framework built on top of Apache Cordova (Phone Gap) that allows web developers to deploy mobile apps using AngularJS, which allows developers to develop serious, robust mobile applications.[8] It supports iOS 6+ and Android 4.1 + platforms.
The UI is clean, simple and based on functionality and speed. Applications are designed to work and display beautifully on all mobile devices and the framework offers a number of mobile components all based on a stunning extensible base theme.

Ionic allows customizing settings, such as headers, tabs, transitions, toggles, action sheets, back-buttons, checkboxes, navigation-buttons, and fonts, for each major platform.

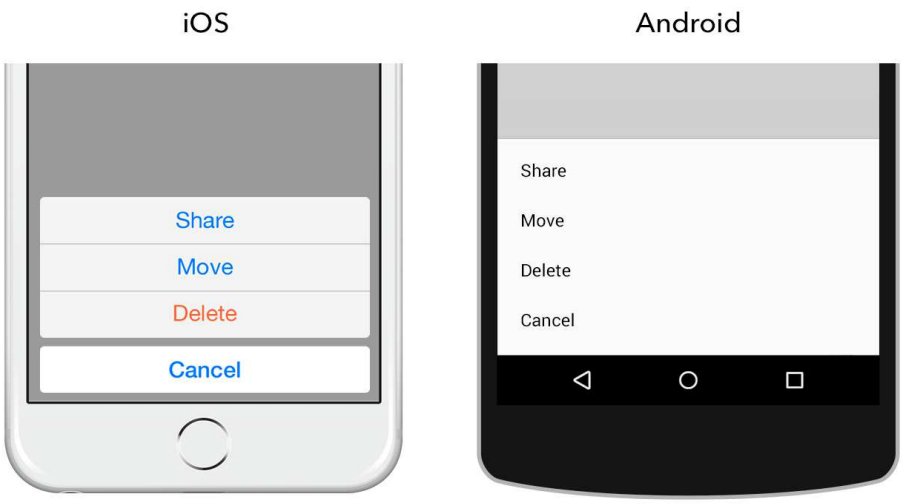


Figure 5 The difference between action sheets for iOS and Android platform in Ionic [9]

The figure 5 illustrates the difference between action sheets for iOS and Android platform. Ionic provides unique UI for each platform. It automatically looks like iOS action sheet on iOS and Android action sheet on Android.

2.2.2 Famo.us

Famo.us is a free, open source JavaScript framework that helps to create smooth, complex mobile applications. [14]
Famo.us is made for hybrid mobile development, it supports iOS and Android platforms.

This kind of framework is suitable for game and animation applications.
This is because Famo.us is the only JavaScript framework that includes an open source 3D layout engine fully integrated with a 3D physics animation engine that can render to DOM, Canvas, or WebGL.[14]

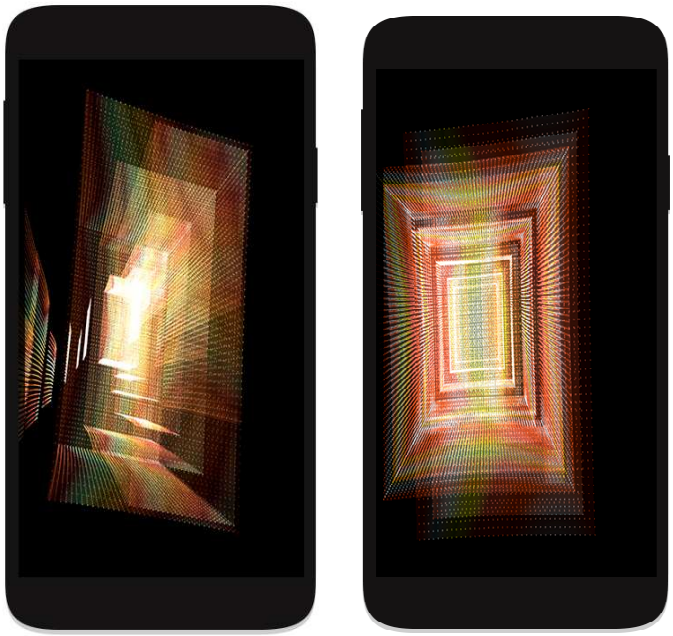


Figure 6 Example of Famo.us rendering to WebGL

The figure 6 is an example of Famo.us rendering to WebGL. It is an artistic transitioning from a sphere to a cube and back. To see it in action there is a reference to famo.us showcase in codepen.io -http://codepen.io/befamous/.

However, Famo.us has some disadvantages. This kind of framework does not provide basic skeletons (menus, tabs, basic app structure/layout). Moreover, it is the newest framework on the market nowadays. For this reason the documentation is pure and it is hard to find real examples and show cases.

2.2.3 Kendo UI

Kendo UI is a product of Telerik and it is not free of charge. Kendo is based on HTML5 and JavaScript and the features of this framework include a complete set of HTML5 widgets and features.

Kendo UI is comes in complect with AngularJS integration and Bootstrap support. Because this is a commercial product it comes in two separate packages: Kendo UI Core- open source and Kendo UI Professional - with additional charge. From the visual perspective, Kendo UI provides an adaptive theming. Kendo UI will recognize application platform and then makes specific adjustments according to it. [13]

Figure 7 The TabStrip for iOS and Android in Kendu UI [13]

The same application will look different on iOS and Android. On iOS, for example, the TabStrip is at the bottom, while on Android it is moved to the top. (Figure 7)
Currently Kendo UI Mobile supports iOS, Android, BlackBerry, and Meego.
Also, the big advantage of the framework is that Kendo UI provides the Telerik AppBuilder – redactor, which makes the development process faster and more comfortable.
Built-in simulator makes it easy to debug and deploy applications for any mobile device.

2.2.4 Summary

It is important to notice that it is only the small part of UI frameworks for developing hybrid mobile applications. In the last year the big number of UI frameworks joined the list, and many more are on the way, presenting more efficient and more performant hybrid experience. The table 3 bellow summarizes all advantages and disadvantages of the three UI frameworks: Ionic framework, Famo.us and Kendo UI mobile framework.

	Ionic	Famo.us	Kendo UI mobile
Advantages	Full UI components, clean and simple UI, native-like UX, fast start, good documentation, widely used .	Provide great opportunity to create powerful animations. Suitable for game developing.	Full UI components, rich ,native-like UX, faster development time. Easy Unit testing, developing with app builder redactor, supporting the variety of mobile platforms.
Disadvantages	Slow translations and animations. Supporting only Android and iOS platforms.	Not providing basic skeletons. Not widely use. It is hard to find real examples and show cases. Supporting only Android and IOS platforms.	Paid model of licensing.

Table 3 The advantages and the disadvantages of the UI frameworks

To draw the conclusion, it can be said that choice of user interface framework for mobile application is depends on result, which exactly needed to be achieved. If the target is to make impressive animations and transition, the best option will be using Famo.US framework. And if the aim is to make native like - UX, the Kendo UI or Ionic framework will be suitable.

3. PROTOCOLS AND STANDARDS. SOAP vs REST.

3.1 REST

REST stands for Representational State Transfer. REST was introduced and defined in 2000 by Roy Fielding in his doctoral dissertation.[22] REST is often used in mobile applications, social networking Web sites and automated business processes. REST is an architecture style for designing distributed systems. It relies on a stateless, client-server, cacheable communications protocol.[21] Simple HTTP is used to make calls between machines.

3.1.1 REST architecture constraints

REST constraints are design rules that are applied to establish the distinct characteristics of the REST architectural style.[25]

All REST architecture constraints bellow are taken directly from the source for evaluation purposes.

- *Uniform Interface*

All interactions should be built around a uniform interface, which supports all the interactions with resources by providing general and functionally sufficient set of methods.

- Resource Identifications.

Each URL identifies a resource and has a strictly defined format.

- Resource representation.

It is how the resource will return to the client. This representation can be in HTML, XML, JSON, TXT, and more. [26]

- Self-explanatory answer.

The passage of meta information is needed in the request and response. Some of this information are HTTP response code, Host, Content-Type. [26]

- *Cacheable*

Because many clients access the same server, and often requesting the same resources, it is necessary that these responses might be cached, avoiding unnecessary processing and significantly increasing performance.[26]

- *Client- server*

This constraint is based on the principle of Separation of concerns. The purpose of this division is to separate architecture and responsibilities in both environments. This constraint requires the existence of a client component that sends requests and a server component that receives requests. (Figure 8) [26]

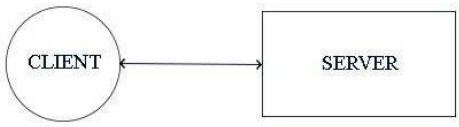


Figure 8 Client-server architecture

- *Stateless Interaction*

Each interaction between a client and a server has to be entirely self-contained. There should be no client state maintained on the server which would allow an interaction to depend on both the exchanged representation and on the session associated with the client. This constraint is important to ensure that the scalability of servers is bound only by the number of concurrent client requests and not by the total number of clients that they have to interact with. [20]

- *Layered System*

The layered system constraints enable network-based intermediaries such as proxies and gateways to be transparently deployed between a client and server using the uniform interface. A network-based intermediary will intercept client-server communication for a specific purpose. Network-based intermediaries are commonly used for enforcement of security, response cashing and load balancing. [19]

- *Code-On-Demand*

This condition allows the customer to run some code on demand, that is, to extend part of server logic to the client, either through an applet or scripts. [26]

3.1.2 HTTP methods

HTTP-request type (also called HTTP-method) instructs the server to what action is needed to make to the resource. RESTful applications use HTTP requests to post data (create and/or update), read data, and delete data. REST uses HTTP for all four CRUD (Create/Read/Update/Delete) operations.

GET - method retrieves information. Get is a safe method – it never modifies resources.

POST - Requests that the resource at the URI does something with the provided entity. Often POST is used to create a new entity, but it can also be used to update an entity.

PUT - Stores an entity at a URI. PUT can create a new entity or update an existing one.

DELETE -A client uses DELETE to request that a resource be completely removed from its parent. Once a DELETE request has been processed for a given resource, clients can not longer find the resource.

3.1.3 Response status codes

REST API-s use the Status – Line part of an HTTP response message to inform clients of their request's overarching result. HTTP defines forty standard status codes that can be used to convey the results of a client's request. The status codes are divided into the five categories presented in the table 4.

Category	Description
1xx: Informational	Communicates transfer protocol-level information.
2xx: Success	Indicates that client's request was accepted successfully.
3xx: Redirection	Indicates that the client must take some additional action in order to complete request.
4xx: Client Error	This category of errors status codes points the finger at clients.
5xx: Server Error	The server takes responsibility for these error status codes.

Table 4 HTTP Response status codes.

3.2 SOAP

The Simple Object Access Protocol (SOAP) based web services interact over XML data which is well defined and standard message format that uses web service description language (WSDL). SOAP has the following features: protocol independence, language independence, platform and operating system independence.

SOAP defines a way to move XML messages from point A to point B. The Figure 9 illustrates a simple one-way message where the sender does not receive a response. The receiver could, however, send a response back to the sender (Figure 10). Usually HTTP is used to exchange SOAP messages. However, it should be noticed that SOAP is protocol independence, so it is not necessary to use SOAP with the HyperText Transfer Protocol transport. There is an actual specification for using SOAP over Simple Mail Transfer Protocol (SMTP) and others. (Figure 9)

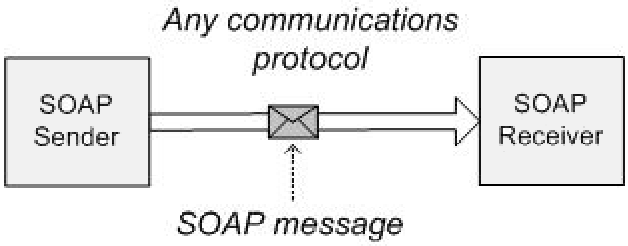


Figure 9 Simple SOAP messaging [27]

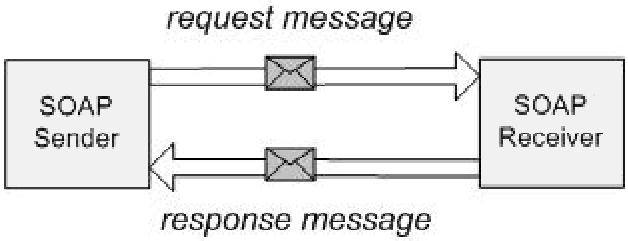


Figure 10 Request/response message exchange pattern [27]

SOAP allows different message exchange patterns, of which request/response is just one. Other examples include solicit/response (the reverse of request/response), notifications, and peer-to-peer conversations. [27]

3.2.1 SOAP MESSAGES

SOAP is based on message exchanges. A SOAP message is an ordinary XML document containing the following elements: An *Envelope* element that identifies the XML document as a SOAP message, a *Header* element that contains header information, a *Body* element that contains call and response information and a *Fault* element containing errors and status information. Figure 11 illustrates the SOAP message structure.

Figure 11 SOAP message structure

Every envelope element must contain exactly one Body element. The body element may contain as many child nodes as are required. The contents of the Body element are the message. If an envelope contains header, it must be the one header and header location is always before body. The header can contain, like a body element, any valid, well-formed XML. Each element contained by the Header is called Header block. The purpose of header blocks is to communicate contextual information relevant to the processing of a SOAP message. If a Fault element is present, it must appear as a child element of the Body element. A Fault element can only appear once in a SOAP message SOAP Fault Codes. (Table 5)

Error	Description
VersionMismatch	Found an invalid namespace for the SOAP Envelope element.
MustUnderstand	An immediate child element of the Header element, with the mustUnderstand attribute set to “1”, was not understood.
Client	The message was incorrectly formed or contained incorrect information.
Server	There was a problem with the server so the message could not proceed.

Table 5 SOAP Fault Codes [17]

3.3 SUMMARY

First of all, let us summarize information about REST and SOAP. Below is the comparison between the two approaches.

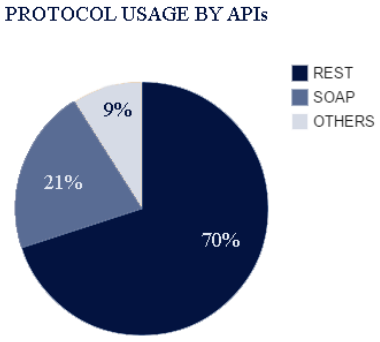
One of the major benefits of RESTful API is that it is flexible for data representation, for example you could serialize your data in either XML, JSON format or any other format.

RESTful services are also lightweight - they do not have a lot of extra XML markup. To invoke RESTful API all what is needed is a browser or HTTP stack and the machine or device with the Internet connection.

RESTful applications typically use normal HTTP methods instead of a big XML format describing everything.

Moreover, it is important to keep in mind that SOAP has significantly more advanced tooling support in some languages, and other languages strongly prefer REST. For example, if it is a Java client for service, there will be no problems to use SOAP. On the other hand, if it is a JavaScript client then it will be better to deal with the REST interface. JavaScript works great with REST.

However, there are two particular scenarios where SOAP may win out over REST. If application needs a high level of end-to-end security and absolute distributed transactional reliability, SOAP extensions WS-Security, WS-Reliable Messaging and WS-Atomic transactions make meeting such requirements far easier to accomplish than in REST. For instance, the classic application in this regard is banking account transfers. Also, SOAP supports distributed communication whereas REST assumes communication between only two endpoints.



The pie chart (figure 12) illustrates the protocol usage statistics by popular APIs. According to the diagram, it could be seen that today the API world is moving more and more to REST based approaches. As of the end of March, REST accounted for 70% of the APIs. (Retrieved from: Programmableweb).

Figure 12 Protocol usage by APIs

The table bellow illustrates the brief comparison between SOAP and REST. (Table 6)

SOAP	REST
A XML- based protocol.	An architectural style protocol.
SOAP has WSDL, which is an XML-based language for describing Web services and how to access them.	REST has no strong interface definition and structure validation.
SOAP only permits XML format. Also, other data types could be encapsulate into XML format.	REST permits many different data formats – XML, plain text, JSON, HTML and others.
Invokes services by calling RCP method.	Simply calls services via URL path.
Transfer is over HTTP. Also it could be used other protocols such as SMTP, FTP and others.	Transfer is over HTTP only.
JavaScript can call SOAP, but it is difficult to implement.	Easy to call from JavaScript.
Caching situations: If the information needs to be cached.	REST services are easily cacheable.
Performance is not great compared to REST.	Less CPU intensive, leaner code.
Using SOAP is little bit complex, It takes more time to learn and develop.	REST is much more lightweight and can be implemented using almost any tool, leading to lower bandwidth and shorter learning curve.

Table 6 Comparison between SOAP and REST

Each protocol has definite advantages and equally problematic disadvantages. The choice to use SOAP or REST is dependent on a number of factors including security, transactional, performance, development and other application requirements.

4. POLLING TECHNIQUES

In this chapter we shall consider request - response model and its two types of receiving a response from server. To begin with, let us introduce the notion of request – response communication model between applications. This type of communication model is generally used by HTTP and all extended protocols based on HTTP, for instance, as discussed in the previous chapter, the REST and SOAP. The main principle of work of this model is: a client, typically a web browser, sends a request for a resource to a server, and the server sends back a response corresponding to the resource. So there are two different ways to poll servers.

4.1 TRADITIONAL POLLING

Traditional polling is a way to get data generated from the server side at known intervals. The definite time interval is set to execute the request to get the server events as soon as possible, the polling interval (time between requests) must be as low as possible. . However, it leads to a problem that half of the request is useless, a waste of bandwidth and server resources. After that, the server creates a response for each request ASAP and sends it back. The new connection to the server must be opened each time the request method is called. (Figure 13)

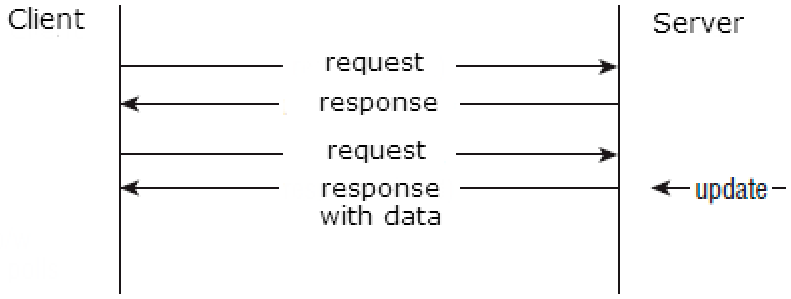


Figure 13 Traditional polling

This method is simple in realization. Although it has some disadvantages. There may be a delay between the appearance and receiving the data, just in the amount of seconds between requests and as it has been said before, it is a waste of bandwidth and server resources due to useless requests.

4.2 LONG-POLLING REQUEST

The main idea of the long-pulling request is: the client polls the server requesting new information and the server holds the request open until new data is available. When there is new information, the server sends it to the client. The client receives the new information and immediately sends another request to the server, starting the waiting process of receiving updates again. (Figure 14)

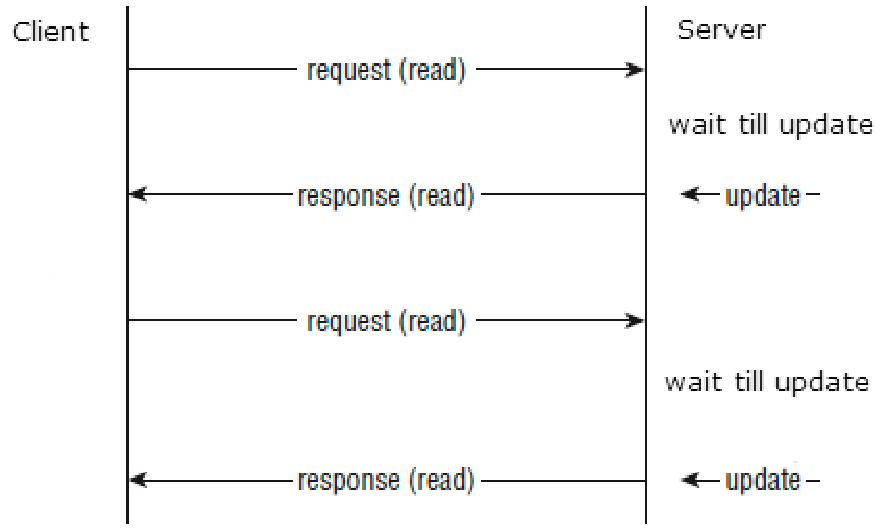


Figure 14 Long-polling request

Disadvantages:

- Long-polling technique works great only with a couple of connections. In other words, Long polling takes significantly more server resources. Server has to handle a big amount of long-lived connections. However, there is a limit - the maximum number of open sockets. Different OS have different limit in number of open sockets.

Advantages:

- You are notified when the server event happens with no delay.
- Instead of sending a request, for example, each five second to check for new messages, requests will be sent continuously on the long-term. For example, instead of 12 requests in one minute, there will be approximately only 1-2 requests.

5. THE CASE STUDY

In this chapter there will be a detailed consideration of project implementation process. The chapter is devoted to a detailed examination of evolution of the project concept, its architecture, design, justification of the design, toolset and process choices.

5.1 PROJECT's TARGET

The "Online Chess Tournaments" system is the project for Chess Rating Agency (International) Ltd for FIDE. FIDE is a French organization and abbreviation means - Fédération Internationale des Échecs. In English it reads: World Chess Federation.

Chess Rating Agency (International) Ltd acts as a customer for the second time. It was already developed a web application to broadcast international tournaments – "Online Chess Tournaments Broadcasting" system. The main idea of the system is to enable users to watch international tournaments and chess games online in real time. Also there is an opportunity to see games already finished. User could see the chess board with the state of each move and its notation.

Now it is needed to implement a system for mobile phone users. The main difference between the web version of the system that for the web application a user must always have an internet connection, however, the mobile application would be implemented in such a way that the user will need to have the internet access only to receive updates or to watch the game in real time. The functionality might not be full without the Internet connection. It means that some updates could not be received without the Internet access. But the application's work will not be disrupted.

Furthermore, the second difference between web applications is that mobile application is responsive and adaptive for mobile screens. Mobile applications are more comfortable to use on mobile devises instead of web applications. Also, the aim is to implement the cross-platformed application. It means that it would be available on two major mobile platforms: Android and IOS .

5.2 PROJECT DESCRIPTION

This chapter presents a list of client-requirements that would help us understand the targets and functionality of the project more clearly. Also it explains the technical choices which have been made in the project architecture, such as selection of application model, framework and the communication protocol with the server side.

5.2.1 Clients-Requirements

Here client-requirements made by customer and mockups of the application to visualize those requirements are shown.

1. The application should show a list of tournaments. User has an opportunity to search and select a particular chess tournament from the list of all possible tournaments. (Figure 15)

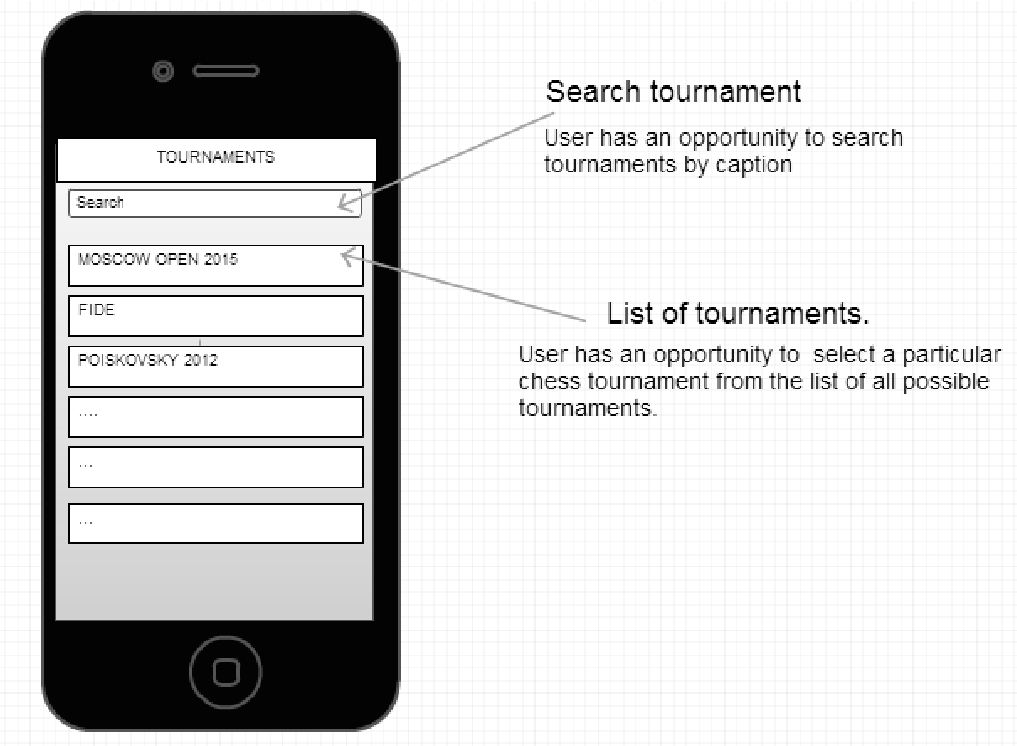


Figure 15 List of tournaments screen mockup

2. The application should display a list of all rounds of the selected tournament with the date of the round in UTC time format. The user should be able to select a particular chess round. (Figure 16)

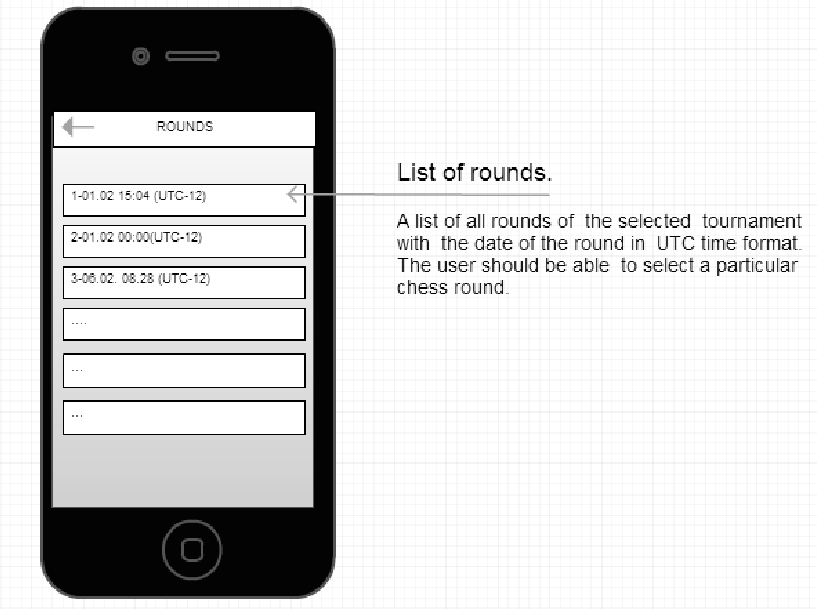


Figure 16 List of rounds screen mockup

3. The application should specify a list of all games of the current tournament or round. The list of games presents the table with information of players (first name, last name, federation , rating, FIDE-ID, the last move, the last move evaluation using the chess engine or if game is over-game score. Also there is the availability to choose a particular chess game. (Figure 17)

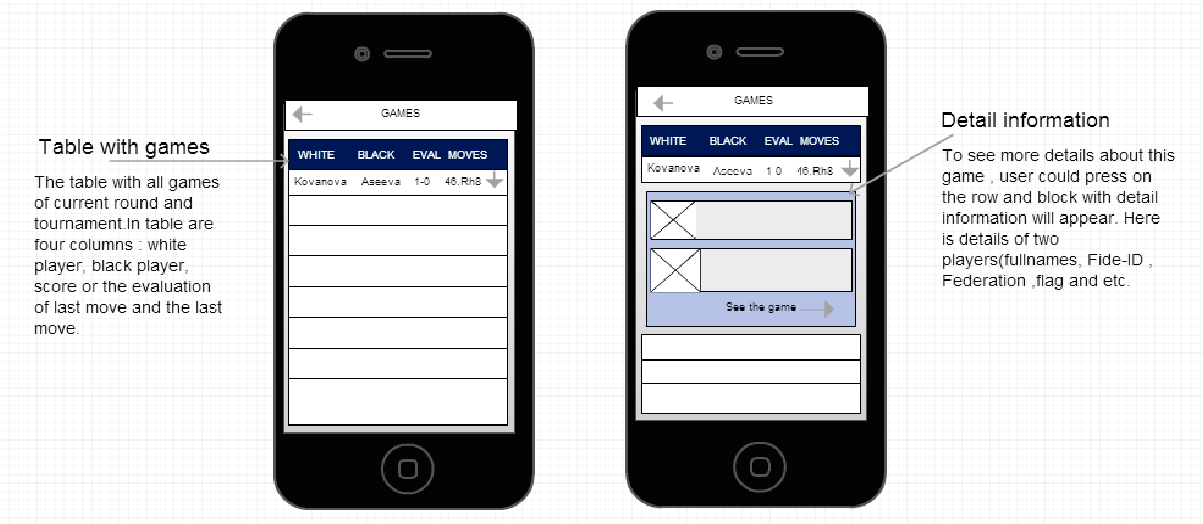


Figure 17 List of game screen mockup

4. After selecting the particular game the current game situation (chessboard, chessboard state) should be displayed. User should be able to see game history on chessboard. Position on the board should correspond to the chosen game move. Move is chosen from table with content of all moves in current game.

5. Integration with the block "Comments" - the application should show comments in the "body" of the game notation.

6. Integration with the block "Analyses from chess engine" – the application should show an evaluation made by the chess engine.

7. Automatic update for moves and the board position during online-broadcasting.

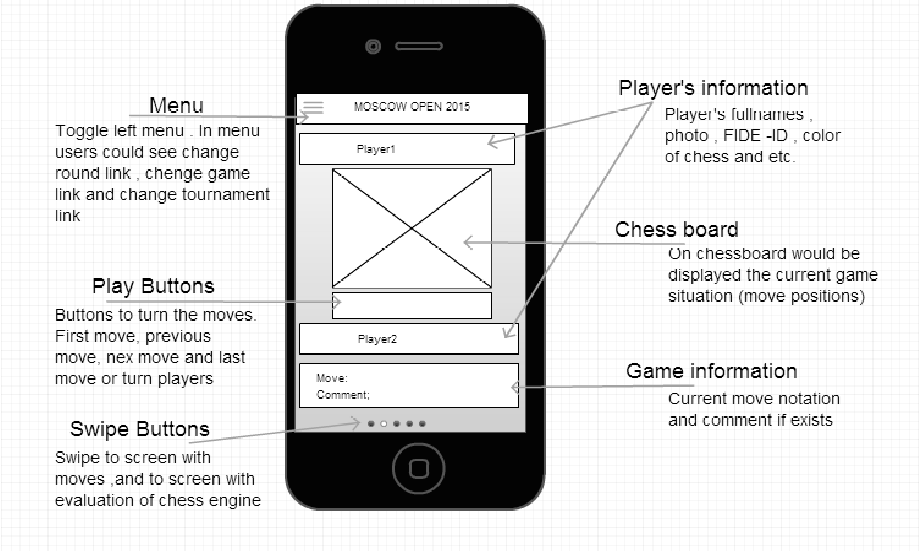


Figure 18 chessboard screen mockup

Figure 19 List of move and chess engine screen mockups

5.2.2 Selection of application model

The choice of mobile application will depend on the characteristics summarized of three main mobile applications models: native, hybrid and mobile web application.

One of the aims of “Online Chess Tournaments” system ensure the applications work on multiple systems with a short period of time.

For that reason it will be more suitable to use mobile web application or hybrid application If we select to develop native applications it would require twice as much time to implement applications for two main mobile platforms such as Android and iOS, because it would be necessary to write two different mobile applications in different programming languages, one for each platform.

The main advantages of the native mobile is clear UI and UX, maintaining of heavy graphics and full access to all hardware and OS tools. However, all these characteristics are not important in “Online Chess Tournaments“ system. This interface of this project does not contain heavy graphics, the UI will be simple without any heavy load on the system. Nonetheless-, nowadays there are a lot of frameworks which provide clear and native-like UI for mobile applications.

Furthemore, no specific hardware and OS functions and features for the implementation the system are needed. As a consequence, the native mobile application is not applicable in this case.

Now let’s compare and decide which model will be more suitable – mobile web application or hybrid mobile application. The technologies of hybrid and mobile web applications are quite similar. So this factor could not influence our choise. However, the web model outperforms the hybrid model because it maintains almost all platforms. All is needed is a browser and an Internet access, while not all frameworks of the hybrid mobile application maintain all mobile platforms. The majority of frameworks make application available only for Android and IOS, leaving out Windows Mobile. Also the development speed of the web mobile applications is faster than in other models. Although all these features of mobile web apps are lost in this case, because for using web mobile app the Internet access is required. But one of the aims of the system is stable work without Internet access. Also, the disadvantage of the web model is that the application is not available the platform-specific marketplace.

For these reasons, the choice has been made in favor of hybrid mobile applications.

5.2.3 Selection of UI framework

As the hybrid mobile application was chosen, it is necessary to choose the correct UI framework to implement hybrid mobile application. As it has been said before, nowadays there is a variety of different UI frameworks for hybrid and web mobile applications. The majority of frameworks are quite similar. For this reason sometimes it is difficult to make a choice. But in chapter about UI frameworks were presented the three most popular frameworks for discussion. Therefore, this chapter will explain the selection on the basis of the results and conclusions of chapter “UI frameworks”.

For the realization of the project, it was decided to choose Ionic as the cross-platform hybrid mobile development framework. Ionic framework is free licensed, what means no additional costs for the use of the framework, and open source.

First of all lets decide if Famo.us will be suitable for realisation the project.

As mentioned above, Famo.us is the one framework which provides wide graphics maintaining. It is an excellent choice for implementation games. However, The “Online chess tournaments” application is not a game with 3d graphics or complicated animation.

This is a first reason why Famo.us framework does not gain advantage for using in the project. However, with Famo.us it is also possible to do casual mobile applications. But it does not provide basic layouts. It will be very difficult to design application with this framework. For instance, in “Online Chess Tournaments” mobile application it is designed to do swipe between three screens: screen with chessboard, screen with moves and screen with evaluation of chess engine. Also the application uses the toggle left menu for navigation system. It will be more suitable to not program it by yourself, but to use the basic skeletons. So the Famo.us framework is not suitable for usage in project.

As a consequence, Kendo UI mobile and IONIC are more suitable for this kind of projects. Kendo UI has a lot of advantages such as maintaining a variety of platforms, while Ionic only supports Android and IOS platforms. Also, the developing with Kendo UI is much easier because it provides developing tools such as a special redactor for developing. However, the Kendo UI mobile is not free of charge. It has a free demo version, but this version has limited opportunities.

Still, one supporting argument for Ionic is that Ionic framework provides the fastest, most reliable and most responsive user experience and its UI is very close to native.

The choice was made in honor of the Ionic. Despite the fact that Kendo UI has a lot of advantages, client was not ready to cover the additional costs that using Kendo would have caused.
Currently platforms, which are supported by Ionic are iOS and Android. It is quite enough. Because iOS and Android make up the largest part of the market , this is still acceptable for a client.

5.2.4 Selection of communication protocol.

The communication protocols and standards are a big diversity. In chapter 3 two approaches for communication between client and server – REST and SOAP were presented. In this chapter we shall discuss which approach is better for “Online Chess Tournaments” system.

First of all, it should be said that because of hybrid application model was chosen, it would be implemented using JavaScript and jQuery. According to the summary of difference between SOAP and REST from the table 6 (chapter 3), it could be seen that REST is easy to call from JavaScript, it simply calls services via URL path, while SOAP is difficult to implement using JS and jQuery.

Secondly, our system does not need the high level end-to-end security and absolute distributed transactional reliability for which SOAP will be the best approach.

REST permits many different data formats. Because data format will be JSON, the SOAP protocol will not be suitable for realization of this system because SOAP only permits XML data format.

Obviously, the REST is the most suitable way of communication between client and server side because it is easier and faster to implement than the communication using SOAP protocol.

5.2.5 Selection of polling technique

After the choice of communication standard has been made, it is necessary to choose what kind of polling technique to use in project implementation. From two types of polling the server the choice was made towards traditional polling for several reasons. As explained in chapter about polling techniques, the long- polling request is better than traditional polling for a lot of reasons. However, it would only work excellent only with a small amount of connections. The Chess Rating Agency (international) Ltd is a global company. In our case, the number of clients increases at a fast pace. So for this reason it will take a lot of server resources to hold all connection in open state. The limits of open sockets are also one of the barriers in implementing the large-scale use of the system.

Moreover, long-polling request require an open connection. This leads to one more problem. Sometimes mobile devices rapidly switch between WIFI and cellular networks or lose connections, and the IP address changes. It is problematic for server to automatically re-establish connections.

With traditional polling there will be an amount with useless requests which leads to the waste of bandwidth and server resources. However, to compare with long-polling request, if there are many connections at once, the server load by traditional polling will be much less. Also, in chapter about traditional polling it was said that one of the disadvantages is that it might be a delay between the appearance and receiving the data. But in “Online Chess Tournaments” system the data will not be updated so often, except when a chess game is translated in the real time.

After discussing all pro- and contra arguments, it is possible to conclude that traditional polling communication type is more suitable for our project than the long-polling way.

5.3 PROJECT IMPLEMENTATION

5.3.1 Domain Object Model

A domain object model of the “Online chess tournament” system is presented bellow. It describes the various entities, their attributes, roles, and relationships. (figure 20)

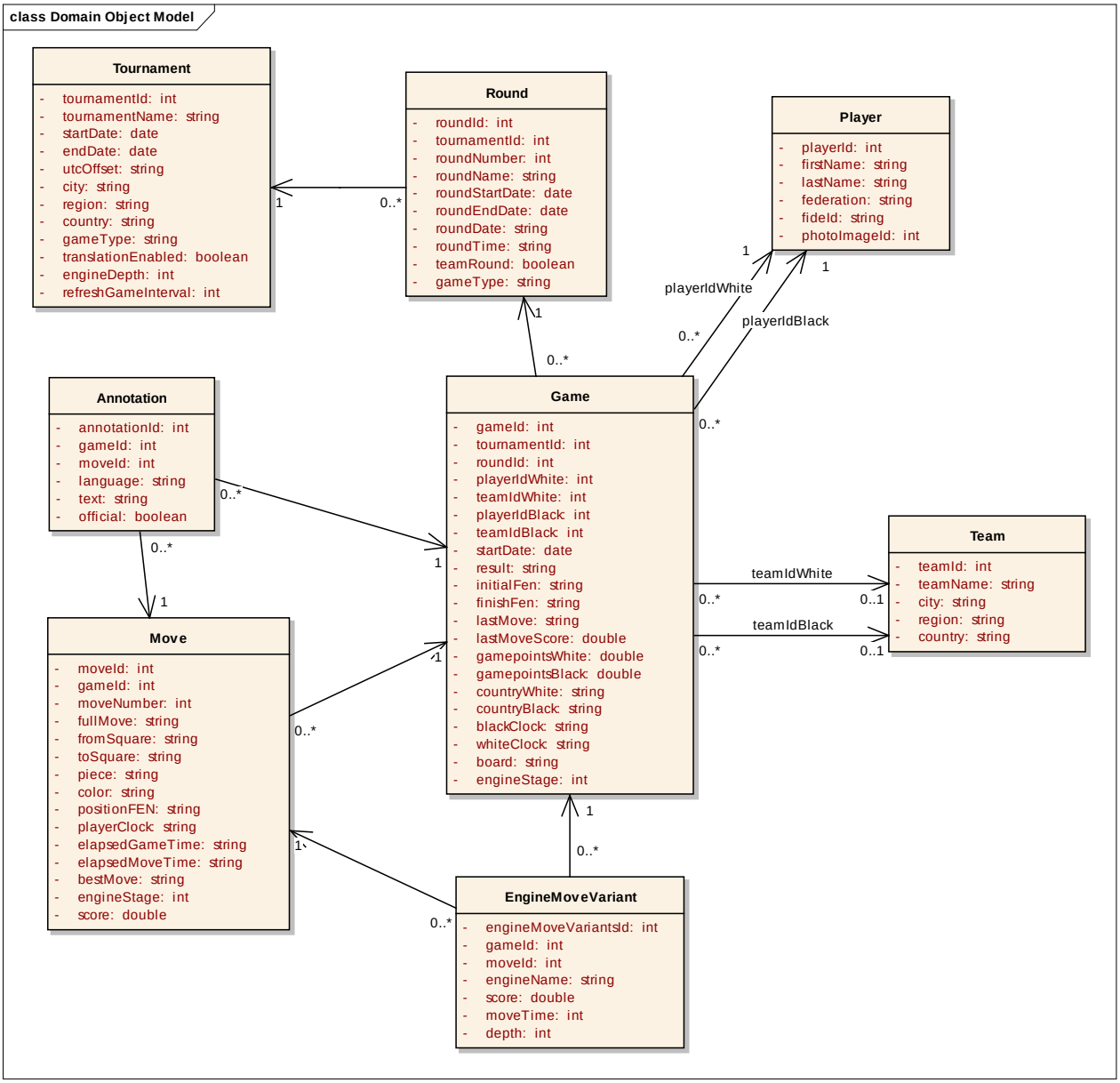


Figure 20 Domain object model

Tournament
- An international chess tournaments for broadcasting.

Attribute	Type	Description
tournamentId	int	Tournament identifier
tournamentName	string	Tournament name
startDate	date	Start date of tournament
endDate	date	End date of tournament
utcOffset	string	This is the offset to add to UTC to get local time. In hours.
city	string	Tournament city
region	string	Tournament region
country	string	Tournament country
gameType	string	Tournament game type
translationEnabled	boolean	Is translation enabled or not
engineDepth	int	Chess engine analyse depth
refreshGameInterval	int	Refresh game interval in ms

Table 7 tournament object's attributes description

Round – a chess round of the selected tournament.

Attribute	Type	Description
roundId	int	Round identifier
tournamentId	int	Reference to the tournament
roundNumber	int	Round number
roundName	string	Round name
roundStartDate	date	Start date of round
roundEndDate	date	Round end date
roundDate	string	Date: the starting date of the game, in YYYY.MM.DD form. "??" are used for unknown values.
roundTime	string	Time: Time the game started, in "HH:MM:SS" format, in local clock time.
teamRound	boolean	Is it a team round or not
gameType	string	Game type

Table 8 round object's attributes description

Game – a chess game of the selected round.

Attribute	Type	Description
gameId	int	Game identifier
tournamentId	int	Reference to the tournament
roundId	int	Reference to the round
playerIdWhite	int	Reference to the player of white pieces
teamIdWhite	int	Reference to the team of white pieces
playerIdBlack	int	Reference to the player of black pieces
teamIdBlack	int	Reference to the team of black pieces
startDate	date	The starting date and time of the game, in local clock time
result	string	the result of the game. This can only have four possible values: "1-0" (White won), "0-1" (Black won), "1/2-1/2" (Draw), or "*" (other, e.g., the game is ongoing).
initialFen	string	The initial position of the chess board, in Forsyth-Edwards Notation. This is used to record partial games (starting at some initial position).
finishFen	string	The last position of the chess board, in Forsyth-Edwards Notation. This is used to record partial games (starting at some initial position)
lastMove	string	Last move

Attribute	Type	Description
lastMoveScore	double	Last move score
gamepointsWhite	double	Current result of the player playing of white pieces (the number of points scored in the tournament by default - 0)
gamepointsBlack	double	Current result of the player playing of black pieces (the number of points scored in the tournament by default - 0)
countryWhite	string	the three-letter International Olympic Committee code for the country of player of white pieces
countryBlack	string	the three-letter International Olympic Committee code for the country of player of black pieces
blackClock	string	Black player clock in game
whiteClock	string	White player clock in game
board	string	Board
engineStage	int	Engine calculation stage: 0 - none, 1, 2

Table 9 Game object's attributes description

Player - a player of white or black pieces.

Attribute	Type	Description
playerId	int	Player identifier
firstName	string	First name of player
lastName	string	Last name of player
federation	string	the three-letter International Olympic Committee code for the country
fideId	string	FIDE player identifier
photoImageId	int	Reference to the image

Table 10 Player object's attributes description

Team – a team of white or black pieces.

Attribute	Type	Description
teamId	int	Team identifier
teamName	string	Team name
city	string	City of team
region	string	Team region
country	string	Team country

Table 11 Team object's attributes description

Move – a move of the selected game

Attribute	Type	Description
moveId	int	Move identifier
gameId	int	Reference to the game
moveNumber	int	Move number
fullMove	string	Full move
fromSquare	string	From square
toSquare	string	To square
piece	string	Piece
color	string	Color – black or white
positionFEN	string	Position after move in FEN
playerClock	string	The clock command. This is the time displayed on the players clock.
elapsedGameTime	string	Elapsed Game Time : the egt command. The elapsed time that a player has used for all moves in the game up to that point.
elapsedMoveTime	string	Elapsed Move Time : the emt command

Attribute	Type	Description
bestMove	string	Best move in UCI notation
engineStage	int	Engine calculation stage: 0 - none, 1, 2
score	double	move score

Table 12 Move object's attributes description

EngineMoveVariant – variants for move made by chess engine.

Attribute	Type	Description
engineMoveVariantsId	int	Engine move variants identifier
gameId	int	Referemce to the Game
moveId	int	Referemce to the Move
engineName	string	Name of engine
score	double	Variant score
moveTime	int	Engine move time
depth	int	Engine depth calculation

Table 13 EngineMoveVariant object's attributes description

Annotation – an annotation of the move or of the game.

Attribute	Type	Description
annotationId	int	Annotation identifier
gameId	int	Reference to the Game
moveId	int	Reference to the Move
language	string	Annotation language.
text	string	Annotation text
official	boolean	Is annotation official or not

Table 14 Annotation object's attributes description

5.3.2 Description of receiving information process.

In this chapter examines the process of receiving information about tournaments, rounds and game data for games broadcasting in two different cases: with an Internet connection and without.

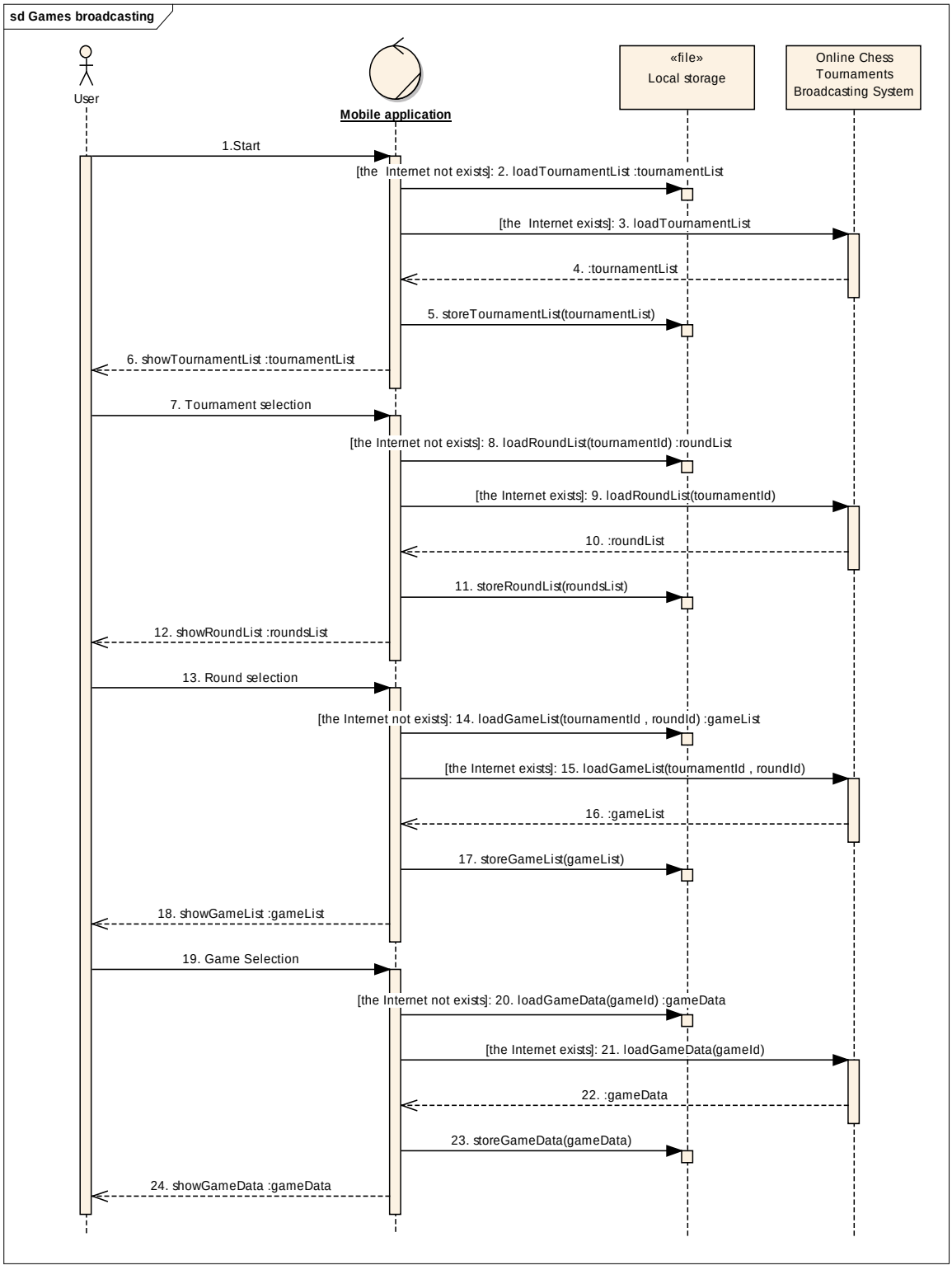


Figure 21 Game broadcasting

Internet connection exists.

1. An user starts an application (1).
2. A request to load a tournament list is sent to the “Online chess tournaments broadcasting” system (3).
3. The tournament list is returned from the “Online chess tournaments broadcasting” system (4).
4. The tournament list is saved to the local storage (5).
5. The tournament list is shown to the user (6).
6. The user selects the tournament (7).
7. A request to load a round list for the selected tournament is sent to the “Online chess tournaments broadcasting” system (9).
8. The round list is returned from the “Online chess tournaments broadcasting” system (10).
9. The round list is saved to the local storage (11).
10. The round list is shown to the user (12).
11. The user selects the round (13).
12. A request to load a game list for the selected tournament and round is sent to the “Online chess tournaments broadcasting” system (15).
13. The game list is returned from the “Online chess tournaments broadcasting” system (16).
14. The game list is saved to the local storage (17).
15. The game list is shown to the user (18).
16. The user selects the game (18).
17. A request to load a game data for the the selected game is sent to the “Online chess tournaments broadcasting” system (21).
18. The game data is returned from the “Online chess tournaments broadcasting” system (22).
19. The game data is saved to the local storage (23).
20. The game data is shown to the user (24).

Alternative: Internet connection not exists.

1. An user starts an application (1).
2. A request to load a tournament list is sent to the local storage. The tournament list is returned from the local storage (2).
3. The tournament list is shown to the user (6).
4. The user selects the tournament (7).
5. A request to load a round list for the selected tournament is sent to the local storage. The local storage returns the round list (8).
6. The round list is shown to the user (12).
7. The user selects the round (13).
8. A request to load a game list for the selected tournament and round is sent to the local storage. The local storage returns the game list (14).
9. The game list is shown to the user (18).
10. The user selects the game (19).
11. A request to load a game data for the selected game is sent to the local storage. The local storage returns the game data (20).
12. The game data is shown to the user (24).

5.3.3 Description of updating information processes.

Updating a tournament list

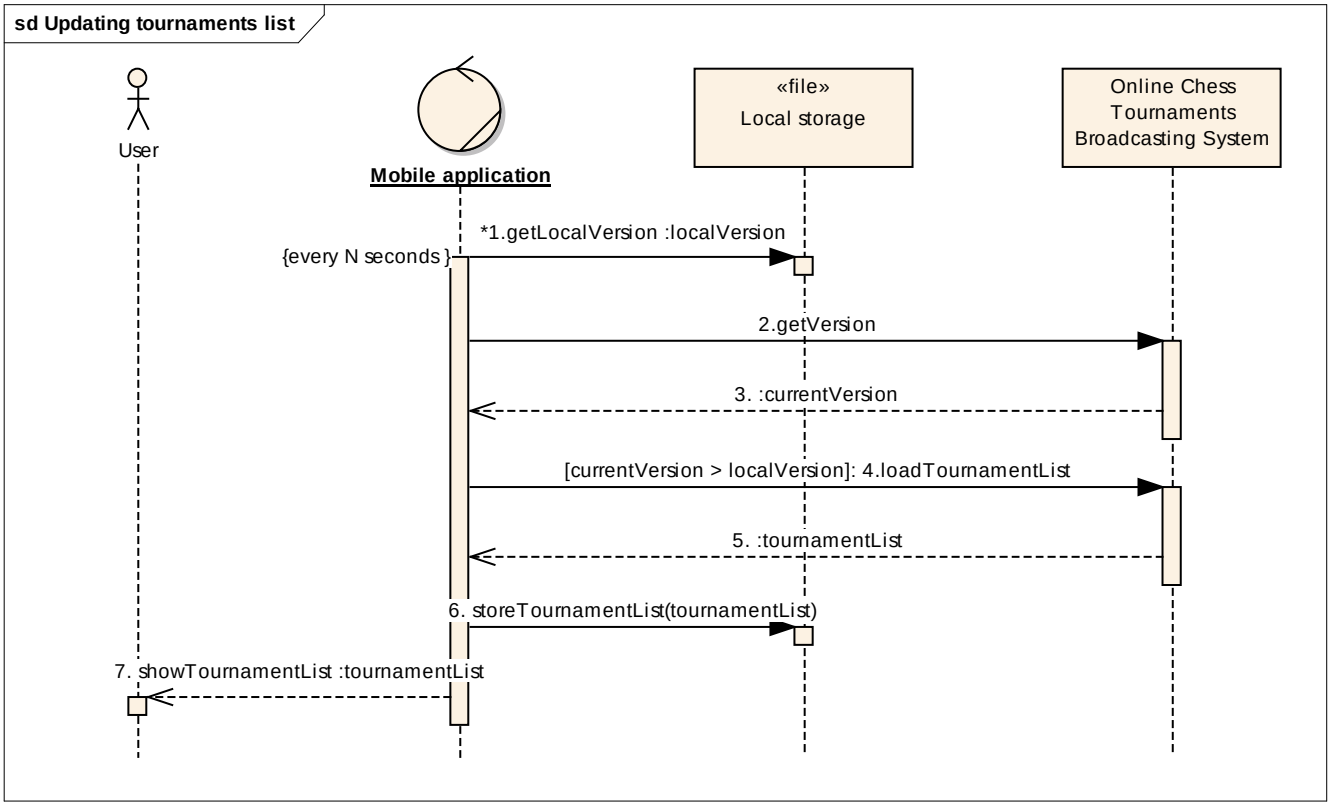


Figure 22 Updating a tournament list

1. Every N seconds a request to get a local version of the tournaments is sent to a local storage. The local storage returns the local version (1).
2. Request to get a current version of the tournaments is sent to the “Online chess tournaments broadcasting” system (2).
3. The “Online chess tournaments broadcasting” system returns the current version (3).
4. If the current version is higher than the local version, the request to load tournaments list is sent to the “Online chess tournaments broadcasting” system (4).
5. The “Online chess tournaments broadcasting” system returns the tournament list (5).
6. The tournament list is saved to the local storage (6).
7. The tournament list is shown to the user (7).

Updating a round list

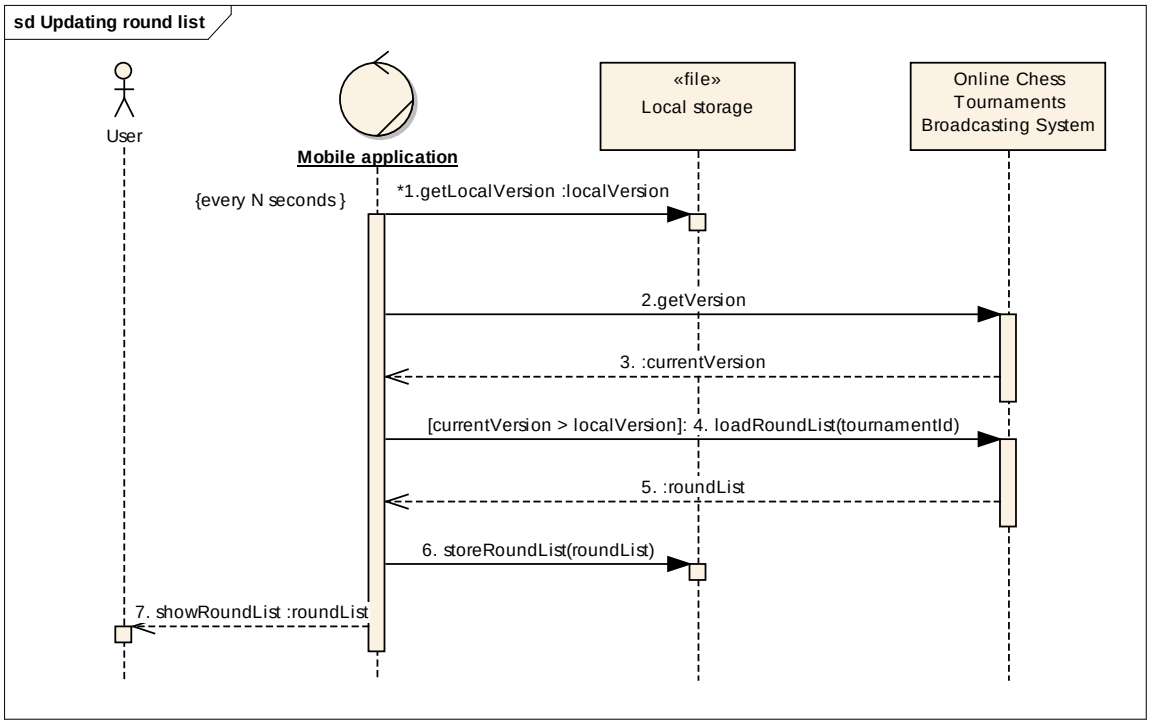


Figure 23 updating a round list

1. Every N seconds a request to get a local version of a round list is sent to a local storage. The local storage returns the local version (1).
2. Request to get a current version of the round list is sent to the “Online chess tournaments broadcasting” system (2).
3. The “Online chess tournaments broadcasting” system returns the current version (3).
4. If the current version is higher than the local version, the request to load the round list is sent to the “Online chess tournaments broadcasting” system (4).
5. The “Online chess tournaments broadcasting” system returns the round list (5).
6. The round list is saved to the local storage (6).
7. The round list is shown to the user (7).

Updating a game data

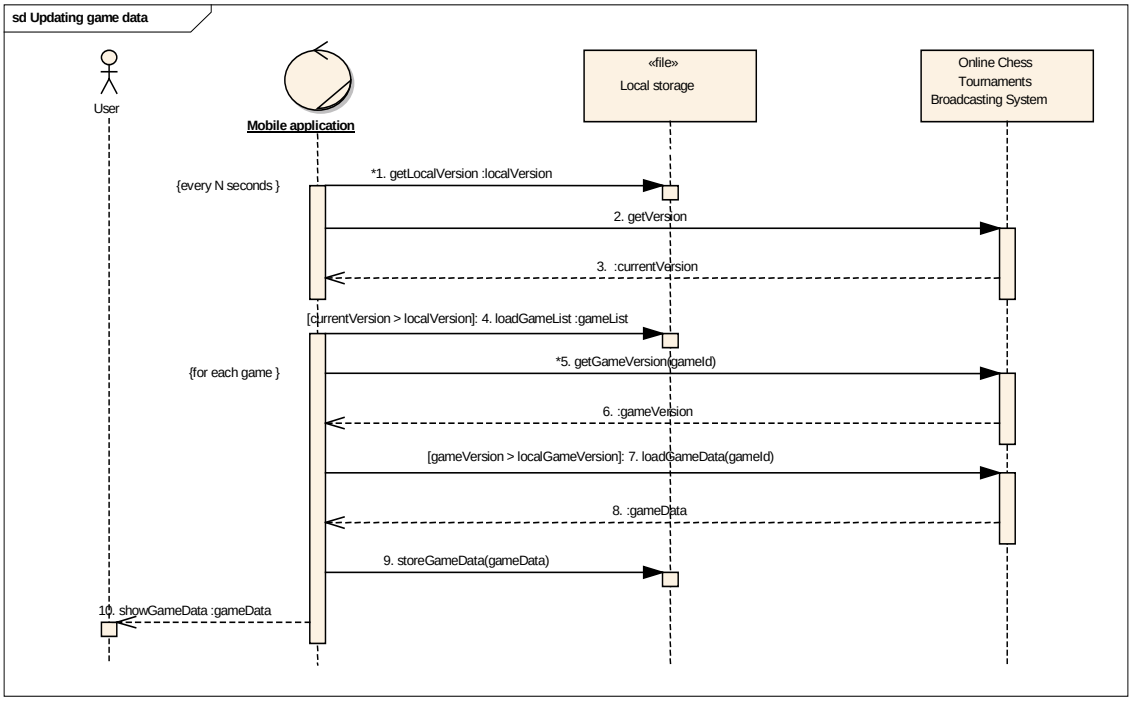


Figure 24 Updating a game data

1. Every N seconds a request to get a local version of a game list is sent to a local storage. The local storage returns a local version (1).

2. The request to get a current version of the game list is sent to the “Online chess tournaments broadcasting” system (2).

3. The “Online chess tournaments broadcasting” system returns the current version (3).

4. If the current version is higher than the local version, the request to load the game list is sent to the local storage (4).

5. For each game in the game list, the request to get a current game version is sent to the “Online chess tournaments broadcasting” system (5).

6. The “Online chess tournaments broadcasting” system returns the game version for each game (6).

7. If the current game version is higher than the local game version, the request to get a game data for each game in the list is sent to the “Online chess tournaments broadcasting” system (7).

8. The “Online chess tournaments broadcasting” system returns the game data for each game (8).
9. The game data is saved to the local storage (9).
10. The game data is shown to the user (10)

5.3.4 User Interface

The main user interface color of FIDE official web-site is blue. For this reason the “Online Chess Tournaments” application was designed in blue tones. Below are presented screenshots of the application. (Figure 25)

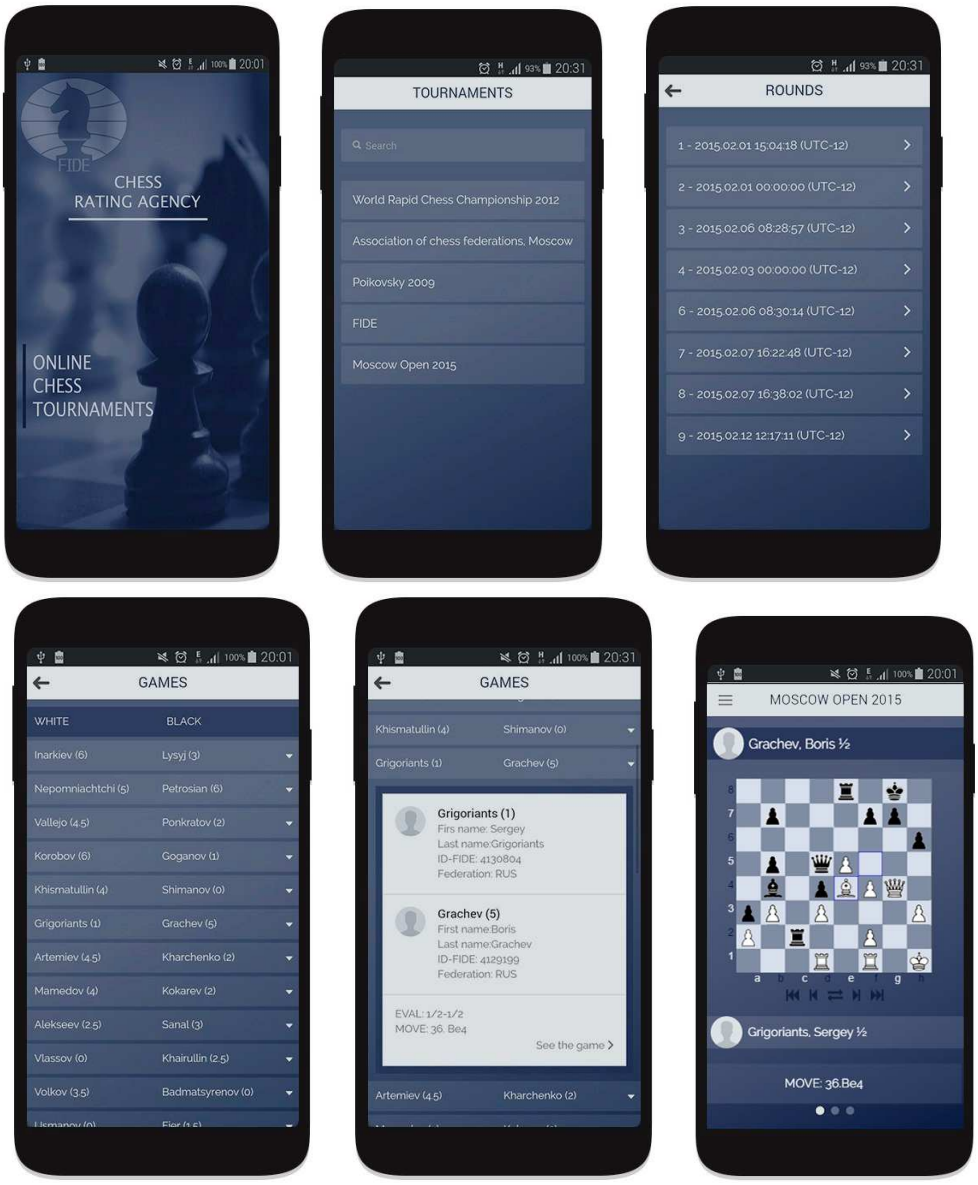


Figure 25 “Online Chess Tournaments” application's screenshots

5.4 REVIEW OF SIMILAR APPS

5.4.1 Lichess

“Lichess” mobile application is made for lichess.org web-site and available in 80 languages**.** The Lichess is an online environment for playing chess games with each other. Moreover, it is available to see games broadcasting move by move. The figure 26 illustrates three screens that are made from Lichess application on the mobile phone with android platform. The Lichess TV – opportunity to see games online move by move. The screen with toggle left menu provides information about all application’s opportunities divided into to modes: online and offline mode. In online mode user could create new game, play with friends and watch Litchess TV. In the offline mode user could play with computer or play for black and white side for himself. Finally, the third screen illustrates the game with AI (computer).

Figure 26 Lichess mobile application

The main advantage is that application is cross-platform. It is available on the App Store for iOS platform and on the Play Market for Android platform. Furthermore, it should be noticed that “Lichess” is open source. So, it is generally known what technical solutions were used for implementation the application. For implementation was used hybrid mobile model and was realisated such tools as Cordova platform, Tarifa toolchain and Mithril JS framework.

The shortcoming of the application is that the Internet connection is requeired for watching the games broadcasts.

5.4.2 Follow Chess

“Follow Chess” is mobile application where user could check out moves from multiple chess tournaments. So, the main application idea is identical with “Online Chess Tournaments” application. The user has opportunity to choose tournament from the list, choose the round, game and then see the game with move’s notation. Screenshots from application are presented bellow. (figure 27)

Figure 27 "Follow chess" mobile application

The system works in offline mode but not smoothly.
“Follow chess” has some disadvantages in comparison to “Online Chess Tournaments” application. This application is made only for Android platform. In Apple Store user could not find this application. It means that it has lost a big part of mobile phones users.

Also, it should be noticed that the board will not auto-refresh if new moves are made in the actual game. Furthemore, for analysis of the game, user should download the separate application: “Chess – Analyze This”. This application provides opportunity to analyze chess game with chess engines, simultaneously. Installing separate application for analyze chess game takes a long time. So it is not comfortable for users.

5.5 RESULTS AND ANALYSIS

As a result “Online Chess Tournaments” mobile application was successfully implemented with all required functionality.

It is achieving the goal to create a multi-platform mobile application. It means that one application is available for both Android and iOS users. For that reason was choosen hybrid mobile application model and Ionic framework for imlementation clear user interface.

Also, the distinctive feature is that the application works in both online and offline mode. The data is stored locally using local storage, which does not allow violating the application work without the Internet connection.

Also, with the Internet access, the application’s state is updated in the real time. To achieve this goal REST architecture style with traditional polling technique with 5 seconds interval was used or administrator could set updating interval by himself.

Furthermore, it has been made the overview of similar applications on the market. To summarize all information, it could be said that despite the fact that “Follow chess” presents the identical main idea, the “Online Chess Tournaments” system introduces a new, more convenient functionality such as opportunity to make evaluation of the game with chess engine, without downloading the separate application or availability to write comments with facebook account for each move.

Also, the “Litchess” presents a good implemented application, but it would not stay in the market on the same level as “Online Chess Tournaments” because it does not translate international tournaments.

To summarize, two main applications features could be distinguished, which set apart from the rest:

- Application is for multi-platforms mobile devises that have inscreased usage two times. According to the latest statistics by Net Market Share (2014), Android is leading the market with 45.86% and iOS is the second popular OS worldwide with 43.15% of share. (Figure 28)
- Application is available in offline mode using localStorage technique.

- Additional features like evaluation of the game with chess engine and the availability to write comments for each move with facebook account.

In conclusion, it should be said that “Online Chess Tournaments” mobile application is a completely unique product on the mobile applications market.

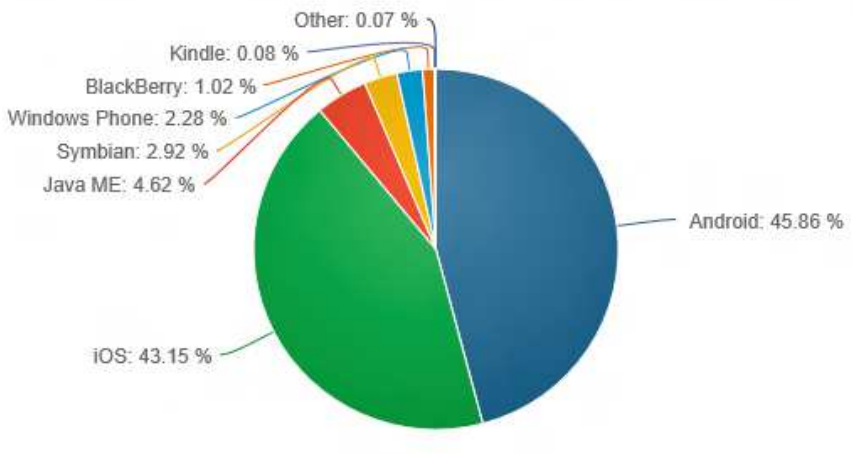


Figure 28 Worldwide mobile market share chart [44]

6. CONCLUSION

In every software development process it is very important to make correct architectural decisions to implement the safe, stable and user – friendly system.

In this thesis the big attention was focused on making grounded architectural decisions for the development of multi-platform mobile application "Online chess tournaments".

Based on the result of the researches the architectural decisions for the implementation “Online Chess Tournaments” application were made. In the course of this work a hybrid mobile application model was chosen. As an UI framework the decision was made in favor of Ionic framework. As an architectural style the REST – the Representational State Transfer was chosen and traditional polling method was used.

All these approaches have been realized in the case study of “Online Chess Tournaments” application. As a consequence, the application works in two major platforms such as: Android and iOS. Application broadcasts the international tournaments in real time. Also, the goal, to ensure the application work in offline mode without disruption, has been achieved successfully.

As future development of the application it could be implemented such things like maintaining the different languages, display information about starting lists of players and information about the course of the tournament pairs paring.

Application passed all customer’s accepting tests and now is available for download on the Google Play Market and in future it will be also available on the Apple Store.

KOKKUVÕTTE

Tarkvaraarenduse protsessi üks oluliseim osa on õigete arhitektuurivalikute tegemine, mille järgi on võimalik usaldusväärse, stabiilse ja kasutajasõbraliku süsteemi realiseerimine.

Lõputöös tehti ja põhjendati arhitektuurivalikud mitmel platvormil töötava “Online Chess Tournaments” mobiilirakenduse arendamiseks.

Tehtud analüüsi alusel põhjendatakse käesolevas töös mitmel platvormil töötava mobiilirakenduse arendamiseks konkreetseid arhitektuurivalikuid. Mitmel platvormil töötava mobiilirakenduse arendamiseks on valitud hübriidne mobiilirakenduse mudel Ionic raamistikku kasutades. Arhitektuuristiilina on valitud REST – the Representational State Transfer. Päringumeetodite variantidest on valitud traditsiooniline päringute meetod.

Kõik sellised arhitektuurilahendused on realiseeritud “Online chess tournament” mobiilirakenduses. Tulemusena töötab rakendus nii Androidi kui iOS platformil. Süsteem edastab mängude seisu rahvusvahelistel turniiridel reaalajas. Samuti saavutati eesmärk, et tagada rakenduse töö võrguta režiimis.

Mobiilrakenduse edaspidine arendamine on võimalik järgmistes suunades: erinevate keelde toetamine, turniiride mängijate nimekirjade lisamine ja info mängijate paaride loosimise kohta.

Tellija võttis rakenduse vastu ja praegu on seda võimalik laadida Google Play Market-ist. Plaanil on mobiili rakenduse kättesaadavus Apple Store-ist.

REFERENCES

Mobile application models

1. Dragan Gaić (2014). Hybrid vs Native mobile apps.
Retrieved from: http://www.gajotres.net/hybrid-vs-native-apps/

2. John Bristowe (2014). What is a Hybrid Mobile App?
Retrieved from: http://developer.telerik.com/featured/what-is-a-hybrid-mobile-app/

3. Mitch Pronschinske (2014). The State of Native vs. Web vs. Hybrid.
Retrieved from: http://java.dzone.com/articles/state-native-vs-web-vs-hybrid

4. Joe Stangarone (2012) . Hybrid vs. Native vs. Mobile web comparison chart.
Retrieved from: http://www.mrc-productivity.com/blog/2012/03/hybrid-vs-native-vs-mobile-web-comparison-chart/

5. Pietro Saccomani (2012) . Native, Web or Hybrid Apps? What's The Difference?
Retrieved from: http://www.mobiloud.com/blog/2012/06/native-web-or-hybrid-apps/

6. Jake Hird (2011). The fight gets technical: mobile apps vs. mobile sites.
Retrieved from: https://econsultancy.com/blog/7832-the-fight-gets-technical-mobile-apps-vs-mobile-sites/

7. Ted Wallin. WEB, HYBRID OR NATIVE?
Retrieved from: http://www.gambitgroup.fi/our-work/web-hybrid-or-native/

UI frameworks for hybrid mobile applications

8. Kevin Meurer(2014). Impressions of the Ionic Framework: Not Quite Native.
Retrieved from: http://kevinmeurer.com/impressions-of-the-ionic-framework-not-quite-native/

9. Katie (2014). Platform Continuity.
Retrieved from: http://blog.ionic.io/platform-continuity/

10. Ben Ripkens (2014). Ionic: An AngularJS based framework on the rise.
Retrieved from: https://blog.codecentric.de/en/2014/11/ionic-angularjs-framework-on-the-rise/

11. Dragan Gaić(2013). Ionic vs OnsenUI
Retrieved from: http://www.gajotres.net/ionic-vs-onsenui/

12. Ionic framework official web-site.
Retrieved from: http://ionicframework.com/

13. Telerik. Kendo UI.
Retrieved from: http://www.telerik.com/kendo-ui

14. Famo.us framework official web-site.
Retrieved from: http://famo.us/

15. Choosing the right HTML5 Framework.
Retrieved from: http://rapidvaluesolutions.com/whitepapers/HTML5-Framework.html

16. Tal Gleichger. Hybrid UI framework shootout: Ionic vs. Famo.us vs. F7 vs. OnsenUI.
Retrieved from: https://www.airpair.com/ionic-framework/posts/hybrid-apps-ionic-famous-f7-onsen

REST / SOAP

17. James Snell,Doug Tidwell,Pavel Kulchenko (2002). Published by O'Reilly & Associates. Programming Web Services with SOAP.

18. M. Papazoglou .(2008) Web Services: Principles and Technology .

19. Mark Masse. (2012) Published by O'Reilly . REST API Design Rulebook .

20. Erik Wilde,Cesare Pautasso. Published by Springer . REST: From Research to Practice.

21. Learn REST: A Tutorial.
Retrieved from: http://rest.elkstein.org/2008/02/what-is-rest.html

22. Understanding REST.
Retrieved from: https://spring.io/understanding/REST

23. Margaret Rouse.(2014). REST (representational state transfer).
Retrieved from : http://searchsoa.techtarget.com/definition/REST

24. Ludovico Fischer(2013). A Beginner's Guide to HTTP and REST
Retrieved from: http://code.tutsplus.com/tutorials/a-beginners-guide-to-http-and-rest--net-16340

25. Manisha Patil. REST Architectural Elements and Constraints.
Retrieved from: http://mrbool.com/rest-architectural-elements-and-constraints/29339

26. Ricardo Plansky (2015). REST Architecture Model: Definition, Constraints and Benefits
Retrieved from: http://imasters.expert/rest-architecture-model-definition-constraints-benefits/

27. Aaron Skonnard(2003). Understanding SOAP
Retrieved from : https://msdn.microsoft.com/en-us/library/ms995800.aspx

28. REST vs. SOAP: Selecting a Web Service (2015)
Retrieved from :http://www.optimusinfo.com/rest-vs-soap-selecting-a-web-service/

29. Jack Cox. SOAP vs. REST For Mobile Services
Retrieved from : https://www.captechconsulting.com/blogs/soap-vs-rest-for-mobile-services

30. Arvind Rai.(2014). When to Use SOAP and REST Web Services
Retrieved from : http://www.concretepage.com/webservices/when-to-use-soap-and-rest-web-services

31. REST Vs SOAP Web Services : Which one to use ?
Retrieved from : http://www.tutorialsdesk.com/2014/10/rest-vs-soap-web-services-which-one-to.html

POLLING TECNIQUES.

32. Joe Hanson (2014) What is HTTP Long Polling?
Retrieved from : http://www.pubnub.com/blog/http-long-polling/

33. Mathieu Carbou (2011) Reverse Ajax, Part 1: Introduction to Comet.
Retrieved from: http://www.ibm.com/developerworks/library/wa-reverseajax1/

34. The HTTP Request/Response Model.
Retrieved from: https://www.safaribooksonline.com/library/view/javaserver-pages-3rd/0596005636/ch02s01.html

35. Mohsen Elgendy. Understanding Ajax Long-Polling Requests
Retrieved from: http://webcooker.net/ajax-polling-requests-php-jquery/

36. Seventh Octave. Simple Long Polling Example with JavaScript and jQuery
Retrieved from: http://techoctave.com/c7/posts/60-simple-long-polling-example-with-javascript-and-jquery

37. Михаил Русаков (2014) Что такое Long-Polling, WebSockets, SSE и Comet.
*Retrieved from:*http://myrusakov.ru/long-polling-websockets-sse-and-comet.html

38. COMET с XMLHttpRequest: длинные опросы.
Retrieved from: https://learn.javascript.ru/xhr-longpoll

ANALOGOUS APPS

39. Asim Pereira.(2014) Follow Chess 1.0 released!! Now follow moves from multiple Chess tournaments.
Retrieved from: http://www.mychessapps.com/2014/04/follow-chess-10-released-now-analyze.html

40. “Follow Chess” on Play Market :
Retrieved from: https://play.google.com/store/apps/details?id=com.pereira.live

41. Lichess official web-site:
Retrieved from: http://lichess.org/

42. Lichess on Play Market:
*Retrieved from:*https://play.google.com/store/apps/details?id=org.lichess.mobileapp

43. Lichess on Apple Store:
Retrieved from: https://itunes.apple.com/us/app/lichess-free-online-chess/id968371784

OTHERS

44. Ahmad Wahid (2015). Windows Phone gains 0.15% of market share in December 2014 with 2.28% worldwide.
Retrieved from: http://www.windows8core.com/windows-phone-gains-0-15-market-share-december-2014-2-28-worldwide/

45. Statista 2015. Number of mobile app downloads worldwide from 2009 to 2017 (in millions)
Retrieved from: http://www.statista.com/statistics/266488/forecast-of-mobile-app-downloads/

46. Benedict Evans (2013). Mobile is eating the world
Retrieved from: http://ben-evans.com/benedictevans/2013/11/5/mobile-is-eating-the-world-autumn-2013-edition

47. Writing your Dissertation: Methodology. From Dissertation Writing section.
Retrieved from: http://www.skillsyouneed.com/learn/dissertation-methodology.html

48. The Case Study as a Research Method.
Retrieved from : https://www.ischool.utexas.edu/~ssoy/usesusers/l391d1b.htm