

MASARYK UNIVERSITY
FACULTY OF INFORMATICS



Cross-platform System for Time Scheduling

BACHELOR'S THESIS

Jan Tomášek

Brno, Spring 2016

MASARYK UNIVERSITY
FACULTY OF INFORMATICS



Cross-platform System for Time Scheduling

BACHELOR'S THESIS

Jan Tomášek

Brno, Spring 2016

Declaration

Hereby I declare that this paper is my original authorial work, which I have worked out on my own. All sources, references, and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source.

Jan Tomášek

Advisor: doc. RNDr. Tomáš Pitner Ph.D.

Acknowledgement

I would like to express my gratitude to doc. RNDr. Tomáš Pitner, Ph.D. for valuable advice concerning formal aspects of my thesis, to Ing. Libor Kubečka and Software Development Europe company for technical support, to all people involved in beta testing, and to my closest family for their support.

Abstract

This thesis deals with a development of cross-platform application for time management.

Author reviewed common approaches for cross-platform development and implemented hybrid mobile application in Ionic Framework for Android and iOS devices, together with web application for desktops. Users' data is stored on a remote server and accessed via Web API.

Mobile applications for different platforms are built from one source code and shares the majority of that code with the web application. That makes the development and maintenance easier, which is the key reason for cross-platform development.

Keywords

cross-platform, mobile application, hybrid, ionic, iOS, android, web application, time management

Contents

1	Introduction	1
2	Time management	3
2.1	<i>Three generations of time management</i>	3
2.2	<i>The fourth generation</i>	3
2.2.1	P/PC Balance	3
2.2.2	Four quadrants of importance and urgency	4
2.2.3	Weekly planning	5
2.3	<i>Current applications</i>	6
3	Initial software analysis	9
3.1	<i>Requirements</i>	9
3.1.1	Non-functional requirements	9
3.1.2	Functional requirements	9
3.2	<i>Use-case diagram</i>	10
3.3	<i>Application structure</i>	10
4	Cross-platform frontend development approaches	11
4.1	<i>Web applications</i>	11
4.1.1	Responsive website	12
4.1.2	Mobile website	12
4.2	<i>Native mobile applications</i>	13
4.3	<i>Cross-platform mobile frameworks</i>	13
4.3.1	Hybrid approach	14
4.3.2	Interpreted approach	15
4.3.3	Cross compiled approach	16
4.4	<i>Conclusion</i>	17
4.4.1	Pros and cons	18
4.4.2	Decision matrix	19
5	Backend development	21
5.1	<i>Database</i>	22
5.1.1	Entity Relationship Diagram	22
5.2	<i>Application server</i>	23
5.2.1	Wildfly	23
5.2.2	Configuration	23
5.3	<i>Object relational mapping</i>	24
5.3.1	Hibernate	24
5.3.2	Mapping and relations	24

5.4	<i>Web API</i>	25
5.4.1	REST architecture	25
5.4.2	Web API specification and implementation . . .	26
5.5	<i>Authentication</i>	28
5.5.1	Sessions	29
5.5.2	Signed tokens	29
5.5.3	Password reset strategy	30
6	Frontend development	31
6.1	<i>Ionic Framework</i>	31
6.1.1	Installation	31
6.1.2	AngularJS components	31
6.1.3	Accessing native API	33
6.1.4	Debugging	33
6.1.5	Ionic evolution	34
6.2	<i>Web application</i>	35
6.3	<i>Synchronization</i>	36
6.4	<i>Application features</i>	37
7	Testing and publishing	39
7.1	<i>Integration testing</i>	39
7.2	<i>Beta testing</i>	40
7.3	<i>Publishing</i>	41
8	Conclusion	43
A	Setup of the backend server	45
B	Web API	47
C	Screenshots from the application	51
D	List of electronic attachments	53

1 Introduction

Time management helps people to be more productive and effective. Calendars, diaries, lists with notes etc, all of these are well-known organizers for time management. Less popular is the organizer for the fourth generation of time management, described by Stephen R. Covey¹, which differs from the others in its philosophy and construction.

Due to the evolution of computing technologies, more and more people use their devices to manage time electronically. Mobile phones are ideal devices for this purpose, because in time management, it is important to have an opportunity to modify a time plan whenever it is needed. However, the control of mobile phones is slower and the display size is smaller, compared to devices like PCs. Therefore, popular time management applications are cross-platform. There are many cross-platform applications for common organizers, but only a few popular applications for Covey's organizer.

This thesis deals with the whole process of development of a cross-platform application including the implementation of cross-platform application, tailored especially for the Covey's organizer. The first part includes introduction to the time management of the fourth generation, the summary of popular time management applications, the software analysis and the analysis of current possibilities of cross-platform development. The second part includes the system design, the description of the implementation, testing and publishing.

1. Well-known leadership instructor and author of bestseller *The 7 habits of highly effective people*.

2 Time management

Stephen Covey divided time management evolution into four generations [1]. Each generation improves the previous one with additional points of interest.

2.1 Three generations of time management

The first generation used a simple list of tasks and notes which helped people to remember what to focus on and also gave them feeling of some kind of order which they should follow.

The second generation started to plan the future. People started to link their tasks with dates and record them into calendars and diaries. This was important because, apart of that, people were able to use time management for reaching long-term goals.

The third generation of planning is about priorities. People set their long-term or short-term goals and try to move towards them. In their daily planning they give priority to their tasks, which should lead them to their goals, sort them according to priority and presumed time spent, and then try to accomplish them in most efficient way. This is the most known time management approach nowadays.

2.2 The fourth generation

2.2.1 P/PC Balance

Stephen Covey pointed out that to be effective and successful in the long-term, it is necessary to maintain the principle of P/PC balance where P stands for production and PC stands for production capability [1]. The idea behind this principle is that when people work all week long, omitting rest and their personal and social needs, they can be successful in short-term, but in the long-term it will result in an unbalanced life, physical and psychical exhaustion, which will take its toll on the production itself.

To maintain the P/PC balance, it is important for people to remember which roles do they play in their lives. Each person can have many of these roles, e.g. parent, partner, manager, lecturer, sportsman and

2. TIME MANAGEMENT

more. In previous generations of time management people could have these roles in mind, but with this generation it requires the need to write them down. After that, people should write one or two long-term goals which they would like to achieve in each role. People should also write down their personal mission, which is typically influenced by their roles. Short-term goals (tasks) are also linked with a specific role and ideally with long-term goals. Having the list of roles written down helps people to remember their work needs, but also personal and social needs, which is necessary to maintain the P/PC balance.

2.2.2 Four quadrants of importance and urgency

Another key thing for effectiveness is to recognize tasks on which a person should focus on. Covey divided tasks into four quadrants according to the importance and urgency [1].

	URGENT	NOT URGENT
IMPORTANT	I Crises Pressing problems Deadline-driven projects	II Prevention, PC activities Relationship building Recognizing new opportunities Planning, recreation
NOT IMPORTANT	III Interruptions, some calls Some mail, some reports Some meetings Proximate, pressing matters Popular activities	IV Trivia, busy work Some mail Some phone calls Time wasters Pleasant activities

Figure 2.1: Four quadrants of importance and urgency [1]

1. Important and urgent

The higher the number of such tasks, the worse the situation will get. Important tasks need to be done and when it is urgent as well, this lack of time causes stress. Some of these tasks do cause unpredictable

bumps in the schedule, but most of these could have been solved earlier and importantly before they became urgent and added additional time pressure.

2. Important but not urgent

Tasks in the second quadrant are tasks on which a person should focus on. These tasks are not yet urgent, so there is some flexibility, but because they are important they still have to be written down.

3. Not important but urgent

These tasks may seem important at first glance, but at the end of the week a person can find out that they were not. E.g. attendance at a public event that a person has already committed to, but that has no actual benefit for them. It is important to establish priorities to distinguish what is important and what is not.

4. Not important nor urgent

Things that people usually do when they feel tired. However, hours spent watching TV is not the right way to renew power. To re-gain their energy, people should follow the seventh habit called Sharpen the Saw, which is a guidance for renewal of four dimensions – physical, social, mental and spiritual.

2.2.3 Weekly planning

The vast majority of systems of the third generation are focused on daily planning of tasks from the first quadrant. People use these systems to write down tasks which must be done in the next day. However, this system will crash when some big rock unexpectedly bumps to the schedule.

Planning in the system of the fourth generation is based on weekly cycles and focus on tasks of the second quadrant. With this approach the time space for tasks is greater and the whole system becomes flexible. Because tasks from the second quadrant are not urgent, they can be moved to another day if some big rock unexpectedly bumps to the schedule.

Weekly planning also supports the concept of roles because the greater time space allows people to find time for all their roles.

2.3 Current applications

The fourth generation time management is not as popular as the previous ones. Some popular applications like Any.do support grouping tasks into lists, which is actually a more general concept of roles. However, there are only few applications which include the Covey's design for weekly planning and share his ideology. Two of such popular applications are My Effectiveness and ZZTasks.

My Effectiveness

My Effectiveness is a good-looking, complex application for Android devices, which strictly follows the Covey's ideology. The application is available on the Google Play store since October 2011, and falls into the category of 100,000–500,000 downloads, which makes it the most famous application for the 4th generation time management for Android devices. Information about development and guides how to use the applications are available at the developers blog¹.

The application enables users to write down their personal mission, roles and their goals, actions and notes. Role goals stands for long-term goals whereas actions are, like tasks, short-term goals. Actions can be created either in the "Weekly Plan" section or in the "First Things First" section, which includes visualization of four quadrants and actions created inside them.

The specific and not very intuitive feature is that an action has assigned a day name instead of a date. Therefore all actions are visible in the "Weekly Plan", independently on the actual date, until they are deleted. Another problem with actions in this application is that it is not possible to simply assign an action to a role. The action must be tight up to a role goal, which is unnecessarily restrictive. The main drawback of the application is that it is very complex but not intuitive, so it takes some time to learn how to use it.

1. <http://andtek.blogspot.cz/>

A big weakness of this application is that it is only available for Android devices, although people have been asking for a web application on the developer's blog since 2012 [2]. According to the Google search statistics [3], the amount of people looking for the application on the internet is now about 90 per month.

ZZ Tasks

ZZ Tasks is a cross-platform application for the fourth generation time management. It can be installed on devices with iOS or Android and it is also accessible at: <https://zztasks.com>. The application, originally named FranklinCovey Tasks, has changed its name and policy in January 2016. Together with name of the application, names of features were renamed also: "personal mission" to "bucket list" and "roles" to "buckets". However the functionality stayed the same.

It supports the necessary parts for fourth generation time management – concept of roles linked with tasks, priority ordering, personal mission etc. An attractive feature is the opportunity to synchronize tasks with Google apps or Microsoft Exchange accounts, although this was not working when tested. The creation of tasks is fast and controls are simple and intuitive, so the application does not look as confusing as the previous one.

The application offers 30 day trial after which the support of history, synchronization and even access to the web application are charged. Another drawback is that the mobile application doesn't use any specific mobile controls like sliding or drag and drop, which are appreciated by users. Everything is handled by buttons, forms and dialog boxes.

3 Initial software analysis

3.1 Requirements

3.1.1 Non-functional requirements

- Platform : Cross-platform application
 - Available as a web application for desktop browsers.
 - Available at least for devices with Android and iOS OS's with use of any technology.
- Server's operating system : Linux

3.1.2 Functional requirements

Functional requirements are based on the Covey's description of a planning system of the fourth generation in the book The 7 Habits of Highly Effective People, habit 3.

- Application shall enable users to write down their personal missions, roles, long-term and short-term goals.
- Application shall display graphical feedback on planning efficiency to motivate users and remind them their roles.
- Application shall support weekly planning system.
- Application shall enable users to update their plans and let them move their tasks to the backlog.
- Application shall not be restrictive and shall let users to use it as they want.
- Application shall enable users to create and manage their tasks in a simply and fast way.



The purpose of the backend part is to store users' data at one, remote, centralized place and communicate with the frontend part via internet. With this approach, once users create their account, they can access their data from whatever device they want. Backend development process is described in Chapter 5.

10

4 Cross-platform frontend development approaches

Cross-platform application should be independent of operating system (Windows, Linux, Android, iOS), type of device (smartphone, tablet, PC) and screen size. It is common that the frontend part of the application is specific for different platforms. It is due to the fact that differences between device types and screen sizes are perceptible.

For desktops and notebooks, there is usually a web application available, because major web browsers are available for major OS's (Linux, Windows, Mac OS) and renders the web content in similar way. However, there are different approaches of how to develop cross-platform frontend for mobile phones and tablets. Technologies which are widely used to achieve the goal are described below.

4.1 Web applications

As mentioned, web application is developed for desktops and notebooks. However, the web application could be also used for mobile phones and tablets.

Technologies like HTML (HyperText Markup Language), CSS (Cascading Style Sheets) and JavaScript, which are used for development of the web application are widely known and easy to learn. Another advantage of web applications is the ability to run on multiple platforms. Applications run on remote servers and in web browsers, which are contained in every OS, so every device with OS (Operating System) can use web application. As they run on remote server, they are portable just by the share of a link and there is no need for installation. Developers can update web applications instantly straight on servers, which is much more comfortable than maintaining updates to applications installed on clients' devices.

On the other hand, all types of web applications have drawbacks of performance compared to other technologies for cross-platform development. Also they are not able to utilize native API (Application Programming Interface) of the platform and its native GUI (Graphical User Interface) components. Another big disadvantage is that the internet connection is needed to communicate with remote server.

A web application which is developed for desktops and notebooks has to be either responsive, or must be modified, when it has to be used by mobiles, because mobiles have much smaller screen size than desktops.

4.1.1 Responsive website

Responsive website adapts its layout according to the platform used by visitor. Screen size is the key attribute in the responsive website design. The layout viewed is wide and full of information for larger screens, while for smaller screens, more information is hidden under website navigation.

The power of responsive websites is the "one size fits all" principle, which means that there is only one version of web application, with one source code, which works well on all platforms. Each device with internet browser and internet connection can use a responsive website as an application. As the user connects, the layout and controls are set according to platform used. This approach is mostly used for informational websites to enable user to find information quickly from any platform. For more complex applications, full of controls and interactivity, this is not the best option [4], since it can't use the full potential of the particular platform. It takes more time to create a good working responsive websites than just common websites. However, the development could be done by using a responsive web framework, e.g. Bootstrap, which makes it easier.

4.1.2 Mobile website

Another approach of creating cross-platform web applications is to have two separate websites – one optimized for larger screens and one for mobile phones. Besides responsive website, with this approach, it is easier to develop complex, component-rich website [4]. Mobile website can be tailored exactly for its purpose. However, mobile website needs its own domain and also code, so the whole maintenance is not so comfortable.

4.2 Native mobile applications

Native mobile applications are developed in specific programming languages according to the specific OS of the device. For example native applications for Android devices are developed in Java while iOS development is done in Objective C.

This approach provides the fastest performance of applications and allows the developer to utilize native API (application programming interface) of the OS for the use of camera, push notifications¹ etc. Native mobile applications can run on a client's side only, which means they can be fully used without the internet connection. Moreover, they can be published at mobile e-shop where most of mobile phone users look for specific applications.

A disadvantage is that to accomplish a cross-platform application, for each different OS (Android, iOS, Windows Phone, Blackberry etc.) a different application must be made. Various OS's use different programming languages and all of them require more skills and are harder to learn than HTML and CSS used for web applications. Therefore, the development is challenging and expensive.

4.3 Cross-platform mobile frameworks

Applications developed using cross-platform frameworks take advantages of both web applications and native applications. They are mostly written in one programming language so they are portable and the development process takes less time than in native mobile applications case. But, unlike web applications, they can use native API. Instead of publishing on the internet, they are installed locally on a device and they can be published on mobile e-shops.

Cross-platform frameworks are generally divided into three groups according to approaches used to attain their purpose [5].

1. Notifications which are displayed to users even if they don't use the application at the moment.

4.3.1 Hybrid approach

Hybrid mobile applications are written, as web applications, in HTML, CSS and JavaScript. However, they are not available on the internet, but installed locally on the device. To achieve that, hybrid frameworks utilizes another frameworks, usually Apache Cordova or PhoneGap, which build application packages from HTML, CSS and JavaScript source codes. Applications then run in the native web browser of the device where only the content of the application is viewed; the panel with URI (Unique Resource Identifier) is hidden. The native browser, can communicate with native API of the device using Cordova plugins. This allows the hybrid mobile application to use functionality like push notifications, camera and others, so the functionality of the application can be complex.

The development of a hybrid mobile application is very similar to the development of a web application. Simply said, hybrid applications are web applications, which uses plugins to access native API and are packaged by Cordova/PhoneGap into native application. The application logic (JavaScript) is shared between all platforms and because the GUI is written in HTML and CSS, it can be also shared between all platforms², including the web platform. However, the layout of the GUI has to be either responsive or optimized for mobiles/desktops as pointed out in Section 4.1.

Because the GUI is written in HTML and CSS and rendered by a web browser, the application does not use native GUI components and animations. This can have negative impact on user-friendliness. To alleviate this lack, some hybrid frameworks provides sets of components, like animations, CSS styles and others, which imitate native look and feel. Another disadvantage of a hybrid application could be weak performance. Because the application depends on a native browser, it can be slower than native mobile application.

Hybrid frameworks

There are plenty of various frameworks for hybrid mobile applications development. These frameworks differs from each other mainly

2. Detailed discussion of how much of the code can be shared can be seen in Section 6.2

in components which they provides for mobile development (CSS styles, JavaScript components) and also in JavaScript libraries/frameworks which are used for development. Popular open source hybrid frameworks are mentioned below.

One of the most popular hybrid framework is Ionic Framework. Ionic is built on the top of AngularJS, which is MVC (Model-View-Controller) JavaScript framework. To develop complex application with Ionic, it is recommended to use AngularJS [6]. With its CSS styles, JavaScript components and native icons, Ionic is trying to imitate the native look and feel of iOS, Android and recently also Windows Phone 10. Great advantage of Ionic is its huge community and complete documentation.

Onsen UI is a less popular Ionic alternative. It is also built on the top of AngularJS but unlike Ionic, Onsen UI is less AngularJS dependent and does not forces developer to use AngularJS for development. It offers support for AngularJS but also for React and jQuery libraries. Onsen UI also tries to imitate native look and feel and supports Android, iOS and Windows Phone 8.1.

Besides others frameworks imitating native look and feel, there are also frameworks which are more likely frameworks for mobile-first responsive website development. However, because Cordova can package every website which is written in HTML, CSS and JavaScript into application package, they can be considered as hybrid frameworks. Most popular framework of this kind is probably jQuery mobile, however there are also others, newer frameworks like Mobile Angular UI. The advantage of such frameworks is that because Cordova supports wide scale of mobile OS's (Blackberry, Firefox OS, Ubuntu) and components are not native-like, the application can be packaged also for these platforms whereas frameworks which provides native-like components usually support only major mobile OS's (Android, iOS, Windows Phone).

4.3.2 Interpreted approach

In interpreted approach case, the application does not run in the native web browser. Instead, the application runs as a native application. The application source code (usually written in JavaScript) is interpreted

at runtime by an interpreter, which accesses native features of the platform through the abstraction layer [5].

The abstraction layer is usually some API, provided by the specific framework. Developers utilize this API in their platform independent source code during development. At runtime, the platform-specific implementation of the API is executed.

To develop the interpreted application, it is necessary to learn the specific API of the framework, which is not needed in case of hybrid approach, where the development is the same as for web applications. On the other hand, interpreted applications are faster, because they don't run in a web browser. They also provides native look and feel, which is appreciated by users. However, to achieve that, it is sometimes necessary to write platform-specific code.

Appcelerator

In the case of Appcelerator, which is the most popular platform for interpreted mobile applications, there is the Titanium API, which maps JavaScript objects and methods to native code. JavaScript objects are paired with native objects and shares properties and methods. With using Titanium API, methods called on JavaScript objects invokes native methods [7].

The GUI of the Appcelerator application is programmed with the Titanium API, using JavaScript objects. Appcelerator also provides Alloy XML (Extensible Markup Language) markup as an abstraction from Titanium API, which makes it easier to create GUI [8], but it still requires additional knowledge. There are also some WYSIWYG (What you see is what you get) tools³ for GUI creation, however, the original one is available only under payed account.

Appcelerator currently supports Android, iOS, Windows Phone and Blackberry platforms, between which 60–90% code can be reused [9].

4.3.3 Cross compiled approach

Cross compiled approach is similar to the interpreted approach, since the result is also native application. However, in cross-compiled ap-

3. Example of Titanium WYSIWYG: <http://www.titaniumui.com/>

proach, the code is not interpreted by an interpreter at runtime but compiled into native binaries by a cross-compiler [5]. Since the application is compiled and not interpreted, the performance can be even better. Drawbacks of this approach are the same as in the interpreted approach case; it is necessary to learn API and to achieve the native look and feel, it is necessary to write some platform-specific code.

Xamarin Framework

Well-known framework for cross compiled application development is Xamarin. In the case of a Xamarin framework, the cross-platform application is written in C#.

For years, Xamarin was a commercial product, which provided very restricted functionality for free. Developers with free accounts were unable to use Xamarin Forms, which is XML markup for cross-platform GUI creation. Therefore they had to create specific GUI for each platform. Moreover, there was a limitation of the application size [10].

In March 2016, Microsoft bought Xamarin which resulted in changes in distribution and pricing. Xamarin is now available as a tool in Visual Studio Community Edition, for free, without previously mentioned restrictions and limits [11].

Xamarin currently supports Android, iOS and Windows Phone and according to Microsoft Visual Studio website, averagely 75% of source code can be reused between these platforms [12]. However, on Xamarin website introducing prebuilt applications there are example applications with code reuse of 93–100% [13].

4.4 Conclusion

It is common to create one web application for desktops and notebooks. Mobile phones have different screen sizes and therefore typically can not use desktop web application if it is not at least optimized. Based on research of cross-platform frontend development approaches, author created lists of pros and cons and decision matrix which summarizes which approach is appropriate to use according to requirements on the application functionality and development.

4.4.1 Pros and cons

Web applications

- + Source code is reused between all mobile platforms
- + Application logic can be shared with desktops web application (GUI partially)
- + Ease of development with HTML, CSS and JavaScript
- + No installation, Instant updates
- Internet connection needed
- Weak performance
- Unable to access native API
- No native look and feel

Native applications

- + Access to native API
- + The best performance
- + Native look and feel
- No code reuse between platforms
- Expensive and challenging development

Hybrid applications

- + Source code is reused between all mobile platforms
- + Application logic can be shared with desktops web application (GUI partially)
- + Ease of development with HTML, CSS and JavaScript
- + Access to native API
 - Partially native look and feel (depends on framework)
- Performance

Interpreted and Cross compiled applications

- + Performance
- + Access to native API
- + Application logic can be shared with desktops web application⁴
 - About 75% of source code reused between mobile platforms
- It is necessary to learn API of the framework

4.4.2 Decision matrix

	Native	Cross compiled/Interpreted	Hybrid	Web
Performance	X	X		
Native look&feel	X	X	x	
Native API	X	X	X	
Works offline	X	X	X	
Code reuse		x	X	X
Easy development			X	X

Considering pros and cons, author chose hybrid approach for the frontend development. Main factors of the decision were ease of the development and code reuse. Web approach has these advantages also but it miss other advantages like access to native API or offline functionality and therefore was not chosen.

From hybrid frameworks, author chose Ionic framework because of its popularity, huge community and complete documentation.

4. In case of Xamarin this requires application server which can run web application written in C#

5 Backend development

The backend is developed in the Java programming language with use of Java Enterprise Edition (Java EE) API. Its architecture consists of four layers. Each of those layers has its own functionality and communicate only with adjacent layers.

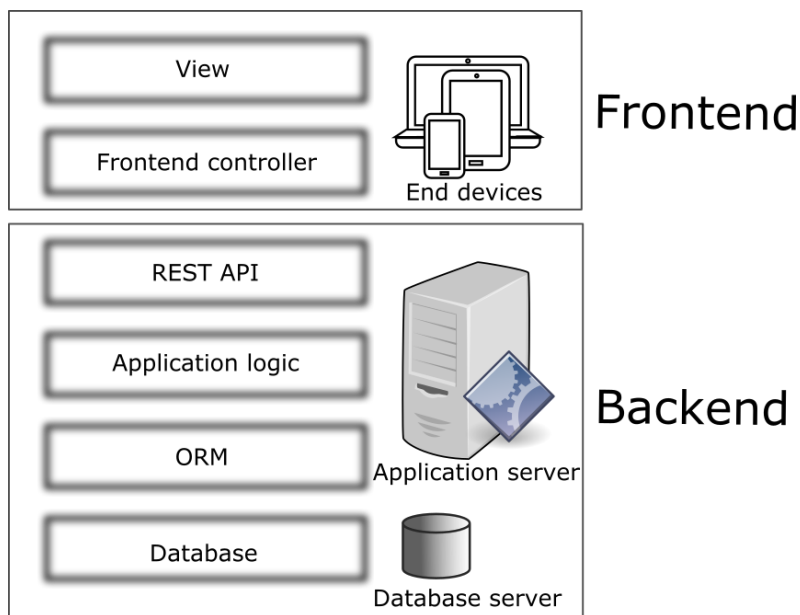


Figure 5.1: Application layers

Despite the fact that the database server and the application server could each run on its separate physical server, for simplicity they both run on the same physical server, provided by Software Development Europe (SDE) company.

The application server uses Web API to manage data requests from a frontend controller. The application logic then solve the request by accessing the database and updating/retrieving requested data.

For communication with the database an ORM (Object Relational Mapping) framework is used. The purpose of the ORM framework is to map Java classes onto tables in database and provide interface for managing data stored in relational database through Java code without using raw SQL queries.

5.1 Database

For the database layer, the relational database management system (RDBMS) has been chosen, because it is the most popular DBMS, suitable for such kind of applications and author already has experience with it. Two of the most popular open source RDBMS's are MySQL and PostgreSQL. They both support Linux server and Java language. The major criterion of decision between these two is performance benchmark from 2014 [14], according to which the PostgreSQL RDBMS is faster in read, insert and also update operations. Therefore PostgreSQL database has been chosen.

5.1.1 Entity Relationship Diagram

The database scheme is shown in the ERD (Entity Relationship Diagram) below. Tables "tokens" and "passwordreset" are related to the authentication process (Section 5.5).

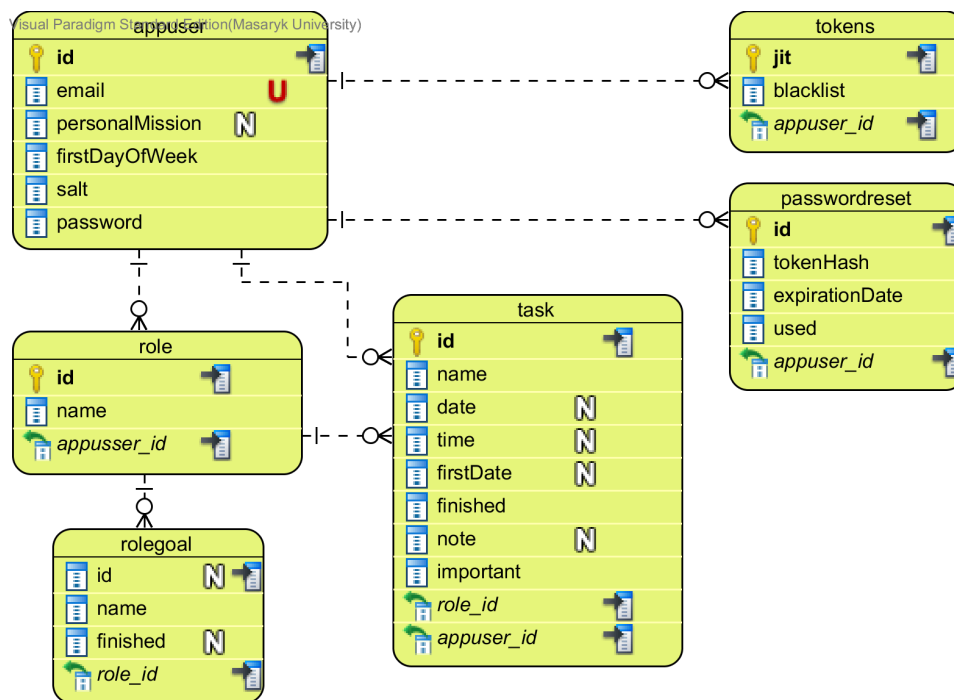


Figure 5.2: ERD diagram

5.2 Application server

5.2.1 Wildfly

Cross-platform time management application is a complex application which needs access to various Java EE technologies and therefore needs to run on an application server which is fully compatible with Java EE platform.

Two open source application servers from the Oracle list of compatible application servers are GlassFish Server Open Source Edition and Wildfly.

Whereas the support of commercial version of the Glassfish Server ended with version 3.x [15], Wildfly is a Red Hat community project which development goes hand-in-hand with commercial JBoss Enterprise Application Platform (JBoss EAP) project [16]. Because those two projects have much of their code base the same, the author concluded that the quality of Wildfly is potentially higher than in GlassFish case.

5.2.2 Configuration

The configuration of the Wildfly server is done in the standalone.xml file. This file can be edited either via graphical admin console, CLI (Command Line Interface) or directly in the standalone.xml source code. Among others, it was necessary to configure logging, data source and mail subsystems and also HTTPS redirecting and SSL (Secure Socket Layers) keys and certificates.

Wildfly logging subsystem is set to write warning and error messages to the periodically rotating file. The rotation period is set to one day, so each day, a new file with name following this pattern: liferoles.log.yyyy-mm-dd, is created. The file is not created if no log appears during the day.

Because the application sends sensitive data via internet, some sort of encryption is required to protect data. Therefore HTTPS (Hypertext Transfer Protocol Secure) protocol was chosen for the communication. To achieve that it is necessary to obtain an SSL certificate ¹. During the development phase it is possible to generate own SSL certificate using the Keytool program, which is contained in every Java distri-

1. The configuration of SSL can be found in Appendix A

bution. However, for the production use it is necessary to obtain a certificate from trusted certification authority. Most of such certificates are paid, however there are also certification authorities which offers free certificates. Author get one of such from StartCom.

5.3 Object relational mapping

5.3.1 Hibernate

Author decided to use ORM framework for communication with the database, because it is one of the most popular approaches nowadays. In Java EE application it is common to use the JPA (Java Persistence API) standard for ORM. However, JPA is only the standard without implementation. Specific ORM implementations usually offer implementation of the JPA standard, but also their native API, which is independent on the JPA standard.

One of the most popular ORM frameworks is the Hibernate ORM framework. Hibernate can be used both with its native API or as the JPA provider. The Hibernate JPA provider is pre-installed and used as the default JPA provider in Wildfly.

Because of the lack of knowledge about Java EE, the author chose to use the native Hibernate API. Later, for educational purposes, the code was remodeled to use the Hibernate JPA implementation. The advantage of using JPA standard is that the ORM provider can be changed without remodeling the code.

5.3.2 Mapping and relations

Although Hibernate allows the opportunity to generate DDL (Data Definition Language) commands from classes and their configuration, author has decided to create the database structure on his own. With this approach, author have full control over the database structure and does not have to rely on the Hibernate generation strategy.

After the tables creation it is necessary to map those tables into Java classes. In Hibernate, this can be achieved with two approaches: XML descriptors or annotations. Author chose annotations mapping, which is newer, less verbose and more intuitive approach.

The mapping between entities is done with respect to the application logic. All relations between Java objects are displayed in Figure 5.3.

For example, the relation between task and role is many-to-one relation (each task has reference to its role). This allows to manage tasks independently on their roles and also each task can display its role in the application dialog. Relation between role and long-term goal is, conversely, many-to-one relation (role has reference to all its long-term goals), because long-term goal is tightly linked with its role and is never managed separately. Even though, according to Covey, task should be ideally linked with some long-term goal, this relation is not implemented in the application. However, the relation is not even shown in the Covey's weekly planning tool [1], and therefore it is left on the user to keep this relation in mind.

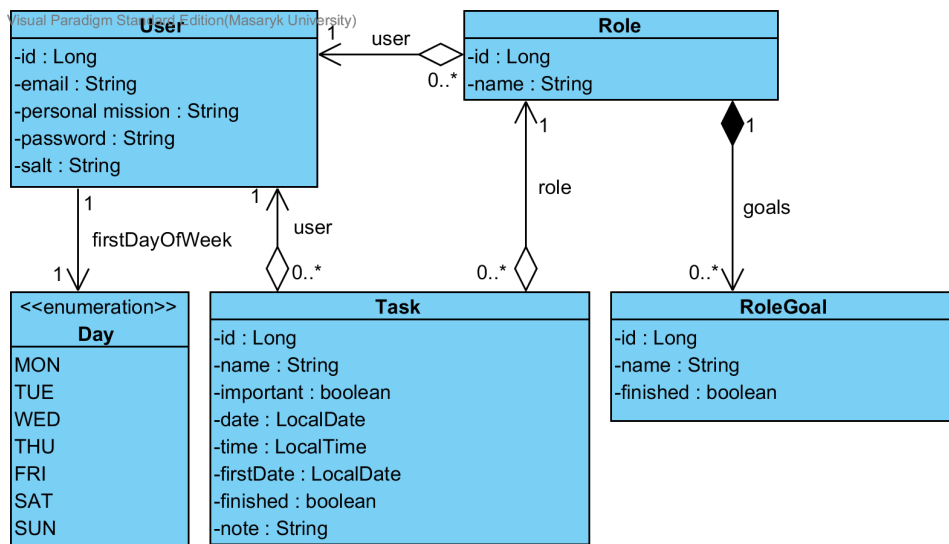


Figure 5.3: Analysis Class Diagram

5.4 Web API

5.4.1 REST architecture

The Web API of the application is designed rather to follow fundamentals of the REST (Representational State Transfer) architectural

style described by Roy Thomas Fielding² in his dissertation [17], then to follow RPC (Remote Procedure Call) architecture.

In RPC protocols like JSON-RPC (JavaScript Object Notation RPC) [18], the HTTP request URI and/or body specify an operation which will be executed on the server, together with optional parameters and data. RPC protocols usually use only two HTTP methods: GET and POST, for all requests, which violates the HTTP specification of these HTTP methods.

In the REST architecture, instead of specifying the operation, the HTTP request URI specifies identifier of the resource, on which the operation is executed. The body of the request specifies the representation of the resource [17] (mostly JSON or XML data). The operation is specified with correct usage of HTTP methods, mainly POST, GET, PUT and DELETE. These HTTP methods are mainly used for CRUD (Create, Retrieve, Update, Delete) operations.

Author chose the REST architecture over RPC mainly because it is more resource-oriented than operation-oriented which is suitable for the application which purpose is to manage CRUD operations on users' data. REST also does not violate the HTTP specification, and with the correct usage of HTTP methods, requests are uniform and self-descriptive [17].

5.4.2 Web API specification and implementation

Author defined four types of resources which are managed via the Web API: User, Role, Task and Authentication, which is more operation-oriented and serves for authentication and authorization purposes, but is still treated as a resource. Because client side uses JavaScript to manage the data, the JSON media type was chosen as a data representation.

Resource URI usually follows this pattern:

`https://liferoles.sde.cz/rest/resourceType/web | m/resourceID`³

The "m" or "web" determines the platform from which the request was sent (mobile/web). This splitting is against REST, however it is necessary to distinguish these requests, because web application

2. One of the authors of the Hypertext Transfer Protocol (HTTP) specification

3. See Appendix B for specific Web API endpoints

and mobile applications uses different form of authentication (see Section 5.5).

Resource ID can be either a numeric identifier of specific task/role or other identifier, which identifies the resource, for example month for which the statistics should be computed and retrieved. Although users have also numeric IDs, these IDs are not sent in URI, because application server recognizes authenticated users⁴.

To implement the Web API, the Java EE JAX-RS (Java API for RESTful Web Services) is used. Author used RESTEasy implementation of JAX-RS, because it is pre-installed in Wildfly.

Mapping URIs to operations and representations to classes is done with annotations. Here is simplified example of Java class used to handle task requests.

```
@Path("/rest/tasks")
public class RestTask {
    @PUT
    @Path("/web/{taskId}")
    @Consumes(MediaType.APPLICATION_JSON)
    public void updateTask(@PathParam("taskId") Long taskId,
        Task task) {
        //method body
    }
}
```

RESTEasy automatically serializes/deserializes⁵ Java object to/from JSON string. However, the application's Web API also defines custom classes for serialization/deserialization of objects. Default classes serializes all data from Java object into JSON, however some of them are useless. As explained in Subsection 5.3.2, each task has reference to its role and each role has reference to list of its long-term goals. However, long-term goals data are never referenced from the task object in the application. To reduce the size of transferred data, these data are not serialized when serializing the task object.

4. Either their IDs are contained in authentication tokens or in session attributes (see Section 5.5)

5. Process of translating data into a format that can be transmitted across a network and reconstructed later in the same or another computer environment

5.5 Authentication

Web application users are authenticated by form-based authentication using sessions managed by the application server, because it is a common type of authentication in the web application development. Whereas this type of authentication is suitable for web applications, it is not so suitable for mobile applications because they are different in many things.

For example, that application server redirects unauthenticated users of the web application to the login web page, when they try to access the Web API. However, mobile application never access any web page, even that one for login. It has its own login section, to which the user should be redirected after unsuccessful authentication.

Another difference between web applications and mobile applications is that in web applications, it is common that users are logged off when they close a web browser. This is because application server uses session cookies, which are destroyed when the browser is closed. However, in mobile application case, it is common that once users logs in, they are never logged off, until they requests it.

One of the most important differences is that whereas web application accesses to the Web API from the same domain, the mobile application doesn't. Therefore it makes cross-origin requests, which must be allowed on the server side. Server has to filter all responses and add CORS (Cross-Origin Resource Sharing) headers to each of them to allow cross-origin requests. The problem is that the "j_security_check" action, which is called during Java EE form-based authentication, is handled by the application server and ignores filters created by the developer.

Because of all these differences between mobile and web platform, author decided to use authentication with signed tokens for the mobile application. The advantage of such authentication is that the authentication process is managed more by the developer than by the application server module. Although it would be possible, with great effort, to use session authentication for both platforms, and it would be also possible to use signed tokens authentication for both, it would still require customization of the authentication process.

5.5.1 Sessions

Application server is configured to reject access for unauthenticated users to all web pages except the login page. The access is also rejected for all data request with path like `"/rest/web/*"`. If unauthenticated users try to access some of these restricted sources, the web application redirects them to the login page, where they can also register their account or send a request for password reset.

When users send their credential, the server authenticates them. In the case of success, the server generates the session id and stores it together with user's id to the memory. The session id is returned in the response and stored on the user's side in a session cookie. The session has no timeout set, so it is killed only when users log out from the application or close their web browser.

The server's authentication method is defined by the Wildfly login module configuration. Wildfly offers database login module, which compares credentials of users with data stored in database. Although this login module supports password hashing, the application uses other hashing functionality. The reason is that the default login module offers only MD5 and SHA1 hashing algorithms, which are too fast to compute and therefore makes the use of brute force attacks easier. Default login module also doesn't support salting, which is an important technique for protection against rainbow tables and lookup tables crack attempts by concatenating password with some other string before computing the hash. Therefore, the hashing provided by the default login module is not considered as appropriate [19]. Login module of the application uses PBKDF2 hashing function which uses salting and is CPU-intensive [19].

5.5.2 Signed tokens

Signed token is a text string, which contains user-specific data and a signature provided by the server. The signature is usually computed by the HMAC (Keyed-hash Message Authentication Code) algorithm, which arguments are user-specific data and a secret key (which is stored on the server). A token is sent to users after registration and stored locally either in a cookie or in HTML5 storage. With each HTTP request, a token is sent back to the server. The server extracts user-

specific data from the token, computes the signature again and compares result with the signature contained in the token.

The application uses JWT (JSON Web Token), which is increasingly popular standard (RFC 7519), for signed token authentication. On the server side, the Java JWT library is used to validate the token.

When users starts the mobile application, application looks for the JWT token in the HTML5 local storage of the device. If the token is not found, the application redirects users to the login section where they have to provide credentials to obtain the token. Otherwise it sends request for users' data to the server with the JWT token in the request header. Application server validates token and either returns data or error status code. Error can occur in two cases; either is the token compromised (signatures do not match) or the token is marked as blacklisted in the tokens table.

Tokens table contains JTI (JSON Token Identifier) of every token created by the server, id of user which own particular token and flag which can indicate that the token is on the blacklist. The purpose of the blacklist is to enable users to block access from their device, if they lost it. From the web application, users can mark all their tokens as blacklisted. Token is also marked as blacklisted every time users log off from they mobile application.

5.5.3 Password reset strategy

If users forgot their password they can request for a reset link. When this happens, server generates a new token, stores its hash into the database and sends an email, containing the token in URI, to users. Users have to click on the link which redirects them to the web application, where they can change their password.

The table used for this purpose contains token hash, id of a user, id of record and also expiration date and "used" flag. Expiration date is there to make sure that no one who gets access to the user's email can't use the token after some time. The expiration date is set to date of creation plus one hour. "Used" flag is there to prohibit multiple use of the same link (attacker with access to the browser history could abuse the link which was already used).

6 Frontend development

Because the hybrid approach was chosen for creation of the mobile application, the whole source code of the mobile application consists of HTML, CSS and JavaScript. Author decided to create the mobile application first and then try to reuse as much code as possible for the web application.

6.1 Ionic Framework

6.1.1 Installation

To begin with Ionic Framework, it is necessary to install Node.js, Cordova and Ionic¹. Application itself can be developed in whatever IDE² (Integrated Development Environment) on whatever platform but to build the application package and debug the application, following stuffs are required:

- Windows OS and Visual Studio for Windows application
- Mac OS and Xcode for iOS application
- Windows/Linux/Mac OS and Android SDK for Android application

Nevertheless, the application can be debugged in a desktop web browser most of the time, so the build is only needed when debugging mobile specific features eg. native API functionality.

6.1.2 AngularJS components

To develop a complex application in Ionic, it is necessary to use AngularJS. Its main components and their usage in the application are described below.

1. Ionic installation guide: <http://ionicframework.com/docs/guide/installation.html>

2. For the development, Eclipse IDE with Ionic and AngularJS plugins was used

Controllers and Services

Controllers are AngularJS components, which variables and functions can be accessed from HTML files. Their purpose is to manage the view layer of the MVC. In the application there is specific controller per each application page (settings controller, statistic controllers ...).

Services should be independent on the view layer. Their purpose is to provide data and methods. Controllers can inject services and then access their data and methods. This way data are shared between controllers. The application uses many third party services but also own service for managing tasks and roles (lists of data and related functions).

Directives

Directives are markers on DOM (Document Object Model) elements (attributes, element names, classes) which purpose is to attach a specific behavior to that element [20].

```
<ion-item ng-repeat="role in roles" ng-click="someFn()" ">
  {{ role.name }}
</ion-item>
```

In the source code example above, "ion-item" is a directive provided by Ionic. On the background, it generates <div> element with specific styling and features.

"ng-repeat" is, on the other hand, directive provided by AngularJS. Element marked with "ng-repeat" is repeated for every role object in roles list. The particular role object is then accessible inside that element. The list of roles is managed in a AngularJS controller file.

{{role.name}} expression is a shortcut for another AngularJS directive (ng-bind), which sets a watcher on the name variable of the particular role. When the variable is updated in a controller, the change is propagated to the HTML page.

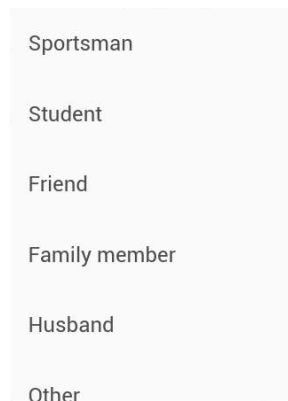


Figure 6.1: ng-repeat

6.1.3 Accessing native API

Native API of the device is accessed via Cordova plugins. Currently, there are more than 1200 plugins available but it is still possible that there are native APIs for which the plugin is not created. In that case, developers can create their own plugins with use of language specific for the OS (Java for Android, Objective C for iOS ...).

The application uses plugin for network information, with which it can recognize when a device has not connection to the internet. If there is no connection, warning message is permanently shown to the user.

Another plugin used by the application is a plugin for native date and time pickers. At first, author wanted to use HTML5 input with date type (`<input type="date">`), which should display a date picker of the web browser. However, this feature is not supported in the native browser of Android 4.1³ on which the application was tested.

6.1.4 Debugging

As earlier mentioned, the application was debugged in a desktop browser most of the time. For this kind of debugging, Mozilla Firefox and its Firebug plugin was used. However, when it comes to debugging usage of native mobile API the application must be built and either deployed to the device or run in a simulator.

Because simulators are very slow⁴ the application has been tested on real devices, mainly on device with Android 4.1.

To debug on an Android device it is only needed to build the application executing "ionic build android" in Ionic CLI, connect the device via USB and execute "ionic run android" command.

Debugging on an iOS device is more complicated. Earlier, it was possible to debug the application on an iOS device only with Apple Developer program, which costs \$99 per year. Now, with Xcode 7, it is possible to deploy the application to iOS device connected via USB [21]. Executing "ionic build ios" generates an Xcode project from which the application can be deployed to the iOS device using Apple ID of the developer.

3. Supported features of web browsers can be found on: <http://caniuse.com>

4. Launching of the application took about five minutes on the author's notebook.

In addition to input with date type, author found few other features which are not supported in Android 4.1 web browser. The Promise object, from ES6 specification (ECMAScript 6), is also not supported. It would be still possible to use promises, since there are some libraries which implements promise functionality, but for simplicity, author used callback functions. Another feature unsupported by Android 4.1 browser is CSS calc() function, which enables for example to set element's proportions as a difference between percentage and specific number of pixels.

6.1.5 Ionic evolution

Lot of things has changed from the time when author started with development of this application, in Ionic framework, in November 2015. Since February 10, 2016 there is a new Ionic Framework 2 Beta [22]. Whereas Ionic 1.x is built on the top of AngularJS 1.x versions, Ionic 2 is built on the top of Angular 2, which is also still in beta version. Angular 2 has different syntax and components than its older versions, just as Ionic 2. Therefore it is not trivial to migrate from Ionic 1 to Ionic 2 and for that reason, the final application of this thesis remains Ionic 1 application.

Whereas, in its beginning in 2013, Ionic 1 focused on imitating native look and feel of iOS, latter it tried to imitate the native look and feel of Android as well, by extending and overriding old CSS styles [23]. Majority of elements in Ionic 1 has usually the same code base and look the same on both platforms. For more Android-like design, developer has to override CSS files. However, in Ionic 2, there is no longer extended, IOs-based, CSS file but separate CSS file for each platform, recently also for Windows Phone. It is also no longer needed to split the source code to gain platform-specific look and feel. Some of differences between styling are shown in Figure 6.2.

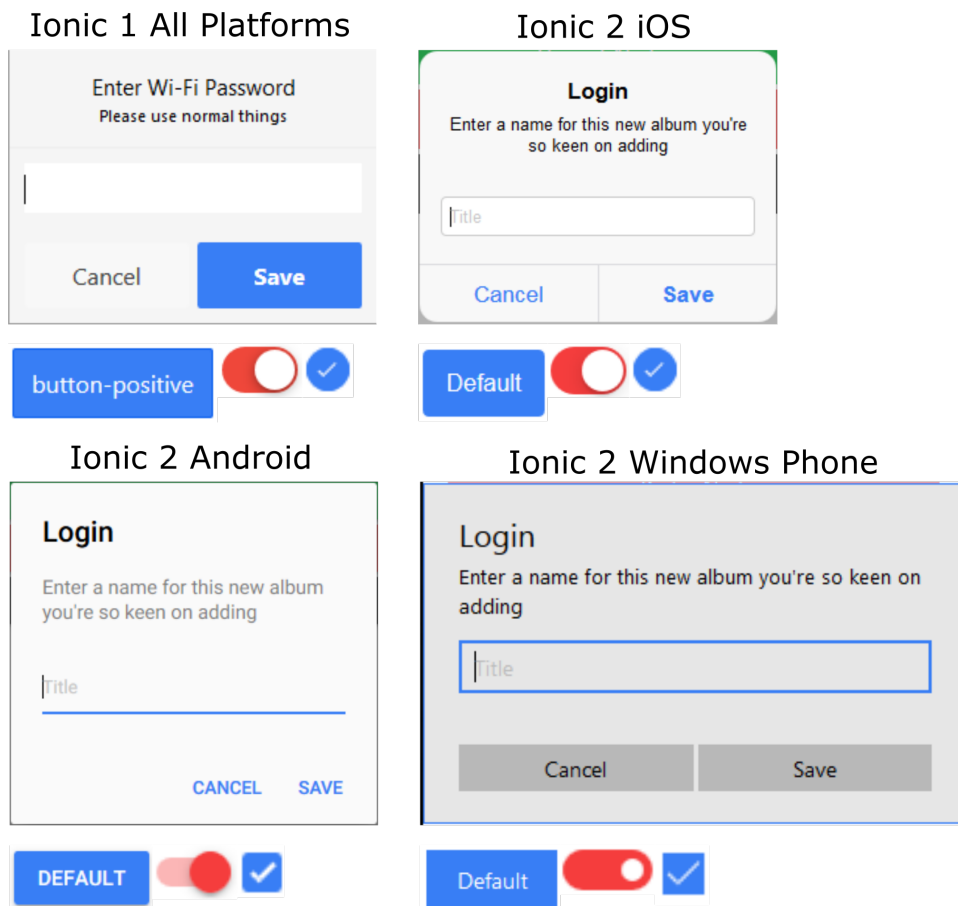


Figure 6.2: Difference in design: Ionic 1 vs. Ionic 2

6.2 Web application

For the development of the web application, author took the source code for mobile application and rewrote some of its parts, like GUI controls layout.

Most of the application logic can be shared between mobile and web platforms. Different code is written usually in cases of authentication and usage of native API of the device. In numbers, ca. 1200 lines of JavaScript code are shared between platforms and about 330 lines are specific for mobile/web platform. That is 78.5% of code reuse.

At first, author has copied JavaScript source codes from the mobile application to the web application and then maintained changes for mobile and web platforms separately. Later, for easier maintenance, source codes were merged, so now instead of updating two files, author can update one file and just overwrite the second one. Platform-specific parts of the code are handled by conditions according to the global variable "platform".

HTML and CSS file were not shared throughout the development. However, in the end author recognized that those CSS styles which do not change the layout of the application can be shared. Splitting the CSS file into shared and platform-specific files resulted in ca. 50% of code reuse. HTML files remains splitted even if they shares vast major of the source code.

6.3 Synchronization

Even though the hybrid application could utilize its potential to work offline, store data locally and synchronize it with the server once in while, for simplicity, it updates the database every time it update its locally stored data, just like the web application.

With this approach, data on the server are never dirty. However data on the device can be dirty. For example, if user deletes task from the mobile application, the web application is not updated, because it has its data cached. To keep the devices' data actual, web application requests data from the server every 5 minutes to update its cache. To reduce the size of transferred data, mobile application update its cache only when the application retrieves from the background, where it was at least 10 minutes.

If a user changes the same data on different devices in a short interval, this synchronization based on refreshes will not work. In that case, the application behaves as follows:

- If a user updates the same data on both devices, the latest version of data is considered as the right.
- If a user tries to update data which were already removed, data are created again.

- If a user tries to delete data which were already deleted, the application prompts the user for manual refresh in the settings section.

6.4 Application features

Among features which are provided by competing applications, like concept of roles, personal mission, and long-term goals, there are two features specific for this application.

Application displays graphical feedback on planning efficiency to motivate users. For frontend implementation, author used Angular-nvD3 library, which provides great amount of highly customizable charts. Statistics are computed on the server, which fills library-specific objects with data and returns those objects in HTTP response. Frontend has only to display charts filled from received data.

Second specific feature of the application is its planning layout. In the tasks section, next to the list of tasks of currently viewed week, there is a column with days of the week. A user can drag task and move it under other day of the week or to/from backlog. This enables users to manage their tasks really quickly. This layout also enables users to create all tasks for the week without assigning a date to them immediately and then drag them one by one into the schedule, which is a way of planning recommended by Covey [1], unsupported in competing applications.

Screenshots from the application are placed in Appendix C.

7 Testing and publishing

7.1 Integration testing

Author wrote unit tests¹ for backend Java classes at first, but due to frequent changes of the backend design, mainly in the beginning of the development, those tests were no longer relevant. Because the majority of the backend logic consists of linking of the Web API with the ORM API, it was rather important to focus on integration testing² than on unit testing. Author found updating unit tests too time consuming and therefore rather spent time with manual integrated tests via frontend Web API. Later, automated integrated tests had been created for this purpose.

For automated integration testing, the Java REST Assured library has been used. REST Assured provides intuitive method chaining API for creation and execution of HTTP requests and subsequent parsing and validation of HTTP responses. Even though REST Assured is a library for integration testing, it can be used with JUnit³ testing framework.

For each Web API endpoint (authentication, user, role and task) the specific test class is created. Test classes test almost all methods of the Web API, except methods which require usage of some third party resources, like registration of the web application, which uses CAPTCHA⁴ and therefore can not be automatically tested this way. Test methods usually test that the response's status code matches with expected status code and/or that returned JSON data matches expected data. If the HTTP request does not return any data but has to update the database (for example update of the resource), in addition to the status code, test method also accesses the database to check if the expected update has been made.

-
1. Tests created individually for each method and runs in isolation
 2. Testing of multiple combined modules
 3. Popular framework for unit testing
 4. Test used to differentiate humans from robots

7.2 Beta testing

To test the GUI, author has released beta version of both, web and mobile application. These applications are still being tested by users. All users has opportunity to report bugs and enhancement tips at liferoles.app@gmail.com, directly from the application.

For better tracking and managing of issues and enhancements, author used the JIRA tool (a tool for agile development) provided by SDE company. Users' emails are rewritten as issues in JIRA and managed with the JIRA tool. Among author and few users, the application was tested also by Libor Kubečka, SW development manager in SDE company, and Daniel Doherty, specialist in digital marketing of social media in Grafform, UK.

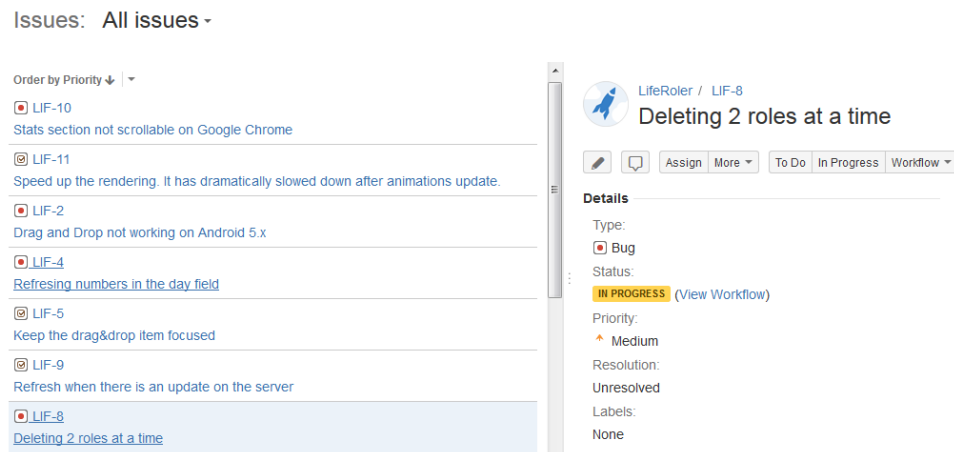


Figure 7.1: JIRA issues

There is one alarming bug, which has not been solved yet. Whereas drag and drop functionality works in desktop web browsers and on iOS 8.3 and Android 4.1 devices, it does not work on Android 5.x devices. Because drag and drop is not provided by Ionic, application uses third party library, which does not guarantee support from browsers. Because the drag and drop functionality is an enhancement, which is not necessary to use the application properly, the problem remains open. However, author will try to solve the problem, probably by using other library.

7.3 Publishing

Web application beta version was published in 23.3.2016 at <https://liferoles.sde.cz>. At first, application used self-signed SSL certificate, which resulted in browsers security warning. Consequently the application was shared only with few testers. In 17.4.2016 the application started using SSL certificate signed by StartCom and also the Android application was published to the Google Play. Since then, until 15.5.2016, 20 users registered to the application, from which, 6 were actively using the application. The count of devices, which have installed the Android application was 16.

Publishing the mobile application to the Google Play market requires the Google developer account, which costs \$25 for registration. The application package can be also distributed outside the Google Play by download the package and installing it via Android installer. To publish the application on Google Play, the Google developer account of SDE company was used. When publishing the application on Google Play, author appreciated Ionic CLI "ionic resources" command, which generates icons and splash screen images for various screen sizes from one source image.

To publish the iOS application to the Apple App Store, the Apple developer account is required, which costs \$100 per year. Neither SDE company, nor Author invested into it and therefore the application for iOS is not available at the Apple App Store. Unlike Google, Apple doesn't let the application package to be distributed outside of the App Store.

8 Conclusion

Time management system described by Stephen R. Covey differs from systems most people use. Because its less popularity, there are only few applications supporting this system.

The main objective of this thesis was to develop cross-platform application for time management of the fourth generation. Based on research of possibilities of cross-platform development, author decided to use Ionic Framework for this purpose, mainly because the ease of development and code reuse. Author used Web API for communication and synchronization of data between end devices and remote server. The final application, called Liferoles, is a cross-platform application, available for desktop browsers and devices with Android and iOS.

Because hybrid mobile development and related topics are new, and technologies are quickly evolving, many things has changed since the development started. Ionic Framework 2 and mainly Angular 2 differs a lot from their older versions. Because the migration is not trivial, the application presented in this thesis is built in Ionic Framework 1.x and AngularJS 1.x versions. However, the migration to the Ionic 2 is definitely one of the most important step that will be done in the future. Another important plan to the future is to upgrade the mobile application to work offline with use of local storage of the device and synchronization.

Among these two big steps there is a great amount of already planned enhancements to make the application more user-friendly. Author believes, that with long-term maintenance and support, the application can become popular tool for time management of the fourth generation.

A Setup of the backend server

PostgreSQL 9.4 database server runs on the physical server with Debian OS. After the installation via "apt-get" command, the "postgres" service is automatically set and started. To create a new database user, it is necessary to login to the "postgres" database as "postgres" superuser. New database can be created by newly created user or by "postgres" superuser. To control the access of other users to the created database, the pg_hba.conf file has to be edited.

Wildfly 9.0.1 application server runs on the same physical server as the database server. Wildfly is distributed via .tgz archive, which has to be unpacked in the desired location. The easiest way to register Wildfly as a service is setting up the wildfly-init-debian.sh script, which is contained in the Wildfly distribution.

To connect the Wildfly server with the PostgreSQL server it is necessary to configure the PostgreSQL datasource in standalone.xml file. Important parts of the datasource configuration are:

- JNDI name, which identifies the resource in the Java application
- path to PostgreSQL JDBC driver
- URL of the connection to the database
- name and password of the database owner

Another important part of standalone.xml is configuration of the SSL certificate (CRT) used by HTTPS protocol. In standalone.xml this is done by reference to a Java keystore file. Author got free SSL CRT from <https://www.startssl.com/>. After obtaining the data from StartSSL, there should be four files: private key, domain-specific CRT, StartSSL's intermediate CRT and StartSSL's root CRT. Three mentioned certificates should be concatenated into one file (bundle.crt) in the same order as they are mentioned. Then, via openssl command, the .pkcs12 file should be exported from the input private key and the bundle.crt file. Java keystore file is then exported from the .pkcs12 file via keytool command. When exporting the Java keystore file, it is necessary to write down its password, password of the key (can be the same) and alias of the key. These three values are then referenced in the standalone.xml file inside the <ssl> element of the <security-realm> ele-

A. SETUP OF THE BACKEND SERVER

ment, which is referenced inside the `<https-listener>` element, which is referenced inside the `<socket-binding-group>` element.

Redirection of HTTP to HTTPS is done in the `web.xml` file by specifying `<web-resource-collection>` of the `<security-constraint>` element with `<transport-guarantee>` set to `confidential`. If the Debian server has to listen to HTTPS on port 443, the Wildfly server must be started under the root user (which is the default behavior if it is running as a service). The port must be also allowed by the server's firewall.

B Web API

In this section, the Web API is documented. This could be useful for third party applications which could use this API to access the data and perform more advanced data analysis (eg. custom statistics, data mining and visualization). However, the Web API was not created for this purpose and therefore would have to be extended to provide more methods suitable for data mining.

To access most of the API, the authentication is required. Third party applications should get the authentication token by sending POST request to: `https://liferoles.sde.cz/rest/auth/m/login` with following JSON data: `{email:'userEmail',password:'usersPassword'}`. Content type must be set to **application/json** as with each other request. If credentials were valid, the server returns a token which must be assigned to every request which requires the authentication. The token must be send in a header of the request, where key is 'Authentication' and value is 'Bearer *token*'.

In the following documentation, in the case of JSON, **bold**-marked properties are required, other are optional. If the optional property is omitted, it is set to its default value.

Users API

URI: `https://liferoles.sde.cz/rest/users/m`

Method	URI parameters, JSON data	Result
GET		Returns user.
POST	<code>/password</code> <code>{"oldP":"old","newP":"new"}</code>	Changes user's password.
POST	<code>/mail</code> <code>{"password":"pass",</code> <code>"email":"new"}</code>	Changes user's email.
POST	<code>/data</code> <code>{"firstDayOfWeek":"4",</code> <code>"personalMission":"work"}</code>	Sets Friday as the first day of planned week and <i>work</i> as a personal mission.
POST	<code>/logout</code>	Invalidates the token.

Roles API

URI: <https://liferoles.sde.cz/rest/roles/m>

Method	URI parameters, JSON data	Result
GET		Returns all roles.
POST	<code>{"name":"role1","goals": {"name":"goal1","finished":false}, {"name":"goal2","finished":false}}}</code>	Creates new role and returns its ID.
PUT	<code>/1 {"name":"role1","goals": {"name":"goal1","finished":true}, {"name":"goal2","finished":false}}}</code>	Updates role with id 1. Omitted goals are deleted.
DELETE	<code>/1</code>	Deletes role with its goals and tasks.
DELETE	<code>/1&newRoleId=2</code>	Deletes role with its goals and moves tasks under other role.

Tasks API

URI: <https://liferoles.sde.cz/rest/tasks/m>

Method	URI parameters, JSON Content	Result
GET	/2016/5/30	Returns tasks from 30.5.2016 and newer, including backlog.
GET	/week/2015/11/10	Returns tasks between 10.11.2015 and 17.11.2015.
POST	<code>{"name":"task1","role":{"id":2}, "date":{"year":"2016","month":"05", "day":"16"},"time":{"hours":"03", "minutes":"30"},"note":"note", "important":false,"finished":false, "firstDate":{"year":"2016", "month":"05","day":"19"}}</code>	Creates new task and returns its ID. "firstDate" should be permanently set to the date on which the task was initially scheduled.
PUT	/1 <code>{"name":"task1","role":{"id":2}, "date":{"year":"2016","month":"05", "day":"22"},"important":true, "finished":false,"firstDate":{" "year":"2016","month":"05", "day":"19"}}</code>	Updates task.
DELETE	/1	Deletes task.
GET	/stats/2015/11?last=false	Returns month statistic. "last" should be true when retrieving data for current month.
POST	/backlog/2016/05/12	Moves uncompleted tasks from 12.5.2016 and older to the backlog.

C Screenshots from the application

Liferoles

Welcome

Welcome to the **Liferoles App!** Our aim is to provide you with a **simple** but **incredibly effective** tool to help improve your **time management**.

If you are looking to manage your time more effectively, please try out the **Liferoles Time Management App**. We have developed this application as an aid to help people follow [The 7 Habits of Highly Effective People](#) as written by Stephen R. Covey. One of the main ideas is that in addition to work, it is also important to find time for yourself and your relationships.

The **Liferoles app** is designed to help you become truly productive with your time.

Important

Update your Android app
With the [last update](#) we changed our database structure. Therefore the mobile application should be updated. If you have disabled automatic updates on your Android device, please find Liferoles in your Google Play and update it from there.

Beta version
This is a Public Beta Version - I.e. the application is still in development and if you want to help us improve it, then please do report bugs (or make suggestions) to us via liferoles.app@gmail.com.

Login Register

Email

Password

Repeat password

☐ I'm not a robot

Register

Figure C.1: Web application registration

Liferoles

Login Register

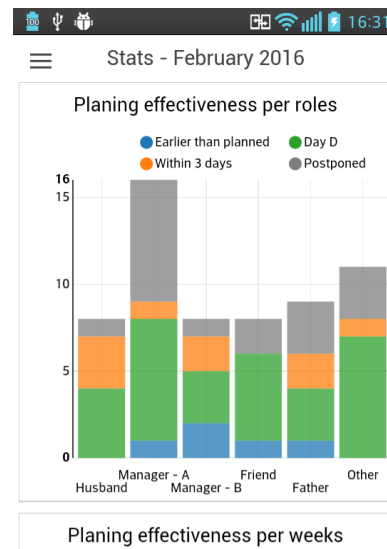
Email

Password

Repeat password

Register

(a) Registration



(b) Statistics

Figure C.2: Mobile application

C. SCREENSHOTS FROM THE APPLICATION

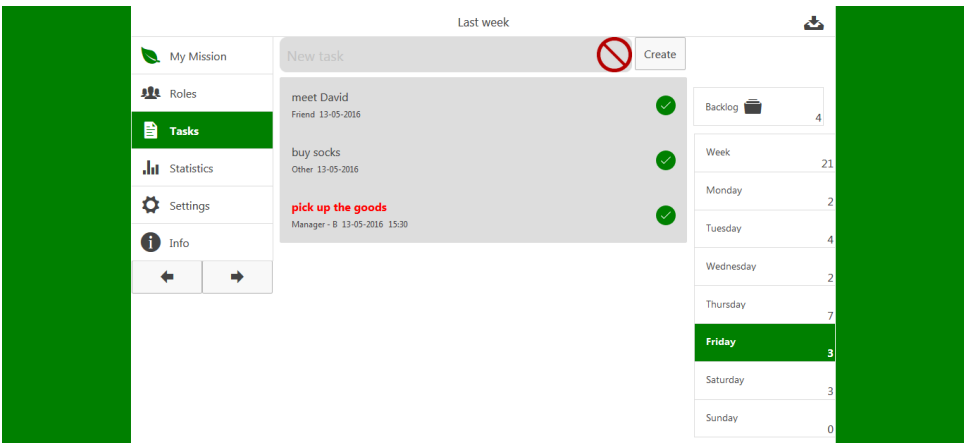


Figure C.3: Web application layout

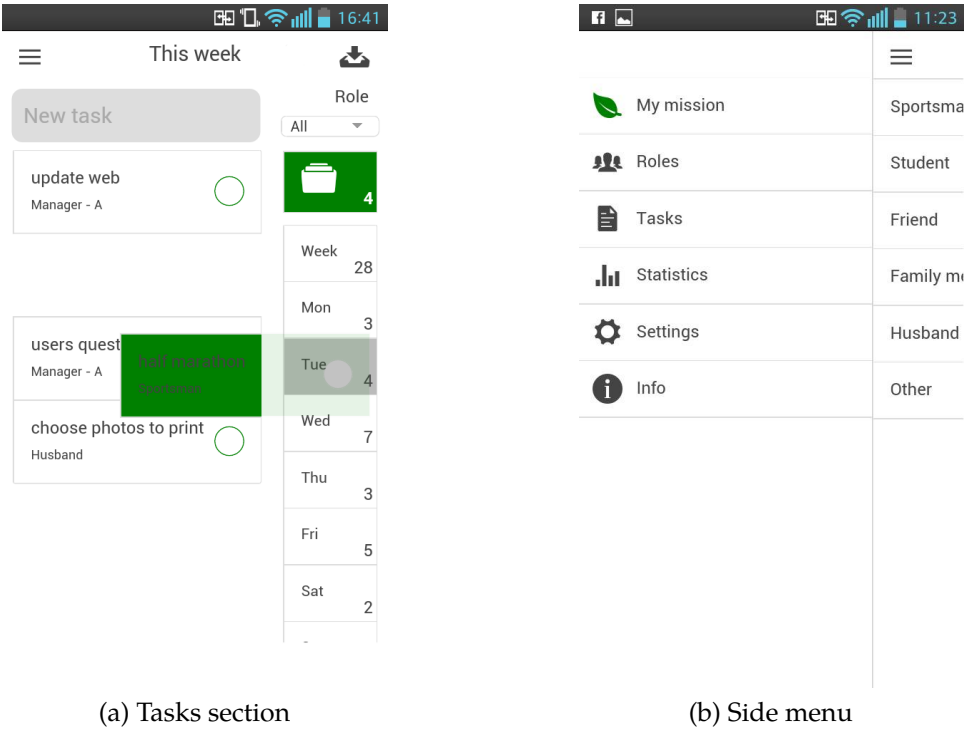


Figure C.4: Mobile application layout

D List of electronic attachments

Following files are submitted along with the thesis:

- *liferoles-web.zip* – source code of the Liferoles web application together with Java classes used for integration testing
- *liferoles-mobile.zip* – source code of the Liferoles mobile application
- *standalone.xml* – configuration file of the Wildfly application server in XML format
- *liferoles.sql* – SQL script for creation of database tables in PostgreSQL
- *thesis.zip* – source of this thesis in \LaTeX format and related resources

Bibliography

- [1] Stephen R. Covey. *The 7 habits of highly effective people*. 3rd ed. Praha: Managment Press, 2014. ISBN: 978-80-7261-282-6.
- [2] Andtek blog, ed. *My Effectiveness Habits. Ideas Post*. 2012. URL: <http://andtek.blogspot.cz/2011/11/my-effectiveness-habits-ideas-post.html> (visited on 04/22/2016).
- [3] Google trends, ed. *my effectiveness app. Interest over time*. 2016. URL: <https://www.google.com/trends/explore#q=my%20effectiveness%20app> (visited on 04/22/2016).
- [4] Erich Vokral. *Building Mobile Friendly Websites. Responsive vs. Mobile Version*. 2016. URL: <http://allwebcodesign.com/responsive-vs-mobile-sub.htm> (visited on 05/05/2016).
- [5] R. Raj and S.B. Tolety. "A study on approaches to build cross-platform mobile applications and criteria to select appropriate approach". In: *India Conference (INDICON), 2012 Annual IEEE*. 2012.
- [6] Ionic documentation, ed. *Ionic Documentation Overview*. 2016. URL: <http://ionicframework.com/docs/overview/> (visited on 05/08/2016).
- [7] Kevin Whinnery. *Comparing Titanium and PhoneGap*. 2012. URL: <http://www.appcelerator.com/blog/2012/05/comparing-titanium-and-phonegap/> (visited on 02/03/2016).
- [8] Appcelerator documentation, ed. *Alloy XML Markup*. 2016. URL: http://docs.appcelerator.com/platform/latest/#!/guide/Alloy_XML_Markup (visited on 01/05/2016).
- [9] Appcelerator website, ed. *Titanium SDK*. 2016. URL: <http://www.appcelerator.com/titanium/titanium-sdk/3/> (visited on 05/08/2016).
- [10] Jason Quackenbush, ed. *PLEASE remove Starter edition!* 2015. URL: <https://forums.xamarin.com/discussion/32947/please-remove-starter-edition> (visited on 05/05/2016).
- [11] Xamarin website, ed. *Xamarin is now a member of the Visual Studio family*. 2016. URL: <https://www.xamarin.com/compare-visual-studio> (visited on 05/05/2016).

BIBLIOGRAPHY

- [12] Visual Studio website, ed. *Now with Xamarin*. 2016. URL: <https://www.visualstudio.com/en-us/features/xamarin-vs.aspx> (visited on 05/08/2016).
- [13] Xamarin website, ed. *Jump-start your next project with a prebuilt app*. 2016. URL: <https://www.xamarin.com/prebuilt> (visited on 05/08/2016).
- [14] Sani Adeyi, Yusuf Abubakara, and Abudu Abimbola Oriyomi. "BENCHMARKING POPULAR OPEN SOURCE RDBMS: A PERFORMANCE EVALUATION FOR IT PROFESSIONALS". In: *International Journal of Advanced Computer Technology* 3 (2014).
- [15] John Clingan. *Java EE and GlassFish Server Roadmap Update*. 2013. URL: https://blogs.oracle.com/theaquarium/entry/java_ee_and_glassfish_server (visited on 04/22/2016).
- [16] Red Hat website, ed. *Red Hat JBoss community or enterprise*. 2016. URL: <https://www.redhat.com/en/technologies/jboss-middleware/community-or-enterprise> (visited on 04/22/2016).
- [17] Roy Thomas Fielding. "Architectural Styles and the Design of Network-based Software Architectures". PhD thesis. University of California, 2000.
- [18] Jeffrey Damick. *JSON-RPC over HTTP*. 2008. URL: <http://www.jsonrpc.org/historical/json-rpc-over-http.html#get> (visited on 05/19/2016).
- [19] Defuse Security Website, ed. *Salted Password Hashing - Doing it Right*. 2016. URL: <https://crackstation.net/hashing-security.htm> (visited on 04/27/2016).
- [20] AngularJS documentation, ed. *Directives*. 2016. URL: <https://docs.angularjs.org/guide/directive> (visited on 05/09/2016).
- [21] Stack Overflow contributors, ed. *Test iOS app on device without apple developer program or jailbreak*. 2015. URL: <http://stackoverflow.com/questions/4952820/test-ios-app-on-device-without-apple-developer-program-or-jailbreak> (visited on 05/11/2016).
- [22] Adam Bradley. *Announcing Ionic Framework 2 Beta*. 2016. URL: <http://blog.ionic.io/announcing-ionic-framework-2-beta/> (visited on 05/10/2016).
- [23] Max Lynch Adam Bradley. *The Ionic Show. Ionic 2*. 2016. URL: https://www.youtube.com/watch?v=u6BFxtv_L-8 (visited on 05/10/2016).