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

Replace this page with a copy of the official signed thesis assignment and the copy of the Statement of an Author.

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

Abstract

Keywords

Contents

1 Introduction

Time management helps people to be more productive and effective. Calendars, diaries, lists with notes, 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. Covey[1], which differs from the others by its philosophy and construction.

From those who use some kind of time organizer, some still use classic paper organizers, but 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 few popular applications for Covey's organizer.

This thesis deals with the whole process of development of a cross-platform application tailored especially for the Covey's organizer. First part includes the summary of popular time management applications and the analysis of current possibilities of cross-platform development. Second part includes the system design, the implementation of user-friendly, attractive, cross-platform application, testing and publishing.

The final application, named Liferoles, meets all requirements and is capable to compete concurrent products. Application is cross-platform and allows user to access his data from more devices without loss of consistency. To achieve this, users' data are stored remotely on database server. User-friendliness is reached by splitting the application into two branches – mobile application[2] for mobile devices with Android or iOS and web application[3] for devices like PCs or mobile devices with other OS. Both applications offer similar looks, but the layout and controls are customized according to the platform used.

---

1. Well-known leadership instructor and author of bestseller The 7 habits of highly effective people.
2. Available for free at Google Play and App Store e-shops.
3. Available at: `https://liferoles.sde.cz`.

In the next chapter time management and its four generations are described and popular applications are mentioned. After that current approaches used for cross-platform development are examined. The system design and the description of the back-end development follows. After that the front-end development, including GUI, authentication and description of both mobile and web application is described, followed by information about testing and publishing.

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 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 works all weeks long, omitting rest and their personal and social needs, they can be successful in short-term, but in long-term it will results in 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 e.g. parent, partner, manager, lecturer, sportsman and more.

Whereas in other generations a person could have these roles in mind, this generation requires to write them down. After that, a person should write one or two long-term goals which would like to achieve in each role and personal mission, which is a statement consequent from roles and their goals. 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 should a person 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 more such tasks, tho worse the situation will get. Important tasks needs to be done and the urgency together with lack of a time cause

stress. Some of these tasks unpredictably bumps into the schedule but most of them could have been solved earlier.

2. Important but not urgent

Tasks of 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 how to renew power. The seventh habit called Sharpen the Saw 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

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 an good-looking, complex application for devices with Android OS, which strictly follows the Covey's ideology. The application is available on the Google Play store since Ocrober 2011, and falls into the cattegory of 100 000 - 500 000 downloads, which makes it the most famous application for the 4th generation time management for Android OS. Information about development and guides how to use the applications are available at the developers blog[1].

The application enables users to write down their personal mission, which is a statement consequent from roles and 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 much 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.

Great weakness of this application is that it is available only for Android devices, although people are asking for a web application

1. `http://andtek.blogspot.cz/`

on the developers blog since 2012 [2]. According to the Google search statistics[2], the amount of people looking for the application on the internet is now about 90 per month, which confirms that the web application demand is high.

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. Attractive feature is opportunity to synchronize tasks with Google apps or Microsoft Exchange accounts, although this was not working when tested. creation of tasks is fast and controls are simple and intuitive so the application does not look 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.

2. `https://www.google.com/trends/explore#q=my%20effectiveness%20app`

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

3.2 Use-case diagram

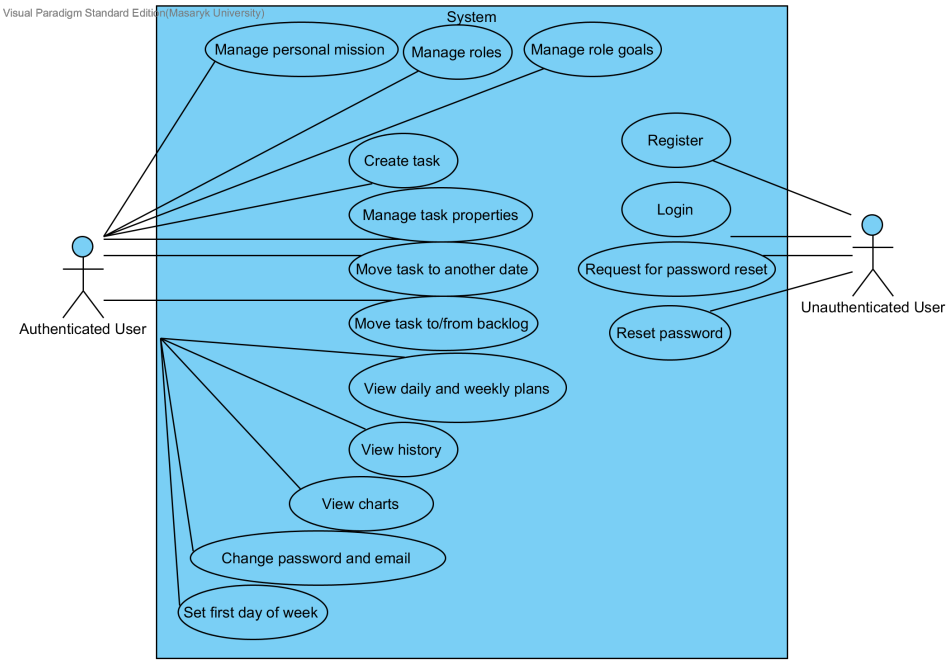

Figure 3.1: Use-case diagram

3.3 Application structure

To fulfill the cross-platform non-functional requirement it is necessary to separate the application into two parts - backend, which runs on a remote server, and frontend, which runs on the end device.

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 the user create his account, he can access his data from whatever device.

The purpose of the frontend part is to get specific data from the backend part via internet, store them locally and display them to the user. If a user update their locally stored data, those data can be either synchronized with the server in real-time, or after some trigger (time, event, etc.).

The backend part is developed in the Java programming language with use of Java Enterprise Edition (Jave EE) API. The development process is described in the chapter 4.

The frontend part depends on the approach to the cross-platform frontend development, which is described in the chapter 5. The development of the frontend part is described in the chapter 6.

4 Backend design and implementation

Backend of the application consists of four layers. Each of those layer has its own functionality and communicate only with adjacent layers.

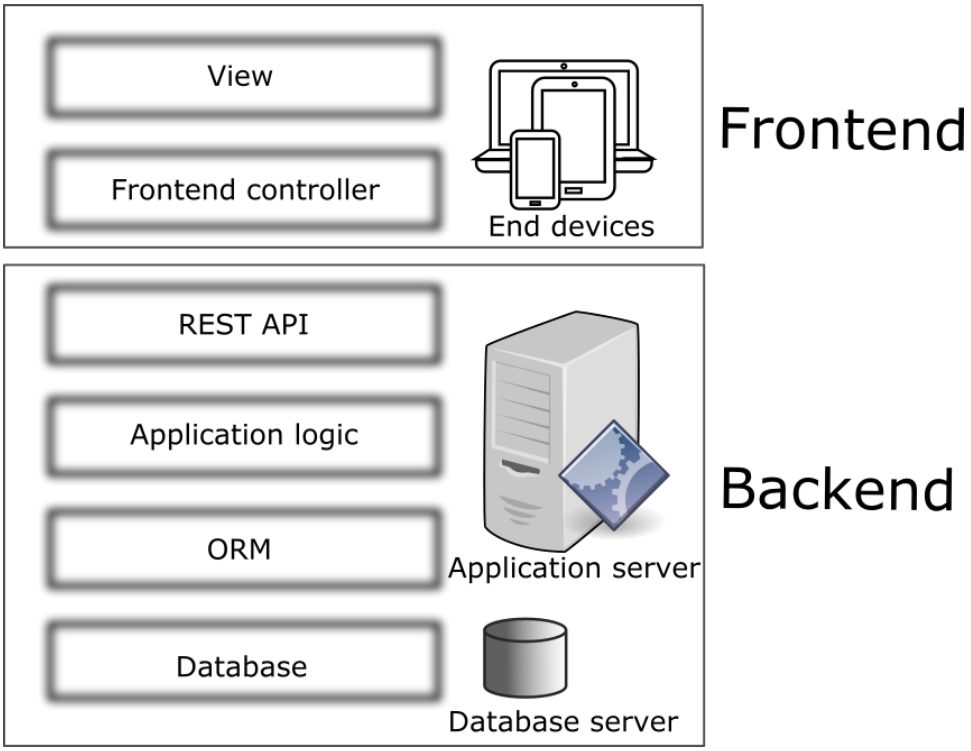

Figure 4.1: Application layers

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 SDE - Software Solutions company.

The application server uses REST (Representational State Transfer) 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 Object Relational Mapping (ORM) framework is used. The purpose of 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.

4.1 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 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 [3], according to which the PostgreSQL RDBMS is faster in read, insert and also update operations. Therefore PostgreSQL database has been chosen.

4.2 Application server

4.2.1 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 [4].

Whereas the support of commercial version of the Glassfish Server ended with version 3.x [5], Wildfly is a Red Hat community project which development goes hand-in-hand with commercial JBoss Enterprise Application Platform (JBoss EAP) project [6]. Because those two projects has much of their code base the same, the author concluded that the quality of Wildfly is potentially higher than in GlassFish case.

4.2.2 Configuration

The configuration of the Wildfly server is done in the standalone.xml file. This file can be edited either via admin console (GUI), CLI (Com-

mand 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 warn 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 it should use HTTPS protocol for the communication. To achieve that it is necessary to obtain an SSL certificate. During the development phase it is possible to generate own SSL certificate using the keytool program[1], which is contained in every Java distribution. However, for the production use it is necessary to obtain a certificate from trusted certification authority. Most of such certificates are payed, however there are also certification authorities which offers free certificates. Author get one of such from StartCom[2].

4.3 Object relational mapping

4.3.1 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 standard without implementation. Specific ORM implementations usually offers 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 [7].

---

1. `http://docs.oracle.com/javase/6/docs/technotes/tools/windows/keytool.html`
2. `https://www.startssl.com/`

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. However, in the end, the author found the native Hibernate API better, because it is more feature-rich and supports new technologies which are not supported by JPA.

4.3.2 Mapping and Relations

Although Hibernate allows a user to generate DDL (Data Definition Language) commands, for better control, the author has decided to create the database structure on his own. After the tables creation it is necessary to map those tables onto Java classes. In Hibernate, this can be achieved with two approaches - XML descriptors or annotations [8]. 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 the section 4.5. Here are two examples of relations with reasoning.

The relation between task and role is many-to-one relation. Task has reference to its role, because in the application section with task's details, there must be displayed which role is assigned to the task. However, it is not needed for a role to have a reference to a list of its tasks, because in the application, tasks are not grouped by roles but by dates. Also, if the relation was one-to-many, with each role all its tasks would be loaded, which is inappropriate, because the application history shouldn't be loaded until the user requests for it.

On the other hand, the relation between role and long-term goals is one-to-many relation. Role is tightly linked with its long-term goals. Every time a role is loaded, all their long-term goals are loaded too. Also, in the application, long-term goals are managed in the same section, where a role is managed. Therefore there is no need for long-term goal to have a reference to a role, because they are never managed independently.

4.4 REST

To communicate between frontend and backend the Jave 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.

First step to configure RESTEasy is to set the URL path pattern for the RESTEasy servlet. Author set the URL pattern to "/rest/*", which means that each request which path starts with "rest" will be handled by RESTEasy servlet.

Mapping HTTP requests to classes and methods is done with annotations. Author divided requests into four groups - authentication requests, user requests, role requests and task request. Here is simplified example of Java class used to handle task requests.

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

This method is fired when client sends HTTP PUT request to URL with path "/rest/tasks/web". The word "web" determines that the request was send from the web application. Request from mobile applications are recognized by the "m" letter. It is necessary to distinguish these requests, because web application and mobile applications uses different form of authentication (see section 4.6).

Last important feature of the application's REST API are custom classes for serialization/deserialization of objects. Default classes serializes all the data from Java object, however some of them are useless. As explained in subsection 4.3.2, each task has reference to its role and each role has reference to list of its long-term goals. However, the long-term goals data are never referenced from the task object in the application. To save the data transfer, these data are not serialized when serializing the task object.

4.5 Design class diagram

cca half page

4.6 Authentication

Web application uses different authentication method than mobile applications. Author chose to authenticate web application users with use of sessions and cookies, because it is most common type of authentication in the web application development. In web application case, it is common that a user is logged of when he close his browser. However, in mobile application case, it is common that once a user logs in, he is never logged of, until he request it. To achieve that behavior the JSON Web Tokens approach is used for authentication of mobile users. Although it was possible to use only one form of authentication and customize it according to platform, author chose to use both, mainly for educational purposes.

4.6.1 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 a user 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 a user sends their credential, the server authenticate them. In 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 the user logs 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 the user with data stored in database. Although this login module supports password hashing, author extended this module with his own hashing functionality (see subsection 4.6.4). The reason is that the default login module offers only

MD5 and SHA1 hashing algorithms and also doesn't support salts and therefore is not considered as secure [9].

4.6.2 JSON Web Tokens

cca 3/4 of page

4.6.3 Password reset strategy

cca half page

4.6.4 Hashing

cca half page

4.7 ERD diagram

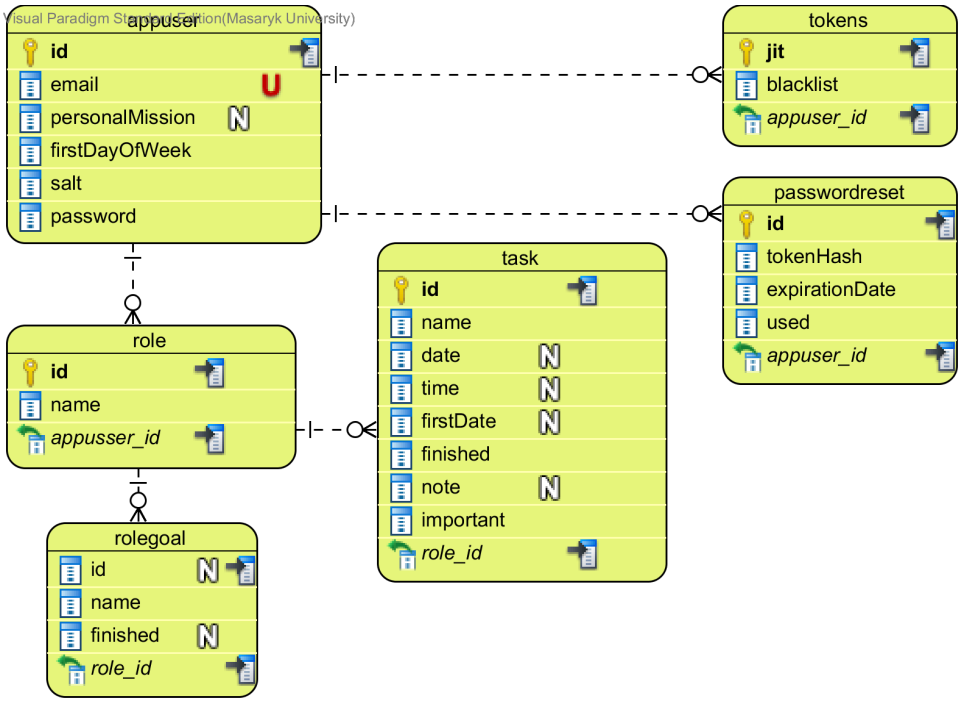

Figure 4.2:

5 Cross-platform frontend development

Cross-platform application should be independent on operating system (Windows, Linux, Android, iOS), type of device (smartphone, tablet, PC) and screen size. It is common that the final product of cross-platform development is an application which has different versions for different platforms. It is due to the fact that differences between device types and screen sizes are perceptible. The technologies which are widely used to achieve the goal are described bellow.

5.1 Web applications

Web applications are programs which allow users to communicate with the internet, where the content is stored, using their web browser. From the technical point of view, a web application is a dynamic website. Its structure and content is written in HTML (HyperText Markup Language) and look is described by CSS (Cascading Style Sheets) files. To make a website dynamic it is necessary to use some scripting language which handles events generated when users interacts with their web browsers.

Scripting language can be server-side or client-side. The difference between these two is that server-side scripts runs on the remote server, whereas client-side scripts runs on the client side - in the web browser. Server side scripting is more secure, because the whole application logic is held on the server and therefore hidden from possible attackers. Another reason why to go server-side is that it is possible to upgrade hardware and software structure of the server according to the application workload. If the scripting is done at user's device, the performance lays on the user's hardware which can be weak and therefore client-side scripting can dramatically slow down the application. On the other hand power of client-side scripting is that the communication between the server and client can be minimized. Because the scripting is done on the client's side, the application can use web caching[1] to reduce http requests to server. Considering the performance of today's

1. A technology for storing web server responses locally.

mobile phones the disadvantage of computational load on the client side becomes insignificant and client-side scripting is widely used.

Main advantage of web applications is the multiplatformicity. Applications run on remote servers and in web browsers, which are contained in every OS (operating system), so every device with OS can use web application without installation. They are portable just by the share of a link and people can find them on the internet by using their web searchers. Developers can update websites 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 and usability of platform specific controls compared to other technologies for cross-platform development. Another disadvantage is that the internet connection is needed to communicate with remote server.

In the past, websites were developed mainly for desktops and other devices with large screens. Nowadays, due to powerful hardware and software, mobile phones enable people to browse websites and the percentage of page views from mobile phones is constantly growing. In May 2012 it was 10.11 % [10]. But there is still one big difference between desktops and mobile phones and that is screen size. Websites which are not optimized for mobile phones can still be viewed by users with zooming, but it is uncomfortable. Consequently, developers try to make both websites and web applications more user friendly with two approaches - responsive websites and mobile websites.

Responsive website

Responsive website adapts its layout according to platform used by visitor. Screen size is the key attribute in 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 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 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, since it can't use full potential of the concrete platform. It takes more time to create 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.

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. When there is already a website optimized for desktops, it is easier to develop a mobile version of the same site than to transform it to responsive design. 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.

Pros and Cons

+ Portability
+ No installation
+ Instant updates
+ Can be found by web searcher
– Internet connection needed
– Weak performance
– Lacking support of native controls

5.2 Native mobile applications

A common way of development of cross-platform applications is to create a web application for devices with large screen and native applications for mobile phones. Native mobile applications are developed

in specific programming languages according to specific OS of the device. For example native applications for devices with Android OS are developed in Java while iOS development is done in Objective C [11].

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 notifications[2] 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[3] 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 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.

Pros and Cons

+ Access to every native API
+ The fastest performance
+ Mobile e-shop
+ Works offline
– Expensive and challenging development

5.3 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

2. Notifications which are displayed to users even if they don't use the application at the moment.
3. Most famous mobile e-shops are Google Play for Android OS and App Store for iOS.

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 approaches used to attain their purpose. [12].

Hybrid approach

Hybrid mobile applications are mainly written in HTML, CSS and JavaScript. They run in the native 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, which allows the hybrid mobile application to use functionality like push notifications, camera and others, so the functionality of the application can be complex [13]. Differences between platforms are handled by specific hybrid mobile framework, so the application source code can be used for different platforms. With these features, the application written as a web application can behave like a native application. However, as the application depends on a native browser, it can be slower than native mobile application.

Native browsers are often used by native mobile applications, for example to refer to the documentation which is stored on the internet. This allows developers to update the documentation without needs to update the local version of the application installed on user's device. In this case the native mobile application occasionally uses the native browser. The hybrid mobile application works the other way round - the application runs in the native browser most of the time and if needed, it uses some native functionality of the device.

As the core of hybrid mobile applications is written in HTML5, CSS and JavaScript, a desktop web browser can display its content. However, web browsers don't necessarily support specific hybrid mobile framework technologies. The reason of possible incompatibility is that hybrid mobile frameworks add their own functionality, for example the use of native API.

Interpreted approach

In interpreted approach case the application source code is written in JavaScript. Unlike in the hybrid mobile application case the GUI (Graphical User Interface) is not written in HTML and CSS because the application typically does not run in the native web browser of the device. Instead, the JavaScript source code is interpreted at the runtime by JavaScript interpreter[4]. All GUI elements of interpreted mobile applications are native and together with native features are accessible from the JavaScript objects' method calls.

In case of Appcelerator Titanium, which is the well-known framework for interpreted mobile applications, there is the Titanium API for this purpose. JavaScript objects are paired with native objects and shares properties and methods. With using Titanium API, methods called on JavaScript objects invokes native methods [14].

Compared to hybrid applications interpreted applications are faster and looks more native. However, to achieve native look and behaving, it is sometimes necessary to write platform specific code, because different platforms have different features and not all of them are handled by framework.

Cross compiled approach

Cross compiled approach is very similar to interpreted approach since the result is also native application. The difference is that cross compiled applications are written in C# and then compiled to native code. Well-known framework for this purpose is Xamarin.

Pros and Cons

+ Cross-platform
+ Works offline
+ Mobile e-shop publish
- Could be slower than native application but faster than web application

4. V8 for Android, JavaScriptCore for iOS

- Access to native API

6 Ionic framework

cca 2 pages

7 Testing and publishing

cca 1 page

8 Conclusion

cca 1 page

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] 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).

[4] Oracle website, ed. *Java EE Compatibility*. 2015. URL: http://www.oracle.com/technetwork/java/javaee/overview/compatibility-jsp-136984.html (visited on 04/22/2016).

[5] 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).

[6] 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).

[7] Scott Marlow. *JPA Reference Guide*. 2014. URL: https://docs.jboss.org/author/display/WFLY9/JPA+Reference+Guide (visited on 04/25/2016).

[8] Emmanuel Bernard. *Hibernate Annotations. Reference Guide*. 2010. URL: https://docs.jboss.org/hibernate/stable/annotations/reference/en/html_single/ (visited on 04/25/2016).

[9] Defuse Security Website, ed. *Salted Password Hashing - Doing it Right*. 2016. URL: https://crackstation.net/hashing-security.htm (visited on 04/27/2016).

[10] Mobiforge website, ed. *Global mobile statistics 2014*. 2014. URL: https://mobiforge.com/research-analysis/global-mobile-statistics-2014-home-all-latest-stats-mobile-web-apps-marketing-advertising-subscriber (visited on 11/22/2015).

[11] Andre Charland and Brian Leroux. "Mobile application development: web vs. native". In: *Communications of the ACM* 54 (2011).

[12] 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.

[13] Patrick Rudolph. *Hybrid Mobile Apps. Providing A Native Experience With Web Technologies*. 2014. URL: http://www.smashingmagazine.com/2014/10/providing-a-native-experience-with-web-technologies/ (visited on 11/22/2015).

[14] Kevin Whinnery. *Comparing Titanium and PhoneGap*. 2012. URL: http://www.smashingmagazine.com/2014/10/providing-a-native-experience-with-web-technologies/ (visited on 02/03/2016).