ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Ryšavý** Jméno: **Filip** Osobní číslo: **420410**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Otevřená informatika**

Studijní obor: **Softwarové inženýrství**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Mobilní aplikace pro MyICPC se sběrem contextu v IoT prostředí

Název diplomové práce anglicky:

Mobile app for MyICPC with context acquisition for IoT environments

Pokyny pro vypracování:

- Prozkoumejte možnosti sběru signálu a meta-informací o síti na WiFi a Bluetooth v mobilních platformách Android, případně iOS.
- Věnujte pozornost WiFi Channel State Information a meta-informacím o dalších zařízení na síti. Dále amplitudě a fázovému posunu [2,3].
- Prozkoumejte systém MyICPC [4] a interakci s mobilním zařízením pro účel interakce a účasti ve hře Quests možností zaslání videa a obrázku.
- Prozkoumejte zabezpečení proti kompromitaci skrze Trusted Execution Environments (TrustZone) [5].
- Navrhněte způsob komunikace mezi mobilním zařízením a MyICPC, tak aby byli mimo jiné předány kontextové informace jako je GPS, a okolní zařízení.
- Implementujte prototyp mobilní aplikace, rozšiřte komunikační část MyICPC a otestujte použití.

Seznam doporučené literatury:

[1]Michals Trnka, Martin Tomasek, and Tomas Cerny. Context-aware security using internet of things devices. In Kuinam Kim and Nikolai Joukov, editors, Information Science and Applications 2017: ICISA 2017, pages 706?713, Singapore, 2017. Springer Singapore.
[2] Cong Shi, Jian Liu, Hongbo Liu, and Yingying Chen. Smart user authentication through actuation of daily activities leveraging wifi-enabled iot. In Proceedings of the 18th ACM In- ternational Symposium on Mobile Ad Hoc Networking and Computing, Mobihoc ?17, pages 5:1?5:10, New York, NY, USA, 2017. ACM.
[3] Ioannis Agadakos, Per Hallgren, Dimitrios Damopoulos, Andrei Sabelfeld, and Georgios Por- tokalidis. Location-enhanced authentication using the iot: Because you cannot be in two places at once. In Proceedings of the 32Nd Annual Conference on Computer Security Applications, ACSAC 16, pages 251?264, New York, NY, USA, 2016. ACM.
[4] Smetana Roman, Next generation of Second-Screen, Realtime application MyICPC, 2016,https://dspace.cvut.cz/handle/10467/62711
[5] Claudio Marforio, Nikolaos Karapanos, Claudio Soriente, Kari Kostiainen, and Srdjan Capkun. Smartphones as practical and secure location verification tokens for payments. In NDSS, 2014."

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Tomáš Černý, MSc., Ph.D., laboratoř inteligentního testování softwaru FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **16.10.2017** Termín odevzdání diplomové práce: **25.05.2018**

Platnost zadání diplomové práce:
do konce letního semestru 2018/2019

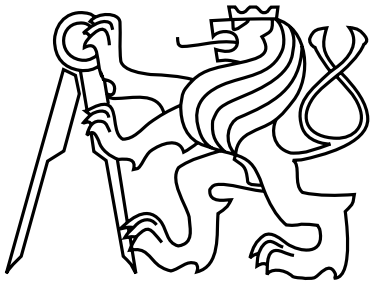
Ing. Tomáš Černý, MSc., Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Ing. Pavel Ripka, CSc. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

.	
Datum převzetí zadání	Podpis studenta

Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Computer Science and Engineering



Master's Thesis

Mobile application for MyICPC with context acquisition for IoT environments

Bc. Filip Ryšavý

Supervisor: Ing. Tomáš Černý, MSc., Ph.D.

Study Programme: Open Informatics, Master's degree

Field of Study: Software Engineering

May 24, 2018

Aknowledgements

I would like to thank my supervisor Ing. Tomáš Černý, MSc., Ph.D. and his colleagues for their continuous guidance and advices. I would also like to thank to all who supported me during studies.

Declaration

I declare that I elaborated this thesis on my own and that I mentioned all the information sources and literature that have been used in accordance with the Guideline for adhering to ethical principles in the course of elaborating an academic final thesis.

In Prague on May 24, 2018 .....

Abstract

This thesis targets to create a mobile application that will acquire the contextual information like a date, a location, network devices and others to support a research of context aware authentication using Wi-Fi enabled IoT devices. Next, this thesis aims to take this application and integrate it with some application from real environment in order to obtain data for the research. In a case of this thesis the integration is with MyICPC that is used during the ACM International Collegiate Programming Contest. This thesis presents a design and an implementation of the solution in a form of a context acquisition application, a MyICPC mobile application and an extension of the current version of MyICPC. The solution presented in this thesis is ready for the deployment during regional contest later this year.

Keywords: Android, context, context acquisition, ICPC, mobile applications, second screen

Abstrakt

Cílem této práce je vytvořit mobilní aplikaci, která dokáže sbírat kontextové informace jako datum, lokace, zařízení v síti a jiné za účelem podpoření výzkumu kontextu uvědomělé autentizace s využitím IoT zařízení využívajících Wi-Fi. Dále je cílem vzít tuto aplikaci a integrovat ji s nějakou aplikací z reálného prostředí za účelem získání dat pro výzkum. V případě této práce se jedná o integraci s aplikací MyICPC, která se používá během ACM International Collegiate Programming Contest. Tato práce představuje návrh a implementaci řešení v podobě aplikace pro sběr kontextu, mobilní aplikace pro MyICPC a rozšíření současné verze MyICPC. Řešení představené v této práci je připraveno pro nasazení během regionálních soutěží konajících se později tento rok.

Klíčová slova: Android, kontext, sběr kontextu, ICPC, mobilní aplikace, second screen

Překlad názvu: Mobilní aplikace pro MyICPC se sběrem kontextu v IoT prostředí

Contents

List of Figures

List of Tables

List of Listings

Chapter 1

Introduction

Today there is already a huge number of mobile devices and technologies overall. There is also a lot of demands on applications to provide a personalized content in this case based on user's context[1]. This property of applications is called a context awareness. Besides that, the context can be used in a context-aware security that is done only by a few applications nowadays[1]. The goal of this thesis is to create a mobile application that will acquire such context and then integrate it in a real mobile application as proof of concept.

This chapter provides an overall description of this thesis. First, this chapter describes a motivation of the context acquisition and how the acquired context will be used in the research dealing with a context-aware authentication using Wi-Fi enable Internet of Things (IoT) devices[2]. Next, this chapter describes an opportunity where the context acquisition can be applied in. The opportunity opens itself during the ACM International Collegiate Programming Contest[3] and consists of using the application called MyICPC[4] and creating a mobile application for it which integrates the context acquisition. Finally, this chapter describes issues that led to the creation of the mobile application and thus opened the opportunity for the context acquisition proof of concept.

1.1 Usage of acquired context in research

The research deals with a context-aware authentication using Wi-Fi enabled IoT devices and it proposes a method that addresses a problem of gathering a contextual information[2]. Every IoT device is by the definition connected to the Internet usually through a computer network. Such devices can serve as a data source of the contextual information which includes some information about network itself[2]. The acquired context needs to be tied to a specific user and the relationship between users and devices can be mapped in various forms[2]. The research is limited to a situation when a device is owned permanently by a single user and focuses on user's virtual location[2].

A large data set of gathered data should be analyzed using data science techniques to determine the security risk level[2]. Then, there should be determined devices that do not change for a given network in the given time and other devices that are tied with the user[2]. A context is acquired and stored during every action performed by the user and

then compared with historical values[2]. If devices present at the given time differ greatly from the devices present at the same time on previous days, it is flagged as suspicious[2].

First, this thesis aims to create a mobile application that can acquire user's context. The context acquired by the application will be used for purposes of the research.

1.2 International Collegiate Programming Contest

The ACM International Collegiate Programming Contest (ACM-ICPC) is a multi-tiered, team-based and programming competition[3]. The contest involves a global network of universities hosting regional competitions that advance teams to the ACM-ICPC World Finals[3]. The closest event is the 42nd Annual World Finals of the ACM-ICPC this year in April that takes place in Beijing, China. The next contests are regional competitions in autumn of this year.

The opportunity to use the context acquisition in a real mobile application is to create one for additional activities of the ACM-ICPC. These activities have been so far handled solely by a web application called MyICPC. The deployment of this mobile application is then expected during one of the closest events.

1.3 MyICPC

MyICPC is a web application that was deployed during all World Finals of the ACM-ICPC during the last few years[4]. The goal of MyICPC is to enhance the experience of the audience and thus it aggregates information from several ACM-ICPC sources and applications such as a social data from Twitter or some analytics of a contest, teams and submissions[4]. Thus, it eases the work of the user. MyICPC consists of several components such as the home page called Timeline, the main contest page called Scoreboard, the Quest game, the schedule, the poll and the gallery[4].

The second goal of this thesis aims to create a mobile application that will be able to send messages directly to the MyICPC home page and to the Quest game. At the same time, the mobile application will integrate the context acquisition, and the acquired context will be used for research purposes motioned in the section 1.1. This will also require designing a communication between the mobile application and MyICPC and since MyICPC does not provide any application programming interface (API), a creation of the API will be required as well.

1.4 The Great Firewall of China

The Great Firewall of China is a colloquial term for mainland China's internet censorship system[5]. It blocks foreign websites, apps, social media, emails, instant messages and other online resources deemed inappropriate or offensive by authorities[5]. The best option how to get around the firewall is to use a virtual private network (VPN), but even some of them are blocked by authorities[5].

This is the main motivation of creating a mobile application for the MyICPC home page and the Quest game because both are using a social data from Twitter that is blocked by the Great Firewall of China. The MyICPC servers run in the United States and they are accessible from China. The mobile application will be useful especially because this year World Finals takes place in Beijing, China as mentioned in the section 1.2. The idea of the mobile application is to be easier to use than setting up some VPN.

1.5 Proposed solution

The solution proposed by this thesis consists of the context acquisition, the mobile application for the MyICPC home page and the Quest game, and the extensions of MyICPC in the form of a REST API for a communication with the mobile application.

First, the mobile application that acquires a context will be created. It will be a tool which can be later integrated with other applications. It will collect a contextual information like a date, a location, some information about the device, a connected Wi-Fi network, available Wi-Fi networks, network devices, a signal strength, Bluetooth devices and user identity. It will mainly focus on the Wi-Fi networks and the network devices to support research described in the section 1.1.

The second step will be to create the mobile application for MyICPC itself. It will serve as front-end for the MyICPC home page and the Quest game. It will replace the usage of a Twitter and thus the concept of it will be quite like the Twitter mobile application. Basically, there will be a sing in screen, a wall of posts and screen with a new post. It will also integrate the context acquisition.

Finally, the MyICPC web application will need some modifications. It will require a REST API to transfer information to and from the mobile application. The REST API will provide functionality of the MyICPC home page and the Quest game. It also will need to handle user's authentication and storing of the acquired context and media files which will be sent by the mobile application instead of using Twitter.

1.6 Structure of thesis

The following part of this thesis is divided into chapters Analysis, Design, Implementation, Testing, Evaluation and Conclusion as follows:

- The chapter **Analysis** will explain what is the context and where the context can be used. The chapter will then analyze mobile platforms and existing MyICPC solution. The chapter will also describe Wi-Fi channel state information, trusted execution environments and some related work.
- The chapter **Design** first describes a design of the mobile application that will be acquiring the context and fulfilling the MyICPC functionality. The chapter then describes changes in the MyICPC web application.
- The chapter **Implementation** will describe the development environment, tools and the implementation of the mobile application and the server application.

- The chapter **Testing** will describe a testing of the MyICPC server application and the MyICPC mobile application. The testing consists of a unit testing of the persistence and the service layer, a stress testing of the MyICPC server application and an usability testing of the MyICPC mobile application.
- The chapter **Evaluation** will describe results of the implementation and the testing of both applications. It will also describe a deployment of the applications.
- The chapter **Conclusion** will summarize this thesis, the applications that were created in it and the overall results.

This thesis contains following appendices: the nomenclature, the content of the included CD that contains the complete source code of this thesis, the deployment guide of the modified MyICPC server application.

Chapter 2

Analysis

First, this chapter explains what the context is and how it is related to the context awareness. This chapter continues the explanation with examples and usages of the context and context awareness mainly in the context-aware security. Then, this chapter describes the context in the mobile environment and analyzes requirements for the context acquisition. This chapter also explains what is the Wi-Fi channel state information, trusted execution environments and some related work.

Second, this chapter describes and compares mobile platforms like Android and iOS. Then, this chapter explains a choice of the mobile platform for the MyICPC mobile application and analyses requirements for it.

Finally, this chapter provides a description of the architecture of MyICPC and describes the MyICPC timeline and the Quest game. Then, this chapter analyses requirements for an extension of MyICPC.

2.1 Context

The context can be understood as an information that characterizes a state of an entity[6]. The entity is an object that is relevant to the interaction between the application and the user[6]. The context can be classified as a computing context, an environmental context, a user context or a physical context.

A computing context can be a network connectivity, a communication cost, a communication bandwidth or nearby resources. An environmental context can be a light intensity, a noise level, traffic conditions or a weather. A user context can be user's identity like his profile, his preferences, his mood, his behavior or his pressure. Physical context can be a time, a date or a location. Note that a computing context like a network information is the most important for purposes of this thesis.

Today's mobile devices are equipped with GPS, Wi-Fi, a gyroscopic sensor and based on a device with various other sensors. The mobile devices are used on a daily basis by many people and thus they are great opportunity for collecting a contextual information. The mobile devices are suitable for collecting the computing context, the environmental context, the user context and the physical context.

2.1.1 Context awareness

A context-aware system uses a context to provide a relevant information to the user[6]. In other words, the context awareness (CA) is an ability of the system to gather an information, process it and adapt to it. Many applications use the CA to a certain degree today. These applications are usually social applications, health care applications or navigational services usually based on a user physical location.

For example, if a user has left his office and his phone starts ringing in his office, the system would detect user's location and redirect the call to his mobile phone and so on. Another usage of the CA is in the context-aware security.

2.1.2 Context-aware security

Applications are usually secured by the traditional ways like Mandatory Access Control, Discretionary Access Control or Role-Based Access control[7]. Only a few applications have the security based on the contextual information[1, 8]. The context-aware security (CAS) is using the context to improve security decisions. The CAS results in less obstructive applications and allows to use different authentication methods based on the context[1, 8]. In other words, the CAS is more personalized, fine-grained and dynamic.

An example of the CAS can be an extra authentication method forced on a change of a location. Another example can be a usage of a different authentication method when accessing from a company network and when accessing from another network. This can go even further as a system can process a bio-metric data and if the user would do some critical operation and his bio-metric data would be abnormal, the system may force addition security features. CAS is suitable in use for IoT devices where it is based on trusted devices[9].

2.2 Wi-Fi channel state information

The Wi-Fi channel state information or just channel state information (CSI) are channel properties in wireless communications that describe how a signal propagates from the transmitter to the receiver[10, 11]. A user can be accurately identified by extracting representative features from CSI measurements of Wi-Fi signals and further processing[10].

2.3 Trusted execution environments

System-wide trusted execution environments (TEEs) are used to secure mobile services despite a mobile operating system compromise[12]. An example of TEEs is ARM TrustZone[12]. TEEs provide an isolate execution of applications and a secure storage of credentials[12].

2.4 Mobile application

The current solution of the MyICPC application is done only as a web application. Nowadays, every mobile device like phones or tablets is equipped with some Internet browser

and thus displaying a web application is not an issue. It has, however, several disadvantages. The most important disadvantage is that a web application cannot access to the mobile device and its sensors. Another disadvantage is a user experience because a web application may not follow design guidelines for a specific mobile platform.

2.4.1 Platforms analysis

Today there are several mobile platforms or operating systems. The two most widely spread system are Android[13] and iOS[14].

The latest version of Android is version 8 called Oreo and it is used approximately by 5.7% of devices[13]. The most used versions of Android are version 6 called Marshmallow and version 7 called Nougat with approximately 25.5% and 31.1% usage respectively[13]. It is not recommended using the latest version as a minimal supported version as the application will target only a small portion of devices. It is also not recommended using the first versions as minimal supported version because the application would require tremendous backward support across all newer versions. For example, there was significant change of the permission system and policy between versions 5 and 6.

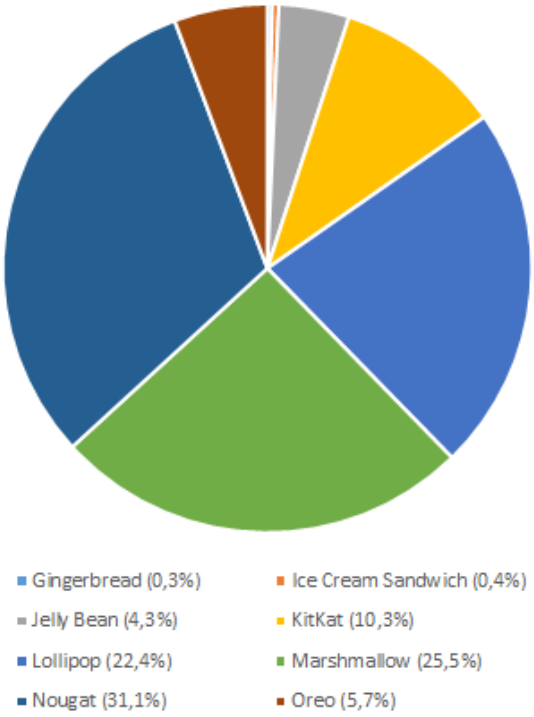


Figure 2.1: Usage of Android versions on May 7, 2018.

The latest version of iOS is version 11 which is used by approximately by 76% devices[13]. Principles about selecting a minimal supported version are like Android. A benefit of using iOS is that most devices are using the latest version of system and its features. However,

iOS is stricter with the security and restrictions of applications. For example, it limited the information that can be obtained like a device information or a network information.

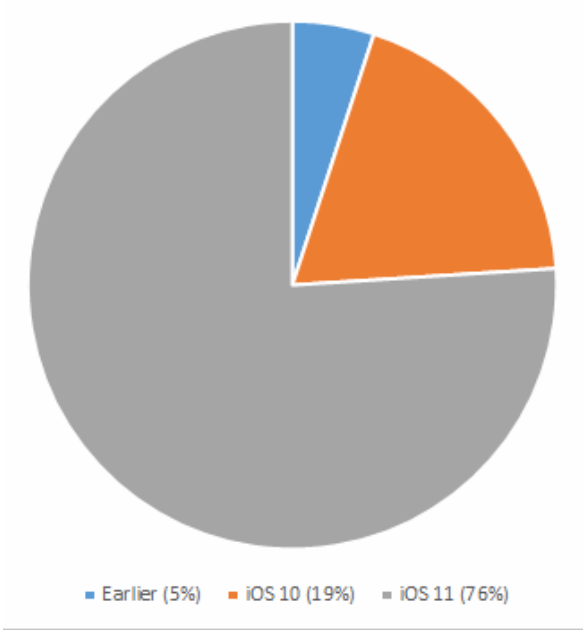


Figure 2.2: Usage of iOS versions on April 22, 2018.

This thesis will use Android as it is more accessible than iOS and it allows to acquire all contextual information specified in this chapter. This thesis will use version 6 as it is a compromise of amount of targeted devices and supported features.

2.5 MyICPC

MyICPC is a web application. The application uses the standard layered architecture[4]. The application consists of persistence layer, service layer, controller layer and presentation layer[4]. The application doesn't contain any API[4]. The application is built using Spring Framework[4]. The persistence layer uses Java Persistence API (JPA) and Hibernate[4]. The presentation layer uses JavaServer Pages (JSP) and JavaScript including AngularJS framework[4].

The application can be run as a single node application on one server as displayed in the figure 2.3[4]. The application can be also run as cluster by running each execution environment on a separate server and by multiplying a node with the application as shown in the figure 2.4[4].

The application consists of several components like the homepage called Timeline, the main contest page called Scoreboard, the Quest game, schedule, poll and gallery[4]. This thesis works with the Timeline and the Quest game.

The MyICPC application can be also classified as a second-screen application[4]. A second-screen application usually involves a device equipped with a screen and access to

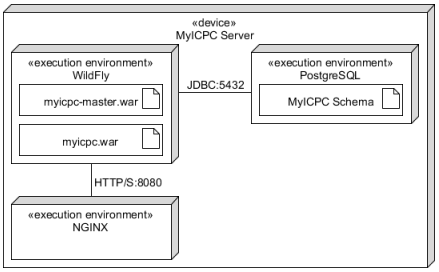


Figure 2.3: Deployment of MyICPC as a single node application.

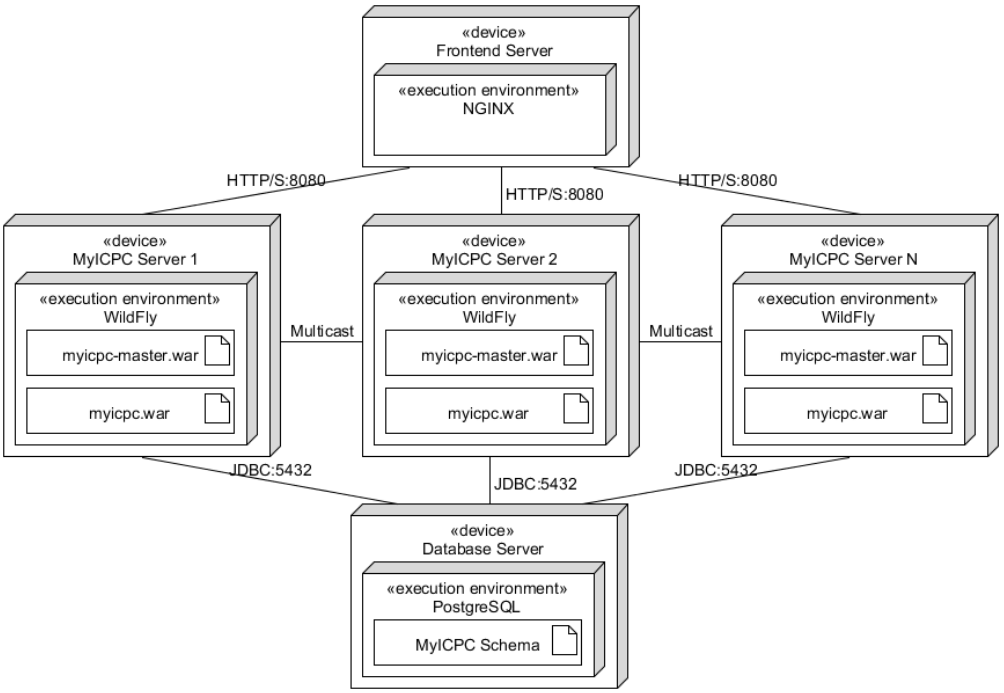


Figure 2.4: Deployment of MyICPC as a single node application.

Internet. Such device enhances user's experience and allows the user to perceive an event from various angles[15]. For example, a second-screen application can be a sport event application that provides current score board etc[4].

2.5.1 Timeline

Timeline is a home page of every contest in the MyICPC server application[4]. It helps to follow what is happening in the contest[4]. It consists of various notifications like the stream of social conversations, contest events or the announcement of a Quest challenge organized in a timeline[4]. The MyICPC timeline is shown in the figure 2.5 and its subset in form of the Quest timeline is shown in the figure 2.6.

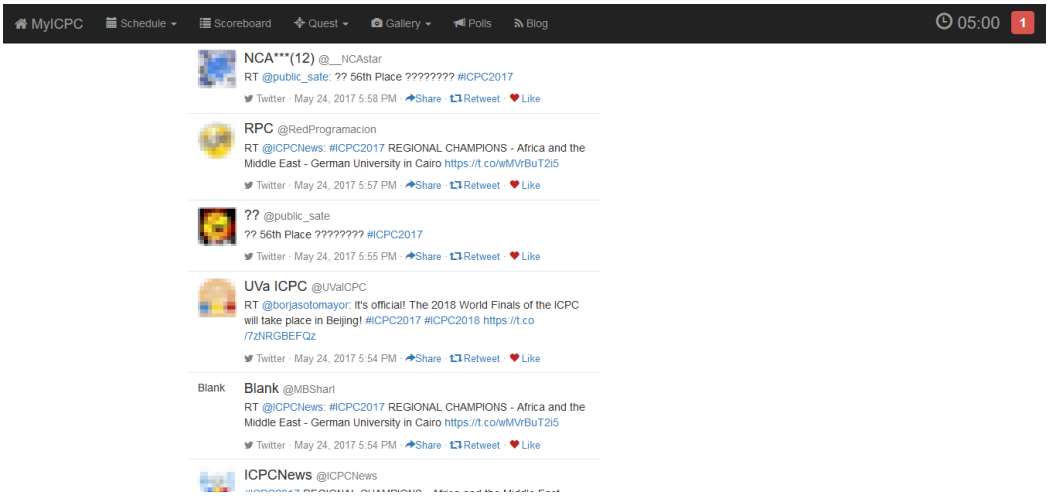


Figure 2.5: Example of the MyICPC Timeline in existing application.

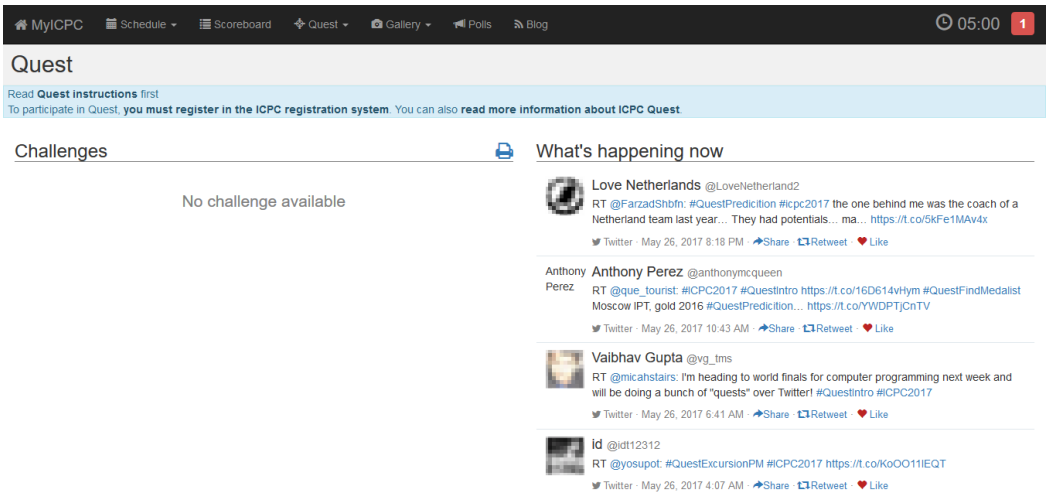


Figure 2.6: Example of the Quest Timeline in existing application.

2.5.2 Quest

MyICPC contains a game called Quest that consists of challenges and reward in form of points for every solved challenge[4]. The user can participate in the Quest game using social networks like Twitter[4]. The Quest timeline is shown in the figure 2.6 and an example of the challenge is shown in the figure 2.7.

2.6 Related work

Several second screens or second-screen applications exists nowadays[15]. They re used to enhance and to support various activities like augmenting TV screens by augmenting a tradition TV program, enhancing sport event experience with overlapping program or usage

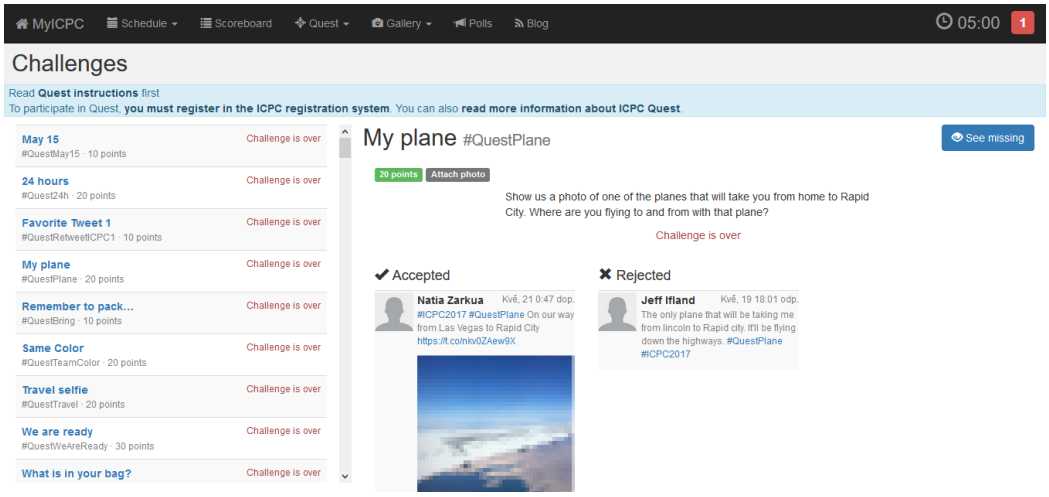


Figure 2.7: Example of challenges in the Quest game in existing application.

in museums and galleries to bring an interactive experience to the audience[15].

There are several libraries, framework or API that focus on the context acquisition. For example, Google Awareness API works with 7 signals including time, location, places, beacons, headphones, activity and weather[16]. These libraries, framework or API usually looks like puzzle pieces that do not fit well together[16]. This means that it is difficult to combine them and thus most of such libraries, framework or API focus on one context information. In case of context information about networks, there are some tools like network scanners but there is no tool that would use the context in term of the context awareness.

2.7 Requirements

The requirements are mainly based on the thesis assignment. The requirements for the acquisition of the contextual information are also based on the context described in this chapter and the research described in the section 1.1. The requirements are as follow:

- Create a mobile application that will acquire the context. The contextual information is a date, a location, a device information, connected network, available Wi-Fi networks, devices in a network, a signal strength, Bluetooth devices and a user identity.
- The context acquisition mainly focuses on network information as described in this chapter.
- The context acquisition should be created as a component that is easy to integrate to other applications as it will be used in a MyICPC mobile application.
- Create a mobile application for MyICPC that will display the MyICPC timeline and posts, challenges and leader boards of the Quest game. It will also allow the user to send messages to the Myicpc timeline and submissions to the Quest game.
- The mobile application will be created for Android version 6.

- The mobile application will integrate the context acquisition described in this chapter and it will send the acquired context with the messages and submissions.
- Create a REST API for communication of with the mobile application. Solve authentication and storage of the context and media files.
- Creating new components in MyICPC is preferable than modifying the existing component due to backward compatibly.

Chapter 3

Design

First, this chapter describes a design of the mobile application for Android devices that will acquire the context as well as take care of the MyICPC functionality. The mobile application will be designed with respect to Android design guidelines. This chapter describes which and how Android components like activities and services will be used in the mobile application as well as which screens the mobile application will use.

Second, this chapter explains changes of MyIPCP. The goal of the changes is to add new components rather than changing existing components. These changes consist of following things: an expansion of the domain model by adding the acquired context and an addition configuration of new components, a description of REST API endpoints that are required for the communication with the MyICPC mobile application and some small or cosmetic changes in MyICPC views.

3.1 Android design guidelines

The users of mobile devices with Android operating system expect applications to behave and look in a way that is consistent with the platform[13]. Google provides guidelines for application developers to design applications in a proper way. These guidelines consist of Material Design Guidelines and App Quality Guidelines[13]. Material Design Guidelines describes visual and navigational patterns and App Quality Guidelines contains information about compatibility, performance, security and more[13].

3.1.1 Context acquisition service

In Android applications, services are one of the essential application components that can perform long-running tasks in the background and services do not provide a user interface[13]. Services can be started by another application components[13]. This description fits the requirements for the context acquisition component.

The context acquisition service will be started at the start of the mobile application. The service will continuously perform the context acquisition in the background. On a request, the service will take a snapshot of the acquired context a send it to a component

that requested the context. The service will be stop when the mobile application will be terminated The service will acquire following information:

- **Date** - It is represented as a Java time stamp that gives the number of milliseconds since the epoch (midnight of January 1st 1970).
- **Location** - It consists of a latitude, a longitude and a time stamp representing a date when the location was obtained.
- **Information about user's device** - User's devices is identified by an operating system an its version, a brand and a modele of the device and a serial.
- **Connected Wi-Fi network** - It consists of SSID, BSSID, a state, RSSI, a link speed, a frequency and an IP address.
- **Available Wi-Fi networks** - It is a list of Wi-Fi networks. Each Wi-Fi network is represented by SSID, BSSID, capabilities, frequencies, a channel width, a level and a time stamp.
- **Devices in the network** - It is a list of network devices. Each device is represented by an IP address and a MAC address.
- **GSM signal strength** - It is a value in dBm units.
- **Bluetooth devices** - It is a list of Bluetooth devices. Each device is represented by a name and a MAC address.
- **User identity** - It consists of an ID, a first name, a last name, social media user names and a profile picture URL.

3.1.2 MyICPC activities

In Android applications, activities are one of the essential components[13]. The Android operating system initiates code in an instance of an activity by invoking specific callback methods that correspond to specific stages of its life cycle[13].

The mobile applications will consist of following activities: a sign in, a timeline, a post detail and a new post. A design of these activities is illustrated in the figure 3.1. The mobile application will also have activities that will display the Quest game posts, their details and allow the user to make a new submission. A design of these activities is more or less identical to a design of the timeline activity, the post detail activity and a new post activity respectively.

After application launch in the sign in activity, there is a sign in button that initializes a sign in processes typically by opening an external dialog, an activity or a browser. After the user signed in, the timeline activity is shown with a list of posts. The detail of a post including an image or a video if there is any is shown after clicking on it. The user will create a new post by cling on a floating action button with a plus sign and then by filling a message and optionally attaching a photo or a video. The user can also agree to provide the acquired context for the research purposes.

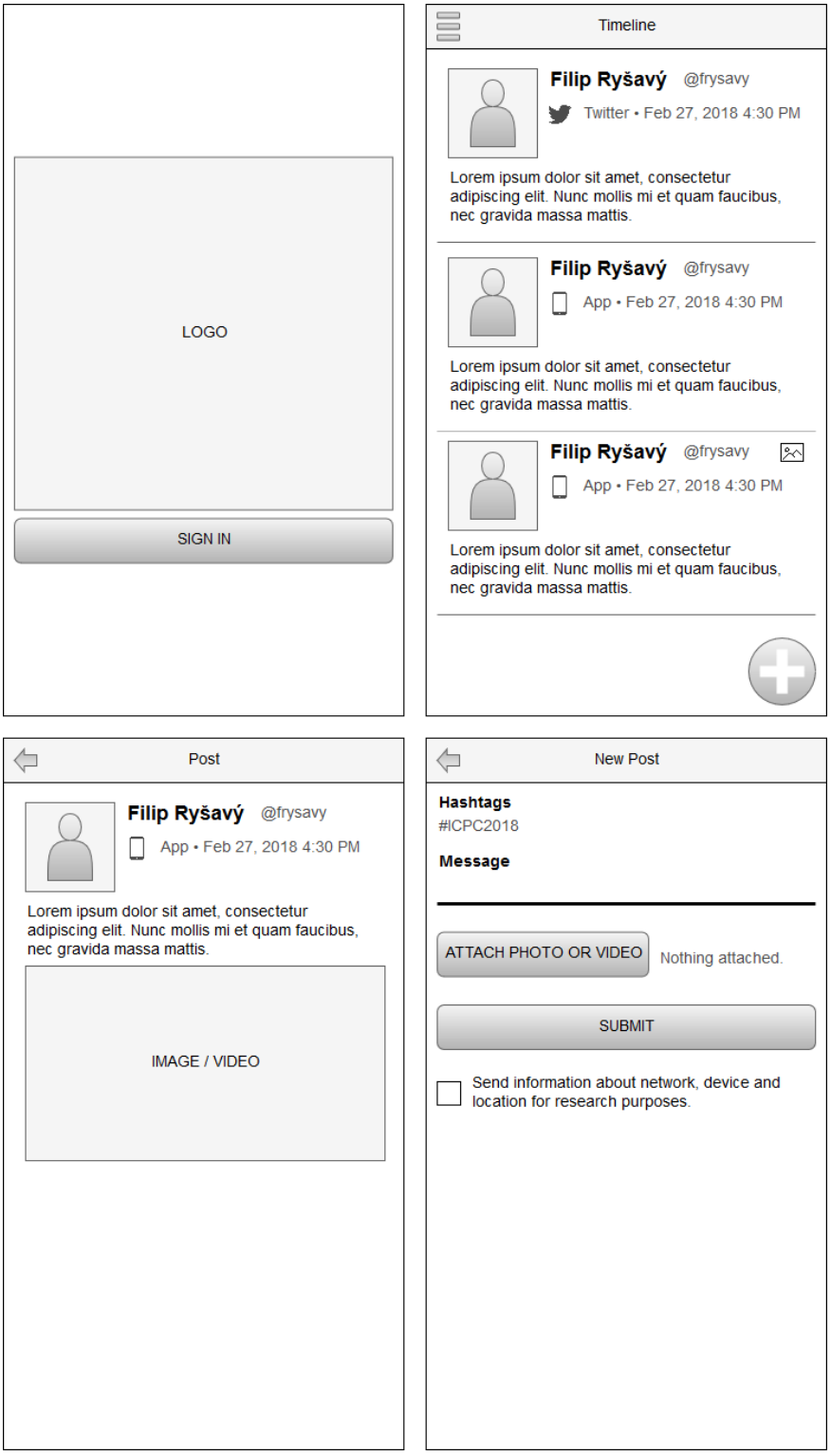


Figure 3.1: Design of activities: a sign in, a timeline, a post detail and a new post.

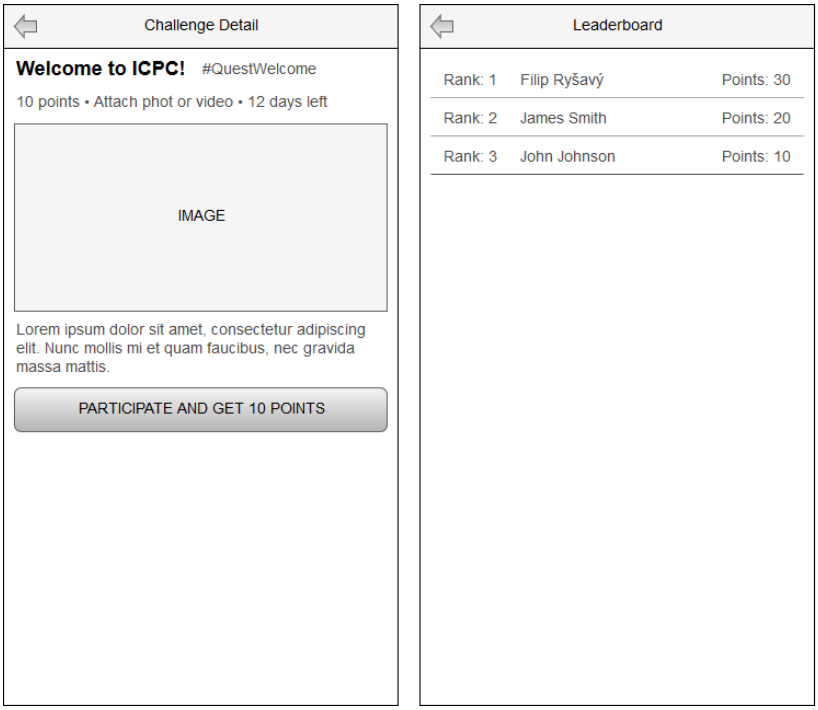


Figure 3.2: Design of activities: a challenge detail and a leader board.

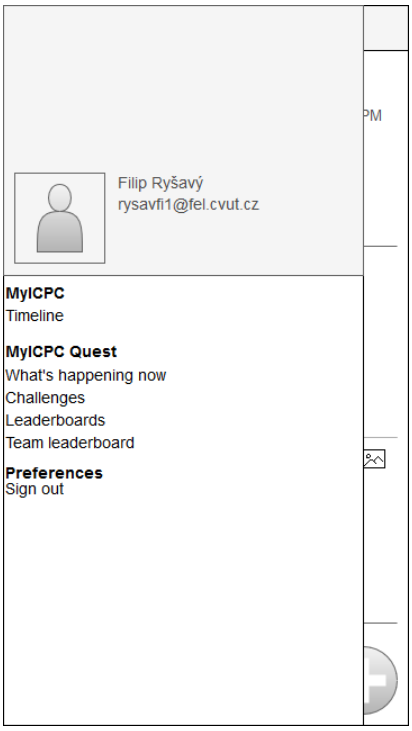


Figure 3.3: Design of a navigation drawer.

The mobile application will next have following activities: a list of challenges, a challenge detail activity, a list of leader boards, a leader board activity and a team leader board activity. A list of challenges and a list of leader boards are simple lists, a design of a team leader board is more or less identical to a design of leader board and a design of rest of these activities is illustrated in the figure 3.2.

The user can select a challenge from a list of challenges by clicking on it. Then the user is presented with a challenge detail containing a name and a hashtag of a challenge, a number of points, an information if a photo or a video is required, a remaining time, a challenge image and a challenge description. The user can participate in a challenge by clicking in a participate button that shows him a new submission activity that is more or less identical to a new post activity.

The user can choose a leader board from a list of leader boards by clicking on it. Then the user is presented with a leader board detail that contains a list of contestants each with a rank, a name and a number of points. The team leader board activity is more or less identical to a leader board detail except it displays a name of a team instead of a name of a contestant.

3.1.3 Navigation

The Android guidelines provide several patterns for a navigation. The mobile application will mainly use a pattern called a navigation drawer[13]. The navigation drawer is a user interface panel that show a main navigational menu of an application[13]. The navigation drawer is hidden when it is not used and it is displayed when the user swipes a finger from the left side of the screen or when the user clicks the drawer button in the application bar[13]. A design of the navigation drawer is illustrated in the figure 3.3. The mobile application will also use patterns called Up navigation[13] and Back navigation[13]. Up navigation refers to a navigation in the application hierarchy by clicking on the Up button in the application bar[13]. Back navigation refers to a navigation in the history of screens by clicking the Android back button[13]. Because the mobile application has relatively small number of activities, both of these navigation pattern appears to be the same.

3.2 Server application architecture

MyICPC does not have any REST API[4] so the first step is to add another layer to its architecture. The layer will contain all REST API endpoints for communication with the mobile application and data transfer objects. This layer will use the service and the persistence layer in a same fashion as the controller layer[4]. Because MyICPC is divided into module this can also be perceived as modifying the controller layer.

The whole idea is to remove usage of Twitter. Because of that an authentication and a storage was removed from the process as well. The second step is to provide new means of an authentication and a storage. The authentication will be provided by an authentication server that uses OpenID Connect, for example Keycloak, and Google Drive will be used as a storage. This deployment is shown in the figure 3.4.

This change preserves the ability of MyICPC to run as a single node application as well as a cluster. In a single node application with one instance of MyICPC, every execution

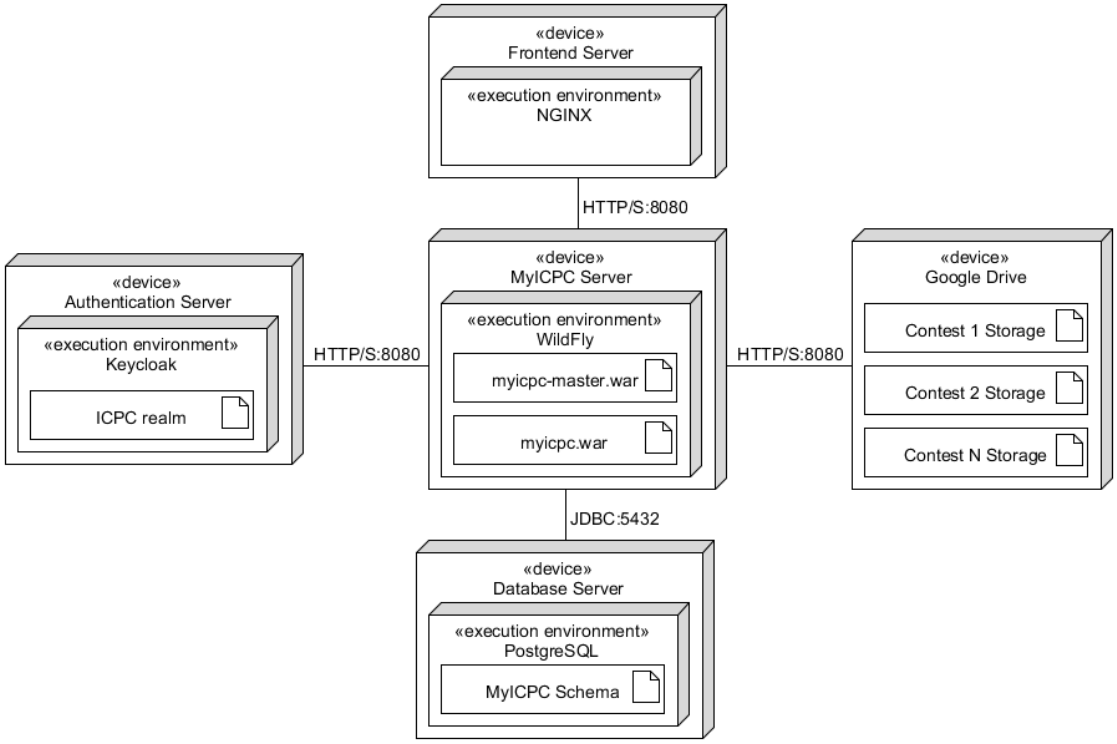


Figure 3.4: Deployment of MyICPC with REST API.

environment can run on a single device with exception of Google Drive or each execution environment can run on separate devices. In a cluster of MyICPC instances, a device running it can by horizontally scaled.

3.2.1 Domain model

This thesis does not modify the existing domain model[4]. This thesis only expands it with the acquired context and the additional configuration of MyICPC.

The acquired context consists of a date, a location, an information about user's device, a connected Wi-Fi network, available Wi-Fi networks, devices in the network, a GSM signal strength, Bluetooth devices and a user identity. This is shown in the figure 3.5.

The additional configuration of MyICPC consists of the configuration of REST API. The domain model consists of true or false value indicating if REST API is enabled in a contest, a Keycloak introspection endpoint, a Keycloak client ID, a Keycloak client secret, a Google APIs service account ID (e-mail), a Google APIs service account private key, a Google APIs service account private key ID, a Google APIs token server encoded URL, a Google APIs service account project ID, a Google Drive parent folder ID and a contest ID. This is shown in the figure 3.6.

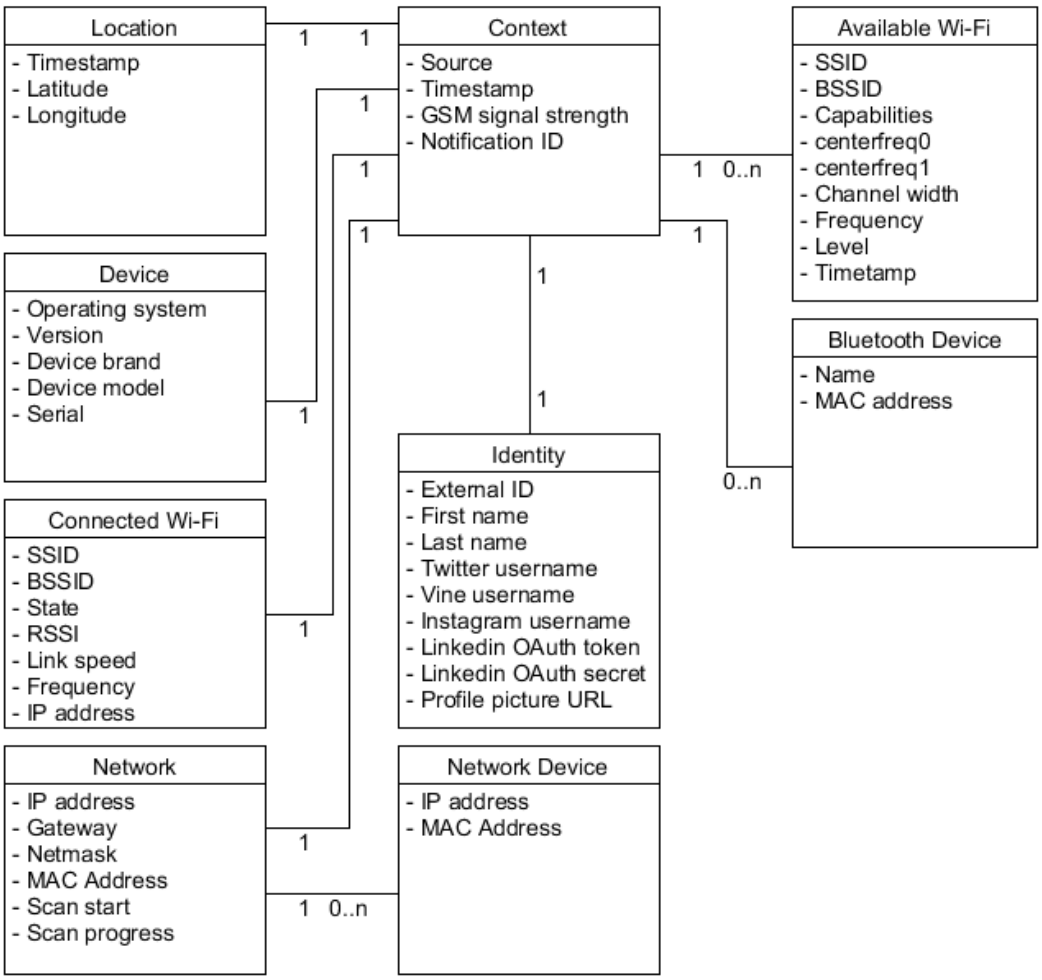


Figure 3.5: Domain model of the acquired context.

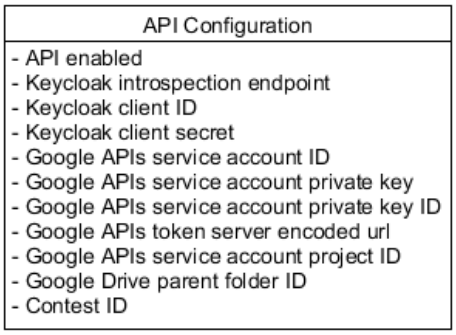


Figure 3.6: Domain model of the REST API configuration.

3.2.2 REST API endpoints

The endpoints are used for communication with the mobile application and they are described in the table 3.1. URL of all endpoints is prefixed with `api/{contestCode}` where `contestCode` is an identifier of a contest.

Endpoint	Method	Description
`user`	GET	Gets a user, status 200.
`timeline`	GET	Gets timeline posts, status 200.
`timeline`	POST	Posts a new post, status 201. Otherwise status 400.
`quest/whats-happening-now`	GET	Gets quest posts, status 200.
`quest/challenges`	GET	Gets challenges, status 200.
`quest/challenges/{hashtagSuffix}`	GET	Gets a challenge, status 200. A challenge is identified by `hashtagSuffix`.
`quest/leaderboards`	GET	Gets leader boards, status 200.
`quest/leaderboards/{urlCode}`	GET	Gets a leader board, status 200. A leader board is identified by `urlCode`.
`quest/team-leaderboard`	GET	Gets a team leader board, status 200.

Table 3.1: REST API endpoints developed for this thesis.

All http requests to the endpoints require Authorization header with "Bearer $accessToken". The endpoints return HTTP status code 401 if user isn't authenticated, 403 if user if authenticated but isn't in MyICPC, 404 if a contest is not found or REST API is disabled. Bodies of all request and responses are in a JSON format.

3.2.3 Changes in MyICPC

The MyICPC contains several notification types and none of them fits the message from the mobile application. Fist change is to add a new notification type called mobile app. This step also includes to modify views and services to work with this type usually by adding if statement.

Because this thesis extends the domain model with context information there needs to be added new JPA entities, repositories and services for the context.

Chapter 4

Implementation

This chapter first describes development environments and development tools. Then, this chapter describes an implementation of the context acquisition. It explains how individual context information are obtained in the Android environment using version 6 of Android. This chapter also describes an implementation of the MyICPC mobile application. It explains how the mobile application uses the OpenID Connect (OIDC) authentication with Keycloak and how it communicates with the MyICPC server application. Finally, this chapter describes an implementation of changes in the MyICPC server application like expanding the domain model, adding REST API, securing the application with Keycloak and uploading media files to Google Drive.

4.1 Development environments

This thesis uses several development environments. First, it uses Android Software Development Kit (SDK) to develop the context acquisition and the MyICPC mobile application. Then, it uses Java Platform, Standard Edition (Java SE) to expand the MyICPC server application. Finally, there are several small environments like an authentication server, an application server and a database.

4.1.1 Android SDK

The Android SDK[13] contains various tools that are used to develop mobile applications on the Android platform. It contains components like SDK tools, SDK build tools and SDK platform tools. SDK tools include development and debugging tools for Android. SDK build tools are required for building Android applications. SDK platform tools include tools that interface with the Android platform and that are required for Android application development. The Android SDK is used through the Java programming language to develop Android applications.

4.1.2 Java Platform, Standard Edition

Java[17] is both a programming language and a platform. The Java programming language is a high-level and object-oriented language. Java SE is one of the Java programming

language platform. The other platform is for example Java Platform, Enterprise Edition. Each platform consists of a Java Virtual Machine (JVM) and an API. JVM is a program that runs Java applications. An API is a set of components that you can use to develop Java applications. Java SE's API provides the core functionality of the Java programming language. The most known implementation of Java SE are Oracles's Java Development Kit (JDK) and OpenJDK.

4.1.3 Keycloak

Keycloak[18] is an open source Identity and Access Management solution. Keycloak uses Single Sign On (SSO). SSO mean that the user authenticates with Keycloak rather than individual applications and that the application do not have to deal with any login forms, authenticating users and storing them because it is all provided by Keycloak. Keycloak supports standard protocols like OIDC, OAuth 2.0 and Security Assertion Markup Language (SAML) 2.0 and various other features.

4.1.4 WildFly

WildFly[19] is a lightweight, flexible and managed application server that runs applications written in the Java programming language. WildFly implements the latest Java standards. WildFly can be run in two different modes called standalone mode and domain mode. The standalone mode allows the application server to run in single JVM. The domain mode allows the application server to run in multiple JVM, in other words in a cluster.

4.1.5 PostgreSQL

PostgreSQL[20] is an open source, object-relational database, structured query language (SQL) database. PostgreSQL has a proven architecture, a reliability, a data integrity, a robust feature set and an extensibility. PostgreSQL successfully conforms most of the mandatory features of the SQL standard.

4.2 Development tools

Development tools are used by software developers to implement, test and debug their software solutions that they are developing. The developers can develop software without the development tools, but the process would consume significantly more time to achieve the same results as with usage of the development tools. Development tools consist of an integrated development environment (IDE), a build tool and a version control system. In this thesis, there were used different development tools for the mobile applications and the server application.

4.2.1 Integrated development environments

Android Studio[13] was used to develop the context acquisition and the MyICPC mobile application on the Android platform. Android Studio is the official IDE for developing

Android applications provided by Google. Android Studio is based on IntelliJ IDEA[21]. IntelliJ IDEA is the commercial IDE for developing applications in the Java programming language developed by the company called JetBrains s.r.o. IntelliJ IDEA itself was used to develop the MyICPC server application.

4.2.2 Apache Maven

Apache Maven[22] is a build tool as well as a software project management tool. Apache Maven uses configuration file called pom.xml to describe project properties, dependencies and build information. Apache Maven uses the pom.xml file to download these dependencies from a central repository and to build project based on its properties. This thesis use it to build the MyICPC server application.

4.2.3 Version control system

Version control systems are used by the developers to store changes of the source code over time. Version control systems allows the developers to recall a specific version later. Version control systems are not limited by the text information only and other types like graphics or multimedia can be stored as a version as well. This thesis uses version control system called Git and its web based repository called BitBucket.

4.3 Third-party libraries

This thesis uses several third-party libraries in both the MyICPC mobile application and the MyICPC server application. These libraries are developed by some third party and are used for a specific function like OIDC authentication, bootstrapping Java application and others.

4.3.1 AppAuth for Android

AppAuth for Android[23] is a library (SDK in Android terminology) for communication with OAuth 2.0 and OIDC provides. AppAuth for Android parses the communication with authentication server into requests and responses and it also provides custom tabs to display the login form. The MyICPC mobile application uses AppAuth for Android to communicate and authenticate the user with Keycloak.

4.3.2 Spring framework

The Spring framework[24, 25] is used to build the enterprise or web applications in the Java programming language. The Spring framework is used to bootstrap the MyICPC server application.

The Spring core provides Inversion of Control container that makes the initialization of classes easier by injecting correct dependencies when beans are created. The Spring framework then consists of Spring Data module that is used to manage JPA entities and access to the database. Next, the Spring framework contains of Spring MVC which contains

controllers for REST API and presentation layer. Finally, the Spring framework contains a security module.

4.3.3 Others

The MyICPC mobile application also uses following third-party libraries to perform some small tasks: Gson, Jsoup and Apache Commons. Gson is a Java library that is used to convert JavaScript Object Notation (JSON) in HTTP requests and responses into Java objects and vice versa. Jsoup is a Java library that is used to display HTML body of MyICPC notifications as a plain text. From Apache Commons, there is used its Net module to obtain network information like a range of IP address.

4.4 Context acquisition implementation

The context acquisition is implemented using Android on version 6. This section explains an implementation of following contextual information on the Android platform: a date, a location, an information about user's device, a connected Wi-Fi network, available Wi-Fi networks, devices in the network, a GSM signal strength, Bluetooth devices and user's identity.

The overall context acquisition is implemented as Android service by extending the `Service` class. This service in its callback method uses components for collection individual context information.

4.4.1 Date

Android applications are built in Java and thus the Android SDK contains JDK. The easiest way to obtain a current date is to create a new instance of the `java.util.Date` object. This object is by default created with current date.

4.4.2 Location

The Android SDK contains two methods how the application can obtain a location. These methods are Location Services from the `android.location` package and the Google Play services location API. Note that both methods require permissions to access a location of the device.

Location Services from the `android.location` package is an older method. The main component of this method is the `LocationManager`. This class is a system service and it provides the API to determine a location. It supports following location providers: GPS provider, network provide and passive provider that obtains location from other application. This method is used via registering a listener.

Google Play services location API is a new method of obtaining a location. It basically offers the same functionality of Location Services from the `android.location` package with some minor implementation and naming changes. However, it is dependent on Google Play services and its configuration.

The context acquisition application uses Location Services from the `android.location` package because it fulfills requirement on the location acquisition and does not have any dependencies. The context acquisition application implements `LocationListener` and registers it in `LocationManager`. It uses all three providers to obtain the location. The listener only updated the location if the currently stored location is older than two minutes or if the new location is more accurate. This implementation is shown in the listing 4.1.

```
public class LocationResolver implements LocationListener {
  private Location location;

  /* ... */

  @Override
  public void onLocationChanged(Location location) {
    if (location != null) {
      if (this.location == null) {
        this.location = location;
        return;
      }
      if (Math.abs(location.getTime() - this.location.getTime())
          <= 120000) {
        if (location.getAccuracy() > this.location.getAccuracy()) {
          this.location = location;
        }
      } else {
        this.location = location;
      }
    }
  }

  /* ... */
}
```

Listing 4.1: Implementation of the date acquisition.

4.4.3 Information about device

In the Androdi SDK, there is the `android.os.Build` class that contains information about the device and the current build that are obtained from system properties. The context acquisition application collects following information: an operating system, a version of the operating system, a brand of the devices, a model of the device and a serial number of the device.

4.4.4 Connected Wi-Fi network

The Android SDK contains a class called `WifiManager`. This class can be used to obtain the list of configured networks, the currently active Wi-Fi network, results of access point scans, etc. The context acquisition application uses this class to obtain information about

the connected Wi-Fi network. Note that this and all following network related information require permissions to access network and Wi-Fi.

The context acquisition application uses following information: the basic service set identifier (BSSID) of the current access point, a fine-grained network connectivity state, the current frequency, the IP address, the current link speed, the received signal strength indicator (RSSI) of the current 802.11 network in dBm and the service set identifier (SSID) of the current 802.11 network.

4.4.5 Available Wi-Fi networks

As mentioned in the previous section, the `WifiManager` class can be used to get results of access point scans. In other worlds, it can be used to get available Wi-Fi networks. This can be done by starting scan using the `WifiManager` class and registering Android component called a broadcast receiver to receive and process the results.

4.4.6 Devices in network

The Android SDK itself doesn't contain any method to obtain a list of network devices. However, Java itself has a few ways how to obtain a list of network devices. The options are either to use ping command or to use a Java class called `InetAddress`.

Because Android is an operating system based on Linux, it contains a ping command in its core. The ping command uses the Internet Control Message Protocol (ICMP) protocol's mandatory echo request datagram to elicit an ICMP echo response from a host or gateway. Next, Java is able to get the operating system run-time and execute commands like ping in it a get command result. This process is shown in the listing 4.2.

```
Runtime runtime = Runtime.getRuntime();
String command = String.format("/system/bin/ping -q -n -w 1 -c 1 %s",
  ipAddress);
Process process = runtime.exec(command);
int exitValue = process.waitFor();
process.destroy();
```

Listing 4.2: Checking if IP address is reachable using ping command.

Java itself can check if an IP address is reachable without using the ping command. Java contains a class called `InetAddress` that represents an IP address. This class has a method to check if the IP address that is represented by the object is reachable. This method also requires a timeout in milliseconds. This process is shown in the listing 4.3.

The context acquisition application first uses the `WifiManager` class to get the network information and estimate available IP addresses range. Then, it tries to use the ping command method as it is more reliable than the pure Java method. However, the ping command might not be available on all Android devices. In such case, the context acquisition application will use the pure Java method.

```
try {
  InetAddress inetAddress = InetAddress.getByName(ipAddress);
  if (inetAddress.isReachable(1000)) {
    /* IP address is reachable */
  } else {
    /* IP address isn't reachable */
  }
} catch (Exception e) {
  e.printStackTrace();
}
```

Listing 4.3: Checking if IP address is reachable using pure Java.

The final stage of this process is to obtain a hardware address also known as a MAC address. The context acquisition application is using the fact that Android is based on Linux to obtain a hardware address. It is using program called Arp or more precisely, it uses its cache directly. Arp manipulates the kernel's ARP cache in various ways. The primary options are clearing an address mapping entry and manually setting up one. This process is shown in the listing 4.4.

```
public static String getHardwareAddress(String ipAddress) {
  String result = "00:00:00:00:00:00";
  try {
    String regex = String.format(
      "^%s\\s+0x1\\s+0x2\\s+([:0-9a-fA-F]+)\\s+\\*\\s+\\w+$",
      ipAddress.replace(".", "\\.")
    );
    Pattern pattern = Pattern.compile(regex);
    BufferedReader br = new BufferedReader(
      new FileReader("/proc/net/arp")
    );
    String line;
    while ((line = br.readLine()) != null) {
      Matcher matcher = pattern.matcher(line);
      if (matcher.matches()) {
        result = matcher.group(1);
        break;
      }
    }
    br.close();
  } catch (IOException e) {
    e.printStackTrace();
  }
  return result;
}
```

Listing 4.4: Obtaining of the hardware address.

Note that each of these methods requires one second to execute and it needs to be executed for each potential IP address. Also, note that network operations like ping cannot be run on the main thread and they need to be executed on a background thread for example by using an Android component called `AsyncTask`.

4.4.7 GSM signal strength

The process of obtaining the GSM signal strength is similar to obtaining the location. The Android SDK contains a class representing a system service called `TelephonyManager`. A listener is registered to this service. The listener is used to update the GSM signal strength when it is changed. The obtained value needs to be converted to dBm by: $(2 * mSignalStrength) - 113$.

4.4.8 Bluetooth devices

The Android SDK contains a class `BluetoothAdapter` that can be used to start a discovery for Bluetooth devices. To get results the context acquisition must register a broadcast receiver to receive and process the results.

4.4.9 User identity

The context acquisition application uses as the user identity the information about the user from the MyICPC server application. These pieces of information are obtain from REST API developed in this thesis. These pieces of information consist of an identifier from ICPC systems, a fist name, a last name, social media user names and a profile picture uniform resource locator (URL).

4.5 Android application implementation

The MyICPC mobile application is implemented using Android on version 6. This section describes an implementation of the application, how the application integrates the context acquisition, how it uses OIDC authentication with Keycloak thought AppAuth library and how it communicate with REST API of the MyICPC server application.

All Android applications are built on the concept of activities. An activity is implemented by extending the `Activity` class from Android SDK and implementing its callback method. The MyICPC mobile application implements following activities: the login activity, the main activity containing the navigation drawer, the post detail activity, the new post activity, the challenge detail activity, the new quest submission activity and the leader board detail activity. Finally, each activity is registered in the application manifest file.

Every screen in the MyICPC mobile application is not implemented as an Android activity, but some screens are implemented as a fragment. A fragment is another Android component. It is a view component that can be embedded into an activity. The following screens are implemented as a fragment embedded into the main activity: the list of challenges, the list of leader boards, the team leader board detail, the MyICPC timeline and

the Quest game timeline. The navigation among the fragments is done using the navigation drawer.

4.5.1 Integration of context acquisition

The context acquisition application is implemented as a simple Android service. The easiest way to integrate it with the MyICPC mobile application is to add the service to the MyICPC mobile application and register it in the application manifest file. The service is then started and ended when the MyICPC mobile application is started and ended respectively.

4.5.2 OpenID Connect authentication with Keycloak

The OIDC authentication with Keycloak is realized through the AppAuth for Android library. The AppAuth for Android library parses authorization requests and responses according to the OIDC standard as well as it hadles a token refreshing. The library provides its functionality through a service. This service needs to be provided with information about Keycloak server like an authorization endpoint, a token endpoint, a client ID and a redirect URL.

The authentication server where Keycloak is deployed might use Hypertext Transfer Protocol Secure (HTTPS). HTTPS is an extension of the Hypertext Transfer Protocol (HTTP) where the communication protocol is encrypted by Transport Layer Security or by Secure Sockets Layer. This is the case with production deployment of the authentication server where Keycloak runs and thus the application requires to know the certificate from the server where Keycloak is deployed.

In a traditional operating system like Windows or Linux, a certificate can be added to Java Run-time Environment (JRE) keystore directly by using the keytool command. However, there is no way how to access JRE keystore in the Android operating system. This would also require that each user would do this method on his or her device. Such approach is completely unusable for the Android environment.

The Adroid SDK or more precisely Java that a part of the Android SDK allows the application to add certificated dynamically to the keystore at the run-time. This process is shown in the listing 4.5.

4.5.3 Communication with MyICPC

The Android SDK uses the `HttpURLConnection` class from the Java programming language to allow Android applications to communicate over HTTP. The MyICPC mobile application implements communication with all endpoint of the MyICPC server application using this class.

Note that this operation requires Internet permissions and if the MyICPC server application is deployed on a server that uses HTTPS, it also requires to add the certificate to the keystore as described in the previous section.

```
private static SSLSocketFactory getSslSocketFactory(Context context) {
  SSLSocketFactory sslSocketFactory = null;
  try {
    CertificateFactory certificateFactory = CertificateFactory
      .getInstance("X.509");
    InputStream inputStream = context.getResources()
      .openRawResource(R.raw.cmplayecsbayloredu);
    Certificate certificate = certificateFactory
      .generateCertificate(inputStream);
    inputStream.close();

    String type = KeyStore.getDefaultType();
    KeyStore keyStore = KeyStore.getInstance(type);
    keyStore.load(null, null);
    keyStore.setCertificateEntry("cmplay", certificate);

    String algorithm = TrustManagerFactory.getDefaultAlgorithm();
    TrustManagerFactory trustManagerFactory = TrustManagerFactory
      .getInstance(algorithm);
    trustManagerFactory.init(keyStore);

    SSLContext sslContext = SSLContext.getInstance("TLS");
    sslContext.init(null, trustManagerFactory.getTrustManagers(),
      null);
    sslSocketFactory = sslContext.getSocketFactory();
  } catch (CertificateException | IOException | KeyStoreException
      | NoSuchAlgorithmException | KeyManagementException e) {
    e.printStackTrace();
  }
  return sslSocketFactory;
}
```

Listing 4.5: Dynamically adding the certificate to keystore.

4.6 Server applications implementation

The MyICPC server application is already build using the Spring Framework and the Java programming language. The MyICPC server application is extended with expanding the domain model, adding REST API, securing the application with Keycloak and uploading media files to Google Drive.

The domain model is extended with the acquired context and the additional configuration of the REST API. It is implemented by JPA entities. For each JPA entity there is a repository generated by the Spring Framework and a service to work with it.

4.6.1 Endpoints

The MyICPC server application implements endpoints according to the table 3.1 using rest services from the Spring framework. The following endpoints were implemented: the user

endpoint, the MyICPC timeline endpoint, the Quest game timeline endpoint, the challenges endpoint, the leader boards endpoint and the team leader board endpoint.

4.6.2 Securing application with Keycloak

REST API that was written only by using pure Java can be secured by implementing a HTTP filter that would filter the HTTP requests for example based on the Authorization header. Even though the Spring framework is for Java application those filter does not work well with the life cycle of the Spring application.

The Spring framework contains a security module. However, the security module is already used in the current version of the MyICPC server application. This situation makes is usage difficult because the already existing security rules does not work with the new security rules well.

Keycloak provides several ways to integrate it with applications. First, Keycloak provides several adapters to use with Java applications, the WildFly server or the Spring framework. The only suitable adapter is the Spring framework adapter, because the production version of the WildFly server can not be modified. However, it does not work well with already existing security rules as mentioned above.

```
public static Principal getPrincipal(String accessToken,
    ApiSettings apiSettings) {
  try {
    String credStr = String.format("%s:%s", apiSettings.getClientId(),
      apiSettings.getClientSecret());
    byte[] credBytes = credStr.getBytes();
    byte[] encCredBytes = Base64.encode(credBytes);
    String encCredStr = new String(encCredBytes);
    String basicAuthStr = String.format("Basic %s", encCredStr);

    HttpHeaders headers = new HttpHeaders();
    headers.set("Content-Type", "application/x-www-form-urlencoded");
    headers.set("Authorization", basicAuthStr);

    String body = String.format("token=%s", accessToken);

    RestTemplate restTemplate = new RestTemplate();
    HttpEntity<String> reqEntity = new HttpEntity<>(body, headers);
    ResponseEntity<Principal> resEntity = restTemplate.exchange(
      apiSettings.getIntrospectionEndpoint(), HttpMethod.POST,
      reqEntity, Principal.class
    );
    return resEntity.getBody();
  } catch (Exception e) {
    return null;
  }
}
```

Listing 4.6: Obtaining user's principal from Keycloak introspection endpoint.

```
public class AuthInterceptor extends HandlerInterceptorAdapter {
  /* ... */
  @Override
  public boolean preHandle(HttpServletRequest req,
      HttpServletResponse res, Object handler) {
    String bTokenStr = req.getHeader("Authorization");
    if (bTokenStr != null) {
      String[] bTokenArr = bTokenStr.split("\\s+");
      if (bTokenArr.length == 2 && bTokenArr[0].equals("Bearer")) {
        Contest contest = getContest(req.getRequestURI());
        if (contest == null) {
          // Unknown contest, return 404 - Not Found
          res.setStatus(HttpStatus.NOT_FOUND.value());
          return false;
        }

        ApiSettings as = asr.findByContest(contest);
        if (as == null || !as.isEnabled()) {
          // API is disabled, return 404 - Not Found
          res.setStatus(HttpStatus.NOT_FOUND.value());
          return false;
        }

        Principal principal = Principal
          .getPrincipal(bTokenArr[1], as);
        if (principal != null && principal.isActive()) {
          ContestParticipant cp = cpr
            .findByContestAndExternalId(contest, principal.getUid());
          if (cp != null) {
            // Valid token and user is in contest
            return true;
          } else {
            // Valid token, but user isn't in contest,
            // return 403 - Forbidden
            res.setStatus(HttpStatus.FORBIDDEN.value());
            return false;
          }
        }
      }
    }
    // Invalid or expired token, return 401 - Unauthorized
    res.setStatus(HttpStatus.UNAUTHORIZED.value());
    return false;
  }
  /* ... */
}
```

Listing 4.7: Implementation of the REST API security.

Another option that Keycloak provides is to use its REST API directly. It provides endpoints according to the OIDC specification like the token endpoint, the authorization endpoint end the introspection endpoint. The introspection endpoint is used to validate a

client tokens from example obtained from the Authorization header of the HTTP request.

The security of the REST API in the MyICPC server application is implemented in a following way. First, an interceptor is created to filter HTTP request by the token in the Authorization header. An interceptor is a Spring framework component that works similarly as HTTP filter except it operates within the life cycle of a Spring framework application. The interceptor expects the Authorization header to contain bearer token. The interceptor validates it against Keycloak and tries to get principal from Keycloak. The user request is accepted or denied based on the result from Keycloak introspection endpoint. This process is shown in the listing 4.7.

A principal is an information about an entity that is authenticated by Keycloak. This information is obtained from Keycloak introspection endpoint in exchange for a valid token. The endpoint is called using a REST client build in the Spring framework and the request follows the OIDC specification. This process is shown in the listing 4.6.

4.6.3 Uploading images and videos to Google Drive

Google provides a Java library that allows the application to upload files to Google Drive service. First, the Google service account needs to be set up in Google APIs. The Google service account contains credentials like a service account ID (e-mail), a service account private key, a service account private key ID, a token server encoded URL and a service account project ID. An ID of a Goodle Drive folder where the files will be stored is also required. Then, an image or a video is uploaded by using the library and an ID is returned that is stored in the database.

Chapter 5

Testing

In this chapter there will be presented how the MyICPC server application and the MyICPC server application were tested. The functionality of the persistence layer and the service layer was tested using unit tests. The overall MyICPC server application was also stress tested using tool called JMeter. A small usability test of the MyICPC mobile application was performed using a method called cognitive walkthrough.

5.1 Unit testing

The unit testing aims to split an application into small pieces (units) and test them if they behave correctly according to a specification. This thesis uses a library called JUnit to test the persistence layer and the service layer of the MyICPC server application. In those layers, every repository or service already fits a description of a unit and thus a test was written for each unit.

5.1.1 JUnit

JUnit[26] is a Java library for unit testing. It consists of a platform serving as a foundation for launching testing and a test engine. It supports annotations, pasteurized tests, an exception testing and a timeout support.

5.2 Stress testing

Stress testing checks upper limits of applications under heavy loads. Such testing allows to find potential bottleneck in applications. The MyICPC server application has the biggest load during a contest event. The MyICPC server application is stress tested using a tool called JMeter.

5.2.1 JMeter

JMeter[27] is an open source tool designed to load test functional behavior and measure performance. It supports several applications and protocols, importantly HTTP, HTTPS and others.

5.3 Usability testing

Usability testing is a technique verifying that a product is usable. There are qualitative and quantitative methods and a testing can be done with or without users. This thesis perform a small usability test in a form of a cognitive walkthrough.

5.3.1 Cognitive walkthrough

Cognitive walkthrough is a qualitative method of usability testing without user. Cognitive walkthrough is used to test a user interface with exactly given scenarios. It tests if the user is able to complete a scenario and how the user deviates from it. At the beginning a task-definition question is asked:

- What does the user want to achieve?

After a task-definition question there are three following questions that are asked in each following step of the scenario:

- Will the correct actions be evident to the users?
- Will the users connect the label of an action with their goals?
- Will the user receive a sensible feedback?

A testing without user requires good knowledge of a user and it is suitable only for non exotic user interfaces. A testing without user is cheap and easy to set up, however, it does not provide an observation of real users. This thesis uses a testing without user because the application is based on usual concept of a social media application and the method is fast and cheap. I, the creator of this thesis, also participate in events where the application is used and thus I have a very good knowledge of actual user.

5.3.2 User description

It is required that all users use the Android operating system. The main group of users consists of university students with computer science specialization. They are expected to have at least a basic knowledge of using mobile phones and social media applications. They might have an experience with the competition from previous years. Next user group forms coaches who are usually teacher of the students. Last users are members of ICPC.

5.3.3 Tested use cases

The cognitive walkthrough is performed with following use cases of the MyICPC mobile application:

- **Sign in** - The user will first start the application on his or her mobile phone. The user will click on the sign in button that shows the user a form. The user will fill in the form and confirms it. The user will end up on the timeline screen.

- **Post submission** - The user start after the sign in. The user will click a button with plus symbol. The user will fill in a form with a given post and submits it. The user will end up on the timeline screen.
- **Challenge submission** - The user start after the sign in. The user will use the navigation drawer to navigate to the list of challenges. The user will select a given challenge and make a submission. The user will end up on the timeline screen.
- **Displaying the leader board** - The user start after the sign in. The user will use the navigation drawer to navigate to the list of leader boards. Then, the user will select a give leader board with his given role. The user will end up by finding himself or herself in the leader board.

Chapter 6

Evaluation

This chapter describes how the requirements were fulfilled and what issues were found during the evaluation of the context acquisition, the MyICPC mobile application and the MyICPC sever application. Then, this section presents results of the unit testing, the stress testing and the usability testing. Finally, this section present resulting application and describes their deployment.

6.1 Requirement fulfillment

First, the context acquisition application was created using Android 6 as a standalone application as well as a component that was later integrated with the MyICPC mobile application. The context acquisition application collects a date, a location, a device information, connected network, available Wi-Fi networks, devices in a network, a signal strength, Bluetooth devices and a user identity. Then, the MyICPC mobile application was created using Android 6. It provided the functionality of the MyICPC timeline and the Quest game. Finally, the REST API was added into the MyICPC server application to allow the communication with the mobile application.

6.1.1 Heavy load of network device discovery

First issue that was found during evaluation is a heavy load of the network device discovery. Every potential IP address must be checked if there is a device. Every potential IP address is pinged with minimal timeout of one second. If a network consists of large amount of IP addresses this process can take several minutes. A solution of this problem might be a parallelization.

Android supports multi-threaded applications. However, even if the context acquisition is run in several threads, the load is still great and thus Android operating system will suspend even the thread that handles a user interface rendering. This results in lags in the user interface. It solution is thus suitable only for the standalone context acquisition application. Note, that there is a great amount of various Android devices with different processor units and each of them is able to run different amount of threads.

6.1.2 Quality of uploaded video

Second issue that was found during evaluation is a quality or a length of uploaded videos. The current design of the REST API expect JSON messages that contain serialized photo or video. This method allows to send a media file of a size approximately 5MB. This is suitable for sending photos, but in a case of sending videos it results in sending a low quality video of sufficient length or a high quality video of extremely short length.

Android in its core allows to capture a low quality video or a video of a maximal quality that a device can capture. One solution is to tweak this video control to get a video of reasonable quality. Second solution is to redesign the REST API to receive media files as multipart message and thus allowing it to receive larger files.

6.1.3 Downloading media files from storage

Last issue that was found during evaluation is a downloading media files from the Google Drive storage. The Google Drive is inaccessible from China due to the Great Firewall of China. An upload of files works fine because all files are first uploaded to the MyICPC server application that is not blocked by the Great Firewall of China and then the files are sent to Google Drive from MyICPC. MyICPC serves as a proxy server in this case. However, the files are downloaded directly from Google Drive.

First solution to this problem might be to set up a proxy server for downloading files that would work in similar way as MyICPC works during file uploading. Second solution is to acquire different storage. This would require small modification of the code. There is a Java interface that is implemented by the class that handles an upload to Google Drive. A new class needs to be added that implements the interface in a similar way for a new storage in this case.

6.2 Testing results

The testing of both the server application and the mobile application described in previous chapter was performed and this chapter summarizes the results of the unit testing, the stress testing and the usability testing.

6.2.1 Unit testing results

The unit tests for the persistence layer and the service layer of the MyICPC server application were performed automatically before the compilation. All tests have successfully passed and the application was successfully build.

6.2.2 Stress testing results

The stress testing was performed using JMeter as described in the previous chapter. The testing was done using MyICPC running on a single server. There were 1000 HTTP requests send to MyICPC with maximal response time approximately 4.4 seconds under heavy load. The production deployment is usually run with multiple servers and a load balancer and response time in such environment should be smaller under heavy load.

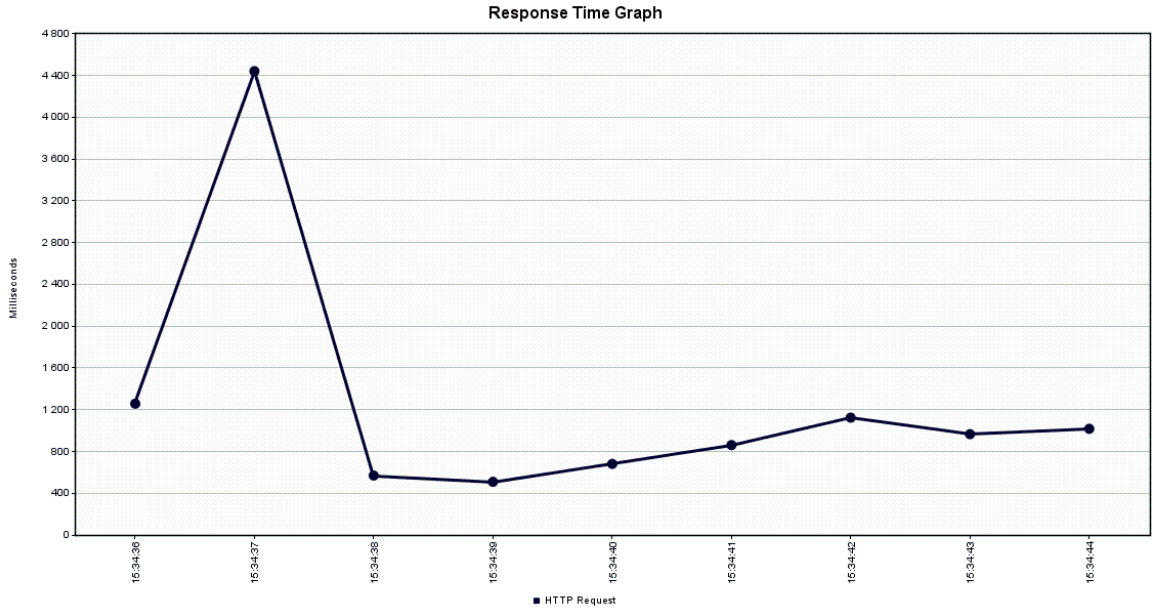


Figure 6.1: Response time graph.

6.2.3 Usability testing results

The usability test in form of cognitive walkthrough of the MyICPC mobile application was performed for all use cases according to the previous chapter. The table 6.1 contains all discovered issues. Each issue has a description, a priority and a proposed fix if such exists. Priorities are high for issues that break the functionally, medium for issues that complicate the process or make it difficult, low for small and cosmetic issues.

Description	**Priority**	**Proposed solution**
Challenge submission is accessible by too many clicks.	Medium	Merge challenge detail and submission.
Almost duplicate screens for two types of submission.	Medium	Merge timeline and quest submissions.
Leader board details are too small	Low	Redesign leader boards.

Table 6.1: Results of the usability testing.

6.3 Deployment

The new version of MyICPC and the MyICPC mobile application were supposed to be deployed during this year World Finals. However, due to technical issues caused by the Great Firewall of China, it was rescheduled to the regional competitions later this year. Thus, the deployment was only in testing environments.

6.3.1 Collected context

The context acquisition application obtains the context during every submission. The context is then stored in the database for later usage in the further research of the context-aware authentication using Wi-Fi enabled IoT devices. An example of the acquired context is shown in the appendix C.

6.3.2 Android application

The mobile application is replacing need for using Twitter in order to participate in the Quest game or simple to share contestant's experience. The resulting application is shown in the figure 6.2.

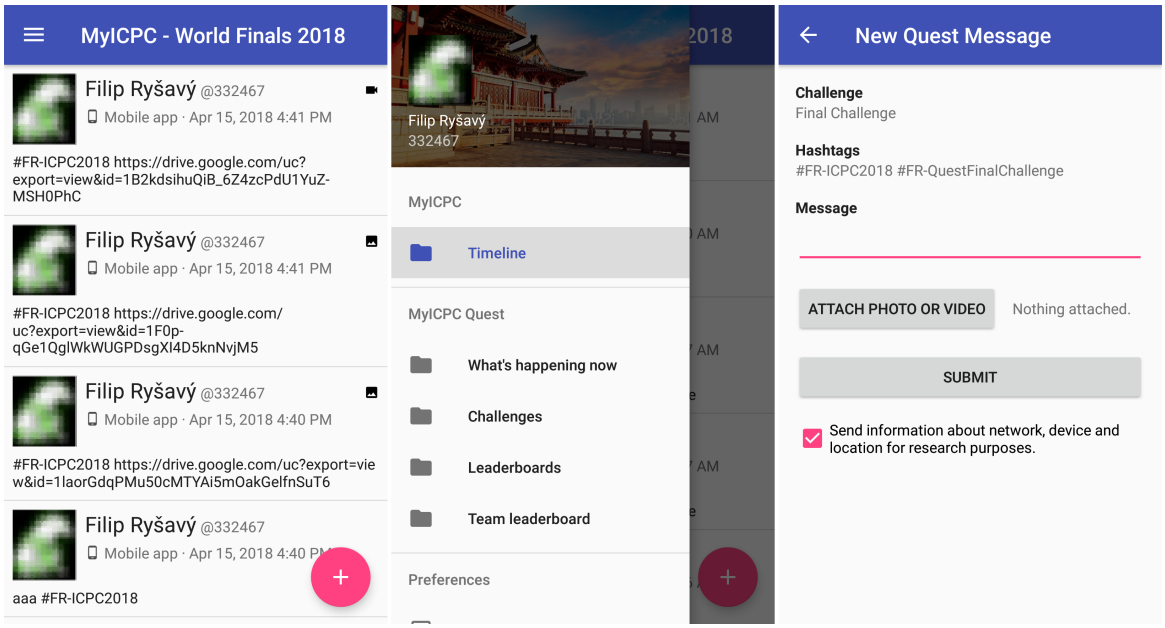


Figure 6.2: MyICPC mobile application.

6.3.3 Server application

The majority of changes in the MyICPC server application were additions and thus amount of work in order to update to the new version is minimal. The new version is also backward compatible with previous version. The deployment guide of the modified version of MyICPC is in the appendix D.

Chapter 7

Conclusion

This chapter summarizes the work that was done during making of this thesis and the requirement fulfillment. This chapter also describes several possibilities of the future work that can be done based on results of this thesis.

7.1 Summary

The first goal of this thesis was to implement a mobile application that would acquire the contextual information like a date, a location, network devices and others to support a research of context aware authentication using Wi-Fi enabled IoT devices. The second goal of this thesis was to take this application and integrate it with some application from real environment in case of this thesis with MyICPC that is used during the ACM International Collegiate Programming Contest for secondary activities of the contest. It would also require extending a current version of MyICPC with REST API.

This thesis presented the design and the implementation of the solution in a form of the context acquisition application, the MyICPC mobile application and extension of the current version of MyICPC. The solution was tested and it successfully fulfilled the requirements. It acquires the context, it allows contestants to post to the MyICPC timeline and the Quest game and it is easy to upgrade from previous version of MyICPC to the new version created in this thesis. The solution is ready for the deployment during regional contest later this year.

7.2 Future work

Even though the context acquisition application is complete at the moment the research of context aware authentication using Wi-Fi enabled IoT devices is still continuing. The application might need to adapt to the direction of research for example with a new type of contextual information that will need to be acquired. There is also possibility of performance improvements especially in the acquisition of the network devices.

The goal of this thesis was to implement the timeline module and the Quest game module into the mobile application. However, MyICPC contains other modules like score board,

schedule or gallery. In the future work, there is possibility to include these modules in the mobile application too. This thesis was focused on the implementation of the functionality and even though the design guidelines were fulfilled the future work might be focused on look and feel of the application to satisfy the user.

Bibliography

[1] M. Trnka, M. Tomasek, and T. Cerny, "Context-aware security using internet of things devices," in *International Conference on Information Science and Applications*, pp. 706–713, Springer, 2017.

[2] M. Trnka, F. Rysavy, T. Cerny, and N. Stickney, "Using wi-fi enabled internet of things devices for context-aware authentication," 2018.

[3] "The acm-icpc international collegiate programming contest." `https://icpc.baylor.edu`.

[4] R. Smetana, "Next generation of second-screen, realtime application myicpc," 2016.

[5] "Great firewall of china - comparitech." `https://www.comparitech.com/privacy-security-tools/blockedinchina/`.

[6] A. K. Dey, "Understanding and using context," *Personal and ubiquitous computing*, vol. 5, no. 1, pp. 4–7, 2001.

[7] R. Ausanka-Crues, "Methods for access control: advances and limitations," *Harvey Mudd College*, vol. 301, p. 20, 2001.

[8] M. Trnka and T. Cerny, "On security level usage in context-aware role-based access control," in *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, pp. 1192–1195, ACM, 2016.

[9] M. Trnka and T. Cerny, "Authentication and authorization rules sharing for internet of things," *Software Networking*, vol. 2017, no. 1, pp. 35–52, 2017.

[10] C. Shi, J. Liu, H. Liu, and Y. Chen, "Smart user authentication through actuation of daily activities leveraging wifi-enabled iot," in *Proceedings of the 18th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, p. 5, ACM, 2017.

[11] I. Agadakos, P. Hallgren, D. Damopoulos, A. Sabelfeld, and G. Portokalidis, "Location-enhanced authentication using the iot: because you cannot be in two places at once," in *Proceedings of the 32nd Annual Conference on Computer Security Applications*, pp. 251–264, ACM, 2016.

[12] C. Marforio, N. Karapanos, C. Soriente, K. Kostiainen, and S. Capkun, "Smartphones as practical and secure location verification tokens for payments.," in *NDSS*, 2014.

[13] "Android developers." `https://developer.android.com`.

[14] "Apple developer." `https://developer.apple.com`.

[15] T. Cerny and M. Donahoo, "Survey on second screen systems," in *IT Convergence and Security (ICITCS), 2016 6th International Conference on*, pp. 1–5, IEEE, 2016.

[16] "Google awareness api." `https://developers.google.com/awareness`.

[17] "Java se at a glance." `http://www.oracle.com/technetwork/java/javase/overview/index.html`.

[18] "Keycloak." `https://www.keycloak.org`.

[19] "Wildfly." `http://wildfly.org`.

[20] "Postgresql: The world's most advanced open source database." `https://www.postgresql.org`.

[21] "Jetbrains: Developer tools for professionals and teams." `https://www.jetbrains.com`.

[22] "Apache maven project." `https://maven.apache.org`.

[23] "Appauth-android: Android client sdk for communicating with oauth 2.0 and openid connect providers." `https://github.com/openid/AppAuth-Android`.

[24] "Spring framework 5.0." `https://spring.io`.

[25] C. Walls and R. Breidenbach, *Spring in action*. Dreamtech Press, 2005.

[26] "Junit 5." `https://junit.org/junit5`.

[27] "Apache jmeter." `https://jmeter.apache.org/`.

Appendix A

Nomenclature

ACM Association for Computing Machinery

ACM-ICPC ACM International Collegiate Programming Contest

API Application programming interface

BSSID Basic service set identifier

CA Context awareness

CAS Context-aware security

CSI Channel state information

GSM Global System for Mobile communications

HTTP Hypertext Transfer Protocol

HTTPS Hypertext Transfer Protocol Secure

ICMP Internet Control Message Protocol

IDE Integrated development environment

IoT Internet of Things

Java SE Java Platform, Standard Edition

JDK Java Development Kit

JPA Java Persistence API

JRE Java Run-time Environment

JSON JavaScript Object Notation

JSP Java Server Pages

JVM Java Virtual Machine

OIDC OpenID Connect

REST Representational State Transfer

RSSI Received signal strength indicator

SAML Security Assertion Markup Language

SDK Software Development Kit

SQL Structured Query Language

SSID Service Set Identifier

SSO Single Sign On

TEEs Trusted execution environments

URL Uniform resource locator

VPN Virtual private network

Appendix B

Content of included CD

`|-- context-acquisition-app`	**Context Acquisition App**
`|   |-- src`	Source code
`|   |-- build`	Build
`|`	
`|-- myicpc-android-app`	**MyICPC Android App**
`|   |-- src`	Source code
`|   |-- build`	Build
`|`	
`|-- myicpc-2.0`	**MyICPC 2.0 with API for Android App**
`|   |-- src`	Source code
`|   |-- build`	Build
`|`	
`|-- rysavfi1-thesis-2018.pdf`	**This thesis in PDF**
`|-- rysavfi1-thesis-2018.zip`	**Source code of this thesis**

Appendix C

Example of acquired context

The following listing contains an example of an acquired context. The acquired context does not contain a user identity as the context was acquired by the standalone context acquisition application that does not have any user identity provider.


```
{
  "availableWifis": [
    {
      "bssid": "c8:3a:35:10:88:c8",
      "capabilities": "[WPA-PSK-CCMP][ESS]",
      "centerFreq0": 2452,
      "centerFreq1": 0,
      "channelWidth": "CHANNEL_WIDTH_40MHZ",
      "frequency": 2472,
      "level": -37,
      "ssid": "Tenda_1088C8",
      "timestamp": 1122619758615
    }
  ],
  "bluetoothDevices": [],
  "connectedWifi": {
    "bssid": "c8:3a:35:10:88:c8",
    "frequency": 2472,
    "ipAddress": "192.168.0.100",
    "linkSpeed": 72,
    "rssi": -40,
    "ssid": "\"Tenda_1088C8\"",
    "state": "COMPLETED"
  },
  "device": {
    "deviceBrand": "HONOR",
    "deviceModel": "FRD-L09",
    "operatingSystem": "Android",
    "serial": "0000000000000000",
    "version": "6.0"
  },
  "gsmSignalStrength": -113,
  "location": { "latitude": 0, "longitude": 0, "timestamp": 0 },
  "network": {
    "gateway": "192.168.0.1",
    "ipAddress": "192.168.0.100",
    "macAddress": "B8:08:D7:3A:AE:46",
    "netmask": "255.255.255.0",
    "networkDevices": [
      { "ipAddress": "192.168.0.1", "macAddress": "c8:3a:35:10:88:c8" }
    ],
    "scanProgress": 0.03543307086614173,
    "scanStart": 1527003220356
  },
  "provider": "Context Acquisition App 1.0", "timestamp": 1527003229593
}
```


Listing C.1: Example of the acquired context.

Appendix D

Deployment guide

MyICPC 2.0 with API for Android App

This version of MyICPC 2.0 is extended by API for Android App. The API is used by MyICPC Android App. The API uses CM5 Keycloak for authentication. The API provides functionality of Quest module. The API provides lists of quest notifications, challenges, leaderboards and their details. The API also allows to participate in challenges directly instead of using Twitter.

Installation

1. Follow MyICPC documentation: `https://icpc.baylor.edu/xwiki/wiki/icpcdev/view/MyICPC2`
2. Populate database also with `apiSchema.sql`.
3. Configure Google API
   - Open `https://console.developers.google.com`
     - Create new project, e.g. `World Finals 2018`
     - Enable Google Drive API
     - Create credentials and choose service account key
       - Create service account, e.g. `Service Account #01`, with role Project - Editor.
       - Select JSON key type
     - Store downloaded JSON file and note its location
   - Open `https://drive.google.com`
     - Create folder to store photos and videos, e.g. `World Finals 2018`, and note its ID (you can find it in URL)
     - Share the folder with the service account
     - Share the folder, choose public on web
4. Configure API per contest, see the listing D.1

```
INSERT INTO apisettings VALUES(
  nextval('apiSettings_id_seq'),
  true,
  'https://.../auth/realms/cm5/protocol/openid-connect/token/introspect',
  'cm5-backend',
  '37ab5f1d-a3ca-4ff2-adc4-b3c41e1d848d',
  'service-account-01@world-finals-2018-199920.iam.gserviceaccount.com',
  '...',
  '5430c5bc00a1e84c84f0ca093548626f52bef56b',
  'https://accounts.google.com/o/oauth2/token',
  'world-finals-2018-199920',
  '15fP2oLlmyholgcMw3j_KdHEqzyNibbHo',
  1
);
```

Listing D.1: Example of API configuration.

Endpoints

All http requests require Authorization header with "Bearer ${accessToken}". All endpoints return 401 if user isn't authenticated, 403 if user if authenticated but isn't in MyICPC, 404 if contest is not found or API is disabled. URL of all endpoints is prefixed with `api/{contestCode}` where `contestCode` is an identifier of a contest.

Endpoint	Method	Description
`user`	GET	Response 200.
`timeline`	GET	Response 200. Optional headers X-Offset and X-Count, default values 0 and 30.
`timeline/count`	GET	Response 200.
`timeline`	POST	Response 201, 400.
`quest/whats-happening-now`	GET	Response 200.
`quest/challenges`	GET	Response 200.
`quest/challenges/hashtagSuffix`	GET	Response 200.
`quest/challenges/hashtagSuffix /accepted-submission`	GET	Response 200.
`quest/challenges/hashtagSuffix /rejected-submission`	GET	Response 200.
`quest/challenges/hashtagSuffix /pending-submission`	GET	Response 200.
`quest/leaderboards`	GET	Response 200.
`quest/leaderboards/urlCode`	GET	Response 200.
`quest/team-leaderboard`	GET	Response 200.

Table D.1: Overview of all REST API endpoints.

Changelog

- New things
  - Add `myicpc-api-android module`
  - Add `myicpc-service-context module`
  - In `myicpc-model`:
    - Add API settings model
    - Add context models
  - In `myicpc-persistence`:
    - Add API settings repository
    - Add context repositories
    - Add DB create script `apiSchema.sql`
  - In `myicpc-webapp`:
    - Add `com.myicpc.tags.notification.MobileAppTile`
- Modifications
  - In `myicpc-parent`:
    - Add new modules and their versions to `pom.xml`
    - Add this readme
  - In `myicpc-webapp`:
    - Add new modules to `pom.xml` in order to compile them to `myicpc.war`
    - Add `com.myicpc.config.ApiConfig` to `web.xml`
    - Set Redis address to `redis://127.0.0.1:6379` in `web.xml`
    - Update `com.myicpc.tags.notification.NotificationTag` to work with `MOBILE_APP` and `com.myicpc.tags.notification.MobileAppTile`
  - In `myicpc-model`:
    - Add `MOBILE_APP` to `com.myicpc.enums.NotificationType`
  - In `myicpc-persistence`:
    - Add method `findByContestAndContestParticipantExternalId` to `com.myicpc.repository.quest.QuestParticipantRepository`
    - Add method `findByContestAndExternalId` to `com.myicpc.repository.teamInfo.ContestParticipantRepository`
    - Fix problems with `scoreboardmodule_05:12:2017.sql`
  - In `myicpc-service`:
    - Add method `addMobileApp` to `com.myicpc.service.utils.lists.NotificationList`
    - Add `MOBILE_APP` to `TIMELINE_TYPES` in `com.myicpc.service.timeline.TimelineService`
  - In `myicpc-service-quest`:

    * Add `MOBILE_APP` to `QUEST_TIMELINE_TYPES` in `com.myicpc.service.quest.QuestService`
    * Add `MOBILE_APP` switch cases to `com.myicpc.service.quest.QuestSubmissionService`
  - In `myicpc-service-social`:
    * Add `MOBILE_APP` to `GALLERY_TYPES` in `com.myicpc.social.service.GalleryService`

Notes

- If you want participant to submit using this API, make sure that he has setup CM5 person ID properly. Also it would be nice if profile picture URL was imported.
- Web application does not have user interface for API settings, it need some modifications.