FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

BACHELOR THESIS

Ekaterina Kurysheva

Home alarm system

Department of Distributed and Dependable Systems

Supervisor of the bachelor thesis: doc. RNDr. Petr Hnětynka, Ph.D.

Study programme: Computer Science

Specialization: Software and Data Engineering

Prague 2018

I declare that I carried out this bachelor thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 paragraph 1 of the Copyright Act.

In.....date..... signature

I would like to thank my supervisor doc. RNDr. Petr Hnětynka, Ph.D. for his patient guidance and valuable advice regarding this thesis.

I would also like to express the gratitude to my family for allowing me to study abroad and for their constant support on this journey.

Title: Home alarm system

Author: Ekaterina Kurysheva

Department / Institute: Department of Distributed and Dependable Systems

Supervisor of the bachelor thesis: doc. RNDr. Petr Hnětynka, Ph.D., Department of Distributed and Dependable Systems

Abstract: The thesis defines a software solution for a home alarm system based on Raspberry Pi computer. It follows the Internet of things concept to put together simple and affordable hardware components. In this thesis, we analyse how to control the alarm system, how to detect an intrusion and capture pictures of the event and how to notify the user. Then we create a solution architecture, which meets the defined functional requirements. The architecture contains a security application controlling hardware on Raspberry Pi, a web application for managing the users, a server and an Android application for delivering the notifications and pictures. The result of this thesis is an implementation of a commercially independent home alarm system, which uses RFID tokens for user identification, detects the intrusion with a motion sensor, takes pictures with a USB camera and delivers the captured information to the Android application on the user's mobile device.

Keywords: Internet of things, Raspberry Pi, Home alarm

Contents

1. Introduction

Technologies are developing at a high rate during the last century. Computers become smaller, and their potential, on the other hand, grows more prominent each day. As a result, there is a massive movement towards automation in all areas of our lives. This movement has played a vital role in the formation of the Internet of things (IoT) concept [23], which became the main inspiration for this thesis.

The Internet of things is a concept of connecting physical devices to a network, where they can exchange data and assist, automate or even manage processes. The system usually includes sensors, electronic devices, the network for communication between the parts and the managing software. This concept has found its application and development in environment monitoring, optimisation in manufacturing or energy management. It brings the advantage of remote control over processes saving time and money for people, like, for example, when the world's first IP camera saved the customers two air flights a day allowing to monitor oil spills in the sea via the Internet [20]. Application of the Internet of things clearly allows an increase of efficiency and minimisation of the human factor in the area.

One of the biggest concerns for people always was the safety of their belongings and privacy of their living and working space. The modern solutions to this problem are approaching it from angles that vary from using advanced door locks to complicated automatic systems with multi-level authentication, all of which are supposed to prevent illegal access to the private territory. Security has become a giant industry and success factor in many businesses, and developing hardware and software technologies find their application in the field.

The evolution of modern technologies has brought a vast amount of affordable and portable devices to the market, which can be used to provide custom security solutions. Motion sensors, sound alarms, identification cards and card readers are easy to purchase and use. Thanks to the availability of learning materials it became possible to create an automatic security system without any complicated prerequisites. Moreover, the own

solution brings the advantage of building the unique application, which meets individuals' needs.

The mentioned trends in the computer technologies field had a significant influence on the idea of the project. It appeared with the intention to create a custom application for monitoring private space, independent of any commercial services and usable in everyday life.

1.1. Thesis goals

The thesis's goal is to create a home alarm system based on widely available and affordable components, which allow easy adjusting of the solution architecture. We want our system to be able to detect an intrusion, to notify the users of the event and to take pictures of the space upon the alert.

We also set a goal to design the system so that it can identify the users with tokens, which trigger switching of the system and provide a user management interface. The system is set to be accessible from the local network and the Internet.

1.2. Structure of the thesis

The thesis goes through problem analysis and project requirements and limitations in Chapter 2. Then the detailed discussion of the software and hardware solution follows in Chapter 3. The evaluation of the work and comparison with the existing systems are described in Chapter 4. Finally, we discuss future perspectives in Chapter 5.

2. Analysis

The goal of the project - named Safe Home - is to implement a custom home alarm system for private spaces. The high-level requirements of the system are to implement detection of an intrusion, collection of data about the intrusion, such as time of the event and pictures from the place at the time, and notification of the user, so he/she can take actions. The system also needs a control interface for switching the modes of operation. In more details, we discuss these requirements in the rest of the chapter.

2.1. Functional requirements

2.1.1. Intrusion alert

There are several ways to detect intrusion into a private space with simple hardware. It can be done with infra-red motion sensors, which react to people moving in the room, or with door or window opening detectors, which are based on magnets or with detectors, which react on sharp noises, such as window breaking. The most appropriate method for making the system flexible and usable for a large variety of room layouts is an infra-red motion sensor since the hypothetical monitored space does not have to have doors or windows.

To take pictures upon the motion detection, we can use an IP-camera, which has a built-in motion detector, or a separate camera, which can be controlled by a computer or via a network. IP-cameras usually use the manufacturer’s cloud storage and mobile applications, which eliminates the idea of creating a custom solution. Therefore the separate camera fits the criteria best.

The system needs to have the active and standby modes for hours when security is required to be on or off. The modes switching must be protected against illegal access. After putting together the motion sensor and the camera, the Safe Home system intrusion alert functionality involves detecting the motion in the monitored space and taking pictures upon the detection when the application is active. Then this component is set to deliver the information to the module responsible for notifications and access to pictures.

2.1.2. User identification

A user is a person who has the authority to access the monitored space and is interested in keeping it from intrusion. One space can have multiple users attached to it, for example, family members.

Hardware options of user identifications may be radio-frequency identification tokens read by RFID reader, fingerprint scanners or pin code entering. Fingerprint scanners are expensive, and the biometric data are sensitive. Pin code entering, on the other hand, does not provide any data about the user, who entered it. The RFID tokens are widely used, affordable and allow the system to identify the user, which makes them the best option from the listed above.

To prevent illegal access to the system, the users and their identification tokens need management. The management interface must provide the opportunity to add and delete accepted tokens.

The user also needs to get notifications from the system and access to pictures taken by the camera remotely. This means the need for the remote access interface for browsing the pictures, and identification of the user in some additional way other than RFID token to deliver notifications and pictures outside of the local network.

2.1.3. Remote access

The common ways to deliver notifications to the users remotely are by SMS, e-mail or in a mobile application. SMS sending requires usage of a SIM-card or a paid third-party SMS service. Sending e-mail does not require paid services, but it lacks flexibility and a possibility of interactions with the user

Remote access to the pictures can be implemented as an e-mail attachment, on a website or in a mobile application. E-mail attachments' flaw is that pictures are not collected at one place. A mobile application offers a possibility to combine notifications delivery with picture browsing, which is why we choose it as a mean of remote access.

2.2. Analysis of the hardware

The application is meant for usage at home with no particular equipment or predispositions for the security enhancements. However, the minimum requirement is access to the Internet from the monitored place since Safe Home is supposed to deliver notifications of the events.

To connect the motion sensor, camera and RFID reader into one system, which is capable of communication with a mobile application, we need to control them via single-board computer. When using them together, it is possible to create an independent security unit, which is capable of user identification and detecting intrusion.

There are various single-board computer options on the market, which are capable of reading the input from the hardware parts we want to use. We can divide them into two categories: computers which have the operating system and which do not. Computers with the operating system include but not limited to Raspberry Pi, Orange Pi, ODROID computers, and an example of a computer without the operating system is Arduino.

Performance and abilities of single-board computers with operating systems are comparable with modern mobile devices. They can act as a smart component of the home alarm system, which manages connected devices, analyses the changes in the environment and handles components' accessibility issues. The computer with the operating system can also act as a server and provide a web application or a web service.

A computer without the operating system reads and writes programmes directly to microcontrollers and is the best suit for controlling the electronics, as, for example, sensors or LEDs. It does not have the time and resources overhead which appears when using the operating system but has a limited number of use cases.

The functional requirements for the project imply that the computer for controlling the hardware must be able to run multiple tasks with different hardware concurrently, that is, managing the RFID reader, the motion sensor and a camera simultaneously when the system is on. The layered logic of the hardware controlling and data exchange with the system's components leads us to a decision to use a computer with an operating system. We choose to use Raspberry Pi computer as the most popular one [18] with a great community support.

2.2.1. Raspberry Pi

The Raspberry Pi [1] is a name of single-board computers produced by the Raspberry Pi Foundation [3]. It is the best-selling computer of its kind, the third after Apple Macintosh and Microsoft Windows PCs [18], highly affordable and widely distributed.

The computers are small in size, which varies from the size of a credit card to the size of a smartphone. It contains system on a chip (SoC), which is an integrated circuit broadly used in mobile computing, which ensures low power consumption. The SoC includes the CPU with a frequency of 700 MHz brought by the Broadcom Corporation [4] free from any license control. It also includes graphics processing unit (GPU) and RAM. Raspberry Pi Model B also has Ethernet circuitry, USB hub and GPIO (general-purpose input/output) pins. The USB ports on Raspberry Pi provide the option of connecting mice, keyboards or USB cameras, which extends the available functionality.

The Model B of Raspberry Pi has 26 GPIO pins including 3.3V and 5V power supply pins. Other pins can be configured to different pin-dependent settings like level-sensitive or edge-sensitive interrupt, or to the I2C bus. This variety of settings allows using the computer as a control unit in robotics and other embedded systems.

Raspberry Pi is also equipped with the Serial Peripheral Interface bus (SPI), which is used for the full duplex communication between devices on a short distance using master-slave architecture, which enables connecting of RFID reader to the computer.

The Raspberry Pi Foundation officially supports Raspbian operating system [5], however other third-party operating systems are available too. Raspbian is Debian-based and has pre-installed software for programming like Python, Java or Mathematica.

2.2.2. Card Reader

To identify the users with RFID tokens, we need the RFID card reader. The most commonly used card reader for this purpose is the model RFID-RC522. It is a communication device for contactless reading of and writing to the MIFARE [6] cards and tags. It uses voltage 3.3V for the power supply and SPI interface to communicate with the master CPU, hence acts as a slave. It is capable of error detection and supports encryption algorithm.

The MIFARE series cards and tags is a registered technology for the radio-frequency identification (RFID). The tags contain memory structure and perform authentication of a reader and are broadly used in all areas where the electronic tagging is appropriate. The examples of cards containing RFID tag include payments cards, ISIC (International Student Identity Card).

2.2.3. Motion sensor

The motion sensor available for this project is a passive infrared sensor which detects the motion based on infrared radiation level changing. Its sensitivity range is up to 6 meters of distance and up to 110º x 70º of width depending on the lens. It cannot detect how far the moving object is nor how many objects there are. It has low power consumption and doesn't wear out over the time.

2.2.4. Limitations for the Internet connection

As was mentioned at the beginning of this chapter, the minimum requirement for the place of Safe Home installation is access to the Internet. Most of Raspberry Pi models support connection to the Internet via Ethernet or Wi-Fi. We can suppose, that access to the Internet available at the place is protected by the firewall, and Raspberry Pi cannot be accessed directly from outside of the network. This means that we cannot consider direct remote access to the security module as a solution.

2.3. Architectural requirements

Application on Raspberry Pi which controls the hardware cannot be accessed directly from the Internet, as was discussed in 2.2.4. section. Therefore, there must be a part of the application available to the user outside of Raspberry Pi, providing the remote access user interface. Then, the Safe Home application concept breaks down into two major parts. The first is security module on Raspberry Pi which controls hardware. The second is the user interface with its supporting components.

We choose to implement application's primary user interface as a mobile application. That is because mobile devices are reported to share more than 50% of the market of computers in 2017 11, winning over desktops. The mobility of the smartphones also

provides the means to notify the user of the intrusion “on the go” whenever the device connects to the Internet, and mobile applications, in general, are more versatile and adjustable than simple e-mail notifications.

The decision to implement the user interface as a mobile application still does not provide a solution for fetching the information from Raspberry Pi security module. In this situation, a client-server pattern fits the concept.

2.3.1. Client-server model

The client-server model [12] is a design of communication between the server and usually multiple clients. The server stands for a communication party that provides data or service to clients upon requests. Clients are the other party; they do not usually share their resources. Clients initiate a connection. The communication has to have a template or protocol to allow interpreting requests and responses.

In case of Safe Home, Raspberry Pi security module and mobile application are clients, and the server is the new component responsible for passing relevant data between them, as illustrated in *Figure 1*. This solution solves the problem of Raspberry Pi being inaccessible because no direct access to the client from the server is required in this model since security module is to establish the communication.

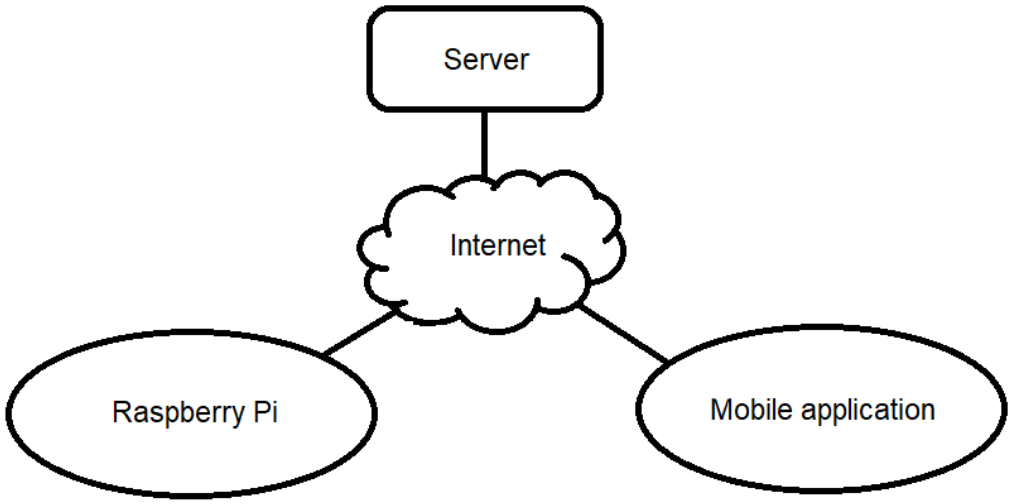

Figure 1. Design concept illustration.

2.4. Tools for the implementation

2.4.1. Java

"The Java programming language is a general-purpose, concurrent, class-based, object-oriented language." [13]. According to the TIOBE index of programming languages popularity [14], Java has been the most used programming language for the majority of years of its existence, as presented on the diagram in *Figure 2*. Java programs can run on a significant number of platforms and have an enormous number of third-party libraries. It has predefined interfaces for development of web applications with various implementations. Also, this language has strong support on the Raspbian operating system. Those factors together make Java language a perfect match for the project development.

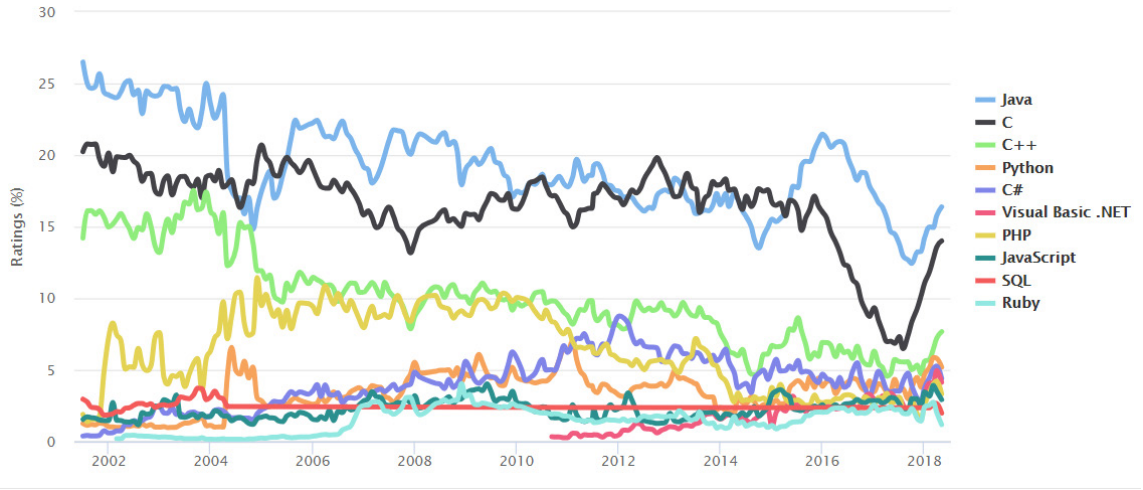

Figure 2. TIOBE Programming Community Index. [14]

2.4.2. Android

Android [2] is a name of the operating system for mobile devices based on Linux kernel. It is designed primarily for interactions via touchscreen with limited power supply and memory available.

Each Android application runs independently on other processes in its virtual machine (VM), which decreases security risks. Applications also have the minimum possible access to the device memory and functionalities, and they have to request the corresponding permissions to perform interactions which may be misused. The source

code and all the resources of the application for Android are compiled into Android package (APK) file, which can be then installed on Android-powered devices.

Choosing to implement the user client for Android is beneficial in the following ways. Firstly, according to Gartner, Inc. [10], smartphones with Android operating system share over 86% of the market by the first quarter of the year 2017. The table with supporting data is presented in *Figure 3*. Secondly, the operating system supports [8] programming in Java [9] language, which was chosen for the implementation of the project. Finally, Android platform provides the means for non-trivial interaction with the user, which can be explicitly customised for the project, and that is impossible when using e-mails or other third-party applications for delivering data from Raspberry Pi.

Operating System	1Q17 Units	1Q17 Market Share (%)	1Q16 Units	1Q16 Market Share (%)
Android	327,163.6	86.1	292,746.9	84.1
iOS	51,992.5	13.7	51,629.5	14.8
Other OS	821.2	0.2	3,847.8	1.1
Total	**379,977.3**	**100.0**	**348,224.2**	**100.0**

Figure 3. Worldwide smartphone sales to end users by operating system in 1Q17 (thousands of units). [10]

2.4.3. Server implementation

Java Enterprise Edition provides technology for easy Web server implementation: Java Servlet [19]. Java Servlet is platform-independent and has access to entire Java APIs. It is mostly used for implementing client-server communication based on HTTP protocol. A servlet is an object, which gets a request and produces a response. Servlets need servlet containers to manage the interactions, URL mapping and servlet lifecycle.

There are several servlet containers, open-source and commercial, to choose from. Apache Tomcat, GlassFish or WildFly (former JBoss) are the most common and actively supported. From those three, Apache Tomcat is the most light-weight and compact,

while the other two are instead considered mainly as application servers with a servlet container support as a feature.

The Apache Tomcat is an open source project which represents servlet container. It implements Java technologies such as JavaServer Page (JSP), Java Expression Language (JEL), Java WebSocket and Java Servlet interface. Tomcat provides the environment for deployment and running dynamic web applications. The technologies provided by Tomcat make it possible to implement different types of internet communication based on client-server architecture, including RESTful web services, SOAP (Simple Object Access Protocol) and others. Furthermore, JSP and JEL technologies deliver tools for the creation of rich HTML content, which makes Tomcat sufficient for server development.

2.4.4. Client-server communication

The communication between the parts is going via the Internet or local network. We are going to use HTTP protocol [7] since it is the most convenient when implementing Java Servlets and is supported by all browsers. The content of the HTTP requests can be further specified by communication protocols or approaches like SOAP (Simple Object Access Protocol), JSON-RPC (remote procedure call protocol encoded in JSON) or REST (Representational state transfer) to unify and make it more robust to changes.

SOAP is a platform-independent communication protocol. It uses XML message encoding with predefined SOAP namespaces. The server endpoints have strict descriptions which all requests and responses accessing the endpoints must follow. Any changes to those descriptions lead to the necessity of client application updates, but the communication goes through a robust validation process, which increases the security aspect.

JSON-RPC is another platform-independent communication protocol, in which messages are encoded in JSON (JavaScript Object Notation). It has a simple predefined structure for the requests, but the endpoint requests and responses do not have descriptions for validation like in SOAP. However, clients have to call specific methods, which leads to tight coupling between server and client.

REST, on the other hand, is an architectural solution, which has no strictly specified standard, except that the RESTful API is based on specific URI construction. The clients do not call specified endpoints, they fill the request URI with documented parameters and get data. The key feature of REST is using stateless unified requests, which is very flexible and easily modifiable solution. The RESTful communication service usually uses HTTP methods GET, POST, PUT and DELETE consistently, where GET requests are meant to fetch the required information without changing it, POST requests initiate the creation of a new entry, PUT requests are used for changing existing data and therefore DELETE requests for deleting data.

For communication between parts of Safe Home, we choose to use RESTful API. That is due to its maintainability and flexibility, since the application has the potential for large extensions in the future, and we would like to avoid strong dependence of clients on server methods.

3. Solution

The solution architecture for the project consists of four major components: Raspberry Pi security unit, user management web application, Android application and server, as demonstrated in *Figure 4*. Raspberry Pi unit is responsible for controlling USB camera, motion sensor and RFID reader. It holds the state of the system and contains the security logic. The user management web application is an implementation of the user interface for registering RFID tokens. It runs on Raspberry Pi and is accessible from the local network. The Android application gets the notifications about events registered by Raspberry Pi unit and allows the user to browse pictures taken by the USB camera. The server stores the event information and pictures from Raspberry Pi unit and shares the data with the Android application. *Figure 5* and *Figure 6* illustrate how the parts interact in two use-cases.

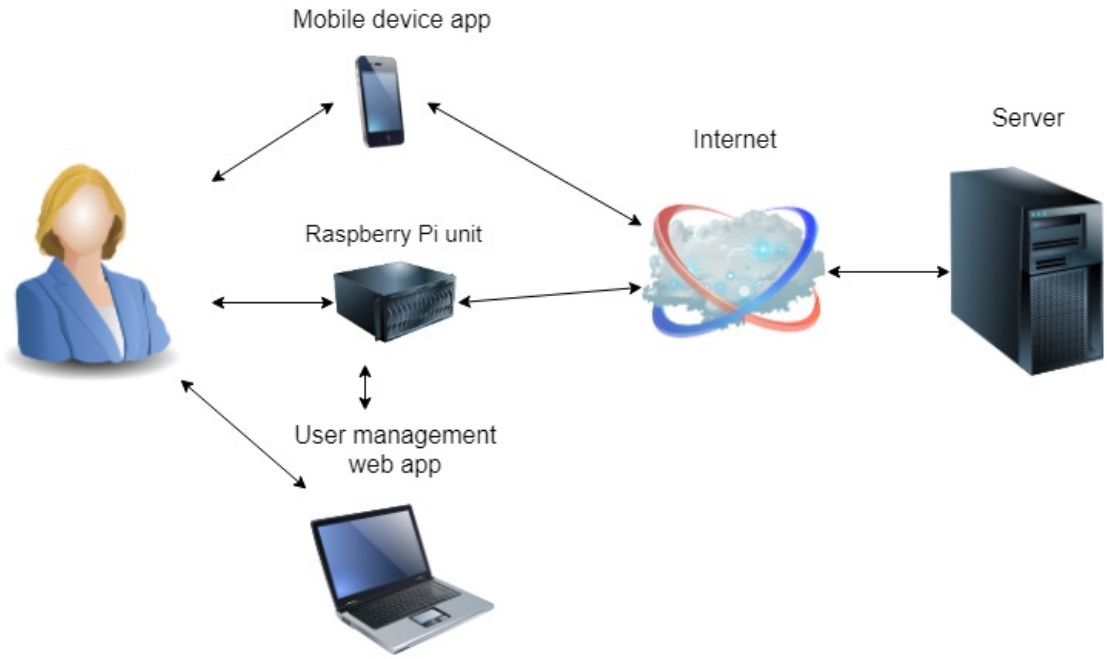

Figure 4. Safe Home components illustration.

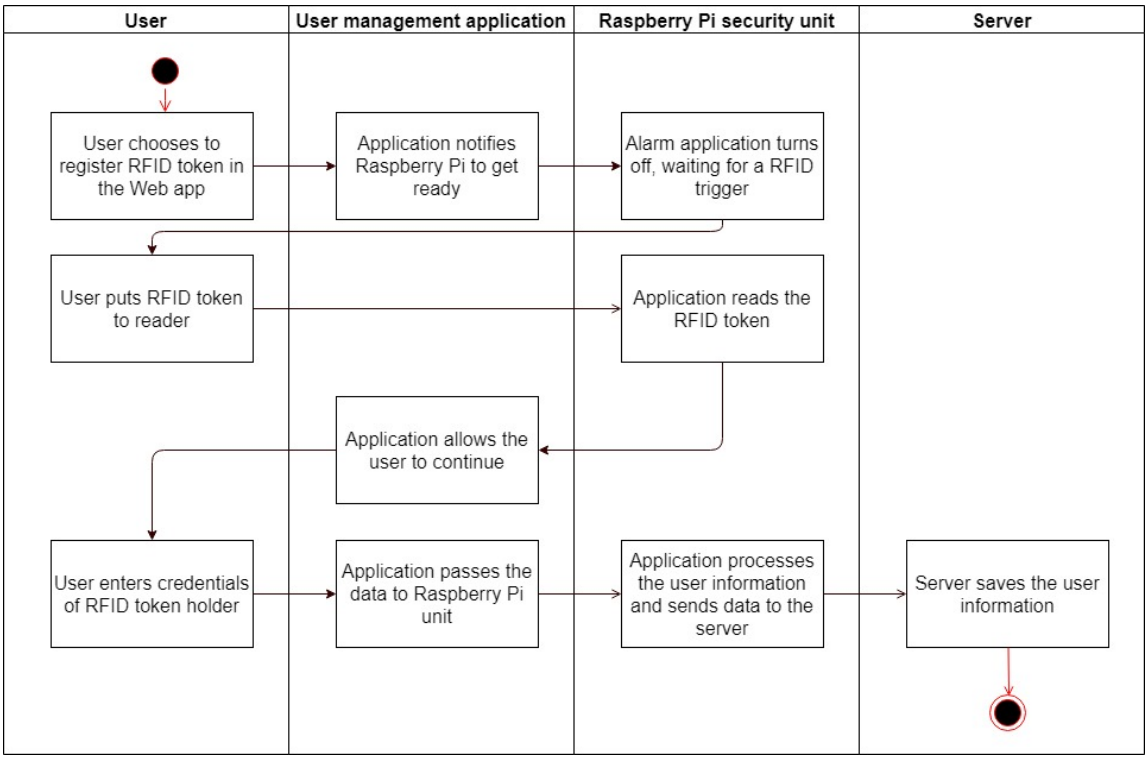

Figure 5. Components interactions during user registration.

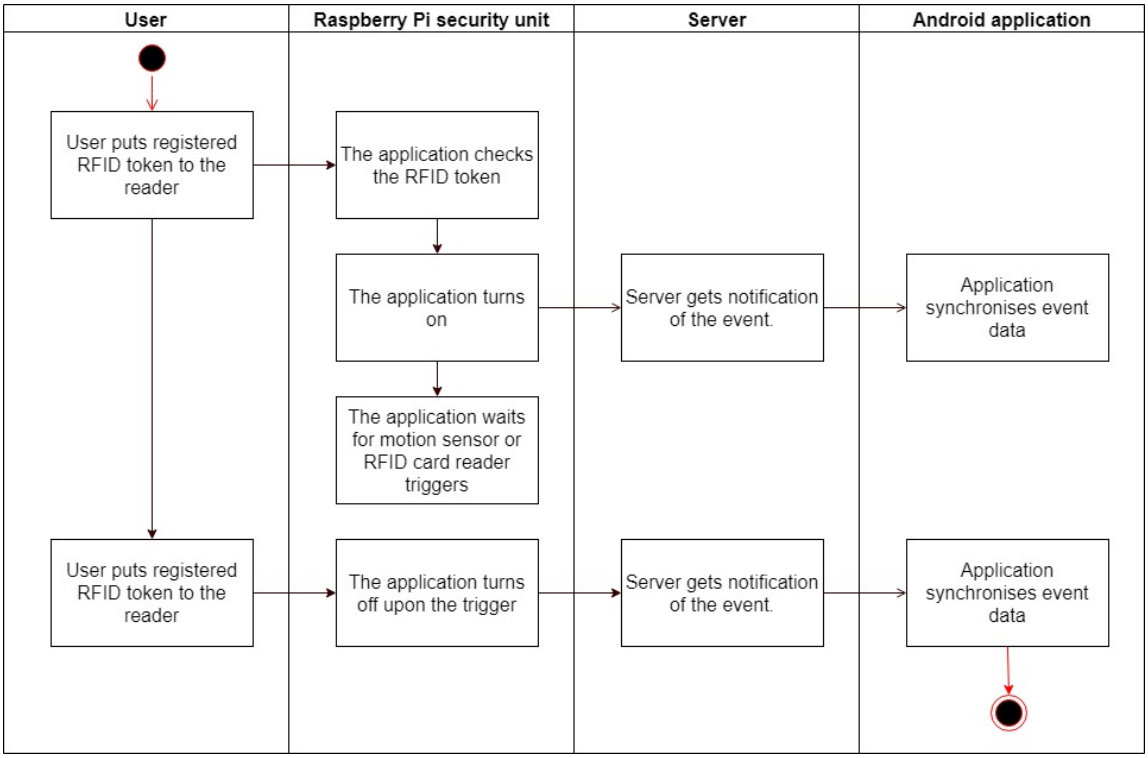

Figure 6. Components interactions while turning the application on and off.

3.1. Raspberry Pi module

Raspberry Pi module includes Raspberry Pi security unit and user management application. Its responsibilities can be divided into three categories. The first category is hardware management. The security logic of the application has RFID card reader, motion sensor and USB camera under control. The second category is user registration and management. It includes a user interface for registering RFID tokens and sharing data about the users. The third category is data exchange with the server. It consists of a communication interface and a communication policy.

Since those three parts of Raspberry Pi security module have a certain level of independence from each over, it is reasonable to apply the division to the module architecture to keep the model clear. The parts apparently have to communicate with each other, and our goal is to make them loosely coupled to improve the quality of the code [17]. For coordination of these parts, we choose the mediator pattern [15] as a solution for cases, when there are many-to-many dependencies in the application.

3.1.1. Mediator pattern

Mediator pattern in software development is a type of encapsulation of interactions between objects of the application. When there are many classes with complex logic and several communication cases, a mediator is an object responsible for the interactions. The classes or objects do not communicate directly with each other but access the mediator, which coordinates the requests and channels them.

This approach decreases dependencies between the objects, which leads to easier future maintenance and applying of modifications and overall improves reusability of the code. This pattern also enables to develop and test the subsystems independently, which helps to reduce the number of flaws in the application [16].

3.1.2. Raspberry Pi module architecture

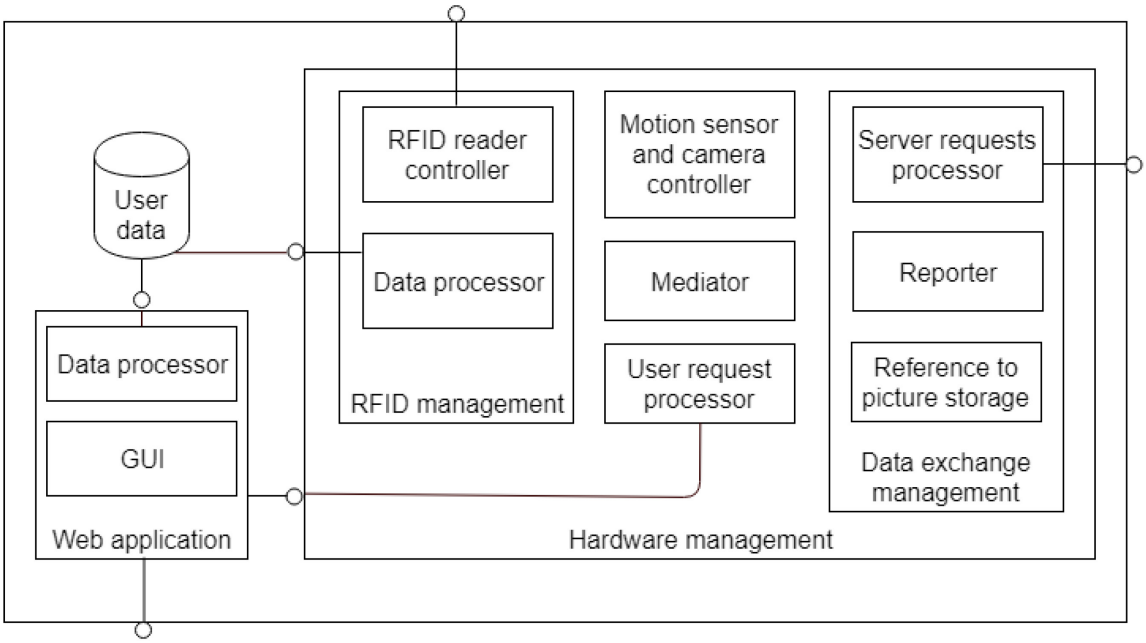

Figure 7. Raspberry Pi module architecture.

Raspberry Pi module communicates with the user and the server. The user can access it from two endpoints: the web application and by using registered RFID token on the RFID reader. The server gets data about events from Raspberry Pi via HTTP.

The web application allows the user to register RFID token as was illustrated earlier in *Figure 5*. The registration process includes filling the credentials of the user, which are later associated with the token. The web application also offers the interface for deleting registered users or changing their credentials. The data about the users gotten from this component stay in a local data storage and are later processed by the main submodule. In order to associate an RFID token with the entered credentials, the User request processor module from Hardware management communicates with the application and then passes the task to the Mediator, when it arrives.

The RFID reader controller endpoint reacts to reading a known RFID token and triggers the change of the application to the state “on” or “off” as was also illustrated in *Figure 6*. It accesses the local storage with the user data to identify the user and does not depend on the server. It accesses the local storage via a data processor, which watches changes in user information and reports it to the server. The RFID management

component performs simple tasks passed through the mediator and the change of state trigger goes through the mediator to the motion sensor and camera controller.

Communication with the server is controlled by the Data exchange management module and consists of the following parts. Raspberry Pi module sends the information about user credentials to the server in case of registering a new user, change in credentials or deleting of an existing user. The information about the associated RFID tokens is not shared with the server and stays inside the module to prevent the module inaccessibility in case of missing Internet connection. The module also notifies the server about turning the application on or off by the user and about detected motions. Moreover, the module sends the pictures taken by the USB camera to the server, and it sends the pictures separately from sending information about an intrusion to prevent the notification delivery failure in case camera is not available. The pictures are stored in the Raspberry Pi file system on SD card.

Coordination of the communication between the parts of the module goes through the mediator and sending information about the events on Raspberry Pi module goes through the reporter. The reporter is not a part of the mediator since its task is specific and does not include communication between the other parts, which does not fit into the idea of the mediator pattern. Its role is to collect the event information from the submodules and to pass it in a required format to the server, and the rest of the module does not depend on it. This architecture decision helps to fulfil the requirement for the parts of Safe Home to be available despite possible Internet connection issues.

The web application and the RFID reader controller do not share data access and the data processor, because the web application is separated from the rest of the module to a different application, which runs on a local server on Raspberry Pi. This decision has the benefit of the possible future extension of the web application with the user interface from the local network to the Internet. It also implies less dependency on the other components, easier component testing and better modifiability and maintainability.

The user data storage is implemented as a file containing all the information in JSON format. There is no database since the available memory on our Raspberry Pi is limited,

and the expected number of users for one household is low enough for this data storage solution to work. This decision is a topic for future enhancements to the system.

The RFID controller is separate from the motion sensor, and USB camera controller since the RFID controller part needs the access to the user data and has its logic for cases of a regular turn on or off operations and the new user registration part. Moreover, both components listen to different triggers (for RFID token and the motion detection), hence need to run concurrently. USB camera usage, on the other hand, depends on the motion detection and does not follow any other specific logic, therefore becomes a part of the motion sensor controller. The motion sensor controller also holds the information about the current state of the application ("on" or "off"). The communication between the components is illustrated in *Figure 8*.

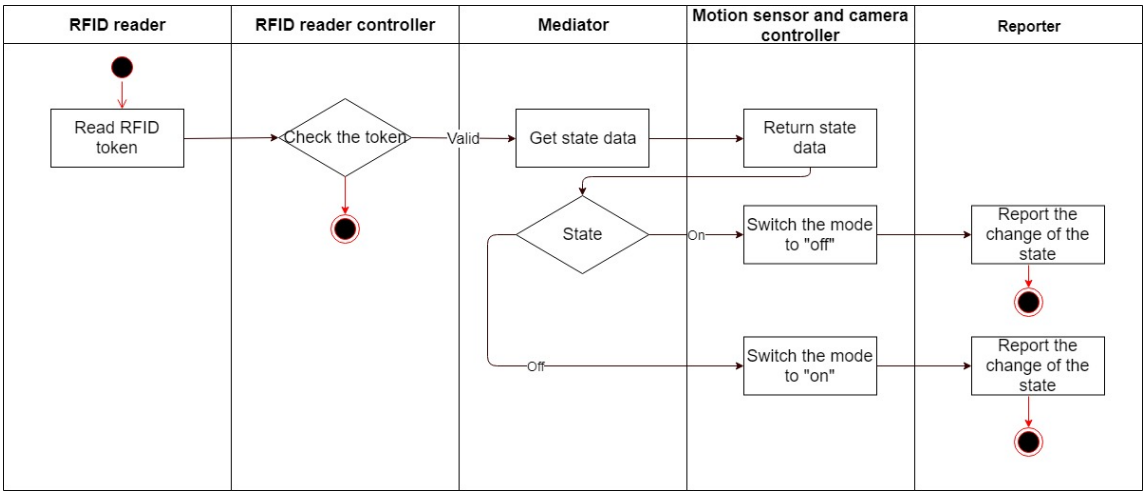

Figure 8. Communication inside Raspberry Pi module upon RFID token usage.

The intrusion alert requirement is satisfied within Raspberry Pi security module, where the hardware components are connected to Raspberry Pi computer and are organised to switch between standby and active modes and to react on motions when in the active state. The pictures from the security module stay on Raspberry Pi until the server does not accept them, therefore the data cannot be lost due to an unstable Internet connection.

The user identification is provided with the usage of RFID tokens and associated user profiles in the Android application. Therefore, the user identification requirement on the Raspberry Pi application side is also fulfilled with the described implementation solution.

3.2. Server module

Servlet module of Safe Home has two types of clients. The first type is a security module, which sends data about clients and events in a monitored space. The second type is a user application, which we decided to implement as an Android application, but it can be extended to other platforms potentially. This type of the client needs to get information about the events and pictures to notify the user and provide access to the pictures. In this clients definition, the data flow goes from the security module to the application via the server, which role is to store data and control access to them, as illustrated in *Figure 9*, which makes the server to act as a cloud.

Figure 9. Data flow illustration

This strict division of the clients to the categories implies that the server's communication with them needs to be separated into different interfaces, and the data shared between the two clients need to be stored. In Java Servlet technology we can easily implement two different servlets for the two clients respectively and to configure a database for our server.

3.2.1. Server module architecture

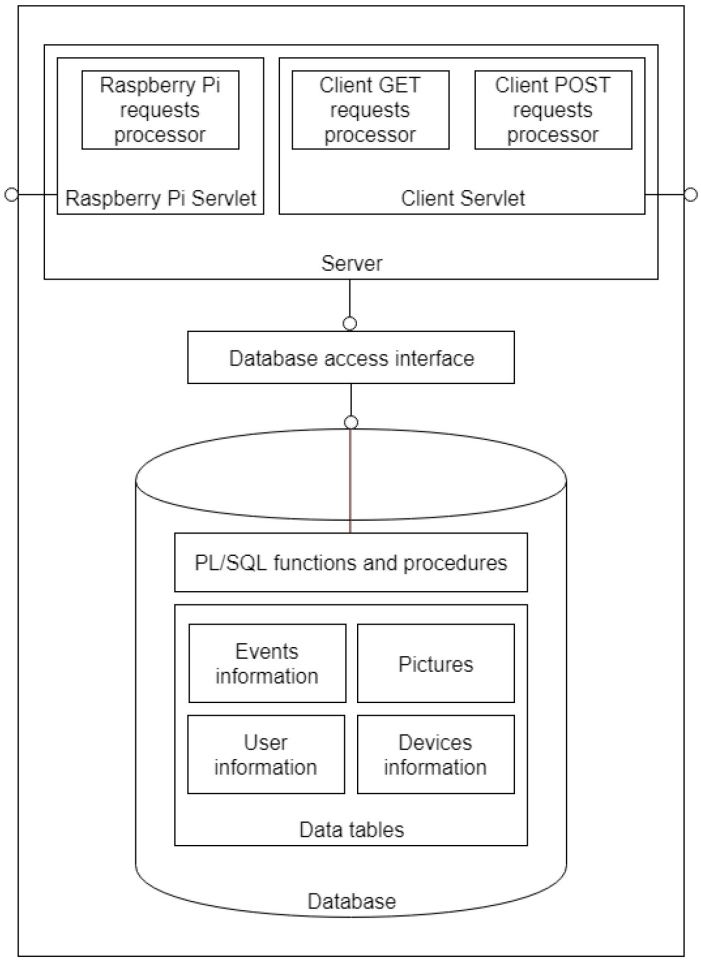

Figure 10. Server module architecture.

Server module of Safe Home consists of two Java Servlets and a database layer. The servlets respond to requests from Raspberry Pi module and Android application respectively. Database layer exists inside the module and cannot be accessed from other modules of the application.

Servlet for communication with the security module has its' REST API. This API has a set of accepted URI parameters, which tell the server what action to perform. In the defined data flow, all data are sent as HTTP POST requests. The server then accesses the database to store events, time of events and photo in tables. It also saves the information about users to identify them, when they try to log in via the mobile application.

Servlet for communication with the mobile application has a similar REST API. The majority of requests from the application are HTTP GET requests. Those requests contain identification tokens for authentication of the user, which the server validates before providing any data. The application requests dates of the latest events, the event information and pictures.

The database layer consists of the database with tables containing data about the events, users and pictures, functions and procedures and an interface for the database access from servlets. It watches the data consistency and saves the updates from the user profiles. The interactions between the components of the module are illustrated in the following figures.

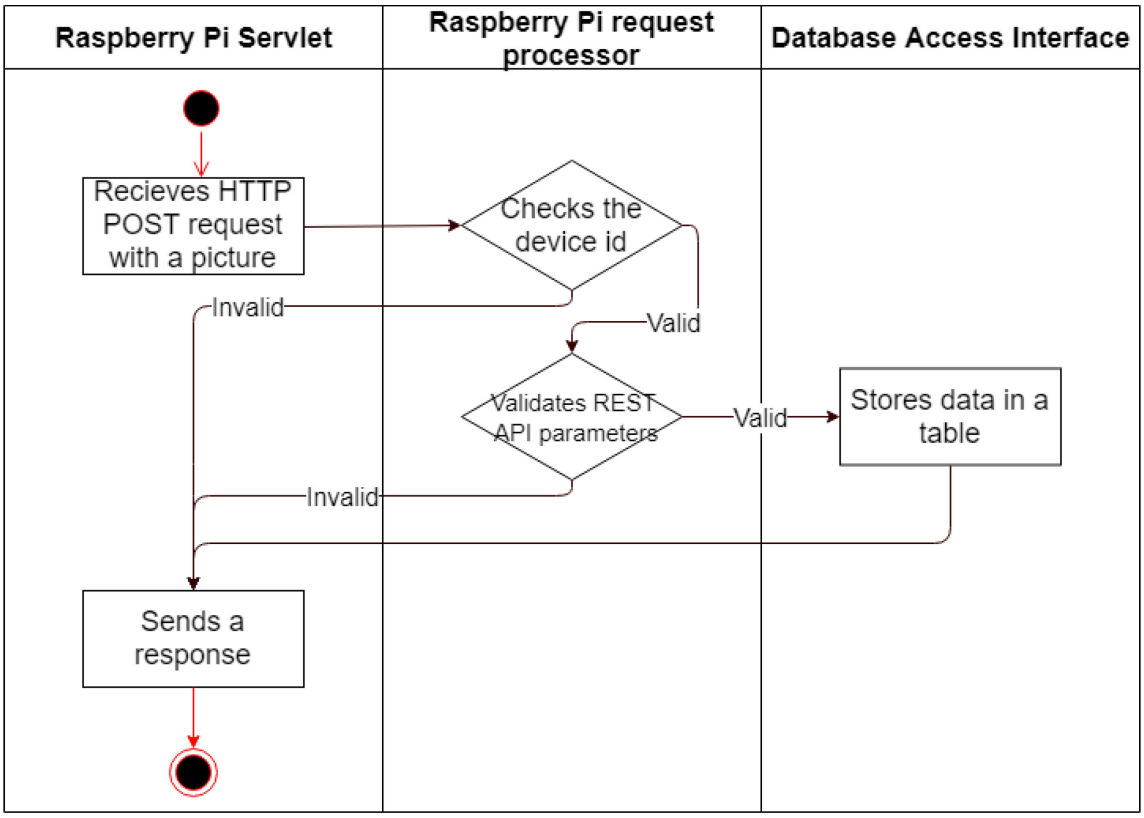

Figure 11. Communication inside the server module components when receiving a picture.

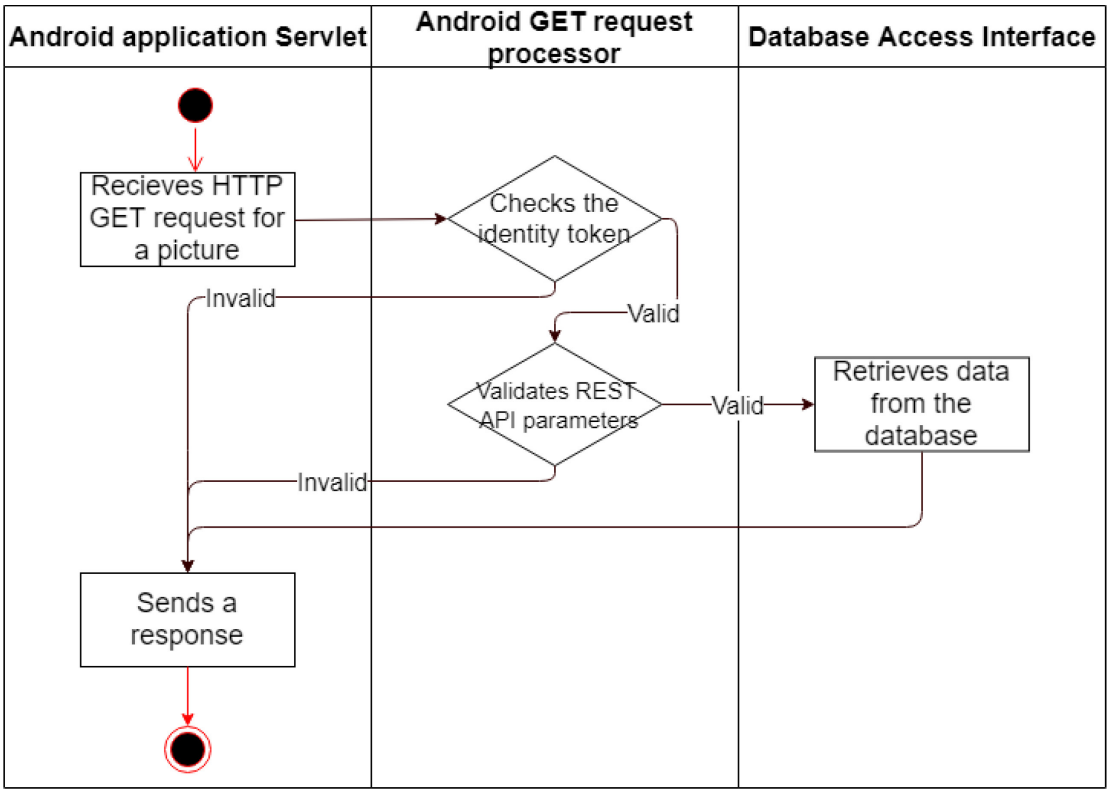

Figure 12. Communication inside the server module components when requested the picture.

3.3. Android application module

Android application module's goal is to identify the users to associate them with the security module. It also needs to display the notifications and to provide an interface for the users to watch what happens at home.

There are several methods for the user identification in applications. The user can log in with a combination of username and password, which can be unique for the system or connected with a third-party profile like Facebook or Google account. Alternatively, the two modules can create a pair with an identification token (barcode, for example). We decided to use a combination of username and password, which the user sets during registration of the RFID token to avoid dependency on other systems and to make the logging in process available even if the web application is not accessible for displaying the pairing token.

3.3.1. Android application module architecture

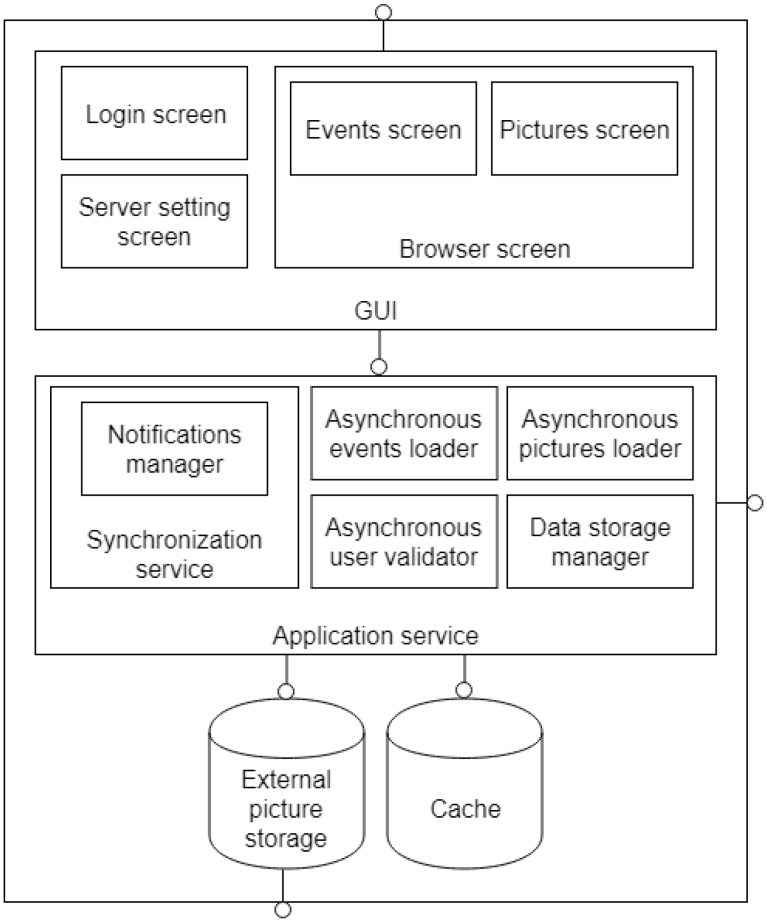

Figure 13. Android application module architecture.

The application's entry point from the user view is a login page. The user types his or her username and password set during the RFID token registration there and the data from the associated Raspberry Pi are loaded if the user is authorised. When the user is identified, the data about the authentication are stored in the cache, and the component with pictures and text information from the security module becomes the entry point for the user. Only authenticated users can switch the states of the alarm systems and get an access to the pictures from the security module.

The other entry point of the module is for the server. The Android application always initiates the connection. It validates the user with the server method, gets pictures and event information with separate requests. The connection with the server is not

continuous to avoid vast battery consumption on the device and it is asynchronous to make the application usage flow smoother.

We considered different options for providing access to the events information to the user. The Android operating system allows the applications to store data in the cache, which is not shared between different applications, but has small capacity and can be cleared by the system. There is also an option to store data in internal or external memory. The internal memory for the application is private and has more space than the cache. The Android operating system allows the applications to have a database in internal memory. The external memory, on the other hand, can be accessed by all applications on the device.

We decided to store pictures as media files in external storage in this module to allow the user to access them as regular files, which is not possible when using other methods. We store other text information delivered by the server in the cache memory since the application's primary role is to display the latest events and not to store the whole history of Safe Home, which is available on the server.

The application's notifications service runs in the background and synchronises updates with the server using the new Android API feature of system task scheduling. The application communicates with the server over HTTP using REST API mentioned in the description of the server module, and the requests for the synchronization are also asynchronous. The application is fully responsible for fetching the data at the right time, but the device, where the application runs, may influence the background tasks execution by delaying or cancelling them. The synchronization flow is depicted in *Figure 14.*

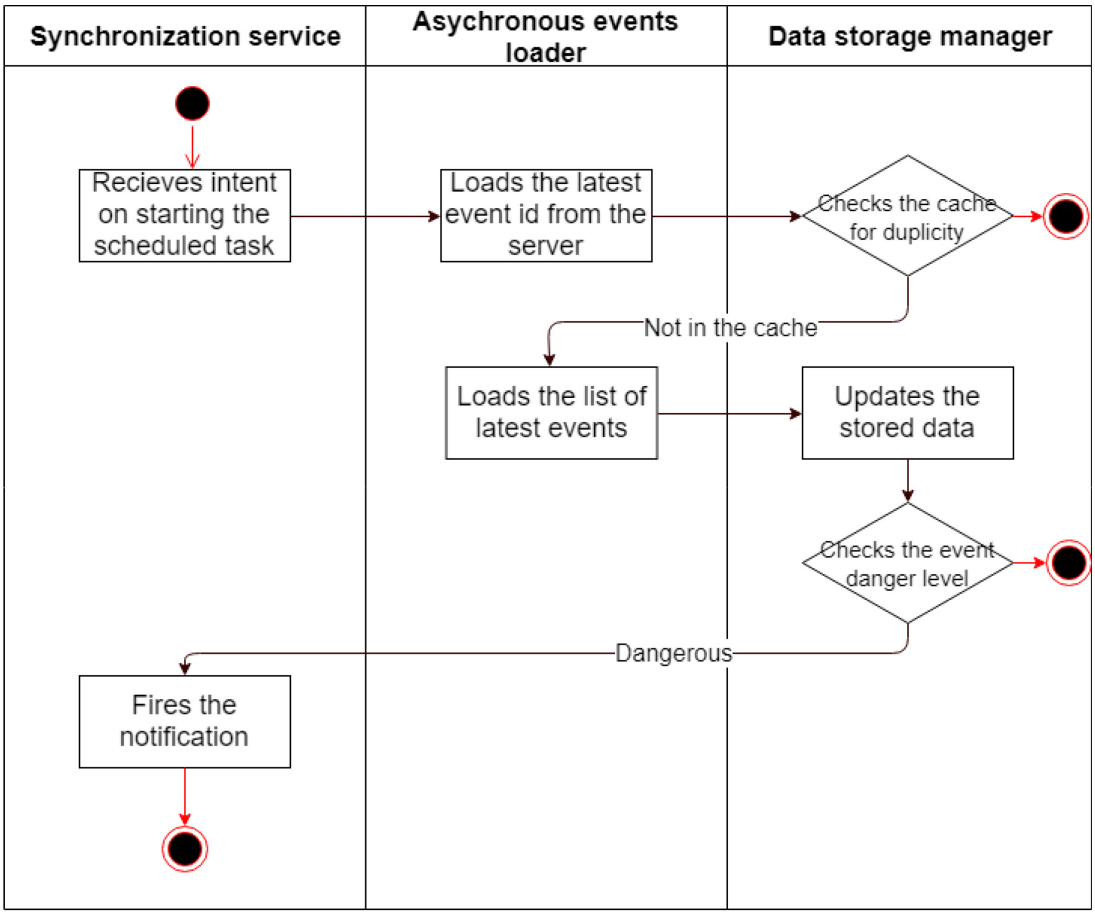

Figure 14. Synchronization flow on the Android application.

4. Usage and related works

The Safe Home application is a system providing security monitoring for the user's home. It consists of Raspberry Pi component, which belongs to space, where we want the monitoring. It needs to be installed in such a way, that the motion sensor and the camera monitor the same area, and that the RFID reader is not situated in this area. The other component is the server on the Internet, which can be installed on the user's personal computer or on the faculty's computer for the testing purposes. And the last component is an Android application on the user's mobile device.

The installation and usage of Safe Home are straightforward. Hardware components include Raspberry Pi computer, RFID RC-522 reader, a PIR motion sensor, a LED, a resistor, jumping wires and breadboard. Then the user needs a USB camera, a micro-USB power supply, an SD card and an Ethernet cable. After the security module hardware is connected as demonstrated in *Figure 15* and the operating system on Raspberry Pi is up and running, the user needs to run an installation script, which loads needed libraries and configures the security module.

The server and the database are also easily put together with a quick deployment and a database script run on a computer devoted to this task. The Android application gets to the mobile device with the standard installation file for Android. The step-by-step guide of how to get together all the components is available in the *User guide* attachment to this thesis.

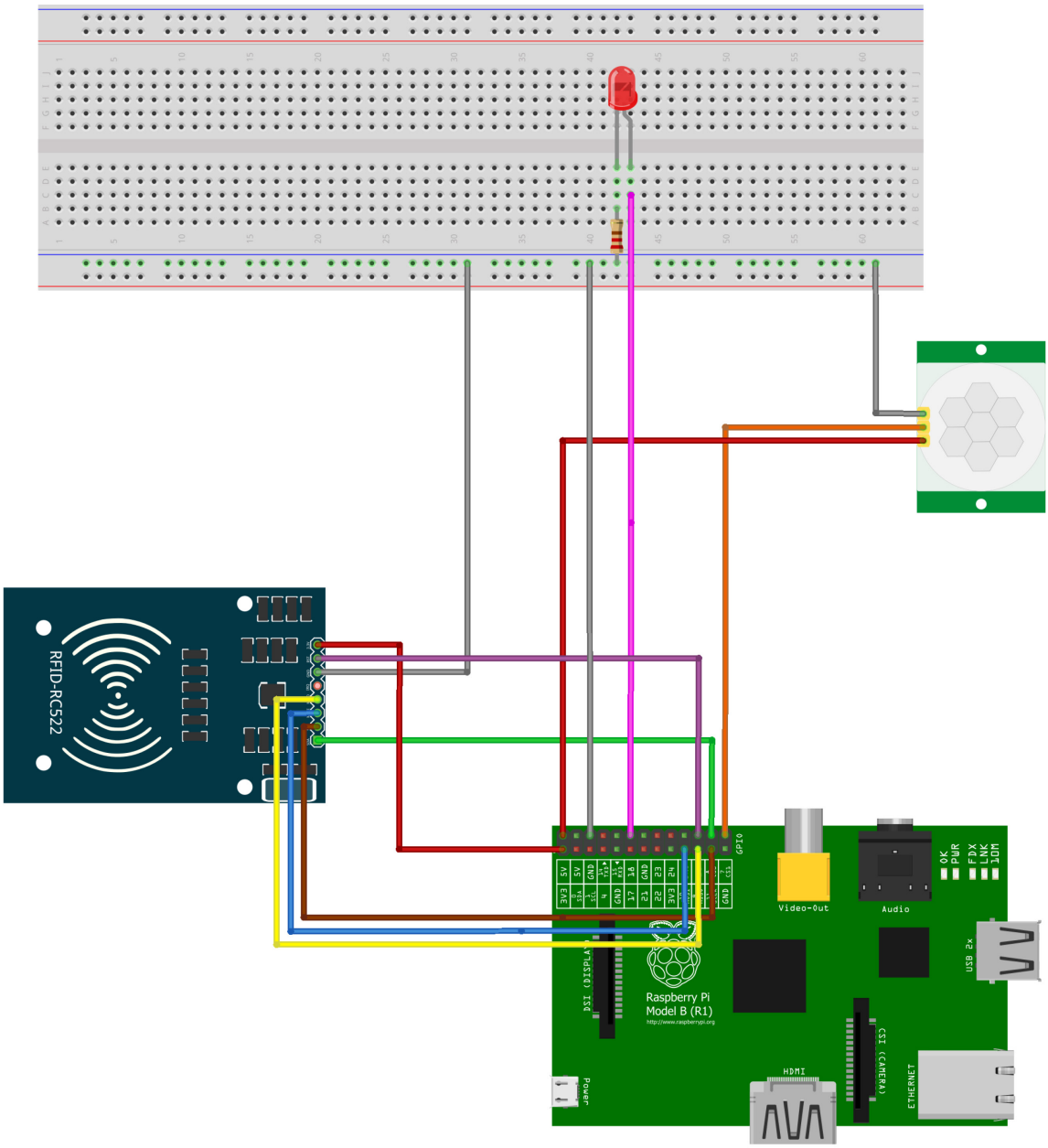

Figure 15. The hardware circuit scheme

4.1. How to use the application

To start with Safe Home usage, a user needs to register an RFID token, which is going to be used for the alarm switching. The token can be in a form of an RFID key, an ISIC card, a debit or credit card or any other card, which contains an RFID chip. The data read by the RFID reader stay exclusively on Raspberry Pi device and are only used to identify the registered card.

So, the first step is to open a website on the Raspberry Pi's IP address, log in as admin and follow the instructions for registering a new user as illustrated in *Figure 16*. After the process is finished, the user can log in to the Android application with the credentials, entered during the registration process. The user can only register the card to one user, but one user can have multiple tokens for identification.

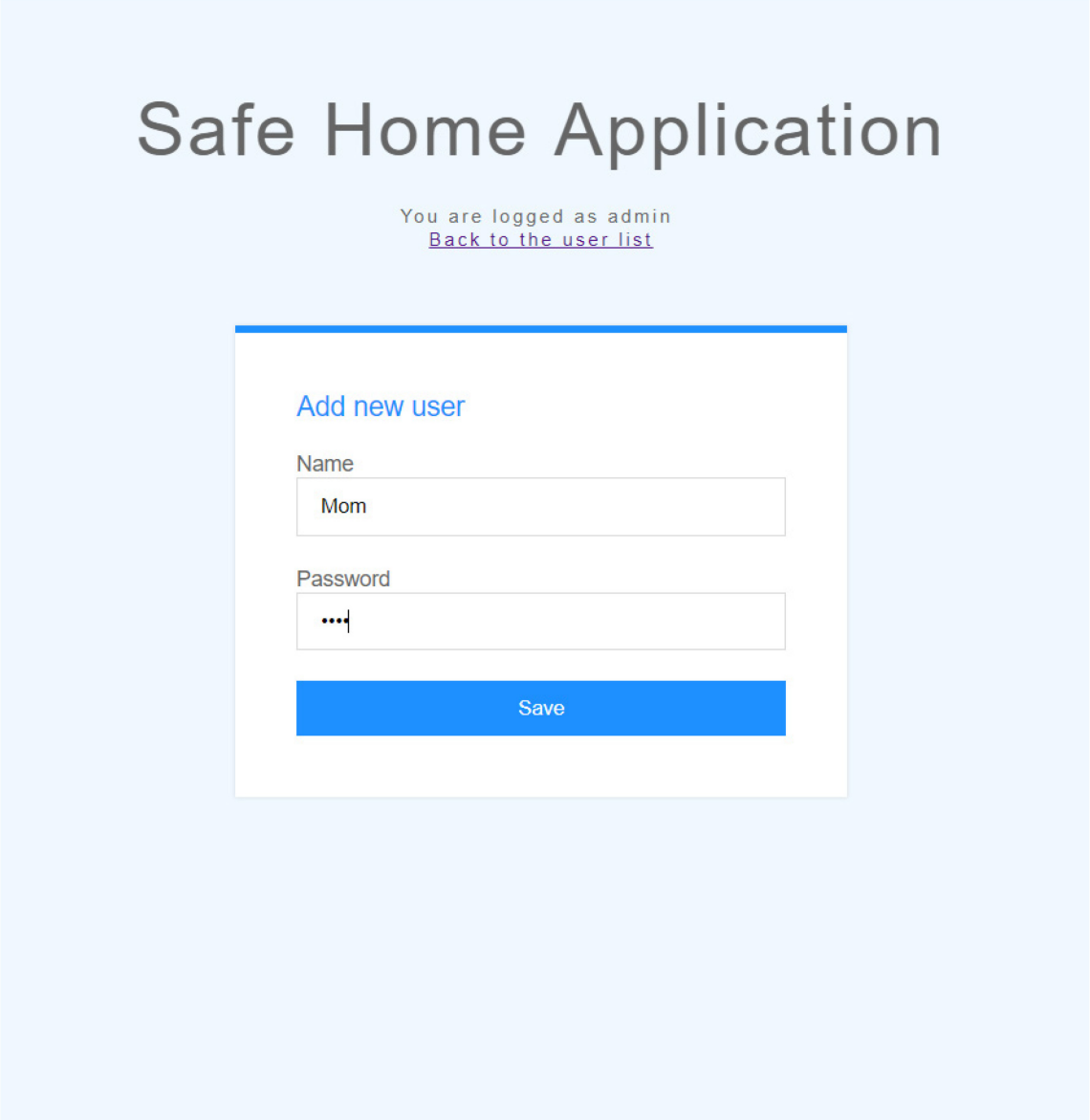

Figure 16. The new user registration process.

The user can test the application with turning the security module on and off with his or her RFID token for a period of at least half a minute. Blinking LED means that the application recognizes the token. The notifications from the Android applications mean,

that the user was successfully logged in the device, and the whole system was installed correctly and delivers messages about the detected motions. Now the user can get the information about intrusions in his or her mobile device as shown in *Figure 17*.

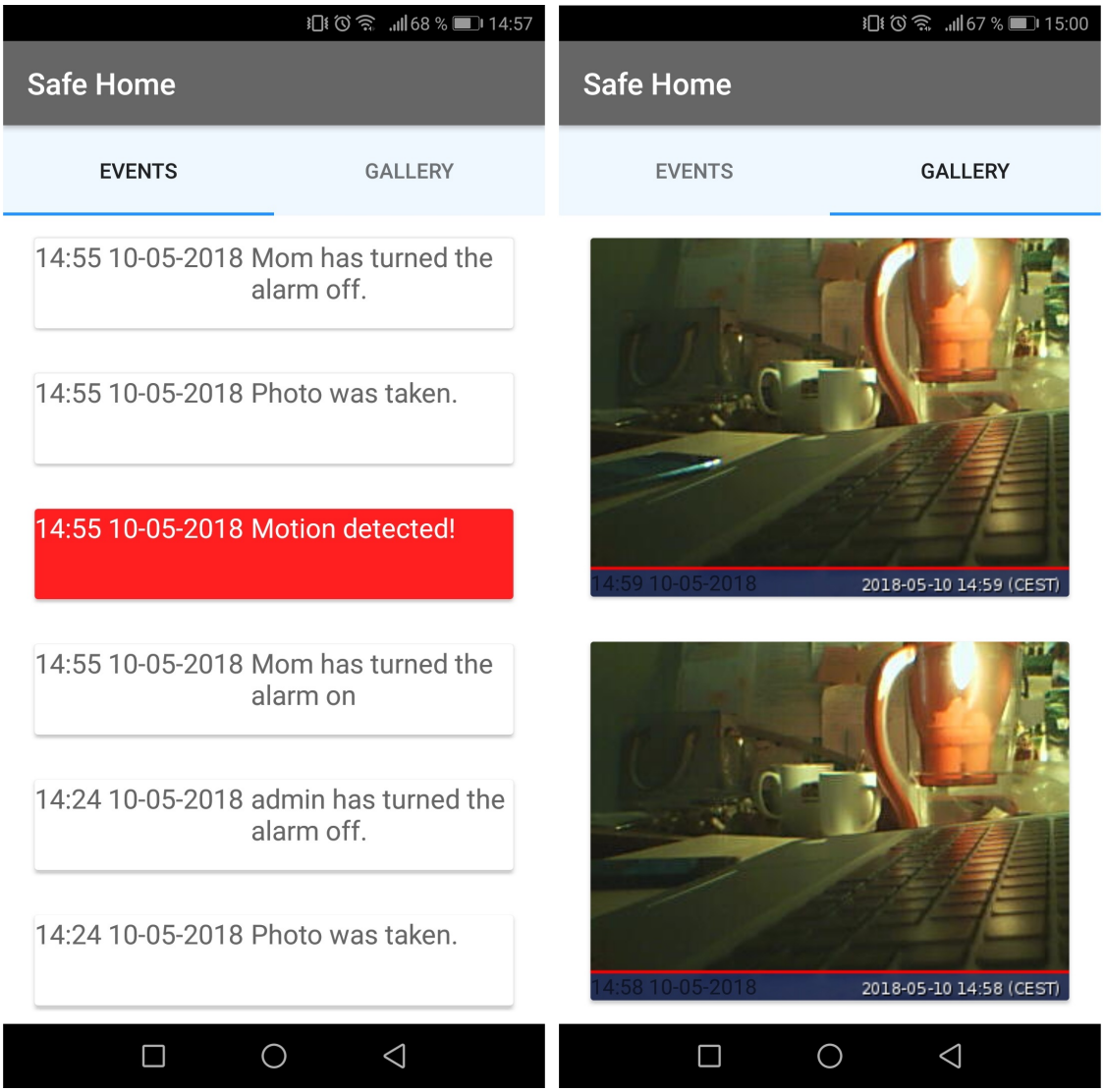

Figure 17. The Android application user interface.

4.2. Comparison with other systems

The idea to connect the camera, motion sensor and a mobile application has been implemented by a large number of commercial companies by the year 2018. The implementation has got the name of an IP camera or a network camera. The first network camera was invented in 1996 by Axis Communications [20]. At that time, there

was no mobile application for it, but the camera and its pictures could be accessed from the Internet. In 1999 was invented the first camera, which was capable of raising an alarm and contained software for managing video recordings with a Linux system by the company named Mobotix [22].

There are a couple of tutorials for Raspberry Pi available, which teach how to turn the computer into an IP camera enabling to use it as a home surveillance system. The tutorials mostly use the Motion [24] software, which allows the user to configure Raspberry Pi to record videos and pictures, to stream online and to save the video footage when the motion was detected. There also exists motionEyeOS [25], which is an operating system using the Motion at the core with an additional web front-end for more user-friendly control over the device. The software supports multiple cameras input or even connecting multiple Raspberry Pi computers into a network. However, the software does not include mobile applications nor user identity management.

IP camera models available on the market also offer a variety of options for surveillance cameras. The price point starts at around 30USD per camera. They usually contain a motion detector and auto record upon the detected motion. Most of them can be accessed remotely from a smartphone, using the manufacturer's mobile applications or an alternative third-party software. Some of them save data on an SD card, some connect to cloud, some provide an option of real-time streaming on a mobile device.

Let us review an IP camera Carneo HomeGuard WIFI, which is the best selling IP camera in the Czech Republic according to the biggest Czech e-shop of electronics Alza.cz [21]. It contains an HD camera with a lens of 290° horizontally and 100° vertically and a limited night vision option. It also has a motion sensor, sound detector, microphone and speaker. It can connect to the Internet via an Ethernet cable or WiFi. The manufacturer provides a mobile application for devices with Google Android and Apple iOS operating systems and a web application. The application allows the user to move the camera and to use it for the two-way communication. It saves the data on a microSD card, and the device's cost is more than 80USD.

Another top product from the Czech market is Hikvision DS-2CD2020F-I (4mm) IP camera. It costs over 135USD and offers full HD videos, integrated motion sensor and

connectivity to an alarm system. It also has microphone and speaker and sends the data via the network to external storage. The camera belongs to standard NVR (Network video recorder) type of cameras and can be controlled by a large number of third-party software, including web applications, but the manufacturer does not offer their application for the remote access.

Safe Home solution overall hardware cost is about 60USD. Unlike the reviewed products and all other available IP cameras, it uses the RFID identity tokens for controlling the system and identifying the users but lacks the two-way communication option. The newer versions of Raspberry Pi can also connect to the Internet via WiFi, but Safe Home does not provide a web application for accessing pictures.

The biggest difference between available solutions and Safe Home is that it is designed for monitoring when the user locks the doors to his or her private space and does not expect any guests, whereas other systems are designed to constantly monitor space, and often not only indoors. Sometimes they explicitly present the product as a tool for monitoring people, including so-called baby monitors. Our application cannot be used to make records about the users without their knowledge. Our system is also unique in connecting the identity management and home surveillance into one piece of software.

5. Conclusion and future work

In this thesis, we successfully achieved all initially set goals. We have created a home alarm system, which uses simple and affordable components: best selling single-board Raspberry Pi computer, a USB camera, an infrared motion sensor and an RFID reader. The architecture of our solution is flexible and easily modifiable because the components of the system are logically separated and are loosely coupled.

The system detects an intrusion on its Raspberry Pi security module with the motion sensor and takes pictures with the USB camera. It delivers notifications to the users via the Internet to the Android application. Its modes are controlled with users' RFID identity tokens, which can be, for example, ISIC student cards or similar. Overall, this home alarm system is a ready-to-use solution for securing the individual's home.

5.1. Possible extensions

The home alarm systems are popular and widely used, and there is high competition on the market. Safe Home has all the potential to be enhanced to the level of commercial products with some points, which need to be further developed.

To compete with existing products, our project needs an implementation of communication security, such as usage of HTTPS protocol for the network exchange and the requests' body encryption. In our project, the parts of the code responsible for forming the requests are separated into a library, which makes this enhancement possible and with minimum changes.

For better user experience, the remote access application needs to support different platforms, such as Apple iOS, and to be accessible on the web or as a desktop application on a personal computer. The API for communication between the parts of the system is designed for such an extension.

The remote access itself also can provide a larger number of options, such as remote switching the system's mode on and off or remote picture taking, or a real-time streaming. That is also possible thanks to a flexible communication API.

The design of the system also plays a huge role in the market. The security module needs to look attractive, be durable and resistant to physical damage. That point, though, does not depend on the software implementation.

The home alarm systems also often include several cameras, which are connected to the same managing software. The current implementation of Safe Home's only limitation as a number of USB ports on Raspberry Pi computers for connecting further USB cameras, otherwise it is ready to communicate with more than one set of motion sensor and camera.

Based on the above discussion of extensions, we can conclude, that our home alarm system has the great potential for future productive work, which can eventually make Safe Home a competitive customised solution for home security.

Bibliography

1. “Raspberry Pi.” Retrieved July 8, 2017, from https://www.raspberrypi.org/
2. “The Android Story.” Retrieved July 8, 2017, from https://www.android.com/history/
3. “About Us.” Retrieved July 8, 2017, from https://www.raspberrypi.org/about/
4. “Company History.” Retrieved July 8, 2017, from https://www.broadcom.com/company/about-us/company-history/
5. “Raspbian.” Retrieved July 8, 2017, from https://www.raspbian.org/downloads/raspbian/
6. “MFRC522. Standard performance MIFARE and NTAG frontend” (2016). Retrieved July 8, 2017, from https://www.nxp.com/docs/en/data-sheet/MFRC522.pdf
7. “Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content.” Retrieved July 10, 2017, from https://tools.ietf.org/html/rfc7231
8. “Application Fundamentals.” Retrieved April 3, 2018, from https://developer.android.com/guide/components/fundamentals
9. “Java Powers Our Digital World.” Retrieved April 3, 2018, from https://go.java/index.html
10. “Gartner Says Worldwide Sales of Smartphones Grew 9 Percent in First Quarter of 2017” (2017). Retrieved April 4, 2018, from https://www.gartner.com/newsroom/id/3725117
11. “Desktop vs Mobile vs Tablet Market Share Worldwide.” Retrieved April 3, 2018, from http://gs.statcounter.com/platform-market-share/desktop-mobile-tablet
12. “Introduction to Client/Server Architecture.” Retrieved April 3, 2018, from https://docs.oracle.com/cd/B13789_01/server.101/b10743/dist_pro.htm

13. Gosling, James; Joy, Bill; Steele, Guy; Bracha, Gilad; Buckley, Alex (2014). "The Java® Language Specification (Java SE 8 ed.)" PDF.

14. "TIOBE Index for May 2018." Retrieved May 15, 2018, from https://www.tiobe.com/tiobe-index/

15. "Mediator design pattern." Retrieved April 11, 2018, from https://sourcemaking.com/design_patterns/mediator

16. Jones, Capers. 1996. "Software Defect-Removal Efficiency", IEEE Computer, April 1996.

17. McConnell, Steve. 2004. "Code Complete, Second Edition". Redmond, WA: Microsoft Press.

18. Miller, Paul. 2017. "Raspberry Pi sold over 12.5 million boards in five years." Retrieved April 15, 2018, from https://www.theverge.com/circuitbreaker/2017/3/17/14962170/raspberry-pi-sales-12-5-million-five-years-beats-commodore-64

19. "Java Servlet Technology." Retrieved April 16, 2018, from http://www.oracle.com/technetwork/java/index-jsp-135475.html

20. "The Axis Story." Retrieved May 5, 2018, from https://www.axis.com/about-axis/history

21. "Carneo HomeGuard WIFI." Retrieved May 6, 2018, from https://www.alza.cz/carneo-homeguard-wifi-d5253636.htm

22. "An Incredibly Unboring History of IP Cameras." Retrieved May 6, 2018, from https://www.protectamerica.com/home-security-blog/tech-tips/draft-an-incredibly-unboring-history-of-ip-cameras-draft_11713

23. "A Very Short History Of The Internet Of Things." Retrieved May 11, 2018, from https://www.forbes.com/sites/gilpress/2014/06/18/a-very-short-history-of-the-internet-of-things/

24. "About Motion." Retrieved May 12, 2018, from https://motion-project.github.io/

25. “motionEyeOS.” Retrieved May 14, 2018, from https://github.com/ccrisan/motioneyeos

List of figures

List of abbreviations

IoT	Internet of Things
RFID	Radio-frequency identification
GPIO	General-purpose input/output
PIR	Passive infrared
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
JSON	JavaScript Object Notation
RPC	Remote procedure call
REST	Representational state transfer
SOAP	Simple Object Access Protocol
HTML	HyperText Markup Language
XML	Extensible Markup Language
API	Application Programming Interface