Maciej Bukowski

THE APPLICATION OF NFC TECHNOLOGY TO CREATE A MOBILE CLOCK-IN/OUT INSPECTION DEVICE.

Case: SPR

Thesis
CENTRIA UNIVERSITY OF APPLIED SCIENCES
Information Technology
March 2018

ABSTRACT

Centria University of Applied Sciences	**Date** March 2018	**Author** Maciej Bukowski
Degree programme Degree programme in Information Technology		
Name of thesis THE APPLICATION OF NFC TECHNOLOGY TO CREATE MOBILE CLOCK-IN/OUT INSPECTION DEVICE. Case: SPR.		
Supervisor Dr Grzegorz Szewczyk	**Pages** 33 + 2	
Supervisor Dr Grzegorz Szewczyk	**Working-life supervisor** Jari Isohanni	

Access mode control in many companies is a basic data necessary for payment purposes. Though computers are present in our everyday life, no low cost and easy to use access mode solution exists. To date, employees at the second hand shops of the Finnish Red Cross (SPR) manage this by signing on paper lists. Afterwards, somebody has to spend a lot of time and energy preparing reports.

The aim of the thesis is to create and develop an application for an Android device which will be a part of a system for access mode control. The application will cooperate with the Firebase database and Web Management Panel. This application will allow the collection of data about the time employees start and end their working day. This data will be saved to the database, from where it will be accessible via the Web Management Panel, which will allow the generation of reports for any given period of time, without the obligation to re-calculate everything. Another very important feature of this system will be that it is quite simple to develop and maintain. In addition, it will be extremely reasonable to install at the destination locations, due to the fact that it is wireless and does not need any particular terminals. The terminal can be almost any Android mobile device, which is consistent with the requirements.

Key words
Access mode control, NFC, TAG

ABSTRACT

Centria University of Applied Sciences	**Data** Marzec 2018	**Autor** Maciej Bukowski
Kierunek Informatyka Stosowana		
Temat pracy WYKORZYSTANIE TECHNOLOGII NFC DO BUDOWY PRZENOŚNEGO URZĄDZENIA KONTROLI CZASU PRACY. Przypadek: SPR.		
Instruktor Dr inż. Grzegorz Szewczyk	**Strony** 33 + 2	
Promotor Dr inż. Grzegorz Szewczyk	**Opiekun** Jari Isohanni	

Podstawą wypłacania wynagrodzeń pracowników najemnych jest zawsze ewidencja czasu pracy. Taka ewidencja jest również prowadzona w sklepach z artykułami używanymi, których właścicielem jest Fiński Czerwony Krzyż – Suomen Punainen Risti (SPR). Dotychczas, w tej instytucji ewidencja czasu pracy była prowadzona za pomocą papierowych list imiennych, na których pracownicy zaznaczali godzinę przyjścia do i wyjścia z pracy. Następnie, listy te były zbierane w okręgu działania SPR i znajdujące się na nich dane były wprowadzane do centralnego systemu ewidencji czasu pracy, co było procesem żmudnym i długotrwałym.

Celem tej pracy było zaprojektowanie i wykonanie aplikacji pracującej pod systemem operacyjnym Android, której zadaniem jest zautomatyzowanie ewidencji czasu pracy w Fińskim Czerwonym Krzyżu. Aplikacja współpracowałaby z bazą danych Firebase i Internetowym Panelem Administracyjnym. Aplikacja pozwoli na zbieranie danych o czasie rozpoczęcia i zakończenia pracy przez pracowników. Dane będą zapisywane do bazy danych skąd będą one dostępne dla Internetowego Panelu Administracyjnego, który pozwoli na generowanie raportów z dowolnego okresu czasu, bez konieczności żmudnego, ręcznego przetwarzania danych.

Inne bardzo ważne cechy systemu to względna prostota produktu i łatwość obsługi. Aplikacja pracująca pod popularnym systemem Android będzie mogła komunikować się z centralną ewidencją czasu pracy SPR w sposób bezprzewodowy, a także nie będzie wymagać specjalistycznego urządzenia końcowego. Terminalem może być dowolne urządzenie pracujące pod kontrolą systemu operacyjnego Android, spełniające wymagania opisane w pracy. Przy takich założeniach koszty instalacyjne i eksploatacyjne aplikacji będą bardzo niskie, co z punktu widzenia klienta stanowi ważny argument.

Słowa kluczowe
Kontrola czasu pracy, metka, NFC

ACRONYMS

1. NFC – Near Field Communication
2. SPR – Suomen Punainen Risti (Finnish Red Cross)
3. DAO – Data Access Object

ABSTRACT [English]

ABSTRACT [Polish]

ACRONYMS

CONTENTS

1 INTRODUCTION

SPR - Suomen Punainen Risti (fin. Finnish Red Cross) “The Finnish Red Cross is one of the largest civic organizations in Finland. The objective of the Finnish Red Cross is to help those who need it most both in Finland and abroad (Suomen Punainen Risti).” The aim of this project is to develop a system to calculate the working time of employees at the second hand shops of the Finnish Red Cross. This solution will replace the current usage of paper attendance lists, which are very inconvenient to use, unreliable and time consuming when preparing reports for payment purposes. The advantages of this system will be ease of use, the ease and low cost of installation (does not require any wires and external services, only 3G or Wi-Fi internet), and portability – the collection devices are totally wireless so it is not a problem to change the location of a shop without incurring any additional costs.

The system will consist of a web service for data management and report generation, an Android application for collecting data about the working time of employees using NFC, an online database to store data about employees, their assignment to particular shops, and overtime by each employee, and personal identification cards containing an NFC tag for each employee.

The SPR mobile application will transform any mobile device (smartphone / tablet) with an Android Operating System into a terminal used for collecting data about the employee’s working time. The application will work in kiosk mode, locked on the screen. The application will provide information about the current status of the employee and will allow the employee to change the status.

The SPR web service will allow managers to manage users’ personal data, assign and manage personal ID numbers, view and manage data about overtime by employees, and compile periodic reports.

Development of the SPR mobile application is commissioned by Centria University of Applied Sciences Research and Development in Kokkola, Finland (APPENDIX 1).

2 NFC COMMUNICATION

2.1 Overview

Near Field Communication (NFC) was invented and developed by an association of Nokia, Sony and Philips established in 2004 called the NFC Forum. Nowadays the number of NFC Forum members is much bigger and still growing. Almost all the bigger digital technology companies can be found on the list of members. (NFC Forum) Near Field Communications is a contactless means of communication for short distances of up to 10 centimetres.

> “Near field communication, or NFC for short, is an offshoot of radio-frequency identification (RFID) with the exception that NFC is designed for use by devices within close proximity to each other. Three forms of NFX technology exist: Type A, Type B, and FeliCa. All are similar but communicate in slightly different ways. FeliCa is commonly found in Japan (Square, Inc.).”

Near Field Communication technology can be used in three possible modes: Tag Reader / Writer, Peer to Peer and Card Emulation.

2.2 Causes of NFC utilization

For the purposes of this project a passive version of NFC was employed, which uses an NFC tag. A tag is a kind of stick that consists of a coil and a microprocessor. Passiveness means that the tag does not contain any energy supply; energy is generated by the coil. The other advantage of this solution is that tags are very cheap, so their cost can practically be omitted when calculating system costs. Some of the alternative solutions are much more expensive. Another possible solution was to use a barcode, but it is more time consuming to using barcodes due to the necessity of taking a sharp photo of the barcode for readout. The advantage of an NFC tag in this case is that the orientation presented to the device containing the employee ID tag does not matter unlike the above-mentioned barcodes. According to one of the client requirements to be able to use this application on most modern smartphones, it was not possible to use any of the other RFID solutions, because they are not commonly implemented in smartphones. In light of these advantages and features of NFC tags in this case, it was decided together with the client to use this technology for identifying employees.

3 APPLICATION DESIGN

Application design is a very complex process which leads to the development of an application responding to clients' requirements. Application design consists of elicitation of requirements, identifying use cases, class selection process, designing user interfaces, projecting flow of data inside the application and exchange of data with external systems if needed. Software engineers basing on properly specified requirements and eligible behaving of the application are able to design classes, user interfaces and flow of data inside the application considering the need of interaction with external systems. Previously mentioned steps are also mandatory to properly construct tests.

3.1 Requirements

Recognition of requirements is an important part of software design. Very well specified requirements are a base for the whole design and development process. More time spent on the identification and deep analysis of requirements leads to software that suits the client needs much better. Tools such as interviews with the client and studies of similar software currently existing on the market can be used to specify the requirements. As a result, requirements can be determined in detail and tell the developers precisely what the application should do and what the limitations should be. (Sommerville, 2011, pp. 88-111)

3.1.1 Functional requirements

FR001	The application shall be able to display the user's current work status.
FR002	The application shall show a screen with a current status indicator and two buttons for choosing an action.
FR003	The application shall be able to display a notification when an action cannot be executed.
FR004	The application shall show a clock and information about the required action from the user on main screen.
FR005	The user shall choose one of the two given directions.
FR006	The application shall show a notification with information on which action was executed.
FR007	The application shall identify the user by the NFC tag number.
FR008	The application shall use the device ID to identify in which shop this device is working in order to exchange data with the database.

3.1.2 Non-functional requirements

NFR01	The application shall show a screen with buttons to choose the direction of logging for 10 seconds.
NFR02	The application shall show a screen with notification about the direction sent to the database for 3 seconds.
NFR03	The application should have access to the Internet for at least 20 minutes per month.
NFR04	The database of one shop should be less than 10 MB.
NFR05	The application shall be able to work offline.

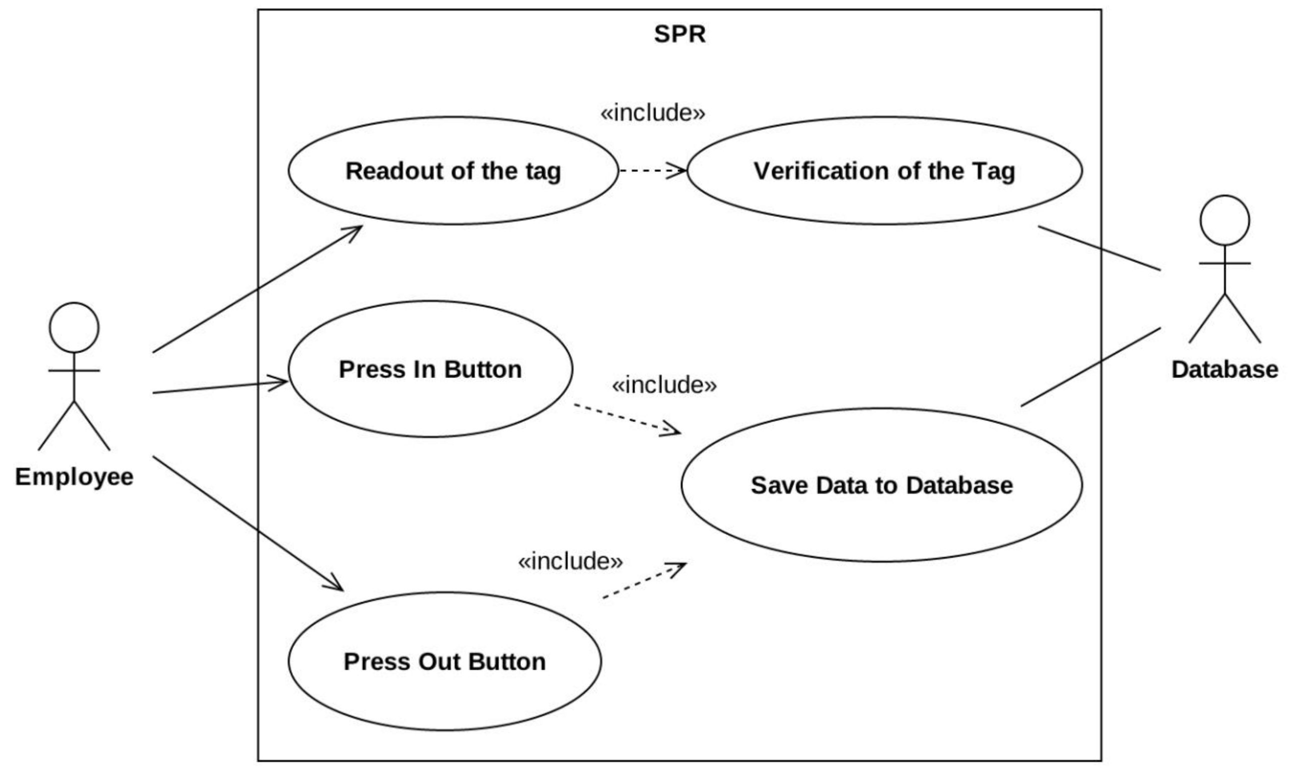
3.1.3 System requirements

SR001	The application shall work on platforms above Android 5.0.
SR002	The application shall use an NFC module.
SR003	The application shall work on devices with a screen with a diagonal of at least 5.7 inches.

3.2 Use cases

Use cases are used to illustrate the actions that the user has to perform to achieve the intended goal. Interactions with other actors in the system are also shown in the use case. This tool also allows software engineers to present all possible exceptions and, something that is even more valuable, what actions will be performed by the system to solve the undesirable situation. (Sommerville, 2011, pp. 124-126)

3.2.1 Diagrams



GRAPH 1. Use cases diagram

3.2.2 Descriptions

Use Case ID:	**001**
Use Case Name:	**Readout of the tag**
Actors:	Employee
Description:	
Trigger:	
Pre-conditions:	
Normal Flow:	The user shows the NFC tag to the device; the device reads the tag and delivers the event to the application.
Alternative Flows:	
Exceptions:	
Post-conditions:	Number of the tag is known.

Use Case ID:	**001**
Use Case Name:	**Readout of the tag**
Includes:	
Frequency of Use:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

Use Case ID:	**002**
Use Case Name:	**Verification of the tag**
Actors:	Database
Description:	
Trigger:	NFC event from operating system.
Pre-conditions:	Correctly read NFC tag
Normal Flow:	The application asks the database for data on the tag. The application displays the actual status of employee access and number of the tag. [**Exception:** Tag Does Not Exist], [**Exception:** Tag Out of Date]
Alternative Flows:	
Exceptions:	**Tag Does Not Exist:** Shows the number of the tag and short description. The application returns to the idle state. **Tag Out of Date:** Shows the number of the tag and short description. The application returns to the idle state.
Post-conditions:	Tag is valid.
Includes:	
Frequency of Use:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

Use Case ID:	**003**
Use Case Name:	**Press “In” Button**
Actors:	Employee
Description:	
Trigger:	Verification of the tag
Pre-conditions:	Read and verified tag
Normal Flow:	User presses the “In” button. [**Exception:** User Not Responding]
Alternative Flows:	
Exceptions:	**User Not Responding:** The application returns to the idle state.
Post-conditions:	
Includes:	
Frequency of Use:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

Use Case ID:	**004**
Use Case Name:	**Press “Out” Button**
Actors:	Employee
Description:	
Trigger:	Verification of the tag
Pre-conditions:	Read and verified tag
Normal Flow:	User presses the “Out” button. [**Exception:** User Not Responding]
Alternative Flows:	
Exceptions:	**User Not Responding:** The application returns to the idle state.
Post-conditions:	
Includes:	
Frequency of Use:	
Special Requirements:	
Assumptions:	

Use Case ID:	**004**
Use Case Name:	**Press “Out” Button**
Notes and Issues:	

Use Case ID:	**005**
Use Case Name:	**Save Data to Database**
Actors:	Database
Description:	
Trigger:	
Pre-conditions:	
Normal Flow:	The application sends the log (tag number, access mode, status update) to the database. [**Exception:** New State is the Same as Actual]
Alternative Flows:	
Exceptions:	**New State is the Same as Actual:** A log is sent with opposite access mode to the database. A log is set with correct access mode.
Post-conditions:	Access control data added to the database.
Includes:	
Frequency of Use:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

3.3 Class selection process

For this purpose the author used “natural language analysis” invented by (Abbot, 1983) and popularized by Grady Booch. This method relies on underlining and mapping different parts of speech to object model components, therefore, as the use cases descriptions are quite poor in this case, focus was placed only on classes and attributes that were mapped according to rules from nouns.

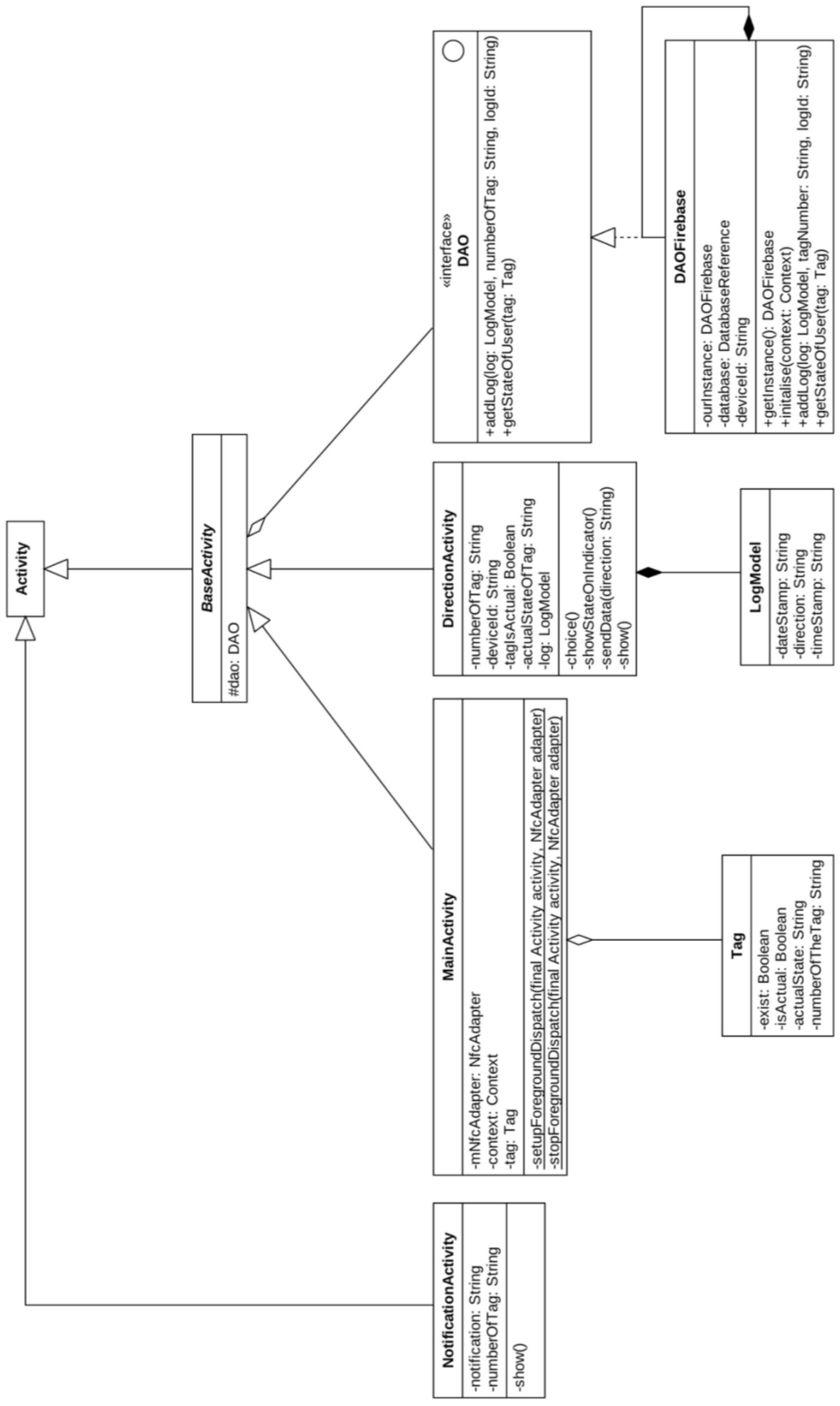
TABLE 1. Lexical analysis of requirements

Use case number	Use case name	Use case description
002	Verification of the Tag	The application asks the database for data on the tag. The application displays the actual status of employee access and the number of the tag.
003	Press “In” Button	User presses the “In” button.
004	Press “Out” Button	User presses the “Out” button.
005	Save Data to Database:	The application sends the log (number of the tag, access mode, and status update) to the database.

Observed nouns: Application, Database, Tag, Application, Status of access of employee, Number of the Tag

As a result of the preliminary analysis, the following classes were received: Application, LogModel, Tag, Database as well as the attributes: Status of access of employee, Number of the Tag.

3.4 Class diagram



GRAPH 2. Class diagram

3.5 User interface design

During interviews with the client it was specified that this application will have a UI, based on three views: main, one used to choose the access mode, and one used for notifications (APPENDIX 2). Images of the screens are presented below.

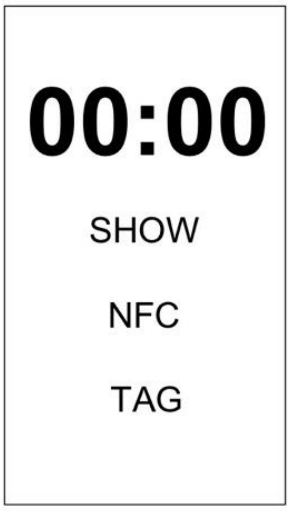


FIGURE 1. Main screen

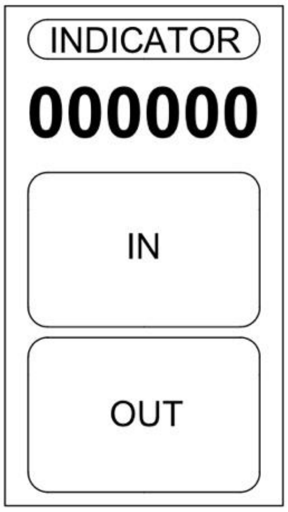


FIGURE 2. Direction screen

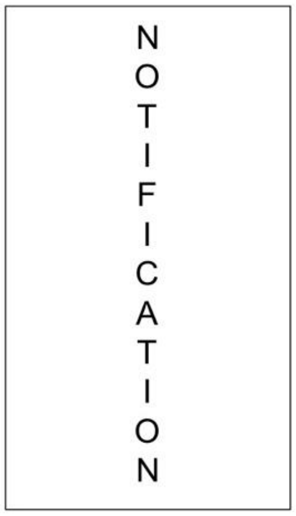


FIGURE 3. Notification screen

The main screen of the application – always shows the current time and information about action required to go further (FIGURE 1). The screen on which the current status of the employee and his/her tag number are shown and two buttons for selecting the action to start or finish work (FIGURE 2). The notification screen – on this screen all information is shown on executed actions like the changed current status of the employee; also used to show notifications about failures (FIGURE 3).

3.6 Flow of data between modules of the entire system

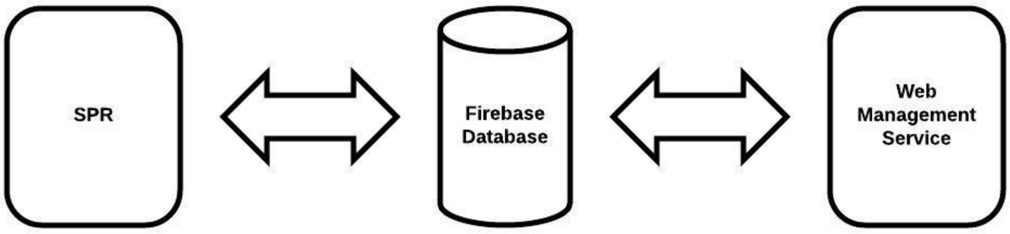


FIGURE 4. Flow of data in the entire system

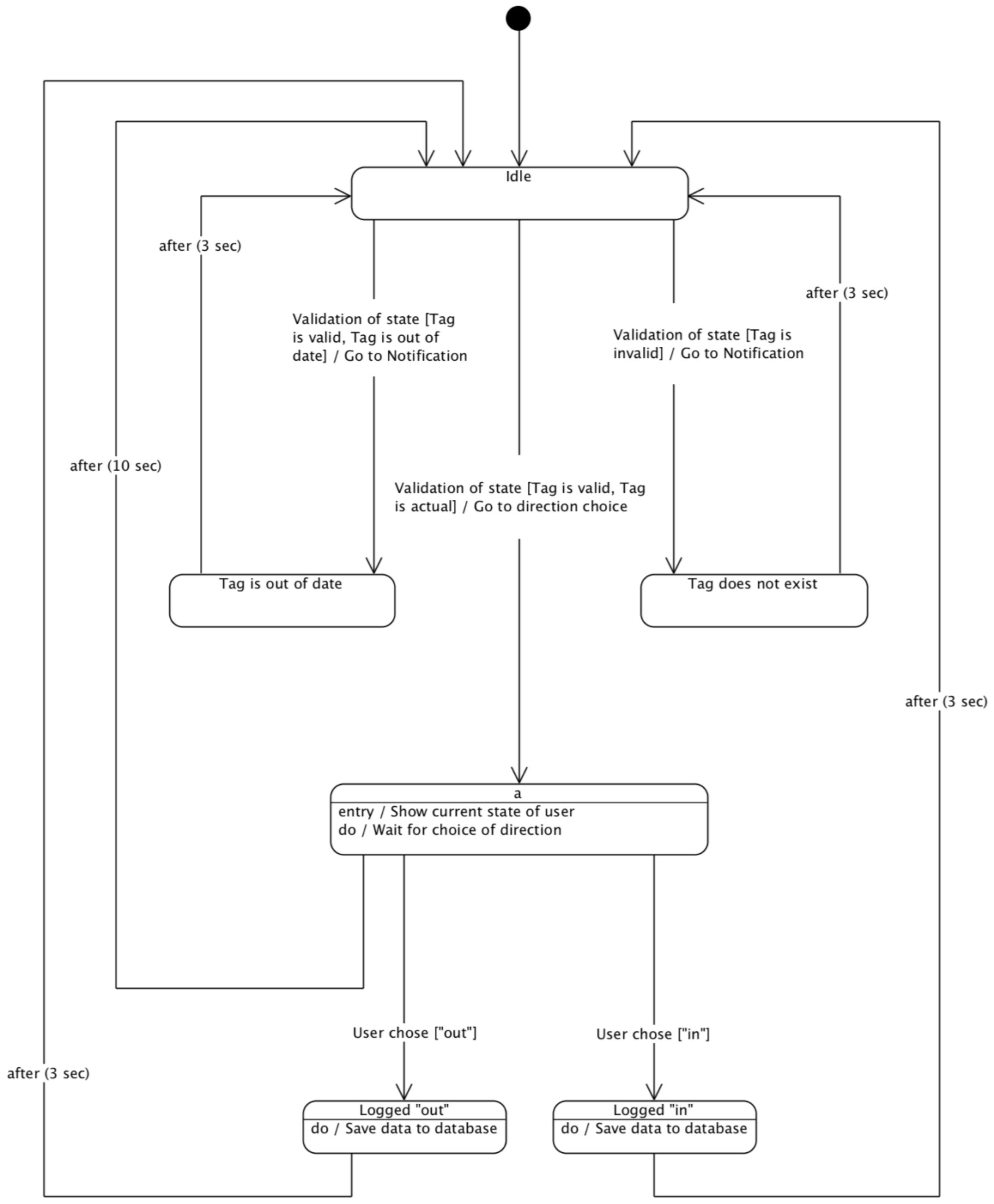
This system is to consist of the SPR application, the Firebase database and the Web Management Service. The Firebase database is at the core of the whole system, while the other components communicate with the database in both directions (FIGURE 4).

3.7 State diagram.

Software engineers can use state diagrams for the event-driven modelling of an application. *Event-driven modelling shows how a system responds to external and internal events* (Sommerville, 2011, p. 135). State diagrams consist of states, activities performed in these states, and events that cause transitions from one state to another. A state diagram does not show the flow of data in the application but may include some additional information about calculations performed in particular states. According to the fact that in state modelling, the number of states increases very quickly, in bigger system models some less important details should be hidden. (Sommerville, 2011, pp. 135-138)

TABLE 2. State diagram description

State name	State Description
Idle	Main state of application. Application is waiting for the tag to be presented by the user.
Direction choice	The application will save the data to the database. A notification will be displayed about successful changed user access mode to “out” for 3 seconds. After this time, the application will go into idle state.
Logged “out”	Application will save data to Database. And will display a notification about successful changed user access mode to “out” by 3 seconds. After this time application will go to the Idle state.
Logged “in”	The application will save the data to the database. A notification will be displayed about successful changed user access mode to “in” for 3 seconds. After this time, the application will go into idle state.
Tag does not exist	The application will display a notification that the tag does not exist for 3 seconds. After this time, the application will go into idle state.
Tag is out of date	The application will display a notification that the tag is out of date for 3 seconds. After this time, the application will go into idle state.


Idle
after (3 sec)
Validation of state [Tag is valid, Tag is out of date] / Go to Notification
Validation of state [Tag is invalid] / Go to Notification
after (3 sec)
after (10 sec)
Validation of state [Tag is valid, Tag is actual] / Go to direction choice
Tag is out of date
Tag does not exist
after (3 sec)
a
entry / Show current state of user
do / Wait for choice of direction
User chose ["out"]
User chose ["in"]
after (3 sec)
Logged "out"
do / Save data to database
Logged "in"
do / Save data to database

GRAPH 3. State diagram

4 APPLICATION DEVELOPMENT AND IMPLEMENTATION

4.1 Design patterns used

Design patterns are general solutions for problems that are quite common. Using a design pattern invented for a particular problem which occurs and should be solved guarantees that it is the most cost-effective solution. Design patterns can be mixed and developed without any limits. They give software engineers a skeleton of behaviour, so instead of inventing a similar solution again from scratch they can focus on fitting it to the current problem. This helps to save a lot of money, time for development and also to produce a code which will be more stable and less expensive to test.

> “The pattern is a description of the problem and the essence of its solution, so that the solution may be reused in different settings. The pattern is not a detailed specification. Rather, you can think of it as a description of accumulated wisdom and experience, a well-tried solution to a common problem. (…) Using patterns means that you reuse the ideas but can adapt the implementation to suit the system that you are developing. When you start designing a system, it can be difficult to know, in advance, if you will need a particular pattern. Therefore, using patterns in a design process often involves developing a design, experiencing a problem, and then recognizing that a pattern can be used. This is certainly possible if you focus on the 23 general-purpose patterns documented in the original patterns book. However, if your problem is a different one, you may find it difficult to find an appropriate pattern amongst the hundreds of different patterns that have been proposed (Sommerville, 2011, pp. 189-193).”

4.1.1 Bridge

The bridge is a design pattern used to separate abstraction and implementation and to allow them to change independently.

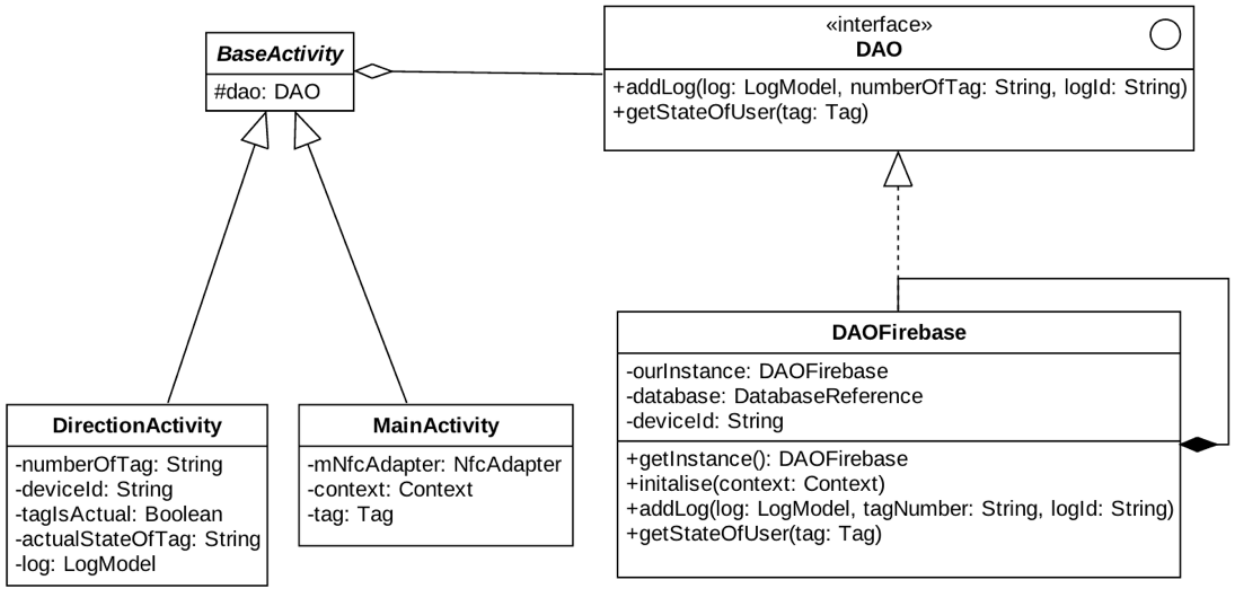

BaseActivity
#dao: DAO
«interface»
DAO
+addLog(log: LogModel, numberOfTag: String, logId: String)
+getStateOfUser(tag: Tag)
DirectionActivity
-numberOfTag: String
-deviceId: String
-tagIsActual: Boolean
-actualStateOfTag: String
-log: LogModel
MainActivity
-mNfcAdapter: NfcAdapter
-context: Context
-tag: Tag
DAOFirebase
-ourInstance: DAOFirebase
-database: DatabaseReference
-deviceId: String
+getInstance(): DAOFirebase
+initalise(context: Context)
+addLog(log: LogModel, tagNumber: String, logId: String)
+getStateOfUser(tag: Tag)

FIGURE 5. Bridge pattern (OODesign, Bridge)

The bridge pattern was used to separate the main part of application from the concrete implementation of the database. This will enable a way of developing this application in the future to change the present database solution to a completely different one and make the changes only in objects directly cooperating with the database without drowning the other parts of the application (FIGURE 5).

4.1.2 Singleton

The singleton pattern is a creational pattern to enable the creation of a single instance of a class object in the whole application. In order to achieve this goal, all constructors must be private, and the function of returning a singleton instance must be static (FIGURE 6).

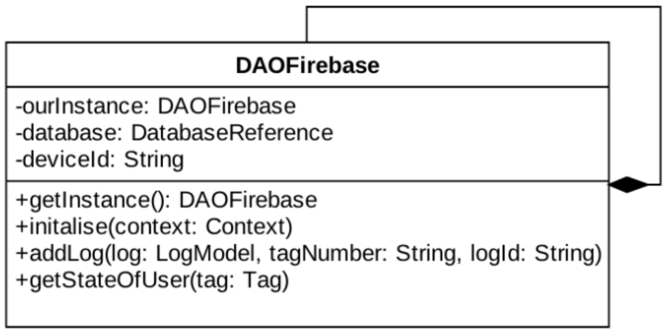

DAOFirebase
-ourInstance: DAOFirebase
-database: DatabaseReference
-deviceId: String
+getInstance(): DAOFirebase
+initalise(context: Context)
+addLog(log: LogModel, tagNumber: String, logId: String)
+getStateOfUser(tag: Tag)

FIGURE 6. Singleton pattern (OODesign, Singleton)

The singleton pattern was used to ensure that two or more objects of the class used for working with the database are not created. This may cause difficulties with the offline Firebase, which may decrease the maximal size of offline copy. The Firebase offline copy can be stored in the cache only if it is smaller than 10MB. If this size is exceeded, then the last data used are purged, which means that in offline mode users would not be able to check their status, and also any change of status without a known real current state may cause damage to the database. In addition, in the case of many references to the database if one reference were set up for a listener to retrieve data and the operation of saving data were occurring on another one, then it might cause a not entirely predictable effect. To avoid these consequences, the application has only one object with one reference to the database in the memory at one time.

4.1.3 Observer

The observer pattern is another design pattern used for communicating between objects in the application via events. It is achieved using two basic object types: observable – objects on which we would like to obtain some information, and observer – objects that are waiting for a notification about the change of state of the observable object. When the observable object changes its state, then it informs all its observers about this change (FIGURE 7).

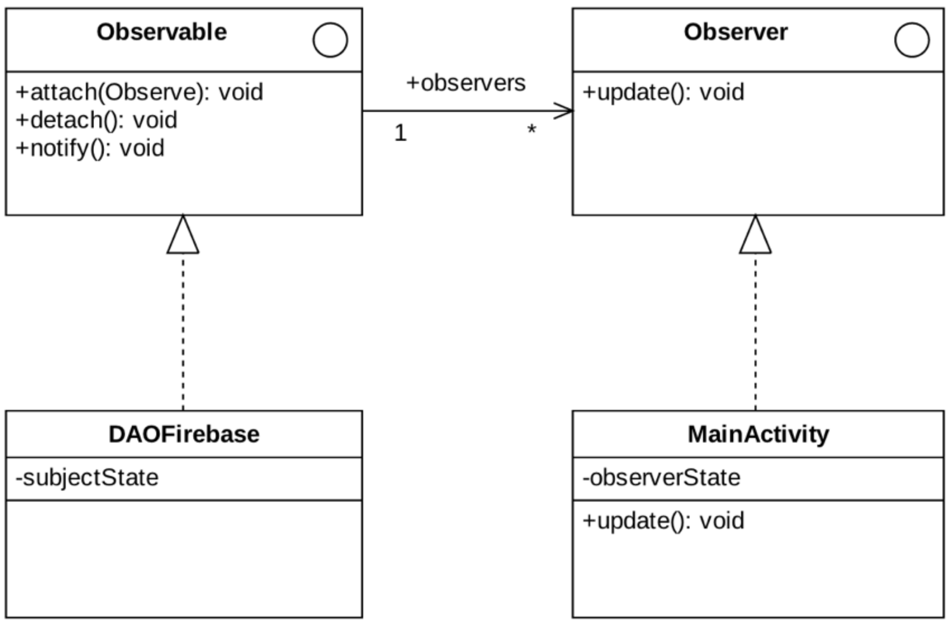


FIGURE 7. Observer pattern (OODesign, Observer)

Due to the fact that mostly operations on the database take quite a long time, they are carried out in asynchronous way. The advantage of this solution is that the main thread of the application is not frozen at the time of operations on the database. However, as a consequence it is not possible to simply pass data to another function, so it can be resolved by using an observer pattern, and the event is thrown when the operation on the database is completed. Then the application can work on this data just at the time they are delivered.

4.2 IDE and additional libraries used

Since the client required that this application be developed in Java for an Android operating system, it was decided to use Android Studio for this purpose due to the fact that the author had earlier used some IDEs supplied by Jet Brains. All their products are consistent and quite similar in use regardless of the supported technology.

The EventBus version 3.1.1 library designed by greenrobot was also used. It is a high performance, well optimized, very popular, very simple-to-use implementation of the observer pattern (GreenRobot, brak daty).

4.3 Database operations

```
public class LogModel
{
     @PropertyName("datestamp")
     private Integer dateStamp;
     @PropertyName("direction")
     private String direction;
     @PropertyName("timestamp")
     private String timeStamp;

     ...
}
```

SOURCE CODE 1. PropertyName annotations in the code

Adding data from log object to the database is realized by passing a LogModel type object to Firebase's setValue(Object) function. This function converts it to JSON using Firebase's "PropertyName" descriptions in the Log Model class (SOURCE CODE 1) and it is appended to the specified node.

The Firebase database was supplied by the client, so there was no opportunity to change the structure, as the Web Management Service was almost complete and any changes in the structure would have caused errors. This is described in more details below in the Observations section. Please see the diagram of the structure of the supplied database in FIGURE 8. Three dots mean that the object on the right side can be multiplied many times.

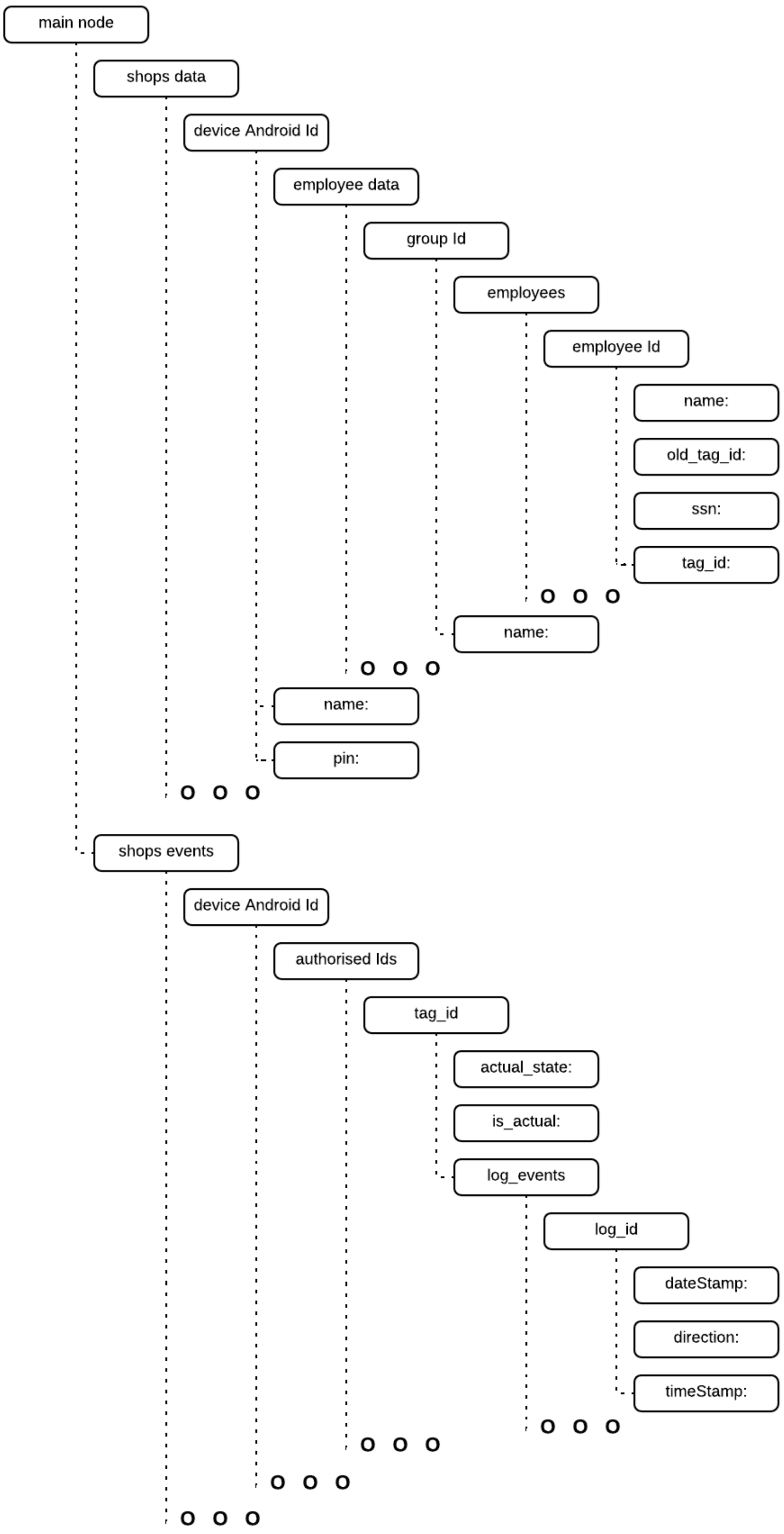


FIGURE 8. Database's structure

5 APPLICATION TESTS

Since this application does not contain any complicated logic, no unit tests were prepared. Instead, tests were carried out using all possible scenarios. This table contains all the scenarios performed with adopted assumptions and obtained results (TABLE 3). To sum up, all of the above tests were carried out correctly. In addition, the acceptance tests were performed with the client's participation. The application fulfills all the client requirements and expectations (Centria Univerity of Applied Sciences Research & Development).

TABLE 3. Assumptions adopted and results obtained in the tests

Tag exists	Tag is current	Current status	Performed action	Expected result	Test result
no	-----------	--------	-----------	Notification that tag does not exist	PASS
yes	no	--------	-----------	Notification that tag is out of date	PASS
yes	yes	in	nothing	After 10 seconds application will go back to initial state	PASS
yes	yes	in	change access mode status to “out”	Notification that change of access control status performed correctly. Added proper records to database.	PASS
yes	yes	out	change access mode status to “in”	Notification that change of access control status performed correctly. Added proper records to database	PASS
yes	yes	in	change access mode status to “in”	Notification that change of access control status performed correctly. Added proper records to database	PASS
yes	yes	out	change access mode status to “out”	Notification that change of access control status performed correctly. Added proper records to database	PASS

5.1 Release notes

There are no issues affected work flow of application, which should be included to release notes.

5.2 Conclusions on applied database solution and concept of development

Usage of the Firebase database was one of the client’s requirements. However, during the development of this application it was noticed that it would be a good idea for the future to change the Firebase database to another solution for several reasons. Firstly, Firebase is a “server-less” solution, which means that the whole code to manage and maintain the database has to be written in the application. This would entail that all changes in the structure of the database would bring with them compulsory updating of the application. Also, independently, it would cause the necessity for changes to the Web Management Panel. Secondly, Firebase downloads all subtrees on load, which means that the data are not downloaded according to needs. Thirdly, another of the client's requirements was that the application should work properly and without breaks even without an Internet connection. For this purpose, in the Firebase case, the copy of the data on the device has to be smaller than 10 MB. After exceeding the limit, more recent data will be removed from the device after being synchronized with the server. This means that, in the event of a loss of Internet connection, the offline database on the device will work on non-current data, which can cause serious mistakes in the database. Therefore, it is the duty of the system manager/administrator to frequently generate reports and remove data from the database.

6 MANUAL

6.1 System requirements

1. Operating system: Android 5.0 or newer.
2. NFC module.
3. Internet connection.
4. Screen with diagonal of 5.7 inches or greater.

6.2 Installation manual

1. Make sure that your device meets all the requirements.
2. Before you start the installation, please find out the Android ID of your device (for this purpose you can use one of the free applications from Google Play Store). It will be necessary to add your device to the shop in the Web Management Panel.
3. The application is delivered as an installable file with an .apk extension. Copy this file to your device.
4. Open file manager on your phone, go to the directory with the copied file.
5. Select the application, and choose "install". Afterwards accept all permissions required by the application.
6. Enable Internet connection, NFC module and Kiosk Mode.
7. Run the application.
8. Pin the application on the screen in Kiosk Mode.
9. Congratulations! The application is ready for use as soon as you have completed the configuration of the store unit in the Web Management Panel.

6.3 User manual

Every time the application is idle then you can check the current time on the clock (1). To check or change your current access mode status, please show your tag. The color of the indicator (2) represents your current access mode status: red – "not working", green – "working". You can see the number of your tag (3). You can change your status by choosing one of the direction buttons (4). If you do not choose either of them, then after 10 seconds the application will return to the beginning state. If you made a selection, then you will see screen (5) with information that the change of status was correct.

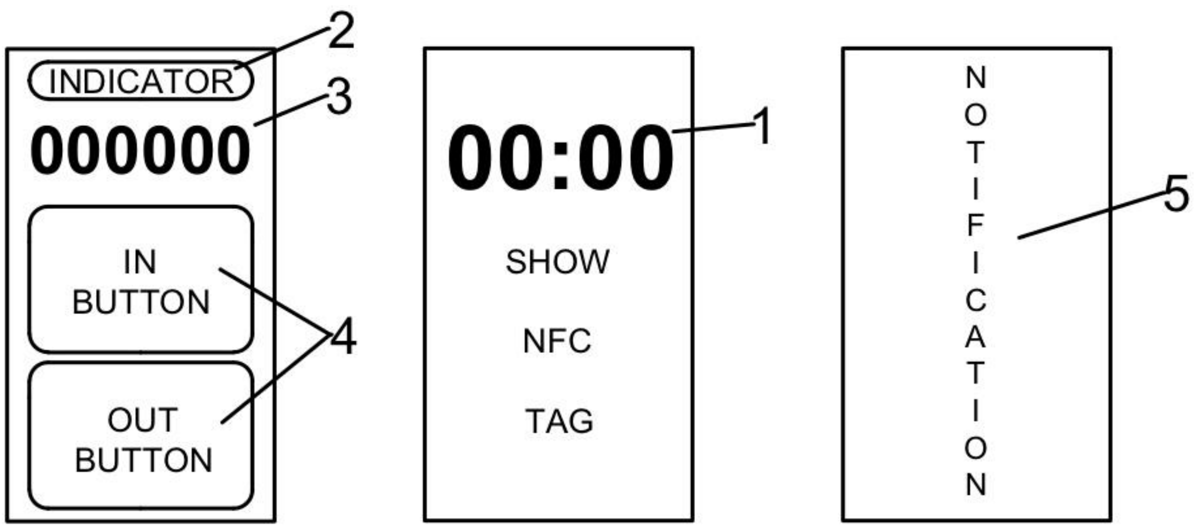


FIGURE 9. Screens in the application

REFERENCES

Abbot, R. J. (1983). Program Design by Informal English Descriptions. *Research Contributions*. Retrieved December 1, 2017, from https://greenbay.usc.edu/csci577/fall2009/site/coursenotes/ep/Program%20Design%20by%20Informal%20English%20Descriptions,%20Russell%20Abbott.pdf

Centria Univerity of Applied Sciences Research & Development. (n.d.). Retrieved 05 15, 2018, from https://tki.centria.fi/project/Xnet%20–%20digitalisaatiota%20edistävät%20verkkosovellukset/1121/news/4345

GreenRobot. (n.d.). Retrieved 03 2018, 01, from https://github.com/greenrobot/EventBus

NFC Forum. (n.d.). *NFC Forum*. Retrieved 05 18, 2018, from NFC Forum Members: https://nfc-forum.org/about-us/our-members-2/

OODesign. (n.d.). *Bridge*. Retrieved 03 01, 2018, from https://www.oodesign.com/bridge-pattern.html

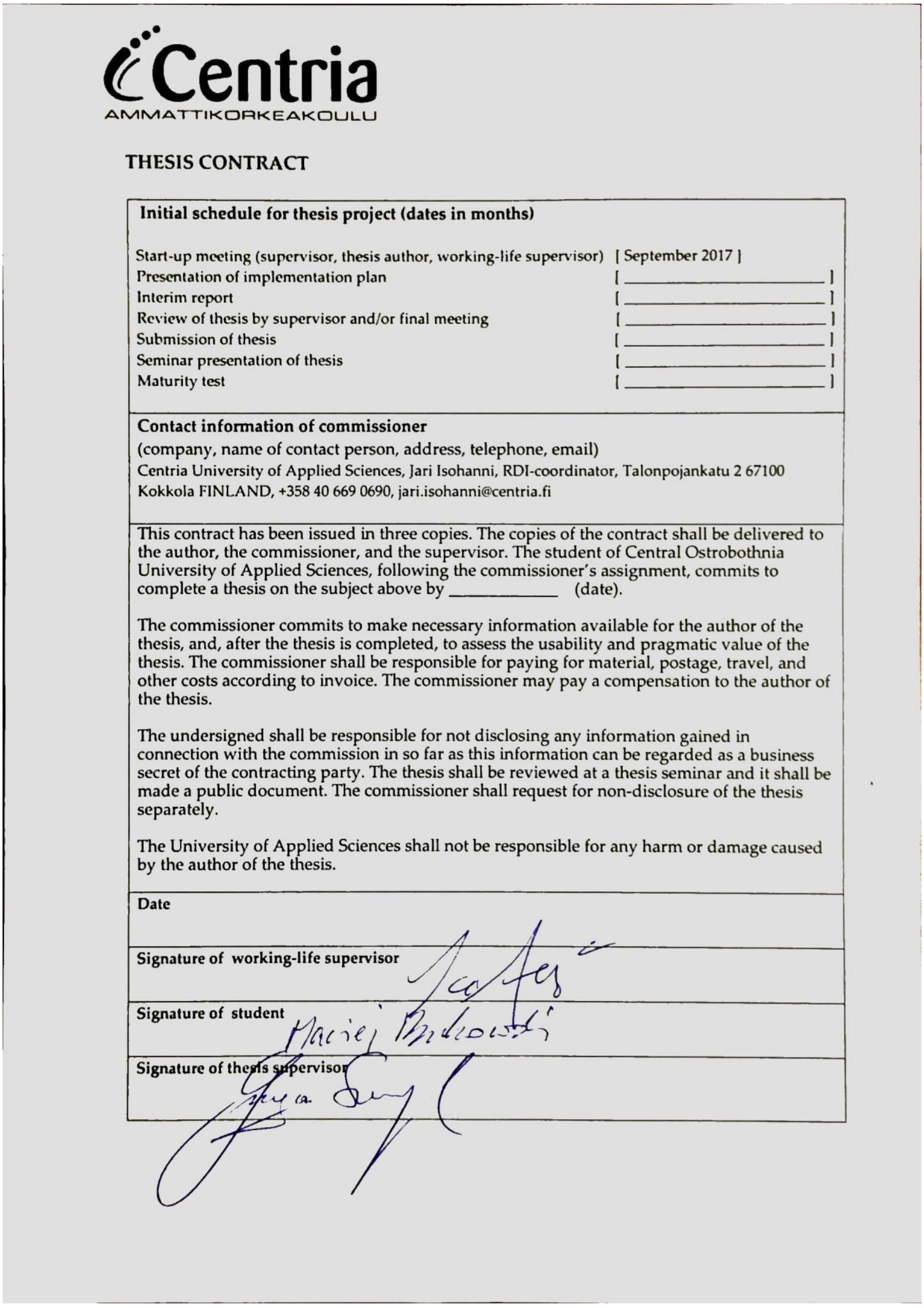
OODesign. (n.d.). *Observer*. Retrieved 03 01, 2018, from https://www.oodesign.com/observer-pattern.html

OODesign. (n.d.). *Singleton*. Retrieved 03 01, 2018, from https://www.oodesign.com/singleton-pattern.html

Sommerville, I. (2011). *Software engineering.*

Square, Inc. (n.d.). *NFC*. Retrieved 05 18, 2018, from NFC: http://nearfieldcommunication.org/how-it-works.html

Suomen Punainen Risti. (n.d.). *punainenristi*. Retrieved November 21, 2017, from what-finnish-red-cross: https://www.redcross.fi/node/1556/what-finnish-red-cross



Centria AMMATTIKORKEAKOULU

THESIS CONTRACT

Initial schedule for thesis project (dates in months)

Start-up meeting (supervisor, thesis author, working-life supervisor)	[September 2017]
Presentation of implementation plan	[ ]
Interim report	[ ]
Review of thesis by supervisor and/or final meeting	[ ]
Submission of thesis	[ ]
Seminar presentation of thesis	[ ]
Maturity test	[ ]

Contact information of commissioner

(company, name of contact person, address, telephone, email)

Centria University of Applied Sciences, Jari Isohanni, RDI-coordinator, Talonpojankatu 2 67100 Kokkola FINLAND, +358 40 669 0690, jari.isohanni@centria.fi

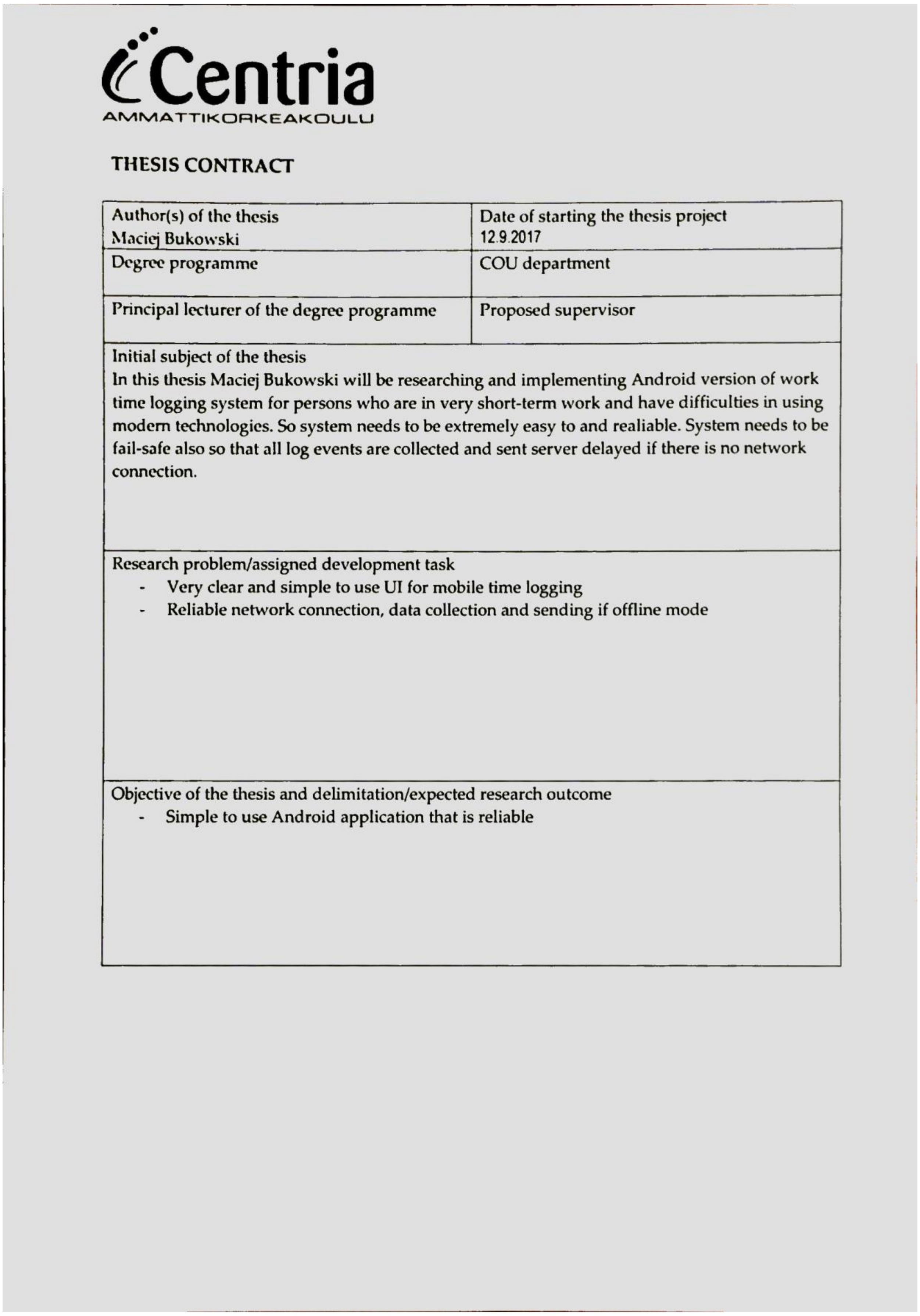
This contract has been issued in three copies. The copies of the contract shall be delivered to the author, the commissioner, and the supervisor. The student of Central Ostrobothnia University of Applied Sciences, following the commissioner's assignment, commits to complete a thesis on the subject above by ____________ (date).

The commissioner commits to make necessary information available for the author of the thesis, and, after the thesis is completed, to assess the usability and pragmatic value of the thesis. The commissioner shall be responsible for paying for material, postage, travel, and other costs according to invoice. The commissioner may pay a compensation to the author of the thesis.

The undersigned shall be responsible for not disclosing any information gained in connection with the commission in so far as this information can be regarded as a business secret of the contracting party. The thesis shall be reviewed at a thesis seminar and it shall be made a public document. The commissioner shall request for non-disclosure of the thesis separately.

The University of Applied Sciences shall not be responsible for any harm or damage caused by the author of the thesis.

Date	
Signature of working-life supervisor	[illegible]
Signature of student	Maciej [illegible]
Signature of thesis supervisor	[illegible]

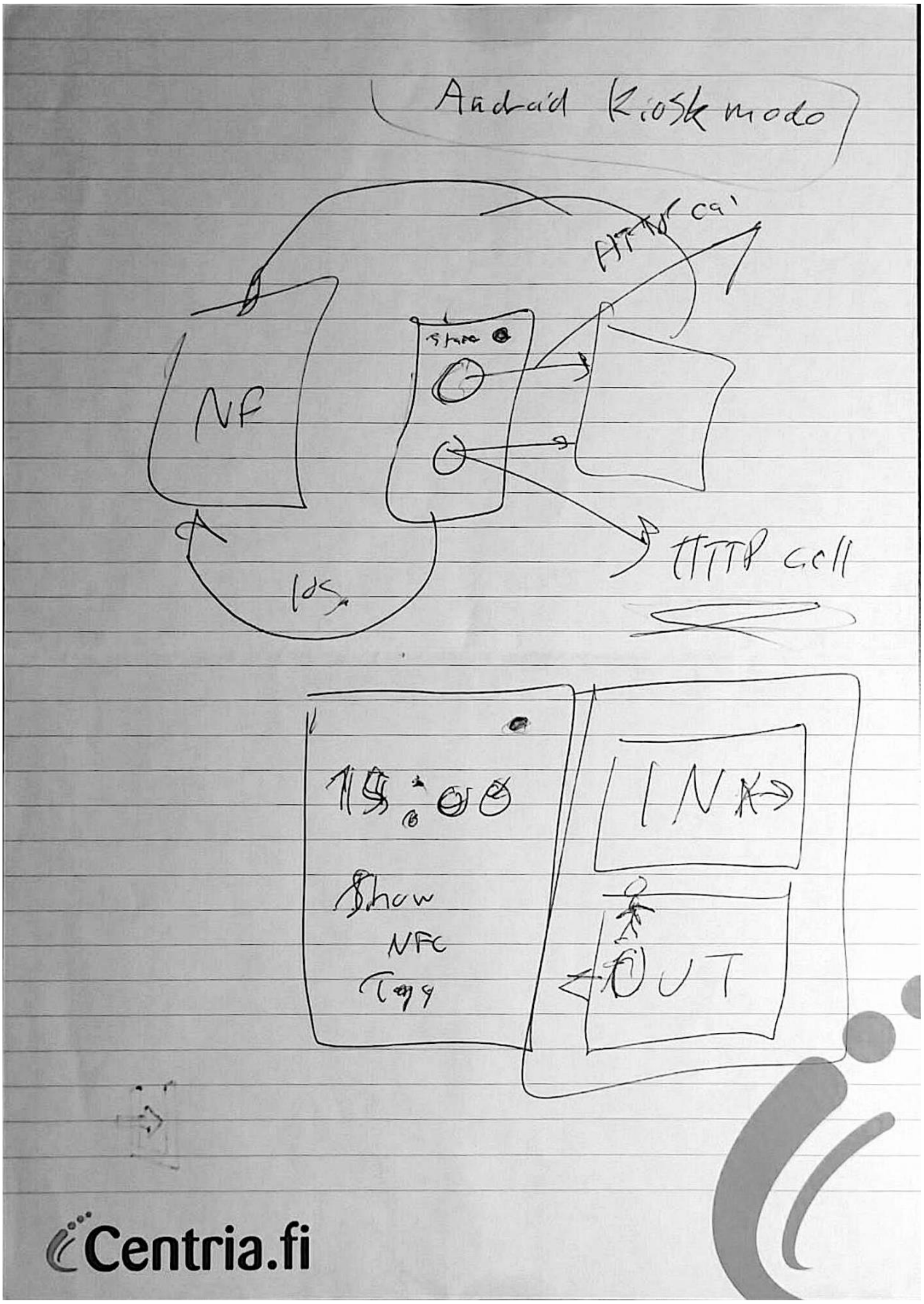


Centria
AMMATTIKORKEAKOULU

THESIS CONTRACT

Author(s) of the thesis Maciej Bukowski	Date of starting the thesis project 12.9.2017
Degree programme	COU department
Principal lecturer of the degree programme	Proposed supervisor

Initial subject of the thesis In this thesis Maciej Bukowski will be researching and implementing Android version of work time logging system for persons who are in very short-term work and have difficulties in using modern technologies. So system needs to be extremely easy to and realiable. System needs to be fail-safe also so that all log events are collected and sent server delayed if there is no network connection.
Research problem/assigned development task - Very clear and simple to use UI for mobile time logging - Reliable network connection, data collection and sending if offline mode
Objective of the thesis and delimitation/expected research outcome - Simple to use Android application that is reliable


Android Kioskmode
HTTP call
NF
HTTP call
IDs
15:00
Show NFC Tag
IN
OUT
Centria.fi