UNIVERSITY OF ZAGREB

FACULTY OF ORGANISATION AND INFORMATICS

V A R A Ž D I N

**Viktor Lazar,** [**vlazar@foi.hr**](mailto:vlazar@foi.hr)

**Martina Šestak,** [**msestak2@foi.hr**](mailto:msestak2@foi.hr)

**Goran Vodomin,** [**gvodomin@foi.hr**](mailto:gvodomin@foi.hr)

**Matej Vuković,** [**mvukovic2@foi.hr**](mailto:mvukovic2@foi.hr)

BLUETOOTH LE SHOWCASE

SOFTWARE ANALYSIS AND DEVELOPMENT PROJECT

Varaždin, 2014.

UNIVERSITY OF ZAGREB

FACULTY OF ORGANISATION AND INFORMATICS

V A R A Ž D I N

**Team number:** T01

**Team name:** Heisenbug

**Team members:**

Viktor Lazar, 0108063551

Martina Šestak, 0016091250

Goran Vodomin, 0016092445

Matej Vuković, 0016094754

BLUETOOTH LE SHOWCASE

SOFTWARE ANALYSIS AND DEVELOPMENT

Mentors:

Dr.sc. Zlatko Stapić

Ivan Švogor, mag.inf.

Evolaris mentors:

Christian Adelsberger

Hermann Moser

Martin Schumann

Varaždin, November 2014.

Content

[1. Introduction 1](#_Toc403765135)

[2. Bluetooth Low Energy Technology 2](#_Toc403765136)

[3. User Requirements Specification 4](#_Toc403765137)

[3.1. Introduction 4](#_Toc403765138)

[3.2. Organisational / Functional Areas Affected 4](#_Toc403765139)

[3.3. Requirements 5](#_Toc403765140)

[4. Software development methodology 6](#_Toc403765141)

[4.1. Scrum team 6](#_Toc403765142)

[4.2. Product backlog 7](#_Toc403765143)

[4.3. Product backlog item 7](#_Toc403765144)

[4.4. Tasks 8](#_Toc403765145)

[5. Technical documentation 9](#_Toc403765146)

[5.1. Conceptual application model 9](#_Toc403765147)

[5.2. UML diagrams 10](#_Toc403765148)

[5.3. System architecture 13](#_Toc403765149)

[5.4. Tools and technologies 15](#_Toc403765150)

[6. SeierFriend mobile application 16](#_Toc403765151)

# Introduction

Evolaris is a company that has been established in 2000 and since then its teams of experts have been developing many web and mobile applications. Besides that, they are also constantly studying new technologies and evaluating if it's possible to apply them to build useful and modern applications.

One of Evolaris' projects is NFC bonus programme that was recently launched at the Shopping City Seiersberg located in Graz. Customers can, by using this application, register online and become „Friend of Seiersberg“. When registering every customer receives a traditional plastic card that contains a “Friend Chip”. When he comes to Shopping City Seiersberg he has to find a so called “Friends Kiosk” and place his “Friend Chip” over the terminal to sign in and begin using the application on his device. Every time he signs in, the user gets some points and vouchers which he can later spend in stores.

A “Friend Chip” which each customer has on his card supports Near Field Communication (hereinafter referred to as NFC) technology. NFC represents a form of contactless communication between devices. To start a communication session a user only needs to wave his device over another device that supports NFC technology. That device acts as a reader/interrogator and he is sending information to the passive user’s device using specific radio frequency.

Figure 1. NFC card,

Source: http://bestnfchardware.com/wp-content/uploads/2013/12/NFCBusinesscards.jpg

Currently Evolaris installed four terminals (readers) inside the shopping center so, whenever a user signs in with his card with NFC tag at that terminal, he can see how many points he has and he can also print his vouchers.

Even though the current system with NFC technology has proven to be successful in practice, they would like to use alternative technologies for the same functionality. This is the area our team will be dealing with; we will do some research about Bluetooth Low Energy (hereinafter referred to as Bluetooth LE) technology and build a mobile application that will mostly have the same functionality as Evolaris’ current application, but instead, it will work using Bluetooth LE.

Everything regarding this project (project documentation, application code and notes from Scrum meetings) will be stored in our public Github repository available at <https://github.com/MartinaSestak/AiR_BLE>.

# Bluetooth Low Energy Technology

Bluetooth Low Energy (also called Bluetooth Smart) is a technology that came public in 2010 and became popular because of lower battery power consumption compared to its predecessor Bluetooth. Thanks to this technology Bluetooth can also be used on smaller devices like watches and toys. It was first supported in mobile devices with Android version 4.3 (Android API 18), but nowadays it’s already available on iOS via iBeacon feature.

Figure 2. Bluetooth LE,

Source:http://upload.wikimedia.org/wikipedia/en/thumb/2/20/Bluetooth\_Smart\_Logo.svg/1280px-Bluetooth\_Smart\_Logo.svg.png

Some features that made this technology stand out from classical Bluetooth technology are:

* Ultra-low peak, average and idle mode power consumption
* Ability to run for years on coin-cell batteries
* Lower implementation costs
* Multi-vendor interoperability
* Enhanced range

The communication between two devices is specified by so called GATT profile (Generic Attribute Profile), which is built on top of the Attribute Protocol (hereinafter referred to as ATT). This profile specifies the way small pieces of data (attributes) are exchanged between devices. Each attribute has its unique identifier (128-bit format for a string ID) and it can be either a characteristic or a service. A characteristic contains a single value and 0 or more descriptors used to describe that value and a service is a collection of characteristics.

For example, when an Android device communicates with a BLE device, we can say that it supports the central role of scanning and looking for an advertisement sent by BLE device, which then supports a peripheral role.

When these two devices establish a connection, they first need to exchange GATT metadata to define data that will be exchanged later. Depending on the nature of that data, a device taking part in this communication can be called a GATT server or a GATT client. In our example, if a BLE device continuously sends data to Android device, we can declare it as a GATT server, where the Android device is a GATT client receiving data from BLE device.

Compared to NFC technology, both Bluetooth LE and NFC define protocols for establishing a radio link and transferring data between two devices, where one device is considered to be a master (reader, scanner) and the other one is a slave (card, advertiser).

However, when it comes to range Bluetooth LE is better than NFC because it works over several meters of distance, while NFC will only work over a couple of centimeters. Also, a NFC communication session between two devices will start only after the user himself holds his device above the other device (reader) and a bidirectional data transfer will start immediately. Unlike that, Bluetooth LE doesn’t require the user to do anything, communication between devices can start when the user’s device is in reader’s range.

However, data transfer begins after the “pairing” process which establishes the connection between devices (for instance, exchanging keys). There is another difference that can be noticed when comparing these technologies, and it’s regarding devices’ power supply. Bluetooth LE demands both devices to be powered with less power consumption compared to the standard Bluetooth. On the other side, when using NFC one device (usually NFC tag) can be passive and inactive until it starts communicating with the other device (reader).

Bluetooth LE has the potential to change the way costumers and retailers interact, but only for costumers with the latest smartphones and tablets.

# User Requirements Specification

### Introduction

#### Objectives

This is the User Requirements Specification for Bluetooth LE Showcase project, for use by Evolaris GmbH, team members and project mentors. In this section we will be determining the project’s scope, user requirements that need to be satisfied in our mobile application and describing our task that was assigned to us by Evolaris’ mentors.

#### History

Evolaris GmbH already developed the entire system and installed terminals inside The Shopping City Seiersberg. New users register online or at a terminal and receive a card with NFC tag. When they arrive at the shopping center, they need to sign in at a terminal in order to gain shopping benefits. After that they can see their “loyality” points and vouchers status and print vouchers. Our assignment is to try to implement the same functionality using Bluetooth LE technology by building a mobile application.

#### Scope

While doing this project we are expected to build a mobile application that will start when the terminal recognizes the user with Bluetooth LE turned on in its range. The user gets a notification signaling his device is recognized and receiving beacons and after that he can start his application by signing in. A signed in user can see how many points and what kind of vouchers he has and receive additional points or vouchers each time he is inside the shopping center.

### Organisational / Functional Areas Affected

#### Assumptions

The application that we build will periodically need to be tested in real environment (The Shopping City Seiersberg) and we will need access to customers’ database and server instance.

### Requirements

All requirements are defined in point form and are rated either Mandatory (M) or Highly Desirable (HD) or Desirable (D), dependent on business need and University Policy.

#### Functional Requirements

##### 3.3.1.1. Common Features

| **Requirement** | **Preference** |
| --- | --- |
| * + - 1. User login | M |
| * + - 1. User can see his points status and receive new ones | M |
| * + - 1. User can see his points history | M |
| * + - 1. User can see his personal info | M |
| * + - 1. User can see his vouchers and receive new ones | D |

##### 3.3.1.2. Reporting

| **Requirement** | **Preference** |
| --- | --- |
| * + - 1. Project documentation | M |
| * + - 1. Notes from SCRUM meetings | M |

#### 

#### System Requirements

##### Hardware

| **Requirement** | **Preference** |
| --- | --- |
| * + - 1. Mobile device with BLE support | M |
| * + - 1. Beacon device | HD |

##### Software

| **Requirement** | **Preference** |
| --- | --- |
| * + - 1. Eclipse IDE | M |
| * + - 1. Min. Android API 18 | M |

# Software development methodology

During this project we decided to use agile software development methodology, which emphasizes team communication and continuous collaboration, functional software product and the flexibility to adapt to emerging business needs. Agile methodology includes methodologies like Scrum, Extreme Programming, Dynamic Systems Development etc.

Since Scrum is currently very popular and enables teams to dynamically plan everything regarding the project (releases, resources and functionalities etc.) in cooperation with stakeholders (in this case Evolaris and project mentors), we decided to use this software development methodology.

Among many tools and services, we decided to use Microsoft Team Foundation Service (Visual Studio Online available at <http://www.visualstudio.com/>) provided by Team Foundation Server (hereinafter referred to as TFS) to implement Scrum methodology in our project. As mentioned in [Introduction](#_Introduction), our application code will be stored on TFS, as well as everything related to our software development process.

### Scrum team

According to Scrum, a Scrum team consists of 3 roles: Product Owner, Scrum Master and Development Team. Each of these roles is assigned to a team member, so every team member has some responsibilities shown in Figure 3. However, in accordance with courses’ policies, each team member will be working on developing some part of the final application.

Figure 3. Scrum roles and assignments

We defined our sprints (iterations) according to course’s checkpoints, so our first sprint called Phase 1 started on 7th November 2014 and ended on 14th November [[1]](#footnote-1). Also, each team member’s working capacity is set to 3 hours per day from Monday to Saturday. As mentioned before, all notes from Scrum team meetings can be found in our Github repository.

### Product backlog

On a product backlog refinement meeting the Product Owner defined and prioritized product backlog items (hereinafter referred to as PBI) that need to be implemented and the Scrum Master defined which PBIs need to be implemented at the end of each sprint.(shown in Figure 4).

Figure 4. Product backlog

Figure 5. Product Backlog

### Product backlog item

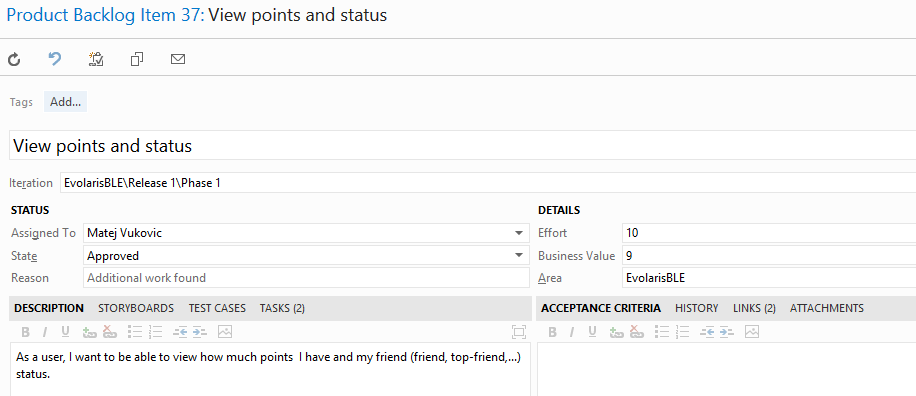
Each PBI is declared to be in a state: *New, Approved, In progress, Commited* or *Done* and changes its state during the sprint. Also, when defining a PBI, a Product Owner sets its Business value and Effort measured in the number of hours the PBI will take to be implemented. He also need to make a description of the PBI which contains a role, what this role want to be able and why.

Figure 5. PBI definition

### Tasks

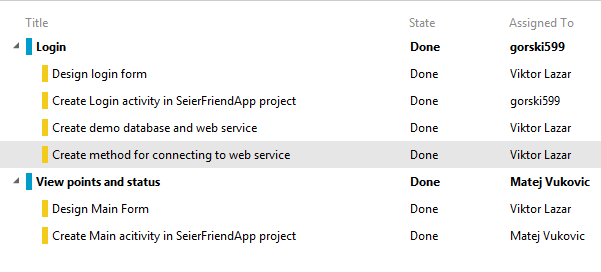
On a sprint planning meeting the Scrum Master in agreement with the entire team analyzed each product backlog item and defined tasks for each PBI. Each task and PBI was assigned to a team member and he has to continuously update task’s status and remaining work time. Figure 6 contains a list of tasks for Login PBI, which is assigned to Goran Vodomin and at that time in “Committed” state with 7 hours of remaining work. During the Phase 1 sprint, we managed to implement the following PBIs and their tasks shown in Figure 6.

Figure 6. Phase 1 product backlog items

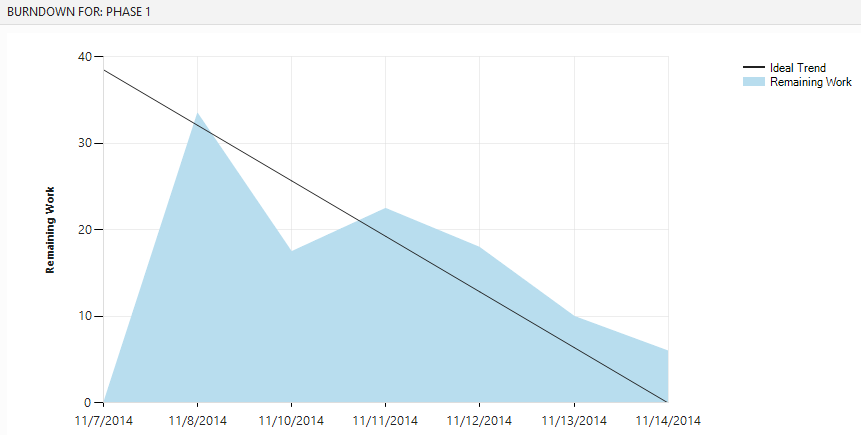
Last but not least, we can analyze our performance by looking at the generated burndown chart. Based on set project parameters (capacity, remaining work on tasks, effort etc.) the tool sets a line representing Ideal Trend. If our remaining work is bellow that line, this means all tasks are done on time, but if it’s above the line, this means the project is late. Our project’s final burndown chart is shown in Figure 7. It shows that the first sprint was at the beginning executed within timelines, which is opposite to a slight delay in the further course of this sprint.

Figure 7. Burndown chart for Phase 1 sprint

# Technical documentation

In Graz our team was introduced with user specifications. Based on these information we created the conceptual model of application representing user stories and additional UML activity and class diagrams to help us build our application more efficiently. The data model (ERA diagram) will be described in the next sprint since we haven’t had access to the system’s database or any other documentation owned by Evolaris. Also, we built our own web service to retrieve user data from test database[[2]](#footnote-2).

### Conceptual application model

The conceptual model for our application was first described and created with Pop tool and is available [online](https://popapp.in/w/projects/545c977595b7e3a228ecdc78/mockups/545f7f92ad823c190b3abf85). This model defines 5 possible activities (user stories) which a user can access. They are connected in the following way:

1. A user starts the application, enters his credentials in a login form and presses the Sign in button
2. If he doesn’t have an account, he can press the Register online link that will redirect him to the web application where he can register
3. If he forgot his password, he can press the Forgot password? Link to receive a notification on his email address.
4. If he signs in successfully the next activity enables him to see how many points he has and what is his “Friend of Seiersberg” status
5. By pressing the Menu icon in the top left corner, a side menu appears. It contains his account information and available options (links to other activities).
6. If the user presses the Points option, he is sent back to the main activity described in step 2.
7. If the user selects the Discounts option, a new activity appears, showing him a list of all vouchers that the system is currently offering. Each voucher is shown as an image with a description. However, he can only access and use vouchers that are available to him based on his points status, while some vouchers can be disabled so he can’t access them.

### UML diagrams

#### 5.2.1 Activity diagram

Activity diagram is used to describe dynamic aspects of a system. Activity diagram shows flow of control or object flow with emphasis on the sequence and conditions. We decided to create activity diagrams (even though it’s not mandatory in Scrum methodology) so that we can better understand the communication flow between different objects. Based on our conceptual model, we can see four dynamic aspects of our system, they are “Starts application”, “Checks points”, “Checks for vouchers” and “Check for voucher details”.

**Starts application**

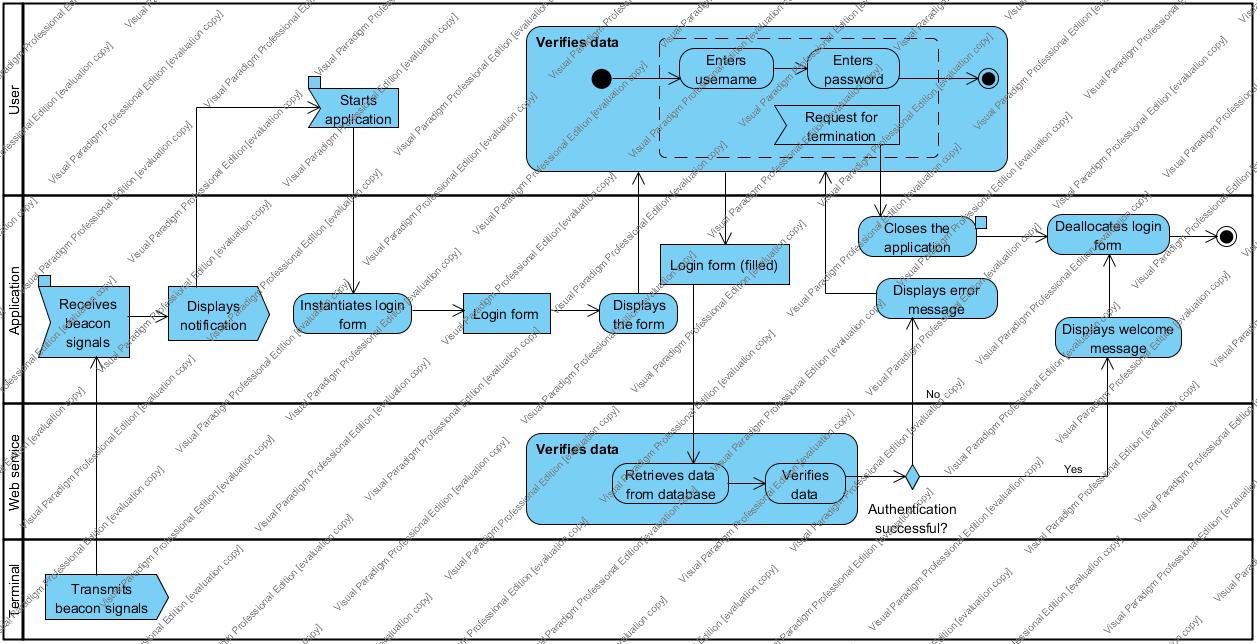
To use the application user has to be signed in, so for that purpose application has the login form where users enter their username and password. When the data is entered, application passes data to web service. Web service verifies the data and sends information if the entered data is correct. Application receives data and based on information moves on or displays an error message. Meanwhile, during this whole time application listens for beacon signals that terminals transmit. If entered data is correct and application received a beacon signal, a welcoming message is displayed and the user can use the application. A user can send request for termination at any time, which closes the application.

Figure 8. Activity diagram for use case Starts application

**Checks points**

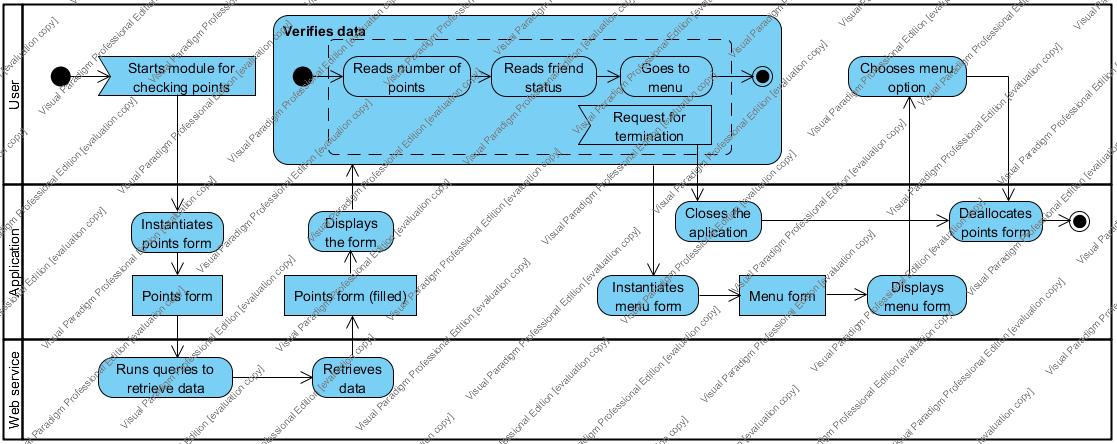
To check points and see his friend status a user has to start the corresponding module. In the beginning application instantiates form and sends data request to web service. Web service then runs queries on database to retrieve data. When the data is retrieved, web service sends it to application which displays the data. A user then reads the number of points and his friend status. Anytime the user can send request for termination which closes the application.

Figure 9. Activity diagram for use case Checks points

**Checks for vouchers**

In order to see list of vouchers the user starts the corresponding module. Further logic is quite the same as in the previous case, application instantiates the form, sends data request to web service. After that web service runs queries on database to retrieve data. In the end when necessary data is retrieved, web service sends it to application which displays the data. User reads the list of vouchers and chooses two options, to see voucher details or to go back to menu. Depending on the choice the application creates menu form or voucher details form. The user can send request for termination which closes the application at any time.

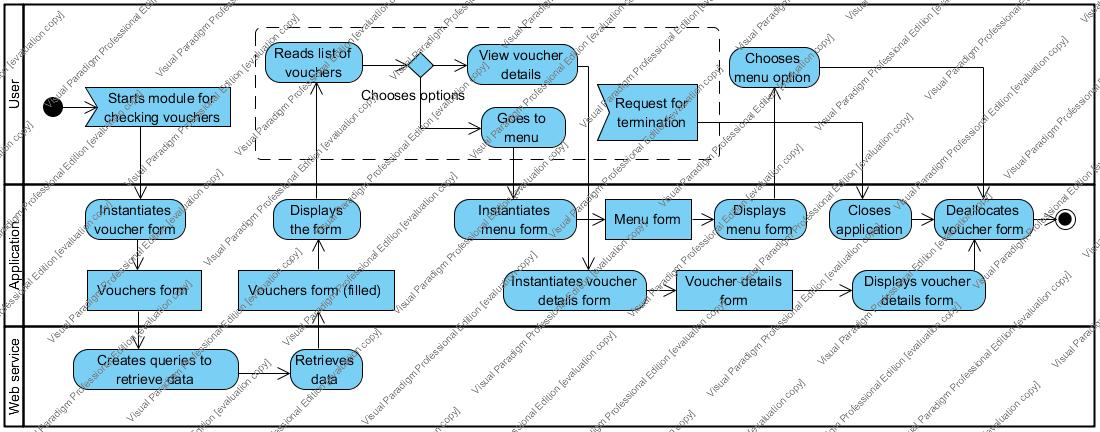


Figure 10. Activity diagram for use case Checks vouchers

**Checks for voucher details**

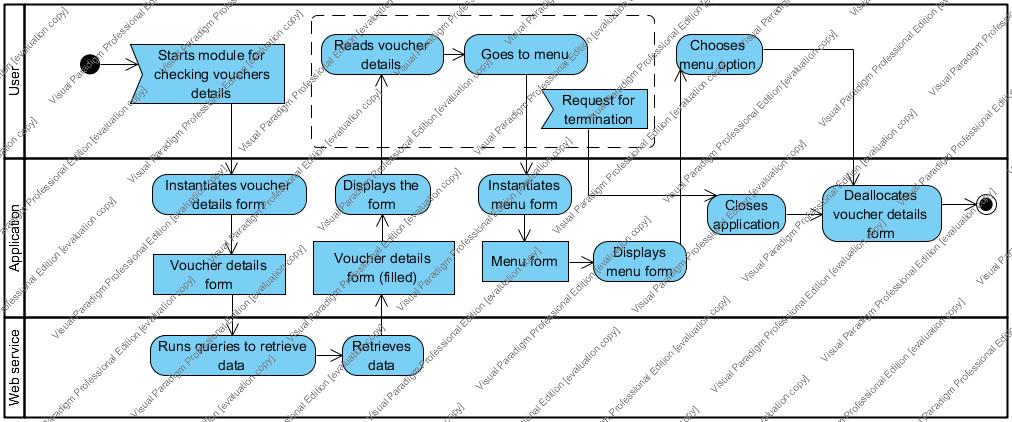
For every voucher displayed the user can see its details and start the corresponding module. After the module has started, application instantiates the form and sends data request to web service. Web service runs queries to retrieve data and forwards retrieved data to application. Application displays data. The user can send request for termination which closes the application at any time.

Figure 11. Activity diagram for use case Checks voucher details

### System architecture

The entire system architecture behind the “Friend of Seiersberg” application includes several participants. Their communication (exchanged data) are described in Figure 10 according to our current presumptions[[3]](#footnote-3).

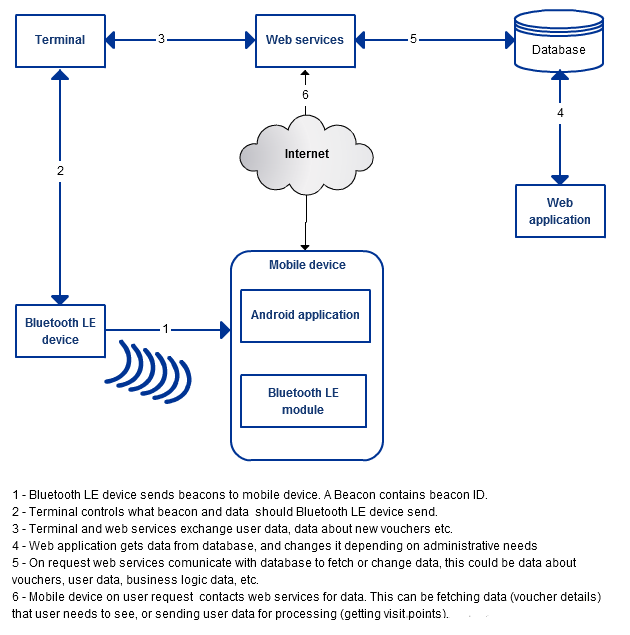


Figure 12. System architecture

If we look more closely at the Android application’s infrastructure, it can be described as follows:

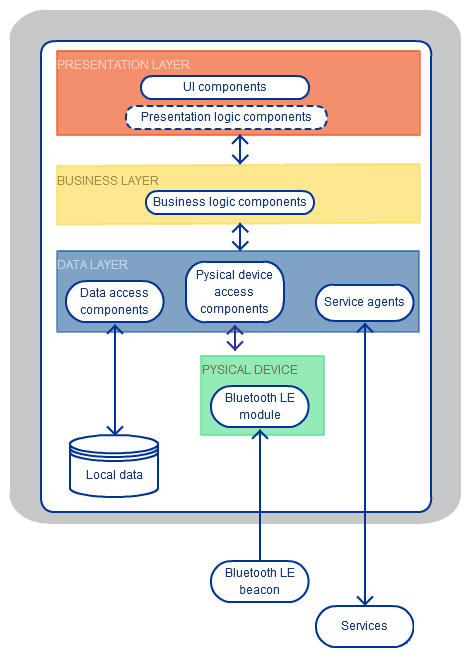
* Presentation layer consists of UI components (view controls used in forms) and Presentation logic components (event listeners that execute part of a code when triggered).
* Presentation layer communicates with Business layer, which has some Business logic components used to implement business logic (for instance, to check whether the user signed in more than once in one day, which vouchers are disabled to him etc.).
* Business layer communicates with Data layer, which contains components for working with the local database, accessing physical device (BLE device, NFC card or something else) and exchanging data with the corresponding module (in this case BLE module) and service agents for exchanging data with services.
* BLE beacon and services represent a supporting infrastructure for the application.

Figure 13. Mobile application infrastructure

The application will be built in a modular way enabling the system to work properly no matter what technology is used in Data layer, whether physical device access components are communicating with a BLE module, WIFI, NFC or any other module.

### Tools and technologies

During this project we decided to use various technologies and tools for different activities (development, design, document versioning etc.).

**Scrum methodology tools**

* Visual Studio Online (Microsoft Team Foundation Service) available at <https://martina-sestak.visualstudio.com/DefaultCollection/EvolarisBLE>

You can access our project by signing in with following credentials:

Email: **team.heisenbug@hotmail.com**

Password: **ProjektAiR**

**Modeling tools (conceptual model, activity and class diagram, system architecture)**

* POP (<http://popapp.in/> ) for conceptual modeling
* Creately to model system architecture
* Visual Paradigm CE to create activity diagrams
* ObjectAiD UML Explorer for Eclipse to generate class diagram

**Version control systems**

* Github repository with all project materials and artifacts is available at <https://github.com/MartinaSestak/AiR_BLE>
* Visual Studio Online for application code checking in and out

**Web service and test database**

* DigitalOcean , PHP, MySQL

**Application development tools**

* Eclipse Luna IDE (minimal Android version 4.3 (API 18), since earlier versions don’t support BLE technology)

# SeierFriend mobile application

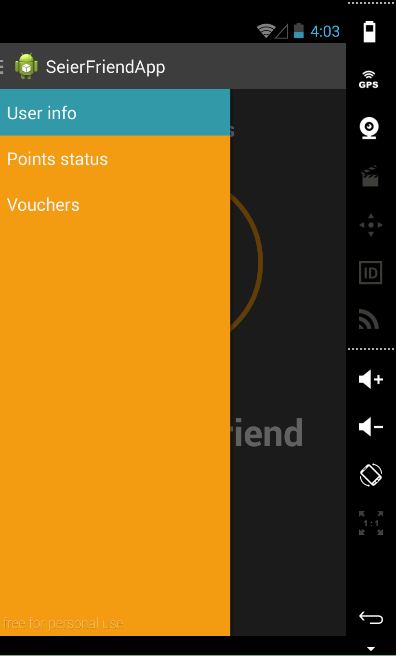
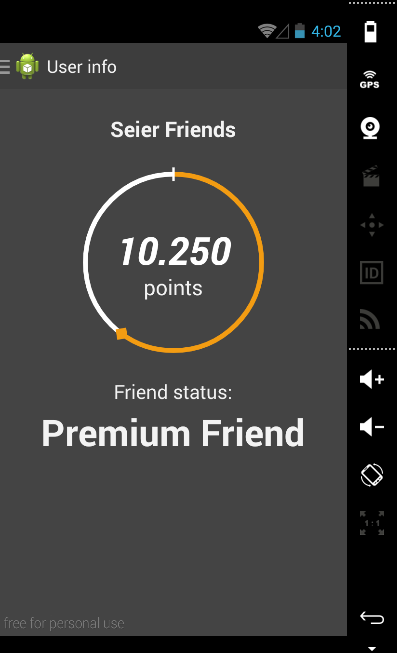
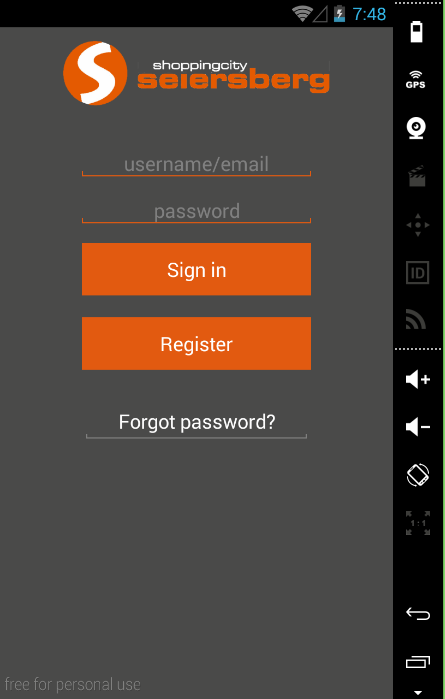
SeierFriend mobile application is built in Eclipse IDE and at the end of Phase 1 sprint has the following functionalities:

* When the user starts the application, a login form is displayed
* User enters his username and password and click Sign in button

**Important**🡪 credentials that need to be entered in the current version of the application in order to sign in are:

**Username:** mbach@mail.at

**Password:** 123456

* If the user signs in successfully, an activity for viewing points and status is displayed to him
* From the side menu he can select the Points status option, which will take him back to PointStatus activity

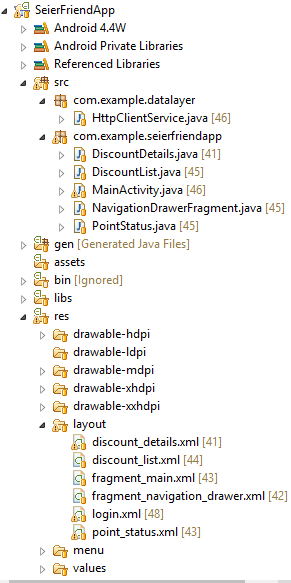
2. View points and status screen

3. Open side menu

1. Login screen

At the end of the first sprint our Android application project has 6 XML files defining the application layout and 2 packages *com.example.seierfriendapp* (implements application’s functionality with 5 Java source files) and *com.example.datalayer* (connects application to web service). The project’s structure is shown in Figure 14.

Figure 14. Android application project partial structure



class containing methods for implementing side menu

class containing methods for displaying PointStatus activity

class containing methods for displaying DiscountList activity

class containing methods to display DiscountDetails activity

class that calls PointStatus activity

class containing a method for connecting to web service

We also defined standard strings to be displayed as a text to a user in *strings.xml* file and modified *AndroidManifest.xml* to require permission to use Internet, so that the application can retrieve data from the web service. Java source files contain event listeners triggered when a user does some action with a certain UI control (for instance, selects the navigation drawer item, clicks the Sign in button etc.).

1. Phase 1 sprint started later because we didn’t know our project scope before our meeting with Evolaris mentors, which took place on 31st October and because we were unofficially studying BLE technology earlier. [↑](#footnote-ref-1)
2. The built web service and test database will be replaced with Evolaris' API in the next sprint. [↑](#footnote-ref-2)
3. During the first sprint we didn't have access to Evolaris' documentation, so we weren't completely familiar with the system architecture and we have some uncertainties whether, for instance, BLE device communicates with terminal as a standalone device or if it's a terminal component. All current uncertanties will be removed one we receive their documentation during the second sprint. [↑](#footnote-ref-3)