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, 2015.

UNIV 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, January 2015.

Content

[1. Introduction 1](#_Toc408868830)

[2. Bluetooth Low Energy Technology 2](#_Toc408868831)

[3. User requirements specification 4](#_Toc408868832)

[3.1. Introduction 4](#_Toc408868833)

[3.2. Organisational / Functional Areas Affected 5](#_Toc408868834)

[3.3. Requirements 5](#_Toc408868835)

[4. Software development methodology 7](#_Toc408868836)

[4.1. Scrum team 7](#_Toc408868837)

[4.2. Iterations 8](#_Toc408868838)

[4.3. Project team capacity 8](#_Toc408868839)

[4.4. Product backlog 9](#_Toc408868840)

[5. Technical documentation 13](#_Toc408868841)

[5.1. Conceptual application model 13](#_Toc408868842)

[5.2. Data model 14](#_Toc408868843)

[5.3. System architecture 15](#_Toc408868844)

[5.4. Tools and technologies 17](#_Toc408868845)

[6. SeierFriendApp mobile application 18](#_Toc408868846)

[6.1. Bluetooth LE scanning module and notification launcher 20](#_Toc408868847)

[6.2. Working with web services and local database 20](#_Toc408868848)

[6.3. Application design and fragments 21](#_Toc408868849)

[6.4. Application launching instructions 23](#_Toc408868850)

# 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>. Application code created in the second project phase is available at <https://github.com/MartinaSestak/AiR_BLE/tree/master/Sprint2/Code> .

# 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 using the appropriate API.

### 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

##### Common Features

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

##### 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. Intellij IDEA 14.0.1 Ultimate Edition | 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.

### 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

For this project we defined 4 sprints (iterations), so our first sprint called Phase 1 started on 7th November 2014 and ended on 14th November [[1]](#footnote-1). As mentioned before, all notes from Scrum team meetings can be found in our Github repository.

### Iterations

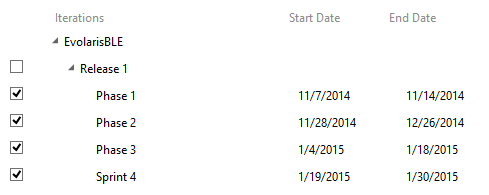
The Scrum team estimated that this project can be completed in 4 sprints with a duration of 1-4 weeks, as shown in Figure 4.

Figure 4. Iterations configuration

### Project team capacity

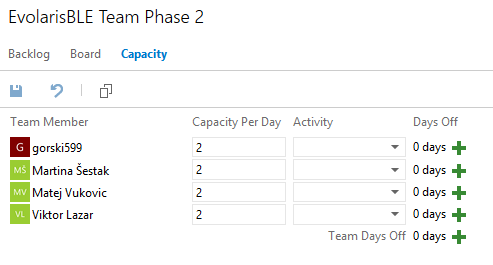
Each team member was assigned to a task while working 2 hours per day with no days off.

Figure 5. Team capacity

### 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. At the beginning of the project in sprint 1, product backlog looked as shown in Figure 6.

Figure 6. Product backlog defined in sprint 1

Figure 7. Product Backlog

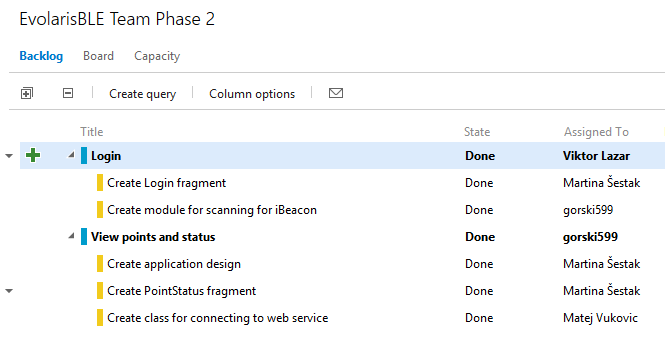
After Phase 1 sprint the Scrum master called a product backlog refinement meeting to improve the existing product backlog and analyze tasks assigned to each PBI. In Phase 2 sprint we decided to implement Login and View points and status PBIs so our product backlog looked as shown in Figure 7.

Figure 7. Product backlog for Phase 2 sprint

On the last product backlog refinement meeting new tasks were added to each PBI, so our product backlog for Phase 3 sprint looks as shown in Figure 8.

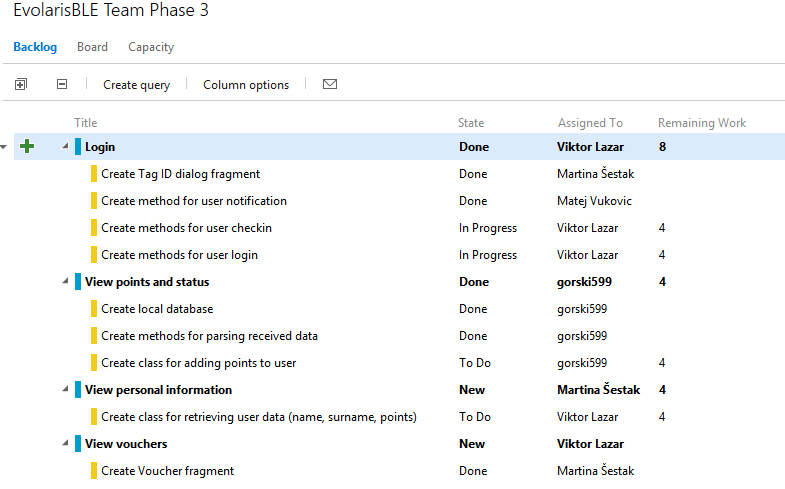


Figure 8. Product backlog for Phase 3 sprint

#### Product backlog item

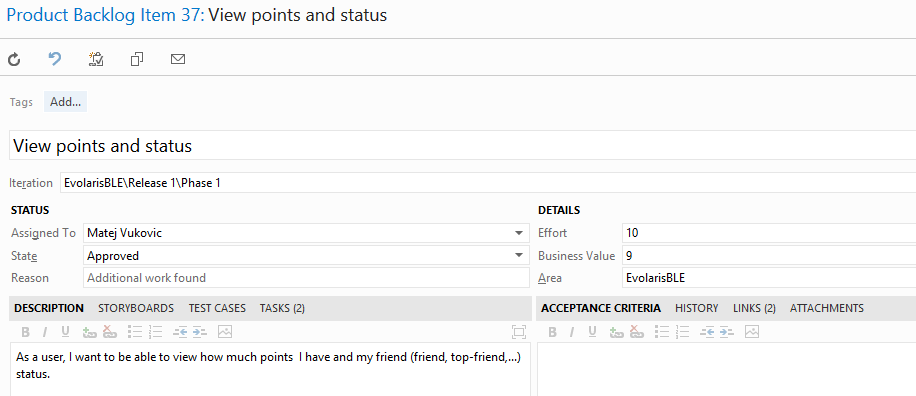
Each PBI is declared to be in a state: *New, Approved, In progress, Commited* or *Done* and changes its state during the sprint. Also, when defining a PBI, a Product Owner sets its Business value and Effort measured in the number of hours the PBI will take to be implemented. He also needs to make a description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

Figure 9. PBI definition

#### Tasks

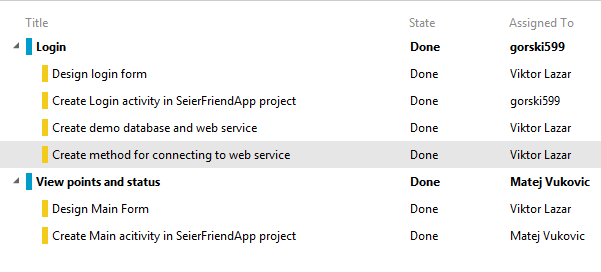
On a sprint planning meeting the Scrum Master in agreement with the entire team analyzed each product backlog item and defined tasks for each PBI. Each task and PBI was assigned to a team member and he has to continuously update task’s status and remaining work time. Figure 10 contains a list of tasks for Login PBI for Phase 1 sprint, 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.

Figure 10. Phase 1 product backlog items

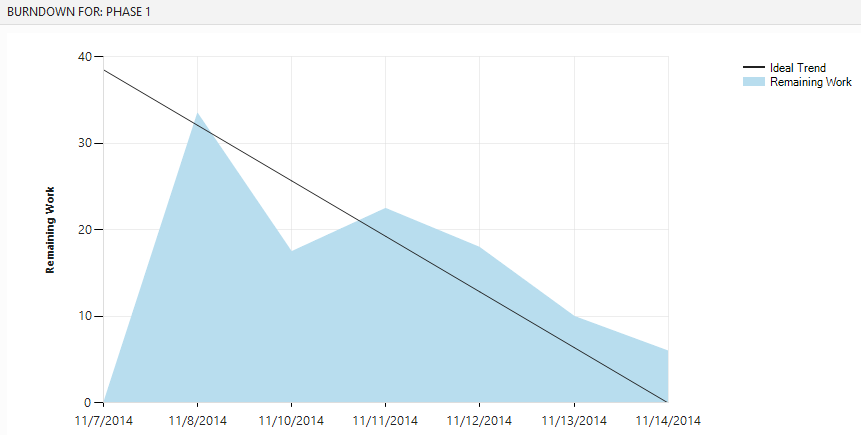
Last but not least, we can analyze our performance by looking at the generated burndown chart. Based on set project parameters (capacity, remaining work on tasks, effort etc.) the tool sets a line representing Ideal Trend. If our remaining work is bellow that line, this means all tasks are done on time, but if it’s above the line, this means the project is late. Our project’s final burndown chart for Phase 1 sprint is shown in Figure 11. 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 11. Burndown chart for Phase 1 sprint

If we take a look at the Phase 2 sprint burndown chart, we can notice that most of the tasks were completed at the beginning of the sprint so the project development was within time limits, except for the last week, but at the end of the sprint all tasks set for Phase 2 sprint were still completed. Since Phase 3 sprint is still running, the burndown chart for this sprint will be shown and analyzed in the final project phase.

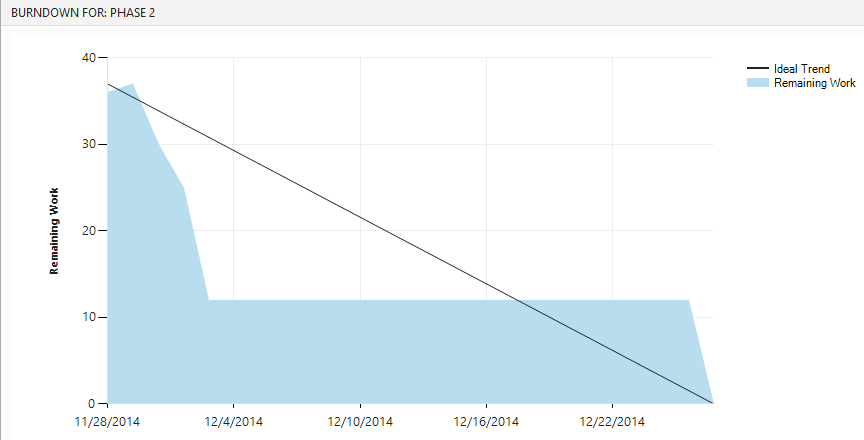


Figure 12. Burndown chart for Phase 2 sprint

# 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.

### 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. If this is his first sign in, he needs to enter Tag ID from his NFC card in order to receive an authorization token that he will later use to receive points, else the main activity is displayed.
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 signs in successfully the next activity enables him to see how many points he has and what is his “Friend of Seiersberg” status.
4. By pressing the Menu icon in the top left corner, a side menu appears. It contains his personal information, point status and available vouchers list.
5. If the user presses the Point status option, he is sent back to the Point status fragment described in step 3.
6. If the user selects the Vouchers option, a new fragment appears, showing him a list of all vouchers that are available to him. Each voucher is shown as an image with a name.

### Data model

As mentioned, at the end of the first sprint we were provided with access to Evolaris web services API and their database instance so the application’s ERA model is the same as the one defined by Evolaris. The database can be accessed at <https://evodeployment.evolaris.net/phpmyadmin/> with following credentials:

Username: **varazdin**

Password: **7X2tGEjb8anu**

As shown in Figure 13, the most important tables in the ERA model with most connections to other tables are table Offers and Participants. Table Offers contains information about vouchers and their image, while table Participants contains information about users, number of points they have and their point level, but also the information about when was the last time they received a push notification form the application and their notification token.

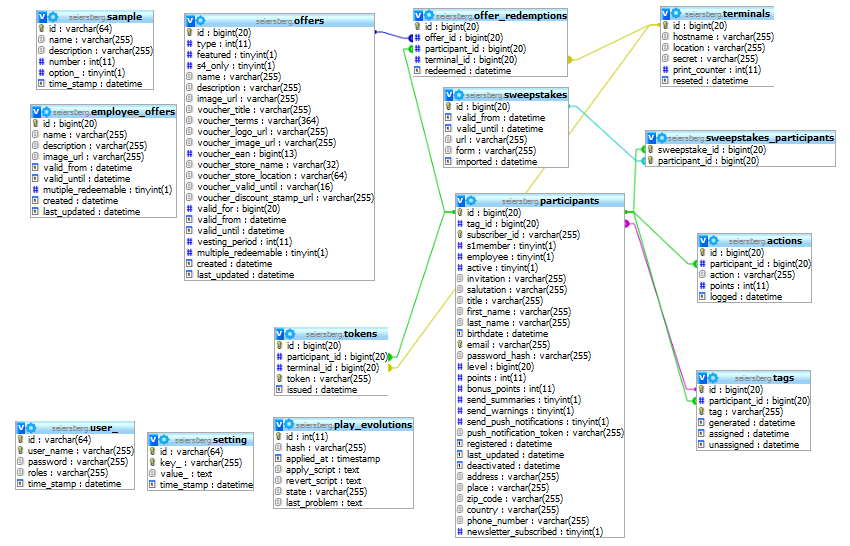


Figure 13. ERA model for Friend of Seiersberg application

### System architecture

The entire system architecture behind the SeierFriendApp application includes several participants. Their communication (exchanged data) is described in Figure 14.

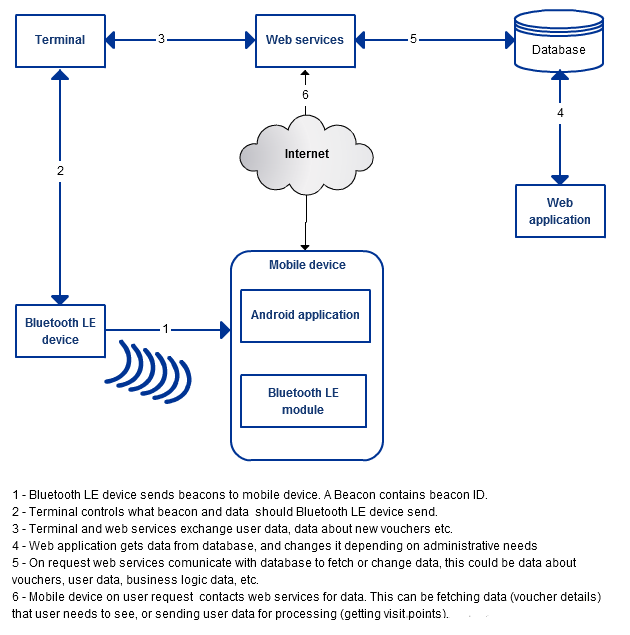


Figure 14. System architecture

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

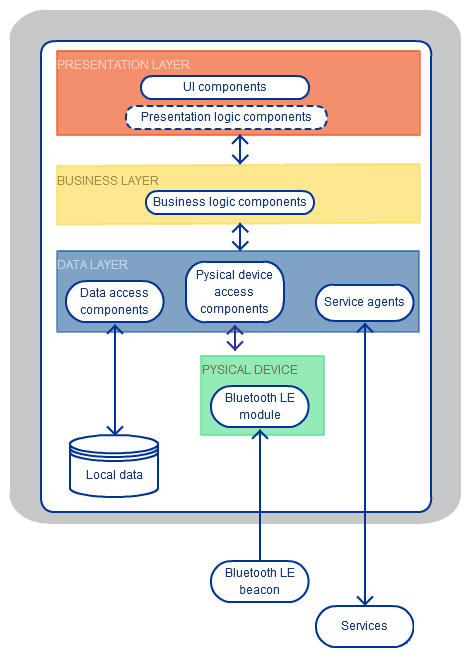
* Presentation layer consists of UI components (view controls used in forms) and Presentation logic components (event listeners that execute part of a code when triggered).
* Presentation layer communicates with Business layer, which has some Business logic components used to implement business logic (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 15. Mobile application infrastructure

The application will be built in a modular way enabling the system to work properly no matter which 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

**Version control system**

* Github repository with all project materials and application code is available at <https://github.com/MartinaSestak/AiR_BLE>

**Web service and test database**

* DigitalOcean , PHP, MySQL

**Application development tools**

* Eclipse Luna IDE, Intellij IDEA 14.0.1 Ultimate Edition (minimal Android version 4.3 (API 18), since earlier versions don’t support BLE technology), Genymotion

# SeierFriendApp mobile application

SeierFriendApp mobile application is built in Intellij IDEA 14.0.1 and at the end of Phase 2 has the following functionalities:

* When he’s in range of beacon devices the application icon is displayed on user’s mobile device notifying him that he can start the application.
* When the user clicks on the displayed notification, SeierFriendApp starts with a login screen. User can’t sign in unless he enters both his username and password.
* If he doesn’t have an account, he can register on the web application by clicking the Register button.
* When the user clicks the Sign in button, a Tag ID dialog is shown, which prompts the user to enter his Tag ID written on his NFC card.
* After clicking the Save button main activity with Point status fragment is displayed to the user where he can see the number of points and his status.
* User can select the menu icon in top left corner, which opens the sidemenu containing other options.
* Vouchers option from sidemenu opens a fragment containing a list of vouchers and their images (hardcoded in the current application version).

After Phase 1 we decided to move our entire application code from Eclipse IDE to Intellij IDEA in order to improve application stability and our efficiency. At the end of Phase 2 application source code is divided into several parts shown in Figure 16.

Figure 16. Project src folder structure

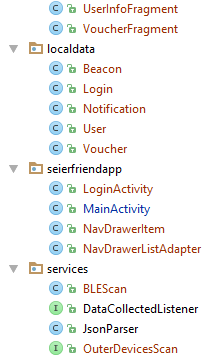
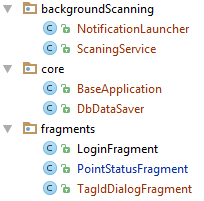
Interface which defines a method for starting scanning for beacon devices and returning device list.

Class for initializing Bluetooth adapter and ActiveAndroid.

Class for displaying application launcher icon on user’s mobile device.

Class containing a method for saving user credentials to local database.

Class containing methods for scanning for beacon devices and their MAC addresses.



Classes containing methods for displaying navigation sidemenu and filling in the options.

Class containing methods to start and stop scanning for beacon devices.

Class containing methods for fetching and parsing data received from web service.

Interface indicating whether data is collected or not.

Classes extending Model class which represent tables in local database.

Fragment returning an inflated view (voucher\_fragment.xml) displayed in MainActivity class.

Fragment returning an inflated view (user\_info\_fragment.xml) displayed in MainActivity class.

Fragment returning an inflated view (tad\_id\_dialog.xml) displayed in LoginActivity class.

Fragment returning an inflated view (login\_layout.xml) displayed in LoginActivity class.

Fragment returning an inflated view (point\_status\_fragment.xml) displayed in MainActivity class.

As mentioned in [chapter 4.4](#_Product_backlog), the most interesting tasks that were assigned for Phase 2 and Phase 3 sprint were: implementing BLE module for scanning for beacon devices and creating a notification launcher, working with web services and saving data to local database. The logic behind this tasks is described in further chapters. We also made some improvements in application design by using fragments.

### Bluetooth LE scanning module and notification launcher

On application startup, application checks if Bluetooth is enabled. If that’s not the case, application enables it and gets MAC addresses of beacon devices from web service and stores them in local database. After that application launches background scanning service. Background service gets previously stored MAC addresses via interface, launches BLE scanning module providing it with MAC addresses.

Besides that, it also implements broadcast receiver which receives broadcast intents from scanning module (class BLEScan). BLE scanning module has few methods, some of which work with the Bluetooth adapter. They are continuously scanning and if there is a MAC address match they make broadcast intent. The scanning service receives that intent and calls a class (class NotificationLauncher) that handles intent. After that NotificationLauncher checks if notification were sent and if there weren’t, it creates notification.

If the user quits application ScanningService continues running in the background scanning for beacon devices. When the user clicks the notification NotificationLauncher launches Login Activity with extra data that indicates if Check method in web service needs to be called. On the other hand, if the user cancels notification, if there is a match, notification is launched again.

### Working with web services and local database

Application communicates with a backend server in order to ensure that all necessary data about the user and his benefits are available . This communication is achieved through API provided by Evolaris. In the domain of our Android application this communication is mainly based on a class called JsonParser which is included in services package. The most important feature of JsonParser class is its nested class called WebServiceDataCollection. Here is the whole logic that stands behind data retrieval from the backend server. Depending on the type of the HTTP request (GET or POST) this class parses forwarded parameters and calls backend server's API. After that data returned in the server's response are validated and converted to Json object. Interface which indicates if data is collected or not is also ensured.

In the application, data is stored in a local SQLite database. Database is achieved through ActiveAndroid and all work and transactions related to database is made through it. ActiveAndroid is an object relational mapper (ORM) which provides easy and fast way to work with local database. At this phase of development local database consists of five tables, i.e. classes. After user login and data synchronization, data is stored in the local database. This action is handled in DbDataSaver class. Other functionalities related to fetching and storing data will be added in the next phase according to our development methodology.

### Application design and fragments

One of Phase 2 sprint tasks was Application design. Also, we also needed to implement fragments inside activities that will later be used to bind and display data received from web service. Figures 18.-22. show the current application screens in a simple use case.

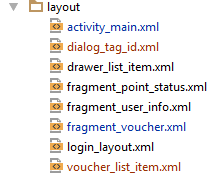
The layout of fragment is defined in the following XML resource files located in layout folder:

Figure 17. Aplication layout files

Login\_layout.xml defines the layout of Login Activity and contains a Sign in button, which when clicked initializes TagIdDialogFragment whose layout is defines in dialog\_tag\_ii.xml. Activity\_main.xml defines a layout of Main Activity which contains a FrameLayout to display different fragments and a ListView for displaying navigation sidemenu. The layout of each sidemenu list item is defines in drawer\_list\_item.xml. The layout of each option fragment is defined in separate XML files (fragment\_point\_status.xml, fragment\_user\_info.xml and fragment\_voucher.xml). Besides that, fragment\_voucher.xml contains a ListView, where the layout of each voucher list item is defined in voucher\_list\_item.xml.

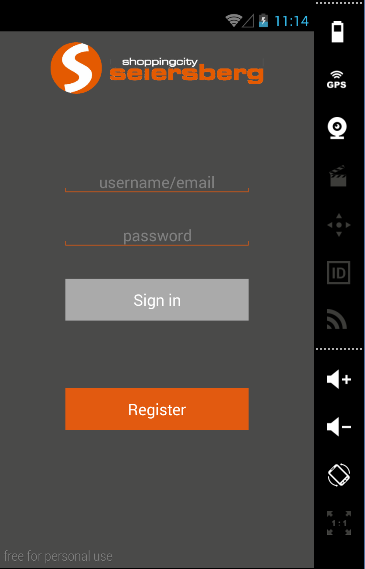
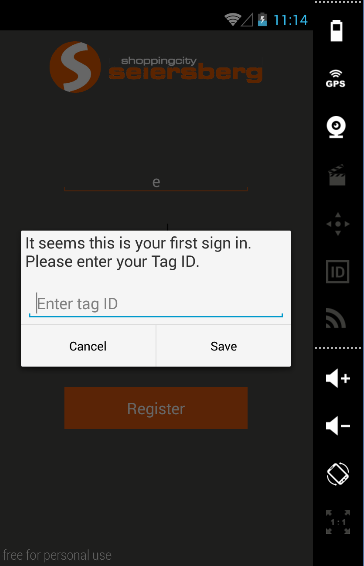
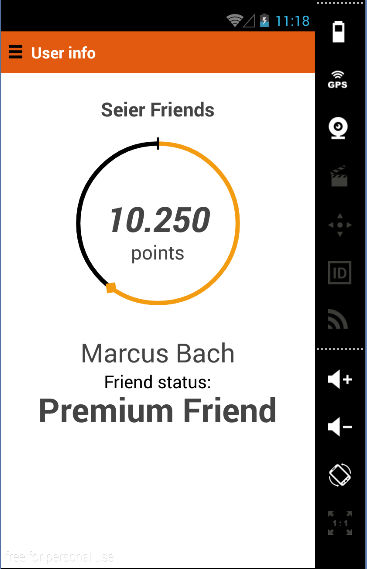
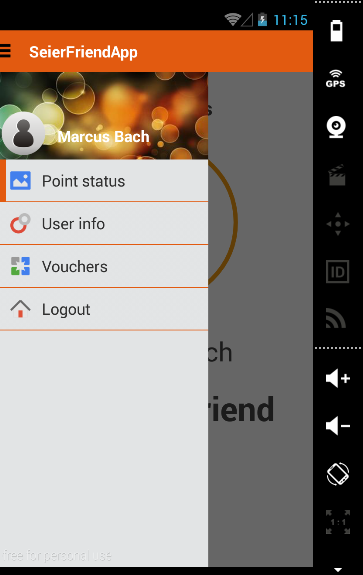
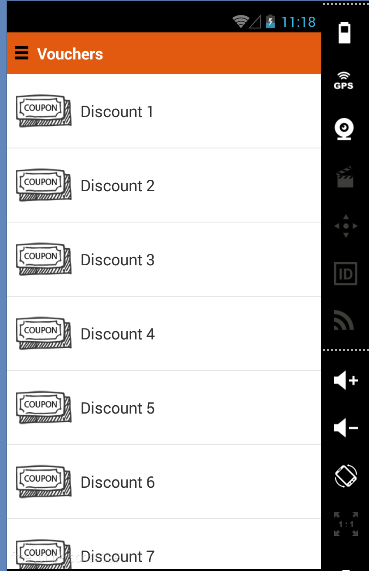


Figure 19. If the user is signing in for the first time, a dialog is shown where he needs to enter his tag ID from NFC card.

Figure 21. By clicking the icon or swipeing, a user can open the sidemenu with other options.

Figure 22. By choosing Vouchers option from sidemenu the user can see a list of available vouchers.

Figure 20. After successful sign in, a user can view his number of points and his status.

Figure 18. Login screen with disabled Sign in button until userame and password are entered.

### Application launching instructions

Since Genymotion emulator doesn’t support Bluetooth LE module, our BLE module for scanning beacon devices can only be tested on a mobile device with BLE support.

Source code of this application available at Github contains classes and methods for scanning and saving MAC addresses to the local database which are currently commented out so that the application can be launched within the emulator (in this case BLE module functionality is not available).

When launching an application on a mobile device, remove comments in the following lines of code:

* BaseApplication.java: *saveMac();* (line 36)
* ScaningService.java: *deviceList = getMacAdresses();* (line 46)

Also, since receiving data from web service is not completely implemented (data is received from web service but not stored right away in local database), when launching application on a mobile device for the first time, comment out following lines of code:

* PointStatusFragment.java:

*points.setText(String.valueOf(u.getPoints()));* (line 43)

*name.setText(u.getFirstName() + " " + u.getLastName());()));* (line 46)

After that, relaunch the application, but this time uncomment previous lines of code. Now the received data is displayed inside a Toast message.

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)