TECHNOLOGICAL EDUCATIONAL INSTITUTE OF CRETE

School of Applied Technology

Department of Informatics Engineering

Thesis

Title: Indoor Location and Tracking

Giannantonakis Ioannis

Logothetis Ilias

Advisor: Papadourakis George

ABSTRACT

The purpose of this thesis is the designing and development of a security system that can help a company and its workers with a variety of capabilities such as indoor location tracking and communication with a security supervisor.

That security system consists of 2 web pages for the administration of the maps and the employees' registration. There is another web page that only the security manager uses which allows the communication with an employee and, with his approval, the location tracking system that is meant for guiding. It also consists of 2 Android applications, one for the employees and the second one for the visitors. Lastly there is the backend system that manages all the data the websites provide making the communication of the websites with the applications feasible.

ΣΥΝΟΨΗ

Ο σκοπός της παρούσας πτυχιακής είναι ο σχεδιασμός και η ανάπτυξη ενός συστήματος ασφαλείας το οποίο θα μπορεί να βοηθήσει μια εταιρεία και τους εργαζομένους σε αυτή με δυνατότητες όπως ο εντοπισμός θέσης σε κλειστό χώρο και η επικοινωνία με τον υπεύθυνο ασφαλείας.

Αυτό το σύστημα ασφαλείας εμπεριέχει 2 ιστοσελίδες για την διαχείριση των χαρτών και των εγγραφών των εργαζομένων. Υπάρχει άλλη μία ιστοσελίδα όπου την διαχειρίζεται ο υπεύθυνος ασφαλείας η οποία καθιστά δυνατή την επικοινωνία του με τους εργαζόμενους και, κατόπιν συγκατάθεσης του εργαζόμενου, την γνωστοποίηση της τοποθεσίας του για καθοδήγηση. Επιπλέον αποτελείται από 2 Android εφαρμογές εκ των οποίων η μία απευθύνεται στους εργαζόμενους και η $2^{η}$ στους επισκέπτες. Τέλος υπάρχει ένα σύστημα (backend) το οποίο διαχειρίζεται όλες τις πληροφορίες που προσφέρουν οι σελίδες καθιστώντας έτσι εφικτή την "επικοινωνία" των ιστοσελίδων με τις εφαρμογές.

ACKNOWLEDGEMENTS

We would like to thank our professor Dr. George Papadourakis for the opportunity he gave us to participate in this project as well as his guidance and advices during it.

Also we would like to thank our friends in the Technological Institute of Crete for supporting us mentally in the procedure of this Thesis.

Finally we would like to thank the students and the professors of the Blended AIM 2017 for the great experience of working with such a team on a big project for a company.

DEDICATION

Giannantonakis Ioannis:
This thesis is dedicated to my parents who helped me all these years.
To the friends that supported me mentally.

Logothetis Ilias:
This thesis is dedicated to my family that supported me all these years and without them it would not be possible to complete it.

Also to all my friends that helped and encouraged me throughout the project.

Table of Contents

Table of figures

Table of pictures

CHAPTER 1: INTRODUCTION

1.1 Summary

This thesis is part of an international Erasmus+ project named Blended-AIM (Academic International Mobility), where students from Educational Institutes around Europe formed a team to develop an application requested by a company. The goal was to design and develop an application that will allow a security supervisor to track, guide and communicate with a company's employees to make everyone feel safer as well as to help with building directions.

To achieve that our team was split further into a backend team, a frontend team, an android team, a design team and a business team. In the backend technologies like Visual Studio IDE, C#, PostrgreSQL, vtortola library for the WebSockets, Npgsql and Entity Framework were used.
In the frontend technologies like AngularJS, NPM and Bower package managers and Jenkins server were used.
In the android team technologies like WebSockets, Rest API, Java, Android and Estimote beacons API.

The project was hosted at a TFS (Team Foundation Server) server from where the android application and the frontend could communicate with the backend services.

1.2 Motive

Nowadays many companies are hosted altogether in huge compounds where each company rents a number of buildings that might not be physically connected and even if they are, the traverse from a room to another might pose a challenge for visitors or newcomers. Furthermore, what happens in an emergency situation? How fast can the employees evacuate the building? What if someone is trapped in a room or even worse if he fainted?

That's where Gabriel kicks in. Gabriel is an Android application that can help a supervisor track, guide and communicate with the employees of a company from his computer. Also through a website the company can configure and edit the building maps allowing them to expand or even add new floors and rooms. Finally, the employee management (add/delete/edit) is also done through another website.

1.3 Blended Aim

Blended AIM (Academic International Mobility) is an Erasmus+ funded project made to promote students' employability and support companies hosting internships. Every year 10 educational institutes from European countries, like Portugal, Greece, Belgium, United Kingdom, Germany, Iraq, Austria and Italy send up to 2 students each to form a team. The purpose of that is to support the students develop soft skills in an international environment by means of blended mobility.

At this moment, a student's professional career depends on mobility and demands certain intercultural skills. However, most institutes don't have courses that provide international exposure. Blended mobility helps the students adapt and learn but it's hardly considered, let alone used, a solution to international mobility's problems.
Blended Aim sets the foundation to promote and test blended mobility by providing the resources, training, supporting tools and information to the students and the companies that host internships.
For that reason Praxis was created. Praxis is a consortium of higher education institutions, companies, associations, research labs and chambers of commerce, all committed to enhance a student's or a company's Project/Internship experience and to promote innovation in the field.

1.3.1 How Blended AIM works

Every year at its beginning students from 10 educational institutes (2 from each) gather to take up on a project given from a company. That project is considered as a course and each student gets ECTS after the project's completion. The students are from different study fields such as computer informatics, graphic design and business management so that the project can be completed. The project is usually product that helps solve some of the company's problems. During the time of the entire project there are two important meetings. In the first one the students meet with each other, their professors and the company's representatives to discuss how to approach the project and how to work on it effectively. In the second one the students present the product to the teachers and the company's representatives and after that they get appraised by the teachers. During the time between the two meetings the students perform online meetings to showcase their work and discuss with each other their ongoing work and the problems that they may be facing.

CHAPTER 2: FUNDAMENTALS

2.1 Analysis and Development Methods

2.1.1 Nexus Scrum

Software development is a complex and challenging work to do on its own. Even companies face difficulties in the development section and these difficulties set them back on their work load. Scrum framework can help but it is not enough on its own. To get a step further Nexus framework is required as well.

Nexus is a framework, based on the principles of Scrum and the Agile Manifesto, that helps the developers by minimizing cross-team dependencies and integration issues. It was developed by Ken Schwaber, the co-creator of Scrum and founder of Scrum.org, in 2015. Nexus framework is an extension to the Scrum framework and it uses an iterative and incremental approach to scaling software and product development. It allows multiple Scrum teams, that work on a product, to unify as a single larger team.

Nexus framework consists of 3-9 scrum teams that work on the same product backlog for the successful development of a product. The main goal is the identification of cross-team issues and making sure that integration tools are understood and used. Integration team is a new feature in the Nexus framework and it is the team that is accountable for the successful integration of all work created by all the Scrum Teams in a Nexus. It consists of a Product Owner, a Scrum master and few team members of each scrum team in a nexus.

Each scrum team needs to be represented by a team member. Then the representatives meet to identify, resolve in-Sprint dependencies and set the goals that they want to achieve in each Sprint. These steps explain the "Nexus Sprint Planning" term which is crucial in a Nexus framework.

Furthermore, Nexus Daily Scrums help the teams to plan correctly their next steps by inspecting the integrated work. Nexus Daily Scrum is a brief meeting between the Scrum teams' members. Not everyone has to attend but each team's representative must.

2.1.2 Microsoft TFS

Team Foundation Server (TFS) is an integrated server suite of developer tools, released by Microsoft, that provide source code management, requirements management, reporting, project management, automated builds, testing and release management capabilities. It covers the entire application lifecycle and can be used as a back-end to numerous Integrated Development Environments.

The source code management can be achieved either with Team Foundation Version Control or with Git. Team Foundation Version Control is a control system that allows teams to store any type of artifact within its repository. It uses two types of workspaces when working with client tools, the Server Workspaces and the Local Workspaces. Server workspaces allow developers to lock files for check- out and provide notifications to the other developers that files are being edited. Local workspaces allow developers to edit the files, as long as they are on his local machine, without checking out before working on them. Git support was added on TFS with the release of TFS 2013 based on the libgit2 library. Because of that, any Git client can be used natively with TFS.

TFS consists of a data warehouse which is a relational database and a SQL Server Analysis Services data cube. Both of these sources are available for reporting through SQL Server Reporting Services when this option is installed allowing for reports that cover Build information, Test results and progress, project management, agile reports and bug data. Also with the release of TFS 2013 a new feature called "light-weight reporting" was introduced. This feature allows the creation of real-time reports based on query results that do not rely on the warehouse or cube.

TFS also includes a build server application called Team Build that consists of MSBuild and Windows Workflow Foundation (WF). MSBuild is a declarative XML language and WF is a Microsoft technology that controls the overall flow of the build process. Team Build has the ability to report on the changes in each build as well as test results. Combined with the testing tools, testers then get an integrated view of what code was changed in each build, but also which bugs, PBIs and other work changed from build to build.

TFS also offers a tool which helps with the project management called Release Management. The Release Management capabilities give teams the ability to perform a controlled, workflow (provided by Windows Workflow Foundation) driven release to Developer, Test and Production environments and provides dashboards for monitoring the progress of one or more releases.

CHAPTER 3: WORK PLAN

3.1 State of the Art

Whether the end goal is customer engagement, improved productivity or risk mitigation, businesses need accurate indoor location information. The signals from the satellites are attenuated and scattered by roofs, walls and other objects rendering the use of GPS systems inadvisable so what technology should a company strive for? There are various technologies that can be used for indoor location like Beacons, Wi-Fi, RFID and even light. The best solution would be a combination of those mentioned above but that is usually more complex and less cost efficient for companies. So as a simple pick the solution of beacons stands out ahead of the competition.

Beacons use low energy Bluetooth technology to broadcast signals with information such as its unique identification number. They can't store and broadcast larger amount of information so these IDs can be mapped to certain areas of the company's buildings by the developers of the indoor location system application or an administrator. These maps are usually stored during the installation of the application on the device so that when it detects the signal of a beacon it can map its own position. With an effective range of up to 50 meters (at this moment) beacons can be used as proximity checkpoints for mapping a building. Achieving an accurate position is possible by using more beacons to compute the device's position by applying the method of triangulation.

3.2 Technologies

For this project technologies like WebSockets, Rest API, Java, Android, Angular, Entity Framework, PostrgreSQL, Estimote and several Design Patterns were used. Bellow, information on the above-mentioned technologies are given.

3.2.1 WebSockets

A WebSocket represents a persistent long-held bi-directional real-time TCP socket connection between a client and a server allowing full duplex messages to be instantly distributed independently with little overhead resulting in a very low latency connection.

WebSockets establish a TCP-style connection through a HTTP request to the desired server which the server should accept, through a response, after the authentication and authorization of the client (handshake). After that the connection remains intact unless redeployment is needed (usually because of workload redistribution).

Also, cross-domain communication has been considered from day one and is dealt with within the connection handshake thus allowing the use of services such as Push and Comet offering a massively scalable real-time platform that can be used by any website, desktop or mobile application.

They are preferred by developers because of their ability to deliver downstream (server to client) messages fast and without the unnecessary headers of HTTP requests/responses but as all modern technologies there are some cons. One of the most significant problems is that if a connection stays up for too long something might “kill” it, making the retransmission of data from the server or the client harder and time consuming.

3.2.2 Android

Android is an open source mobile operating system based on the Linux kernel and designed for touch screen mobile devices. Android was initially developed by Android Inc. which Google bought in 2005. As of May 2017, Android has 2 billion monthly active users and the largest installed base of any mobile operating system.

Its open source is what attracted many developers but more importantly technology companies that require a low-cost and customizable operating system for their high-tech devices. By having such a huge development community, the variety of applications, tutorials on development, development tools and supported programming languages is increasing.

Applications are developed using the Android Software Development Kit (SDK) and usually the Java programming language through the Android Studio or Eclipse Integrated Development Environments (IDEs).

3.2.3 Java

Java is a powerful class-based and object-oriented computer programming language that was developed by James Gosling for Sun Microsystems which was acquired by Oracle Corporation. Java is one of the most popular programming languages in use especially for client-server web applications, with a reported 9 million users. It was based on C/C++ syntax design which application developers found familiar. Originally it was designed for interactive television but it was too advanced for the digital cable television at the time (1991).

Java's major goal is portability which is accomplished by compiling the Java code to Java bytecode instead of machine code. Java bytecode instructions are analogous to machine code, but they are intended to be executed by a virtual machine written specifically for the host hardware. Because Java bytecode runs on any platform that supports a Java Virtual Machine the term "write once, run anywhere" was introduced to application developers.

After the acquisition of Sun Microsystems by Oracle Corporation on January 27, 2010 the Java platform implementation was split into two different distributions. The Java Runtime Environment (JRE) which contains the parts of the Java SE platform required to run Java programs, and the Java Development Kit (JDK) which is intended for software developers and includes developments tools such as the Java complier, Javadoc and a debugger.

As mentioned Java is an object-oriented programming language. That means that the memory management gets more difficult for the programmers due to pointer references. For example, if the programmer holds the reference to an object that no longer exists or is needed then a "null pointer exception" error is thrown. To solve these problems Java is accompanied by Automatic Garbage Collectors that keep the memory up to date by managing the pointers of the created objects. That is the reason why Java does not support C/C++ style pointer usage.

Java is used for the development of Android applications through Android Studio or Eclipse IDEs but the Java bytecode runs on its own virtual machine, optimized for low memory devices.

3.2.4 Rest API

The term Representational state transfer (REST) REST was defined by Roy Fielding in his 2000 PhD dissertation “Architectural Styles and the Design of Network-based Software Architectures”. REST is a way of providing web services between computer systems over the Internet. RESTful web services allow the manipulation of Web resources by using a predefined set of stateless operations. Other web services such as SOAP expose their sets of operations and that is why REST web services are preferred at the moment.

In a RESTful web service by using the HTTP verbs GET, POST, PUT, DELETE we get responses from the resource provider that are usually in XML, HTML, JSON or some other defined format. REST systems aim for fast performance, reliability, and the ability to grow, by re-using components that can be managed and updated without affecting the system as a whole, even while it is running.

There are six constraints that define a RESTful system. By operating within these constraints, the service gains performance, scalability, simplicity, modifiability, visibility, portability and reliability. These constraints are:

- Client-server architecture
  By separating the user interface concerns from the data storage concerns, there is an improvement in the portability of the user interface across multiple platforms and improve scalability by simplifying the server components.

- Statelessness
  Each request from any client contains all the information necessary to service the request, and session state is held in the client.

- Cacheability
  Responses must therefore, implicitly or explicitly, define themselves as cacheable or not to prevent clients from reusing stale or inappropriate data in response to further requests.

- Layered system
  A client cannot ordinarily tell whether it is connected directly to the end server, or to an intermediary along the way. Intermediary servers may improve system scalability by enabling load balancing and by providing shared caches.

- Code on demand
  Servers can temporarily extend or customize the functionality of a client by transferring executable code.

- Uniform interface
  Decoupling the architecture enables each part to evolve independently.

3.2.5 Retrofit

Retrofit is a library for REST client for Android or Java in general, developed by Square. Retrofit provides a framework for authenticating and interacting with APIs and sending network requests with OkHttp. The response that usually contains JSON or XML files, is handled easily by parsing it into a Plain Old Java Object. For the parsing to be done the definition of those objects must be defined as classes according to the response.

In the past, Retrofit relied on the Gson library to serialize and deserialize JSON data but with the release of Retrofit 2 there are more parsers for processing network response data such as Moshi and Protobuf.
Although, if a user wants to manually create new Java classes for resources he should use the Gson library.

3.2.6 Estimote Beacons

Estimote Beacons are developed by Apple on top of Bluetooth Smart technology, allowing for location tracking application development. They work by broadcasting packets of data, containing their beacon ID and information such as signal strength and battery level so that the phone knows which beacon it detects and how far it is. It also broadcasts two more values (major and minor) that allow further spatial information such as rooms or other areas of interest. These values can be modified manually by the user.

Estimote SDK can be used for proximity or indoor location applications. The device that runs that application can distinguish the signals that are being constantly sent from the beacons and estimate its location in a known area.

3.2.7 AngularJS

AngularJS is a JavaScript framework that lets you extend HTML vocabulary for your application. The resulting environment is extraordinarily expressive, readable, and quick to develop. It is fully extensible and works well with other libraries.

AngularJS uses data binding which allows the updating of the view whenever the model changes, as well as the updating of the model whenever the view changes, thus eliminating the DOM manipulation. Models are plain old JavaScript objects making the code easy to test, maintain and reuse. It also supports deep linking so that users can bookmark and email links to locations within the app. Client-side form validation is also supported by AngularJS as it lets you declare the validation rules of the form without having to write JavaScript code.

Dependency injection is a core to AngularJS. This means that there is no need for a main() method in the application and that any component that doesn't fit the user's needs can easily be replaced. AngularJS was designed to be testable so it encourages behavior-view separation and takes full advantage of dependency injection.

3.2.8 PostgreSQL

PostgreSQL (Postrgres) is an object-relational database management system that focuses or extensibility. It was an evolution of the project Ingres at the University of California, Berkeley lead by Michael Stonebraker. On 1996 its name was changed to PostgreSQL from Postgres95 to reflect its support for SQL. At this time PostgreSQL is the most advanced and features-rich open-source database management system.

The concurrency is dealt through a system known as multiversion concurrency control (MVCC) that allows changes to be made without being visible to other transactions until the changes are committed. This ensures that PostgreSQL remains ACID-compliant (atomicity, consistency, isolation, durability).

PostgreSQL includes binary and synchronous replication. Binary replication replicates nodes asynchronously, with the ability to run read-only queries against these replicated nodes, allowing better efficiency in splitting the read traffic among multiple nodes. Synchronous replication ensures that, for each write transaction, the master waits until at least one replica node has written the data to its transaction log. This can be useful for workloads that do not require such guarantees but it has some negative effect on performance due to the requirement of the confirmation of the transaction reaching the synchronous standby.

It also includes support for regular B-tree and hash indexes, and four index access methods: generalized search trees (GiST), generalized inverted indexes (GIN), Space-Partitioned GiST (SP-GiST) and Block Range Indexes (BRIN). Complex queries are dealt by using multiple indexes together to create temporary in-memory bitmap index operations. User-defined index methods can also be created but it is a complex process.

The data types supported by PostgreSQL vary and include data types such as Boolean, Dates, Bit strings, Composite, HStore, Arrays up to 1 GB, Geometric primitives JSON and a faster binary JSONB. In addition to these the users can create their own data types which can usually be made fully indexable via PostgreSQL's indexing infrastractures.

3.2.9 Entity Framework

The Entity Framework is an open source object-relational mapping framework that was included in .NET framework until the release of Entity Framework version 6 with which it separated from .NET framework.

With the Entity Framework, the developers can work on a higher level of abstraction when they deal with data stored in databases, and can create and maintain data-oriented application with less code than in traditional applications.

The architecture of Entity Framework, from the bottom up, consists of the following:

- Data source specific providers.
  The database that the framework interfaces.

- Map provider.
  It translates the Entity SQL command tree into a query in the native SQL of the interfaced database.

- EDM parser and view mapping.
  It creates views, from the relational schema, of the data and aggregates information from multiple tables in order to aggregate them into an entity and splits an update to an entity into multiple updates to whichever table(s) contributed to that entity.

- Query and update pipeline.
  It converts queries to canonical command trees which are then converted into store-specific queries by the map provider.

- Metadata services.
  They handle all metadata related to entities, relationships and mappings.

- Transactions.
  Integration of transactional capabilities of the underlying store.

- Conceptual layer API.
  The runtime that exposes the programming model for coding against the conceptual schema by following the ADO.NET pattern.

- Disconnected components.
  They locally cache datasets and entity sets for use.

- Embedded database.
  A lightweight database for client-side caching and querying of relational data.

- Design tools.
  Tools, such as Mapping Designer, that simplify the mapping of a conceptual schema to a relational schema and specify which properties of an entity type correspond to which table in the database.

- Programming layer.
  Services that expose the EDM as programming constructs which can be consumed by programming languages (Object services, Web services).

- High-level services.
  Services which work on entities rather than relational data.

Entities represent the individual instances of the objects to which the information pertains. Entity types define the class an entity belongs to and also defines what properties an entity will have. Also, all entity types belong to a namespace and have a unique EntityKey property that identifies each instance.

Entity SQL queries are parsed into a command tree, segregating the query across multiple tables, which is handed over to the EntityClient provider. The EntityClient then converts the command tree into an SQL query in the native flavor of the database, which after the execution of that query returns an Entity SQL ResultSet.

3.2.10 SvgPanZoom library

This is a library that adds pan and zoom features to Scalable Vector Graphic (SVG) images. Four tools accompany that library:

- Pan.
  A tool which allows the user to drag an image around within the viewer.
- Zoom
  The user can scale the image either with a point click or by selecting a region to zoom the specified area.
- None
  The user can interact with SVG child elements and trigger events in contrast to the above-mentioned tools where he cannot interact with SVG child elements.
- Auto
  The user can perform all of the above-mentioned actions on the SVG child elements.

3.2.11 Materialize library

Materialize is a responsive front-end framework based on Material Design design language developed by Google in November 8, 2015. It provides default stylings, refined animations and transitions making the life of a developer easier. Additionally, a single underlying responsive system across all platforms allows for a more unified user experience.

The Designer and Google's Vice President of Design Matias Duarte stated: "unlike real paper, our digital material can expand and reform intelligently. Material has physical surfaces and edges. Seams and shadows provide meaning about what you can touch." Material Design makes more liberal use of grid-based layouts, responsive animations and transitions, padding, and depth effects such as lighting and shadows.

3.2.12 Design Patterns

Design patterns are general reusable solutions on commonly occurring problem in software design. They can be described as formal best practices that can be used from a programmer on designing an application or system to solve problems or to prevent them. Design patterns gained popularity in computer science by the book "Design Patterns: Elements of Reusable Object-Oriented Software" that published in 1994 by the "Gang of 4".

Design patterns can help to speed up the development process and make the code more readable to those familiar with them. To use them requires consideration of issues that might occur later, so by using them it will help to prevent the problems. On every application, a design pattern must be programmed from start. Patterns usually show relations and interactions between classes or objects, without specifying the final application classes. They often described using UMLs so the relations and interactions can be easily understandable by developers.

Design patterns were originally grouped into three categories: creational patterns, behavioral patterns and structural patterns. They were 23 in number, but new patterns are presented by the years, having a result of new categories such as the architectural design pattern that may be applied at the architecture level of the software such as the Model–View–Controller pattern.

CHAPTER 4: MAIN PART

4.1 Problem Analysis

Trilogis is an Italian computer company, innovative, dynamic and constantly growing. Trilogis has been founded in 2006 by Gianni Rangoni, Nicola Giuliani and Massimo Barozzi. The name Trilogis comes from the combination of the words "trilogia" (trilogy) and the acronym "GIS" (Geographic Information System). Trilogis is specialized in advanced solutions in the fields of geography and computer science and in the management and tracking of people and equipment both outdoor and indoor. In close collaboration with the customer, Trilogis models and customizes geographical and information systems, as needed, with the expertise in many business contexts.

Despite its relatively recent foundation, Trilogis accomplished and participated in many projects in collaboration with companies, universities, research institutions and government agencies. Some of these projects are Mepi, MoPAL, iCore, Centric, Seneca, giCASES and i-locate.

One of Trilogis' most recent projects and the "progenitor" of the project analyzed in this thesis is the i-locate project. The goal of i-locate is to simplify the life of users by helping them to navigate inside buildings to reach their destination and by providing any other supportive information available using their smartphone. I-locate also provides indoor tracking of objects and portable equipment for their management and maintenance.

4.1.1 Problem description

During the first meeting in Rovereto the Project manager of the company addressed a huge problem for companies by staging a story. The story's main subject was the injury of an employee in a factory. The incapability of the medical team knowing his location did cost them 2 hours of searching, risking the employee's life. That incident inspired Trilogis to act by assigning on Blended AIM 2017 team the development of a security system that supports indoor position tracking and communication.

Trilogis also decided to expand their field of interests to not just employees but also the visitors of a company's buildings. So, the Blended AIM team was called to make another application for the guidance of the visitors through the company's buildings.

4.1.2 System Requirements

To complete the project that was given the students had to decompose the main task into simpler and shorter ones that could be handled more effectively. The basic need for the project was a site, an android application and the backend services. The site would allow a security supervisor to manage the building maps, the employees and the communication with them. The android application would be the employee's tool to navigate through the building safely and communicate with the supervisor in cases of emergency. The backend services would take care of data storage and manipulation thus allowing the correct connection of the site with the android application.

For better workflow management and easier achievable tasks, the students split the components into two additional sites and one additional application to scale down the work load of each individual task. So, the final decomposition consisted of 3 web pages (one for the map editing, one for employee management and one that would be used by the security supervisor), 2 android applications (one for usage by the employees and one for visitors) and the backend services.

Dummy users were created to give the notion of personalization and to help with the user stories separation. These users were Gabriel (the security supervisor), Joe (the employee), Maria (the employee manager) and the map editor. For these users, user stories were created in each sprint as upcoming goals.

Sprint 1
As Gabriel, I want to send pings to Joe.
As Gabriel, I want to know Joe's last known approximate position.
As Gabriel, I want to modify Joe's routes.
As Gabriel, I want to get alerted if Joe is off course.

Sprints Table 1

Sprint 2
As Gabriel, I want to get alerted if Joe does not clear a checkpoint on time.
As Gabriel, I want to modify Joe's routes.

Sprints Table 2

Sprint 3
As Maria, I want Joe's application to collect information from the beacons and notify me if a beacon is running low on battery.
As Joe, I want to be able to send text messages to Gabriel.

Sprints Table 3

Sprint 4
As the map editor, I want to be able to snap to the grid and change the grid size.
As the map editor, I want to be able to create rooms.
As Gabriel, I want to be able to send text messages to Joe.

Sprints Table 4

Sprint 5
As Gabriel, I want to be able to configure the list of predefined ping messages.
As Gabriel, I want quick access to emergency procedures when an alert is triggered.
As Gabriel, I want to be able to call emergency services directly from the interface.

Sprints Table 5

Sprint 6
As Joe, I want to be notified if Bluetooth on my device is off.
As Joe, I want to be allowed to change my user password whenever I want.
As Joe, I want to be able to delay my estimated time of arrival.
As Joe, I want my mobile phone to monitor inactivity periods.
As Joe, I want to have a visual reminder that I am being tracked.
As Joe, I want to call Gabriel directly from within the application.

Sprints Table 6

Sprint 7
As Gabriel, I want to modify Joe's route.
As the map editor, I want to be able to delete walls/objects.
As Joe, I want to answer to Gabriel's ping.
As Joe, I want to get information about the nearest exit.

Sprints Table 7

Sprint 8
As Maria, I want to be able to review the accepted user agreements.
As Maria, I want to be able to create new accounts/profiles and assign roles.
As Gabriel, I want the mobile application to track Joe's progress through the route.
As Gabriel, I want to get notified if Joe disables the tracking feature on his mobile application.
As Joe, I want to be able to accept the usage agreements the first time I use the application.
As Joe, I want the mobile application to track my progress through the route.
As Joe, I want to trigger the start of a new round from the mobile application.
As Gabriel, I want to know Joe's estimated time of arrival.

As Gabriel, I want to have access to basic information about Joe.
As Gabriel, I want to set Joe's route duration (approximately).
As Gabriel, I want to be able to delay Joe's estimated time of arrival.
As Gabriel, I want to receive Joe's inactivity alerts.
As Gabriel, I want to be notified when Joe delays the estimated time of arrival.
As Gabriel, I want to be notified if Joe appears offline.
As Gabriel, I want to be notified if Joe's phone is running low on battery.
As the map editor, I want to be able to snap to a segment.
As the map editor, I want to be able to draw doors/stairs.
As the map editor, I want to be able to edit existing walls/rooms.

Sprints Table 8

4.2 Implementation

4.2.1 Project Implementation

In Blended AIM 2017 participated 20 students from universities in Germany, Portugal, Greece, Austria, Belgium, Slovenia, Scotland and Iraq. At first, they had to get acquainted with each other and also with the Scrum framework that they would use for the agile software development. Following the Nexus Scrum, they rearranged the initial team into separate smaller ones consisting of the design team, the frontend team, the backend team, the android application team, the business team and the integration team. The rearrangement was done according to the students' educational field. The first Sprint took place in Rovereto, Italy (Trilogis company headquarters) and it went mostly smooth after resolving some issues on the technologies that would be used and on the final product. After that meeting each student returned to his country and that made the project management a lot harder. Each Monday night the students discussed about the upcoming week's user stories and what was needed to accomplish them. Also, every Thursday night a sprint review meeting was held were every student had to present the work that was done.

The whole project was divided in 8 sprints, one every two weeks, until its completion. There was daily communication between the students through the Slack application. Meetings were hosted on the Adobe Connect that was provided by the University of Paderborn (as a trial version). That type of communication posed many problems such as very long meetings, no clarification on the tasks, adobe

plug-ins not being able to identify some microphones and the scrum master being unable to keep everyone under control.

All the user stories were held in the Microsoft Team Foundation Server 2015. Every scrum team had its own user stories and each user story had its own tasks. In the end of each Sprint the product owner had to assign new user stories for the upcoming Sprint. When a user story wasn't complete for the Sprint it was transferred to the next Sprint marked as delayed.

The last Sprint took place in Graz, Austria (in FH Joanneum University) where the final presentation of the product would take place. During the last week of the last Sprint students worked together again to finalize their work, fix existing bugs and problems. Demo runs were prepared by mapping specific places in the building to make sure that every feature of the application was presented and preserve the certainty of a smooth run. Also voice covers were created by each student to create an inspiring speech about the product. At the end of the product presentation the students' work was evaluated by the company representatives and the professors. The students also were rated for their participation by each other.

4.2.2 Detailed Android Implementation

The students of TEI of Crete were assigned with the development of the Android application so bellow a further explanation of the user stories concerning the application is shown. Furthermore, some of the code as well as certain techniques that were used for the completion of the application will be explained.

As Joe, I want to be able to send text messages to Gabriel.	
1. Send text messages through WebSockets. 2. UI for the chat.	A new activity/fragment was created for the chat. New WebSocket message type was created. New Json deserializer was created.

Android User Story 1

As Joe, I want to be notified if the Bluetooth on my device is off.	
1. Check Bluetooth connectivity and state. 2. Notify the user if the Bluetooth is off or not supported.	Through the Android API we check if the Bluetooth is supported and if it is we then check its state. Then we notify the user with a notification or an alert box.

Android User Story 2

As Joe, I want to answer to Gabriel's ping.	
1. Display a ping message on screen and push notifications. 2. Send back a ping reply using WebSockets. 3. UI for messages. 4. UI for Joe emergency messages. 5. Implement a flashlight alert. 6. Implement a vibration alert. 7. Implement a sound alert. 8. Understand WebSocket and Json. 9. Implement the simple SOS WebSocket emergency message and the Json deserializer. 10. Implement the WebSocket message for each emergency type. 11. Send emergency messages using the WebSockets.	A folder that holds all notification types such as flashlight, vibration and sound was created. WebSockets were implemented. Custom message types were created, one for each emergency message, for sending through WebSockets. Json deserializers for each message type were implemented. User Interface to display the messages. User Interface to allow Joe to answer to a message.

Android User Story 3

As Joe, I want to be able to accept the usage agreements the first time I use the application.	
1. Create UI for usage agreements. 2. Run this activity after the application is installed. 3. Check for new license agreements every time the application starts. 4. Save the user that accepted the user agreements on a given day at a given time.	A new activity for the User Interface of the usage agreements was created. After login check if the user has accepted the latest usage agreements. If not start the activity of the usage agreements so that the user can read and accept them. If the user accepts send on the backend the day and the time that the user accepted the terms.

Android User Story 4

As Joe, I want the mobile application to track my progress through the route.	
1. UI to start the route. 2. Beacon listener. 3. REST request to load route, checkpoints and associated beacon data. 4. REST request to send the last beacon passed. 5. Implement the REST API controller that handles	A new activity/ fragment for the User Interface was created. Implemented beacon listener classes. Implemented the REST requests to handle these tasks. Implemented the classes that the REST calls

the getting and posting of routes and checkpoints. 6. Implement the REST API controller that handles the last beacon passed in the route. 7. Implement the route domain entity. 8. Implement the checkpoint domain entity. 9. Implement the checkpoint route and beacon classes. 10. UI showing progression through the route (beacon passed and the next).	required. Implemented the classes on the backend. Implemented the REST API handlers on the backend.

Android User Story 5

As Joe, I want to trigger the start of a new route from the mobile application.	
1. Send a REST request to trigger a new route. 2. UI to start a shift. 3. Send a REST request to trigger the start of a new shift. 4. Refactor the code of the main activity so it won't go back to the main page if a bearer token is already on system. 5. Check if the application goes to the background. 6. Check if the application comes back to the front. 7. Implement REST API controller to allow basic CRUD for a route. 8. Implement the domain entity Route. 9. Implement the domain entity Shift. 10. UI to show the completed Routes. 11. REST request to fetch the Route history.	The Shift starts automatically when the user logs in. Log in and main activity were refactored to support the changes. New REST calls were created. A new class was created to check if the application goes to the back or to the front. On the backend all the CRUD (Create, Read, Update, Delete) for the route were created. On the backend the Route and Shift entities (classes) were created. A new activity to show the completed Routes was created.

Android User Story 6

As Joe, I want to be allowed to change my user password whenever I want to (in-app).	
1. UI for the password change. 2. Check if the old password was given correctly by the user. 3. Send the new password to the backend.	A new Activity for the User Interface was created. The user is prompted to type his old password. On success he is prompted to type a new password twice for the evaluation. The new password is sent to the backend to be saved in the database.

Android User Story 7

As Joe, I want to be able to delay my estimated time of arrival.	
1. New button for the delay. 2. New WebSocket message type. 3. Send the delay message through WebSockets.	A new button was created on the Route activity/fragment. A new WebSocket message type for the delay was created. A new Json deserializer was created. A new WebSocket message type for the delay was created on the backend. A new Json deserializer was created on the backend. On button click a delay message is sent.

Android User Story 8

As Joe, I want my mobile phone to monitor inactivity periods.	
1. Check if the user uses the application. 2. Check if the user is moving. 3. Alert if inactivity period is detected. 4. New WebSocket message type. 5. Send alert message through WebSockets.	Detect touch events and accelerometer. Pop-up alert with sound/flashlight/vibration to check if the user is alright. If not answered in a given time an alert message is sent through WebSockets. New WebSocket message type was created both on Android and on the backend. New Json deserializer was created both on Android and on the backend.

Android User Story 9

As Joe, I want to have a visual reminder that I am being tracked.	
1. UI for the tracking reminder. 2. Alert box that shows that the tracking is on.	A User Interface with animation, when the tracking is on, was created. On tracking activation an alert box reminds the user that he's being tracked.

Android User Story 10

As Joe, I want to call Gabriel directly from within the application.	
1. Emergency button for phone call. 2. Android permissions for the direct call.	A new button was added to call a predefined number. Android permissions were added to allow a direct call through an application.

Android User Story 11

4.2.2.1 User Interface Design

At first the design was made by following the demo that was created at Rovereto by the design team. The login functions were built on top of that first design. The prototypes of the emergency and chat functions were also added. Even though it was functional, the other teams requested the use of more colors. So, the second approach was given influenced by material design standards. The application felt like an entertainment application and not one used by employees for their work. Also, the tracking interface was not sufficient for the team's needs. The application's performance was found to be worse than expected due to the use of cut images from the files of the design team.

A third release of the design was necessary to rectify the issues mentioned above and to improve the looks of the application. The new design was a mix of the first and second ones with revised colors and a new text font called Montserrat. Following the Google design standards, a menu bar was added on the left side with the default menu button. All the icons and logos were defined as Scalable Vector Graphics (SVG) files to relieve the workload and achieve better performance. As a last change, a new application logo was created by the design team.

Every part consisting the design was placed under the resources folder which is auto generated by Android Studio IDE. The resources folder consists of sub-folders also auto generated by Android Studio. These folders help to organize the structure of the project by separating the parts that compose the application, according to their purpose. These folders are:

- Drawable.
  This folder contains graphic components that can be drawn to the screen. They can be retrieved with API calls such as getDrawable(int), or applied to another XML resource with attributes such as android:drawable.

- Layout.
  In this folder the activity and fragment layouts are generated.

- Mipmap.

  This folder contains all the images that are used by the application. Every image loaded in this folder is set in 5 different dimensions to match every phone or tablet display size.

- Values.

  In the Values folder general information about the application's design is being kept. It consists of more sub-folders:

  - Colors.

    In this folder a value is assigned to a desired name that will be used to define the application's main color. For example:

```
<color name="gabrielBlue">#0000ff</color>
```

  - Dimens.

    In this folder values are assigned to desired names that will be used for certain dimension variables. For example:

```
<dimen name="activity_horizontal_margin">16dp</dimen>
<dimen name="activity_vertical_margin">16dp</dimen>
```

  - Strings.

    In this folder every text is saved as a string value for correct text management. For example:

```
<string name="navigation_drawer_open">Open navigation drawer</string>
<string name="navigation_drawer_close">Close navigation drawer</string>
```

  - Styles.

    In this folder styles and names for them as well as their values are defined. For example:

```
<style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">

    <item name="colorPrimary">@color/gabrielBlue</item>
    <item name="colorPrimaryDark">@color/gabrielBlue</item>
    <item name="colorAccent">@color/gabrielBlue</item>

</style>
```

Under the Drawable folder the SVG files were placed for proper display. An SVG file consists of code similar to the following:

```
<?xml version="1.0" encoding="utf-8"?>
<vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:viewportWidth="258"
    android:viewportHeight="258"
    android:width="258dp"
    android:height="258dp">
    <path
        android:pathData="M129 5c33.1 0 64.3 12.9 87.7 36.3 23.4 23.4 36.3 54.6 36.3 87.7 0
33.1 -12.9 64.3 -36.3 87.7C193.3 240.1 162.1 253 129 253 95.9 253 64.7 240.1 41.3 216.7
17.9 193.3 5 162.1 5 129 5 95.9 17.9 64.7 41.3 41.3 64.7 17.9 95.9 5 129 5m0 -5C57.8 0 0
57.8 0 129 0 200.2 57.8 258 129 258 200.2 258 258 200.2 258 129 258 57.8 200.2 0 129 0l10
0z"
        android:fillColor="#303030" />
    <path
        android:pathData="M102.1 169.2l-19.6 0c-0.4 0 -0.8 -0.4 -0.8 -0.8l0 -50.7c0 -0.4
0.4 -0.8 0.8 -0.8l19.7 0c0.4 0 0.8 0.4 0.8 0.8l0 50.7c-0.1 0.4 -0.4 0.8 -0.9 0.8l0 0zm0 0"
        android:fillColor="#303030" />
    <path
        android:pathData="M137.9 79.5c-5.8 -4.2 -13.7 -1.8 -14 -1.7 -0.8 0.2 -1.3 1 -1.3
1.8l0 17.4c0 5.9 -2.8 11 -8.4 14.9 -4.3 3.1 -8.7 4.3 -8.8 4.4 0 0 -0.1 0 -0.1 0l-1 0.3c-0.6
-1 -1.7 -1.7 -3 -1.7l-18.1 0c-1.9 0 -3.4 1.6 -3.4 3.4l0 49.1c0 1.9 1.6 3.4 3.4 3.4l18.1
0c1.6 0 3 -1.1 3.3 -2.7 2.4 2.6 5.9 4.2 9.7 4.2l42.8 0c9.1 0 14.9 -4.8 15.9 -13.1l5.5 -
34.7c0.8 -5 -1.3 -10.1 -5.3 -13 -2.3 -1.6 -4.9 -2.5 -7.7 -2.5l-22.3 0 0 -15.9c0.2 -6.4 -1.7
-11 -5.3 -13.6l0 0zm-36.8 87.8l-17.6 0 0 -48.6 17.6 0 0 48.6zM165.6 113c2 0 3.9 0.6 5.6 1.8
2.9 2.1 4.3 5.8 3.8 9.4l-5.4 34.8 0 0.1c-1 8.1 -7.2 9.8 -12.3 9.8l-42.8 0c-5.3 0 -9.6 -4.3
-9.6 -9.6l0 -38.8 1.7 -0.6c0.6 -0.2 5.1 -1.6 9.8 -4.8 6.6 -4.7 10 -10.9 10 -18l0 -16.1c2.1
-0.4 6.4 -0.8 9.5 1.4 2.6 1.9 3.9 5.5 3.9 10.6l0 18.1c0 1 0.8 1.9 1.9 1.9l123.9 0 0 0zm0 0"
        android:fillColor="#303030" />
</vector>
```

The described data paths are used to create the following image.

Picture 1, Thumbs Up

Shapes can be defined under the Drawable folder like the following shape that is being used to decorate layout components.

Shape example:

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
    <solid android:color="@color/gabrielIncomingMsg"/>
    <stroke
        android:width="0dip"
        android:color="#dddddd" />
    <corners android:radius="10dip" />
    <padding
        android:bottom="0dip"
        android:left="0dip"
        android:right="0dip"
        android:top="0dip" />
</shape>
```

Layout example that uses the shape above as a background:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_marginLeft="5dp"
        android:background="@drawable/msg_incoming"
        android:orientation="horizontal">

        <TextView
            android:id="@+id/txt_msg_gabriel"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:maxWidth="300dp"
            android:padding="12dp"
            android:textSize="18dp"
            android:textColor="@color/gabrielBlack" />

    </LinearLayout>

</RelativeLayout>
```

Above Layout as it is shown on display:

Picture 2, Chat Layout

The light-colored casing (empty at its initial phase) is used to wrap the messages and can expand according to the message's length and height.
The above images show the "incoming messages" layout implementation. The same implementation applies for the outgoing messages with a right orientation and a different background color for the casing.

A new Resource type folder was added to the Resources folder named Menu. This folder was used to define two application menus, a side menu and a bottom one.

A Menu Resource type code example:

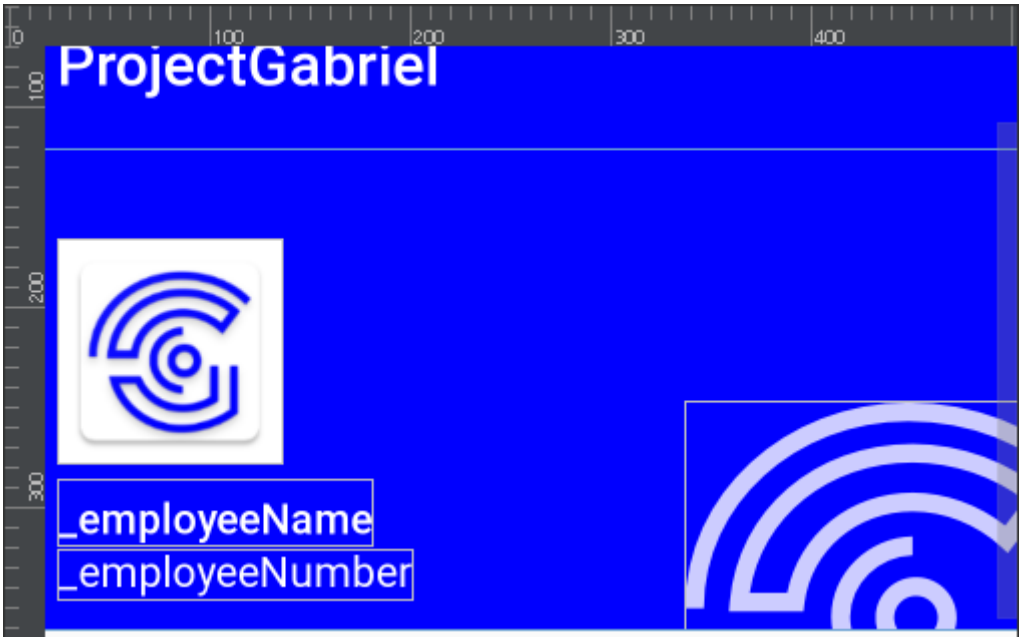
```
<menu>
    <item
        android:id="@+id/nav_changePassword"

        android:icon="@drawable/ic_edit"

        android:title="@string/change_password" />

    <item
        android:id="@+id/nav_endShift"
        android:icon="@drawable/ic_logout"
        android:title="@string/end_shift" />

</menu>
```

The above menu definitions are then called in the according layout files as shown below, first for the side menu layout and then for the additional navigation items.

The side menu's Navigation Layout:

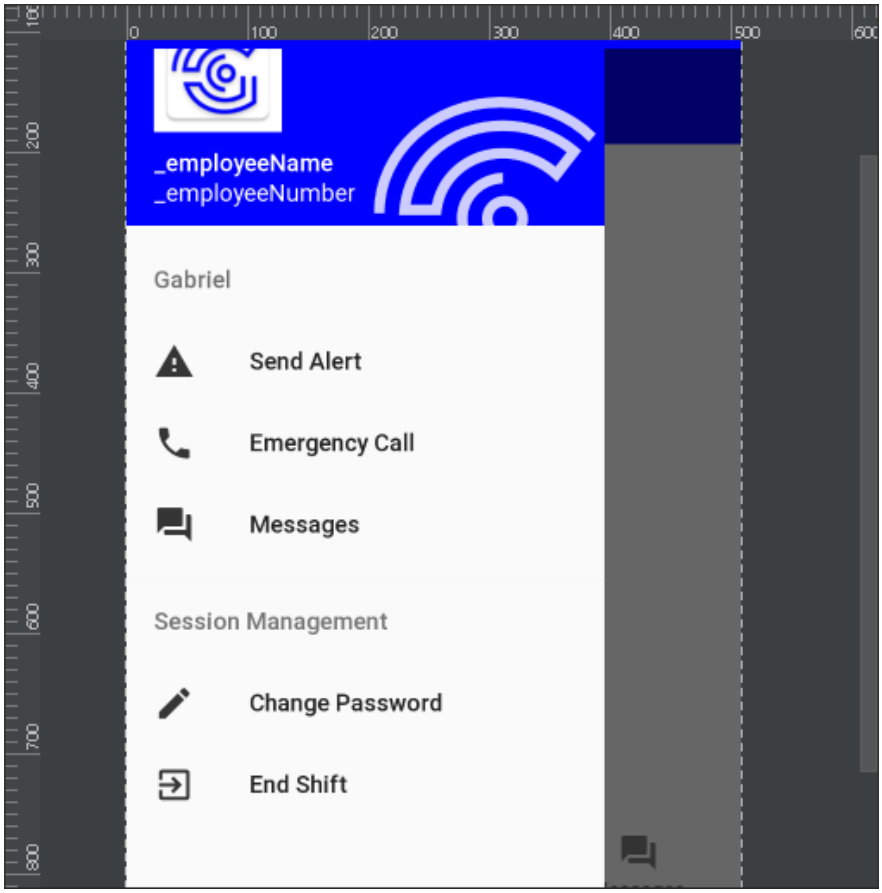
```
<android.support.design.widget.NavigationView
    android:id="@+id/nav_view"
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:layout_gravity="start"
    android:fitsSystemWindows="true"
    app:headerLayout="@layout/navigation_layout"
    app:menu="@menu/activity_main_drawer" />
```

The appearance of the upper part of the side menu was defined separately:



Picture 3, Side menu upper part

The appearance of the side menu:

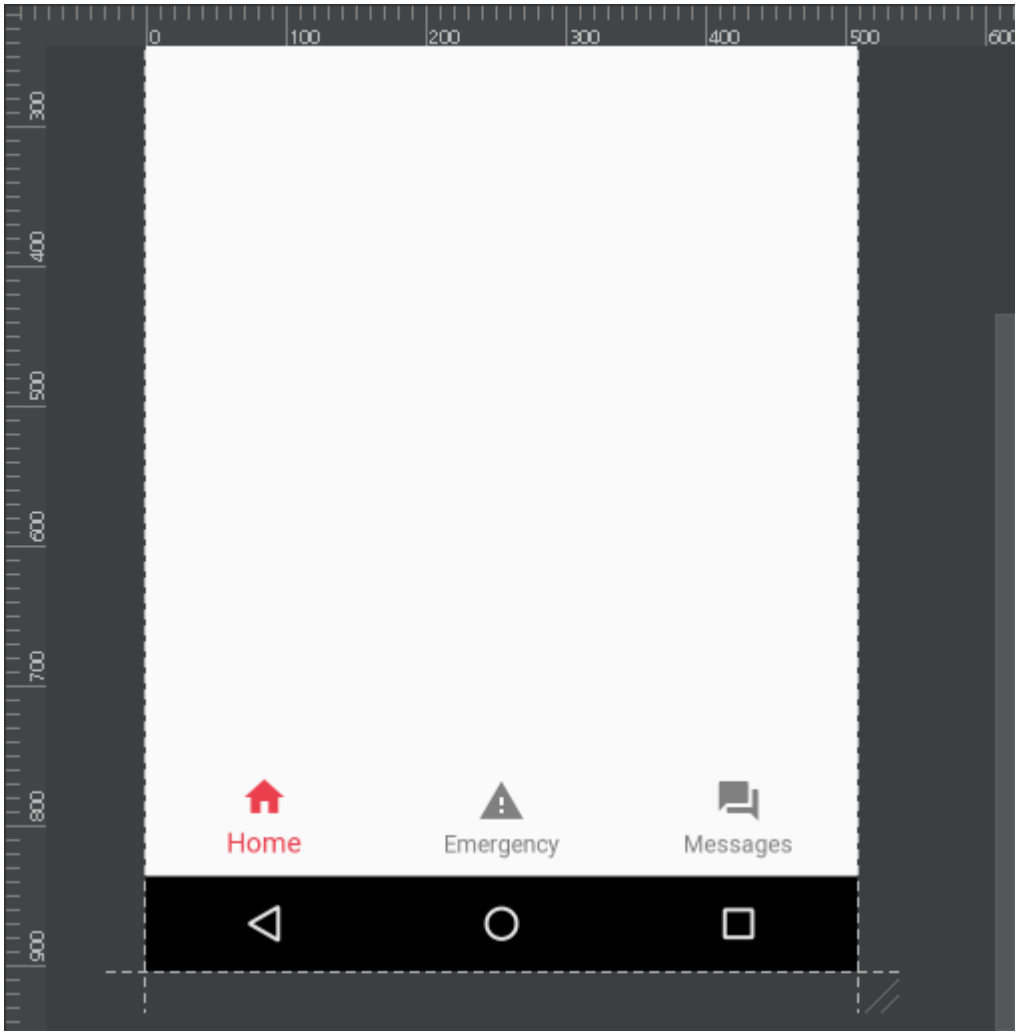


Picture 4, Side menu

Bottom Navigation View for the additional items:

```
<android.support.design.widget.BottomNavigationView
    android:id="@+id/navigation"
    android:layout_width="match_parent"
    android:layout_height="60dp"
    android:layout_gravity="bottom"
    app:itemIconTint="@drawable/selector_navigation"
    app:itemTextColor="@drawable/selector_navigation"
    app:menu="@menu/bottom_nav_items" />
```

The appearance of the bottom menu:



Picture 5, Bottom navigation menu

The checked items from the above menu change appearance according to the selector_navigation file that is shown below:

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:state_checked="true" android:color="@color/gabrielRed" />
    <item android:color="@android:color/tab_indicator_text"  />
</selector>
```

Having built the components that would mostly be used on the application's layouts the implementation of those layouts took place. The application consists of 4 activities and 3 fragments, which means that the creation of 7 layouts was needed. The activities are:

- The activity_login.
  In this layout, there are 2 images with the logo of the application, 2 text fields for the username and the password, a checkbox for the “remember me” feature and a button for the login.

- The activity_main.
  In this layout the layouts of the menus (side and bottom navigation) are contained. On top of that the layout of the fragment that is in use is shown accordingly.
  These fragments are:

  - The fragment_home.
    This is the main fragment of the application and it is used to display the tracking. When the tracking is on a circle animation is shown and when it is off the animation stops. There is also a button to start the route. When the route starts the next checkpoint and a delay button are visible. On the top left corner, the last checkpoint is shown regardless of the route state.

  - The fragment_emergency.
    This fragment is used to display 6 emergency buttons each for a predefined emergency situation and an extra button for a direct call to the security supervisor.

  - The fragment_message.
    This fragment is used to display the messages sent and received. It has a list view to display the messages, a type box where the user can type his message and send it. This type box is a separate layout that contains an edit text field and an image view used as a button. Finally, there are 2 buttons for quick response messages (thumbs up and thumbs down).

- The activity_password.
  In this layout there are 3 text fields, one for the old password another for the new password and the last one for the confirmation of the new password. There is also a button for the password change and an image with the application logo.

- The activity_usage_agreements.
  In this layout there is an image with the application logo and name, a text view for the display of the usage agreements, a button for the acceptance and one for the decline of the usage agreements.

4.2.2.2 Rest API

In the Rest API folder is the implementation of the Rest API services for the application. It was split into 3 classes each class for a specific purpose.

The first class, the AuthorizationRequestInterceptor is there to provide information about the user. It holds the bearer Token, that is a Json WebToken provided from the backend when the user logs in successfully. The token is the identifier of the user for then backend when calls on it are made. The Class implements the interceptor interface from okhhtp3 and must override its method intercept. This method has a return value type of "response". In that method, the token is checked and if there isn't an empty token (that means that the user has successfully logged in) then it is added to the header of the request that will make the call.

The snippet of the code is shown below.

```
@Override
public Response intercept(Chain chain) throws IOException {
    if(BearerToken!=null){
        Request request = chain.request().newBuilder()
                .addHeader("Authorization","Bearer "+BearerToken).build();
        return  chain.proceed(request);
    }

    return chain.proceed(chain.request());
}
```

The second class, is the RestController. In that class, the information needed for the calls is set. Firstly, the BASE_URL is set to the url that the calls will be made. After that, in the class constructor, a gson object is setup to use it as a custom converter on the calls. It contains the date format, as well as a deserializer that is another custom-made class to use it on the gson object. Then a custom setup call for call instances will be made. All these setups will be used by the Retrofit builder that is implemented at that point. Finally, we instantiate the gabrielRestApiService to the restAdapter that is made so that it can create the calls.

```
Gson gson  = new GsonBuilder()
        .setDateFormat("yyyy-MM-dd'T'HH:mm:ss.SSSSSS")
        .registerTypeAdapter(JsonMessageWrapper.MessageType.class,new
JsonMessageWrapperMessageTypeDeserializer())
        .create();
OkHttpClient.Builder httpClientBuilder = new OkHttpClient.Builder();
httpClientBuilder.addInterceptor(new AuthorizationRequestInterceptor());
httpClientBuilder.connectTimeout(30, TimeUnit.SECONDS);
OkHttpClient client = httpClientBuilder.build();
Retrofit restAdapter = new Retrofit.Builder()
        .baseUrl(BASE_URL)
        .addConverterFactory(GsonConverterFactory.create(gson))
        .addConverterFactory(ScalarsConverterFactory.create())
        .client(client)
        .build();
```

In the end of the class a GET function is implemented so the retrieval of the REST function will be easier to be made.

The third class is the interface that holds the REST services. In that interface named GabrielRestApiService all the services/calls that are used on the application, are held. First, we set the headers for the calls and after that all the calls. A code snippet will be given that will show the log in / log out calls and the headers.

```
@Headers({
        "Content-Type: application/json",
        "Accept: application/json"
})
@POST("Auth/PostLogin")
public Call<String> login(@Body LoginDto login);

@POST("Auth/PostLogout")
public Call<Void> logout();
```

To give the headers, the @Headers is used so it can identify that these are headers.

For the log in and out the @POST is used followed by (“example”), is the “path” of the desired service. On the next line the functions are implemented, on login there is a return value of String assigned inside the tag <> and void for logout. The login function has a @Body annotation also, which means that information is needed to be passed to the backend so it can respond back with a proper answer. This information is given by a class that the backend handles.

4.2.2.3 WebSockets

The WebSocketController contains the functionality of the WebSockets to be used in the application. In that class, the WebSockets can be started and stopped. It also contains the instantiation of them, as well as the function to send data through WebSockets. In the instantiation method if an instance already exists it gets returned else a new instance is created. In start method, a new WebSocketImpl object is created.

In the WebSocketImpl class is the implementation of the WebSockets in the application. Firstly the Uri, that the WebSockets will connect, is given. Then a string tag to map the possible exceptions during the development of the application is provided. The default constructor of the class is defined. There is also a "connect" method which creates a WebSocket with the given Uri, adds the necessary listeners and finally connects. A "reconnect" method was created to allow the reconnection to the server in case of a disconnection. A "get connection" method was created to return the existing connection. A "disconnect" method was created to allow the disconnection of the WebSocket from the server. An inner class named "SocketListener" was created which extends the WebSocketAdapter class and overrides the onConnected, onTextMessage, onError, onDisconnected, onUnexpectedError, onPongFrame and onPingFrame methods. In these methods the "behavior" of a WebSocket is formed for each situation represented by the methods.

In the WebSocketMessages folder the chain of responsibility pattern is used so the code would be more efficient, easily reusable and expandable. As the name suggests, the chain of responsibility pattern creates a chain of receiver objects for a request. This pattern decouples sender and receiver of a request based on type of request. This pattern comes under behavioral patterns.

In this pattern, normally each receiver contains reference to another receiver. If one object cannot handle the request then it passes the same to the next receiver and so on.

A Chain of Responsibility Pattern says that just **"avoid coupling the sender of a request to its receiver by giving multiple objects a chance to handle the request".**

In other words, we can say that normally each receiver contains reference of another receiver. If one object cannot handle the request then it passes the same to the next receiver and so on.

Advantages

- It reduces the coupling.
- It adds flexibility while assigning the responsibilities to objects.

- It allows a set of classes to act as one; events produced in one class can be sent to other handler classes with the help of composition.

Usability

- When more than one object can handle a request and the handler is unknown.
- When the group of objects that can handle the request must be specified in dynamic way.

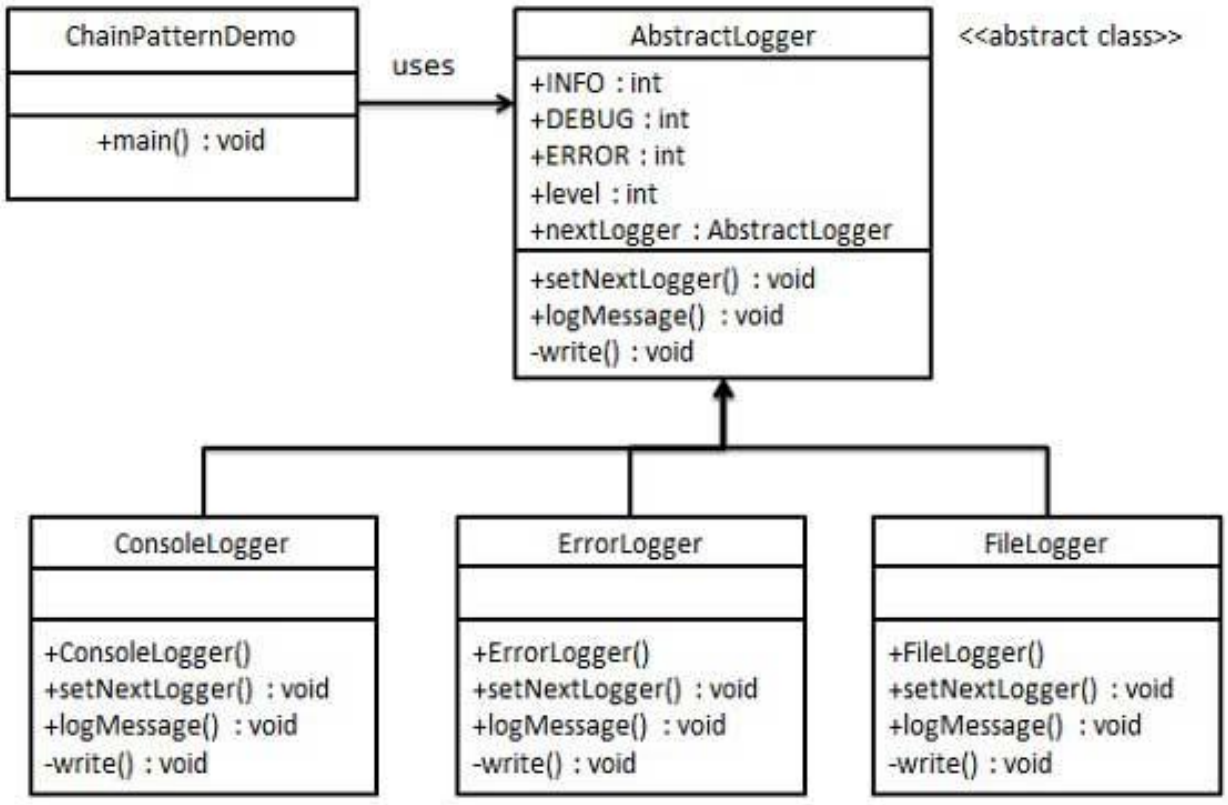
Rules of Thumb

- Chain of Responsibility, Command, Mediator, and Observer, address how you can decouple senders and receivers, but with different trade-offs. Chain of Responsibility passes a sender request along a chain of potential receivers.
- Chain of Responsibility can use Command to represent requests as objects.
- Chain of Responsibility is often applied in conjunction with Composite. There, a component's parent can act as its successor.

As the UML below shows the basic form of the pattern there are these steps that must be followed for its implementation:

1. The base class maintains a "next" pointer.
2. Each derived class implements its contribution for handling the request.
3. If the request needs to be "passed on", then the derived class "calls back" to the base class, which delegates to the "next" pointer.
4. The client (or some third party) creates and links the chain (which may include a link from the last node to the root node).
5. The client "launches and leaves" each request with the root of the chain.
6. Recursive delegation produces the illusion of magic.

UML example of Chain of Responsibility design pattern



Picture 6, Chain of Responsibility UML

An abstract class is implemented named JsonMessageDeserializer, after that, various implementations of that class are set, one for every message on the application through WebSockets. To hold the data of the messages an interface called IMessage helps once every different message has different data in it. So, for every message an implementation to hold its data was created, all these classes were implementing the IMessage interface. Finally, in the JsonMessageWrapper class the chain of responsibility pattern is finished. In the getMessageDeserializer method all the deserializers are instantiated and after that, using the setNextDeserializer method from JsonMessageDeserializer abstract class, the deserializers are chained.

In the JsonMessageWrapper class an enum value named MessageType was set too. This value holds all the different message types so they can be indicated on setType and identified to get deserialized properly. Different seters and geters were implemented in this class so all the data that is needed can be known.

The InitCommsMessage class contains information about the user that sends a message. This class is used to allow the backend to decide correctly where the message should be delivered. For example if an application user sends a message, then the backend has to decide to which site the message should go.

4.2.2.4 Activities

1. Login Activity
   In the Login Activity the user has to type his credentials to log in. Using the shared preferences provided by the Google API the "Remember me" functionality was implemented allowing the user to save the credentials for the next log in as an xml file in the application's data file. In this activity, the most recent license agreement acceptance is also checked. If the user did accept the application continues as normal else he is redirected the LicenseAgreement Activity. After the log in and before the application steps to the next activity a toast will be shown reminding that the application will track the user's position.

2. LicenseAgreement Activity
   These days the terms of use and license agreements are mandatory for every official application. The license agreements contain rules that the user has to abide by and certain personal information permissions that it will require from the user. The activity LicenseAgreement was made to be shown right after the installation of the application, or if there is an update on them, so that the user can decide whether to use the application by accepting or not.
   If the user accepts the terms of use his choice is stored in the database with a REST call to the backend and if he declines the application does not continue.

3. Password Activity
   In this activity the user is prompted to type his old password, a new password and a confirmation for the new password. Then, if the new password matches the confirmation, a REST call is made to send the old password and the new one to the backend. Then the backend replies accordingly if the replacement was successful or not. If it was successful a message is shown and then the user is redirected to the main activity.

4. Main Activity
   This activity encapsulates the layouts of the fragments that are being used and takes care of the navigation between them.

4.2.2.5 Fragments

1. Emergency Fragment

   For each emergency situation a different message is being sent through WebSockets as the example shown below:

```
    SOSFireMessage message = new SOSFireMessage();
    JsonMessageWrapper wrapper = new JsonMessageWrapper();
    Gson jsonConverter = new Gson();
    String jsonMessage = jsonConverter.toJson(message);
    wrapper.setType(JsonMessageWrapper.MessageType.SOS_FIRE_NOTIFICATION);
    wrapper.setDeliverToUsers(null);
    wrapper.setJsonPayload(jsonMessage);
    wrapper.setBearerToken_Username(AuthorizationRequestInterceptor.BearerToken);
    wrapper.setFrom("Client");
    WebSocketController.Instance().sendJsonData(jsonConverter.toJson(wrapper));
```

First the type of the situation is identified depending on the button the user pressed and the message gets converted to a json string. Then the Json wrapper sets the data that is required to send the message and then that message is sent through the WebSockets to the site that the supervisor is using. In the Emergency fragment, there is also a direct call button that uses the code below:

```
buttonEmergencyCall.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        try {
            String uri = "tel:0123456789";
            Intent emergencyCall = new Intent(Intent.ACTION_CALL, Uri.parse(uri));
            startActivity(emergencyCall);
        }catch(Exception e){
            Toast.makeText(getActivity().getBaseContext(), R.string.call_failure,
                    Toast.LENGTH_LONG).show();
            e.printStackTrace();
        }
    }
});
```

When the button is pressed the ACTION_CALL intent is launched. The ACTION_CALL intent is provided by Google API and allows the dial of an explicit phone number that can be predefined.

2. Home Fragment

   This is the main fragment of the application and it is used to display the tracking. When the tracking feature is turned on an activation message is sent through the WebSockets to the

supervisor's site, a rest call is made to the backend and a function is called to activate the beacon listener. Also, the Bluetooth turns on after prompting the user for permission to allow its use. If the tracking is turned off the Bluetooth turns off, a deactivation message is sent through WebSockets to the site, a rest call is made to the backend, and a function is called to deactivate the beacon listener. Then a runnable is created to update the route's checkpoint names that are being displayed on the interface.

Home fragment also contains the "start route" button. If the button is used a request is sent to the backend which then replies with a route. Then the application checks the response and if it is null, which means that there is no route, an appropriate message is shown and the button gets disabled. If the response is not null the contained data are assigned to a route type object (REST API DTO).

3. Messages Fragment

   This is the fragment where the chat feature was implemented. It uses WebSockets to send text messages and predefined quick responses. Two inner classes are also contained, the LoadMessages class and the AddMessages class. The first one loads the previously sent messages and by filtering them it displays only the desired ones. The latter adds the messages to a chat Adapter. The chat Adapter was necessarily implemented to allow the display of the messages on the ListView of the fragment. A chat message type object is created which contains the content of the message, the date and the sender. The Adapter checks the sender and the content and "chooses" the correct layout to display the message. For example, if a user sends a message the outgoing layout is used which aligns the message to the right with a grey background. Also, if a quick response message is sent, an image is displayed rather than the content of the message.

4.2.2.6 Beacons

To implement the classes needed for the use of beacons the Design Pattern Observer – Observable was used. Observer pattern is used when there is one-to-many relationship between objects such as if one object is modified, its dependent objects are to be notified automatically. Observer pattern falls under behavioral pattern category. Java has implemented the observer pattern components to help the programmers by shorting the code.

Advantages

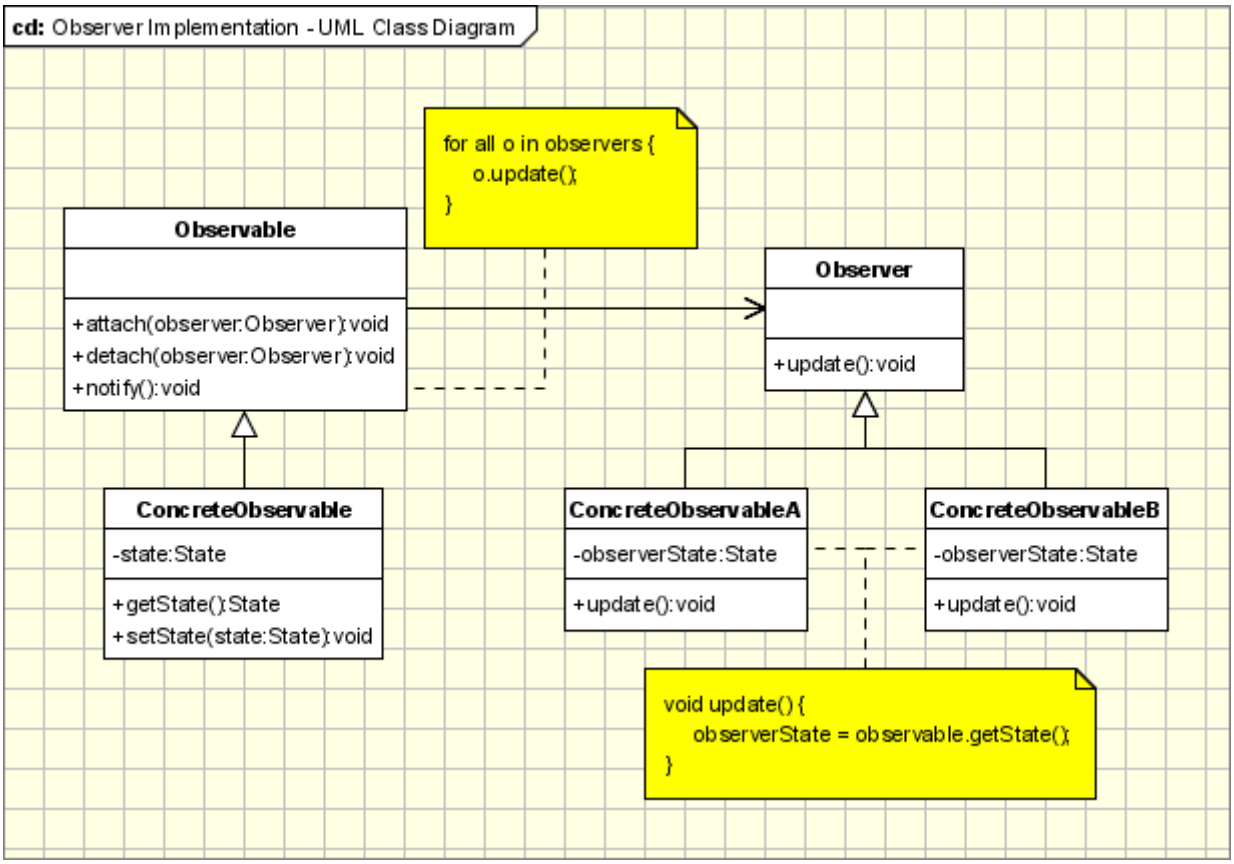
- Supports the principle to strive for loosely coupled designs between objects that interact.
- Allows the programmer to send data to many other objects in a very efficient manner.
- No modification is needed to be done to the subject to add new observers.
- The programmer can add and remove observers at any time.

Usability

- When the change of a state in one object must be reflected in another object without keeping the objects tight coupled.
- When the framework we are writing needs to be enhanced in future with new observers with minimal changes.

Usage examples

- Model View Controller Pattern. The Observer pattern is used in the model view controller (MVC) architectural pattern. In MVC this pattern is used to decouple the model from the view. View represents the Observer and the model is the Observable object.
- Event Management. This is one of the domains where the Observer pattern is extensively used. Swing and .Net are extensively using the Observer pattern for implementing the events mechanism.



Picture 7, Observer pattern UML

In the ‘Beacons’ folder the beacon listener was implemented which consists of two classes, the BeaconDetector class and the BeaconController class. The first one extends the Observable class and implements the BeaconManager.RangingListener from which the onBeaconDiscovered method is overridden. This method checks if a beacon is detected nearby and if there is one it notifies the Observers of this class.

```
@Override
public void onBeaconsDiscovered(Region beaconRegion, List<Beacon> list) {
    if(list.size()>0) {
        Beacon bm = list.get(0);
        this.setChanged();
        this.notifyObservers(bm);
    }
```

The latter extends the Observable class and implements the Observer interface. In the class constructor, the values that are shown below are initialized. These values are necessary for the use of the Estimote API.

```
public BeaconController(BeaconManager manager){
    this.beaconManager = manager;
    this.detector = new BeaconDetector();
    this.detector.addObserver(this);
    this.beaconManager.setRangingListener(this.detector);
    this.region = new Region("ranged region", null, null, null);
    Assert.assertTrue(oneInstance);
    oneInstance = false;
}
```

Then methods to get and set the instance of this class and methods called in the home fragment, to activate or deactivate the beacon listener, were implemented.

4.2.2.7 AppStatusApplication class

This class extends the Application class of the Android API which maintains the global application state and implements the Application.ActivityLifecycleCallbacks, ComponentCallbacks2, Observer and SensorEventListener interfaces. The Application class offers the ability to update the application based on the changes in that class. That helps the developers immensely because the updates of the application’s tasks are applied independently of the application’s current state or current activity.
The ActivityLifecycleCallbacks and the ComponentCallbacks2 interfaces give access to the activity

callback methods. This way the current state of the application is determined at any time. More precisely these callbacks were used to know when the application is in use or not, when it is sent to the background or to the foreground and when it gets destroyed.

In the OnCreate method several initializations take place. One of these is the initialization of the adapter that allows the connection with the device's Bluetooth. Another is that of the internet state and of the broadcaster that notifies when the internet state changes. Then an Intent filter is created to distinguish under which conditions the internet state changes and a receiver is registered. A sensor manager is created to allow the access to accelerometer and register a listener to it. Another broadcast receiver is registered to check if the device's screen is off. Activity lifecycle callbacks are then registered and finally a beacon controller is initialized.

As this class controls the whole application functions are implemented to discriminate the currently used activity. These functions are crucial for the alert notifications which require the activity that is under use at that moment. Also for the safety of the user the inactivity periods are tracked with the use of a timer. If an inactivity period is detected an alert box pops up using the flashlight and sound notifications to check up on the user and prompt him to assure his safety with the push of a button. If a button is not pressed in a given time window an alert will be shown in the supervisor's site. If the "Not Ok" button is pressed the user get redirected to the emergency fragment to choose an emergency situation. If the "Ok" button is pressed the inactivity timer is reset and the alert dialog closes.

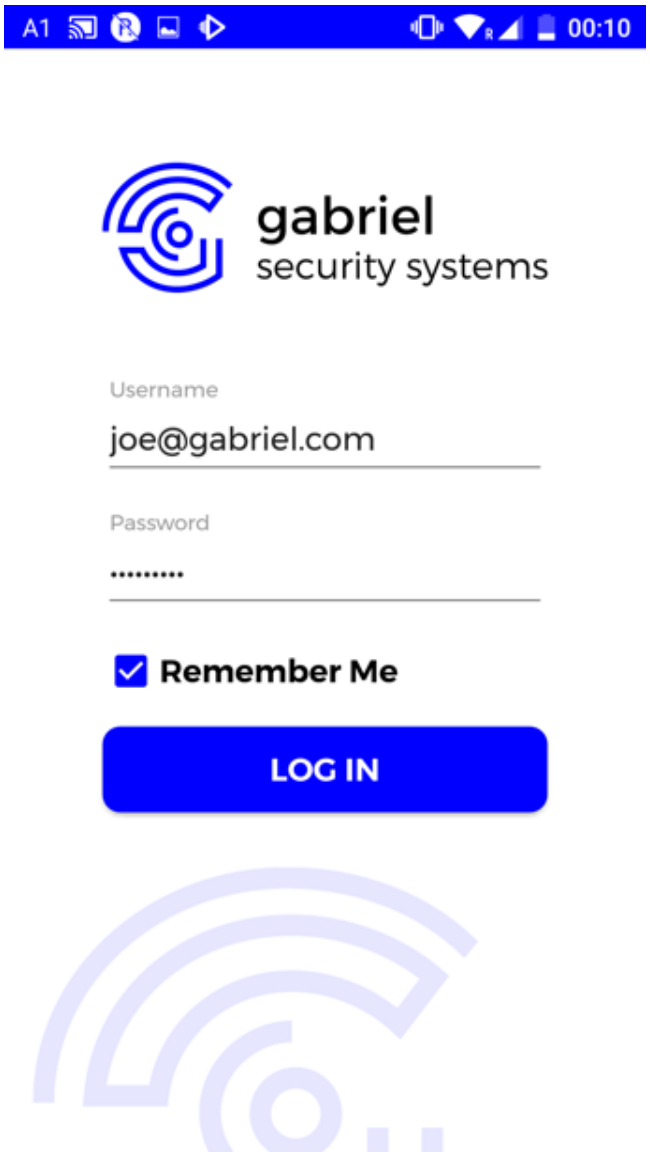
In the overridden method onSensorChanged, the values of the accelerometer get filtered to lower its sensitivity and trigger only on major movements because the accelerometer detects movement even when the device is still.

The final functionality of the Application class is to update the backend, the site and the UI of this application with the detected beacons if and when it is needed. If a new beacon is detected while the tracking is on information about its identity and battery level are sent to the website through WebSockets regardless of the current route. If there is an active route with active remaining checkpoints and the nearest beacon is the same with the beacon that represents the first checkpoint of the route, a REST call is made to clarify that the mentioned checkpoint was cleared. Then if that checkpoint is also the last, the route ends and appropriate actions take place else the next checkpoint is pushed on the list of beacons to check. The passed checkpoint as well as the next checkpoint, if there is one, are shown through a notification if the user is not in the home fragment of the application.

4.3 Manual

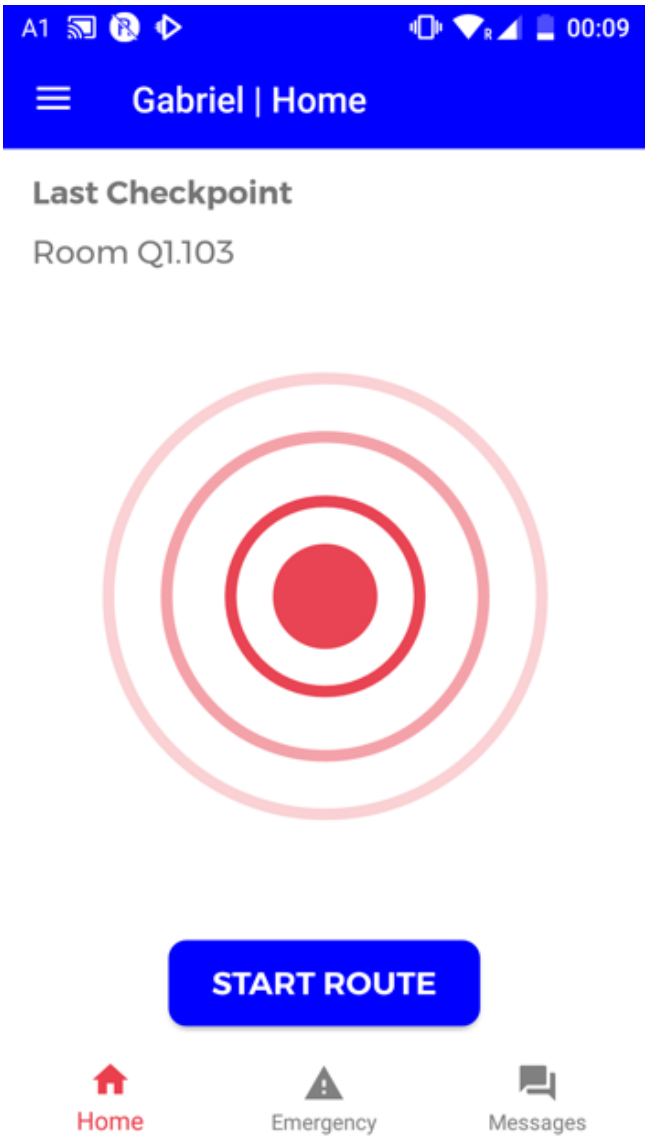
Below is a presentation of Gabriel's core features.

The first contact of the user with the mobile application is through the Login screen. As shown below the user is prompted to type the credentials provided by the company. For ease of use a "Remember Me" checkbox was added.



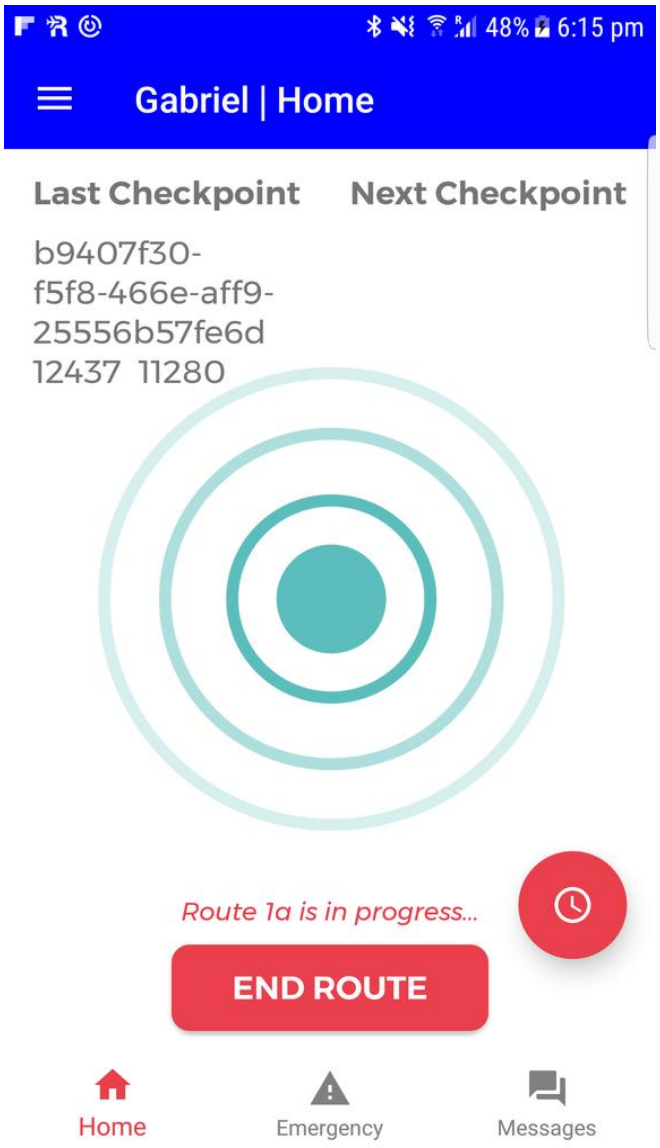
Picture 8, Manual, Login screen

After a successful Login the user gets redirected to the Tracking page which is the main Activity of the mobile application. Initially the tracking feature is disabled. To activate it the user can click on the animated circle. In the top left corner there is a button for the side menu of the application and at the bottom there is the application navigation menu. Informations regarding the checkpoints are displayed above the animated circle.



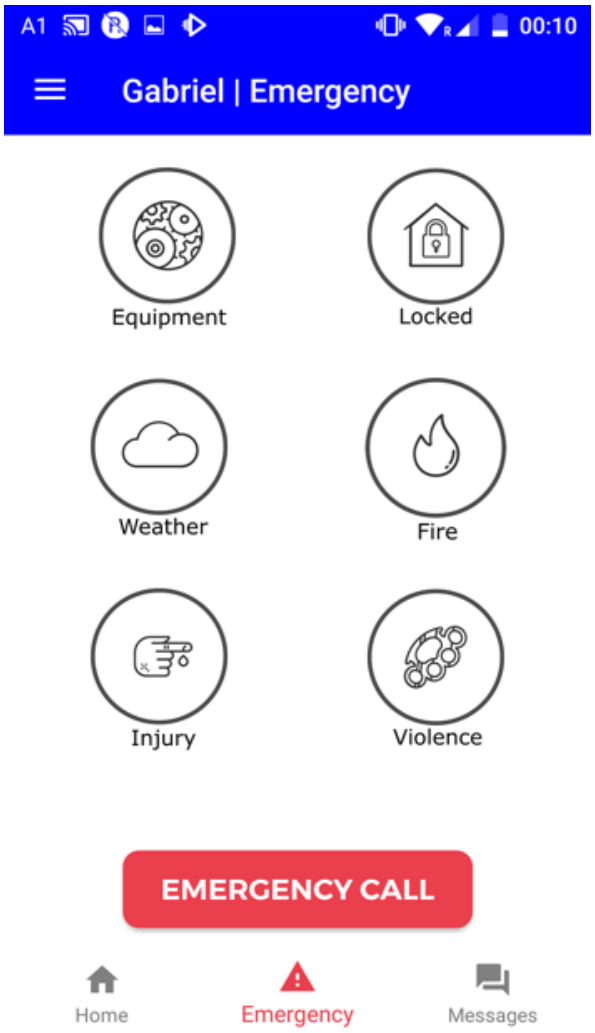
Picture 9, Manual, Tracking Screen

This image shows the tracking feature and the routing are set to active. An additional field appears which displays the name of the next checkpoint if there is one. Also a delay button was added so the user can inform the supervisor about time delays. To remind the user that the tracking is active the colors of the circle and the button were changed and an animation was set to the circle.



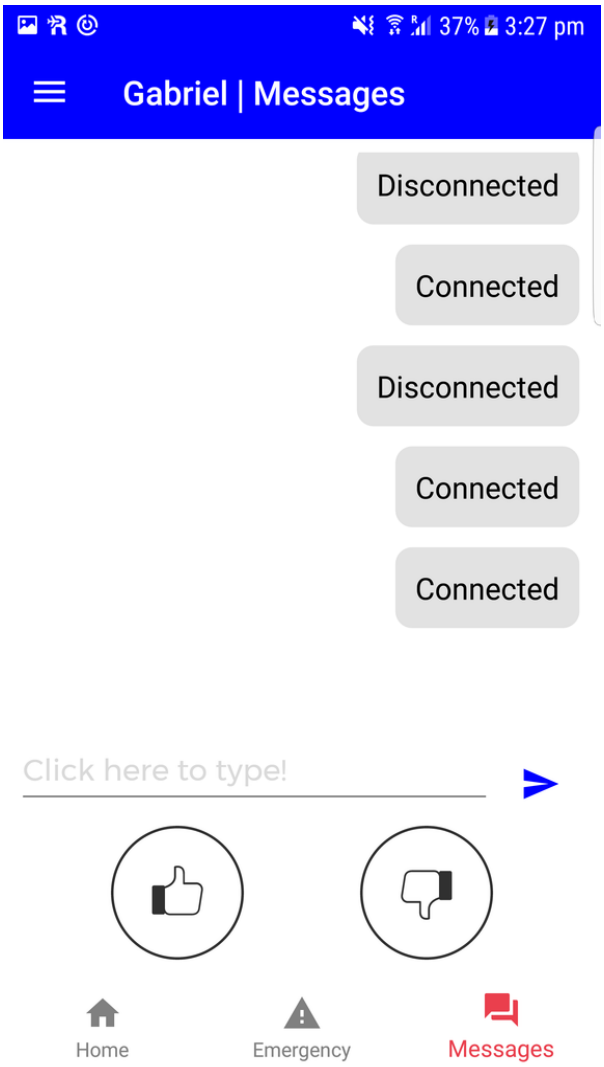
Picture 10, Manual, Activated tracking screen

This is the second page of the Android application. In this page there are emergency buttons, each for a different emergency situation. By pressing any of these buttons a message will be sent to the supervisor website indicating the emergency situation. There is also an Emergency Call button which allows the user to make a direct call to a predefined number.



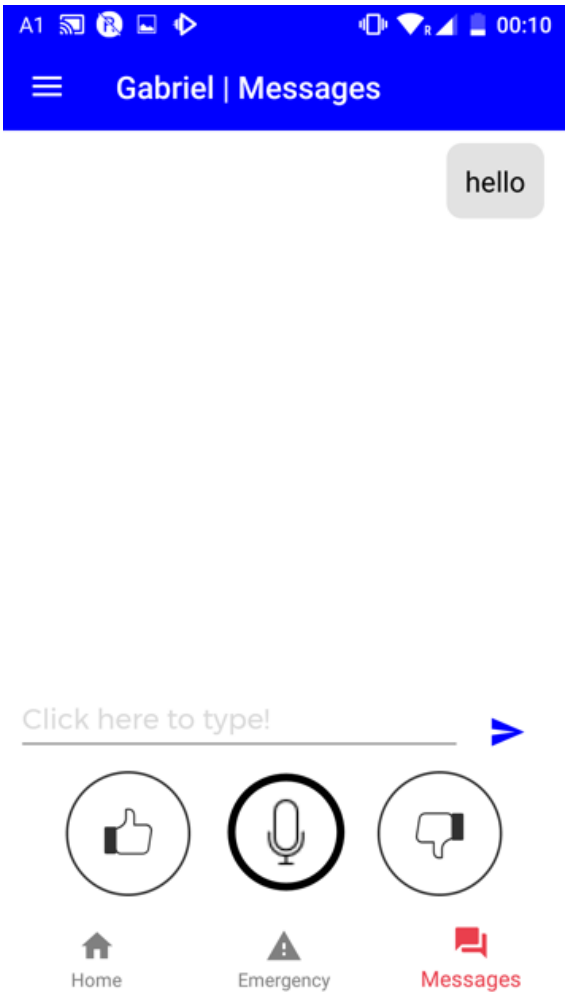
Picture 11, Manual, Emergency situation screen

This is the chat section of the Android Application. The user is provided with a text box to type the messages. Also quick response buttons were added for additional ease of use.



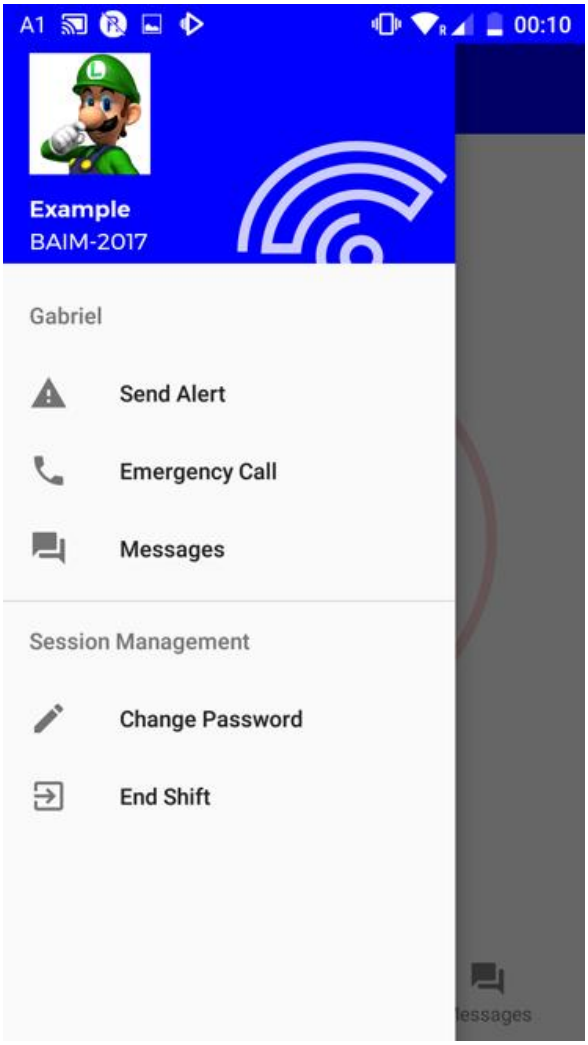
Picture 12, Manual, Chat screen

This is a version of the chat messages which was not released. An audio message is available with the push of the microphone button. The message has duration of up to thirty seconds and is sent to the website.



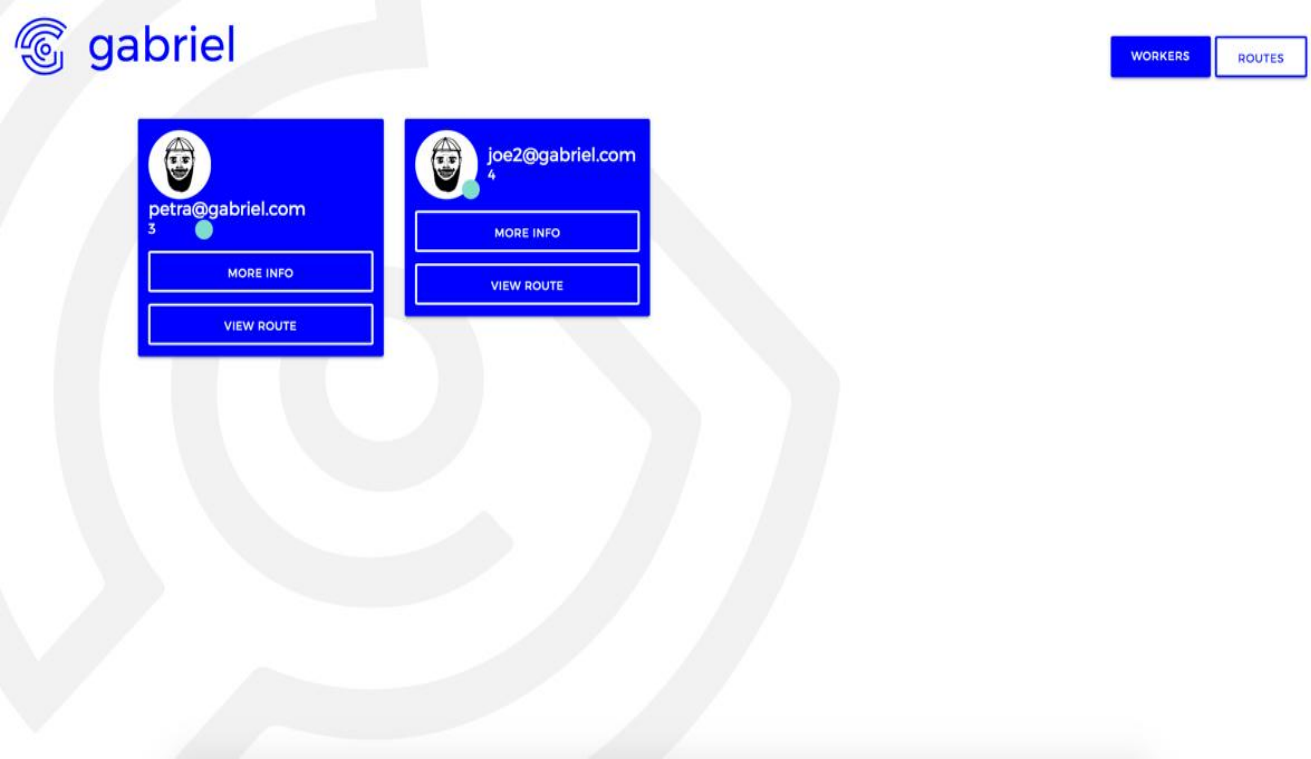
Picture 13, Manual, Chat with Audio screen

This is the side menu of the application. It allows the redirection of the user to the other pages of the application as well as the management of his password and the log out option. Also information about the user such as image and name are displayed on the top section of the menu.



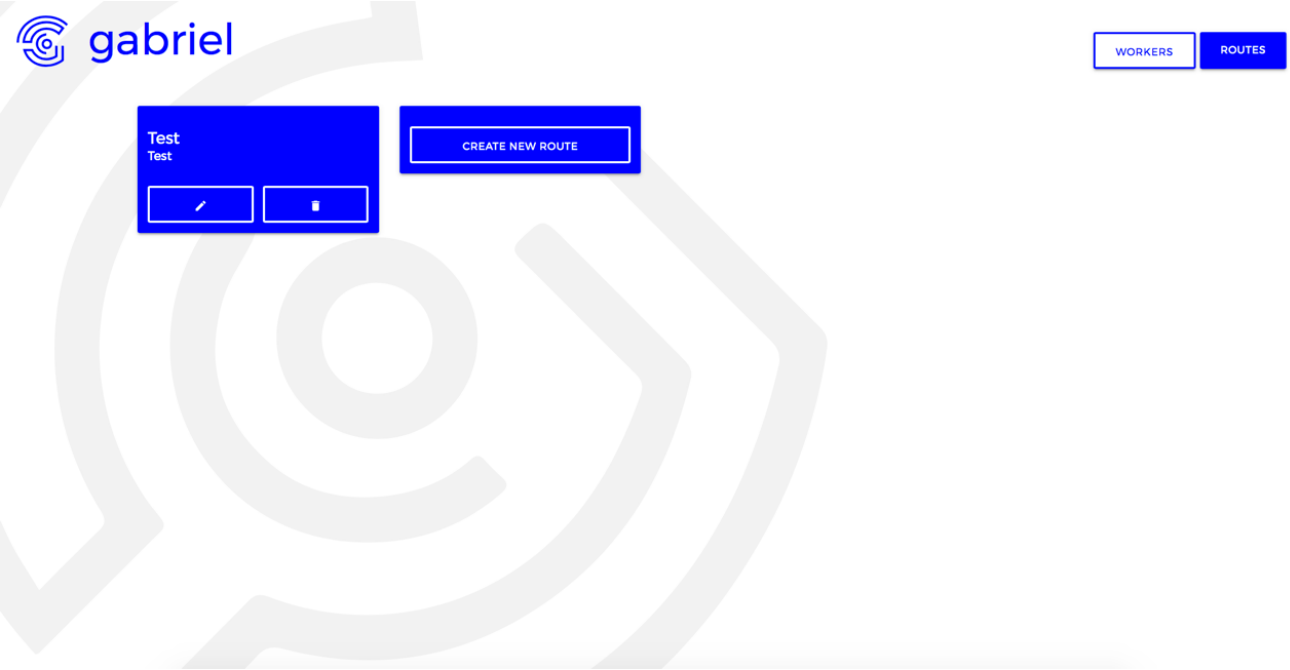
Picture 14, Manual, Side menu

This is a page of the supervisor's website. In this page the employees that are logged in are displayed. The supervisor can view details and what routes are assigned on each employee. There is an option to change to the route management page and also with the selection of an employee the supervisor can open a chat box with that employee.



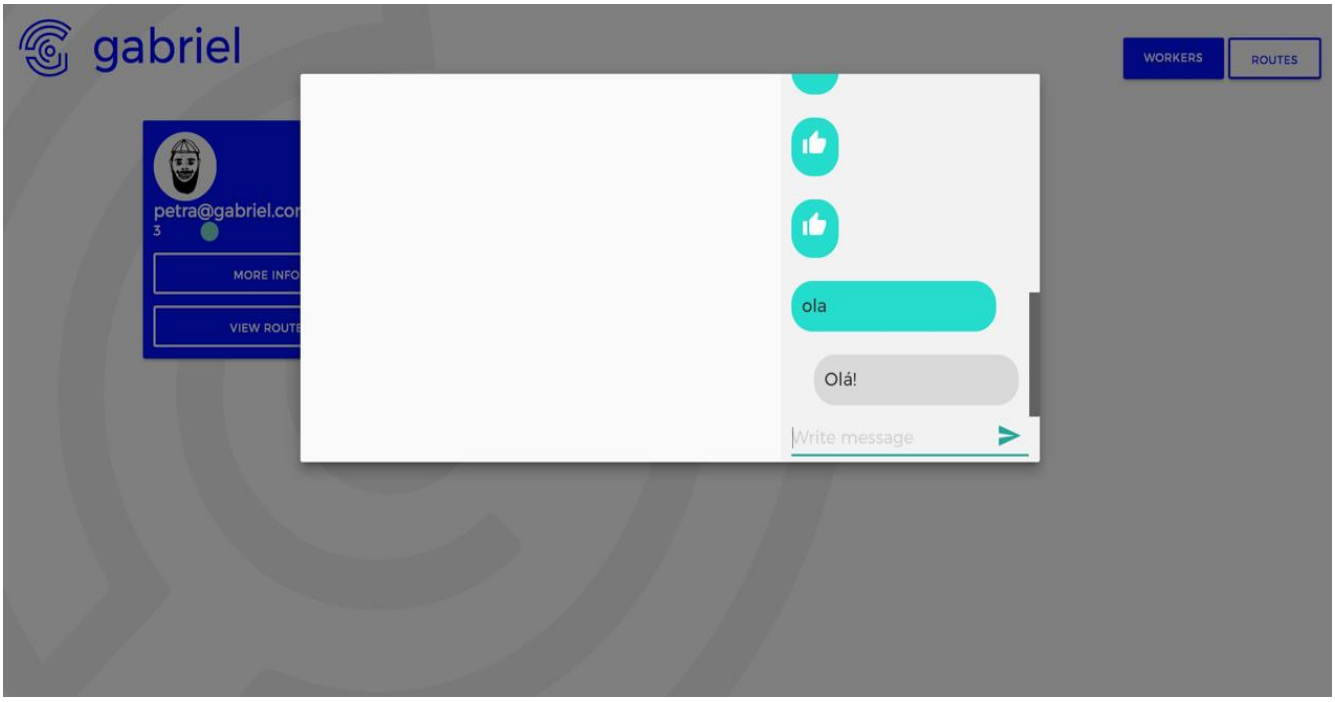
Picture 15, Manual, Supervisor website, Worker Tab

This is the route management page of the supervisor's website. A new route can be created with the push of a button and then assigned to employee/s. The supervisor can also edit and delete the already existing routes.



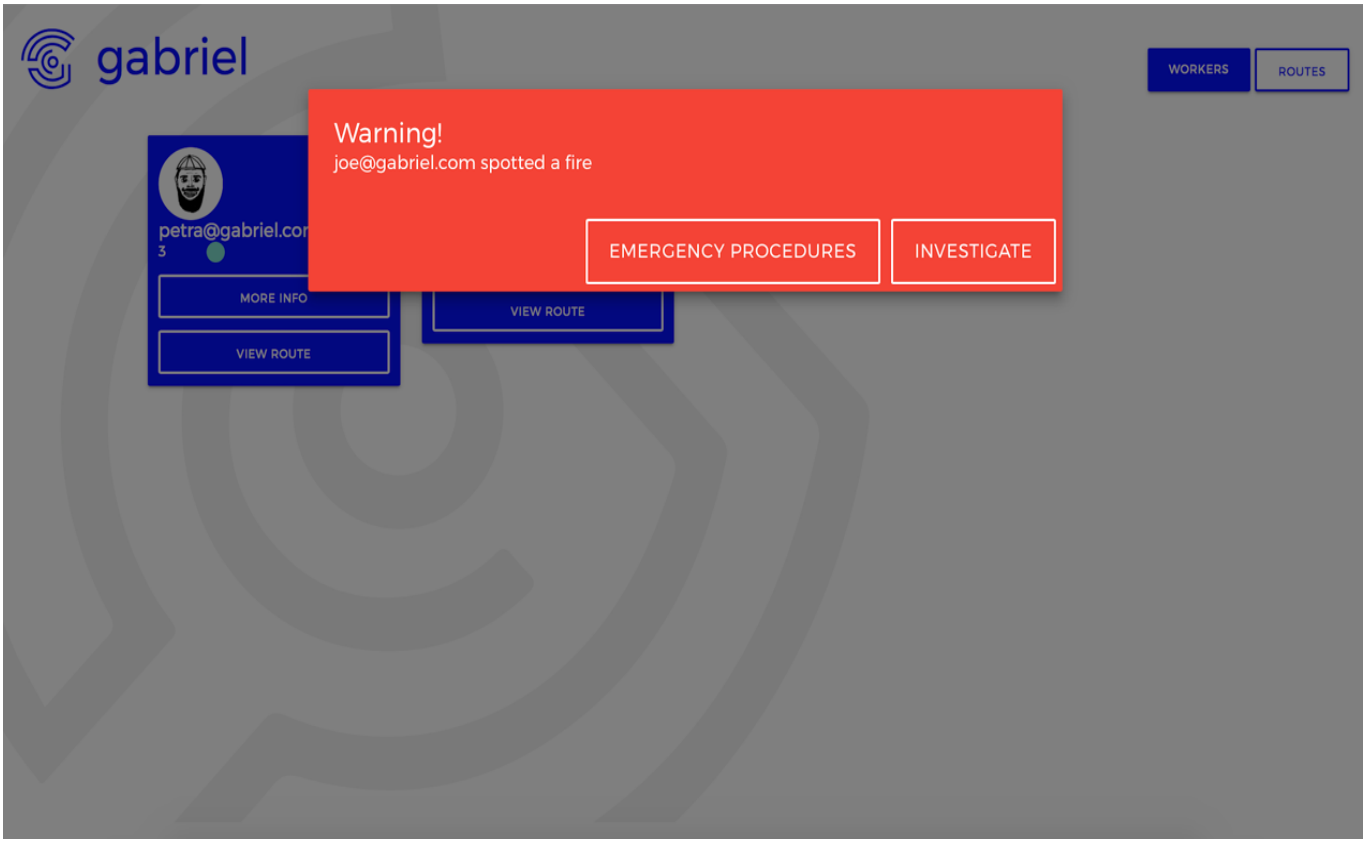
Picture 16, Manual, Supervisor website, Route Tab

This is the chat that appears when the supervisor selects an employee.



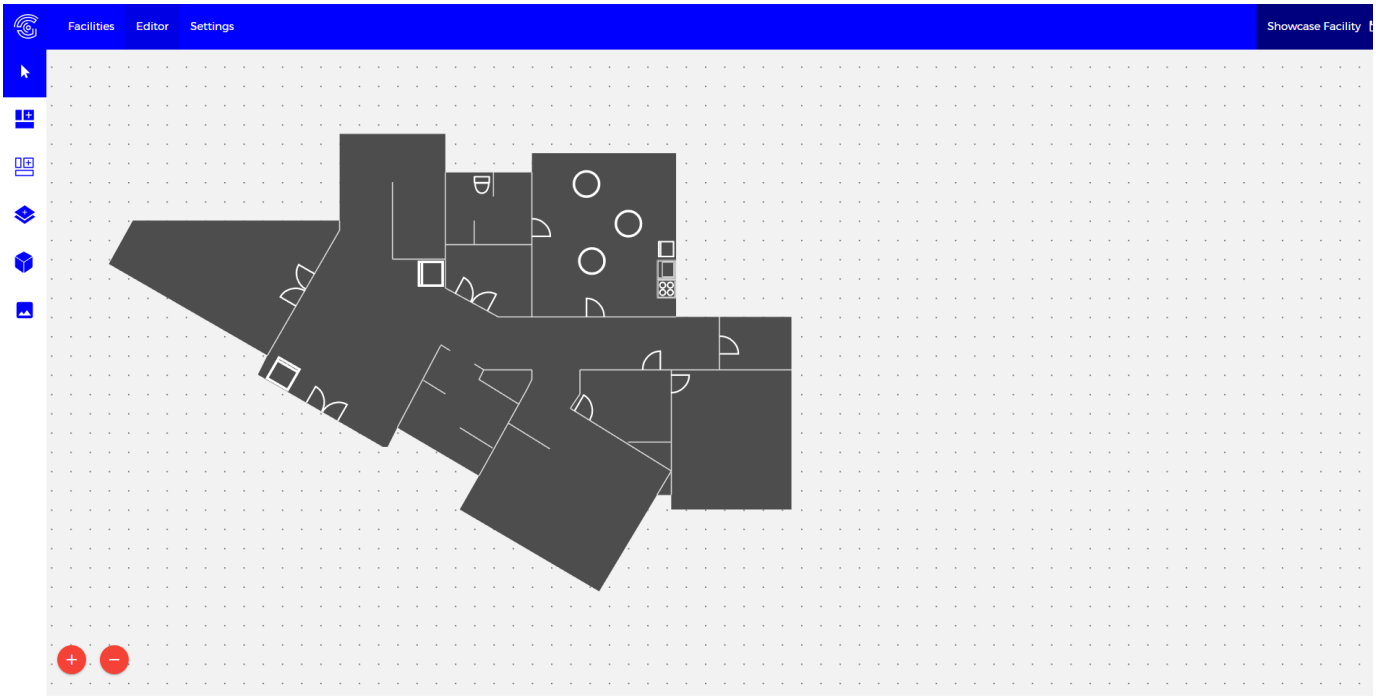
Picture 17, Manual, Supervisor website, chat with an employee

This is the emergency alert box that the supervisor gets when an employee triggers an emergency button. This will also pop up if the inactivity alert is triggered. The supervisor is prompted to choose an action, either proceed to emergency procedures or proceed to investigate the situation. Through the emergency procedures the supervisor can contact the appropriate security services. Through the investigate button the supervisor can view the emergency situation that was triggered.



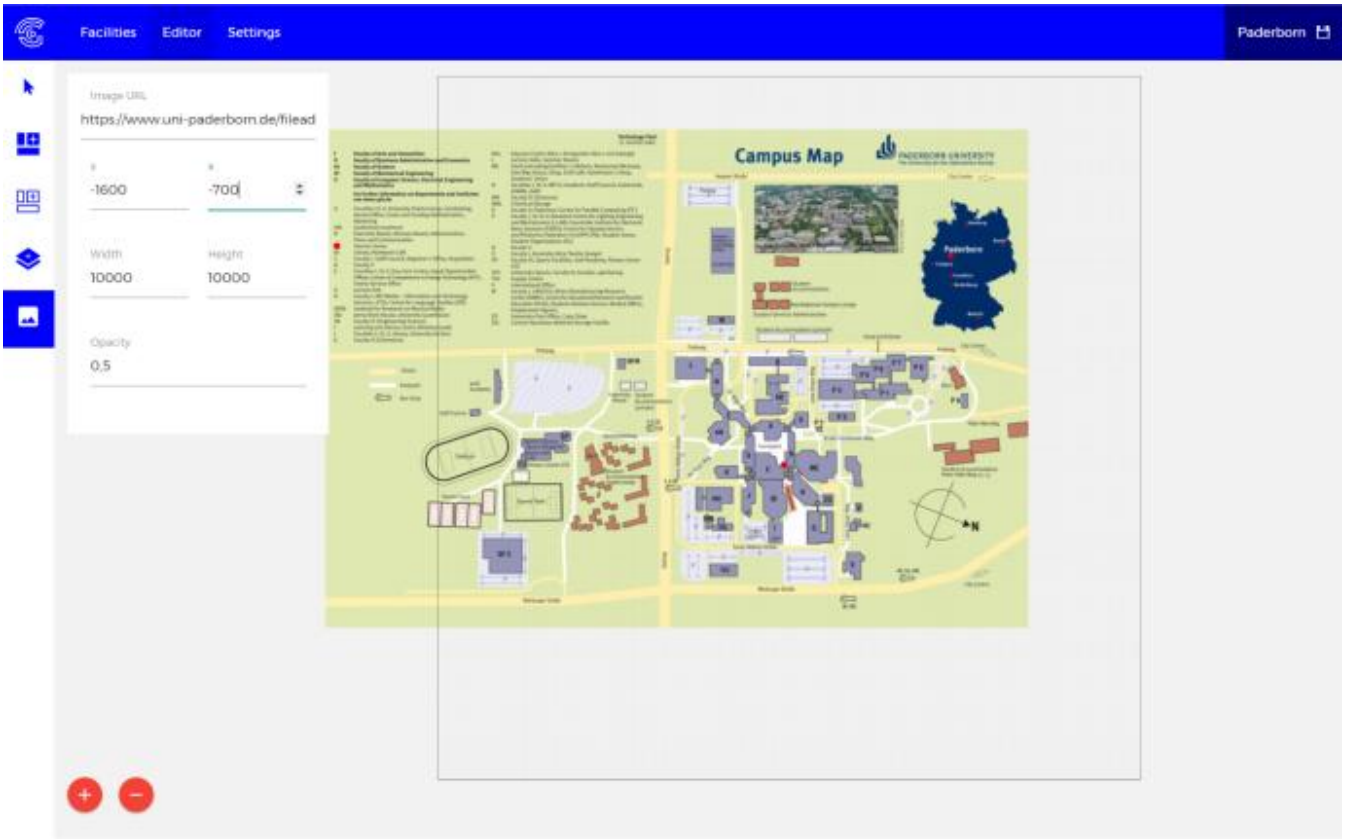
Picture 18, Manual, Supervisor website, emergency

This is the map editor website. In this website a user can create a map or modify an existing one. For easier use the user is provided with tools that help on the creation or modification of a map. Some of these tools are: Zoom in/out, set size, floor selection if the map has more than one floor, door insertion and a wall drawer.



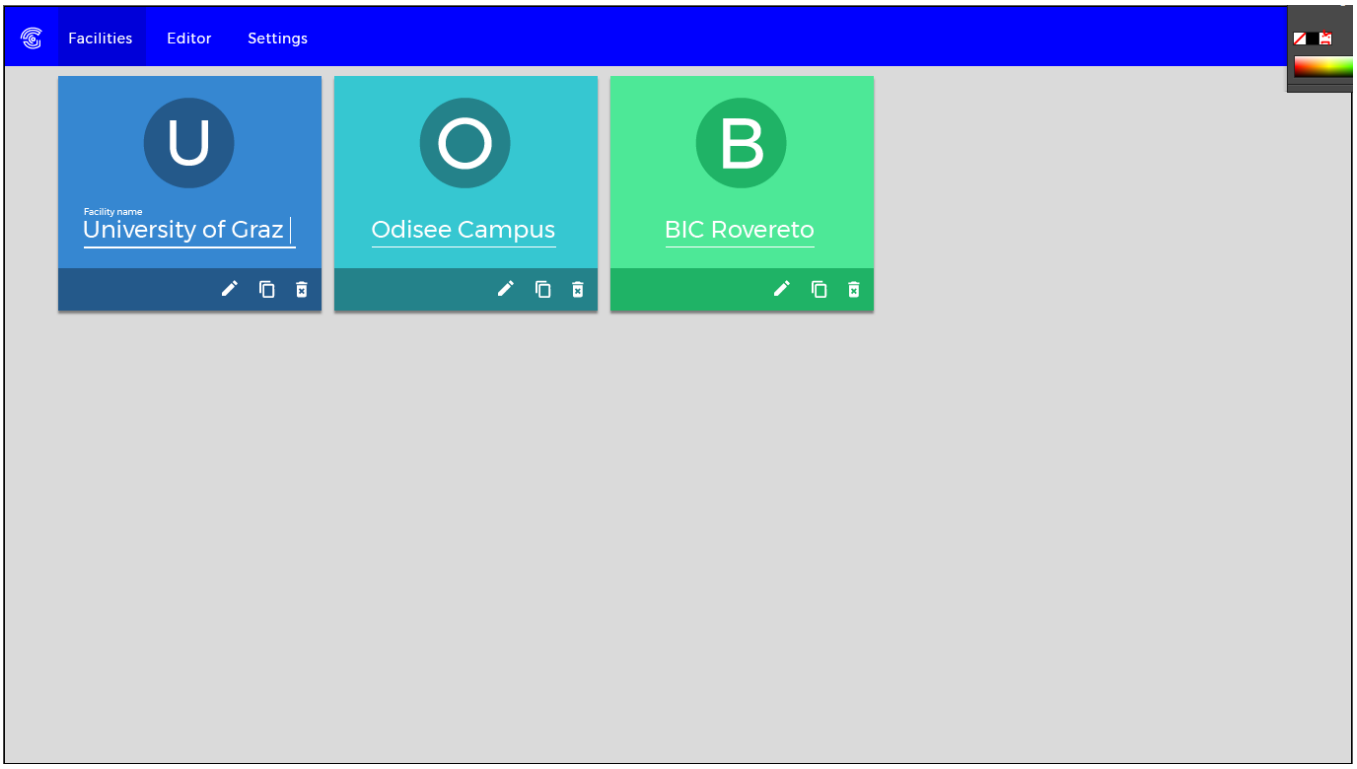
Picture 19, Manual, Map editor website

Another feature of the editor website is the ability to import an image to the editor allowing the user to draw the building on top of this image. This feature is helpful because it facilitates the procedure of the map drawing as the user can see directly the map as a guide.



Picture 20, Manual, Map editor website, image import

After its creation the facility's map can be displayed in the facilities section of the editor website. In this page the user can edit, copy and delete an existing facility.



Picture 21, Manual, Map editor website, facilities display

CHAPTER 5: RESULTS

5.1 Conclusion

This thesis was attained within the Blended Aim project. Through Blended Aim Students across Europe were assigned with the development of a product meant to keep a company's employees safe. This product was appointed by Trilogis, an Italian company specialized in advanced solutions in the fields of geography and computer science. The outcome of the project was a product named Gabriel which consists of two Android applications, three websites and the backend services.

Gabriel was designed to help companies manage the safety of their employees during their work shifts. As companies grow in size, accidents also grow in numbers and severity. The important part of accidents is to manage it quickly but in order to do so the accident has to be acknowledged fast. To achieve that, Gabriel features direct communication between an employee and a security supervisor. The supervisor has access to the employee's health record and with his discretion the position as well. With safety as its main goal, Gabriel is a tool for every company and every employee that strives for safety in his workplace.

The opportunity to work and collaborate with students from different educational institutes and fields of study was a great experience and practice. As the project was split into smaller parts the separation of the students into sub-teams was required. The students shared their thoughts, worries, issues, solutions, knowledge and different points of view on a subject as one team. This served as a learning ground for the students who were not familiar with each other's field of study, thus providing a better and more complete understanding of their areas of expertise. It also improved the perception of how to approach a problem more efficiently. Though it might seem imaginary, basic intercommunication and efficient work flow is sometimes difficult for people of different countries because of contrasting habits and work procedures. Communication and better understanding was developed further due to the intercultural background of the team.

By developing this project the students acquired more knowledge and experience. Through the usage of new technologies, which are considered state of the art in the field of informatics engineering, they became familiar and more comfortable using such technologies. Technologies like WebSockets, REST services and beacons keep rising by the day in the software development section. The combination of the above – mentioned with Android Application development is a significant foundation for a successful career.

5.2 Future work and extensions

Gabriel security system is at its first release version and it fulfills the goals for which it was developed. More features will be added to complement and improve the already existing ones.

Upcoming features	
Audio messages in the chat of the application.	Audio recording and transmission services were implemented in the application and the backend but not used.
Audio playback in the websites.	A media player needs to be implemented to play the received audio messages.
Route and checkpoint management.	This feature will allow the user to browse the routes assigned to him and modify them if there is an issue.
Route scheduling in the backend.	Implement a scheduling system to properly save the routes in the database.
Improved backend security.	Hyper Text Transfer Protocol Secure (HTTPS) and WebSocket Secure (WSS) can be used to secure the connection between the client and the server without risking the loss of data. A security but also a performance improvement would be the limitation of what REST calls a user is allowed to make and how the backend responds. (A simple user cannot make a REST call that

	returns information about other users).
Worker tasks.	This will allow the supervisor to assign “special” tasks at any moment that might come up for the employee’s shift.
Accept worker task and task management.	From the mobile application the employee can get notifications for the worker tasks. The details of the tasks can be viewed before accepting or declining the task. If the employee accepts the task a new option will be available to notify the supervisor when the task is finished.
More languages.	More languages will be added in all the websites and the application so that the user can choose.
Nearest exits and points of interest.	This feature will allow the supervisor to add points of interest such as nearest exits by modifying the map.
Visitor Application.	The visitor application was a bonus feature to allow navigation in a company’s premises. Though it is working properly it can be improved. These improvements could include a revised navigation display and more navigation options for the user.

Future Work

REFERENCES

[1] Wikipedia-WebSockets
<https://en.wikipedia.org/wiki/WebSocket>

[2] What are WebSockets?
<https://www.twilio.com/docs/glossary/what-are-websockets>

[3] Wikipedia-Android
<https://en.wikipedia.org/wiki/Android_(operating_system)>

[4] Representational state transfer
<https://en.wikipedia.org/wiki/Representational_state_transfer>

[5] Praxis
<http://www.praxisnetwork.eu/>

[6] Retrofit 2
<http://square.github.io/>

[7] GitHub-Retrofit
<Consuming APIs with Retrofit · codepath/android_guides Wiki · GitHub>

[8] What is an iBeacon
<http://developer.estimote.com/ibeacon/>

[9] Estimote Beacons
<https://estimote.com/?gclid=CjwKCAjw2s_MBRA5EiwAmWIaczKsjeShK-zdEdZpiIiPBH5xdS6AuC3gFlPZyuJSvBSGLu3necV9fBoCZukQAvD_BwE>

[10] AngularJS
<https://angularjs.org/>

[11] Wikipedia-PostgreSQL
<https://en.wikipedia.org/wiki/PostgreSQL>

[12] Wikipedia-Entity Framework
<https://en.wikipedia.org/wiki/Entity_Framework>

[13] Scrum
<https://www.scrum.org/>

[14] Wikipedia-Team Foundation Server
<https://en.wikipedia.org/wiki/Team_Foundation_Server>

[15] Microsoft TFS
<https://www.visualstudio.com/tfs/>

[16] Indoor location technologies
<https://lighthouse.io/indoor-location-technologies-compared/>

[17] Trilogis
<http://www.trilogis.it/?lang=en>

[18] Tutorialspoint-Design Patterns
<https://www.tutorialspoint.com/design_pattern/chain_of_responsibility_pattern.htm>

[19] Chain of Responsibility Pattern
<https://sourcemaking.com/design_patterns/chain_of_responsibility>

[20] Javatpoint-Chain of Responsibility Pattern
<https://www.javatpoint.com/chain-of-responsibility-pattern>

[21] JWT.io-Json Web Token
<https://jwt.io/introduction/>

[22] Wikipedia-Json Web Token
<https://en.wikipedia.org/wiki/JSON_Web_Token>

[23] Developer.android Menu
<https://developer.android.com/guide/topics/resources/menu-resource.html>

[24] Wikipedia-Material Design
<https://en.wikipedia.org/wiki/Material_Design>

[25] Materialize
<http://materializecss.com/>

[26] Svg-pan-zoom
<https://www.npmjs.com/package/react-svg-pan-zoom>

[27] Wikipedia-Observer pattern
<https://en.wikipedia.org/wiki/Observer_pattern>

[28] Tutorialspoint-Observer pattern
<https://www.tutorialspoint.com/design_pattern/observer_pattern.htm>

[29] Oodesign-Observer pattern
<http://www.oodesign.com/observer-pattern.html>

[30] Oracle Java
<https://www.oracle.com/java/index.html>