EKETour - Software Platform for Organizing Tours

Bence Bakó*, Vivien Emőke Bartha*, Károly Simon‡, Csaba Sulyok‡ and Levente Kintzel§
* Babeș Bolyai University, Cluj-Napoca, Romania
‡ Codespring LLC, Babeș Bolyai University, Cluj-Napoca, Romania
§ Codespring LLC, Cluj-Napoca, Romania
bakobence@yahoo.com
bartha_vivien@yahoo.com
simon.karoly@codespring.ro
csaba.sulyok@gmail.com
kintzel.levente@codespring.ro

***Abstract*—The target audience of the EKETour software system is any organization that manages hikes, tours and/or other outdoor events. The purpose is to create a unified registration interface for tours. This makes it easier to sign up for returning hikers, because the general information requires completion only in the first occurrence of the registration. The interface provides the possibility for a single user to register the information of multiple people into several profiles. Hikers receive NFC tags to validate at checkpoints. Organizers can use a mobile application to check passers, validate and log their data. The software is designed and implemented at the request of and based on requirements from the Transylvanian Carpathia Society (EKE). The system aims to replace/amend the current personnel on-paper hiker tracking mechanism with a digital solution.**

I. Introduction

The "Erdélyi Kárpát Egyesület"[1] (The Transylvanian Carpathia Society, abbr. EKE) is a well-known organization in Transylvania whose main purpose is to regularly organize hiking and bike tours for interested nature lovers. The tours can be distinguished in tour versions by its path and distance. The first version of the EKETour project presents a specialized software system that suits the requirements posed by the EKE association. As such, the first intention of the project is to provide a unified registration platform for the existing tours directed by the EKE team.

EKETour offers a convenient platform both for the users and the administrators. Currently, the management of the tours is elementary: if an interested party for a certain tour completes the necessary registration steps, they must reiterate the same steps every subsequent time they wish to take part.

Another factor is the management of the events; the responsibility of the organizers is the manual authentication of the participants at each specific checkpoint.

There are many difficulties presented by such a system. For example, the administrator, at the end of the day, summarizes all the information collected and written down by the organizers. The risk of misspelled or otherwise incorrect input values must be considered.

The EKETour project introduces a new solution for this problem, the card system, which provides a simpler and better organization. The participants with the received NFC cards can walk through the appointed path and they can authenticate themselves each referee. The point is that, besides the authentication simultaneously is resolved the tracking of the hikers, and from the saved and summarized data can be considered correct and relevant statistics, to be used later to fine-tune the events.

There have been similar attempts at projects aimed towards hikers in recent years. For example, Dow et al. [1] present a project at groups of tourists; when some members get torn away from the group, they can easily rejoin with the help of the application. Huang et al. [2] specialize on the Yushan National Park; their intended audience are hikers who can easily find the appointed sightseeing spots and pavilions. The presented works both try to resolve a reoccurring problem: the localization, navigation and the data collection for tracking the hikers or tourists. The main requirements share the navigation on a route with wearable devices. A posed problem is reducing excessive network communication in such scenarios.

The EKETour shows similarity with the aforementioned projects in its target audience. The implementation in the current case puts emphasis on solving a more complex problem: the management of multiple tours, a tracking system using NFC tags, and a multi-profile registration platform.

II. The EKETour Project

A. Background

In this section, the expressions used in the EKETour project are presented.

The EKE association organizes multiple events each year, usually dubbed Memorial Endurance Tours; these are dedicated to a person well-known in local history. Examples include the Mór Jókai, Károly Kós and Pál Vasvári Memorial Endurance Tours.

A tour may cover larger distances, with checkpoints specified in certain resting places en route. These are named after the area where they are placed, for example, in the Jókai Mór tour, there is the Turda Gorge (Tordai-hasadék), Tureni Gorge (Túri-hasadék), Adventure Park, etc. The participants are awaited at each checkpoint by the organizers, also known referees. They validate and record the data of the arrived

[1]Official website: http://teljesitmenyturak.ekekolozsvar.ro/en

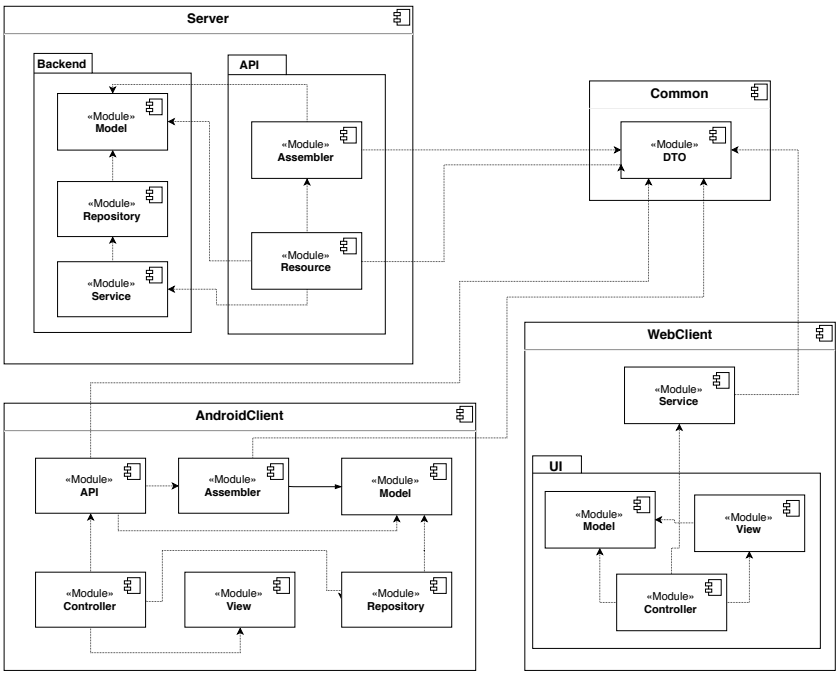

Fig. 1. Architecture of the EKETour project.

participants and guide them to the next checkpoint. The tour has one or more tour versions, which are different from each other by type (hike or bicycle), route, length, start and endpoint.

B. Functionalities

The EKETour project uses of a centralized server, which serves two types of clients: an Android mobile app and a web client. The web interface may be used by three types of users: guest, authenticated user and administrator. Each of them are given access to different functionalities. The target audience of the Android app introduces a fourth user type: the organizer. Currently, the mobile app may only be used by organizers.

The web user interface provides functionalities both for the administrator and the simple user. An administrator has the privilege to visualize the users and the profiles related to them. He/she can create tours by defining a label, a general description and a date. After that, it can be assigned multiple trip versions, which differ from each other by path, the type, the starting and the finish points. He/she then determines a registration fee in multiple currencies for each version.

An authenticated user may create user profiles. A profile contains required information, such as T-shirt size, EKE membership or the name of the organization and dining preferences (e.g. vegetarian). In the interest of a well organized event, this information is required in advance and later on used for each registration, eliminating monotone form completing.

A possible arising situation is that a younger participant (e.g. child accompanied by a parent or guardian) does not possess either a phone number or an e-mail address, but by means of the profile system, the parents can easily record their information, then achieving the registration for the suitable tour version. The user processes this registration with the profiles one by one to any of the available tours, therefore every profile is connected to the most suitable tour version independent from the user. Another possibly beneficial scenario of the profile system is a company representative (e.g. secretary, organizer) registering the employees in bulk. Everyone selects a suitable tour version for themselves and later on the collected registration fees are payed by the company collectively.

The authentication and the logging/tracking of the participants is resolved in the Android application. The content of the application is internationalized; the loaded pages adapt the default language of the device. After this, the system synchronizes the data with the server and visualizes the available tours. By selecting the current tour where the user has an organizer position, the application provides the possibility to pair a participant with an NFC card and a unique number. During the trip, the participants who pass a checkpoint are authenticated by reading the NFC card with the application, afterwards this information is saved into the local storage. If a network connection is available, the system synchronizes the new logs with the server.

C. Architecture

The EKETour project is separated into three main components: the server, the Android mobile application and the web application. Both the server and Android use a database. If there is a network connection present, the Android application synchronizes its data with the server, allowing usage even without a working Internet connection. Figure 1 shows a fourth

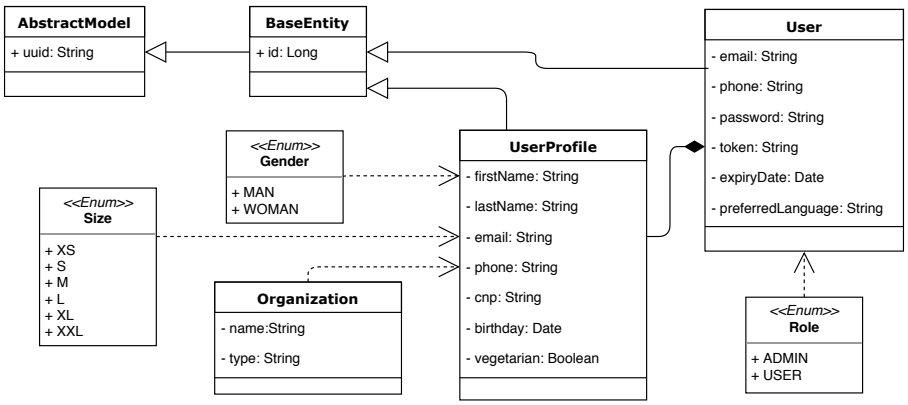

Fig. 2. Partial representation of the data model. Besides the generalization of entities, the one-to-many relationship between the User and UserProfile entities/tables are presented.

component as well: the common module, which contains the Data Transfer Object (DTO) implementation. The connection between the other three components is realized via this one, providing a uniform model representation with possible undercover data fractions, like passwords or IDs.

The server component consists of two subcomponents: the Backend and the API. The Backend contains the Model module: a collection of the entities. The Repository model implements the database related operations. It uses the MySQL relational database, with connections established via the Java Persistence API (JPA). The Service module handles the business logic of the server, requesting the Repository package where necessary.

The API subcomponent provides a RESTful web service for the clients, allowing HTTP requests. In the API subcomponent, the Assembler module performs conversions between the Backend's Model and DTOs. The Controller module responsibility is to forward the request to the Service layer. The received answers are processed by the Assembler, where the data is converted into DTOs, which are serialized and sent back to the client.

The web client is separated into four modules. The Model module contains the entity representations. The Service module implements the communication with the server; this layer requests necessary information from the RESTful API of the server. The Controller manages the View and it uses the functionalities of the Service. The Controller communicates with the Service with an Observable data stream, by subscribing to the Service methods.

The Android client contains six modules. The Model represents the entities of the application. The Repository module implements the communication with the local SQLite database. The Controller processes the events generated on the View. The API module handles the communication with the server's RESTful API.

D. Data model

The entities of the EKETour system are represented as Java Beans with corresponding Java Persistence API annotations. The entities possess private properties, which can be accessed by the public getter and setter methods that follow a naming convention, respectively have a no-argument constructor.

The `AbstractModel` class, which implements the `Serializable` interface by having a universally unique identity property, provides the possibility to easily distinguish the objects created in the system. The `BaseEntity` class extends it with another id property, which functions as a per-entity unique identifier.

The descendant classes use the corresponding JPA annotations to create their schema; with this, each class is mapped with the corresponding table from the database. Some entities are considered secondary information collections or have only one referrer, therefore mapping them to a table becomes unnecessary. These entities are embedded into the table of referrer. These classes are similar to the entity ones: they extend directly the `Serializable` interface but do not contain the id property.

The `User` data model represents authenticated users. Its shell contains some specific properties like the one which stores the lastly chosen language on the web interface, by this attribute the system handles in which language the content must be displayed. In the other hand, there is a token and a set expiry date property pair, which maintain the possibility to change the forgotten password.

The `UserProfile` entity stores the personal information about a person: full name, phone number, email, CNP[2], birthday. Furthermore, the name and type of an affiliated organization may be defined, as well as information about the required T-shirt: the size and the type (sewing). For security reasons, the User entity contains a property (`UserType`) which defines their role, taking on predefined values from an enum.

III. Technologies

A. Server

The EKETour server program is built using the Spring lightweight framework, which is based on the Inversion of

[2]Cod Numeric Personal, regional equivalent of a Social Security Number

Control[3] (IoC) design principle. IoC manages the life cycle of the components and resolves the dependencies between them automatically on runtime, by applying the Dependency Injection pattern. The scope of the IoC container is to manage objects from a particular class, configure, assemble them by reading the configuration metadata (in this case, the Java annotations).

Spring Boot[4] aids the creation of the EKETour server-side project. It provides predefined configurations that are common for certain project types. Based on the provided dependency packages, it inserts particular modules into the application.

The Spring Data[5] dependency provides the possibility to switch between different technologies that provide access to different types of databases. By abstracting the methods, it significantly reduces the amount of repeated code required to implement data access. In the EKETour project, the Spring Data JPA module is used, which creates a higher abstraction layer on top of the Hibernate ORM (Object Relational Mapping) framework.

The JPA module supports defining a query manually or creating it dynamically based on the method name. Most of the time the second strategy is used, but in some cases the solution is to annotate the query method with `@Query`. The value of the annotation is the query defined in JPQL language.

The applied communication in the EKETour project is based on the REST software architecture and is realized by a RESTful web service, namely the Spring Web MVC module. This module provides serialization and deserialization of all payloads in JSON format.

The client applications can indirectly access or modify the resources provided by the server, depending on which HTTP method (POST, GET, PUT, DELETE) and Unified Resource Identifier is used in the particular request.

To secure the aforementioned communication, the server uses the Spring Security framework, which provides a restriction for URL accessibility affected by the particular role. For example, paths starting with the admin-specific prefix can be accessed only with administrator privileges. Password encryption with BCrypt is applied for system security.

B. Web

The web interface of the EKETour project is a single-page application, provided as static files served by the server. The application is developed in the Angular 4 frontend framework [6]. Its primary language is TypeScript, which is a strict superset of the ECMAScript 6, which, along with event-driven, functional and prototype programing styles provided by the JavaScript, also supports object-oriented programing paradigms.

The architecture of the Angular framework is characterized by a component hierarchy, supports many design patterns, like reactive programing, unidirectional data flow and centralized state management. In the project, the MVC design pattern was applied.

For handling the asynchronous events, the Observable data stream is used, which is implemented in RxJS. Therefore the unidirectional data flow is also implied, which is fulfilled by the following steps: on the view an action is triggered, by this action the state of the application changes which modifies the certain view.

The Angular client application consumes the Spring MVC-based RESTful web service. This communication is realized with the HttpClient[7] technology. This uses the well-known XMLHttpRequest browser API for the HTTP request execution. It is possible to directly access the JSON response by subscribing to the aforementioned Observable which is the return value from the certainly executed method.

The web user interface obtains a pleasing view by using the Material Design framework. Material is an adaptable system of guidelines, components and tools. The internationalization (i18n) library used by the Angular provides the translation of the web page's content.

C. Android

The EKETour Android application entities are represented as Java Beans, with the corresponding OrmLite[8] annotations, showing similarities to the server-side model both in terms of structure and hierarchy.

In the EKETour mobile application, communication with the server is implemented through RESTful services, using the Retrofit package. For operations requiring network access, the data is synchronized: the data retrieved from the server will also be saved in the database of the mobile device.

The data models include a modification date field, so the application only requests data altered since the last modification date from the server.

The android.nfc package contains an `NfcAdapter` class, which gives the possibility to make a connection with the NFC[9] sensor.

IV. Tools and methodologies

The development phase of the project employs the Scrum iterative and incremental framework. Efficient teamwork is a principal intent, therefore the Git version control system[10] is used, with older project versions simply available for restoration and comparison. The development phase uses git flow: for each functionality, a new branch is created, and if one of them is reviewed and is considered finished, the changes can be merged into the stable development branch.

GitLab is a project management tool by providing a Kanban Board where the progress of the project can be traced, is a repository management system, therefore it provides solution for storing the source code and the pipeline system, which also realizes the Continuous Integration process. A pipeline is built up from different jobs, which are grouped into stages. The defined jobs separate the code analysis from the building phase and from the test phase. When a change is pushed into the main repository, the CI system first builds the application and then runs the automatic tests. The developers are notified by the system if either stage fails.

During the development, the Java source code quality assurance is assisted by two static code analyzers: FindBugs and

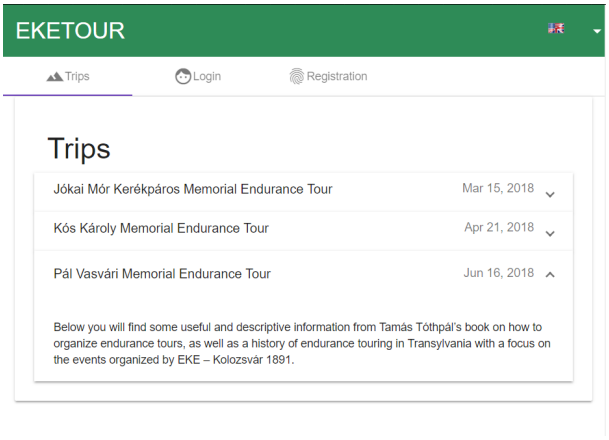

Fig. 3. Home page

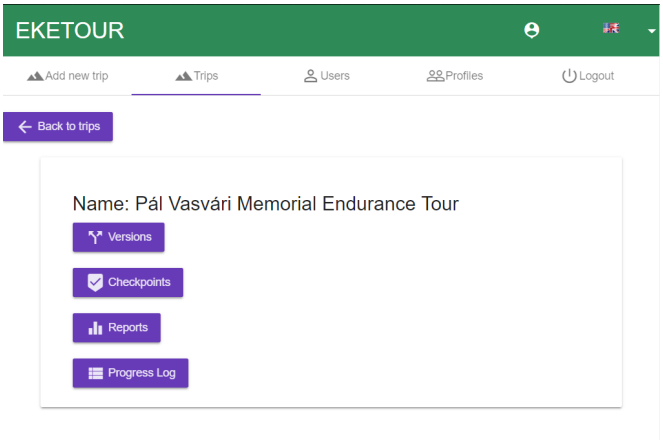

Fig. 4. Page view after selecting a tour.

Checkstyle. For TypeScript, code conventions are ensured using the TSLint static code analyzer. The individual methods in the Java source code are unit tested with the JUnit framework. Furthermore, integration tests are provided with the Arquillian framework to examine the communication and connection between components. The Android application is also tested with an automation testing tool, namely the Espresso UI tester.

Gradle compiles the Android application and the server module, and automatically retrieves dependencies. In the case of the web module, npm (node package manager) resolves the compilation, building and front-end dependency management.

V. The Usage of the EKETour Project

A. Web interface

The users can use the web application to access the software system functionalities. The default language of the web application is Hungarian, but it can be changed by selecting the flag icon in the header of the web page, as seen in Figure 3.

In the menu, the guest users can check the registered tours and the information connected to the activities: description, date, tour versions, registration fees, start and finish points. In order to access other functionalities, the users have to log in or register into the system, providing their e-mail addresses and phone numbers.

The authenticated user can create a new profile. This contains some information that is required for tour registration, like T-shirt size, membership of EKE community and meal preferences. A user has the opportunity to create more than one profile; they can add family members, friends or colleagues.

The users can browse the available programs under the *Tour* menu point. Here, all tour versions related to the chosen tour are shown. The user performs the registration for a tour with one of their profiles by selecting the most suitable version.

The users with administrator privileges can access more functionalities, such as listing the information of other users, managing the tours and tour versions, etc.

Under the *Add new tour* menu the administrators can create new tours. The created ones can be listed under the *Tour* menu point. The administrator user manages the information connected to a selected tour: the available versions, checkpoints, statistics and progress logs.

The administrator is able to define registration fees for a selected tour version. This can be done by clicking the *$ icon* in the respective row. The registration fee is specified in different currencies (currently RON, HUF and EUR), since the tours are addressed to an international audience.

The administrator user can set checkpoints connected to the tours by providing a meaningful name and the time interval(s) they are open. These checkpoints are listed under the *Checkpoints* menu in the tour settings page.

Under the *Tours statistics* page, the administrator can create tabulations where the statistics are listed. For example, they can request the number of participants who ordered T-shirts, and from which types.

Under the *Progress Log* menu, the progress of the participants is listed: which user passes a given checkpoint and the time of event occurrence. The progress logs can be filtered by participant.

B. Android client

The Android application is mainly designed for organizers and referees. To use the app, the designated smartphone must support the reading and writing of NFC tags.

The referee automatically receives the necessary information from the server by opening the app: the organized tours and the lists of participants. They can also request synchronization manually by tapping the *menu icon*, then selecting the *Sync* option from the appeared list. Afterwards they can choose the tour they are involved in as organizer from the list of tours. There are two options available on the displayed surface, as shown in Figure 5.

The organizer selects the first option (*Pair with NFC*) at a starting point, where their task is to register the arriving hikers to that specific tour. They search the participant from the list by name, insert the previously assigned sequence number into the blank field, after which they pair an NFC tag with the participant by moving it close to the phone (see Figure 6).

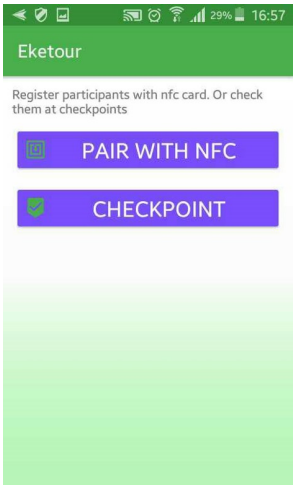

Fig. 5. After selecting a tour

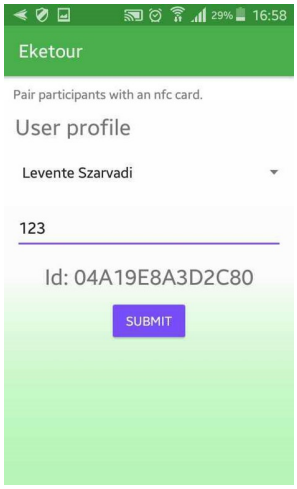

Fig. 6. Pairing NFC with a participant

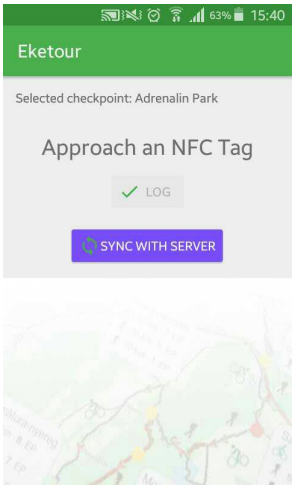

Fig. 7. NFC authentication

Clicking the Submit button the pairing is uploaded to the server.

The organizers waiting at the next checkpoints are responsible for identifying the passing participants. The second option(*Checkpoint* button) provides the opportunity for this, which displays all the checkpoints on the tour route. The organizer selects the checkpoint representative of his/her location and starts logging the passing hikers. Approaching an NFC tag prompts (see Figure 7) the display to show the name and the received sequence number. By tapping the Log button the data is validated, then by selecting the *Sync with server* button, the synchronization process will start, which sends the locally saved logs to the server, if there is a network connection.

VI. Conclusions and further development plans

The EKETour software program successfully provides a unified platform for the management and registration of hikes/tours, facilitating the tasks of both the organizers and the participants. The Android application provides the opportunity to register hikers by pairing them with NFC cards, to identify them and to log information about the performed operations. In the web user interface, the administrator can create tours, version of the respective tours and checkpoints. The administrator is able to set participation fees for every tour version, is able to view reports about the completed tours, for example, the requested T-shirts or the progress of participants at a specific tour.

As a further development, snapping and storing profile pictures for all participants at the starting point could help identify them. The referees would be able to check the pictures at checkpoints by reading just an NFC tag.

Because the tours can take a longer time, finding a hotel or a means of transportation back home may be a problem for some participants. As further development, transportation and accommodation options could be introduced into the system, the preferences of which could be provided by participants upon registration, similarly to the ordering T-shirts.

The online payment of the tour fee would simplify the work of the organizers. The fees for the tours could also vary for different age groups, after reaching tour deadlines or based on other discounts. Adding these rules to the system, the participants could also pay the registration fee over an online bank transfer immediately after the registration.

As another planned extension, displaying the checkpoints on a dynamic map would show the route of the tour. In case of multiple tour versions, users could easily see the difference between routes. For the referees, it would possible to validate a checkpoint by GPS tracked location of the smart phone.

References

[1] C.-R. Dow, Y.-Y. Chang, C.-W. Chen, and P.-Y. Lai, "A mobile group tour tracking and gathering system," in *Information Technology: New Generations*, S. Latifi, Ed. Cham: Springer International Publishing, 2016, pp. 293–302.

[2] Y. T. Huang, Y. C. Chen, J. H. Huang, L. J. Chen, and P. Huang, "Yushannet: A delay-tolerant wireless sensor network for hiker tracking in yushan national park," in *2009 Tenth International Conference on Mobile Data Management: Systems, Services and Middleware*, May 2009, pp. 379–380.

[3] K. D. Rod Johnson, Juergen Hoeller. Spring framework reference documentation, part iii. core technologies. [Online]. Available: https://docs.spring.io/spring/docs/3.0.x/spring-framework-reference/html/beans.html

[4] J. L. Phillip Webb, Dave Syer. Spring boot reference guide. [Online]. Available: https://docs.spring.io/spring-boot/docs/current-SNAPSHOT/reference/htmlsingle/

[5] Spring data jpa - reference documentation. [Online]. Available: https://docs.spring.io/spring-data/jpa/docs/current/reference/html/

[6] Y. Lasorsa. The missing introduction to angular and modern design patterns. [Online]. Available: https://bit.ly/2rSB29O

[7] S. Kryvets. Simply about new httpclient in angular. [Online]. Available: https://sergeome.com/blog/2017/11/26/simply-about-new-httpclient-in-angular/

[8] G. Watson. Ormlite - lightweight object relational mapping (orm) java package. [Online]. Available: http://ormlite.com

[9] Android nfc. [Online]. Available: https://developer.android.com/reference/android/nfc/package-summary

[10] V. Driessen. A successful git branching model. [Online]. Available: http://nvie.com/posts/a-successful-git-branching-model/