Diaspora Mapping and Collaboration Platform for Expatriates

Kincső Tüzes-Bölöni
Babeş–Bolyai University
Cluj-Napoca, Romania
tuzesbkincso@gmail.com

Zsuzsanna Borsay
Babeş–Bolyai University
Cluj-Napoca, Romania
zsborsay@yahoo.com

Csaba Sulyok
Babeş–Bolyai University
Cluj-Napoca, Romania
csaba.sulyok@gmail.com

Károly Simon
Babeş–Bolyai University
Cluj-Napoca, Romania
simon.karoly@codespring.ro

***Abstract*—In the current globalized world, it is a frequent phenomenon that people from different regions uproot and move to other countries, working and living there temporarily or long-term. But how could the origin regions follow the actual residency of their habitants, and how could they keep the data of the population up to date? The aim of the E-migrated web application is to measure and dynamically visualize the dispersion of specialists originating from a given country. It also gives the opportunity of interconnecting these people and to create a community, encouraging its users to build relationships and cooperate. It is a virtual link for the professionals originating from a common region, independent of current location.**

This paper aims to present the E-migrated platform, describing the architecture of the system, mentioning the details of the implementation, the main technologies, tools and methods used during the development process. It presents the functionalities of the application, and demonstrates its behavior, using some case studies.

***Index Terms*—collaboration platform;dispersion of expatriate specialists;professional knowledge transfer**

I. Introduction

There are many regions worldwide from where a multitude of people decide to uproot and start working, studying and living in other countries. But how large are these groups, where do they settle down and in what proportion are they distributed in other countries? Csala Dénes is searching for the answers to these questions in his project called SZÉKELYDATA [1]. The goal of his project is to measure the dispersion of Székely professionals. Székelys are a subgroup of Hungarian people, who mostly live in Eastern Transylvania.

The technique of Csala Dénes for uncovering this data is to manually gather and log relevant social media (Facebook) information. The results are shown on a non-interactive world map with clustered markers and pie charts, representing the number of Székelys currently living in a given country and the Transylvanian locations from which they originate.

In this paper, we present the E-migrated software project; an initiative inspired by the ventures of Csala Dénes. It is the part of the project Digitális Székelyföld (Digital Székely Land), that was initiated by the IT Plus Cluster of Székely Land. E-migrated is an interactive and rethought alternative of SZÉKELYDATA; a web application depicting its participants clustered on a world map. It offers to dynamically filter specialists by profession, country or location. If a location of a user changes, he/she can also modify it on the platform to keep dispersion statistics up to date. Besides visually presenting the diaspora of a territory, it also represents a collaboration platform. It creates a community and attracts potential users with a multitude of functionalities.

The general goal of the project is to explore experts originating from a given region and to create a comprehensive register of them. Furthermore, it allows professional knowledge transfer and the exchange of experience for its users. It creates an elite social network for prominent members of general fields, and would cooperate to contribute the economic and technological development of their originating country. Although the application is configured to be used by users from Székely Land, it is easily adaptable for any region in the world.

II. The E-migrated project

A. Functionalities

The target audience of the software system are professionals originating from a common region, who have achieved remarkable results in their respective fields. The joining of a user to the system is based on invitation. After registration, he/she can invite other people, but the number of invitations is limited, in order to encourage everyone to thoroughly consider who deserves to be the part of this community.

When joining E-migrated, one must read the policy of the application about data protection. This policy meets the European GDPR (General Data Protection Regulation) standards and describes the data handling and data storage of the software. By clicking to Registration or any Save button at profile modification, users agree that they have read and agree with the policy, which is also available at any time after registration.

User profiles include all the personal data given at registration (full name, profession, residency etc.) and a short biography. Furthermore, users can share posts and list them separately or together with the posts of every member of the system (see Figure 1).

An authenticated user may delete his/her E-migrated account, after confirming their password. He/she can choose complete deletion, in which case all personal data will be deleted from the database, including all posts. But he/she can also choose to keep their posts, in order to make them available to others. In this case, every relevant post is assigned to an

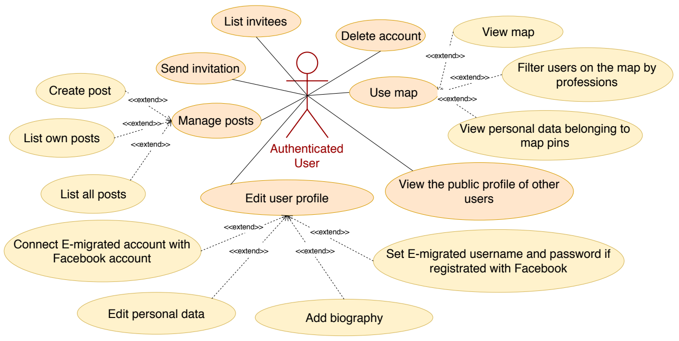
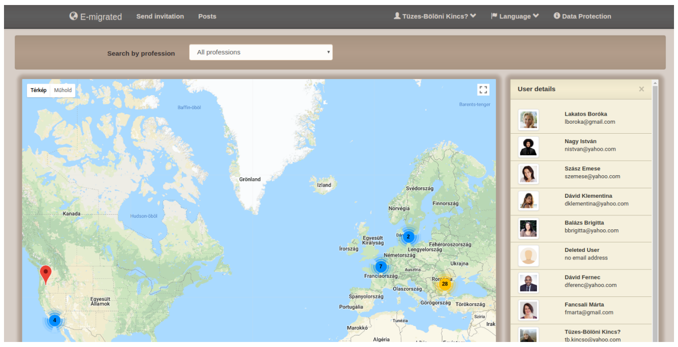

List invitees
Delete account
View map
Send invitation
Filter users on the map by professions
Create post
Use map
Manage posts
List own posts
View personal data belonging to map pins
Authenticated User
List all posts
Edit user profile
View the public profile of other users
Connect E-migrated account with Facebook account
Set E-migrated username and password if registrated with Facebook
Edit personal data
Add biography
<<extend>>

Fig. 1: Use case diagram for authenticated users



Lakatos Boróka
lboroka@gmail.com
Nagy István
nistvan@yahoo.com
Szász Emese
szemese@yahoo.com
Dávid Klementina
dklementina@yahoo.com
Balázs Brigitta
bbrigitta@yahoo.com
Deleted User
no email address
Dávid Fernec
dferenc@yahoo.com
Fancsali Márta
fmarta@gmail.com
Tüzes-Bölöni Kincs?
tb.kincso@yahoo.com

Fig. 2: Home page for logged in users

"anonymous" account, after which the account of the user may be safely deleted.

The main page of the application contains a world map, which contains pins and marker clusters representing the users of the platform, so they can be filtered by location, region or profession (see Figure 2). Authenticated users can see the personal data belonging to the pins, and they can contact each other with the contact information on the profile pages. Guests can only see the pins; they are not allowed to see any personal information about others.

In case an interested party wishes to join the platform, but lacks a valid invitation, he/she is able to request one by sending a curriculum vitae and a short motivational letter.

Upon receiving an invitation, there are two possible ways to join. Simple registration involves filling out a form with the personal data required by the standard profile, like name, profession, e-mail address etc. The alternate way is registration using a social network. Its advantages are that registration is simple, fast, and it does not require filling out forms, because the data appearing on the profile is received from the given social network, and it can later facilitate the log in as well. The E-migrated platform currently supports Facebook, with plans to include support of Google+ and LinkedIn.

The administrators are also users of the system, but after switching to admin mode, they are able to evaluate the invitation requests and to edit the list of available professions. They can list all the users, and in justified cases may suspend or reactivate accounts.

B. Architecture

The project consists of two main components: the server and the web client (see Figure 3). The communication between them is based on HTTP requests which conform to the REST (Representational State Transfer) conventions.

The persistent data of the system is stored in a MySQL database, and the server-database communication is built on Hibernate: the Java Persistence API (JPA) implementation provided by the Spring framework.

The modularity of the server comes from the multilayer architecture of the component [2]. Its advantages are that different modules are reusable, easily replaceable without changing the other layers, the code becoming clearer, more understandable, and the exception handling more efficient.

The *data access layer* performs the data handling of the application on database level. The Model and Repository components belong to this layer. The *Model* contains the main entities; these are simple POJOs (Plain Old Java Object) with private attributes and public getters and setters. The server-database communication is accomplished by the *Repository* component, which applies the DAO (Data Access Object) design pattern. This provides interfaces which ensure simple data access to higher layers, separating the application logic from the lower level database access.

The *business logic layer* specifies the interaction possibilities with the domain data of the software. The presentation layer is not allowed to access directly the essential entities and it does not have to know the details of the data access implementation. Therefore it uses the services provided by the business logic layer. The *Service* component is based on the "facade" design pattern, providing available services for the API through different interfaces. The corresponding implementations of the interfaces use the methods of the Repository component. The *API* ensures REST resources for the presentation layer, so it receives HTTP requests and sends HTTP responses based on the REST principles.

Nevertheless, the presentation layer needs different properties, or property combinations of business logic objects. In order to reduce the HTTP requests, the serialization of the messages between web client-server is implemented with DTOs (Data Transfer Object). A DTO contains all the attributes required by a view, and the *Assembler* component performs the necessary conversions between the Model and DTO elements.

The *presentation layer* is responsible for the behavior of the user interface. The *Web Service* serves the requests of the Web Controller, communicating with the API component of the server, and uses the DTO elements of the server to represent data. The *Controller* component controls the View, using the Service layer. The *View* layer provides the appearance of the user interface. The presentation layer applies a client side MVC (Model-View-Controller) pattern, so it separates the view from the control and data. The traditional MVC is moved to the client side, so the server responses

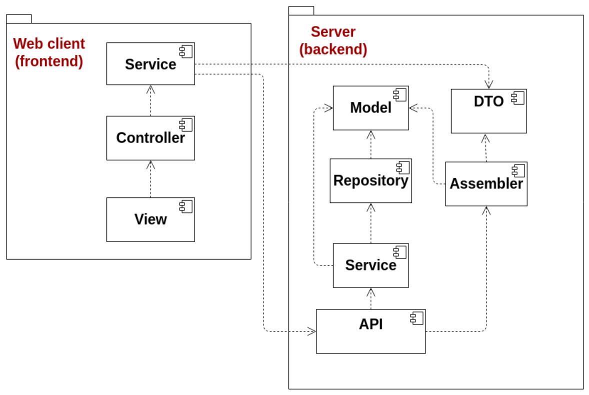


Fig. 3: The main components of E-migrated software system

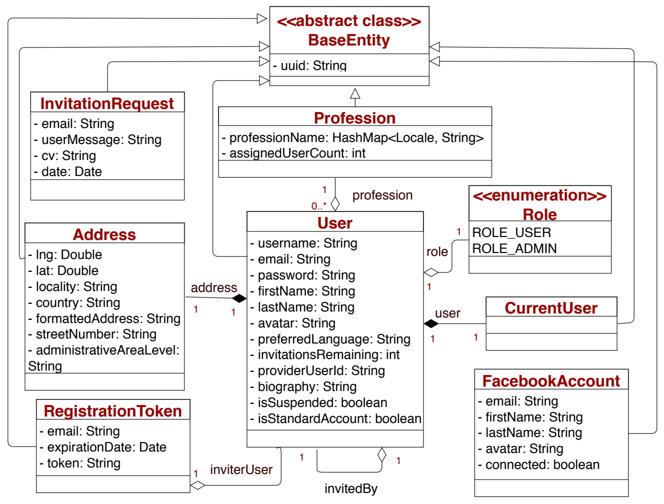


Fig. 4: The Domain model diagram of E-migrated software system

are not fully rendered views simply shown by the browser. Instead, the server provides the static resources of the web client together with the REST endpoints. The view rendering is afterwards accomplished by the browser, using the data received asynchronously from the REST endpoints.

C. Domain Model

The entities of E-migrated are Java Beans, which are serializable classes containing getters, setters and a constructor without parameters. Every bean is the descendant of the `BaseEntity` class, which has a single attribute, the UUID: a unique, system-level identifier, which is the primary key of the tables that represent the principal entities in the relational database (see Figure 4).

The main entity of the application is the `User` class, which contains user-related attributes, such as full name, password, e-mail address, profile picture, biography, preferred language, etc.

Every user has a home address, which is stored in a separate table in the database and is assigned to the User table with a one-to-one relation. It is possible for multiple users to share a common address, but for simpler modification of an address without impacting other accounts, a one-to-one relation is chosen. The `Address` class has a primary role at representing the users in a clustered form in a world map: the `lat` and `lng` attributes store the latitude and longitude of an address.

The professions are also represented by a separate table in the database, and the two entities are connected with one-to-many relationship. The `Profession` class contains the number of the belonging users, and a key-value pair, encompassing the name of the profession in the different supported languages.

The `Post` class symbolizes the posts created and shared by users. It contains the title, text, upload date and the uploader.

The `InvitationRequest` entity represents pending invitation requests from those wanting to join the E-migrated community. It includes the e-mail address of the interested person, a curriculum vitae, short biography and the date of the request.

The invitation-based registration is provided by the `RegistrationToken` class, which stores a unique token, its expiration date, the e-mail address of the invited person, and the inviter.

The role based access management is implemented using the `Role` enum. Its possible values: `ROLE_USER` or `ROLE_ADMIN`. A role is assigned to every user, that is automatically set to `ROLE_USER` at registration.

III. Server side technologies

The server side of E-migrated has been written in Java using the Spring framework, which provides a comprehensive infrastructure for Java-based applications [3]. In order to implement a runnable version of the application quickly and easily, the Spring Boot framework is also used, whose power lies in the ”convention over configuration” principle [4]. This section presents the main technologies used on the server side, the different architectural layers, the behavior of components and their communication.

The communication between server and the relational database is realized using Spring Data JPA, which provides a modular and general solution for data persistence/data access [5]. Using the Spring Data project, the programmers do not need to write queries in order to implement the data access layer, they just have to implement the interfaces provided by Spring Data and the framework generates the implementations based on the method names.

In the service layer of the project, there are Service Spring beans, all of them injecting a corresponding Repository interface in order to use its methods for implementing the main part of the application logic. They publish interfaces used by the API layer, with no knowledge of the underlying implementation details. These beans handle and log errors in a general way, throwing layer-specific exceptions. The `UserService` interface is responsible for the user-related operations, such as registration, login, changing a username or password, and it returns the users based on the incoming parameters. The FacebookConnectionService interface handles the registration and account connection with Facebook.

The API layer manages the requests from the client, publishing its resources through RESTful endpoints. It handles the requests, processes them and sends appropriate responses. In order to implement the API layer easily, the spring-web package is used.

When a request arrives, a handler defined as a Resource class is executed. These handlers use the interfaces to the service layers and pass the requests to them, if they need more complex processing. They also handle the different exceptions, incorporating them and sending appropriately transformed error messages to the client.

The data sharing between the client and server happens in JSON message format, by serializing DTOs. Each DTO has an associated Assembler interface, which is structured in a hierarchy with an abstract assembler on top containing conversion methods between model and DTO objects. Some assemblers incorporate default values for model objects, read from configuration files.

To implement the application security, the Spring Security framework is used, which offers comprehensive security solutions for enterprise Java applications [6]. The application-specific usage of Spring Security dictates the aspects of authentication and authorization; BCrypt is used as the password hashing library, and forbidden resource access is handled in a general role-based way.

The project provides a custom HTTP path-based authorization system: all paths starting with /guest/** are available for everyone, but paths starting with /user/** or /admin/** are available only for users with the respective role. The same convention is used also in the case of REST resource paths, starting with /api.

In favor of the integration of social networks, for making the registration and login more efficient, the Spring Social module is used. It provides plausible solutions for creating connections between a Spring project and a Software-as-a-Service API [7]. Since registration into the platform is based on invitations and Spring Social offers no support for that by default, a customization is applied by implementing a range of adapters and interceptors.

E-mail sending is resolved using the HTTP API of Mailgun, which supports the SMTP, POP3 and IMAP protocols. In order to use the services of the mail sender, a range of configuration is needed, which are read from a properties file. The dynamically built mail bodies are rendered based on Freemarker templates using the Spring Web MVC (Model View Controller) framework. Sending is realized using the JavaMailSender interface supported by the Spring framework [8].

IV. Client side technologies

The web component of E-migrated is a single-page application, where the handling of the different views and the building of the page is implemented with Angular.js [9]. The Bootstrap framework [10] has a proper role in the representation of the user interface and in the realization of its responsive design. This section presents the structure of the web client, the functioning of its components and the main technologies that make up its implementation.

Angular.js is a JavaScript based framework for web applications, that extends the dictionary of HTML with directives. Its main advantages are that it supports the client side MVC, two-way data binding and dependency injection. The client side MVC pattern facilitates the expansion, maintenance and testability of the software by separating the business logic from data handling and view representation. Bidirectional data binding ensures the automatic synchronization between the model elements and the DOM elements of the user interface, sparing the programmers from manual refreshing and shortening the amount of corresponding code. The dependency injection helps to create clear and testable code, because it supports the automatic injection of external dependencies from Angular components, without knowing their implementation details.

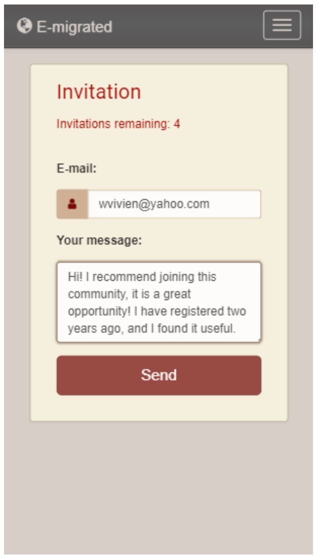
E-migrated uses the Angular UI-Router [11] library in order to place more views within a single HTML page, every view having its own name and controller. Thus the views are reusable. The main page of the application contains two ui-view directives, one containing the menu situated in the header and one that includes the main content. These ui-views are replaced dynamically with different states. The state-view mapping is set in the main configuration file of the front-end of the application, where a template URL, an effective URL, a controller and corresponding permissions are assigned to every state. When a user does not have the permission to reach a state, the he/she is redirected to the home page.

The Angular Google Maps [12] third party library is used in order to display the world map in the home page. It ensures directives that facilitate the integration of the Google Maps API into Angular.js applications, without knowing all the implementation details of it. It also provides well known Google Maps objects, such as markers, windows, lines on the map.

The Angular Translate [13] library is responsible for the internalization (i18n) of the project, happening on the client side. Its lazy loading mechanism ensures that language specific data requests from the server are only sent out when needed. It contains built-in directives and filters, the corresponding dependency has to be injected in the main module, and a JSON file has to be created for every supported language.

The look and responsiveness of pop-up windows, and informative or warning messages in the application are implemented using the Sweetalert2 third-party library, which is advantageous by supporting the personalization of the appearing content.

The testing of the front-end is implemented using the Jasmine framework and Karma test runner. Jasmine is a behavior-driven framework that supports the testing of the user interface, while Karma helps the run of Jasmine tests



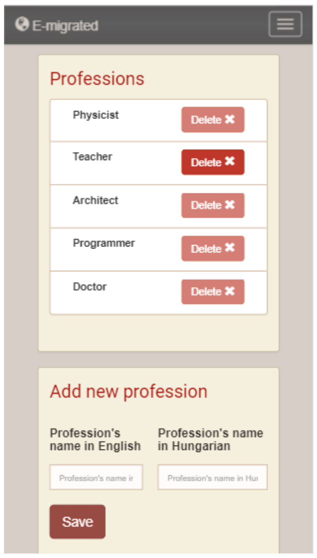
Invitation

Invitations remaining: 4

E-mail: wvivien@yahoo.com

Your message: Hi! I recommend joining this community, it is a great opportunity! I have registered two years ago, and I found it useful.

Fig. 5: Send invitation page



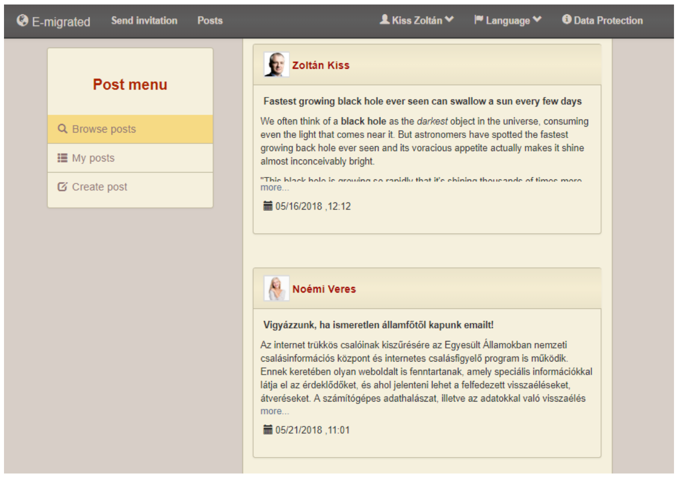
Professions

Physicist	Delete ✖
Teacher	Delete ✖
Architect	Delete ✖
Programmer	Delete ✖
Doctor	Delete ✖

Add new profession

Profession's name in English	Profession's name in Hungarian
Profession's name ir	Profession's name in Hu

Fig. 6: Edit professions page



Zoltán Kiss

Fastest growing black hole ever seen can swallow a sun every few days

We often think of a **black hole** as the *darkest* object in the universe, consuming even the light that comes near it. But astronomers have spotted the fastest growing back hole ever seen and its voracious appetite actually makes it shine almost inconceivably bright.

05/16/2018 ,12:12

Noémi Veres

Vigyázzunk, ha ismeretlen államfőtől kapunk emailt!

Az internet trükkös csalóinak kiszűrésére az Egyesült Államokban nemzeti csalásinformációs központ és internetes csalásfigyelő program is működik. Ennek keretében olyan weboldalt is fenntartanak, amely speciális információkkal látja el az érdeklődőket, és ahol jelenteni lehet a felfedezett visszaéléseket, átveréseket. A számítógépes adathalászat, illetve az adatokkal való visszaélés

05/21/2018 ,11:01

Fig. 7: Browse posts page

in different browsers and devices. Karma also summarizes the test results and provides visualization options to the developer team.

V. Development Methods and Tools

The development methodology of the project respects the Scrum standards, ceremonies and roles. Accordingly, the development process is divided into two weeks long iterations. At the end of every sprint, a new functional version of the application is completed.

For version control, the team uses git, while GitLab serves as project management, code base and CI/CD (Continuous Integration/Continuous Deployment) system. GitLab offers development tools for the full life cycle of the software, like a built-in Kanban board, sprint milestones, repository management and continuous integration system. As version control method, the team uses the Git Flow branching model. Accordingly, the implementation of every user story is done on a new dedicated feature branch. After approving the corresponding merge requests, they are merged into the `master` branch, which holds the stable version of the application.

The continuous integration provides the correct functioning of the software after integrating new functionalities. After every push, a CI pipeline is started and every stage that is defined in `gitlab-ci.yml` is executed. In the case of E-migrated, if the push is performed on the `production` branch, the CI pipeline consists of four stages: building and Docker image creation, running of tests, static code analysis and deployment to a staging test server. Otherwise, the first stage does not include Docker image creation, and there is no fourth stage executing the deployment. The order of the stages guarantees that only a functioning, tested version of the application is deployed to the test server, because if one stage fails, the subsequent ones are not executed.

The deployment of the project uses Docker. A container is created for the server, and another for the MySQL database, so it is necessary to use docker-compose [14], which enables the configuration and management of programs that consists of more, separate containers.

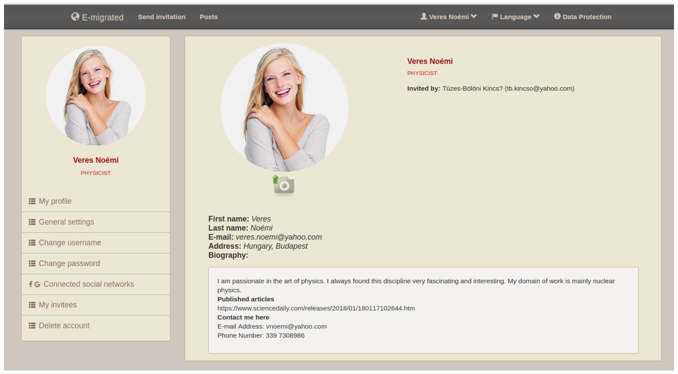
The dependencies of the client side are managed by npm (Node Package Manager) and automatically packed with the source code by Gulp. In the `gulpfile.js` file, different tasks are defined for building the code base or for static code analysis using JSHint, FindBugs and Checkstyle. It is possible to create an observer task, which automatically refreshes the modified files (ex. js, css) also in the build folder of the application. Another task is the 'gulp-concat', that concatenates all the JavaScript files to one, called `main.js`, which can be referred to from the `index.html`.

As a build and dependency management tool on the server side, Gradle is used, which supports the automatic download and configuration of external dependencies, and helps to create multi-module projects. It also generates Eclipse projects, runs tests and static code analyzer tools, and extended with different external plugins, it enables the full automation of the build process. In case of E-migrated, the full project is built using Gradle, because its Gulp plugin also builds the front-end of the application, resulting in one end-to-end tool chain.

VI. The Behavior of the E-migrated Application

This section presents the functioning of the E-migrated application and describes how users may interact with it.

The main page of the application contains a world map powered by Google Maps, including multiple marker clusters that show the number of registered users, distributed by region. By zooming in on the map, these clusters are split up into more pins, representing the users of the system more precisely. Guests are not allowed to see any personal information about the users represented on the map. They can only see the markers and filter them by profession. An authenticated user can also see who lives in a given location or country by clicking on the pins, furthermore he/she can navigate to the



Veres Noémi
PHYSICIST
Invited by: Tüzes-Bölöni Kincs? (tb.kincso@yahoo.com)

First name: *Veres*
Last name: *Noémi*
E-mail: *veres.noemi@yahoo.com*
Address: *Hungary, Budapest*
Biography:

I am passionate in the art of physics. I always found this discipline very fascinating and interesting. My domain of work is mainly nuclear physics.
Published articles
https://www.sciencedaily.com/releases/2018/01/180117102644.htm
Contact me here
E-mail Address: vnoemi@yahoo.com
Phone Number: 339 7308986

Fig. 8: Edit profile page

public profile page of a user from the map. If a marker cluster includes more than three users, then a panel appears on the right side of the window, with the list of those users (see Figure 2).

The main menu is on top of every page in the application. It allows every user to select their preferred language, and for guests, it allows logging in. By clicking on the *Login* menu item, they can choose between the default *Login* and *Login with Facebook* buttons, or they can navigate to the request invitation page, if they are not registered yet.

For authenticated users, the top menu is extended with other menu items. For example, by clicking on the *Send invitation* item, users may invite their friends to join (see Figure 5). Under the *Posts* menu item, they can navigate between public posts, create new posts, or list their own posts (see Figure 7).

By clicking on their name, they may navigate to the edit profile page, where users can perform a range of operation with their profile, such as connecting their account with Facebook, changing their general data, etc. (see Figure 8).

The administrators are also normal users of the system; they may switch between user mode and admin mode. If they are in admin mode, by selecting the *Users* menu item, they can block and reactivate a user account, sending them an auto-generated e-mail, containing a private message from the administrator. Under the *Edit* menu item, they can manage the professions list (see Figure 6), with the deletion of a profession only available if not assigned to any user. On this page, they can list the invitation requests and accept, decline or delete them.

VII. Conclusion and future work

In the beginning of the development phase, the team has planned to create a prototype of an application that can follow and dynamically visualize the dispersion of people originating from a common region. That goal is successfully achieved: the E-migrated web application was built, which is a software system rich in functionalities, offering a wide range of possibilities over showing different dispersion statistics. E-migrated also serves as a professional collaboration platform, creating a community of people with similar origins.

From discussions formed at several presentations of the application, many new ideas and further development possibilities have arisen. These include: integrating other social networks, like Google+ and LinkedIn to the application; setting some posts as public in order to make them available also for guests; uploading and sharing educational videos and documents; requesting professional assistance or consultation from other users.

Furthermore, supporting collaboration between users with the help of gamification tools is also planned. These consist of the introduction of a *Thank You* point system: users could thank each other for the professional assistance by awarding such Thank You points. Thereby the helper receives public acknowledgement, and the users with the most Thank You points could obtain other privileges or perks, such as more invitations.

References

[1] C. Dénes. Hol vagytok székely(földi)ek? – a teljes diaszpóra! [Online]. Available: http://csaladenes.egologo.ro/?p=773
[2] M. Fowler. (2015) PresentationDomainDataLayering. [Online]. Available: https://martinfowler.com/bliki/PresentationDomainDataLayering.html
[3] Spring Framework Reference Documentation. [Online]. Available: https://docs.spring.io/autorepo/docs/spring-framework/3.2.17.RELEASE/spring-framework-reference/pdf/spring-framework-reference.pdf
[4] F. Gutierrez, *Pro Spring Boot*. Apress, 2016.
[5] Dao support. [Online]. Available: https://docs.spring.io/spring/docs/4.2.x/spring-framework-reference/html/dao.html
[6] P. Mularien, *Spring Security 3*. Packt Publishing, 2010.
[7] Spring Social Reference. [Online]. Available: https://docs.spring.io/spring-social/docs/2.0.0.M4/reference/htmlsingle/
[8] J. Hoeller. (2003) JavaMail. [Online]. Available: https://docs.spring.io/spring/docs/current/javadoc-api/org/springframework/mail/javamail/JavaMailSender.html
[9] B. Green and S. Seshadri, *AngularJS*. O'Reilly Media, 2013.
[10] The official page of Bootstrap. [Online]. Available: https://getbootstrap.com/
[11] The official page of Angular UI-Router. [Online]. Available: https://ui-router.github.io/about/
[12] The official page of Angular Google Maps. [Online]. Available: http://angular-ui.github.io/angular-google-maps/#!/
[13] PascalPrecht. The official page of Angular Translate. [Online]. Available: https://angular-translate.github.io/docs/#/guide
[14] The official page of Docker Compose. [Online]. Available: https://docs.docker.com/compose/