UTALK

# Internet technologies - project

Team :

Amariei Marinel – Team Leader

Blejusca Oana

Ciobanu Denis

We chose to make our own version of a virtual social network to find out more about how these big companies succeeded and the work behind those beautiful html landing pages.

Preexisting such applications are already well known, such as Facebook, Twitter, Instagram. They excel through a big, scalable infrastructure that can sustain big loads of requests and accesses every second.

## Specifications and requirements

Our application is going to be based on:

-database server

-web services server

-web server

-web client(browser)

-mobile app

Main implemented features:

-Personal profile – Register, Create and Edit profile

-Friends list – Add, remove, search friends

-Messaging – Send and post messages

Hardware/software resources:

-Visual Studio Code, Sublime and Notepad++ – text editor

-IntelliJ/Eclipse – Java IDE

-ApacheTomcat 8.5 – server

-MySql – Database server

-Github – Project management

- Google Chrome – Browser

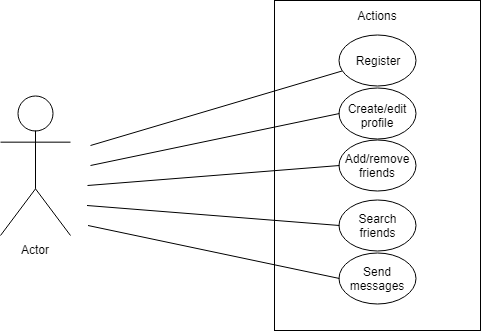
The main protagonist is our user. He has the possibilities to register or log in, in order to have access to his information.

Application components: User interface, Mobile application, Web server, Web services server, Database server.

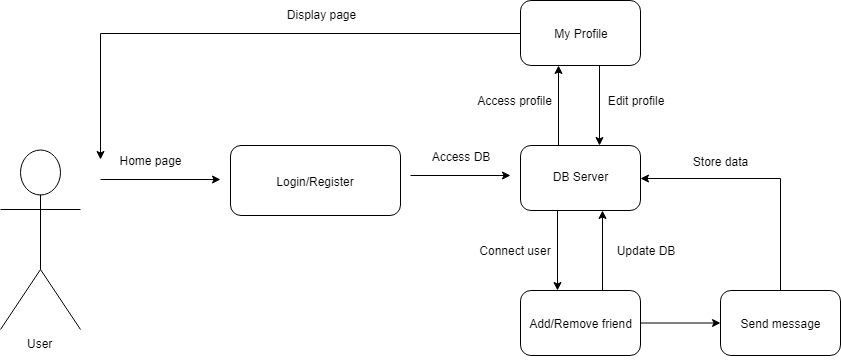
## UML DIAGRAMS

• Use-case diagram: This diagram offers a general description of the system way of usage, provides an overview of the features that the system offers and shows how it interact with the user.

The user has the possibility to create a profile, to publish posts/articles, he can visualize the friends list and add new friends in list. All the operations are available, depending of the each user rights, at the graphic interface level, through hyperlinks, buttons or forms.



• Information-Flow Diagram: Emphasizes the path the information covers from source to destination.



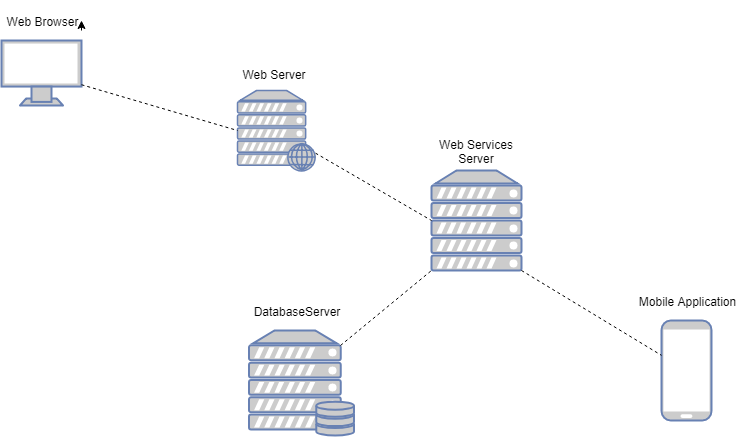
• Components Diagram: It shows the app components and the dependencies between them. A component is a class that represents a modular part of a system with encapsulated content. Utalk has as principal components a web services server, a web server and a database server. A separate component will be the mobile app. There are 5 interconnected components. This diagram has the role to present the physic structure of the code and its grouping per components using a mapping on the logic view of components.

A component can contain a source code or it can be in binary or executable form.

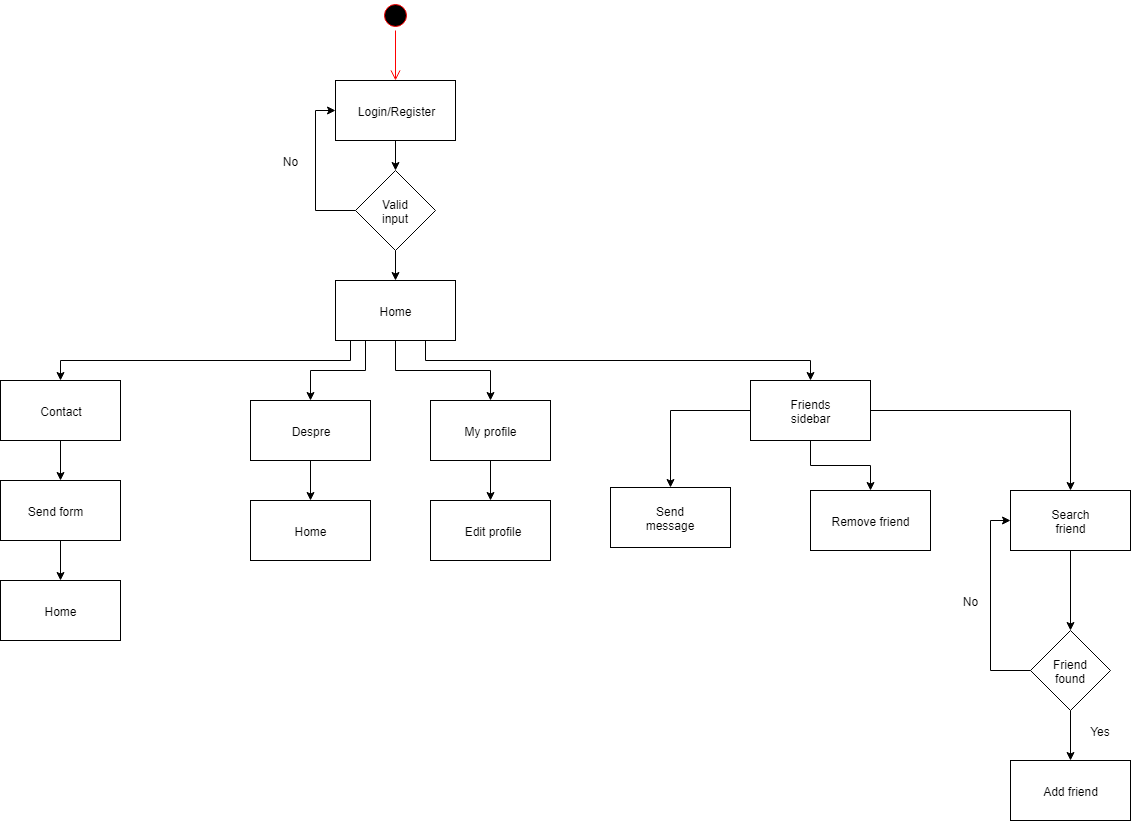
The diagram describe the path of the information, the relationships and dependencies between components.

The user use an interface given by the app or the web server, so he can log in.

Any action of the user is transmitted from the web services server that transmits further to the database server. There the information processing the answer is transmitted to the web services server what will transmit to the user messages that describe the action results.

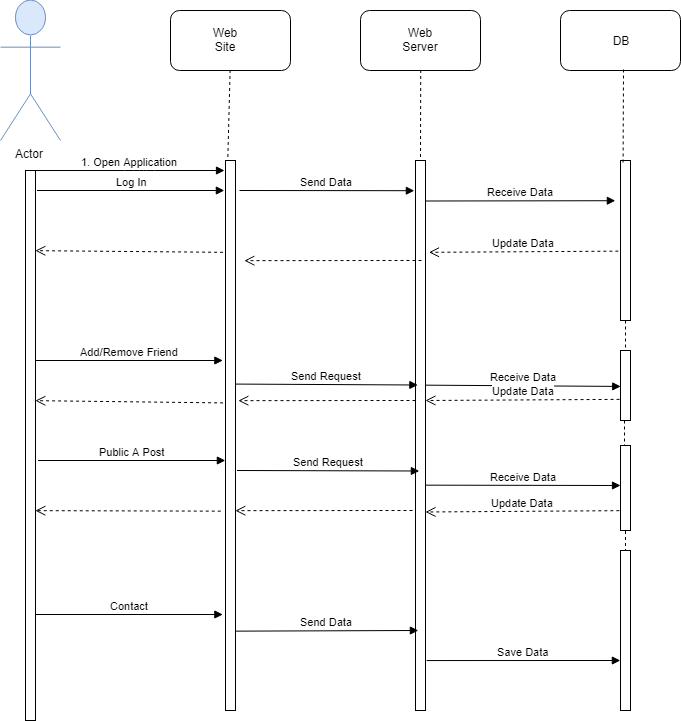


• Activity diagram: It presents the actions that the user can do on the site, together with the possible transactions and the conditioning between them.

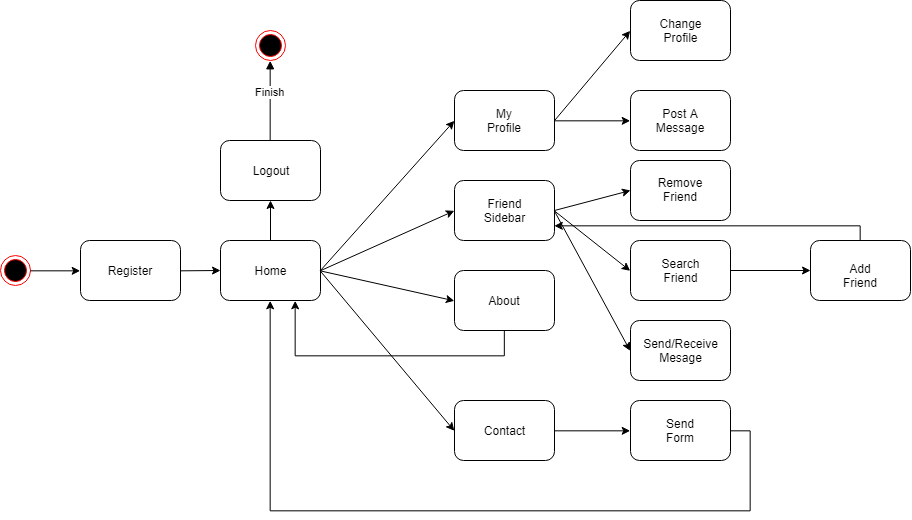


• Sequence diagram: It focus on the messages exchange between classes, components, subsystems or actors. The transmitted messages are organized chronologically from up to down, represented through arrows with continuous line, drawn from source to destination. Messages that represents responses are drawn using interrupted line.

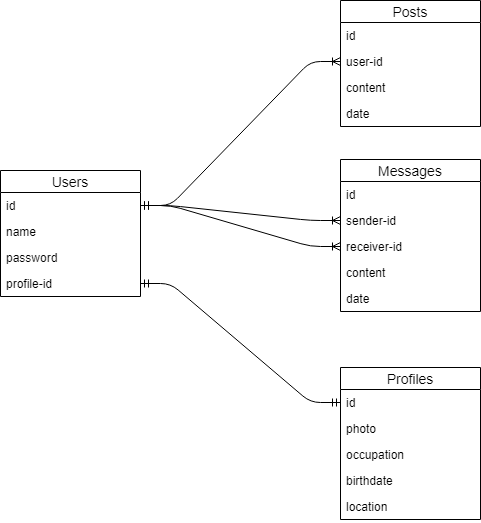
The entities that exchange messages are: user, app ( web or android ), the services server and the entire database.



• State machine diagram: It shows the states sequence that the system can take, from the user point of view. At the same time, the events caused by the user that lead to state change, the final state getting reached at logout.



• Entity-Relationship diagram: This diagram displays the layout of our database which consists of 4 tables: Users (with id, name, password and profile id), Profile(user-id – foreign key to users.id) one to one relationship with Users, Messages (sender-id, receiver-id – foreign key to users.id) many to one with Users, Posts(user-id – foreign key to users.id) many to one with Users.



## DATABASE

Connection with the database has been made using the file the module services. DatabaseConnection as the name implies produces the methods used for linking the application to the database. It contains the following methods: setUpBasicDataSource, getConnection, initDatabase, checkIfTableExists, createTable.

This module also makes the logic of this connection, implementing functions like : getById – as the name says it makes a request for an entity based on its id, getAll – get all the entities, create – for the creating profiles, update- for updating elements in tables, delete- for deleting an entity based on its id.

CRUD (Create, Read, Update, Delete) has been made inside repository module . We used CrudRepository interface in order to define the functions : create, getById, getAll, update, delete, deleteAll. Inside of this module we also have other functions who ease the work with the database. These can be found in module ProfileUtils.

Our database consists of 4 tables: Users (with id, name, password and profile id), Profile(user-id – foreign key to users.id) one to one relationship with Users, Messages (sender-id, receiver-id – foreign key to users.id) many to one with Users, Posts(user-id – foreign key to users.id) many to one with Users.

The database that we use is Oracle SQL. Oracle Database is a proprietary multi-model database management system produced and marketed by Oracle Corporation. It is a database commonly used for running online transaction processing, data warehousing and mixed database workloads

## Web Services

Using Spring Framework we defined functions inside of module controller. This module is used for allocating the resource profiles and it is also used for calling methods GET, POST, PUT, DELETE .

Controller use several methods like: getProfiles and getProfileById that used Get (annotation @GetMapping was used, it also produces a string in json format), createProfile that use Post (annotation @PostMapping was used, it also consumes a string in json format) , updateProfile that use Put (annotation @PutMapping was used, a string in json format), deleteProfile that use Delete (annotation @DeleteMapping was used).

The module services also contains class Application that extends SpringBootServletInitializer in order to realize the application in accordance with Spring Framework functionality.

These functions call methods belonging to profileServices field, procedure that follow the flow: controller – services – repository. This can be explained as it follows: the frontend makes a request, and this way functions are called from controller (this module allocate the resource profile and when it is needed it calls the functions within it ), then this module, through its methods, calls functions from services ( which realize the connection with the database and its logic ), the last step being for repository functions to be called in order to implement the CRUD logic.

Then all this road is followed from the end (repository) to the start (frontend).

## Web Server

For web server we used Angular.It is JavaScript Framework for building web applications and apps in JavaScript, HTML, and TypeScript, which is a superset of JavaScript. The code is written in TypeScript, which compiles to JavaScript and displays the same in the browser. We used this framework because of several advantages: it is easy to start the server you only use the ng serve command, it is fast and it is easy to work with.

We used Java Server Pages (JSP) for creating Contact web page. JSP is a technology for developing Webpages that supports dynamic content. This helps us insert java code in HTML pages by making use of special JSP tags, most of which start with <% and end with %>.

For formatting our frontend pages we used HTML5, CSS and also JavaScript in order to create functionality.

## FUTURE FUNCTIONALITIES

In the next stage we want to expand the application. We are going to create login logic on the site, Posts page, Message send/receive and deploying a desktop application or mobile application.

## THE LAST STAGE

**Difficulties:** connection and population with the database, framing multiple components into the page **Design patterns:** we use Singleton for connection with the database. Make the class of the single instance object responsible for creation, initialization, access, and enforcement. Declare the instance as a private static data member. Provide a public static member function that encapsulates all initialization code, and provides access to the instance.The client calls the accessor function (using the class name and scope resolution operator) whenever a reference to the single instance is required.

**The contribution of each member:**

* Blejusca Oana-Maria: posts component;
* Amariei Marinel: profiles and login compenents;
* Ciobanu Denis: friends component.

**Critical appreciation of members by the leader:**

* Blejusca Oana**:** Teamworker, with strengths in research and perseverance who has been much involved in the development of the project
* Ciobanu Denis**:** Knows how and when to impose his point of view, eager to do the hard work and contribute with new ideas.

**Critical appreciation of the leader by each team member**

* Blejusca Oana**:**
* Ciobanu Denis**:**

**Conclusions:**

From our point of view, it was a rather complex project to accomplish, from which we had to learn a lot of new stuff.