Technische Universität Dortmund

Development of a short-message sharing Web Application in Tapestry5

Arthur Xavier Gomes Ribeiro, Rafaella Chaves, Nathan Fabiano

July 20, 2015

1 Introduction

In this paper we develop a system used to broadcast short messages between the users. It is a social network inspired by famous websites and web applications which are currently in trend. The system developed within this paper is to be considered for academic-only purposes. It is therefore open source and its source code can be download from `http://github.com/arthur-xavier/croak`.

1.1 Terms used

In this report there are several terms with which the reader might not be familiarized. And to the fully understanding of the paper, a brief explanation of these terms is understandably needed.

User A user is an identified and specific person who uses the system described here and can perform actions within the system and alter its state.

CROAK A croak is a short message (maximal 140 characters) a user can write. These messages are the core of the system. In the main page of the system the user will be able to view many diferent croaks.

FOLLOWER A follower is a *User* who follows the user in question and can therefore see the croaks this user writes.

FRIEND A friend is a *User* who either is a *Follower* of the user in question or a *User* who the user in question follows.

1.2 PROBLEM DEFINITION

The system designed is the product of a set of actions and restrictions applied to the model. Below is a detailed description of these.

1. Croak a croak
   a) The user must be able to write short messages with content up to 140 characters.
   b) These messages must be stored within the database.
   c) The user's friends must be able to see his new message.
2. Following list
   a) The current user must be allowed to maintain a list of another users whose croaks it follows.
   b) The user must be able to manage this list, begin able to delete or add new friends at will.
3. Followers list
   a) Alongside the *Following list*, its counterpart, the *Followers list* is a list of another users who follow the current user's croaks.
   b) The user can't but manage this list, as it relies on the other users' behaviour.
4. Follow me
   a) The user must be able to invite other users to follow his croaks.

The definition of these parameters was done before anything else, and is the starting point of the development of the system.

2 Project specifications

This project is being development as part of the course **Webtechnologien II** for the Summer Semester 2015 at the Technische Universität Dortmund.

2.1 Environment

The environment and tools used in the development of the project were quite strict and could not be chosen freely. On our hands we had a toolset of Java Frameworks and libraries which, despite not much deep developed, documented or known, have had its own benefits and use within this project.

The whole set of technologies used are listed below:

- Java 8
- Maven 4
- Tapestry5
- JPA
- Hibernate
- HSQLDB
- JAX-RS
- Apache Shiro

2.2 Modelling

For the perfect functioning of the system were needed to be created just two Model classes: **User** and **Croak**. A *User* entity has for properties: username; password; first name; last name; email; avatar picture; and a quotephrase shown in his profile page. A *Croak* entity has only three properties: text; color; and a *User* author. A *Invitation* entity has but only two properties: the sender *User*; and the receiver *User*.

Using the conventions stated by Tapestry5, we have divided the system in 9 pages and 2 RESTful resources (one for each entity) used to handle AJAX requests. The pages used in the system are:

1. **Login** - when the user is not authenticated in the system, this page is shown, either so that it logs in or signs up into the system.

2. **Home** - the homepage, where a user can see a short preview of his profile and all the croaks its friends recently wrote.
3. **Friends** - this page contains both the *Following list* and the *Followers list* of the current user.
4. **Profile** - provides a form where the user can edit and update his profile.
5. **Search** - where the user can search for another users or croaks by typing keywords.
6. **ViewCroak** - when the user clicks on a croak it is redirected to this page, where it can see the croak alone and bigger.
7. **ViewUser** - when the user clicks on a user or a username it is redirected to this page, where it can see the profile and all the croaks written by this user.
8. **Logout** - simple action page where the user logs out of the system (no template).
9. **Error** - when an error or uncatched exception occurs in the system, this page is shown to the user.

3 Implementation

The implementation of the system was quite difficult and problematic, as there are not many resources and good documentation for the tools, libraries and frameworks used. The Tapestry5 framework is also not fully developed and thus creates new small difficulties for developers as it tries - most of the time successfully - to make it easier.

For the implementation, all the entities were firstly designed along with their respective RESTful Resources and Data Access Objects (DAOs) interfaces. Then the DAO interfaces were firstly implemented to store the entities within the application server, using Java Collections.

The Tapestry pages were then created, in order to test the current state of development.

The Hibernate library was then fully integrated in the system as well as the Hibernate implementation for each DAO interface which now store the entities in a HSQLDB database.

The Apache Shiro authentication and authorization library was then integrated with Tapestry and Hibernate, and each page and service fully secured.

The Login and Logout pages were finally created and the last bugs corrected.

The Invitation mechanism was implement after the system was already fully functional.

3.1 Details

The tapestry-routing library (by Tynamo) was used in order to provide better URLs for the system, as seen in the user profile URLs (e.g. `/@username`).

The system provides two DAO implementations for each model class: one using Java Maps and the other one using Hibernate. This was done in order to provide an easier way of implementing the system in parts - testing the system functions and specifications before integrating with Hibernate and a database. This also makes it possible to extend the system in the future, as it exposes generic DAO interfaces for the entities.

The frontend of the application was developed following the principles stated in Google's Material Design Guidelines (`http://www.google.com/design/spec`). This was done so that the application would suit better modern users, attract them with beautiful visuals and work well in mobile devices.

3.2 Problems

During the development we encountered some problems which might be worth reporting.

Tapestry5, showing one more time that, despite the version number, is not mature enough to compete with another vanguard Java Frameworks for web development, conflicts with some JavaScript frontend frameworks such as jQuery. This happens because Tapestry5 makes use of Prototype, another JavaScript frontend framework which uses the same global variable as jQuery. This problem could be easily solved by guarding this global variable within the Tapestry dependencies and JavaScript files in a protected scope.

The libraries developed by Tynamo (such as tynamo-routing and tynamo-security used in this project) must be correctly labeled as for compatibility with Tapestry5 versions, as they are not backward compatibles. We had a big problem figuring out that the latest version of tynamo-routing does not work with Tapestry 5.3.8, as it does not generate any errors or clues to the fact.

Another problem with which we had to deal during the development was the filtering system used by the tynamo-security library which implements Apache Shiro for Tapestry5. As the login page of the application contains two Tapestry forms (which generate two diferent response/action URLs), the tynamo-security filtering system (authc) was not able to deal well with forwarding requests after the login. We went around this problem by leaving the *authc* filter and using a simple *user* filter for protected pages and adding an automatic redirection function in the Error page when it shows authentication errors.

3.3 Known issues

The page for profile edition lacks validations which could be easily developed by adding Tapestry5 annotations over the properties for the form fields in page's server-side class.

As for the database, our passwords are stored in plain text, as the application is not meant to be used in Production environment. In order to facilitate the testing and development, the Sha-256 Hash which was used at first to store the passwords for the users was replaced by simple plain text storage. This change can but be reversed by uncommenting both lines at the following files: src/com/com/croak/dao/hibernate/CroakDAOHibernateImpl.java:44 and src/com/com/croak/security/UserRealm.java:24.

And last but not least, the system lacks and administrator user. That means it is still not possible to administrate the system (users and croaks) via the own application. The system administrator would need to make use of the REST API.

4 Results

The final results of the development process are to be shown in this section. Quite satisfactory, following all of the project specifications (but the *Follow me* feature), the application presents no errors or runtime exceptions when executing any function. It is known to be fully functional and working well also in mobile devices.

The following screenshots intend to show the application's final design and features.

4.1 Login page

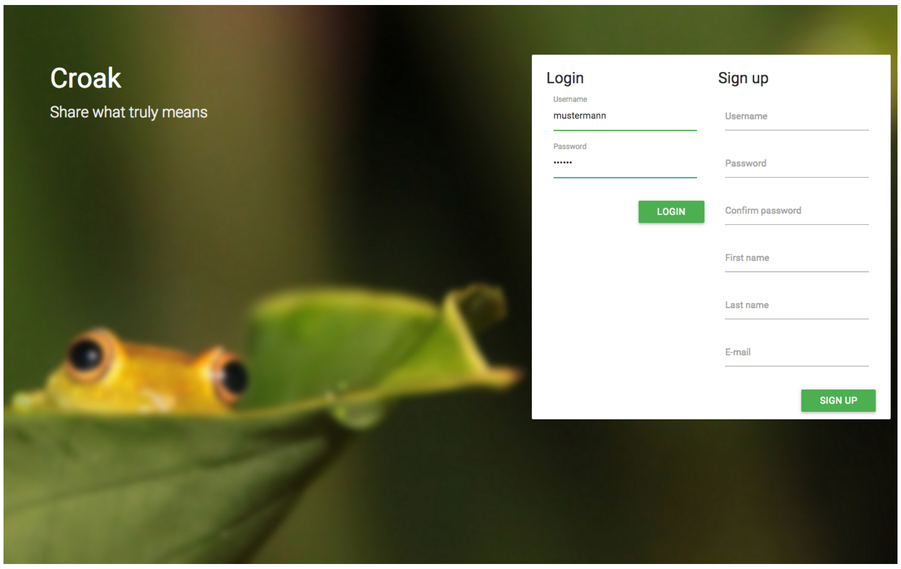
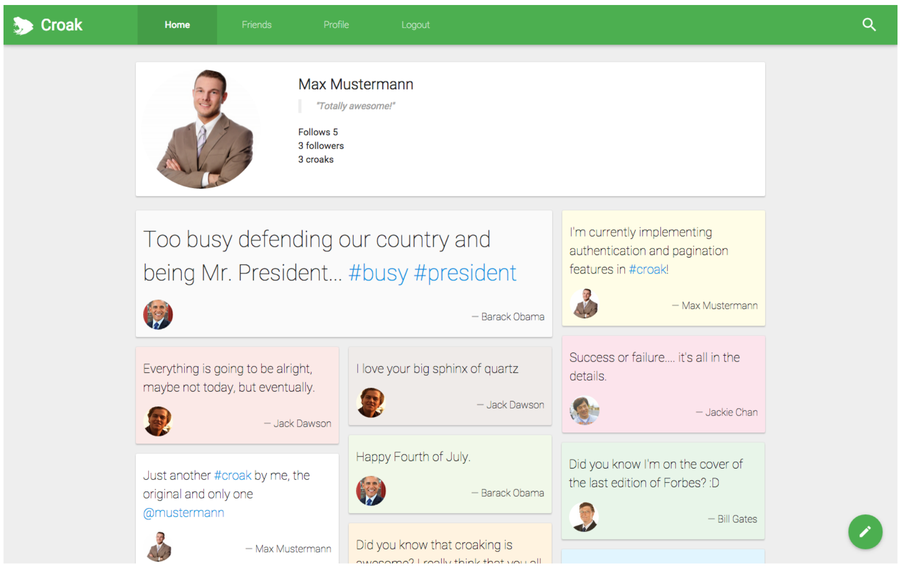


Figure 4.1: Screenshot of the login page

Containing two forms - one for login and the other for signing up -, this page was quite complicated in development for the difficulties in integrating Apache Shiro's filtering (authc) functions with the system URLs. As the login page itself contains two Tapestry5 forms, and is also target of redirection when the user tries to access any page of the system when not authenticated, the Apache Shiro framework does not integrate quite well with the Tapestry5 way of routing the forms' actions.

Tapestry5 validations were added to this page to prevent the users from registering with invalid information.

4.2 Home page



Max Mustermann

"Totally awesome!"

Follows 5
3 followers
3 croaks

Too busy defending our country and being Mr. President... #busy #president

— Barack Obama

I'm currently implementing authentication and pagination features in #croak!

— Max Mustermann

Everything is going to be alright, maybe not today, but eventually.

— Jack Dawson

I love your big sphinx of quartz

— Jack Dawson

Success or failure.... it's all in the details.

— Jackie Chan

Just another #croak by me, the original and only one @mustermann

— Max Mustermann

Happy Fourth of July.

— Barack Obama

Did you know I'm on the cover of the last edition of Forbes? :D

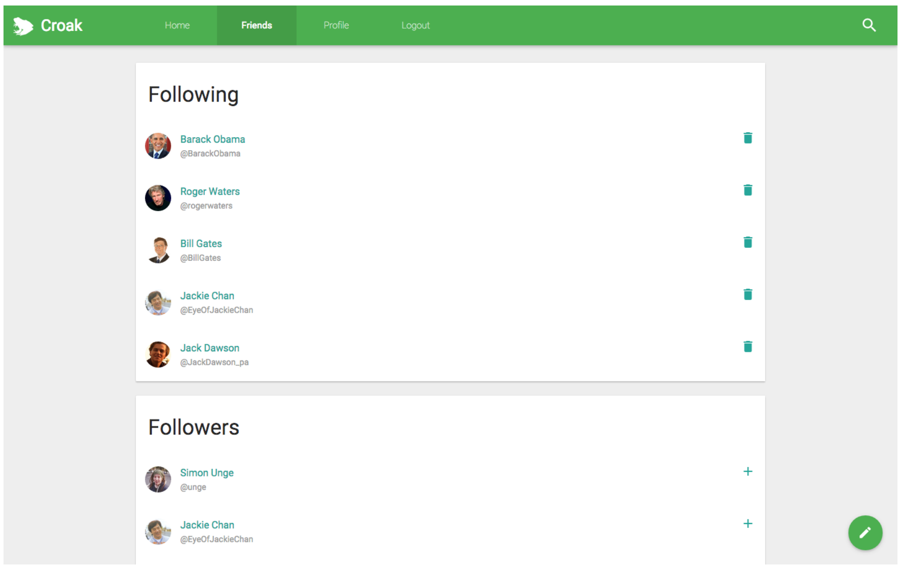
— Bill Gates

Did you know that croaking is

Figure 4.2: Screenshot of the home page

The home page is obviously the center of the system. Here the user has access to the latest croaks written by the ones he follows. He has also a short overview of his profile and some statistics about himself. By clicking at the button in the bottom right corner of the screen (the button is present in all the pages), the user is shown a form for writing a croak.

4.3 Friends page



Following

Barack Obama
@BarackObama

Roger Waters
@rogerwaters

Bill Gates
@BillGates

Jackie Chan
@EyeOfJackieChan

Jack Dawson
@JackDawson_pa

Followers

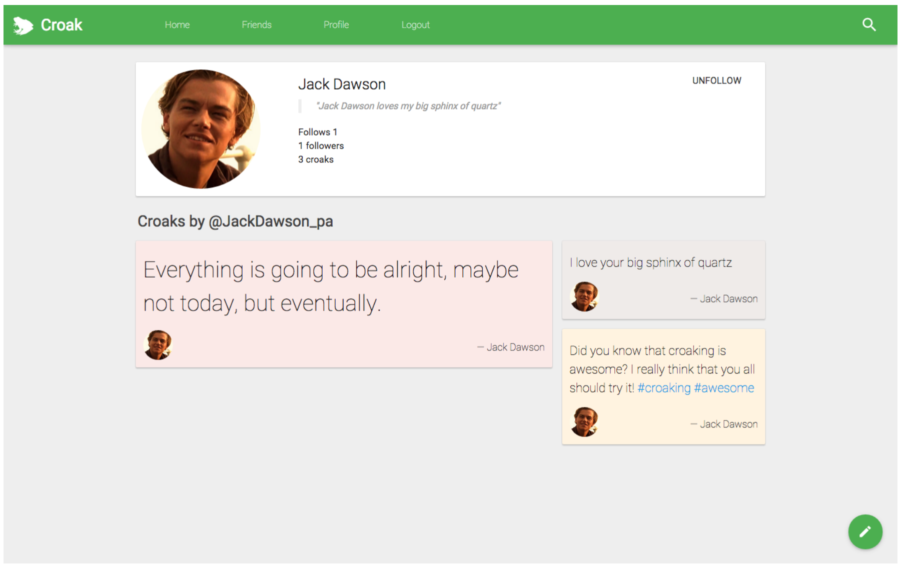
Simon Unge
@unge

Jackie Chan
@EyeOfJackieChan

Figure 4.3: Screenshot of the friends page

In the friends page, the user can view, select, add or remove friends from his *Following list* or *Followers list*. Shortcut buttons for adding or removing someone from the lists are provided in the right side of the list after each element, so that the user doesn't need to go to a friend's profile and click another button to follow or unfollow this person.

4.4 View user page



Jack Dawson

"Jack Dawson loves my big sphinx of quartz"

Follows 1
1 followers
3 croaks

Croaks by @JackDawson_pa

Everything is going to be alright, maybe not today, but eventually.

— Jack Dawson

I love your big sphinx of quartz

— Jack Dawson

Did you know that croaking is awesome? I really think that you all should try it! #croaking #awesome

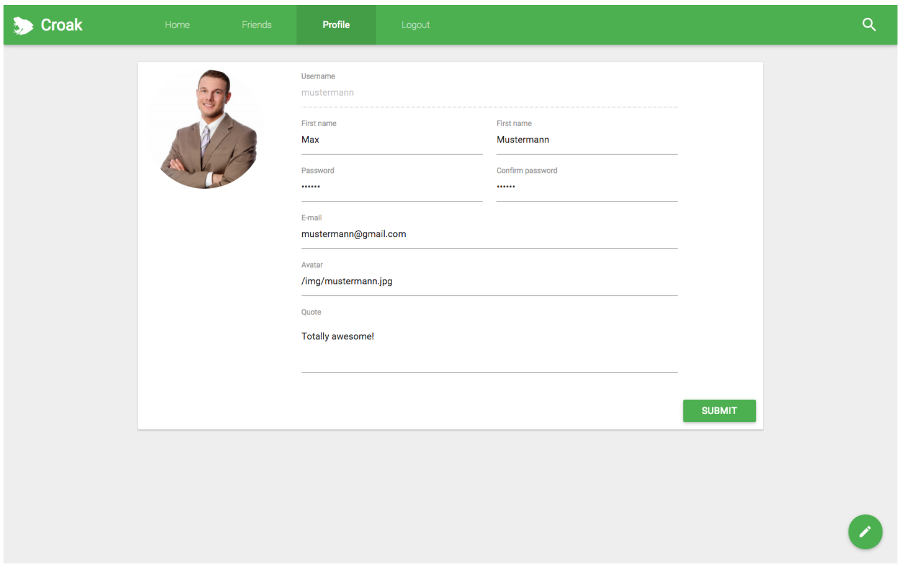
— Jack Dawson

Figure 4.4: Screenshot of the view user page

In this page the user can view informations about another user. Just as in the home page the user has access to the same informations displayed there but for another user. He can also click a button in the top right corner of the information card to follow or unfollow this another user.

Below the information card, the user can view all the croaks posted by the another user in question.

4.5 Profile page



Username	mustermann
First name	Max
First name	Mustermann
Password	••••••
Confirm password	••••••
E-mail	mustermann@gmail.com
Avatar	/img/mustermann.jpg
Quote	Totally awesome!

Figure 4.5: Screenshot of the profile page

Last page shown in this report, the profile page contains a form where the user can view and edit informations regarding his own profile.

5 Conclusion

Tapestry5, despite being still not fully developed, well-known nor well-documented, provides really what it promises: an easy-to-use environment Rapid Application Development. The choice for this framework in order to create a Java based simple/prototype Web Application is, because of that, remarkable.

Having more time for the development, guidance and organization (both in the development team itself and from the Tapestry5 developers), this project could have been fully developed up to the final stage of a refined, production ready industrial application.

Nevertheless the results from the project are quite exciting, showing that, although the team does not master the environment, a well developed application came out of it.

5.1 Future work

A better password storage algorithm is also a possibility for future work in this application. It is much more a feature missing in the system than a future work idea, because when it comes to security guarantees, this is a must in any modern web service.

User authorizations and roles, such as a super user (admin user) would be likely to be implemented in future versions, as this provides an easier way of managing the system.

Still in the security topic, prevention against SQL Injection can easily be implemented in future versions by making use of some well developed Java libraries for security. Using these libraries it would also be easy to add further security improvements, such as captchas in the Login page.

Another feature which would come right in hand of the users is the ability to upload your own avatar pictures instead of referencing and external image by its URL in the profile. This was not done here in order to simplify the development, as it is considerably complicated to implement uploads and server-side storage in Tapestry5.

And a last feature thought for implementation in future versions is a simple AJAX-based pagination for the viewing of croaks, both in the Home page and in the View User page. This is not only a nice feature, but would also be needed as the system grows due to a high count of users and messages.