Multimedia Report Forum Application

Multimedia modelleren en programmeren

Joris Schelfaut

Katholieke Universiteit Leuven

Contents

Chapter 1

Introduction

1.1 About this report

The course *Multimedia: modelleren & pogrammeren* is given by prof. E. Duval at KULeuven. The objective of this course to create a mobile application using different technologies, namely `iOS`, `Android` and `HTML5`, to eventually gain insight in the advantages and disadvantages of each technology.

The technologies that will be discussed on this blog are `iOS` and `Android`. All code for this project is open source and can be found on and downloaded from Github[1]. The progress of the development is communicated through a blog.

1.2 About the application

The application is an online forum where the main question of each tread is directed at one expert or a group of experts. Each member can enlist him/herself as an expert in certain areas and will get notified when a question is posted, related to his/her domain of expertise.

Although similar websites already exist, e.g. *Stackoverflow*[2] and *Yahoo! Answers*[3], one idea would be to direct the application rather at students than at the general public.

[1] `https://github.com/mumedev`
[2] `http://stackoverflow.com/`
[3] `http://answers.yahoo.com/`

Chapter 2

Requirement analysis

2.1 User story and storyboard

The use of the application can be illustrated by the following user story. Figure 2.1 depicts the same story line.

Jake is a student at the department of computer science at KULeuven. For his project of Multimedia he has to develop an Android application. Unfortunately he has encountered a particular problem as he was working on the application. He decides to look for some help.

He starts the application and enters his question. He also attaches a number of specialities as tags to his post. Users that are listed as specialists in these technologies are then notified and can start working on a solution.

Several solutions are proposed. Jake finds one that works and is able to continue with his work.

2.2 Use case diagram

The use case diagram for the application is shown in figure 2.2. Each use case is elaborated in tables 2.1, 2.2 and 2.3.

2.3 Screen transition diagram

Figure 2.1: Story board depicting a potential scenario of use.

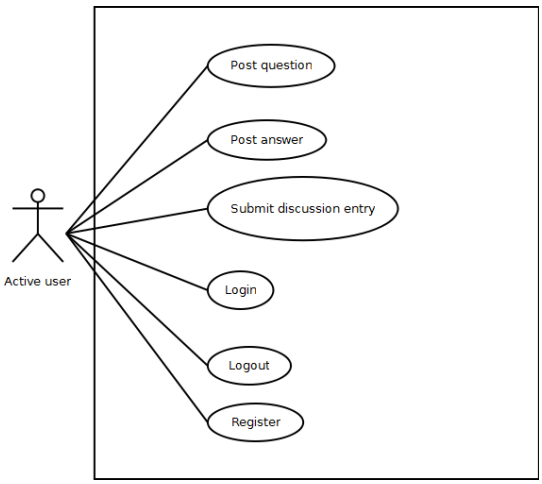

Figure 2.2: The use case diagram showing the functionality in the system from a user's perspective.

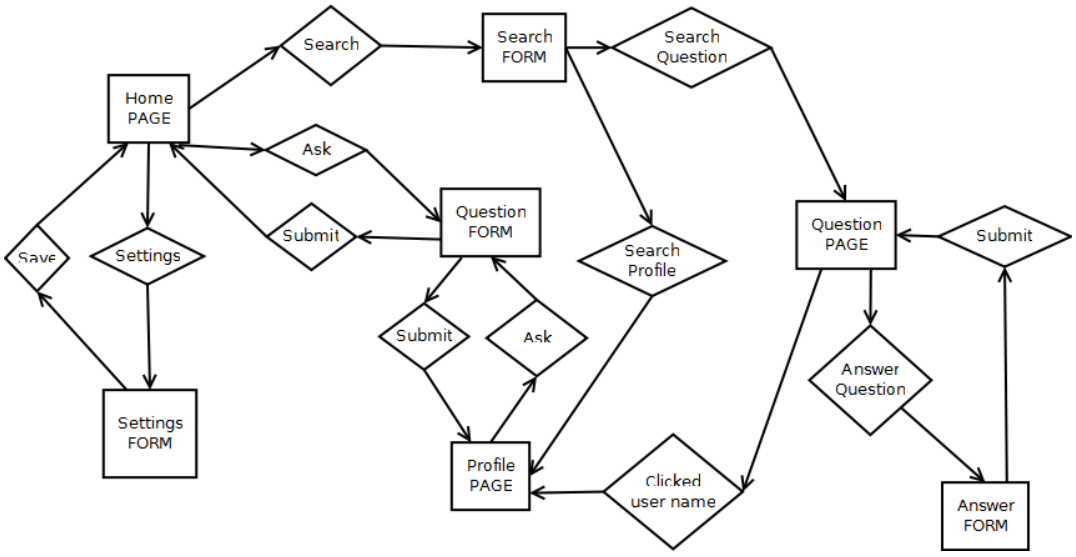

Figure 2.3: The transitions between the screens.

Table 2.1: Use case 1 *Post question*

Primary actor:	Active user
Preconditions:	User is logged in;
Basic flow:	(1) The question form is loaded.
	(2) The user enters his question.
	(3) The user selects recipients, including specific people and groups.
	(4) The user submits the form data.
	(5) The systems shows a confirmation message.

Table 2.2: Use case 2 *Post answer*

Primary actor:	Active user
Preconditions:	User is logged in;
	The user has selected a question to answer.
	User is allowed to participate in the discussion.
Basic flow:	(1) The user selects an option to answer the question.
	(2) The user enters his/her answer.
	(3) The user submits the answer.
	(4) The systems shows a confirmation message.

Table 2.3: Use case 3 *Take part in discussion.*

Primary actor:	Active user
Preconditions:	The user is logged in;
	The user has found a question he/she would like to provide further comments on;
	The user is allowed to participate in the discussion.
Basic flow:	(1) The user selects an option to create a discussion entry.
	(2) The user types in the discussion entry.
	(3) The user submits the entry.
	(4) The entry is added to the discussion.

Chapter 3

Architecture

3.1 Data model

Figure 3.1 shows the entity relationship diagram (ERD) of the entities in the database.

3.2 REST API

To allow access to the database, each application can talk to a public interface. The REST API consists out of a series of methods, listed in table 3.1. The parameter names of the returned data are based on the database table and column names.

Table 3.1: REST API

Resource	**Method**	**Description**
`USER`		
	get_info	Retrieve profile information about a user.
	update_info	Update profile information about a user.
	get_questions	Get questions submitted by a user.
	register	Register a new user account.
	delete	Delete a user account.
	search	Search a user given certain parameters.
	get_skills	Get the skills of a user.
	get_answers	Get the answers submitted by a user.
`SKILL`		
	search	Search skills.
	create	Create a new skill and add it to the database.
	get_questions	Get questions related to a given skill.
`QUESTION`		
	search	Search questions, given a number of search parameters.
	get_answers	Get the answers for a question.
	get_skills	Get the skills related to this question.
	create	Post a new question.
	get_info	Get information on a question.
	delete	Delete a question.
`ANSWER`		
	search	Search an answer.
	get_question	Get the question this answer was posted on.
	create	Create a new answer.

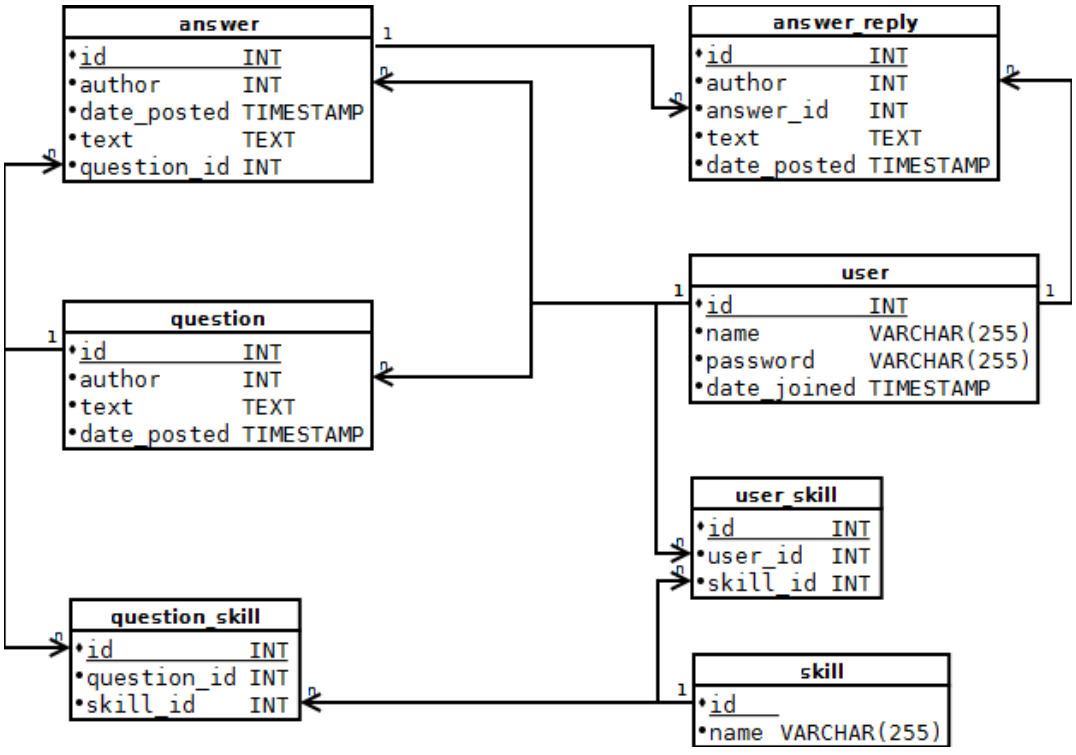

Figure 3.1: Entity relationship diagram of the database.

Chapter 4

Comparison

4.1 Web versus native mobile applications

In [1] a table is presented of the required skill set to be able to write mobile applications for each platform. From the nine mobile technologies that were included, the market shares for Android and iOS operating systems are 68.3% and 18.8% respectively, clearly dominating the market [2]. Numbers suggest that the Windows Phone will establish itself as another giant on the market, however still considerably less significant than Android or iOS. Even though this suggests development of native applications can be reduced to development for two to three technologies, the cost of developing and maintaining applications in these technologies would still create a considerable overhead. Charland et al. [1] look at mobile web technology to address this issue of heterogeneity.

4.1.1 Phonegap hack

The 'Phonegap hack' has grown from the fact that all mobile operating systems are equipped with a mobile browser, in which the native API can be called by using JavaScript. Unfortunately there are differences between the Webkit implementations of these browsers. In recent years many of these issues are being addressed through various libraries, e.g. jQuery Mobile. Also, the W3C has a device API working group that is working to bridge the gap between lower level native APIs and web technology [1]. JavaScript virtual machine technology is quickly getting more powerful, driven by competition between browsers. In the end, in [1] is stated that 'if you want to add a native capability to a browser, then you can either bridge it or recompile the browser to achieve that capability'.

> If a browser does not support a native capability, it's not because it can't or that it won't; it just means it hasn't been done yet (Charland et al.).

4.1.2 Trade-offs

A brief list of trade-offs between web and native mobile applications: Size of the code base - the code base for a mobile web application will be significantly smaller than the code base for two or more native applications; Efficiency of maintenance - the cost of maintenance may be directly related to the number of versions of an application that need to be updated; Application performance - native applications still outperform mobile web applications; Development time; Application objectives - related to application performance, as nice looking visuals do not fulfill business requirements, but may require significantly more processing power; GUI guidelines: creating only one application for different platforms, may require a lot of conditional statements to provide user interface code that is conform with the guidelines for each seperate platform;

Chapter 5

Conclusion

Bibliography

[1] A. Charland and B. Leroux. Mobile application development: web vs. native. *Commun. ACM*, 54(5):49–53, May 2011.

[2] D. Graziano. Mobile market share 2012: Android continues its success, ios follows. URL: `http://bgr.com/2012/12/04/mobile-market-share-2012-android/`, 2012. Online; Accessed January 22, 2013.

List of Figures

List of Tables

Appendix A

Codeigniter and REST API Setup

Appendix B

API Usage

Appendix C

Statistics