WaaS - Documentation

WebE - FFHS

Nicola Pfister & Jonas Meise

February 10, 2019

Table of Contents

1 Introduction

This is the documentation of the project WaaS, which was done as part of the module Web Engineering at the FFHS. The goal of the project is to create a concept and implement a web application which fullfills at least the following criteria:

- Basic user authentication (register, login/logout, delete and update user)
- Public and dynamic/user specific content
- Persist user data (in database or file)
- Basic validation and error handling
- Support for sessions and cookies

2 Requirements Engineering

This chapter contains purpose and context of the Webapplication WaaS aswell as the functional and non-functional requirements.

- **Name of the application:** WaaS
- **Purpose:** WaaS is a webapplication that allows it's users to crawl urls with specified search patterns and notifies them as it finds the pattern.
- **Names:** Nicola Pfister, Jonas Meise

2.1 Purpose & Context

WaaS (Webcrawler as a Service) is a service, that allows users to keep track of news on their favourite websites, by giving them the possibility to create "crawls". A crawl is defined with a URL, a search pattern and an email address. WaaS will regularly check those URLs for the given search patterns and notifies the users via their email address when it finds the pattern it was searching for.

TODO: Context

2.2 Functional Requirements

The following chapter contains the functional requirements for WaaS.

2.2.1 Target Group

TODO

2.2.2 Use Cases

UST-1 Register	
Goal	A user registers a useraccount for WaaS.
Actors	Eventual user of WaaS
Precondition	The user owns a valid email address that is not yet used for an existing useraccount
Trigger	Click on the button: "Register"
Main path	1 The user clicks the button: "SignUp". 2 The user enters his credentials into the register form (email address, and password). 3 The user clicks on: "Register". 4 WaaS creates a useraccount entry in the database. 5 The user gets redirected to the Login Page.
Alternative path	1a The user enters a invalid input data. 2a A validation error message shows up.
Postcondition	A new useraccount was created in the database with the credentials the user entered into the register form.

Table 1: UST-1

UST-2 Login	
Goal	A user logs in with an existing user account.
Actors	registered users
Precondition	UST-1
Trigger	Click on the button: "LogIn"
Main path	1 The user enters a valid email address an password into the login form. 2 The user clicks on the button: "LogIn".
Alternative path	1a The user enters a invalid input data. 2a A validation error message shows up.
Postcondition	The user is now logged in and is situated on the overview page

Table 2: UST-2

UST-3 **Create Crawl**	
Goal	A user creates a new crawl.
Actors	logged in users
Precondition	UST-1 & UST-2
Trigger	Click on the button: "+"
Main path	1 The user enters a URL and a search pattern into the New Crawl form. 2 The user clicks on the button: "Save".
Alternative path	1a The user clicks the button "Cancel". 2a The user redirected back to the overview page.
Postcondition	The newly created crawl is persisted in the database & The new crawl gets displayed on the users overview page.

Table 3: UST-3

UST-4 **Delete Crawl**	
Goal	A user deletes an existing crawl.
Actors	logged in user
Precondition	UST-3
Trigger	Click on the delete crawl button.
Main path	1 The user clicks on the delete button of the crawl he wishes to remove. 2 A confirmation dialogue is shown. 3 The user clicks the "Delete" button. 4 The crawl gets deleted from the database.
Alternative path	3a The user clicks the button "Cancel". 4a The confirmation dialogue closes.
Postcondition	The crawl is deleted from the database & The crawl does not get displayed on the users overview page anymore.

Table 4: UST-4

UST-5 **Edit Crawl**	
Goal	A user edits an existing crawl.
Actors	logged in user
Precondition	UST-3
Trigger	Click on the edit crawl button.
Main path	1 The user clicks on the edit button of the crawl he wishes to edit. 2 The Edit Crawl form is shown. 3 The user changes the Crawls URL. 4 The user clicks "Save". 5 The changes get persisted in the database.
Alternative path	4a The user clicks the button "Cancel". 5a The Edit Crawl form closes.
Postcondition	The updated crawl gets displayed on the users overview page.

Table 5: UST-5

2.3 Non-functional Requirements

1. **Performance**
   (a) For user interactions, the application should respond 99% of the requests in 2 seconds.
   (b) The application should score at least 40 points of performance in Google Chromes inbuilt Lighthouse audit.
2. **Usability**
   (a) If the application experiences any kind of error or delay, users should be made aware of this.
   (b) When a website that is subject to a Crawl changes to contain the looked for pattern, users should receive a notification within an hour.

3. **Other non-functional requirements**

   (a) The application should be implemented according to best practices and should therefore score 100 points in the best practices section of the Lighthouse audit.

   (b) Due to the use of relational database technology, horizontal scalability is too big of a task to be realized in the given time frame and is therefore outside of the scope of this project.

List of Tables

References