WaaS - Documentation

WebE - FFHS

Nicola Pfister & Jonas Meise

February 22, 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 fulfills 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

The technologies for the project were not predefined, so we decided to use .NET Core for the web application and AngularJS for the front end due to personal preferences.

2 Requirements Engineering

This chapter contains purpose and context of the web application WaaS as well as the functional and non-functional requirements.

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

2.1 Purpose & Context

WaaS (Web Scraper 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 "Scrapes". A Scrape 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.

The following graphic visualizes the context of the app with it's use cases.

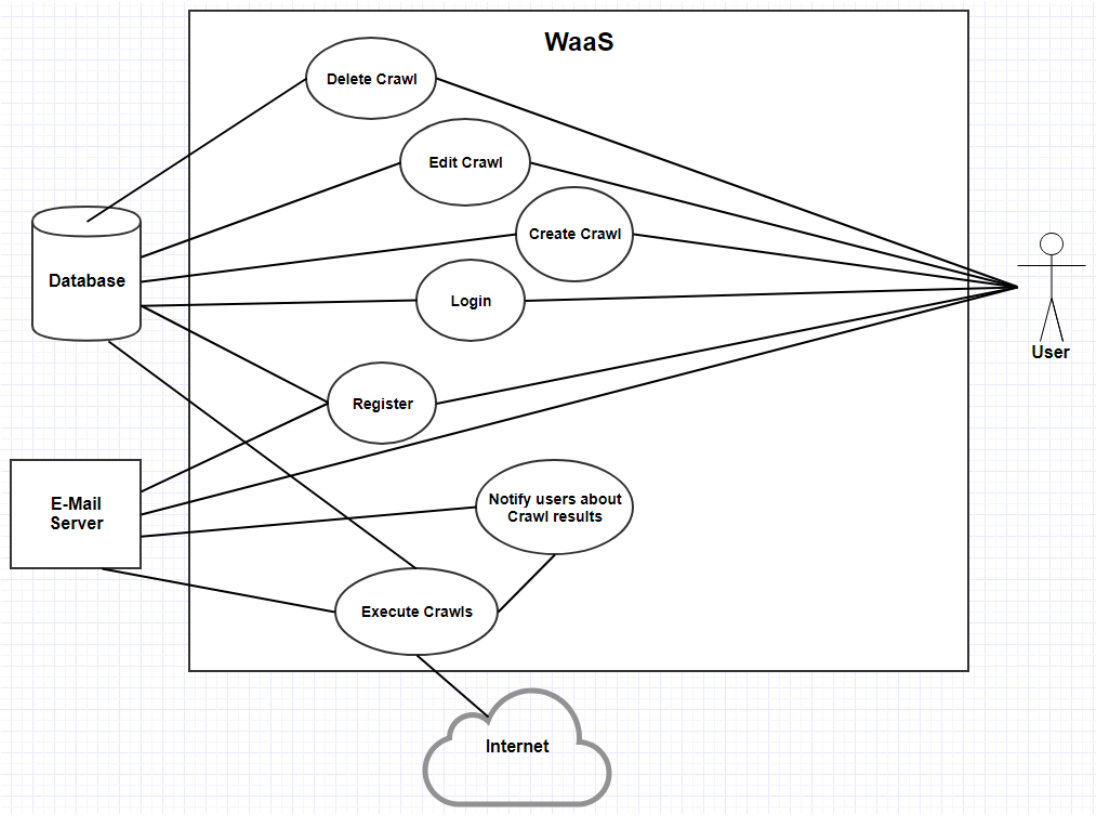

Figure 1: Use Case Diagram

2.2 Functional Requirements

The following chapter contains the functional requirements for WaaS.

2.2.1 Target Group

The target group of WaaS are mainly people that are regularly checking websites for news. That concludes people of all ages who understand what a web scraper is. It can also be used from people who want to check wether or not the new episode of their favourite tv series is already online. All of those people are generally expected to have some basic skills in the field of IT or are at least interested in it.

2.2.2 Use Cases

UST-1	**Register**
Goal	A user registers a user account for WaaS.
Actors	Eventual user of WaaS
Precondition	The user owns a valid email address that is not yet used for an existing user account
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 user account 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 user account 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 Scrape**
Goal	A user creates a new scrape.
Actors	logged in users
Precondition	UST-1 & UST-2
Trigger	Click on the button: "+"
Main path	1 The user enters a URL into the New Scrape form. 2 The form shows if the entered URL is valid. 3 The user enters a Scrape pattern into the New Scrape form. 4 The user clicks on the button: "Save".
Alternative path	1a The user clicks the button "Cancel".
Postcondition	The newly created scrape is persisted in the database & The new scrape gets displayed on the users overview page.

Table 3: UST-3

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

Table 4: UST-4

UST-5	**Edit Scrape**
Goal	A user edits an existing scrape.
Actors	logged in user
Precondition	UST-3
Trigger	Click on the edit scrape button.
Main path	1 The user clicks on the edit button of the Scrape he wishes to edit. 2 The Edit Scrape form is shown. 3 The user changes the Scrape's URL. 4 The form shows if the entered URL is valid. 5 The user clicks "Save". 6 The changes get persisted in the database.
Alternative path	2a The user clicks the button "Cancel". 3a The Edit Scrape form closes.
Postcondition	The updated Scrape gets displayed on the overview page.

Table 5: UST-5

UST-6	**Receive and Dismiss Notification**
Goal	A user gets notified about a Scrape having been triggered.
Actors	user
Precondition	UST-3
Trigger	WaaS found defined Scrape pattern on Scrape URL.
Main path	1 The user receives an E-Mail notifying him about the triggered Scrape. 2 The user clicks on the link in the E-Mail. 3 The notification gets dismissed by the system. 4 The user gets redirected to the URL of the Scrape.
Alternate path: Notification Tray	1a The user logs in to WaaS. 1b The user opens the notification tray. 2a The user clicks on the notification in the tray
Alternate path: Notification Tray Dismiss All	1a The user logs in to WaaS. 1b The user opens the notification tray. 2a The user dismisses all notifications. 3a All previously unread notifications get marked read.
Postcondition	The user has been notified about his scrape being triggered.

Table 6: UST-6

UST-7	**Show Scrape History**
Goal	A user can see all past triggers of a Scrape.
Actors	logged in user
Precondition	UST-3
Trigger	User opens Scrape details.
Main path	1 The opens the Scrape details. 2 Details include an overview of past triggers for the Scrape.
Postcondition	The user has gotten information about his Scrape.

Table 7: UST-7

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 Scrape 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

List of Figures

References