|  |  |
| --- | --- |
|  |  |
| **M151 Doku**  **ShortURL** |  |
|  |  |
|  | *20.6.2022*  *M151* |
|  | *Ramon, Sven M.* |

Contents

[Einleitung 3](#_Toc106522958)

[Zieldefinition 4](#_Toc106522959)

[Meilensteine 5](#_Toc106522960)

[Benutzung von Frameworks 6](#_Toc106522961)

[Arbeitsjournal 7](#_Toc106522962)

[Vermeiden von Angriffen 8](#_Toc106522963)

## Einleitung

Wir haben im Modul 151 den Auftrag bekommen, eine Webapplikation zu realisieren. Diese sollte mindestens eine 4-Tier Architektur besitzen, eine verschlüsselte Verbindung besitzen und speziell auf Datensicherheit achten. Dieser Auftrag wird in einer Gruppe erledigt.

Nachdem wir uns organisiert hatten, besprachen wir, was für eine Webseite wir machen sollten. Wir hatten zuerst eher weniger Ideen, aber schlussendlich einigten wir uns auf einen URL-Shortener Service. Das ist eine Website, mit der man lange URLs (z.B. <https://dies-ist-eine-sehr-lange-domain.com/noch-mehr-text-hier>) kürzen kann, so dass sie nur noch ein Bruchteil ihrer Originallänge waren. So eine gekürzte URL würde etwa so aussehen: <https://short.cc/AHftl9>

## Zieldefinition

Damit wir uns immer schön orientieren können während dem Entwickeln setzen wir uns vor dem Beginn ein paar Ziele, welche wir mit unserer Website erreichen möchten. Somit wissen wir immer klar, woran noch gearbeitet werden muss.

#### Ziel 1

Die Webseite soll eingegebene URLs kürzen und dem User die gekürzte URL zurückgeben.

#### Ziel 2

Es sollte keine Short-URL zwei Mal existieren.

#### Ziel 3

Es darf keine SQL-Injection, XSS oder Request-Forgery möglich sein.

## Meilensteine

Anhand der Meilensteine können wir unseren Fortschritt messen. Da wir in Etappen arbeiten, wäre es gut, wenn wir jede Woche einen Meilenstein erreichen können. Meilenstein 1 steht hierbei für die 1. Woche, Meilenstein 2 für die 2. und so weiter.

#### Meilenstein 1

Die Planung des Projektes ist abgeschlossen

#### Meilenstein 2

Das Frontend steht.

#### Meilenstein 3

Die Datenbank ist fertig gestellt und es sind Funktionen im Backend, die mit der DB interagieren.

#### Meilenstein 4

Das Backend ist fertig.

#### Meilenstein 5

Die Dokumentation ist abgeschlossen und somit ist das Projekt fertig.

## Architektur

Hier beschreiben wir kurz, wie unsere Software aufgebaut ist und weshalb wir die Technologien verwendeten, die wir schlussendlich brauchten.

#### Frontend

Fürs Frontend haben wir uns für Plain HTML, CSS und jQuery verwendet, da das am einfachsten ist für unsere Webapplikation. Ausserdem kennen wir uns mit den 3 Technologien am besten aus und müssen uns nicht noch auf andere Dinge wie Components fokussieren.

#### Backend

Beim Backend ist die Situation etwa gleich, wir entschieden uns für PHP, da es einfach zum Hosten ist und wir es schon kannten. Ausserdem lässt es sich einfach integrieren.

#### Datenbank

Als Datenbank nahmen wir MariaDB, da Ramon zuhause ein Raspberry Pi hat und dort MariaDB 24/7 zur Verfügung steht.

#### Sicherheit

Da wir die Daten der User schätzen, hat unsere Webseite ein SSL Zertifikat, so das jeder Traffic verschlüsselt wird. Ausserdem haben wir dafür gesorgt, dass die Daten nicht aus der DB durch SQL-Injection entwendet werden können. Mehr dazu unter dem Kapitel *Vermeiden von Angriffen*.

## Arbeitsjournal

#### Woche

##### Ramon & Sven

In der ersten Woche haben wir zusammen das Projekt definiert. Wir einigten uns auf einen URL-Shortener, da es einfach zum Entwickeln ist, man aber dennoch auf gewisse Sicherheitsmassnahmen achten muss. Ausserdem haben wir in der 1. Woche noch die Grobplanung gemacht, also festgelegt welche Meilensteine wir in jeder Woche erreichen wollen.

#### Woche

##### Ramon

Das Frontend samt design habe ich in dieser Woche gemacht. Allerdings noch ohne JavaScript, das kommt später dazu, denn dafür muss zuerst das Backend stehen.

##### Sven

In dieser Woche habe ich den Projektbeschrieb geschrieben und das Grundgestell der Dokumentation realisiert

#### Woche

##### Ramon

In dieser Woche habe ich die Datenbank entworfen und realisiert. Die gekürzten URLs sollten schliesslich gespeichert werden. Ausserdem habe ich noch ein hCaptcha hinzugefügt auf dem Frontend, da so eine Seite bestimmt Bots anlocken wird und man sich mit einem Captcha, welches zum Teil nicht mal Menschen können, einfach schützen kann.

##### Sven

In dieser Woche habe ich einen grossen Teil der Doku, wie Journal, Frameworks, usw., geschrieben und genauer spezifiert. Zudem habe ich noch ein wenig CSS für die Seite geschrieben.

#### Woche

##### R**amon & Sven**

Es war schulfrei, dementsprechend wurde an diesem Tag eher weniger erledigt. Das war auch so eingeplant und sollte im Zeitplan aufgehen, da wir sehr gut mit dabei sind.

#### Woche

##### Ramon

In der 5. Woche habe ich das Backend fast fertig gemacht. Es steht noch nicht ganz, aber das sollte nächstes mal fertig werden. Ausserdem habe ich das JS beim Frontend hinzugefügt, mit welchem man über jQuery und Ajax mit dem Backend interagiert.

##### Sven

Fertigstellung des Arbeitsjournals und grammatikalische Korrektur der Doku. Des Weiteren habe ich noch ein paar Bugs im CSS behoben.

## Vermeiden von Angriffen

Eine Website ist ein beliebtes Ziel von Hackern und anderweitigen Angreiffern, daher muss der Entwickler sich darum kümmern, dass die Sicherheit und Integrität der Systeme und Besucher Daten gewahrt bleibt. Unter diesem Kapitel beschreiben wir kurz, was für Sicherheitsmassnahmen wir getroffen haben, um typische Angriffe zu verhindern.

#### SQL-Injection

Jeder Input vom User, der in irgendeinem Weg Kontakt mit der Datenbank hat, muss «escaped» werden. Das bedeutet, dass Spezielle im Zeichen wie ‘ und ` modifiziert werden, damit sie nicht von der Datenbanksoftware als SQL-Syntax erkennt werden können.

#### XSS

Cross-Site Scripting, kurz XSS, ist eine Angreifbarkeit, mit der man auf dem Webserver JavaScript speichern kann. Wenn danach ein anderer User die Webseite aufruft, wird dieses JS Script ausgeführt und kann potenziell die Login Daten stehlen oder im Hintergrund Crypto Minen.   
Das kann man auch relativ einfach verhindern, in dem Jeder Userinput, der auf der Webseite geladen/gerendert werden könnte, durch die Funktion *htmlspecialchars(«input»)* geschickt wird.

#### Request-Forgery

Dieser Angriff beschreibt das Modifizieren der URL Parameter, so dass ein ungewolltes Ergebnis rauskommt. Natürlich ist klar, dass wenn man den Key nach der Domain (short.cc/**keyHere**) ändert, man zu einer anderen Webseite kommt. Daher kann man nicht wirklich etwas ändern, ausser beim Registrieren der URLs aber auch dort gibt es nur 3 POST Parameter, die gelesen werden. Einer ist die lange URL, der andere der Key und der dritte ist für das Captcha.

#### Cross-Origin-Authentication

Da Login Systeme häufig kompliziert werden können, haben wir uns dafür entschieden, das jemand anderem zu überlassen. In unserem Fall wäre das Google, da praktisch jeder ein Google Account hat. Auf unserer Website befindet sich also ein *Login with Google* Button, der den User zum externen Login führt. An unserem Ende wird dann der Login Token, der Google zurück gibt, validiert und der User wird angemeldet.

#### Cross-Site-Authentication

    \*Ramon\*

#### Cross-Site-Cooking

Wir haben authentifizieren unsere Cookies mit den Sessions und vermeiden so fremde Angriffe.

#### Cross-Site-Tracing

Wir speichern natürlich so wenig Benutzerdaten wie möglich, damit diese auch nicht missbraucht werden können. Wir haben trotzdem alle HTTP Trace disabled, damit niemand ein Trace verwenden kann und unsere Daten so sicher sind

#### Session-Hijacking

Wir verwenden Google Login, was den User authentifiziert und Session Hijacking verhindert.

#### Directory-Traversal

Auch hier Escapen wir alle Filepaths, damit nichts injected werden kann.

#### E-Mail-Injection

Wir haben keine E-Mail angaben bei unserer Webseite, weshalb auch keine E-Mail gestohlen werden kann

#### Man-in-the-Middle

Da können wir nicht viel machen, ausser die User sensibilisieren, damit sie keine Verdächtigen Links klicken und somit gar nicht in diese Situation.

#### Phishing

Wir haben keine E-Mails, weshalb wir unsere User relativ einfach darauf sensibilisieren können, da wir auch keine Mails versenden können.

#### Shell-Injection

Wir haben die Shell() Funktion nicht verwendet, deshalb kann da auch nichts passieren.

#### Login

Wir haben kein Login, deshalb ist es nicht notwendig.