Pflichtenheft Badger

SWP-OP | 5AHWII | 2020/21

Mario Petrovic
HTL ANICHSTRASSE | WIRTSCHAFTSINGENIEURSWESEN & BETRIEBSINFORMATIK

Inhalt

Inhalt

Mitwirkende

Name	Rolle	E-Mail
Sabo Rubner	Betreuung	s.koelloe@tsn.at
Mario Petrovic	Entwicklung	mpetrovic1@tsn.at

Projektbeschreibung

Badger ist eine web-basierende Plattform für Textkommunikation, welche von einem Schüler der Abschlussklasse der HTL Anichstrasse entwickelt wird. Als Inspiration dient hierbei die Chatting-App WhatsApp. Die Motivation für das Projekt ist die Sammlung von Erfahrung im Bereich Web-Development und Client-Server-Kommunikation.

Ziel

Schlussendlich soll eine Webseite entstehen, über die sich mehrere Nutzer mittels Accounts einloggen und miteinander kommunizieren können. Der Nutzer kann hierbei Kontakte zu anderen Nutzern einspeichern und aufrufen. Die Chatverläufe sowie die Nutzer sollen über eine Datenbank gespeichert werden und jederzeit aufrufbar sein. Während der Kommunikation soll es dem Nutzer möglich sein formatierte Texte und Bilder senden und empfangen zu können.

Web-Sockets

Web-Sockets dienen dazu, dass die Kommunikation zwischen Client und Server aus beiden Richtungen erfolgen kann. Normalerweise fragt bei HTTP nur der Client den Server nach Dateien. Dieser sendet diese dann zwar an den Client, jedoch geschieht dies nur auf Anfrage. Mittels Web-Sockets kann nun der Server selbstständig Daten ungefragt an einen Client senden.

Dies ist für dieses Projekt vor allem sinnvoll, da die Kommunikation zwischen zwei Usern über einen HTTP-Server läuft. Somit sendet ein User Daten an den Server, welcher diese dann ohne Anfrage gleich an einen anderen User sendet. In diesem Fall wird die Node.JS-Technologie Socket.io benutzt.

Socket.io besteht allgemein aus zwei Teilen:

- Ein Server, welcher das Node.JS-Modul socket.io integriert.
- Eine Client-Library, welches die Technologie auf die Browser-Seite ladet (in die HTML-Datei) namens socket.io-client. Diese Library ist im Node-Modul enthalten.

Ablauf des Projekts

Zu Beginn soll das Wichtigste erstellt werden: Die Grundfunktion. In diesem Fall wäre das die Kommunikation zwischen zwei Usern über einen Server. Somit wird sichergestellt, dass das Grundprinzip hinter dem Projekt funktioniert. Dazu wird ein Server benötigt, auf dem eine Webseite läuft, welcher deshalb als erstes erstellt wird.

Als nächstes soll die Datenbankeinbindung funktionieren und zeitgleich das Prinzip der User-Accounts erstellt werden.

Zuletzt wird endgültig die Webseite erstellt, welche die zuvor erstellten Funktionen einbindet.

Beim Ablauf wird nicht unbedingt strikt nach Plan entwickelt. In jedem Bereich wird abwechselnd weiterentwickelt, damit zu jedem Zeitpunkt ein Vorzeigemodell existiert, welches mit der Zeit verbessert wird.

Rahmenbedingungen

- Das Projekt soll im Zeitraum von September bis Jänner/Februar entwickelt und beim Betreuer eingereicht werden.

Technologien

In diesem Projekt wurden zahlreiche Technologien benutzt, welche zuvor von mir nie verwendet wurden. Dies liegt vor allem auch an der ursprünglichen Motivation an dem Projekt. Mir sollen möglichst viele neue Technologien, anhand ihrer Anwendung und welche Vorteile sie bieten, nähergebracht werden.

Es folgt nun eine Auflistung inklusive kurzer Erklärung der verwendeten Technologien.

Visual Studio Code

Da es sich bei diesem Projekt um eine Web-Anwendung handelt, wird eine entsprechende Entwicklungsumgebung benötigt. Hierbei war keine großartige Recherche notwendig, da mir im Laufe der HTL immer wieder positive Erfahrungen mit VSCode zu Ohren gekommen sind.

Mit VSCode ist die Programmierung von Web-Anwendungen sehr bequem. Neben dem Erstellen von Vorlagen anhand Shortcuts hat VSCode auch eine integrierte Syntax-Überprüfung, welche Programmfehler frühzeitig erkennt.

Als Alternative wäre noch Notepad++ in Frage gekommen, jedoch war ich nach dessen Verwendung in den bisherigen Jahren der HTL nicht sonderlich überzeugt davon.

NodeJS

Mithilfe von NodeJS kann ein Backend für den Server erstellt werden. Bei NodeJS handelt es sich um eine JavaScript basierte Runtime-Engine, welche außerhalb von Webbrowsern läuft. Somit können hierdurch JavaScript-Programme entwickelt werden, welche nur Daten verarbeiten und nicht unbedingt eine GUI benötigen (ideal für Server geeignet).

Zudem besitzt NodeJS eine Vielzahl von integrierten Libraries, sogenannten Node-Modules, welche den Umgang mit der Umgebung sehr vereinfachen.

Express

Express ist eines dieser Node-Modules und ermöglicht die Aufsetzung eines Servers. Dieser läuft bei diesem Beispiel auf dem Port 80 (Standard-HTTP-Port). Seine Aufgabe wird es hauptsächlich sein Daten vom Client zu empfangen und diese in die Datenbank einzubinden bzw. Daten von der Datenbank an Clients zu senden und als Verteiler der Nachrichten zwischen verschiedenen Clients zu dienen.

Da hierbei Dateipfade angegeben werden müssen (z.B.: die Index.html Seite), wurde noch das Node-Module „path“ verwendet, welche eine einfache Konkatenierung von Dateipfaden ermöglicht.

SocketIO

Das wesentliche Problem bei diesem Projekt ist die Umsetzung für eine Echtzeit-Kommunikation zwischen Usern. Somit wird der wichtigste Bestandteil des Projekts eine Lösung für dieses Problem zu finden und umzusetzen. Dies kann durch Web-Sockets ermöglicht werden (siehe Seite 3). Für die Implementierung von Web-Sockets gibt es bei NodeJS das Modul Socket.io, welches durch sogenannte „emits“ eine einfache, ungefragte Datenübertragung von dem Server zu Clients ermöglicht.

JQuery

JQuery ist eine API, welche nichts anderes macht, als JavaScript-Code einfacher darzustellen und entwicklungsfreundlicher zu machen. Alles, was mit JQuery möglich ist, ist auch mit normalem JavaScript möglich, jedoch wird es bei JQuery meist verständlicher dargestellt.

XMLHttpRequest

Der Client muss Daten laufend an den Server senden (z.B.: Chat-Nachrichten, Nutzerdaten, etc.). Dies geschieht durch sogenannte HTTP-Requests (POST bzw. GET). XMLHttpRequest ist eine, in JavaScript integrierte, API, welche genau dies ermöglicht. Wie auch im Namen beschrieben, werden diese Daten in XML-Form (Extensible Markup Language) weitergeleitet.

Diese Daten werden dann durch den Express-Server weiterverarbeitet.

MySQL

Bei MySQL handelt es sich um eine Datenbank-Umgebung, welche wir schon seit dem dritten Jahr in der HTL kennen. Der Umgang mit der Erstellung und Verwaltung von Datenbanken wurde uns schon vorher beigebracht und somit gibt es hierbei keine Probleme. Mithilfe dieser Datenbank werden die Nutzer, ihre Kontakte und die gesendeten Nachrichten gespeichert (siehe Abbildung 1: Datenbankschema).

In der Integration in die Web-Anwendung gibt es hierbei wieder ein passendes Node-Module namens mysql. Hiermit kann eine Datenbank-Verbindung sehr einfach hergestellt werden und Daten abgefragt bzw. abgespeichert werden.

Das Datenbankmodell laut Abbildung wurde mit dem Zeichenprogramm DIA erstellt.

HTML, CSS, JavaScript

Hierbei handelt es sich um die Grundbausteine des Web-Developments. Diese sollen seit der ersten Klasse schon bekannt sein. Jedoch lässt unsere Erfahrung mit JavaScript noch zu wünschen übrig, weswegen diese Programmiersprache noch Recherche mit sich bringt.

Bootstrap

Bootstrap ist eine HTML/CSS-Library, welche ein dynamisches Webseitendesign ermöglicht. HTML-Elemente können hiermit in Rows und Columns gruppiert werden, welche ihre Anordnung abhängig von der Bildschirmgröße dynamisch verändern. Somit ist auch eine ansehnliche Darstellung auf Handys ganz einfach möglich.

TypeScript

Bei TypeScript handelt es sich um eine Erweiterung von JavaScript. Mit TypeScript ist alles möglich, was auch mit JavaScript möglich ist, jedoch bietet TypeScript in vielerlei Hinsicht mehr Möglichkeiten. Mit jeglichen Eigenschaften, wie Interfaces, Membervariablen und Datentypen ist diese Sprache ähnlicher zur Programmiersprache Java, welche uns schon seit der dritten Klasse bekannt ist.

GitHub

Zur Abspeicherung und Dokumentation der wesentlichen Projektdateien ist der Dienst GitHub klar von Vorteil. Der Unterschied zu anderen onlinebasierten Speicherdiensten ist die Versionsverwaltung. Jede Abspeicherung wird einzeln eingetragen und bleibt bestehen. Somit können potenzielle, erhebliche Softwarefehler wieder verhindert werden, indem eine vorherige Version wiederhergestellt wird.

HTTPS

Bei der Verwendung von HTTP kann jeder Host, welcher sich zwischen dem Server und einem anderen Client verbindet, die Kommunikation mitverfolgen. Da auf dieser Seite viele empfindliche Informationen gesendet werden (z.B.: Passwörter, Nachrichten, andere Zugangsdaten, etc.) muss die Kommunikation verschlüsselt werden. Dies geschieht durch HTTPS.

Bei HTTP sendet der Client die Daten mit dem dazugehörigen Key an den Server. Dieser erhält diese zwei Komponenten und kann die Daten somit ablesen. Der Server sendet dann sein eigenes Datenpaket mit dem gleichen Schlüssel an den Client. Jeder Host, welcher auf der gleichen Verbindung anliegt, erhält den Key und das Datenpaket und kann demnach die Daten ablesen.

Bei HTTPS hat der Server noch zusätzlich zwei Komponenten:

- Ein Zertifikat, indem ein Public-Key steckt, mit dem die Clients Daten verpacken, jedoch nicht entpacken können.
- Ein Private-Key, mit dem der Server die mit dem Public-Key verpackten Daten entpacken kann.

Der Server sendet beim Start der Kommunikation das Zertifikat an den Client. Dieser verschlüsselt seine Daten dann mit dem Public-Key und sendet diese mit dem eigenen Session-Key an den Server. Nur er kann diese Daten mit dem Private-Key entschlüsseln. Der Private-Key wird vom Server geheim gehalten. Beide Nutzer haben nun den gleichen Session-Key, mit dem die nachfolgende Kommunikation stattfindet.

Bei der Erstellung eines HTTPS-Servers müssen somit diese zwei Komponenten mit angegeben werden. Diese können mit dem Toolkit „OpenSSL“ in GitBash mit folgendem Befehl erstellt werden:

```
$ openssl req -nodes -new -x509 -keyout server.key -out server.cert
```

Daraufhin werden noch einige Informationen (z.B.: Standort, Organisation, etc.) abgefragt und die Dateien „server.key“ und „server.cert“ erstellt. Diese werden dann im Programmcode vom Server eingelesen und die HTTPS-Connection funktioniert.

Screenshots

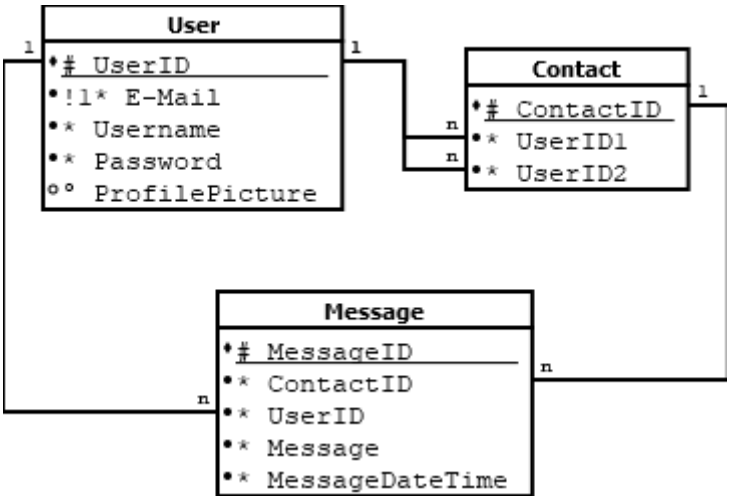

Abbildung 1: Datenbankschema

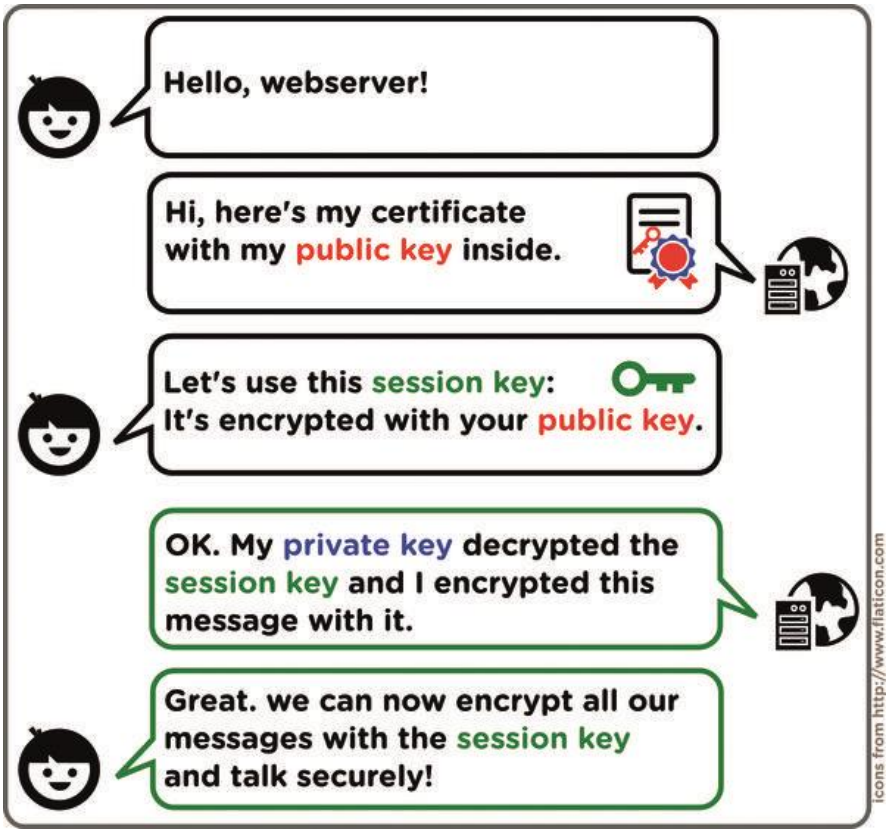

Abbildung 2: Funktionsweise HTTPS

Meilensteine

Erstes Semester

Vorbereitung	
Einlesen in die unbekannten Technologien	28.09.2020
Verfassung des Pflichtenheftes	30.09.2020
Start des Projekts	30.09.2020
Entwicklung	
Entwicklung des Webseitenlayouts (HTML + CSS)	05.10.2020
Aufsetzen des NodeJS - Servers	19.10.2020
Kommunikation Client - Server	26.10.2020
Datenbankeinbindung	30.11.2020
Einbindung von Accounts	14.12.2020
Bonusziele	
Formatierung des Texts im Markdown-Format + Bilder	28.12.2020
Endphase	
Optimierung des Codes	25.01.2021
Abgabe	01.02.2021

Zweites Semester

Entwicklung	
Entwicklung einer Freundesliste (bzw. System, um Kontakte hinzuzufügen + zu entfernen)	?
Login- und Registrierungsseite erstellen	?
Redirect auf die Hauptseite einbauen (evtl. mit Cookies, damit man sich nicht ständig einloggen muss)	?
Settings-Seite erstellen	?
Letzte Kleinigkeiten (Profilbilder, etc.) einbauen	?
Endphase	
Optimierung des Codes	?
Abgabe	?