

Web Engineering – Programmmentwurf

WordCounter

Erstellt von

Damion Häberle, Philipp Thümmler

Friedrichshafen, 14.03.2022

Erklärung

Wir versichern hiermit, dass wir unsere Projektarbeit im Modul Web Engineering selbstständig verfasst und keine anderen als die in der Dokumentation zum Programmentwurf angegebenen Quellen und Hilfsmittel benutzt haben.

Friedrichshafen, den 14.03.2022

Damion Häberle, Philipp Thümmler

Inhaltsverzeichnis

1.	Dokumentation der Arbeit	- 1 -
1.1	Dateistruktur und Aufbau	- 1 -
1.1.1	Stylesheets.....	- 1 -
1.1.2	JavaScript.....	- 2 -
1.1.3	HTML	- 5 -
1.1.4	responsive Design	- 5 -
2.	Installationsanleitung	- 6 -
3.	Verwendete Technologien	- 7 -

Abbildungsverzeichnis

Abbildung 1.1: darkmode-Switch im lightmode (links) und darkmode (rechts).....	- 1 -
Abbildung 1.2: Registrierungs-Formular.....	- 3 -
Abbildung 1.3: Anmelde-Formular	- 4 -
Abbildung 1.4: WordCounter	- 5 -
Abbildung 2.1: Kontextdiagramm login.html.....	- 6 -
Abbildung 3.1: localStorage mit Testbenutzern	- 7 -

1. Dokumentation der Arbeit

1.1 Dateistruktur und Aufbau

Da das Projekt ausschließlich aus HTML-, CSS- und JavaScript-Dateien besteht kann eine einfache Ordnerstruktur gewählt werden, bei der – ausgehend vom Hauptverzeichnis `WebEngineering_WordCounter` – alle Stylesheets im Verzeichnis mit dem Namen `./styles` und alle JavaScript-Dateien im Verzeichnis mit dem Namen `./scripts` gespeichert werden können. Außerdem werden icons im Verzeichnis `./icons` und Bilder wie das Hintergrundbild der Webseite im Verzeichnis `./images` gespeichert. Die Dokumentation für die JavaScript-Dateien mittels JSDoc sind ebenfalls in einem separaten Verzeichnis mit dem Namen `JSDoc` zu finden.

1.1.1 Stylesheets

Der Webauftritt greift auf insgesamt sieben Stylesheets zu, von denen einige in mehreren Dateien Anwendung finden. So bietet die Datei `main.css` das Grundgerüst jeder Website – mit Ausnahme der Datei `impressum.html` – und enthält Gestaltungsanweisungen für den Hintergrund, die Schriftart und das responsive Design des Hintergrundes. Die Datei `style_nav.css` beinhaltet Anweisungen für die Navigationsleiste am oberen Bildschirmrand, die ebenfalls auf jeder Website zu finden ist. Außerdem wird hier die Darstellung der Überschrift und der Links in der Navigationsleiste definiert, sowie deren Anordnung für diverse Bildschirmgrößen. Ebenfalls auf jeder Website zu finden ist ein Schalter (siehe Abb. 1.1), mit dem man durch Anklicken die Darstellung der jeweiligen Website zu einem dunklen Design ändern kann. Dieses wird in der Datei `darkmode.css` für die dafür benötigten HTML-Elemente jeder Datei definiert, was sie zur größten CSS-Datei des Projekts macht. Neben den Stylesheets, die – mit einer Ausnahme – auf jeder Website Verwendung finden gibt es für jede der vier HTML-Dateien noch ein extra Stylesheet, das Gestaltungsanweisungen für die spezifischen Elemente der jeweiligen Datei enthält. Diese sind `style_login.css`, `style_register.css`, `style_counter.css` und `style_impressum.css`.



Abbildung 1.1: darkmode-Switch im lightmode (links) und darkmode (rechts)

1.1.2 JavaScript

Auch bei den JavaScript-Dateien gibt es eine, die in alle HTML-Dateien eingebunden wird, nämlich `darkmode.js`. Der Dark Mode ist ein Feature, auf das auf allen Webseiten zugegriffen werden kann und er bleibt auch beim Wechseln zwischen den Webseiten aktiviert bzw. deaktiviert, je nachdem in welcher Position sich der Dark Mode Schalter befindet. Gesteuert wird der Dark Mode über die Klasse „dark“, die dem `body`-Element der jeweiligen HTML-Datei hinzugefügt bzw. entfernt wird. Besitzt das `body`-Element die Klasse „dark“, treffen die CSS-Selektoren der Datei `darkmode.css` zu und die jeweilige Website erhält ihr dunkles Design. Der Zustand des Dark Mode (aktiviert oder deaktiviert) wird daraufhin im internen Browserspeicher, im Folgenden als `localStorage` bezeichnet, mit einem Schlüssel-Wert-Paar gespeichert, wobei der Wert entweder den Wert „true“ für aktiviert oder „false“ für deaktiviert annehmen kann. Ist das `document`-Objekt einer Website vollständig geladen und es existiert noch kein Eintrag „darkmode“ im `localStorage`, wird dieser initial mit dem Wert „false“ hinzugefügt. Somit erscheint die Website für jeden neuen Nutzer, der die Seite zuvor noch nicht besucht hat immer im hellen Modus (Standardmodus).

Zwar ist die Website, auf die ein Nutzer standardmäßig zuerst gelangen sollte die Anmelden-Seite, allerdings soll hier zuvor auf die Registrierung-Seite eingegangen werden, um erklären zu können, wie neue Nutzer angelegt werden können, da dies zuerst nötig ist, um deren Existenz im Speicher beim Anmelden überprüfen zu können. Für die Registrierung eines Nutzers sind folgende Informationen und Anforderungen im Registrierungsformular (siehe Abb. 1.2) anzugeben und zu erfüllen:

- Vorname: muss mit einem Großbuchstaben beginnen und mindestens ein Zeichen enthalten; darf Umlaute und „ß“ enthalten
- Nachname: muss mit einem Großbuchstaben beginnen und mindestens ein Zeichen enthalten; darf Umlaute und „ß“ enthalten
- Benutzername: darf nur Großbuchstaben, Kleinbuchstaben, Zahlen, Punkte, Bindestriche und Unterstriche enthalten; darf keine Umlaute, „ß“ und andere Sonderzeichen als die genannten enthalten und muss mindestens ein Zeichen enthalten
- Email: darf vor dem `@`-Zeichen dieselben Zeichen mit denselben Anforderungen wie bei „Benutzername“ enthalten und nach dem `@`-Zeichen nur Kleinbuchstaben, Großbuchstaben, Zahlen und Bindestriche, sowie einen Punkt gefolgt von zwei oder drei Kleinbuchstaben

- Passwort: unterliegt keinen Einschränkungen bei der Wahl der Zeichen, muss aber mindestens ein Zeichen enthalten

The image shows a registration form with the following elements:

- Input field: Vorname
- Input field: Nachname
- Input field: Benutzername
- Input field: Email
- Input field: Passwort
- Button: Werte löschen
- Button: Registrieren
- Toggle switch (currently turned off)

Abbildung 1.2: Registrierungs-Formular

Drückt der Benutzer nach der Eingabe auf den Knopf „Registrieren“ wird zuerst mit der nativen HTML-Eingabeüberprüfung überprüft, ob alle Felder ausgefüllt wurden und im Fall eines leeren Feldes eine Fehlermeldung mit roter Schrift im unteren Bereich des Eingabebereichs angezeigt. Wurden dagegen alle Felder ausgefüllt, wird die Eingabe des Benutzers zuerst in einem Objekt mit dem Namen „user“ gespeichert. Zur Sicherheit wird im Anschluss daran noch einmal überprüft, ob eines der benötigten Eingabefelder leer gelassen wurde, bzw. die Information im erstellten Objekt nicht mehr vorhanden ist.

Im nächsten Schritt wird die Benutzereingabe auf korrekte Syntax überprüft. Mithilfe von drei Funktionen wird über das „user“-Objekt iteriert, jede Information – mit Ausnahme des Passworts – gegen einen regulären Ausdruck geprüft und ggf. eine passende Fehlermeldung ausgegeben. Verläuft die syntaktische Validierung erfolgreich, d.h. die Eingaben des Nutzers erfüllen die oben genannten Anforderungen, kann der neue Benutzer gespeichert werden. Um eine mehrfache Speicherung von Nutzern zu vermeiden, wird vor der Speicherung mithilfe von zwei Funktionen überprüft, ob ein Nutzer mit demselben Benutzernamen oder derselben Email, wie die, mit dem der neue Nutzer gespeichert werden soll, bereits existiert. Dafür werden iterativ alle existierenden Benutzer aus dem localStorage geladen und deren Benutzername und Email mit dem Benutzernamen und der Email des zu speichernden Nutzers verglichen. Stimmen diese überein, wird wieder eine Fehlermeldung ausgegeben und der Nutzer wird nicht gespeichert. Andererseits kann mit der Speicherung fortgefahren werden. Da alle Benutzer im localStorage mit dem Schlüssel (key) gespeichert werden, der sich aus „user“ und einer

aufsteigenden, eindeutigen Zahl zusammensetzt, muss zunächst noch überprüft werden ob es bereits einen key mit derselben Zahl wie die, mit der der neue Nutzer gespeichert werden soll gibt. Dafür werden mit einer Hilfsfunktion alle bestehenden Nutzer in einem Array gespeichert und anschließend die Zahl, die „user“ angefügt werden soll so oft erhöht, bis kein Nutzer mit dieser Zahl mehr existiert und dann der Nutzer mit seinem eindeutigen key-value-Paar im localStorage gespeichert.

Der Registrierungsprozess endet mit der Speicherung des neuen Nutzers im localStorage, woraufhin dieser auf die Anmelden-Seite zurückgeleitet wird, um sich dort mit seinen Anmeldeinformationen – dem eindeutigen Benutzernamen und zugehörigen Passwort – anzumelden. Auch hier wird zunächst die Eingabe des Benutzers im Anmeldeformular (siehe Abb. 1.3) auf Leerheit überprüft, sobald dieser auf „anmelden“ drückt wird anschließend die syntaktische Validierung durchgeführt, die hier allerdings nur den Benutzernamen umfasst.

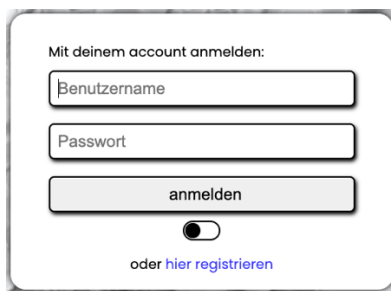
Das Bild zeigt ein Webformular für die Anmeldung. Oben steht der Text 'Mit deinem account anmelden:'. Darunter befinden sich zwei Eingabefelder: 'Benutzername' und 'Passwort'. Unter diesen Feldern ist ein breiter Button mit der Aufschrift 'anmelden'. Darunter befindet sich ein Toggle-Schalter, der currently ausgeschaltet ist. Am unteren Rand des Formulars steht der Text 'oder [hier registrieren](#)'.

Abbildung 1.3: Anmelde-Formular

Nun kann überprüft werden, ob ein Benutzer im localStorage mit dem Benutzernamen und dem zugehörigen Passwort existiert, das den eingegebenen Daten des Nutzers entspricht, der sich versucht anzumelden. Trifft dies zu, wird der Nutzer angemeldet und auf die Hauptseite (wordcount.html) weitergeleitet.

Die Funktionalität der Website wordcount.html umfasst das Zählen von Wörtern und Zeichen einer Nutzereingabe, deren Anzahl dann unter dem Eingabefeld der Nutzereingabe ausgegeben wird (siehe. Abb. 1.4). Der Zähler wird bei jedem Auftreten des Events „keyup“ aktualisiert. Der Zeichen-Zähler wird bei jeder Eingabe erhöht, der Wort-Zähler nur dann, wenn ein Trennzeichen auftritt. Als Trennzeichen sind dabei folgende definiert: ".", ",", ":", ";", "!", "?", " ", "\n". Da bei der Überprüfung der Nutzereingabe gegen einen regulären Ausdruck mit der Funktion match(), die auf dem Inhalt des value-Attributs der textarea aufgerufen wird ein Array mit allen auf den regulären Ausdruck zutreffenden Wörtern zurückgegeben wird, lässt sich deren Anzahl direkt über die Größe des Arrays bestimmen.

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Zeichen: 591

Worte: 100

Abbildung 1.4: WordCounter

1.1.3 HTML

Wie oben bereits erwähnt, besteht der Webauftritt aus insgesamt vier Websites, der jeweils eine HTML-Datei für die Struktur zugrunde liegt. Diese sind namentlich login.html, register.html, wordcount.html und impressum.html. Jede dieser Dateien enthält eine Navigationsleiste mit den Links zu den Webseiten impressum.html und wordcount.html. Letztere ist somit auch ohne eine Anmeldung oder Registrierung verwendbar. Im main-Element befindet sich in login.html und register.html jeweils das Formular für die Benutzereingaben und in counter.html die Hauptfunktionalität mit der Benutzereingabe, sowie der Ausgabe der Anzahl an Wörtern und Zeichen. In impressum.html finden hier die Kontaktangaben und der Disclaimer Platz.

1.1.4 responsive Design

Das responsive Design wird durch media-queries in den CSS-Dateien realisiert. Die Datei main.css bietet dabei die Gestaltungsanweisungen für den Hintergrund und die Datei style_nav.css die für die Navigationsleiste bei unterschiedlichen Displaygrößen. Die spezifischen CSS-Dateien für jede HTML-Datei definieren die restlichen Anweisungen wie die Darstellung der Eingabeformulare für Anmeldung und Registrierung oder des Eingabe-Containers in counter.html. media-queries wurden dabei für die folgenden Displaygrößen und -ausrichtungen definiert:

- 0px - 640px (40em): Smartphones
- 640px - 1024px (40em – 64em): Tablets (landscape)
- 640px - 1024px (40em – 64em): Tablets (portrait)

Die Gestaltungsanweisungen außerhalb der media-queries sind demnach für Monitore mit einer Größe von mindestens 1024 px und größer definiert.

2. Installationsanleitung

Durch die Beschränkung auf eine Clientseitige Anwendung, ist kein Webserver nötig und auch keine weitere Installation oder Programme. Für die Nutzung der Website wird diese einfach in einem Browser aus dem lokalen Dateisystem geöffnet. Als Startseite ist die login.html vorgesehen, von ihr ausgehend gelangt man auf jede andere Website, wie man in ihren Kontextdiagramm (siehe Abb. 1.5) erkennt. Eine Internetverbindung ist allerdings notwendig, da die JavaScript Bibliothek einer hash-Funktion über eine Website eingebunden wird.

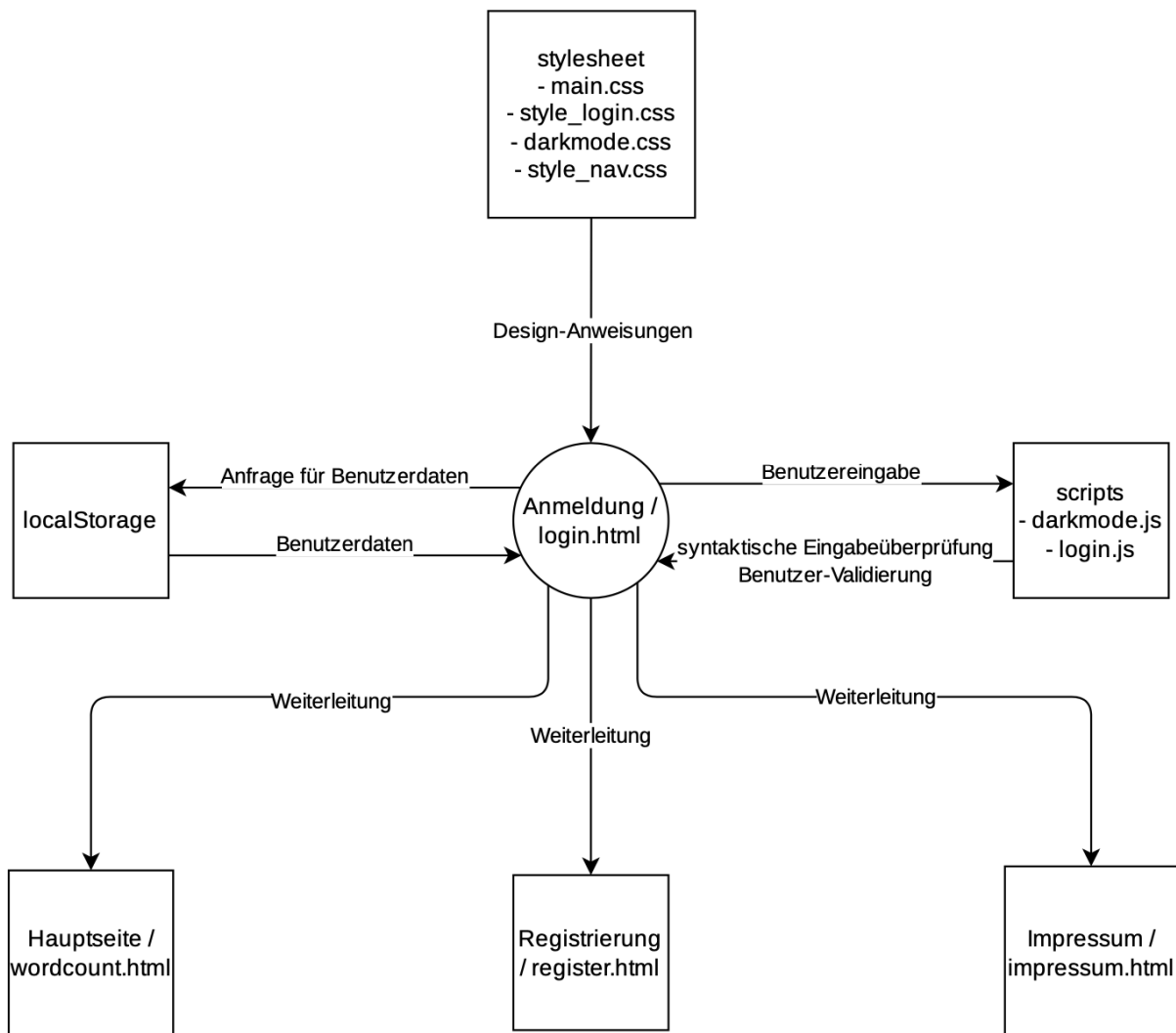


Abbildung 2.1: Kontextdiagramm login.html

3. Verwendete Technologien

Bei den verwendeten Technologien handelt es sich – wie bereits ausführlich beschrieben – um HTML, CSS und JavaScript, den localStorage (siehe Abb. 3.1) der Web Storage API, sowie der über eine Website eingebundenen Bibliothek blueimp-md5 zum Speichern der Passwörter als hash. Der localStorage wird verwendet, da die uneingeschränkte Funktionalität der Website auch ohne eine Anmeldung oder Registrierung verwendet werden kann und somit eine aufwändige Sicherung nicht notwendig ist, außerdem reichen die verfügbaren 5 MB für das Speichern der nicht sehr umfangreichen Nutzerdaten aus.

Key	Value
darkmode	false
user2	{ "firstname": "TestUser_firstname", "lastname": "TestUser_lastname", "username": "test.user", "email": "test@user.de", "password": "76a2173be6393254e72ffa4d6df1030a" }
user3	{ "firstname": "TestUser2_firstname", "lastname": "TestUser2_lastname", "username": "test.user2", "email": "test2@user.de", "password": "76a2173be6393254e72ffa4d6df1030a" }
user4	{ "firstname": "TestUser3_firstname", "lastname": "TestUser3_lastname", "username": "test.user3", "email": "test3@user.de", "password": "363b122c528f54df4a0446b6bab05515" }
user5	{ "firstname": "Max", "lastname": "Mustermann", "username": "max-mustermann", "email": "max.mustermann@example.com", "password": "2ffe4e77325d9a7152f7086ea7aa5114" }
user6	{ "firstname": "Erika", "lastname": "Mustermann", "username": "Erika_Mustermann", "email": "mustermann.erika@example.com", "password": "f7f7591403c6c431053920223069550a" }

```
▼ { "firstname": "Max", "lastname": "Mustermann", "username": "max-mustermann", ... }  
  email: "max.mustermann@example.com"  
  firstname: "Max"  
  lastname: "Mustermann"  
  password: "2ffe4e77325d9a7152f7086ea7aa5114"  
  username: "max-mustermann"
```

Abbildung 3.1: localStorage mit Testbenutzern