|  |  |
| --- | --- |
|  | Praxisarbeit «Aufgabenverwaltung» Webtechnologie 1  Neven Kljajić |

Inhaltsverzeichnis

[1 Einleitung 3](#_Toc128745996)

[2 Planung 3](#_Toc128745997)

[3 Architektur 4](#_Toc128745998)

[3.1 Architekturbild 4](#_Toc128745999)

[3.2 Komponenten 5](#_Toc128746000)

[4 Use-Case-Diagramm 7](#_Toc128746001)

[5 ERM (Entity-Relationship-Modell) 9](#_Toc128746002)

[6 Testcases und Testplan 10](#_Toc128746003)

[7 Anleitung zur Inbetriebnahme der Webseite 11](#_Toc128746004)

[8 Lessons Learned 11](#_Toc128746005)

[9 Selbstständigkeitserklärung 12](#_Toc128746006)

# Einleitung

Diese Dokumentation dient als umfassender Leitfaden für das Praxisprojekt «Aufgabenverwaltung», eine Webseite, die zur Verwaltung von eigenen Aufgaben entwickelt wurde.

Ziel dieses Projekts war es, ein einfach zu bedienendes Aufgabenverwaltungssystem, um Aufgaben effizienter zu verwalten. Die Anwendung besteht aus zwei Teilen:

* Tasks: Übersicht aller Aufgaben, sortiert nach Fälligkeit. Es kann nach Aufgabentypen gefiltert werden. Zudem können Aufgaben erfasst, editiert und gelöscht werden.
* Admin-Bereich: Verwaltung von Aufgabentypen (erfassen, editieren, löschen) sowie Personen, für welche Aufgaben zugewiesen werden können.

Diese Dokumentation beschreibt die verschiedenen Komponenten des Projekts, einschließlich der Architektur, der verwendeten Technologien, das Datenbankdesigns des Systems, die Planung, Testcases etc. Darüber hinaus bietet sie eine Anleitung zur Inbetriebnahme der Webseite, damit dies der Dozent testen und bewerten kann.

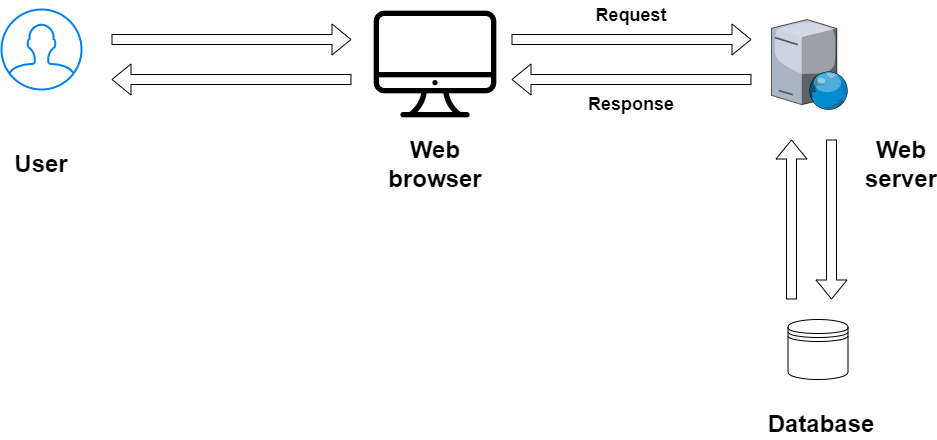
# Planung

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Aufgabe | Geplanter STart | Geplantes Ende | Tatsächlicher Start | Tatsächliches Ende | Aufwand |
| ERM erstellen | 28.12.22 | 28.12.22 | 13.02.23 | 13.02.23 | 1 h |
| Datenbank + tabellen SQL erstellen | 28.12.22 | 28.12.22 | 28.12.22 | 28.12.22 | 2 h |
| Task Seite erstellen | 28.12.22 | 28.12.22 | 28.12.22 | 29.12.22 | 2 h |
| Admin Seite + Personen + Aufgabentyp | 29.12.22 | 30.12.22 | 29.12.22 | 04.01.23 | 4h |
| CRUD Operationen implementieren | 02.01.23 | 05.01.23 | 05.01.23 | 19.01.23 | 6h |
| Aufgabe als abgeschlossen markieren | 05.01.23 | 05.01.23 | 19.01.23 | 19.01.23 | 1 h |
| Aufgaben importieren Funktion | 22.01.23 | 22.01.23 | 22.01.23 | 29.01.23 | 3 h |
| Überarbeitung des Codes | 05.02.23 | 05.02.23 | 05.02.23 | 05.02.23 | 2 h |
| Design des Projektes | 05.02.23 | 09.02.23 | 05.02.23 | 06.02.23 | 2 h |
| Dokumentation Erstellen | 20.02.23 | 05.03.23 | 20.02.23 | 05.03.23 | 5 h |

Die Planung der Arbeit wurde von Beginn an verfolgt und ständig dokumentiert. Die geplanten Starts und Ende wurden nicht von Anfang an definiert, jedoch bei Start der Aufgabe, legte der Verfasser immer fest bis wann er dies erledigt haben wolle. Dies funktionierte im Normalfall sehr gut und die Schätzung, bis wann dies erledigt, sein sollte, passte meistens. Leicht rot markiert sieht man, das es bei einigen Aufgaben ein deutlich anderes Ende hat als geplant. Dies liegt vor allem daran, dass der Verfasser länger an einer Lösung arbeitete als geplant. Diese 3 Aufgaben, die rot markiert sind, die waren von der Komplexität her auch am schwierigsten zu implementieren. Das Design des Projekts empfand der Verfasser als eine eher experimentelle Aufgabe und hatte Spass, mehreres auszuprobieren. Dies wurde jedoch auch sehr schnell erledigt und war sehr zufriedenstellend.

# Architektur

## Architekturbild



**Web-Browser**:

Der Webbrowser ist die wichtigste Komponente für den Zugriff auf die webbasierte Anwendung. Er stellt die Benutzeroberfläche bereit, die den Inhalt der Anwendung anzeigt und den Benutzern die Interaktion mit dem System ermöglicht. Der Browser ist für das Rendern von HTML und CSS sowie für die Verarbeitung von Benutzereingaben zuständig. Die Benutzeroberfläche der Anwendung wird mit HTML und CSS erstellt, wobei Bootstrap zusätzliche Styling- und Layout-Optionen bietet. Für Bootstrap hat sich der Verfasser deshalb entschieden, weil es eine moderne und einfache Alternative zu einfachem CSS darstellt. Es besteht die Möglichkeit einfach und zeitentsprechende Vorlagen zu entnehmen oder die vorkonfigurierten Klassen zu benutzen.

Die Aufgabe des Browsers besteht darin, HTTP-Anfragen an den Server zu senden und die Antwort für den Benutzer anzuzeigen. Die PHP-Skripte der Anwendung generieren auf der Serverseite dynamisch HTML, das dann als Antwort auf eine Benutzeranfrage an den Browser gesendet wird.

Insgesamt ist der Webbrowser eine Schlüsselkomponente in der Client-Server-Architektur der Anwendung, die es den Benutzern ermöglicht, über eine moderne, reaktionsschnelle und benutzerfreundliche Schnittstelle auf die Funktionen der Anwendung zuzugreifen und mit ihnen zu interagieren.

**Webserver**:

Der Webserver ist eine entscheidende Komponente der Systemarchitektur, da er als Gateway für den Zugang der Benutzer zu den Systemfunktionen dient. Der Webserver ist für den Empfang von Benutzeranfragen, deren Verarbeitung und die Rückgabe der entsprechenden Antwort verantwortlich. Die in diesem Projekt verwendete Webserver-Software ist Apache, ein weit verbreiteter Open-Source-Webserver. Apache wurde aufgrund seiner Stabilität, Leistung und Sicherheitsfunktionen ausgewählt, jedoch auch weil dies im Unterricht vorgestellt wurde und die Arbeit damit sehr gut funktioniert.

Der Webserver ist so konfiguriert, dass er mit PHP arbeitet, einer serverseitigen Skriptsprache, die zur Erstellung dynamischer Webseiten verwendet wird. Die serverseitigen Skripte sind für die Interaktion mit der Datenbank, die Verarbeitung von Benutzereingaben und die Generierung dynamischer Inhalte verantwortlich, die dem Benutzer über den Webserver zur Verfügung gestellt werden. Zusätzlich zu Apache und PHP verwendet das System MySQL als Datenbankmanagementsystem, um Daten zu speichern und zu verwalten.

Insgesamt ist die Webserver-Komponente ein wichtiger Bestandteil der Systemarchitektur.

**Datenbank**:

Die Datenbankkomponente bildet die Grundlage des Systems und ist für die Speicherung, Verwaltung und den Zugriff auf die Anwendungsdaten zuständig. Die für dieses Projekt gewählte Datenbanktechnologie ist MySQL, ein open source relationales Datenbankmanagementsystem. Die Wahl fiel auf MySQL, weil uns dies im Unterricht vorgestellt wurde, jedoch punktet MySQL auch wegen seiner Zuverlässigkeit, Skalierbarkeit und Benutzerfreundlichkeit sowie wegen seiner weiten Verbreitung und der großen Nutzergemeinde.

Das Datenbankschema wurde entwickelt, um die für das System erforderlichen Daten zu speichern, darunter Aufgaben, Aufgabentypen und Personen. Jede Tabelle in der Datenbank verfügt über einen Primärschlüssel und geeignete Indizes, um einen effizienten Datenabruf und eine effiziente Datenmanipulation zu gewährleisten.

Insgesamt bietet die Wahl von MySQL als Datenbanktechnologie für dieses Projekt eine zuverlässige, skalierbare und sichere Grundlage für das System, und das Datenbankdesign gewährleistet eine effiziente und konsistente Datenspeicherung und -verwaltung.

## Komponenten

Für die Praxisarbeit wurden mehrere PHP-files erstellt die notwendig sind damit die Aufgabenverwaltung komplett funktioniert. Hinzu gibt es noch ein einfaches CSS-file, ein SQL-file, sowie auch ein XML- und JSON-file. Diese werden kurz genauer beschrieben:

**addPerson.php**:

Diese Datei wird verwendet, um eine neue Person in die Tabelle "persons" in der Datenbank "todo" aufzunehmen. Die Person steht dann bei der Erstellung eines Tasks zur Auswahl zur Verfügung.

Es überprüft simpel ob alle Felder ausgefüllt sind und schreibt dann die Person mit einem «Insert»-Befehl in die Datenbank.

**addType.php**:

Diese Datei wird verwendet, um einen neuen Aufgabentyp zur Tabelle "task\_types" in der Datenbank "todo" hinzuzufügen. Der Aufgabentyp steht dann bei der Erstellung eines Tasks zur Auswahl zur Verfügung.

Es überprüft simpel ob alle Felder ausgefüllt sind und schreibt dann den Aufgabentyp mit einem «Insert»-Befehl in die Datenbank.

**admin.php**:

Diese Datei ist die Verwaltungsseite für die Anwendung "Aufgabenverwaltung", die es dem Benutzer ermöglicht Personen und Aufgabentypen hinzuzufügen und auch XML- und JSON-Dateien hochzuladen, welche Tasks enthalten.

Hier wird ein ganzer DOM dargestellt welche ein Nav und einen Footer im Body enthalten, sowie zwei Forms für das Hinzufügen von Personen und Aufgabentypen und auch eine Form für das Importieren der Dateien welche auf «importTask.php» verweist

**createNewTask.php**:

Diese Datei wird verwendet, um eine neue Aufgabe in der Tabelle "tasks" in der Datenbank "todo" zu erstellen.

Hier werden die Werte entgegengenommen, die vom Benützer eingefüllt worden sind und überprüft, ob alle Werte vorhanden sind. Danach wird überprüft, ob das Fälligkeitsdatum nicht in der Vergangenheit liegt. Und dann wird die Aufgabe mit einem «Insert»-Befehl in die Datenbank geschrieben. Hinzu noch das HTML welches die Form darstellt mit den jeweiligen Eingabefeldern und den zwei Dropdown für die Personen und Aufgabentypen.

**db\_Connection.php**:

Diese Datei stellt eine Verbindung zur 'todo'-Datenbank her, wobei die in der Datei angegebenen Anmeldeinformationen verwendet werden.

Diese Datei wird dann überall dort verwendet, wo eine Verbindung nötig ist, mit einem «inlcude».

**deleteTask.php**:

Diese Datei wird verwendet, um eine Aufgabe aus der Tabelle "tasks" in der Datenbank "todo" zu löschen.

Es überprüft die ID der Aufgabe und löscht dann die Aufgabe mit einem «Delete»-Befehl aus der Datenbank.

**displayTask.php**:

Diese Datei wird verwendet, um die Aufgaben in der Tabelle "tasks" in der Datenbank "todo" anzuzeigen.

Hier wird zuerst die Möglichkeit implementiert, damit man die Aufgaben nach Aufgabentyp filtern kann. Danach werden alle Aufgaben in einer Tabelle dargestellt die nicht als «completed» markiert worden sind und sie sind per Default nach der Fälligkeit sortiert. Wenn eine Aufgabe bereits in die Fälligkeit des jeweiligen Aufgabentyps gefallen ist, wird diese in der entsprechenden Farbe dargestellt. Es besteht in der Auflistung der Aufgaben die Möglichkeit, die Aufgabe zu bearbeiten, löschen oder als «Done» / »completed» zu markieren.

**editTask.php**:

Diese Datei wird verwendet, um eine Aufgabe in der Tabelle "tasks" in der Datenbank "todo" zu bearbeiten.

Dies ist möglich in der Übersicht der Aufgaben, wenn man bei der jeweiligen Aufgabe auf «Edit» klickt. Es überprüft die ID der Aufgabe und bearbeitet dann die Aufgabe mit einem «Update»-Befehl in der Datenbank.

**importTask.php**:

Diese Datei wird verwendet, um Aufgaben aus einer XML- oder JSON-Datei zu importieren und sie der Tabelle "tasks" in der Datenbank "todo" hinzuzufügen.

Es wird zuerst überprüft, ob die Datei ausgewählt wurde und überprüft dann die Dateiendung. Möglich ist nur .xml oder .json. Danach werden dann die Aufgaben in der Datei überprüft und zuerst kontrolliert, ob das Fälligkeitsdatum nicht in der Vergangenheit liegt. Danach schreibt es die Aufgabe mit einem «Insert»-Befehl in die Datenbank.

**markTask.php**:

Diese Datei wird verwendet, um eine Aufgabe in der Tabelle "tasks" in der "todo"-Datenbank als erledigt zu markieren.

In der Aufgabenübersicht kann man eine Aufgabe als «Done» markieren und dann ist sie erldeigt und man sieht die Aufgabe nicht mehr in der Übersicht. Es überprüft die ID der Aufgabe und mit einem «Update»-Befehl setzt es das «Completed» auf 1 und ist somit erledigt.

**tasks.php**:

Diese Datei ist die Aufgabeübersicht für die Anwendung "Aufgabenverwaltung", die es dem Benutzer ermöglicht die Aufgaben anzuschauen und die gewünschten CRUD-Operationen durchzuführen.

In diesem DOM befindet sich das Nav ganz oben. Danach sofort der Button für die Erstellung einer neuen Aufgabe. Darunter dann die ganze Übersicht der Aufgaben.

**Stylesheet.css**

Da grösstenteils Bootstrap verwendet wird, wurde in dieser Datei nicht viel definiert ausser das Nav Div um das Margin anzupassen.

**CreateDB+.sql**

In dieser Datei wird die Datenbank «todo» erstellt und gleich die benötigten Tabellen «tasks», «task\_types» und «persons».

Es werden auch gleich 3 Beispiel Personen hinzugefügt, 3 Aufgabentypen und 8 Aufgaben. Es wird auch gleich ein User erstellt, der gleich ist wie im db\_Connection.php.

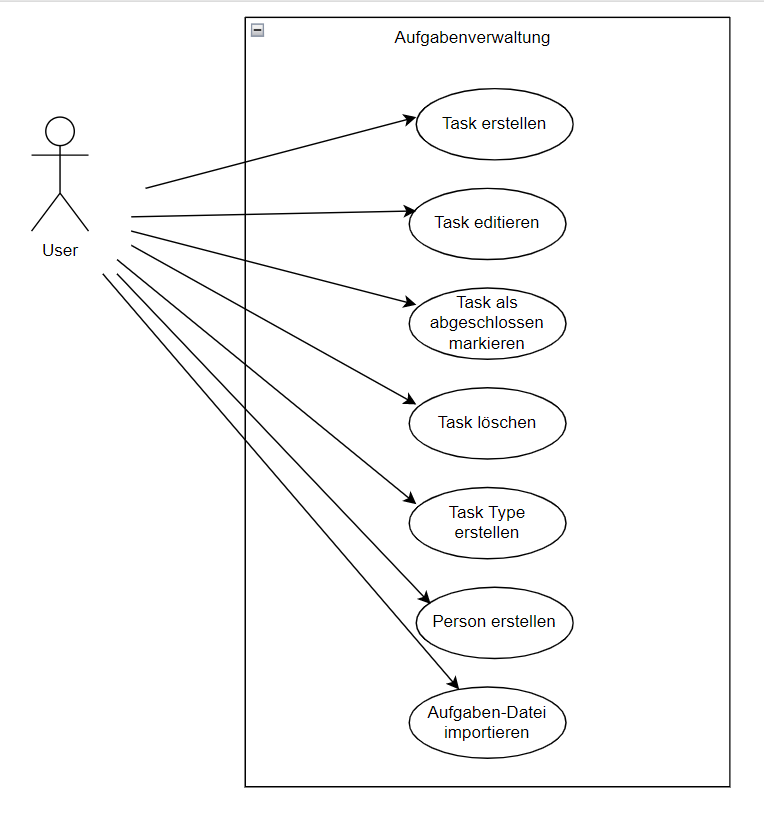
**BeispielTaskXML.xml**

In dieser Datei befinden sich 3 Beispiel Aufgaben in XML-Form, welche importiert werden können.

**TaskBeispiel.json**

In dieser Datei befinden sich 3 Beispiel Aufgaben in JSON-Form, welche importiert werden können.

# Use-Case-Diagramm



**Task erstellen**:

Der User erstellt einen neuen Task, indem er ein Formular ausfüllt, das den Aufgabentitel (Title), die Beschreibung (Description), die Person (Person), der die Aufgabe zugewiesen werden soll, den Aufgabentyp (Task Type) und das Fälligkeitsdatum (Due Date) enthält. Nach dem Absenden des Formulars wird der Task zur Liste der Tasks hinzugefügt.

**Task editieren**:

Der User wählt einen Task aus der Liste der Tasks aus und klickt auf die Schaltfläche "Edit". Er gelangt zu einem Formular, das mit den Details des ausgewählten Task vorausgefüllt ist. Der User kann jedes beliebige Feld des Formulars bearbeiten und dann das Formular absenden, um die Änderungen am Task zu speichern.

**Task als abgeschlossen markieren:**

Der User wählt einen Task aus der Liste der Tasks aus und klickt auf die Schaltfläche "Done". Der Task wird als erledigt markiert und ist dann nicht mehr ersichtlich in der Liste.

**Task löschen**:

Der User wählt einen Task aus der Liste der Tasks aus und klickt auf die Schaltfläche "Delete". Danach ist der Task gelöscht und nicht mehr ersichtlich.

**Task Type erstellen**:

Der User erstellt einen neuen Task Type im Admin Bereich, indem er ein Formular ausfüllt, das den Namen (Name), die Farbe (Color) und die Frist (Due Date (in days)) des Task Type enthält. Nach dem Absenden des Formulars wird der neue Task Type im Dropdown der Task Types vorhanden sein.

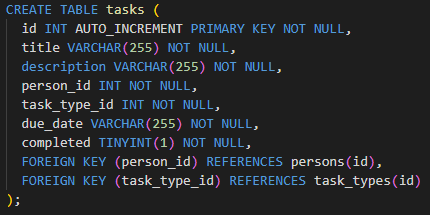
**Person erstellen**:

Der User erstellt eine neue Person im Admin Bereich, indem er ein Formular ausfüllt, das den Vornamen (First Name), den Nachnamen (Last Name) und die E-Mail-Adresse der Person enthält. Nach dem Absenden des Formulars wird die neue Person im Dropdown der Personen vorhanden sein.

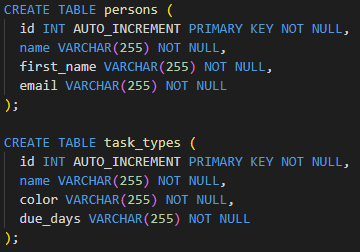
# ERM (Entity-Relationship-Modell)

Die Implementierung des ERM erfolgte mit einer MySQL-Datenbank. Der Verfasser erstellte zunächst die Tabellen mit Hilfe der CREATE TABLE-Anweisungen und definierten dann die Beziehungen zwischen ihnen auf der Grundlage der Projektanforderungen.

Zuerst wurde die Tabelle «tasks» erstellt:

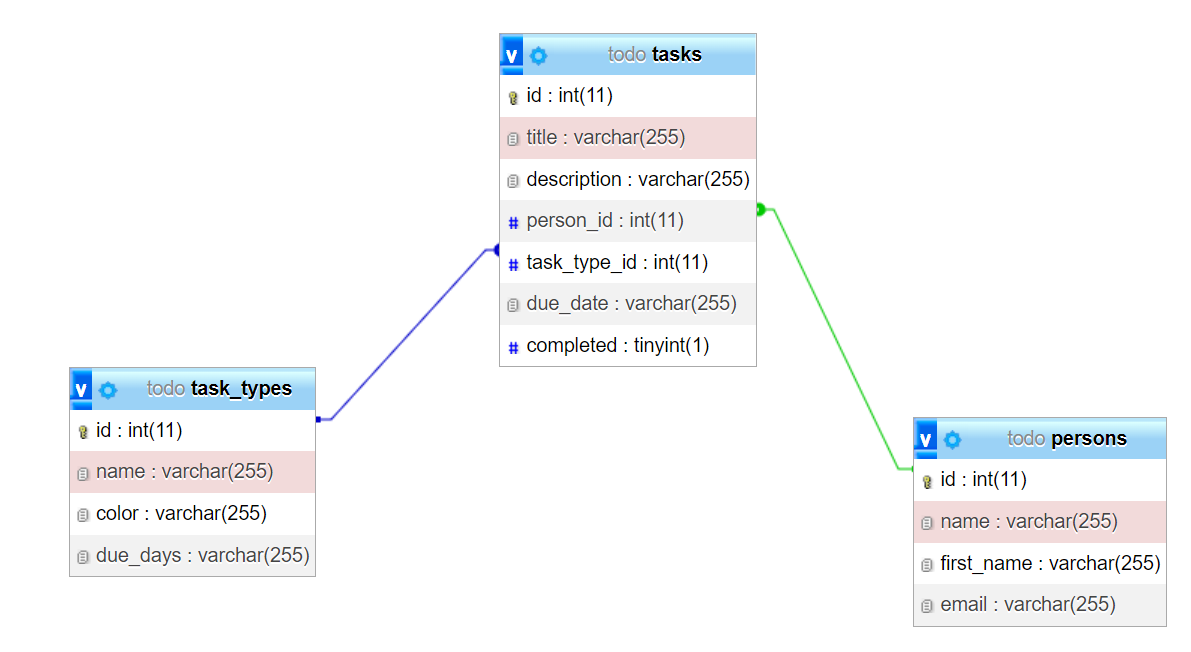


Wie in der Projektanforderungen beschrieben, wurde ein Titel, eine Beschreibung, eine Person, ein Aufgabentyp sowie ein Fälligkeitsdatum angegeben. Zusätzlich wurde auch eine «completed» Spalte erstellt, damit man später einen Task als abgeschlossen markieren kann. Für die Person und den Aufgabentyp wurde jedoch eine separate Tabelle erschaffen:



Als Primärschlüssel wurde für alle 3 Tabellen, jeweils die ID gewählt. Mit den Fremdschlüsseln «person\_id» und «task\_type\_id» wurde in der Tabelle «tasks» die Beziehung sichergestellt.

Die Tabellen wurden so angelegt, dass sie die für unsere Webanwendung erforderlichen Informationen, wie Benutzerdaten und Aufgaben, speichern. Daraufhin konnte ganz einfach das ERM-Diagramm erstellt werden, dass die Beziehungen zwischen den Tabellen grafisch darstellen:



# Testcases und Testplan

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Test-Case | Tester | Geplant |
| 1 | Task erstellen | Neven | Ja |
| 2 | Task editieren | Neven | Ja |
| 3 | Task als abgeschlossen markieren | Neven | Ja |
| 4 | Task löschen | Neven | Ja |
| 5 | Task Type erstellen | Neven | Ja |
| 6 | Person erstellen | Neven | Ja |
| 7 | Aufgaben-Datei importieren | Neven | Ja |

|  |  |  |  |
| --- | --- | --- | --- |
| ID | 1 | | |
| Name | Task erstellen | | |
| Beschreibung | Auf der «Tasks» Seite auf «Create a new Task» klicken und dann die Aufgabe erstellen und bestätigen auf der neuen Seite. | | |
| Schrittnummer | Aktion | Daten | Erwartet |
| 1. | Der Benutzer klickt auf den Button «Create a new Task» | - | Beim Klick auf den Button öffnet sich die neue Seite, um eine Aufgabe zu erstellen mit den gewünschten Daten |
| 2. | Der Benutzer füllt die Felder Title, Description aus, wählt aus dem Dropdown Menü eine Person und beim zweiten Dropdown einen Task Type aus und wählt das gewünschte Datum bei Due Date aus. | - Task 1  - Test Description 1  - Potter Harry (id: 1)  - A (id: 1)  - 04.03.2023 | Die Felder lassen sich ausfüllen. Aus den Dropdowns lassen sich Personen und Aufgabentypen auswählen und beim Datum lässt sich ein Datum aus dem Kalender auswählen. |
| 3. | Klick auf «Create Task» | - | Nach dem Klick wird man auf die «Tasks» Seite geführt und dort ist der erstellte Task ersichtlich in der Übersicht. |

|  |  |  |  |
| --- | --- | --- | --- |
| ID | 2 | | |
| Name | Task editieren | | |
| Beschreibung | Auf der «Tasks» Seite in der Übersicht der Aufgaben, rechts von der Aufgabe auf «Edit» klicken | | |
| Schrittnummer | Aktion | Daten | Erwartet |
| 1. | Der Benutzer klickt auf «Edit» rechts von der Aufgabe «Task 1» | - | Beim Klick auf «Edit» öffnet sich die neue Seite, um eine Aufgabe zu bearbeiten. Die Felder sind mit den bestehenden Daten bereits gefüllt. |
| 2. | Der Benutzer ändert im Feld Description den Text auf: «Test Description 1.1!!!» und ändert die Person auf «Parker Peter» | - Test Description 1.1!!!  - Parker Peter | Die Felder lassen sich bearbeiten und eine neue Person lässt sich auswählen. |
| 3. | Klick auf «Save» | - | Nach dem Klick wird man auf die «Tasks» Seite geführt und dort ist der bearbeitete Task ersichtlich in der Übersicht mit den neuen Daten. |

|  |  |  |  |
| --- | --- | --- | --- |
| ID | 3 | | |
| Name | Task als abgeschlossen markieren | | |
| Beschreibung | Auf der «Tasks» Seite in der Übersicht der Aufgaben, rechts von der Aufgabe auf «Done» klicken | | |
| Schrittnummer | Aktion | Daten | Erwartet |
| 1. | Der Benutzer klickt auf «Done» rechts von der Aufgabe «Task 1» | - | Beim Klick auf «Done» verschwindet die Aufgabe und ist nicht mehr ersichtlich in der Übersicht. |

|  |  |  |  |
| --- | --- | --- | --- |
| ID | 4 | | |
| Name | Task löschen | | |
| Beschreibung | Auf der «Tasks» Seite in der Übersicht der Aufgaben, rechts von der Aufgabe auf «Delete» klicken | | |
| Schrittnummer | Aktion | Daten | Erwartet |
| 1. | Der Benutzer klickt auf «Delete» rechts von der Aufgabe «Ollivanders» | - | Beim Klick auf «Delete» verschwindet die Aufgabe und ist nicht mehr ersichtlich in der Übersicht. |

|  |  |  |  |
| --- | --- | --- | --- |
| ID | 5 | | |
| Name | Task Type erstellen | | |
| Beschreibung | Auf der «Admin» Seite das Task Type Formular ausfüllen. | | |
| Schrittnummer | Aktion | Daten | Erwartet |
| 1. | Der Benutzer füllt die Felder Name, Due Date (in days) aus und wählt eine Farbe aus. | - Type Test 1  - RGB: 26 47 219  - 6 | Die Felder lassen sich ausfüllen und aus der Farbpalette lässt sich eine Farbe auswählen. |
| 2. | Klick auf den Button «Add Type». | - | Neuer Aufgabentyp ist erstellt. |
| 3. | Zurück zur «Task» Seite eine neue Aufgabe erstellen und im Dropdown «Task Type» neu erstellten Aufgabentyp zur Auswahl sehen | - | Im Dropdown des Aufgabentyps zur Erstellung einer Aufgabe ist nun der neu erstellte Aufgabentyp ersichtlich und zur Auswahl bereit. |

|  |  |  |  |
| --- | --- | --- | --- |
| ID | 6 | | |
| Name | Person erstellen | | |
| Beschreibung | Auf der «Admin» Seite das Personen Formular ausfüllen. | | |
| Schrittnummer | Aktion | Daten | Erwartet |
| 1. | Der Benutzer füllt die Felder First Name, Last Name und E-Mail Adress aus. | - Ron  - Weasley  - ron.weasley@gmail.com | Die Felder lassen sich ausfüllen. |
| 2. | Klick auf den Button «Add Person». | - | Neue Person ist erstellt. |
| 3. | Zurück zur «Task» Seite eine neue Aufgabe erstellen und im Dropdown «Person» neu erstellte Person zur Auswahl sehen | - | Im Dropdown der Person zur Erstellung einer Aufgabe ist nun die neu erstellte Person ersichtlich und zur Auswahl bereit. |

|  |  |  |  |
| --- | --- | --- | --- |
| ID | 7 | | |
| Name | Aufgaben-Datei importieren | | |
| Beschreibung | Auf der «Admin» Seite lässt sich eine XML oder JSON Datei importieren. | | |
| Schrittnummer | Aktion | Daten | Erwartet |
| 1. | Der Benutzer klickt auf Datei auswählen und wählt eine JSON-Datei aus | - | Der Explorer öffnet sich und man kann eine XML oder JSON Datei Öffnen. |
| 2. | Klick auf den Button «Import». | - | Gelangt auf die «Tasks» Seite und in der Übersicht sind die importierten Aufgaben zu sehen. |

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Test-Case | Tester | Erfolgreich |
| 1 | Task erstellen | Neven | Ja |
| 2 | Task editieren | Neven | Ja |
| 3 | Task als abgeschlossen markieren | Neven | Ja |
| 4 | Task löschen | Neven | Ja |
| 5 | Task Type erstellen | Neven | Ja |
| 6 | Person erstellen | Neven | Ja |
| 7 | Aufgaben-Datei importieren | Neven | Ja |

# Anleitung zur Inbetriebnahme der Webseite

# Lessons Learned

Während der Entwicklung dieses Projekts wurden mehrere wichtige Lektionen gelernt. Diese Lektionen haben nicht nur die Qualität des Endprodukts verbessert, sondern auch zur Entwicklung besserer Programmierpraktiken beigetragen.

Erstens sind die richtige Planung und der richtige Entwurf für den Erfolg eines jeden Projekts von entscheidender Bedeutung. Die Zeit, die man sich für die Erstellung eines soliden Architektur- und Use Case Diagramm nahm, half dabei, den Umfang und die Anforderungen des Projekts zu definieren. Dies ermöglichte eine bessere Verwaltung von Projektzeitplänen und Ressourcen.

Zweitens sollte das Testen ein kontinuierlicher Prozess während des gesamten Projekts sein. Durch regelmäßiges Testen einzelner Komponenten und Funktionen sowie des Systems als Ganzes konnten Probleme bereits in einem frühen Stadium des Entwicklungsprozesses erkannt werden. Dies ermöglichte eine schnellere Lösung von Problemen und trug dazu bei, das Auftreten größerer Probleme im weiteren Verlauf zu verhindern.

Drittens ist die Dokumentation von wesentlicher Bedeutung für die langfristige Pflege der Codebasis. Eine ordnungsgemäße Dokumentation des Codes, einschließlich Kommentaren und klaren Namenskonventionen, trug dazu bei, die Lesbarkeit und Wartbarkeit des Codes zu verbessern.

Für die notwendigen Fähigkeiten und Kenntnisse für dieses Projekt, hat der Verfasser zunächst gründlich alle Unterrichtsmaterialien nochmals durchgearbeitet und Recherchen zu den für die Webentwicklung erforderlichen Technologien und Tools durchgeführt. Ich habe Online-Tutorials, Dokumentationen und Foren konsultiert, um mir ein klareres Bild vom Projekt zu machen. Hierbei sind die Technologien HTML, CSS, Bootstrap, PHP und MySQL sehr wichtig gewesen.

Außerdem hat der Verfasser bereits einige Erfahrungen von früheren Webprojekten, sowie einige Kenntnisse von der Berufstätigkeit, bevor das Projekt begann. Dies half, mit dem Entwicklungsprozess vertraut zu werden und eventuelle Wissenslücken zu erkennen, die geschlossen werden mussten.

Im Laufe des Projekts stieß der Verfasser auch auf neue Herausforderungen, bei denen er neue Fähigkeiten aneignen musste. In diesen Situationen nutzte er seine Problemlösungsfähigkeiten und suchte weiterhin nach Ressourcen und Anleitung, um die Hindernisse zu überwinden.

Zusammenfassend kann man sagen, dass die notwendigen Fähigkeiten und Kenntnisse für die Durchführung dieses Projekts durch eine Kombination aus Forschung, Praxis und Lernen am Arbeitsplatz angeeignet werden konnten.

Zusammenfassend lässt sich sagen, dass dieses Projekt eine wertvolle Lernerfahrung war, die die Bedeutung von Planung, Tests, Dokumentation und der Programmierung der Webseite verdeutlicht hat. Diese Lektionen werden zweifellos bei zukünftigen Projekten nützlich sein und die Entwicklungspraktiken auch in Zukunft leiten.

# Selbstständigkeitserklärung

Hiermit bestätige ich, Neven Kljajić, die Arbeit selbstständig und ohne Hilfe von anderen Personen durchgeführt zu haben. Des Weiteren bestätigt sie hiermit, dass er keine weiteren Quellen, ausser den angegebenen, verwendet zu haben.

Neven Kljajic   
Zürich, 14. September 2022