# Pflichtenheft To-Do-Listen Webseite

|  |  |
| --- | --- |
| **Kunde** | Herr Zimmermann |
| **Entwickler** | Luca Hutter, Jonathan Forker |
| **Benutzer** | Unsere Mitschüler |
| **Projekt** | To-Do-Listen Webseite |
| **Version** | 1.0.0 |

## Einleitung

Das Ziel des Projektes ist es eine funktionierende Webseite für eine To-Do-Liste. Man soll auf der Webseite To-Do-Elemente erstellen, bearbeiten, und löschen können. Es soll möglich sein einen Account erstellen um seine Einträge abspeichern zu können.

Zielgruppen:

Abgrenzung:

## Ist- Zustand

## Soll-Zustand

## Funktionale Anforderungen

Systemfunktionen

**Registrieren:** Einen eigenen Account erstellen mit Username, E-Mail und Passwort erstellen

**Login:** Mit dem erstellten Account auf der Webseite einloggen

**Logout:** Am Ende der Nutzung die wieder auszuloggen

**Teams:** Um Projekte mit anderen zu teilen und zusammenzuarbeiten

**Projekte:** Jedes Projekt ist seine eigenen Listen. Dies dient dazu Struktur und Übersicht in die Listen zu bringen

**Prioritätsgruppen:** Jedes Projekt hat drei Prioritätsgruppen „High“, „Medium“ und „Low“. Damit ist es möglich die einzelnen To-Do’s nach Wichtigkeit aufzuteilen.

**To-Do’s:** Jedes To-Do hat seinem eigenen Titel Inhalt und sein eigens Enddatum. Die To-Do‘s sollen über Drag and Drop zwischen den verschiedenen Prioritätsgruppen und untereinander verschiebbar sein.

Anwendungsfälle

**Registrierung:** Die Registrierung beginnt mit der Eingabe seiner Daten z.B. Username, E-Mail und Passwort diese werden dann durch einen HTML-Form Post an das dazugehörige PHP-skript gesendet. Im Skript werden die Daten dann auf Richtigkeit und Doppelte Einträge überprüft.

**Login:** Die Login Seite wird sehr wahrscheinlich fast identisch zur Registrierung Seite aussehen. Die Daten werden genauso über eine Form an den Server weitergegeben und über PHP verarbeitet. Hier wird jedoch nichts in die Tabellen geschrieben. Wichtig ist hier aber das wie bei der Registrierung eine Session in PHP gestartet wird und beim Logout natürlich wieder beendet wird.

**To-Do erstellen:** Die Erstellung eines To-Do’s soll ebenfalls über eine Form im besten Falle über ein HTML-Form entstehen hier wird fast keine Überprüfung der Daten gebraucht da es möglich sein kann das der Nutzer zwei sehr ähnliche To-Do’s anlegen kann. Wichtig ist jedoch die Zuweisung des Projekts und Teams indem das To-Do gerade angelegt wurde.

Schnittstellen

Haben wir bisher keine

Datenanforderungen

Userdaten, Teamdaten, Projektdaten und To-Dodaten müssen alle erfasst, verarbeitet und gespeichert werden. Eine Erweiterung wäre dann auch die Historie von To-Do’s anzusehen. Um Änderungen in Projekten

## Nicht-funktionale Anforderungen

**Performance:** Erstellen, bearbeiten und löschen von To-Do’s soll möglichst schnell geschehen und so wenig Ladezeit wie möglich beanspruchen. Das Laden von Teams und deren Projekte genau so wie das Laden von Projekten und deren To-Do’s mit geringen Ladezeiten geschehen.

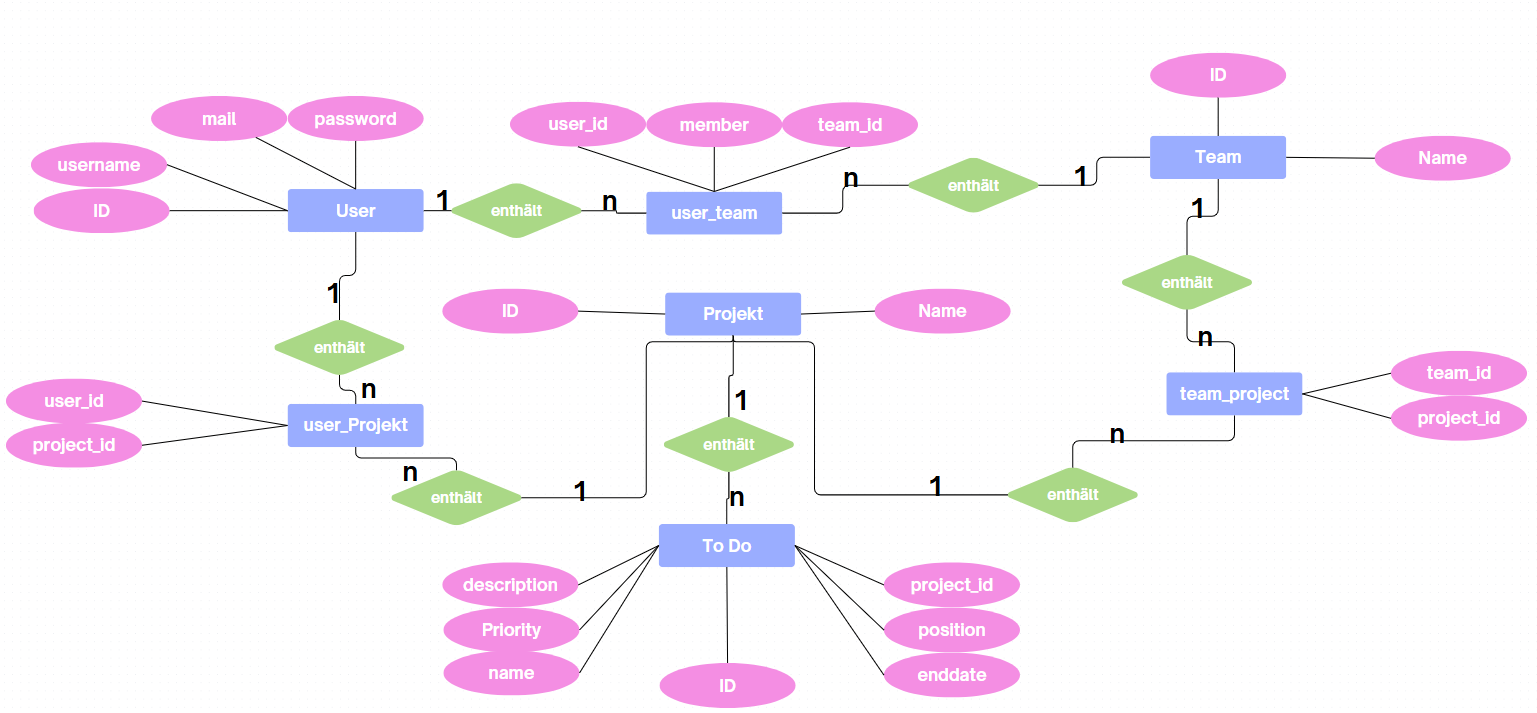
**Sicherheit:** Sicherheit beginnt mit simplen Dingen wie der Login um auf seine Daten zuzugreifen zu können, das hashen von Passwort beim Registrieren und Einloggen. Sicherheit bei Webanfragen können über die Weiterleitung von http anfragen auf https und SSL-Zertifikate und   
SSL-Zertifikatverifizierung erlangt werden.

**Verfügbarkeit:** Das System soll eine Erreichbarkeit von 99,9% gewährleisten. Dies entspricht einer maximalen Ausfallzeit von 8,76 Stunden pro Jahr.

**Usability:** Die Benutzeroberfläche soll einfach zu verstehen sein und nicht zu Überladen wirken. Es soll eine klare Aufteilung geben zwischen Team, Projekt und den To-Do-Listen. Es soll immer lesbar sein in welchem Team und Projekt man sich befindet.

**Skalierbarkeit:** Die Skalierbarkeit von Teams und Projekten ist wichtig, ein Team soll viele User und Projekte haben können.

## Daten Modell

Er Diagramm

## Datenbank Design

Tabelle User:

User (ID, Username, Mail, Password)

|  |  |
| --- | --- |
| Felder | Datentypen |
| ID | INTEGER |
| Username | VARCHAR |
| Mail | VARCHAR |
| Password | VARCHAR |

Tabelle Team:

Team: (ID, Name)

|  |  |
| --- | --- |
| Felder | Datentypen |
| ID | INTEGER |
| Name | VARCHAR |

Tabellen Projekt:  
Projekt (ID, Name)

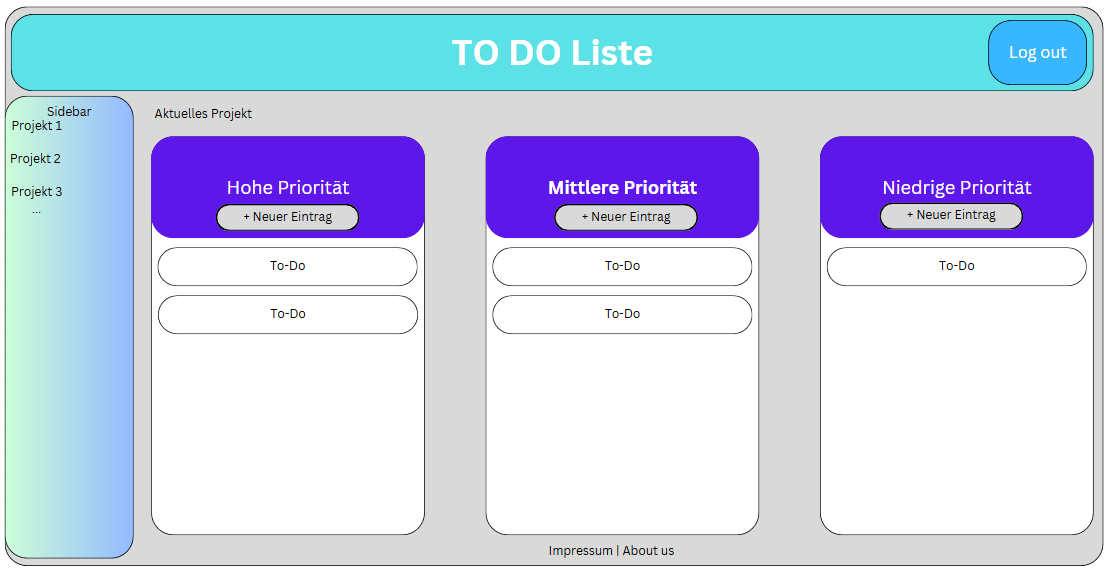
|  |  |
| --- | --- |
| Felder | Datentyp |
| ID | INTEGER |
| Name | VARCHAR |

Tabelle To-Do:

To-Do (ID, Priority, Name, Description, Enddate, Position, Project\_ID)

|  |  |
| --- | --- |
| Felder | Datentypen |
| ID | INTEGER |
| Priority | INTGER |
| Name | Text |
| Description | Text |
| Enddate | Date |
| Position | INTEGER |
| Project\_ID | REFERENCE project(id) |

## Benutzeroberfläche (UI/UX)

Skizze der Benutzeroberfläche****

Navigationskonzept  
 Die Seite ist in drei wesentliche Hauptelemente aufgeteilt.

* Der Header ist für die Globale Navigation verantwortlich hier könnten noch Buttons für andere Funktionen eingefügt werden.
* Im Hauptbereich werden die To-Do-Listen angezeigt hier passiert die Hauptaufgaben der Webseite. Außerdem befindet sich hier auch noch ein Footer mit Verlinkungen zum Impressum und anderen Seiten wie eine About us Seite
* Das dritte Element ist die Sidebar, diese ist dafür da zwischen den Projekten zu wechseln. Eine Idee ist es der Sidebar eine Ein- und Ausklappfunktion zu geben

Responsive Design:  
Das To-Do-Listen-Projekt sollte ein responsives Design haben, das sich flexibel an verschiedene Bildschirmgrößen anpasst. Dabei müssen Bedienelemente gut erreichbar und klar lesbar sein, mit ausreichendem Abstand zwischen den interaktiven Elementen. Für Barrierefreiheit sollte eine gute Farbkontrastwahrnehmung bieten und Tastaturnavigation ermöglichen.

## Systemarchitektur

Client-Server-Modell

Das Projekt ist als drei-Schichten-Architektur aufgebaut. Dies verbessert Struktur, die mögliche Wartungsarbeiten und die Erweiterbarkeit des Systems.

**Das Frontend** stellt die GUI dar und ist für die Interaktion mit Nutzern verantwortlich. Dazu gehört unter anderem das Anzeigen von To-Do-Listen und das Empfangen und Darstellen von Daten aus der Datenbank über die API.

**Das Backend** dient als Vermittler zwischen Frontend und Datenbank. Es ist für die Verarbeitung von Abfragen des Frontendes sowie die Validierung von Benutzereingaben verantwortlich.

**Die Datenbank** speichert alle persistenten Daten, die für das System benutzt werden. Als Datenbank haben wir vor einen Anbieter Namens Turso zu verwenden. Als Backup falls es Probleme geben sollte würden wir auf SQLite umsteigen.

Technologie-Stack

**Frontend**

|  |  |
| --- | --- |
| Technologie | Erklärung |
| HTML | Für die Struktur der Benutzeroberfläche |
| CSS | Zur Gestaltung und Anpassung des Designs |
| JavaScript | Für alle Funktionen die auf Client- Seite ausgeführt werden. |
| Drag and Drop API | Ermöglicht das Verschieben von To-Do’s untereinander und zwischen Prioritätsgruppen |

**Backend**

|  |  |
| --- | --- |
| PHP | Zur serverseitigen Verarbeitung von Benutzereingaben, Bereitstellung von Daten und Anbindung an die Datenbank |

**Datenbank**

|  |  |
| --- | --- |
| Turso | Ein externen Datenbanken Anbieter zur Speicherung der Daten |
| SQLite (Backup) | Eine leichtgewichtige Datenbank für die Speicherung der Daten |

## Test- und Abnahmekriterien

Testfälle

Wichtige Dinge die getestet werden müssen ist das Überprüfen von korrekten Tabellen INSERTS, SELECTS und DELETS, dazu gehören die auch die Funktionen im Frontend die diese auslösen. Besonders wichtig wäre das Registrieren und Einloggen sowie Erstellen von To-Do’s da diese zu den Grundfunktionen des Projektes gehören

Abnahmekriterien

Das System kann nur übergeben werden, wenn alle Grundfunktionen problemlos ablaufen. Außerdem sollte das Erstellen von Teams und Projekten ebenfalls funktionieren.

## Zeit und Ressourcenplan

Meilensteine

|  |  |
| --- | --- |
| Meilenstein | Deadlines |
| Grundlegendes Design | 07.10.2024 |
| Sidebar und Drag and Drop | 17.10.2024 |
| Erstellung von Einträgen | 01. 11.2024 |
| Datenbank Anbindung von PHP | 07.11.2024 |
| Registrieren und Einloggen | 21.11.2024 |
| Erstellen von Teams und Projekten | 05.12.2024 |
| Design für mobile Geräte anpassen | 15.12.2024 |

Ressourcenplan:

|  |  |
| --- | --- |
| Ressourcenart | Übersicht |
| Hardware | Laptops und/oder Desktops für die Entwickler |
| Software | Visual Studio Code |
| Personal | 2 Entwickler |

## Budget (nur Simulation mäßig)

Kostenübersicht und Ressourcenallokation

|  |  |
| --- | --- |
| Entwicklung | 8.000,00€ |
| Test | 1.200,00€ |
| Deployment | 400,00€ |
| Gesamtkosten | 9.600,00€ |