Ein Bild, das Uhr, Kreis, Symbol, Wanduhr enthält.

Automatisch generierte Beschreibung

Hermes

Das bessere ZEUS

Von Niel Müller

Lehrender des MGB

# Inhaltsverzeichnis

Inhaltsverzeichnis 2

Einleitung: 2

Vorwort: 2

Projekt Idee 2

Planung 4

Projektantrag 4

Konzept: 5

Grundidee: 5

Use Cases: 5

Risiken 6

Technologien 6

Neue Dinge 6

Zeitplanung 6

Testfälle 7

# Einleitung:

## Vorwort:

Schon bald ist das erste Lehrjahr und die Zeit im ZLI vorbei. Dabei gibt es am Ende des Jahres noch ein Abschlussprojekt. Im Gegensatz zu den Ferienprojekten ist dieses Projekt eine Einzelarbeit. Auch wird von einem erwartet, dass man einen Projektantrag mache und all dies auch dokumentiert. Das Projekt besteht aus der Planung, der Umsetzung und der Finalisierung. Dafür haben wir für all das nur wenige Wochen Zeit.

## Projekt Idee

Im ZLI wird sich immer wieder über das nicht besonders benutzerfreundliche Zeus beschwert. Es ist umständlich und das CSS kommt auch aus dem letzten Jahrhundert. Darum kam immer wieder die Idee ein neuer Zeus zu machen. Diese Idee habe ich in meine Projekt Idee umgewandelt. Da es etwas mit Zeus zu tun haben soll aber der Name Zeus 2 langweilig klingt, habe ich mich dafür entschieden eine andere griechische Gottheit zu nehmen. Da habe ich mich dann für Hermes entschieden, weil mir der Name gefallen hat.

# Planung

## Projektantrag

|  |  |  |
| --- | --- | --- |
| Projekttitel | | |
| Hermes (kann sich noch ändern) | | |
| **Projektleiter (Name, E-Mail)** | **Coach (Name, E-Mail)** | |
| Niel Müller [lehreniel@gmail.com](mailto:lehreniel@gmail.com) | Fabian Dennler fabian.dennler@zli.ch | |
| **Projektstart** | **Projektende** | |
| 07.06.2023 | 06.07.2023 | |
| **Idee / Kurzbeschreibung** (Um was geht es bei diesem Projekt?) | | |
| Ich will ein Zeus machen nur benutzerfrendlicher | | |
| **Ausgangslage** (Was ist los? Warum wird der Projektauftrag gestellt?) | | |
| Nichts los | | |
| **Projektziele** (Erwartete Resultate. Forderungen und Wünsche) | | **Prior.1** |
| Man kann ein Team erstellen | | A |
| Man kann Benutzer zu dem Team hinzufügen | | A |
| Man kann die Zeit erfassen | | A |
| Man hat eine Arbeitszeit | | A |
| **Vorgehensziele** (Forderungen und Rahmenbedingungen) | | **Prior.1** |
| Ich produziere clean code (ich versuchs zumindest) | | B |
| Ich mache regelmässig commits | | A |
| **Lernziele** (Welche Kompetenzen werden angestrebt?) | | **Prior.1** |
| Das ich lerne wie man programmiert | | A |
| Das ich lerne wie man clean code macht | | C |
| **Ressourcen** (Personal, Sachmittel, Material?) | | **Menge** |
| PC | | 1 |
| Niel | | 1 |
| **Projektentscheidung** (Freigabeunterschrift) | | **Datum** |
| Freigabe durch: | |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 1) A = Must have | B = Starkes Wunschziel, wenn irgendwie möglich | C = “Nice to have“ |  |

## Konzept:

### Grundidee:

Die Idee ist ein Zeiterfassungs Tool zu machen. Dabei sollte es 2 Bereiche geben.

* Den Manage Bereich indem nur der Admin User Zugriff, hat. Von dem gibt es bei jedem Team nur einen. Er kann in der Manage Page neue User erstellen und diese einem Team zuweisen. Außerdem kann er die Passwörter die Arbeitszeit und die Ferien Tage festlegen. Dies sollte er über eine Konfiguration machen können. Dann sollte man die Konfigurationen und User verbinden können. Somit kann man eine Hierarchie erstellen können mit dem Verbinden von Feldern
* Den User Teil: Der User kann sich mit den Daten, die definiert wurden. Einloggen. Da kann er dann seine Arbeitszeit seine Ferien und seine Arzttermine eintragen. Alles ausser Arbeit müssen vom Vorgesetzten genehmigt werden und wird dann erst in der Zeitberechnung berücksichtigt.

### Use Cases:

Admin:

* Er kann sich einloggen
* Er kann sich ausloggen
* Er kann eine neue Person erstellen
* Er kann den Namen einer Person ändern
* Er kann das Passwort eine Person ändern
* Er kann eine Konfiguration erstellen
* Er kann den Namen der Konfiguration ändern
* Er kann die Arbeitsstunden der Konfiguration ändern
* Er kann die Ferien Anzahl der Konfiguration ändern
* Er kann Personen einer Konfiguration zuweisen
* Er kann Konfigurationen einer Person zuweisen
* Er kann seine Änderungen speichern

User:

* Er kann sich einloggen
* Er kann sich ausloggen
* Es sieht seine Arbeitszeit
* Er sieht seine Ferien Zeit, die er noch hat
* Er sieht seine Ferien Zeit, die er noch nicht eingetragen hat
* Er sieht eine Übersicht wann er gearbeitet hat
* Er kann einen Eintrag erstellen
* Er sieht was für Einträge genehmigt werden müssen
* Er kann Einträge genehmigen
* Er kann einen Eintrag bearbeiten
* Er kann einen Eintrag löschen

### Risiken

* Es wird nicht in der Zeit fertig
* Es ist zu viel Bugs, um es zu nutzen
* Es gibt während der Entwicklung einen Logikfehler und man muss von vorne Anfangen
* Es wird zu kompliziert
* Zu viele Sicherheitslücken
* Zu kompliziert zu bedienen

### Technologien

* Java Scirpt
* HTML
* CSS
* Node.js
* SQL
* SQLite
* Express
* VSCode
* Firefox
* DB-Browser for SQLite
* Lodash

### Neue Dinge

* Lodash
* Vertieftes SQL
* Clean Code

### Zeitplanung

1. Tag
   1. Alle HTML-Files erstellen und im Backend verlinken
   2. Login und Register Page machen mit Backend
   3. Manage Page design machen
2. Tag
   1. Manage Frontend mit Lodash realisieren
3. Tag
   1. Manage Speichern programmieren
   2. Manage Laden programmieren
   3. Im Backend von Manage Daten in User Daten umwandeln
4. Tag
   1. User Frontend realisieren
   2. User Frontend Tabelle erstellen
5. Tag
   1. User Frontend Einträge darstellen
6. Tag
   1. Puffer für unvorhergesehenen
7. Tag
   1. User Frontend Einträge erstellen
   2. User Backend Einträge speichern
8. Tag
   1. Genehmigungen erstellen

# Testfälle

Test Fall 1

|  |  |
| --- | --- |
| **Testfall Nr.** | 1 |
| Beschreibung | Ein neues Team erstellen |
| Voraussetzung | Auf den index Seite |
| Testschritte | 1. Team Name eingeben 2. Username eingeben 3. Passwort eingeben 4. Auf Register als Admin drücken |
| Erwartetes Ergebnis | Man kommt auf die Login Page |
| Ergebnis |  |
| Testdatum |  |
| Bemerkung |  |

|  |  |
| --- | --- |
| **Testfall Nr.** | 2 |
| Beschreibung | Login als Admin |
| Voraussetzung | Auf der Login Seite sein |
| Testschritte | 1. Team Name eingeben 2. Username eingeben 3. Passwort eingeben 4. Auf Login als Admin drücken |
| Erwartetes Ergebnis | Man kommt auf die Manage Page |
| Ergebnis |  |
| Testdatum |  |
| Bemerkung |  |

|  |  |
| --- | --- |
| **Testfall Nr.** | 3 |
| Beschreibung | Einen neuen User erstellen |
| Voraussetzung | Man ist eingeloggt auf der Manage Seit |
| Testschritte | 1. Drückt auf den neue Person button 2. Gibt einen Username ein 3. Gibt ein Passwort ein oder drückt Enter 4. Drückt auf Abbrechen |
| Erwartetes Ergebnis | Es erscheint ein neuer Text mit dem Namen des Users |
| Ergebnis |  |
| Testdatum |  |
| Bemerkung |  |

|  |  |
| --- | --- |
| **Testfall Nr.** | 4 |
| Beschreibung | Configuration erstellen |
| Voraussetzung | Man ist eingeloggt auf der Manage Seit |
| Testschritte | 1. Drückt auf den neue Config button 2. Gib die nötigen Daten in Form ein 3. Drücke auf Speichern |
| Erwartetes Ergebnis | Es kommt ein neuer Text mit dem Namen der Configuration und es ist Grün |
| Ergebnis |  |
| Testdatum |  |
| Bemerkung |  |

|  |  |
| --- | --- |
| **Testfall Nr.** | 5 |
| Beschreibung | Buchung Eintragen |
| Voraussetzung | Man ist eingeloggt auf der view Seit |
| Testschritte | 1. Auf den button Eintragen klicken 2. Die Daten im From ausfüllen 3. Auf Speichern drücken |
| Erwartetes Ergebnis | Man sieht den Eintrag dann danach im Kalender |
| Ergebnis |  |
| Testdatum |  |
| Bemerkung |  |

|  |  |
| --- | --- |
| **Testfall Nr.** | 6 |
| Beschreibung | Genehmigungen anschauen |
| Voraussetzung | Man ist eingeloggt auf der View Seit |
| Testschritte | Auf Anfragen drücken |
| Erwartetes Ergebnis | Man sieht die Anfragen mit Datum, User und Typ |
| Ergebnis |  |
| Testdatum |  |
| Bemerkung |  |

|  |  |
| --- | --- |
| **Testfall Nr.** | 5 |
| Beschreibung | Anfrage genehmigen |
| Voraussetzung | Man ist eingeloggt auf der Manage Seit und hat offene Anfragen |
| Testschritte | 1. Auf Anfrage drücken 2. Auf Annehmen drücken |
| Erwartetes Ergebnis | Die Anfrage ist genehmigt und die Anfrage verschwindet |
| Ergebnis |  |
| Testdatum |  |
| Bemerkung |  |

# Code

## Server.js (backend)

### Endpoints abfangen

app.get("/", function (req, res) {

  res.sendFile(\_\_dirname + "/public/index.html");

});

app.get("/login", function (req, res) {

  res.sendFile(\_\_dirname + "/public/login.html");

});

app.get("/view", function (req, res) {

  res.sendFile(\_\_dirname + "/public/view.html");

});

app.get("/manage", function (req, res) {

  res.sendFile(\_\_dirname + "/public/manage.html");

});

Dieser Code definiert die Endpoints für das HTML. Dabei werden die verschiedenen GET request abgefangen und es wird ein file zurückgeschickt.

app.post("/getSandbox", async function (req, res) {

  let result = await handle\_getSandbox(req);

  res.send(JSON.stringify(result));

})

app.post("/getWeekData", async function (req, res) {

  let result = await handle\_getWeekData(req);

  res.send(result);

})

app.post("/updateBooking", async function (req, res) {

  let result = await handle\_updateBooking(req);

  res.send(result);

})

app.post("/createBooking", async function (req, res) {

  let result = await handle\_createBooking(req);

  res.send(result);

})

Dieser Code fängt die POST End Points ab. Dabei wird eine function aufgerufen und die beiden Variabeln req, res definiert. Diese beinhalten die Infos für die request und die response. Dann wird die Variable result definiert. Diese beinhaltet das Ergebnis der handle function. Dieses result wird dann an das Frontend zurückgeschickt.

### Handle Endpoints