TH Lübeck

Medieninformatik B.Sc. Online

Web-Programmierung WS20/21

Dozentin: Fr. Dorina Gumm

Projektbericht

Gruppe 15

AUFGABENPLANER

TASKERIZR

Tom Gibson: 314576

Felix Hansmann: 271842

Frederic Karliczek: 332820

Marc Andre Fischer: 244031

**Inhaltsverzeichnis**

Inhalt

[1 Einleitung 3](#_Toc63024520)

[1.1 Die Web-Anwendung 3](#_Toc63024521)

[1.2 Die Zielgruppe 3](#_Toc63024522)

[1.3 Technische Daten 3](#_Toc63024523)

[2 Konzeption 4](#_Toc63024524)

[2.1 Der Schwerpunkt der Anwendung 4](#_Toc63024525)

[2.2 Das Datenmodell 4](#_Toc63024526)

[2.3 Das Layout der Anwendung 5](#_Toc63024527)

[2.3.1 Die Startseite der Anwendung 5](#_Toc63024528)

[2.3.2 Der Wireframe für die Settings 6](#_Toc63024529)

[2.3.3 Der Wireframe für das Hinzufügen von Projekten 6](#_Toc63024530)

[3 Umsetzung 8](#_Toc63024531)

[3.1 Die Organisation des Quellcodes 8](#_Toc63024532)

[3.2 Umsetzung der Anforderungen 9](#_Toc63024533)

[3.2.1 Allgemeine Anforderungen 9](#_Toc63024534)

[3.2.2 Zusätzliche Anforderungen 9](#_Toc63024535)

[3.3 Erläutern von Beispielcode 10](#_Toc63024536)

[3.3.1 HTML 10](#_Toc63024537)

[3.3.2 JavaScript 10](#_Toc63024538)

[3.4 Besonderheiten der Anwendung und Herausforderungen 13](#_Toc63024539)

[3.4.1 Besonderheiten der Anwendung 13](#_Toc63024540)

[3.4.2 Welche Lerneffekte gibt es 13](#_Toc63024541)

[3.5 Abweichungen von der ursprünglichen Planung 16](#_Toc63024542)

[4 Die Gruppenarbeit 17](#_Toc63024543)

[4.1 Aufteilung und Organisation der Gruppenarbeit 17](#_Toc63024544)

[4.2 Das Vorwissen der Gruppenmitglieder 18](#_Toc63024545)

[4.3 Wichtige Lernergebnisse und Erkenntnisse 19](#_Toc63024546)

# Einleitung

## Die Web-Anwendung

Die Anwendung soll dem Nutzer ermöglichen, seine anstehenden Aufgaben zu organisieren. Auf dem Dashboard hat er eine Übersicht, welche Aufgaben/Termine noch offen sind, welche Termine an dem Tag, an dem die Seite aufgerufen wird, anstehen, und welche bereits erledigt sind.

Auf der Projektseite kann der User neue Projekte anlegen. Anschließend können zu den Projekten Aufgaben angelegt werden. Im weiteren Verlauf können diese auch verändert werden, und der User kann sie als erledigt markieren bzw. löschen.

Die Kalenderansicht ist dazu gedacht, dem Nutzer einen einfachen Kalender bereitzustellen, ohne dass dieser die Anwendung verlassen muss. Gleichzeitig werden ihm für den jeweiligen Tag fällige Termine angezeigt.

In den Einstellungen kann der User auswählen, ob er ein helles oder ein dunkles Theme haben möchte.

## Die Zielgruppe

Unsere Zielgruppe sind Computeranwender, sowohl im privaten, als auch im beruflichen Bereich. Sie notieren sich viel, müssen sich viele Termine merken und gehen sehr organisiert vor. Die Anwendung ist optimal für Menschen geeignet, die die selbstständig arbeiten, und für die Selbstmanagement ein wichtiger Teil der Organisation ist.

Sie möchten alles an einem Ort gespeichert haben. Durch den Offlinespeicher kann die Anwendung auch bei schlechtem Internet verwendet werden. Genauso kann ein Geschäftsmann, der auf seinem Laptop kein Internet zur Verfügung hat, auf seinen Planer zugreifen. Es wurde bei der gesamten Planung der Anwendung darauf geachtet, die App für so viele Personen wie möglich sinnvoll nutzbar zu machen.

## Technische Daten

Die Anwendung befindet sich auf dem Web-Space unter folgender Adresse:

<https://gruppe-15.wp20.mylab.th-luebeck.de>

Zum Entwickeln wurde der Browser Chrome verwendet. Auf diesen ist der gesamte HTML-, CSS- und JavaScript-Code auch optimiert.

# Konzeption

## Der Schwerpunkt der Anwendung

Mit der Anwendung soll ein User verschiedene Projekte anlegen können. Ein Projekt ist dabei zu sehen wie ein Thema. Das kann ein beliebiges Thema aus dem privaten oder beruflichen Bereich sein. Im privaten Bereich wären Projekte möglich wie: Planung eines Urlaubs, Organisieren einer Familienfeier oder ähnliches.

Anlegen und Organisieren von Aufgaben. Im weiteren Verlauf können die Aufgaben auch bearbeitet oder gelöscht werden. Eine Statistik der Aufgaben wird auf dem Dashboard angezeigt.

Im Headerbereich findet sich eine Uhr. Je nach Tageszeit, wird der User mit einem anderen Gruß willkommen geheißen. Beim Drücken auf den Button, wird ein zufälliges Zitat angezeigt.

Im Kalender kann man sich zu einem Tag die jeweiligen Aufgaben anzeigen lassen, die an diesem Tag anliegen.

## Das Datenmodell

**IndexedDB**

Zur Datenhaltung wird die IndexedDB verwendet. Dazu dienen mehrere ObjectStores als „Tabellen“. Ein ObjectStore mit dem Namen „Projects“ (**database.js, Zeile 120**) dient zum Speichern der Projekte.

Der ObjectStore „Tasks“ (**database.js, Zeile 131**) dient zur Datenhaltung der Aufgaben. Dabei gehört jede Aufgabe zu einem spezifischen Projekt.

Ein weiterer ObjectStore „Settings“ (**database.js, Zeile 147**) wird zum Speichern der Einstellungen (des jeweiligen Themes) verwendet. Für alle drei Objektstrukturen existiert jeweils eine Klasse, mit der Objekte erstellt werden können. Durch das Speichern in Arrays können die aus der IndexedDB gelesenen Daten mehrmals pro Seite verwendet werden, und müssen nicht jedes Mal neu geladen werden.

**Die Daten für die Zitate**

Die Zitate stammen von der Webseite git.githubuser.com. Sie werden mit Hilfe der Ajax-Technologie eingelesen.

Das Ganze findet in der Datei ajaxQuotes.js statt. Anschließend wird mit einer Zufallszahl ein Zitat herausgefiltert. Bei jedem Klicken auf den Button „Nächstes Zitat“, wird zufällig ein neues Zitat von dieser externen Webseite geladen und angezeigt.

## Das Layout der Anwendung

### Die Startseite der Anwendung

Wie unter Punkt 4.3 „Wichtige Lernergebnisse / Erkenntnisse“ nochmals erwähnt wird, wurde das Design der Wireframes während der Umsetzung mehrmals angepasst. Das Dashboard, aber auch die Settings und die Navigation an sich, wurden überarbeitet.

Dadurch konnte erreicht werden, dass die Webseite den Look einer Applikation bekommt. Für den Nutzer ergibt sich ein besseres Nutzererlebnis, weil er bei dieser Art von Webseite die Steuerung auf der linken Seite erwarten würde. Außerdem hat die Navigation auf der linken Seite durch diese Änderung eine zweite Navigationsebene bekommen, was als Anforderung gefragt war.

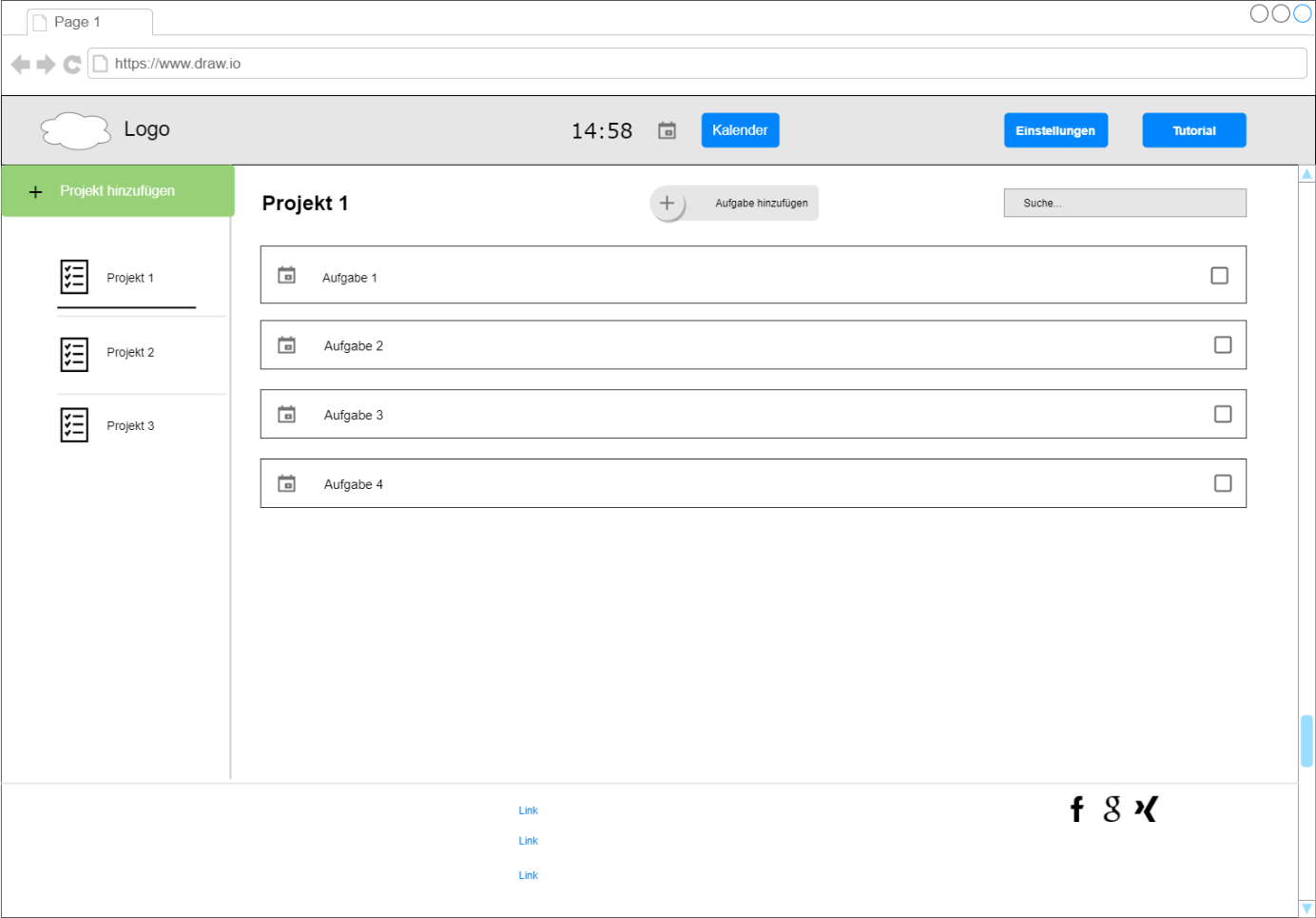


Abbildung 1: Das erste Wireframe, noch mit der Menüleiste im Header-Bereich

Für die Wireframes haben sich die Gruppenmitglieder auf das Tool draw.io geeinigt. Der große Vorteil bei diesem Tool ist, das es entweder als Webapp direkt im Browser genutzt werden kann oder als Desktop-App heruntergeladen werden kann. Außerdem bietet draw.io eine große Palette an Elementen. Neben allgemeinen Formen und deren Erweiterungen bietet dieses Tool auch spezielle Elemente für Webanwendungen, Android-Apps und vieles mehr. Fertige Dateien konnten auf dem Computer gespeichert werden. Sie können in die Webanwendung geladen werden oder auf dem Desktop verwendet werden.

### Der Wireframe für die Settings

Auch bei den Settings hat sich im Gegensatz zu den Wireframes noch eine Änderung ergeben. Anstatt das Menü über ein Overlay anzeigen zu lassen, wurde die Ansicht auf einer eigenen HTML-Seite (settings.html) erstellt.

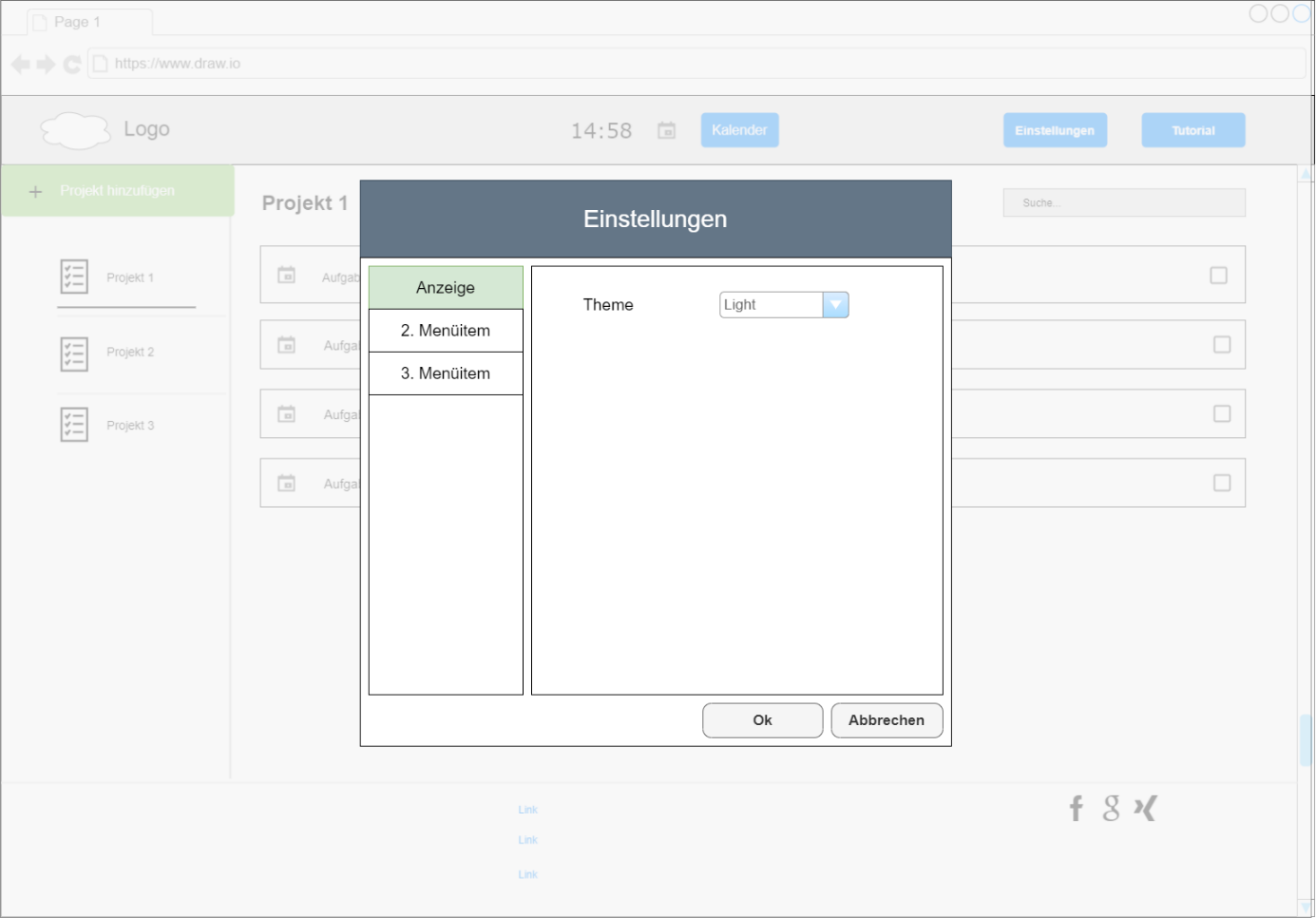


Abbildung 2: Auch bei der Ansicht der Settings hat sich im Nachhinein noch einiges geändert

Was sich aber auf den Wireframes schon gut erkennen lässt, und während der gesamten Entwicklung auch nicht mehr geändert wurde, war die Gestaltung mit großen Flächen, das Arbeiten mit dem „Flat-Design“ sowie viel Weißraum.

Die Gruppe hat sich dabei immer wieder kreative Designs einfallen lassen, haben sich aber stillschweigend immer an die Design-Vorgaben gehalten. So hat die Webseite ein einheitliches Bild erhalten.

### Der Wireframe für das Hinzufügen von Projekten

Bei manchen Anwendungen ist zu sehen, dass der Anwender beim Absenden eines Formulars auf eine neue Seite geleitet wird. Dann wird ein Skript aufgerufen, um die Daten in die Datenbank zu speichern. Anschließend erfolgt eine Weiterleitung entweder auf eine neue Seite oder zurück zum Formular. Für dieses Projekt, so war sich die Gruppe einig, ist es besser, wenn sich das Formular als Overlay öffnet. Erstens weiß der User so besser, wo er sich gerade befindet. Zweitens kann er das Fenster jederzeit, auch ohne das Formular abzuschicken, wieder schließen. Er befindet sich dann immer noch auf der vorherigen Seite und muss nicht erst umständlich zurück navigieren.

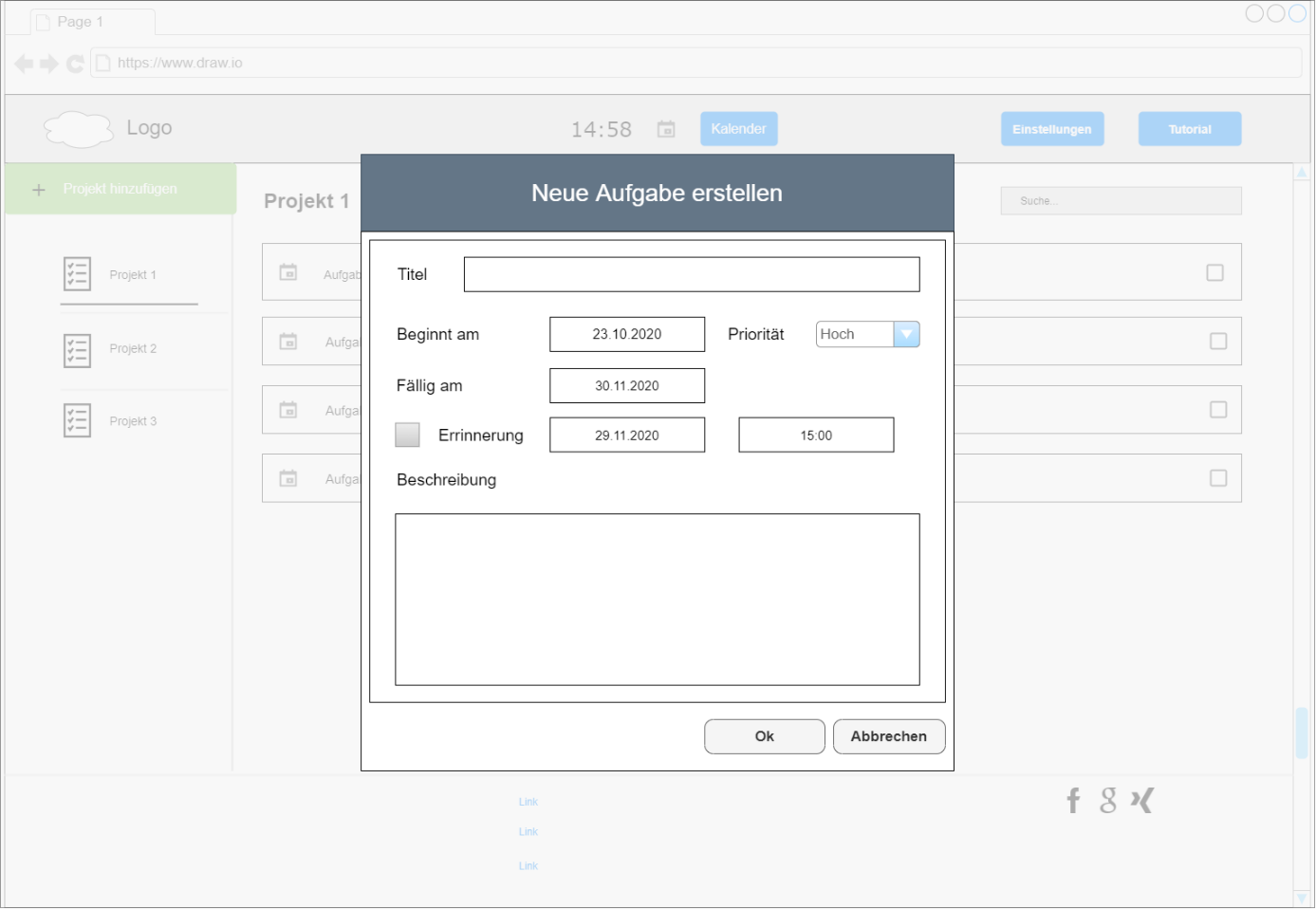


Abbildung 3: Beim Klicken eines Projektes öffnet sich ein Overlay

Das Projekt wurde in die Datenbank eingetragen (die Methoden der Datei database.js wird in „3.2.2 JavaScript“ noch genauer erklärt. Da es sofort auch wieder aus der Datenbank geholt wird, wird es sofort angezeigt. Der Nutzer weiß also, dass das Hinzufügen erfolgreich war.

Möchte der Nutzer weitere Aufgaben anlegen – angenommen er möchte gleichzeitig 5 Aufgaben für ein Projekt anlegen – geht das auf diese Weise deutlich einfacher, als wenn er von einer anderen Seite wieder zurücknavigieren müsste.

# Umsetzung

## Die Organisation des Quellcodes

**HTML**

Der Quellcode des HTML verteilt sich auf 4 Seiten. Dabei gibt es die Seiten (index.html, project.html, calendar.html und settings.html). Die weitere Seite table.html, auf der sich die Tabelle aus Einsendeaufgabe 1b c) befindet, wird hier nicht mit aufgezählt.

Die Navigationselemente, zu denen der Header-Bereich gehört, die seitliche Navigation und der Footer, wiederholen sich auf allen Seiten. So bleibt die Seite konsistent und der User kann sich auf jeder einzelnen Seite viel besser zurechtfinden.

**CSS**

Der CSS-Code wurde in mehrere Dateien ausgelagert. Damit muss nur das CSS geladen werden, was für die aktuelle Seite gerade benötigt wird. Bei der Benennung der Dateien wurde auf aussagekräftige Namen geachtet, damit Erweiterungen oder Änderungen einfacher durchgeführt werden können.

Nur das Styling, das auf allen Seiten zum Einsatz kommt, befinden sich in der Datei „style.css“. Deshalb befinden sich dort die Anweisungen für die Bereiche Header, Seitennavigation und Footer.

Alle CSS-Dateien befinden sich in dem Ordner css. Wenn Anpassungen vorgenommen werden sollen, weiß jeder sofort, wo er suchen soll. Außerdem befinden sich alle CSS-Files an einem Ort. Zudem kann ein Ordner in den meisten Editoren in der seitlichen Projektstruktur zugeklappt werden. Dadurch erreicht man eine deutlich bessere Übersichtlichkeit.

In der Datei themes.css werden die Farben über Variablen in CSS-Klassen gespeichert, um die Design-Gestaltung zu vereinfachen.

**JavaScript**

Auch das JavaScript wurde in verschiedene Dateien aufgeteilt. Die Benennung der Dateien deutet auf die jeweilige Funktionalität hin, die innerhalb der Datei implementiert ist. Für die Scripts wurde ebenfalls ein separater Ordner angelegt.

Eines der wichtigsten Skripte ist die Datei „database.js“. Mit Hilfe dieser Datei wird mit der IndexedDB kommuniziert. Zuerst wurde überlegt, die verschiedenen Themen (Projects, Tasks und Settings) in verschiedene Dateien zu packen. Daraus hätten sich aber möglicherweise wieder schwer auffindbare Abhängigkeiten ergeben. Deshalb wurde sich schlussendlich dafür entschieden, die gesamte Kommunikation mit der Datenbank zentral in einer Datei zu implementieren. Diese Implementierung hat sich im Nachhinein als sehr wartungsfreundlich herausgestellt.

## Umsetzung der Anforderungen

### Allgemeine Anforderungen

Die Anforderungen an den Scope wurden erfüllt. Die Anwendung hat vier Ansichten. Dazu gehören die Seiten index.html, calendar.html, project.html und settings.html. Alle vier Seiten sind über Links miteinander verbunden. Der Vollständigkeit halber wurde die Datei tabelle.html, in der die Übung mit der Tabelle gemacht wurde, ebenfalls verlinkt. Sie ist also Bestandteil der Anwendung.

Die Anwendung besteht aus den Bereichen Header, Footer und auf der linken Seite über ein Navigationsmenü. Um die Konsistenz zu wahren, befinden sich die Elemente auf allen Seiten am gleichen Platz.

Wo immer möglich und wo es Sinn gemacht hat, wurden zur Strukturierung sogenannter „Sectioning-Content“ verwendet. Auf der Seite index.html findet sich in den **Zeilen 20-33** der Header-Bereich. Weitere Unterteilungen bilden der Navigationsbereich (**Zeilen 37-71**) und der Hauptbereich, der durch das Element main (**Zeilen 73-137**) gekennzeichnet wird. Beim Gebrauch eines div-Elementes wie <*div id=“main“>* ergibt sich keine sichtbare Änderung der Ansicht im Browser. Aber der große Vorteil ist, dass Suchmaschinen und Bildschirmlesegeräte die neuen HTML5-Elemente unterscheiden können. Für Sie ergibt sich eine semantische Struktur, die alleine durch die Vergabe von Id’s nicht entsteht.

Einige Elemente tauchen nur einmal pro Seite auf – auch wenn sie theoretisch öfters vorkommen können. Aber mit den Elementen header, aside, main und footer ergibt sich eine klare semantische Struktur. Andere Tags tauchen mehrmals auf der Seite auf. So z.B. section (**Zeilen 75, 78, 88**), wobei sogar mehrere section-Tags ineinander verschachtelt werden. Das article-Element kommt in **Zeile 74** vor.

Es gibt in der Anwendung mehrere Tabellen. Auf dem Dashboard (index.html ab **Zeile 90**). Die Tabelle zeigt dem User die Aufgaben an, die heute fällig sind. Eine weiter Tabelle gibt es in project.html ab **Zeile 102**. Hier werden für das jeweilige Projekt die verfügbaren Aufgaben (Tasks) aufgelistet.

Beim Anlegen einer neuen Aufgabe öffnet sich ein Overlay mit einem Formular. Neben einer Validierung, die über die Standartvalidierungsmöglichkeiten von HTML5 hinausgeht, gibt es verschiedene Interaktionselemente. Es gibt Textfelder, Date-Picker, Date-Time-Picker und ein Select-Element mit dem zwischen drei Prioritätsstufen gewählt werden kann.

Es kam in großem Umfang JavaScript zum Einsatz. Damit wurden unter Anderem folgende Funktionen umgesetzt: Die Kreisdiagramme (charts.js) auf dem Dashboard. Das Anzeigen der Uhrzeit (clock.js). Für die Zitate (ajaxQuotes.js) wurde neben reinem JavaScript auch die Technologie mit Ajax umgesetzt. Insgesamt gibt es 16 JavaScript Dateien, die zum Teil mit einander Verknüpft sind, teilweise aber auch unabhängig agieren.

### Zusätzliche Anforderungen

Des Weiteren wurden auch statische und dynamische Effekte umgesetzt, wo es Sinn ergibt. Zu den verwendeten statischen Effekten zählen: neben den Buttons hat auch der Kalender einen „Box-Shadow“. Dieser wird in **Zeile 29** von calendar.css erzeugt. Auch bei den Buttons wurden mehrere statische Effekte umgesetzt. Außerdem wurden die Ecken abgerundet. Die Effekte wurden in der style.css in **Zeile 163** und **172** umgesetzt.

Außerdem wurden dynamische Effekte hinzugefügt. Buttons und Navigationselemente ändern beim „Hovern“ ihre Hintergrundfarbe. Um den Effekt zu verstärken, wurde mit CSS-Transitions dafür gesorgt, dass die Effekte verzögert auftreten. Somit ergibt sich ein besonders weicher Übergang, was das Nutzererlebnis verbessert. Implementiert wurde das ebenfalls in der style.css in **Zeile 83** und **154**.

Die Gruppenmitglieder haben sich dazu entschieden, für das Layout der Seiten, aber auch für einzelne Abschnitte mit dem Flexbox-Layout-Modell zu arbeiten. Mit wenigen Befehlen, lassen sich Elemente sehr gut strukturieren. Wenn nicht alle Elemente den gleichen Inhalt haben, kann es zu unterschiedlichen Größenverhältnissen kommen. Auch das kann man mit Flexbox sehr gut lösen.

## Erläutern von Beispielcode

### HTML

**Rendern zur Laufzeit mit dem template-Tag**

Gelegentlich kann es vorkommen, dass nicht nur kurze Texte oder ein Element in das bestehende HTML eingefügt werden sollen, sondern möglicherweise ein übergeordnetes Element mit mehreren Kindelementen.

Mit Hilfe des template-Tags kann eine Vorlage erstellt werden. Elemente die von den Template-Tags umschlossen werden, sind zunächst nicht Bestandteil des DOM. Mit JavaScript wird hieraus ein Objekt erstellt werden und zum DOM hinzugefügt.

Erst nach dem Hinzufügen zum DOM ist es wirklich existent und kann auch im weiteren Verlauf von CSS und JavaScript aus angesprochen werden.

Das Konzept wurde in den HTML-Dateien calendar.html (**Zeile 107**) und index.html (**Zeile 100**) verwendet. In project.html sogar 2mal (**Zeilen 74** und **113**).

Die verwendeten neuen HTML5-Elemente wurden bereits im Abschnitt 3.2.1 Allgemeine Anforderungen erläutert.

### JavaScript

Von Java ist bekannt das zumindest jede öffentliche Klasse in eine eigene Datei geschrieben werden muss. In JavaScript gibt es solche Konventionen nicht. Um trotzdem eine gute Aufteilung des Quellcodes zu schaffen, gibt es verschiedene Design-Pattern. Mit Hilfe dieser Design-Pattern, können Variablen, aber auch mehrere Methoden zu einem Ganzen zusammengefasst werden. Zum einen wird dadurch erreicht, dass der Code besser strukturiert wird, und bei Änderungen oder Erweiterungen fällt es deutlich leichter, diese an der richtigen Stelle zu implementieren.

Im JavaScript Quellcode wurden verschiedene Design-Pattern umgesetzt. In der Datei charts.js z.B. wurde das Konstruktor-Design-Pattern angewendet. In der Datei database.js wurde das Module-Design-Pattern verwendet. Es gibt aber auch einige Scripts, z.B. die calender.js, in der einfach verschiedene Funktionen implementiert sind. So ergibt sich ein Lerneffekt aus dem Ausprobieren unterschiedlicher Code-Strukturierungen.

**Die Datei database.js**

In der database.js finden sich alle Methoden zum Erstellen der Datenbank, der ObjectStores und zum Lesen und Schreiben in diesen. Gleich zu Anfang der Datei ab **Zeile 13** gibt es 3 Klassen. Die 3 Klassen modellieren unsere Objekte, die in den ObjectStores gespeichert werden. Daraus sollen Objekte erstellt werden können, und so gewährleistet werden, dass sie eine einheitliche Datenstruktur haben.

In **Zeile 58** beginnt die JavaScript Funktion Database. Hierfür wurde das Module-Design-Pattern verwendet. Alle Variablen und Methoden sind so gekapselt und sind nach außen nur über Schnittstellen zugreifbar.

Die Methode *getAllItemsFromObjectStore()* in **Zeile 194** ist asynchron (mehr zu asynchronen Methoden später). Mit Hilfe dieser Methode werden aus einem ObjectStore alle gespeicherten Elemente geholt. Die Methode hat einen Parameter, durch den der gewollte ObjectStore aufgerufen werden kann. Die Methode gibt ein Promise zurück. Mit diesem kann dann an anderer Stelle auf die Elemente als eine Art Liste zugegriffen werden.

In **Zeile 246** hilft die Methode *getItemsFromObectStoreByIndex()* dabei, ein einzelnes Element anhand der als Parameter übergebenen Id aus der Datenbank zu holen. Aus welchem ObjectStore das Element geholt werden soll, wird ebenfalls als Parameter übergeben.

Es sollten aber nicht nur Elemente aus dem Store gelesen werden, es sollen auch neue Daten hineingeschrieben werden. Mit den Methoden *addItemToObjectStore()* ab **Zeile 291** und addItemArrayToObjectStore() ab **Zeile 312** kann auch ein einzelnes Element bzw. eine Reihe von Elementen (mit Hilfe eines Arrays) in den ObjectStore geschrieben werden. Dabei wiederholt sich das Schema, das der ObjectStore und die Daten als Parameter übergeben werden können.

Die Methoden werden in der Datei project.js verwendet.

**Kreisdiagramme anzeigen lassen mit der Datei charts.js**

In dieser Datei kam das Konstruktor-Design-Pattern zum Einsatz. Es wurde eine Blaupause eines Chart-Objekts erstellt. Der Vorteil dieser Variante ist, dass mehrere Chart-Objekte damit erstellt werden können. In **Zeile 11** werden der Konstruktor-Methode 3 Parameter übergeben. Der erste Parameter canvasName dient dazu, später im HTML ein Canvas-Element ansprechen zu können. Als weitere Parameter dienen die Daten, die mit dem Chart visualisiert werden sollen, sowie die Farben. Mit dem Chart wird das Verhältnis der bereits abgeschlossenen Projekte/Aufgaben zu den noch Offenen dargestellt.

Hierfür werden Farben als Array übergeben. Dadurch können Charts mit unterschiedlichen Farben erzeugt werden. Ebenfalls als Parameter werden die Daten übergeben, anschießend wird mit den Methoden drawWedge2() in **Zeile 45** und der Methode drawDonut() ab **Zeile 77** für jeden Chart jeweils ein äußerer und ein innerer Kreis gezeichnet. Durch die Überlagerung entsteht der Donut Effekt. Die Methode drawWedge2() wird in der Methode drawDonut() in **Zeile 80** aufgerufen. Mit der if-Abfrage in **Zeile 50** wird sichergestellt, dass auch dann ein vollständiger Kreis angezeigt wird, wenn es noch gar keine Aufträge gibt. Die if-Abfrage in **Zeile 65** sorgt dann dafür, dass der gesamte Donut in grau dargestellt wird.

**Das Skript project.js**

In diesem Skript findet die Verarbeitung von angelegten Projekten statt. In **Zeile 58** in der Funktion createProject() wird ein Project in der Datenbank gespeichert. Der Name des Projektes wird als Parameter übergeben. **Zeile 59** überprüft den String auf einen validen Projektnamen. Nur wenn dieser den Konventionen genügt, wird das Projekt gespeichert. Zum Speicher des Projektes wird sich wieder einer Methode aus der database.js Datei bedient.

Die Methode addProject() ab **Zeile 113** nimmt sich das Template, welches für die Erzeugung von Projekten erstellt wurde, und fügt das neue Projekt in die HTML-Seite ein.

Existiert erst einmal eine Liste mit verschiedenen Projekten, soll zu jedem Projekt, welches ausgewählt wird, auch die passenden Aufgaben geladen werden. Zuerst werden alle vorher angezeigten Aufgaben gelöscht, und erst dann werden aus der Datenbank alle Aufgaben geladen, welche zu dem ausgewählten Projekt gehören.

**Wo wird eine neue Aufgabe, engl. Task angelegt?**

Dazu wird das JavaScript-File task.js näher angeschaut. In der Datei wird in **Zeile 41** mit Beginn der Methode createTask() eine neue Aufgabe angelegt. Die Werte werden aus dem Formular, welches sich als Overlay öffnet, verwendet. Der Task wir nur dann gespeichert, wenn der Titel keinen leeren String enthält.

Hat der User einen gültigen Titel eingegeben, wird der Task in der Datenbank gespeichert (**Zeile 44**). Damit er anschließend nicht nur gespeichert ist, sondern auch gleich in der Liste der erstellten Aufgaben auftaucht, müssen alle Tasks aus der Datenbank neu geladen werden. Das geschieht in **Zeile 46**. Die dazu verwendete Methode wird in ab **Zeile 60** implementiert.

Mit der Methode addTaskToTaskList() ab **Zeile 87** wird die neue Aufgabe in das HTML-Gerüst eingefügt. Weil die Tasks als Tabelle implementiert sind, werden die wichtigsten Attribute eines Tasks in die jeweiligen Tabellenzellen eingefügt (**Zeile 97-101**).

In mehreren Methoden dieses Scripts kommen Methoden von database.js zum Einsatz.

## Besonderheiten der Anwendung und Herausforderungen

### Besonderheiten der Anwendung

Eine der Besonderheiten der Anwendungen ist sicher, dass es gleich 3 verschiedene ObjectStores gibt. Diese werden alle am Anfang beim ersten Besuch der Startseite angelegt. Zwei der drei ObjectStores, nämlich „Projects“ und „Tasks“ sind direkt miteinander verbunden. Für ein Projekt können mehrere Tasks angelegt werden. Dabei kann ein Task aber immer nur zu einem Projekt gehören. Diese Verbindung herzustellen, war am Anfang ziemlich knifflig. Auf die Lösung wird unter „3.3 Erläutern von Beispielcode“ noch näher eingegangen.

### Welche Lerneffekte gibt es

**An einem kompletten Projekt arbeiten**

Alle Gruppenmitglieder waren sich einig, dass eine tolle und wichtige Erfahrung war, von der ersten Programmzeile bis zur Fertigstellung an einem eigenen Projekt zu arbeiten. Dadurch hat sich das Verständnis vor allem von JavaScript enorm verbessert. Bereits vorhandene Programmierkenntnisse, z.B. aus Java konnten mit integriert werden. Genauso interessant war es, zu sehen wie eine komplette Anwendung aufgebaut ist und wie die einzelnen Komponenten zusammenspielen.

**Organisieren von JavaScript-Dateien**

Beim Erstellen von neuen JavaScript-Dateien und vor allem beim Ausprobieren der neuen Funktionen, ist den Gruppenmitgliedern immer wieder aufgefallen, dass der geschriebene Code nicht wie gewünscht funktionierte. Codefragmente wurden in anderen JavaScript-Dateien aufgerufen.

Zunächst wurde aber nicht darauf geachtet, in welcher Reihenfolge die Skripte eingebunden werden. Nachdem bemerkt wurde, dass die Reihenfolge eine Rolle spielt, wurde diese beliebig von den Gruppenmitgliedern geändert. Aber auch das schien nicht die Lösung zu sein. Es funktionierten nämlich nach dem Tausch andere Codebausteine nicht mehr, bei denen es zuvor keine Schwierigkeiten gab.

Schließlich hat die Gruppe herausgefunden, dass beim Aufrufen von Methoden und der Verknüpfung mit der Oberfläche der Befehl

*window.onload = function()…*

nicht verwendet werden sollte. Dabei kann es zu dem Problem kommen, dass das Abfeuern eines Events von einem anderen Skript, was ebenfalls bei diesem Event zum Einsatz kommen soll, überschrieben wird. Um dieses Problem zu umgehen, wurde stattdessen auf die Möglichkeit mit eventListenern zurückgegriffen.

*window.addEventListener(‚load‘, function()…)*

fügt dem Element nur einen weiteren EventListener hinzu, aber ohne diesen zu überschreiben.

**Absprachen über die Benennung von Variablen und Methoden**

Es wurde ein Styleguide erstellt, indem festgehalten wurde, wieviel die Zeichen pro Ebene eingerückt werden sollen. Bei HTML und CSS war es ein Einzug von 2 Zeichen. Bei JavaScript einigte man sich auf 4 Zeichen. Allerdings wurde anfangs nicht über die klare Benennung von Variablen und Methoden gesprochen. Es haben sich aber stillschweigend alle Beteiligten an übliche Konventionen gehalten. Methodennamen, bis auf Module-Bezeichnungen, und Variablen beginnen mit einem Kleinbuchstaben und werden dann mit der „Camel-Case-Notation“ fortgesetzt.

Aufgrund der fehlenden Absprachen zu Beginn, wurden in einigen Dateien die Kommentare auf Deutsch, in anderen auf Englisch verfasst. Beim nächsten Projekt würde im Vorfeld auch über diese Dinge gesprochen werden.

**Den Umfang einzelner Funktionen unterschätzt**

Wie in „3.5 Abweichungen von der ursprünglichen Planung“ schon beschrieben, hat sich die Implementierung eines einfachen Timers mit den zur Verfügung stehenden Technologien als quasi nicht umsetzbar herausgestellt.

Die Mitglieder haben daraus gelernt. Eine Aufgabe die auf den ersten Blick einfach erscheint, stellt sich im Nachhinein als nur schwer lösbar oder unmöglich heraus. In diesem Fall wurde das Lösen des Timers unterschätzt.

**Asynchrones JavaScript**

Werden Daten aus der IndexedDB geholt, muss darauf geachtet werden, wie man das implementiert. Den Mitgliedern der Gruppe ist es am Anfang passiert, dass eine Methode zum Auslesen von Daten erstellt wurde, beim Aufrufen aber kein Ergebnis vorlag.

Erstellt man einfach eine Methode, und ruft dies ohne weitere Schritte an anderer Stelle auf, kann es genau zu beschriebenen Verhalten kommen. Grund dafür ist, dass es passieren kann, dass an der Stelle, an der die Methode aufgerufen wird, noch gar kein Ergebnis vorliegt.

Mit Promises wird nicht der Wert direkt zurückgegeben, sondern ein Versprechen (Promise). Diesem Versprechen werden zwei Parameter als Methoden übergeben – resolve und reject. Wenn das Versprechen eingelöst werden kann, werden mit Hilfe von resolve die Daten zurückgegeben (zusehen z.B. in der Datei database.js ab Zeile 197. Die gesamte Methode wird als async gekennzeichnet. Das bedeutet, beim Aufrufen der Methode wird die Anwendung nicht blockiert, bis ein Ergebnis vorliegt.

**Es gibt immer mehr als einen Lösungsweg**

Ebenfalls sehr interessant war, zu sehen, dass es sehr oft mehrere Lösungswege gibt. Der Umgang damit ist aus mehreren Gründen wichtig. Zum einen Gibt es für einen selbst oft mehrere Möglichkeiten. Dann muss abgewägt werden, welcher Weg der sinnvollere ist. Oftmals muss in die Überlegungen dann noch mit einfließen, dass ein bestimmter Lösungsweg an einer Stelle, bestimmte Voraussetzungen an einer anderen Stelle nach sich zieht. Dann gilt es zu prüfen, ob diese Voraussetzungen vorhanden sind.

Es kann aber auch der Fall sein, dass einem selbst nur ein Lösungsweg einfällt, während der Kollege einen anderen Lösungsansatz verfolgt. Dann gilt es, darüber zu sprechen, seine Argumente vorzubringen, und, sollte der andere Lösungsansatz tatsächlich besser sein, dies auch einzugestehen.

**Doppelter Code**

Es wurde sich zwar immer wieder abgesprochen, wer welche Methoden für welchen Funktionsumfang übernimmt. Trotzdem ist am Schluss aufgefallen, dass redundanter Code vorhanden war, was teilweise noch ausgebessert wurde. An einem Beispiel ist das recht deutlich geworden. Anstatt eine Methode zu schreiben mit der ein Overlay geöffnet wird, wurde dies anfangs in mehreren Skripten implementiert. Auch hier hätte eine noch bessere und sorgfältigere Planung wahrscheinlich zusätzliche Arbeit gespart.

Es lohnt ein anfänglicher Blick auf Funktionen, die sich im Laufe des Projekts auf verschiedenen Seiten wiederholen.

**Fremder Code**

Ein weiterer Lerneffekt hat sich durch das Zurechtfinden in fremden Code ergeben. Diesen zu lesen und zu verstehen war anfangs gar nicht einfach. Schließlich hatte man nach fast jedem „pullen“, um auf dem aktuellen Stand zu sein, mit neuem Code zu tun. Und das nicht nur in einer Datei, sondern oft in Mehreren. Und dass auch sowohl im HTML, CSS oder JavaScript. Es war nicht immer einfach, diese verschiedenen Änderungen zu verknüpfen und zu verstehen. Mit der Zeit hat sich die Gruppe aber immer besser mit dem fremden Code zurechtgefunden.

**Die Sache mit der IndexedDB**

Es gab Mitglieder in der Gruppe, die bereits etwas Erfahrung mit MySQL und Datenbanken hatten. Umso überraschender war es, dass es am Anfang doch einige Schwierigkeiten mit der IndexedDB gab. Die Funktionsweise der IndexedDB unterscheidet sich grundlegend von der Kommunikation des herkömmlichen Backends mit einer gewöhnlichen Datenbank.

Die Begriffe, wie z.B. der „ObjectStore“ waren am Anfang ungewohnt. Das holen eines DB-Objectes und das Anwenden von Funktionen auf dieses Objekt schienen am Anfang nicht selbstverständlich. Die Entwicklertools in Chrome (F12) haben da einen sehr wertvollen Dienst erwiesen.

## Abweichungen von der ursprünglichen Planung

**Nachträgliche Änderung des Seitenlayouts**

Als das erste Online-Meeting anstand, wurde nach der Themenfindung umgehend über erste Funktionen gesprochen. Wenige Tage später hatten die einzelnen Gruppenmitglieder bereits erste Wireframes erstellt.

Von der Startseite wurde auch das Wireframe in HTML und CSS implementiert. Mit der Zeit hat sich aber herausgestellt, dass das Design der Website noch optimiert werden sollte. Die Aufteilung der beiden Menüpunkte im Kopfbereich schien noch nicht ausreichend. Außerdem war es so nicht möglich, die Vorgabe einer tieferen Navigation umzusetzen. So wurde das Design überarbeitet, indem das Menü nach links an die Seite ausgelagert wurde.

Dadurch konnte unterhalb des Punktes „Projekte“ eine zweite Ebene eingefügt werden, indem alle vorhandenen Projekte aufgelistet wurden.

Außerdem sieht die Webseite so eher wie eine Applikation aus, was dem Nutzer zu Gute kommt. Die Elemente zur Bedienung sind da, wo er sie vermutet.

**Einen Timer implementieren**

Es sollte ein Timer erstellt werden, durch den der User automatisch nach 25 Minuten daran erinnert wird, dass er eine Pause einlegen sollte. Ein sogenannter Pomodorotimer. Die Idee war es, dass der Countdown zunächst nur auf der Startseite zu sehen sein sollte, aber auf allen Seiten eine Benachrichtigung erscheint, wenn die Zeit um ist.

Nach der ersten Codierung stellte sich heraus, dass sich der Timer bei jedem Aufruf des Dashboards zurücksetzt und der Countdown von Neuem beginnt. Im weiteren Verlauf ist die Gruppe zu dem Ergebnis gekommen, dass für eine sinnvolle Implementierung einer solchen Funktion eine serverseitige Technologie benötigt würde. Mit den zur Verfügung stehenden Mitteln wäre eine Überlegung gewesen, den Laufenden Timer anzuhalten und den Wert in der Datenbank zu speichern. Dann müsste aber auf allen Seiten, die der Nutzer besucht, der Wert aus der Datenbankweiterlaufen. Das wäre nicht sinnvoll umsetzbar gewesen.

Mit PHP beispielsweise könnte eine Session erstellt werden und innerhalb dieser Session läuft der Timer weiter.

**Der Kalender**

Ursprünglich war der Kalender als Overlay geplant. Es sollte nur eine kleine Funktion sein, die dem User ermöglichen sollte, das aktuelle Datum herauszufinden. Daraus hat sich aber die Idee entwickelt, dass sich der User für jedes Datum anzeigen lassen kann, ob er an einem bestimmten Datum Termine hat, und um welche es sich handelt.

Bei den Einstellungen verhält es sich ähnlich. Auch diese waren am Anfang als Overlay geplant. Es wurde sich dann aber für die Variante entschieden, für die Settings eine komplett neue Seite zu erstellen, da dies die Usability für den User verbessern würde. Bei so einem großen Overlay wäre die Wirkung deutlich besser, wenn der Hintergrund ausgegraut würde (Siehe Abb. 2). Dabei könnte es nicht mehr eindeutig sein, dass das Theme wirklich geändert würde. Deshalb ist man von dieser Variante abgekommen.

**Das funktioniert alles noch nicht**

Wird in das Searchfield im Header rechts ein Wert eingegeben, werden zwar Suchergebnisse angezeigt. Die Funktionalität, dass die Datenbank durchsucht wird, ist nicht implementiert worden.

Wenn eine Aufgabe angelegt wird, gibt es eine Option, ein Erinnerungsdatum mit Uhrzeit zu setzen. Laut Planung sollte der User beim Aufrufen der Webseite nach Ablauf des Erinnerungs-Countdowns eine Nachricht über ein Overlay angezeigt bekommen. Diese Funktionalität wurde auch nicht mehr implementiert.

Bei dem Kalender wird im aktuellen Jahr der Februar übersprungen. Woher dieser Fehler kommt, konnte leider auch intensivem Debugging und einer Online-Konferenz mit allen Teilnehmern nicht geklärt werde. Mit etwas mehr Zeit hätte die Gruppe daran sicher weitergearbeitet.

# Die Gruppenarbeit

## Aufteilung und Organisation der Gruppenarbeit

**Organisation**

Vereinbarungen und Absprachen, auch für Termine, wurden per WhatsApp kommuniziert. Auf Microsoft Teams wurden Dateien und Ideen geteilt. Mit wenigen Handgriffen erlaubt es MS Teams, Dateien für alle Teilnehmer zugänglich zu machen.

Zum gemeinsamen Coden wurde vorwiegend ebenfalls MS Teams verwendet, in Verbindung mit der IDE VS Code. Es konnte abwechselnd einer der Teilnehmer die Bildschirmfreigabe erlangen und dann auch auf einem anderen Bildschirm arbeiten. Es wurde vereinbart, dass jeder während alleine Code geschrieben wurde selbstverständlich seinen favorisierten Editor benutzen kann.

Es wurde auch in Betracht gezogen, mit dem Plugin „VS Life Share“ zu arbeiten. Die Methode mit MS Teams hat sich dann aber durchgesetzt, weil so neben dem reinen Code auch noch andere Inhalte auf dem Bildschirm geteilt werden konnten.

**Aufteilung der Gruppenarbeit**

Bei der Aufteilung wurde zunächst darauf geachtet, dass Jeder mit allen Technologien arbeiten kann, die bei dem Projekt zum Einsatz kommen. Bei Online-Konferenzen wurden stets die nächsten Schritte besprochen. Dabei hat sich herausgestellt, dass die Gruppenmitglieder auch vor vermeintlich schwierigen Aufgaben nicht zurückschreckten. Jeder hat sich bereit erklärt, Aufgaben zu übernehmen, die zumindest auf den ersten Blick knifflig erschienen.

Besonders herausfordernde Implementierungen wie z.B. der asynchrone Aufruf von Methoden wurde entweder in Online-Konferenzen gemeinsam bearbeitet oder die Vorgehensweise wurde im Nachhinein mit den anderen Gruppenmitgliedern besprochen.

WhatsApp diente dabei stets als Hauptkommunikationsmittel. Spontan wurden Fragen gestellt und beantwortet. Auch um Konflikte mit Git zu vermeiden, wurde WhatsApp so zu sagen zur Synchronisation eingesetzt. Trotzdem ließen sich Konflikte nicht völlig vermeiden. Beim Beheben der Probleme halfen die Gruppenmitglieder, die bereits mehr Erfahrung mit Git hatten den Mitgliedern, bei denen die Konfliktlösung Schwierigkeiten bereitete.

**Mit diesen Tools wurde gearbeitet**

Bei der Gruppenarbeit wurde meistens der Editor VS Code eingesetzt. Man hat sich für diesen Editor entschieden, weil er sehr gut an die eigenen Bedürfnisse angepasst werden kann und vor allem viele zusätzliche Plugins bietet. Darunter waren auch zusätzliche Plugins wie „Git Graph“ und „GitLens“. Mit „Git Graph“ kann die Historie des Git-Repository visuell dargestellt werden. Mit GitLens kann man sich die Änderungen in einem Dokument auch zeilenweise anzeigen lassen. Das war vor allem hilfreich, beim Lösen von Konflikten.

Bei der Farbauswahl kam der JCColorPicker zum Einsatz. Findet man auf einer Webseite eine Farbe, die man auch in seinem eigenen Projekt verwenden möchte, findet der JCColorPicker dafür den passenden Wert, entweder in hexadezimaler Form oder als RGB/RGBA-Wert. Für die eingefügten Icons wurde sich bei IconMonstr und Flaticon bedient. Die Icons dürfen kostenfrei verwendet werden.

In wenigen Fällen wurde auch ein Icon oder Bild nachbearbeitet. Das wurde mit Hilfe von Adobe Photoshop gemacht. Die meisten Icons wurden aber so ausgesucht, dass keine nachträgliche Bearbeitung nötig war.

Das Logo von Taskerizr im Header links wurde mit Adobe Illustrator erstellt.

## Das Vorwissen der Gruppenmitglieder

Das Vorwissen der Gruppenmitglieder war weitestgehend auf dem gleichen Level. Bei einigen Dingen konnten sich die Gruppenmitglieder jeweils helfend unter die Arme greifen. So war das Tool Git allen Mitgliedern schon bekannt. Trotzdem gab es untereinander hilfreiche Tipps wie das Arbeiten mit Git optimiert werden kann. Auch bei dem Befehl „merge“ und dem Lösen von Konflikten beim „mergen“ haben sich die Teilnehmer untereinander geschult und weitergeholfen.

Auch von den eingesetzten Technologien HTML, CSS und Javascript kannte jedes Gruppemitglied die Grundlagen. Dabei war interessant zu sehen, wie das vorhandene Wissen eingesetzt wird, da keines der Mitglieder bisher professionelle Erfahrung in der Webentwicklung hatte und auch noch nicht in einem Team mit 4 Teilnehmern programmiert hatte. Trotzdem ist die Gruppe immer wieder auf kreative Ideen gekommen und hat viel ausprobiert.

## Wichtige Lernergebnisse und Erkenntnisse

**Steht das finale Wireframe schon?**

Im Vorfeld wurden sich viele Gedanken gemacht. Relativ schnell haben sich die Gruppenmitglieder auf ein Thema geeinigt. Danach ging es an das Erstellen von Wireframes. Obwohl sich alle Mitglieder daran beteiligt haben und auch zeitnah Ergebnisse besprochen wurden, hat sich das Design der App im Nachhinein noch einmal geändert. (siehe „3.5 Abweichungen von der ursprünglichen Planung“)

Vielleicht wäre es sinnvoller gewesen, sich vor Beginn der Implementierung noch weitergehend mit dem Wireframe auseinanderzusetzen. Trotzdem haben die sich die Änderungen aufgrund von agilem Arbeiten bezahlt gemacht.

**Das HTML-Grundgerüst ist wichtiger als gedacht**

In einer Online-Konferenz wurde über eine mögliche HTML-Struktur gesprochen. Es gab dazu unterschiedliche Meinungen und diese waren mit Begründung für sich gesehen auch richtig. Bei der Auflistung der vorhandenen Aufgaben wurde beispielsweise überlegt, welches Element als umgebenden Container genutzt werden sollte.

Dabei gab es Überlegungen, ein Section-, Article-, Figure oder ein Listenelement zu nehmen. Es gibt Elemente, die nicht in andere Elemente hineingepackt werden dürfen. So entschied man sich schließlich für das „ul“-Element. Auf der anderen Seite gab es Elemente die selbstverständlich erschienen, und aufgrund des Feedbacks von Frau Gumm geändert wurden. Der [HTML-Validator](https://validator.w3.org) war dabei eine wichtige Hilfestellung.

Um wirklich gute HMTL-Seiten zu erstellen, gehört neben dem Wissen über HTML-Elemente, zumindest den Wichtigsten, auch Erfahrung. Nur so bekommt man das richtige Gefühl, wann welches HTML-Tag am sinnvollsten ist. Da der Browser, dabei sehr großzügig ist und sehr viele Fehler verzeiht, sollte man sich ganz auf seine Erfahrung und den Validator verlassen. Die Gruppenmitglieder waren sich darin einig, dass die richtigen HTML-Elemente schwieriger zu finden waren, als man sich am Anfang vorstellen konnte.

**Konflikte mit Git**

Trotz sorgfältiger Absprachen und Synchronisation über WhatsApp, kam es immer wieder zu Konflikten. Diese wurden zwar schnell gelöst. Trotzdem haben die Mitglieder auch hier gemerkt, wie wichtig es ist, sich abzusprechen und wenn möglich an getrennten Dateien zu arbeiten. Ist erstmal ein Konflikt aufgetreten, ist es deutlich schwieriger und zeitaufwändiger, diesen Konflikt wieder aufzulösen.

**Das Organisieren von JavaScript-Files**

Wie schon in „3.4.2 Welche Lerneffekte gibt es“ erwähnt, hatte die Gruppe Schwierigkeiten die JavaScripts in einer logischen und gut funktionierenden Weise zu organisieren. Dabei wären noch mehr Absprachen untereinander hilfreich gewesen. Der Lernprozess, wie der Code am besten organisiert werden kann, ist mit diesem Projekt natürlich auch noch nicht abgeschlossen. Vermutlich gehört auch da eine gehörige Portion Erfahrung dazu, um das richtige Gefühl dafür zu entwickeln.

Darüber hinaus wurden auch noch nicht alle Vorteile von ECMA-Script 6 ausgeschöpft. Fronted-Frameworks helfen zusätzlich dabei, den Code besser zu strukturieren. Sowohl in React als auch in Angular wird mit Komponenten gearbeitet, die sich beliebig schachteln lassen. Dadurch ist automatisch eine bessere Organisation des Codes möglich.

**Fazit**

Die Gruppenmitglieder sind sich einig darüber, dass mit diesem Projekt ein guter Einstieg in die Entwicklung von Webanwendungen gelungen ist. Der Fokus in diesem Semester lag auf der Entwicklung des Frontend-Bereichs. Alle Teilnehmer waren sich einig, dass die größte Herausforderung darin bestand, zu lernen, wie man die IndexedDB sinnvoll einsetzt.

Wohin die Reise für die einzelnen Gruppenmitglieder gehen soll, ist noch nicht klar, aber alle freuen sich darauf, später auch mit Frontend Frameworks wie React oder Angular zu arbeiten, die das Leben wesentlich vereinfachen, genauso wie mit Backend-Technologien wie dem Springframework von Java oder PHP (Frameworks) kennenzulernen.

Lübeck, 31.01.2021

Frederick Karliczek,

Tom Gibson,

Felix Hansmann,

Marc A. Fischer