Homepage-Projekt e3fi2 2020/21

Ivanne Thijssen

Inhaltsverzeichnis

Einleitung

Im Rahmen des Homepage-Projektes e3fi2 2020/21 habe ich Ende 2020 die Webseite *note.ivanne.de* erstellt. Der Menüpunkt *Info* führt zu einer Video-Vorführung.

Diese Projekt-Dokumentation ist Voraussetzung für eine gute Note im Berufsschulfach Anwendungsentwicklung und beschreibt, inwiefern Gestaltungskriterien eingehalten und bestimmte Technologien eingesetzt wurden. Das Wissen über MVC (Model-View-Controller) habe ich im Wesentlichen aus einer Video-Reihe vom Youtube-Kanal *Code With Dary* über OOP (Objektorientierte Programmierung), MVC und CRUD (Create, Read, Update, Delete) in der PHP.

Meine Homepage ist im Internet öffentlich zugänglich und verlinkt direkt auf ihren offenen Quellcode bei GitHub. Hauptzweck der Webapplikation ist eine persönliche Terminverwaltung mit optionalen Erinnerungen per E-Mail bei Fälligkeit. Jedem Termin können bis zu 50 Bilder angehängt werden.

1. Datenhandling

Bis auf das PHPMailer-Paket, das ich aus einem externen Repository unverändert übernommen hatte, ist die gesamte Struktur vom Hauptverzeichnis *clean* hier abgebildet:

https://note.ivanne.de/clean/public/doc/tree.html (Anhang 1)

Wer im Browser *note.ivanne.de* eintippt, wird durch .htaccess-Konfiguration auf den Pfad clean/public/ weitergeleitet. An der Wurzel dieses Ordners gibt es wiederum eine .htaccess-Datei, die mittels Rewrite-Engine für sog. saubere URLs (Clean Uniform Resource Locators) sorgt. So muss sich kein Website-Besucher die unschöne Zeichenkette *index.php?url=* ansehen. Stattdessen wird jede GET-Anfrage von einer Instanz der Klasse app/mains/Core.php nach dem Format /<*controller*>/<*methode*>/<*parameter*> behandelt. Beispielsweise wird mit den URL-Parametern /pages/impressum die Methode `impressum()` der Klasse app/controllers/Pages.php aufgerufen, wo die View app/views/pages/impressum.php – mit assoziativem Array als Argument zur Datenübergabe ans Frontend (z.B. Seitentitel) – gerendert wird. Noch ein Beispiel: Mit /notes/delete/42 wäre die Notiz Nummer 42 gelöscht. Die Methode `delete()` ruft übrigens die Lösch-Methode aus dem Model app/models/Note.php auf. Model-Klassen sind immer die Singularform der zugehörigen Controller-Kindsklasse und besitzen als statische Eigenschaft ein Objekt der Klasse app/mains/Database.php.

In public/index.php wird lediglich `require_once '../app/require.php'` befohlen. In app/require.php werden zunächst alle Konstanten aus der config-Datei geholt. Letztere ist wegen geheimen Daten in .gitignore eingetragen und somit nicht bei GitHub zu finden. In der ebenfalls *required* Datei app/helpers/start.php wird durch Umleitung auf HTTPS sichergestellt, dass die Verbindung zum Apache-Server in Dresden nicht ohne Verschlüsselung stattfinden kann. Dann wird schließlich der Core-Konstruktor aufgerufen.

Die o.g. URL-Regel mit Schrägstrichtrennungen gilt nur dann nicht, wenn an der Stelle <*controller*> der Verzeichnisname *scripts* steht. Diese Ausnahme erlaubt insbesondere die Anwendung von XHR (XMLHttpRequest) ohne MVC-Logik. Ich habe versucht, einen AjaxController in das MVC-Konzept zu integrieren und bis jetzt

nicht ganz verstanden, warum ich damit statt der gewünschten View nur die Standard-Index-Seite in das HTML-Zielelement ausgeben konnte.

2. JavaScript

In meinem Projekt wird das JavaScript-Objekt XHR gebraucht, um PHP-Skripte auszuführen und das Ergebnis davon in einem DIV-Element auszugeben, ohne dass das komplette HTML-Dokument neu laden muss. So kann der Applikationsnutzer seine durch Textsuche gefilterten Notizen schon während der Eingabe (HTML-Attribut *oninput*) angezeigt bekommen. Er kann das Suchergebnis auch auf einen Fälligkeitsdatumsbereich einschränken und beliebig umsortieren – alles ohne Seitenaktualisierung.

Da die Notizen in der Listenansicht direkt editierbar sind, ist z.B. die Funktion `transformToArea(obj)` dafür da, angeklickte Notiz-Texte in TEXTAREA-Elemente umzuwandeln, deren Größe veränderbar ist, um die Textbearbeitung benutzerfreundlicher zu gestalten. Wenn der Benutzer mit der Maus über das Kreuz-Symbol fährt (HTML-Attribut *onmouseover*), wird die entsprechende Notiz-Zeile mit einem niedrigeren Opazitätsanteil versehen. D.h. Die Notiz, die im Klick-Fall ganz gelöscht wäre, wird erst mit durchsichtigeren Farben markiert, sobald der Cursor die Schaltfläche auch nur berührt – wobei der Effekt durch die CSS-Eigenschaft *transition* absichtlich leicht gebremst wird.

Auf eine jQuery-Bibliothek verzichte ich bewusst, nicht zuletzt weil wir im Unterricht pures JavaScript geübt haben. Als Ersatz für die grob äquivalente load-Methode aus jQuery dient die Funktion `vanillaLoad(php, data, callback)`. Dabei steht `php` für den Skript-Dateinamen, `data` für das FormData-Objekt, das die POST-Variablen enthät und `callback` für die Funktion, die dann mit dem Ajax-Rückgabewert als Argument ausgeführt werden soll. Während der Wartezeit, die vor allem beim Hochladen von Bildern einige Sekunden lang sein kann, blende ich mit `showRunning()` ein animiertes GIF (Graphics Interchange Format) mit einem joggenden Menschen ein – passend zum Hintergrund-Bild. Mit `inputPicture(id)` wird ein Bilddatei-Inputfeld hinzugefügt, wenn der Nutzer

auf das Plus-Icon in der Bild-Tabellenzelle (app/views/notes/create.php) klickt. Jedes neue Bild-INPUT-Element trägt den selben Namen *pictures[]*, sodass später bei der POST-Anfrage ein Bilddatei-Array übergeben wird.

Hier sind ca. 30 weitere Funktionen gesammelt, auf die an der Stelle nicht näher eingegangen wird:

https://note.ivanne.de/clean/public/javascript/functions.js

3. PHP

Als Projekt-Vorlage für Login und Registrierung mit Auslösen von bedingten Fehlermeldungen hat mir am Anfang das Tutorial-Repository vom Youtuber Dary geholfen:

https://github.com/Darynazar/login-register-script-mvc

Der Navigationspunkt *Notizen* ist nur für angemeldete Nutzer sichtbar. Ob ein Besucher eingeloggt ist oder nicht, wird durch Prüfung des globalen Server-Session-Array-Elementwertes mit dem Schlüssel *userid* festgestellt. Denn bei jedem erfolgreichem Login wird u.a. genau diese Variable gesetzt. Bei Logout wird sie logischerweise zerstört. Welche Seite gerade ausgewählt ist, finde ich anhand des Wertes von `$view` im Backend heraus. Die dafür programmierte Vergleichsfunktion `currentActive($view, $menu)` wird in app/view/includes/navheader.php aufgerufen, um den aktuellen Menüpunkt über die CSS-Klasse *.active* mit Textdekoration (einem Strich obendrüber) zu kennzeichnen.

Das Homepage-Projekt umfasst folgende Anwendungsfälle:

https://note.ivanne.de/clean/public/doc/UseCaseDiagram.png (Anhang 2)

In der Mitte des Anwendungsfall-Diagramms geht es um die Auflistung der Notizen. Je Notiz werden neben Text, Fälligkeit, Wiederholungsintervall und Alarm-Checkbox auch ggf. hochgeladene Bilder in Miniaturgröße angezeigt. Damit selbst mobile Geräte sämtliche Bilddateien darstellen können, greift die PHP-Funktion `resizePicture($pixdata, $pixtype, $divisor)` auf die beim Laden

des BODY-Elementes (HTML-Attribut *onload*) in JavaScript ermittelte und via Ajax-Call in der PHP-Variable `$_SESSION['screenwidth']` gespeicherte Bildschirmbreite zu, um die Bilder optimal zu skalieren.

Mit den Anwendungsfällen auf der Rechten Seite des Diagramms kommen wir zum Thema Fälligkeitserinnerung. Jede Notiz bekommt einen Zeitpunkt für ihre erste Fälligkeit zugewiesen – entweder vom Benutzer über Datum- und Zeit-INPUT-Elemente (der Feld-Typ *datetime* wird nicht mehr unterstützt), oder automatisch mit der heutigen Zeit als Standard-Wert. Eine Notiz kann deshalb auch als Aufgabe, Termin oder ToDo bezeichnet werden. Ist bei einer Notiz die Alarm-Checkbox angewählt, wird bei Eintritt der Fälligkeit eine Erinnerungsmail an die bei Registrierung angegebene E-Mail-Adresse versendet – und zwar mit 10-Minuten-Genauigkeit, da der Cronjob nur alle 10 Minuten den Mailing-Algorithmus in app/crontrollers/Alarms.php startet. Wenn unter Intervall nicht *Einmalig* steht, dann werden E-Mails solange automatisch weitergesendet, bis das Alarm-Häkchen oder die Notiz als solche entfernt wird.

4. XML

Falls der Nutzer all seine Notizen ungefiltert sehen möchte, kann er auf den Link *Notizliste ohne Bilder im XML-Format öffnen* klicken. Andersherum hat er auch die Möglichkeit, Notizen in einer XML-Datei hochzuladen. Dafür wird die PHP-Klasse `XMLWriter` bzw. die Funktion `simplexml_load_file()` eingesetzt.

5. Datenbank

Mit der Software *MySQL Workbench* habe ich 3 Tabellen in der 3. Normalform sowie folgendes ERM (Entity-Relationship-Modell) erzeugt:

https://note.ivanne.de/clean/public/doc/MySQLdesign.PNG (Anhang 3)

In der Relation *users* wird das Attribut *tempcode* mit einer zufälligen Zeichenkette befüllt, sobald der Benutzer einen sog. Reset-Link anfordert, um sein Passwort zurückzusetzen. Dieser Token hat eine Gültigkeitsdauer von 15 Minuten. Bei

Versand des Reset-Links wird in der Spalte *lastreset* der aktuelle Zeitstempel geschrieben, um den Ablauf des temporären Reset-Codes gewährleisten zu können.

In der Tabelle *notes* sind in der Spalte *repeat* nur folgende Intervall-Werte zu finden: `Einmalig|Täglich|Wöchentlich|Monatlich|Jährlich|[1-9][0-9]`. Diesem Muster wird beim Speichern garantiert entsprochen, weil die Optionen innerhalb eines SELECT-Elements präsentiert werden. Wenn allerdings auf die letzte Option *Tagesintervall…* geklickt wird, erscheint ein INPUT-Feld vom Typ *number*.

In der Relation *pictures* sei zu bemerken, dass der Typ von *pixdata* nicht BLOB (Binary Large OBject), sondern MEDIUMBLOB ist. Grund dafür ist die Größeneinschränkung von BLOB auf nur 64 KB. Mit MEDIUMBLOB kann die Dateigröße theoretisch bis zu 16 MB betragen, jedoch wird sie in app/models/Note.php auf 5 MB begrenzt. Mehr ist – schätze ich – nicht nötig, da unüblich für Bilddateien.

Wie bereits im ersten Kapitel erwähnt, hat jede Model-Klasse ein Database-Objekt. In PHP gibt es zwei bekannte APIs (Application Programming Interfaces) für Datenbankverbindungen: *mysqli Extension* und PDO (PHP Data Objects). Ich hatte zuvor nur von *mysqli* gehört aber in den MVC-Framework-Beispielen, die ich bei GitHub gefunden habe, wird mit PDO gearbeitet. Der DBH (Database Handler) ist also auch in meinem Projekt eine PDO-Instanz. So lerne ich wieder was Neues in OOP.

6. Canvas

Unter dem Menüpunkt *Zitate* auf der linken Seite wird ein CANVAS-Element ausgegeben. Dort kann man mit einstellbarer Strichbreite und Farbpalette frei zeichnen. Daneben wurde außerdem eine Spielerei eingebaut: Wenn man 10 Mal auf das Apfelmännchen klickt, wird der JavaScript-Code zur Darstellung der Mandelbrotmenge im Canvas bis zum Ende ausgeführt. Das resultierende 400px breite fraktale Bild (mit 3D-Look aufgrund der 10 Überlappungen) kann man auf Knopfdruck herunterladen – genauso wie eigene Skizzen. Leider funktioniert das Programm noch nicht für Touch-Events. Daher ist die Anzeige von Elementen mit

CSS-Klasse *.mouse-only* bei Browser-Fensterbreite von maximal 770px in einer Medienabfrage verhindert.

7. Style

Die Medienabfragen befinden sich ganz unten im Stylesheet:

https://note.ivanne.de/clean/public/css/style.css

Die Tabelle zur Auflistung der Notizen mit Bildern sollte RWD-Ansprüchen gerecht sein (Responsive Webdesign). TD-Elemente verfügen nämlich über das HTML-Attribut *data-label*, das jeweils den gleich Wert hat wie der TH-Inhalt, also die Spaltenbezeichnung. Dieser Wert wird innerhalb der Medienabfrage als *content*-Eigenschaft der Pseudo-Klasse *td::before* gesetzt, damit auch bei umgekippter Tabelle in Blockanzeige und verstecktem THEAD-Element die Spaltennamen ersichtlich bleiben.

Selbstbewertung

So würde ich mein eigenes Projekt bewerten:

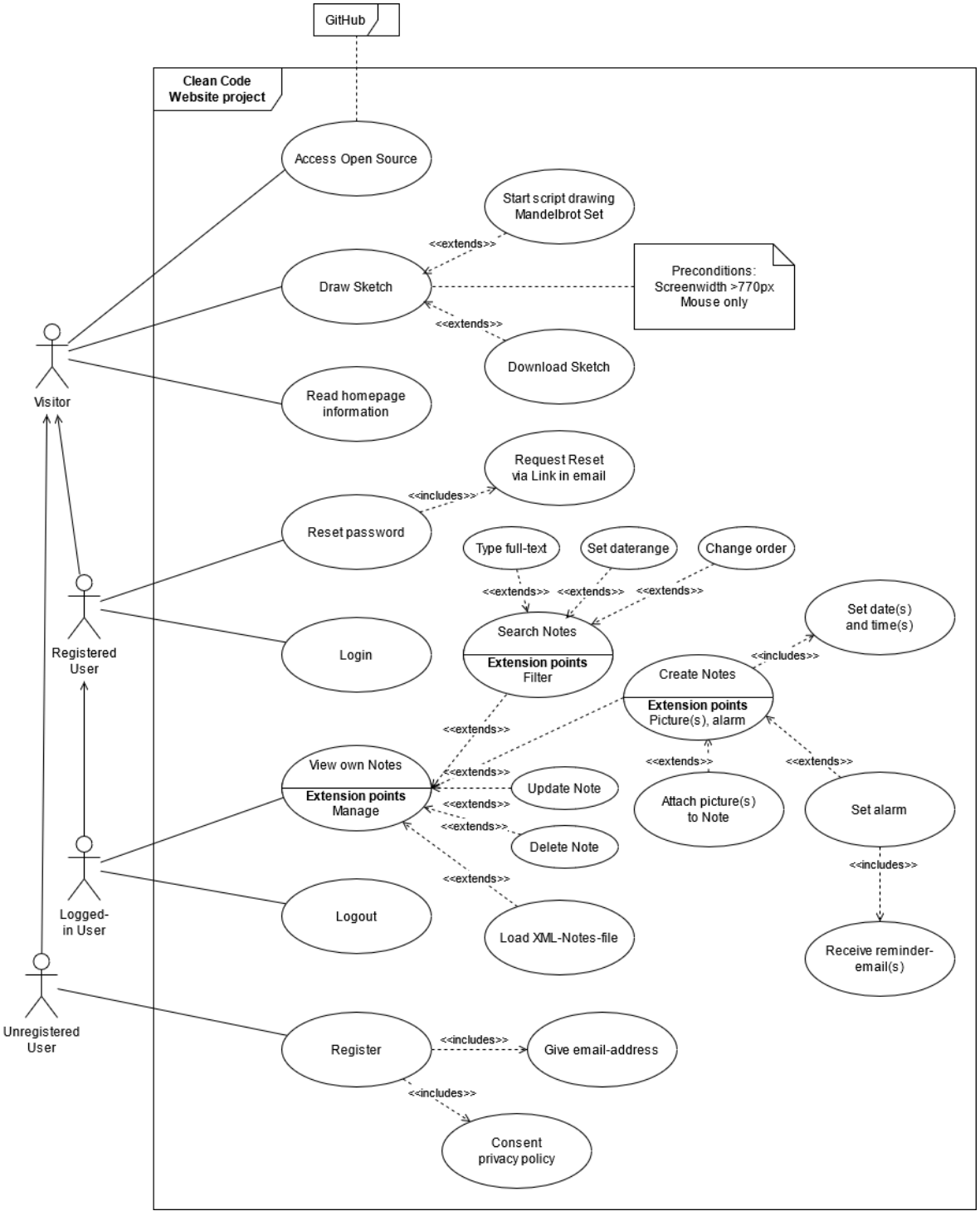
https://note.ivanne.de/clean/public/doc/bewertung.ods

	Bereich	Umsetzungsniveau	Kontext	Beispiel-Auszug	Punkte
Gestaltung	Navigation	Links mit Style und Pseudoklassen	css	`.top-nav ul li a:hover, footer a:hover, .active`	3
	Bilder	Bilder mit alt-Text	notes	`<img src="<?php echo URLROOT; ?>/notes/minpix/<?php echo $id ?>" Alt="Notiz <?php echo $note->noteid; ?>">`	3
		Styling sichtbar	css	`.pix img {box-shadow: 0 .1em .2em #000;border-radius: .2em 0;} .pix img:hover {transform: scale(1.2);transition: .1s ease-in;}`	
		Öffnen bei Klick in neuem Fenster	scripts	`<?php header('Content-type: image/' . $data['pixtype']); echo $data['pixdata'];`	
	Layout	Header und Navigation einspaltig	includes	`<header> <nav class="top-nav">`	3
		Inhaltsbereich zweispaltig	css	`.row {display: flex;}`	
		Fußzeile einspaltig	includes	`<footer>`	
	Typografie	Mehrere Schriften für Content und Überschriften	css	`html, body {font-family: 'oxygen-mono', monospace;} h1, h2 {font-family: 'la-belle-aurore', cursive;}`	3
		eingebundene Font	css	`@font-face`	
	Farben	Mehrere Farben für Header/Footer	css	`header {background-color: #0e151b;} footer {background-color: #000;}`	3
		Schriftfarben sinnvoll gesetzt zu Hintergrund	css	`.top-nav ul li a {color: #fff;} footer a {color: #fff;}`	
	Dokumentation	Gestaltung und Komplexität Diagramme	doc	`https://note.ivanne.de/clean/public/doc/UseCaseDiagram.png https://note.ivanne.de/clean/public/doc/MySQLdesign.PNG`	3
	Gesamteindruck	Sorgfältig ausgewählte Bilder	css	`background: url('https://note.ivanne.de/clean/public/img/pixabay-running.jpg');`	3
		Responsive Webdesign	css	`@media (max-width: 1070px)`	
	Datenhandling	Dateibenennung	doc	`tree.html`	3
		Ordnerstruktur	doc	`https://note.ivanne.de/clean/public/doc/tree.html`	
		Externe Stylesheet/JavaScript im Head-Element	includes	`<link type="text/css" rel="stylesheet" href="<?php echo URLROOT; ?>/public/css/style.css"> <script src="<?php echo URLROOT; ?>/public/javascript/functions.js"></script>`	
Technik	JavaScript	Sinnvolle Interaktion durch Mauseingabe/Tastatur mit HTML-Elementen über Styleeigenschaft	notes	`<td data-label="Text" onclick="expandCreate()">`	6
	PHP	Anwendungsmechanik mit eigenen Klassen und Lese- und Schreibfunktion sowie Darstellung des Inhalts	controllers	`class Notes extends Controller`	7
		Bildgrößenberechnung und -ausgabe	helpers	`function resizePicture($pixdata, $pixtype, $divisor)`	
	Datenbank	Auslesen und Darstellung von Inhalten aus Datenbank mit Filterfunktion bei mehreren verknüpften Tabellen	models	`function findNotesByFilters()`	8
		sinnvolles Schreiben von Inhalten	models	`public function addNote($data)`	
	XML mit PHP	Inhalte aus XML lesen	models	`public function storeFromXML()`	7
	Ajax	XMLHttpRequest	javascript	`function vanillaLoad(php, data, callback)`	
	Canvas	Canvas-Element mit komplexer nutzergenerierter Zeichnung	javascript	`function freeDraw(x1, y1, x2, y2)`	7
		Fraktalbild-Generierung	javascript	`function fillFractal(a, b, color)`	

Anhang 1: Verzeichnisstruktur (Hauptverzeichnis *clean*)

```
+---app
|   |   .htaccess
|   |   require.php
|   |
|   +---config
|   |       config.php
|   |
|   +---controllers
|   |       Alarms.php
|   |       Notes.php
|   |       Pages.php
|   |       Users.php
|   |
|   +---external
|   |   \---PHPMailer
|   |       |   PHPMailerAutoload.php
|   |
|   +---helpers
|   |       mail.php
|   |       resize.php
|   |       start.php
|   |
|   +---mains
|   |       Controller.php
|   |       Core.php
|   |       Database.php
|   |
|   +---models
|   |       Alarm.php
|   |       Note.php
|   |       User.php
|   |
|   +---scripts
|   |       notes.php
|   |       pix.php
|   |       size.php
|   |       xml.php
|   |
|   \---views
|       +---includes
|       |       canvas.php
|       |       footer.php
|       |       head.php
|       |       navheader.php
|       |
|       +---notes
|       |       create.php
|       |       filter.php
|       |       list.php
|       |       maxpix.php
|       |       minpix.php
|       |       table.php
|       |
|       +---pages
|       |       datenschutz.php
|       |       impressum.php
|       |       index.php
|       |       info.php
|       |       quotes.php
|       |
|       \---users
|               login.php
|               register.php
|               reset.php
|               sendlink.php
|
\---public
    |   .htaccess
    |   index.php
    |
    +---css
    |   |   style.css
    |   |
    |   \---fonts
    |       +---La_Belle_Aurore
    |       |       LaBelleAurore-Regular.ttf
    |       |
    |       \---Oxygen_Mono
    |               OxygenMono-Regular.ttf
    |
    +---doc
    |       bewertung.ods
    |       MySQLdesign.PNG
    |       tree.html
    |       UseCaseDiagram.png
    |
    +---img
    |       animated-running.gif
    |       favicon-16x16.png
    |       favicon-32x32.png
    |       favicon.ico
    |       gesture-24px.svg
    |       man-running.PNG
    |       mandelbrot-icon.svg
    |       palette-24px.svg
    |       pixabay-running.jpg
    |
    \---javascript
            functions.js
```

Anhang 2: Anwendungsfalldiagramm



Anhang 3: Datenbank ERM