Simon Besendorfer

Rettenpacherstraße 15/7  
5020 Salzburg

Tel.: 0043 (0) 699 10 61 56 16

e-Mail: simon.besendorfer@me.com

**Dokumentation des Projekts “Wichtel Bot“**

**Eine Projektaufgabe im Zuge der Ausbildung der Developer Akademie**

Github: https://github.com/SimonBesendorfer/impBot

Demo: http://simon-besendorfer.developerakademie.com/impBot/index.html

Inhaltsverzeichnis

[1. Einleitung 3](#_Toc56778193)

[2. Problemstellung 3](#_Toc56778194)

[3. Tech-Stack 3](#_Toc56778195)

[4. Funktionsweise der Anwendung 3](#_Toc56778196)

[4.1. Persönlich 3](#_Toc56778197)

[4.2. Online 4](#_Toc56778198)

[4.3. Erklärung, Impressum, Datenschutz 5](#_Toc56778199)

[5. Dateien und Ordnerstruktur 5](#_Toc56778200)

[6. Projektstart 6](#_Toc56778201)

[7. Klassendiagramm 6](#_Toc56778202)

[8. Methodenübersicht (JSDoc) 6](#_Toc56778203)

# Einleitung

Im Zuge der Ausbildung der Developer Akademie wurde dieses Projekt gewählt, um gelerntes zu festigen und um in dem Projekt auftretende Probleme zu erkennen und zu beheben.

# Problemstellung

Bei der Suche nach einem Eigenen Projekt und den zum Zeitpunkt bestehenden Gegebenheiten (social-distancing aufgrund der Corona Pandemie, Projektstart November 2020) kam die Idee, eine Wichtel App zu erstellen, welche nicht nur vor Ort, sondern auch Online funktioniert und eine Anonyme Wichtelauslosung ohne persönliches Treffen ermöglicht.

# Tech-Stack

Basierend auf der Aufgabenstellung wurden für die technische Umsetzung HTML, CSS und JavaScript verwendet. Da es sich um eine kleinere Aufgabe handelt, welche innerhalb weniger Wochen zu einem fertigen Ergebnis führen sollte, wurde auf die Verwendung eines komplexen JavaScript Frameworks verzichtet.

# Funktionsweise der Anwendung

Beim Öffnen der Startseite erscheint eine kurze Erklärung, was die Anwendung machen kann und wozu diese dient. Es erscheint am unteren Ende ein „Click-to-Action“ Button sowie ein ausklappbares Burger Menü im rechten oberen Eck, welches zu jedem Moment in der App aufrufbar ist.

Nach klicken auf den „jetzt starten“ Button folgt eine erklärung über zwei durchführbare Varianten:

## Persönlich

Die Persönliche Variante führt zu einem weiteren Bildschirm, welcher dazu aufruft, die Namen der Teilnehmer einzugeben und mittels klicken auf den „hinzufügen“ Button in einem Array hinterlegt werden. Bereits eingegebene Namen erscheinen im unteren Teil des Bildschirms.

Eine doppelte Eingabe des gleichen Namens ist nicht möglich, da vor Ablage im Array überprüft wird, ob sich der eingegebene Name bereits im Array befindet. Bei Eingabe eines bereits abgelegten Namens, erscheint ein Popup Fenster mit dem Hinweis, dass dies nicht möglich ist und der Name wird aus dem Eingabefeld entfernt.

Sind drei oder mehr Namen im Array hinterlegt, erscheint ein Button („weiter“) welcher eine zeitlich begrenze Lade Animation zeigt, das Array mischt, darauf achtet, dass es nicht dazu kommt, dass ein Name dem gleichen Namen zugeordnet wird und in weiterer Folge den Endnutzer dazu auffordert, das Endgerät an einen entsprechenden Teilnehmer weiter zu geben.

Nach dem klicken auf „weiter“ folgt eine Abfrage, ob es sich tatsächlich um die Person handelt, für die das folgende Ergebnis bestimmt ist.

Durch klicken auf „ja“ erscheint das individuell mittels Zufalls Formel ausgeloste Ergebnis.

Im weiteren wird jeder Nutzer duch diesen Prozess geleitet um sein Ergebnis zu erhalten.

## Online

In der Option Online wird der Nutzer zunächst auf eine Erklärungs Seite weitergeleitet und bekommt im Anschluss einen persönlich generieten Link welcher auf die seite mit der Endung personalPage.html? als Endung id=1605879079533 anhängt. Die 13 Stellige Zahlenkombination ist ein Unix Time Stamp und wird zur Identifizierung der einzelnen Gruppen genutzt.

Mit dem Öffnen der Individuellen Seite werden auf dem Server hinterlegte Daten (JSON) abgerufen und mit der ID des Endnutzers verglichen. Ist die ID bereits im JSON hinterlegt, wird auf die jeweilige Position zugegriffen, anderenfalls wird die neue ID im JSON erstellt.

Jeder Nutzer, welcher nun mit der individuellen ID zugreift kann sich als Teilnehmer eintragen und für die spätere Abfrage ein Passwort erstellen um sein Ergebnis nicht öffentlich einsehbar zu machen.

Alle bereits eingetragenen Teilnehmer erscheinen am unteren Ende des Bildschirms.

Wird der dritte Nutzer hinzugefügt, erscheint zusätzlich der Button „Ziehung starten“, welcher das JSON-Array mischt und erneut am Server ablegt.

Nachdem das JSON-Array gemischt wurde, ist ein neuerliches mischen nicht mehr möglich.

Bei einem erneuten aufrufen der Seite erscheint die Meldung, dass diese Gruppe bereits eine Ziehung gestartet hat und jeder individuell sein Ergebnis durch Eingabe seines Namens und seines Passworts abfragen kann.

## Erklärung, Impressum, Datenschutz

Im Burger Menü finden sich die Punkte Erklärung, Impressum und Datenschutz.

Im Punkt Erklärung wird kurz auf die Funktionsweise der App eingegangen und erklärt, dass Daten im Online JSON welche älter sind als 30 Tage gelöscht werden.

Impressum und Datenschutz Angaben wurden mithilfe des Datenschutz Generators von AdSimple erstellt und den entsprechenden Komponenten angepasst.

# Dateien und Ordnerstruktur

Das Projekt Wichtel Bot ist in mehrere Dateien und Unterordner gegliedert. Die Datei **script.js** enthält wesentliche Animationsschritte des Burger Menüs, die Datei **personal.js** enthält die Logik für die Ausführung der persönlichen Spielvariante, die Datei **online.js** stellt die Datei mit den komplexesten Funktionen dar, welche für die Durchführung einer online Auslosung notwendig ist. Die Datei **snow.js** ist für die loading Animation zuständig.

Das Projekt nutzt zwei CSS Dateien: **style.css** und **burger-btn-style.css** welche auf allen Seiten eingebunden sind.

Um eine übersichtliche Struktur zu behalten wurden mehrere HTML Dateien erstellt welchen die jeweiligen JavaScript Dateien zugeordnet wurden. Folgende HTML Dateien sind in diesem Projekt enthalten:

* Index.html
* Dataprotection.html
* Decide.html
* Explaination.html
* Imprint.htm l
* Online.html
* Personal.html
* personalPage.html

Die Ordnerstruktur beinhaltet drei Ordner:

* /img  
  Der Ordner /img enthält sämtliche Bild und Grafik Dokumente.
* /php  
  Der Ordner /php enthält die save\_json.php und die JSON Datei online.json.
* /out  
  Der Ordner /out enthält die mit JSDoc erstelle Funktionsdokumentation.

# Projektstart

Das Projekt benötigt keine zusätzliche Software um gestartet zu werden. Es reicht, die Datei **index.html** in einem Webbrowser zu starten. Die Funktionen wurden **getestet unter Safari und Chrome** sowohl auf einem PC als auch auf einem Smartphone mit iOS Betriebssystem.

# Klassendiagramm

Klassendiagramm erstellen

# Methodenübersicht (JSDoc)

Eine Umfangreiche Funktionsdokumentation wurde mit JSDoc (Vgl.: https://jsdoc.app/) erstellt welches einen gängigen Standard für die Dokumentation von JavaScript Methoden darstellt und einen genauen Überblick über Funktionen und Parameter darstellt.

#### **addMessage()**

addMessage() shows information in html after adding a new member to the group

Source:

* [online.js](online.js.html), [line 236](online.js.html" \l "line236)

#### **addParticipant()**

addParticipant() adds a new member to the individual group and saves it to the Server, starts the function getData() and addMEssage().

Source:

* [online.js](online.js.html), [line 178](online.js.html" \l "line178)

#### **addParticipant()**

addParticipant() adds the entered Name to the Array participants and shuffled. Before the Name is added, the entered value is checkt, if it is already in the array. If it is already in the array, an Error Message will be displayed and the name will NOT be added to the Array. The function checkForNext() will be started.

Source:

* [personal.js](personal.js.html), [line 11](personal.js.html" \l "line11)

#### **checkForConflict()**

checkForConflict() compares participants and Shuffled at the same Array position. If there is anywhere the same name, the function shuffle will be startet again. If there is no match, the JSON will be saved to the server.

Source:

* [online.js](online.js.html), [line 362](online.js.html" \l "line362)

#### **checkForConflict()**

checkForConflict() checks, if the name[i] in the participants Array is the same name as it is in at the same position in the Array shuffled. All Array positions will be checkt. If there is the same name on any place, the function shuffle() will be started again. If there is no conclict, the function solution() will be started.

Source:

* [personal.js](personal.js.html), [line 65](personal.js.html" \l "line65)

#### **checkForMatch()**

checkForMatch() checks, if the individual Page ID matches with an ID in the JSON. If there is a Match, the i is saved as indexOfGroup and all the participants will be displayed by the function showParticipants(). If there are more then 3 Player, the start Shuffle Button will be displayed. If there is no matching ID in the JSON, a new Group (newFam) will bei pusht to the Array, saved to the Server and reloaded. It will start the function checkIfShuffled().

Source:

* [online.js](online.js.html), [line 135](online.js.html" \l "line135)

#### **checkForNext()**

checkForNext() checks, if there are already 3 or more participants in the Arrays (participants and shuffled)

Source:

* [personal.js](personal.js.html), [line 38](personal.js.html" \l "line38)

#### **checkIfShuffled()**

checkIfShuffled() checks if the loaded family is new. If not, it will check, if the current group already is shuffled.

Source:

* [online.js](online.js.html), [line 156](online.js.html" \l "line156)

#### **checkPerson(i)**

checkPerson() will Display Result Information and is one of the functions, whick leads you thru the results.

Source:

* [personal.js](personal.js.html), [line 126](personal.js.html" \l "line126)

#### **copyToClipboard()**

copyToClipboard() will copy the individual Link to the Device Clipboard

Source:

* [online.js](online.js.html), [line 34](online.js.html" \l "line34)

#### **createImpFamily()**

createImpFamily() Creates a new empty Group (family) which will be added to the JSON impData and later saved on the server.

Source:

* [online.js](online.js.html), [line 222](online.js.html" \l "line222)

#### **deleteOldFamily()**

deleteOldFamily() is started after klicking on the button "zurück zum Start"

Source:

* [online.js](online.js.html), [line 397](online.js.html" \l "line397)

#### **determineProxySettings()**

determineProxySettings() contains functionality for testing in offline mode.

Source:

* [online.js](online.js.html), [line 119](online.js.html" \l "line119)

#### **finish()**

finish() shows a final Screen and gives the option for going back to index.html

Source:

* [personal.js](personal.js.html), [line 166](personal.js.html" \l "line166)

#### **generatePersonalPage()**

The function generatePersonalPage() generates an individual link for every Group by using the unix Time Stamp as an extention for URL "http://\*\*\*.com/personalPage.html?ID=1605171452000" Before you get on the Page it is displayd in online.html and shoud be copied and sent to the invitet Members in the Group.

Source:

* [online.js](online.js.html), [line 16](online.js.html" \l "line16)

#### **getImp(i)**

getImp() will Display Result information and is one of the functions, which leads you thru the results.

Source:

* [personal.js](personal.js.html), [line 95](personal.js.html" \l "line95)

#### **load()**

The function load() starts the function loadJSONFromServer() when the personalPage is loaded. When the JSON is loaded, a statement will be displayed. in the console, the JSON will be parsed to impData and the function checkForMatch() will be startet. If there is a Problem with loading, a Message will be displayed in the console.

Source:

* [online.js](online.js.html), [line 51](online.js.html" \l "line51)

#### **loadJSONFromServer()**

loadJSONFromServer() contains all the nessesary information for getting the JSON from the server.

Source:

* [online.js](online.js.html), [line 70](online.js.html" \l "line70)

#### **next(i)**

next() will Display Result Information and is one of the functions, whick leads you thru the results. adds +1 th the parameter which is responsible for showing the array position.

Source:

* [personal.js](personal.js.html), [line 154](personal.js.html" \l "line154)

#### **onClickMenu()**

the button for the burger menue in the uper right corner

Source:

* [script.js](script.js.html), [line 5](script.js.html" \l "line5)

#### **saveJSONToServer()**

saveJSONToServer() contains all the nessesary information for saving the JSON on the Server. It is an post statement and will replace the current available JSON with a totaly new one.

Source:

* [online.js](online.js.html), [line 97](online.js.html" \l "line97)

#### **serverError()**

Displays an Error Message if there is any trouble with the connection to the server

Source:

* [online.js](online.js.html), [line 389](online.js.html" \l "line389)

#### **showParticipants()**

The function showParticipants() displays all members of a group in html.

Source:

* [online.js](online.js.html), [line 167](online.js.html" \l "line167)

#### **showPersonalImp()**

showPersonalImp() checks, if the enterd name and password matches with the data from the JSON. If there is a match, the function showPersonToGivePresent() will be startet, if the password is not matching, a popup Info will apear for 4 sec.

Source:

* [online.js](online.js.html), [line 282](online.js.html" \l "line282)

#### **showPersonToGivePresent()**

showPersonToGivePresent() displays the Information which person in the individual goup matches after shuffeling.

Source:

* [online.js](online.js.html), [line 305](online.js.html" \l "line305)

#### **showResult()**

showResult() shows information in html, that the function shuffle() allready has been startet and displays the the form and button which can start the function showPersonalImp().

Source:

* [online.js](online.js.html), [line 259](online.js.html" \l "line259)

#### **showResult(i)**

showResult() will Display Result Information and is one of the functions, whick leads you thru the results.

Source:

* [personal.js](personal.js.html), [line 140](personal.js.html" \l "line140)

#### **shuffle()**

shuffle() Mixes the group Array Shuffled and starts the function checkForConflict().

Source:

* [online.js](online.js.html), [line 346](online.js.html" \l "line346)

#### **shuffle()**

shuffle() is started by clicking on the button "weiter". It will shuffle the Array Position of the Names in the Array shuffled. After the array is shuffled, the checkForConflict() function will be started.

Source:

* [personal.js](personal.js.html), [line 49](personal.js.html" \l "line49)

#### **shureToShuffle()**

shureToShufle() displays a warning message and explans, that the function shuffle() only one time can be startet.

Source:

* [online.js](online.js.html), [line 324](online.js.html" \l "line324)

#### **solution()**

It defines the parameter i and starts the getImp() function

Source:

* [personal.js](personal.js.html), [line 86](personal.js.html" \l "line86)