Installationsdokumentation

Implementationsdokumentation

//TODO Implementierung dokumentieren, sowie Abweichungen von MS2 argumentieren

In dieser Implementationsdokumentation, werden wir unseren UseCase “Wohnort finden” nochmals erläutern und anschließend die einzelnen Installationsschritte aufzeigen.

Im Use Case geht es darum, dass der User Fragen beantwortet auf Grund derer der Server ihm geeignete Wohnorte ausgibt. Wie genau das funktionieren soll wird im Folgenden dargestellt.

Die Anforderungen, die zur Erfüllung wichtig sind sind folgende

Client

Präsentationslogik

Hinzufügen des Buttons um zur nächsten Frage zu gelangen

Da wir eine SeekBar haben ist der automatische Übergang zur nächsten Frage nicht sinnvoll. Der Benutzer wird den regler wahrscheinlich mehrmals verschieben

Daher wurde die Entscheidung getroffen einen Button hinzuzufügen. Aus Gründen der Einheitlichkeit.

Die geeigneten Dörfer werden auf einer Karte angezeigt (siehe GoogleMaps API).

Anwendungslogik

Auf der Clientseite wird berechnet was für ein Typ der Benutzer ist. Damit kann die Suche nach passenden Wohnorten weiter individualisiert werden. Zur Berechnung gibt der Benutzer über den Regler jeweil an wie wichtig ihm Natur und wie wichtig ihm ein vielfältiges Freizeitangebot ist. Je nachdem was höher ist wird der Typ berechnet und zum Objekt User hinzugefügt.

Aus allen eingegebenen Informationen wird eine JSON Datei erstellt, die an den Server gepostet wird.

POST /userwish application/json

Daraufhin berechnet der Server die passenden Wohnorte (siehe Implementierung des Servers). Dem User werden diese automatisch ausgegeben. Intern sendet der Client eine GET Anfrage.

GET /userwish/dorf

Der Benutzer bekommt eine JSON mit den Dörfern und den  jeweiligen Informationen in der Reihenfolge, in der sie angezeigt werden soll. (siehe Server)

GoogleMaps API

Server

Der Server läuft über Nodejs. Er hat die Aufgabe alle Dörfer zu verwalten, sowie die Wünsche der User zu einem geeigneten Ort zuzuweisen.

Die Datenhaltung sieht folgendermaßen aus:

Userwish

id: int

stadt: string

radius: int

miete: int

aktivitaet [ ] : string

typ: string

Dorf

id: int

name: string

nächstestadt: string

entfernung: int

aktivitaeten [ ] : string

gruenflaeche: int

mietspiegel: int

Erhält der Server ein POST auf userwish so berechnet er direkt, welche Orte geeignet sein könnten mit folgendem Algorithmus:

if userwish.stadt == dorf.nächstestadt && userwish.radius <= dorf.entfernung && (userwish.miete - dorf.miete) < 10

for dorf.aktivitaet [ ]

for userwish.aktivitaet[ ]

if dorf.aktivitaet == userwish.aktivitaet

geeignetesDorf[ ] = Dorf;

for geeignetesDorf[ ]

if userwish.typ == natur

if dorf.gruenfläche >= 20

uswerwish.dorf[ ] = dorf;

if userwish.typ == freizeit

if dorf.aktivitaet.lenghth > 10

userwish.dorf[ ] = dorf;

Auf die Get-Anfrage GET userwish/dorf sendet er die geeigneten Dörfer dem User. Dabei wird auch die Reihenfolge der Darstellung je nach Typ berücksichtigt. (z.B. ist der Benutzer ein Naturtyp, werden die Informationen zur Natur als erstes angezeigt)

Installationsdokumentation