Inleiding

N.a.v. het functioneel ontwerp (link), zijn de technische eisen en randvoorwaarden opgesteld voor het concept ‘Wegwijs in de Markthal’. Aan de hand van de genummerde functionaliteiten uit het FO wordt de flow van het systeem bepaald. Aan de hand hiervan wordt er ingezoomd op wat er op technisch gebied nodig is om dit te bereiken, in de vorm van pseudo-code. De gemaakte technische keuzes zullen worden toegelicht en verantwoord, en ook het bijbehorende technische onderzoek zal worden beschreven.

Systeemflow

Functionaliteit #1, #2 en #3.

1) Toon beginscherm, bepaal regelmatig of de gebruiker zich in de geofence bevindt.

    Ja → Haal overzicht kramen op uit back-end.
    Nee → Terug naar 1.

    Pseudo code

```
interval elke twee seconden {
        als ( gebruiker in geofence ) {
                Haal overzicht kramen op uit back-end.
                Toon overzicht kramen.
        }
        anders {
                toon beginscherm.
        }
}
```

Functionaliteit #5, #6 en #7.

2) Back-end haalt informatie uit database en API’s, stuurt JSON response terug.

    Pseudo code

    Verbind met database en API’s.
    Haal naam en foto van kramen op.
    Haal Instagram foto’s en Yummly gerechten op.
    Zet resultaat in een array in JSON en stuur dit terug naar front-end.

Functionaliteit #4.

3) Front-end laadt response dynamisch in.

    Pseudo code

    Wacht op response van stap 2.
Bouw overzicht met JSON response.

Technische eisen en randvoorwaarden

Voor stap 1 van de workflow moet er gebruik worden gemaakt van de HTML5 Geolocation API. Deze API is aan te roepen met JavaScript, en kan de lengte- en breedtegraad van de gebruiker teruggeven. Voordat deze API gebruikt kan worden, dient de gebruiker om privacy redenen toestemming te geven voor het delen van locatiegegevens. Wanneer deze toestemming niet wordt gegeven, of wanneer de device de API niet ondersteund, moet dit netjes worden afgehandeld.

De API bevindt zich in de navigator namespace en is aan te roepen met navigator.geolocation.

Het geofence moet een vierkant worden, omdat dit het makkelijkst is om te controleren zonder al te ingewikkelde berekeningen. Het geofence wordt een JavaScript object in de vorm van:

```
var geofence = {
        xMin: decimal,
        xMax: decimal,
        yMin: decimal,
        yMax: decimal
};
```

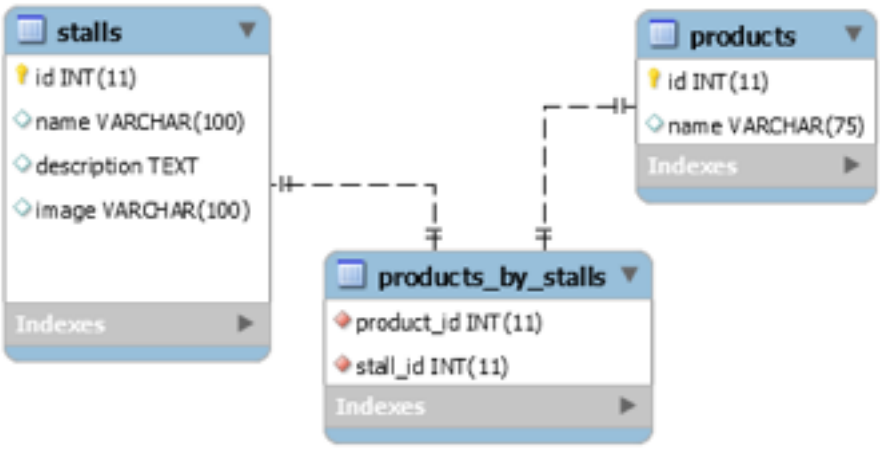
Door te controleren of de x- en y-coördinaten van de gebruiker tussen de xMin, xMax en yMin yMax liggen, kan worden bepaald of de gebruiker binnen het geofence is.
Dit moet elke twee seconden worden gecontroleerd, voor zolang de app open is.

Als de gebruiker in het geofence is, moet er een AJAX GET request naar de back-end worden gestuurd. Dit gebeurt met de native XMLHttpRequest class. De back-end vangt deze request op en haalt de benodigde data op uit de database en eventuele API's. De database driver is MySQL en de verbinding wordt gelegd met PDO. Hiervoor wordt gebruik gemaakt van een reeds door ons ontwikkelde database class, die fungeert als interface om op een veilige manier met de database te communiceren.

De back-end moet verbinding kunnen maken met API's van Instagram en Yummly. Hiervoor zijn API keys nodig. De back-end moet cURL requests kunnen maken. Het resultaat uit de database en API's moet worden samengevoegd in een JSON response. Dit moet een array van objecten zijn, waarin elk object een marktkraam vertegenwoordigd. Wanneer een specifieke marktkraam wordt opgevraagd, moet er een object worden teruggestuurd.

De JSON moet worden omgezet in een JavaScript object, dit kan m.b.v. de JSON.parse() functie. Er moet door arrays en door objecten kunnen worden ge-looped, dit gaat met een for-loop met daarin een for-in loop.

Opzet database



De database bestaat uit drie tabellen; een tabel met marktkramen (stalls), een tabel met producten (products) en een koppeltabel die deze twee aan elkaar koppelt. Een kraam kan namelijk meerdere producten verkopen, en producten kunnen door meerdere kramen verkocht worden. Dit is een veel-op-veel relatie, vandaar de tussenkomst van een koppeltabel. Dit draagt allemaal bij aan het principe om de data maar een keer op te slaan.

Van de kramen wordt de naam, beschrijving en pad naar afbeelding opgeslagen. Het pad is de naam van de afbeelding in kleine letters en hoge strepen als spaties plus de extensie. Van de producten wordt alleen de naam opgeslagen. Beide tabellen hebben een id als primary-key. Deze worden als foreign-keys opgeslagen in de koppeltabel. Let erop dat de koppeltabel geen primary-key heeft, dus vanuit phpMyAdmin is er verminderd CRUD-functionaliteit. (enkel via sql).

Technische specificaties

Het systeem krijgt de vorm van een web-app. De front-end wordt volledig in vanilla Javascript i.c.m. HTML5 en CSS. D.m.v. AJAX wordt er verbinding gelegd met de back-end. De communicatie zal altijd in JSON-formaat zijn.

De back-end wordt een API die los staat van de front-end en op zichzelf kan functioneren. Deze wordt geschreven in vanilla PHP.

Er wordt bewust voor vanilla gekozen, zodat er geen extra library of framework door de developers geleerd hoeft te worden. Dit in verband met de beschikbare tijd. De benodigde functionaliteiten vereisen in principe geen libraries of frameworks. Het zou het project voor de lange termijn werkbaarder maken, maar er wordt nu bewust voor gekozen om hier van af te zien.

Technisch onderzoek

De beperkingen van de technieken voor locatiebepaling zijn middel een functionele test onderzocht. De GPS van mobiele apparaten hebben een maximale nauwkeurigheid van ongeveer 65 meter. Dit is voldoende om een geofence om de Markthal heen te plaatsen.

Ook is er uit het onderzoek naar voren gekomen dat mobiele browsers geen toegang hebben tot de camera van het apparaat. Native applicaties hebben deze toegang wel. Het is dus niet mogelijk om in een website QR-codes te scannen.