

Design Rationale

bRAIN

Meesterproef



Door

Nigel Fijnheer

Casper den Nijs

Dennis de Graaf

Opdrachtgever

Spatwater

Brain

Communication and
Multimedia Design

Hogeschool van
Amsterdam

Minor Web Design and
Development

2023

Inhoud

Probleem definitie

3

Debrief

4

Ontwerp

6

Home

6

Onboarding

7

Secondary Flows

8

Code

9

Svelte

9

Hygraph

10

API

11

Probleem definitie

Weerextremen in Nederland nemen steeds verder toe. Enerzijds komen periodes van droogte tekorten vaker voor en anderzijds hebben we vaker te maken met extreme neerslag, met wateroverlast tot gevolg. Hier willen we als Nederland zijnde op voorbereid zijn.

Volgens Spatwater is het essentieel dat ook de private ruimte klimaatadaptief wordt ingericht, omdat in de gemiddelde gemeente is 50 tot 60 procent van het totaal oppervlak privaat terrein. Dit betekent dat je bewoners moet overtuigen om klimaatadaptief te worden.

Hun idee is om dit te doen waarin je zowel droogte als extreme neerslag voor een deel kan bufferen is doormiddel van regentonnen. Het is veel voorkomend dat het regenwater nu het riool ingaat en vervolgens terecht komt in sloten en rivieren. Hierdoor verdwijnt het water uiteindelijk in de zee en wordt het zout. Het is beter als het water via de regenpijp in een regenton komt. Zo kan het water bijvoorbeeld worden gebruikt voor het bespoeien van de tuin zodat het in de grond terecht komt.

Debrief

Contactgegevens

Spatwater

Timo van den Berg

t.vdberg@spatwater.nl

Mees van Milligen de Wit

m.vmilligen@spatwater.nl

Jesse Schoenmakers

j.schoenmakers@spatwater.nl

Startdatum 30-5-2023

Datum oplevering 30-6-23

Achtergrondinformatie

SPATwater bestaat uit een nieuwe generatie enthousiaste hydrologen die met een technische achtergrond nét wat anders kijken naar ruimtelijke opgaven. De uitdagingen zijn enorm, maar de kansen die het biedt nog groter.

Met expertise op het gebied van waterkwaliteit, de Kaderrichtlijn Water en klimaatadaptatie helpt SPATwater overheden en bedrijven om toekomstbestendig en waterrobust te worden.

Ze voeren geen analyse uit zonder handelingsperspectief en bieden zowel creatieve als praktische oplossingen.

Debrief

Aanleiding

Weerextremen in Nederland nemen steeds verder toe. Enerzijds komen periode van droogte tekorten vaker voor en anderzijds hebben we vaker te maken met extreme neerslag, met wateroverlast tot gevolg. Hier willen we als Nederland zijnde op voorbereid zijn. In de gemiddelde gemeente is 50 tot 60 procent van het totaal oppervlak privaat terrein. Daarom is het essentieel dat ook de private ruimte klimaatadaptief wordt ingericht. Dit betekent dat je bewoners moet overtuigen om klimaatadaptief te worden. Een van de manieren om dit te doen waarin je zowel droogte als extreme neerslag voor een deel kan bufferen is doormiddel van regentonnen.

Doelstelling

Het doel is om bewoners klimaatadaptief te worden zodat regenwater zo goed mogelijk gebruikt kan worden.

Oplevering

Er moet een deelbare link naar het prototype zijn die Spatwater en kan gebruiken om het idee te pitchen. Er moet een duidelijke beschrijving zijn hoe het prototype te gebruiken is. Ook moet het project overdraagbaar zijn voor developers die hierop door gaan.

Randvoorwaarden

- Een tool voor bewoners waarmee er berekend wordt wanneer de regenton vol zit
- Er moet te zien zijn wanneer het weer zal gaan regenen
- Er moet te zien zijn hoe vol de regenton is
- Een melding gestuurd worden als hij vol is en als hij geleegd moet worden
- Het moet visueel aantrekkelijk en gebruiksvriendelijk zijn
- Aan de hand van het prototype moet het idee goed naar boven komen

Extra:

- Op basis van locatie het weer & berekening van het dakoppervlak
- Het deel voor de gemeente

Gebruikers van het eindresultaat

Het prototype wat wij maken gaat naar Spatwater. Zij gaan het prototype pitchen bij de gemeente Amsterdam, met de bedoeling dat de gemeente dit idee kan gebruiken. Zowel de bewoners als de gemeente zullen de app gebruiken.

Relatie met andere projecten

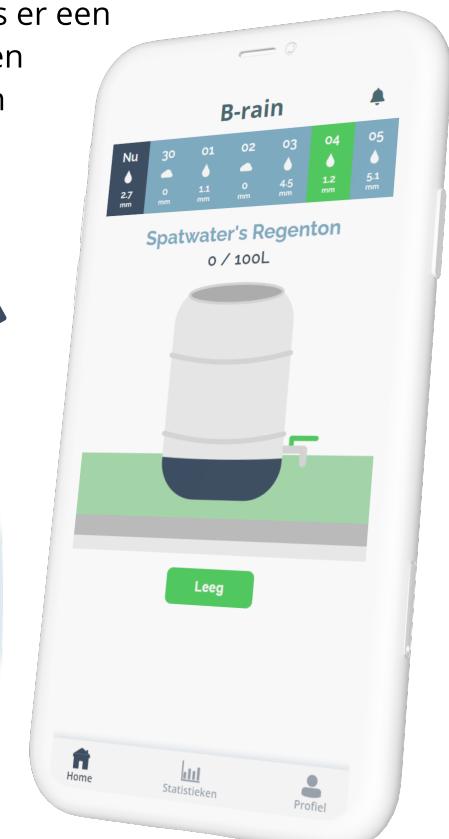
De app Perceelwijzer die beschikbaar is op android. Met deze app kan je door middel van de locatie er achter komen hoe groot je dakoppervlak is. In een ideale wereld zou die api ook hier gebruikt kunnen worden voor het dakoppervlak van de bewoners..

Ontwerp - Home

De core functionaliteit van de pagina zit in de homepage van B-Rain. Dit is waar de regenton te zien is, en waar hij geleegd kan worden.

Er is ervoor gekozen om de regenton in het midden te plaatsen van de pagina, omdat dit het belangrijkste is van de applicatie. Verder is er een tijdlijn aan toegevoegd. Dit zorgt ervoor dat de gebruiker kan zien wanneer het gaat regenen, en wanneer de ton vol komt te zitten

Zodra je de app binnenkomt krijg je een zero state te zien. Vanaf hier kan je de onboarding beginnen.



Flow

6

Ontwerp - Onboarding

Bij het ontwerpen van de onboarding hebben wij de uiteindelijke gebruiker in ons hoofd gehouden. Wij willen dat de gebruiker van B-Rain gemakkelijk de juiste informatie kan invullen.

Het dak type is het eerste wat de gebruiker moet aangeven nadat hij de onboarding is gestart. Doormiddel van bolletjes geven wij aan hoe ver de gebruiker in het proces zit, en wat hij nog moet doen. De forms zijn required, en de gebruiker krijgt feedback als niks is ingevuld.

B-rain

Hoe ziet je dak eruit?

Deze informatie is nodig om de oppervlakte van je dak te berekenen

● ○ ○ ○ ○

Volgende

B-rain

Hoeveel vierkante meter(m²) is je huis?

Deze informatie is nodig om de oppervlakte van je dak te berekenen

aantal vierkantmeter (m²)

○ ● ○ ○ ○

Terug **Volgende**

B-rain

Hoeveel regenpijpen heb je?

Deze informatie is nodig om te bepalen hoeveel regen er in de regenton terecht komt

aantal regenpijpen

○ ○ ● ○ ○

Terug **Volgende**

B-rain

Hoeveel liter water kan je er in je regenton?

Deze informatie is nodig om uit te rekenen wanneer de regenton vol zit

aantal liter (L)

○ ○ ○ ● ○

Terug **Volgende**

De illustraties helpen de gebruiker in het proces

Dak type

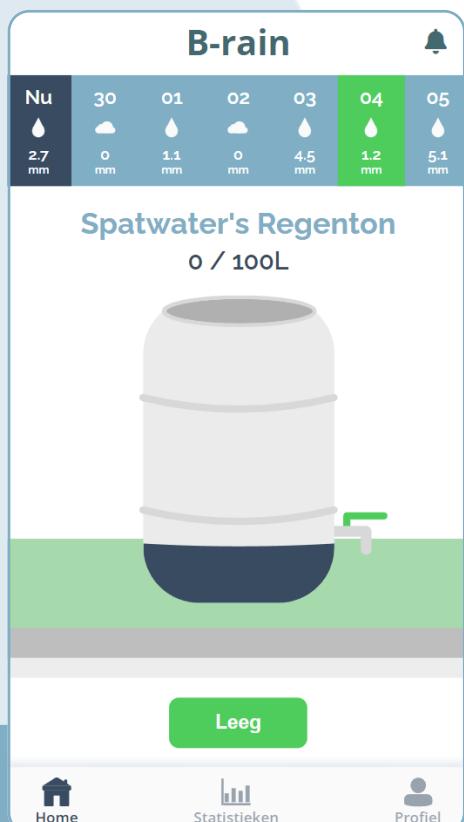
Oppervlakte

Regenpijpen

Inhoud regenton

Ontwerp - Secondary flow

Op de gegevens pagina krijgt de gebruiker nog een keer alles te zien van wat hij heeft ingevuld, en kan hij nog zijn gegevens wijzigen. Wanneer de gebruiker op opslaan klikt zal hij gebracht worden naar zijn regenton.

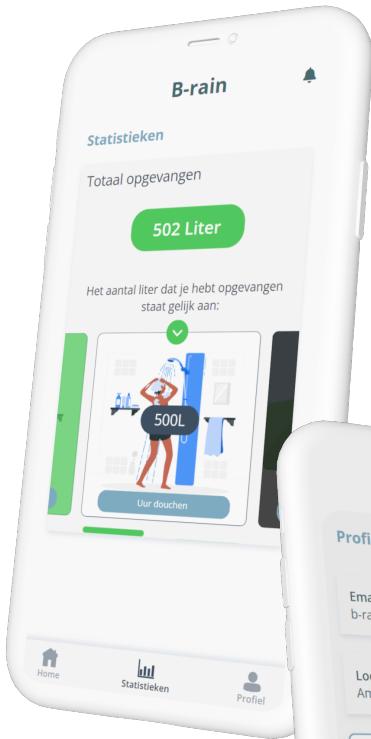


Gegevens

Home page

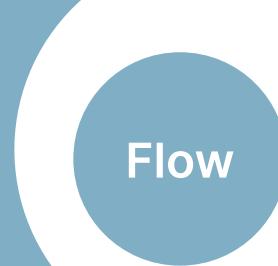
Statistieken

Op de statistieken pagina krijgt de gebruiker te zien hoe veel water hij al heeft bespaard.



Profiel

Op de profiel pagina kan de gebruiker zijn gegevens aanpassen, of regentonnen / daken toevoegen



Uitleg code - Svelte(Kit)

Routing

Routing word door Svelte heel snel geregeld. Je maakt een mapje aan binnen de routes map en je stopt er een +page.svelte in en je kan al bijvoorbeeld navigeren naar de pagina (bijv /home).

In de +page.svelte kan je vervolgens JavaScript schrijven binnen de <script> tag, de HTML die je in de <body> stopt en de CSS die erbij hoort in een <style> tag.

Hierdoor kan je plaatselijk alles bijhouden en onderhouden en heb je geen groot document voor je CSS en JavaScript met alles er in of in modules.

Templating

Verder biedt Svelte ondersteuning voor dingen zoals templating. Je kan bijvoorbeeld heel makkelijk de data die je in JavaScript ophaalt in een tag in je HTML stoppen.

Voorbeeld hiervan is een if else statement met wanneer meer dan 0mm regen valt, je regendruppels moet zien en anders een wolk. Dit kan met de tags toegankelijk gemaakt door Svelte.

Ook wordt de check gedaan doormiddel van data die we uit een API fetchen. Hier halen we neerslagsom op van vandaag en die gebruiken we om te kijken of er regen gaat vallen.

Code

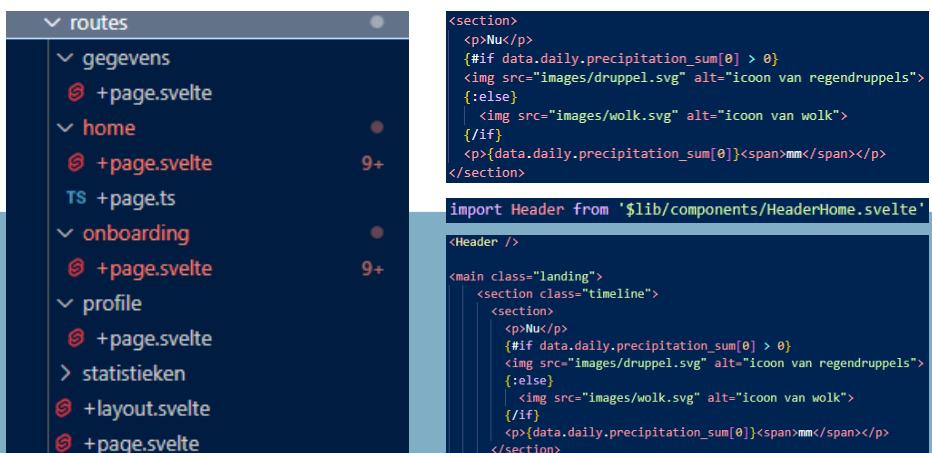
Componenten

Ook kan je in Svelte werken in componenten, dit zijn kleine stukjes HTML met ieder ook aparte CSS en JavaScript. Wij hebben bijvoorbeeld een component gemaakt voor onze header. Je kan het ook een beetje zien als een partial.

In de component hebben we in dit geval alleen HTML en CSS, aangezien we verder geen JavaScript nodig hebben.

Maar omdat dit element vaker voorkomt, kunnen we nu deze toepassen op de pagina's die we willen. We hoeven dan alleen het bestand te importeren en er een tag voor in de HTML te plaatsen. Verder zorgt Svelte voor al het andere werk.

Svelte kan nog veel meer dingen doen, maar gezien de korte tijd en het ons eerste keer is dat we dit gebruiken, hebben we voornamelijk deze onderdelen toegepast.



The screenshot shows a file explorer interface with a sidebar containing a tree view of files and a main panel displaying code snippets. The sidebar shows a 'routes' folder with several sub-folders like 'gegevens', 'home', 'onboarding', 'profile', and some unnamed '+page.svelte' files. The main panel has two parts: the top part shows an HTML snippet with an if-else condition for precipitation, and the bottom part shows a component import followed by its template code.

```
<section>
  <p>Nu</p>
  {#if data.daily.precipitation_sum[0] > 0}
    
  {:else}
    
  {/if}
  <p>{data.daily.precipitation_sum[0]}<span>mm</span></p>
</section>

import Header from '$lib/components/HeaderHome.svelte'

<Header />

<main class="landing">
  <section class="timeline">
    <section>
      <p>Nu</p>
      {#if data.daily.precipitation_sum[0] > 0}
        
      {:else}
        
      {/if}
      <p>{data.daily.precipitation_sum[0]}<span>mm</span></p>
    </section>
  </section>
</main>
```

Uitleg code - Hygraph

Hygraph

Dit is ons platform die we gebruiken voor het opslaan van alle data die we nodig hebben zoals de regentonnen en gebruikers. Je kunt het als een soort API gebruiken waarbij je data kan oproepen (een query) en kan bewerken (een mutation)

Je hebt er alles makkelijk en overzichtelijk op een rijtje en is gebruikersvriendelijk. Het oproepen en bewerken van data wordt gedaan doormiddel van een module GraphQL. In combinatie met node module GraphQL-request hebben we het werkende gekregen.

	Stages	ID	Created At	Created By	Updated At	Updated By	Dak Oppervlakte	Type Dak	Aantal Regenpij... pjes	Inhoud Regenton	User ID
□	Published	clilp7uud0fgt0...	Jun 7, 2023, 2:41 f	Casper	Jun 26, 2023, 11:0	Gegevens tok 1000	Plat	4	1	0	
□	Published	clilgg9wcOrtp...	Jun 7, 2023, 3:15 f	Casper	Jun 26, 2023, 11:0	Gegevens tok 500	Punt	2	1	0	
□	Draft	cliviehdd9oe...	Jun 14, 2023, 11:21	Casper	Jun 26, 2023, 11:0	Gegevens tok 10	Plat	10	10	0	
□	Draft	clivwpxr9t9e0...	Jun 14, 2023, 11:4	Casper	Jun 26, 2023, 11:0	Gegevens tok 11	Plat	10	10	0	
□	Draft	clj42yjmd4ooy...	Jun 20, 2023, 11:2	Public API	Jun 26, 2023, 11:0	Gegevens tok 1	Plat	1	1	0	
□	Draft	clj42yw714oue...	Jun 20, 2023, 11:2	Public API	Jun 26, 2023, 11:0	Gegevens tok 1	Plat	1	1	0	

```
const getData = async () => {
  console.log("getdata");
  const endpoint = "https://api-eu-central-1-shared-euc1-02.hygraph.com/v2/clilodskp031201uhbhpdaltk/master";

  const graphQLClient = new GraphQIClient(endpoint, {
    headers: {
      authorization: `Bearer eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImtp2CI6ImdjXMTbWFpbj1wcm9kdHNVoaW9uIn0.eyJ...
```

GraphQL (-request)

In de afbeeldingen rechts zie je een voorbeeld van hoe een GraphQL connectie opgezet wordt en een query en mutation. Elke connectie wordt opgezet met een url naar de API. Om gebruik te kunnen maken van de API moet je een endpoint token aanmaken met de juiste rechten. Deze token zet je dan neer bij authorization met ervoor `Bearer`.

Daarna verklaar je welke query je wilt uitvoeren, in het eerste voorbeeld willen we gegevens ophalen en dan alleen de id, huidigOpgevangenWater en inhoudRegenton. Uit de lijst gegevens willen we alleen de laatste hebben.

In het tweede voorbeeld voeren we een mutation uit (een bewerking aan de database content). Met deze mutation willen we de huidigOpgevangenWater van de ton resetten naar 0, oftewel legen. Dit doen we door 0 mee te geven en de id van de specifieke ton.

In het laatste voorbeeld hebben we weer een simpele query die de laatste id ophaalt uit de gegevens lijst. Deze gebruiken we voor de mutation hiervoor om de regenton te bepalen.

```
const query = gql`mutation {
  updateBRAIN_data(data: {huidigOpgevangenWater: 0}, where: {id: "${gegevensId}"}) {
    huidigOpgevangenWater
  }
}
```

```
const query = gql`query {
  gegevens(stage: DRAFT, last: 1) {
    id
    huidigOpgevangenWater
    inhoudRegenton
  }
}

const gegevens = await graphQLClient.request(query);`
```

Uitleg code - API

KNMI API

Als allereerst hebben we geprobeerd de KNMI API te gebruiken die aangegeven stond bij de projectomschrijving. Als snel bleek dat deze vrij onduidelijk was in hoe je het moest gebruiken. Ook werkte de fetch's op hun site zelf niet en na zelf het proberen op te halen kregen we geen succes.

In de eerste afbeelding probeerde we een simpele fetch uit te voeren.

```
const apiUrl = 'https://api.dataplatform.knmi.nl/open-data/v1/datasets/ROYK/radius';
const apiKey =
'eyJvcmcI0IiIzTUNGUoOTI3NGE5NjAwMDYtNTIYjEiLCJpZC16IjI4ZmZ1OTZkNDk2ZjQ3ZmE5YjMzMjY5MDU3MjQyMjVIIiieCT
6ImicmIicjEyOC09'; // Vervang dit met je eigen KNMI API-sleutel

// Functie om het fetch-verzoek uit te voeren
async function fetchNeerslagData() {
  const start = '2023-01-01T00:00:00Z'; // Vervang met de gewenste startdatum
  const end = '2023-01-31T23:59:59Z'; // Vervang met de gewenste einddatum
  const lat = '52.0'; // Vervang met de gewenste breedtegraad
  const lon = '5.0'; // Vervang met de gewenste lengtegraad
  const radius = '10'; // Vervang met de gewenste straal in kilometers

  const url = `${apiUrl}?start=${start}&end=${end}&lat=${lat}&lon=${lon}&radius=${radius}&format=json`;

  try {
    const response = await fetch(url, {
      headers: {
        Authorization: apiKey,
        'Access-Control-Allow-Origin': '*'
      }
    });

    if (response.ok) {
      const data = await response.json();
      console.log(data); // Hier kun je de ontvangen gegevens verder verwerken
    } else {
      console.log('Er is een fout opgetreden bij het ophalen van de gegevens..');
    }
  } catch (error) {
    console.log('Er is een fout opgetreden bij het maken van het verzoek:', error);
  }

  // Roep de functie aan om de neerslaggegevens op te halen
  fetchNeerslagData();
}
```

Open Meteo API

Toen hebben we onderzoek gedaan en hebben we besloten om de Open Meteo API te gaan gebruiken. Die bevatte alle informatie die we nodig hadden en is gratis en simpel in gebruiken.

In de tweede afbeelding zie je hoe wij gebruik hebben gemaakt van deze API. We hebben als allereerst twee datums nodig, begin en eind datum. Deze geven we namelijk mee in de arguments voor de endpoint. Deze data gebruiken we namelijk voor onze kleine kalender op de homepagina.

Eerst halen we een nieuw datum voor beide op en tellen we bij het eind datum twee erbij op.

Daarna halen we de maanden op en tellen we er eentje bij op aangezien het begint met tellen vanaf 0 en de API vanaf 1.

Aangezien de API altijd dagen in twee getallen wilt zoals 15 maar ook 06, moeten we een 0 toevoegen wanneer het een enkel getal wordt.

Uiteindelijk voegen we dit samen in het goede formaat en geven we die mee verderop in endpoint. Niet te zien in de afbeelding.

Vervolgens doen we een fetch en nemen we de data mee naar de pagina.

Code

```
export const load = (async () => {
  const startDate = new Date();
  const endDate = new Date();

  startDate.setUTC(startDate.getUTCDate());
  endDate.setUTC(endDate.getUTCDate() + 6);

  const startMonth = startDate.getMonth() + 1;
  const endMonth = endDate.getMonth() + 1;

  const startMonthString = ('0' + startMonth).slice(-2);
  const endMonthString = ('0' + endMonth).slice(-2);

  const startDay = startDate.getDate();
  const endDay = endDate.getDate();

  const startDayString = ('0' + startDay).slice(-2);
  const endDayString = ('0' + endDay).slice(-2);

  const startDateFormat = startDate.getFullYear() + '-' + startMonthString + '-' + startDayString;
  const endDateFormat = endDate.getFullYear() + '-' + endMonthString + '-' + endDayString;

  const endpoint = `https://api.open-meteo.com/v1/forecast?latitude=52.37&longitude=-4.89&daily`;
  const response = await fetch(endpoint);
  const data = await response.json();

  return data;
}) satisfies PageLoad;
```