

Transferable Code Project Eurocom

Inhoudsopgave

Inleiding.....	2
Project Eurocom.....	2
Git Repository opzetten	2
Branches	3
Readme.md.....	3
App Shell.....	4
Chart.js	5
MongoDB Database & MongoDB Compass	7
Feedback	9
Reflectie.....	9

Inleiding

Voor veel oefeningen en projecten zoals ons Eurocom project schrijf ik code in visual studio code. Deze code word bijgehouden in een repository op Gitlab. In dit document zie je hoe een deel van deze code tot stand is gekomen en wat ik geleerd en toegepast heb.

Project Eurocom

Voor ons Eurocom project moest er een app worden gemaakt. In deze app word data op een overzichtelijke manier weergegeven. Voor de app zelf moet eerst een appshell gemaakt worden. Omdat er ook data opgehaald moet worden, moet er een database gemaakt worden. Om de informatie overzichtelijk weer te geven moet deze in grafieken worden gezet. Uiteindelijk hebben we een library gevonden die hier erg goed voor is genaamd Chart.js. De code van onze app is te vinden op <https://git.fhict.nl/I411167/eurocom>

Git Repository opzetten

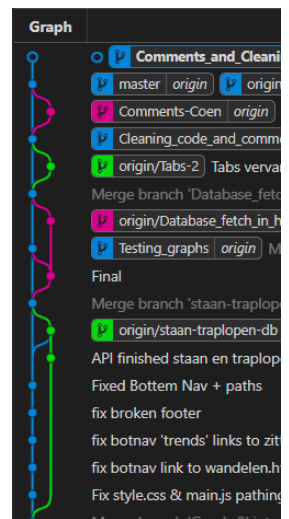
Omdat we met meerdere mensen aan dit project werken, hebben we een repository gemaakt op Gitlab. Vervolgens hebben we deze lokaal op onze computer gezet. Nu iedereen een lokale versie heeft kunnen we code veranderen en dit pushen, hierdoor kunnen andere mensen de code weer pullen en verder werken in de meest recente versie.

The screenshot shows the GitLab interface for a repository named 'Eurocom'. At the top, there's a header with the repository name, a shield icon, and statistics: 48 Commits, 13 Branches, 0 Tags, 573 KB Files, and 573 KB Storage. Below this is a progress bar and a navigation bar with buttons for 'History', 'Find file', 'Web IDE', and 'Clone'. A merge notification banner indicates a merge of 'Comments_and_Cleaning_code' into 'master' by Dijk.Joep J.A.V. van, 3 days ago. Below the banner are buttons for 'README', 'Add LICENSE', 'Add CHANGELOG', 'Add CONTRIBUTING', 'Enable Auto DevOps', and 'Add Kubernetes cluster'. At the bottom, there's a table listing recent commits.

Name	Last commit	Last update
client	Comments toegevoegd	3 days ago
models	Prettier	4 days ago
routes	Final	6 days ago
.env	Complete filesystem overhaul, add API	6 days ago

Branches

Omdat we in een team werken moeten we gebruik maken van branches, gelukkig heb ik dit in de development workshops geleerd en konden we dit makkelijk toepassen. We kregen wel de feedback dat je niet voor ieder persoon een branch aan moest maken maar eerder per onderdeel van je app. Dus een branch voor de appshell en een voor de database connecties etc. Dit kregen we uiteindelijk goed onder de knie en we hebben veel geleerd over het mergen en oplossen van merge conflicts.



Overview

Active

Stale

All

Filter by branch name

Oldest updated

Delete merged branches

New branch

Protected branches can be managed in [project settings](#).

✓

appshe11

merged

15577dec


Make Gaan zitten pagina · 1 week ago


38

0

Merge request

Compare





✓

database

merged

d815eae


Graph colors · 6 days ago


35

0

Merge request

Compare





✓

Graphs

merged

d39002b6


Merge branch 'database' into 'master' · 6 days ago


33

0

Merge request

Compare





✓

Trend_Pages

merged

d1e9ac66


dag, week, maand bar weggehaald bij overzicht · 6 days ago


28

0

Merge request

Compare





Readme.md

In elke repository hoort ook een Readme bestand (en soms ook meerdere). Hierin geef je een soort introductie van je project en de informatie die iemand nodig heeft om deze te gebruiken. Je verteld dus kort wat het doet, wat je nodig hebt om dit te gebruiken of te installeren en andere handige commands of functies die je kunt gebruiken. Hieronder zie je een deel van de readme uit ons Eurocom project.

In onze readme staat bijvoorbeeld dat je Nodejs geïnstalleerd moet hebben, de repository moet clonen en de nodemodules moet installeren door `npm -i` in de console te typen. Als je dit niet doet heb je geen verbinding met de database en werken de grafieken niet.

README.md148 KB

↩

Edit

Web IDE

Replace

Delete

🔍

📄

📁

Eurocom

CONTENTS OF THIS FILE

- Introduction
- Requirements
- Installation
- Configuration
- Maintainers

INTRODUCTION

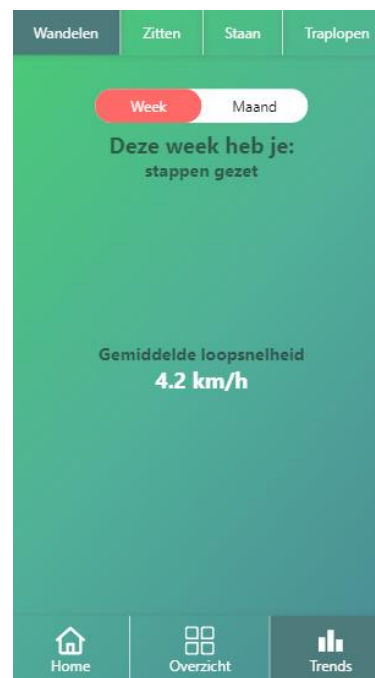
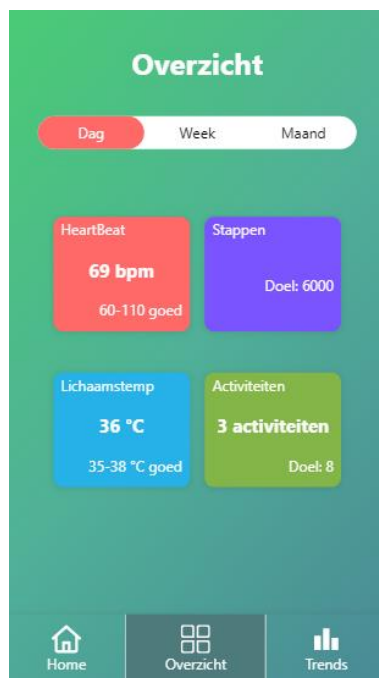
This is a project made for Eurocom group. This project consists of an API and a mobile website. The goal of this project was to give insight on problems pertaining to mobility by using sensor data gathered from a smart device.

App Shell

Ik ben begonnen met het maken van de app shell. Omdat ik een goed prototype had gemaakt in Adobe XD was dit makkelijk na te maken. Met de kennis van Html, Css en Javascript die ik van vorig semester had én die ik nieuw geleerd had door de development workshops kon ik snel deze app shell in elkaar zetten. Ik had hiervoor een branch aangemaakt. Uiteindelijk zouden er ook andere mensen werken aan de app shell files, daarom heb ik overal meteen duidelijke comments bij gezet. Ik vertelde bijvoorbeeld dat een block waarin data weergegeven word een bepaalde class moet hebben en dat alle kopjes weer een andere specifieke class moest hebben en hoe alles werkte. Hierdoor konden mijn groepsleden makkelijk zien hoe de code in elkaar zat en dingen kopiëren waar dat nodig was.

Ik heb uiteindelijk de homescreen, data overzicht, trendpagina's, onderste navigatie en de dag/week/maand knoppen gemaakt. Baldur heeft de andere trendpagina's gekopieerd en de titels vervangen zodat deze goed aansloten op de data die weergegeven werd. Ik heb voor nu nog placeholders gebruikt op de plekken waar de data en grafieken komen te staan.

<https://git.fhict.nl/l411167/eurocom/-/blob/master/client/index.html>



```
<body>
  <!-- Header -->
  <header>
    <nav class="headNav"> <!-- Dit is de bovenste navigatie waarmee je naar alle trendpagina's kunt gaan -->
      <div class="headTabs">
        <a href="traplopen.html" class="headTab">Traplopen</a>
        <a href="zitten.html" class="headTab">Zitten</a>
        <a href="staan.html" class="headTab">Staan</a>
        <a href="wandelen.html" class="headTab active">Wandelen</a>
      </div>
    </nav>
    <section class="dataBlocks"> <!-- Dit zijn de knoppen waarmee je tussen week en maand kunt switchen -->
      <nav class="topNav">
        <div class="tabsTop">
          <a href="#week" class="tabTop active">Week</a>
          <a href="#maand" class="tabTop">Maand</a>
        </div>
      </nav>
    </section>
  </header>

  <section class="contentSlide"> <!-- In deze section zitten 2 divs. De eerste div laat het week scherm zien en de tweede div het maandscherm -->
    <div id="week" class="contentblock screen">
      <!-- Wandelen eerste screen -->
```

Chart.js

Uit ons onderzoek bleek dat Chart.js een handige library was om json data om te zetten naar grafieken. Dit heb ik eerst in een losse poc getest en vervolgens in de app gezet. Om de cdn van de library op te roepen moet je deze code in de head plaatsen.

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/2.6.0/Chart.min.js"></script> <!-- Dit is de cdn van chart.js -->
```

Nu deze code erin staat kun je json data gebruiken uit de database en die omzetten in grafieken. Omdat we nog geen database hadden op dit punt deed ik dit door zelf json data in de html erbij te plaatsen.

```
<canvas id="canvas"></canvas> <!-- Dit is het canvas waar de grafiek staat, de naam moet overeenkomen met de variabele in de javascript -->
```

Dit is het canvas van de grafiek, deze plaats je in de html en de naam moet overeenkomen met de variabele naam uit de javascript. Hieronder zie je de json data die in de grafiek weergegeven word. Uiteindelijk word dit direct uit de database gehaald.

```
var jsonfile = { //Dit is de json data, deze wordt uiteindelijk uit de database gehaald
  "jsonarray": [{
    "month": "Jan",
    "amount": 100
  }, {
    "month": "Feb",
    "amount": 200
  }, {
    "month": "Maart",
    "amount": 364
  }, {
    "month": "April",
    "amount": 872
  }, {
    "month": "Mei",
    "amount": 211
  }, {
    "month": "Juni",
    "amount": 50
  }, {
    "month": "Juli",
    "amount": 362
  }]
};
```

Ook moet je aangeven wat er word weergegeven op de x en de y as, dit doe je door de code hieronder, in het voorbeeld word de maand en het aantal weergegeven.

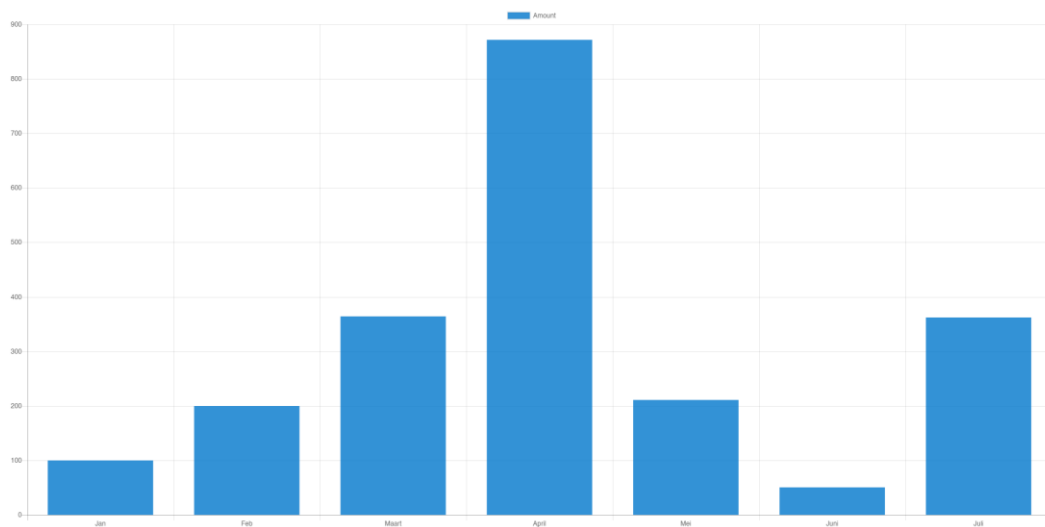
```
var labels = jsonfile.jsonarray.map(function(e) { // Hiermee geef je de x en y waardes aan, hier word de maand opgevraagd (x)
  return e.month;
});
var data = jsonfile.jsonarray.map(function(e) { // Hiermee geef je de x en y waardes aan, hier word het aantal opgevraagd (y)
  return e.amount;
});
```

Het uiterlijk van de grafiek is ook aanpasbaar, je kunt bijvoorbeeld verschillende type grafieken gebruiken zoals een lijngrafiek of bargrafiek. Ook kun je de kleuren aanpassen.

```
var ctx = canvas.getContext('2d'); // Hier kun je dingen aanpassen van de grafiek
var config = {
  type: 'bar', // Je kunt type bijvoorbeeld naar line veranderen voor een lijngrafiek
  data: {
    labels: labels,
    datasets: [[
      label: 'Amount',
      data: data,
      backgroundColor: 'rgba(0, 119, 204, 0.8)' // Hier kun je de kleuren van de grafiek aanpassen
    ]]
  }
};

var chart = new Chart(ctx, config);
```

Met deze code word de grafiek uiteindelijk weergegeven zoals hieronder te zien is.



Met een werkende database is er niet veel verschil. Je moet alleen de data fetchen zoals hieronder te zien is. In dit voorbeeld word alleen de wandelsnelheid van de eerste 7 dagen opgevraagd. Vervolgens worden de x en y waarde weer bepaald.

```
// WANDELSNELHEID week
// De eerste 7 records ophalen vanaf de database via de URL met de limit functie
fetch('/api/wandelsnelheid', {
  headers: {
    limit: 7,
  }
}).then(result => {
  return result.json();
}).then(dataSnelheid => {

  // Variabelen declareren
  var gemiddeldSnelheid = 0;

  dataSnelheid.forEach(element => {
    gemiddeldSnelheid += element.snelheid;
  });
});
```

Ook willen we data uit de database halen en dit weergeven in tekst. Dit kan met de code hieronder. Ook word het getal afgerond zodat je geen mega groot getal te zien krijgt die uit het beeld gaat.

```
// Tekst in de HTML wijzigen door de ID te gebruiken
document.getElementById("gemiddeldSnelheid").innerText = Math.round(gemiddeldSnelheid / 7 * 100) / 100 + " km/h";
```

Hier kun je de code van de app zien:

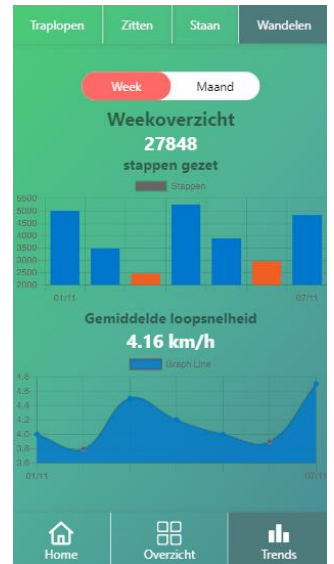
<https://git.fhict.nl/I411167/eurocom/-/tree/master/>

Hier kun je de code zien waar ik chart.js uittest (poc)

<https://git.fhict.nl/I437759/semester-3-media/-/tree/master/Poc's/Chart.js%20poc>

Door een onderstaande code toe te voegen bij de styling van de grafieken kun je een bepaalde kleur geven wanneer een waarde onder een bepaald getal is. Hierdoor kan je een oranje waarschuwingskleur geven wanneer er een gevaarlijke waarde zou zijn. Wanneer de grafieken in de app zijn gezet ziet de app er uit zoals de afbeelding hiernaast.

```
data: data,
backgroundColor: function (context) {
  var index = context.dataIndex;
  var value = context.dataset.data[index];
  return value < 3000 ? 'rgba(240,94,35)' : // draw negative values in red
    // index % 2 ? 'blue' : // else, alternate values in blue and green
    'rgba(0, 119, 204)';
}
}]
```



MongoDB Database & MongoDB Compass

Net zoals in de development workshop hebben we een MongoDB database opgezet. Deze hebben we op dezelfde manier verbonden. Ik kreeg de tip dat er iets bestond genaamd MongoDB Compass. Dit is een programma waarmee je makkelijk verbinding maakt met je database en hier data kunt toevoegen of aanpassen. Hierdoor kon ik makkelijk data toevoegen en kijken hoe de grafiek er uit ziet. Hieronder zie je de verschillende databases in MongoDB Compass.

MongoDB Compass - cluster0.kmwz1.azure.mongodb.net:27017/testData

Connect View Help

Local

6 DBS 16 COLLECTIONS

☆ FAVORITE

HOSTS

cluster0-shard-00-01.kmw...

cluster0-shard-00-02.kmw...

cluster0-shard-00-00.kmw...

CLUSTER

Replica Set (atlas-oik8qo-s-...)

3 Nodes

EDITION

MongoDB 4.2.10 Enterprise

Q Filter your data

> Mobilo

> admin

> config

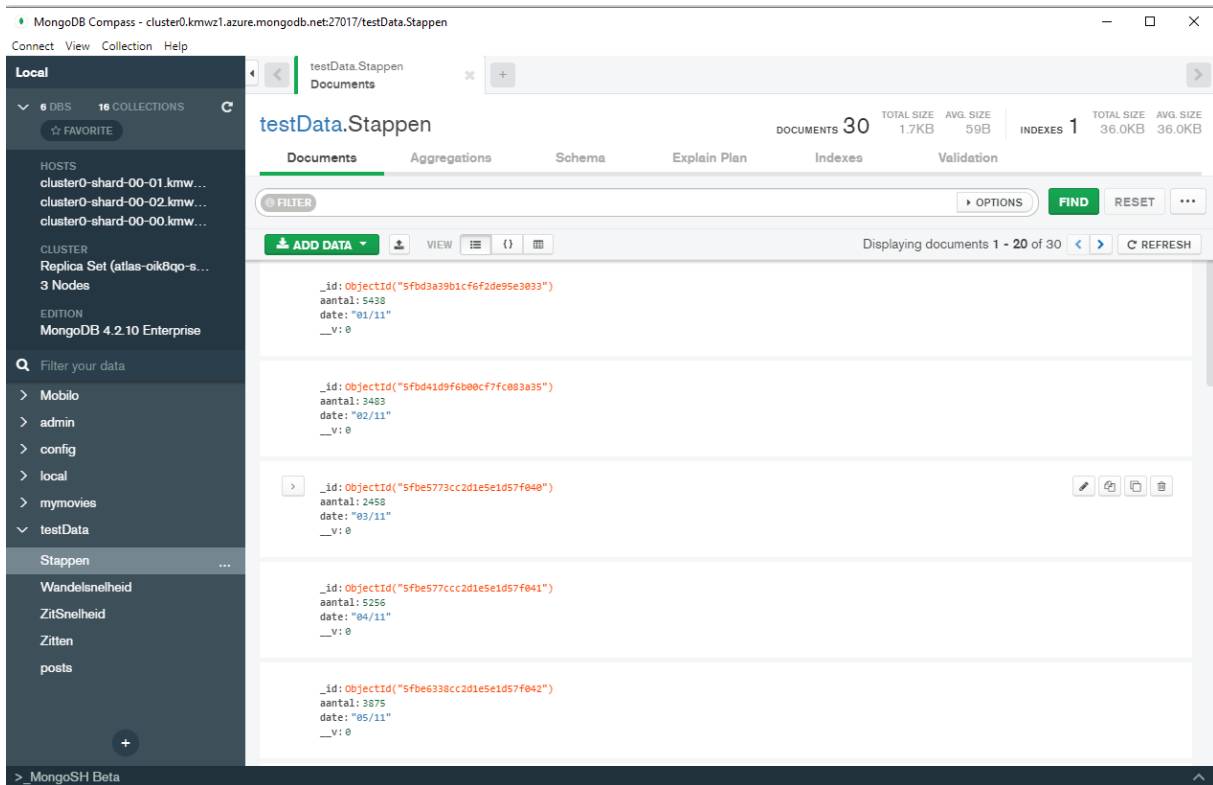
> local

Collections

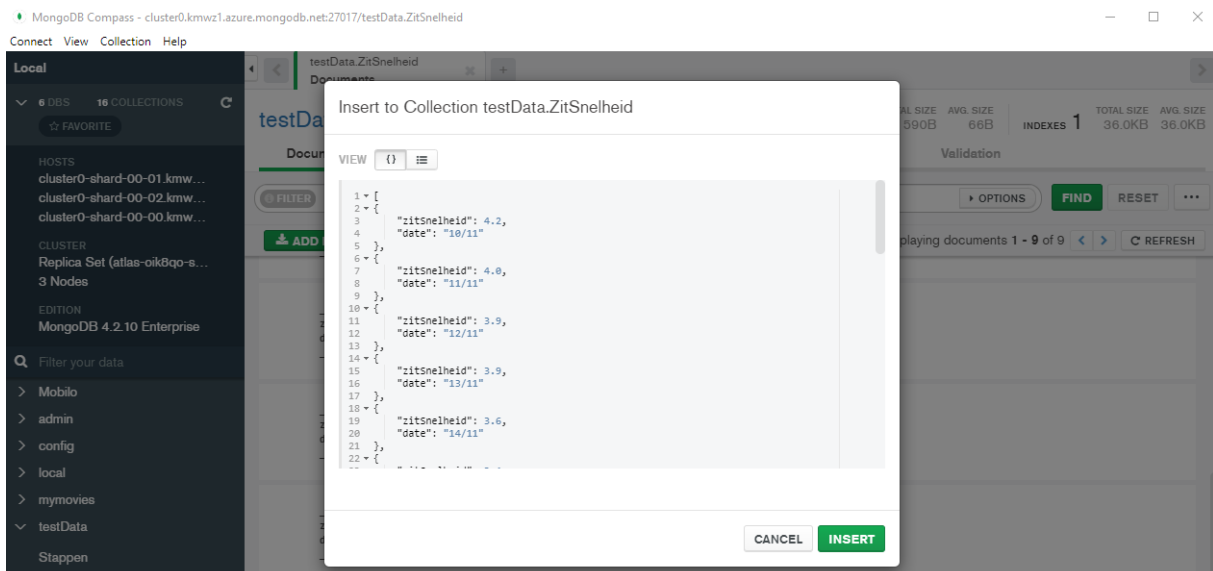
CREATE COLLECTION

Collection Name	Documents	Avg. Document Size	Total Document Size	Num. Indexes	Total Index Size	Properties
Stappen	30	59.0 B	1.7 KB	1	36.0 KB	
Wandelsnelheid	7	63.9 B	447.0 B	1	36.0 KB	
ZitSnelheid	7	67.4 B	472.0 B	1	36.0 KB	
Zitten	7	59.0 B	413.0 B	1	36.0 KB	
posts	4	108.8 B	435.0 B	1	36.0 KB	

Om data toe te voegen hoef je alleen maar op add data te klikken zoals hieronder te zien is. Ook zie je dat er al wat data in de database is toegevoegd.



Als je eenmaal op de add data knop klikt zie je het onderstaande scherm. Hier kun je je eigen data invoeren. Eerst deed ik dit één voor één maar toen realiseerde ik me dat je veel meer data tegelijk kan toevoegen door deze in een array te zetten zoals je hieronder ziet. Als je daarna op insert klikt is de data toegevoegd aan de database en worden de grafieken in de app meteen geupdate.



Deze data word op dezelfde manier gefetcht als eerder aangegeven bij het gedeelte over chart.js.

Feedback

Ik ben een aantal keer langs geweest bij Maikel omdat ik vaak vast kwam te zitten met de grafieken en de directe verbinding naar de database. Maikel vertelde me dat ik goed de documentatie van Chart.js door moest leren en uiteindelijk bleek dat de oplossing erg simpel was als ik de documentatie goed had gelezen.

Ook kregen we de feedback dat we de branches niet per persoon moeten maken maar eerder per onderdeel van de app. Je maakt dus geen branch aan voor ieder groepslid maar bijvoorbeeld één voor de appshell en één voor de database connecties.

Reflectie

In het begin liep ik vaak vast met de grafieken en de database. Daarvoor ben ik vaak bij Maikel geweest die me steeds een stukje verder hielp. Het is belangrijk om goed de documentatie te lezen van een library, voor de volgende keer moet ik dit beter doen omdat ik dan niet tegen stomme fouten aanloop.

Ik heb ook geleerd dat het belangrijk is om rustig te blijven en alles in stapjes uit te werken. Ik wist echt niet waar ik moest beginnen toen ik de grafieken direct moest verbinden met de database. Nadat ik de tip kreeg om dit op te delen kon ik eerst de grafieken werkend krijgen met json data die ik in de html had gestopt. Daarna lukte het om data te fetchen uit de database. Door dit te combineren lukte het uiteindelijk toch wel.

Verder is mijn kennis over javascript ook toegenomen. Ik was hier eerst nooit echt super in maar door goed met mijn groepsleden te communiceren heb ik veel geleerd over het fetchen van data, gebruiken van variabelen, rekenen met variabelen en functies.

Door het constante gebruik van Git en Gitlab heb ik veel geleerd over het werken in een team met Git. Door het mergen van de branches en de merge conflicts op te lossen weet ik ook hoe dit nu in zijn werk gaat.