Logboek

Datavisualisatie

Janne Cools, Lander Durie, Jens Pots *Mei 2024*

Onze website focust op het visualiseren van data in verband met vertragingen bij treinen. Data is verkregen van infrabel (`https://opendata.infrabel.be/explore/dataset/ruwe-gegevens-van-stiptheid-d-1/information/?disjunctive.train_no&disjunctive.relation&disjunctive.train_serv&disjunctive.line_no_dep&disjunctive.relation_direction&disjunctive.ptcar_lg_nm_nl&disjunctive.line_no_arr`). Deze volledige database bevat ongeveer 80GB aan data over vertrektijden, aankomsttijden en vertragingen van treinen. Aangezien we met een statische pagina werken hebben we ons gelimiteerd tot het gebruik van de data van 2023.

Taakverdeling:

- Jens: setup database + stations pagina
- Janne: route pagina + selectie opties
- Lander: general pagina

Naast deze taakverdeling was er ook overlap bij het debuggen, zoeken naar functionaliteiten van observable, kiezen van grafieken ...

1 Dataverwerking

Infrabel voorziet een ruwe datadump met daarin iedere aankomst en ieder vertrek van een trein in een Belgisch station, inclusief de afwijkingen van de geplande tijdstippen in seconden. Deze data is omvangrijk. Een enkele maand komt overeen met een tekstbestand in het CSV-formaat van ruwweg 300 megabyte, goed voor zo'n twee miljoen datapunten.

Het is in de praktijk onmogelijk om deze ruwe data rechtstreeks aan te spreken bij het compileren van de Observable Framework website. De bestandsgrootte is de eerste limiterende factor en maakt het moeilijk om de static-site-generator als DevOps dienst in te zetten om automatisch in te zetten via GitHub Pages. We zouden namelijk gigabytes aan data aan het versiebeheersysteem moeten toevoegen, of deze herhaaldelijk moeten downloaden van Infrabel's API.
Het gebrek aan structuur vormt ook een probleem. Stel dat we alle datapunten voor een station zoals "Gent Sint-Pieters"wensen te vinden. We kunnen hier geen gebruik maken van identifiers, want deze zijn afwezig in de CSV-bestanden. We moeten dus rij-per-rij een string-match algoritme uitvoeren met complexiteit $\mathcal{O}(n)$. Het process wordt dus bijzonder traag.

Om deze data dus efficiënt te kunnen verwerken, schreven we een Golang programma om de CSV-bestanden in te laden, structuur aan te brengen en in een PostgreSQL databank

te laden. De queries die we wensen uit te voeren om data te verkrijgen om de grafieken weer te geven, zijn gedefinieerd in de map *./observable/docs/data*.

1.1 Databank

De SQL queries die uitgevoerd worden om de resulterende grafiekdata te verkrijgen, zijn geen deel van de static-site-generator zelf, voor vele van dezelfde redenen als hierboven. We zouden hiervoor namelijk een draaiende PostgreSQL databank moeten voorzien tijdens het genereren, wat onmogelijk is zonder eerst de ruwe data zelf te verkrijgen uit het versiebeheersysteem of de data. Daarom hebben we ervoor gekozen om niet te steunen op een externe databank. Onze oplossing is dus eenvoudigweg de SQL queries uitvoeren tijdens het ontwikkelen, en het resultaat als CSV-bestand opslaan en toevoegen aan Git.

1.2 Online filteren

Stel dat we data wensen te generen die de gemiddelde vertraging per kalendardag, per station beschrijft. Ons eerste idee splitste de data op per station, waarbij we voor n stations dus ook n CSV-bestanden moesten genereren. Deze zijn individueel relatief klein, en kunnen eenvoudig opgevraagd worden door de browser. Helaas is dit idee niet mogelijk, want de FileAttachment API verwacht een compile-time string literal, waardoor het volgende stukje code onmogelijk is.

```
const stationIdentifier = view(Inputs.select(stations));

// Throws an compile-time string literal error.
const data = await FileAttachment(`./data/${stationIdentifier}`);
```

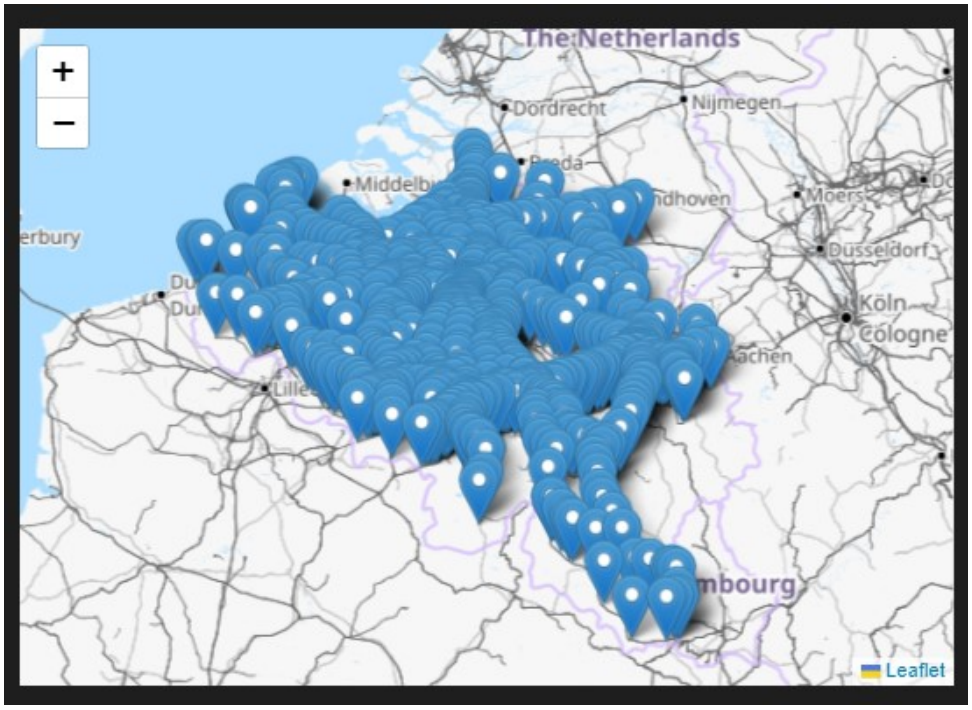
Hetzelfde geldt voor de ingebouwde DuckDB connector, waar SQL queries geen variabelen mogen bevatten.

Daarom kiezen we ervoor om één CSV-bestand (voor bijvoorbeeld alle stations) te genereren en op te slaan. Deze bestanden zijn dan klein genoeg voor een statische pagina en hieruit kunnen we in JavaScript de correcte waarden *client-side* filteren om de data van een specifiek station te verkrijgen. Een performantere werkwijze laat Observable Framework niet toe.
Een vereenvoudigd voorbeeld van zo'n CSV-bestand ziet er als volgt uit:

> station,vertraging_aankomst,vertraging_vertrek,kalenderdag
> Gent-Sint-Pieters,100,40,2023-01-01
> Brugge,152,64,2023-01-01
> ...

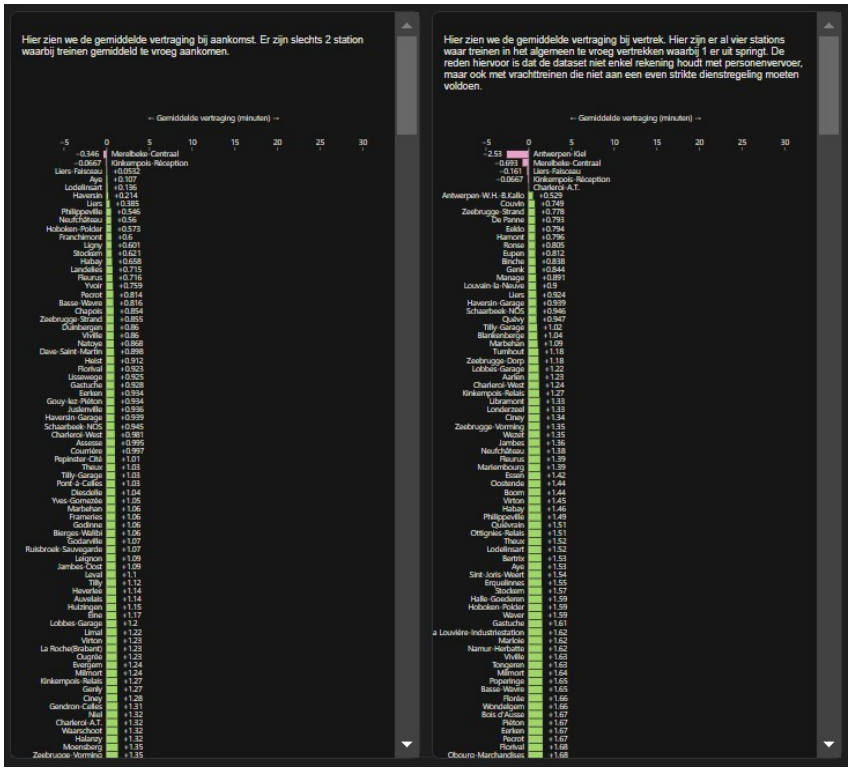
2 Algemeen

Wanneer je naar onze site gaat, is de algemene pagina de eerste die je ziet. Deze toont alle stations in België op de kaart, zoals te zien op figuur 1.



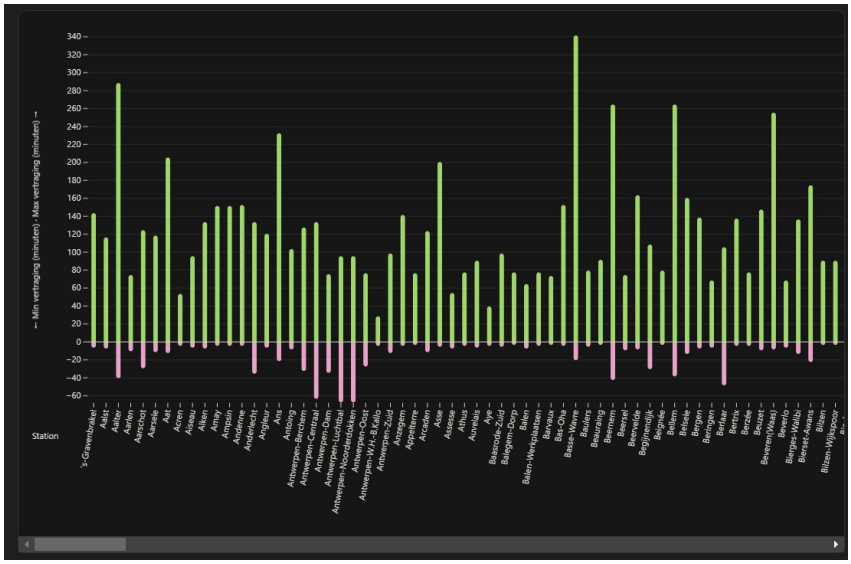
Figuur 1: Kaart met alle stations van België

Vervolgens worden de gemiddelde vertragingen bij aankomst en vertrek per station gevisualiseerd op figuur 2. Eerst probeerden we om alle stations hierop af te beelden, maar dit werd een veel te lange lijst aangezien er honderden stations in België zijn. Vervolgens toonden we enkel de dertig stations met de laagste gemiddelde vertraging en de dertig stations met de hoogste gemiddelde vertraging. Hierna kwamen we echter op het idee om de grafiek scrollbaar te maken zodat we toch alle stations konden tonen. Het scrollbaar maken van een grafiek was wat zoekwerk, maar uiteindelijk is het ons wel gelukt.



Figuur 2: Vergelijkingen van gemiddelde vertragingen bij stations

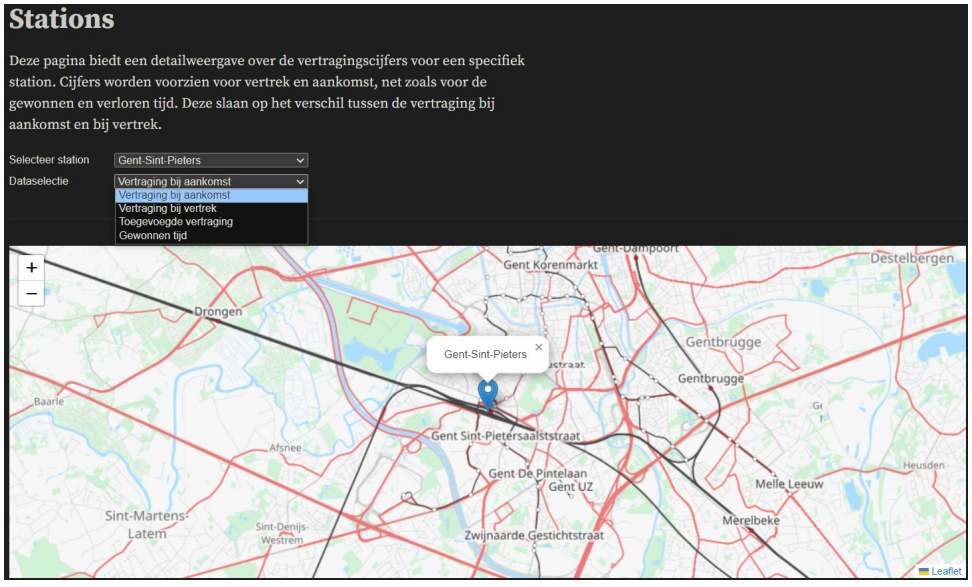
Niet enkel de gemiddelde vertragingen, maar ook de maximale en minimale vertraging kunnen veel informatie geven. Daarom hebben we een grafiek geconstrueerd die voor elk station de grootste en kleinste vertraging tegenover elkaar plot. Aangezien er bij elk station minstens één trein was die te vroeg vertrok, is de kleinste vertraging voor elk station een negatieve waarde. Dit zorgt ervoor dat de grootste en kleinste vertraging van eenzelfde station eenvoudig tegenover elkaar geplot kunnen worden, de ene met een groene staaf in de positieve richting (grootste vertraging) en de andere een roze staaf in de negatieve richting (grootste tijd van te vroeg zijn). Dit wordt afgebeeld op figuur 3, waarbij we opnieuw de grafiek scrollbaar hebben gemaakt zodat we alle stations kunnen afbeelden.
Ons doel van deze visualisatie is om te demonstreren dat de maximale vertragingen veel extremer zijn dan treinen die te vroeg vertrekken.



Figuur 3: Vergelijkingen van maxima van vertragingen bij stations

3 Stations

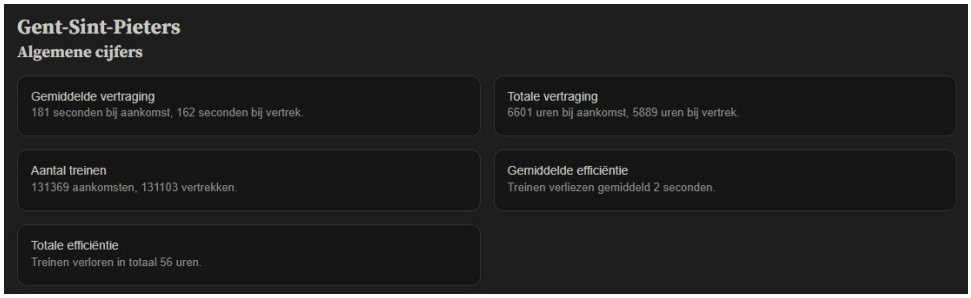
Via de zijbalk kan je naar onze station-pagina navigeren. Hierbij krijg je een detailweergave over de vertragingscijfers van een specifiek station. Een station kan selecteerd worden, samen met het type data waarvan visualisaties getoond moeten worden. Dit omvat vertraging bij aankomst en bij vertrek, alsook de toegevoegde vertraging (verschil tussen vertraging bij vertrek en vertraging bij aankomst) en de gewonnen tijd (verschil tussen vertraging bij aankomst en vertraging bij vertrek).
Het geselecteerde station wordt getoond op een kaart, zoals te zien op figuur 4.



Figuur 4: Keuzelijst voor stations en datatype, kaart met geselecteerd station

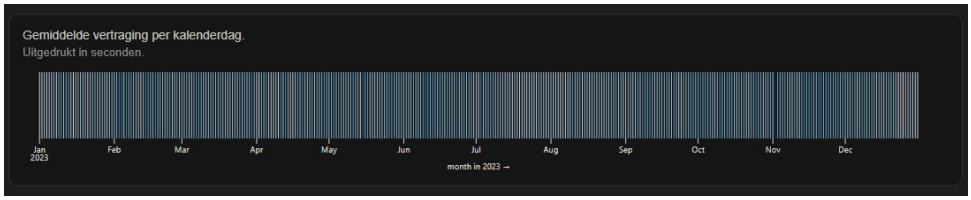
Eerst wordt algemene informatie gegeven over het station (zie figuur 5), zoals gemiddelde vertragingen en het aantal treinen dat er in 2023 gepasseerd is. Deze cijfers kunnen een

zicht geven op de hoeveelheid vertraging in dat station over een heel jaar. Zo kan je enkele stations vergelijken op basis van hun totale vertraging of efficiëntie om te ontdekken bij welke stations er meer vertraging is.



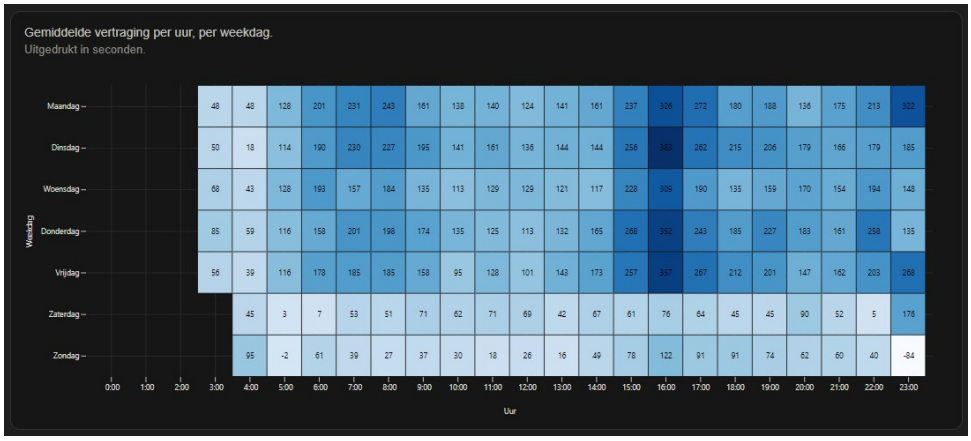
Figuur 5: Algemene informatie over het geselecteerd station

Vervolgens wordt de gemiddelde vertraging gedemonstreerd per dag over het hele jaar 2023 in figuur 6. Donkerdere kleuren komen overeen met grotere vertragingen. Via deze voorstelling willen we een soort correlatie zien tussen data en vertragingen. Zo verwachten we dat badsteden drukker zullen zijn in de zomermaanden en dat verlofdagen ook een invloed kunnen hebben op de vertragingen bij stations.



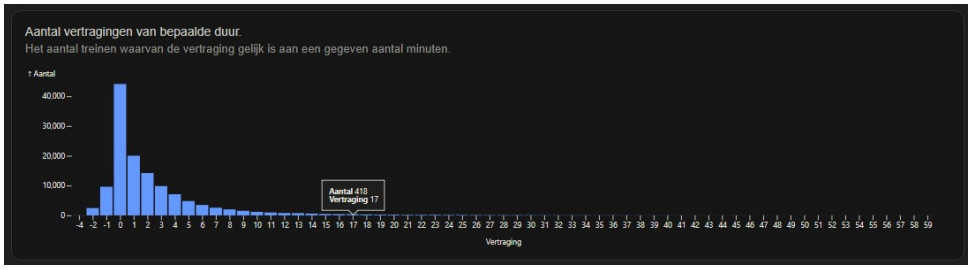
Figuur 6: Gemiddelde vertraging per dag over het hele jaar

De volgende grafiek toont de gemiddelde vertraging per dag en per uur, aangezien er een verschil zou kunnen zijn tussen weekdagen en het weekend. Bovendien verwachten we meer vertragingen rond spitsuren, wanneer iedereen de trein neemt om te gaan werken en het dus drukker is. Om deze correlaties voor te stellen, gebruiken we een heatmap, waarbij je snel kan zien waar grotere vertragingen zijn met behulp van de kleuren. In figuur 7 zie je bijvoorbeeld dat er minder vertragingen zijn in het weekend en de vertragingen extremer zijn tussen de spitsuren (tussen 15 uur en 17 uur en tussen 6 uur en 8 uur).



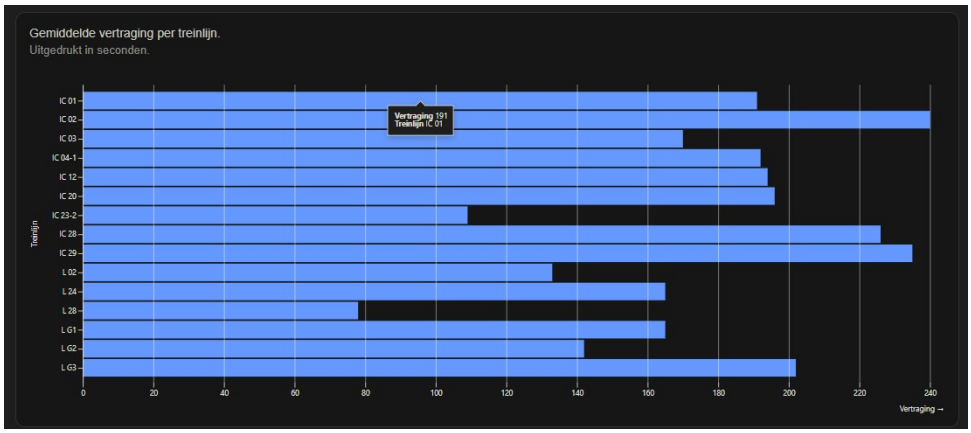
Figuur 7: Gemiddelde vertraging per dag en per uur

Het tonen van gemiddelde vertragingen is echter niet altijd eenvoudig te interpreteren en kan snel beïnvloed worden door een zeer extreem geval van vertraging. Daarom willen we ook een distributie weergeven, waarbij je kan zien hoeveel treinen er een gegeven aantal minuten vertraging liepen. Dit hebben we gevisualiseerd met een bar chart in figuur 8. Zo kan je uit deze figuur afleiden dat de meeste treinen toch minder dan vijf minuten vertraging hebben.



Figuur 8: Distributie van de vertragingen

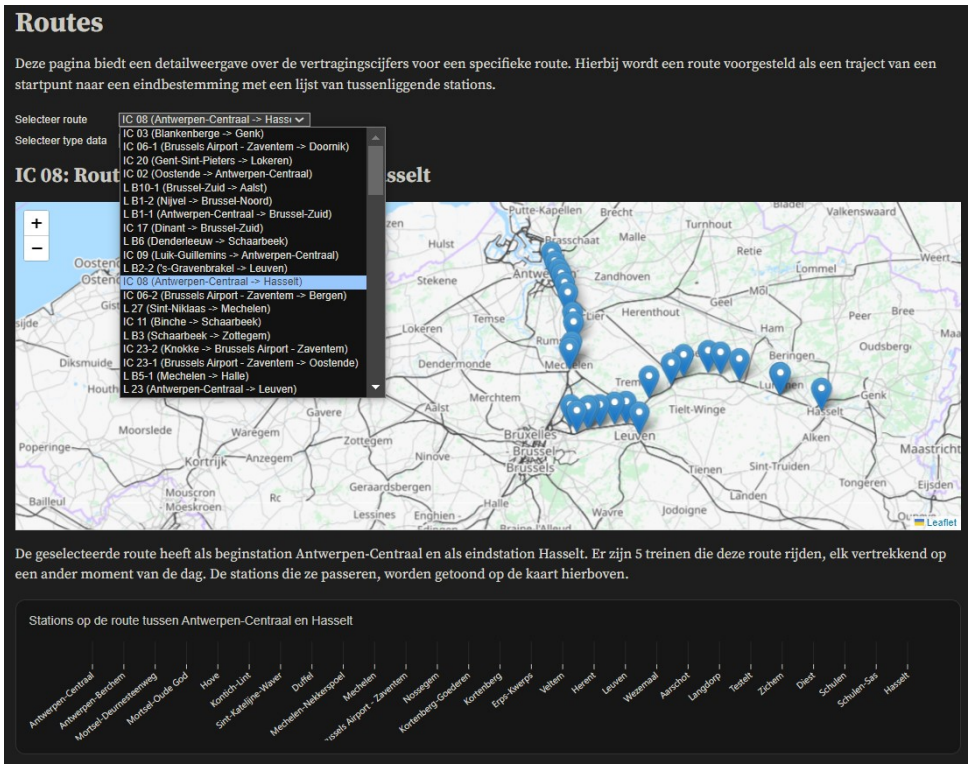
Als laatste hebben we een grafiek gemaakt die de gemiddelde vertraging aan een station vergelijkt over de verschillende treinlijnen die er passeren. De route van de trein kan immers ook een invloed hebben op de vertraging.



Figuur 9: Distributie van de vertragingen

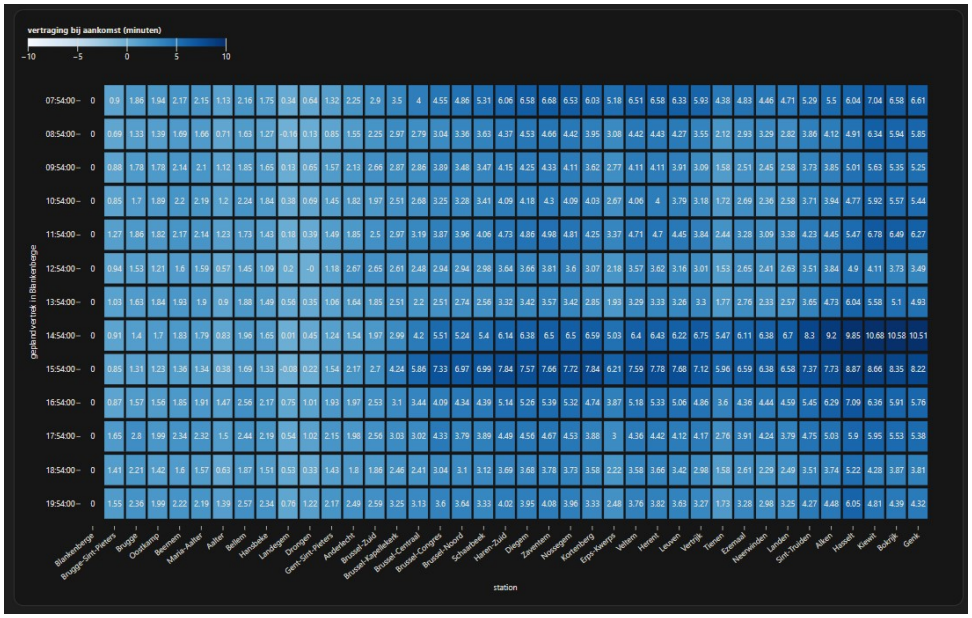
4 Routes

Onze laatste pagina toont verschillende routes die door treinen worden gereden. Het idee is dat een gebruiker uit een lijst van mogelijke routes kan kiezen. Deze bevatten onder andere de IC 03 route tussen Blankenberge en Genk of de IC 26 tussen Kortrijk en Sint-Niklaas. Na het aanduiden van de gewenste route worden de stations getoond die op deze route liggen, zowel op een kaart als in een lijst in de juiste volgorde. Bovendien kan een gebruiker aanduiden of hij vertragingen wilt zien bij vertrek of aankomst. Deze info wordt getoond op figuur 10.



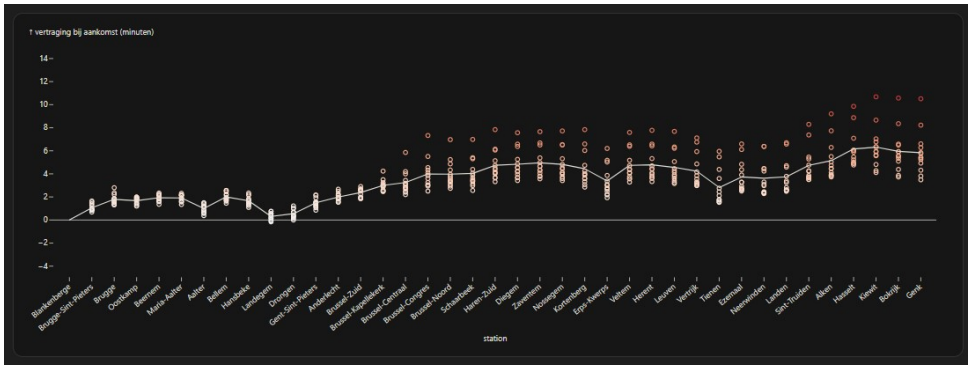
Figuur 10: Keuzelijst voor routes, kaart met stations en oplijsting van station

Vervolgens wordt een heatmap gebruikt om vertragingen van de treinen per station te vergelijken, te zien op figuur 11. Op de x-as staan de stations in de volgorde waarin de treinen ze passeren en de y-as stelt het vertrekuur van elke trein voor. De vertragingen worden dan als een matrix voorgesteld waarbij hogere vertragingen een donkerdere kleur krijgen. Op die manier kan je overgangen eenvoudiger waarnemen. Op de figuur is bijvoorbeeld te zien dat de vertragingen sterk toenemen tijdens de spitsuren, i.e. rond acht uur 's ochtends en tussen 15 uur en 17 uur. Daarnaast worden vertragingen groter naarmate de treinen het eindstation naderen. Hoe verder in de route, hoe groter de kans op vertragingen.



Figuur 11: Vertragingen van treinen op de route

Als laatste visualiseren we op figuur 12 de vertragingen van de treinen per station samen met de gemiddelde vertraging op dat station. De cirkels stellen de vertragingen voor van de treinen op dat station en de lijn verbindt de gemiddelde vertragingen van de stations. Ook hier willen we de trend demonstreren waarbij de vertragingen stijgen richting het eindstation toe. Je kan op de grafiek eveneens waarnemen tussen welke stations er tijd gewonnen kan worden. Dit zijn de stations waartussen de gemiddelde vertraging daalt.



Figuur 12: Vertragingen per station op de route

5 Algemene problemen

De grootste problemen die we tijdens het project tegenkwamen, waren beperkingen van het Observable Framework zelf. Aangezien we met statische pagina's moeten werken, konden we ten eerste niet alle data van Infrabel gebruiken, een totaal van ongeveer 80GB. Daarom hebben we gekozen om slechts de data van 2023 te gebruiken, deze op te slaan in een databank, hierop gerichte queries uit te voeren en de resultaten ervan op te slaan in csv-bestanden. Op die manier konden we per pagina enkele csv-bestanden maken en inladen in de markdown-bestanden.

Daarnaast was het moeilijk om dynamische pagina's te gebruiken waarbij een gebruiker zelf kiest van welk station of welke route hij data wilt zien. Dit hebben we gelukkig kunnen oplossen via de select-boxen waarbij de gebruiker het station of de route kan aanduiden. Voor het maken van de grafieken werden de csv-bestanden gefilterd op het gekozen station/route.

De overige problemen die we tegenkwamen waren kleiner en relatief eenvoudig op te lossen, Een paar ervan zijn al vermeld zoals het scrollbaar maken van grafieken.