

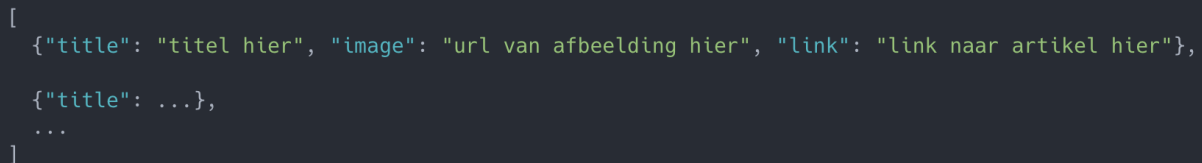
Rapport - Sprint 2

In deze sprint hebben we ons vooral gefocust op het overstappen naar React voor de Frontend, enkele prototypes te maken van de pagina's en het afwerken van de database.

Flask API voor artikels

Aangezien we nu React gebruiken voor de Frontend, is het niet langer de bedoeling dat Flask de artikelen op de homepage weergeeft. Het moet deze enkel toegankelijk maken voor de FrontEnd.

De manier waarop we dit doen is door Flask al de nodige informatie te laten schrijven naar een "/api" -route. In het geval van de artikels ziet deze informatie er zo uit:



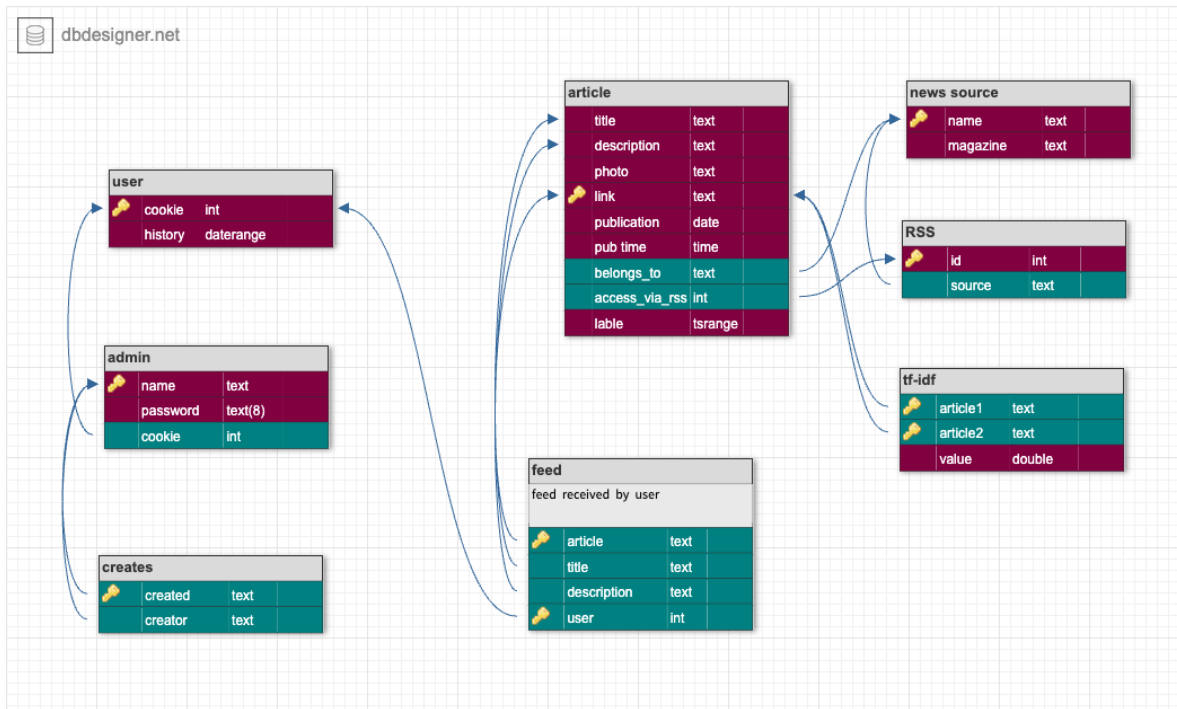
```
[
  {"title": "titel hier", "image": "url van afbeelding hier", "link": "link naar artikel hier"},
  {"title": "..."},
  ...
]
```

Het is dan aan React om deze informatie te lezen en de HTML te genereren. Dit gebeurt door middel van Axios.

We gebruiken hier dus het principe van een Restful API.

Database

De database werd aangemaakt in SQLAlchemy. De reden hiervoor is dat het overzichtelijk is om zo veel mogelijk in dezelfde programmeertaal te schrijven. Het is niet meer nodig om te schakelen tussen python en psql wanneer er gewerkt wordt met de database. Aan de hand van SQLAlchemy kan de database in python worden geschreven en worden de tabellen automatisch in postgres gegenereerd. Hier is een overzicht van hoe de database er (voorlopig) uitziet:



Database Updaten:

Als de database al bestaat maar de tabellen niet overeenkomen met de laatste versie, moeten die worden geüpdatet.

Dit doe je als volgt:

1. Verwijder de tabellen in je (lokale) database die geupdate moeten worden
2. In `src/server/config.py` verander de import: `from sql.sql_config import engine` tijdelijk naar `from .sql.sql_config import engine``
3. Open de python console en tip het volgende:

```
from src.server.config import app
from src.server.config import db
db.create_all()
```

4. Pas de import in `config.py` terug aan

Database Gebruiken:

Om ervoor te zorgen dat de database gemakkelijk gebruikt kan worden zonder voorkennis over de inhoud van de database of postgres, werd de wrapper klasse `ConnectDB`

aangemaakt. Deze klasse heeft eenvoudige functies zoals `addUser()`, `addRSS()`, ... om de inhoud van de database te manipuleren.

Daarnaast zijn er checks voorzien om zeker te zijn dat de database op een correcte manier gebruikt wordt.

Resemblance

Met behulp van het TF-IDF algoritme heb ik een functie geschreven die met behulp van NLTK en gensim twee .txt-files met titels vergelijkt en een dictionary teruggeeft met waarden die duiden op hoe gelijk de titels zijn. De code is gebaseerd op [deze post](#).

De resemblance folder bevat 2 demo-files. Door 'python3 resemblance.py' kan je ze testen.

User Creation

Voor de User Creation heb ik eerst een User model aangemaakt voor onze database. In dit model wordt ook het wachtwoord opgeslagen. Het wachtwoord dat wordt opgeslagen in de database heeft een hash + salt met behulp van bcrypt. We kunnen bcrypt ook gebruiken om het wachtwoord later opnieuw te verifiëren.

Registration

Bij het registreren van een nieuwe user moeten ze via de react front-end een WTForm invullen. Deze ziet er als volgt uit:

```
class RegisterForm(FlaskForm):
    username = StringField(label='User Name', validators=[Length(min=3, max=30), DataRequired()])
    email_address = StringField('Email Address', validators=[Email(), DataRequired()])
    password1 = PasswordField(label='Password:', validators=[Length(min=6), DataRequired()])
    submit = SubmitField(label='Create Account')
```

React zal deze informatie dan naar de `/api/register` route sturen waar Flask het zal opvangen en gebruiken om een nieuwe User aan te maken. Ten slotte zal Flask weer naar React sturen of het gelukt is.

React Router en Error-Handler Componenten

Om vlot te kunnen navigeren tussen pagina's, wordt de Router v6 functionaliteit van React gebruikt.

```
import {BrowserRouter, Route, Routes, Link, NavLink} from "react-router-dom";
```

Elke pagina heeft zijn eigen React component die wordt geassocieerd met een route. Als je een bepaalde route wilt bereiken, wordt de respectievelijke component geladen.

```
<BrowserRouter>
  ...
  <Routes>
    <Route path="/" element={<Homepage />} />
    <Route path="/admin" element={<Admin />} />
    <Route path="*" element={<Error404 />} />
  </Routes>
</BrowserRouter>
```

Er zijn tot nu toe twee error-handler componenten geïmplementeerd, Error 404 en Error 403. Error 404 wordt weergegeven als er een ongeldige of onbekende route wordt bereikt.

Als een gebruiker echter een pagina vraagt waartoe ze geen toegang hebben (omdat ze geen admins zijn bijvoorbeeld), dan krijgt hij een 403 error te zien. Het component meldt dat de gebruiker zich moet inloggen en stuurt hem terug naar de homepage.

Front-end

Tools



We gebruiken dit als front-end framework. Dit geeft ons de mogelijkheid om de site in componenten op te delen. Zo kunnen we de site makkelijk aanpassen en kunnen we taken beter verdelen.

Een voorbeeld hiervan is onze header, die is makkelijk toe te voegen aan elke pagina van onze site en als we aanpassingen maken aan de header dan gaat dit op elke pagina automatisch aanpassen.



Vite betekent in het Frans “snel”, maar in dit geval gaat het over een snelle build tool. Bij development update de site sneller bij updates omdat het werkt met HMR (Hot module replacement). Dit betekent dat het niet heel de site van nul opnieuw built maar enkel de aangepaste componenten, bij uitbreiding van de site zullen we dan niet last hebben van een trager bouwproces.

Bij het debuggen kunnen we dan ook makkelijker vinden in welke file het probleem ligt omdat Vite de source code niet wordt gebundeld in 1 grote file.

Bij deployment wordt de webapp, dankzij Vite, met Rollup gebundeld. Rollup doet ook enkele optimizations en code-splitsing, wat de webapp vlotter laat runnen voor de klanten.



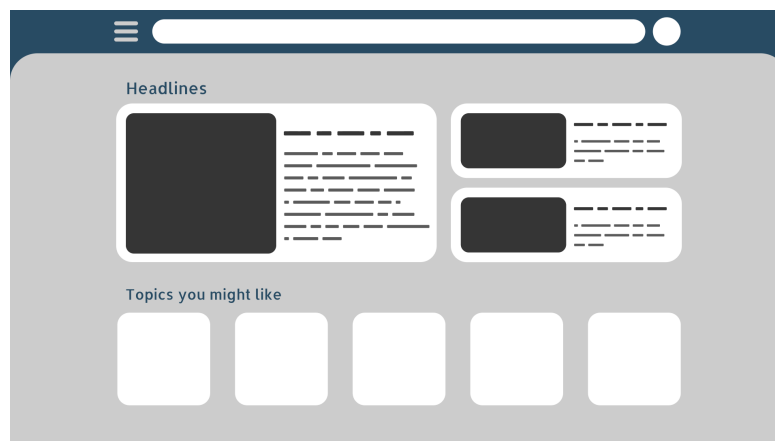
Om samenwerking te vergemakkelijken gebruiken we Sass (scss) omdat we makkelijk variabelen kunnen definiëren voor heel het project en het design consistent te maken over al onze pagina's. Het geeft ook de mogelijkheid om de stylesheets overzichtelijker te maken.

Design

(<https://www.figma.com/file/Csu2SlJyBfe77w5xmg6Wjp/News-app?node-id=0%3A1&t=374jmdViviNGn5Lf-1>)

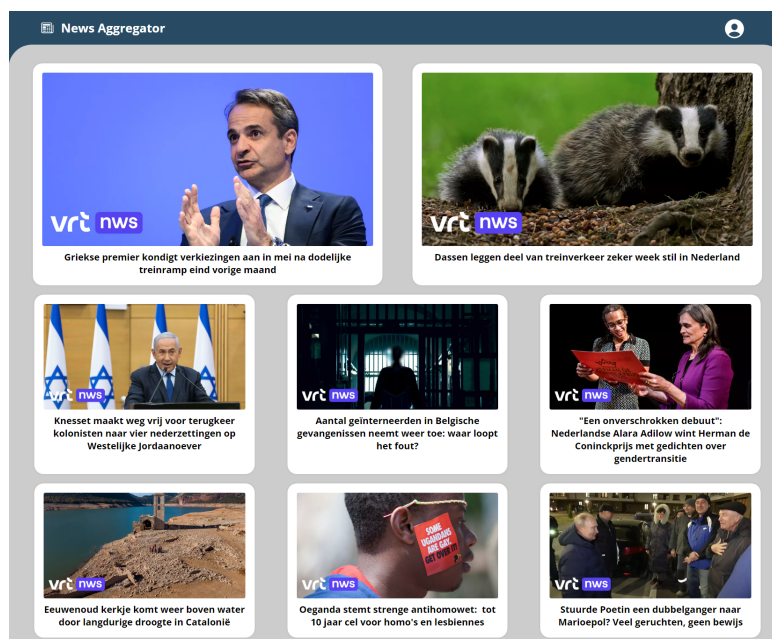
In deze sprint hebben we enkele veranderingen gemaakt aan onze plannen in verband met design. Hier en daar hebben we wat veranderingen gemaakt om het realistisch te houden.

Homepage

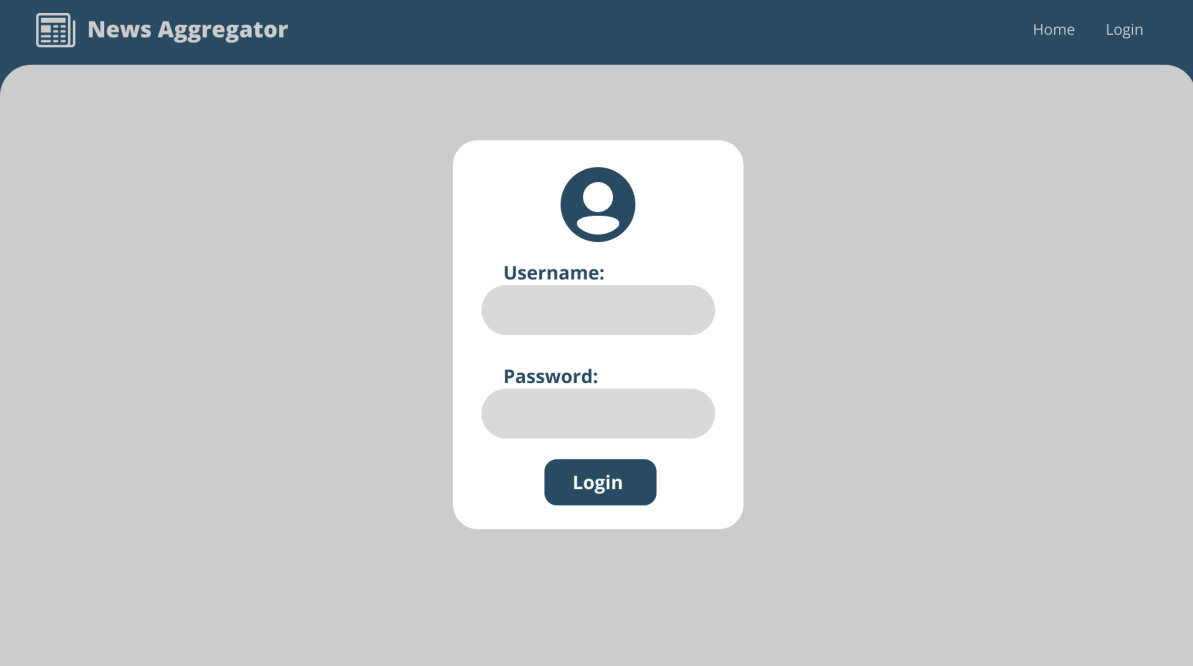


Het originele design heeft een hoop features hier die zichtbaar worden gemaakt. Origineel hadden we het idee om met een AI artikels te scrapen om zo tags hieruit te kunnen halen, meer belangrijke artikels eerst te plaatsen en ook een search functie te hebben.

Voor nu hebben we dit even in ons achterhoofd gehouden en de basisfuncties geïmplementeerd. Dit zie je ook op onze site nu. We hebben nu het gehouden om de artikelen te tonen en de meest recente groter te maken. In de header zie je wel dat er een gebruiker-knopje is. Momenteel is dit een prototype die we de volgende sprint hopelijk zullen uitwerken.




Login-page



The image shows a prototype of a login page for a 'News Aggregator'. At the top, there is a dark blue header bar. On the left side of the header is a small icon of a newspaper and the text 'News Aggregator'. On the right side of the header are two links: 'Home' and 'Login'. The main content area has a light gray background. In the center, there is a white rounded rectangle containing a user profile icon (a dark blue circle with a white person silhouette). Below the icon, there are two labels: 'Username:' and 'Password:'. Each label is followed by a light gray rounded input field. Below the input fields is a dark blue button with the text 'Login' in white.


Dit is slechts een prototype-pagina, we hebben het design wel deels versimpeld. Om alle pagina's aparte routes te geven, hebben we de login pagina van een pop-up naar een aparte pagina veranderd



This prototype shows a user profile icon (a gray circle) at the top. Below it are two buttons: a dark blue button labeled 'Register' and a light gray button labeled 'Login'.



This prototype shows a user profile icon (a gray circle) at the top. Below it is the text 'Hello, {user}' and a light gray button labeled 'Log out'.



This prototype shows a user profile icon (a gray circle) at the top. Below it is the text 'Hello, admin', a dark blue button labeled 'Admin page', and a light gray button labeled 'Log out'.

Admin-page

Hier hadden we nog niet echt een design voor gemaakt in Figma. Dus was dit vroeger puur prototype om te tonen wat we hier specifiek willen.

Het design hebben we maar recent af en hopen we in de volgende sprint in onze site te implementeren, dus momenteel zie je wel wat de basisfuncties zijn.

The image shows two side-by-side form prototypes on a light gray background. The left form is titled "Add new RSS Feed" and contains two input fields: "Feed URL:" and "Feed Name:". Below these fields are two buttons: a blue "Add Feed" button and a red "Delete Feed" button. The right form is titled "Add new Admin" and contains two input fields: "Username:" and "Password:". Below these fields are two buttons: a blue "Add Admin" button and a red "Delete Admin" button.

In de toekomst hopen we deze pagina wat complexer te maken met de mogelijkheid om de RSS-feeds te managen op de site-side, in de plaats van dat we dit in de backend enkel kunnen aanpassen.

Het design is dus nog niet volledig af maar geeft een basis weer

The image shows a full-page prototype of the News Aggregator admin interface. It features a dark blue header with the title "News Aggregator" and links for "Home" and "Login". The main content area has a light gray background and contains two white rounded rectangular cards. The left card is titled "Add RSS Feed" with an RSS icon and fields for "Feed Name:" and "Feed URL:". The right card is titled "Add Admin" with a key icon and fields for "Username:" and "Password:".

Error-pages

Hier hebben we in Figma nog geen specifiek ontwerp voor gemaakt en is dit momenteel nog een prototype op de site.



Error 404: Page not found

The page you are looking for does not exist.



[Return to Home](#)