01 - Rijksmuseum applicatie

Bouw een applicatie die gebruikmaakt van de Rijksmuseum API. Met de applicatie kunnen mensen zoeken op naam van kunstenaar (*Rembrandt*, *Vermeer*) en zien ze een lijst met kunstwerken van deze kunstenaar. Vervolgens kunnen gedetailleerde gegevens over het betreffende kunstwerk worden getoond.

Je kunt dit opzetten volgens het principe van microservices:

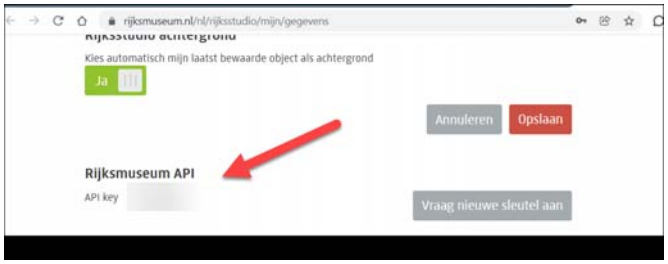
1. **Je app praat met jouw (gateway-) server.**
2. **Server stuurt request naar de API en transformeert/manipuleert desgewenst de resultaten.**
3. **Server stuurt kant-en-klare gegevens terug naar je app.**

De applicatie heeft de volgende requirements:

- De app gebruikt een front-end [framework] naar keuze: Angular, Vue , React, Flutter, Nuxt, Nest.js, Vanilla HTML/CSS/JavaScript, of nog anders.
- De architectuur van de app gebruikt het API Gateway-principe. De Rijksmuseum-API *is* geschikt om rechtstreeks aan te spreken vanuit je frontend en JSON retour te geven, maar probeer dit op te zetten volgens het in de sessie besproken microservices-principe.
- Zorg voor aantrekkelijke vormgeving. Als je een webapp maakt, mag je een CSS-framework zoals Bootstrap gebruiken, maar dit is niet verplicht. Je mag ook aanvullende libraries als Angular Material of PrimeNG gebruiken. Dit is ook niet verplicht. Bij Flutter/Ionic/… gebruik je natuurlijk de mobiele widgets die voor dat platform beschikbaar zijn.
- De app moet bruikbaar zijn op mobiele apparaten (responsive).
- De app bevat een zoekvak en een knop met voldoende aanwijzingen in de userinterface.
    - Als de bezoeker een kunstenaar invult en op *Zoeken* klikt, wordt een lijst met kunstwerken getoond.
    - Bij elke beschrijving wordt een kleine afbeelding van het kunstwerk getoond.
- Selectie van een kunstwerk zorgt ervoor dat meer details voor dat kunstwerk worden getoond.
- Zorg ervoor dat de app schaalbaar is. Gebruik het Single responsibility-principe voor componenten. Gebruik services voor communicatie met het backend.
- Optioneel: gebruikers kunnen een kunstwerk als Favoriet aanmerken. Het wordt dan bewaard in HTML5 localStorage op het eigen device.

WERKWIJZE:

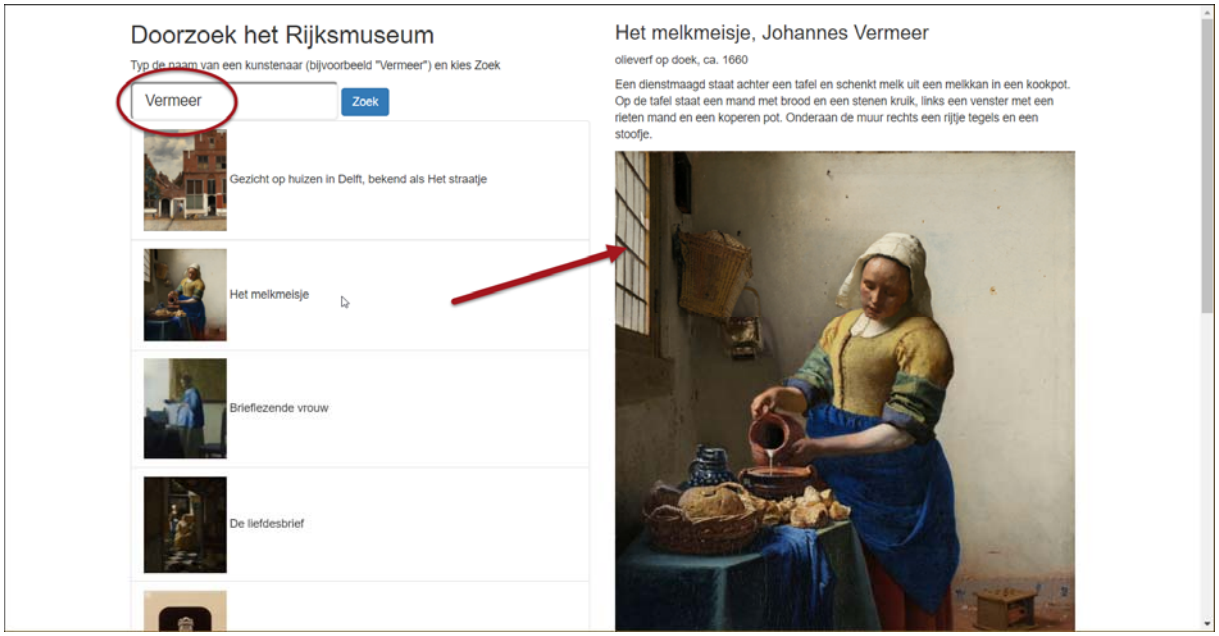
- **Frontend**: gebruik een CLI van jouw voorkeur om het project te starten. Voeg eventueel aanvullende libraries of packages toe.

- Schets op papier hoe de lay-out eruit komt te zien. Houd rekening met desktopgebruik en mobiel gebruik (bijvoorbeeld via Chrome DevTools).
- **Backend**: maak een kleine Node.js-server.
  - Deze moet geschikt zijn voor het ontvangen van requests, het eventueel transformeren en doorzetten van de request naar de API en het terugzenden van de ontvangen data in een response.
  - Gebruik hiervoor bijvoorbeeld de packages `express`, `request`, `node-fetch`, `got`, of nog een andere.
  - Het gebruik van TypeScript is optioneel. Als je dit doet, zorg dan zelf voor goede compilatie.
- Meld je aan voor gebruik van de Rijksmuseum API. Dit gaat via https://data.rijksmuseum.nl/object-metadata/api/ en/of http://rijksmuseum.github.io/. Er zijn meerdere stappen:
  - Maak een nieuw account voor *Rijksstudio*. Gebruik je Facebook-account om aan te melden, of meld je aan via e-mail/wachtwoord.
  - Bevestig eventueel je aanmelding (als je hebt gekozen voor e-mail/wachtwoord).
  - Vraag een API-key aan via Mijn Rijksstudio (https://www.rijksmuseum.nl/nl/rijksstudio/mijn/gegevens). Dit doe je via *Gegevens*, *Geavanceerde gegevens* (onderin). Je krijgt je key via mail toegestuurd.
  - 

  - Lees de mogelijkheden voor API-gebruik op Github pages en bekijk eventueel de voorbeelden. Een request om alle kunstwerken over Vermeer op te vragen, ziet er bijvoorbeeld zo uit: https://www.rijksmuseum.nl/api/nl/collection?q=vermeer&key=<jouw_key>&format=json
  - Onderzoek zelf hoe detailgegevens over een werk kunnen worden opgevraagd.
  - Optioneel: bouw *paginering* in. De user ziet dan bijvoorbeeld eerst een lijst met 10 kunstwerken en links voor Volgende / Vorige. De parameters hiervoor staan beschreven op Github, onder het kopje Collection.
- Plaats in jouw app de userinterface en logica om de service aan te roepen.
- Schrijf services met bijvoorbeeld als methodes `getArtist(<name>)` en `getDetails(<object-number>)`. Laat deze service retourneert asynchroon de opgehaalde data.
  - Als je er tijd voor hebt, mag je een eigen typing/interface voor artiesten en kunstwerken schrijven (aanbevolen).
  - Breidt de service uit met methoden die jij nodig acht.

- Lees ook de documentatie, om parameters mee te sturen naar de API die je wilt (bijvoorbeeld de toevoeging `imgonly=true`, als je alleen resultaten wilt waarvoor ook een afbeelding beschikbaar is. Zie http://rijksmuseum.github.io/ voor meer parameters).

De applicatie kan er bijvoorbeeld als volgt uitzien (hier als webapp):



- Tot slot, optioneel: genereer een distributie-build van je applicatie en publiceer deze op internet. Je kunt hiervoor bijvoorbeeld gebruik maken van gratis hosting op Github Pages, Netlify of Heroku.
- Stuur de URL naar de docent ter beoordeling ;-)
- In de volgende sessie (28 februari 2022) worden enkele random cursisten gevraagd om zijn/haar project(en) te presenteren.

Veel succes!

Peter Kassenaar, info@kassenaar.com
