

Starter ved å lage to mapper, «frontend» og «backend».

1. npm create vite@latest pokedex-app --template react.
2. Select a framework: React
3. Select a variant: JavaScript
4. cd pokedex-app
5. npm install
6. npm run dev
7. Får http://localhost:5174/

Pokedex-app er nå under frontend-mappen. Må innstallere avhengigheter, først bruker jeg `npm install react-router-dom axios`.

React-router-dom brukes for å routing, slik at flere sider kan lages der det kan navigeres mellom uten å måtte laste siden inn på nytt. Får komponentene `<BrowserRouter>`, `<Link>`, `<Switch>` og `<Route>` ved å bruke denne.

Axios brukes for å lage asynkrone HTTP-forespørsler fra Node og nettleser. Siden jeg skal hente inn data fra API, vil dette brukes.

Lager filene

Home.jsx som vil være forsiden og håndterer Main Pokemons og Types.

Teams.jsx som viser de tre teams.

SearchResults.jsx som tar for seg søkeresultater.

Type.jsx som tar for seg pokemon som tilhører en bestemt gruppe.

Pokemon.jsx viser detaljer som hører til en spesifikk pokemon.

I **App.jsx** så legges routing til, det er her jeg importerer inn sidene som er lagt til for at det skal være mulig å navigere seg på sidene.

Måtte bruke `npm install react-dom@latest`.

I **App.jsx** endret jeg fra `<Switch>` til `<Routes>` pga feilmelding.




Måtte utføre `npm install --save @fortawesome/react-fontawesome @fortawesome/free-solid-svg-icons @fortawesome-svg-core` for å få ikonet i søkefeltet.

Semantisk tagger:

Det er lagt inn etter semantiske regler, etter å ha undersøkt nærmere så er det greit at `<section>` er innefor `<article>` i følge HTML5. Det finnes noen få `<div>`-tagger, men de har gjerne fått `className` i seg.

Jeg la inn først kode i **Pokemon.jsx** som hentet data ved å bruke axios for å gjøre enkel GET-forepørsel, men denne henter ikke detaljer bortsett fra navn på Pokemonen men på bildet, på GitHub så viser det også detaljer derfor må jeg gjøre det på en annen måte. Den første koden brukte staten data for å lagre og `front_detail` sprite blir brukt for bilde av Pokemon. Derfor endrer jeg på koden, der den kan hente Pokemon-data ved å bruke `fetch` API og gjør et tilleggskall for å hente ut detaljer av evne(abilities), effekter (effects) og korteffekter(short effect).

Den håndterer to states, en for data om pokemon og en for evnene. Bruker section i HTML-tager og CSS klasser for struktering av innhold i UI-struktur. Har lagt til feilhåndtering, både for hoveddataforespørsel og tilleggskall for evner, viser da feilmelding dersom en mislykket forespørsel. Bytter ut front_default til official-artwork som også gir høyere kvalitet. Byttet ut Pokemon mot PokeDetail. Sånn så den ut først:

```
frontend > pokedex-app > src > pages >  Pokemon.jsx >  Pokemon >  data.stats.map() callback

1 //pokedex-app/src/pages/Pokemon.jsx
2 import React, { useEffect, useState } from 'react';
3 import axios from 'axios';
4 import { useParams } from 'react-router-dom';
5
6 const Pokemon = () => {
7   const { pokemon } = useParams();
8   const [data, setData] = useState(null);
9
10  useEffect(() => {
11    const fetchPokemon = async () => {
12      const response = await axios.get(`https://pokeapi.co/api/v2/pokemon/${pokemon}`);
13      setData(response.data);
14    };
15    fetchPokemon();
16  }, [pokemon]);
17
18  if (!data) return <div>Loading...</div>;
19
20  return (
21    <article>
22      <h1>{data.name}</h1>
23      <img src={data.sprites.front_default} alt={data.name} />
24      <section>
25        <h2>Types</h2>
26        <ul>
27          {data.types.map((typeInfo) => (
28            <li key={typeInfo.type.name}>{typeInfo.type.name}</li>
29          ))}
30        </ul>
31      </section>
32      <section>
33        <h2>Stats</h2>
34        <ul>
35          {data.stats.map((stat) => (
36            <li key={stat.stat.name}>
37              {stat.stat.name}: {stat.base_stat}
38            </li>
39          ))}
40        </ul>
41      </section>
42      <section>
43        <h2>Abilities</h2>
44        <ul>
45          {data.abilities.map((abilityInfo) => (
46            <li key={abilityInfo.ability.name}>{abilityInfo.ability.name}</li>
47          ))}
48        </ul>
49      </section>
50    </article>
51  );
52 };
53
54 export default Pokemon;
```

Oppretter så en fil som heter PokeDetail.jsx.

Sanity-delen:

Bruker npm install -g @sanity/cli for å installere sanity klient. Går så videre inn til mappen som har sanity, altså backend ved *cd backend*, for å skrive inn *sanity init --create-project «pokedex-backend»*. Altså kaller prosjektetnavnet for pokedex-backend, dataset vil bli production. Når dette er gjort, kan jeg åpne ved å skrive inn *npm run dev* i terminal. I og med jeg har gjort de høyere arbeidskrav så vil porten være opptatt, må avslutte disse for

å kunne bruke denne i dette prosjektet (bruker taskkill /PID 7756 /F). Deretter kan jeg logge meg inn på Sanity, der jeg bruker GitHub-kontoen. Der finner jeg prosjekt-id, som er «glvgqvzp».

Check-list:

- To hovedmapper, frontend (inneholder React-prosjekt med Vite) og backend (inneholder Sanity prosjektet). ✓
- Prosjekt satt opp med React og Vite, ikke brukt *create-react-app*. ✓
- Ingen genererte sider har mer enn tre div-tagger. ✓
- Bruk av semantiske tagger. ✓

Når det gjelder hvilken karakter som ønskes å oppnå, gikk jeg tidligere for høyest i Pokedex- men pga noen feilmeldinger som ikke løste seg i Sanity så valgte jeg å lage nytt prosjekt med nye filer, og heller velge den laveste karakteren istedenfor. Jeg har tatt noen koder og gjenbrukt dem, fra det prosjektet jeg trodde skulle bruke.

Jeg begynner først å gjøre det som karakter E er, og måten jeg har gjort det på er beskrevet overfor.

Har satt en maks-størrelse for skjerm, som 1200px.

Lagt til responsivt design.

Karakter C/D:

Jeg har gjort litt blandet her, siden jeg i tillegg går utifra demo-videoen som er lagt ut og på GitHub. Begynner først med å endre siden om Pokemonene, for å se ut som det som skal i karakterkrav D. Går deretter på Home.jsx.

For å hente bilde av pokemons på home-siden, brukes axios som henter de første ni pokemons fra API, for hver pokemon hentes full data for den spesifikke pokemonen for å gi tilgang til bilde, kaldt for sprite.

Jeg regner med å få en E, men ikke dårligere siden jeg har gått for minst alle minstekravene men siden det er en blanding av C og D i tillegg, så kanskje det virker rotete og derfor rakkes ned på karakter. Jeg mener det fortjenes å få minst E.

Det er bare jeg som har stått for dette eksamensdokumentet, så Ifcrazycat er Louise Fosdahl.

Kilder:

https://www.w3schools.com/react/react_usestate.asp

https://www.w3schools.com/react/react_useeffect.asp

https://www.w3schools.com/react/react_router.asp

https://www.w3schools.com/react/react_forms.asp

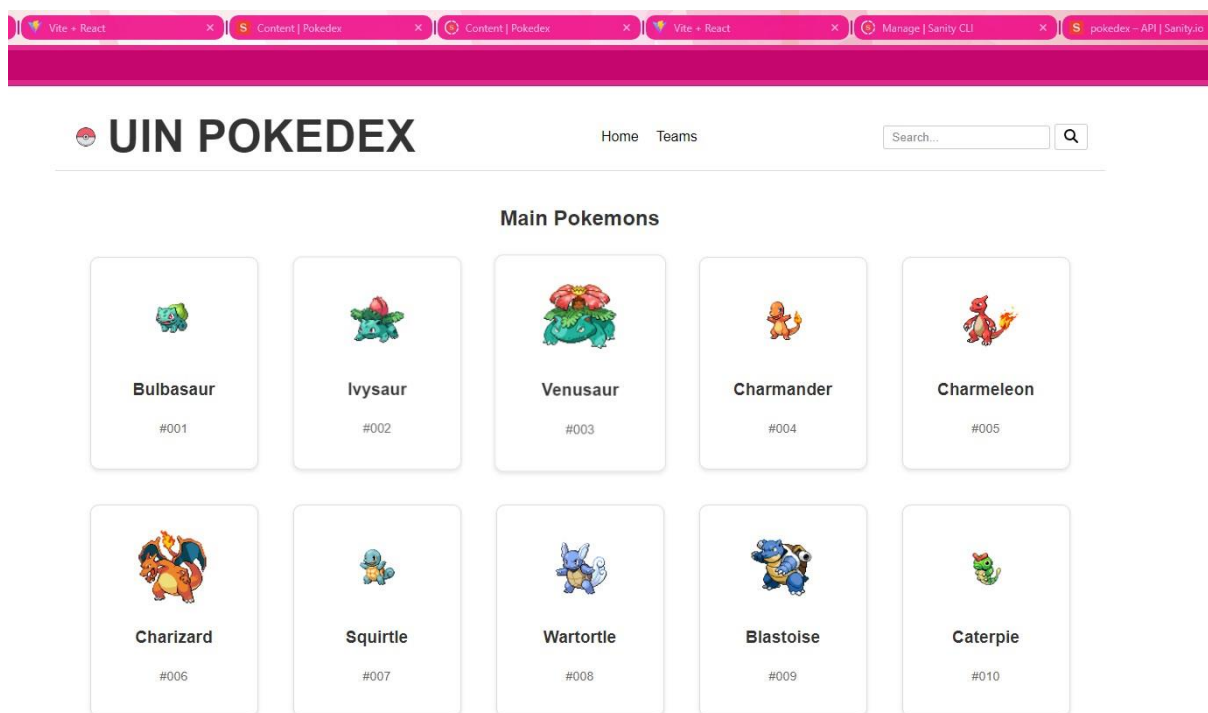
https://www.w3schools.com/react/react_props.asp

https://www.w3schools.com/react/react_render.asp

Hentet søkefelt-ikon fra fontawesome. <https://fontawesome.com/icons/magnifying-glass?f=classic&s=solid>

Vedlegg:







Screenshot fra tidligere, i et annet prosjekt som ikke ville fungere til slutt og derfor ikke kunne leveres.



CORS origins

+ Add CORS origin

Hosts that can connect to the project API.

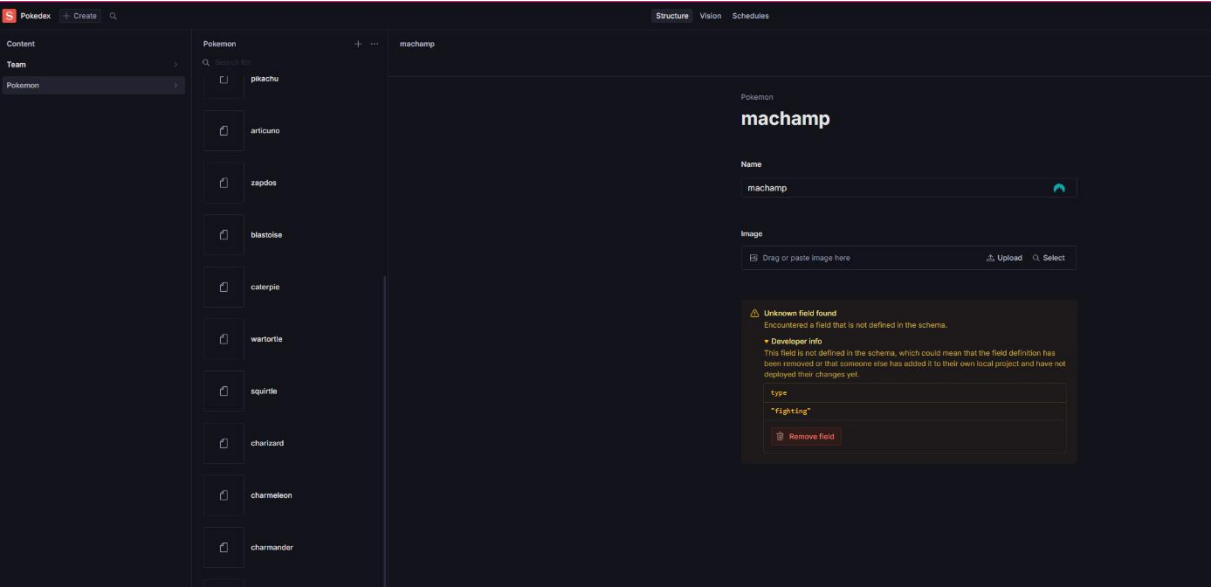
ORIGIN	CREDENTIALS	CREATED	
http://localhost:5173	Allowed	10 minutes	
https://pokedex-fosda.sanity.studio	Allowed	21 minutes	
http://localhost:5176	Allowed	8 hours	
http://localhost:3350	Allowed	21 hours	
http://localhost:3334	Allowed	yesterday	
http://localhost:3333	Allowed	2 days	
...			

Tokens

+ Add API token

Tokens are used to authenticate apps and scripts to access project data.

NAME	PERMISSIONS	CREATED
Upload Pokemons	Editor	22 hours



UIN POKEDEX

[Home](#) [Teams](#)

Search...



Grass Poison

Bulbasaur

Stats

Hp: 45
Attack: 49
Defense: 49
Special-attack: 65
Special-defense: 65
Speed: 45

ABILITIES

Overgrow. When this Pokémon has 1/3 or less of its HP remaining, its grass-type moves inflict 1.5× as much regular damage.
Chlorophyll: Während strong sunlight ist die speed eines Pokémon mit dieser Fähigkeit doppelt so hoch wie normal. Dieser Bonus zählt nicht als stat modifier.