Din Mægler Rapport / Dokumentation

Tech Stack:

Grundlæggende teknologier:

- **HTML:** Bruges til at strukturere vores indhold på siden.

- **Tailwind:** Bruges til at style siden så det passer til figma'en vi har fået udleveret.

- **JavaScript/TypeScript:** Bruges til at skabe/tilføje funktionalitet, bl.a. for at skabe interaktion for brugeren. Der bruges TypeScript i dette projekt.

Frameworks:

- **CSS Frameworks:** Tailwind benyttes for at gøre det lettere at designe hjemmesider hurtigere. Helt basic så har Tailwind lavet classes for os, som vi så kan benytte i stedet for, at vi selv skal skrive diverse classes.

- **JavaScript Framework:** NextJS er et fork af React. NextJS benyttes for at hjælpe os med at bygge siden op i små dele (komponenter) som kan genbruges flere gange. + Ved hjælp af dette frameworks så er det også nemmere at videreudvikle / vedligeholde. Det hjælper bl.a. også med ting som server rendering og states.

Biblioteker:

- **Tailwind**: Bruges til at style siden så det passer til figma'en vi har fået udleveret.

- **react-icons:** Bruges til svg ikoner. Ved brug af dette bibliotek slipper vi for at skulle downloade hver enkelt svg fra figma'en, hvilket kan tage et godt stykke tid + ved brug af dette bibliotek så får vi også mange muligheder som f.eks. at farve ikoner, gøre dem større osv.

- **zod:** Bruges til form validering. F.eks. skal man i et type:email input ikke kunne skrive følgende; "sebastian@.dk" da det er en ugyldig email. Ved hjælp af zod sørger den for at det kun er gyldige mails, der kan skrives.

- **cookies-next:** Bruges til at få adgang til cookies på et client komponent. Ved brug af Next's cookie, der kan man ikke få adgang til cookies på client komponent, kun på server komponent.

API:

- **DinMægler:** Følgende API brugt: https://dinmaegler.onrender.com/

Dynamisk Indhold og implementering:

Dynamisk indhold er blevet implementeret ved hjælp af at fetche på *https://dinmaegler.onrender.com/homes* på den måde har vi modtaget alle husene som vores udleveret API indeholder. Derefter er de blevet gemt i en useState så vi kan få fat i vores data og vise det på siden. Og vores fetch data har vi console.logget for at være sikre på at vi får de ønskede dataer. (Naturligvis slettet ved aflevering)
Eksempel:

```
const fetchHouses = async () => {
    try {
        const result = await fetch(`https://dinmaegler.onrender.com/homes`)
        const data: Home[] = await result.json()
        console.log(data)
        setHouses(data)
    } catch (error) {
        console.error("Failed to fetch houses:", error)
    }
}
```

Dataindsendelse til API'et

Jeg har også implementeret så man kan melde sig til nyhedsbrevet på forsiden. Normalt når man bruger en API så bruger man GET metoden, men når man skal sende noget til API'en skal man benytte POST metoden.
Eksempel:

```
const [email, setEmail] = useState<string>("")
const [message, setMessage] = useState<string>("")

async function handleSubmit(event: React.FormEvent<HTMLFormElement>) {
event.preventDefault()

try {
  const response = await fetch("https://dinmaegler.onrender.com/subscribers", {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
    },
    body: JSON.stringify({ email }),
  })
```

Her der har vi brugt POST metoden og det vi sender til API'et er vores "email" som står i body: JSON.stringify({ email }) - Vores email er den email som brugeren skriver ind i vores input.
Mailen bliver valideret af zod, for at sikre os at det kun er gyldige mails der bliver sendt til vores API.

Test af siden

- **API:** API'en er blevet testet og debugget ved hjælp af chromes developer tools f.eks. ved at tjekke diverse fejl i console og ved hjælp af console.logs som vi selv har implementeret i koden.

Ændringer af design ift. det udleveret design?:

- **Hover effects:** Der er blevet lavet en hover effect med farven Orange (text-orange-400) da den passer godt til den mørkeblå farve i designet. Hover effects kan findes på diverse knapper på siden samt, links, mails, telefonnumre.

- **Mine Favoritter:** Mine Favoritter i navbaren er gjort "usynlig" hvis man ikke er logget ind og det er gjort ved at fortælle navbaren at hvis dm_userid ikke findes i cookies så skal der ikke vises "Mine favoritter" i navbaren. Dette er en fin måde at gøre det på i denne API, men i andre API'er så vil en token udløbe efter et hvis antal timer/dage og det er der ikke i denne API.

- **Footer:** Footeren er ikke lavet 100% ud fra figma'en grundet at den ikke er blevet prioriteret ift. de andre ting på siden.

- **Kontakt en mægler:** Kontakt en mægler er ikke lavet, grundet at den ikke er blevet prioriteret ift. de andre ting på siden

- **Søgefunktion:** Søgefunktionen har ikke noget funktionalitet igen grundet at den ikke er blevet prioriteret ift. de andre ting på siden. Men måden jeg vil have lavet den på er at den skulle "filtrere" gennem vores huse med det valgte søgeord man havde skrevet og på den måde finde frem til huset eller de huse som indeholder de søgeord.

  Man kunne også bruge api'ets eget søge funktion, jeg ville have valgt løsning 1, da jeg allerede havde lavet en context med alle husene og på den måde undgår jeg at skulle fetche igen: eksempel på API'ets søgefunktion:

  

  Det dårlige ved denne er at du kun vil kunne søge på ovenstående og ikke på f.eks. energylabel, price, postalcode, city, rooms, livingspace osv.

Konklusion:

- **Hvad lykkedes godt?:** Design af hjemmesiden lykkedes rimelig godt, jeg synes at designet ligner figma'en udmærket. Det er ikke til perfektion, men det ligner figma'en udmærket.

  API'et var nemt at fetche på og en udmærket dokumentation om den også. Så overall så var API'et nemt at benytte.

  Er du tilfreds?: Jeg er udmærket tilfreds med det færdige produkt.