**Sluttraport it prosjekt vår**

**Produktbeskrivelse**

Denne gangen bestemte jeg meg for å lage et nettsted hvor man kan lære programmeringskonsepter, algoritmer og skjerpe ferdighetene innen programmering og problemløsning. Inspirert av sider som “W3schools”, “free code camp” og “leet code”. Planen var at man får en oppgave som å slå sammen to sorterte arrays til en sortert array. Man skal kunne skrive programmet på nettsiden i en IDE lignende de man kan finne på pc’en som “Atom” og “VS code”. Dette programmet blir sendt til en API som kjører koden på en trygg måte og tester den opp mot noen gitte tester (testcases). Hvis koden passerer alle testene har man klart oppgaven. I tillegg til å passere testene gjelder det å lage en løsning med optimal tids og lagrings kompleksitet. Dette burde også bli beskrevet i oppgaven. Det skal gå ann å legge inn tips man kan se på hvis man ikke helt vet hva man skal gjøre samt en chat for å diskutere løsninger med andre. For å gjøre dette må det være mulig å logge inn. Landings siden inneholder diverse informasjon og forklaring på hvordan man bruker nettstedet og hva man kan lære der. Nettsiden burde i hovedsak være responsiv mellom en pc skjerm og en ipad siden disse enhetene er nok de mest brukte enhetene for programmering. Hjemmesiden derimot burde være responsiv helt ned til telefon størrelse ettersom dette kan være hvor mange søker opp nettstedet og leser om det før man begynner å prøve å løse oppgavene.

**Her er link til nettsted skissen min.**

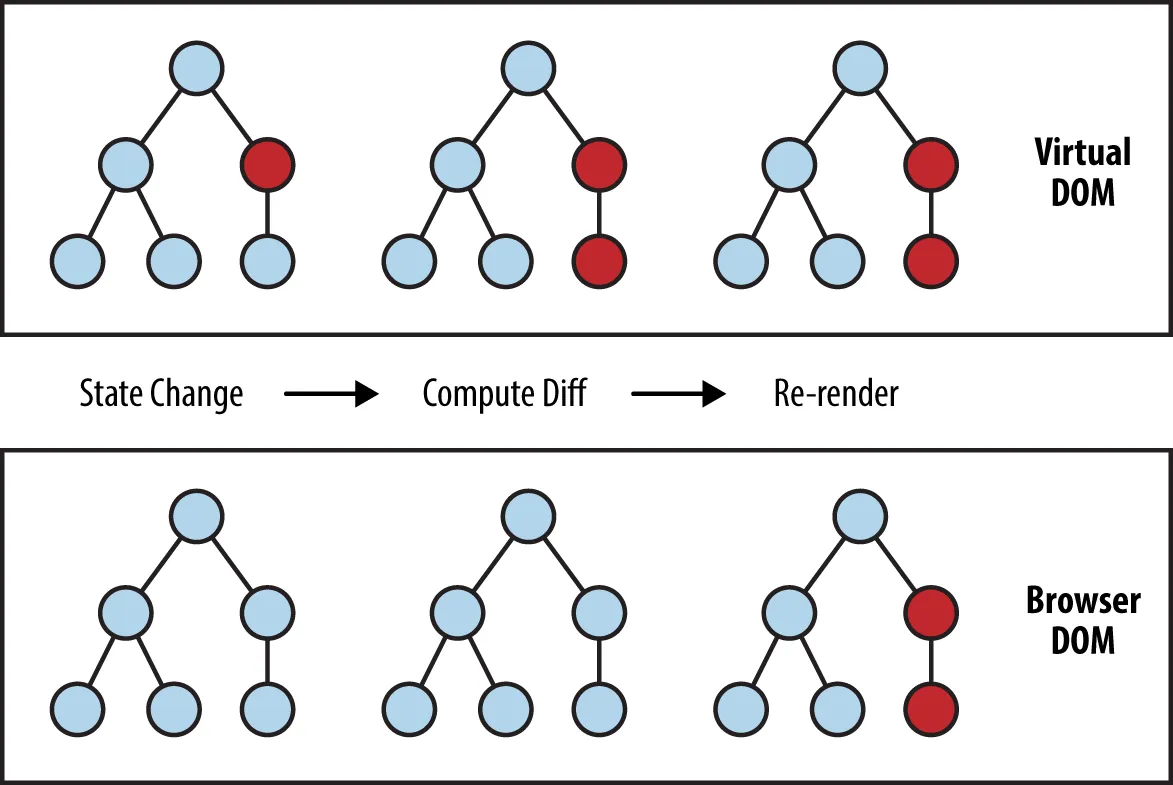
[https://wireframepro.mockflow.com/view/M8a68bb86783835fc4b987b77941a81be16](https://wireframepro.mockflow.com/view/M8a68bb86783835fc4b987b77941a81be1614850924128#/page/928cd0cb085b4720920114e12b5e9f95)

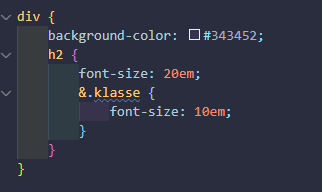
[14850924128#/page/928cd0cb085b4720920114e12b5e9f95](https://wireframepro.mockflow.com/view/M8a68bb86783835fc4b987b77941a81be1614850924128#/page/928cd0cb085b4720920114e12b5e9f95)

**Og dette var Planen min.**

|  |  |  |
| --- | --- | --- |
| **Ukenr** | **Plan** | **Frister** |
| 7 | Bestemte meg for nettside og designet planla og tegnet siden. |  |
| 8 | VINTERFERIE |  |
| 9 | Startet å lage text editor nettsiden. Har startet med utseende (css og html). | **Prøve 4/3 (datamodellering)**  **5/3: Innlevering av skisse og framdriftsplan** |
| 10 | Programmere API og fortsette med html og css text editoren og spørsmål siden |  |
| 11 | jobbe videre med API og koble til database | **Innlevering av tre skjermbilder av nettstedet 19/3** |
| 12 | koble til database |  |
| 13 | PÅSKEFERIE |  |
| 14 | Lage landingssiden |  |
| 15 | forbedre siden med ekstra funksjoner | **Innlevering av tre skjermbilder av nettstedet 16/4** |
| 16 | Ferdigstille nettstedet og fikse bugs |  |
| 17 | Ferdigstille nettstedet og fikse bugs | **Innlevering av ferdig prosjekt 30/4** |

For å lage denne nettsiden lærte jeg meg å bruke mange nye redskaper. Noen av nødvendighet andre av interesse og nysgjerrighet. Kanskje den største forandringen var å lære meg React js.

React er en javascript library for å lage brukergrensesnitt (UI). React ble gitt ut i 2013 av facebook med åpen kildekode og vedlikeholdes og oppdateres stadig av facebook i et felleskap med andre selskaper. React brukes oftest til å lage enkelt-side nettsteder eller “single page application”/SPA’s og er optimalisert for å raskt hente og skifte data på nettstedet. I praksis vil dette si at “hele” nettstedet vil bli lastet inn en gang når man går inn på siden. Dette betyr også at det ikke er noen lastningstider mellom sidene som får det til å føles veldig ut som en vanlig applikasjon på pc’en og ikke som en nettside. Dette ser man også i koden min hvor man kun vil finne et html dokument. Dette er fordi man i stedet for vanlig javascript skriver man “JSX” i React. Disse JSX filene må så kompileres til vanlig javascript før det kan leses av nettleseren. Ett av kjerne konseptene i React er komponenter. Komponenter er små deler av nettsiden som kan brukes om igjen flere plasser. Ett godt eksempel på dette kan være navigasjonsbaren eller footeren. I stedet for å holde står på footeren i hver av de forskjellige nettsidene kan man nå importere navigasjonsbaren og bruke den som et HTMl DOM element . React bruker også et virtuelt DOM, dette gjør at oppdateringer av nettsiden blir svært mye raskere. Når man endrer et element i DOM vil React sammenligne endringene i det virtuelle DOM et med vanlige DOM og endre kun de nodene som blir påvirket. React kode compilers web hjelp av Webpack, og bruker Babel for bakoverkompatibilitet som vil si at man kan skrive ES6 javascript som kompilert til en tidligere javascript versjon for å virke på gamle enheter. Webpack er også innstilt til å minske filstørrelse gjennom mange lure triks som å minifisere koden (fjerne alle unødvendige tegn fra kildekoden). React har også innebygd ESLint som hjelper med å finne feil og skadelige mønstre i koden. For ekstra sikkerhet bruker man også ofte å kombinere React med Typescript som introduserer statiske typer til javascript som gjør det enklere å oppdage bugs. 

Sammen med React bruker jeg også SCSS. SCSS er et “supersett” av CSS som betyr at den legger til ekstra funksjoner men at all gyldig CSS er gyldig SCSS. Noen eksempler på ekstra funksjoner er variabler, nestede regler og mixins. Den funksjonaliteten jeg har brukt mest er nok nestede regler som vi ser til høyre. Alle diver har en bakgrunnsfarge, alle h2 inne i en div har skriftstørrelse 20em og hvis h2 elementet har klassen “klasse” vil skriftstørrelsen være 10em. SCSS har som mål å gjøre det enklere og mer effektiv å skrive css. SCSS er et forhåndskompilert språk og kompileres til CSS ved hjelp av webpack. 

Jeg har brukt github for versjonskontroll gjennom hele prosessen og det har reddet meg flere ganger. For å publisere nettsiden min brukte jeg Netlify, dette fungerer veldig bra i kombinasjon med github og hver gang jeg dytter (git push) filene til github kompilerer Netlify automatisk applikasjonen til et optimalisert produksjons “build” og gjør den tilgjengelig på nettet.

**Mappestruktur**

Når vi åpner mappe ser vi en netlify.toml fil, dette er konfigurasjonsfilen til Netlify som sørger for at alt publiseres på riktig måte. Videre ser vi to mapper en remote code mappe, dette er koden til API’en som kjører kode fra nettsiden. Vi ser også en folder nettside,dette er kildekoden til selve nettsiden (Frontend). Her ser vi en mappe som heter node modules, dette er forskjellige avhengigheter til react og prosjektet. Vi ser også en .gitignore fil som sier hva som er unødvendig å laste opp til github. Vi ser to filer package.json og packagelock.json disse holder styr på forskjellige regler avhengigheter, script som skal kjøres for å kompilere nettstedet. de to viktigste mappene er src (source/source code) og public. I public finner vi det ene html dokumentet jeg snakket som inneholder en div med id root som er referert til i fra javascriptet. Det ligger også et favicon der som er det lille bildet som vises øverst i nettleseren, samt et robots.txt for å ekskøludere webcrawlere fra nettsiden. Det ligger også en manifest.json fil der som gir informasjon til mobiler om app ikon og lignende. hvis vi går ut igjen og inn i src mappen finner vi selve koden. I App.js finner vi koden som definerer hva de forskjellige subdomenene skal vise og om man må være logget inn for å se de. Videre ser vi den komponent baserte stilen til React. vi ser tre mapper den første er pages. Inne i pages finner vi de forskjellige sidene på nettstedet. Vi kan ta for oss landing\_page mappen. Inne i Denne mappen finner vi en javascript fil og en tilhørende SCSS fil. Dette er de to hoved filene som bygger opp landingssiden. På grunn av landingssiden sin statiske natur er det få eksempler på komponenter på akkurat denne siden. I mappen ser vi også flere foldere med bilder brukt på landingssiden. Ett stykke nede i javascript filen finner vi en referanse til teksteditor komponenten. Denne komponenten brukes på landingssiden som et eksempel for å vise hvordan teksteditoren ser ut og fungerer. Hvis vi går ut igjen finner vi components mappen. Denne mappen inneholder forskjellige komponenter som ikke i seg selv er en hel nettside men bare en liten del av den. Et godt eksempel på dette er navbar mappen. Denne mappen inneholder all koden både SCSS og JSX til navigasjons baren og blir brukt på både landingssiden og spørsmålsliste siden.

Den siste mappen er helper. Her ligger all ikke visuelle komponentene som for eksempel firebase konfigurasjonsfilen config.js hvor firebase authentication, realtime DB og Storage blir initialisert. Det ligger også en screen-size.js fil her som oppdaterer state til skjerm bredde og høyde hver gang vinduet blir justert. Det er også en delay på dette så den maks oppdaterer dette hvert tiende millisekund. authentication-context ligger også her og holder styr på all informasjonen tilknyttet en bruker og forskjellige metoder å oppdatere dette på som en login og logut funksjon.

**Her er link til nettsiden min**

<https://frederik-it.netlify.app/>

**Dokumentasjon**

PS: har bare fyll tekst på nettsiden men forklarer hvordan jeg hadde sett for meg teksten hvis dette skulle blitt en ordentlig nettside.

Når man åpner nettstedet havner man på landingssiden. På denne siden er målet å overbevise brukeren hvorfor de burde bruke dette nettstedet. Det første vi ser er en stor logo som sier gir oss en ide om hva nettsiden handler om. Siden er delt opp i seksjoner markert med en skrå endring i bakgrunnsfarge og hver seksjon har forskjellig informasjon. Overskriften overlapper halvveis med med en skrå bakgrunn som endrer farge i et slags bølge mønster. Fargen på overskriften endrer seg med fargene i bakgrunnen og er et resultat av fire overskrift elementer oppå hverandre for å få den ønskede effekten. Til høyre for overskriften ligger det et bilde skrudd i motsatt retning fra seksjonen for å skape kontrast. I seksjon to finner vi det fotoshoppede bildet. Bildet har en dobbel eksponerings effekt og er inspirert av lignende bilder med natur eller bylandskap i bakgrunnen. Jeg valgte å prøve dette med kode i bakgrunnen for å knytte det til temaet på nettsiden. Under dette ligger det mer i dybden hva nettsiden er om og hvordan man laster opp koden til en server som kjører koden for deg og slipper å tukle med installering av diverse språk IDEer og annet oppsett. Linker til spørsmål siden er også spredt utover seksjonene for å gjøre det enkelt å gå videre i prosessen.

I neste seksjon står det informasjon om hvilke programmeringsspråk som er støttet. Ikke alle er foreløpig implementert og bare python, javascript, java, C og C++ fungerer foreløpig. Neste seksjon handler om konseptene man kan lære om på nettstedet og man holder musen over hver av konseptene for å lese litt mer om de. Den femte seksjonen hadde jeg telt feil under skisseringen siden jeg ville at teksteditoren skulle være i en farget seksjon men dette gikk ikke opp så denne seksjonen er foreløpig tomm. Neste seksjon viser hvordan tekst editoren ser ut og fungerer. Den siste seksjonen er annmeldelser hvor folk kan skrive anmeldelser og hjelper i å overbevise brukeren om hvor bra produktet er. den siste delen er footeren som linker til diverse passende sosiale medier som github og youtube samt linker til ting som brukervilkår osv.

Hvis vi går opp til navigasjons baren igjen ser vi at siden du befinner deg på er markert med strek under. about siden er foreløpig tomm teksteditor linken har egentlig ingen nytte men linker til teksteditoren med et tilfeldig valgt spørsmål (ville byttet ut denne linken for noe mer meningsfylt i en ordentlig nettside). Questions linken bringer oss videre til siden hvor man kan velge en oppgave. Det er også en sign in knapp som bringer oss til login siden. Vi vil også bli brakt til denne siden hvis vi prøver å gå til en side man trenger bruker til for å være på, eksempler på dette er teksteditoren, legg inn spørsmål eller skrive en anmeldelse. Når du har logget inn/ laget en bruker vil du bli brakt tilbake til siden man prøvde å gå inn på.

På logg inn siden vil man kunne lage en bruker med google, github eller facebook eller med email og passord, dette gjelder også login. Hvis man har en bruker trykker man på logg inn på høyre siden som ikke har som er nesten lik menn inneholder login felt for email (google github og facebook har samme funksjon på begge sidene pga Oauth). Men den inneholder også en reset passord link som bringer deg til en ny side hvor man kan skrive inn emailen sin og firebase sender en email for å endre passordet. Hvis man logger inn en gang vil man fortsatt være innlogget når man kommer tilbake etter å ha lukket fanen.

Den neste siden er Questions siden. Her kan man velge hvilket spørsmål man vil prøve å løse. Man kan justere hvor mange spørsmål man vil se på en side nederst og klikke vidre til neste side (pagination). Man kan også legge til et spørsmål ved å trykke på submit a question. Ikke så mye spessielt her. kategoriene i dropdownen er bestemt av det som ligger i databasen som gjør det enkelt å endre senere.

Går vi tilbake og trykker på et spørsmål kommer vi til hovedfunksjonen på nettstedet. Her er nesten all dataen hentet fra databasen. Denne siden er ikke anbefalt å bruke på mobil siden det er en generelt dårlig opplevelse, men hvis man absolutt vil er nettsiden responsiv nokk til at det går fint. Øverst på skjermen ser vi forskjellige innstillinger som språk som bestemmer språk brukt av serveren og språk brukt til syntaks fremhevning. man kan velge skriftstørrelse og utseende på syntaks fremhevning. Bytte mellom en lys og mørk modus, med forskjellige syntaks fremhevinger slik at skriften alltid er synlig. Man har også en stoppeklokke hvis man vil ta tiden eller øve på å løse oppgavene fort som for eksempel er nyttig i programmerings turneringer. Skjermen er delt i fire(plassering avhenger skjermstørrelse men antar stor skjerm). Øverst til høyre ser man kategorien, vanskelighetsgraden, tittelen og selve spørsmålet, det ligger også hint og informasjon om optimal tid og lagrings kompleksitet I big O notasjon (generelt mer brukt enn andre som Big-omega, Big-theta og small-O men kunne gjerne også hatt med disse). Nede til venstre ligger testene dette er de testene serveren skulle ha testet koden mot. Øverst til høyre er teksteditoren ganske selvforklarende men inneholder mange nyttige snarveier som finnes i andre IDEer og kan stilles inn til å oppføre seg som F.eks atom eller kanskje nyttigere Vim eller Emacs (ikke foreløpig implementert). Det går også ann å skru på “code completion” men tenker dette kan ødelegge litt for læringsprosessen. nederst til høyre er hvor resultatet av koden vises. i vinduet øverst til høyre er det også en navigasjonsbar, trykker man på discuss kommer man til en chat for å diskutere spørsmålene. De fire vinduene er også justerbar og kan justere ved å trykke/holde på deleren mellom vinduene og dra de til den størrelsen man vil.

kan for eksempel være fin å gjemme alt annet enn programmerings vinduet når man fokuserer på programmeringen. man kan også da samtidig ha resultat vinduet litt åpent for å teste koden, på denne måten får man en veldig lik opplevelse som den man ville fått i en ordentlig IDE. Det er vel å merke at hvis man klikker på linken fra spørsmåls siden vil den ta med seg all den informasjonen som alleredet er lastet inn, men man kan også skrive inn URL’en og den vil da laste inn den samme oppgaven.

På landingssiden hvis man blar helt til høyre i anmeldelsene er det en link til å skrive en anmeldelse. Dette skjemaet er stylet helt likt de andre men her kan man også legge inn et profilbilde og logo til selskapet/ skolen man befinner seg i. Dette lastes da opp til firebase Storage og returnerer en link som blir lagret under anmeldelsen i databasen. Det er også en “loading bar” som forteller hvor lang tid det er igjen siden store bilder kan ta litt lengre tid.

**Endringer**

Har egentlig fulgt planen svært nøyaktig.

På landingssiden har jeg fjernet informasjonen over teksteditoren (hadde glemt den og merket det nå), her burde det stått informasjon om hvordan teksteditoren fungerer.

Anmeldelser seksjonen har jeg endret ettersom jeg la til muligheten å skrive inn selv trengte jeg noe som kunne håndtere flere anmeldelser. Jeg har også da lagt til siden for å skrive inn anmeldelser.

Jeg la også til en ekstra men foreløpig tom seksjon siden farg på grunn av tellefeilen.

Login siden hadde jeg ikke tegnet opp men hadde alltid planlagt å ha den.

På spørsmål siden har jeg endret bakgrunnen siden det så tomt ut uten den. navigasjons baren har også fått en skrå seksjon som jeg ser ut til å ha glemt å tegne in.

Jeg har lagt til muligheten å legge inn spørsmål selv, (pga tok lang tid å skrive direkte i firebase).

På teksteditor siden har svært lite forandret seg. Men har lagt til små navigasjonsbarer til hver boks, dette var nødvendig for å kunne bytte til diskusjonssiden som også var planlagt og skrevet ned i plan men glemt under skisse prosessen.

**Hva gikk bra**

Syntes det aller meste gikk veldig bra og er veldig fornøyd med å ha lært react ettersom dette var et svært nyttig hjelpemiddel.

**Hva kunne ha blitt gjort annerledes?**

anmeldelse seksjonen på landingssiden fungerer svært dårlig. Jeg prøvde å bruke en library siden jeg ikke visste hvordan man kunne få den til å fungere på mobil og ville heller fokusere på andre ting. Den laster ikke ordentlig når man laster inn siden (trykk på dottene neders for å få den til å virke) er ikke responsiv på noen måte og det hadde vært mye enklere å lage den selv.

På spørsmåls siden laster siden inn alle spørsmålene på en gang og mye annen informasjon som ikke er nødvendig for linkene. dette kan fikses med å restrukturere databasen litt å gruppere nødvendig informasjon til spørsmål siden og hente dette istedenfor å hente hele spørsmålet. Det var veldig vanskelig å få realtime DB til å sende bare noen spørsmål samtidig for å få pagineringen til å være mer effektiv ettersom det egentlig ikke er noe poeng i den nå siden alt er lastet inn uansett.

Kunne fått dette til menn må indeksere spørsmålene og sortere dem etter indeks for så å hente bare noen spørsmål mellom F.eks index 20 og 30.

I legg til spørsmål siden er det bare mulig å definere enn testcase. Hadde planer om og gjøre dette ved at man kan skrive inn tre “”” for å markere en ny testcase. skulle gjøre dette med regex på denne måten (**/"""([^"""]+)"""/g**) men har ikke rukket dette enda. det samme gjelder også hint. Kunne også brukt dette til å bruke kode eksempel i spørsmålet slik at alt mellom tre “”” erkode og blir framstilt med syntaks fremhevning. Input boksene for de forskjellige start funksjonene burde også hatt syntaks fremhevning. Liker ikke helt stilen på disse skjema nettsidene og ville endret den til å passe mer med resten av sidene.

I teksteditoren fungerer det meste som det skall men ble litt rotete kode på slutten for å fikse noen bugs veldig kjapt. navigasjons baren i teksteditoren ble også gjort responsiv veldig raskt med en dårlig men veldig enkel løsning som jeg ville gjort litt bedre hadde jeg hatt tid.

Diskusjon seksjonen er heller ikke ideell. Tenker den beste løsningen hadde vært noe lignende stack overflow og reddit hvor man kan publisere en thread som folk kan svare på og bruke kode eksempler med syntaks fremhevning. Det kunne også vært en ide og kunne skru av syntaks fremhevning for en ekstra utfordring.

Testcasene burde “blurres” ut fra starten av slik at man ikke ser de fra starten av siden dette kan gjøre det lettere å løse oppgaven. Dette Gjelder også hintene hvor jeg ville lagt til en funksjon at man må klikke på de for å vise hintene (dette ser vi også planlagt i skissen).

**Hva var mest utfordrende?**

Vanskelig å peke på en ting som den vanskeligste men det tar litt tid å bli vant til javascript sin ikke blokkerende event loop, altså at ikke alt nødvendigvis kjører i den rekkefølgen man har skrevet programmet i. Men jeg har lært mye og begynner nå å bli ganske solid med asynkrone funksjoner.

Jeg har lært veldig mye gjennom denne prosessen både om frontend og backend og skriver kode mye lettere nå enn når vi startet.

**Skisse og framdriftsplan**

Fulgte fremdriftsplanen veldig godt og har jobbet med stabilt tempo gjennom hele prosessen.

**Database**

Database strukturen er skrevet opp i databasestruktur.json filen.

Ikke så mye spesielt, men diskusjonen tilhørende en oppgave lagres utenfor oppgaven for å

hindre at den blir lastet ned på spørsmåls siden. Bruker istedenfor en “falsk relasjon” mellom

oppgaven og den tilsvarende chatten.

**Libraryer brukt**

Ace editor ble brukt for syntaksfremhvningen og de forskjellige IDE lignende funksjonene.

Reflex js ble brukt for å bevege på vinduene i teksteditoren.

En veldig dårlig library ble brukt på anmeldelsene.

**API**

Har ikke forklart så mye om hvordan kode kjørings API’en fungerer siden det kanskje ikke er like relevant, men den tar inn et json objekt.

{

"src": “kode”, // koden som skal kjøres

"stdin":"",

"lang": "python", // språket som skal brukes

"timeout":5 // hvor lang tid den får på å fullføre

})

Tar imot dette med en POST request. dette blir håndtert av express js.

Alt kjøres i docker kontainere for å hindre muligheten for å skade pcen med farlig kode. kan også sette maksgrense for ram og cpu kraft. Den ene docker containeren sender det videre til en taskQueue bygget med RabbitMQ. den andre kontainer en lytter på denne køen øg kjører koden. resultatet blir cache’et i en redis database (lett database key: value par lagret i ram). den returnerer en link som man sender en GET request for å få tilbake dataen. Serveren bruker da også node js som runtime.

Serveren er Port forwardet fra en raspberry pi.

**Den enkleste måten å se nettsiden på er denne linken**

<https://frederik-it.netlify.app/>

**Hvis ikke kjør**

**cd ./nettside**

**npm i**

**npm run Start**