Oppgave 069
Nordif.com
Systemdokumentasjon

Versjon <1.1>

Revisjonshistorie

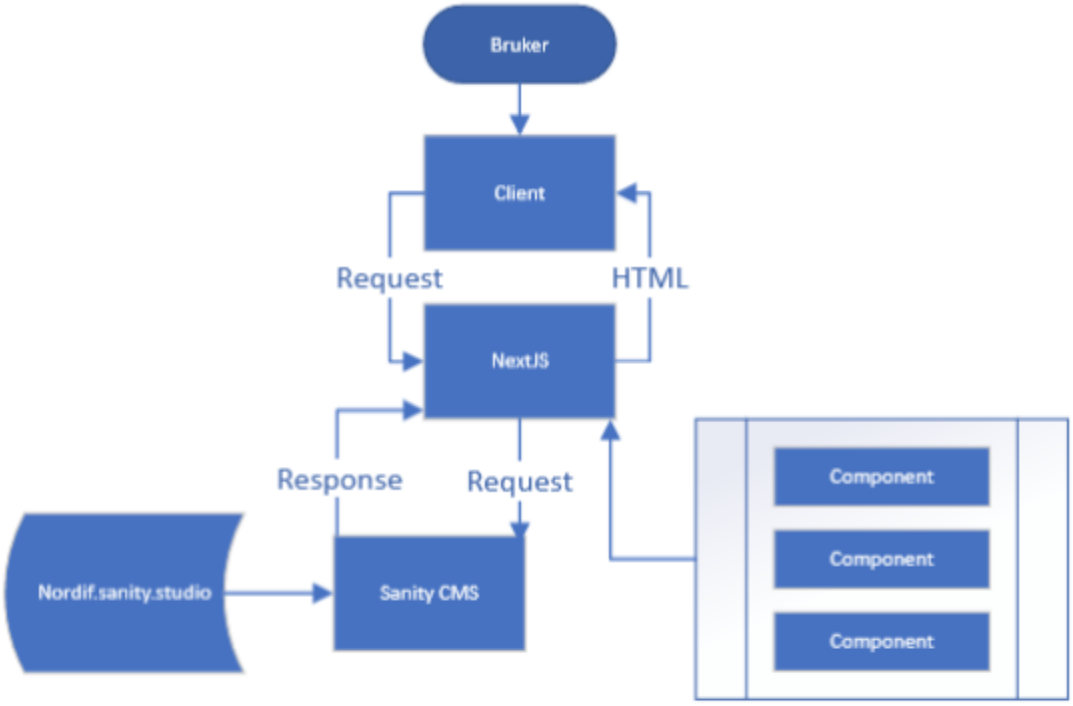
Dato	**Versjon**	**Beskrivelse**	**Forfatter(e)**
21.02.2023	0.1	Oppsett av kapitler	Magnar Dybwad Dæhli, Tore Andre Orheim
06.04.2023	0.2	Prosjektstruktur, Installasjon og kjøring	Tore Andre Orheim
19.05.2023	1.0	Sikkerhet, Databaseemodell	Tore Andre Orheim
21.05.2023	1.1	Revisjon før innlevering	Magnar Dybwad Dæhli, Tore Andre Orheim

Innholdsfortegnelse

1 Introduksjon

Dette dokumentet skal gi en oversikt over den overordnede strukturen til system som utvikles i forbindelse med bacheloroppgave i faget INFT2900. Hensikten med dokumentet er å gi en dyp forståelse for systemets oppbygging og hvordan deler av systemet kommuniserer seg imellom. Dokumentet inneholder informasjon som omhandler arkitektur, prosjektstruktur, tjenester, sikkerhet, installering og utføring av endringer, samt dokumentasjon av kildekode.

2 Arkitektur

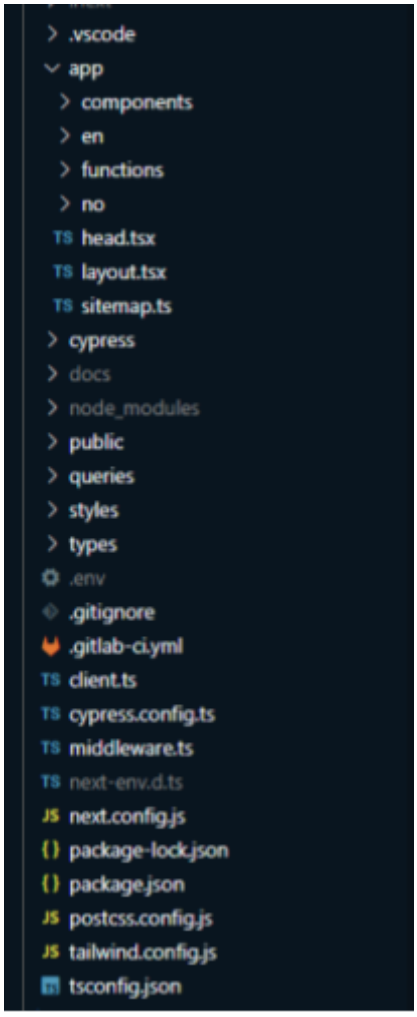


Figur: 1

Figur 1 gir et overordnet blikk på hvordan systemet er satt opp. Brukeren kobler seg til systemet gjennom klienten ved å navigere til domenet nordif.com. Klienten er gitt av Next.js ved bruk av sammensetning av komponenter som blir til sidene som brukeren kan navigere på domenet. Når brukeren navigerer sidene på klienten, gjør klienten etterspørsler til Next.js om hvilke komponenter som skal vises fram.
Next.js kommuniserer også med Sanity CMS for å hente data som komponentene trenger for å vise tekst og grafikk på domenet, Next.js etterspør komponentene, derfra etterspør Next.js CMS for data som blir hentet fra databasen på nordif.sanity.studio. Når daten er gitt tilbake til Next.js og kombinert med komponentene kan klienten se siden som er etterspurt.

3 Prosjektstruktur

3.1 Frontend



Figur 2.1

Figur x.x viser den overordnede filstrukturen til front-end delen av systemet.

Mapper:

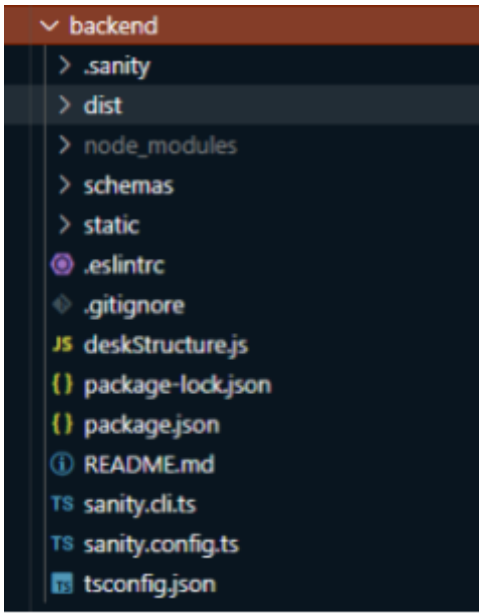
- App-folderen inneholder sidene, komponentene, og komponent-testene som blir brukt for å sette opp nettstedet som brukerne ser.
- Cypress mappen inneholder konfigurasjon til Cypress og E2E testene til systemet.
- Docs filen vil inneholder dokumentasjon som genereres ved kjøring av `NPX TYPEDOC`.
- Node_modules mappen inneholder avhengighetene som frontend systemet trenger for å bygge nettstedet.
- Public mappen holder filer som må være tilgjengelige for bruker ved kjøring.
- Queries mappen inneholder api-kall mot Sanity sitt CMS system for å hente innhold som blir vist på nettstedet.
- Styles mappen er i hovudsak ubrukt, med unntak for å aktivere TailwindCSS for utvikling.
- Types-mappen inneholder type-/interface-definisjoner som blir brukt gjennom Typescript for å hjelpe med utvikling.

Filer:

- .env inneholder sensitiv informasjon, som API-nøkler, noe som ikke kan bli vist til brukerne.
- .gitignore inneholder hvilke filer/mapper som ikke skal leveres til git-repoet som oppbevarer kildekode på nettet.
- Client.ts er frontend-klienten som lar Sanity kommunisere fra backend-delen av systemet.
- cypress.config.ts er konfigurasjonsfilen for Cypress sitt testrammeverk
- middleware.ts er en fil som inneholder kode som kan kjøres før en forespørsel er fullført, og basert på denne koden kan den endre responsen til forespørselen før den blir brukt i andre filer.
- Next-env.d.ts inneholder typedeklarasjoner for Next.js.
- Next.config.js er konfigurasjonsfilen som lar utviklere konfigurere Next.js for å passe til systemet som skal utvikles.
- Package.json og package-lock.json er avhengighets-filer, som definerer hvilke avhengigheter som blir brukt i frontend-delen av systemet og hvilke avhengigheter som vil installeres ved kjøring av `NPM INSTALL` inne i frontend-mappen.
- postCss.config.js er konfigurasjonsfilen for PostCSS.

- Tailwind.config.js er konfigurasjonsfilen for TailwindCSS, denne filen lar utviklere konfigurere TailwindCSS til systemets behov.
- Tsconfig.json er konfigurasjonsfilen for TypeScript, denne filen lar utviklerne konfigurere TypeScript til utviklernes behov.

3.2 Backend



Figur 2.2

Figur x.x+1 viser den overordnede mappestrukturen for backend-delen av systemet.

Mapper:

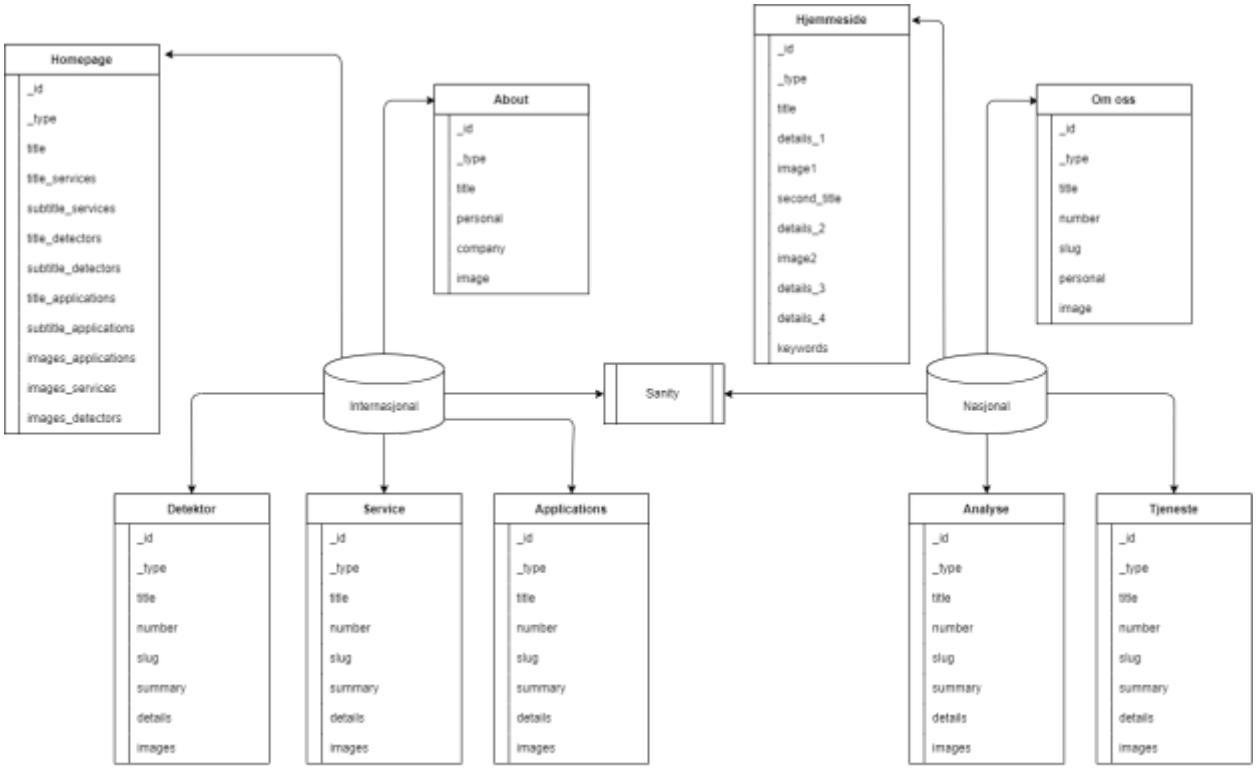
- .sanity er automatisk generert ved kjøring av `NPM RUN BUILD`.
- Dist inneholder produksjonsmiljøet til sanity og kommer ved kjøring av `NPM RUN BUILD`.
- Node_modules mappen inneholder avhengighetene som backend-systemet trenger for å bygge backend-delen av systemet.
- Schemas inneholder schemaer for innholdssidene som ligger i NORDIF sitt Sainty CMS.
- Static inneholder statiske filer for Sanity studio.

Filer:

- .eslintrc definerer hvordan kildekode skal formateres under utvikling slik at EsLint kan formatere koden som blir skrevet av utviklere.
- .gitignore inneholder hvilke filer/mapper som ikke skal leveres til git-repoet som oppbevarer kildekode på nettet.
- deskStructure.js lar utviklerne opprette strukturen til systemet sitt Sainty CMS.
- Package.json og package-lock.json er avhengighets-filer, som definerer hvilke avhengigheter som blir brukt i backend-delen av systemet og hvilke avhengigheter som vil installeres ved kjøring av `NPM INSTALL` inne i backend-mappen.
- Readme.md er en dokumentasjonsfil som forklarer hvordan en kan utnytte backen-delen av systemet.
- Sanity.cli.ts er konfigurasjonsfilen for Sanity CLI-en, denne inneholder prosjekt ID og datasettet som CLI-en skal koble seg opp mot.
- Sanity.config.ts er konfigurasjonsfilen for Sanity slik at utviklerne kan konfigurere Sanity slik at avhengigheten passer til systemets behov.
- Tsconfig.json er konfigurasjonsfilen for TypeScript, denne filen lar utviklerne konfigurere TypeScript til utviklernes behov.

5 Databasemodell

Systemet som er utviklet tar i bruk et CMS (Content Management System) fra Sanity.io. Dette CMS gir systemet data som er lagret i databasen. Vi valgte å ta i bruk Sanity CMS for å utvikle et system som er dynamisk og lett å utføre endringer/oppdateringer i, uten å måtte gå inn i kildekoden til systemet. Kommende figur gir en modell av databasen.



Figur xx

6 Server-tjenester

6.1 Endepunkter

Systemet har endepunkter mot Sanity sitt CMS, så disse endepunktene vil ikke returnere data ved bruk som et vanlig API f.eks. ved bruk av Postman.

Metode	**Endepunkt**	**Beskrivelse**	**Rettigheter**
GET	/en/about	Henter data med typen about	Bruker på siden
GET	/en/applications	Henter all data med typen applications	Bruker på siden
GET	/en/products	Henter all data med typen products	Bruker på siden
GET	/en/products/:id	Henter data med typen product som har id	Bruker på siden
GET	/en/services	Henter data med typen services	Bruker på siden
GET	/en/services/:id	Henter data med typen service som har id	Bruker på siden
GET	/no/analyser	Henter all data med typen analyser	Bruker på siden
GET	/no/analyser/:id	Henter data med typen analyser som har id	Bruker på siden
GET	/no/tjenester	Henter all data med typen tjenester	Bruker på siden
GET	/no/tjenester/:id	Henter data med typen tjenester som har id	Bruker på siden
GET	/no/omOss	Henter data med typen omOss	Bruker på siden

Tabell: 1

6.2 Rettigheter

Systemet sine endepunkter kan kjøres av brukere på nettsiden. Dette siden alle endepunktene er GET metoder som betyr at endepunktene bare kan hente data fra systemet sin CMS.

7 Sikkerhet

Sikkerhet er noe som er satt i fokus under utviklingen av systemet. Hemmeligheter er lagret gjennom Vercel, som også hoster domenet. Ved å lagre hemmeligheter her kan vi sikre at disse ikke blir tilgjengelige til klienten, og derfra en uautorisert bruker.

A01 - Broken Access Control

OWASP A01 Broken Access Control handler om å beskytte api-kall mot brukere, slik at ingen kan ta i bruk kall utenfor sine rettigheter. Dette har vi implementert gjennom SANITY sin Access Control-, CORS- og medlemsroller. Foruten å for dypt i temaet har dette punktet åtte underpunkter:

1. Beskyttelse av API med medlemsprivelegier, eller blokker som standard.
   Sanity er dekket av dette, men ikke EmailJS. Gjennom Sanity er det ikke tillatt for brukere å utføre API-kall uten å ha korrekte API-nøkler.
   EmailJS lar ikke tjenesten beskyttes fra andre domener uten å betale for et oppgradert abonnement, av denne grunn er dette nevnt, anbefalt, samt også gjennomgått i Vedlegg C3 Tjenestedokumentasjon EmailJS.
2. Systemet tillater ikke endring av objekter basert på brukerdata.
3. API-en vår tillater ikke bruken av andre forespørsler enn GET
4. Det er ingen innlogging, så alle brukere er på samme "nivå" i systemet, dette gjør det umulig å etterligne en administrator.
5. Systemet tar i bruk CORS å har avgrenset bruken av Sanity til nordif.com/ domenet og sub-domener av denne.
6. Det er ingen autentisering så Force Browsing er ikke en vulnerabilitet for nettsiden.

A02 - Cryptographic Failures

Vi lagrer sensitive hemmeligheter i .env, som Next.js holder "server-side", ergo: disse blir ikke tilgjengelige for klienten.

A03 - Injection

Dataen som blir levert til EmailJS gjennom de eneste brukerinput-feltene på kontaktskjemaet blir validert og desinfisert før den leveres til EmailJS.

A04 - Insecure Design

Systemet er utviklet med sikkerhet i tankene, men dybden som OWASP A04 etterspør har vi ikke gått i, noe som vi vil anbefale å se på for videre utvikling.

A05 - Security Misconfiguration

Det er en mulig sikkerhetshull med ikke-implementerte sikkerhetsvalg i diverse tjenester, disse er ikke valgt i bruk for utviklingsprosessen, men har dokumentasjon og anbefales aktivert av kunde.

A06 - Vulnerable and Outdated Components

Komponenter i bruk av systemet er nylig opprettet og dokumentert, dette fører til god oversikt over komponentene som er i bruk og fører til godt vedlikehold.

A07 - Identification and Authentication Failures

Systemet utfører ikke noe form for identifisering eller autentisering, og av den grunn er ikke dette punktet av høy relevans.

A08 - Software and Data Integrity Failures

Produktet tar i bruk minimalt med avhengigheter og avhengighetene som er brukt er sett gjennom og tatt i vurdering mtp sikkerhet. Dette punktet kan arbeides mer med, ved tanke på CI:CD-segregering og -tilgangskontroll.

A09 - Security Logging and Monitoring Failures

Dette punktet er ikke mye utbedret og slike logger burde tas vare på for å sikre systemet på en bedre måte, dette burde utbedres ved videre utvikling.

A10 - Server-Side Request Forgery

Dette er atter et punkt som vi burde ha satt inn mer tid til å bli kjent med, gitt at vi utvikler et system som bruker SSR. For videre utvikling burde systemet ikke tillate domener som ikke er nødvendige for dette systemet.

8 Installasjon og kjøring

8.1 Nedlasting

For å installere systemet slik at en kan utføre endringer/videreutvikle systemet, må en først laste ned kildekoden fra repositoriet sin nettside på GitLab: https://gitlab.com/nordif/nordif. Dette kan en gjøre på flere måter, her skal vi gå gjennom to varianter med en anbefalt metode.

Nedlasting ved å klone repositoriet (Anbefalt):

Denne anbefalte metoden krever at en har installert git på sin datamaskin, om en ikke har git installert kan en installere siste versjon her: https://git-scm.com/downloads.

Inne på nettsiden til repositoriet kan en klone repositoriet ved å bruke HTTPS, eller SSH, begge disse klone-metodene utgjør samme resultat. Dersom en ønsker å klone med HTTPS, kopierer en lenken som ligger under dette valget. Deretter navigerer en i en terminal (for eksempel i windows, CMD) til mappen der en ønsker å lagre kildekoden lokalt. Etter skriver en inn i terminalen `git clone` og deretter den kopierte lenken fra nettsiden til repoet, for dette systemet: `git clone https://gitlab.com/nordif/nordif.git`. Etter en liten stund (avhengig av maskinspesifikasjoner) vil repositoriet være klonet til din maskin.

Nedlasting som fil:

På nettsiden til repositoriet kan en laste ned systemet som komprimert-fil (ZIP/TAR), disse filtypene må ekstraheres der en ønsker å lagre kildekoden lokalt.

8.2 Installasjon

Etter fullført nedlasting av repositoriet, vil en ha en mappe som heter NORDIF, inne i denne mappen ligger to mapper, backend og frontend.

For å installere backend-delen av systemet, må en navigere seg inn i backend-mappen gjennom en terminal så kjøre kommandoen `NPM INSTALL`, etter fullføring av denne kommandoen skal backend-delen av systemet være installert.

For å installere frontend-delen av systemet, må en navigere seg inn i frontend-mappen gjennom en terminal så kjøre kommandoen `NPM INSTALL`, etter fullføring av denne kommandoen er avhengighetene som frontend-delen trenger for å kompilere. Etter dette må en også opprette en fil inne i frontend-mappen som heter .env.local, inne i denne mappen må en skrive inn de nødvendige hemmelighetene (se kapittell 8.3 Miljøvariabler) som systemet trenger for å fungere normalt. Etter denne filen er opprettet, har en fullført installering av systemet.

8.3 Miljøvariabler

Som nevnt under installasjon kapittelet er det nødvendig å opprette miljøvariabler, uten disse kommer deler av systemet ikke kjøre som planlagt.

8.4 Kjøring

For å kjøre frontend-delen av systemet må en i en terminal som er navigert til frontend-mappen, kjøre kommandoen `npm run dev`, denne kommandoen starter en lokal versjon av nettsiden på localhost:3000. I en nettleser, naviger til localhost:3000 for å se nettsiden og få kontinuerlige oppdateringer ved endring av kildekode.

For å kjøre backend-delen av systemet må en i en terminal som er navigert til backend-mappen, kjøre kommandoen `npm run dev`, denne kommandoen starter en lokal versjon av Sanity som backend-del av systemet på localhost:3333. I en nettleser kan en navigere til localhost:3333 for å se NORDIF sitt Sanity-studio og kontinuerlige endringer som en utfører i kildekode. For å laste opp ny kildekode til NORDIF sitt Sanity CMS som ligger på nettet, må kommandoen `npx sanity deploy` kjøres i terminal som er navigert til backend-mappen.

9 Dokumentasjon av kildekode

Kildekoden er lagret på GitLab repositoriet https://gitlab.com/nordif/nordif. I kildekoden er det tatt steg for å skrive ryddig kode, ved hjelp av type-sjekking fra TypeScript, og gode navngivelses-prinsipper til variabler, funksjoner, klasser, O.L. Dette fører til at koden selv skal fungere som dokumentasjon, altså den er self-documenting. Vi har uansett tilført kommentarer i kildekoden som er TsDoc-kompatibel, dette betyr at en vil få hjelpetekst på funksjoner O.L. ved å holde musepekeren over disse. Siden kommentarene er TsDoc kompatible er det også mulig å bruke `NPX TYPEDOC` kommandoen i terminalen inne i frontend-mappen, for å generere en ny mappe, kalt "docs" som inneholder en .html fil som kan åpnes i nettleser. Denne samler alle kommentarer i koden og gjør den navigerbar.

10 Kontinuerlig integrasjon og testing

10.1 CI:CD

Systemet tar i bruk kontinuerlig integrasjon i regi av Gitlab sin CI:CD pipeline. Denne kjøres av skriptet i gitlab-ci.yml filen som ligger under frontend/ mappen (se kapittel 3.1 for filplassering).
Vercel har også en CI:CD pipeline som blir kjørt ved utvikling av ny build, dersom denne vil fungere som en "fail-safe" dersom det er kritiske problemer med versjonen som blir bygget.dersom denne feiler, vil det ikke være mulig å oppdatere nettsiden til ny versjon..

10.2 Testing

Cypress tester er utviklet for prosjektet, både på E2E (end to end) og CT (component) nivå. E2E testene tester systemet fullt ved å følge en brukerhistorie. CT testene er komponent-tester for å passe på at komponentene fungerer slik de skal ved endringer, dette vil da også sies at de må utvides om komponentene blir videreutviklet for å inneholde testkriterier for ny funksjonalitet.Disse testene vil kjøres gruppevis i tre nettlesere: Chrome, Firefox og Safari.
For å kjøre de forskjellige testene, kjør kommandoen i tabellen under i en terminal inne i /frontend mappen.

CY:CT	Komponent tester
CY:E2E:CHROME	E2E tester kjørt i Chrome
CY:E2E:FIREFOX	E2E tester kjørt i Firefox
CY:E2E:SAFARI	E2E tester kjørt i Safari