Teknisk dokumentasjon

Dawid Musial 2IMC

Introduksjon

Denne tekniske dokumentasjonen beskriver de ulike teknologiene og arkitekturene som ble brukt i utviklingen av Postly, en bloggplattform for lesing og publisering av blogginnlegg. Dokumentasjonen dekker frontend- og backend-komponenter, serverkonfigurasjon, databaseløsning, og integrasjon av versjonskontrollsystem.

Arkitektur

Prosjektet er bygget med en moderne webapplikasjonsarkitektur som skiller frontend og backend i separate moduler. Dette gir fleksibilitet i utviklingen og gjør det lettere å skalere og vedlikeholde applikasjonen.

1. **Frontend**
    - **Rammeverk**: React med Vite
    - **Program**: Visual Studio Code
    - **Stylingbibliotek**: Material UI (MUI)
    - **Rutestyring**: React Router
    - **Utviklings miljø**: Node, npm, pnpm
    - **Programmeringsspråk**: Typescript, Javascript, CSS og JSX
2. **Backend**
    - **Rammeverk**: ASP.NET med MVC-arkitektur
    - **Program**: Visual Studio
    - **Utviklings miljø**: Swagger UI
    - **Programmeringsspråk**: C#
3. **Server**
    - **Operativsystem**: Debian-VM
    - **Cloud**: Microsoft Azure
    - **Hosting**: Apache, dotnet SDK
4. **Database**
    - **Databasehåndteringssystem**: MongoDB

Frontend

Frontend-delen av applikasjonen er bygget med React, valgt på grunn av dens effektive utviklingsopplevelse og sterke fellesskap. Vite ble valgt som byggeverktøy for sin raske oppstartstid og optimaliseringsevne.

- **React**
    - **Komponentbasert arkitektur**: Reacts komponentbaserte tilnærming gjør det enkelt å utvikle og vedlikeholde modulær kode.
    - **Hooks**: Bruk av React Hooks for å håndtere tilstand og livssyklusmetoder.
- **Vite**
    - **Rask utviklingsserver**: Tilbyr en lynrask utviklingsserver med hot module replacement (HMR).
    - **Optimering**: Produserer optimalt byggetid og mindre bundler.
- **Material UI (MUI)**
    - **Komponentbibliotek**: Et omfattende sett av gjenbrukbare UI-komponenter.
    - **Tematisering**: Enkel tematisering og tilpasning av utseende.
- **React Router**
    - **Rutestyring**: Håndtering av applikasjonens navigasjon og URL-ruting.
- **Node**
    - **Utvikling:** Utvikling miljø med en localhost server.

Backend

Backend er implementert med ASP.NET, valgt på grunn av C#'s robusthet og ASP.NETs velutviklede verktøy for å bygge skalerbare webapplikasjoner.

- **ASP.NET MVC**
    - **MVC-arkitektur**: Separasjon av logikk, brukergrensesnitt, og data for bedre organisering av kode.
    - **Routing**: Konfigurerbar routing for å dirigere HTTP-forespørsler til de riktige kontrollerne.
- **C#**
    - **Språkfunksjoner**: Sterk typet, med moderne språkfunksjoner som async/await for asynkron programmering.

Server og Hosting

Applikasjonen er hostet på en Debian-VM i Azure med apache og SDKM som gjør det mulig å hoste frontend og backend, som gir pålitelig skybasert infrastruktur.

- **Debian-VM**
    - **Pålitelighet**: Stabil og sikker Linux-distribusjon.
    - **Konfigurerbarhet**: Full kontroll over servermiljøet.
- **Microsoft Azure**
    - **Skalerbarhet**: Enkel skalering av ressurser basert på trafikkbehov.

    - **Pålitelighet**: Høy tilgjengelighet og pålitelighet.

Database

MongoDB ble valgt på grunn av dens fleksibilitet og skjemaløse natur, som gjør det enkelt å tilpasse datamodellen over tid.

- **MongoDB**
    - **Dokumentbasert**: Lagrer data i JSON-lignende dokumenter for fleksibel struktur.
    - **Skalerbarhet**: Enkel horisontal skalering og replikering.

Versjonskontroll

GitHub ble brukt som versjonskontrollsystem, noe som muliggjør effektivt samarbeid og sporbarhet i kodede endringer.

- **GitHub**
    - **Versjonshåndtering**: Sporer endringer og tillater tilbakeføring til tidligere versjoner.
    - **Samarbeid**: Muliggjør samarbeid gjennom pull requests og kodegjennomganger.

Diagram:
https://app.eraser.io/workspace/EoRiqDosKC68Hy9T0bba?origin=share