

Vi har valgt å bruke MVC struktur for dette prosjektet. Dette er en struktur vi har jobbet med tidligere og er derfor et trygt valg. Denne strukturen er oversiktlig og gjør koden enklere å lese og navigere.

```
projekt/
----app/
    --- Controllers/
        --ChatbotController.php

    ---Models/
        --KassalappAPI.php
        --EANScannerModel.php
        --ChatBotModel.php

    ---Views/
        --ChatbotView.php
        --ProduktView.php
        --layout/
            -filer for eventuell styling

----public/
    --index.php
    --css/
    --js/
    --images/ ? -om nødvendig

----config/
    --config.php

----vendor/
```

----composer.json
----.gitignore

Controller

- tar imot handlinger fra bruker, henter data fra modeller og sender det til riktig view

Modeller

- inneholder forretningslogikk som f.eks. API-kall, prishenting, datastrukturering

Views

- Ingen logikk, bare HTML og PHP for å vise data

Public

- inneholder bare filer som er tilgjengelig fra nettleseren, som css, index, js og bilder

Config

- Inneholder API-nøkkel og eventuelle miljøvariabler

Vendor

- Composer-avhengigheter

Composer.json

- konfigurasjon for composer

Klasser som inkluderes

- kassalappAPI. Har ansvar for kommunikasjonen med API-et. Henter ut produktinformasjon
- strekkodeScanner. Har ansvar for å hente og formtere informasjonen som blir hentet via API-et, siden denne dataen er rå og ubehandlet. Tillater søk på produkt via EAN-kode.
- ChatbotController. Håndterer brukerens spørsmål og genererer svar tilbake.
- ChatbotModel. Tolker meldingen og henter relevant data.
- ChatView. Viser meldinger og svar i chatformat
- *DatabaseConnector. tilkobling til database (hvis vi skal implementere database)*

Funksjoner som må inkluderes i de forskjellige klassene

KassalappAPI

- `__construct`. Initialiserer Guzzle-klient med API-nøkkel
- `request`. Utfører GET-call til API og returnerer JSON data
- `handleError`. Håndterer API-feil og gir forståelige feilmeldinger til bruker

StrekkodeScanner

- `__construct`. Tar inn en API-instans
- `skannProdukt`. Henter produktdata basert på EAN-kode
- `formaterProduktInfo`. Rydder og strukturerer data fra API-et, trekker ut relevante data.

ChatbotModel

- `__construct`. Initialiserer med API-instans
- `hentRespons`. Tolker meldingen, henter data via StrekkodeScanner og returnerer tekstsvar
- `formaterSvar`. Lager forståelig svar basert på produktinformasjon

ChatbotController

- `__construct`. initialiserer ChatbotModel med API
- `handleMessage`. Tar imot melding og returnerer svar

Dette må løses:

blå markering betyr løst

- En måte å motta og tolke brukerens spørsmål
- En måte å hente prisdata fra [kassalapp.no](#) basert på EAN
- en måte å sammenligne prisdataen fra forskjellige butikker
- En måte å gi et enkelt svar tilbake til bruker
- eventuelt lagre svar og spørsmål i database?
- eventuelt ha forhåndsdefinerte svar i database?