Projektopgave - Del 3

Github url: [https://github.com/larsk7cdk/vejrportalen](https://github.com/larsk7cdk/vejrportalen-api)

Branch: del-3

Vejrportalen url: <http://40.127.170.50/vejrportalen/>

Til del 3 af projektopgaven har jeg arbejdet med følgende

* REST API i PHP som kan
  + oprette, opdatere, slette og hente et abonnement
* MySQL
* Postman
* Virtual Machine med PHP og MySQL i Azure

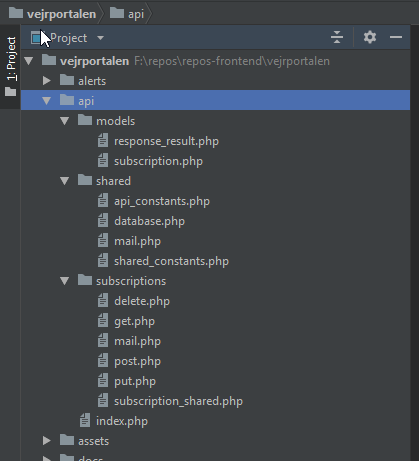
I løsningen for del 3 af projektopgaven har jeg lavet et REST API i PHP som gemmer data i en MySQL database. Derudover en HTML side hvor man kan indtaste sine oplysninger og dermed blive tilmeldt en abonnement service. For at kunne tilgå det, har jeg oprettet en LAMP server i Azure, ved brug af Bitnami.

REST API’et er lavet i PHP og har kun backend funktionalitet. MySQL er valgt som database, i stedet for .json filer. Jeg har testet API’et ved hjælp af postman.

HTML siden er opbygget med bootstrap i en form. For at kunne sende data til backend, er JQuery ajax benyttet.

## Vejrportalen API

Koden er placeret i sin egen mappe under api, i projektet på github. Mappe og fil strukturen for .php filerne er som vist på figur 1.



Figur 1 Mappe og fil struktur i api

Det er muligt at læse oprette, opdatere og slette poster via API’et. Hver af de nævnte operationer har deres egen Url

* vejrportalen/api/subscriptions/get.php – henter alle poster
* vejrportalen/api/subscriptions/get.php?id= – henter enkelt post ud fra id
* vejrportalen/api/subscriptions/post.php – opretter en post
* vejrportalen/api/subscriptions/put.php – opdaterer en post
* vejrportalen/api/subscriptions/delete.php?id= – sletter en post ud fra id

### Shared mappen

Her er samlet funktionalitet som er generel for API’et.

#### api\_constants.php

For at samle information et sted om API’et, så man f.eks. ikke skal ind i hver enkel fil og håndtere ERROR\_LEVEL, er dette samlet i api\_constants.php. Her vil også være information om MySQL indstillinger mail osv.

#### database.php

Denne fil indeholder funktionalitet som benyttes i forbindelse med MySQL databasen.

#### mail.php

For at kunne sende en mail når der oprettes et abonnement, er funktionalitet med mail samlet her.

#### shared\_constants.php

Her er konstanter som benyttes i hele API’et.

### Models mappen

Der er 2 forskellige klasser til brug for data. Klassen response\_result bruges ved svar på et request og indeholder status (”success” eller ”error”), message (beskrivende tekst) og i tilfælde af fejl, selve fejlen fra f.eks. MySQL.

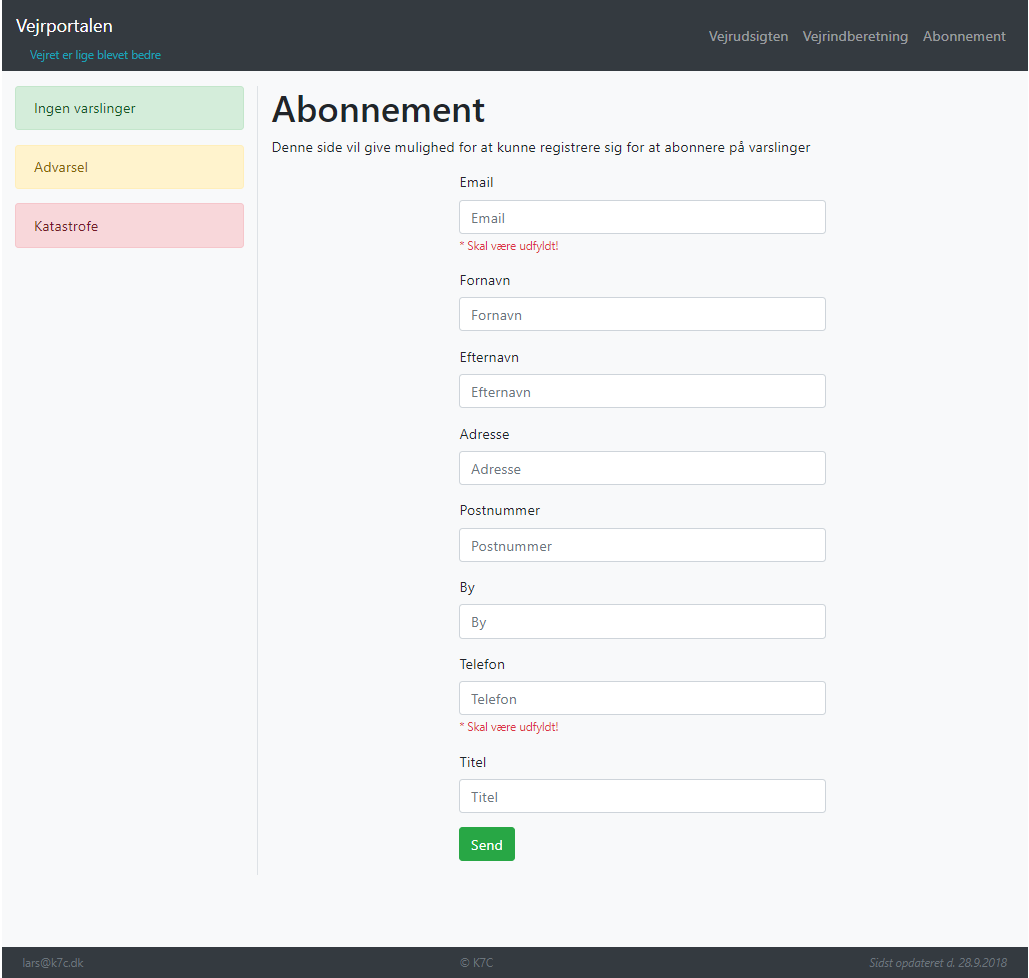
Klassen subscription indeholder strukturen for et request abonnement.

### Subscriptions mappen

De ovennævnte operationer er placeret i denne mappe. Alle operationer er bygget op efter den samme skabelon og selve afviklingen af hvad der skal ske ligger i subscription\_shared.php.

## Abonnement HTML siden

Denne side er bygget i et html form element, med bootstrap komponenter som vist på figur 2.



Figur 2 HTML abonnement siden

Ved at udfylde input felterne og klikke på send, sendes data til backend’en. Her vil der blive foretaget en kontrol af om e-mailen eksisterer i forvejen. Er dette ikke tilfældet, oprettes der en ny post som abonnement og sendes en mail (mail virker ikke pt.).

## Virtual Machine med PHP og MySQL i Azure

For at kunne teste portalen og give andre brugere adgang, har jeg oprettet en LAMP server i Azure ved brug af Bitnami.

Siden kan tilgås på url’en nævnt i starten af dokumentet.