


# Pakke portallen

## Eksamensprojekt - Programmering A

Mads Gørup Gjellerod Christiansen

Vejledt af: Gorm








 **Pakke Portalen**

[Forsiden](#) [Om os](#) [Min Side](#) [Test Side](#)

Velkommen oliversbutik@gmail.com! [Log Ud](#) →

Velkommen tilbage  
**Mads!**

Dine leveringer

 postnord	<b>DELIVERED</b> Galten, 8464 - Danmark	Afsender: ArduinoTech.dk Service: PostNord MyPack Collect Vægt: 0.040kg	<a href="#">Mere info →</a>
 postnord	<b>DELIVERED</b> Danmark, 0000 - Danmark	Afsender: Techbitshop Service: PostNord MyPack Collect Vægt: 5.760kg	<a href="#">Mere info →</a>
 postnord	<b>DELIVERED</b> Danmark, 0000 - Danmark	Afsender: Techbitshop Service: PostNord MyPack Collect Vægt: 5.760kg	<a href="#">Mere info →</a>
 postnord	<b>DELIVERED</b> Risskov, 8240 - Danmark	Afsender: NIKE Yusen Herentals Service: PostNord MyPack Home Vægt: 1.180kg	<a href="#">Mere info →</a>
 postnord	<b>DELIVERED</b> Risskov, 8240 - Danmark	Afsender: NIKE Yusen Herentals Service: PostNord MyPack Home Vægt: 1.180kg	<a href="#">Mere info →</a>
 postnord	<b>DELIVERED</b> Galten, 8464 - Danmark	Afsender: Batter1 Energ1 ApS Service: PostNord MyPack Collect Vægt: 0.010kg	<a href="#">Mere info →</a>
 postnord	<b>DELIVERED</b> Galten, 8464 - Danmark	Afsender: Batter1 Energ1 ApS Service: PostNord MyPack Collect Vægt: 0.010kg	<a href="#">Mere info →</a>

## Abstract

# Indhold

Abstract . . . . .	1
<b>1 Indledning</b>	<b>3</b>
<b>2 Fremgangsmåde</b>	<b>4</b>
<b>3 Problemformulering</b>	<b>5</b>
<b>4 Produktprincip</b>	<b>6</b>
4.1 Flowchart over programmet . . . . .	6
4.2 Produktkrav . . . . .	6
4.3 Tech-stack . . . . .	7
4.4 Database . . . . .	7
4.5 Asp.net Identity . . . . .	7
4.6 Wireframes . . . . .	8
<b>5 Test af produkt</b>	<b>9</b>
<b>6 Gennemgang af kode</b>	<b>10</b>
6.1 Frontend . . . . .	10
6.2 Backend . . . . .	11

# 1 Indledning

## 2 Fremgangsmåde

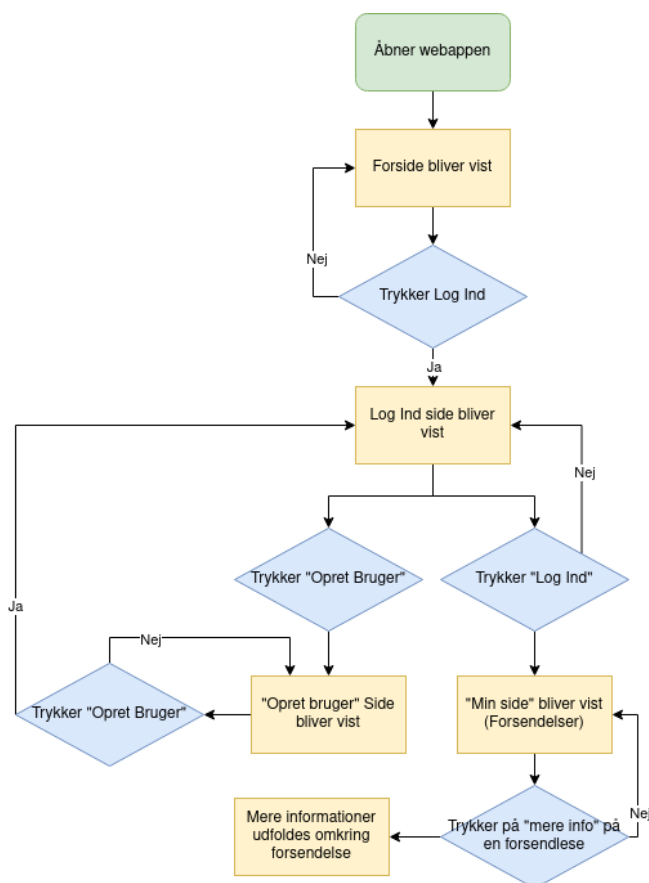
### 3 Problemformulering

Når man bestiller et produkt eller en vare online, findes der lige nu et væld af forskellige fragtfirmaer som udfører håndteringen og fragten af din varer, hver af disse tilbyder hver deres form for sporing eller tracking af en sådan forsendelse, men hvis man venter flere forsendelser fra forskellige af disse fragtfirmaer kan det hurtigt blive svært at finde rundt i. og der findes dermed heller ikke en løsning som samler alt dette sporingsinformation omkring sin pakke et samlet sted.

## 4 Produktprincip

Produktets vil være en webapp løsning, som ved brug af Googles bruger API og fragtfirma api'er såsom GLS's og Postnords API'er. skal kunne gennemse den brugerens gmail og derudfra hente trackingnummer på forsendelser fra en bred vifte af fragtfirmaer. denne tracking information skal derefter fremvises til brugeren på webappen, således tracking informationen fra alle brugerens pakker samles.

### 4.1 Flowchart over programmet



Figur 4.1: General funktion af programmet.

### 4.2 Produktkrav

Der er her opstillet en række hårde og bløde krav som det udviklede web app skal opnå. de er som følger:

#### Hårde krav

- Skal kunne tracke pakker fra mindst 2 forskellige fragtfirmaer
- Skal kunne fremvise pakke tracking fra mindst 2 forskellige fragtfirmaer det samme sted Oprette bruger
- siden skal have en lav kompleksitet (tilgå tracking på 2 klik)

#### Bløde krav

- Skal have et moderne stilrent design
- Oprette bruger med Google
- Skal fungere upåklageligt på mobile enheder (være responsivt)
- Skal selv kunne hente mails fra brugerens google konto

## 4.3 Tech-stack

Webapplikationens tech stack består af Html hvor vi bruger razorpages og til styling bruges css her bruger vi Tailwind som er et css framework. Der bruges også javascript og her bruger vi også javascript biblioteket jQuery som har til formål at programmere webapplikationer. Til backend bruges Csharp og som database bruges mssql.

## 4.4 Database

Til Databasen bruges MS SQL som er en relations database(RDBMS) udviklet af Microsoft. Hvilket giver mest mening når man laver Asp.net core mvc webapps da mange dele af dette bygger på at man bruger MS SQL. Til kommunikation mellem MS SQL og Asp.net core 7 mvc projektet.

Til at lave migrations til databasen bruges Entity Framework Core .NET Command-line tools. Hvordan så kan skrive Models i projektet og putte dem ind i vore Application database context. Her brugesto kommando'er første kommando laver en migration fra ApplicationDbContext.cs som er den databasecontext til vores database hvor der opbevares data om brugeren der skal gemmes. Den næste kommando opdateres så databasen med migration som vi lavede før. Af sikkerheds grunde er der valgt at databasen er delt i to en hvor alt login data bliver gemt og en hvor data om bruger bliver gemt. Det gør at hvis der bliver udført et sql angreb er det kun den en af databaserne der bliver ramt.

```
1 > dotnet ef migrations add IntialUserData --context ApplicationDbContext
2 > dotnet ef migrations add IntialUserLogin --context IdentityDbContext
3 > dotnet database update --context ApplicationDbContext
4 > dotnet database update --context IdentityDbContext
```

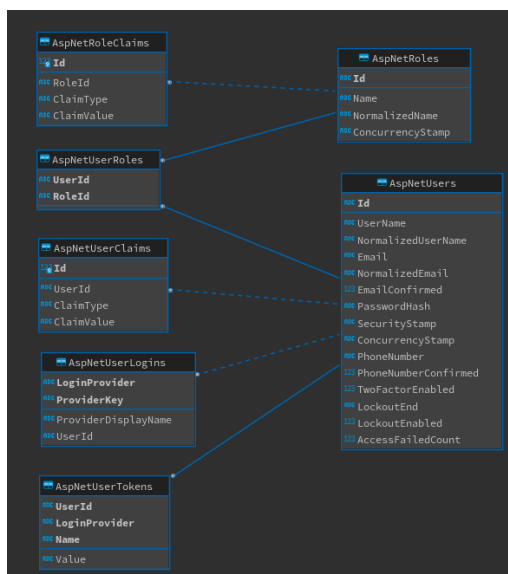
På figur 4.2 kan man se den overordnet database struktur. Denne database er lavet med Asp.net core Identity som er en API som giver nogle login funktioniteter til ens program og kan findes som en nuget package. Denne Api genere selv modellerene til systemet.

## 4.5 Asp.net Identity

Asp.net core er det som bliver brugt til håndtere autentificering og autorisation i webapplikationen. Denne måde at gør det på giver en standardiseret måde at håndtere dette på. Identity giver også indbygget funktioniteter til at håndtere brugerregistrering, login, håndtering af adgangskoder og meget mere.

### 4.5.1 Google OAuth

Google OAuth bruger OAuth som er en åben protokol som alle kan bruge. Denne protokol bygger på HTTP protokolen. Ved hjælp at OAuth kan man som tredjepartsapplikation få adgang til brugers data. Det er det vi skal bruge den data vi godt vil have fundt i fra brugeren er at læse brugers emails. Til at få emails fra Google bruges Googles gmail API. som gør det muligt at læse alle brugers emails når de er logget ind med OAuth.

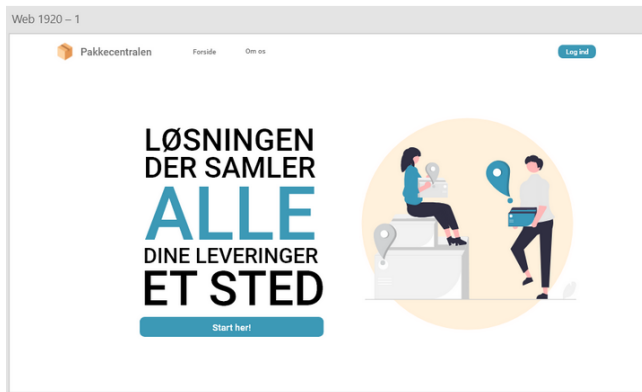


Figur 4.2: Uklip af User tablen i Databasen

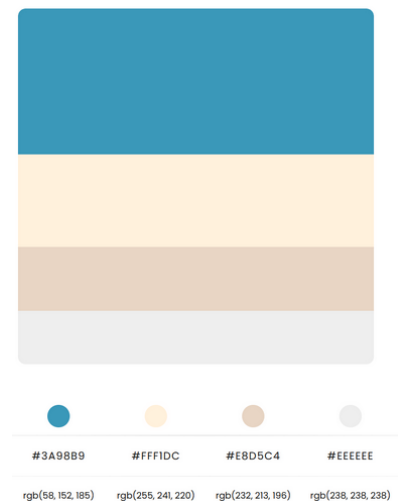


## 4.6 Wireframes

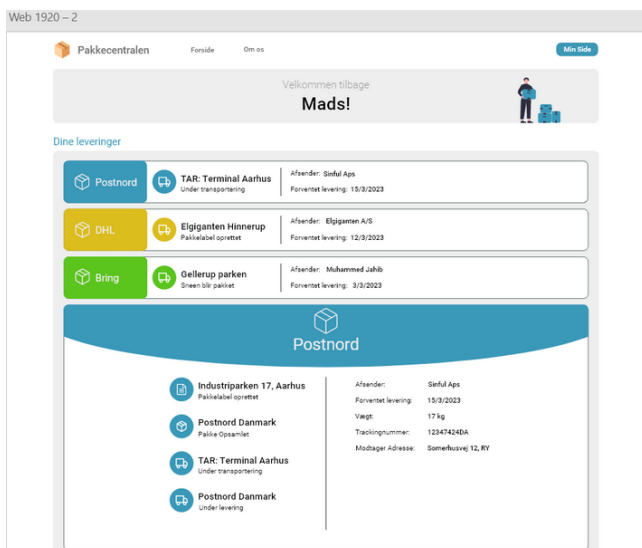
som man kan se på figur 4.3 har vi forsiden. Her er der en navbar. Hvor man kan vælge nogle muligheder. man kan gå til om os som er en side der fortæller noget om virksomheden. så kan gå til login hvor man også kan registrere sig. hvis man også trykker på start vil man blive ført til registrering. Farve valget er ikke tilfældeligt vi bruger den samme farve palette igenmen hele webapplikationen.



Figur 4.4: Wireframe af forsiden af webapplikationen



Figur 4.3: Colorpalette der er brugt.



Figur 4.5: Wireframe af hvor man kan se sin

## 5 Test af produkt

For at teste vores webapp efter færdiggørelse af 1. iteration vil vi teste om de hårde / bløde krav er opnået, dette vil blive gjort f.eks. ved at forsøge at foretage os at tracke 2 pakker fra 2 forskellige fragtfirmaer. I tilfældet af visuelle krav "Skal have et moderne stilrent design" vil dette kunne testes på en række testpersoner hvor deres holdning til blive taget og dermed vil der evalueres på dette krav.

## 6 Gennemgang af kode

### 6.1 Frontend

#### 6.1.1 Pakke visning

---

```
1      @model PackageTrackingApp.Viewmodels.MyPageVM;
2      @{
3          int iteration = 0;
4
5          foreach(var package in Model.shipments) {
6
7              iteration++;
8
9              string boksId = $"box{iteration}";
10
11              // Her ligger alt koden til at fremstille Pakke visning
12
13          }
14      }
```

---

Figur 6.1: Shipments.cshtml

Dette er koden der genere hver boks som viser hvilke pakker man har i ens email. Vi starter med at definere modellen MyPageVM som indeholder data om ens pakker, som bliver generet i HomeControlleren. Herefter laver vi en variabel der indeholder tal som skal indexe hvilke pakke den er nået til. Der er så et foreach løkke som for hvert element i Modellen shipments er en variabel "package" som indeholder information om pakken. Så bliver iteration plusset en til. der bliver så deklareret "boksId" streng som er ligmed en streng interpolation hvor "box" sættes sammen med "iteration" nummeret. under dette ligger der så en masse kode der ikke var plads til i opgaven.

```

1 <div class="bg-white rounded-lg flex flex-row" style="visibility: flex;" id="@boksId">
2 <div class="bg-primaryBlue rounded-l-lg h-auto p-4 flex flex-col md:flex-row items-center mx-a
3     <h1 class="text-sm md:text-lg text-white">@package.info.courrier</h1>
4 </div>
5 <div class="w-full p-4 rounded-r-lg flex flex-row space-x-4">
6     <div class="flex flex-col my-auto">
7         <h1 class="text-xl font-semibold">@package.currentStatus</h1>
8         @{{
9             string currntEventLocationtxt = package.events.Last().location.city + ", "
10        }}
11        <h1 class="text-sm">@currntEventLocationtxt</h1>
12    </div>
13    <div class="h-full bg-none md:bg-primaryBlue rounded-full w-1 flex-grow md:flex-initial"><
14    <div class="flex-grow hidden md:block">
15        <table class="table-auto border-separate border-spacing-x-4 text-sm text-gray-900">
16            <tr class="mx-4">
17                <td class="font-semibold">Afsender:</td>
18                <td>@package.info.consignor.name</td>
19            </tr>
20            <tr>
21                <td class="font-semibold">Service:</td>
22                <td>@package.info.service</td>
23            </tr>
24            <tr class="">
25                <td class="font-semibold">Vægt:</td>
26                @{{string weight = package.info.weight + "kg";}}
27                <td>@weight</td>
28            </tr>
29        </table>
30    </div>
31    <div class="my-auto items-center float-right" >
32        <a class="btn-primary" onclick="moreInfo(@iteration)">
33            <p>Mere info</p>
34        </a>
35    </div></div></div>

```

Figur 6.2: Shipments.cshtml

Dette kode er inde i foreach løkken. på linje 2 har vi et div html tag som definere en division af koden her som indpakker h1 tagget på linje 3 hvor teksten er sat til at være variabelen "package.info.courrier" som er inde i h1 tagget. div rundt om bruger klasserne "bg-primaryBlue", "rounded-l-lg", og "h-auto" Dette gør at baggrundsfarven er blå og har afrundet hjørner til venstre.

## 6.2 Backend

### 6.2.1 Oprettelse af Konto

---

```

1      // denne funktion sender en Email med sendgrid ved at bruge sendgrids smtp server.
2      public async Task SendEmailAsync(string toEmail, string subject, string message)
3      {
4          var apiKey = "-----"; // sendgrid api key
5          await Execute(apiKey, subject, message, toEmail);
6      }
7
8      public async Task Execute(string apiKey, string subject, string message, string toEmail)
9      {
10         _logger.LogInformation(apiKey);
11         var client = new SendGridClient(apiKey);
12         var msg = new SendGridMessage()
13         {
14             From = new EmailAddress("mads.gjellerod@gmail.com", "Password Recovery"),
15             Subject = subject,
16             PlainTextContent = message,
17             HtmlContent = message
18         };
19         msg.AddTo(new EmailAddress(toEmail));
20         // fjerner klik tracking fra koden
21         msg.SetClickTracking(false, false);
22         var response = await client.SendEmailAsync(msg);
23         // sender en besked til konsollen til at debug evtuelle fejl
24         _logger.LogInformation(response.IsSuccessStatusCode
25                                ? $"Email to {toEmail} queued successfully!"
26                                : $"Failure Email to {toEmail}");
27     }

```

---

Figur 6.3: EmailSender.cs

Dette er koden der sender en email når man eksempelvis registrere sig på siden og skal konfirmitere ens email. når man vil sende en email i programmet kalder man metoden `SendEmailAsync`. Denne metode har signature af at være `public` hvilket betyder at andre metoder kan kalde den. Så har den også signaturene "`async Task`" som gør at metoden gøre parrelet med andre opgaver så når man venter på en Api response så stopper programmet ikke. Denne metode tager tre parametere som er strege af tekst. Det første der sker i metoden er at vi laver en variabel der indeholder vores Sendgrid api key. herefter laves der et kaldt til metoden `Execute` som bliver kaldt asynkront ved hjælp af "`await`" operatøren.

I `Execute` metoden bliver der først oprettet et instans af `SendGridClient` som indeholder api nøglen. Derefter bliver der lavet et instans af Klassen `SendGridMessage` som kaldes `msg` som bruges til at opbygge. Beskenden som skal sendes. Her bliver der sat nogle variabler som hvem den er fra og beksendens indhold. den næste er at metoden `AddTo` bliver brugt til at sætte emailen som beskenden skal sendes til. Herefter bruges metoden `SetClickTracking` som der gives to false værdier til at deaktivere Click tracking. først defineres der en implicit variabel som kaldes "`response`" som vil være den type som der returneres fra `SendEmailAsync` metoden. Der bruges "`await`" operatøren som venter på at metoden fuldfører udførelsen, før koden fortsætter med at køre. `client` er et objekt af typen `SendGridClient` som har metoden `SendEmailAsync` som er en asynkron metode.