


Pakke portallen

Eksamensprojekt - Programmering A

Mads Gørup Gjellerod Christiansen

Vejledt af: Gorm Drachmann








 **Pakke Portalen**

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

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

Velkommen tilbage
Mads!

Dine leveringer

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

Abstract

Formålet med denne synopse er at give et overblik over hjemmesiden "Pakke portalen". Problemet der ønskes at løse præsenteres, hvorefter kravspecifikationer, design af siden og flowdiagrammer også præsenteres. På baggrund af denne information vil man bedre kunne forstå sidens funktionalitet og kan derfor bedre forstå kode gennemgangen, som kommer herefter. Koden gennemgås og ved en detaljeret beskrivelse så man er sikker på funktionaliteten. Alt kode og andet materiale kan findes under bilag. Hjemmesiden, alle diagrammer og design er lavet af Oliver Kempel og Mads Gjellerod, mens denne synopse er fremstillet af Mads Gjellerod.

Indhold

Abstract	1
1 Indledning	3
2 Problemformulering	4
3 Produktprincip	5
3.1 Flowchart over programmet	5
3.2 Produktkrav	5
3.3 Tech-stack	6
3.4 Database	6
3.5 Asp.net Identity	6
3.6 Wireframes	8
4 Test af produkt	9
4.1 Test af Hårde krav	9
4.2 Test af bløde krav	9
5 Gennemgang af kode	10
5.1 Frontend	10
5.2 Backend	12
6 Konklusion	16
7 Bilag	17
7.1 Video	17
7.2 Kode	17

1 Indledning

Vi har valgt i dette projekt at lave en webapplication. Hvor man kan se alle levering oplysning et sted. Så man ikke skal ledde efter links til de forskellige fragtfirmaens hjemmesidere. Dette bliver automatisk gjort når man logge ind og går ind på siden vil den hente alle ens pakker, der skal leveres fra ens mail og vise information om dem.

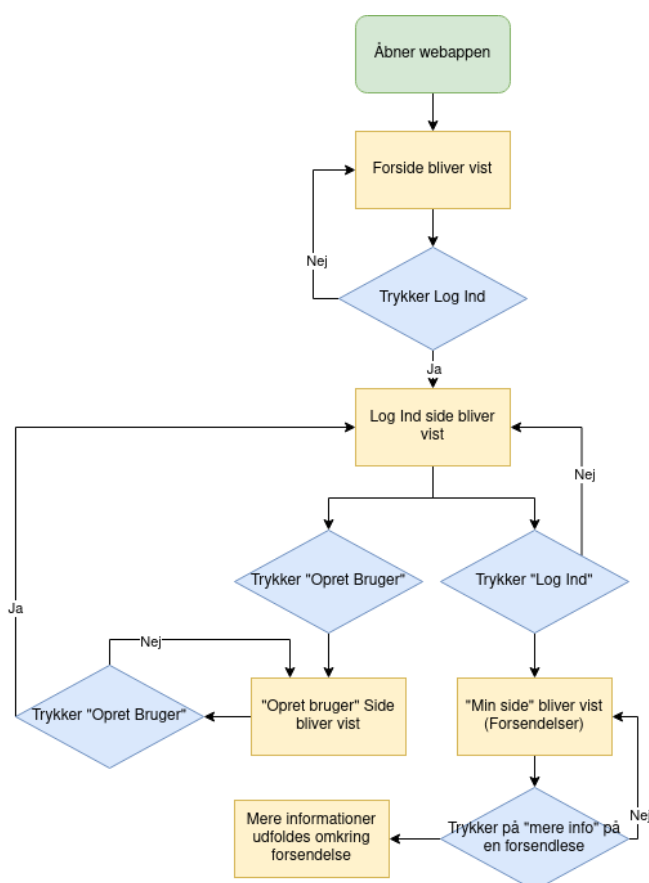
2 Problemformulering

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.

3 Produktprincip

Produktet 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.

3.1 Flowchart over programmet



Figur 3.1: General funktion af programmet.

3.2 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

3.3 Tech-stack

Webapplikationen er bygget som en Asp.Net Core 7.0 MVC webapplication, som er et webframework. Tech stacken består af Html hvor vi bruger razorpages og til styling bruges css her bruger vi Tailwind som er et css framework. TailwindCSS er et framework som gør det nemt at have et uniformt design system på en webapplication, det gør det muligt at style dine HTML tags direkte i HTML koden ved at bruge Tailwinds klasser. 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.

3.4 Database

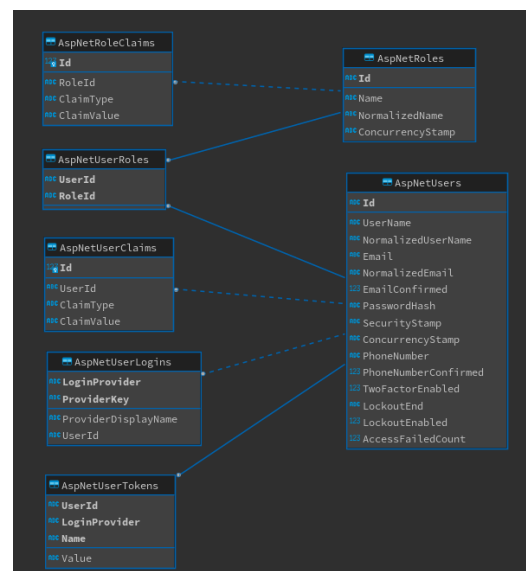
Til Databasen bruges MS SQL som er en relations database(RDBMS) udviklet af Microsoft. Hvilket giver mest mening når man laver Asp.net core mvc webapps da mange dele af dette bygger på at man bruger MS SQL.

Til at lave migrations til databasen bruges Entity Framework Core .NET Command-line tools. Hvor man 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.

```
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 funktionaliteter til ens program og kan findes som en nuget package. Denne pakke har allerede en måde databasen skal se ud på og den kan genereres med en simpel command som genere de forskellige modeller og dette kan så migreres til databasen.

der er fem tabeller AspNetRoles, AspNetUsers, AspNetRoleClaims og AspNetUserClaims. Disse tabeller er til at opbevare data relateret til brugerautorisation i vores webapplikation, Dette kan ses på figur 4.2. På denne figur kan vi se at AspNetUsers har en til en relation med tabellerne AspNetUserLogins og AspNetUserClaims. Dette er fordi at der kun kan være en bruger med det navn og login. AspNetUsers har også en til mange realtion til AspNetUserTokens og AspNetUserRoles. AspNetUserRoles har så en mange til en relation med AspNetRoles som har en til en relation med AspNetRoleClaims.



Figur 3.2: Uklip af User tablen i Databasen

3.5 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 funktionaliteter til at håndtere brugerregistrering, login, håndtering af adgangskoder og meget mere.

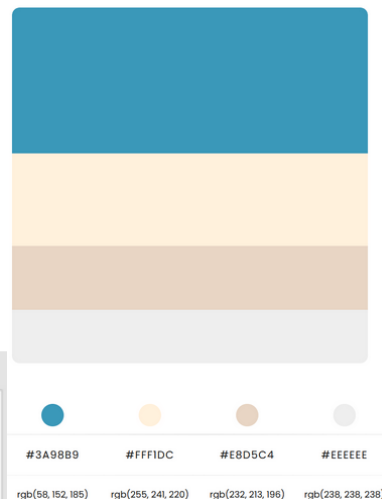
3.5.1 Google OAuth

Google OAuth bruger OAuth2 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

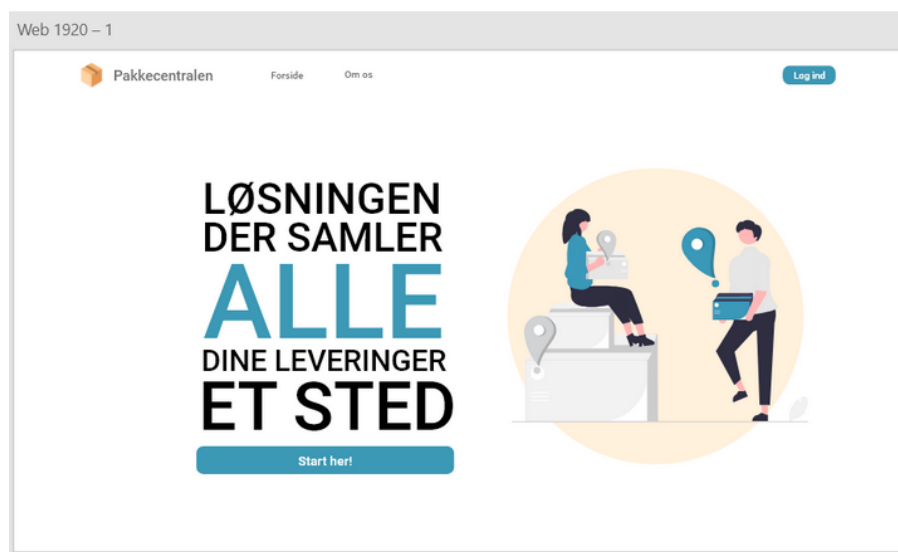
er det vi skal bruge den data vi godt vil have fat i fra brugeren er at læse brugers emails. Til at få emails fra Google bruges Googles gmail API. som gør det muligt at læse alle brugers emails når de er logget ind med OAuth.

3.6 Wireframes

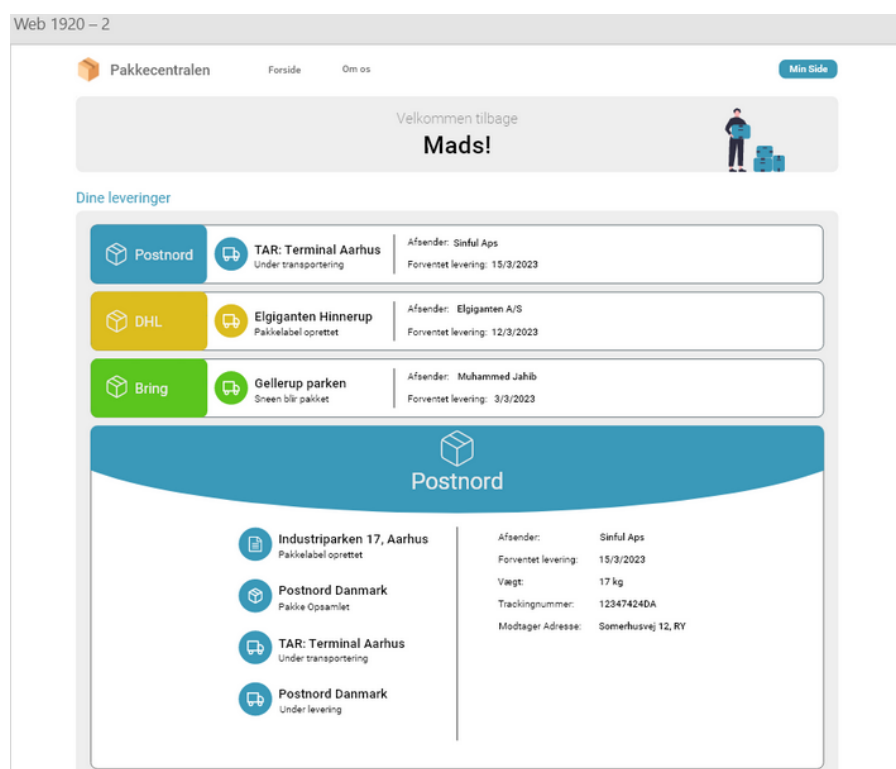
som man kan se på figur 4.3 har vi forsiden. Her er der en navbar. Hvor man kan vælge nogle muligheder. man kan gå til om os som er en side der fortæller noget om virksomheden. så kan gå til login hvor man også kan registrere sig. hvis man også trykker på start vil man blive ført til registrering. Farve valget er ikke tilfældeligt vi bruger den samme farve palette igenmen hele webapplikationen. Dette er gjort for at give et stilrent design gennem hele hjemmesiden. Det også valgt at hvert fragtfirma har hver sin farve som man kan se på figur 3.5. dette gør det let for brugeren hurtigt at se hvilket fragtfirma pakken er fra. Disse wireframe viser hvordan det endelige design helst skal se ud.



Figur 3.3: Colorpalette der er brugt.



Figur 3.4: Wireframe af førsiden af webapplikationen



Figur 3.5: Wireframe af hvor man kan se sin

4 Test af produkt

4.1 Test af Hårde krav

Skal kunne tracke pakker fra mindst 2 forskellige fragtfirmaer og Skal kunne fremvise pakke tracking fra mindst 2 forskellige fragtfirmaer det samme sted Oprette bruger

Dette er muligt og vi har tilgang til to største fragtfirmaers Api'er. Vi kan os vise pakker for disse fragtfirmaer i vores webapplikation. Så dette er også opnået.

siden skal have en lav kompleksitet (tilgå tracking på 2 klik)

Dette er opfyldt når man er logget ind er der kun et klik og man er så på siden hvor man kan se ens pakker. De bliver helt automatisk fundet i ens mail. så dette krav er også opfyldt. Med et klik mere kan man se alt information om levering af ens pakke.

4.2 Test af bløde krav

Skal have et moderne stilrent design

Kravet er testet ved at lave en undersøgelse hvor vi har sprugt klassekammerater, venner og familie om deres subjektive holdning om vores Webapplikationen. vi sprugt om de synes at webapplikationen havde et stilrent og moderne design, det svarede de fleste ja til, ud fra dette kan vi konkludere at dette krav er opnået.

Oprette bruger med Google og Skal selv kunne hente mails fra brugerens google konto

Dette krav har vi opfyldt man kan både oprette en bruger med google og med manuel registrering. Når man logge ind med google acceptere man også at vores application kan læse brugers mails. Så dette krav har vi også opnået.

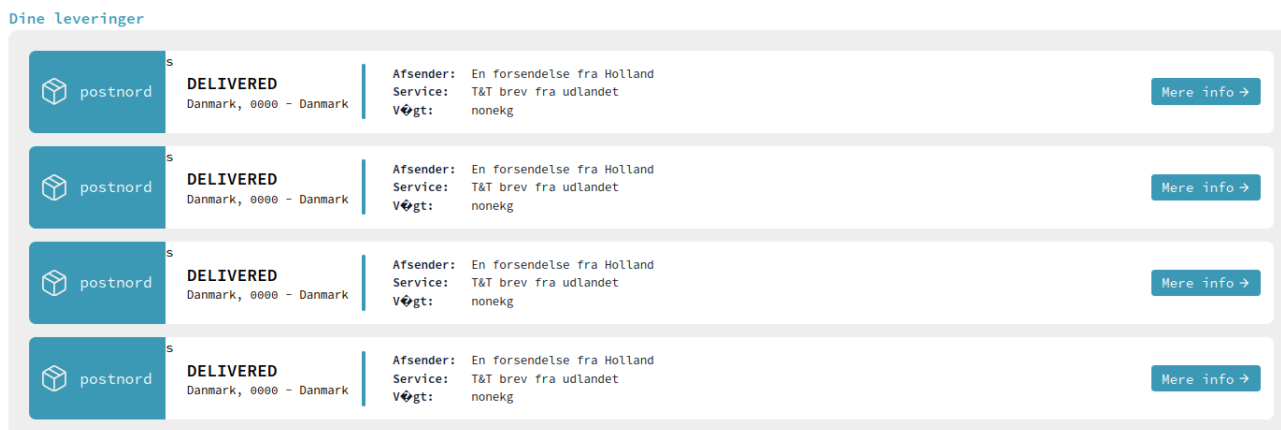
Skal fungere upåklageligt på mobile enheder (være responsivt)

Dette krav er testet ved at bruge Firefox developer tools til at sætte hjemmesiden i mobil tilstand så man kan se hvordan det ser ud på en mobil og teste med forskellige mobiler. Hjemmesiden fungere meget godt dog ikke upåklageligt. så dette krav er ikke helt opfyldt

5 Gennemgang af kode

5.1 Frontend

5.1.1 Pakke visning



Figur 5.1: 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 5.2: Shipments.cshtml

Dette er koden der genere hver boks som viser hvilke pakker man har i ens email se figur 5.1. 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            + @package.events.Last().location.zipCode + " - "
11            + @package.events.Last().location.country;
12        }
13        <h1 class="text-sm">@currntEventLocationtxt</h1>
14    </div>
15    <div class="h-full bg-none md:bg-primaryBlue rounded-full w-1 flex-grow md:flex-initial"><
16    <div class="flex-grow hidden md:block">
17        <table class="table-auto border-separate border-spacing-x-4 text-sm text-gray-900">
18            <tr class="mx-4">
19                <td class="font-semibold">Afsender:</td>
20                <td>@package.info.consignor.name</td>
21            </tr>
22            <tr>
23                <td class="font-semibold">Service:</td>
24                <td>@package.info.service</td>
25            </tr>
26            <tr class="">
27                <td class="font-semibold">Vægt:</td>
28                @{string weight = package.info.weight + "kg";}
29                <td>@weight</td>
30            </tr>
31        </table>
32    </div>
33    <div class="my-auto items-center float-right" >
34        <a class="btn-primary" onclick="moreInfo(@iteration)">
35            <p>Mere info</p>
36        </a>
37    </div></div></div>

```

Figur 5.3: Shipments.cshtml

Dette kode er inde i foreach løkken. på linje 2 har vi et div html tag som definere en division af koden. 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'en rundt om bruger klasserne "bg-primaryBlue", "rounded-l-lg", og "h-auto" Dette gør at baggrundsfarven er blå og har afrundet hjørner til venstre.

på linje 5-6 er der to nye div html tags. det her er den hvide del af hver pakke kasse. Klassen "w-full" gør at boksen fylder 100% af kontaineren. klasserne "flex" og "flex-row" laver elementet til en fleksibel kontaineren. klasserne "p-4" og "space-x-4" først giver vi elementet en padding på 4. Derefter tilføjes der en margin på 4 mellem elementerne som er mellemrummet mellem elementerne. klassen "rounded-r-lg" gør at elementet har afrundet med en stor radius. det andet div element på linje 6 bruger klasserne "flex flex-col my-auto". Dette gør Ddette element til en fleksibel container med en enkelt kolonne, der justeres centralt i en vandret linje af det primære element. "my-auto" klassen justere elementerne vertikalt så det er i midten af det primære element på linje 5. inde i det sekundære div-element bruges h1 tag til at lave en tekst med pakkes status her bruges klasserne "text-xl font-semibold" som gør at teksten er stor og fremhævet. under dette

bruges der razor-syntaks til at lave en teskt. der bliver lavet en streng hvor flere varialer bliver plusset sammen til en streng. Herefter bliver denne her variabel brugt i et h1 tag med klassen "text-sm" som gør teksten lille.

På linje 15 laves et div-element som laver en blå vandret streg med runde hjørner som deler information delene i to. Derefter er der et div-element med klasserne "flex-grow hidden md:block" hidden gør at dette element er skjult men kun på små skærme ved brug af klassen "md:block". Derefter laves et table element med klasserne "table-auto border-separate border-spacing-x-4 text-sm text-gray-900" som gør at tablen selv justere sig til dens indhold. "border-separate" og "border-spacing-x-4" gør at der er en synlig grænse mellem hver celle hvor der er en afstand på 4 pixels mellem grænserne. "text-sm" gør at teksten i felterne er i en mindre skriftstørrelse, , og "text-gray-900" klassen angiver, at teksten skal være mørkegrå. Herefter angives der en tabelrække med "tr" elementet som bruger klassen "mx-4" som laver en vandret margen. inde i dette element er der så to celler defineret med elementet "td". på linje 19 laves en celle hvor der står "Afsender" med en halv fed skrifttype. Der er så et celle mere hvor der indsættes en navn på afsenderen ved hjælp af objektet "package.info.consignor.name". der bliver så lavet to rækker mere med andre informationer.

på linje 33 er der et div element med klasserne "my-auto items-center float-right" dette gør at elementet bliver centreret i kontaineren og både horisonalt og vertikalt. "float-right" placerer element i højre side af kontaineren. inden i dette element er der et a link element med klassen "btn-primary" som gør at elementet har stilet som en knap. dette element bruger så "onclick="moreInfo(@iteration)" hvor angiver en JavaScript-funktion, der køres, når knappen klikkes på. I dette tilfælde vil funktionen "moreInfo" køres, og @iteration vil blive sendt som et argument til funktionen. Denne funktion vil folde boksen ud med mere information om pakke leveringen. inde i a elementet er der så et p element hvor der står "Mere info". Dette er vist som synlig tekst på knappen.

5.2 Backend

5.2.1 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 5.4: 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 beskendens 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.

Dette er metoden der bruges når der skal laves en ny bruger. Denne funktion bliver kaldt når brugeren har

trykker registre på hjemmesiden eller når brugeren logge ind med en ny Google konto. når denne metode bliver kaldt kører metoden asynkront i baggrunden.

```
1 public async Task<IActionResult> OnPostAsync(string returnUrl = null)
2 {
3     // her sættes hvilke url der skal bruges når registreringen er færdig
4     returnUrl ??= Url.Content("~/");
5     // her henter den en liste af andre logins
6     ExternalLogins = (await _signInManager.GetExternalAuthenticationSchemesAsync()).ToList();
7     // hvis dette er okay går den videre
8     if (ModelState.IsValid)
9     {
10         // bruger metoden RegisterModel.CreateUser til at lave en model for registreringen
11         var user = CreateUser();
12         // her gemmer vi data brugeren har sat i felterne i databasen
13         await _userStore.SetUserNameAsync(user, Input.Email, CancellationToken.None);
14         await _emailStore.SetEmailAsync(user, Input.Email, CancellationToken.None);
15
16         var result = await _userManager.CreateAsync(user, Input.Password);
```

Figur 5.5: Register.cshtml.cs

Dette første vi gør i metoden. er at sætte variablen returnUrl hvis den er null dette gøres ved at bruges "??=" som er en sammensat operatør som kun tildeler en værdi hvis værdien af variablen er null. Herefter på linje 6 bliver der initialiseret en liste af eksterne login metoder der er tilgængelige for brugerne. Til dette bruges "_signInManager" som er en feltvariabel af typen "SignInManagerIdentityUser;" som er en klasse i Identity framework. Metoden "GetExternalAuthenticationSchemesAsync" bliver så brugt til at returnere alle tilgængelige eksterne login metode. Dette bliver så lavet til en liste ved at bruge metoden "ToList".

Det der så sker er at vi tjekke om "ModelState" er gyldig dette er den ved at der ikke er sket nogle fejl i programmet. Den her "ModelState" er en egenskab i ASP.NET Core som indeholder en samling af fejl som kunne være under sket under at data er blevet konverteret til modeller. Hvis så "ModelState" er gyldig bliver der først lavet en variabel "user" hvor metoden "CreateUser" som laver en ny bruger model.

Herefter gemmes bruges navn og email. Ved at bruge "_userStore" og "_emailStore" bliver instansiliseret længere oppe i koden det her instans af "IUserStore" og "IUserEmailStore". Her bruges metoderne "SetUserNameAsync" og "SendEmailAsync" som gemmer dette i databasen. disse metoder tager tre parametere ind. først tager den en model for registrering af bruger som var den variabel vi kaldte "user". I anden parametere tager metoden "SetUserNameAsync" et navn som input. Hvor "SetEmailAsync" tager en email begge parametere her er sat til variablen "Input.Email" da ens navn som bruger er ens email den "Input" klasse indeholder data fra formdata fra registrering siden. Den sidste parametere som metoderne tager ind er "CancellationToken" den er sat til none da webapplicationen ikke gøre brug af det, men denne token kan bruges til at afbryde registrering hvis der sker en fejl eller at forbindelsen til databasen forsvinder. Herefter bruges et instans af klassen "userManager" hvor metoden "CreateAsync" bliver brugt til at lave en ny bruger i databasen. den tager "user" variablen og et kodeord som parametere.

```

1      // hvis bruger kunne laves kører koden videre
2      if (result.Succeeded)
3      {
4          // dette er til at debug problemer i koden
5          _logger.LogInformation("User created a new account with password.");
6          // her laver vi en variabel med bruges id som vi lavet før.
7          var userId = await _userManager.GetUserIdAsync(user);
8          // her generere vi en token som bruges når brugeren skal konfirmere sin email
9          var code = await _userManager.GenerateEmailConfirmationTokenAsync(user);
10         // vi kryptere så denne token med Base64 encoding.
11         code = WebEncoders.Base64UrlEncode(Encoding.UTF8.GetBytes(code));
12         // her efter laves en ny midlertidig side hvor konfirmationen af emailen ventes på
13         var callbackUrl = Url.Page(
14             "/Account/ConfirmEmail",
15             pageHandler: null,
16             values: new { area = "Identity", userId = userId, code = code, returnUrl = returnUrl,
17             protocol: Request.Scheme});
18         // her bliver emailen sendt med linket til den nye midlertidig side.
19         await _emailSender.SendEmailAsync(Input.Email, "Confirm your email",
20             $"Please confirm your account by <a href='{HtmlEncoder.Default.Encode(callbackUrl)}'>click here</a>");
21         // her tjekkes om brugeren har accepteret sin email.
22         if (_userManager.Options.SignIn.RequireConfirmedAccount)
23         {
24             // her bliver brugeren diageret til forsiden
25             return RedirectToPage("RegisterConfirmation", new { email = Input.Email, returnUrl = returnUrl });
26         }
27         else
28         {
29             // hvis ikke emailen bliver konfirmation vil den ikke virke
30             await _signInManager.SignInAsync(user, isPersistent: false);
31             // her bliver brugeren diageret til forsiden
32             return LocalRedirect(returnUrl);
33         }
34     }
35     // Hvis koden når her til er noget fejlet under registrering af bruger.
36     return Page();
37 
```

Figur 5.6: Register.cshtml.cs

Hvis så resultatet af "`_userManager.CreateAsync`" er "`Succeeded`" vil koden køre videre. Det første der sker er at der bliver sendt en besked til konsolen til at debug applicationen. Derefter laves der en variabel "`userId`" som henter et user id fra objektet "`user`" ved hjælp af metoden "`GetUserIdAsync`" som er en metode der kører asynkront. her generere vi en token som bruges når brugeren skal konfirmere sin email ved hjælp af metoden "`GenerateEmailConfirmationTokenAsync`". derefter bliver token kryptere med base64 encoding. På linje 13 bliver der så lavet en ny side der bruger stien "`Account/ConfirmEmail`". Derefter bliver emailen sendt ved hjælp af `EmailSender.cs` på figur 5.4 ved hjælp af metoden "`SendEmailAsync`" som kører asynkront. Denne metode tager tre parameter ind først er det hvilken email beskeden skal sendes til. Herefter Emnet og til sidst selve beskeden som i dette tilfælde er linket til at acceptere sin mail. på linje 22 tjekkes der så om mailen er blevet accepteret ved at kalde metoden "`RequireConfirmedAccount`" som returnere en boolean. hvis det så er sandt vil metoden returnere en "`RedirectToPage`" til "`RegisterConfirmation`". Ellers bliver brugers konto slettet og brugeren bliver sendt til forsiden.

6 Konklusion

Der er i dette løst alle krav der var stillet og det er muligt at se ens pakke. Det også muligt at logge ind med en google konto. Det er lige nu muligt at tracke pakker fra to fragtfirmaer. Der skal selvfølgelig være flere hvis dette skal være en rigtig webapplikation. Klussion er at denne webapplication har formået at løse problemet.

7 Bilag

7.1 Video

Youtube link: <https://youtu.be/l1dpl4ObZoM>

7.2 Kode

7.2.1 Github

Link til Github repo: <https://github.com/Oliverkempel/PackageTrackingApp.git>

7.2.2 Program.cs

```
1      using Microsoft.EntityFrameworkCore;
2      using Microsoft.AspNetCore.Identity;
3      using Microsoft.AspNetCore.Identity.UI.Services;
4      using Microsoft.AspNetCore.Authentication.Cookies;
5      using Microsoft.AspNetCore.Authentication;
6      using System.Threading.Tasks;
7      using System.Security.Claims;
8      using Microsoft.AspNetCore.Authentication.Google;
9      using Microsoft.AspNetCore.Authentication.OAuth;
10     using Microsoft.AspNetCore.HttpOverrides;
11     using Google.Apis.Auth.OAuth2;
12     using Google.Apis.Gmail.v1;
13     using Google.Apis.Gmail.v1.Data;
14     using Google.Apis.Services;
15     using Google.Apis;
16     using PackageTrackingApp.Services;
17     using PackageTrackingApp.Data;
18     using System.Net.Http;
19     using Google.Apis.Auth.OAuth2.Requests;
20     using NuGet.Protocol.Plugins;
21     using Microsoft.Identity.Client.Platforms.Features.DesktopOs.Kerberos;
22     using static Google.Apis.Gmail.v1.GmailService;
23
24     var builder = WebApplication.CreateBuilder(args);
25     // Add services to the container.
26     builder.Services.AddControllersWithViews();
27
28     //GmailApiReader.UserAuthorization();
29     //Console.WriteLine(GmailApiReader.ListMessages);
30     // Add Database
31     var connectionString = builder.Configuration.GetConnectionString("DefaultConnection");
32
33     builder.Services.AddDbContext<ApplicationDbContext>(options =>
```

```

34         options.UseSqlServer(connectionString));
35
36     builder.Services.AddDbContext<PackageTrackingAppIdentityDbContext>(options =>
37         options.UseSqlServer(connectionString));
38
39
40     builder.Services.AddDefaultIdentity<IdentityUser>(options => options.SignIn.RequireCon
41
42     builder.Services.AddRazorPages();
43
44     //builder.Services.AddHttpsRedirection(options => {options.HttpsPort = 7084;});
45
46     builder.Services.Configure<IdentityOptions>(options =>
47     {
48         // Password settings.
49         options.Password.RequireDigit = true;
50         options.Password.RequireLowercase = true;
51         options.Password.RequireNonAlphanumeric = true;
52         options.Password.RequireUppercase = true;
53         options.Password.RequiredLength = 6;
54         options.Password.RequiredUniqueChars = 1;
55
56         // Lockout settings.
57         options.Lockout.DefaultLockoutTimeSpan = TimeSpan.FromMinutes(5);
58         options.Lockout.MaxFailedAccessAttempts = 5;
59         options.Lockout.AllowedForNewUsers = true;
60
61         // User settings.
62         options.User.AllowedUserNameCharacters =
63         "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789-._@+";
64         options.User.RequireUniqueEmail = true;
65     });
66
67     builder.Services.AddAuthentication()
68     .AddCookie()
69     .AddGoogle(googleOptions =>
70     {
71         googleOptions.ClientId = "395816641246-60e0nalb4ruip3ptrvp1a34dg87v9q2a.apps.googl
72         googleOptions.ClientSecret = "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";
73         googleOptions.CallbackPath = new PathString("/signin-google");
74         googleOptions.Scope.Add(GmailService.Scope.GmailReadonly);
75         googleOptions.SignInScheme = IdentityConstants.ExternalScheme;
76         googleOptions.SaveTokens = true;
77         googleOptions.AccessType = "offline";
78     });

```

```

79
80
81     //.HttpContext.GetTokenAsync("access_token");
82
83     builder.Services.ConfigureApplicationCookie(options =>
84     {
85         // Cookie settings
86         options.Cookie.SameSite = SameSiteMode.Strict;
87         options.Cookie.SecurePolicy = CookieSecurePolicy.Always;
88         options.Cookie.HttpOnly = true;
89         options.ExpireTimeSpan = TimeSpan.FromMinutes(5);
90
91         options.LoginPath = "/Identity/Account/Login";
92         options.AccessDeniedPath = "/Identity/Account/AccessDenied";
93         options.SlidingExpiration = true;
94     });
95
96     builder.Services.AddTransient<IEmailSender, EmailSender>();
97     builder.Services.Configure<AuthMessageSenderOptions>(builder.Configuration);
98     builder.Services.AddHttpContextAccessor();
99     builder.Services.AddScoped<IGmailService, GmailApiReader>();
100    builder.Services.AddScoped<IMailHandler, MailHandler>();
101    builder.Services.AddScoped<ITrackingInfo, TrackingInfo>();
102
103
104
105    var app = builder.Build();
106
107    // Configure the HTTP request pipeline.
108    if (!app.Environment.IsDevelopment())
109    {
110        app.UseExceptionHandler("/Home/Error");
111        // The default HSTS value is 30 days. You may want to change this for production s
112        app.UseHsts();
113    }
114
115    app.UseHttpsRedirection();
116
117    app.UseStaticFiles();
118
119    app.UseRouting();
120
121    app.UseAuthentication();
122    app.UseAuthorization();
123

```

```

124     app.MapControllerRoute(
125         name: "default",
126         pattern: "{controller=Home}/{action=Index}/{id?}");
127
128     app.MapRazorPages();
129
130     app.Run();
131

```

7.2.3 HomeController.cs

```

1     using Microsoft.AspNetCore.Authorization;
2     using Microsoft.AspNetCore.Mvc;
3     using PackageTrackingApp.Models;
4     using PackageTrackingApp.Services;
5     using System.Diagnostics;
6     using Google.Apis.Auth.OAuth2;
7     using Google.Apis.Gmail.v1;
8     using Google.Apis.Gmail.v1.Data;
9     using Google.Apis.Services;
10    using Microsoft.AspNetCore.Authentication;
11    using Microsoft.AspNetCore.Http;
12    using Microsoft.Extensions.Configuration;
13    using System.Linq;
14    using System.Threading.Tasks;
15    using PackageTrackingApp.Services;
16    using PackageTrackingApp.Viewmodels;
17
18    namespace PackageTrackingApp.Controllers
19    {
20        public class HomeController : Controller
21        {
22            private readonly IGmailService _gmailService;
23            private readonly ILogger<HomeController> _logger;
24            private readonly IMailHandler _mailHandler;
25            private readonly ITrackingInfo _trackingInfo;
26
27            // Konstruktor som gør tilgang til services muligt
28            public HomeController(IGmailService gmailService, ILogger<HomeController> logger)
29            {
30                _gmailService = gmailService;
31                _logger = logger;
32                _mailHandler = mailHandler;
33                _trackingInfo = trackingInfo;
34            }

```

```

35
36 //Returnere forsiden (Index siden)
37 public IActionResult Index()
38 {
39     return View();
40 }
41
42 //Returnere Privacy siden
43 public IActionResult Privacy()
44 {
45     return View();
46 }
47
48 //returnere viewet myPage hvis bruger er authorized (logget ind)
49 [HttpGet]
50 [Authorize]
51 [Route("token")]
52 public async Task<IActionResult> myPage()
53 {
54     //opretter en instans af klassen mailInfos og venter svar fra mailHandler
55     AllMailInfo mailInfos = await _mailHandler.getAllTrackingNumbers();
56
57     //Initializere en liste af Shipments og tildeler den det der returneres fra
58     List<Shipment> allShipmentsFromUserInbox = _trackingInfo.getTrackingInfoAll
59
60     //En ny viewmodel initialiseres, denne viewmodel er forventet af myPage vi
61     MyPageVM vm = new MyPageVM();
62
63     //Shipments i viewmodel bliver derefter sat til de hentede shipment inform
64     vm.shipments = allShipmentsFromUserInbox;
65
66     //Der returneres til myPage viewet med viewdataen sendt med.
67     return View(vm);
68 }
69
70
71 [ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore =
72 public IActionResult Error()
73 {
74     return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext
75 }
76 }
77 }
78

```

7.2.4 TrackingInfo.cs

```
1      using PackageTrackingApp.Models;
2      using Newtonsoft.Json;
3      using RestSharp;
4      using PackageTrackingApp.Models.ApiReturnModels;
5      using PackageTrackingApp.Models.ShipmentSubModels;
6
7      namespace PackageTrackingApp.Services
8      {
9          //interface til alle metoderne i klassen.
10         public interface ITrackingInfo
11         {
12             List<Shipment> getTrackingInfoAllCourriers(AllMailInfo allMailInfo);
13             PostnordReturnWrapper.PostNordReturn getPostnordTrackingInfo(string trackingNu
14             GlsReturnContainer.GlsReturn getGlsTrackingInfo(string trackingNumber);
15         }
16
17         public class TrackingInfo : ITrackingInfo
18         {
19             public List<Shipment> getTrackingInfoAllCourriers(AllMailInfo allMailInfo)
20             {
21                 //initialisere liste af Shipments
22                 List<Shipment> shipmentsList = new List<Shipment>();
23
24                 //looper igennem alle postnordMailInfos i allMailinfos fra parametrene i
25                 foreach(var postnordMailInfo in allMailInfo.postNordMailInfos)
26                 {
27                     //initialisere ny postnordreturn til opbevaring af datga fra api kald.
28                     PostnordReturnWrapper.PostNordReturn shipmentData = new PostnordReturn
29
30                     // henter postnord tracking info via metoden med den nuværende iterati
31                     shipmentData = getPostnordTrackingInfo(postnordMailInfo.trackingNumber
32
33                     // tjekker om lengden ad shipments er større end eller lig en
34                     if(shipmentData.TrackingInformationResponse.Shipments.Length >= 1)
35                     {
36                         // initialisere ny liste af SmipmentEvents
37                         List<ShipmentEvent> Events = new List<ShipmentEvent>();
38
39                         // Looper igennem events i sporingsinformationerne
40                         foreach (var ev in shipmentData.TrackingInformationResponse.Shipme
41                         {
42                             //tilføjer nyt shipmentevent til eventlisten
43                             Events.Add(new ShipmentEvent
```

```

44         {
45             //tilføjer informationen omkring eventet
46             dateTime = ev.EventTime.UtcDateTime,
47             description = ev.EventDescription,
48             location = new Address
49             {
50                 city = ev.Location.City,
51                 country = ev.Location.Country,
52                 zipCode = ev.Location.Postcode,
53             },
54             status = ev.Status,
55         });
56     }
57
58     //tjekker om vægten af smipmenttet er null
59     if (shipmentData.TrackingInformationResponse.Shipments.First().TotalWeight == null)
60     {
61         //tilføjer shipment til listen
62         shipmentsList.Add(new Shipment
63         {
64             //tilføjer informationen til shipment
65             currentStatus = shipmentData.TrackingInformationResponse.ShipmentStatus,
66             //events bliver tilføjet til shipment
67             events = Events,
68             //informationer vedrørende shipment bliver tilføjet
69             info = new ShipmentInfo
70             {
71                 //weight sættes til strengen none da der ingen vægt er
72                 weight = "none",
73                 //tracking nummer sættes
74                 trackingNumber = shipmentData.TrackingInformationResponse.TrackingNumber,
75                 //fragtfirma sættes
76                 courier = postnordMailInfo.Courier,
77                 //servicen sættes
78                 service = shipmentData.TrackingInformationResponse.ShipmentService,
79                 //Afsender sættes
80                 consignor = new Person
81                 {
82                     name = shipmentData.TrackingInformationResponse.ShipmentConsignorName,
83                     address = new Address
84                     {
85                         city = shipmentData.TrackingInformationResponse.ShipmentConsignorCity,
86                         country = shipmentData.TrackingInformationResponse.ShipmentConsignorCountry,
87                         zipCode = shipmentData.TrackingInformationResponse.ShipmentConsignorZipCode,

```



```

89         },
90     },
91     //modtager sættes
92     consignee = new Person
93     {
94         name = "No name (You)",
95         address = new Address
96         {
97             city = shipmentData.TrackingInformationResponse.City,
98             country = shipmentData.TrackingInformationResponse.Country,
99             zipCode = shipmentData.TrackingInformationResponse.ZipCode,
100         },
101     },
102 }
103
104 }
105
106 });
107 }
108 //Hvis vægt ikke er null
109 else
110 {
111     //tilføjer shipment til listen
112     shipmentsList.Add(new Shipment
113     {
114         //tilføjer informationen til shipment
115         currentStatus = shipmentData.TrackingInformationResponse.CurrentStatus,
116         //events bliver tilføjet til shipment
117         events = Events,
118         //informationer vedrørende shipment bliver tilføjet
119         info = new ShipmentInfo
120         {
121             //weight sættes til strengen none da der ingen vægt er
122             weight = shipmentData.TrackingInformationResponse.Weight,
123             //tracking nummer sættes
124             trackingNumber = shipmentData.TrackingInformationResponse.TrackingNumber,
125             //fragtfirma sættes
126             courier = postnordMailInfo.Courier,
127             //service sættes
128             service = shipmentData.TrackingInformationResponse.Service,
129             //Afsender sættes
130             consignor = new Person
131             {
132                 name = shipmentData.TrackingInformationResponse.ConsignorName,
133                 address = new Address

```

```

134         {
135             city = shipmentData.TrackingInformationResponse.city;
136             country = shipmentData.TrackingInformationResponse.country;
137             zipCode = shipmentData.TrackingInformationResponse.zipCode;
138
139         },
140     },
141     //Modtager sættes
142     consignee = new Person
143     {
144         name = "No name (You)",
145         address = new Address
146         {
147             city = shipmentData.TrackingInformationResponse.city;
148             country = shipmentData.TrackingInformationResponse.country;
149             zipCode = shipmentData.TrackingInformationResponse.zipCode;
150
151         },
152     }
153
154     }
155
156     });
157 }
158 }
159
160
161 }
162
163 //løpper igennem alle glsMailInfo i allMailInfos.GlsMailInfo
164 foreach(var glsMailInfo in allMailInfo.glsMailInfos)
165 {
166     //opretter instans af Glsreturn til at indeholde data sendt fra api
167     GlsReturnContainer.GlsReturn shipmentData = new GlsReturnContainer.GlsReturn();
168
169     // henter tracking information på nuværende iteration af loopet
170     shipmentData = getGlsTrackingInfo(glsMailInfo.trackingNumber);
171
172     //tjekker at status af shipment returneret af api ikke er "ERRORNOTFOUN
173     if (shipmentData.TuStatus.FirstOrDefault().TuNo != "ERRORNOTFOUN
174     {
175         //Intitalizere instans af ShipmentEvent
176         List<ShipmentEvent> Events = new List<ShipmentEvent>();
177
178         //løper igennem events i shipmentData

```

```

179         foreach (var ev in shipmentData.TuStatus.First().History)
180         {
181             //tilføjer en shipmentevent til events listen
182             Events.Add(new ShipmentEvent
183             {
184                 //data tildeles
185                 dateTime = ev.Date.UtcDateTime,
186                 description = ev.EvtDscr,
187                 location = new Address
188                 {
189                     city = ev.Address.City,
190                     country = ev.Address.CountryName,
191                     //de informationer som ikke findes i deres tracking in
192                     zipCode = "zip Not provided",
193                 },
194                 status = "status Not Provided",
195             });
196         }
197
198         //tilføjer ny instans af Shipment til shipmentsList listen
199         shipmentsList.Add(new Shipment
200         {
201             //data tildeles
202             currentStatus = shipmentData.TuStatus.First().ProgressBar.Stat
203             events = Events,
204             info = new ShipmentInfo
205             {
206                 weight = shipmentData.TuStatus.First().Infos.Where(x => x.
207                 trackingNumber = shipmentData.TuStatus.First().TuNo,
208                 courier = glsMailInfo.Courier,
209                 service = shipmentData.TuStatus.First().Infos.Where(x => x
210                 consignor = new Person
211                 {
212                     // data som ikke findes i fratfirmaets sporingdata til
213                     name = "Name Not provided",
214                     address = new Address
215                     {
216                         city = "City not provided",
217                         country = "Country not provided",
218                         zipCode = "Zipcode not provided",
219
220                     },
221                 },
222                 consignee = new Person
223                 {

```

```

224         name = "No name (You)",
225         address = new Address
226         {
227             city = "City not provided",
228             country = "Country not provided",
229             zipCode = "Zipcode not provided",
230
231         },
232     }
233
234     }
235
236     });
237 }
238
239 }
240
241     //returnere shipmentsList
242     return shipmentsList;
243 }
244
245
246 public PostnordReturnWrapper.PostNordReturn getPostnordTrackingInfo(string tra
247 {
248     //opretter options til https kald, med url som parameter
249     var options = new RestClientOptions("https://api2.postnord.com")
250     {
251         //sætter timeout til -1, så der ikke er nogen timeout
252         MaxTimeout = -1,
253     };
254     //opretter ny restclient med options som parameter
255     var client = new RestClient(options);
256     // initialisere ny instans af restrequest og giver endpoint og get metode
257     var request = new RestRequest("/rest/shipment/v5/trackandtrace/findByIdent
258
259     // tilføjer parametre til https kald, heriblandt apikey, tracking id og l
260     request.AddQueryParameter("apikey", "aa5bab080e542f8f20d09e27a48321e0");
261     request.AddQueryParameter("id", trackingNumber);
262     request.AddQueryParameter("locale", "da");
263
264     // initialisere ny instans af response og køre kaldet med client.Get(reque
265     RestResponse response = client.Get(request);
266
267     //initialisere ny instans af PostNordReturn til opbevaring af data fra api
268     PostnordReturnWrapper.PostNordReturn data = new PostnordReturnWrapper.Post

```

```

269         //konvertere json svar fra api til PostNordReturn type og gemmer i data va
270         data = JsonConvert.DeserializeObject<PostnordReturnWrapper.PostNordReturn>
271
272         //returnere data
273         return data;
274     }
275
276     public GlsReturnContainer.GlsReturn getGlsTrackingInfo(string trackingNumber)
277     {
278         //opretter options til https kald, med url som parameter
279         var options = new RestClientOptions("https://gls-group.com")
280         {
281             //sætter timeout til -1, så der ikke er nogen timeout
282             MaxTimeout = -1,
283         };
284
285         //opretter ny restclient med options som parameter
286         var client = new RestClient(options);
287         // initialisere ny instans af restrequest og giver endpoint og get metode
288         var request = new RestRequest("/app/service/open/rest/DK/da/rstt001", Metho
289
290         // tilføjer trackingnumber parameter til kald
291         request.AddQueryParameter("match", trackingNumber);
292
293         // initialisere ny instans af response og køre kaldet med client.Get(reque
294         RestResponse response = client.Get(request);
295
296         //initialisere ny instans af GlsReturn til opbevaring af data fra api kald
297         GlsReturnContainer.GlsReturn data = new GlsReturnContainer.GlsReturn();
298
299         //konvertere json svar fra api til GlsReturn type og gemmer i data variabel
300         data = JsonConvert.DeserializeObject<GlsReturnContainer.GlsReturn>(respons
301
302         //tjekker om http kaldet returnere med statuskode 404 not found
303         if(response.StatusCode == System.Net.HttpStatusCode.NotFound)
304         {
305             //opretter ny instans af TuStatus arrayet
306             GlsReturnContainer.TuStatus[] notFoundCatch = new GlsReturnContainer.T
307             //giver plads 0 i arrayet ny instans af TuStaus
308             notFoundCatch[0] = new GlsReturnContainer.TuStatus
309             {
310                 // sætter TuNo lig med "ERRONOTFOUND404"
311                 TuNo = "ERRORNOTFOUND404"
312             };
313             //returnere GlsReturn med tustaus lig notFoundCatch

```

```

314         return new GlsReturnContainer.GlsReturn
315         {
316             TuStatus = notFoundCatch,
317         };
318     } else
319     {
320         //Returnere data
321         return data;
322     }
323 }
324
325 }
326 }

```

7.2.5 EmailSender.cs

```

1     using Microsoft.AspNetCore.Identity.UI.Services;
2 using Microsoft.Extensions.Options;
3 using System;
4 using System.Threading.Tasks;
5 using SendGrid;
6 using SendGrid.Helpers.Mail;
7
8 namespace PackageTrackingApp.Services;
9
10 public class EmailSender : IEmailSender
11 {
12     private readonly ILogger _logger;
13
14     public EmailSender(IOptions<AuthMessageSenderOptions> optionsAccessor,
15                     ILogger<EmailSender> logger)
16     {
17         Options = optionsAccessor.Value;
18         _logger = logger;
19     }
20
21     public AuthMessageSenderOptions Options { get; } //Set with Secret Manager.
22
23     public async Task SendEmailAsync(string toEmail, string subject, string message)
24     {
25         var apiKey = "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";
26         if (string.IsNullOrEmpty(apiKey))
27         {
28             throw new Exception("Null SendGridKey");
29         }

```

```

30         await Execute(apiKey, subject, message, toEmail);
31     }
32
33     public async Task Execute(string apiKey, string subject, string message, string toEmail)
34     {
35         _logger.LogInformation(apiKey);
36         var client = new SendGridClient(apiKey);
37         var msg = new SendGridMessage()
38         {
39             From = new EmailAddress("mads.gjellerod@gmail.com", "Password Recovery"),
40             Subject = subject,
41             PlainTextContent = message,
42             HtmlContent = message
43         };
44         msg.AddTo(new EmailAddress(toEmail));
45
46         // Disable click tracking.
47         // See https://sendgrid.com/docs/User_Guide/Settings/tracking.html
48         msg.SetClickTracking(false, false);
49         var response = await client.SendEmailAsync(msg);
50         // sends response to debug console
51         _logger.LogInformation(response.IsSuccessStatusCode
52                                ? $"Email to {toEmail} queued successfully!"
53                                : $"Failure Email to {toEmail}");
54     }
55 }

```

7.2.6 myPage.cs

```

1         @model PackageTrackingApp.Viewmodels.MyPageVM;
2     @{
3         ViewData["Title"] = "Test page";
4     }
5
6     <section class="bg-primaryGray p-6 rounded-xl text-center">
7         <h1 class="text-lg text-gray-500 font-thin">Velkommen tilbage</h1>
8         <h1 class="text-2xl text-gray-900 font-semibold">@User.Identity.Name</h1>
9     </section>
10
11     <section class="mt-6">
12         <h1 class="text-lg text-primaryBlue font-semibold text-center md:text-left">Dine leveringer</h1>
13         <div class="bg-primaryGray rounded-xl flex flex-col p-6 space-y-4">
14
15             @{
16                 int iteration = 0;

```

```

17
18     foreach(var package in Model.shipments) {
19
20         iteration++;
21
22         string boksId = $"box{iteration}";
23
24         <div class="bg-white rounded-lg flex flex-row" style="visibility: flex;" id="@boksId">
25             <div class="bg-primaryBlue rounded-l-lg h-auto p-4 flex flex-col md:flex-row items
26                 <svg xmlns="http://www.w3.org/2000/svg" class="h-8 pr-4 mx-auto" viewBox="0 0
27                     <g id="Icon_feather-package" data-name="Icon feather-package" transform="t
28                         <path id="Path_87" data-name="Path 87" d="M31.568,18.032,11.25,6.315"
29                         <path id="Path_88" data-name="Path 88" d="M45.136,34.6V16.544a4.515,4.
30                         <path id="Path_89" data-name="Path 89" d="M4.905,10.44119.708,11.4,19.
31                         <path id="Path_90" data-name="Path 90" d="M18,40.756V18" transform="tr
32                     </g>
33                 </svg>
34                 <h1 class="text-sm md:text-lg text-white">@package.info.courrier</h1>
35             </div>
36             <div class="w-full p-4 rounded-r-lg flex flex-row space-x-4">
37                 <div class="flex flex-col my-auto">
38                     <h1 class="text-xl font-semibold">@package.currentStatus</h1>
39                     @{
40                         string currntEventLocationtxt = package.events.Last().location
41                     }
42                     <h1 class="text-sm">@currntEventLocationtxt</h1>
43                 </div>
44                 <div class="h-full bg-none md:bg-primaryBlue rounded-full w-1 flex-grow md:fle
45                 <div class="flex-grow hidden md:block">
46                     <table class="table-auto border-separate border-spacing-x-4 text-sm text-g
47                         <tr class="mx-4">
48                             <td class="font-semibold">Afsender:</td>
49                             <td>@package.info.consignor.name</td>
50                         </tr>
51                         <tr>
52                             <td class="font-semibold">Service:</td>
53                             <td>@package.info.service</td>
54                         </tr>
55                         <tr class="">
56                             <td class="font-semibold">V<|gt:</td>
57                             @{
58                                 string weight = package.info.weight + "kg";
59                             }
60                             <td>@weight</td>
61                         </tr>

```



```

62         </table>
63     </div>
64     <div class="my-auto items-center float-right" >
65         <a class="btn-primary" onclick="moreInfo(@iteration)">
66             @* onclick="moreInfo(@package.info.trackingNumber)*@
67             <p>Mere info</p>
68             <svg fill="none" stroke="currentColor" stroke-linecap="round" stroke-l
69                 <path d="M5 12h14M12 5l7 7 7"></path>
70             </svg>
71         </a>
72     </div>
73 </div>
74 </div>
75
76     string bigboksId = $"boxbig{iteration}";
77
78     <div class="bg-white rounded-lg" style="display: none;" id="@bigboksId">
79         <div class=" bg-primaryBlue rounded-t-lg p-4 flex flex-row">
80             <div class="text-center flex-1 items-center">
81                 <h1 class="text-2xl text-white text-center">Postnord</h1>
82             </div>
83
84         </div>
85
86         <div class="grid grid-cols-1 lg:grid-cols-2">
87             <div class="border-none lg:border-2 lg:border-r-primaryBlue flex flex-col p-6
88
89                 @{{
90                     foreach(var evt in package.events) {
91
92                         <div class="flex flex-row items-center">
93                             <div class="h-12 w-12 rounded-full bg-primaryBlue mr-4"></div>
94                             <div class="flex-1">
95
96                                 @{{
97                                     string eventLocationtxt = evt.location.city +
98                                     }
99                                     <h1 class="text-lg font-semibold">@eventLocationtxt
100                                     <p class="text-base font-light">@evt.description</p>
101                                 </div>
102                                     <p class="">@evt.dateTime</p>
103                             </div>
104                         }
105                     }
106

```

```

107
108     </div>
109     <div class="border-none lg:border-2 lg:border-l-primaryBlue p-6">
110         <table class="table-auto border-separate border-spacing-x-5 border-spacing
111             <tr class="mx-4">
112                 <td class="font-semibold">Afsender:</td>
113                 <td>@package.info.consignor.name</td>
114             </tr>
115             <tr class="mx-4">
116                 <td class="font-semibold">Afsender Adresse:</td>
117                 @{
118                     string consingorAdressTxt = package.info.consignor.add
119                 }
120                 <td>@consingorAdressTxt</td>
121             </tr>
122             <tr>
123                 <td class="font-semibold">Service:</td>
124                 <td>@package.info.service</td>
125             </tr>
126             <tr class="">
127                 <td class="font-semibold">V<del>gt</del>:</td>
128                 @{
129                     string packageWeightTxt = package.info.weight + " kg";
130                 }
131                 <td>@package.info.weight + " kg"</td>
132             </tr>
133         </table>
134     </div>
135 </div>
136 <div class=" w-full">
137     <a class="btn-secondary float-right m-4" onclick="lessInfo(@iteration);">
138         <p>Vis Mindre</p>
139     </a>
140 </div>
141
142 </div>
143 }
144 }
145
146 @section scripts {
147
148     <script type="text/javascript">
149         function moreInfo(elementIndex) {
150             var element = document.getElementById("boxbig" + elementIndex);
151             var element1 = document.getElementById("box" + elementIndex);

```

```

152         if(elementIndex == 1) {
153             element.style.marginTop = 0;
154         }
155         element1.style.display = "none";
156         element.style.display = "block";
157
158     }
159     function lessInfo(elementIndex) {
160         var element = document.getElementById("boxbig" + elementIndex);
161         var element1 = document.getElementById("box" + elementIndex);
162         element1.style.display = "flex";
163         element.style.display = "none";
164     }
165     </script>
166
167 }
168

```

7.2.7 GmailApiReader.cs

```

1     using Google.Apis.Auth.OAuth2;
2     using Google.Apis.Gmail.v1;
3     using Google.Apis.Gmail.v1.Data;
4     using Google.Apis.Services;
5     using Microsoft.AspNetCore.Authentication;
6
7     namespace PackageTrackingApp.Services;
8
9     public interface IGmailService
10    {
11        Task<List<Message>> GetAllMails(string fromEmail);
12    }
13
14    public class GmailApiReader : IGmailService
15    {
16
17        private readonly IHttpContextAccessor _httpContextAccessor;
18
19        public GmailApiReader(IHttpContextAccessor httpContextAccessor)
20        {
21            _httpContextAccessor = httpContextAccessor;
22        }
23        public async Task<List<Message>> GetAllMails(string fromEmail)
24        {
25            // henter Auth properties indeholdende accesstoken asynkront og gemmer det i authPro

```

```

26     var authProps = await _httpContextAccessor.HttpContext.AuthenticateAsync();
27
28     // gemmer accessToken i variabel fra authprops
29     var accessToken = authProps.Properties.GetTokenValue("access_token");
30
31     //omdanner accesstoken til google credential
32     var credential = GoogleCredential.FromAccessToken(accessToken);
33
34     //opretter ny instans af Gmailservice
35     var service = new GmailService(new BaseClientService.Initializer()
36     {
37         HttpClientInitializer = credential,
38         ApplicationName = "Gmail API Sample"
39     });
40
41     //opretter en request til at hente mails fra brugeren
42     var emailListRequest = service.Users.Messages.List("me");
43     //tilfjer et search query som sger efter mails fra mail "fromEmail" som indsttes fra f
44     emailListRequest.Q = $"from:{fromEmail}";
45     //email requestet sendes og resultatet gemmes i emailListRespinse variabelen
46     var emailListResponse = await emailListRequest.ExecuteAsync();
47     //kontrolstruktur der tjekker om svaret er null eller om der er nogle beskeder i email
48     if (emailListResponse?.Messages != null && emailListResponse.Messages.Any())
49     {
50         //hvis der er instanseres en ny liste af Message
51         List<Message> messagesWithData = new List<Message>();
52
53         //lkke der gennemgr hver message i emailListResponse
54         foreach (var message in emailListResponse.Messages) {
55
56             // opbygger request til at hente selve beskeden med idet der tilhrende nuvrend
57             var emailInfoRequest = service.Users.Messages.Get("me", message.Id);
58             //requestet sendes og svaret gemmes i variabelen emailInfoResponse
59             var emailInfoResponse = await emailInfoRequest.ExecuteAsync();
60             // Dette tilfjes derefter til listen af messages
61             messagesWithData.Add(emailInfoResponse);
62         }
63
64         //efter alle messages er tilfjet til listen messagesWithData returneres denne
65         return messagesWithData;
66     }
67     else
68     {
69         //hvis der ingen messages er i resultatet returneres null
70         return null;

```

71 }
72 }
73 }
