Dokumentation for Hotel Vesterly projekt

[Aspire 1](#_Toc966136974)

[Database 2](#_Toc201010125)

[API 3](#_Toc57539515)

[Endpoints 4](#_Toc793543252)

[Rooms 4](#_Toc1639133434)

[Reservations 4](#_Toc1641579829)

[Users 4](#_Toc806317226)

[Tickets 5](#_Toc1030115152)

[Front end 5](#_Toc1633422014)

[Pages 6](#_Toc1298653081)

[Reservations 6](#_Toc720722196)

# Aspire

Dette projekt bruger .Net Aspire til diverse dele af løsningen. Dette gør at vi undgår at skulle bruge en database som bliver hostet separeret fra vores projekt.

Vi blandt andet brugt [CSharpFritz’s tutorials](https://www.youtube.com/watch?v=jwDC_UQ8H70) og diverse [aspire dokumentation sider](https://learn.microsoft.com/en-us/dotnet/aspire/) for at få vores DB opsat med Entity Framework Core.

Der er brugt referencer via aspire så vi undgår at skulle bruge alt muligt i forhold til gemte connection strings.

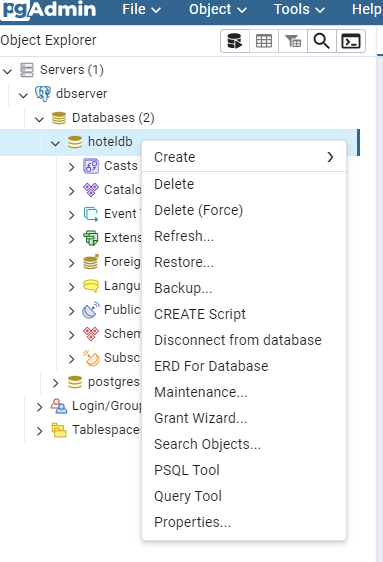
Vi har oplevet enkelte problemer med vores migreringer og at boote up DB gennem Aspire men løsningen var bare at slette HotelDB databasen på vores postgres instans bootet op igennem Aspire.

(Dette kan blive fikset gennem bedre forståelse af EFCore og hvordan den fungerer med Aspire men problemet er vitterligt bare slet og prøv igen hvis det sker.)

# Database

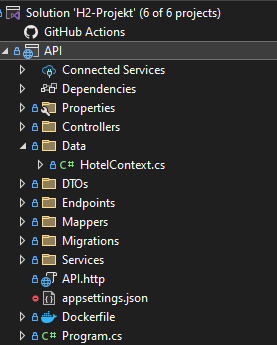
Vi kører Postgresql med en HotelDB database og har løst problemet med connection strings ved hjælp af Aspire.

Skulle man opleve problemer med databasen så skal man bare slette hoteldb som set her

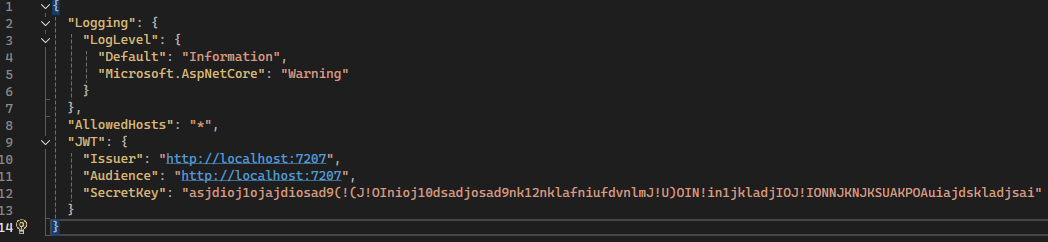


# API

Vi har opsat login og authorization på vores api. Til det har vi også en appsettings opsat. Her er et billede som viser strukturen af vores appsettings.json fil.



Den indeholder dette tekst:

Issuer og audience er egentligt bare ip addressen på api’en som man ser i aspire dashboardet. Din secret key skal bare være et langt unikt string af en eller anden form.

## Endpoints

* Vi har ikke de mest brugervenlige endpoints, men de skal selvfølgelig bruges i sammenhæng med vores Domain Models. De kan ses på swagger.

### Rooms

* Der er blevet lavet en controller for at kunne hente, oprette, ændre og slette data indenfor rooms. Det er lavet i sammenhæng med vores Domain Models, i dette tilfælde ville det være room.

### Reservations

* Der er blevet lavet en controller for at kunne hente, oprette, ændre og slette data indenfor reservations. Det er lavet i sammenhæng med vores Domain Models, i dette tilfælde ville det være reservation.

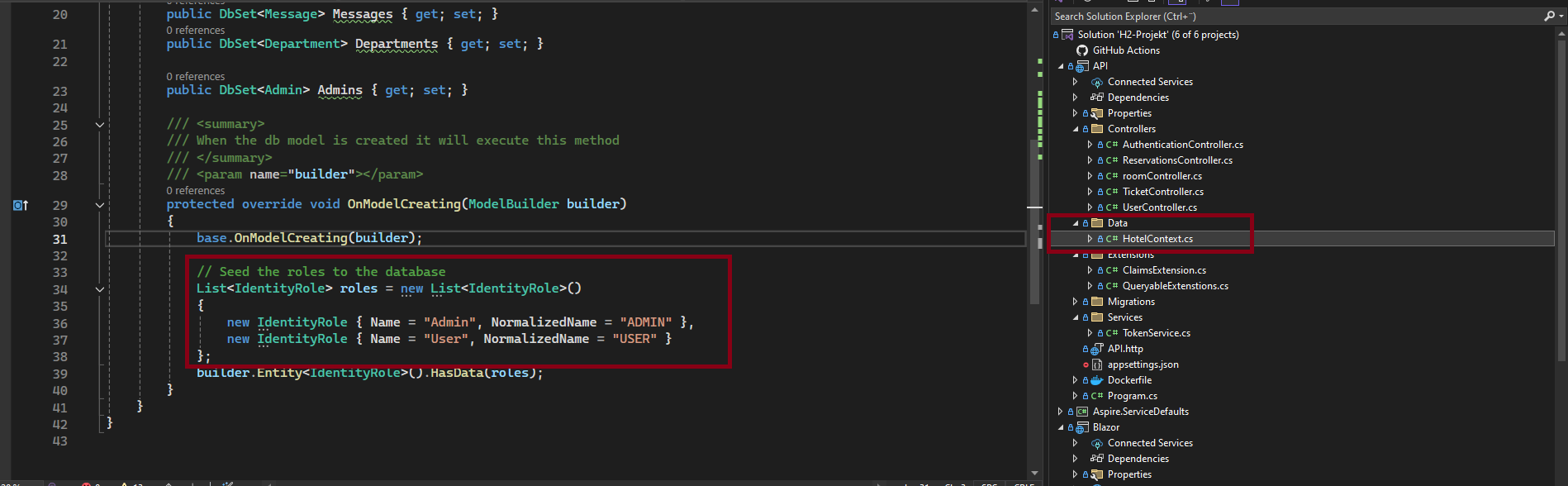
### Users

* Vi har 7 endpoints i User controlleren. 5 til almene crud operations på user objektet ned mod databasen. Ud over det er der også to endpoints til specifikt medarbejdere(employees) som er et endpoint til at lave en medarbejder og så en til at opdatere medarbejderen. I endpoints’ne har jeg gjort brug af til håndtering af at lave en bruger/medarbejder, da vi ikke synes der er en grund til at udstille id og userid i responset. Til de DTO’er, har jeg også lavet en mapper, hvori DTO for user og employee bliver mappet til User og Employee objektet i Domain models mappen.

### Tickets

* Der er blevet sat en simpel controller op som kan hente, oprette, ændre og slette data som hører til reservations. Det er sat op sammen men vores domain models, hvor det vil være ticket der bliver brugt.

Authentication

* I forhold til authentication og authorization, generere vi en JWT (Json Web Token) som indeholder det der hedder claims på baggrund af brugeren. Og det gør vi inde i vores TokenService.cs fil, som ligger inde i Services mappen, i API projektet.
* Lidt mere teknisk, så er er det en ClaimsPrincipal som indeholder det der hedder en ClaimsIdentity, og den her ClaimsIdentity kan så indeholde de her claims, altså oplysninger på brugeren.
* Du kan se et ClaimsPrincipal lidt lige som et pas, ClaimsIdentity er så hvem det her pas tilhøre, og Claims i ClaimsIdentity vil så være oplysninger på personen i deres pas.
* Det er så i vores AuthenticationController.cs (inde i Controllers mappen I API projektet) at vi gør brug af TokenService.cs i forhold til at når bruger registrere sig, og når de vil logge ind.
* Ud over det har vi gjort brug af Authorize attributen i flere af vores controllere for at kunne granulere ned hvem der har adgang til at kunne kalde dem, ved nogen af dem skal man som det mindste være logget, andre skal man være admin for at kunne gøre det. (eg. [Authorize(Roles = “Admin”)]
* Hvis man har lyst til at tilføje flere roller, det kunne være rengøring, reception osv. Så kan man gøre det i HotelContext.cs på linje 34. Du kan finde HotelContext.cs i Data mappen i API projektet.
* 

# Front end

Der er blevet sat en front end op, med noget hjælp fra et framework kaldet MudBlazor, hvilket er en simpel side til at kunne finde nemt front end kode mv. Hvor der er blevet brugt nogle templates, samt nogle andre komponenter som kan findes, ved at gå igennem de components, der kan findes på dette link: <https://mudblazor.com/components/alert#api>

MudBlazor frameworket er installeret ved at bruge den manual install guide, som kan findes på MudBlazors hjemmeside gennem linket her: <https://mudblazor.com/getting-started/installation#using-templates>

## Pages

Der er blevet sat flere forskellige pages op, som arbejder sammen med nogle af vores endpoints.

### Reservations

Der er blevet sammenkoblet en funktion, fra reservations end point til front end. Så at en Admin, har mulighed for at Delete en reservation. Dette vil også sige at det er nødvendigt at have logget ind som Admin, for at kunne bruge delete funktionen.

### Authentication og Authorization

I Blazor håndtere vi login ved