|  |  |
| --- | --- |
| **Øvelse** | RosBil Razor Pages app |
| **Projekt** | RosBilRP |
| **Formål** | Forstå opbygningen af en Razor Pages app der har forbindelse til en database, samt tilføje kode til appen. |
| **Beskrivelse** | Den givne Visual Studio solution rummer en Razor Pages app, der imple­­menterer grundlæggende funktioner (se alle, opret og slet) for tre domæne-klasser **Kunde**, **Bil** og **Leje**. **Kunde** og **Bil** er ret simple klasser uden objekt-referencer, mens et **Leje**-objekt refererer til et **Kunde**-objekt og et **Bil**-objekt. |
| **Trin** | 1. Download og unzip (VIGTIGT) filen **RosBilRP.zip** fra Moodle. 2. Hvis du ikke allerede har en lokal database **RosBilDB** med de tre tabeller **Kunde**, **Bil** og **Leje** (se filen **DBScript.sql**), skal du oprette denne database og køre scriptet på den. NB: Hvis du ikke kører på en lokal database, skal du selv tilrette connection-string i **RosBilDBContext.cs** (Linje 27). 3. Kør appen. Du skulle gerne kunne lave operationerne ”se alle” (via link fra start-siden), ”opret” (via link i bunden af ”se alle”-siden) og ”slet” (via knap på de enkelte elementer på ”se alle”-siden) for alle tre klasser. Hvis du stød­er på problemer, er det sandsynligvis din database, der ikke er sat korrekt op. 4. En Razor Pages app er kompleks, og der er mange filer at holde styr på. Overordnet er koden delt op i disse foldere:    1. *Models*: her ligger de auto-genererede (ved brug af *EF Core Power Tools*) domæne-klasser og **DBContext**-klassen, interfacet **IHarId** samt filen **Extensions.cs**.    2. *Pages*: Her ligger alle vores Pages. De pages der er specifikke for en domæne-klasse ligger i en tilsvarende sub-folder. I *Shared*-folderen ligger **\_Layout.cshtml**, som også er vigtig.    3. *Services*: Her ligger alle klasser relateret til de repositories, hvori domæne-objekterne er opbevaret når appen kører.    4. *(Roden)* Her er det primært **Program.cs** der er vigtig, samt database-scriptet. 5. Start med at orientere dig i *Models*-folderen. Husk, at det kun er interfacet **IHarId** samt **Extensions.cs** der er ”håndlavet”, resten er auto-genereret. Samlet set repræsenter dette vores domæne-model. Se filerne igennem, og tag notater hvis der er noget du føler du ikke forstår. 6. Fortsæt til *Services*-folderen. Alle klasser og interfaces i denne folder imple­men­­terer repositories, som ikke selv er domæne-klasser, men derimod ”software”-klasser, der hjælper os med at holde styr på alle vores domæne-objekter, når programmet kører. I denne app bruges Entity Framework Core (**EFCore**) som teknologi til at interagere med databasen. Det afspejler sig i klassen **EFCRepositoryBase**, som implementerer det ”teknologi-neutrale” interface **IRepository**, ved brug af **EFCore**. Prøv også her at oparbejde en forståelse af disse klasser/interfaces, og tag notater hvis der er noget du ikke forstår. 7. Den sidste store folder er *Pages*. Her ligger alle vores Razor Pages (og det er faktisk en regel, at alle pages skal ligge i denne folder). For hver domæne-klasse er der tre Razor Pages (Alle, Opret og Slet). Hver enkelt Razor Page består af et ”page layout” (**.cshtml**-filen) samt en tilhørende ”page model” (**.cshtml.cs**-filen). Arbejd dig igennem de tre Razor Pages for **Kunde**, indtil du har en rimelig forståelse for strukturen. Herefter kan du arbejde dig gennem de samme pages for **Bil** og **Leje**. Du vil nok se store ligheder, men også nogle forskelle. Tag igen notater hvis der er noget ud ikke forstår. 8. Nu får du den første opgave, hvor du selv skal kode noget 😊. Brugerne af appen synes at det i ”se alle” for **Leje** burde fremgå, hvad den samlede pris for den enkelte udlejning er. Hvis man f.eks. har lejet en bil i 4 dage som koster 750 kr/dag, er den samlede pris 3000 kr. **Du skal nu lave denne ændring til appen!** Det er ikke noget der kræver mange linjers kode, men det kræver man har opbygget en rimelig forståelse af appens struktur. 9. Vores domæne-model bliver nu udvidet med en klasse **Ansat**. En **Ansat** har et Id, et Navn og en Email. Du skal nu implementere alle de elementer i programmet, som er nødvendige for at kunne lave de samme operationer (se alle, opret og slet) som vi kan foretage for **Kunder** (du kan med fordel lægge dig tæt op af implementationen af **Kunde**). Du får sikkert (mindst) brug for at gøre følgende:    1. Udvide database-scriptet med en **Ansat**-tabel (brug **Kunde**-tabel som inspiration).    2. Køre det opdaterede script på databasen.    3. Køre *EFCore Power Tools*, og få genereret den nye klasse, samt en opdateret **DBContext**-klasse.    4. Lave relevante ”extensions” til den nye **Ansat**-klasse i **Extensions.cs**.    5. Tilføje **IAnsatRepository** og **AnsatRepository** til *Services*-folderen, og registrere denne nye service i **Program.cs**.    6. Lave en ny folder **Ansatte** i *Pages*-folderen, og til den tilføje tre nye Razor Pages (Alle, Opret og Slet)    7. Tilføje et nyt ”nav-item” til **\_Layout.cshtml**, så man kan navigere hen til Alle-page for **Ansat**. 10. Vores domæne-model bliver nu udvidet med en mere kompleks klasse **Opgave**. En **Opgave** har en reference til en **Ansat** og en **Leje**. Et enkelt **Opgave**-objekt repræsenterer således en form for opgave, som en ansat skal udføre i forbindelse med en udlejning (f.eks. ”Tjek at bilen er rengjort”, ”gå den afleverede bil efter for skader”, og så videre). På den måde har **Opgave** en struktur der minder meget om **Leje**, så derfor kan man også her lade sig inspirere af, hvordan implementationen ser ud for en eksisterende klasse (i dette tilfælde **Leje**-klassen). Du skal således gennem de samme trin som for **Ansat**-klassen, men de enkelte trin kan godt være mere komplekse. 11. Nu har vi fem domæne-klasser, og fem varianter af ”se alle”, ”opret” og ”slet”. Hvis du sammenligner f.eks. de fem udgaver af ”page model”-klasser­ne for ”slet”, vil du nok se at de minder rigtig meget om hinanden. Minder de så meget om hinanden, at vi kan definere en base-klasse for disse fem klasser? Overvej dette, og prøv at se hvor langt du kan komme med at implementere og bruge en sådan base-klasse. Og mon ikke også det gælder for ”page model”-klasserne for ”se alle” og ”opret”? |