24-timers opgave - Hotel Booking System

Programmering-2, 3. semester KEA, Efterår 2023

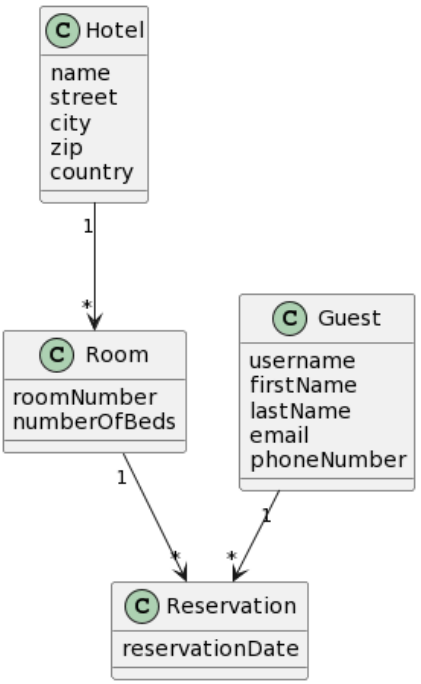
I dette eksamensprojekt skal du designe en simpel full-stack web application, QuickBook, for et "nyt Hotel Booking Firma". Fokus vil være på at håndtere hoteller, hotelværelser og gæsters reservationer.

Backend-delen skal implementeres med Spring-Boot, JPA/Hibernate og en MySQL-database. Frontend skal være en **separat** HTML/CSS/JavaScript-applikation.

Det forventes, at du arbejder alene på øvelsen og ikke deler nogen form for kode/idéer med andre, hvilket vil blive betragtet som eksamenssnyd. *Se i denne forbindelse også afsluttende afsnit om brug af ChatGPT og lignende.*

Læs **hele opgaven** før du starter, og hvis du mener du vil kunne implementere sikkerhed (se senere), vil det nok være nemmest at tænke dette ind helt fra start.

Sammen med "Product Owner" er følgende indledende domænemodel udarbejdet, som du kan bruge som inspiration for dit design.



Adskillige detaljer mangler i denne model. Eksempelvis bør du håndterer, at et rum har en pris, og en reservation får en pris herfra, som ikke må ændres hvis rummets generelle pris ændres efter reservationen.
Bemærk også at modellen ikke tager stilling til hvordan primary- og foreign keys skal håndteres i den endelige database.

Reelt havde product owner ønsket at en reservation havde haft en *fra*- og *til*-dato. For denne opgave er det dog acceptabelt at nøjes med en enkelt dato. Vil man reservere værelset for en længere periode, må man lave flere reservationer.

Følgende opgaver er stillet af product owner

Bemærk at selvom det ikke er nævnt specifikt under følgende delopgaver, skal du sørge for at inkludere relevante testcases i din backend.

Opgave-1: Backend-projektet og database

Analyser den foreslåede domænemodel og vær sikker på at du, ideelt med dine tilføjelser, forstår hvordan de forskellige entiteter interagerer med hinanden. Hvad repræsenterer hver entitet, og hvad er deres ansvar i systemet?
Implementer et Spring Boot-projekt med database og JPA for din tilrettede version af oplægget. Tilføj, for alle entiteter, to ekstra felter, *created* og *updated*, begge af typen `LocalDateTime`.

Opgave 2a: Hoteller og værelser → Test Data

Product Owner regner med at der på sigt kommer rigtigt mange hoteller med i ordningen. Implementer kode, som ved opstart laver 250 "dummy hoteller" og for hvert hotel opretter et antal værelser (10-40) med fra 1-4 senge.
Overvej i næste trin hvordan dette antal (og flere) håndteres mest hensigtsmæssigt på både klient og server

Opgave-2b: Hoteller og Værelser

Impleplementere et REST API + en tilsvarende HTML/CSS/JavaScript-klient, der implementerer funktionalitet angivet herunder:

A. Oprette et Hotel
B. Se alle Hoteller, **kun** → id, navn, adresse og antal Værelser
C. Find og se detaljer for et enkelt hotel, **kun** → id, navn, adresse og antal Værelser
D. Opdatere et hotel
E. Slette et hotel (når du til opgave 3, bør et hotel ikke kunne slettes hvis der er værelser reserveret)
F. Oprette et Værelse og tilknytte det til et Hotel

Opgave-3: Værelser og Gæster

Implementer følgende, både backend og frontend:

A. Mulighed for en ny bruger for at oprette sig selv (glem, edit, slet mm her, kun opret)
B. Et endpoint til at oprette en Reservation for en Gæst af et værelse for en given dato (hvis det er ledigt)
C. Et endpoint til at annullere en specifik reservation for et værelse

*Vælg nu **en**, **og kun en**, af følgende muligheder, **4a)** eller **4b)***

Opgave-4a)

Product owner har yderligere et ønske om, at et hotel skal kunne klassificeres via en eller flere typeangivelser (havudsigt, parkering, morgenmad inkluderet, tæt på city, swimmingpool, mm)

A. Tilføj muligheden for **kun** at se hoteller af en vis type (familie, havudsigt, mm)
B. Se alle værelser i et hotel, **kun** →(id, roomNumber, numberOfBeds)
C. Se alle ledige værelser i et hotel, for en given dato
D. Find og se detaljer for ét værelse, **kun** → (id, roomNumber, numberOfBeds)

Opgave-4b: Sikkerhed

Tilføj sikkerhed med en login-funktion. Opdel dine endpoints i nedenstående grupper og tilføj kode nødvendig for at håndhæve disse restriktioner

- Anonyme brugere
- Brugere logget på som gæst
- Brugere logget på som administrator

Opret et, eller ændr et eksisterende, endpoint, der tillader gæster logget ind, at se alle deres reservationer (kun token må sendes med i request, ikke username).
Passwords må selvfølgelig ikke gemmes i plain text

Tips:
Brug ikke for meget tid på, hvem der kan gøre hvad (ud over hvad der lige er bedt om). Tag bare en hurtig beslutning for hvert endpoint

Du er velkommen til at bruge hardkodede admins/brugere opsat i koden til denne del.

Yderligere Features - hvis du har tid/lyst

Du er velkommen til at tilføje yderligere features, som du finder relevante for at vise dine færdigheder, hvis de ikke er inkluderet ovenfor. Dette kan omfatte den anden valgmulighed for del-4, pagination, responsive design, eller andet.

Start **KUN** på denne del, hvis/når opgave 1-4 er fuldført.

Krav til aflevering

Du skal aflevere **et enkelt dokument** via Wiseflow med følgende oplysninger senest kl. 09.00 dagen efter, du har modtaget opgaven:

Dit fulde navn og KEA-e-mail

Et link til dit **PRIVATE** GitHub-repo (to når backend og frontend er i separate repositories)

Bemærk → Offentlige repositories vil IKKE blive accepteret

Når du opretter dine repositories, skal du tilføje følgende **to (tre)** GitHub-brugere (vælg din klasse herunder) som collaborators (husk dette for både frontend og backend).

A-klassen	ERLM@kea
	MANY@kea.dk
B-klassen	LMOR@kea.dk
	IANB@kea.dk
C-klassen	JART@kea.dk
	SIEB@kea.dk
	NIFR@kea.dk

Dette dokument skal også indeholde en kort beskrivelse (8-15 linjer), der beskriver, hvor langt du er kommet med projektet. Hvis du er nået til del-4, bør det fremgå hvilken mulighed du har valgt.

Vigtigt:

Du må IKKE uploade til dit GitHub-repo efter kl. 09.00, 24 timer efter du har modtaget opgaven. Dette vil gøre din aflevering ugyldig.

Forberedelse til den efterfølgende mundtlige eksamen

Selvom du ikke må uploade til dit repo som forklaret ovenfor, er du mere end velkommen til at lave lokale ændringer i din kode, og hvis du beder om det, vil du få et par minutter til at præsentere disse ideer.

Under eksamen vil du sandsynligvis blive bedt om at foretage mindre ændringer i din kode.

Om brug af ChatGPT eller lignende AI-værktøjer

Du er velkommen til at bruge ChatGPT, eller en tilsvarende sparringspartner, men hvis du gør, er det vigtigt at forstå, at **du** er ansvarlig for den kode, du afleverer. Det betyder, at du forventes at kunne forklare, og eventuelt rette, koden, som, at du selv havde skrevet den. Husk altid at læse ChatGPT's supplerende forklaringer og generelt vurdere, om et AI-forslag nødvendigvis er det rigtige. At skrive supplerende tests (selv) for code snippets givet af en AI, er en god måde at både komme til at forstå koden bedre, og sikre at den virker som forventet.