24 timers eksamensprojekt

Programmering 2, 3. semester KEA, E23.

Denne eksamensopgave udgives: 16. januar 2025 kl. 09:00
Besvarelse skal afleveres senest: 17. januar 2025 kl. 09:00

Du skal i dette eksamensprojekt bygge en simpel **full-stack REST web-applikation**. Opgavens tema er **Drone Pizza** og består af en front-end og en back-end. Back-end skal laves med brug af Spring Boot, H2 og MySQL. Front-end skal laves med HTML/CSS og JavaScript.

Eksamensopgaven består af **3 separate delopgaver**.

Vigtigt:

- Læs hele opgaveteksten før du går i gang, så du bedre ved hvor meget tid hver del tager.
- Det forventes at du under hele eksamensperioden arbejder alene på din besvarelse. Enhver deling af kode eller idéer med andre vil blive betragtet som eksamenssnyd.
- Når afleveringsfristen har passeret, er det ikke tilladt at push'e ændringer til GitHub. Sørg derfor for at aflevere løbende ved at commit'e og push'e løbende. Vent ikke til der er får minutter tilbage, før du afleverer noget som helst. Afprøv at aflevering virker.
- Det er tilladt at bruge AI-baserede hjælpeværktøjer som ChatGPT, Claude eller Copilot. Til den efterfølgende mundtlige prøve vil du dog blive bedt om at tilføje rettelser eller udvidelser til din kode, uden andre hjælpemidler end IntelliJ, så vær forberedt på det.

Sådan afleverer du:

- Til programmeringsdelen skal du lave et Spring Boot-projekt, hvor du gør brug af Spring Web, JPA, MySqL og H2. Du kan også inkludere Thymeleaf for at distribuere dit HTML og JavaScript. Alle tre delopgaver kan løses inden for dette projekt. Sørg for at **push'e til GitHub senest kl. 09:00** dagen efter du har modtaget opgaven. **Derefter er det ikke tilladt at push'e ændringer til GitHub.**
  Programmeringsdelen er afleveret, når du har push'et den. Sørg for at aflevere både backend med JUnit tests og frontend.
  Når du opretter dine **private** repositories skal du tilføje GitHub-brugere for din klasse som collaborators til dine repositories (husk både frontend og backend-projekt, hvis du har to).

Klasse	GitHub-brugere
DAT-E23A	• ERLMkea • nynnejc
DAT-E23B	• sshine • SigneBorch • kommuniker
DAT-E23C	• JarlTuxen • mindhack-dk

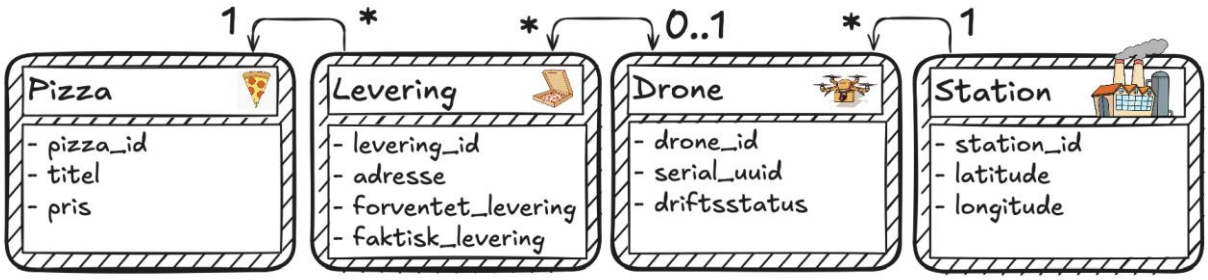
- Via Wiseflow skal du **aflevere et enkelt dokument senest kl. 09:00** dagen efter du har modtaget opgaven med følgende oplysninger:
    - Dit fulde navn og KEA-email
    - Et link til dit **private** GitHub repository (to når backend og frontend er i separate repositories)
      Bemærk: Offentlige repositories vil ikke blive accepteret.
    - En kort beskrivelse (5-15 linjer) af hvor langt du er kommet med projektet. Hvis enkelte dele har åbenlyse mangler, må du gerne skrive det her. En åbenlys mangel kunne være noget der ikke er lavet færdigt, eller noget som forårsager at programmet fejler på uønskede tidspunkter. Du må også gerne skrive, i hvilken grad din back-end er afprøvet med unit tests, og hvordan du ville udvide dine tests, hvis du havde mere tid.

Delopgave 1: Database med JPA og H2/MySQL

Modellér følgende problemdomæne:

[Følgende er fiktion.] For at styrke erhvervslivet, har regeringen som prøveordning gjort det lovligt at flyve droner til varelevering i København. Du er blevet hyret til at afprøve, om der er penge i at levere pizzaer fra luften. Efter at have undersøgt problemdomænet, har du lært at:

- En **Drone** har et offentligt kendt serienummer (UUID), som ikke skal være primærnøgle.
- En **Drone** er enten "i drift", "ude af drift" eller "udfaset".
- En **Pizza** har en titel og en pris i hele kroner. Da DronePizza laver pizzaer med robotter på samlebånd, kan man kun bestille pizzaer fra menuen uden ekstra fyld. Det er for dyrt!
- En **Drone** flyver altid fra samme **Station** som har et GPS-koordinat (latitude, longitude), som på dansk hedder længdegrad og breddegrad og er komma-tal.
- En **Drone** har mange **Leveringer**, og en **Levering** har én **Pizza**.
- En **Levering** bliver udført af én **Drone** til én adresse på et *forventet* leveringstidspunkt, som altid er defineret, og med et *faktisk* leveringstidspunkt som kan være udefineret. En **Levering** starter sit liv uden en **Drone** og uden et *faktisk* leveringstidspunkt.



Sørg for at der er mindst tre Stationer og fem Pizzaer i databasen, når applikationen starter. Centrum af København er i opgaven defineret til (55°41N 12°34E) som betyder "55,41 grader nord og 12,34 grader øst". Sørg for at Stationerne har forskellige koordinater, og ikke er alt for langt fra Centrum af København.

Delopgave 2: Byg et REST API

DronePizza har brug for en back-end til at koordinere leveringen af pizzaer. Back-end'en skal levere JSON-værdier som kan oplyse om dronernes og leveringernes tilstand, ændre dronernes tilstand, samt oprette nye leveringer, tilskrive en levering til en drone, og markere en levering som afsluttet. Her er en række krav til endpoints:

- **/drones** skal give en liste af alle Droner, deres UUID, driftsstatus, og Station.
- **/drones/add** skal oprette en ny Drone og koble den til Stationen som har færrest Droner. Hvis der er flere stationer med lige få droner, er det underordnet hvilken der får dronen. Nye Droner, skal have et tilfældigt offentligt UUID, og deres driftsstatus skal være "i drift". Hvis der ikke er nogen Stationer i databasen, skal oprettelse af Dronen fejle.
- **/drones/enable** skal skifte en givet Drones status til "i drift".
- **/drones/disable** skal skifte en givet Drones status til "ude af drift".
- **/drones/retire** skal skifte en givet Drones status til "udfaset".
- **/deliveries** skal give en liste af alle leveringer, der **ikke** er leveret.
- **/deliveries/add** skal tilføje en bestilling af en bestemt pizza. Forventet levering skal være en halv time senere end det øjeblik som endpoint'et kontaktes, og der skal ikke vælges nogen drone koblet til leveringen.
- **/deliveries/queue** skal give en liste af alle leveringer, der mangler en drone.
- **/deliveries/schedule** skal tage en levering, der mangler en drone, og tilskrive den en drone. Om det er en drone som er angivet med en request-parameter, eller om det er en tilfældig drone er op til dig. Dronen må også gerne være i gang med en anden levering. Men en levering som allerede er i gang med at blive leveret (dvs. har en Drone) skal fejle, hvis den bliver skeduleret. Og brug af en drone, som ikke er "i drift" skal fejle.
- **/deliveries/finish** skal tage en givet levering, der er i gang, og markere den som færdig i det øjeblik som endpoint'et kontaktes. Hvis leveringen ikke har en Drone, skal den fejle.

For hvert endpoint:

- Afgør om endpoint'et skal benytte **GET eller POST** (vi ser bort fra andre metoder).
- Afgør hvilke **request-parametre**, hvis nogen, endpoint'et er bedst tjent med (fx ID'er).
- Overvej, hvis endpoint'et kan fejle, hvilken HTTP-statuskode, der passer, og hvad en god fejlmeddelelse ville være. Benyt exceptions til at levere fejlen.

For at afprøve endpoint'et:

- Prøv at kalde hvert endpoint mindst én gang når du arbejder med opgaven
- Benyt en API-klient som fx Postman til manuel kontakt af endpointet
- Benyt JUnit til at afprøve endpoint'et programmatisk

Fokus for evaluering af delopgave 2 er, at kravene ovenfor er korrekt forstået og implementeret.

Delopgave 3: Front-end med HTML/JavaScript

DronePizza har mange konkurrenter, og vil gerne i gang med at sende droner ud før deres software er helt færdig. De forventer derfor kun et enkelt skærmbillede er tilgængeligt via HTML/CSS og JavaScript:

- Man skal kunne se en liste af alle leveringer, der ikke er leveret endnu, sorteret så ældste leveringer står øverst. Det må gerne fremgå om leveringen mangler en drone, eller om den har en drone, men bare mangler at blive leveret.
- Listen må gerne opdatere sig selv hvert 60. sekund
- En knap for hver levering, der mangler en drone, som tilføjer en drone til den.
- En knap til at oprette nye droner.

For at en kundedemonstration af front-end'en virker overbevisende, mangler der dog oprettelse og afslutning af levering. Man kan trods alt ikke vælge en drone for leveringer, eller markere en levering for afsluttet, hvis en levering ikke er oprettet. Da det er op til kunderne og dronerne at oprette og afslutte leveringer, vil DronePizza derfor gerne se noget funktionalitet, der efterligner kunder og droner:

- Mulighed for at simulere, at en levering af en pizza bliver oprettet (af kunden)
- Mulighed for at simulere, at en levering bliver leveret (af dronen)

Hvordan simulationen skal foregå er op til dig: Det kan være nogle objekter, som bliver oprettet i databasen, når applikationen starter, eller nogle knapper i front-end'en som simulerer kunder og droner, eller nogle baggrundsprocesser i back-end'en som udfører database-operationer. Det er ikke vigtigt at lave noget sofistikeret her:

Fokus for evaluering af delopgave 3 er, at listen med leveringer virker, at knapperne virker, og at en demo kan gennemføres. Dvs. at der ikke mangler relevant data i databasen som fx droner, pizzaer, stationer, eller at leveringer oprettes og afsluttes.