IT-Case:

Webbutveckling

2020

Lunicore

Lunds universitets studentkonsultbolag

1. Introduktion

Nu har du förhoppningsvis satt dig in i någon av programstackarna som du kan välja att genomföra ditt case i. Detta dokument innehåller samtliga instruktioner som krävs för att kunna genomföra caset.

Vi är väl medvetna om att detta är en tidskrävande uppgift och har förståelse för ifall en komplett lösning inte kan presenteras under intervjun. Det viktigaste är att du visar en god förståelse för uppgiftens olika delar, kan föra ett tekniskt resonemang och kan motivera tillvägagångssätt.

2. Inlämning

Lämna in din lösning genom att skicka ett mejl till itcase@lunicore.se. Du har 3 dagar (72h) på dig från det att du fick detta dokument. I ditt mail ska du skicka en länk till ett git-repository där hela din lösning ska finnas incheckad. Vi mottager inga filer bifogade i mejlet.

3. Nästa steg

Du har redan blivit inbokad för en case-intervju där du kommer få presentera din lösning. Under intervjun förväntas du bland annat motivera val av språk och verktyg. Du ska även kunna presentera en övergripande beskrivning av vilka komponenter ditt program består av och hur dessa kommunicerar med varandra.

Efter presentationen kommer du ställas inför vidare utmaningar. Du kan förbereda dig genom att känna till ditt program, det språk det är byggt i, samt tänka igenom möjligheter för vidareutveckling av ditt program.

4. Frågor

Du kan ställa frågor genom att mejla itcase@lunicore.se.

5. Uppgift 1

Du ska göra en grundläggande webbserver för en bilhandel. Du får välja bland följande databaser och stacker:

- Node.js + React (rekommenderas)
- Ruby on Rails (rekommenderas)
- Node.js + HTML/CSS/JavaScript
- Node.js + Vue • Node.js + Angular
- PHP + HTML, CSS & JavaScript (ok att använda PHP-ramverk)

Du får även använda andra hjälpmedel för både back-end (t.ex. express, restify) och front-end (t.ex. Bootstrap).

För databaser:

- SQLite
- MySQL
- PostgreSQL
- MongoDB
- Andra SQL- eller NoSQL-system

Om du väljer att använda Ruby on Rails finns separata instruktioner för uppgiften under sektionen *För dig som har valt Ruby on Rails* på efterkommande sidor. I detta fall behöver du inte läsa informationen nedan, utan gå i stället direkt till sektion som behandlar Ruby on Rails.

Bifogat med detta dokument har du fått en JSON-fil. Denna fil innehåller all data men ska inte användas som en databas i din lösning. Använd en av databas-systemen som listats ovan. Skriv gärna en funktion eller ett script som läser in JSON-filen och importerar datan till din databas. Försök dessutom lösa åtminstone en av extrauppgifterna.

Ditt program ska vara portabelt, och vara trivialt att starta. Det ska alltså vara enkelt att återskapa miljön som ditt program kräver (så som databaser och språk), samt att starta servern. Ditt program ska vara körbart på macOS och Linux. Du bestämmer själv hur du uppnår detta. En optimal setup innebär att programmet kan starta med ett enda kommando, naturligtvis efter att all nödvändig programvara installerats. Detta moment är lika viktigt som att ditt program uppfyller kraven, men fokusera först på att lösningen innehåller en bra implementation.

Din webbserver ska definiera fyra endpoints. Data som skickas till och från servern, skall skickas i body och vara representerad i JSON-format. Ha i åtanke att det ska finnas möjlighet att på ett enkelt sätt utveckla ditt program vidare.

6. Endpoints

1. GET /employees
   Skall returnera en lista över samtliga anställda, samt deras parametrar.

2. GET /carmodels
   Skall returnera en lista över samtliga sparade bilmodeller, samt deras parametrar.

3. POST /carmodels
   Skall spara en ny bilmodell, samt returnera den.

4. DELETE /carmodels
   Ta bort en bilmodell, samt returnera den.

5. GET /total_sales
   Skall returnera en lista över samtliga anställda, samt deras parametrar. Dessutom skall en parametes ”sales” finnas för varje anställd, som ackumulerar deras säljbelopp.

Exempel:
{
”*name*”: ”*Kalle*”,
... övriga employee-parametrar...
”*sales*”: 133000,
}

7. Uppgift 2

När endpointsen är färdiga ska du ta fram en front-end som visar datan och gör följande funktionalitet tillgänglig. Prioritera kvalitet över kvantitet:

1. Skapa en front-end-lösning som visar all data genom att använda de implementerade endpointsen. Det ska också gå att utföra åtgärder som att lägga till och ta bort bilmodeller.

2. Implementera funktionalitet för att skapa konton och logga in som en användare. En användare ska kunna ha en direkt koppling till en anställd, eller ingen alls. Om en användare, som också är anställd, loggar in, ska denne kunna lista sina parametrar (id, namn och genomförda sälj) i en profilvy.

3. Snygga till din design med hjälp av HTML/CSS och skapa en användarvänlig frontend. Ta inspiration från hemsidor som du tycker ser bra ut.

Extrauppgifter

Finns det tider över får du gärna försöka dig på följande extrauppgifter, som självklart är meriterande för din bedömning.

4. Om uppgift 3 är färdigställd: Inför två access-nivåer för anställda, en admin och en standard. Användare som har standard access-nivå ska inte kunna ändra, ta bort eller skapa nya användare. Admins ska ha access till allt.

5. Om uppgift 4 är färdigställd: Implementera funktionalitet för att användare som glömt sitt lösenord ska kunna ändra sitt lösenord, alternativt få ett nytt, genom ett mejl till den mejladdress som registrerats vid skapandet av användaren.

Lunicore
Lunds universitets studentkonsultbolag

För dig som har valt Ruby on Rails

1. Uppgiftsbeskrivning

Du ska göra en grundläggande webbapplikation för en bilhandel. Du har valt att använda Ruby on Rails och får välja en tillhörande databas till din applikation. Rekommenderat är att använda PostgreSQL som är standard för Heroku, en molntjänst för att hosta webbapplikationer. Du får gärna använda front-end-verktyg som Bootstrap för att underlätta designen av din applikation.

Bifogat med detta dokument har du fått en JSON-fil med information som ska populera databasen. Ditt program ska vara portabelt, och trivialt att starta. Det ska alltså vara enkelt att återskapa miljön som ditt program kräver (så som databaser och språk), samt att starta servern. Ditt program ska vara körbart på macOS och Linux. Du bestämmer själv hur du uppnår detta. Molntjänsten Heroku är här ett bra val.

Om du har tid över, är det meriterande om du försöker dig på någon av extrauppgifterna.

2. Applikationens funktionalitet

Webbapplikationen som du skapar ska ha följande funktionalitet. Prioritera kvalitet över kvantitet:

1. Modeller med tillhörande controllers för: användare (user), anställda (employee), bilmodeller (carmodel) och sälj (sales). Alla controllers ska ha tillhörande metoder för att returnera, skapa, ändra och ta bort respektive modell.
   - User: User-modellen ska användas för inloggning till webbapplikationen. Lämpliga attribut är alltså email och password. En user kan antingen ha en koppling till en employee, eller ingen alls.
   - Employee: Employee-modellen ska ha de attribut som är beskrivna i jsonfilen under “emloyees”. En employee kan ha flera sales-modeller kopplade till sig. Se struktur på nästa sida.
   - Carmodel: Carmodel-modellen ska ha de attribut som är beskrivna i jsonfilen. En Carmodel kan ha flera sales-modeller kopplade till sig. Se struktur på nästa sida.
   - Sales: Sales-modellen ska ha de attribut som är beskrivna i json-filen.

2. Lägg till och koppla ihop de employees, carmodels och sales som finns beskrivna i json-filen.

3. Följande vyer ska finnas för att visualisera och ändra databasens innehåll:
   - User: I users-vyn ska det vara möjligt att ändra användarnamn och lösenord. Om user-modellen är kopplad till en employee, ska det vara möjligt att lista den kopplade employee:ns attribut. Det ska även vara möjligt att se hur mycket employee:n har sålt för.
   - Employees: I denna vy ska det vara möjligt att lista samtliga anställda, samt deras attribut.
   - Carmodels: I denna vy ska det vara möjligt att lista samtliga bilmodeller med tillhörande attribut, samt vara möjligt att lägga till/ta bort bilmodeller ur databasen.

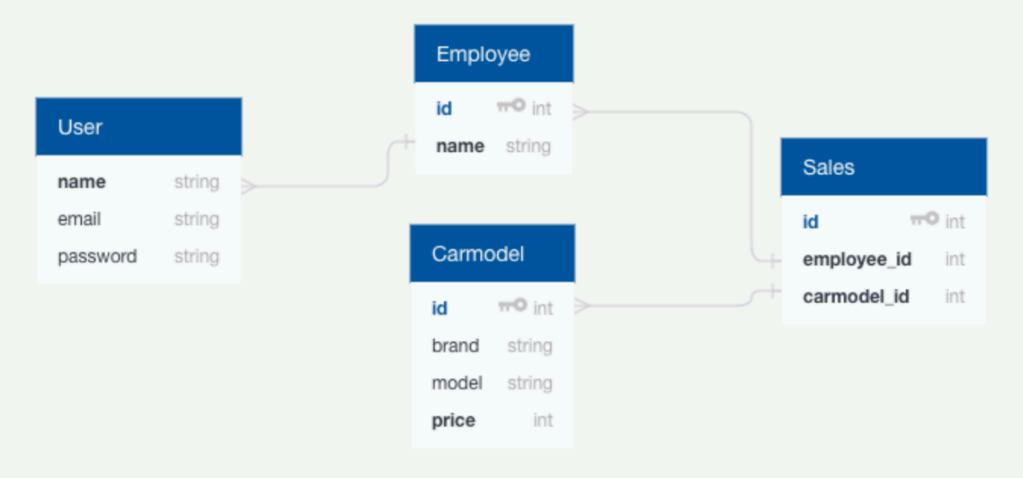
4. Snygga till din design med hjälp av HTML/CSS och skapa en användarvänlig applikation. Använd gärna Bootstrap eller dylika web-frameworks för att underlätta designen. Ta inspiration från hemsidor som du tycker ser bra ut.

Extrauppgifter

Finns det tider över får du gärna försöka dig på följande extrauppgifter, som självklart är meriterande för din bedömning.

5. Lägg till möjligheten att, i employee-vyn se den ackumulerade säljsumman för företaget.

6. Om uppgift 3 är färdigställd: Inför två access-nivåer för användare, en admin och en standard. Användare som har standard access-nivå ska inte kunna ändra, ta bort eller skapa nya bilmodeller. Admins ska ha access till allt.



Arbeta med framtiden idag!

AF-borgen
Sandgatan 2
223 30 Lund

www.lunicore.se
info@lunicore.se

Lunicore

Lunds universitets studentkonsultbolag