YH, Systemutvecklare – agil webbprogrammering 400 Yh-poäng (utbildningsnummer: 201406900)

Fortsättningskurs webbserverprogrammering
Gruppinlämningsuppgift

Scenario
En bilverkstad vill kunna lagra (skapa, läsa, ändra och ta bort) information om följande i en databas:

1) Vilka fordon som för tillfället är under reparation (reg. nr, modell etc).
2) Kunder (personnummer, adress etc)
3) Vilka kunder som lämnat in vilka fordon.
4) Vilken skador/fel som ska åtgärdas på varje fordon.
5) Vilka anställda man har.
6) Vilka anställda som arbetar/har arbetat med varje fordon.
7) Vilka reservdelar man har i lager och vilka fordonsmodeller de passar.
8) Om en reparation/åtgärd av ett fordon är kommande, pågående eller avslutad.
9) Arbetstid nedlagd på ett visst fordon.
10) Vilka tider olika anställda har semester.

YH, Systemutvecklare – agil webbprogrammering 400 Yh-poäng (utbildningsnummer: 201406900)

Fortsättningskurs webbserverprogrammering

Gruppinlämningsuppgift

35 yrkeshögskolepoäng

För att få uppnå betyget G:

- Arbeta i grupp, i ett agilt team, och använd versionshantering via GitHub under hela detta arbete.
- Fundera på hur man bryter ned scenariot i olika entiteter (som i nästa steg blir Mongoose-modeller).
- Använd era tidigare kunskaper om MongoDB och Mongoose för att skapa Mongoose-modeller som på ett bra sätt representerar scenariot (se ovan). Skapa minst 5 olika Mongoose-modeller.
- Skriv en programklass som utifrån Mongoose-modulerna tillhandahåller ett generellt REST-gränssnitt, där routes automatiskt skapas för varje Mongoose-datamodell. (Hitta på ett sätt att registrera Mongoose-datamodeller via ett metod-anrop till en instans av din klass.)
- Via REST-gränssnittet din kod skapar ska det för alla modeller gå att skapa, läsa, ändra och radera data (CRUD) i MongoDB.
- Separera er kod i olika filer utifrån moduler och klasser.
- Strukturera er kod så att asynkrona anrop till filsystem och databaser oftast används för att undvika blockering av Node.js enda tråd.
- Genomför en grundläggande serveroptimering – antingen så att kompression används vid dokumentöverföring eller att data, som webbläsaren redan har en aktuell version av, inte skickas igen.
- Använd sessioner för att på ett grundläggande sätt för att identifiera en specifik klient. Pröva att detta fungerar genom att som en svarsheader för varje REST-anrop lägga till ett unikt id som identifierar klienten. Headern ska heta *X-Client-Id*.
- Ändra er kod så att förenklad version av applikationen påbörjas, i vilken MongoDB och Mongoose ersätts med en MySQL-databas. (Som påbörjat räknas att du skapar en MySQL-databas med en tabell motsvarande varje Mongoose-modell, samt att du lyckas få Node att ställa frågor till MySQL-tabellerna.)

OBS! Ni får **INTE** lov att använda er av bibliotek som automatiskt skapar REST-router från datamodeller (t.ex. Mongresto).

Du bedöms individuellt utifrån att du

- Ska ha haft ett grundläggande deltagande under planering och samarbete i det agila teamet.
- Samarbetat genom att versionshantera din kod tillsammans med gruppen på ett i grunden korrekt sätt.
- Har skrivit en andel av koden som motsvarar minst din andel i teamet delat med två.

YH, Systemutvecklare – agil webbprogrammering 400 Yh-poäng (utbildningsnummer: 201406900)

Giltig från 2016-08-25.

Fortsättningskurs webbserverprogrammering

Gruppinlämningsuppgift

35 yrkeshögskolepoäng

För att få betyget VG krävs att ni (*skillnader mellan G-kraven i möjligaste mån markerade med fetstil*):

- Arbeta i grupp, i ett agilt team, och använd versionshantering via GitHub under hela detta arbete.
- Fundera på hur man bryter ned scenariot i olika entiteter (som i nästa steg blir Mongoose-modeller).
- Använd era tidigare kunskaper om MongoDB och Mongoose för att skapa Mongoose-modeller som på ett bra sätt representerar scenariot (se ovan). Skapa minst 5 olika Mongoose-modeller.
- Skriv en programklass som utifrån Mongoose-modulerna tillhandahåller ett generellt REST-gränssnitt, där routes automatiskt skapas för varje Mongoose-datamodell. (Hitta på ett sätt att registrera Mongoose-datamodeller via ett metod-anrop till en instans av din klass.)
- Via REST-gränssnittet din kod skapar ska det för alla modeller gå att skapa, läsa, ändra och radera data (CRUD) i MongoDB.
- Separera er kod **noggrant** i olika filer utifrån moduler och klasser.
- Strukturera er kod så att asynkrona anrop till filsystem och databaser **alltid** används för att undvika blockering av Node.js enda tråd.
- Genomför en serveroptimering – så att kompression används vid dokumentöverföring **och** data, som webbläsaren redan har en aktuell version av, inte skickas igen.
- Använd sessioner för att på ett **heltäckande sätt** för identifiera en specifik klient. **(Gör detta genom att skapa en klass som tillhandahåller routes för inloggning, samt lagra användare och lösenord i databasen.)** Pröva att detta fungerar genom att som en svarsheader för varje REST-anrop lägga till ett unikt id som identifierar klienten. Headern ska heta *X-Client-Id*. **Lägg även till en header med namnet *X-Username* med den inloggade användarens användarnamn.**
- Ändra er kod så att förenklad version av applikationen **slutförs,** i vilken MongoDB och Mongoose ersätts med en MySQL-databas. **(Som slutförd räknas att du skapar en MySQL-databas med en tabell motsvarande varje Mongoose-modell, samt att du lyckas modifiera din programklass som skapar REST-gränssnittet till att fungera gentemot MySQL-tabellerna.)**

OBS! Ni får **INTE** lov att använda bibliotek som automatiskt skapar REST-router från datamodeller (t.ex. Mongresto).

Du bedöms individuellt utifrån att du

- Ska ha haft ett **aktivt och engagerat deltagande** under planering och samarbete i det agila teamet.
- Samarbetat genom att versionshantera din kod tillsammans med gruppen på ett **nyanserat och korrekt** sätt.
- Har skrivit en andel av koden som motsvarar minst din andel i teamet delat med två.

YH, Systemutvecklare – agil webbprogrammering 400 Yh-poäng (utbildningsnummer: 201406900)

Fortsättningskurs webbserverprogrammering
Gruppinlämningsuppgift

Individuellt genomförande
Efter överenskommelse med utbildningsledare kan gruppinlämningsuppgiften genomföras individuellt.

I dessa fall gäller följande (som avsteg från de beskrivningar som ges under **G**- och **VG**-krav):

- All kod ska vara skriven av den studerande.
- Den studerande behöver inte arbeta i grupp, men versionshantering via GitHub används fortfarande under hela arbetet.
- Minst 3 Mongoose-modeller skapas.
- Ett justerat scenario bestående av 4 istället för 10 punkter tillämpas (se nedan).

Justerat scenario vid individuellt genomförande av gruppuppgiften
En bilverkstad vill kunna lagra (skapa, läsa, ändra och ta bort) information om följande i en databas:

1) Vilka fordon som för tillfället är under reperation (reg. nr, modell etc).
2) Kunder (personnummer, adress etc)
3) Vilka kunder som lämnat in vilka fordon.
4) Vilka anställda man har.