FACULTEIT INGENIEURSWETENSCHAPPEN
EN ARCHITECTUUR

Labo frameworks voor serverapplicaties
Groepsopdracht: ORM, nodeJS en websockets

november - december 2020

1 Inleiding

De groepsopdracht uit vier delen:

- het maken van een datalaag met ORM in een javascript-framework,
- een REST-API voor deze datalaag definiëren in OpenAPI-specificatie,
- op basis van de OpenAPI-specificatie een REST-webservices genereren en implementeren in NodeJS,
- een webapplicatie maken in NodeJS die gebruik maakt van websockets en de REST-webservices.

2 Datalaag met ORM

Maak een datalaag in javascript die de principes van ORM toepast. De keuze van het ORM-framework in javascript en de databank is vrij. Hou er wel rekening mee dat de applicatie (en dus ook de databank) gepubliceerd worden op een linux-server (zie puntje 6) en dat de REST-webservices in NodeJS geschreven zullen worden.

De voorwaarden waaraan de datalaag moet voldoen zijn de volgende:

- De datalaag bestaat uit minstens één DAO-object waaraan je objecten kan opvragen, toevoegen, aanpassen, ...
- De datalaag bevat minstens één 1-1-relatie, 1-n-relatie en één n-n-relatie. Voorzie een relatie met en zonder cascade.
- De datalaag bevat overerving.
- De datalaag heeft één of meerdere value-objecten.
- Met de datalaag kan je één object toevoegen.
- Met de datalaag kan je een verzameling van objecten toevoegen.

- Met de datalaag kan je objecten opvragen. Voorzie zowel een “lazy” als niet “lazy” opvraging.
- De datalaag heeft ook een opvraging die gebruik maakt van parameters.
- Met de datalaag kan je objecten aanpassen.

3 Specificatie REST-webservices in OpenAPI

De functionaliteit die ontwikkeld werd in de datalaag moet toegankelijk gemaakt worden via REST-webservices. Pas hiervoor de “API first approach” toe. Dit betekent dat je eerst functionaliteit van de REST-webservices vastlegt en pas daarna start met implementeren. Om de functionaliteit van de REST-webservices te beschrijven gebruik je de OpenAPI-specificatie. Maak een JSON- of YAML-bestand aan op https://editor.swagger.io/ met deze specificatie.

4 Implementatie REST-webservices

Gebruik Swagger om op basis van de specificatie (in JSON of YAML) de REST-webservices te genereren in NodeJS. Werk deze stubs verder uit zodat ze gebruiken van de datalaag uit puntje 2.

5 Webapplicatie in NodeJS

Ontwikkel een webapplicatie in NodeJS met het Express-framework. De webapplicatie maakt gebruik van minstens één van de webservices uit puntje 3 en krijgt ook geregeld updates voor één of meerdere van zijn webpagina’s van de server via websockets.

6 Publiceren op de Virtual Wall

De webapplicatie en een aantal webpagina’s waarmee de REST-webservices interactief getest kunnen worden, zoals bv. op het rechtse deel van https://editor.swagger.io/, moeten gepubliceerd worden op Virtual Wall.

Om de applicatie te deployen krijgen jullie het IP-adres van een machine en een login en wachtwoord. Je kan via een consolevenster inloggen op de server gebruik makende van ssh. (handleiding)

```
ssh login@ipadres
```

Om mappen en bestanden te kopiëren naar de server maak je gebruik van scp. (handleiding)

```
scp padnaarmap login@ipadres:/padopserver
```

Gebruik je liever een grafische omgeving, dan is WinSCP een optie.

Op de server staat poort 3000 open voor HTTP. Je kan de webpagina’s opvragen door in de browser het adres `http://ipadres:3000/pad` in te geven

7 Indienen opdracht

Bewaar geregeld jullie oplossing in jullie github-repository. De locatie van deze repository is https://github.ugent.be/frameworks2020/groep-XX, waarbij XX vervangen wordt door het groepsnummer. Zorg ervoor dat elk teamlid zijn code pusht onder zijn naam. Je oplossing moet ten laatste afgewerkt zijn op dinsdag 15 december 2019 om 22u00. Beschrijf in de README.md hoe de verschillende onderdelen getest kunnen worden.