Examenopdracht Web Expert

60% van de punten van het vak Web Expert worden bepaald door het resultaat van de examenopdracht in combinatie met de mondelinge verdediging. Op de verdediging kan gevraagd worden een kleine uitbreiding of aanpassing in je code te maken. Ook worden er vragen gesteld over je code. Mogelijke voorbeelden van vragen zijn:

Waarvoor dient … in je code?
Hoe zou je je code herschrijven zonder gebruik te maken van … ?

Je mag tot de dag van het examen aan je opdracht werken en plaatst je code op in de repository die je via onderstaande link kan maken.

https://classroom.github.com/a/arGtxc9X

Maak onderstaande REST-API (en client) (Voor alle duidelijkheid: voor een klassieke web-applicatie met templating engine zoals pug, hbs of ejs krijg je geen punten).

De requests & responses van en naar de REST-API gebruiken JSON als data-formaat.

- Splits je code op in routes, controllers, services en models.
- Voorzie voldoende uitgebreide validatie in de models.
- Zorg dat alle input gevalideerd wordt zodanig dat XSS en injection vermeden wordt.
- Voorzie uitgebreide validatie in je models.

Maak een een REST-api om de resultaten van studenten op te volgen. Binnen het systeem worden de rollen teacher en student aangeboden. De authenticatie gebeurt via JWT-tokens.

Binnen het systeem worden evaluaties per vak en per student bijgehouden. Een vak is verbonden met één lesgever.

Deel1 REST-api (/6pt)

De volgende acties moeten geïmplementeerd worden.

(0) Elke gebruiker kan inloggen en krijgt dan een JWT-token

```
POST /login
{"username": "Timmer", password: "tim12321"}
```

(1) Lesgevers krijgen een overzicht van hun vakken.

```
GET /teacher/:userId/courses
```

(2) Lesgevers krijgen per vak een overzicht van alle studenten.

```
GET /teacher/:id/courses/:courseId/students
```

(3) Lesgevers kunnen per vak en per student een evaluatie aanmaken.
Elk resultaat is een cijfer op tien. Een negatief getal is een indicatie dat de student niet deelgenomen heeft aan de evaluatie. Het gewicht is een positief geheel getal kleiner dan 20 en duidt aan hoe sterk de evaluatie door weegt op het eindresultaat. Er kan een optionele boodschap in de evaluatie geplaatst worden.
gBij de evaluatie wordt telkens de datum bewaard.

```
POST /teacher/:id/courses/:courseId/students/:studenId/evaluations
    { "result":10, "weight": 3  }

POST /teacher/:id/courses/:courseId/students/:studenId/evaluations
    { "result":-1, "weight": 1  }

POST /teacher/:id/courses/:courseId/students/:studenId/evaluations
    { "result":10, "weight": 2, "message": "outstanding result"  }
```

(4) Bij actie (2) kan een boven en/of ondergrens vermeld worden. De score wordt berekend als de sum van alle deelgenomen evaluaties vermenigvuldigd met de gewichten gedeeld door de maximaal haalbare score maal 100 (afgerond).

Dus voor het vak wiskunde heeft student Sofie Schuermans
10 voor toets 1 met gewicht 4
-1 voor toets 2 met gewicht 2
8 voor toets 3 met gewicht 7
Het resultaat is hier
`(10*4 + 0*2 + 8*7) / (10*4 + 10*2 + 10*7) * 100 =`
`( 96 / 130) * 100 =`
`73,84.`
Dit wordt afgerond naar 74.

```
GET /teacher/:id/courses/:courseId/students?score_lowerbound=70

GET /teacher/:id/courses/:courseId/students?score_upperbound=80

GET /teacher/:id/courses/:courseId/students?…
             …score_lowerbound=70&score_upperbound=80
```

(5) Een student krijgt een overzicht van zijn vakken

```
GET /student/:id/courses
```

(6) Een student krijgt een overzicht van zijn evaluaties per vak

```
GET /student/:id/courses/:courseId/evaluations
```

Voorzie in alle routes foutmeldingen met correcte statuscode wanneer iemand probeert aan resources te komen waar hij niet geauthenticeerd voor is. Bijvoorbeeld een lesgever probeert de resultaten van een vak waar hij niet aan verbonden is te bekijken. Een student probeert de resultaten van een andere student te bekijken. Voor lower_bound wordt geen getal ingegeven.

Gebruik ondertaande middleware:
- Mongosanitize (https://www.npmjs.com/package/express-mongo-sanitize)
- Helmet (https://helmetjs.github.io/)
- Express rate-limit (https://www.npmjs.com/package/express-rate-limit)

Illustreer de werking van de REST-api via postman of http-client

Bij de evaluatie gelden de onderstaande malussen:
-1.5pt per niet (of foutief) uitgewerkte actie (0-6)
-30% als er een <script> tag in je databank geplaatst kan worden

Deel2 Testing via JEST: (/3)

3 punten verdien je via de tests voor de toepassing. Je maakt functionele tests via supertest en schrijft tests voor de models en services.

Resultaat = code coverage* resultaat deel1

Deel3 Frontend (/3)

Voorzie de frontend voor de acties (0), (5) & (6). Je mag hier kiezen of je werkt met Vue, Angular, React of een ander frontend-framework. Ook mag je werken zonder framework en de client side zelf in Javascript schrijven. Indien je kiest voor een ander frontend-framework dien je eerst via mail toestemming te vragen aan je lector.

Je maakt een gebruiksvriendelijke, elegante toepassing waar de gebruiker eenvoudig tussen de verschillende delen kan navigeren. Zorg er dus zeker voor dat een gebruiker gemakkelijk resultaten kan selecteren (zonder het id te moeten intypen).

Wil je het maximum van de punten halen dan moet je ervoor zorgen dat de code van de REST-api voldoet aan de regels opgelegd door Airbnb (enkel de code niet de tests). Je bewijst dit aan de hand van eslint.

Deel4 Extra punten (/8)

• (/1.2) Maak minstens 5 bijkomende integration tests voor actie (4) die gebruik maken van een in-memory database.
• (/1.2) Voer de REST-api uit op een virtuele machine. Zorg er voor dat bij de communicatie tussen client en server https gebruikt wordt. Je kan bijvoorbeeld mkcert gebruiken om de certificaten te maken.
• (/1.2) Implementeer een blacklist (of denylist) om JWT-tokens te invalideren.
• (/1.2) Voorzie 2 aparte servers voor de REST-api De eerste server dient als identity provider (deze server behandelt de login-requests en stuurt het access-token naar de client). Bij requests naar de 2e server stuurt de client het access-token mee met elk request.
• (/1.2) Voorzie REST-api en client voor de volgende actie. De admin kan een student aanmaken en bij het aanmaken een foto van de student bewaren in de mongodb-databank (dus geen foto uploaden naar een public map op de server maar wel de foto bewaren in de databank).
• (/1.2) Illustreer het gebruik van een dependency injection container in je code.
• (/1.2) Integreer passport in de authenticatie.