**Project: SOA & Cloud computing [SOACL] – MusicPlaylist door Sven Baerten (1540637)**

**Docenten: Prof. Kris AERTS & Stijn SCHILDERMANS**

**Academiejaar: 2018-2019**

**Korte samenvatting:**

De applicatie ‘MusicPlaylist’ laat gebruikers toe om afspeellijsten te maken van hun favoriete liedjes. In de applicatie zijn de volgende tabbladen voorzien: ‘Play music’, ‘Make playlist’, ‘Add song’, ‘User’ en ‘Documentation’. ‘Play music’ toont de afspeellijsten waarop men kan klikken om de liedjes te tonen. In ‘Make playlist’ kan een afspeellijst gemaakt worden. Dan in de tab ‘Add song’ worden liedjes toegevoegd aan een afspeellijst d.m.v. de YouTube-code die gevonden wordt in de YouTube video URL. ‘User’ laat toe om een account te maken in de applicatie en om een API token te verkrijgen. Ten slotte toont ‘Documentation’ de documentatie van de applicatie en zijn REST API.

Extra:

* De documentatie is beschikbaar in ‘./resources/views/documentation.blade.php’ (bijvoorbeeld openen met Mozilla Firefox) en in de webapp.
* Bij het lokaal uitvoeren van de applicatie is het belangrijk om de .env file correct in te stellen (bv. juiste db kiezen). Mijn huidige webapp kan ik lokaal uitvoeren met ‘php artisan serve --env=production’.

**Sterke punten:**

* Laravel webapp wordt gehost als een ‘Azure App Service’.
* Laravel gebruikt een MySQL database ‘musicplaylistdb’ met hierin 2 tabellen in een 1-N relationship: playlists en songs. De database wordt gehost op ‘Azure Database for MySQL-server’.
* De webapp gebruikt ‘Bootstrap CSS’ voor de layout.
* Javascript zorgt voor interactiviteit voor de gebruiker:
  + Met de navigation bar wordt bepaald welke HTML code afkomstig van Laravel wordt ingeladen (geen paginaverversing).
  + De afspeellijsten en liedjes worden dynamisch geladen m.b.v. de ‘fetch API’.
  + Om liedjes af te spelen wordt de ‘YouTube Player API‘ (zie <https://developers.google.com/youtube/iframe_api_reference>) toegepast.
  + De formulieren worden door Javascript afgehandeld.
* In Laravel roep ik de volgende web services op (zie documentatie van de Laravel webapp en documentatie van de web services voor meer informatie):
  + YouTube Data REST API (zie <https://developers.google.com/youtube/v3/getting-started>).
  + Eigen Python Flask REST web service welke op de achtergrond de Spotify web REST API oproept (zie [https://developer.spotify.com/documentation/ web-api/](https://developer.spotify.com/documentation/%20web-api/)). Deze laat toe om a.d.h.v. een artiest en liedtitel extra informatie op te vragen (bv. album naam, publicatiedatum …).
  + Eigen Node.js REST web service die een datum ‘jaar-maand-dag’ splitst naar het jaar, de maand (inclusief de naam) en de dag.
  + Eigen C# SOAP web services:
    - YouTubeSplitterService.asmx: Splitst een YouTube muziekvideotitel van formaat ‘artiest – titel’ naar de artiest en de titel.
    - DateSplitterService.asmx: Deze biedt dezelfde functionaliteit als de bovenstaande Node.js web service en wordt niet langer meer gebruikt.
  + Extra informatie:
    - De Flask en Node.js web services worden gehost als Docker containers in ‘Azure Containers’, en de SOAP web services als een ‘Azure App Service’.
    - De Flask web service houdt ook data bij in een MySQL database ‘flaskspotifydb’ die gehost wordt op dezelfde ‘Azure Database for MySQL-server’ als voor Laravel.
* De Laravel webapp biedt 2 REST web services aan:
  + De afspeellijsten met liedjes opvragen. Deze is beveiligd met een ‘authentication token’ (a.d.h.v. ‘Laravel Passport’) die eerst aangevraagd moet worden aan de webapp vooraleer de service opgeroepen kan worden.
  + De songtekst van een liedje opvragen. Intern wordt hier een SOAP web service opgeroepen van ‘<http://www.chartlyrics.com/api.aspx>’ om songteksten te zoeken.
* De Node.js web service Docker container wordt ook op de Marcel server geplaatst.