Academie voor Engineering en ICT
Opleiding Informatica

Tentamenopdracht Programmeren 4

Juni 2017

Inhoudsopgave

Inleiding

Tijdens de module Programmeren 4 heb je een webserver gemaakt met nodejs, en je hebt een Android app gemaakt die van deze server gebruik maakte. Je hebt ook gewerkt met versiebeheer in GIT, met deployment naar Heroku, en met het uitvoeren van testen.

De toetsing van Programmeren 4 bestaat uit het maken van een server- en een client-applicatie die aansluit bij de onderwerpen die in de lessen zijn behandeld. Dit document beschrijft die opdracht.

Je maakt deze opdracht in een tweetal, dus met een medestudent. Je kiest zelf met wie je deze opdracht maakt.

Inleveren

Inleveren van je uitwerking kan alleen tijdens het ingeroosterde tentamenmoment voor Programmeren 4. Dat moment is ingeroosterd op dinsdag 20 juni 2017, 9:35 – 13:00u. Bij het inleveren wordt je aanwezigheid geregistreerd. Tijdens het tentamenmoment kun je nog werken aan je opdracht; verstandiger is om de opdracht van tevoren al in detail uit te werken.

Inleveren doe je via de inleverlink op Blackboard, die tijdens het tentamenmoment beschikbaar is.

Beoordeling

De beoordeling van jullie uitwerking vindt plaats tijdens een mondeling assessment dat je per tweetal met de docent hebt. In dat gesprek lichten jullie de uitwerking toe, en beantwoorden jullie vragen van de docent. Aan het eind van dat gesprek krijg je te horen wat je cijfer is. Je beoordeling is individueel.

Het mondeling assessment wordt gehouden op woensdag 21 juni 2017. Per tweetal schrijf je je in voor één van de beschikbare momenten. Voor het inschrijven vind je op Blackboard bij het materiaal van Programmeren 4 een link naar een document.

De volgende tekst geeft een beknopt beeld van de opdracht. De precieze eisen waaraan alle onderdelen van de opdracht moeten voldoen worden beschreven in de rest van dit document.

De opdracht

Maak per tweetal een nodejs server en een Android app rond de filmdatabase die bij dit tentamen wordt meegeleverd. De filmdatabase is een verkleinde versie van de Sakila database die je al kent.

De server moet een API bieden die de gegevens in de database beschikbaar maakt voor de app. Die gegevens komen uit de database die je online installeert. De API is beveiligd voor ongeoorloofde toegang via JavaScript Web Tokens (JWT). De server biedt een login- en een registreer-endpoint zodat gebruikers zich aan kunnen melden en een geldig token ontvangen.

De Android app biedt functionaliteit om informatie over films op een Android smartphone te kunnen bekijken, en om een film te kunnen 'lenen' en een lening te kunnen opheffen. In de app moet je ingelogd zijn om films te kunnen lenen. Een uitgeleend exemplaar van een film moet eerst vrijgegeven worden (er is dan geen actieve uitlening meer op dat exemplaar) Als de gebruiker nog geen account heeft moet hij zich kunnen registreren.

Om dit project samen te kunnen uitvoeren gebruiken jullie Git als versie managementsysteem. Jullie gebruiken GitHub en werken gezamenlijk aan alle code – zowel aan de server als de Android app. Jullie samenwerking moet duidelijk worden aan de hand van de historie van jullie *push*- en *pull*-activiteiten op jullie gedeelde repository.

Om de kwaliteit van jullie werk te verhogen maken jullie tests voor de server. Optioneel: laat de resultaten van de tests zien op de Travis CI website via https://travis-ci.org. Optioneel monitor je ook de kwaliteit van de code met behulp van SonarQube. De resultaten daarvan zijn dan inzichtelijk via de website https://sonarcloud.io.

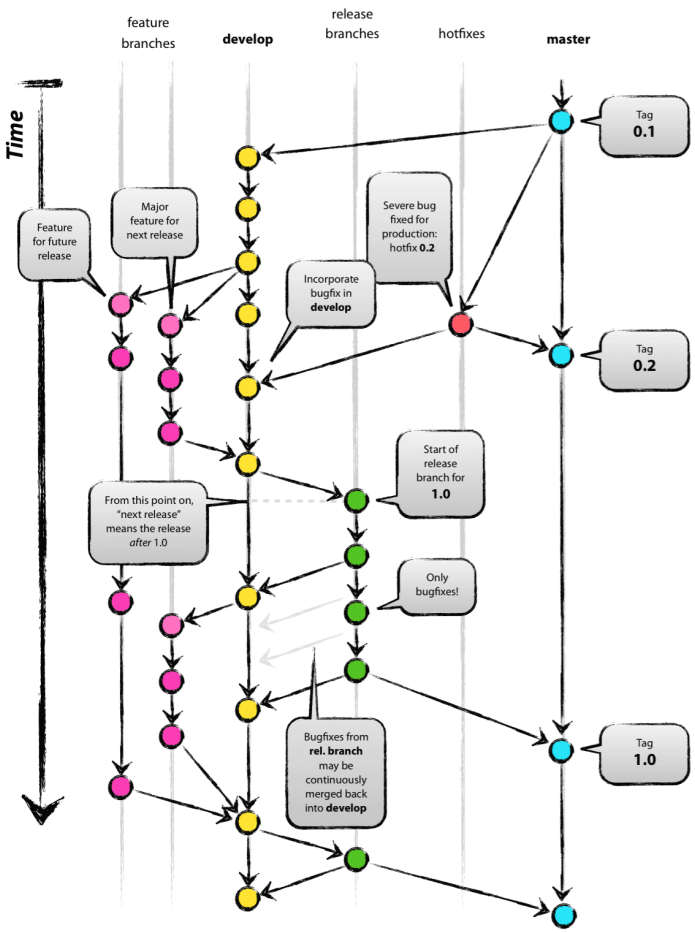
De volgende onderdelen worden beoordeeld.

Ontwikkelproces (20 punten)

De toepassing van een goed ontwikkelproces is een verplichte basisvoorwaarde.

- Jullie gebruiken Git als versie managementsysteem om samen te werken aan de code. Jullie passen de Git Workflow toe zoals die besproken in de les. Concreet:
  - Er is een master branch die altijd de actuele, geteste en deploybare code bevat.
    Je commit nooit direct code naar de master branch! Code in de master branch wordt gemerged uit de develop branch.
  - Er is een develop branch. Hierin komt de code die jullie ontwikkelen bij elkaar. De code in de develop branch is bij voorkeur afkomstig uit feature branches.
  - Feature branches zijn gewenst. In een feature branch werk je samen aan een onderdeel van de applicatie (zowel server als Android app).
  - Een release branch is in jullie geval optioneel; je mag direct mergen van develop naar master.

  Figuur 1 geeft een mogelijke workflow weer. In jullie uitwerking hoef je geen hotfixes of tags te gebruiken. Informatie over de Git Workflows die we in de les hebben behandeld vind je op http://nvie.com/posts/a-successful-git-branching-model/.



FIGUUR 1 BRON: HTTP://NVIE.COM/POSTS/A-SUCCESSFUL-GIT-BRANCHING-MODEL/

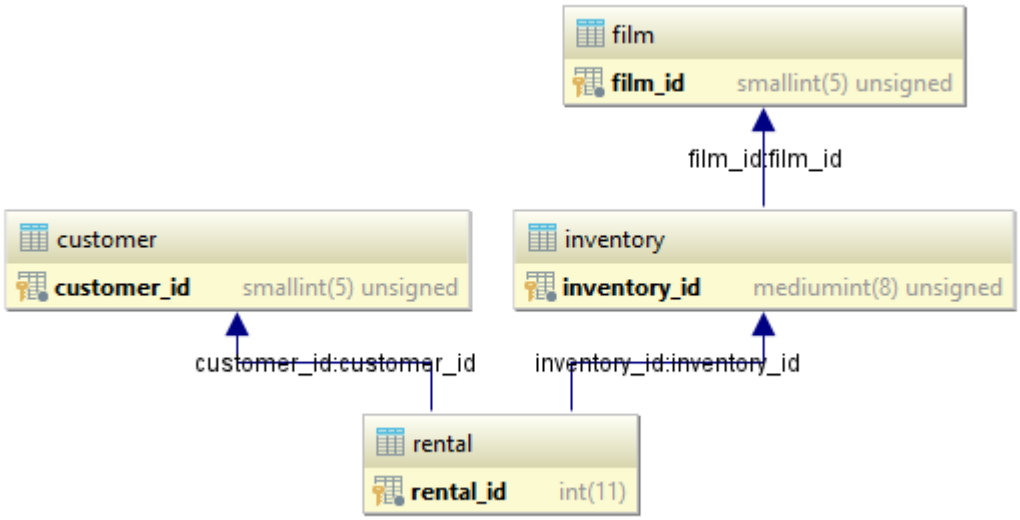
- Het is belangrijk dat jullie gezamenlijk werken aan zowel de server als aan de Android app. Dat betekent dat jullie beide programmacode schrijven, in de repository committen en naar de online repository pushen. Dit proces moet inzichtelijk zijn in de historie van jullie Git repositories.

- Jullie gebruiken een database die online beschikbaar is. In de lessen is gewerkt met een online MySql database. Informatie daarover vind je op Blackboard bij het onderwijsmateriaal van les 6 van Programmeren 4. In deze database importeer je het script met de film-informatie dat bij deze opdracht is meegeleverd.

Database

De database bestaat uit gegevens uit de Sakila database, waarbij we ons beperken tot informatie met betrekking tot verhuur van films. Onze database bevat alleen de film, inventory, rental en customer tabellen. De overige tabellen en views zijn uit het database-script verwijderd. Er is wel een view toegevoegd, 'view_rental', die een JOIN biedt op de vier tabellen, en een aantal geselecteerde kolommen laat zien. Wanneer dat nodig is voor jouw uitwerking mag je de view 'view_rental' aanpassen. Je vindt het script voor de database bij de opdracht op Blackboard.

Het volgende diagram geeft een overzicht van de relaties en tabellen in de database. Bij de opdracht op Blackboard is ook een gedetailleerder diagram beschikbaar. Daarin zie je ook de kolomnamen en datatypen.



De tabel *customer* bevat informatie over klanten die een film huren. Je mag deze tabel gebruiken om gebruikers in te loggen. Je hoeft dus niet een nieuwe tabel gebruiker of user te maken.

De tabel *inventory* geeft aan hoeveel exemplaren van een film beschikbaar zijn voor verhuur.

Server (40 punten)

De server voldoet aan de volgende voorwaarden:

- De server is geschreven in Nodejs en JavaScript, en biedt via zijn API toegang tot de filminformatie in de MySql database.
- De server is beschikbaar via een URL op Heroku.
    - De server wordt automatisch gedeployed wanneer alle servertesten slagen.
- De API is beveiligd tegen ongeoorloofde toegang met behulp van JavaScript Web Tokens (JWT).
- De server biedt ten minste de API endpoints in de volgende tabel.
    - Het kan zijn dat je voor jouw systeem meer enpoints nodig hebt. Die mag je zelf toevoegen.

- Het kan zijn dat je gegevens mee moet sturen in een request. Je bepaalt zelf welke gegevens dat zijn.
- Endpoints die niet door jouw server afgehandeld worden sturen een foutbericht met een duidelijke toelichting terug.

	API endpoint	JWT	Toelichting
POST	/api/v1/login	Nee	Geeft een geldig token wanneer de gebruiker (customer) is ingelogd; anders een duidelijke foutmelding.
POST	/api/v1/register	Nee	Maakt een nieuwe gebruiker (customer) aan de hand van de meegestuurde gebruikersgegevens. Als de gebruiker al bestaat stuurt het systeem een duidelijke foutmelding.
GET	/api/v1/films?offset=:start&count=:number	Nee	Geeft alle informatie van de gevraagde films. Offset en count kunnen als opties worden gegeven. Offset is het startpunt; count is het aantal films vanaf de offset. Voorbeeld: /api/v1/films?offset=50&count=20 retourneert 20 films vanaf index 50.
GET	/api/v1/films/:filmid	Nee	Geeft alle informatie van de film met het gegeven filmid, inclusief alle verhuur informatie
GET	/api/v1/rentals/:userid	Ja	Geeft alle uitgeleende films van de gebruiker met het gegeven userid.
POST	/api/v1/ rentals/:userid/:inventoryid	Ja	Maakt een nieuwe uitlening voor de gegeven gebruiker van het exemplaar met gegeven inventoryid.
PUT	/api/v1/ rentals/:userid/:inventoryid	Ja	Wijzig bestaande uitlening voor de gegeven gebruiker van het exemplaar met gegeven inventoryid.
DELETE	/api/v1/ rentals/:userid/:inventoryid	Ja	Verwijder bestaande uitlening voor de gegeven gebruiker van het exemplaar met gegeven inventoryid.

- Alle endpoints hebben bijbehorende testcases, die zowel de succesvolle (het lukt b.v. om in te loggen) als de falende (inloggen mislukt omdat b.v. password incorrect is) gebruik testen.
  - De testen worden gepubliceerd op Travis CI (https://travis-ci.org).

Android app (40 punten)

De Android app voldoet aan de volgende voorwaarden:

- De gebruiker heeft pas toegang tot de film- en verhuurinformatie via de app nadat hij succesvol ingelogd is.
  - Wanneer de gebruiker is ingelogd heeft de app een JWT token. Dat token wordt binnen de app bewaard.
  - Een ingelogde gebruiker kan uitloggen. Het JWT token wordt daarmee ongeldig.
- Een nieuwe gebruiker kan zichzelf registreren. Hiervoor is een gebruikersnaam en een wachtwoord nodig. Als een gebruiker geregistreerd is kan deze inloggen.
- De app geeft een overzicht van beschikbare films en exemplaren.
  - Bij een exemplaar is zichtbaar of deze uitgeleend of beschikbaar is.
- De app biedt de mogelijkheid om een exemplaar van een film te huren.
  - Een gehuurd exemplaar kan niet door een andere gebruiker gehuurd worden.

- De app biedt de mogelijkheid om een gehuurde film 'in te leveren'.
    - De verhuurgegevens blijven bewaard in de database, maar het exemplaar is daarna niet meer gehuurd.

Bij het maken van de app mag je naar keuze gebruik maken van Volley, of van http requests via de Async methode.

Hier eindigt de opdracht.