Jeroen Smienk

Evert Duipmans

Webtechnologie

8 oktober 2016

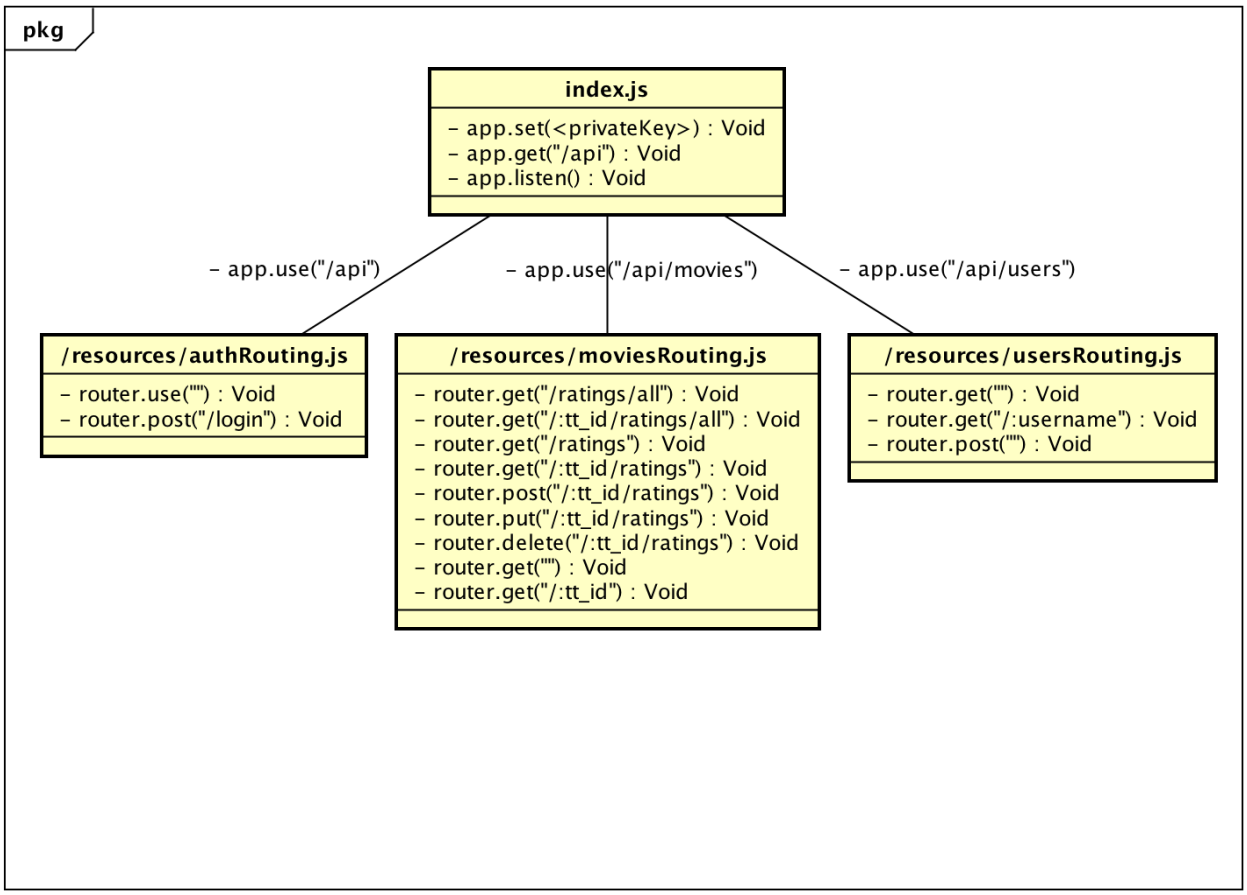
REST Specificate Notflix

Applicatie

De applicatie maakt gebruik van Express, Body-Parser, Mongoose, Blue Imp-MD5 en JSON Web Token.

Om te beginnen wordt een geheime sleutel gegenereerd om te gebruiken voor de JWT en opgeslagen in het applicatie geheugen. Hierna wordt geprobeerd om verbinding te maken met de lokale MongoDB database. De routing wordt gedefinieerd voor verschillende onderwerpen als authenticatie, gebruikers en films. De routing onder deze onderwerpen zijn ondergebracht in verschillende JavaScript bestanden. Nu begint de server met luisteren.

Wachtwoorden worden gehashd voordat ze opgeslagen worden in de database. Dit wordt gedaan wanneer een nieuwe gebruiker aangemaakt wordt. Er wordt een willekeurige salt gemaakt, waarmee het wachtwoord gehashd wordt. Het salt wordt ook opgeslagen bij de gebruiker. Zo heeft elke gebruiker een gehashd wachtwoord met hun eigen willekeurige salt.



Datamodel

Alle gegevens worden opgeslagen in twee documenten. Namelijk gebruikers en films, respectievelijk gedefinieerd in */model/user.js* en */model/movie.js*.

GEBRUIKER

sleutel	type	verplicht	uniek	beschrijving
last_name	String	ja	nee	Achternaam van de gebruiker.
infix	String	nee	nee	Tussenvoegsel(s) van de gebruiker.
first_name	String	ja	nee	Voornaam van de gebruiker.
username	String	ja	ja	Gebruikersnaam van de gebruiker.
password_hash	String	ja	nee	Gehashd wachtwoord van de gebruiker.
password_salt	String	ja	nee	Hash salt behorend tot het huidige wachtoord.

FILM

sleutel	type	verplicht	uniek	beschrijving
tt_id	String	ja	ja	Uniek IMDB nummer van de film.
title	String	ja	nee	Titel van de film.
publish_date	Date	ja	nee	Verschijningsdatum van de film.
length	Number	nee	nee	Lengte van de film in minuten.
director	String	nee	nee	Regisseur van de film.
description	String	nee	nee	Beschrijving van inhoud van de film.
ratings	Array	nee	nee	Verzameling objecten met een gebruikersnaam en waardering van één tot en met tien punten.
average_rating	Number	nee	nee	Gemiddelde waardering van de film.

RATING

sleutel	type	verplicht	uniek	beschrijving
username	String	ja	ja	Gebruikersnaam van de gebruiker.
points	Number	ja	nee	Waardering van de gebruiker van één tot en met tien punten.

Authenticatie

Bijna alle verzoeken naar de REST API dienen voorzien te zijn van authenticatie. Dit wordt gedaan in de vorm van een JSON Web Token in de header onder de naam AuthToken.

Paden die niet voorzien hoeven worden van een authenticatie token, zijn gedefinieerd in de array genaamd *excluded*. Hierin staan objecten met een url, methode en eventueel een boolean. De url is het pad dat uitgezonderd wordt van authenticatie (in combinatie met de bijbehorende methode). Met de boolean kan ervoor gekozen worden om een url alleen tot aan de laatste schuine streep na te kijken. Dit is gedaan om variabele paden, bijvoorbeeld die een tt_id of gebruikersnaam bevatten, te kunnen nakijken. Hierbij hoort de array *includedBackStrings* waarin Strings zitten die wel nagekeken moeten worden, ondanks dat zij achter een pad komen waarbij alleen de basis nagekeken wordt.

Alle verzoeken worden voorgegaan door de authenticatie router. Deze bekijkt of authenticatie nodig is. Zo ja, wordt de inhoud van de *AuthToken* header geverifieerd met behulp van *jwt.verify()*. Als een error opduikt wordt er een 401 teruggestuurd.

Een authenticatie token wordt uitgegeven nadat er succesvol ingelogd is met een combinatie van gebruikersnaam en wachtwoord.

POST /api/login

Een gebruiker authenticeren om een authenticatie token te verkrijgen.

VERZOEK

header	body	query parameters
	username	
	password	

ANTWOORD

status code	body	beschrijving
201	{ token: <AuthToken> }	Authenticatie succesvol.
400	{ error: “Invalid credentials.” }	Gebruikersnaam of wachtwoord niet meegegeven.
401	{ error: “Invalid credentials.” }	Gebruikersnaam niet bekend.
401	{ error: “Invalid credentials.” }	Wachtwoord komt niet overeen.
500	{ error: <MongoError> }	Interne fout opgetreden.

Films en waarderingen

Films kunnen worden opgevraagd zonder authenticatie. Alles te doen met waarderingen vereist echter wel authenticatie.

GET /api/movies

Een lijst met alle films ophalen zonder waarderingen.

Ondersteunt limitering. Met de query parameter limit kan het aantal resultaten bepaald worden en met page de pagina. Bijvoorbeeld de eerste vijf met *?limit=5* of de tweede vijf met *?limit=5&page=1*.

VERZOEK

header	body	query parameters
		limit
		page

ANTWOORD

status code	body	beschrijving
200	{ [<film>] }	Alle films als objecten in een Array succesvol opgehaald.
500	{ error: <MongoError> }	Interne fout opgetreden.

GET /api/movies/:tt_id

Eén film ophalen zonder waarderingen.

VERZOEK

header	body	query parameters

ANTWOORD

status code	body	beschrijving
200	{ <film> }	Eén film als object succesvol opgehaald.
404	{ error: “Movie not found.” }	De ingevulde tt_id heeft geen bijbehorende film.
500	{ error: <MongoError> }	Interne fout opgetreden.

GET /api/movies/ratings/all

Een lijst met alle films ophalen die gewaardeerd zijn.

Ondersteunt limitering. Met de query parameter limit kan het aantal resultaten bepaald worden en met page de pagina. Bijvoorbeeld de eerste vijf met *?limit=5* of de tweede vijf met *?limit=5&page=1*.

VERZOEK

header	body	query parameters
AuthToken		limit
		page

ANTWOORD

status code	body	beschrijving
200	{ [<film>] }	Alle films als objecten in een Array succesvol opgehaald.
401	{ error: <JWTError> }	Niet geauthenticeerd verzoek.
500	{ error: <MongoError> }	Interne fout opgetreden.

GET /api/movies/:tt_id/ratings/all

Eén film ophalen die gewaardeerd is.

VERZOEK

header	body	query parameters
AuthToken		

ANTWOORD

status code	body	beschrijving
200	{ <film> }	Eén film als object succesvol opgehaald.
401	{ error: <JWTError> }	Niet geauthenticeerd verzoek.
404	{ error: “Movie not found/yet rated.” }	De ingevulde tt_id heeft geen bijbehorende film of de bijbehorende film is nog niet gewaardeerd.
500	{ error: <MongoError> }	Interne fout opgetreden.

GET /api/movies/ratings

Een lijst met alle films ophalen die gewaardeerd zijn door de geauthenticeerde gebruiker.

Ondersteunt limitering. Met de query parameter limit kan het aantal resultaten bepaald worden en met page de pagina. Bijvoorbeeld de eerste vijf met *?limit=5* of de tweede vijf met *?limit=5&page=1*.

VERZOEK

header	body	query parameters
AuthToken		limit
		page

ANTWOORD

status code	body	beschrijving
200	{ [<film>] }	Alle films als objecten in een Array succesvol opgehaald.
401	{ error: <JWTError> }	Niet geauthenticeerd verzoek.
500	{ error: <MongoError> }	Interne fout opgetreden.

GET /api/movies/:tt_id/ratings

Eén film ophalen die gewaardeerd is door de geauthenticeerde gebruiker.

VERZOEK

header	body	query parameters
AuthToken		

ANTWOORD

status code	body	beschrijving
200	{ <film> }	Eén film als object succesvol opgehaald.
401	{ error: <JWTError> }	Niet geauthenticeerd verzoek.
404	{ error: “Movie not found/yet rated.” }	De ingevulde tt_id heeft geen bijbehorende film of de bijbehorende film is nog niet gewaardeerd door de geauthenticeerde gebruiker.
500	{ error: <MongoError> }	Interne fout opgetreden.

POST /api/movies/:tt_id/ratings

Een waardering plaatsen van de geauthenticeerde gebruiker bij één film.

VERZOEK

header	body	query parameters
AuthToken	points	

ANTWOORD

status code	body	beschrijving
201	{ username: <gebruikersnaam>, points: <waardering> }	Een waardering is succesvol geplaatst.
400	{ error: “Invalid body content.” }	Waardering niet meegegeven.
400	{ error: “Amount of points is unvalid.” }	Waardering is kleiner dan één of groter dan tien.
401	{ error: <JWTError> }	Niet geauthenticeerd verzoek.
404	{ error: “Movie not found.” }	De ingevulde tt_id heeft geen bijbehorende film.
409	{ error: "Movie already rated." }	De film is al gewaardeerd door de geauthenticeerde gebruiker.
500	{ error: <MongoError> }	Interne fout opgetreden.

PUT /api/movies/:tt_id/ratings

Een waardering wijzigen van de geauthenticeerde gebruiker bij één film.

VERZOEK

header	body	query parameters
AuthToken	points	

ANTWOORD

status code	body	beschrijving
201	{ username: <gebruikersnaam>, points: <waardering> }	Een waardering is succesvol gewijzigd.
400	{ error: “Invalid body content.” }	Waardering niet meegegeven.
400	{ error: “Amount of points is unvalid.” }	Waardering is kleiner dan één of groter dan tien.
401	{ error: <JWTError> }	Niet geauthenticeerd verzoek.
404	{ error: “Movie not found/yet rated.” }	De ingevulde tt_id heeft geen bijbehorende film of de bijbehorende film is nog niet gewaardeerd door de geauthenticeerde gebruiker.
500	{ error: <MongoError> }	Interne fout opgetreden.

DELETE /api/movies/:tt_id/ratings

Een waardering verwijderen van de geauthenticeerde gebruiker bij één film.

VERZOEK

header	body	query parameters
AuthToken		

ANTWOORD

status code	body	beschrijving
204		Een waardering is succesvol verwijderd.
401	{ error: <JWTError> }	Niet geauthenticeerd verzoek.
404	{ error: “Movie not found/yet rated.” }	De ingevulde tt_id heeft geen bijbehorende film of de bijbehorende film is nog niet gewaardeerd door de geauthenticeerde gebruiker.
500	{ error: <MongoError> }	Interne fout opgetreden.

Gebruikers

Alle verzoeken, op één na, die te maken hebben met gebruikers vereisen authenticatie. Alleen een nieuwe gebruiker aanmaken vereist geen authenticatie.

GET /api/users

Een lijst met alle gebruikers ophalen zonder wachtwoord gerelateerde velden.

Ondersteunt limitering. Met de query parameter limit kan het aantal resultaten bepaald worden en met page de pagina. Bijvoorbeeld de eerste vijf met *?limit=5* of de tweede vijf met *?limit=5&page=1*.

VERZOEK

header	body	query parameters
AuthToken		limit
		page

ANTWOORD

status code	body	beschrijving
200	{ [<gebruiker>] }	Alle gebruikers als objecten in een Array succesvol opgehaald.
401	{ error: <JWTError> }	Niet geauthenticeerd verzoek.
500	{ error: <MongoError> }	Interne fout opgetreden.

GET /api/users/:username

Eén gebruiker ophalen zonder wachtwoord gerelateerde velden.

VERZOEK

header	body	query parameters
AuthToken		

ANTWOORD

status code	body	beschrijving
200	{ <gebruiker> }	Eén gebruiker als object succesvol opgehaald.
401	{ error: <JWTError> }	Niet geauthenticeerd verzoek.
404	{ error: “User not found.” }	De ingevulde gebruikersnaam heeft geen bijbehorende gebruiker.
500	{ error: <MongoError> }	Interne fout opgetreden.

POST /api/users

Een nieuwe gebruiker aanmaken.

VERZOEK

header	body	query parameters
	last_name	
	infix *(optioneel)*	
	first_name	
	username	
	password	

ANTWOORD

status code	body	beschrijving
201	{ last_name: <achternaam>, infix: <tussenvoegsels>, first_name: <voornaam>. username: <gebruikersnaam> }	Een gebruiker is succesvol aangemaakt.
400	{ error: “Invalid body content.” }	Eén of meer verplichte velden niet meegegeven.
409	{ error: “Username occupied." }	Er bestaat al een gebruiker met de meegegeven gebruikersnaam.
500	{ error: <MongoError> }	Interne fout opgetreden.

Testverslag

Met behulp van Mocha, ShouldJS en Supertest is de REST API uitvoerig getest.

Bij alle tests wordt een antwoord verwacht met *Content-Type: application/json* in de header. Dit zal ik dan ook niet overal vermelden. Wel wordt de verwachte status code vermeldt en het verwachte JSON object.

Authenticatie (/tests/authTests.js)

GOOD WEATHER

verzoek	invoer	uitvoer	beschrijving
POST /api/login	{ username: jsmienk, password: jj}	201 { token: <AuthToken> }	Inloggen met juiste gebruikersnaam en wachtwoord combinatie.

BAD WEATHER

verzoek	invoer	uitvoer	beschrijving
POST /api/login	{ username: jsmienk, password: jr}	401 { error: <error> }	Inloggen met onjuiste gebruikersnaam en wachtwoord combinatie.
POST /api/login	{ username: kk, password: jj}	401 { error: <error> }	Inloggen met onbekende gebruikersnaam.
/POST api/login	{ password: jj }	400 { error: <error> }	Inloggen zonder gebruikersnaam mee te geven.
POST /api/login	{ username: jsmienk }	400 { error: <error> }	Inloggen zonder wachtwoord mee te geven.
POST /api/login	{ }	400 { error: <error> }	Inloggen zonder gebruikersnaam en wachtwoord mee te geven.

Films (/tests/moviesTests.js)

De database waarmee ik test bevat zeven films.

GOOD WEATHER

verzoek	invoer	uitvoer	beschrijving
GET /api/movies		200 { [<film>] } result.length == 7	Een Array met 7 film objecten succesvol opgehaald.
GET /api/movies?limit=5		200 { [<film>] } result.length == 5	Een Array met 5 film objecten succesvol opgehaald met behulp van limitering.
GET /api/movies?limit=5&page=1		200 { [<film>] } result.length == 2	Een Array met 2 film objecten succesvol opgehaald met behulp van limitering.
GET /api/movies?limit=-1&page=-4		200 { [<film>] } result.length == 7	Alle films opgehaald vanwege negatieve invoer bij limitering.
GET /api/movies/0120338		200 { <film> } tt_id == “0120338”	Eén film object succesvol opgehaald.

BAD WEATHER

verzoek	invoer	uitvoer	beschrijving
GET /api/movies/222222		404 { error: <error> }	De film kan niet gevonden worden.

Gebruikers (/tests/usersTests.js)

De database waarmee ik test bevat vijf gebruikers.

Voordat de tests beginnen wordt een authenticatie token opgevraagd en opgeslagen om te gebruiken bij enkele tests.

Na alle tests wordt de *after()* aangeroepen. Hierin worden de aangemaakte gebruikers “evert” en “wouter” weer verwijderd. Dit werkt alleen om een of andere reden niet altijd. Als we de test vaker draaien, slagen sommige testen wel en sommige niet. Afhankelijk van het feit of bij de vorige test de ‘nieuwe’ gebruikers succesvol verwijderd zijn.

GOOD WEATHER

verzoek	invoer	uitvoer	beschrijving
GET /api/users		200 { [<gebruiker>] } result.length == 5	Een Array met 7 gebruiker objecten succesvol opgehaald.
GET /api/users?limit=3		200 { [<gebruiker>] } result.length == 3	Een Array met 5 gebruiker objecten succesvol opgehaald met behulp van limitering.
GET /api/users? limit=3&page=2		200 { [<gebruiker>] } result.length == 0	Een Array met 2 gebruiker objecten succesvol opgehaald met behulp van limitering.
GET /api/users? limit=-1&page=-4		200 { [<gebruiker>] } result.length == 5	Alle gebruikers opgehaald vanwege negatieve invoer bij limitering.
GET /api/users/jj		200 { [<gebruiker>] } username == “jj”	Eén gebruiker met de juiste gebruikersnaam is succesvol opgehaald.
POST /api/users	{ username: evert, password: evert, first_name: eVeRT, infix: , last_name: DUIpmans }	201 { <gebruiker> } Zelfde/gecorrigeerde waarden in velden.	Een gebruiker succesvol aangemaakt met de voornaam, tussenvoegsels en achternaam gecorrigeerd wat betreft hoofdletters.
POST /api/users	{ username: wouter, password: wouter, first_name: WOuter, last_name: geenIdee }	201 { <gebruiker> } Zelfde/gecorrigeerde waarden in velden.	Een gebruiker succesvol aangemaakt zonder tussenvoegsels mee te geven met de voornaam en achternaam gecorrigeerd wat betreft hoofdletters.

BAD WEATHER

verzoek	invoer	uitvoer	beschrijving
GET /api/users		401 { error: <error> }	Geen AuthToken in de header meegestuurd.
GET /api/users/jj		401 { error: <error> }	Geen AuthToken in de header meegestuurd.

verzoek	invoer	uitvoer	beschrijving
GET /api/users/jk		404 { error: <error> }	De gebruiker kan niet gevonden worden.
POST /api/users	{ password: evert, first_name: eVeRT, last_name: DUlpmans }	400 { error: <error> }	Een gebruiker proberen aan te maken zonder gebruikersnaam mee te geven.
POST /api/users	{ username: evert, first_name: eVeRT, last_name: DUlpmans }	400 { error: <error> }	Een gebruiker proberen aan te maken zonder wachtwoord mee te geven.
POST /api/users	{ username: evert, password: evert, last_name: DUlpmans }	400 { error: <error> }	Een gebruiker proberen aan te maken zonder voornaam mee te geven.
POST /api/users	{ username: evert, first_name: eVeRT, password: evert }	400 { error: <error> }	Een gebruiker proberen aan te maken zonder achternaam mee te geven.
POST /api/users	{ username: evert, password: evert, first_name: eVeRT, infix: , last_name: DUlpmans }	409 { error: <error> }	Een gebruiker proberen aan te maken met een gebruikersnaam die al in bezit is van een andere gebruiker.

Waarderingen (/tests/ratingsTests.js)

De database waarmee ik test bevat zeven films waarvan vier met waarderingen.

Voordat de tests beginnen wordt een authenticatie token opgevraagd en opgeslagen om te gebruiken bij enkele tests.

GOOD WEATHER

verzoek	invoer	uitvoer	beschrijving
GET /api/movies/ratings/all		200 { [<film>] } result.length == 4	Een Array met 4 film objecten en gemiddelde waardering velden succesvol opgehaald.
GET /api/movies/ratings/all?limit=3		200 { [<film>] } result.length == 3	Een Array met 3 film objecten en gemiddelde waardering velden succesvol opgehaald.
GET /api/movies/ratings/all?limit=3&page=1		200 { [<film>] } result.length == 1	Een Array met 1 film object en gemiddelde waardering velden succesvol opgehaald.
GET /api/movies/ratings/all?limit=-2&page=-1		200 { [<film>] } result.length == 4	Een Array met 4 film objecten en gemiddelde waardering velden succesvol opgehaald.
GET /api/movies/234487/ratings/all		200 { <film> } tt_id == “234487” average_rating == 5	Eén film object met het gemiddelde waardering veld succesvol opgehaald.
GET /api/movies/ratings		200 { [<film>] } result.length == 3	Een Array met 3 film objecten en waardering veld succesvol opgehaald.
GET /api/movies/ratings?limit=2		200 { [<film>] } result.length == 2	Een Array met 2 film objecten en waardering veld succesvol opgehaald.
GET /api/movies/ratings?limit=2&page=1		200 { [<film>] } result.length == 1	Een Array met 1 film objecten en waardering veld succesvol opgehaald.
GET /api/movies/ratings?limit=-2&page=-1		200 { [<film>] } result.length == 3	Een Array met 3 film objecten en waardering veld succesvol opgehaald.
GET /api/movies/0345666/ratings		200 { <film> } tt_id == “0345666” rating == 2	Eén film object met de waardering van de geauthenticeerde gebruiker succesvol opgehaald.
POST /api/movies/0114369/ratings	{ points: 8 }	201 { username: jsmienk, points: 8 }	Eén waardering bij een film zonder waarderingen succesvol geplaatst.
POST /api/movies/234487/ratings	{ points: 6 }	201 { username: jsmienk, points: 6 }	Eén waardering bij een film met andere waarderingen succesvol geplaatst.

verzoek	invoer	uitvoer	beschrijving
PUT /api/movies/0345666/ ratings	{ points: 6 }	201 { username: jsmienk, points: 6 }	Eén waardering bij een film succesvol gewijzigd.
PUT /api/movies/0345666/ ratings	{ points: 2 }	201 { username: jsmienk, points: 2 }	Eén waardering bij een film succesvol gewijzigd.
PUT /api/movies/0198940/ ratings	{ points: 10 }	201 { username: jsmienk, points: 10 }	Eén waardering bij een film met andere waarderingen succesvol gewijzigd.
DELETE /api/movies/0114369/ ratings		204 {}	Eén waardering bij een film succesvol verwijderd.
DELETE /api/movies/234487/ ratings		204 {}	Eén waardering bij een film succesvol verwijderd.

BAD WEATHER

verzoek	invoer	uitvoer	beschrijving
GET /api/movies/ratings/all		401 { error: <error> }	Geen AuthToken in de header meegestuurd.
GET /api/movies/0123456/ ratings/all		401 { error: <error> }	Geen AuthToken in de header meegestuurd.
GET /api/movies/222222/ ratings/all		404 { error: <error> }	De film kan niet gevonden worden.
GET /api/movies/9728456/ ratings/all		404 { error: <error> }	Bij de film zijn geen waarderingen gevonden.
GET /api/movies/ratings		401 { error: <error> }	Geen AuthToken in de header meegestuurd.
GET /api/movies/0123456/ ratings		401 { error: <error> }	Geen AuthToken in de header meegestuurd.
GET /api/movies/222222/ ratings		404 { error: <error> }	De film kan niet gevonden worden.
GET /api/movies/234487/ ratings		404 { error: <error> }	Bij de film is geen waardering van de geauthenticeerde gebruiker gevonden.
POST /api/movies/098711/ ratings		400 { error: <error> }	Geen punten meegestuurd.

verzoek	invoer	uitvoer	beschrijving
POST /api/movies/098711/ ratings	{ points: -3 }	400 { error: <error> }	Invalide hoeveelheid punten meegegeven.
POST /api/movies/098711/ ratings	{ points: 3 }	401 { error: <error> }	Geen AuthToken in de header meegestuurd.
POST /api/movies/222222/ ratings	{ points: 7 }	404 { error: <error> }	De film kan niet gevonden worden.
POST /api/movies/0114369 /ratings	{ points: 8 }	409 { error: <error> }	De geauthenticeerde gebruiker heeft de film al gewaardeerd.
PUT /api/movies/0123456/ ratings		400 { error: <error> }	Geen punten meegestuurd.
PUT /api/movies/0123456/ ratings	{ points: -7 }	400 { error: <error> }	Invalide hoeveelheid punten meegegeven.
PUT /api/movies/0123456/ ratings	{ points: 7 }	401 { error: <error> }	Geen AuthToken in de header meegestuurd.
PUT /api/movies/222222/ ratings	{ points: 7 }	404 { error: <error> }	De film kan niet gevonden worden.
PUT /api/movies/098711/ ratings	{ points: 7 }	404 { error: <error> }	De geauthenticeerde gebruiker heeft de film nog niet gewaardeerd.
DELETE /api/movies/9728456/ ratings		401 { error: <error> }	Geen AuthToken in de header meegestuurd.
DELETE /api/movies/222222/ ratings		404 { error: <error> }	De film kan niet gevonden worden.
DELETE /api/movies/9728456/ ratings		404 { error: <error> }	De geauthenticeerde gebruiker heeft de film nog niet gewaardeerd.