Spotitube REST API

Opleverdocumentatie

**Versie 1.0.**

Bart van der Wal – OOSE Semester | Course DEA  
Rick Jellema – 614621 – OOSE-C-s | 10-15-2020

Hogeschool Arnhem en Nijmegen

Inhoud

[Versiebeheer 2](#_Toc53695824)

[1. Inleiding. 3](#_Toc53695825)

[2. Begrippenlijst 4](#_Toc53695826)

[3. Deployement diagram 5](#_Toc53695827)

[4. Package diagram 6](#_Toc53695828)

[4.1. Ontwerpbeslissingen. 7](#_Toc53695829)

# Versiebeheer

|  |  |  |
| --- | --- | --- |
| Versie | Door | Veranderingen |
| 1.0. | Rick Jellema | Document opgesteld. Inhoudsopgave aangemaakt; Voorblad toegevoegd; Inleiding toegevoegd; Begrippenlijst toegevoegd; Hoofdstuk 3. Deployment diagram toegevoegd en afgemaakt;  Hoofdstuk 4. Package diagram en 4.1. Ontwerpkeuzes toegevoegd en afgemaakt; Bronnen toegevoegd;  Bijlages toegevoegd; |
|  |  |  |
|  |  |  |

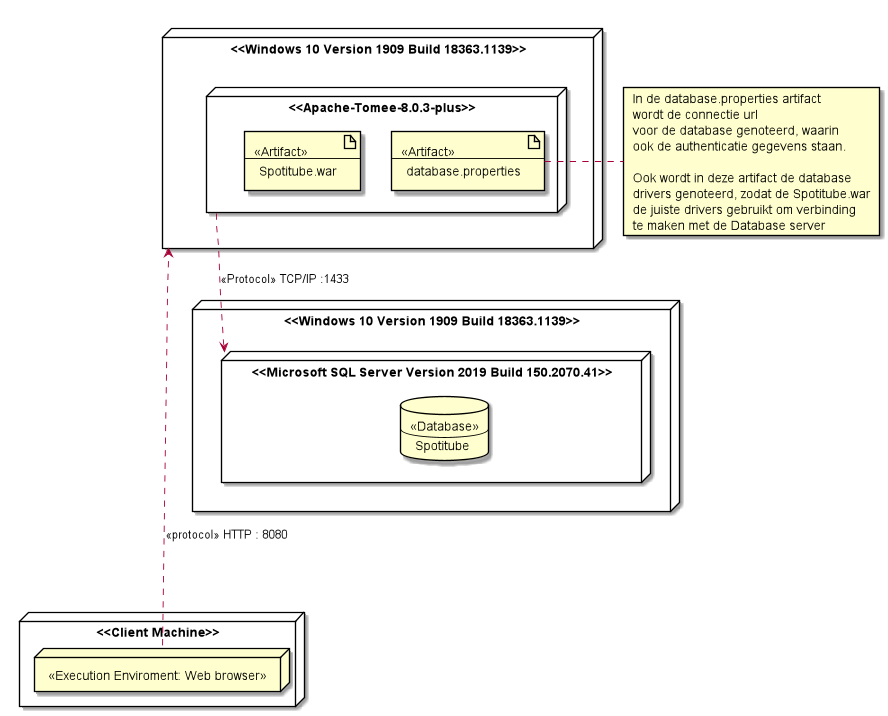
# 1. Inleiding.

Dit document is opgesteld zodat de lezer duidelijk inzage krijgt over hoe de spotitube API opgesteld is.

# 2. Begrippenlijst

|  |  |
| --- | --- |
| Woord | Betekenis |
| Client | De webapplicatie die gebruikt wordt om met de spotitube API te communiceren. |
| API | Een API (Application Programming Interface) is een software-interface die het mogelijk maakt dat twee applicaties met elkaar kunnen communiceren. In deze situatie is dit de spotitube server. |
| Artifact | Een artifact is een bestand dat gebruikt wordt in de uitvoerende omgeving (Execution Enviroment). Zoals Apache-Tomee. |
| Apache-Tomee | Apachee-tomee is software waarmee java applicaties mee gedraaid kunnen worden en makkelijk verbonden mee kan worden via het internet. |
| Singleton | Is een ontwerppatroon om het aantal objecten van de zogenoemde singleton klasse te beperken tot 1 instantie. Toegang tot informatie of het uitvoeren van methoden uit deze singleton klasse gaat altijd via de gemaakte instantie. |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

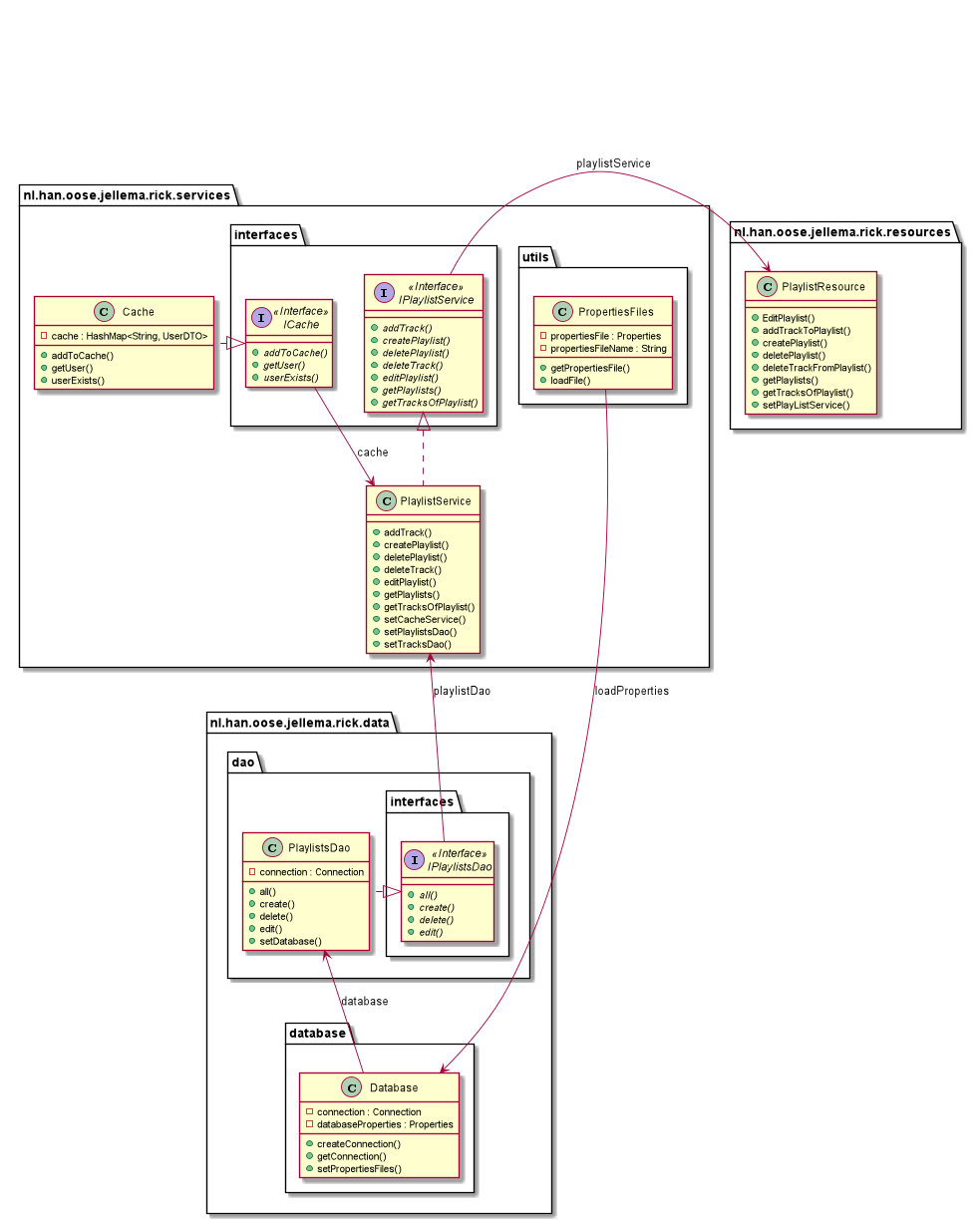
# 3. Deployment diagram

In dit hoofdstuk is het deployment diagram beschikbaar gesteld. In dit diagram is te zien hoe de client, de API en de database server met elkaar communiceren en op welke manier dit gebeurt.  
  
De client wordt niet opgeleverd in dit project omdat de client al ontwikkelt is door de Hogeschool Arnhem en Nijmegen1.  
1. de client versie is beschikbaar op de [github repository](https://github.com/HANICA-DEA/spotitube). 1A. De live is beschikbaar op [deze link](http://ci.icaprojecten.nl/spotitube/).   
 Bijlage 3A Deployment diagram van de Spotitube API.

# 4. Package diagram

In dit hoofdstuk is de package diagram van de spotitube API beschikbaar gesteld. In dit diagram is te zien hoe de spotitube API gestructureerd is in verschillende lagen namelijk:

* De resource layer;
* De service layer;
* De data access layer.

  
4A. Package diagram van de Spotitube API.

## 4.1. Ontwerpbeslissingen.

Er zijn voor de API verschillende ontwerpbeslissingen vastgelegd die beschikbaar zijn in dit hoofdstuk.

**Ontwerpbeslissing ICache/Cache.**Er is gekozen om een Interface ‘ICache’ aan te maken. Die vervolgens op verschillende manieren geïmplementeerd kan worden zodat er gekozen kan worden om verschillende items op te slaan in een ‘Singleton’ opslag systeem.  
  
De Singleton klasse ‘Cache’ wordt gebruikt om de gecreëerde tokens van de gebruikers op te slaan, zodat de gebruiker na het inloggen de verschillende endpoints kan aanroepen en hiervan gebruik kan maken in de client.

**Ontwerpbeslissingen Database:**Er is gekozen om een Database connectie te laten aanmaken via de klasse Database. Deze wordt geïnstantieerd om vervolgens de ‘createConnection()’ functie aan te roepen met als parameter de bestandsnaam van de properties bestand.   
  
Dit bestand wordt uitgelezen door de ‘PropertiesFiles’ klasse en de inhoud teruggegeven aan Database, zodat deze klasse genoeg informatie heeft om een verbinding aan te maken aan de database en deze terug te geven.

Deze verbinden worden gebruikt in de ‘connection: Connection’ variabele in de verschillende DAO’s (Data Access Objects) zoals ‘PlaylistsDAO’ om zo informatie vanuit de database op te halen.

**Algemeen ontwerpbeslissing:**Er is gekozen om de volgende patterns aan te houden voor de spotitube API. Deze patterns zijn de eisen van de opdrachtgever:  
  
**Data access layer.  
Service Layer pattern.  
Remote Facade pattern.**

Deze patterns zijn duidelijk terug te zien in het package diagram (zie afbeelding 4A), zoals de package namen:   
**- nl.han.oose.jellema.rick.services;**   
**- nl.han.oose.jellema.rick.resources;**   
**- nl.han.oose.jellema.rick.data;**

# Bijlage.

Bijlage 3A in hoofdstuk 3. [Deployment diagram](Deployement%20diagram.png).  
Bijlage 4A in hoofdstuk 4. [Package diagram](Deployement%20diagram.png).

# Bronvermelding.

Brouwer, M. & Hogeschool van Arnhem en Nijmegen. (2017, 9 juli). *Github repository spotitube client*. Github Hogeschool Arnhem en Nijmegen spotitube. <https://github.com/HANICA-DEA/spotitube>

Hogeschool Arnhem en Nijmegen. (z.d.). *Live client spotitube.* Spotitube. Geraadpleegd 28 september 2020, van <http://ci.icaprojecten.nl/spotitube/>