[](https://pbs.twimg.com/profile_images/378800000667811953/a72181bdf6dd0813116ccfe11bba95a6.png)

**Thijs Thijssen**

**560295**

**OOSE B-N feb**

**Meron Brouwer**

**07-04-2017**

**Distributed Enterprise Applications**

Contents

[1 Deployment Diagram 3](#_Toc479328265)

[2 Package Diagram 4](#_Toc479328266)

[3 Cross-Reference to requirements 5](#_Toc479328267)

[3.1 Functional Requirements 5](#_Toc479328268)

[3.2 Non-Functional requirements 5](#_Toc479328269)

[3.2.1 Testability 5](#_Toc479328270)

[3.2.2 Technical Constraints 5](#_Toc479328271)

[3.2.3 Interoperability 5](#_Toc479328272)

[3.2.4 Maintainability 6](#_Toc479328273)

[3.2.5 Interfaces 6](#_Toc479328274)

[3.2.6 Implementation 6](#_Toc479328275)

[3.2.7 Design 6](#_Toc479328276)

# **Deployment Diagram**

In het deployment diagram wordt duidelijk welke componenten met elkaar communiceren. In het deployment diagram dat hieronder is weergegeven is te zien dat het Operation System Linux Mint 18.1 Cinnamon is. Op dit Operation System draait een instantie van apache-tomee-plus-7.0.2. In de tomcat instantie is de Spotitube.war gedeployt. Ook draait er een instantie van MySQL 5.7.17. In de MySQL instantie zit een database genaamd Spotitube.

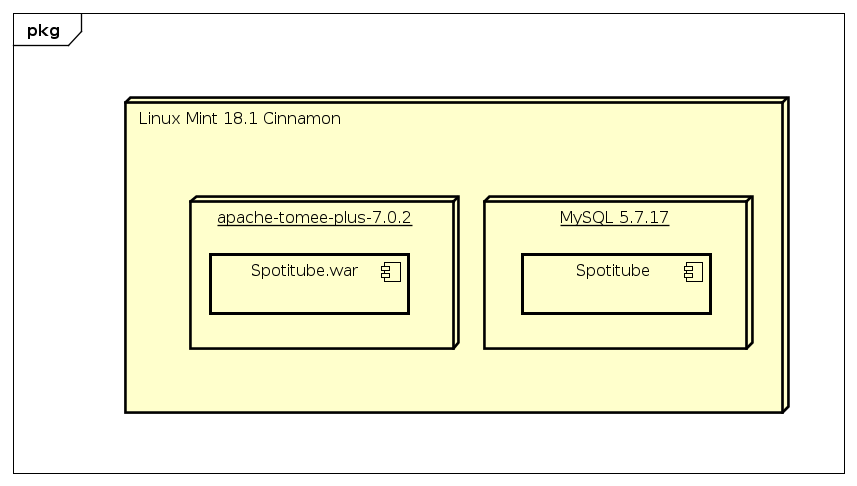


Figure Deployment Diagram Spotitube

# Package Diagram

In het package diagram is te zien wat de packagestructuur van het project is. Hieronder is het package diagram te zien.

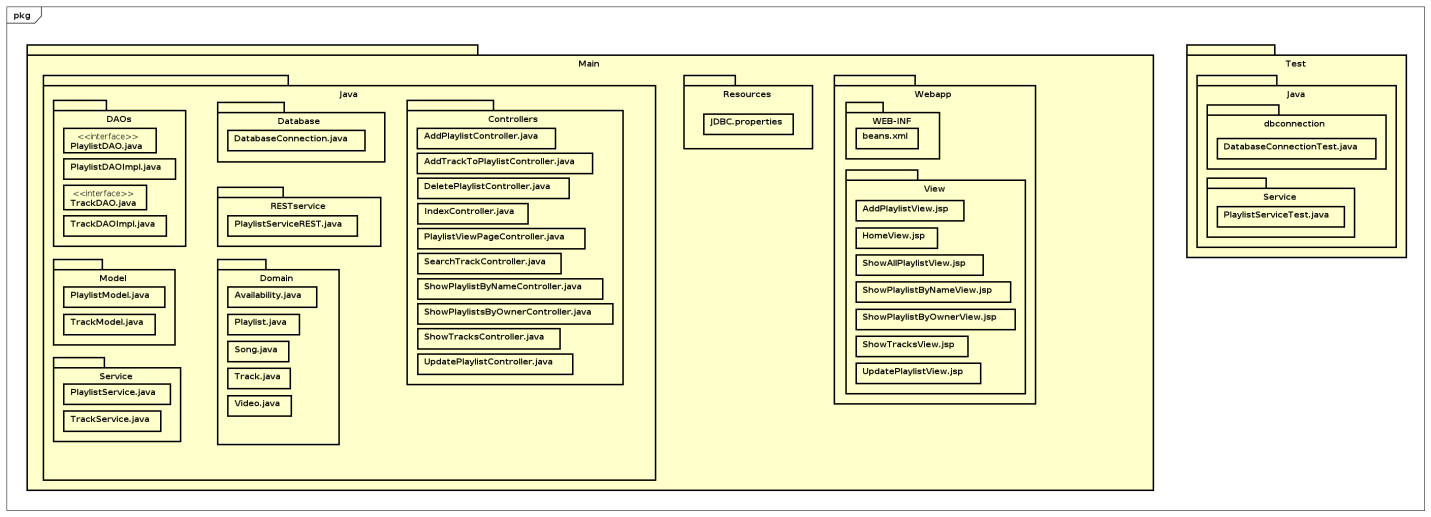


Figure Package Diagram

# Cross-Reference to requirements

Om duidelijke te krijgen waar in de code de requirements zijn afgehandeld, is hieronder een cross-reference tabel te vinden met daarin de requirement en waar deze is afgehandeld.

## Functional Requirements

|  |  |  |
| --- | --- | --- |
| **Nr** | **Requirement** | **Locatie van implementatie** |
| FR1 | Beheer Playlists | ShowAllPlaylistView.jsp en PlaylistViewPageController.java |
| FR2 | Aanmaken Playlist | AddPlaylistView.jsp en AddPlaylistController.java |
| FR3 | Wijzigen Playlist | UpdatePlaylistView.jsp en UpdatePlaylistController.java |
| FR4 | Verwijderen Playlist | ShowAllPlaylistView.jsp en DeletePlaylistController.java |
| FR5 | Toevoegen Track Aan Playlist | ShowPlaylistByNameView.jsp en ShowPlaylistByNameController.java en ShowTracksView.jsp en AddTrackToPlaylistController.java en ShowTracksController.java en SearchTrackController.java |
| FR6 | Details Bekijken Playlist | ShowPlaylistByName.jsp en ShowPlaylistByNameController.java |

## Non-Functional requirements

### Testability

|  |  |  |
| --- | --- | --- |
| **Nr** | **Requirement** | **Locatie van implementatie** |
| NFR1 | De communicatielaag en databasetoegang moeten los van elkaar testbaar zijn. | DatabaseConnectionTest.java en PlaylistServieTest.java |

### Technical Constraints

|  |  |  |
| --- | --- | --- |
| **Nr** | **Requirement** | **Locatie van implementatie** |
| NFR2 | De applicatie moet draaien in TomEE. | If (mvn clean install == BUILD SUCCESS) {  Mv target/Spotitube.war ~/apache-tomee/webapps  }  <http://localhost:8080/Spotitube-1.0-SNAPSHOT/home> |

### Interoperability

|  |  |  |
| --- | --- | --- |
| **Nr** | **Requirement** | **Locatie van implementatie** |
| NFR3 | De front-end en back-end van de applicatie moeten samen in 1 container kunnen draaien zodat ze lokaal met elkaar kunnen  communiceren. | Webapp/view/\*  Java/Controllers/\*  JSP maakt gebruik van Controller. De Controller roept het Model aan. Het Model roept de Service aan. De Service roept de DAO aan. |
| NFR4 | De front-end en back-end van de applicatie moeten elk in een eigen container kunnen draaien zodat ze via een protocol met elkaar kunnen communiceren. | PlaylistServiceREST.java |

### Maintainability

|  |  |  |
| --- | --- | --- |
| **Nr** | **Requirement** | **Locatie van implementatie** |
| NFR5 | Views (JSP-bestanden) bevatten geen Java-code, alleen JSP of JSTL markup. | Webapp/view/\* |
| NFR6 | PageControllers (Servlets) bevatten geen markup, alleen Java-code. | Java/Controllers/\* |

### Interfaces

|  |  |  |
| --- | --- | --- |
| **Nr** | **Requirement** | **Locatie van implementatie** |
| NFR7 | De front-end en back-end moeten Restful (JSON over HTTP) kunnen communiceren. | PlaylistServiceREST.java |

### Implementation

|  |  |  |
| --- | --- | --- |
| **Nr** | **Requirement** | **Locatie van implementatie** |
| NFR8 | De applicatie moet naast MySQL5.1 ook  minimaal 1 andere relationele database  kunnen ondersteunen. | DatabaseConnection.java en JDBC.properties |
| NFR9 | De applicatie maakt gebruik van de  volgende APIs en frameworks:   * JSP * Servlet v3.0 * JAX-RS v2.0 (REST, JSON) * CDI (Context & Dependency injection) * JDBC * JDBC driver v5.1.34 voor MySQL | Webapp/view/\* = JSP  Java/Controllers/\* = Servlet  PlaylistServiceREST.java = JAX-RS  Java/model/\* en Java/service/\* @Dependent boven de class.  En @Inject boven een instantie = CDI  DatabaseConnection.java en  JDBC.properties = JDBC |

### Design

|  |  |  |
| --- | --- | --- |
| **Nr** | **Requirement** | **Locatie van implementatie** |
| NFR10 | Bij het wisselen van database moet de code niet opnieuw gecompileerd hoeven te  Worden. | JDBC.properties |
| NFR11 | De applicatie moet eenvoudig kunnen wisselen van een relationele (RDBMS) opslag naar een andere opslag zoals NoSQL of flat files. | Java/daos/\* |
| NFR12 | De broncode is georganiseerd op functionaliteit, niet op technische rol. | Zie mappenstructuur. |