

Universidade de Coimbra

Faculdade de Ciências e Tecnologia

**Departamento de Engenharia Informática**

**Projecto N.º 6**

**Programação Avançada em Java**

João Pedro Oliveira Brandão Martins

Rafaela David Fernandes Lourenço

Coimbra, 4 de Julho de 2015

Índice

[1 Introdução 1](#_Toc423792997)

[1.1 Eclipse 1](#_Toc423792998)

[1.2 Maven 1](#_Toc423792999)

[1.3 JBoss - WildFly 1](#_Toc423793000)

[1.4 PostgreSQL 1](#_Toc423793001)

[1.5 GitHub 1](#_Toc423793002)

[2 Camada de Apresentação 2](#_Toc423793003)

[2.1 Tecnologias utilizada 2](#_Toc423793004)

[2.1.1 UI Composion Templates 2](#_Toc423793005)

[2.1.2 JSF 2](#_Toc423793006)

[2.1.3 AJAX 2](#_Toc423793007)

[2.1.4 AUTENTICAÇÃO 2](#_Toc423793008)

[2.1.5 Javascript 2](#_Toc423793009)

[2.2 Estrutura das páginas XHTML 3](#_Toc423793010)

[2.2.1 resources/template/common 3](#_Toc423793011)

[2.2.2 Zona não autenticada 3](#_Toc423793012)

[2.2.3 Zona autenticada 3](#_Toc423793013)

[2.3 Interface no servidor 4](#_Toc423793014)

[2.3.1 dei.uc.pt.ar/UserInput.java 4](#_Toc423793015)

[2.3.2 dei.uc.pt.ar/EditLyrics.java 4](#_Toc423793016)

[2.3.3 dei.uc.pt.ar/SearchMusicServer.java 4](#_Toc423793017)

[2.4 Funcionalidades acrescentadas ao Front-End 4](#_Toc423793018)

[2.4.1 Popup Lyrics 4](#_Toc423793019)

[2.4.2 Edição de Lyrics 4](#_Toc423793020)

[2.5 Diagrama UML da camada de apresentação 5](#_Toc423793021)

[2.6 testes 5](#_Toc423793022)

[3 Camada de Negócio 5](#_Toc423793023)

[3.1 Tecnologia Utilizada 5](#_Toc423793024)

[3.2 Data Transfer Objects 5](#_Toc423793025)

[3.2.1 LyricDAO.java 5](#_Toc423793026)

[3.1 Classes Adicionais 6](#_Toc423793027)

[3.1.1 UserLoged.java 6](#_Toc423793028)

[3.2 Diagrama UML da camada de negócio 6](#_Toc423793029)

[4 Camada de Serviços Web 6](#_Toc423793030)

[4.1 Tecnologia Utilizada 6](#_Toc423793031)

[4.2 Classes 6](#_Toc423793032)

[4.3 Pojos 6](#_Toc423793033)

[5 Camada de Dados 6](#_Toc423793034)

[5.1 Tecnologia Utilizada 6](#_Toc423793035)

[6 Empacotamento 6](#_Toc423793036)

[6.1 Parent 7](#_Toc423793037)

[6.2 aweb-web2 7](#_Toc423793038)

[6.2.1 Pom.xml 7](#_Toc423793039)

[6.2.2 Web.xml 7](#_Toc423793040)

[6.2.3 Jboss-web.xml 7](#_Toc423793041)

[6.3 thews-wsLyrics 8](#_Toc423793042)

[6.4 thews-ws2 8](#_Toc423793043)

[6.5 aservice-services2 8](#_Toc423793044)

[6.6 adomain-domain2 8](#_Toc423793045)

[6.6.1 Pom.xml 8](#_Toc423793046)

[6.6.2 Persistence.xml 8](#_Toc423793047)

[6.7 anear-ear2 8](#_Toc423793048)

[7 Conclusões 8](#_Toc423793049)

# Introdução

Este projeto consiste em extender o projecto nº4 de forma a disponibilizar aos utilizadores as letras das músicas presentes nas suas playlists. As letras são obtidas através de web services públicos. Esta aplicação recorre a uma base de dados para guardar letras das músicas editadas pelo utilizador.

Em baixo estão listadas as ferramentas de *software* *opensource* utilizadas neste projeto.

## Eclipse

O *software* de desenvolvimento utilizado foi o Eclipse JEE Luna SR2.

Este foi configurado para aceder diretamente à base de dados por configuração de um Data Source adicionando o driver JDBC “*postgresql-9.4-1201.jdbc41.jar*”.

## Maven

Neste projeto recorreu-se à utilização de *Maven*, como ferramenta de *build* de todo o trabalho. No capítulo 5 será descrito em maior pormenor a estrutura do projeto utilizada.

## JBoss - WildFly

Como web server foi utilizado o Wildfly integrado no Eclipse. Foi configurado no wildfly para acesso directo à Base de Dados por *Data Source* “***p6lyrics***” em conformidade com o ficheiro de configuração “***persistence.xml***” adicionando o driver JDBC comum ao Eclipse “*postgresql-9.4-1201.jdbc41.jar*”. Foi também efetuada uma configuração para efetuar *logging* associado ao package deste projeto.

## PostgreSQL

O software de base de dados utilizado foi PostgreSQL, versão 9.4.1, tendo sido criada a base de dados **p6lyrics** para utilização neste projecto.

## GitHub

Para o desenvolvimento e conceção simultâneos e sincronizados do projeto, foi utilizado o *GitHub*. Este consiste num serviço de *Web Hosting* compartilhado para projetos que usam o controlo de “versionamento” *Git*.

# Camada de Apresentação

A camada de apresentação foi efetuada a pensar num *layout* que fosse bastante intuitivo para o utilizador, sem demasiada informação em cada página, mas com o suficiente para não deixar o utilizador sem saber o que fazer.

Nos passos seguintes vamos descrever as principais funcionalidades na utilização do projecto desenvolvido.

## Tecnologias utilizada

### UI Composion Templates

Esta camada foi desenvolvida segundo um template de CommonLayout, onde existem quarto secções que são comuns a todas as páginas: um CommonHeader, um CommonMenu, um CommonContent e um CommonFooter.

O conteúdo da secção relativa ao *CommonContent*, na figura é a área onde está a tabela com as músicas, vai sendo atualizada de acordo com a página que estamos a consultar.

A utilização deste tipo de template na construção de um website permite que não seja necessário repetir o código dos elementos que são comuns a todas as páginas, sendo apenas escrito na página xhtml correspondente a essa secção. Permite também que a correção e formatação desse código seja efetuada de um modo mais fácil e rápido.

### JSF

Neste projeto utilizamos para a construção de interfaces de utilizador baseadas em componentes para aplicações *web* é feita através do *framework:* *JavaServer* *Faces* (*JSF*). O JSF possui um modelo de programação dirigido a eventos, abstraindo os detalhes da manipulação dos eventos e organização dos componentes, permitindo que nos focássemos somente na lógica da nossa aplicação.

### AJAX

Um dos aspetos fundamentais deste trabalho residiu no recurso à tecnologia *AJAX* (*Asynchronous Javascript and XML*). De forma a tornar as páginas *web* do projeto mais interativas com o utilizador implementou-se o AJAX, como método de uso de solicitações assíncronas de informação. Deste modo, conseguiu-se garantir que o *browser* não carregasse constantemente toda a página, nomeadamente, aquando o clique de botões, resultando todas as interações existentes em chamadas ao servidor.

### AUTENTICAÇÃO

Neste trabalho foi desenvolvido um sistema de proteção recorrendo à tecnologia autenticação pelo container. O Wildfly pode ser configurado no standalone.xml um security domain por forma a ter a função de efetuar o acesso à BD para validar credenciais do utilizador por ***form*** e respectivas ***roles***, por forma a impedir o acesso direto de um utilizador a uma area protegida. Essa protecçao é complementada com configurações no ficheiro web.xml. Neste caso, protegeu-se o diretório ***resources/authorized***. Quando existir um utilizador que não efetuou o “login” com sucesso e tenta aceder à área protegida, é redirecionado para a página *index.xhtml*.

### Javascript

Neste projeto foi utilizado *Javascript residualmente*, incorporado nas paginas xhtml para operações associadas a botões.

## Estrutura das páginas XHTML

Nos pontos seguintes será efetuada uma pequena descrição dos ficheiros do front-end deste projecto.

### resources/template/common

Esta camada foi desenvolvida segundo um template de CommonLayout, onde existem quarto secções que são comuns a todas as páginas: um CommonHeader, um CommonMenu, um CommonContent e um CommonFooter.

### Zona não autenticada

#### index.xhtml

Página de arranque inicial que permite o utilizador efetuar sign-in e sign-up.

### Zona autenticada

#### resources/Authorized

O container ***resources/Authorized*** agrupa as páginas cujo acesso requer autenticação.

##### addMusics.xhtml

Página que permite ver as músicas de todos os utilizadores.

##### addPlaylist.xhtml

Página que permite adicionar playlists.

##### allMusics.xhtml

Página que permite adicionar músicas.

##### changeAccount.xhtml

Página que permite ao utilizador editar os seus dados pessoais e credenciais de acesso.

##### editLyrics.xhtml

Página que permite editar letras de músicas.

##### myMusics.xhtml

Página que permite editar dados das suas músicas.

##### myPlaylist.xhtml

Página que permite gerir as suas playlists.

##### searchMusics.xhtml

Página que permite efectuar pesquisa de músicas.

##### thePlaylist.xhtml

Página que permite ver e apagar musicas da playlist seleccionada.

## Interface no servidor

Para efetuarmos a interface das novas funcionalidades com a camada de negócio, alteramos e acrescentamos as classes descritas nos passos seguintes.

### dei.uc.pt.ar/UserInput.java

Foi alterada esta classe para compatibilizar a autenticação por container, tendo sido alteradas os metodos de login e logout.

### dei.uc.pt.ar/EditLyrics.java

Ainda FALTA FAZER.

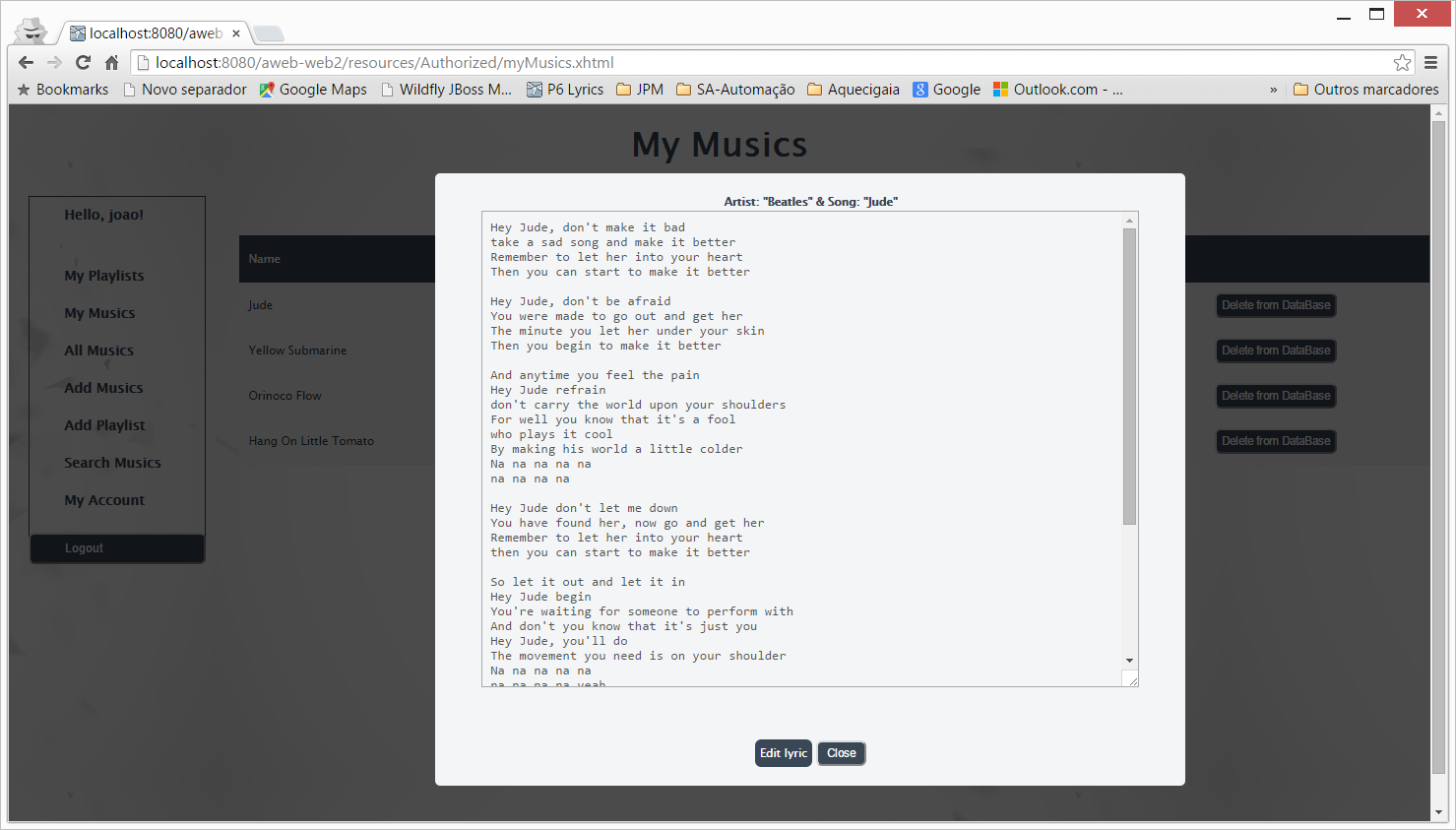
### dei.uc.pt.ar/SearchMusicServer.java

Ainda FALTA FAZER.

## Funcionalidades acrescentadas ao Front-End

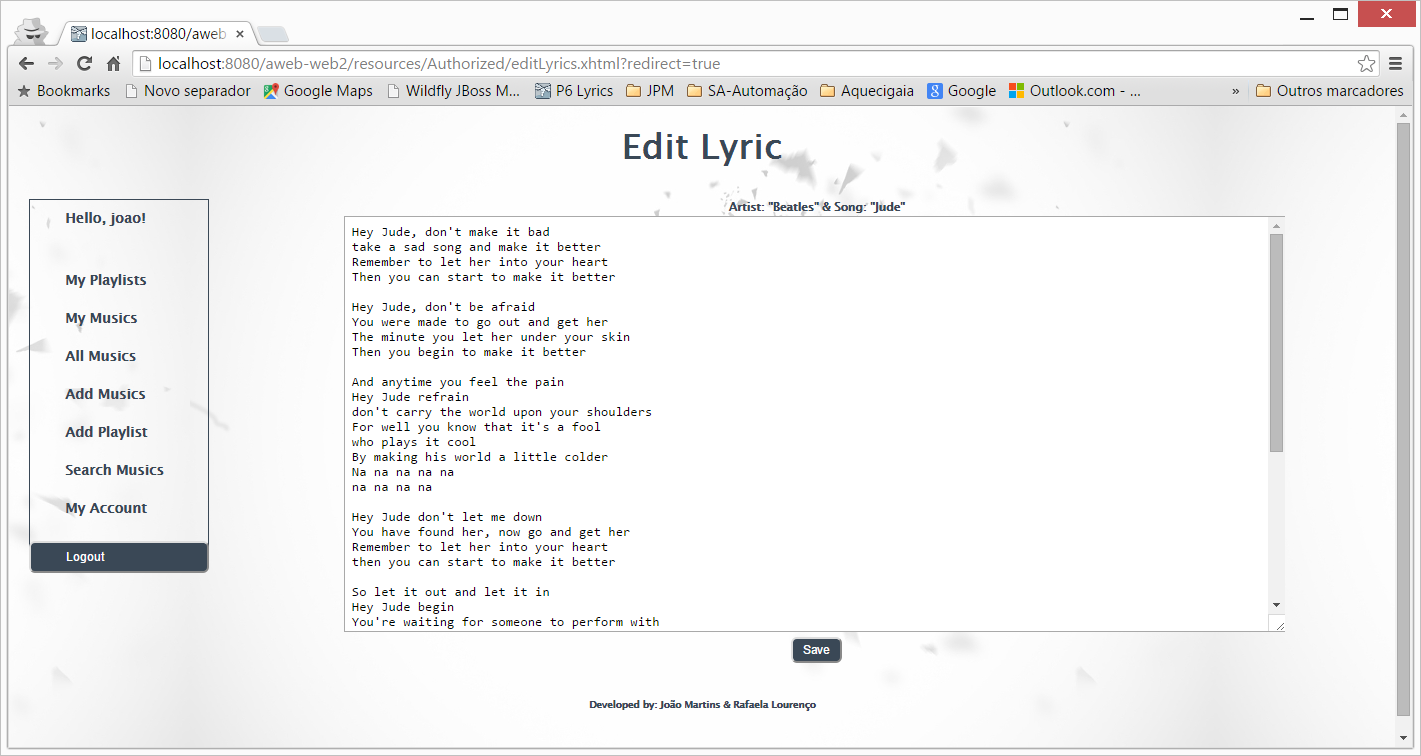
### Popup Lyrics

Ainda FALTA FAZER.



### Edição de Lyrics

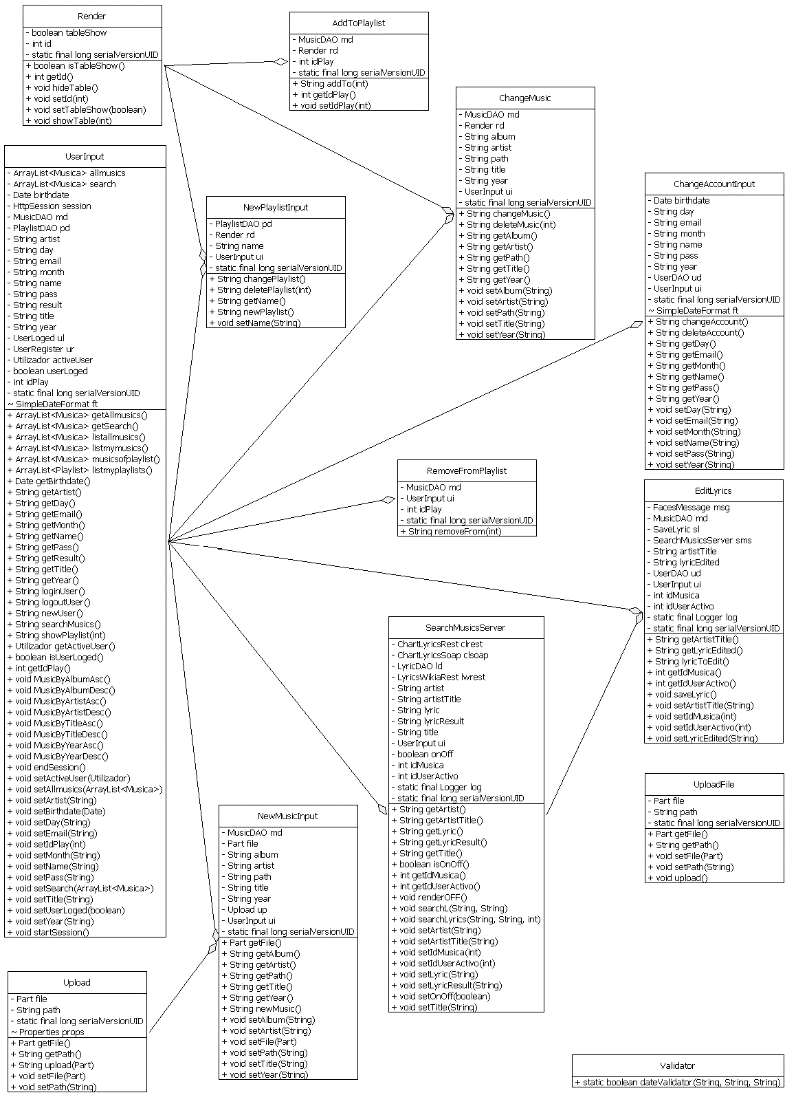
Ainda FALTA FAZER.



.

As letras das músicas são guardadas na base de dados.

## Diagrama UML da camada de apresentação



## testes

Ainda FALTA FAZER.

# Camada de Negócio

## Tecnologia Utilizada

Na camada de negócio a tecnologia utilizada foi EJB (Enterprise Java Beans), consistindo em componentes que são executados nos servidores de aplicação e possuem como principais objetivos, fornecer facilidade e produtividade no desenvolvimento de componentes distribuídos, transacionados, seguros e portáteis

## Data Transfer Objects Adicionados

Ainda FALTA ACRESCENTAR DAO acrescentados das Lyrics.

### LyricDAO.java

Ainda FALTA FAZER.

## Classes Adicionais

### UserLoged.java

Foi adicionada esta classe para possibilitar o controlo de utilizadores logados. Sempre que são utilizadas os métodos login e logout nas classes de interface da camada de apresentação, são guardados de forma não persistente os utilizadores que se autenticam. Para evitar erros na contagem de utilizadores logados em varias sessões recorreu-se a HashMaps.

### SaveLyric.java

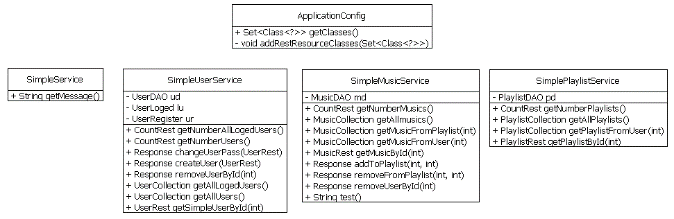
Ainda FALTA FAZER.

# Camada de Serviços Web

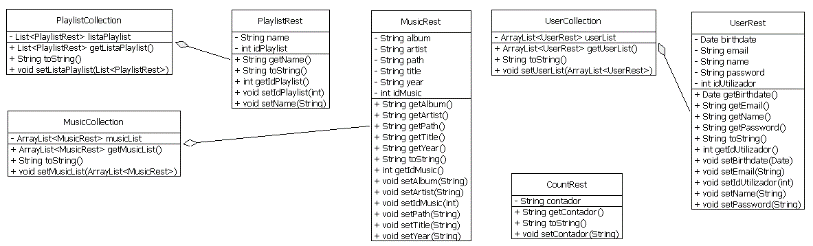
## Tecnologia Utilizada

Na camada de serviços Web foi utilizada a tecnologia Rest.

## Classes



## Pojos

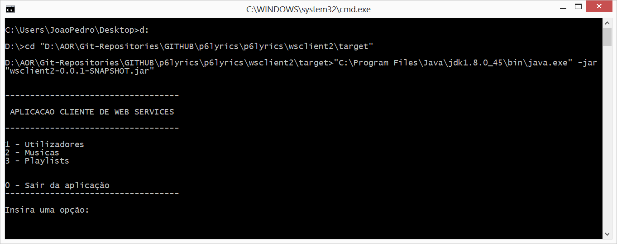
Foram utilizados POJOS utilizando e utilizamos o próprio construtor de cada pojo para converter os dados provenientes da camada de negocio quer para os objetos já existentes para o projecto 4, quer para as novas funcionalidades desenvolvidas neste projeto.

# Camada de Dados

Na camada de dados a tecnologia utilizada foi JPA (*Java Persistence API*), que consiste numa API padrão do Java para a persistência de dados que o utiliza o modelo relacional. Esta API trás uma vantagem de possibilitar uma integração com a base de dados ao nível de entidades sem sequer ser necessário escrever uma linha de SQL, o que liberta o programador para a camada de negócio.

Ao desenho da base de dados existente, acrescentamos 2 entidades. Uma para as roles e outra para as lyrics.

# Cliente WebServices

Foi criada uma aplicação cliente de Web Services Stand-alone. Trata-se de uma aplicação muito simples em modo texto. No entanto permite testar todas as funcionalidades desenvolvidas no modulo WebServices producer.

# Empacotamento

O *Maven* recorre a um arquivo *XML* (*POM* – *Project Object Model*) para descrever todo o projeto de *software* em causa, sendo construídas as suas dependências sobre módulos e componentes externos, a ordem de compilação, diretórios e *plug-ins* que sejam necessários. Para além de realizar os propósitos de compilação de código e seu empacotamento, o *Maven* descarrega bibliotecas Java e seus plug-ins dinamicamente de um ou mais repositórios, como o “*Maven 2 Central Repository”*, e armazena-os numa área de *cache* local. A construção do *Maven* assenta assim numa arquitetura baseada em *plugin*, a qual permite que ele faça uso de qualquer aplicação controlável através da entrada padrão.

Este projecto consiste em 7 sub-projetos.

* **aweb-web2** consiste na camada de apresentação.
* **aservice-services2** consiste na camada de negócio.
* **thews-ws2** consiste na camada de servidor de web services.
* **thews-wslyrics** consiste na camada de cliente de web services externos.
* **adomain-domain2** consiste na camada de dados.
* **anear-ear2** consiste na junção dos 5 anteriores.
* **wsclient2** trata-se de uma aplicação cliente stand-alone capaz de utilizar os web services do projecto.

## Parent

O *POM* (*Project Object Model*) é a peça fundamental de um projeto como este, do tipo *Maven*, pois possui as informações básicas de um projeto, bem como as diretivas de como o artefacto final deste projecto deve ser construído. O ficheiro “pom.xml” deste projeto é constituído pelo artefacto: “theparent2” e possui diversas dependências tais como as necessárias para as tarefas comuns nos sub-projetos.

## aweb-web2

### Pom.xml

As principais dependências são JAVA EE, JAVAX, JUNIT, HAMCREST, JFACES e SERVLET.

### Web.xml

O ficheiro “web.xml” consiste num descritor de implantação que é utilizado como arquivo de configuração do artefacto do projecto.

#### Welcome file list

No ficheiro “web.xml” é descriminado o “*welcome-file-list*” que possui indicação da página de acesso à nossa aplicação:

<welcome-file-list>

<welcome-file>index.xhtml</welcome-file>

</welcome-file-list>

#### Autenticação

Neste projecto o ficheiro “web.xml” possui também informação relativa à configuração da autenticação por web container com o FORM do login, definição dos containers com acesso autenticado por utilizador/password e roles. Essas configurações tem de estar em conformidade com as configurações utilizadas no standalone.xml do Wildfly.

### Jboss-web.xml

O ficheiro “jboss-web.xml” consiste na configuração do security domain utilizado nas configurações do Wildfly.

<jboss-web>

<security-domain>p6lyricsDomain</security-domain>

</jboss-web>

## thews-wsLyrics

As principais dependências são o sub-projecto adomain-domain2, JAVAEE-API, org.apache.axis, JUNIT, RESTEASY, commons-discovery, javax.activation, com.sun.mail, commons-logging, javax.xml.rpc e SLF4J.

## thews-ws2

As principais dependências são o sub-projecto adomain-domain2, JAVAEE-API, JUNIT, RESTEASY, HAMCREST, MOCKITO, SLF4J.

## aservice-services2

As principais dependências são o sub-projecto adomain-domain2, JAVAX, JUNIT, HAMCREST, MOCKITO, SLF4J.

## adomain-domain2

### Pom.xml

As dependências são JAVAX e HIBERNATE.

### Persistence.xml

O datasource responsável pela ligação à base de dados é: java:jboss/datasources/p6lyrics

As classes acrescentadas à entity manager são:

<class>dei.uc.pt.ar.Roles</class>

<class>dei.uc.pt.ar.Lyric</class>

Roles foi criada para agregar Roles aos utilizadores para possibilitar acesso ao container.

Lyric agrega letra de músicas editadas associadas às playlists.

## anear-ear2

As suas dependências são os sub-projectos anteriores .

# Conclusões

Com este trabalho comprovamos a vantagem na utilização de serviços web.