

1 - Feita a configuração do ambiente Eclipse Neon2 - 4.6.2

2- Feita a configuração através de Maven (gerenciador de pacotes Java)

3 - Criado a arquitetura server utilizando Tomcat 9

4 – Arquitetura:

Técnicas e Design Pattern:

- DAO
- DAO Factory
- Controller
- Generics
- Singleton
- RESTFull
- Annotation

5 - APIs Utilizadas

- Maven 3.3.9
- Hibernate-Core 5.2.7.Final
- Hibernate-Validator 5.2.7.Final
- Hibernate-Search 5.5.6.Final
- Resteasy 3.1.0.Final
- Hibernate-Entitymanager 5.0.11.Final
- Mysql Driver 5.1.31
- Javax Servlet-api 3.0.1

6 - Descrição procedimentos utilizados e arquitetura:

→ Utilizo para desenvolver: Eclipse Neon2 com plugins (JBoss Tools destaque para Maven, JSF Rich Faces e AngularJs, Server)

→ Rodar Maven Install ou Build / caso tenha problema de erro em algum ponto desse procedimento rodar Maven clean e fazer rebuild do projeto pelo eclipse.

→ Para levantar o Hibernate implemento a interface ServletContextListener na classe ContextLoaderListener.java e assim o hibernate é levantado com o Tomcat.

→ A sessão hibernate fica disponível para o DAO pela implementação no DAOFactory eu injeto a sessão hibernate para o DAOFactoryHibernate que é instanciado pelo ControllerGenerico por Singleton, eu utilizo no controller especifico de cada UC no caso MusicController.java, poderia ter utilizado a API Spring para gerenciar isso, mas não teria tempo hábil para fazê-lo.

→ Publicado classe RESTFull (WsMusic.java) com RESTEasy + Jackson JSON através de Annotation para facilitar a conversão dos dados recebidos pelo lado AngularJS, preferi publicar um unico rest para facilitar a implementação e devido ao tamanho da aplicação

7- Descrição Web

- Utilizando AngularJS 1.6.1
- Utilizado interface bem simples mas funcional
- Implementado Crud Music, utilizando um único controller para implementação, lista, insert, update e delete, e uma lista de artista associando a uma música a um artista pré cadastrado o CRUD de artista não foi implementado, \$http service está sendo utilizado para fazer a comunicação com o RESTFull java server.
- Ao entrar na tela ele já traz todas as musicas cadastradas, e um input para fazer busca que filtra somente texto que seja maior ou igual a 3 caracteres.

8 – Rest publicados e utilizados:

Salvar música faz o papel de gravar ou fazer update se o ID for zero ele grava um novo registro caso chegue com um ID maior que zero ele atualiza.

PUT - http://localhost:8080/angular_sinax/rest/ws_music/salvar_music

Body data JSON - exemplo: -

```
{
  "id": 5,
  "name": "Musica 2",
  "style": "Free",
  "artist": {
    "id": 7,
    "name": "Led",
    "description": "desc 1"
  }
}
```

DELETE - http://localhost:8080/angular_sinax/rest/ws_music/deletar_music/{id}

GET - http://localhost:8080/angular_sinax/rest/ws_music/list_artist

GET - http://localhost:8080/angular_sinax/rest/ws_music/list_music

GET - http://localhost:8080/angular_sinax/rest/ws_music/list_music/{name}

Exemplo List JSON:

```
[{"id": 5, "name": "Musica 2", "style": "Free",
  "artist": {"id": 7, "name": "Nirvana", "description": "desc 1"}
},
{"id": 6, "name": "Musica 1", "style": "rock",
  "artist": {"id": 7, "name": "Nirvana", "description": "Desc 1"}
}]
```