



# Rapport Projet

## *IMDB : Internet Movie Database*

[INFO-H-303]

---

JACOBS Alexandre ENGELMAN David ENGELMAN Benjamin  
BA2 Informatique

---

Mai 2017

# Table des matières

<b>1</b>	<b>Introduction</b>	<b>2</b>
1.1	Objectif du projet . . . . .	2
<b>2</b>	<b>Première Partie</b>	<b>3</b>
2.1	Modèle entité-association . . . . .	3
2.1.1	Contraintes et hypothèses . . . . .	3
2.2	Modèle relationnel . . . . .	4
2.2.1	Contraintes modèle relationnel . . . . .	5
<b>3</b>	<b>Deuxième Partie</b>	<b>6</b>
3.1	Méthode d'extraction des données . . . . .	6
3.2	Méthode d'insertion dans la base de données . . . . .	6
3.3	Requêtes demandées . . . . .	6
3.3.1	SQL . . . . .	6
3.3.2	Algèbre relationnel . . . . .	9
3.3.3	Calcul relationnel tuple . . . . .	10
<b>4</b>	<b>Site Web</b>	<b>12</b>
4.1	L'administrateur . . . . .	12
4.2	La recherche . . . . .	12
4.3	API . . . . .	12
4.4	Statistiques . . . . .	13
4.5	Commentaires . . . . .	13
<b>5</b>	<b>Conclusion</b>	<b>13</b>

# 1 Introduction

Dans le cadre du cours de "Base de données", nous avons dû réaliser un projet mettant en application les différents concepts vus en cours et lors des séances de travaux pratiques.

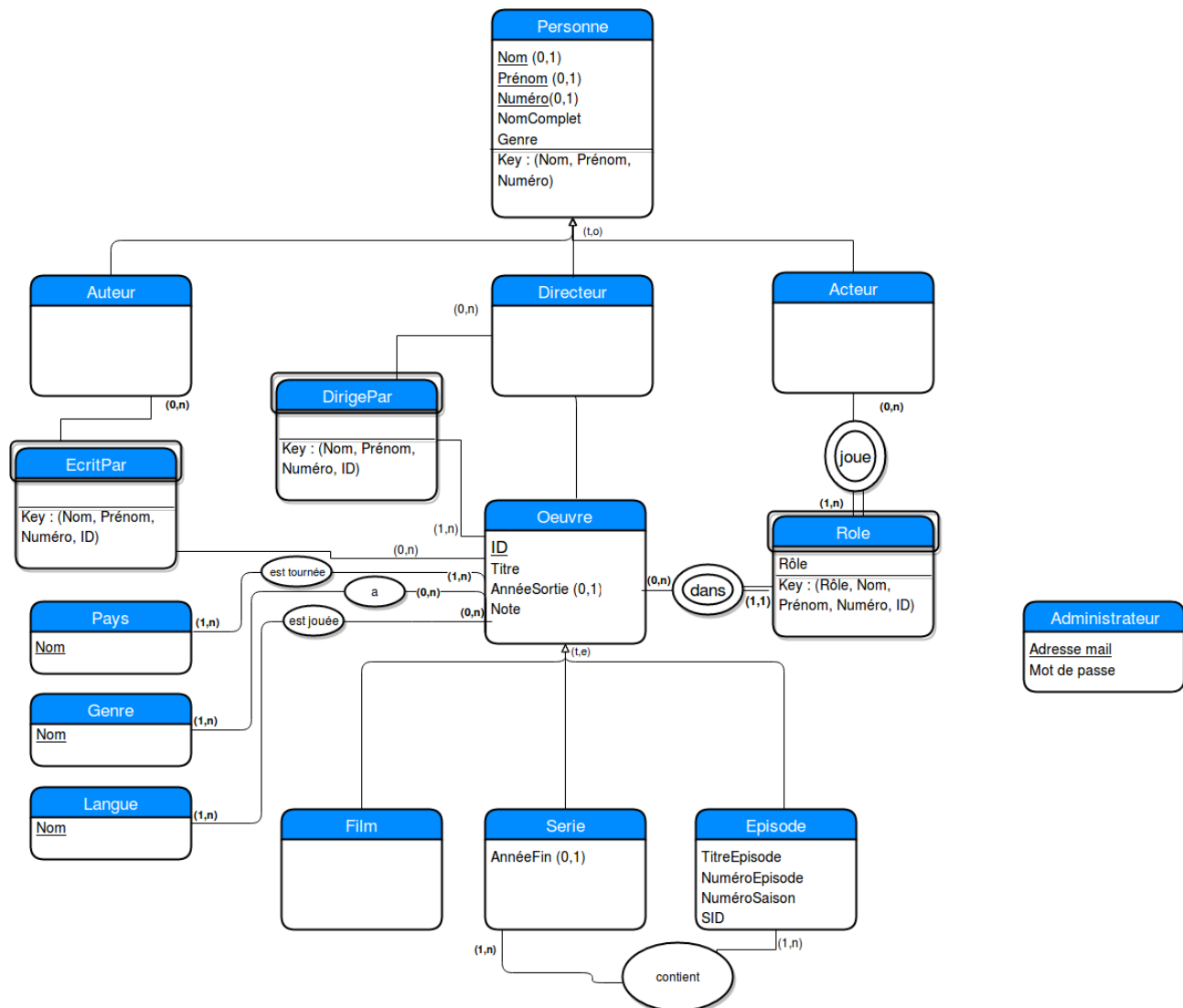
## 1.1 Objectif du projet

L'objectif du projet était de recréer une base de données du type de "IMDB : Internet Movie Database" à partir de fichiers disponibles sur internet. Le projet était divisé en deux grandes parties. La première partie (section 2) consistait en la création d'un diagramme entité-association et d'un diagramme relationnel. La seconde partie (section 3), la plus pratique, était l'implémentation de la base de données et la création du site web.

En plus de devoir faire une base de données, nous avons dû faire les scripts qui permettaient la création de la celle-ci et l'extraction des données qui y sont stockées. Aussi, afin de pouvoir utiliser notre base de données, nous avons créé un site web permettant de rechercher en fonction de critères simples ou avancés des œuvres(film, série ou épisodes) ou des personnes(acteur, directeur, auteur).

## 2 Première Partie

### 2.1 Modèle entité-association



#### 2.1.1 Contraintes et hypothèses

##### Contraintes

- L'année de fin d'une série doit être postérieure à l'année de commencement de la série.
- Un numéro d'épisode ne peut pas être négatif. Une note doit être comprise entre 0 et 10.
- La date de sortie d'un épisode doit être postérieure ou égale à la date de sortie de la série auquel il appartient.

##### Hypothèses

- Une personne (Auteur, Producteur, Directeur, Acteur) peut n'avoir participé à aucune œuvre. Une oeuvre peut avoir été tournée dans plusieurs pays, avoir plusieurs langues ainsi que différents genres.
- L'année de sortie d'une œuvre peut être inconnue.
- Si un pays se trouve dans la base de données, c'est qu'au moins une Œuvre y a été tournée.
- Si un genre se trouve dans la base de données, c'est qu'il existe au moins une Œuvre de ce genre.
- Si une langue se trouve dans la base de données, c'est qu'elle est parlée dans au moins une œuvre.
- Un acteur peut jouer plusieurs rôles dans une même œuvre.

Un rôle peut être joué par plusieurs acteurs (exemple : enfant et adulte).

Plusieurs personnes peuvent avoir le même nom et le même prénom, dans ce cas nous leur attribuons également un numéro pour les différencier.

Le nom ou le prénom d'une personne peut être inconnu mais pas les deux en même temps.

## 2.2 Modèle relationnel

**Personne**(Nom, Prénom, Numéro, Genre)

**Auteur**(Nom, Prénom, Numéro)

Auteur.(Nom, Prénom, Numéro) référence Personne.(Nom, Prénom, Numéro)

**EcritPar**(Nom, Prénom, Numéro, OID)

EcritPar.(Nom, Prénom, Numéro) référence Auteur.(Nom, Prénom, Numéro)

AuteurOeuvre.OID référence Œuvre.ID

**Directeur**(Nom, Prénom, Numéro)

Directeur.(Nom, Prénom, Numéro) référence Personne.(Nom, Prénom, Numéro)

**DirigePar**(Nom, Prénom, Numéro, OID)

DirigePar.(Nom, Prénom, Numéro) référence Directeur.(Nom, Prénom, Numéro)

DirigePar.OID référence Œuvre.ID

**Acteur**(Nom, Prénom, Numéro)

Acteur.(Nom, Prénom, Numéro) référence Personne.(Nom, Prénom, Numéro)

**Rôle**(Rôle, Nom, Prénom, OID)

Rôle.OID référence Œuvre.ID

**Œuvre**(Référence, Titre, Note, AnnéeSortie)

**Pays**(Nom)

**PaysOeuvre**(Nom, OID)

Nom référence Pays.Nom

OID référence Œuvre.ID

**Genre**(Nom)

**GenreOeuvre**(Nom, OID)

Nom référence Genre.Nom

OID référence Œuvre.ID

**Langue**(Nom)

**LangueOeuvre**(Nom, OID)

Nom référence Langue.Nom

Référence référence Œuvre.ID

**Film**(FilmID)

Film.FilmID référence Œuvre. ID

**Série**(SerieID, AnnéeFin)

Série.SerieID référence Œuvre.SerieID

**Episode**(EpisodeID,TitreE, NuméroE, SID)

Episode.EpisodeID référence Œuvre.ID

Epidode.SID référence Serie.SerieID

**Administrateur**(Adresse mail, mot de passe)

### **2.2.1 Contraintes modèle relationnel**

Episode.Date de sortie  $\geq$  Serie.Date de sortie

RéférenceS est unique dans la relation saison (une saison ne peut appartenir qu'à une série).

RéférenceS est unique dans la relation Episode (un Episode ne peut appartenir qu'à une saison).

Référence est unique dans la relation Rôle (un Rôle ne peut appartenir qu'à une Œuvre).

## 3 Deuxième Partie

### 3.1 Méthode d'extraction des données

Pour l'extraction des données qui seront insérées dans notre base de données, nous avons tout d'abord réalisé un script bash permettant le téléchargement des fichiers qui nous étaient nécessaires. En ce qui concerne le traitement des fichiers téléchargés, nous avons réalisé un autre script bash qui lui fait appel à différents parsers écrit en Python. Ces scripts servent à formater les fichiers selon un standard que nous nous sommes fixés et ainsi permettre l'insertion des données dans la base de données. Lors du Parsing, nous avons choisi de garder uniquement les oeuvres tournées entre 2000 et 2017. Quand la date de sortie d'un film ou d'une série était inconnue, nous ne l'avons pas retenue. Lorsqu'une série commençait avant 2017, nous avons pris tous ses épisodes.

### 3.2 Méthode d'insertion dans la base de données

Pour insérer toutes nos données le plus rapidement possible, nous avons choisi d'utiliser la commande SQL "LOAD DATA INTO", utilisant les fichiers résultant de notre parsing. Si des épisodes avaient une date inconnue, nous leur avons attribué la valeur 0. Pour les notes nous avons choisis -1 car 0 n'était pas une valeur aberrante.

Pour garantir l'intégrité des données, nous avons utilisé un trigger lors de la modification des dates. Ce trigger permet de s'assurer que la date de fin d'une série est inférieure ou égale à la date de début.

Pour accélérer, certaines requêtes, nous avons également créé des index à des endroits clés.

### 3.3 Requêtes demandées

Les requêtes demandées étaient les suivantes :

- R1 : Les acteurs qui ont joué toutes les années entre 2003 et 2007.
- R2 : Les auteurs qui ont écrit au moins deux films pendant la même année.
- R3 : Les acteurs Y qui sont à une distance 2 d'un acteur X. Un acteur Y est à une distance 1 d'un acteur X si ces deux acteurs ont joué dans le même film.
- R4 : Les épisodes de série où il n'y a aucun acteur masculin.
- R5 : Les acteurs qui ont joué dans le plus de séries.
- R6 : Les séries avec leur nombre total d'épisodes, le nombre d'épisodes moyen par an et le nombre d'acteurs moyen par saison depuis leur année de création et ce pour toutes les séries dont la note est supérieure à la moyenne des notes des séries.

#### 3.3.1 SQL

```
1 /*R1*/
2 SELECT Nom, Prenom, Numero
3 FROM Acteur a
4 WHERE (SELECT count(distinct AnneeSortie)
5        FROM (
6              SELECT AnneeSortie, Nom, Prenom, Numero
7              FROM Oeuvre o INNER JOIN Role r
8              ON ID = OID
9              WHERE AnneeSortie BETWEEN 2003 AND 2007) t
10        WHERE t.Prenom = a.Prenom and t.Nom = a.Nom and t.Numero = a.Numero) = 5;
```

Listing 1 – Requête 1

```
1 /*R2*/
2 SELECT distinct Nom, Prenom, Numero
3 FROM (
4       SELECT AnneeSortie, Nom, Prenom, Numero
5       FROM Oeuvre o
```

```

6     INNER JOIN Film f ON o.ID = f.FilmID
7     INNER JOIN EcritPar ON ID = OID
8     group by AnneeSortie, Nom, Prenom, Numero
9     having count(*) >=2 )t;

```

Listing 2 – Requête 2

```

1  /*R3*/
2  SELECT DISTINCT Nom, Prenom, Numero
3  FROM (
4      SELECT OID
5      FROM (
6          SELECT R2.OID
7          FROM (
8              SELECT T2.Nom, T2.Prenom, T2.Numero, T2.OID
9              FROM (
10                 SELECT R1.Prenom, R1.Nom, R1.Numero, R1.OID
11                 FROM (
12                     SELECT *
13                     FROM Role
14                     WHERE Nom = 'Elliott' AND Prenom = 'Missy' AND Numero = 'NA
15 ') AS T
16                 JOIN Role R1 ON T.OID = R1.OID) AS T2
17             INNER JOIN Film F ON F.FilmID = T2.OID) AS T3
18             JOIN Role R2 ON T3.Prenom = R2.Prenom AND T3.Nom = R2.Nom AND T3.Numero = R2
19             .Numero) AS T4
20             INNER JOIN Film F2 ON F2.FilmID = T4.OID) AS T5
21 JOIN Role R3 ON T5.OID = R3.OID;

```

Listing 3 – Requête 3

```

1  /*R4*/
2  SELECT DISTINCT EpisodeID
3  FROM Episode e
4  WHERE NOT EXISTS (
5      SELECT Genre, EpisodeID
6      FROM Role INNER JOIN Personne ON Personne.Nom = Role.Nom AND
7      Personne.Prenom = Role.Prenom AND Personne.Numero = Role.Numero
8      WHERE genre = 'm' AND OID = e.EpisodeID);

```

Listing 4 – Requête 4

```

1  /*R5*/
2  SELECT Prenom, Nom, Numero, count(*)nb
3  FROM (
4      SELECT SID, Prenom, Nom, Numero
5      FROM Episode
6      INNER JOIN Role on OID = EpisodeID
7      GROUP BY SID, Prenom, Nom, Numero)t
8  GROUP BY Prenom, Nom, Numero
9  ORDER BY nb desc
10 LIMIT 1

```

Listing 5 – Requête 5



```

1  /*R6*/
2  SELECT T3.SID, ep_num, avg_ep_by_year, avg_actor_by_season
3  FROM (
4      (SELECT ID, count(*) as ep_num
5       FROM(
6           SELECT ID, EpisodeID
7           FROM(
8               SELECT ID
9               FROM Serie INNER JOIN Oeuvre on Oeuvre.ID = Serie.SerieID
10              WHERE note >(
11                  SELECT AVG(note)
12                  FROM(
13                      SELECT note
14                      FROM Serie INNER JOIN Oeuvre on Oeuvre.ID = Serie.SerieID
15                      WHERE note != -1 and AnneeSortie !=0) as t)) as Series_OK
16              INNER JOIN Episode ON Series_OK.ID = Episode.SID) as t
17          GROUP BY ID)T1
18
19      INNER JOIN
20
21      (
22          SELECT SID, avg(num) as avg_actor_by_season
23          FROM(
24              SELECT SID, count(*) as num
25              FROM(
26                  SELECT Nom, Prenom, Numero, SID, Saison
27                  FROM(
28                      SELECT SID, EpisodeID, Saison
29                      FROM Episode INNER JOIN Serie on SerieID = SID
30                      WHERE Saison != -1)t
31                  INNER JOIN Role on EpisodeID = OID
32                  GROUP BY Nom, Prenom, Numero, SID, Saison)t2
33              GROUP BY SID, Saison)t3
34          GROUP BY SID)T2
35
36      on T1.ID = T2.SID
37
38      INNER JOIN
39
40      (
41          SELECT t5.SID, AVG(num) as avg_ep_by_year
42          FROM(
43              SELECT t6.SID, count(*) as num
44              FROM(
45                  SELECT SID, ID, AnneeSortie
46                  FROM Oeuvre
47                  INNER JOIN Episode on EpisodeID = ID)t6
48              WHERE AnneeSortie != 0
49              GROUP BY t6.SID, AnneeSortie) as t5
50          GROUP BY t5.SID)T3
51
52      on T1.ID = T3.SID
53  )

```

Listing 6 – Requête 6

### 3.3.2 Algèbre relationnel

\* Requête 1

$RoleOeuvre \leftarrow \sigma_{AnneeSortie > 2002 \wedge AnneeSortie < 2008}(Oeuvre \bowtie_{r.OID=o.ID} Role \ r)$   
 $RoleOeuvre \leftarrow \pi_{AnneeSortie, Nom, Prenom, Numero}(RoleOeuvre)$   
 $AnneeSortie \leftarrow \pi_{AnneeSortie}(RoleOeuvre)$   
 $R1 \leftarrow \pi_{Nom, Prenom, Numero}(RoleOeuvre / AnneeSortie)$

\* Requête 2

$FilmOeuvre \leftarrow Oeuvre \bowtie_{ID=FilmID} Film$   
 $FilmAuteur \leftarrow FilmOeuvre \bowtie_{ID=OID} EcritPar$   
 $FilmAuteur' \leftarrow FilmAuteur$   
 $Auteur2Film \leftarrow FilmAuteur \bowtie_{f.ID \neq a.ID \wedge f.AnneeSortie=a.AnneeSortie \wedge f.Nom=a.Nom \wedge f.Prenom=a.Prenom \wedge f.Numero=a.Numero}$   
 $FilmAuteur' \leftarrow a$   
 $R2 \leftarrow \pi_{Nom, Prenom, Numero}(Auteur2Film)$

\* Requête 3 (exemple avec Bruce Willis)

$T \leftarrow \pi_{Nom, Prenom, Numero}(\sigma_{Nom="Willis", Prenom="Bruce"}(Role))$   
 $T2 \leftarrow \pi_{R1.Prenom, R1.Nom, R1.Numero, R1.OID}(T \bowtie_{T.OID=R.OID} Role \ R1)$   
 $T3 \leftarrow \pi_{R2.OID}(T2 \bowtie_{T2.Prenom=R2.Prenom \wedge T2.Nom=R2.Nom \wedge T2.Numero=R2.Numero} Role \ R2)$   
 $Res \leftarrow \pi_{Nom, Prenom, Numero}(T3 \bowtie_{T3.OID=R3.OID} Role \ R3)$

\* Requête 4

$Episode \leftarrow \pi_{EpisodeID}(Episode)$   
 $PersonneRoleM \leftarrow Personne \bowtie_{p.Nom=r.Nom \wedge p.Prenom=r.Prenom \wedge p.Numero=r.Numero \wedge p.Genre='m'} Role \ r$   
 $PersonneRoleMEpisodes \leftarrow \pi_{EpisodeID}(PersonneRoleM \bowtie_{OID=EpisodeID} Episode)$   
 $R4 \leftarrow \pi_{EpisodeID, SID}(Episode - PersonneRoleMEpisodes)$

### 3.3.3 Calcul relationnel tuple

\* Requête 1

$$\{a.Nom, a.Prenom, a.Numero | Acteur(a) \wedge \exists r1, r2, r3, r4, r5, o1, o2, o3, o4, o5($$

$$Oeuvre(o1) \wedge Oeuvre(o2) \wedge Oeuvre(o3) \wedge Oeuvre(o4) \wedge$$

$$Oeuvre(o5) \wedge Role(r1) \wedge Role(r2) \wedge Role(r3) \wedge$$

$$Role(r4) \wedge Role(r4) \wedge$$

$$r1.Nom = a.Nom \wedge r1.Numero = a.Numero \wedge$$

$$r1.Prenom = a.Prenom \wedge$$

$$r2.Nom = a.Nom \wedge r2.Numero = a.Numero \wedge$$

$$r2.Prenom = a.Prenom \wedge$$

$$r3.Nom = a.Nom \wedge r3.Numero = a.Numero \wedge$$

$$r3.Prenom = a.Prenom \wedge$$

$$r4.Nom = a.Nom \wedge r4.Numero = a.Numero \wedge$$

$$r5.Prenom = a.Prenom \wedge$$

$$r5.Nom = a.Nom \wedge r5.Numero = a.Numero \wedge$$

$$o1.ID = r1.ID \wedge o1.AnneeSortie = 2003 \wedge$$

$$o2.ID = r2.ID \wedge o2.AnneeSortie = 2004 \wedge$$

$$o3.ID = r3.ID \wedge o3.AnneeSortie = 2005 \wedge$$

$$o4.ID = r4.ID \wedge o4.AnneeSortie = 2006 \wedge$$

$$o5.ID = r5.ID \wedge o5.AnneeSortie = 2007) \}$$

\* Requête 2 :

$$\{a1.Nom, a1.Prenom, a1.Numero | EcritPar(a1) \wedge \exists o1, o2, a2, f1, f2($$

$$Oeuvre(o1) \wedge Oeuvre(o2) \wedge EcritPar(a2) \wedge$$

$$Film(f1) \wedge Film(f2) \wedge$$

$$o1.ID \neq o2.ID \wedge$$

$$o1.ID = f1.ID \wedge o1.ID = a1.ID \wedge$$

$$o2.ID = f2.ID \wedge o2.ID = a2.ID \wedge$$

$$a1.Nom = a2.Nom \wedge$$

$$a1.Prenom = a2.Prenom \wedge$$

$$a1.Numero = a2.Numero \wedge$$

$$o1.AnneeSortie = o2.AnneeSortie) \}$$

\* Requête 3 (exemple avec Bruce Willis)

$$\{r1.Nom, r1.Prenom, r1.Numero | Role(r1) \wedge \exists f1 \exists f2 \exists f3 \exists r2 \exists r3 \exists r4 \exists r5 ($$

$$Role(r2) \wedge$$

$$Role(r3) \wedge$$

$$Role(r4) \wedge$$

$$Role(r5) \wedge$$

$$Film(f1) \wedge$$

$$Film(f2) \wedge$$

$$Film(f3) \wedge$$

$$r1.OID = f1.FilmID \wedge$$

$$r2.OID = f2.FilmID \wedge$$

$$r3.OID = f3.FilmID \wedge$$

$$r4.OID = r2.OID \wedge$$

$$r4.Nom = r1.Nom \wedge$$

$$r4.Prenom = r1.Prenom \wedge$$

$$r4.Numero = r1.Numero \wedge$$

$$r5.OID = r3.OID \wedge$$

$$r5.Nom = r2.Nom \wedge$$

$$r5.Prenom = r2.Prenom \wedge$$

$$r5.Numero = r2.Numero \wedge$$

$$r3.Nom = 'Bruce' \wedge$$

$$r3.Prenom = 'Willis' \wedge$$

$$r3.Numero = 'NA'\}$$

\* Requête 4

$$\{e.EpisodeID, e.SID | Episode(e) \wedge \forall a, p (Personne(p) \wedge Role(a)$$

$$a.Nom = p.Nom \wedge a.Prenom = p.Prenom \wedge p.Numero = a.Numero \wedge$$

$$p.Genre = 'f' \wedge a.OID = e.EpisodeID)\}$$

## 4 Site Web

Le site est divisé en 4 parties principales :

- La page d'accueil, d'où un utilisateur peut lancer une recherche pour une Œuvre ou des personnes.
- L'espace administrateur, permettant d'ajouter des oeuvres et personnes à la base de données.
- La recherche avancée.
- La page de statistiques, offrant quelques informations intéressantes sur les données stockées dans la base de données.

### 4.1 L'administrateur

Lorsqu'un administrateur s'identifie, celui-ci profite de nombreuses options supplémentaires. Outre le fait de posséder le droit d'insertion dans la base données, aussi bien pour une oeuvre qu'une personne, celui-ci peut également complètement modifier les pages du site. En effet, l'interface des pages change lorsque un administrateur est identifié. Cette interface lui permet pour une oeuvre de :

- Modifier le titre.
- Modifier la date de sortie (et date de fin si l'oeuvre est une série).
- Modifier les détails (genres, langues, pays).
- Modifier (ou ajouter) un résumé.
- Ajouter et supprimer des rôles.
- Ajouter et supprimer des directeurs.
- Ajouter et supprimer des personnes.
- Supprimer l'oeuvre.

Et dans le cadre d'une personne de :

- Modifier ses rôles.
- Modifier (ajouter et supprimer) les films dont elle est l'auteur.
- Modifier (ajouter et supprimer) les films dont elle est le directeur.
- Supprimer la personne.

### 4.2 La recherche

L'utilisateur a la possibilité de rechercher des oeuvres (films, séries, épisodes) et des personnes au travers de notre moteur de recherche. Pour trouver un maximum de résultats le plus rapidement possible (et conserver leur pertinence), nous avons choisi d'utiliser les requêtes *MATCH (...) AGAINST (... IN BOOLEAN MODE)* utilisant des fulltext indexes.

Le contenu de l'input entré par l'utilisateur est d'abord traité pour augmenter la pertinence des résultats en ajoutant des "+" devant chaque mot, cela force tous les mots entrés à se trouver dans le résultat et nous évitons donc un nombre trop important de "matches".

Aussi, si jamais l'utilisateur entre des mots composés exclusivement d'une seule lettre, par exemple pour rechercher le film *A (2002)*, une simple requête *WHERE =* sera exécutée pour éviter d'avoir un nombre aberrant de résultats.

### 4.3 API

Nous nous sommes servis de l'api du site [www.themoviedb.org](http://www.themoviedb.org) pour ajouter du contenu à notre site web.

Pour les films ainsi que les séries, nous avons récupéré le poster et une image de fond. Nous avons également pu récupérer les photos des personnes et les trailers des films. Par contre, nous n'avons pas pu certifier la concordance du contenu récupéré avec l'oeuvre ou la personne car l'API utilise un système d'id différent du notre. Il se peut donc que du contenu non cohérent apparaisse dans le cas où 2 oeuvres ont le même titre et la même année de sortie ou lorsque 2 personnes ont le même nom. Les posters sont disponibles pour les films et les séries alors que les trailers sont uniquement disponibles pour les films.

## 4.4 Statistiques

La section statistiques permet à l'utilisateur d'obtenir des informations intéressantes sur les données stockées dans la base de données. Ces informations sont illustrées grâce à des graphiques.

Dans cette section sont disponibles les informations suivantes :

- Le proportion acteurs/actrices.
- La proportion de films/séries/épisodes.
- Le nombre d'oeuvres par pays (top 30).
- L'évolution des notes en fonction du temps.
- L'évolution du nombre de films et séries tournés chaque année.

## 4.5 Commentaires

Sur chaque page d'oeuvre, les utilisateurs ont la possibilité de laisser un commentaire et d'évaluer l'oeuvre en question grâce à un système d'étoiles.

## 5 Conclusion

Pour conclure, ce projet nous aura permis de mettre en pratique les éléments vus au cours théorique ainsi que d'approfondir et conforter nos compétences acquises lors des travaux pratiques. Ce projet nous aura également permis d'apprendre à réaliser un site web et a été un excellent moyen pour nous faire découvrir les concepts de base de php et javaScript. C'est, au final, un des projets les plus complets que nous avons dû réaliser car il nécessitait l'utilisation de différentes technologies qui mises ensemble forment un tout.