

Thomas Parpaillon
Oukache Ayas

Rapport projet Web sémantique

31/01/2021

Lien du dépôt github : <https://github.com/Ayasouk/ProjetWS>

Source des données : <https://www.kaggle.com/justinas/nba-players-data>

Cadre du projet	1
Compétences mises en oeuvre	1
Outils utilisés	1
Représentation en RDF	2
Préparation des données	3
Création du squelette RDF	4
Quelques requêtes utilisées dans l'application	5
Extension des données avec dbpedia	7
Possibilités d'amélioration	7
Captures d'écran	8

1) Cadre du projet

Ce projet est réalisé dans le cadre de la 5ème année de l'école d'ingénieur Polytech Nantes. Il consiste à réaliser un projet mettant en œuvre les concepts appris dans le module de Web sémantique. Ce module est encadré par Mr Fabrice Guillet. Il s'agit, suivant un thème choisi, de restituer et présenter des données qui auront été préalablement formalisées sous la forme de fichier rdf. La présentation doit être sous forme d'une application html/css/js et les données doivent être récupérées à l'aide de requêtes SPARQL.

2) Compétences mises en oeuvre

Parmis les compétences que l'on a vu durant ce cours il y a :

- Savoir utiliser des ressources de données sémantiques comme dbpedia.
- Savoir conceptualiser des données en fichier RDF de différent type (turtle, xml/rdf)
- Savoir requêter une base de données avec des requêtes SPARQL
- Savoir restituer des données sous forme d'une application html/css/js

3) Outils utilisés

Les outils qui ont été utilisés sont les suivants :



Open Refine : Outil google permettant de lire, nettoyer, préparer les données ainsi que gérer les confusions entre les données.

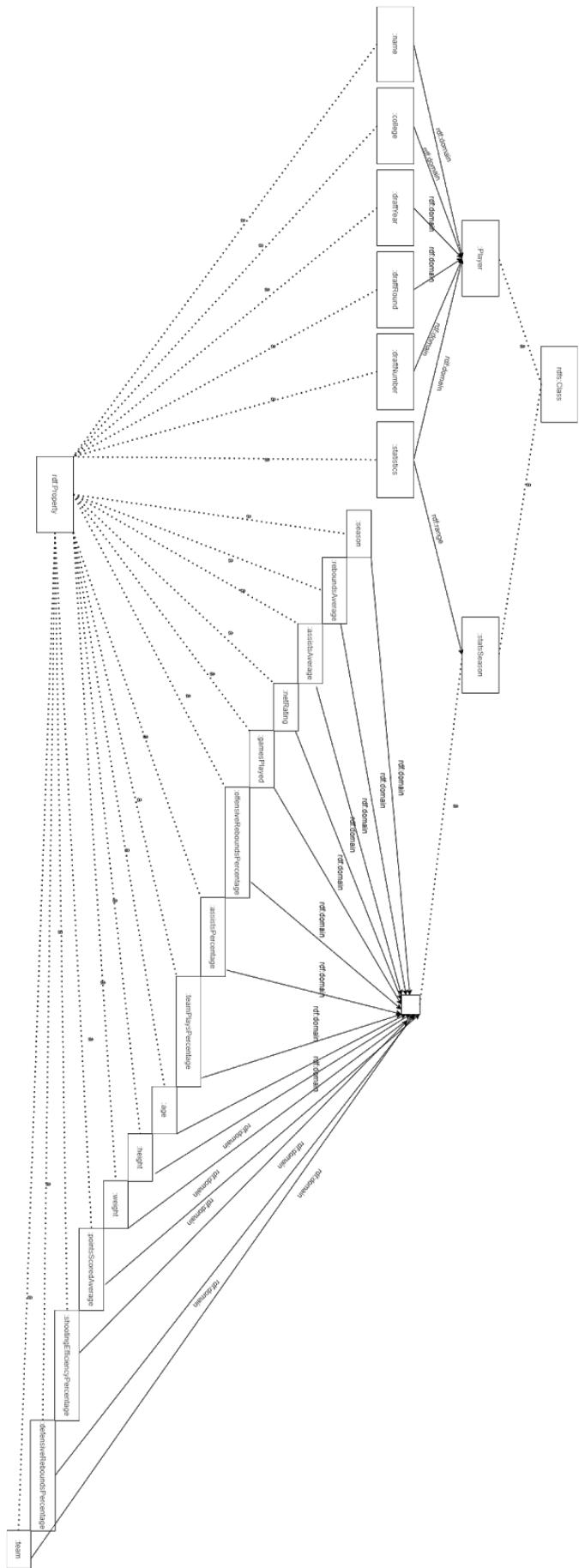


Apache Jena fuseki : Outil d'hébergement de serveur SPARQL permettant d'interroger la base de données choisie avec des requêtes SPARQL.

Pour présenter les différents graphiques dans l'application, nous avons utilisé la librairie D3js.

Pour la présentation globale de l'application, nous avons utilisé la librairie jquery.

4) Représentation en RDF



5) Préparation des données

Pour la préparation des données nous avons traité colonne par colonne les données de telle sorte qu'il n'y ait pas de redondances cachées dans les données. On parle de redondance cachée lorsque deux éléments représentent une même entité mais ont un nom qui diffère.

Pour repérer les redondances cachées nous avons utilisé openrefine, qui propose de plus deux outils pour repérer ce type de redondance :

- Le clustering avec comme algorithme le *key collision*, qui consiste à repérer les mots qui se ressemblent le plus.
- Le clustering avec comme algorithme le *nearest neighbours*, qui consiste à repérer les mots qui ont le plus de lettres en commun en utilisant un paramètre *radius*. Ce paramètre précise le nombre de lettres différentes qu'il faut au minimum pour considérer des mots comme étant proches.

Voici la liste exhaustive des redondances que l'on a trouvé :

- Pour le nom des joueurs:

P. J. Tucker → PJ Tucker

T. J. Warren → TJ Warren

Nous avons arrondi la taille et le poids des joueurs à 2 chiffres après la virgule.

- Pour l'université des joueurs:

St. John's → St. John's NY

Miami → Miami FL

Louisiana-Lafayette → Louisiana Lafayette

- Pour le pays de naissance des joueurs:

US Virgin Islands → U.S. Virgin Islands

Bosnia and Herzegovina - Bosnia & Herzegovina

Nous avons arrondi les pourcentages de rebonds offensifs, le pourcentage de rebonds défensifs, le pourcentage d'efficacité de shoot, le pourcentage de shoots réussis, le pourcentage d'assistances et le pourcentage de jeux en équipe à 3 chiffres après la virgule.

Nous avons trouvé d'autres redondances dans les données, cependant sans que les outils de repérage de cluster ne les détectent :

Au niveau des universités

- Cal-Santa Barbara & California-Santa Barbara
- USC & Southern California
- UNLV & Nevada-Las Vegas
- Cal State-Long Beach & Long Beach State
- Nevada & Nevada-Reno
- California-Los Angeles & UCLA
- Saint Mary's (CA) & St. Mary's (CA)
- Miami & Miami (FL)
- Virginia Commonwealth & Va Commonwealth
- Saint Louis & St. Louis

Nous avons également remarqué que certains joueurs ont un élément manquant ou inexact, que nous avons modifié :

- Danuel House, au niveau du nom (Conflit avec son autre appellation, Danuel House Jr)

- DeAndre' Bembry, au niveau de l'université (l'université renseignée pour une saison, mais pas pour les suivantes (None))
- George Hill, au niveau de l'université (l'université renseignée pour une saison, mais pas pour les suivantes (None))
- Jameer Nelson, au niveau de l'université (l'université renseignée pour une saison, mais pas pour les suivantes (None))
- Langston Galloway, au niveau de l'université (l'université renseignée pour une saison, mais pas pour les suivantes (None))
- Marcus Morris, au niveau du nom (Conflit avec son autre appellation, Marcus Morris Sr)
- Patrick Beverley, au niveau de l'université (l'université renseignée pour une saison, mais pas pour les suivantes (None))

6) Création du squelette RDF

Ensuite au niveau de la définition des triplets pour créer le rdf, voici les différents éléments que nous avons mis en place :

- Transformation des noms des joueurs en URI :
Nous avons remplacé les espaces par des “_”, supprimé les points et remplacé les apostrophes par des “_” sur les noms des joueurs pour les utiliser comme URI.
- Spécification du type de l'URI en :**Player**.
- Ajout des propriétés suivantes :
 - :**draftYear** : l'année de sélection
 - :**draftRound** : le tour de sélection
 - :**draftNumber** : le numéro de sélection
 - :**name** : le nom du joueur
 - :**college** : l'université du joueur lors de sa sélection
 - :**country** : le pays de naissance du joueur
 - :**statistics** : les statistiques du joueur selon les saisons dans lesquelles il a joué
- Spécification du type de :**statistics** en :**statsSeason**
- Ajout de propriétés à :**statistics** :
 - :**season** : Saison jouée
 - :**height** : Taille du joueur
 - :**weight** : Poids du joueur
 - :**age** : Âge du joueur durant la saison
 - :**gamesPlayed** : Nombre de matchs joués
 - :**pointsScoredAverage** : Nombre de points moyens marqués
 - :**reboundsAverage** : Nombre moyen de rebonds
 - :**assistsAverage** : Nombre moyen de passes décisives
 - :**netRating** : Différentiel de points pour 100 possessions
 - :**offensiveReboundsPercentage** : Pourcentage de rebonds offensifs
 - :**defensiveReboundsPercentage** : Pourcentage de rebonds défensifs
 - :**teamPlaysPercentage** : Pourcentage de jeux en équipe
 - :**shootingEfficiencyPercentage** : Pourcentage d'efficacité de shoots
 - :**assistsPercentage** : Pourcentage d'assistance au jeux d'équipe
 - :**team** : Équipe du joueur pour la saison concernée

Une fois le squelette développé nous avons exporté le fichier RDF correspondant, et après que le rdf a été généré nous avons utilisé l'outil easyRDF permettant de nettoyer le rdf et de le rendre plus lisible. Ensuite nous avons mis en ligne le rdf dans un nouveau projet Fuseki, nous permettant de finalement lancer des requêtes sur le serveur fuseki.

Le serveur est lancé à l'adresse 127.0.0.1:3030

Pour requêter le serveur avec du sparql, il suffit d'envoyer une requête http à l'adresse :
127.0.0.1:3030/nom_du_projet_sur_fuseki/sparql

7) Quelques requêtes utilisées dans l'application

Voici la liste des requêtes que l'on utilise pour récupérer des données du serveur fuseki:

`PREFIX : <http://project#>

SELECT DISTINCT ?season

WHERE {

 :_b :statistics [
 :season ?season
]
}

ORDER BY asc(UCASE(str(?season)))`;

>Permet de récupérer toutes les saisons possibles.

`PREFIX : <http://project#>

SELECT DISTINCT ?season

WHERE {

 :NomDuJoueur :statistics [
 :season ?season
]
}

ORDER BY asc(UCASE(str(?season)))`;

>Permet de récupérer les saisons jouées par un joueur.

SELECT DISTINCT

*

where {

 :NomDuJoueur :name ?n;
 :draftYear ?dy;
 :draftRound ?dr;
 :draftNumber ?dn;
 :country ?cou;
 :college ?col

```
}`;
```

>Permet de récupérer les informations ne dépendant pas des saisons liées à un joueur en particulier.

```
`PREFIX : <http://project#>
```

```
SELECT DISTINCT
```

```
*
```

```
where {
```

```
  :NomDuJoueur :statistics [  
    :season "+res_season+";  
    :team ?team;  
    :gamesPlayed ?gp;  
    :pointsScoredAverage ?pts;  
    :reboundsAverage ?rb;  
    :assistsAverage ?as;  
    :netRating ?net;  
    :offensiveReboundsPercentage ?orb;  
    :defensiveReboundsPercentage ?drb;  
    :teamPlaysPercentage ?tp;  
    :shootingEfficiencyPercentage ?s;  
    :assistsPercentage ?asp
```

```
  ]
```

```
};
```

>Permet de récupérer les statistiques liées à un joueur durant une saison donnée, ainsi que l'équipe et l'âge du joueur durant la saison.

```
`PREFIX : <http://project#>
```

```
SELECT DISTINCT ?td ?obj ?season
```

```
WHERE {
```

```
  :NomDuJoueur :statistics [  
    :StatistiqueVoulue ?obj;  
    :season ?season
```

```
  ]
```

```
}
```

```
  ORDER BY asc(UCASE(str(?season)))`;
```

```
}
```

>Permet de récupérer toutes les valeurs d'une statistique sur toutes les saisons pour un joueur donné, et cela ordonné par saison.

```
`PREFIX : <http://project#>
```

```
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
```

```
SELECT ?name ?draft_number
```

```
WHERE {
```

```
  ?name :draftNumber ?draft_number .
```

```
}
```

```
 ORDER BY xsd:decimal(?draft_number)`;
```

>Permet de récupérer les noms et numéros de sélection de tous les joueurs, ordonné par les numéros

8) Extension des données avec dbpedia

Nous avons ajouté la possibilité de rajouter des informations provenant de dbpedia concernant les joueurs. Lorsque le joueur en question est présent sur dbpedia et que l'on retrouve bien le nom correspondant à celui présent dans notre dataset, alors nous avons la possibilité d'afficher des informations supplémentaires. Nous avons décidé d'ajouter le lieu et la date de naissance du joueur à titre d'exemple ; ces informations proviennent de dbpedia.

9) Possibilités d'amélioration

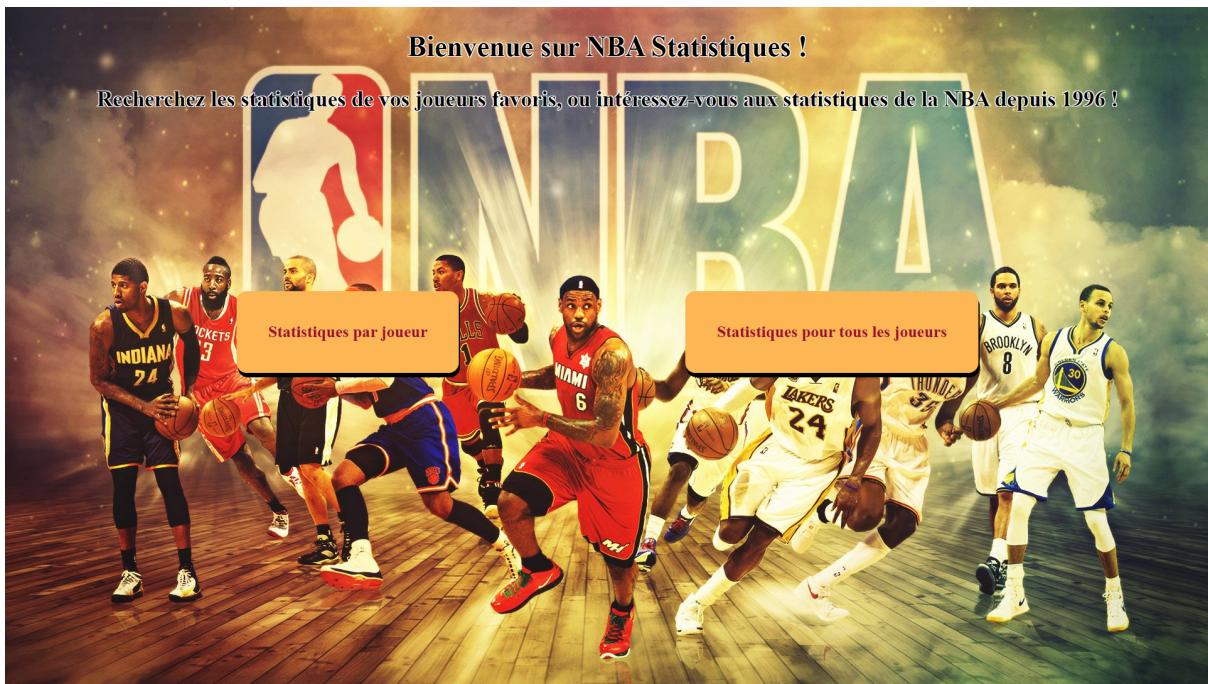
Dans notre application, concernant la partie liée à la recherche d'un joueur, nous pouvons adapter les différents types de graphiques pour représenter les données. Pour le moment nous avons utilisé seulement un graphique en barres, or pour les données de type *pourcentage d'une mesure* ou même *nombre de matchs joués durant toutes les saisons* il aurait été possible de mettre un graphique de type nuage de points connectés qui aurait rendu les données plus lisibles. De plus, il reste le fait d'afficher une seule donnée à la place d'un graphique lorsque l'on sélectionne une statistique pour un joueur donné et pour une saison précise. Il serait bien aussi de donner la possibilité de rechercher un joueur à partir d'un nom incomplet ou ayant une faute.

Du côté de l'affichage des statistiques pour tous les joueurs, deux points sont à améliorer. Pour tous les tableaux : un meilleur affichage grâce à un slider.

Meilleure lisibilité pour les graphiques de la statistique Université grâce à la possibilité de zoomer par exemple sur le graphe.

De manière globale, l'aspect esthétique du site est à améliorer et il faudrait adapter l'affichage des informations à l'écran et au navigateur de l'utilisateur, afin d'éviter que des informations soient plus difficilement lisibles.

10) Captures d'écran



Recherche par statistique

Statistique : **Age** Saison : **Toutes les saisons** Choisir

Affichage des données : Graphique Tableau Afficher

Saisons	Moyenne âge pour tous les joueurs
Saison 1996-97	27.964
Saison 1997-98	28.084
Saison 1998-99	28.002
Saison 1999-00	28.107
Saison 2000-01	28.098
Saison 2001-02	27.673
Saison 2002-03	27.53
Saison 2003-04	27.457
Saison 2004-05	27.297
Saison 2005-06	26.891
Saison 2006-07	26.858
Saison 2007-08	27.193
Saison 2008-09	26.944
Saison 2009-10	27.079
Saison 2010-11	27.135
Saison 2011-12	27.09
Saison 2012-13	27.066
Saison 2013-14	26.963
Saison 2014-15	26.982
Saison 2015-16	27.053
Saison 2016-17	26.848
Saison 2017-18	26.524
Saison 2018-19	26.351
Saison 2019-20	25.574

