

# **Évaluation des connaissances (contrôle continu)**

## **Module « Ontologie & Web Sémantique »**

**Laila Sabar**

Université de Paris  
UFR de Mathématiques et Informatique  
VMI2

Email:

[lailasb2@gmail.com](mailto:lailasb2@gmail.com)

### Table of Contents

<b><i>Introduction .....</i></b>	<b><i>2</i></b>
<b><i>1. Critères de conception des ontologies .....</i></b>	<b><i>2</i></b>
<b><i>2. Structure de l'ontologie.....</i></b>	<b><i>3</i></b>
<b><i>3. Description de l'ontologie MusicOntology .....</i></b>	<b><i>3</i></b>
<b><i>4. Présentation de l'ontologie de musique réalisée .....</i></b>	<b><i>6</i></b>
<b><i>5. Interface web.....</i></b>	<b><i>10</i></b>
<b><i>6. Requêtes SPARQL.....</i></b>	<b><i>15</i></b>
<b><i>Conclusion .....</i></b>	<b><i>16</i></b>

## Introduction

L'ontologie est une spécification explicite de la conceptualisation. Le terme est emprunté à la philosophie, où l'ontologie est un compte rendu systématique de l'existence.

En informatique, l'ontologie est une représentation formelle de la connaissance par un ensemble de concepts dans un domaine et des relations entre ces concepts ce qui permet de donner le sens à l'information. Elle est utilisée pour raisonner sur les propriétés de ce domaine et peut être utilisée pour décrire le domaine. L'ontologie fournit un vocabulaire partagé, qui peut être utilisé pour modéliser un domaine, c'est-à-dire le type d'objets, et/ou de concepts qui existent, ainsi que leurs propriétés et relations.

Les ontologies sont utilisées en intelligence artificielle, dans le Web sémantique, en ingénierie des systèmes, en génie logiciel, en informatique biomédicale et en bibliothéconomie.

La création d'ontologies de domaine est également fondamentale pour la définition et l'utilisation d'un cadre d'architecture d'entreprise.

### 1. Critères de conception des ontologies

Une ontologie doit respecter un ensemble préliminaire de critères de conception afin de garantir le but qui est le partage des connaissances et l'interopérabilité entre les programmes reposant sur une conceptualisation partagée :

- Clarté : Une ontologie doit communiquer la signification proposée des termes définis. Les définitions doivent être objectives et indépendantes du contexte social et informatique.
- Cohérence : L'ontologie doit permettre des inférences qui sont cohérentes avec les définitions. Les axiomes de définition doivent être logiquement cohérents. La cohérence doit également s'appliquer aux concepts qui sont définis de manière informelle, tels que ceux décrits dans la documentation et les exemples en langage naturel.
- Extensibilité : L'ontologie doit être conçue pour anticiper les utilisations du vocabulaire partagé. Elle doit offrir une base conceptuelle pour une gamme de tâches anticipées, et la représentation doit être conçue de manière à ce que l'on puisse étendre et spécialiser l'ontologie de manière monotone.
- Biais d'encodage minimal : La conceptualisation doit être spécifiée au niveau de la connaissance sans dépendre d'un codage particulier au niveau des symboles. Le biais d'encodage doit être minimal, car les agents de partage des connaissances peuvent mettre en œuvre les ontologies dans des systèmes et styles de représentation différents, donc elles ne doivent pas dépendre d'un langage de codage spécifique.

## 2. Structure de l'ontologie

Pour une utilisation appropriée, les ontologies doivent faciliter la communication entre l'homme et la machine, en se référant à la terminologie spécifiée dans l'ontologie, ou même la communication inter-machine et interhumaine.

Une ontologie, en tant que description formelle d'un domaine de discours, repose sur des classes, parfois aussi appelées concepts. Par exemple, une classe de livres représente tous les livres. Les livres spécifiques sont des instances de cette classe.

En outre, une classe peut avoir des sous-classes qui représentent des concepts plus spécifiques que la superclasse. Par exemple, nous pouvons diviser la classe de tous les livres en fiction et non-fiction. Ou encore, nous pouvons diviser la classe de tous les livres en livres pour adultes et livres pour enfants. Les propriétés de chaque classe-concept décrivant diverses caractéristiques et attributs du concept sont appelées slots (parfois rôles ou propriétés)).

Les slots décrivent les propriétés des classes et des instances : *Orgueil et préjugés* est une romance historique ; il est écrit par l'auteur Jane Austin. Nous avons deux slots décrivant le livre dans cet exemple : le slot genre avec la valeur romance et le slot auteur avec la valeur Jane Austin. Au niveau de la classe, nous pouvons dire que les instances de la classe Book auront des slots décrivant leur genre, leur tendance littéraire, leur auteur, etc.

## 3. Description de l'ontologie MusicOntology

MusicOntology fournit un vocabulaire permettant de publier et de relier un large éventail de données relatives à la musique sur le Web. Les données de l'ontologie musicale peuvent être publiées par n'importe qui dans le cadre d'un site web ou d'une API et liées à des données existantes, créant ainsi un réseau de données liées à la musique.

L'ontologie musicale fournit un cadre utile pour :

- Construire un site web musical qui est aussi une API.
- Ouvrir les données relatives à la musique tout en s'assurant qu'elles peuvent être utilisées avec d'autres sources.
- Intégrer des données relatives à la musique provenant de sources multiples.
- Enrichir les résultats des moteurs de recherche autour des titres, des artistes, des œuvres musicales, etc.
- Concevoir un schéma de base de données ou un modèle de domaine lié à la musique.

Voici l'architecture de l'Ontology de music après avoir l'ouvrir sur protégé avec le lien

<http://musicontology.com/specification/index.ttl>



Figure 1: Classes de musicOntology

L'ontologie de la musique est basée sur quatre ontologies principales :

- FOAF, un vocabulaire pour décrire les personnes, les groupes de personnes et les organisations.
- L'ontologie des événements, un vocabulaire pour décrire les événements, Un concept d'événement est défini par un certain nombre de facteurs (comme un instrument de musique, par exemple), d'agents (comme un interprète particulier) et de produits (comme le son physique produit par une interprétation).
- L'ontologie des lignes de temps, un vocabulaire permettant de décrire les intervalles de temps et les instants sur plusieurs lignes de temps (éventuellement liées), par exemple la ligne de temps d'un signal audio. L'information temporelle est la première chose que nous voulons exprimer lorsque nous traitons des connaissances liées à la musique.

- L'ontologie FRBR, un vocabulaire pour décrire les œuvres, les expressions, les manifestations et les éléments et leurs relations, comme défini par les exigences fonctionnelles pour les notices bibliographiques.

Le tableau suivant fournit les références de toutes les ontologies utilisées dans musicOntology.

Prefix	XML Namespace	Specification
<b>frbr</b>	<a href="http://purl.org/vocab/frbr/core#">http://purl.org/vocab/frbr/core#</a>	<a href="#">Expression of Core FRBR Concepts in RDF</a>
<b>timeline</b>	<a href="http://purl.org/NET/c4dm/timeline.owl#">http://purl.org/NET/c4dm/timeline.owl#</a>	<a href="#">The TimeLine ontology</a>
<b>event</b>	<a href="http://purl.org/NET/c4dm/event.owl#">http://purl.org/NET/c4dm/event.owl#</a>	<a href="#">The Event ontology</a>
<b>time</b>	<a href="http://www.w3.org/TR/owl-time/">http://www.w3.org/TR/owl-time/</a>	<a href="#">The Time ontology</a>
<b>dc</b>	<a href="http://purl.org/dc/elements/1.1/">http://purl.org/dc/elements/1.1/</a>	<a href="#">Dublin Core Element Set v1.1</a>
<b>dcterms</b>	<a href="http://purl.org/dc/terms/">http://purl.org/dc/terms/</a>	<a href="#">Dublin Core Element Refinements and Encoding Schemes</a>
<b>foaf</b>	<a href="http://xmlns.com/foaf/0.1/">http://xmlns.com/foaf/0.1/</a>	<a href="#">Friend of a Friend (FOAF) Vocabulary</a>
<b>rel</b>	<a href="http://purl.org/vocab/relationship/">http://purl.org/vocab/relationship/</a>	<a href="#">Relationship: A vocabulary for describing relationships between people</a>
<b>sim</b>	<a href="http://purl.org/ontology/sim/">http://purl.org/ontology/sim/</a>	<a href="#">Similarity Ontology: A vocabulary for describing similarity level between relations</a>

Figure 2: références des ontologies utilisées dans musicOntology

L'ontologie musicale est divisée en trois niveaux d'expressivité - du plus simple au plus complexe - afin de répondre à un large éventail de cas d'utilisation.

Tout est regroupé autour des catégories suivantes :

- Niveau 1 : fournit un vocabulaire pour les informations éditoriales simples (pour exprimer, par exemple, "ce morceau était sur cet album particulier et cette compilation et a été créé par cet artiste").
- Niveau 2 : fournit un vocabulaire pour exprimer le flux de travail de la création musicale (pour exprimer, par exemple, "j'ai assisté à un concert hier soir, que j'ai enregistré avec mon téléphone portable, et voici le flux audio correspondant").
- Niveau 3 : Fournit un vocabulaire pour la décomposition des événements (pour exprimer, par exemple, ce qui s'est passé pendant une performance particulière, quelle est la ligne mélodique d'une œuvre particulière, etc.)

Scénario :

"Un chanteur compile un disque qui est publié par une maison de disques. Ce chanteur se produit également sur une scène et son numéro est enregistré".

Selon la théorie ci-dessus, la première phrase de l'expression est une expression de 1er niveau, tandis que la seconde phrase se réfère au 2ème niveau, comme on peut le constater sur la figure ci-dessus.

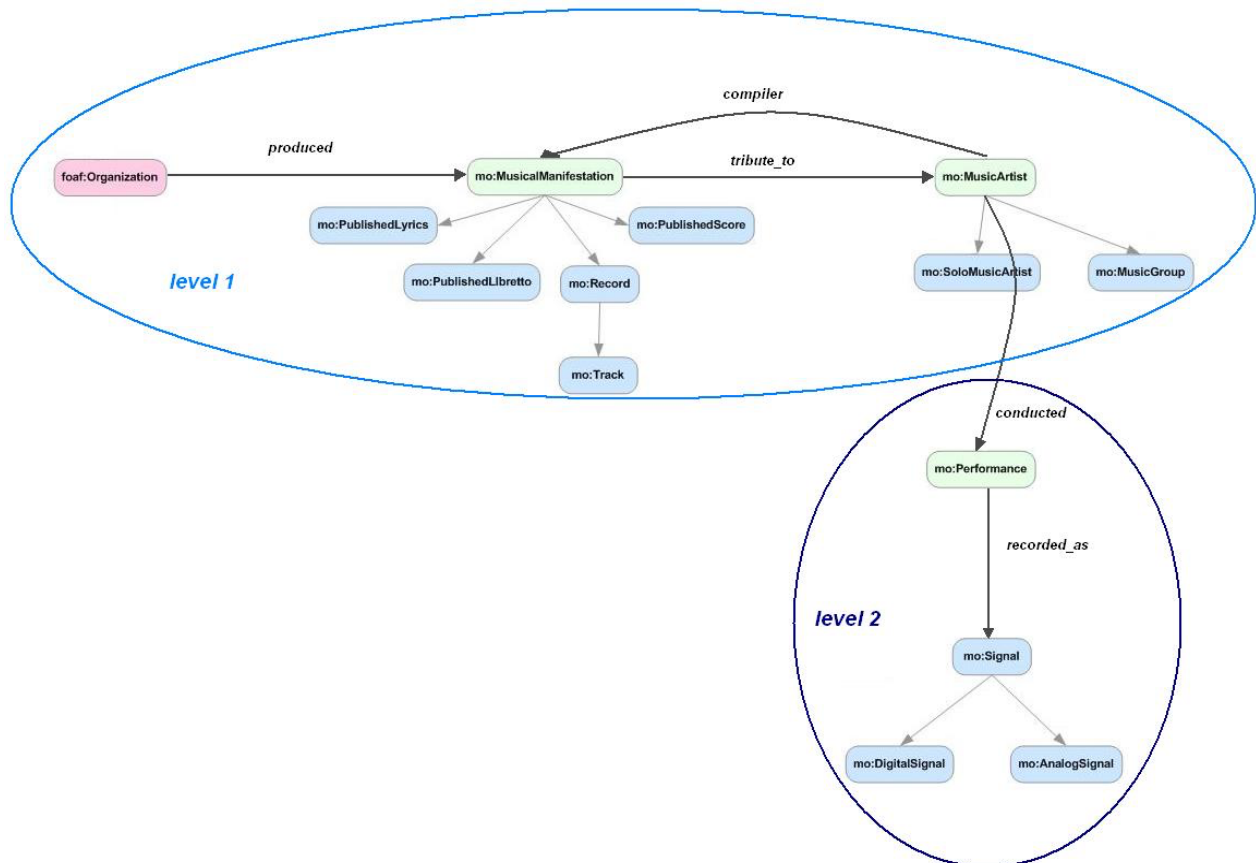


Figure 3 : Description de workflow de musique

#### 4. Présentation de l'ontologie de musique réalisée

Cette partie concerne la présentation de l'ontologie de musique réalisée avec protégé et utilisée au sein d'une application web afin d'interagir avec l'ontologie et permettre une recherche fluide des classes et des instances définis.

## a. Environnement de travail

**Protégé :** Protégé est un système auteur pour la création d'ontologies. Il a été créé à l'université Stanford, il est très populaire dans le domaine du Web sémantique et au niveau de la recherche en informatique. Protégé est développé en Java. Il est gratuit et son code source est publié sous une licence libre. Dans notre projet il était utilisé pour réaliser l'ontologie.

**Python :** Python est un langage de programmation interprété, multi-paradigme et multiplateformes. Il favorise la programmation impérative structurée, fonctionnelle et orientée objet. Dans notre projet il était utilisé comme un langage pour le backend.

**Flask :** Flask est un micro framework open-source de développement web en Python. Il est classé comme microframework car il est très léger. Flask a pour objectif de garder un noyau simple mais extensible. Dans notre projet il était utilisé pour assurer l'interaction entre le backend et le frontend.

**Bootstrap :** Bootstrap est une collection d'outils utiles à la création du design de sites et d'applications web. C'est un ensemble qui contient des codes HTML et CSS, des formulaires, boutons, outils de navigation et autres éléments interactifs, ainsi que des extensions JavaScript en option. Dans notre projet bootstrap a été utilisé afin de bien présenter les pages web.

## b. Ontologie

### Classes

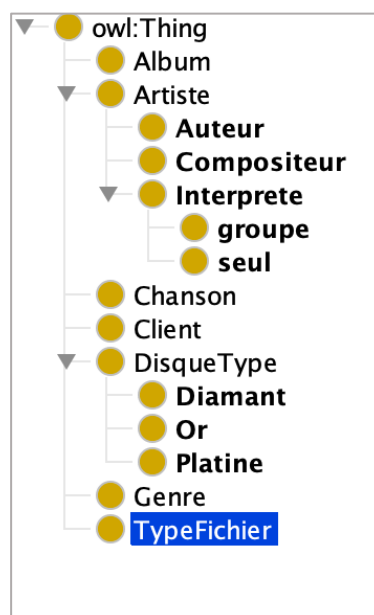


Figure 4 : Classes de l'ontologie musique

Un ensemble de classes a été utilisé afin de décrire l'ontologie musique, la classe album représente les albums des artistes, la classe artiste a comme sous classes auteur, compositeur et interprète, et un interprète peut être soit en groupe ou bien seul, un artiste peut être au même temps auteur, compositeur et interprète. Il y a une classe qui représente les chansons des albums, puis une classe

pour les clients qui ont acheté des albums. La classe DisqueType est pour catégoriser un album selon le nombre des ventes, soit diamant, or, ou platine. La classe genre sert à donner le genre de la musique si elle est rock, pop, classique ...etc, et finalement le type de chanson qui peut être vidéo, audio ou autre.

#### Propriétés /relations



Figure 5: propriétés de l'ontologie musique

Un ensemble de propriété a été proposé afin de lier les classes et les individus entre eux. Ces propriétés sont définies comme suit :

- Un client achète un album
- Un album appartient à un disqueType
- Un compositeur compose une chanson
- Un album contient des chansons
- Une chanson est de genre métal, classique, ...
- La chanson est de type vidéo ou audio
- Un album peut être vendu à un client
- Un interpréteur interprète une chanson
- Un auteur rédige une chanson
- Un artiste peut vendre un album ou une chanson

#### Propriétés data




Figure 6 : propriétés data de l'ontologie musique





Un ensemble de propriétés data a été utilisé afin de définir les propriétés indispensables pour les classes qu'on a créé, la totalité des propriétés data ont été utilisés pour décrire la classe album. Un album a une date de sortie, un prix, un rating, et un nombre de ventes.

### Restrictions

**Description: Or**

Equivalent To 

SubClass Of 

 **DisqueType**





 **etre\_vendu min 50000 Album**

Figure 7 : restriction sur le nombre de ventes d'un album

**Description: Diamant**

Equivalent To 

SubClass Of 

 **DisqueType**


 **etre\_vendu min 500000 Album**

Figure 8 : restriction sur le nombre de ventes d'un album

**Description: Platine**

Equivalent To 

SubClass Of 

 **DisqueType**

 **etre\_vendu min 100000 Album**

Figure 9 : restriction sur le nombre de ventes d'un album

Des restrictions ont été mis afin de catégoriser l'album selon le nombre de ventes, un album qui était vendu au minimum 50000 fois est considéré comme un album d'or, un album vendu au minimum 100000 est considéré album de platine et un album vendu au minimum 500000 est un album de diamant.

## Instances

On a essayé d'instancier chaque classe comme vous pouvez le constater dans les figures ci-dessous

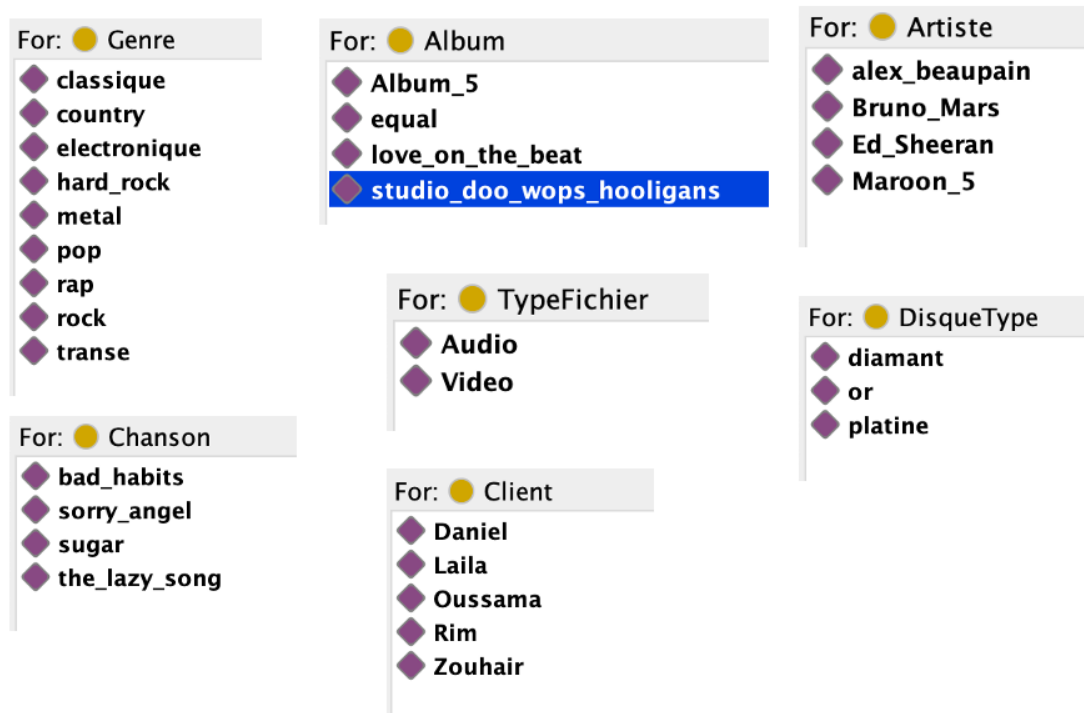


Figure 10 : instances des classes de l'ontologie

## 5. Interface web

Afin de faciliter la recherche des informations de l'ontologie, une application web a été proposée, cette application web est réalisée en utilisant le langage python pour le backend et le frontend et développée avec bootstrap, pour le langage de requête on a utilisé les commandes SPAQL pour interagir avec l'ontologie.

### Page d'accueil

Voici la page d'accueil qui affiche deux moteurs de recherches, un moteur de recherche qui utilise l'ontologie wikipedia afin de récupérer de l'information, et un autre moteur de recherche qui utilise l'ontologie réalisé pour envoyer la réponse.

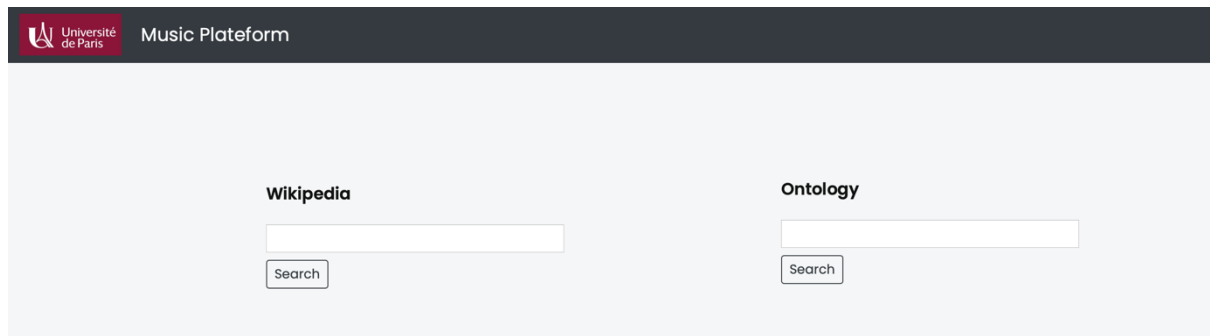


Figure 11 :home page

La page d'accueil contient aussi une liste des albums qui représente les instances créées dans l'ontologie, avec un lien get détails pour avoir plus d'information sur chaque album.

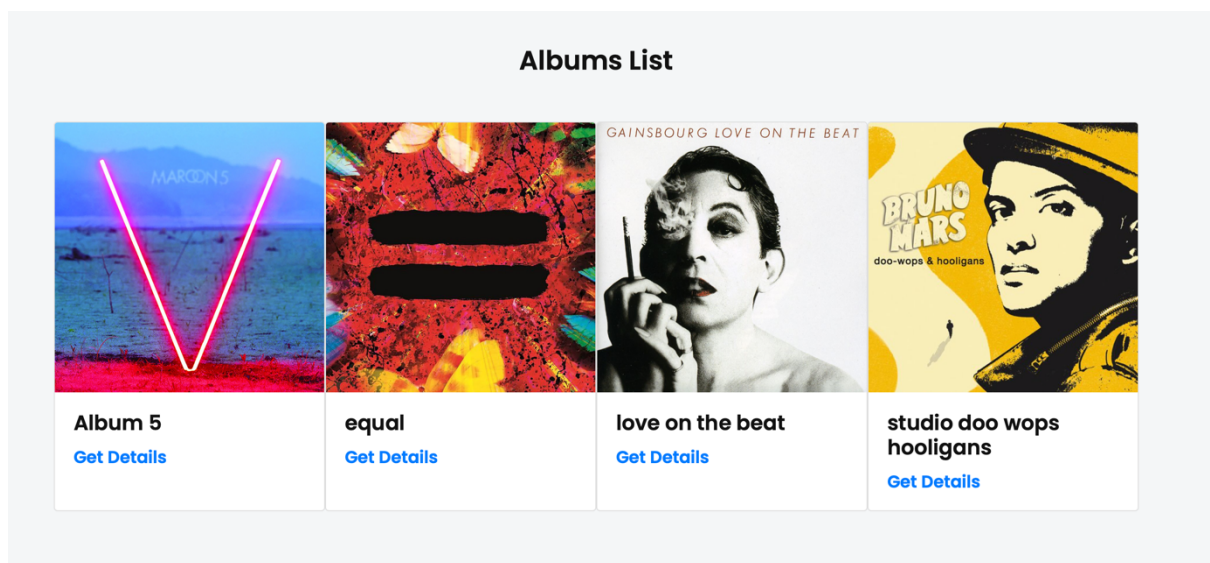


Figure 12 : home page

### Détails album

Lorsqu'on clique sur le lien get details, la page suivante est affichée qui contiens des informations sur l'album sélectionné, comme le nom de l'artiste, la date de sortie, le prix, le nombre de ventes et le rating.

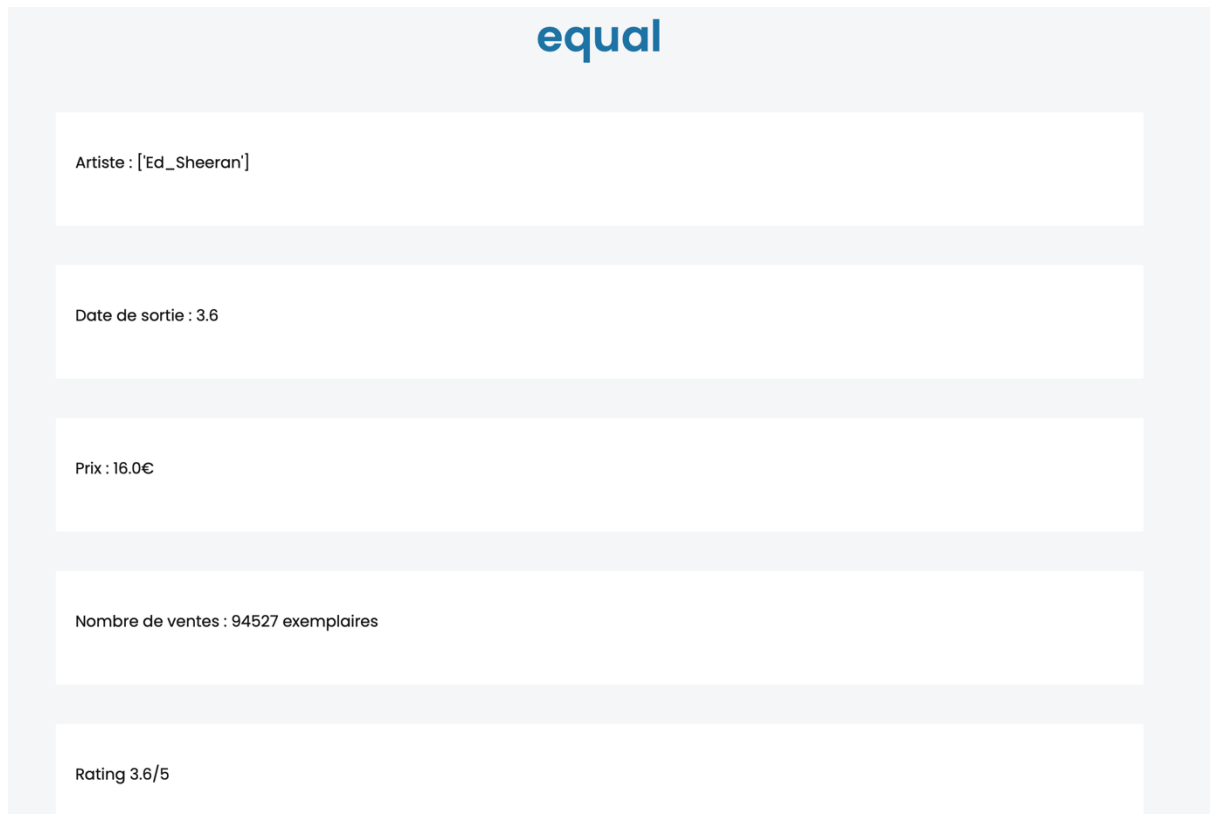


Figure 13 : détails album

La page d'accueil affiche aussi les noms des clients qui ont déjà acheté les albums affichés au-dessus, avec un lien get details pour afficher plus d'information sur l'achat.

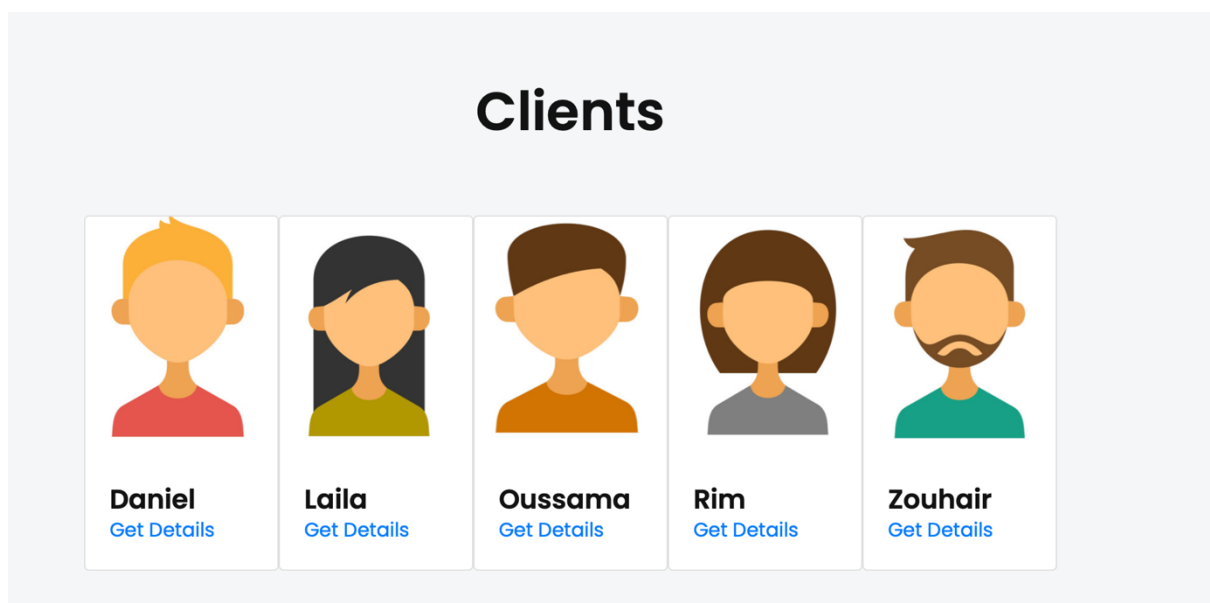


Figure 14 : home page

### Détails achat client

Lorsqu'un client est sélectionné une page avec les détails de l'achat s'affichent, ces détails sont le nom de l'album acheté et le nom de l'artiste.

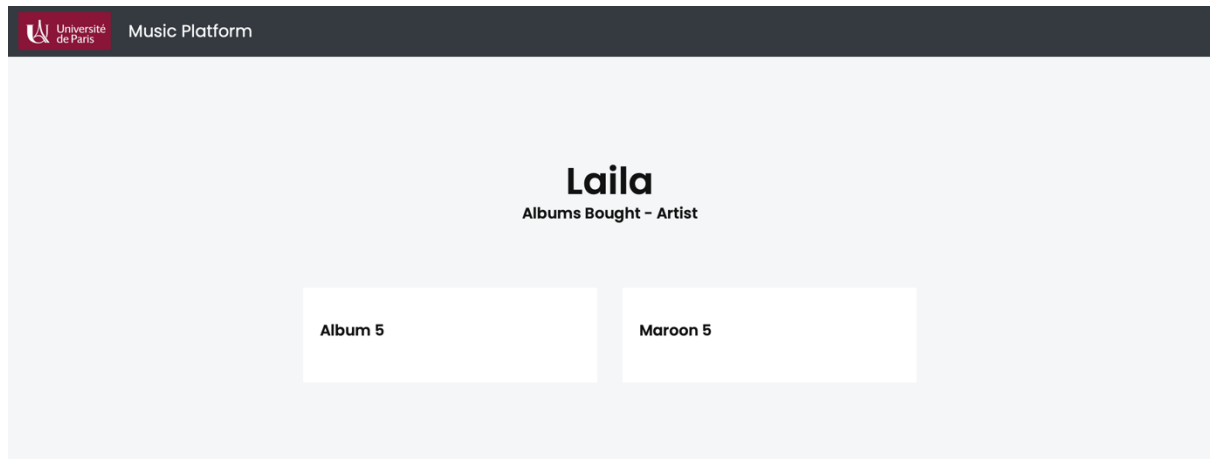


Figure 15 : détails achat client

### Moteur de recherche wikipedia

Ici on peut chercher plus des données sur la musique qui n'existe pas dans notre ontologie

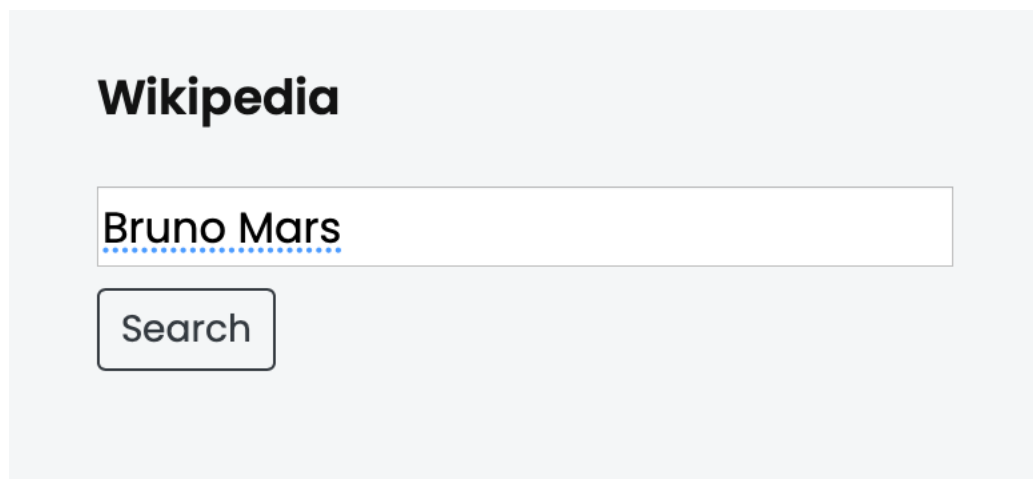
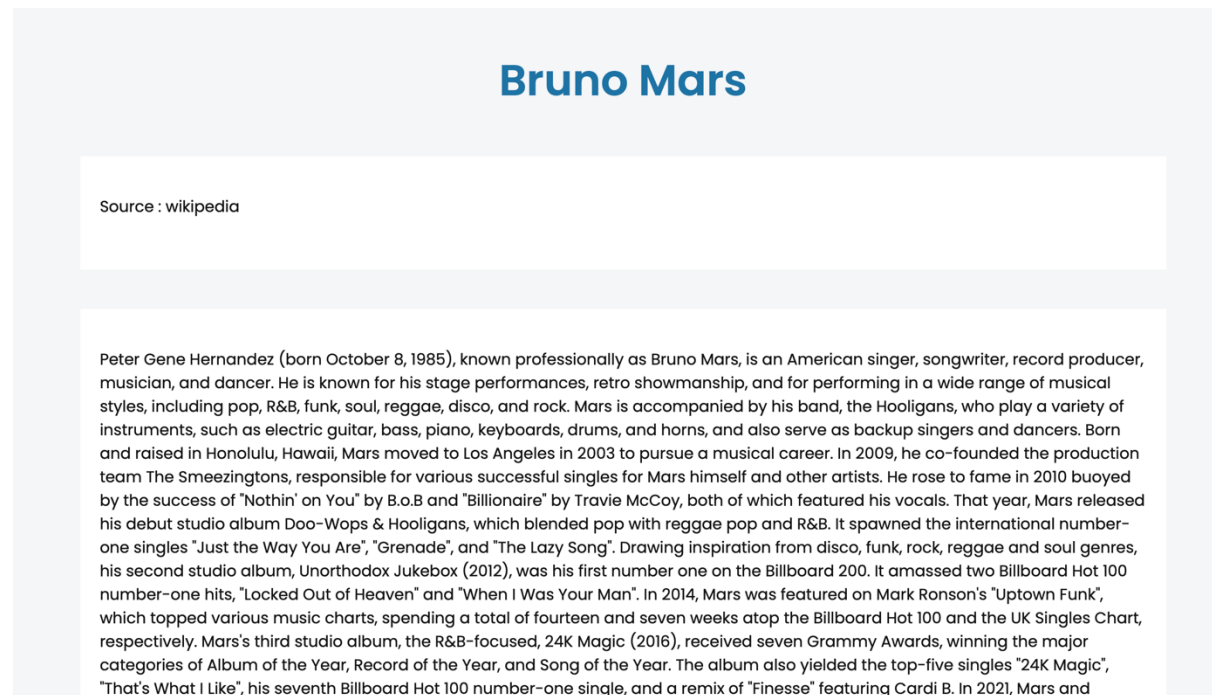


Figure 16 : moteur de recherche wikipedia

Voici le résultat de la recherche



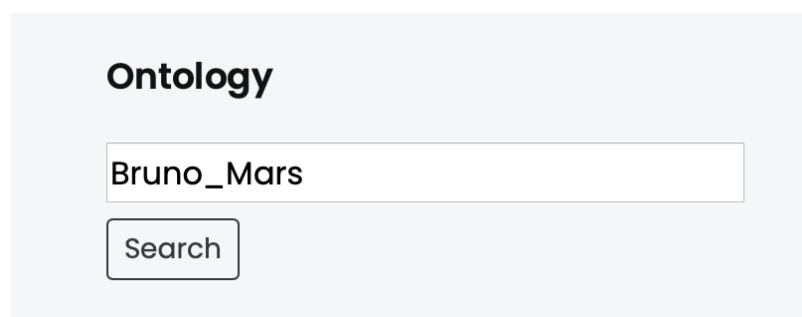
The screenshot shows a search result for 'Bruno Mars' on Wikipedia. At the top, the name 'Bruno Mars' is displayed in a large, bold, blue font. Below it, the source is cited as 'Source : wikipedia'. The main body of the text provides a detailed biography of Peter Gene Hernandez, known as Bruno Mars, including his birth date (October 8, 1985), his musical genres (pop, R&B, funk, soul, reggae, disco, and rock), and his career milestones. It mentions his band, the Hooligans, his debut album 'Doo-Wops & Hooligans' (2010), his second album 'Unorthodox Jukebox' (2012), and his third album '24K Magic' (2016), which won several Grammy Awards. It also notes his collaborations with Mark Ronson and Cardi B.

Figure 17 : résultat recherche wikipedia

### Moteur de recherche ontologie

Ici on peut chercher les noms des artistes, et les noms des albums, mais on doit écrire le nom exactement comme il est défini dans l'ontologie pour avoir la réponse

.



The screenshot shows a simple web interface for an ontology search engine. The title 'Ontology' is centered at the top in a bold, black font. Below the title, there is a text input field containing the text 'Bruno\_Mars'. Underneath the input field is a button labeled 'Search'.

Figure 18 : moteur de recherche ontologie

Voici le résultat de la recherche

## Bruno\_Mars

Artiste , Auteur , Compositeur , Interprete , seul

Chanson : the lazy song

Album : studio doo wops hooligans

## 6. Requêtes SPARQL

Voici quelques requêtes SPARQL utilisées :

Afficher les instances des albums

```
qres = g.query("""PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
                PREFIX owl: <http://www.w3.org/2002/07/owl#>
                PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
                PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
                PREFIX ex: <http://www.semanticweb.org/macbook/ontologies/2021/11/Music#>
                SELECT ?subject
                WHERE { ?subject rdf:type ex:Album }""")
```

Figure 19 : requête SPARQL

Afficher les instances des clients

```
qres = g.query("""PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
                PREFIX owl: <http://www.w3.org/2002/07/owl#>
                PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
                PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
                PREFIX ex: <http://www.semanticweb.org/macbook/ontologies/2021/11/Music#>
                SELECT ?subject
                WHERE { ?subject rdf:type ex:Client }""")
```

Figure 20 : requête SPARQL

Afficher à quelle classe appartient un certain artiste

```
qres2 = g.query("""PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX xd: <http://www.semanticweb.org/macbook/ontologies/2021/11/Music#>

SELECT ?subject
WHERE { xd:""" + artiste + """ rdf:type ?subject}""")
```

Figure 21 : requête SPARQL

Affiches les propriétés data d'un certain album

```
qres2 = g.query("""PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX xd: <http://www.semanticweb.org/macbook/ontologies/2021/11/Music#>

SELECT ?prix ?ventes ?rating ?date_sortie
WHERE { xd:""" + individu + """ xd:prix ?prix.
        xd:""" + individu + """ xd:ventes ?ventes.
        xd:""" + individu + """ xd:rating ?rating.
        xd:""" + individu + """ xd:date_sortie ?date_sortie.
}""")
```

Figure 22 : requête SPARQL

## Conclusion

Ce projet a été très bénéfique pour se familiariser et mettre en pratique la notion des ontologies. Le logiciel Protégé est très pratique pour générer ce type de fichiers, avec différents formats, il permet de visualiser et découvrir des ontologies déjà existantes, et aussi de tester les requêtes sparql afin de voir directement le résultat. La partie application web est aussi intéressante car elle permet de mettre en valeur le travail effectué et aussi permet d'avoir une interface graphique facile à utiliser et plus compréhensible.