Polytech'Nice-Sophia

École d'ingénieurs

SemanticFM

Rapport de projet

Maximilien Perrin - Camille Roux

2008

SOMMAIRE

Introduction

Le web, depuis ses débuts, est soumis à des effets de mode, comme dans la plupart des domaines concernant les nouvelles technologies. On peut facilement découper l'histoire du web et de l'internet en génération. La première génération est ce que l'on appelle couramment le web 1.0. A cette époque, le web était statique. Une page était comme un document sorti tout droit d'un traitement de texte. Ensuite est venu l'époque des forums puis des wiki. Cette période souvent appelée web 1.5 correspond à l'apparition de site ou le contenu des pages est généré par les utilisateurs eux-mêmes (ex: Wikipedia). Après, on a vu venir la génération web 2.0, celle-ci est correspond à de nombreuses évolutions : design (nuages de tags, reflets, rayure, mots écrits en très gros, design avec peu de colonnes), réseaux sociaux, hiérarchies et catégories remplacées par les tags, partage de fichiers, profil utilisateur, mash-up de webservices, ... Cette époque a vraiment été une révolution dans le monde du web et de l'internet; tellement qu'elle a influencé d'autres milieux comme celui de l'entreprise (on entend de plus en plus parler de l'entreprise 2.0) et il y a même certains projets proposant d'appliquer les concepts du web 2.0 aux systèmes scolaires.

Aujourd'hui, nous entrons dans une nouvelle ère, celle du web 3.0. Celle-ci s'axe sur plusieurs points : les applications web, l'intelligence artificielle, l'adaptation aux navigateurs mobiles mais aussi et surtout le web sémantique. Nous sommes certains que le web sémantique va lui aussi marquer la cybersphère. Il suffit de voir des sites comme PowerSet qui sont capables de répondre à des questions comme "what did Steve Jobs say about Apple?" en utilisant les données "sémantiques" contenues sur différents sites web et en essayant d'interpréter les langues naturelles.

C'est pour cette raison que nous avons choisi l'option web sémantique pour notre dernière année à Polytech'Nice-Sophia et que nous avons été très heureux de réaliser le projet que nous allons vous présenter sans plus attendre.

Présentation du projet

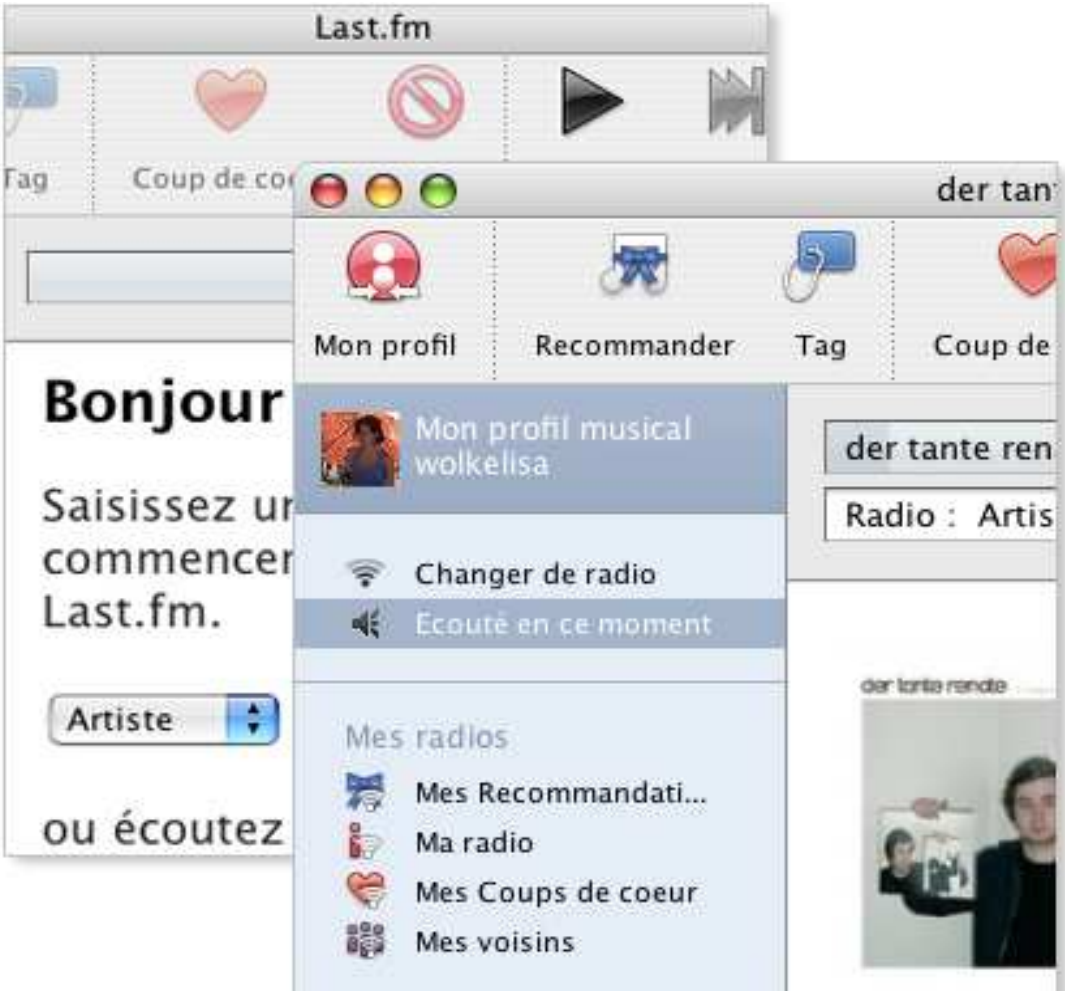
API LastFm

LastFm est un site internet proposant l'écoute de web-radios générées suivant les demandes et goûts de l'utilisateur. Il permet de choisir des morceaux de musique suivant le style (rock, jazz, classique...), la ressemblance à un artiste, les préférences de l'utilisateur si celui-ci possède un compte, etc.

Pour distinguer les différents morceaux de musique, LastFm utilise la technologie AudioScrobbler. Celle-ci se charge de 'tagger' les différents morceaux et artistes, leur attribuant des mots-clefs (les tags) et un coefficient indiquant la correspondance à ce tag. Ce qui permet au final d'obtenir une base de données contenant les informations sur les morceaux et les artistes.

Un des intérêts principaux est de générer pour tout élément (style de musique, morceau, artiste…) la liste de ses proches, avec les correspondances chiffrées. On peut donc ainsi naviguer à partir d'un artiste connu pour découvrir les artistes similaires, et ainsi élargir son horizon musical. C'est ainsi que sont créées les radios écoutées sur LastFm.

Les informations sont fournies par les utilisateurs, qui peuvent tagger une chanson ou un artiste. On peut ainsi obtenir beaucoup d'informations en peu de temps sur une vaste base de musiques. LastFm, par son site AudioScrobbler.com, a choisi d'autoriser l'accès à ces données au moyen de services web, et fournit une API simple et complète pour cela.

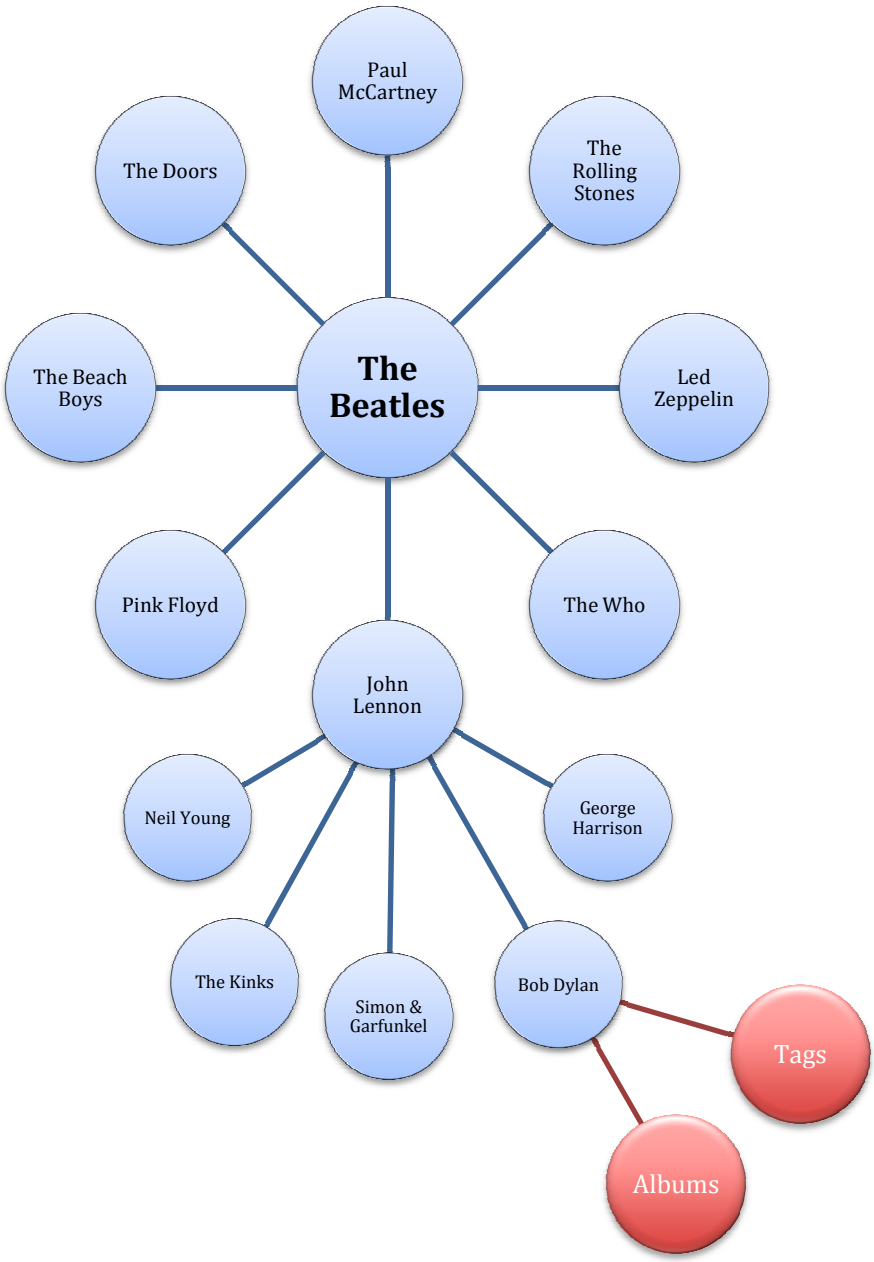


Objectifs

L'objectif de notre projet est d'exploiter ces données pour fournir une navigation directe et intuitive parmi les artistes et styles de musiques. L'utilisateur pourra ainsi découvrir de nouveaux artistes et des morceaux de musique originaux suivant ses goûts musicaux, et obtenir des informations sur ce qu'il souhaite.

Notre projet comporte deux parties fondamentales. La première partie consiste à développer un utilitaire destiné à récupérer un ensemble d'informations à partir des services web d'AudioScrobbler, afin de constituer une base sémantique représentative des musiques existantes. L'API mise à disposition ne permet pas de récupérer les données de façon globale, mais seulement de façon ponctuelle : on peut par exemple récupérer à partir d'un artiste précis ses artistes proches, ses tags, ses albums... mais pas l'ensemble des artistes existants. Nous avons donc décidé d'utiliser un parcours en largeur sur le graphe des artistes proches afin de constituer une liste d'artistes, puis pour chaque récupérer ses tags et ses albums. Cette méthode permet de constituer rapidement une base de connaissances à partir d'un nom d'artiste

La seconde partie de notre projet consiste en un site web permettant l'accès et la navigation parmi les données récupérées. Ce site, basé sur le moteur sémantique Sewese, devait respecter un certain nombre de contraintes que nous nous étions fixées. Tout d'abord, nous souhaitions que le site soit ergonomique afin qu'il puisse être ludique pour l'utilisateur. Nous tenions également à respecter certaines normes d'accessibilités afin que le site puisse être utilisé par des déficients visuels ainsi que pour favoriser le référencement par les moteurs de recherche.



Graphe de récupération des données

Scénarios d'utilisation

Comme nous l'avons dit précédemment, nous avons décidé de faire en sorte que le site puisse offrir la possibilité à l'utilisateur de réaliser un voyage dans la musique. Nous voulions qu'il puisse facilement naviguer entre les styles de musique et les artistes.

Pour cela, nous avons décidé d'afficher un nuage de tags en première page représentant les tags les plus utilisés dans notre base. Un clic sur un tag permet d'afficher la liste des artistes qui correspondent le plus au tag en question, avec la photo du groupe si elle est disponible. Enfin, un clic sur un artiste affiche la page de l'artiste lui-même qui est divisée en trois parties : la liste de ses albums, un nuage des tags correspondant à l'artiste et aussi, les artistes qui ont un style proche de celui de l'artiste.

Il est à noter qu'il est possible de revenir sur le nuage de tags du départ à tout moment en cliquant sur le logo du site.

Enfin, nous avons ajouté un champ de recherche dans la bannière qui permet à l'utilisateur de faire une recherche parmi les tags et les artistes, ce qui lui permet d'accéder rapidement à l'information s'il le souhaite.

Déroulement

Démarche

Notre projet a comporté deux étapes :

Le Crawler

La première étape de notre projet a été de développer la partie de récupération des données, baptisée crawler. Pour cela, nous avons tout d'abord défini le schéma des données que nous allions exploiter au moyen d'un fichier RDFS. Celui-ci nous a servi de base et de modèle pour la suite.

Nous avons ensuite partagé le travail entre nous, Camille s'occupant de coder le cœur du crawler (fonctionnement et récupération des informations utiles au crawler, entre autre les noms des artistes) et Maximilien s'occupant de la transformation des fichiers XML reçus en fichiers RDF par le biais de feuilles de styles XSLT. Nous avons au cours de cette partie affiné le schéma utilisé, afin d'améliorer son efficacité et de rajouter certaines données non initialement prévues.

Cette partie n'a pas posé de difficulté technique particulière. Le fonctionnement du crawler est simple : à partir du nom d'un artiste, il récupère les informations utiles (artistes proches, albums, tags), puis recommence avec la liste des nouveaux artistes ainsi découverts, etc.

Les transformations XSLT sont également simples : les fichiers XML récupérés depuis AudioScrobbler sont très proches du format RDF, et les transformations ont donc été rapides.

http://ws.audioscrobbler.com/1.0/artist/The+Beatles/similar.xml

```
<similarartists artist="The Beatles" streamable="1"
picture="http://userserve-ak.last.fm/serve/160/880929.jpg"
mbid="b10bbbfc-cf9e-42e0-be17-e2c3e1d2600d">
     <artist>
          <name>John Lennon</name>
          <mbid>4d5447d7-c61c-4120-ba1b-d7f471d385b9</mbid>
          <match>100</match>
          <url>http://www.last.fm/music/John+Lennon</url>
          <image_small>http://userserve-
ak.last.fm/serve/50/171806.jpg</image_small>
          <image>http://userserve-
ak.last.fm/serve/160/171806.jpg</image>
          <streamable>1</streamable>
     </artist>
     <artist>
          <name>The Rolling Stones</name>
          <mbid>b071f9fa-14b0-4217-8e97-eb41da73f598</mbid>
          <match>79.97</match>
          <url>http://www.last.fm/music/The+Rolling+Stones</url>
          <image_small>http://userserve-
ak.last.fm/serve/50/8139.jpg</image_small>
          <image>http://userserve-ak.last.fm/serve/160/8139.jpg</image>
          <streamable>1</streamable>
     </artist>
</similarartists>
```

The+beatles relatedArtists.rdf

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns="http://www.polytech.unice.fr/semanticfm#">
     <Artist rdf:about="The Beatles">
          <name>The Beatles</name>
          <pic>http://userserve-ak.last.fm/serve/160/880929.jpg</pic>
          <similar>
               <SimilarArtist>
                    <artist>
                         <Artist rdf:about="John Lennon"/>
                    </artist>
                    <match>100</match>
               </SimilarArtist>
          </similar>
          <similar>
               <SimilarArtist>
                    <artist>
                         <Artist rdf:about="The Rolling Stones"/>
                    </artist>
                    <match>79.97</match>
               </SimilarArtist>
          </similar>
     </Artist>
</rdf:RDF>
```

Le site

La seconde partie du projet a consisté à développer le site web, interface destinée à la navigation parmi les données collectées.

Nous souhaitons tout d'abord que celle-ci soit ergonomique. Nous avons essayé de la rendre intuitive en la testant à plusieurs reprises avec des amis. Nous avons à chaque fois pris en compte leurs remarques quand elles étaient réalisables dans le temps que nous avions. Nous avons aussi opté pour des techniques héritées du web 2.0 comme les nuages de tags pour faciliter la navigation. Nous avons aussi mis en place un système de recherche accessible à partir de toutes les pages.

Une autre exigence de notre part était que le site soit accessible. Pour cela nous avons respecté certaines normes comment celle du XHTML 1.0 Transitionnal. Nous avons également optimisé un minimum nos pages pour le référencement en faisant attention quant à l'utilisation des balises de titre (h1, h2, ...), des balises strong, des balises méta, du titre des pages, ...

Enfin, nous souhaitions que le site soit plus compatible possible avec les navigateurs modernes. Pour cela, nous avons utilisé un framework CSS. Celui-ci nous a permis de rapidement mettre en place le design que nous voulions avec le bon nombre de colonnes. Le framework CSS propose en fait plusieurs feuilles CSS qui permettent de faire un reset de tous les styles HTML de base, de mettre en place un design cross-browser et des patches pour Internet Explorer afin d'améliorer la compatibilité.

Problèmes rencontrés

Pas d'accès à une liste des artistes

Le premier problème que nous avons rencontré a été que l'API d'AudioScrobbler, bien que très complète pour une recherche précise, ne permet pas d'accéder à l'ensemble des données rapidement. Si l'on peut obtenir très simplement la liste des albums d'un artiste, on ne peut en revanche pas obtenir la liste des artistes existants. Cette difficulté nous a forcés à rechercher une méthode de parcours de la base de LastFm, afin de récupérer autrement les données nous intéressant. La solution que nous avons adoptée a été de nous baser sur la liste des artistes proches, partant d'un artiste précis et rayonnant ainsi de proche en proche pour constituer la liste des artistes existants. Une fois cette liste créée, on peut très simplement récupérer les tags associés à chaque, ainsi que leurs albums.

Risques d'incomplétude des données

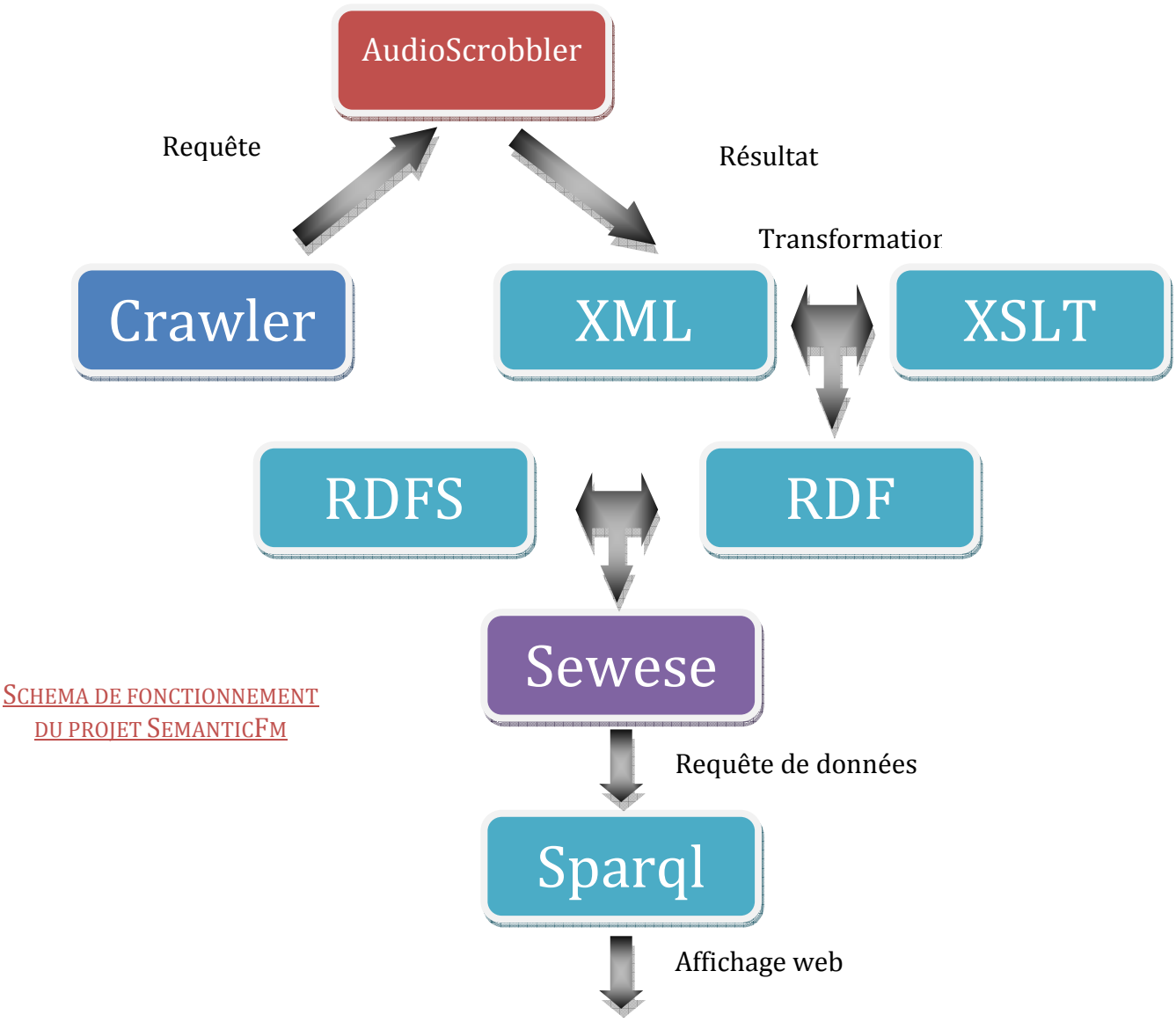
Le second problème que nous avons envisagé a été le risque que notre stratégie de récupération de données ne donne des résultats incomplets. En effet, s'étendre en cercles concentriques à partir d'un artiste peut ne couvrir qu'un domaine limité autour de cet artiste, sans atteindre les artistes plus lointains et laissant ainsi des pans entiers non couverts. Mais après avoir testé cette méthode, nous nous sommes aperçus que quelques générations seulement suffisent pour atteindre des points très éloignés. Nos tests ont montré que les connexions entre les artistes sont suffisamment riches pour ne pas risquer de manquer certaines données. De plus, lancer une seconde session à partir d'un autre artiste permet d'agrandir encore le champ de recherches s'il est besoin.

Limites de Corese

Un troisième problème que nous avons rencontré a été les limites techniques du moteur sémantique. En effet, on note un ralentissement certain sur les requêtes complexes lorsque les données deviennent importantes. Par exemple, une de nos requêtes recherche les occurrences des tags dans la base ; avec sept cent artistes en base, son exécution demande vingt secondes. Cela nous impose de limiter les données enregistrées dans l'état actuel de l'application.

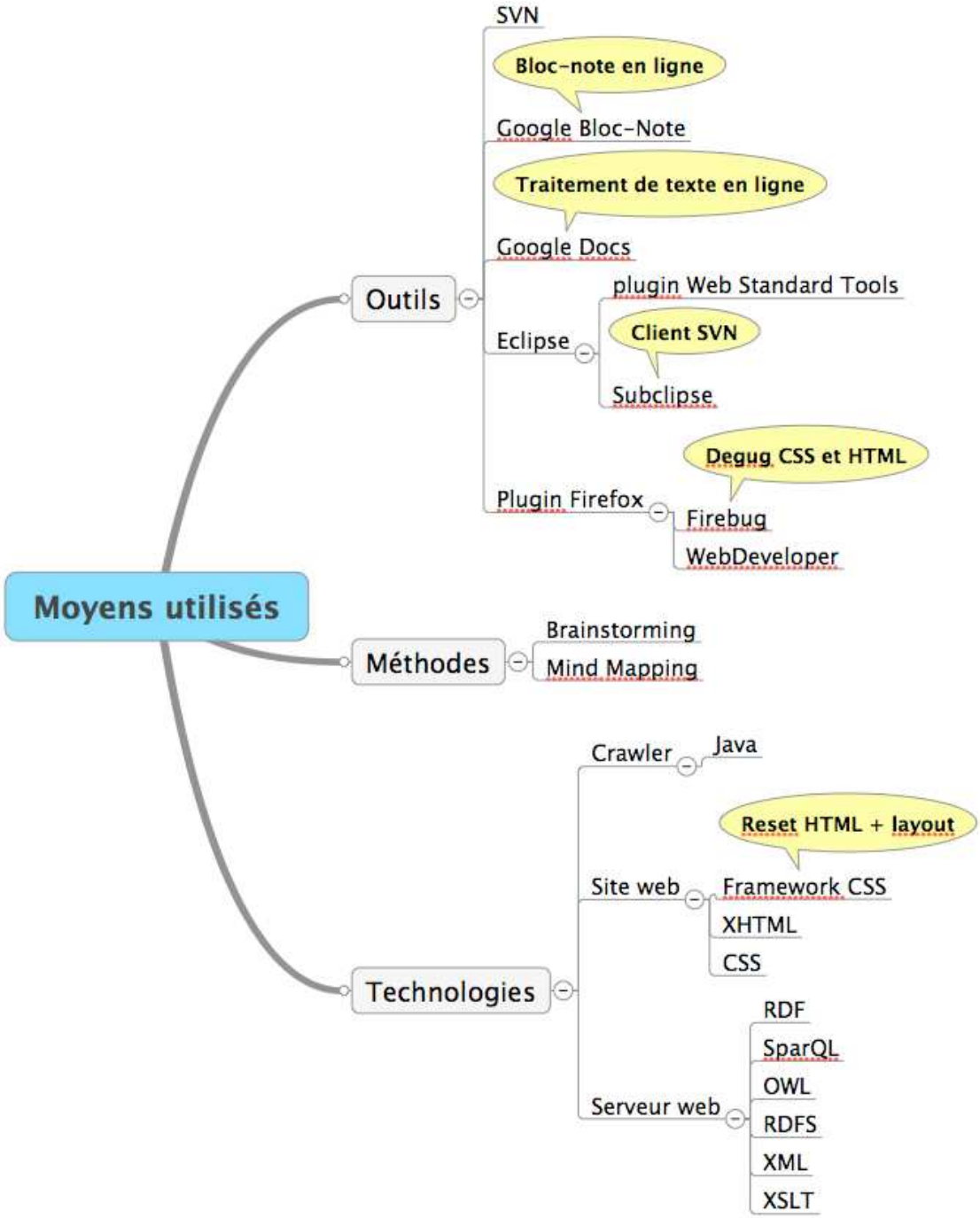
Resultat final

Fonctionnement



SCHEMA DE FONCTIONNEMENT DU PROJET SEMANTICFM

Outils et technologies utilisés



Interêt du web sémantique

Notre projet utilise fortement les outils du web sémantique. Nous transformons les XML reçu grâce à l'API de Last.FM en RDF que nous ajoutons ensuite au serveur web sémantique Sewese.

Ce moteur nous a permis de manipuler des objets ayant des relations complexes entre eux (formant des graphes) avec une facilité déconcertante. Le concept de graphe

est plutôt étranger aux bases de données relationnelles alors que cette notion est très naturelle pour les serveurs web sémantique.

Ajouts possibles

Plus d'informations

Un des apports principaux qu'il resterait à développer serait l'ajout d'informations supplémentaires dans la base et le site. En effet, l'API d'AudioScrobbler permet l'accès à de très nombreux domaines que nous n'avons pas eu le temps d'exploiter. Comme par exemple, les titres d'un album, ou les concerts prévus dans une région donnée. Ces informations sont très nombreuses et accessibles, et participeraient grandement à l'enrichissement de l'application.

Lien avec le profil LastFm

Dans le même contexte, il pourrait être intéressant d'incorporer les données des comptes utilisateurs de LastFm, afin de permettre à une personne possédant un de ces comptes de retrouver directement ses goûts. Les informations de ces comptes sont directement accessibles par l'API, mais nécessitent un traitement avant d'être exploitables.

Intégration du player

Une autre fonctionnalité conséquente, initialement prévue dans le développement mais supprimée faute de temps, est l'ajout du player LastFm dans les pages web. Ce player consiste en un objet flash permettant de jouer un morceau de musique précis, un extrait, ou une radio spécifiée (artiste, tag, etc.). Cette amélioration permettrait de se faire une idée concrète des découvertes.

Conclusion

Ce projet nous a permis de mettre en œuvre toutes les connaissances acquises durant le cours. Nous avons pu confirmer les idées que nous avions au départ au sujet du web sémantique. Nous avons pu grâce à Sewese, développer rapidement et simplement un site internet basé sur des objets aux relations complexes.

Cette option nous a permis de nous conforter dans l'idée que le web 3.0 et plus précisément le web sémantique va sans aucun doute marquer fortement l'avenir de l'internet. Ces concepts deviendront incontournables vu que la quantité de connaissances sur le web croit de manière exponentielle.

Annexes

Fichier RDFS global :

```
      <?xml version="1.0" encoding="utf-8"?>

      <!DOCTYPE rdf:RDF [
      <!ENTITY cos   "http://www.inria.fr/acacia/corese#">
      <!ENTITY rdf   "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
      <!ENTITY rdfs   "http://www.w3.org/2000/01/rdf-schema#">
      <!ENTITY xsd    "http://www.w3.org/2001/XMLSchema#">
      <!ENTITY owl    "http://www.w3.org/2002/07/owl#">
      <!ENTITY sfm   "http://www.polytech.unice.fr/semanticfm#">
      ]>

<rdf:RDF     xml:base="&sfm;"     xmlns:rdfs="&rdfs;"     xmlns:rdf="&rdf;"
      xmlns:cos="&cos;"
      xmlns:owl='&owl;'  >

      <rdfs:Class rdf:ID='Artist'/>
      <rdfs:Class rdf:ID='Tag'/>
      <rdfs:Class rdf:ID='Album'/>


      <rdf:Property rdf:ID='name'>
            <domain rdf:resource="#Artist"/>
      </rdf:Property>

      <rdf:Property rdf:ID='pic'>
            <domain rdf:resource="#Artist"/>
      </rdf:Property>

      <rdf:Property rdf:ID='tagged'>
            <domain rdf:resource="#Artist"/>
      </rdf:Property>

      <rdf:Property rdf:ID='similar'>
            <domain rdf:resource="#Artist"/>
      </rdf:Property>


      <rdf:Property rdf:ID='tagName'>
            <domain rdf:resource="#Tag"/>
      </rdf:Property>

      <rdf:Property rdf:ID='url'>
            <domain rdf:resource="#Tag"/>
      </rdf:Property>


      <rdf:Property rdf:ID='albumName'>
            <domain rdf:resource="#Album"/>
      </rdf:Property>

      <rdf:Property rdf:ID='url'>
            <domain rdf:resource="#Album"/>
      </rdf:Property>

      <rdf:Property rdf:ID='pic'>
            <domain rdf:resource="#Album"/>
      </rdf:Property>

      <rdf:Property rdf:ID='reach'>
            <domain rdf:resource="#Album"/>
      </rdf:Property>

</rdf:RDF>
```

Fichier XSLT SimilarArtists.xsl

```
<?xml version="1.0" encoding="utf-8"?>

<xsl:stylesheet version="1.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:output method='xml' indent='yes' />
<xsl:template match='similarartists'>

<rdf:RDF xmlns="http://www.polytech.unice.fr/semanticfm#"
            xmlns:rdf ="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
            xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
      <Artist>
            <xsl:attribute name='rdf:about'>
                  <xsl:value-of select="@artist"/>
            </xsl:attribute>
            <name><xsl:value-of select="@artist"/></name>
            <pic><xsl:value-of select="@picture"/></pic>

            <xsl:for-each select='artist'>
                  <similar>
                        <SimilarArtist>
                              <artist>
                                    <Artist>
                                          <xsl:attribute
name='rdf:about'>
                                                <xsl:value-of
select="name"/>
                                          </xsl:attribute>
                                    </Artist>
                              </artist>
                              <match><xsl:value-of
select="match"/></match>
                        </SimilarArtist>
                  </similar>
            </xsl:for-each>
      </Artist>
</rdf:RDF>

</xsl:template>
</xsl:stylesheet>
```

Glossaire

RDF Resource Description Framework. Modèle de description de données destiné au web sémantique.

RDFS : RDF Schema. Langage de définition d'ontologies destiné à structurer des ressources RDF.

OWL Web Ontology Language. Extension de RDF et RDFS permettant d'étendre l'expression d'ontologies.

SPARQL langage de requête sur une base de données RDF.

XSLT : eXtensible Stylesheet Language Transformations. Langage de transformation de XML.

API : Application Programming Interface. Spécification des interactions entre deux composants informatiques.

last.fm site internet de musique en ligne proposant des radios suivant divers critères.

Powered by Audioscrobbler AudioScrobbler : technologie permettant de trier des notions musicales (artistes, morceaux, genres…) et de les relier entre elles.

Corese : section sémantique de Sewese.

Sewese : SErveur WEb SEmantique.

Cross-Browser : dit d'un site pouvant être affiché sur tous les navigateurs de façon identique.