Systèmes d'Information Web

Pokédata

Projet

AHMED Sahim – GOODWIN Jean-Paul
18 décembre 2017

Contents

I – Introduction

Chaque binôme travaillera sur une application Web qui combine et présente de manière innovante (ex. des monuments projetés sur un fond cartographique de type Google Maps ou OpenStreetMap) des données d'une ou de plusieurs sources du Web de données (DBpedia, YAGO, Wikidata, etc). Cette application doit récupérer ces données via des requêtes SPARQL ou, exceptionnellement, via des APIs (dans le cas où c'est le seul moyen d'accès à ces données).

L'application, se présentant sous la forme d'un site web, aura pour but de répertorier l'ensemble des Pokémons. Un Pokémon est une créature imaginaire tiré du jeu vidéo du même nom. Les différents Pokédex auxquels appartiennent ces Pokémons seront aussi référencés. Un Pokédex est une librairie virtuelle recensant des Pokémons, ils peuvent porter sur la totalité des Pokémons ou bien sur les Pokémons autochtones d'une région donnée.

II – Cahier des charges

L'application doit se présenter sous forme d'un site web responsive. L'utilisateur n'a pas besoin de s'authentifier. Le site web présentera une page d'accueil présentant le projet et l'équipe. Les utilisateurs peuvent naviguer sur la page depuis n'importe quel terminal, par exemple, un ordinateur, un smartphone, une tablette. Les données du site web devront être récupérées de l'Open Linked Data (DBpedia, Wikidata…).

L'application Web devra respecter les exigences techniques suivantes :

- Implémentation de l'architecture MVC
- Application C# .NET avec ASP.NET Core
- Récupération de données à l'aide de requêtes SPARQL
- Accès à l'Open Linked data, (Wikidata, DBpedia ou autres...)

L'application devra présenter les Pokédex de la série des jeux Pokémon. Un Pokédex a un nom et peux être accompagné d'un commentaire texte. Chaque Pokédex devra quant à lui permettre d'afficher les Pokémons qu'il recense classés par leurs numéros de Pokédex. Un Pokémon peut avoir plusieurs numéros de Pokédex s'il est recensé dans différents Pokédex. Un Pokémon a un nom et appartient à une famille de type de Pokémon. Un Pokémon a obligatoirement un type mais peut en avoir plusieurs. Un Pokémon a une taille, un poids, une couleur et peut être associé à une image et un commentaire.

III – Diagrammes et conception

1 - Diagramme de cas d'utilisations

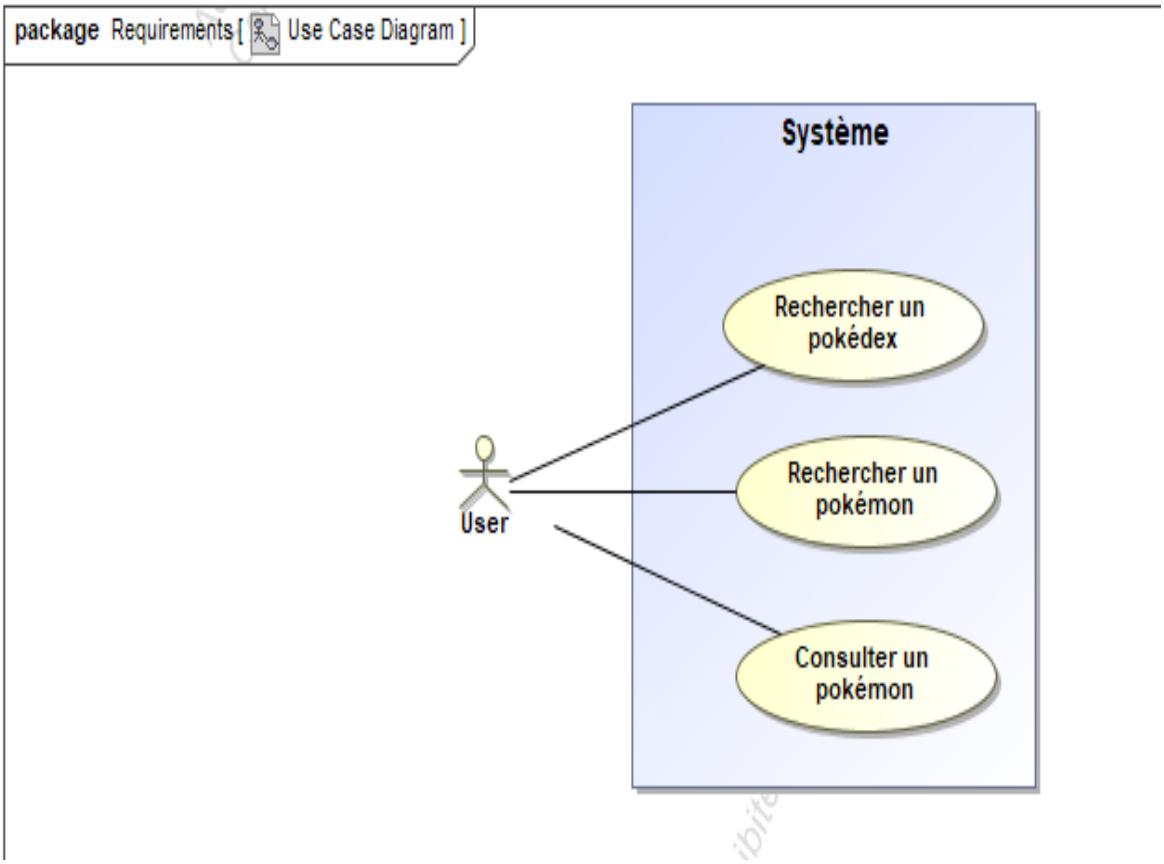


Figure 1 : Diagramme de cas d'utilisation

Un utilisateur peut rechercher un Pokédex, et rechercher un Pokémon. L'utilisateur peut aussi consulter un Pokémon afin de voir les caractéristiques de celui-ci.

2 - Diagramme de classes

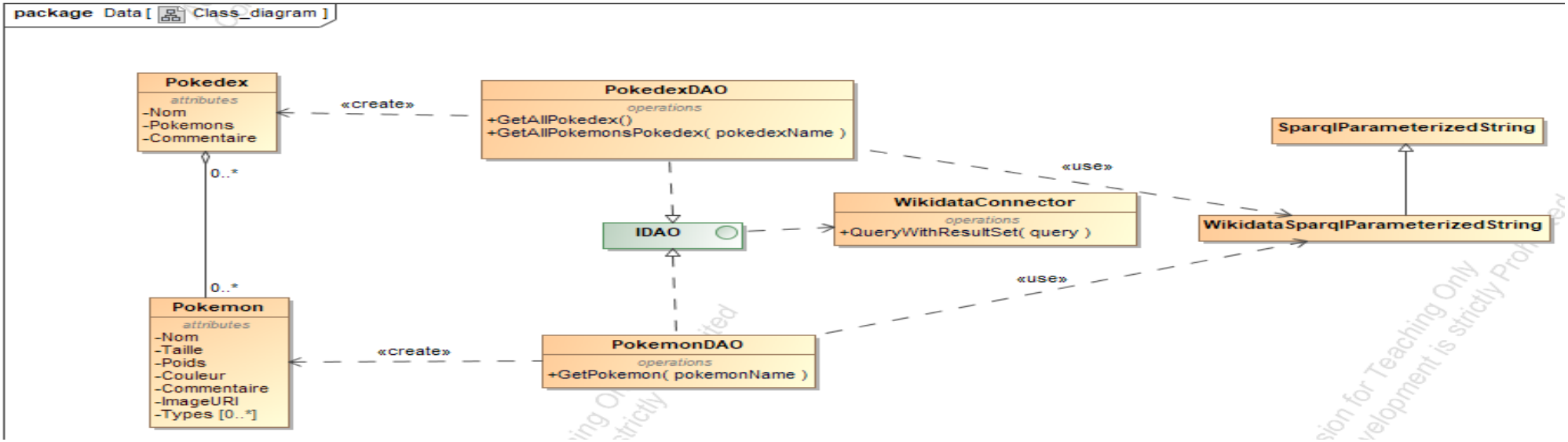


Figure 2 : Diagramme de classe

Un Pokédex a un nom et un commentaire. À un Pokédex est lié une liste de Pokémons. Un Pokémon a un nom, une taille, un poids, une couleur, un commentaire, une image (sous forme d'URI) et au moins un type (sauf s'il est inconnu). Un Pokédex est créé à l'aide d'un DAO (Data Access Object) appelé PokedexDAO. Ce DAO permet de récupérer tous les Pokédex ou bien tous les Pokémons d'un Pokédex donné. Un Pokémon est créé lui aussi à l'aide d'un DAO appelé PokemonDAO. Ce DAO permet de récupérer les données d'un Pokémon donné. Ces DAO dépendent d'un WikidataConnector afin de récupérer les données et utilisent une WikidataSparqlParameterizedString héritant de la classe SparqlParameterizedString (importée de la librairie dotnetrdf). La classe WikidataConector permet d'accéder directement au endpoint SPARQL Wikidata. La classe WikidataSparqlParameterizedString permet de fixer les préfixes demandés pour les requêtes sur le endpoint Wikidata avant de spécifier directement la requête SPARQL.

3 - Diagramme de Navigation

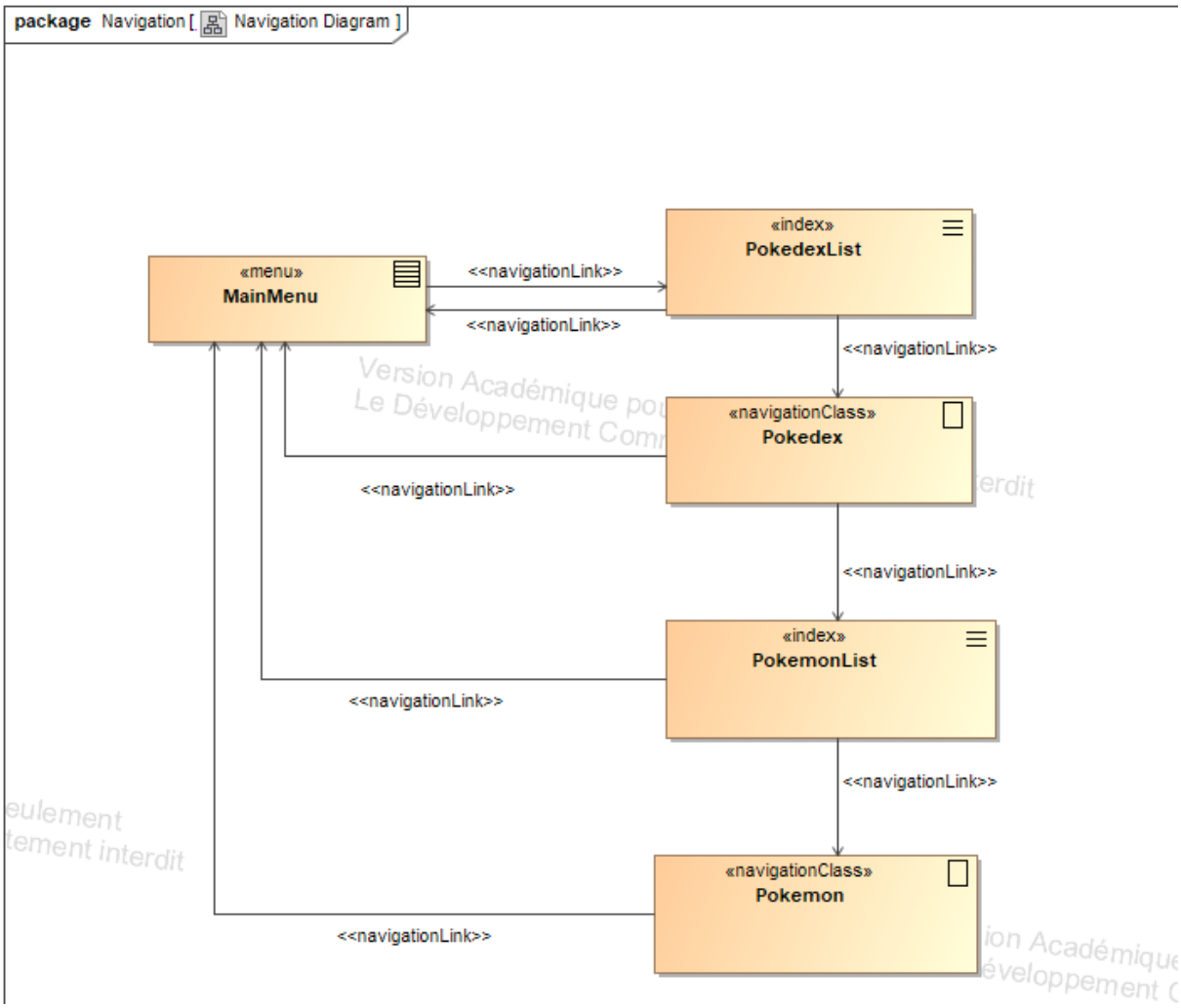


Figure 3 : Diagramme de navigation

L'utilisateur peut accéder à un menu principal. Ce menu principal permet d'accéder seulement à une liste de Pokédex. La liste de Pokédex est un index de Pokédex.

Une fois sur cet index, l'utilisateur peut sélectionner le Pokédex de son choix.

La sélection d'un Pokédex donne accès à un index de Pokémons.

4 - Diagramme de présentation

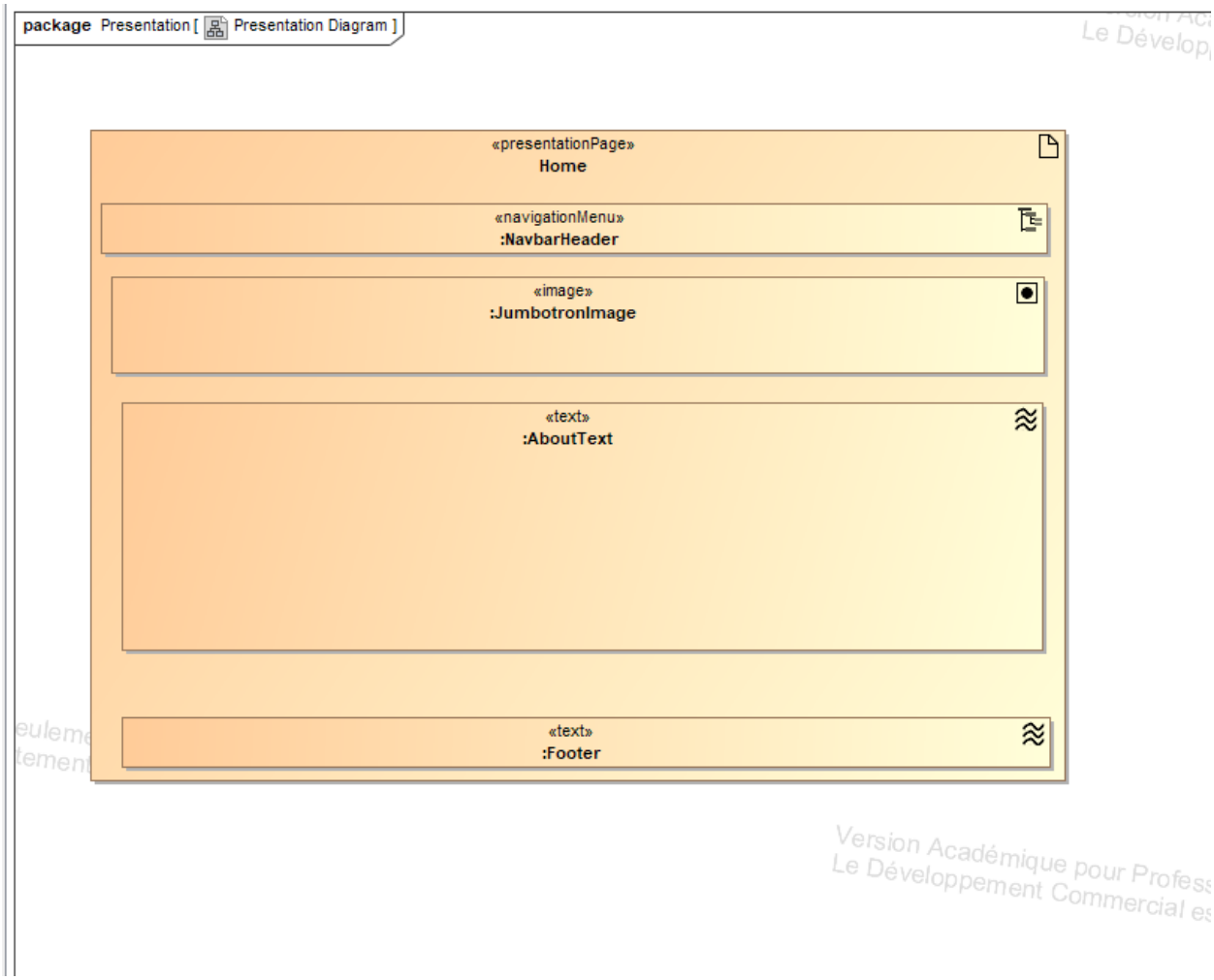
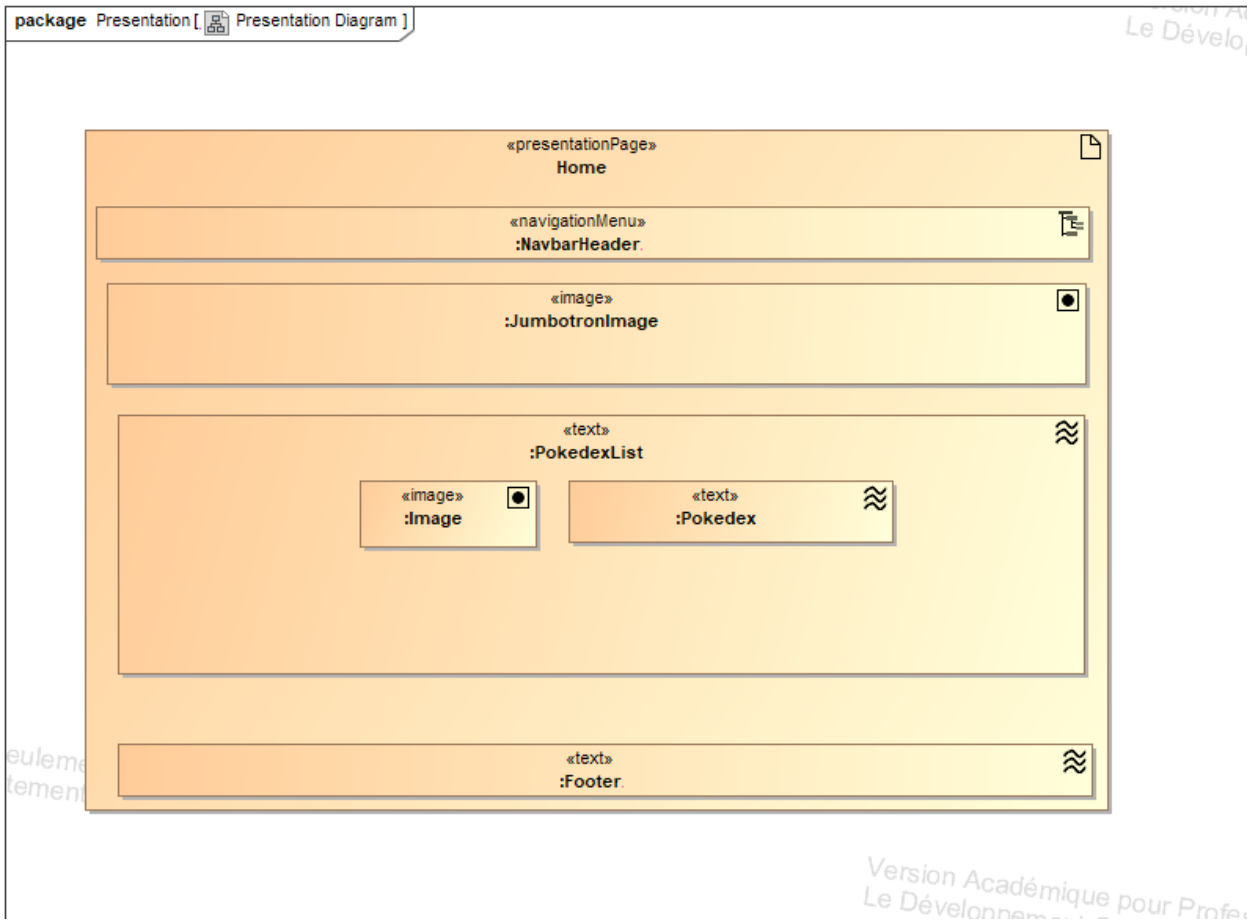


Figure 4 : Diagramme de présentation de la page d'accueil

Une page d'accueil simple, avec la présentation du projet dans le contenu au centre (About Text).



<u>Figure 5</u> : Diagramme de présentation de la page liste de Pokédex

IV - Développement

1 – Requête SPARQL

```
PREFIX dbo:<http://dbpedia.org/ontology/>
SELECT DISTINCT ?typelabel ?pokemonName ?height ?weightKg ?colorLabel ?comment ?image WHERE {
  ?pokemontype wdt:P279* wd:Q3966183;
    rdfs:label ?typelabel.
  ?pokemon wdt:P31 ?pokemontype;
    rdfs:label ?pokemonName.
  FILTER(CONTAINS(?typelabel, 'type')).
  FILTER(LANG(?typelabel) = 'fr').
  FILTER(LANG(?pokemonName) = 'fr').
  FILTER(CONTAINS(?pokemonName, 'Pikachu')).
  OPTIONAL{?pokemon wdt:P2048 ?height}.
  OPTIONAL{?pokemon p:P2067/psv:P2067 [
    wikibase:quantityAmount ?weightKg;
    wikibase:quantityUnit [
      p:P2370/psv:P2370 [
        wikibase:quantityAmount ?unitMass;
        wikibase:quantityUnit wd:Q11570
      ]
    ]
  ].
  BIND(?baseMass * ?unitMass AS ?mass).
  MINUS { ?object wdt:P31 wd:Q3647172. }.
           FILTER(?unitMass = 1 ).
  }
  OPTIONAL{?pokemon wdt:P462 ?color.
          ?color rdfs:label ?colorLabel.
          FILTER(LANG(?colorLabel)= 'fr')}.
  OPTIONAL{
    SERVICE <http://dbpedia.org/sparql> {
      ?pokemonDbpedia owl:sameAs ?pokemon;
                      rdfs:comment ?comment;
                      dbo:thumbnail ?image.
      FILTER(LANG(?comment)='fr').
      }
    }.
}
```

Cette requête SPARQL permet de récupérer les informations du Pokémon Pikachu.

La requête récupèrera d'abord tous les types de Pokémons avant de récupérer tous les Pokémons appartenant à ces types. On filtrera les résultats en français et on vérifiera que le Pokémon soit Pikachu. On récupèrera ensuite sa taille si elle existe et sa masse (en kilogramme). On récupèrera aussi sa couleur si elle est disponible. Enfin, si une page DBpedia existe et correspond à ce

Pokémon, on y accèdera à l'aide d'une requête fédérée afin de récupérer un commentaire en français et une image.

2 - Classe PokemonDAO

```
public Pokemon GetPokemon(String pokemonName) {
    WikidataConnector connector = WikidataConnector.Instance;
    WikidataSparqlParameterizedString queryString = new WikidataSparqlParameterizedString();
    queryString.CommandText = GetPokemonQueryString;
    queryString.SetLiteral("pokemonName", pokemonName);

    SparqlResultSet rset = connector.QueryWithResultSet(queryString.ToString());

    return BuildPokemon(rset);
}
```

Cette méthode, située dans la classe PokemonDAO permet de récupérer un Pokémon. On récupèrera le WikidataConnector (singleton) et on créera une nouvelle requête paramétrable (WikidataSparqlParameterizedString). On placera alors la requête SPARQL dans l'objet avant de changer le paramètre "pokemonName" par le nom du Pokémon souhaité. On exécutera ensuite la requête dans le connecteur Wikidata avant de créer le Pokémon à l'aide de ce résultat.

3 - Classe WikidataConnector

```
public class WikidataConnector{
    private static WikidataConnector instance;

    private SparqlRemoteEndpoint endpoint;

    private WikidataConnector() {
        endpoint = new SparqlRemoteEndpoint(new Uri("https://query.wikidata.org/sparql"), "http://wikidata.org");
    }

    public static WikidataConnector Instance {
        get {
            if (instance == null) {
                instance = new WikidataConnector();
            }
            return instance;
        }
    }

    public SparqlResultSet QueryWithResultSet(String query) {
        return (SparqlResultSet)endpoint.QueryWithResultSet(query);
    }
}
```

La classe WikidataConnector est une classe wrapper d'un objet SparqlRemoteEndpoint (importée de la librarie dotnetrdf) permettant d'exécuter des requêtes sur le Endpoint SPARQL de Wikidata. Cette classe est un singleton car représentant un endpoint unique et un objet n'ayant pas la nécessité d'être instantié de multiples fois.

4 – Classe WikidataSparqlParameterizedString

```
public class WikidataSparqlParameterizedString : SparqlParameterizedString {
    public WikidataSparqlParameterizedString () : base() {
        //Add namespace declaration
        this.Namespaces.AddNamespace("wd", new Uri("http://www.wikidata.org/entity/"));
        this.Namespaces.AddNamespace("wdt", new Uri("http://www.wikidata.org/prop/direct/"));
        this.Namespaces.AddNamespace("rdfs", new Uri("http://www.w3.org/2000/01/rdf-schema#"));
        this.Namespaces.AddNamespace("wikibase", new Uri("http://wikiba.se/ontology#"));
        this.Namespaces.AddNamespace("p", new Uri("http://www.wikidata.org/prop/"));
        this.Namespaces.AddNamespace("ps", new Uri("http://www.wikidata.org/prop/statement/"));
        this.Namespaces.AddNamespace("pq", new Uri("http://www.wikidata.org/prop/qualifier/"));
        this.Namespaces.AddNamespace("dbo", new Uri("http://dbpedia.org/ontology/"));
    }
}
```

La classe WikidataSparqlParameterizedString est une classe héritant de la classe SparqlParameterizedString (importée de la librairie dotnetrdf). Son seul but est de poser les préfixes sur chacune de nos requêtes afin de ne pas avoir à les déclarer à chaque fois.

V – Problèmes rencontrés et solutions

1 - Problème d'importation de librairie

Il nous a été très difficile d'importer la librairie dotnetrdf qui nous avait été conseillée. En effet, la librairie ne fonctionne pas nativement avec le framework ASP .Net Core et nous avons donc été obligé après des heures de recherches infructueuses d'importer les frameworks manquants.

2 - Problème de structure sur DBpedia

Au début du projet nous étions plus à l'aise sur la plateforme DBpedia. Cependant, les Pokémons sont classés comme une entité de type ImaginaryBeings au même titre que les Looney Tunes, Mickey et ses amis. Nous avons donc exploré Wikidata et avons trouvé que tous les Pokémons étaient recensés (au contraire de DBpedia). De plus, sur Wikidata les Pokémons sont classés comme des entités de type "Pokemon species". Cela nous a permis de récupérer tous les Pokémon recensés.

3 - Manque de données sur Wikidata

Les données de Wikidata sont très structurées, ce qui nous a permis de récupérer plus facilement les données. Cependant, cette structure nous a aussi fait perdre des données utiles présentes sur des datasets comme DBpedia. Nous avons donc dû faire un lien entre nos données collectées sur Wikidata et les données présentes sur DBpedia grâce à des requêtes fédérées sur le endpoint de WIkidata. Les données que nous avons alors pu récupérer ainsi ont été les images des Pokémon et des commentaires en langage naturel.

VI - Conclusion

Le projet nous a permis de connaître et pratiquer des nouveaux langages, aussi bien en matière de programmation, qu'au niveau des requêtes SPARQL. Il nous a permis d'utiliser ces nouveaux aspects du web 3.0, tel que le requêtage sur des bases de données liées publiques. Nous avons pu aussi renforcer notre pratique du modèle MVC appliqué au web.

Nous pouvons apercevoir quelques limites tout de même, au niveau de l'apprentissage. En effet, il nous a fallu consulter pas mal de tutoriels sur internet afin d'être à l'aise avec les technologies utilisées. Cette courbe d'apprentissage pouvait prendre beaucoup de temps par rapport au temps qu'on avait pour le projet.

Bibliographie

- Nicolas HILAIRE. Apprenez ASP. NET MVC - OpenClassrooms [en ligne]
  Disponible sur https://openclassrooms.com/courses/apprendre-asp-net-mvc

- dotnetrdf. UserGuide . dotnetrdf/dotnetrdf Wiki [en ligne]
  Disponible sur https://github.com/dotnetrdf/dotnetrdf/wiki/UserGuide