Titre : *Développement d'une plateforme serveur (back-end) pour la collecte et la gestion de données issues de différentes sources.*

Cours	Technologies de l'inforoute (INF37407)
Session	Automne 2025
Contact	yacine_yaddaden@uqar.ca

Table des matières

1. Objectif du travail

L'objectif de ce premier travail pratique est de développer une plateforme côté serveur permettant de collecter et de récupérer des données à partir de différentes sources. Ces données devront ensuite être organisées et structurées avant d'être stockées dans une base de données. Par la suite, elles seront rendues accessibles de manière sécurisée via une API REST ou GraphQL, avec un mécanisme d'authentification. Enfin, une interface graphique simple offrira la possibilité de visualiser certaines informations contenues dans la base de données.

2. Technologies à utiliser

Dans le cadre du projet, il sera nécessaire d'utiliser les technologies suivantes :

- → Environnement de développement :
  - Microsoft Visual Studio Code
- → Langages de programmation :
  - Python
- → Bibliothèques Python :
  - Django (*Framework principal*)
  - Django REST Framework (*conception d'API REST*)
  - SWAGGER ou **drf-yasg** (*documentation et test*)
  - Graphene-Django (*mise en place de GraphQL*)
  - GraphiQL (*documentation et test*)
- → Interface graphique :
  - Twitter Bootstrap 5 (*Framework CSS*)
  - Font Awesome (*icônes*)
- → Logiciels :
  - Postman
  - MySQL Server
  - MySQL Workbench

Il est recommandé de faire de la *gestion de version* avec l'outil **Git**. Afin de faciliter le travail d'équipe, il est recommandé de créer un dépôt **GitHub**.

INFORMATION : il est possible d'utiliser d'autres paquets si c'est jugé pertinent pour la réalisation du projet.

3. Description détaillée

Afin de réaliser le projet, il est nécessaire de suivre les étapes suivantes :

3.1. Les sources de données

Les quatre catalogues ciblés (OpenGouv, CanWin, Données Québec, Borealis) sont basés sur CKAN et accessibles via une **API**. La première étape consistera à définir les filtres de moissonnage (mots-clés, variables essentielles, programmes, producteurs ou localisation).

☑ **OpenGouv**
- **Métadonnées à récupérer** → Lien

☑ **CanWin**
- **Métadonnées à récupérer** → Lien

☑ **Données Québec**
- **Métadonnées à récupérer** → Lien

☑ **Boréalis**
- **Métadonnées à récupérer** → Lien

Le développement s'appuiera sur **Python** et sera orchestré dans **Visual Studio Code**. Les interactions avec les APIs CKAN seront gérées via des scripts Python intégrés au projet Django.

3.2. Tests & définition des schémas des données

Une phase exploratoire permettra d'identifier la structure des métadonnées disponibles dans chaque catalogue, puis de définir un **schéma de données commun** pour leur intégration dans le catalogue de l'OGSL.

Les tests se feront à l'aide de **Postman** (pour les appels API bruts) et d'outils comme **Swagger/drf-yasg** (pour documenter et tester les endpoints internes du projet).

3.3. Récupération des données

La récupération des données se fera via des scripts Python intégrés au framework **Django**. Les données seront ensuite stockées dans la base de données relationnelle gérée par Django ORM.

Le **Django REST Framework (DRF)** sera utilisé pour gérer la sérialisation, la validation et la normalisation des jeux de données récupérés.

3.4. Interface d'administration

Une interface d'administration sera mise en place à l'aide du **Django Admin** et enrichie avec **Twitter Bootstrap 5** et **Font Awesome** pour offrir une meilleure ergonomie.

Cette interface permettra de visualiser les métadonnées moissonnées, d'éditer les jeux de données et de gérer les paramètres de moissonnage (sources, filtres, planification).

3.5. Mise en place de l'accès aux données – API REST

Une **API REST** sera développée avec **Django REST Framework** pour permettre la diffusion des données moissonnées.

La documentation et le test des *endpoints* seront facilités par **Swagger** ou **drf-yasg**, qui généreront automatiquement une interface interactive pour les développeurs et utilisateurs.

3.6. Mise en place de l'accès aux données – GraphQL

En complément du REST, une **API GraphQL** sera mise en place à l'aide de **Graphene-Django**, permettant des requêtes flexibles et précises sur les métadonnées.

L'interface **GraphiQL** sera intégrée pour documenter et tester ces requêtes de manière interactive.

3.7. Page des statistiques

Une page dédiée, intégrée à l'interface Django et stylisée avec **Bootstrap 5**, affichera des statistiques d'utilisation et de couverture (nombre de jeux moissonnés par catalogue, répartition thématique, tendances temporelles, etc.).

Ces visualisations permettront de valoriser le travail réalisé et d'assurer un suivi quantitatif de l'intégration.

3.8. Sécurisation de l'accès

Des mécanismes de **sécurité et permissions** seront mis en place pour contrôler l'accès aux données et aux fonctionnalités d'administration. Django

offre déjà un système robuste d'authentification et de gestion des rôles, qui sera renforcé au besoin (e.g. authentification par jeton pour les API REST et GraphQL).

3.9. Déploiement de l'API REST – Bonus

À titre d'étape bonus supplémentaire, il est demandé de déployer l'**API REST** réalisée sur une plateforme gratuite : https://render.com/

4. Structure du projet

Pour la structure du projet, il est demandé d'utiliser l'outil en ligne de commande pour créer le projet Django et ses différentes applications. En effet, afin de mieux structurer le projet, il est recommandé de séparer les différentes parties en applications distinctes. Cela facilitera le processus de développement.

5. Modalité d'évaluation

Le travail pratique comptera pour **30%** de la note du cours. Ces points sont répartis de la manière suivante :

Partie	Points
Tests & définition des schémas des données	2.0%
Récupération des données	4.0%
Interface d'administration	3.0%
Mise en place de l'accès aux données – API REST	5.0%
Mise en place de l'accès aux données – GraphQL	5.0%
Page des statistiques	3.0%
Sécurisation de l'accès	3.0%
Déploiement de l'API REST – Bonus	3.0%
Qualité du code source	1,0%
Rapport rédigé en Français	2,0%
Démonstration vidéo de l'application	2,0%
Total	**30%**

6. Date de remise

→ La date limite de remise est le **30 octobre 2025 avant 23h00**

→ Fichiers à remettre :

- ✓ Un fichier compressé (.zip) contenant le code source,
- ✓ Le rapport (un modèle **LaTex** est fourni),
- ✓ Une courte vidéo de démonstration faite avec **OBS Studio**.

Tous les fichiers doivent être remis sur Moodle avant la date limite.

Si la vidéo est trop volumineuse, il est possible de la téléverser sur YouTube et d'inclure le lien dans le rapport et en commentaire de la remise.

7. Assistance

Si vous avez besoin d'assistance, il est possible de contacter le professeur directement par courriel afin d'organiser une rencontre ZOOM. Des questions peuvent également être posées à la fin des séances de cours. Si un auxiliaire d'enseignement est disponible, il pourra également vous assister.

8. Points importants

Note I :

→ Le travail est en équipe de deux,

→ Le professeur peut poser des questions liées au travail pratique,

→ Le non-respect de l'énoncé entraînera une perte de points,

→ Le plagiat sera sanctionné selon la politique de l'université,

→ Le retard dans la remise du projet entraînera des pénalités.

Note II : *Dans le cas où il y a des aspects qui ne sont pas clairs, n'hésitez pas à m'en faire part afin que je puisse apporter des éclaircissements et éventuellement mettre à jour l'énoncé du travail pratique.*

9. Documentation

La documentation à utiliser est disponible sur les sites officiels (*voir plus haut*). Vous pouvez également vous référer à celle fournie dans le plan de cours.