Jérémie D’Amours   
Felicya Lajoie Jacob   
Jasmin Dubuc

Techniques de l’informatique   
groupe 4318

Projet 2  
Webscraping et MongoDB Atlas

Rapport présenté à   
M. Nicolas Payre  
Département des Techniques de l’informatique   
pour le cours   
*Exploration de nouvelles technologies*

Cégep de Sherbrooke   
20 octobre 2025

**Table des matières**

[Webscraping 3](#_Toc211798176)

[Description du site et des données choisis 3](#_Toc211798177)

[Présentation de l’outil de webscraping utilisé et de son fonctionnement 3](#_Toc211798178)

[MongoDB 5](#_Toc211798179)

[Méthode d’importation des données 5](#_Toc211798180)

[Description du schéma des documents (champs et types) 6](#_Toc211798181)

[3 requêtes de sélection avec explication 7](#_Toc211798182)

[Lien vers la base MongoDB Atlas 8](#_Toc211798183)

[GitHub 8](#_Toc211798184)

[Lien vers dépôt GitHub 8](#_Toc211798185)

[Conclusion 9](#_Toc211798186)

[Tableau de contribution 9](#_Toc211798187)

# Webscraping

### Description du site et des données choisis

Pour notre première expérience de webscraping, nous avons choisi un site web dont les données sont accessibles : elles sont toutes affichées sur la même page, dans un tableau. Le site choisi est un site listant les items et leur prix pour un jeu vidéo.

Le site a un défaut : il n’offre ni nombre décimal, ni texte avec des accents, mais nous avons trouvé une valeur décimale cachée (Return of Interest => roi) et nous contentons de cela par souci de temps.

<https://prices.runescape.wiki/osrs/all-items>

Aussi, choisir un site nommé Runescape nous donne la chance d’appeler notre fichier de webscraping « Rune-scrape » :)

### Présentation de l’outil de webscraping utilisé et de son fonctionnement

Suite au laboratoire avec MongoDB pour lequel on a utilisé un script Python, nous avons choisi une approche similaire. On a commencé par lire la documentation de l’API du site web [[1]](#footnote-1) pour découvrir qu’il utilise déjà des fonctionnalités avec Python Requests. Il comprend l’URL de l’API à utiliser et des astuces au niveau User-Agent, ainsi que des liens vers Stack Overflow qui aident avec le format de la requête. En suivant cette documentation, ainsi que la documentation de Python Requests, il était assez facile de récolter toutes les informations qu’on voulait.

Exemple de code:

BASE = "<https://prices.runescape.wiki/api/v1/osrs>"

HEADERS = {

"User-Agent": "MyOSRSPriceScraper/1.0 (contact: [myemail@example.com](mailto:myemail@example.com))"

}

def fetch\_mapping():

url = f"{BASE}/mapping"

resp = requests.get(url, headers=HEADERS)

resp.raise\_for\_status()

return resp.json()

Dans cet extrait, nous allons chercher la colonne et son contenu.﷟

Après, il suffisait de créer quelques fonctions qui allait retourner un JSON contentant les informations retirées de chaque requête URL et une boucle pour attacher toutes ces informations ensemble. Lorsque le JSON a été parfaitement assemblé, il n’a resté qu’à l’enregistrer dans un fichier CSV.

# MongoDB

### Méthode d’importation des données

Nous avons testé deux méthodes fonctionnant également : un script Python (voir « conversion CSV, envoi à MongoDB et requêtes.py » en annexe) et une commande en ligne de commande.

Le script Python pouvait envoyer autant le CSV que le JSON à MongoDB. Nous avons finalement choisi le JSON, car ce dernier crée les sous-objets dans MongoDB pour une meilleure lisibilité, alors que le CSV ne peut créer qu’un niveau dans le document.

La commande en ligne de commande, elle, utilise le fichier CSV. Elle est utilisée ainsi :

Se placer dans le dossier bin :

ex: C:\Users\damou\OneDrive\Bureau\Session 5\Exploration\Projet2\mongodb-database-tools-windows-x86\_64-100.13.0\bin

Commande dans le terminal :

.\mongoimport --uri mongodb+srv://damoursjeremie\_db\_user:mTDTE04xFuSuZl9M@projet2.wlw3tln.mongodb.net/db --collection item\_runescape --type csv --headerline --file "..\..\osrs\_prices.csv"

### Description du schéma des documents (champs et types)

Les documents possèdent l’id par défaut de MongoDB, les champs id (entiers) et name (texte) ainsi que deux sous-objets : metadata et price.  
Metadata contient plusieurs champs dont « members », un booléen. Price contient uniquement des entiers.

|  |  |  |
| --- | --- | --- |
| id | 20011 |  |
| name | "3rd age axe" |  |
| examine | A beautifully crafted axe, shaped by ancient smiths. |  |
| metadata | Object |  |
|  | members | true |
|  | lowalch | 22000 |
|  | limit | 40 |
|  | value | 55000 |
|  | highalch | 33000 |
|  | icon | "3rd age axe.png" |
|  | name | "3rd age axe" |

|  |  |  |
| --- | --- | --- |
| price | Object |  |
|  | high | 500000000 |
|  | highTime | 1757807353 |
|  | low | 2147483647 |
|  | lowTime | 1755056562 |
| roi | -0.7671693562377101 |  |

### 3 requêtes de sélection avec explication

Les requêtes sont faites dans le fichier « conversion CSV, envoi à MongoDB et requêtes.py » en annexe. Elles sont exécutées lors de la connexion avec la DB.

#### Requête 1:

print("Requête #1")

result = collection.find\_one({"name": "3rd age amulet"})

print(result)

Cette requête trouve l’objet doté du nom « 3rd age amulet » et l’affiche.

#### Requête 2 :

print ("Requête #2")

result = collection.find\_one({"metadata.value": 50500})

print(result)

Cette requête trouve un objet dont le champ « value » dans le sous-objet « metadata » vaut 50 500 et l’affiche.

#### Requête 3 :

print("Requête #3")

result = collection.find({"price.low": {"$gt": 500000}}).limit(4)

for doc in result:

    print(doc)

Cette requête récupère les 4 premiers items dont le champ « low » dans le sous-objet « price » est supérieur à 500 000.

### Lien vers la base MongoDB Atlas

[https://cloud.mongodb.com/v2/68e3c5f9967deb2ba8142d40#/metrics/replicaSet/68ee54311d83526611fb24f5/explorer/Projet2/Rune/find](https://cloud.mongodb.com/v2/68e3c5f9967deb2ba8142d40" \l "/metrics/replicaSet/68ee54311d83526611fb24f5/explorer/Projet2/Rune/find)

Une image contenant texte, capture d’écran, Police, logiciel

Le contenu généré par l’IA peut être incorrect.

# GitHub

### Lien vers dépôt GitHub

<https://github.com/FelicyaLJ/projet2_ent>

# Conclusion

Le webscraping a été un léger défi, mais en utilisant un script plutôt qu’un outil pré-conçu, nous avons profité d’une grande flexibilité qui a rendu le travail beaucoup plus facile.

MongoDB Atlas est doté d’une interface Web conviviale et assez simple d’utilisation, bien que certaines options comme l’invitation de membres soient un peu plus cachées. Bien qu’il soit un peu contre-intuitif de travailler en NoSQL pour la première fois, la flexibilité offerte par les collections et les documents peut permettre de contourner la rigidité du SQL dans certaines situations. De plus, l’accès aux collections possibles à partir d’un script Python ouvre les portes à des applications dont la conservation des données en serait nettement facilitée.

# Tableau de contribution

|  |  |
| --- | --- |
| Tâche | Membre(s) |
| Recherches sur le webscraping | Felicya, Jasmin |
| Écriture du script de webscraping | Jasmin |
| Conversion du JSON en CSV | Jérémie et Felicya |
| Importation des documents vers la DB | Felicya et Jérémie |
| Requêtes de sélection | Felicya et Jérémie |
| Création du dépôt GitHub | Felicya |
| Structure du rapport | Felicya |
| Complétion du rapport | Principalement Felicya |

1. <https://oldschool.runescape.wiki/w/RuneScape:Real-time_Prices> [↑](#footnote-ref-1)