

Your News Anchor

E1 - Mise en situation

Développeur en Intelligence Artificielle, titre professionnel enregistré au RNCP - École IA
Microsoft by Simplon



par Vincent Papelard

9 février 2024

Table des Matières

Introduction	1
1 Présentation des outils, plateformes et langages utilisés	1
2 Gestion du projet	1
3 Les composants du projet	1
3.1 Collecte des données	2
3.2 Nettoyage des données	4
3.3 Stockage des données	4
3.4 Accès aux données et API	5
4 Tests et intégration continue	6
5 Déploiement	6
5.1 Base de données	6
5.2 API	6
5.3 Script d'extraction des données	7
Conclusion	7
Annexes	8

Introduction

Dans un contexte de diffusion rapide de l'information via Internet et face à la multitude de sources d'informations disponibles, il est parfois difficile de savoir où s'informer. Où trouver les actualités qui nous intéressent vraiment, en allant à l'essentiel ? C'est la problématique à laquelle répond ce projet qui nous est confié par notre client, un gros acteur dans le domaine des médias.

Your News Anchor est une application web qui offre à ses utilisateurs un résumé concis des actualités du jour. Ce résumé est automatiquement généré par intelligence artificielle à l'aide d'une liste de sources personnalisable par l'utilisateur, puis présenté par une voix synthétisée comme un vrai journal télévisé. Ce dossier se penche sur toute la partie du projet relative à la collecte, au nettoyage et au stockage des données nécessaires à la génération automatique de résumés. **Le développement de l'application à proprement parler ainsi que la conception d'un modèle d'intelligence artificielle de génération de résumé ne sont pas couverts dans ce dossier.** La totalité du code est disponible sur GitHub :

- Collecte, nettoyage et agrégation des données : https://github.com/vinpap/your_news_anchor
- API de la base de données : https://github.com/vinpap/your_news_anchor_db_api

Notre client souhaite que le projet soit terminé dans un délai d'un mois. Pour cela, il nous alloue un budget de 4200€. Ce budget couvre uniquement nos honoraires, soit 21 jours travaillés facturés 200€ chacun. Tous les frais réguliers qui devront être payés suite à la livraison du projet, tels que le paiement d'abonnement à des services de cloud, sont à la charge de notre client.

1 Présentation des outils, plateformes et langages utilisés

Les outils et plateformes suivants ont été utilisés dans le cadre de ce projet :

- Développement : **Python**
- Base de données : **PostgreSQL**
- Visualisation de la base de données : **DBeaver**
- Plateformes de déploiement : **Microsoft Azure, PythonAnywhere**
- Versionnage et gestion de projet : **GitHub**
- Prise de notes : **Obsidian**
- Rédaction du rendu : **LATeX**
- Diagrammes et schémas : **draw.io**

2 Gestion du projet

Le projet a été réalisé au cours de plusieurs sprints d'une semaine chacun. GitHub a été utilisé pour assurer le suivi des tâches et des sprints, ainsi que le versionnage du code.

3 Les composants du projet

Le projet est divisé en trois composants :

- une **base de données** PostgreSQL hébergée sur Azure qui stocke les sources d'information et les articles de journal extraits ainsi que les informations concernant les utilisateurs de *Your News Anchor*

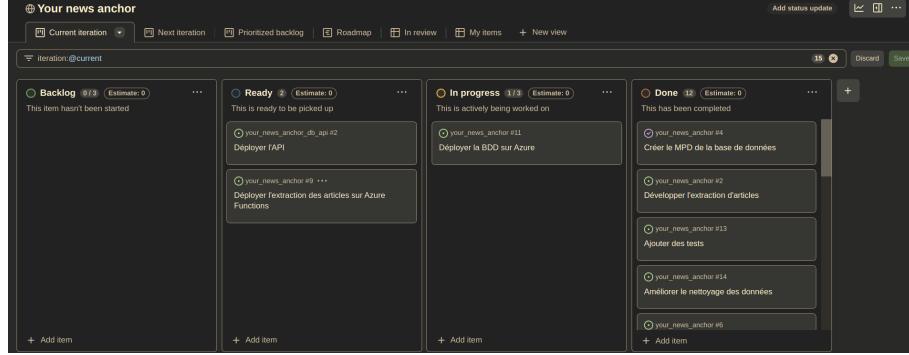


Figure 1: Tableau des tâches de GitHub Projects

- une [API](#) qui expose cette base de données
- le script [extract_articles.py](#). Ce script effectue **la collecte, le nettoyage et l'agrégation des articles de journal qui nous serviront à générer des résumés**, avant de les stocker dans la base de données en passant par l'API.

Chaque composant est développé et déployé indépendamment des deux autres, selon une architecture de microservices. Les différentes étapes de l'exécution du projet sont détaillées dans la figure 2. Par ailleurs, des **captures d'écran** sont présentées dans les annexes pour illustrer les différentes étapes du processus.



Figure 2: Fonctionnement général du projet, de la récupération de la liste des sources d'information à utiliser à l'enregistrement des articles extraits dans la base de données.

3.1 Collecte des données

La collecte des données est réalisée par le script [extract_articles.py](#), qui fait lui-même appel à des fonctions définies dans [extraction_functions.py](#). Afin que les articles stockés dans la base de données restent à jour, ce script est **automatiquement exécuté toutes les 24 heures** pour récupérer les derniers articles de journaux publiés.

La collecte des données est réalisée en plusieurs étapes. Tout d'abord, le script appelle l'API pour récupérer une liste de sources d'information d'où extraire des articles ([lien vers le code](#)).

On va ensuite utiliser le module **feedparser** de Python pour parcourir les flux RSS et en extraire une liste d'URLs d'articles avec d'autres informations telles que le titre de chaque article. Dans la mesure où l'objectif final est de réaliser des résumés concis des actualités du jour, on ne récupère pas la totalité des articles de chaque flux RSS. À la place on récupère les n premiers articles de chaque flux (les premiers articles correspondent souvent aux gros titres du jour, et ont donc plus de chance d'être pertinents). Enfin, on récupère le code HTML des pages web de chaque article et on en extrait le texte de l'article ainsi que plusieurs métadonnées utiles (auteurs, date de publication...).

Cette étape représente un défi particulier. En effet, l'approche traditionnelle consiste à identifier dans le code HTML les classes et id qui permettent de localiser sur la page web le contenu que l'on souhaite extraire, en utilisant des outils tels que BeautifulSoup. Or, la structure de chaque page web est différente. Si l'on voulait collecter le contenu des articles de façon fiable, il faudrait techniquement écrire du code propre à chaque site web. Cette approche a plusieurs inconvénients qui la rendent inadaptée à nos besoins :

- Écrire un code différent pour chaque source de données est un travail long et laborieux. De plus certains sites nous compliquent la tâche, par exemple en obfuscuant les noms des classes et des id dans leur code source
- L'application prévoit que les utilisateurs puissent eux-mêmes rajouter des sources de données. Cela est impossible si le code ne nous permet de gérer qu'une liste de sites web fixe
- Par ailleurs, cette technique est très peu robuste. Le moindre changement apporté au code source des pages web peut rendre notre code obsolète.

Pour toutes ces raisons, on préférera s'orienter vers d'autres méthodes. L'idéal serait de pouvoir coder un "web scraper universel", qui nous permettrait d'extraire le texte de n'importe quel article publié sur une page web quelconque. Cette tâche est extrêmement difficile à mettre en place en pratique, mais il est néanmoins possible de s'en approcher grâce au package Python **newspaper**, qui est utilisé dans le cadre de ce projet. Newspaper est un package créé pour extraire et traiter des articles depuis des pages web. Il permet ainsi d'extraire un article depuis une page web en quelques lignes :

```
from newspaper import Article

url = 'https://url-de-votre-article.com'
article = Article(url)
article.download()
article.parse()
text = article.text
```

Newspaper fonctionne en analysant le code des pages web avec BeautifulSoup. Certaines informations (titre, date...) sont retrouvées en cherchant des balises spécifiques telles que la balise <h1> par exemple, où grâce à des expressions régulières. Newspaper extrait aussi d'autres informations telles que les mots-clés d'un article grâce au module de NLP nltk. Dans notre cas, on récupérera les informations suivantes :

- le texte de l'article
- les auteurs
- la date de publication
- l'url de l'image en une de l'article (s'il y en a une)

Il est important de noter que cette solution n'est pas parfaite. Certains types de contenu, tels que les vidéos, ne sont pas exploitables. Newspaper a aussi du mal à traiter certains sites particuliers (il échoue à récupérer les articles du site 'L'Équipe', par exemple). Malgré tout, les résultats obtenus ont été concluants avec la grande majorité des sites d'information testés, en français comme en anglais.

3.2 Nettoyage des données

Le nettoyage des données est également réalisé par le script `extract_articles.py`. Après avoir collecté les articles, le script Python va passer par une étape de nettoyage des données. Malgré la bonne qualité générale du texte extrait, certains types de texte ne sont pas exploitables pour générer des résumés :

- Les reportages vidéo, qui ne comportent généralement qu'un bref texte de description sous la vidéo
- Les articles réservés aux abonnés, où seul le début du texte est visible
- Les flashes info et autres articles très (trop) courts pour en faire un résumé, de manière générale.

Les noms des auteurs automatiquement extraits par `newspaper` posent également problème. La liste d'auteurs générée identifie souvent de façon erronée des auteurs dans le corps de l'article, et il n'est pas rare d'obtenir des listes d'auteurs telles que ['Par', 'Julien Dupont', 'Le Président'], par exemple.

Le premier problème peut être résolu simplement : il suffit de filtrer les articles en fonction de leur longueur, pour ne garder que ceux dont le nombre de caractères dépasse une certaine seuil. En ce qui concerne la liste des auteurs, une solution un peu plus complexe a été trouvée : réaliser une détection d'entités dans la liste à l'aide de **Spacy** pour ne garder que les chaînes de caractère qui correspondent à des noms propres qui désignent des personnes. Cela a permis d'éliminer la quasi-totalité des erreurs d'extraction de noms d'auteur.

Après avoir été nettoyés, les articles extraits sont rassemblés dans une liste Python avant d'être enregistrés via une API dans la base de données décrite dans la section suivante.

3.3 Stockage des données

Toutes les informations extraites (articles et métadonnées) ainsi que la liste des sources d'information à traiter et toutes les autres données nécessaires au fonctionnement de l'application Your News Anchor sont stockées sur une base de données **PostgreSQL**. Le Modèle Physique de Données qui décrit la structure de notre base de données est présenté dans la figure 3.

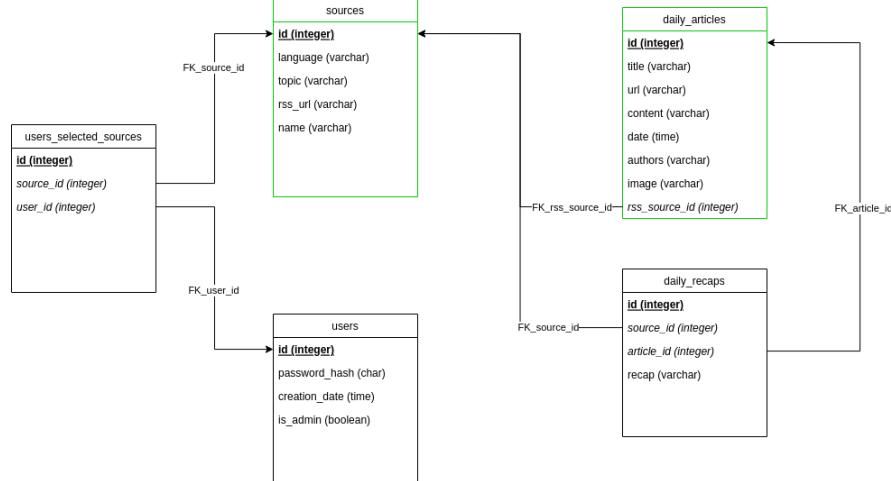
La base de données est composée des tables suivantes :

- *standard_sources* : contient une liste de sources d'information, avec le lien vers leur flux RSS. Cette liste est définie par les administrateurs de l'application et ne peut pas être modifiée par les utilisateurs. Les flux RSS disponibles dans cette table peuvent être utilisés par n'importe quel utilisateur
- *user_sources* : cette table recense les flux RSS ajoutés manuellement par les utilisateurs. Seul l'utilisateur qui a ajouté un flux peut l'incorporer dans sa liste de sources d'informations
- *users* : la table qui contient les informations concernant les utilisateurs (identifiant, hash du mot de passe...)
- *daily_articles* : contient tous les articles extraits par le script de collecte des données. Le contenu de cette table est renouvelé à intervalles réguliers (par exemple toutes les 24 heures), lorsque le script de collecte des données est exécuté
- *daily_recaps* : la table où sont stockés les résumés d'articles générés. Ces résumés sont eux aussi renouvelés à intervalles réguliers.

Différentes requêtes et insertions SQL sont utilisées pour réaliser les tâches suivantes :

- Récupération de la liste des flux RSS disponibles pour un utilisateur spécifique
- Récupération de la liste de tous les flux RSS enregistrés
- Mise à jour des articles de journal enregistrés dans la base de données avec une nouvelle liste d'articles.

Une liste complète des requêtes SQL utilisées dans le cadre de ce projet est disponible **en annexe**. On notera que ces requêtes ont été pensées pour optimiser la vitesse d'accès aux données. Ainsi :



Les tables en vert sont utilisées dans le cadre de ce dossier

Figure 3: MPD de notre base de données. Ce MPD a été réalisé en suivant la méthode MERISE, une méthode de conception qui formalise la conception de systèmes informatiques. Ici, nous appliquons cette méthode pour modéliser les différentes tables de notre base de données et les relations qui les lient. Cette méthode a été choisie car elle nous permet de nous assurer que notre base de données est cohérente et répond bien aux besoins de notre application.

- L'ancien contenu de la table *daily_articles* est supprimé avant d'y insérer les articles de journal du jour. L'objectif est de ne garder que des articles pertinents et de limiter la taille de cette table, ce qui accélère l'exécution des requêtes sur cette table
- On modifie les requêtes pour ne sélectionner dans les tables que les informations dont on a besoin. Dans la **requête qui sélectionne la liste des flux RSS disponibles pour un utilisateur**, par exemple, on nomme explicitement toutes les colonnes que l'on veut sélectionner dans notre requête plutôt que de tout sélectionner en utilisant 'SELECT *', qui est plus lent. À titre d'exemple, le temps d'exécution de la requête ci-dessus est passé de 18 ms à 15 ms en effectuant ce changement, avec environ dix lignes dans la table *user_sources* et la table *standard_sources*.

On notera que ces requêtes ne sont en aucun cas utilisées par notre script `extract_articles.py`, qui n'a pas accès à la base de données. À la place, le script accède à la base de données en passant par une API que nous allons présenter dans la section suivante. Cette API est définie dans un dépôt séparé, dans un script `app.py` ([lien](#)). C'est ce script qui exécute les insertions et requêtes SQL présentées dans les annexes.

3.4 Accès aux données et API

Afin de simplifier l'accès aux données, une API a été développée pour exposer la base de données PostgreSQL. Cela nous évite de devoir gérer nous-mêmes les connexions et requêtes SQL dans notre application, et nous permet de mettre en place des sécurités pour contrôler qui accède à nos données et/ou les modifie. L'API a été développée à l'aide de **FastAPI**, un framework Python destiné à développer rapidement des API REST performantes. Cette API propose deux endpoints pertinents dans le cadre de notre dossier :

- **/feeds**, qui permet de récupérer (mais pas de modifier) le contenu de la table 'sources'
- **/update_articles**, qui sert à renouveler entièrement le contenu de la table 'daily_articles'. Cette opération implique d'effacer le contenu existant de la table pour le remplacer par les données envoyées dans un objet JSON joint à la requête.

Dans la mesure où la requête envoyée à ce deuxième endpoint supprime des données, il a fallu sécuriser l'accès à la base de données. Dans ce but, une clé d'API secrète doit être envoyée avec la requête. Cette clé est stockée dans une variable d'environnement afin de ne pas être visible "en clair" sur un dépôt GitHub, par exemple. Plus d'informations sont disponibles dans la section consacrée au déploiement du projet.

4 Tests et intégration continue

Afin de garantir la qualité du code et de s'assurer que ni notre script `extract_articles.py` ni le script de notre API `app.py` ne vont cesser de fonctionner, plusieurs mesures ont été prises :

- des tests unitaires ont été codés dans le fichier `test_extraction_script.py` pour vérifier que les fonctions qui permettent d'extraire les articles de journaux de pages web fonctionnent bien comme attendu. De la même manière, le `fichier test_api.py` contient des tests unitaires qui vérifient que les différents points de terminaison de l'API fonctionnent comme attendu.
- ces tests ont été automatisés avec GitHub Action. Ils sont ainsi automatiquement exécutés à chaque fois que du code est intégré à la branche 'main' du dépôt 'your_news_anchor' ou du dépôt 'your_news_anchor_db_api'.

5 Déploiement

Comme évoqué plus haut, les trois composants de ce projet sont développés et déployés indépendamment les uns des autres.

5.1 Base de données

La base de données PostgreSQL a d'abord été mise en place localement pendant la période de développement. Elle a ensuite été exportée via un script SQL accessible via [ce lien](#) sur le service Azure Database for PostgreSQL de Microsoft Azure. Pour plus de sécurité, la base de données a été paramétrée de sorte à n'autoriser l'accès qu'à une adresse IP spécifique, où est déployée notre API.

5.2 API

L'API est elle aussi déployée sur Microsoft Azure, plus spécifiquement sur le service **Azure Web App**. Le déploiement est automatisé grâce à GitHub Actions afin que les mises à jour apportées au code soient automatiquement déployées sur Azure. Comme nous l'avons vu, l'API a besoin de définir une clé secrète pour fonctionner. Cependant, il est impossible d'inclure cette clé dans notre code pour des raisons évidentes : celle-ci serait visible de tous sur notre dépôt GitHub. Pour régler ce problème, on utilisera des secrets GitHub. Cette fonctionnalité de GitHub nous permet de créer des valeurs secrètes liées à notre dépôt mais dont la valeur est cachée. On peut ensuite ajouter les lignes suivantes au fichier YAML qui gère le pipeline de déploiement de GitHub action :

```
- name: Create secret values in environment (API token and DB password)
run: |
  export API_TOKEN=${{ secrets.SECRET_API_TOKEN }}
  export DB_PWD=${{ secrets.SECRET_DB_PWD }}
```

Lors du déploiement, ces instructions créeront sur le serveur des variables d'environnement qui contiennent nos valeurs secrètes (le mot de passe de la base de données et notre token d'API) sans pour autant exposer leur valeur dans le code.

5.3 Script d'extraction des données

Notre script d'extraction d'articles doit être automatiquement exécuté à intervalles réguliers. La solution retenue a été de l'exécuter sur [PythonAnywhere](#). PythonAnywhere est une plateforme basée sur AWS qui permet de déployer des scripts et applications Python avec un minimum d'efforts.

Conclusion

Ce dossier a couvert toutes les problématiques liées à la mise à disposition des données dans le cadre du projet Your News Anchor. Il a notamment mis en évidence les difficultés liées à la mise en place d'une solution de web scraping universel, et les réponses qui peuvent être apportées à ce problème. L'utilisation de packages tels que newspaper, qui combinent techniques de NLP et analyse de code HTML, en est un exemple.

Annexes

Illustration de l'exécution du projet

	id	url	language	name	topic
1	1	https://www.france24.com/fr/rss	FR	France 24	General
2	2	https://www.lemonde.fr/rss/une.xml	FR	Le Monde	General
3	3	https://www.20minutes.fr/rss-monde.xml	FR	20 Minutes	General

Figure 4: Notre base de données PostgreSQL stocke une liste de flux RSS à utiliser pour récupérer les adresses web d'articles de journaux. Cette capture d'écran montre la table 'standard_sources' qui contient une liste de flux RSS disponibles pour tous les utilisateurs, mais ces derniers peuvent aussi ajouter leur propre liste de flux RSS qui seront stockés dans la table 'user_sources'.

```

Server response
Code Details
200 Response body
{
  "source_id": 1,
  "name": "BFM",
  "url": "https://www.bfmtv.com/rss/actualites/",
  "topic": "General",
  "language": "FR"
},
{
  "source_id": 1,
  "name": "France 24",
  "url": "https://www.france24.com/fr/rss",
  "topic": "General",
  "language": "FR"
},
{
  "source_id": 4,
  "name": "20 Minutes",
  "url": "https://www.20minutes.fr/Feeds/rss-monde.xml",
  "topic": "International",
  "language": "FR"
},
{
  "source_id": 2,
  "name": "Google News",
  "url": "https://news.google.com/rss",
  "topic": "General",
  "language": "EN"
}
]
Response headers
Content-Length: 574
Content-Type: application/json
Date: Thu, 04 Apr 2024 08:45:46 GMT
Server: uvicorn

```

Figure 5: Le script extract_articles.py accède à la base de données via une API. Cette image montre le résultat d'une requête envoyée à l'API pour récupérer la liste des sources d'information à utiliser pour un utilisateur spécifique. Ces sources sont soit des sources 'standard', c'est-à-dire que tous les utilisateurs y ont accès, soit des sources personnalisées que l'utilisateur a lui-même enregistrées.

```

</description>
<guid isPermaLink="true">http://www.lemonde.fr/international/live/2024/04/04/en-direct-guerre-en-ukraine-onze-drones-russes-abattus-une-quinzaine-ont-vise-la-ville-de-kharkiv-faisant-4-morts-et-10-
blesse-3210.html</guid>
<link>https://www.lemonde.fr/international/live/2024/04/04/en-direct-guerre-en-ukraine-onze-drones-russes-abattus-une-quinzaine-ont-vise-la-ville-de-kharkiv-faisant-4-morts-et-10-blesses-6225223_3210.htm</link>
<media:credit type="plain">Le ministre de la défense russe, Sergueï Chougo, le 30 mars 2024. </media:credit>
<media:credit scheme="urn:ebu">AP/mediacredit</media:credit>
</media:content>
</item>
<item>
<title>
<![CDATA[ PFRAS : les ustensiles de cuisine exclus par les députés de la loi visant à réduire l'exposition aux « polluants éternels »]]>
</title>
<pubDate>Thu, 04 Apr 2024 12:45:17 +0200</pubDate>
<description>
<![CDATA[ La veille, alors qu'il était à 18 et sa fille à 6 ans à « céder aux lobbyings de Seb, au détriment de la santé des Français », ont réagi les députés écologistes. La veille, des salariés de Seb avaient manifesté devant l'Assemblée nationale pour dénoncer le retrait de la proposition de loi. ]]>
</description>
<guid isPermaLink="true">https://www.lemonde.fr/planete/article/2024/04/04/pfas-les-deputes-excluent-les-ustensiles-de-cuisine-du-perimetre-de-la-loi-destinee-a-reduire-l-exposition-de-la-population-aux-polluants-eternels-6225927_3245.html</guid>
<link>https://www.lemonde.fr/planete/article/2024/04/04/pfas-les-deputes-excluent-les-ustensiles-de-cuisine-du-perimetre-de-la-loi-destinee-a-reduire-l-exposition-de-la-population-aux-polluants-eternels-6225927_3245.html</link>
<media:content type="image"><img alt="Photo of a protest in front of the National Assembly against the Seb company's influence on the law." data-bbox="500 100 900 300"/>
<media:credit type="plain">Des salariés de Seb manifestent devant l'Assemblée nationale. Mercredi 3 avril 2024. </media:credit>
<media:credit scheme="urn:ebu">ALAIN JOCARD / AFP</media:credit>
</media:content>
</item>
<item>
<title>
<![CDATA[ PFRAS : les ustensiles de cuisine exclus par les députés de la loi visant à réduire l'exposition aux « polluants éternels »]]>
</title>

```

Figure 6: Les flux RSS nous permettent de récupérer les adresses web des derniers articles publiés sur un site d'information. Cette capture d'écran nous montre une partie du flux RSS du site du journal Le Monde.



Figure 7: Le flux RSS présenté ci-dessus permet à notre script `extract_articles.py` d'accéder à l'article suivant.

The screenshot shows the MySQL Workbench interface with the 'daily_articles' table selected. The table has columns: id, aux_title, aux_url, aux_content, date, aux_author, aux_image, standard_source_id, and is_from_user. The data grid displays 36 rows of news articles, each with a unique ID and details about the source and author. The last row, which corresponds to the protest article shown in Figure 7, is highlighted in red.

	aux_title	aux_url	aux_content	date	aux_author	aux_image	standard_source_id	is_from_user
16	Satellite image shows cyclone #1	https://	A big chunk of the U.S. sitting	00:50:00	Apostol Milice	https://www.m	[NULL]	[v]
17	Gouvernement : C'est quoi ce défilé ?	https://	Le défilé de la Fête des	00:00:00		https://img.20r	[NULL]	[v]
18	Brésil : Un coup de coude à la	https://	Le Monde est au cœur de	00:00:00		https://img.20r	[NULL]	[v]
19	Bresil - « Un coup de coude à la	https://	Luis Claudio Luiz da Silva, m	00:00:00		https://img.20r	[NULL]	[v]
20	Séisme à Taiwan : La course con	https://	A lendemain du puissant s	00:00:00		https://img.20r	[NULL]	[v]
21	Séisme à Taiwan : La course con	https://	Un coup de téléphone et dei	00:00:00		https://img.20r	3 ↗	[]
22	Guerre en Ukraine : C'est quoi ce	https://	Un coup de téléphone et dei	00:00:00		https://img.20r	3 ↗	[]
23	Mexique : Une candidate à la ma	https://	Le Mexique est actuellement	00:00:00		https://img.20r	[NULL]	[v]
24	Japon : Un nouveau record d'earth	https://	La Terre a une nouvelle fois	00:00:00		https://img.20r	1 ↗	[]
25	Haiti : la CEDH condamne le P	https://	Discours à la nation#C'est	00:00:00	Grégoire Sauvage	https://france	1 ↗	[]
26	Au Sénégal, le président Bassine	https://	L'Etat français a été condam	00:00:00	Grégoire Sauvage	https://france	1 ↗	[]
27	Européennes : à Sotteville-lès-R	https://	REPORTAGE#MA Sotteville-lès	00:00:00	Romain Brunet	https://france	1 ↗	[]
28	● En direct : Israël accusé d'	https://	A savoir : le nombre des vict	00:00:00		https://france	1 ↗	[]
29	Comment la Chine lamine l'Indus	https://	Le symbole le plus évidente	00:00:00	Jean-Michel Bezat	https://lem	2 ↗	[]
30	● En direct, guerre en Ukraine : Em	https://	que direct, guerre en Ukraine : Em	00:00:00	Jean-Michel Bezat	https://img.20r	2 ↗	[]
31	Démission surprise de Jean-Baptiste	https://	Des salariés de Seb manifeste	00:00:00	Jean-Michel Bezat	https://img.20r	2 ↗	[]
32	● PAS les responsables de culture ?	https://	Des salariés de Seb manifeste	00:00:00	Jean-Michel Bezat	https://img.20r	3 ↗	[]
33	Bresil - « Un coup de coude à la	https://	Luis Claudio Luiz da Silva, m	00:00:00		https://img.20r	3 ↗	[]
34	Séisme à Taiwan : La course con	https://	A lendemain du puissant s	00:00:00		https://img.20r	3 ↗	[]
35	Japon : Un séisme de magnitude	https://	La terre a une nouvelle fois l	00:00:00		https://img.20r	3 ↗	[]

Figure 8: Après avoir extrait et nettoyé le contenu de l'article depuis la page web, le script `extract_articles.py` enregistre les articles extraits dans la base de données via le point de terminaison `'/update_articles'` de l'API. Cette capture d'écran montre le contenu de la table 'daily_articles' dans la base de données, incluant l'article de journal ci-dessus.

Architecture générale du projet

Sources de données utilisées pour les tests

Voici la liste des flux RSS avec lesquels l'extraction de données a été testée avec succès pendant le développement :

- <https://www.france24.com/fr/rss>
- <https://www.lemonde.fr/rss/une.xml>
- <https://news.google.com/rss?hl=fr&gl=FR&ceid=FR:fr>
- <https://www.ouest-france.fr/rss/france>
- <https://www.francetvinfo.fr/monde.rss>

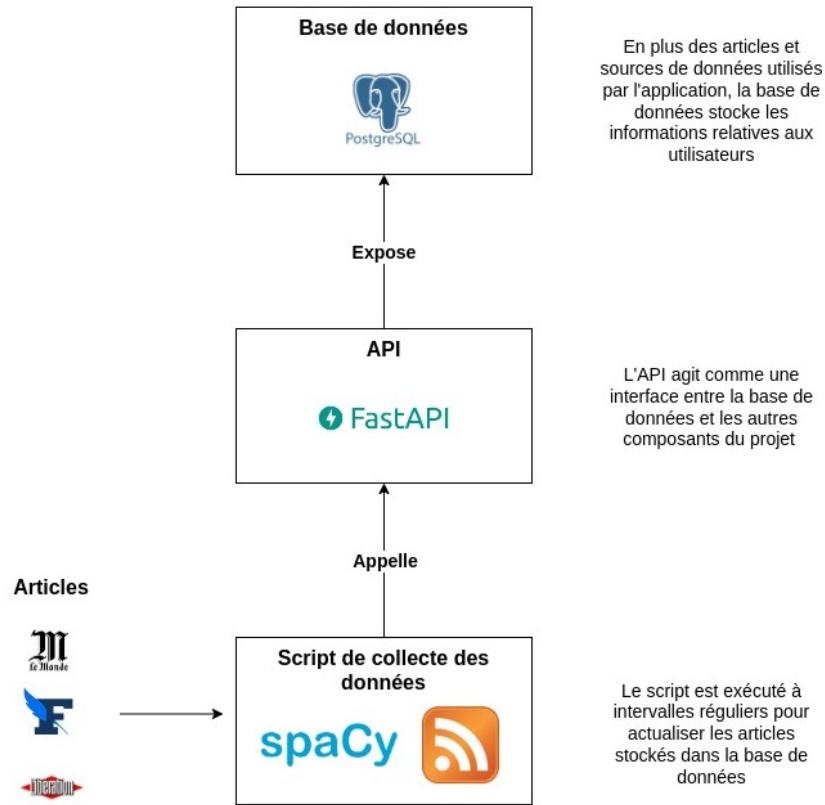


Figure 9: Les différents composants du projet

- <https://www.20minutes.fr/feeds/rss-monde.xml>
- <https://rmc.bfmtv.com/rss/actualites/>
- <https://feeds.leparisien.fr/leparisien/rss>
- <https://www.ladepeche.fr/rss.xml>

En plus des sources ci-dessus, le script a été testé avec le flux RSS du journal l'Équipe. Les résultats n'ont pas été concluants (Seul les premières phrases de chaque article sont détectées).

Plusieurs flux RSS de sources en anglais ont aussi été testés, avec succès :

- <https://www.independent.co.uk/news/world/rss>
- <https://rss.app/feeds/D0SUH2NYYoOzypYN.xml>
- <https://www.huffpost.com/section/world-news/feed>
- <https://news.google.com/rss>

SQL

Voici la liste des requêtes et insertions SQL utilisées par l'API 'your_news_anchor_db_api dans le cadre de ce projet :

- point de terminaison /feeds : récupération de la liste des flux RSS disponibles pour un utilisateur

- Pour vérifier que l'utilisateur concerné existe bien :

```
SELECT * FROM users WHERE username='{username}';
```

- Pour récupérer l'identifiant de l'utilisateur :

```
SELECT id from users where username='{username}';
```

- Pour récupérer la liste des flux RSS disponibles :

```
SELECT url, language, topic, name FROM (select url, language, topic, name
from user_sources where user_id={user_id}) union (select url, language, topic, name
from standard_sources where not exists (select * from user_sources where user_sources.url=standar
```

- point de terminaison /all_feeds : récupération de la liste de tous les flux RSS enregistrés

- Récupération des flux RSS enregistrés par les utilisateurs :

```
SELECT id, url, language, topic, name FROM user_sources;
```

- Récupération des flux RSS disponibles pour tous les utilisateurs :

```
SELECT id, url, language, topic, name from standard_sources;
```

- point de terminaison /update_articles : mise à jour des articles stockés dans la base de données

- Sélection des identifiants de tous les articles déjà enregistrés, permet d'identifier les articles obsolètes qui doivent être supprimés après l'ajout des nouveaux articles :

```
SELECT id FROM daily_articles;
```

- Insertion des nouveaux articles :

```
INSERT INTO daily_articles(title, url, content, author, image, is_from_user,
user_source_id) VALUES('{article.title.replace("", "")}', '{article.url.replace("", "")}',
'{article.content.replace("", "")}', '{article.authors.replace("", "")}', '{article.image}', true, {article.source_id});
```

Dans cette insertion, les valeurs des différentes informations concernant l'article sont définies par le script Python.

- Suppression des anciens articles :

```
DELETE FROM daily_articles WHERE id IN (", ". join(old_article_ids));
```

Ici 'old_article_ids' est la liste Python de tous les identifiants des articles déjà présents dans la base de données.

Captures d'écran

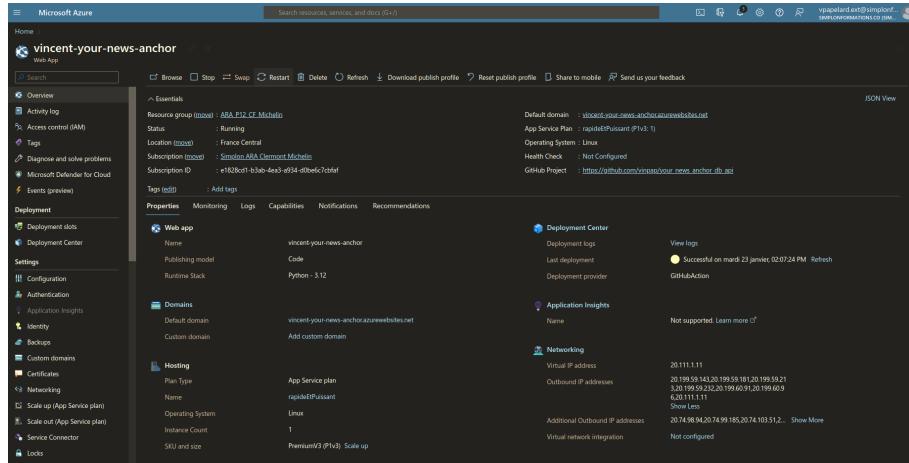


Figure 10: Dashboard de notre API sur Azure

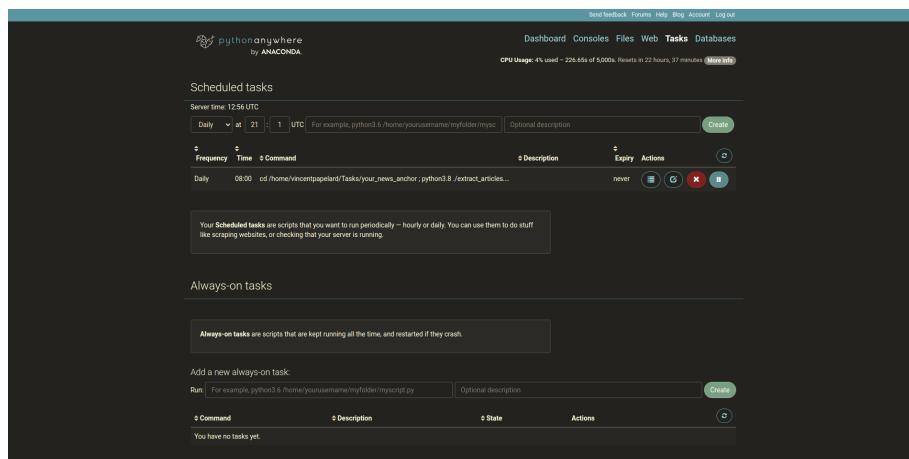


Figure 11: Dashboard de PythonAnywhere. Cette page nous permet de planifier l'exécution de scripts à intervalles réguliers, ici toutes les 24 heures