Rapport Final

API REST Zenless Zone Zero

DEWEERDT Simon - M2 CYBER

Sommaire

1 - Introduction

1.1 - Contexte

Zenless Zone Zero est un jeu développé par HoYoverse, intégrant des personnages appelés agents, des créatures appelées bangboo, ainsi que divers équipements influençant le gameplay, tels que les moteurs amplis et les disques de poussée.

1.2 - Objectifs

L'objectif de ce projet est de développer une API REST permettant d'accéder aux données détaillées de ces éléments avec une documentation détaillée sur l'utilisation de cette API, tout en s'appuyant sur du scraping afin de maintenir à jour la base de données.

1.3 - Positionnement face au projets similaire

Les développeurs du jeu (HoYoverse) ne fournissent pas d'API public pour le jeu Zenless Zone Zero. Pour cela, j'ai recherché des API public réaliser par des développeurs et j'ai trouvé un post Reddit (lien) où l'auteur compter développer cette API. Étant donné que le post daté déjà de plusieurs mois au démarrage de mon projet, qu'on n'avait pas de nouvelle sur le projet et que la publication avait été archivé, j'ai considéré que le projet avait été abandonné.

En revanche, on trouve beaucoup de site web regroupant les données du jeu comme Honey Hunter World, Prydwen, le Wiki officiel de HoYoverse (Wiki HoYoLAB), etc, ce qui ma donnée l'opportunité d'effectuer du scraping.

2 - Méthodologie et organisation

Pour mener à bien ce projet, il a été structuré autour de trois grands axes : la collecte des données, la structuration des modèles, et l'exposition des données via une **API REST**.

Comme je n'avais que peu d'expérience préalable en scraping, j'ai d'abord dû me former aux bonnes pratiques et aux outils adaptés. Pour cela, j'ai suivi une formation pratique via une formation en ligne (lien). Cette formation m'a permis de comprendre :

- Les différences entre les bibliothèques de scraping (ex. : BeautifulSoup, Selectolax, Playwright)
- La manière de cibler efficacement les éléments HTML
- Les pièges courants à éviter (comme les sites dynamiques, les chargements JavaScript, etc.)
- Comment structurer proprement un script pour qu'il soit maintenable
- Comment effectuer du scraping éthique, notamment avec l'intervention de **Rony Shalit**, Vice-président chargé de la conformité et de l'éthique chez Bright Data

Une fois cette base acquise, l'étape suivante a consisté à identifier les sources fiables de données concernant le jeu Zenless Zone Zero. Deux sites web ont été retenus pour leur richesse et leur fiabilité : Honey Hunter World et Wiki HoYoLAB. Une analyse manuelle de leur structure HTML a permis de concevoir des scripts de scraping adaptés.

Une fois les sources identifiées, la mise en place de l'environnement de développement a été réalisée avec l'aide de **Python**, **FastAPI**, **Docker**, **PostgreSQL**, et un outil de migration de base de données (Alembic). L'organisation du projet s'est ensuite articulée autour de plusieurs éléments :

- **Scraping et structuration des données** : extraction des informations pertinentes, nettoyage, et organisation en modèles Python.
- **Modélisation de la base de données** : création des entités via SQLAlchemy, puis génération des tables via Alembic.
- **Développement de l'API** : création progressive des endpoints RESTful pour chaque type de ressource (agent, bangboo, moteur ampli et disque de poussée).
- **Tests et documentation** : vérification des réponses API et documentation automatique générée par Swagger (intégré à FastAPI).

Le projet a été géré de manière individuelle, en utilisant **GitHub** comme système de versionnage et de sauvegarde. Un Makefile a été mis en place pour simplifier l'exécution des tâches de développement courantes, comme le démarrage de l'API, le lancement du scraping ou la création et l'application des migrations.

Lien du repository GitHub : https://github.com/GaruXer/api.zenless-zone-zero

3 - Conception de la solution

3.1 - Architecture

API REST : Interface permettant d'accéder aux données.

Base de données : Stockage des données récolté via le scraping

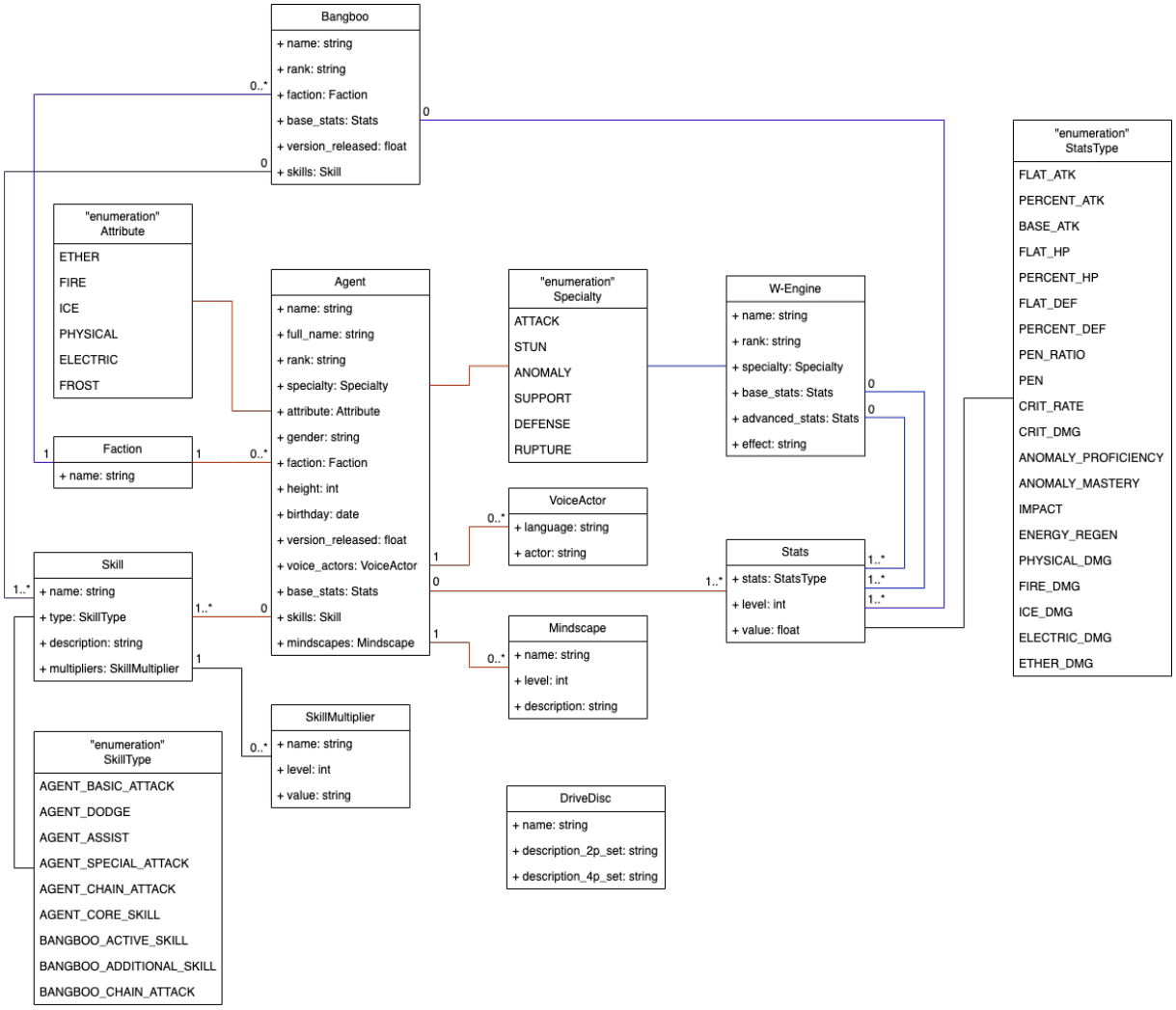
Scraping : Script permettant de récolter les données du jeu

Documentation API : Mode d'emploi pour l'utilisation de l'API

3.2 - Structure des endpoints de l'API

Ressource	Méthode	URL
Documentation	GET	/docs
Récupérer les agents	GET	/agent/all
Récupérer un agent	GET	/agent/{id}
Récupérer les bangboo	GET	/bangboo/all
Récupérer un bangboo	GET	/bangboo/{id}
Récupérer les moteurs ampli	GET	/w-engine/all
Récupérer un moteur ampli	GET	/w-engine/{id}
Récupérer les disques de poussée	GET	/drive-disc/all
Récupérer un disque de poussée	GET	/drive-disc/{id}

3.3 - Modélisation des données



4 - Réalisation

4.1 - Choix des technologies

Pour l'API j'ai décidé d'utiliser **FastAPI** qui est un framework web moderne et rapide (haute performance) pour la création d'API avec Python, basé sur les annotations de type standard de Python. Les principales fonctionnalités sont :

- **Rapidité** : De très hautes performances, au niveau de NodeJS et Go (grâce à Starlette et Pydantic). L'un des frameworks Python les plus rapides.
- **Intuitif** : Excellente compatibilité avec les IDE. Complétion complète. Moins de temps passé à déboguer.
- **Concis** : Diminue la duplication de code. De nombreuses fonctionnalités liées à la déclaration de chaque paramètre. Moins de bugs.
- **Robuste** : Obtenez un code prêt pour la production. Avec une documentation interactive automatique.

Pour la base de données j'ai utilisé **PostgreSQL** qui est un puissant système de base de données relationnelle, développé activement depuis plus de 35 ans, qui lui a valu une solide réputation en termes de **fiabilité**, de **robustesse** des fonctionnalités et de **performances**.

Pour le scraping j'ai utilisé plusieurs technologies : **Playwright** qui est un outil permettant d'automatiser les interactions avec des navigateurs web. Il est principalement utilisé pour faire du end-to-end testing, du scraping ou pour automatiser des tâches dans un navigateur comme remplir des formulaires, naviguer sur des pages, cliquer sur des éléments, etc. Et **BeautifulSoup** qui est une bibliothèque facilitant l'extraction d'informations à partir de pages web. Elle se place au-dessus d'un analyseur HTML ou XML, fournissant des idiomes Pythoniques pour l'itération, la recherche et la modification de l'arbre d'analyse. Afin de récolter les données, j'ai utilisé 2 sites web : Honey Hunter World et Wiki HoYoLAB.

Pour la documentation de l'API, **Swagger** est déjà intégré au Framework FastAPI. Swagger est un langage de description d'interface permettant de décrire des API exprimées à l'aide de JSON.

4.2 - Résultats

Le projet a permis de mettre en place une **API REST fonctionnelle et documentée**, permettant d'accéder de manière centralisée aux données du jeu Zenless Zone Zero. Les principaux résultats obtenus sont les suivants :

- **Extraction automatisée de données** : les scripts de scraping permettent de collecter automatiquement les informations depuis les deux sites ciblés. Cela garantit une mise à jour régulière de la base de données.
- **Base de données relationnelle cohérente** : les entités issues du jeu (agents, bangboos, moteurs, disques) sont modélisées et stockées dans une base PostgreSQL structurée.
- **API REST opérationnelle** : l'ensemble des endpoints permet de récupérer toutes les ressources du jeu, à la fois individuellement et globalement.
- **Documentation interactive** : via Swagger, chaque endpoint est détaillé automatiquement, facilitant la prise en main de l'API pour les utilisateurs.

L'un des résultats clés de ce projet réside dans la mise en place d'un **scraping éthique**, respectueux des sites web sources. Dès la conception, une attention particulière a été portée à la manière dont les données sont extraites afin de minimiser l'impact sur les serveurs de Honey Hunter World et du Wiki HoYoLAB. Plusieurs principes ont été suivis pour garantir un comportement responsable :

- **Fréquence d'exécution limitée** : le script de scraping est conçu pour être lancé manuellement environ une fois par mois, afin de maintenir la base de données à jour sans générer de trafic excessif.
- **Respect des temps de chargement** : des temporisations (wait_for_timeout) sont intégrées entre les requêtes pour éviter un comportement assimilable à un robot agressif.
- **Navigation réelle via navigateur headless** : en utilisant Playwright, les pages sont chargées comme dans un vrai navigateur, permettant d'interpréter correctement les scripts JavaScript tout en imitant le comportement d'un utilisateur humain.
- **Volume de requêtes maîtrisé** : chaque session de scraping cible un nombre limité de pages (par exemple, une par personnage ou équipement), sans itérations massives sur l'ensemble du site.

Ce soin apporté à l'éthique du scraping permet non seulement d'assurer une cohabitation respectueuse avec les sites web communautaires, mais aussi de prolonger la viabilité du projet sans risque de blocage ou de bannissement.

4.3 - Perspectives d'amélioration

Plusieurs axes d'amélioration ont été identifiés pour renforcer la robustesse, l'éthique et la pérennité du projet :

La mise en place de **tests automatisés** avec coverage permettrait d'assurer la qualité et la fiabilité du code, il serait pertinent de développer une suite complète de tests unitaires et fonctionnels. L'intégration d'outils de couverture de code (coverage) permettrait de mesurer précisément la part du code testée, facilitant ainsi la maintenance et l'évolution du projet tout en réduisant les risques de régressions.

L'optimisation du scraping avec des solutions plus éthiques avec l'utilisation de services professionnels comme **Bright Data** permettrait d'améliorer la gestion des requêtes, en répartissant les accès via des proxies rotatifs et en réduisant encore davantage le risque d'impact sur les sites web. Cette solution offrirait une meilleure scalabilité et un respect accru des limites d'utilisation imposées par les sites cibles. De plus, pour garantir une mise à jour régulière et fiable des données, la mise en place de tâches planifiées (**cronjobs**) serait une amélioration notable. Cela permettrait de lancer automatiquement le script de scraping à intervalles définis, assurant ainsi que la base de données reste synchronisée avec les sources d'information tout en respectant la fréquence éthique définie. Cette automatisation faciliterait la maintenance opérationnelle et garantirait une actualisation continue des données.

Pour améliorer la flexibilité et la pertinence des données retournées, il serait intéressant de permettre aux utilisateurs d'appliquer des **filtres** lors des appels à l'API. Par exemple, filtrer les agents par type, les équipements par rareté ou attribut, ou encore trier les résultats selon différents critères. Cela rendrait l'API plus puissante et adaptée à des cas d'usage variés.

Le déploiement en environnement de **production** sécurisé permettrait d'assurer une disponibilité et une performance optimale de l'API, la mise en place d'un environnement de production dédié est envisagée. Cela inclut l'hébergement sur un serveur cloud, la configuration d'un serveur web, ainsi que la sécurisation des communications via HTTPS. L'acquisition d'un nom de domaine personnalisé permettra également d'offrir une interface plus professionnelle et accessible à l'API, facilitant son intégration dans des projets externes.

5 - Conclusion

Ce projet a permis de concevoir et de développer une API REST fonctionnelle et documentée, centrée sur les données du jeu Zenless Zone Zero. Grâce à l'intégration de techniques de scraping web adaptées et éthiques, la solution assure une mise à jour régulière et automatisée d'une base de données relationnelle, offrant ainsi un accès centralisé à des informations détaillées sur les agents, bangboo, moteurs amplis et disques de poussée.

La réalisation a été l'occasion de se former aux bonnes pratiques du scraping, de maîtriser des outils modernes tels que FastAPI, Playwright et BeautifulSoup, et d'appliquer les principes de conception d'API robustes et performantes. Le respect de l'éthique lors du scraping a été une priorité, garantissant une charge maîtrisée sur les sites sources et une pérennité du projet.

Les résultats obtenus confirment la faisabilité technique du projet et ouvrent la voie à de nombreuses améliorations futures, notamment sur la qualité du code, l'enrichissement des fonctionnalités API et la mise en production professionnelle.

En somme, ce projet constitue une base solide et évolutive pour offrir à la communauté de joueurs et de développeurs un outil fiable d'accès aux données du jeu Zenless Zone Zero, tout en respectant les contraintes techniques et éthiques inhérentes à ce type d'application.