L3 MIAGE Apprentissage

Sylvain VOCALE
Pierre RAMBERT
Isaïa GUILLE

Comment identifier et analyser les contenus de nature "conspiratoire" sur YouTube ?

MIAGE_

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

Sommaire

Remerciements

Dans un premier temps, nous souhaitons remercier Madame Floriane Owczarek qui nous a permis de rejoindre le projet et a répondu à nos questions.

Ensuite, nous remercions grandement Monsieur Nicolas Herbaut, enseignant-chercheur à l'université Paris 1 Panthéon - Sorbonne, membre de la chaire PcEN et chercheur au Centre de Recherche en informatique, pour nous avoir permis de participer au projet Prime-Space, et qui nous a épaulé sur le plan technique, notamment en réalisant divers scripts, comme organisationnel tout au long de ce dernier.

Nous remercions également Léna Albert, étudiante en Master 2 MIAGE IKSEM à l'université Paris 1 Panthéon - Sorbonne, pour ses conseils avisés, son expertise quant au sujet qu'elle nous propose et sa direction du projet, ainsi que pour son aide en concevant une application générant automatiquement des scripts pour le robot.

Pour finir, nous remercions aussi Oualid Guennif, Grasaan Gration et Nasser Harti, tous les trois en L3 MIAGE à l'université Paris 1 Panthéon - Sorbonne, pour l'outil de communication qu'ils ont mis en place, ainsi que tous les autre membres participant à ce méta-projet.

Introduction

Avec l'importante expansion de sa plateforme, le réseau social de partage de contenus vidéo en ligne, YouTube, a dû mettre en place de performants algorithmes pour aider ses utilisateurs à trouver du contenu susceptible de correspondre à leurs centres d'intérêts. L'objectif principal étant de maximiser l'engagement et le temps passé sur la plateforme par ces derniers, et ainsi générer des revenus publicitaires.

Cependant, le contenu conspirationniste et complotiste peut aussi se retrouver mis en avant. C'est ce dont la plateforme a été accusée dernièrement, notamment par le journal "The Guardian" et Monsieur Guillaume Chaslot, ex-employé chez Google ayant travaillé sur l'algorithme de recommandations de YouTube ; et face à quoi elle a répondu avoir remanié cet algorithme dans l'optique de réduire le visionnage de ce type de vidéos.

YouTube promeut-il encore les théories du complot ? Comment identifier et analyser les contenus de nature "conspiratoire" sur YouTube ? Ce sont les questions sur lesquelles se penche Léna Albert. Dans le cadre de ce travail de recherche, nous devons développer une solution d'outils analytiques permettant la collecte d'informations de navigation et l'extraction de données sur cette plateforme.

Ce projet fait partie du méta-projet "Streaming for Good", réunissant des étudiants en L3/M1/M2 MIAGE classique ou apprentissage à l'université Paris 1 Panthéon - Sorbonne, autour de recherches sur les plateformes de diffusion de vidéo.

L'ensemble s'articule autour des travaux du projet "Prime-Space", des recherches de l'école des médias et du numérique de la Sorbonne (EMNS), et de la chaire Pluralisme culturel et Éthique du Numérique (PcEN).

Analyse

1. Analyse du terrain

Comme précisé précédemment, l'objectif est d'accumuler un grand nombre de vidéos afin que Léna Albert puisse les étudier dans son travail de recherche. Ainsi, il faut trouver un moyen de collecter une grande quantité de données dont on peut retracer le parcours : quelles sont les actions effectuées par les robots, les mots clefs trouvés et utilisés, les recherches réalisées et les différentes vidéos visionnées qui ont amené au visionnage, ou à l'apparition dans les recommandations de cette vidéo. Elles seront ensuite identifiées à l'aide du titre, des tags qui lui sont attachés, les commentaires postés en dessous cette vidéo, ses éventuels sous-titres, et cætera...

Afin de procéder à ces analyses statistiques, il faut premièrement disposer d'une quantité conséquente de vidéos, et de couvrir un large éventail de thèmes différents (lifestyle, vlog, jeux vidéos, tutos, podcast, et cætera...) de manière à affiner l'analyse au maximum. La seconde étape consiste à envoyer les données récoltées à une base de données, qui permettra d'archiver et d'exploiter toutes les informations ainsi récupérées. Dans un troisième temps, toutes les informations indépendantes du comportement du robot (tel que le titre, les commentaires, les sous-titres, et cætera...) devront être traitées et conservées dans la base de données, de manière à être en capacité de décomposer et comprendre ce qui a amené aux résultats obtenus.

Ainsi, le projet se veut le plus flexible possible mais le plus simple quant à son utilisation. À partir du moment où les paramètres du robots sont donnés, tout le système se met en place et archive toutes les données récoltées.

L'outil qui est nécessaire à Léna pour la réalisation de son mémoire est en soi assez spécifique, d'autant plus qu'il s'agit d'un projet de recherche, et que donc les besoins sont amenés à évoluer avec le temps et l'avancée de ce dernier. De ce fait, avoir une équipe de développement à disposition et avec laquelle interagir est un réel bénéfice. Il n'y aurait pour alternatives que de réaliser la collecte de données manuellement ou de développer une solution logicielle par elle-même.

2. Conduite du projet

La réalisation du projet s'est accompagnée de problématiques organisationnelles que nous avons résolu comme suit :

Chaque une à deux semaines, nous organisions des réunions avec l'ensemble des acteurs du projet, incluant Monsieur Nicolas Herbaut et Léna Albert. Ces entretiens duraient généralement une à deux heures et avaient pour objectif d'informer tous les membres de l'avancée des différentes étapes et tâches, ainsi qu'à poser les différents problèmes rencontrés, d'éventuelles questions, et à planifier le travail pour la ou les semaine(s) suivante(s).

Pour ces réunions, nous avons majoritairement utilisé deux outils. Dans un premier temps, nous utilisions "Zoom" notamment grâce à la licence détenue par l'université. Monsieur Nicolas Herbaut nous fournissait une salle de réunion dans laquelle nous avons pu échanger en interne au sein de l'équipe, mais aussi à de plus rares occasions avec l'ensemble du méta-projet. Dans un second temps nous avons utilisé "Discord". Un serveur regroupant tous les membres du méta-projet qui a été mis en place par les L3 classiques et nous a permis d'organiser des réunions sans avoir le besoin de créer de salle de réunion au préalable à chaque rendez-vous.

Un des avantages majeurs de Discord par rapport à Zoom, c'est la possibilité de créer des canaux textuels et d'échanger des messages avec les différents utilisateurs, à l'image de Slack, mais en centralisant à la fois les appels audio/vidéo, et les canaux textuels. Nous nous sommes grandement servi de cet outil pour communiquer. Nous avions à la fois le serveur créé par les L3 classique à disposition pour informer l'ensemble des membres du projet et donner une visibilité aux membres du méta-projet ; le salons de messagerie privée entre deux utilisateurs pour les échanges rapides d'informations, de fichiers, les demandes d'aide, … entre les membres du projet et la "cliente" ; et un groupe ne réunissant que les trois membres directs du projet sur lequel nous communiquions aussi.

En début de projet, nous avions envisagé d'utiliser les méthodes agiles, notamment SCRUM, pour la planification des tâches. Un tableau Trello a été mis en place, et le travail réparti en User Stories. Cependant, l'idée a vite été écartée car cela ajoutait une charge organisationnelle supplémentaire qui ne semblait pas nécessaire au vu de la taille réduite de l'équipe de développement. Les différentes tâches ont été réparties en fonction des compétences, envies et disponibilités de chacun lors des réunions (bi)hebdomadaires.

Aussi, tout au long du projet, chaque membre a utilisé GitHub pour conserver des traces de son travail, des versions, et pouvoir regrouper facilement l'intégralité des codes en un seul et même endroit.

3. Qualité attendue

Il est attendu que toute l'architecture fonctionne avec le moins d'interaction humaine possible. A partir du lancement du robot, qui peut être automatisé à l'aide d'un planificateur de tâches, tout se fait par la suite sans intervention humaine, et ce jusqu'à l'analyse des données.

Il serait préférable de pouvoir envoyer un script d'instruction au robot, en variant les paramètres, le tout à distance. Ainsi, le robot va lire des scripts disponibles sur un serveur, et s'exécuter. De plus, des tests automatisés validant les fonctions existantes serait un bon ajout. En terme de maintenance, un fichier retraçant les logs du robot pourrait être un bon ajout. Il faut aussi que d'éventuelles nouvelles fonctions puissent s'implémenter de la manière la plus simple à l'existant. Les échanges de données avec l'API doivent pouvoir évoluer de manière simple et rapide, si jamais une restructuration de la base de données devait arriver.

Sachant que les données récoltées par le robot ne sont pas suffisantes, il faut effectuer plus de traitement avant de les stocker. Ainsi, un système de queue pour effectuer ses traitements est une bonne manière de procéder pour effectuer cette deuxième récolte de données. Ensuite, il faut que les données puissent être accessibles par un individu pour les observer et les mettre à jour via une visualisation HATEOAS (Hypermedia As The Engine of Application State) utilisant des hypermédias.

Un dernier point important est le développement de chaque partie de l'architecture globale comme indépendante des autres parties ; si jamais une partie est hors service pour une raison ou une autre, que cela n'empêche pas le reste de la structure de fonctionner. Il s'agit donc de développer les technos comme étant des blocs indépendant les uns des autres, ne communiquant entre elles que pour alimenter leurs données.

4. Le système : vue fonctionnelle

Système entier

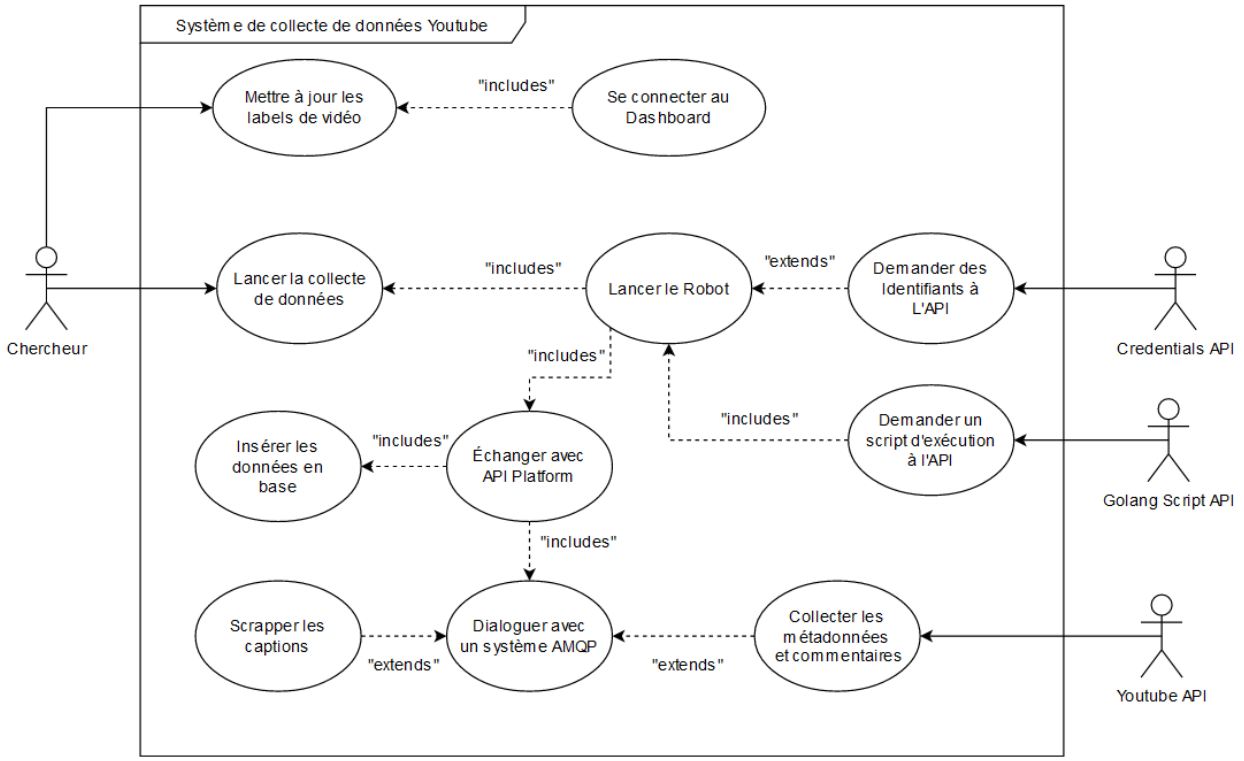


Figure 1 : Diagramme de cas d'utilisation du système

Le chercheur va lancer une collecte de données. Le robot va donc démarrer, demander à l'API la génération de scripts JSON en Golang qui est une suite d'instructions (il s'agit d'une liste de dictionnaire qui indique au robot quelle action effectuer) dont voici un court exemple, qui va ensuite être exécutée par le robot. Si le robot doit se connecter à un compte Google, il va pouvoir faire appel à Credentials API, qui contient des comptes créés spécialement pour le robot. Au fur et à mesure du scraping de YouTube, le robot va envoyer les données récupérées à l'API qui elle-même va implémenter toutes ces informations à la base de données, qui va les allouer dans une table en indiquant que ces données doivent être traitées. De manière asynchrone, les données vont être mises dans une queue et des scripts vont récupérer les sous-titres mais aussi les métadonnées et les commentaires de chaque vidéos, et les renvoyer à la base de données, où elles seront stockées pour que le chercheur, en se connectant à son dashboard, puisse voir les données traitées et modifier les labels.

Robot Python / Selenium

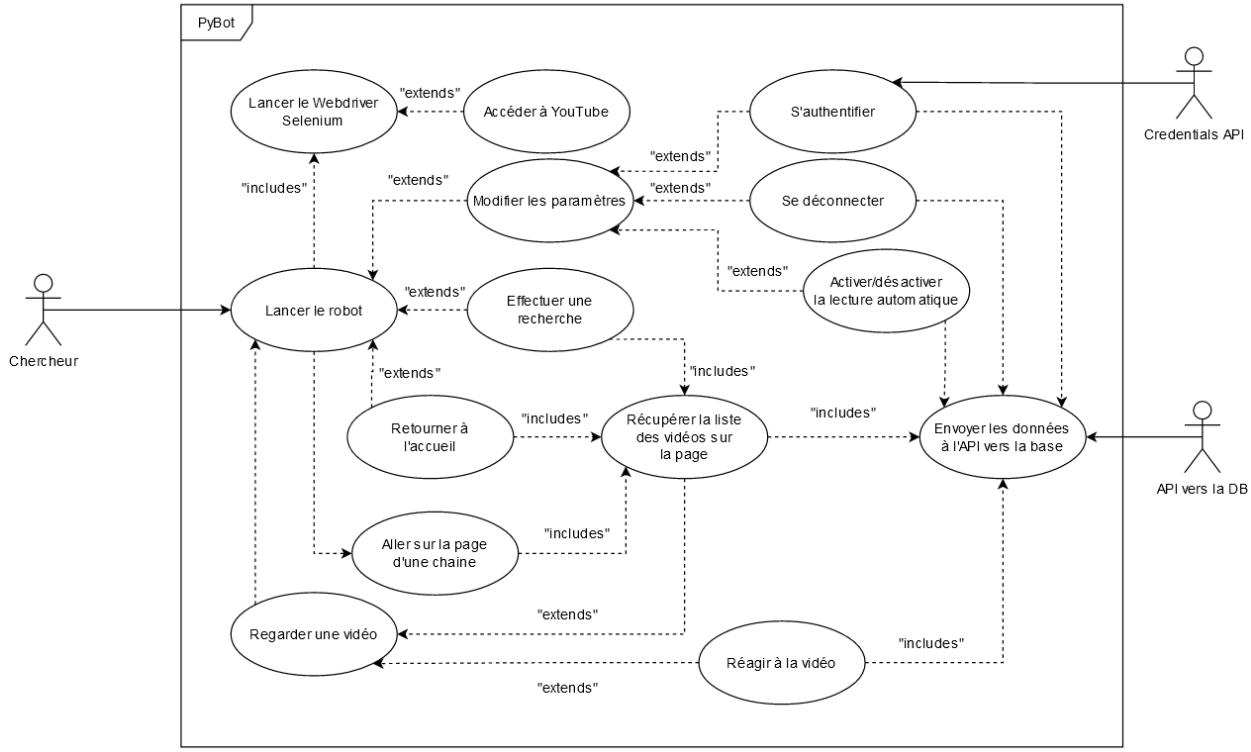


Figure 2 : Diagramme de cas d'utilisation du robot Python

La philosophie derrière le robot est de décomposer chaque action du programme, dans une logique Agile, à travers une fonction, ce qui permet d'avoir une action par fonction. On peut différencier deux grands types de fonctions : il s'agit des fonctions qui jouent sur le web drivers et les données YouTube, et les fonctions qui enveloppent le tout pour que le robot puisse agir selon une liste d'instructions.

Un des objectifs principaux est de minimiser les interactions utilisateur de manière à limiter l'impact de ce dernier sur les résultats obtenus. Avec un système automatisé, on limite les différents biais induits par les comportements humains tout en étant en capacité de simuler une navigation humaine sur la plateforme en créant une forme d'aléatoire de par la multitude d'options de navigation disponibles.

5. Le système : vue statique

Système entier

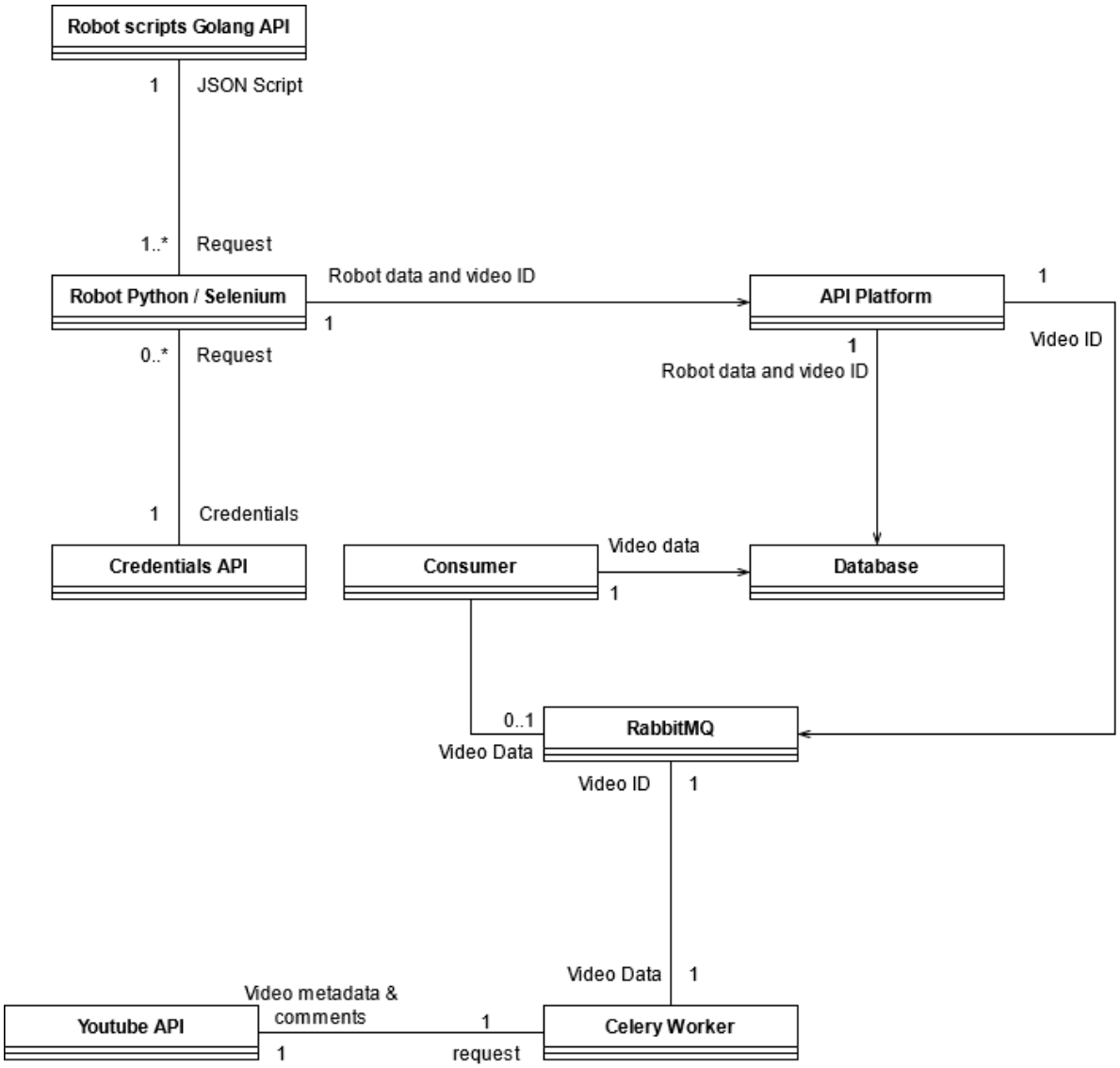


Figure 3 : Diagramme de classe des échanges du système

API Platform

Au sein d'API Platform, afin de communiquer avec le robot PyBOT, nous avons réalisé deux méthodes "*POST*". La première consiste à récupérer la session déclenchée par le robot qui renverra au format JSON le paramètre suivant :

```
{
   id": "sessionID"
}
```

La seconde permet de récupérer les différentes actions effectuées par PyBOT qui transmettra au format JSON les paramètres suivants:

```
{
  "session": "sessionID",
  "action": 0,
  "index": 0,
  "position": 0,
  "currentVideo": "videoID",
  "videos": [
    "videoID"
  ],
  "key_word": "word"
}
```

Par ailleurs, pour afficher les données à l'aide du modèle architectural "HATEOAS", nous avons utilisé plusieurs méthodes "*GET*" :

- **All Sessions**
  - /api/sessions: permet d'afficher toutes les sessions existantes.
  - /api/session/{id}: permet d'afficher une session {id}.

- **All Logs**
  - /api/logs: permet d'afficher toutes les logs existantes.
  - /api/log/{id}: permet d'afficher une log {id}.

- **All Captures**
  - /api/captures: permet d'afficher toutes les captures existantes.
  - /api/capture/{id}: permet d'afficher une capture {id}.

- **All Videos**
  - /api/vidéos: permet d'afficher toutes les vidéos existantes.
  - /api/video/{id}: permet d'afficher une vidéo {id}.

- **All VideosLabel**
  - /api/vidéos/label: permet d'afficher toutes les vidéos existantes avec leur label associé.
  - /api/video/label/{id}: permet d'afficher une vidéo {id} avec son label associé.

- **All Comments**
  - /api/comments: permet d'afficher tous les commentaires existants.
  - /api/comment/{id}: permet d'afficher un commentaire {id}.

- **All Captions**
  - /api/captions: permet d'afficher l'intégralité des sous-titres existants.
  - /api/caption/{id}: permet d'afficher une caption {id}.

- **All Actions**
  - /api/actions: permet d'afficher toutes les actions existantes.
  - /api/action/{id}: permet d'afficher une action {id}.

- **All Channels**
  - /api/channels: permet d'afficher tous les channels existants.
  - /api/channel/{id}: permet d'afficher un channel {id}.

- **All ChannelsLabel**
  - /api/channels/label: permet d'afficher tous les channels existants avec leur label associé.
  - /api/channel/label/{id}: permet d'afficher un channel {id} avec son label associé.

Cf => Annexes : Architecture HATEOAS API Platform

Enfin, en ce qui concerne les méthodes, nous avons utilisé 2 "*PATCH*" qui permettent de modifier le contenu d'une ligne liée aux tables "video_label" et "channel_label" en renvoyant un JSON:

```
{
  "videoLabelID": id,
  "label": "labelID",
  "description": "text"
}
{
  "channelLabelID": id,
  "label": "labelID",
  "description": "text"
}
```

En parallèle, lors de l'utilisation de la méthode "*POST*" permettant de récupérer les différentes actions effectuées par le robot, nous lançons un message dans la queue "Celery" de RabbitMQ si la/les vidéo(s) reçue(s) n'ont aucune information dans la base de données. Suite à ça, les messages en attente dans la queue seront consommés par des scripts Python pour récupérer les informations à partir de l'API Youtube (metadata & commentaires) ou directement depuis la page vidéo_id Youtube (captions) et les renvoient dans une queue "youtube-response". A l'aide du bundle symfony RabbitMQ nous consommons à des heures précises tous les messages en attente afin de les incorporer en base.

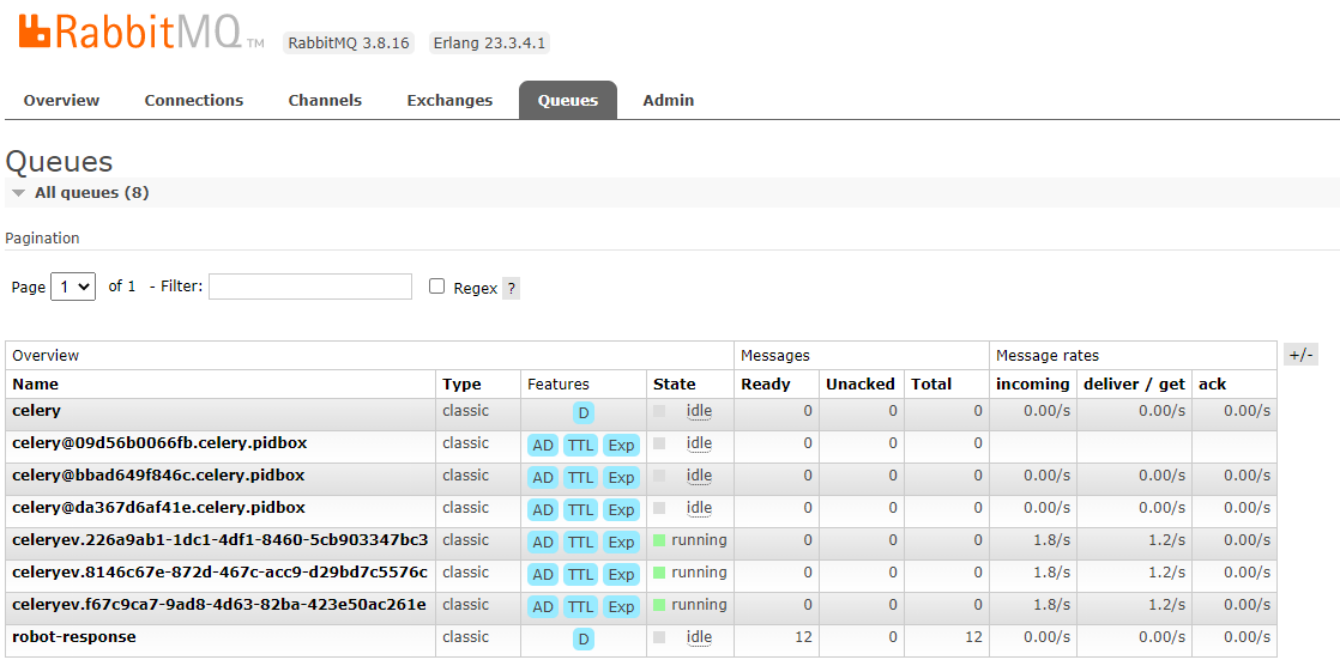


Figure 4 : Capture d'écran de RabbitMQ

Robot Python / Selenium

La fonction “launch()” va sur la YouTube, puis soumet une requête à “https://scriptgenyoutube.miage.dev/generate” pour récupérer un json avec comme clef “actions” et en valeur une liste de dictionnaire ; chaque dictionnaire étant une action que doit effectuer le robot. Le robot est ensuite appelé avec cette liste. Il va itérer sur chaque dictionnaire, et exécuter ce qui lui est demandé. Voici un exemple de fichier récupéré.

Dans la structure de “robot(file)”, il est important de garder en tête quelques variables clefs.

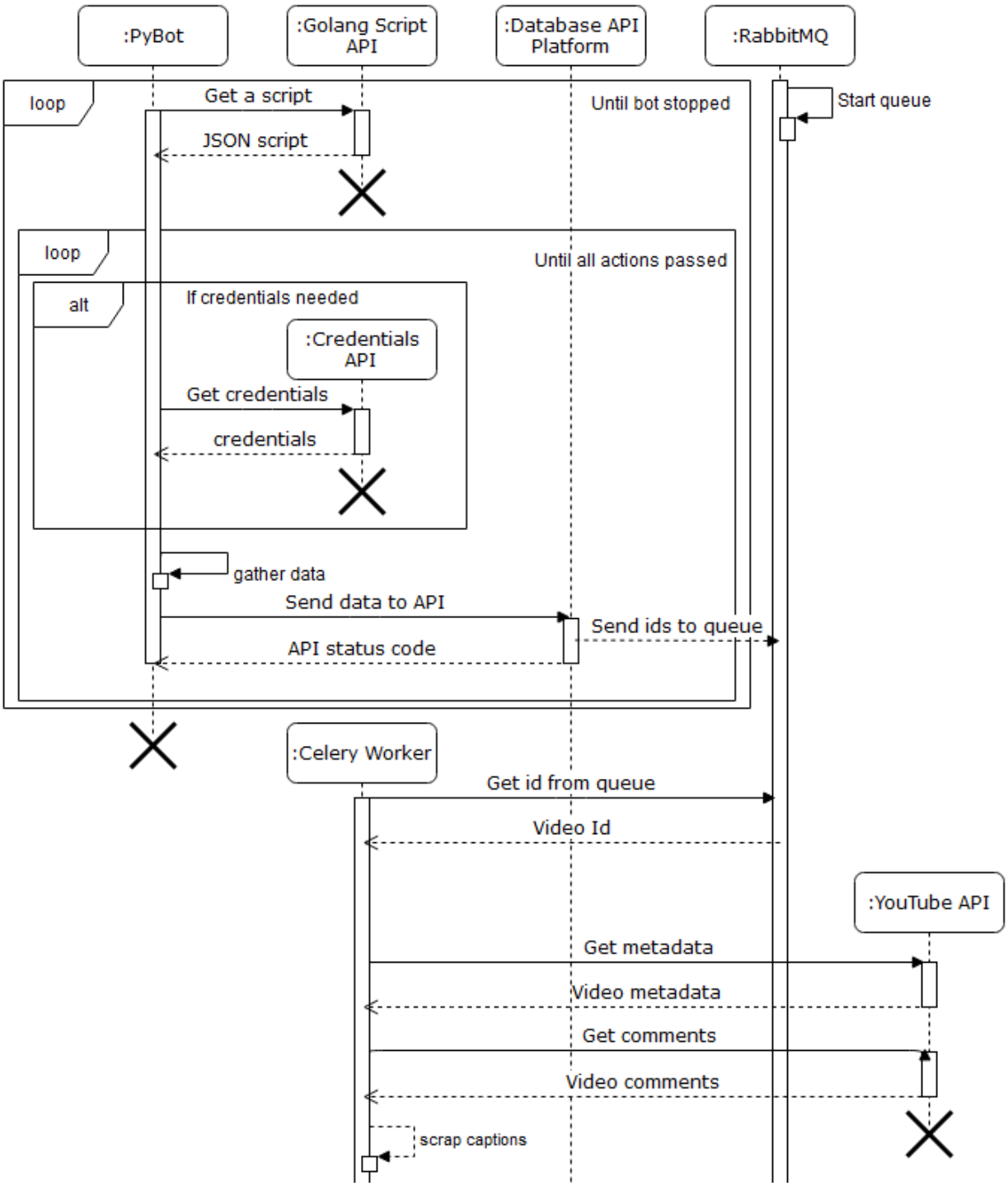
- currentAction est un entier qui est relié dans la base de donnée à une action. À chaque envoi de requête vers la base de données, currentAction est envoyé en paramètre.
- actionNumber est un entier qui est implicitement lié à la session. Il permet d’historiser toutes les actions que fait le robot. A chaque envoi de requête vers la base de données, actionNumber est envoyé en paramètre et est systématiquement incrémenté.

Voici une tableau récapitulatif :

NOM	IDENTIFIANT	PARAMÈTRE ENVOYÉS
settings	1	{"session", "action", "position"} {"session", "action", "email", "position"} *(email est un paramètre optionnel)*
search	2	{“session”, “action”, “videos”, “key_word”, “position”}
watch	3	{“session”, “currentVideo”, “action”, “videos”, “index”, “position”}
like	4	{“session”, “action”, “position”}
dislike	5	{“session”, “action”, “position”}
goToChannel	6	{“session”, “action”, “videos”, “position”}
home	7	{“session”, “action”, “videos”, “position”}

6. Le système : vue dynamique

Système entier



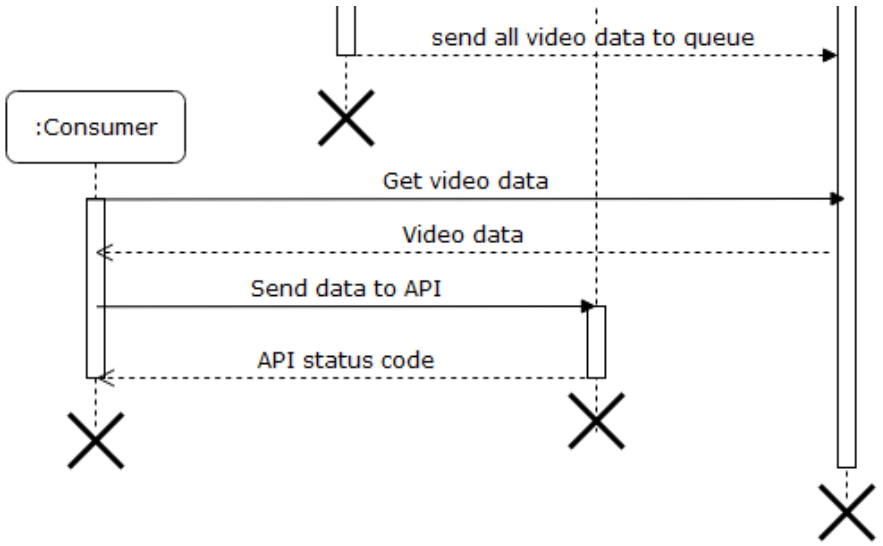


Figure 5 : Diagramme de séquence du système global

Sur le plan des interactions internes au système, le robot, dès son lancement, va interroger l'API de génération de script pour recevoir ses instructions, et ce en boucle jusqu'à être arrêté.

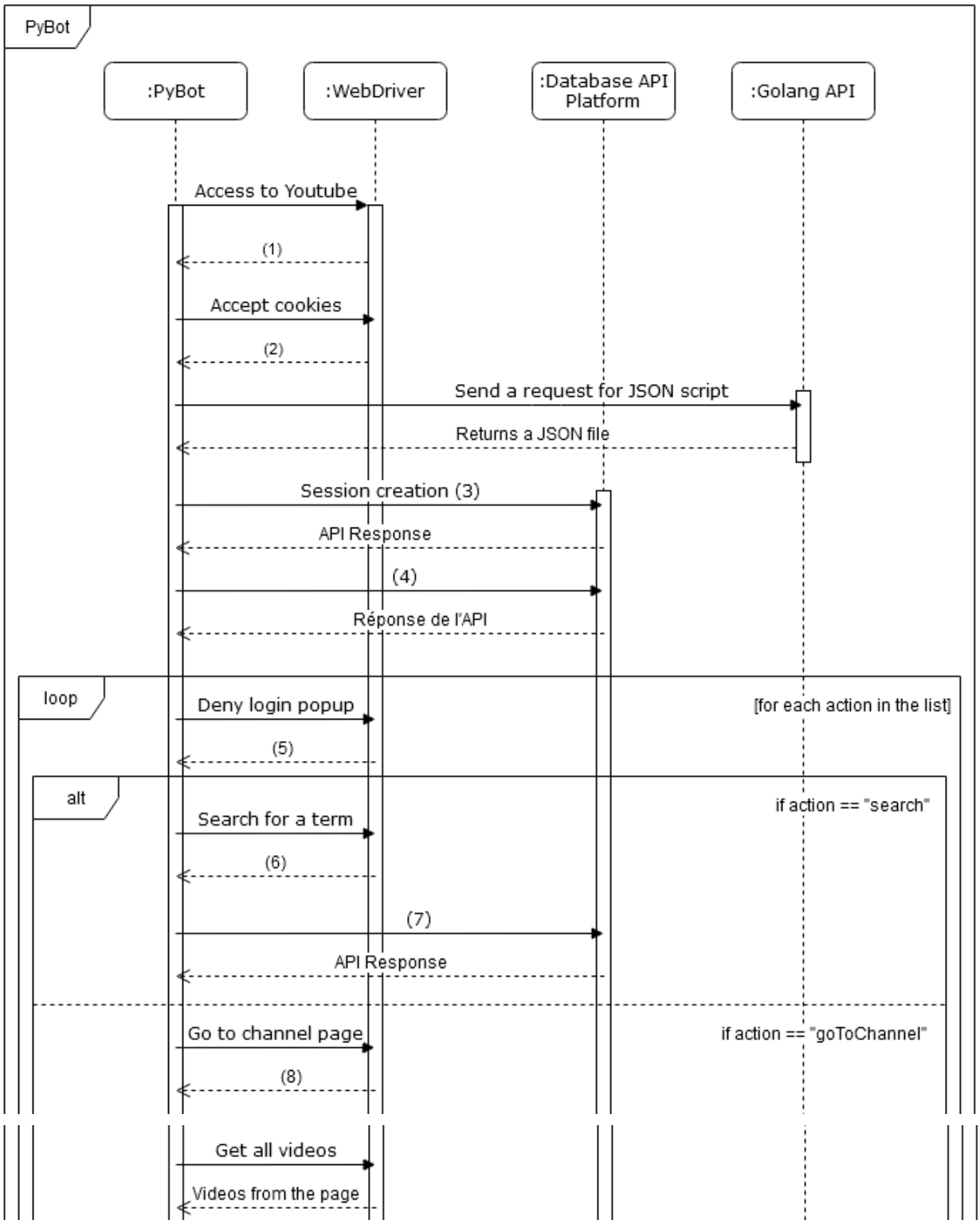
Il va itérer sur chacune des instructions et les exécuter dans le webdriver, et si nécessaire, il pourra aussi récupérer de quoi s'authentifier auprès de l'API d'obtention d'identifiants. Chacune des actions retourne des données qui seront collectées et envoyées à l'API Platform qui se chargera de les intégrer à la base de données et aussi de les mettre en queue RabbitMQ.

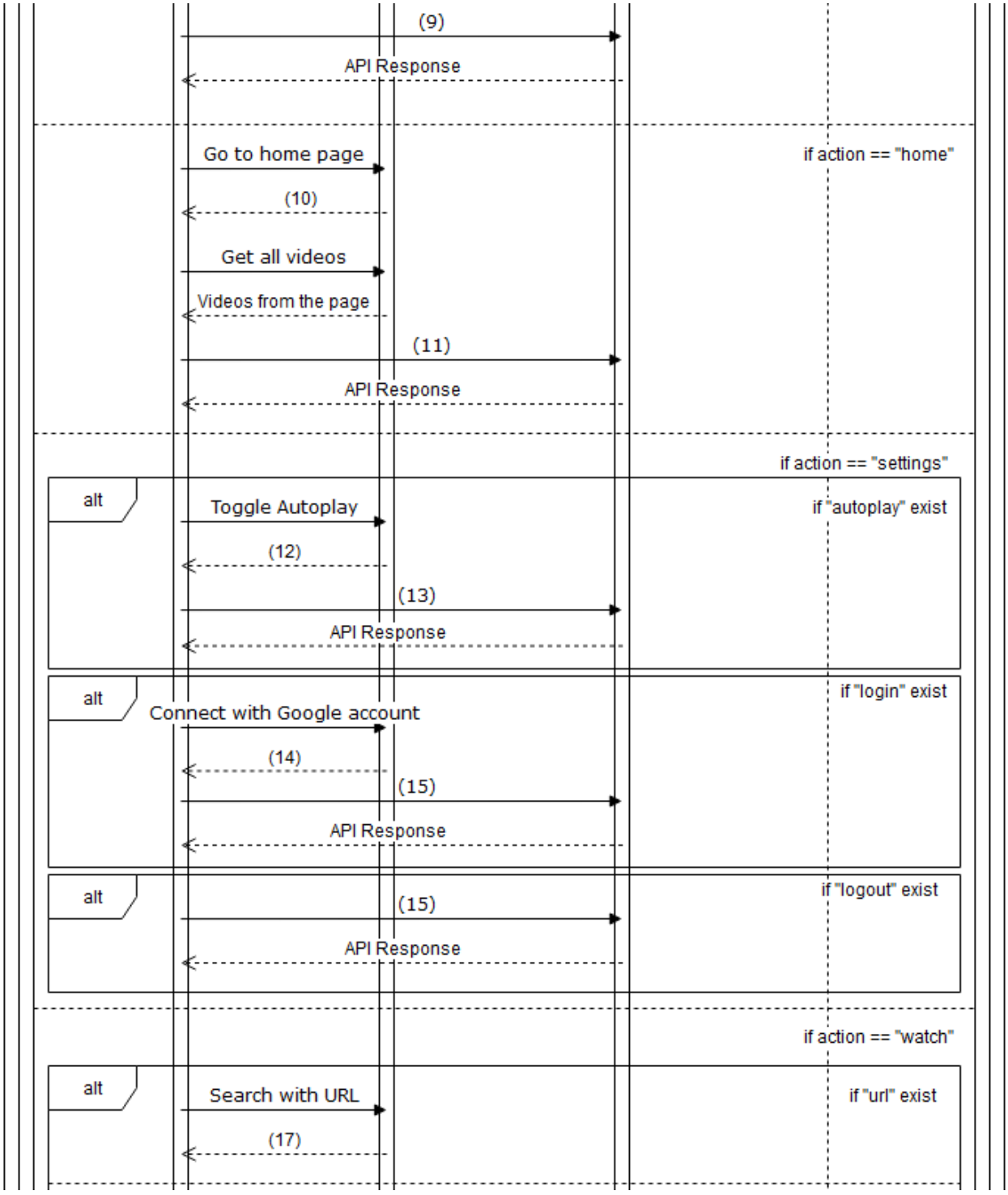
Des workers iront consommer les identifiants de vidéos dans la queue pour ensuite aller interroger l'API de YouTube pour en extraire les métadonnées et les commentaires, puis aller scrapper les sous-titres directement sur le site YouTube avant de renvoyer toutes ces informations dans une autre queue.

Finalement, un consommateur se chargera de récupérer ces données et de les envoyer à l'API qui insèrera tout ce qui a été récolté dans la base de données.

Toutes ces opérations sont engagées dans une boucle infinie qui ne s'arrêtera que lorsque les différents services et clients seront stoppés.

Robot Python / Selenium




(9)
API Response
Go to home page
if action == "home"
(10)
Get all videos
Videos from the page
(11)
API Response
if action == "settings"
alt
Toggle Autoplay
if "autoplay" exist
(12)
(13)
API Response
alt
if "login" exist
Connect with Google account
(14)
(15)
API Response
alt
if "logout" exist
(15)
API Response
if action == "watch"
alt
Search with URL
if "url" exist
(17)

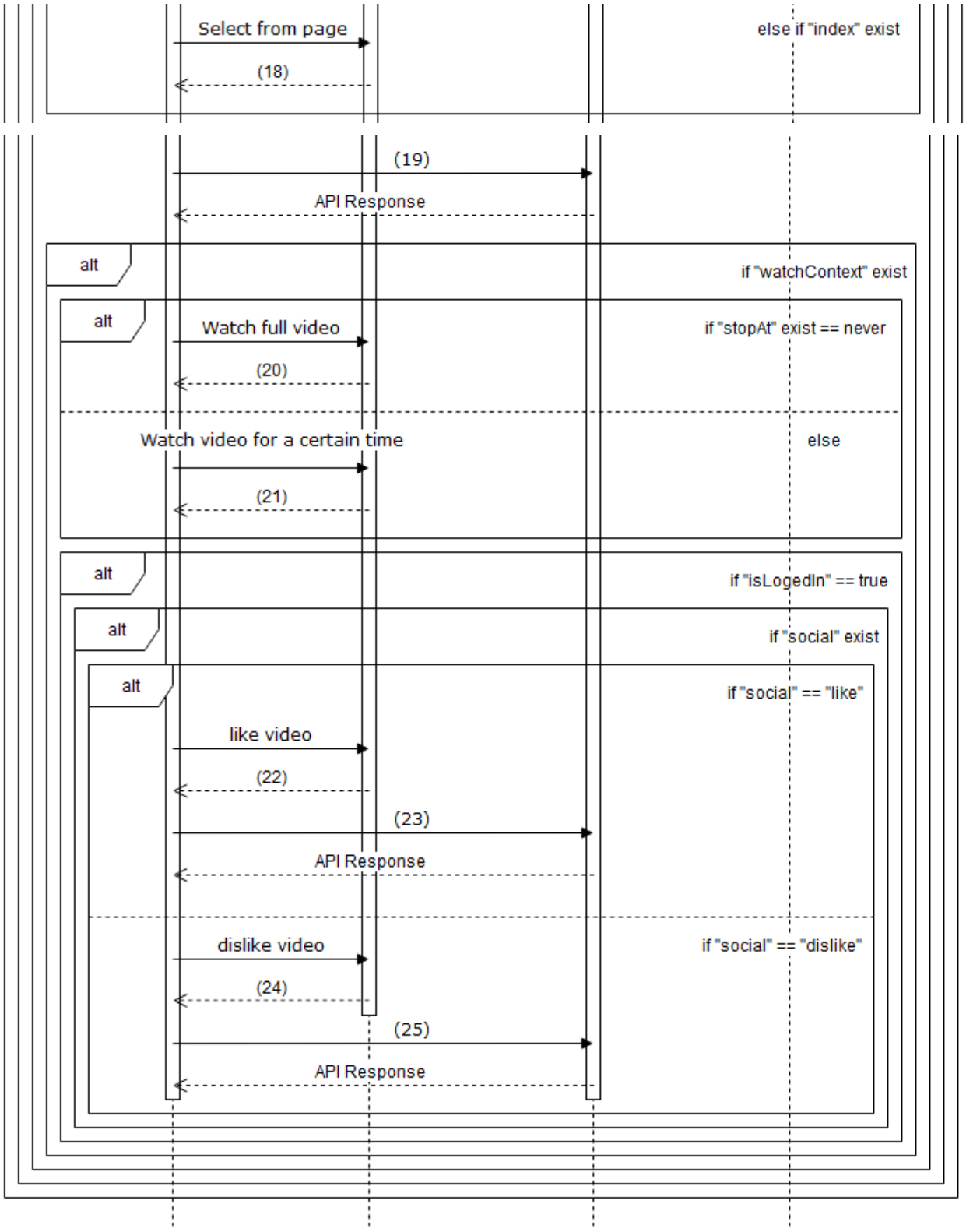


Figure 6 : Diagramme de séquence du Robot Python

(1) Réponse du webdriver, qui renvoie un objet contenant le code de la page actuelle
(2) Lors d'une création de session, un identifiant unique est envoyé à la base de données.
(3) Les données envoyées sont l'identifiant unique créé, la liste de vidéos chargées sur la page actuelle, l'identifiant de l'action et le numéro de l'action effectué
(4) Réponse du webdriver, qui renvoie un objet contenant le code de la page actuelle
(5) Réponse du webdriver, qui renvoie un objet contenant le code de la page actuelle
(6) Les données envoyées sont l'identifiant unique créé, la liste de vidéos chargées sur la page actuelle, les mots clefs de la recherche, l'identifiant de l'action et le numéro de l'action effectué
(7) Réponse du webdriver, qui renvoie un objet contenant le code de la page actuelle
(8) Les données envoyées sont l'identifiant unique créé, la liste de vidéos chargées sur la page actuelle, l'identification de l'action et le numéro de l'action effectué
(9) Réponse du webdriver, qui renvoie un objet contenant le code de la page actuelle
(10) Les données envoyées sont l'identifiant unique créé, la liste de vidéos chargées sur la page actuelle, l'identifiant de l'action et le numéro de l'action effectué
(11) Réponse du webdriver, qui renvoie un objet contenant le code de la page actuelle
(12) Les données envoyées sont l'identifiant unique créé, l'identifiant de l'action et le numéro de l'action effectué
(13) Réponse du webdriver, qui renvoie un objet contenant le code de la page actuelle
(14) Les données envoyées sont l'identifiant unique créé, l'email passé en paramètre (qui peut être une chaîne de caractère vide) l'identifiant de l'action et le numéro de l'action effectué
(15) Les données envoyées sont l'identifiant unique créé, un string contenant "log out", l'identifiant de l'action et le numéro de l'action effectué
(16) Réponse du webdriver, qui renvoie un objet contenant le code de la page actuelle
(17) Réponse du webdriver, qui renvoie un objet contenant le code de la page actuelle
(18) Les données envoyées sont l'identifiant unique créé, l'identifiant de la vidéo,la liste de vidéos chargées sur la page actuelle, l'identifiant de l'action et le numéro de l'action effectué
(19) Réponse du webdriver, qui renvoie un objet contenant le code de la page actuelle
(20) Réponse du webdriver, qui renvoie un objet contenant le code de la page actuelle
(21) Réponse du webdriver, qui renvoie un objet contenant le code de la page actuelle
(22) Les données envoyées sont l'identifiant unique créé, l'identifiant de l'action et le numéro de l'action effectué
(23) Réponse du webdriver, qui renvoie un objet contenant le code de la page actuelle
(24) Les données envoyées sont l'identifiant unique créé, l'identifiant de l'action et le numéro de l'action effectué

Chronologiquement, le robot commence par soumettre une requête pour récupérer un programme à exécuter du côté de l'API de génération de scripts, qu'il parcourt ensuite. À chaque bloc du JSON qu'il a reçu, il va parcourir la liste de ses actions disponibles stockée dans un dictionnaire jusqu'à trouver celle correspondante. Chacune d'entre elles est composée de une à plusieurs sous-tâches qui seront ou non elles aussi réalisées. Les actions actuellement possibles sont les suivantes et sont décomposées comme suit :

- Settings :
    - Activer / désactiver la lecture automatique des vidéos
    - Se déconnecter
    - Se connecter
        - Demande ou non des identifiants à l'API concernée
- Search :
    - Entre des mot-clef dans la barre de recherche, exécute cette dernière et en récupère les résultats
- goToChannel :
    - Va sur la page d'une chaine YouTube, dans l'onglet "Vidéos" et récupère la liste de toutes les vidéos chargées
- Home :
    - Va sur la page d'accueil de Youtube, et récupère la liste de toutes les vidéos chargées
- Watch (se fait soit depuis un URL passé en paramètres soit depuis un click sur une vidéo) :
    - Récupérer la liste de toutes les vidéos chargées
    - Suit les différentes demandes du script (regarder la vidéo en entier, partiellement, s'il y a un compte authentifié : aimer ou ne pas aimer une vidéo)

Chacune des sous-tâches de chaque action retourne en une ou plusieurs requêtes des données à la base de données au travers de l'API.

7. Choix d'implémentation

Base de données

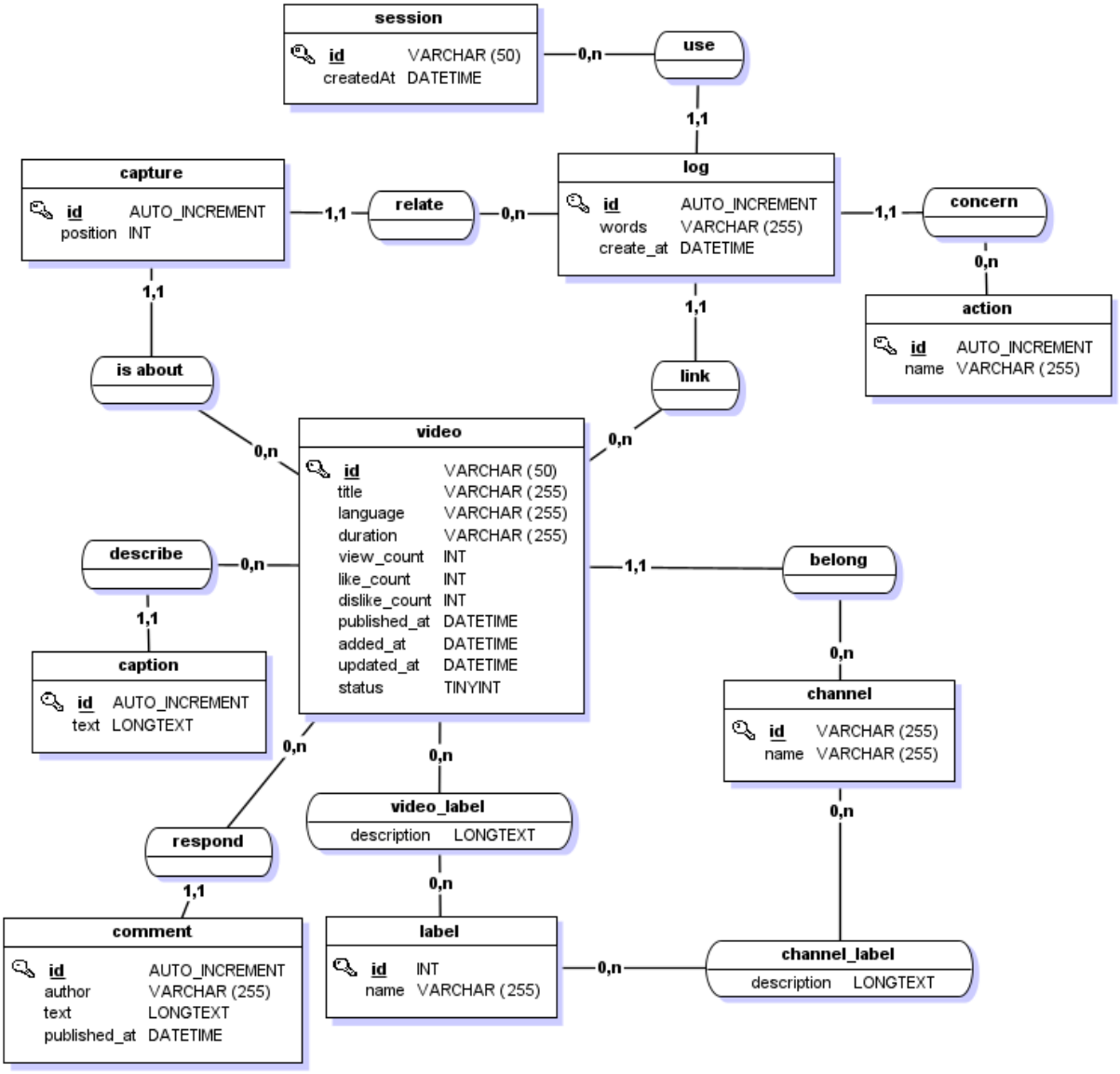


Figure 7 : Modélisation Merise (équivalent E/R) de la base de données

Dans un premier temps, il a été primordial d'établir une base de données. Pour cela, nous nous sommes posés la question suivante: Doit-on partir sur une base de données SQL ou NoSQL ?

Par des contraintes de temps, nous avons opté pour SQL car il nous a paru compliqué de partir sur du NoSQL étant donné qu'aucun membre de notre équipe n'en maîtrisait son fonctionnement. Pour résumer la figure 5, la conception de cette base de données s'articule en plusieurs points :

Une première partie permet de récupérer les données du Robot PyBOT et de les mettre dans les tables :

- "Session" **:** elle permettra d'identifier de façon unique le lancement d'un nouveau script json
- "Log" **:** elle a pour but de récupérer les différentes actions de PyBOT et d'en avoir un historique. Elle est liée directement à la table capture.
- "Capture" **:** elle permet de récupérer les actions {2,3,6,7} envoyées initialement dans la table "Log" et d'incorporer plusieurs vidéos youtube qui sont sur la page que le PyBOT visite.

La seconde partie permet de collecter les informations renvoyées par l'API Youtube & le script de récupération de captions afin de les incorporer dans les tables:

- "Vidéo" **:** elle récupère les informations de chaque vidéo youtube.
- "Comment" **:** récupère les commentaires renvoyés par l'API Youtube.
- "Caption" **:** récupère les captions renvoyées par le script associé.

En ce qui concerne les tables "video_label", "channel", "channel_label" et "label", nous avons décidé de les dissocier de la table "video" afin d'avoir une meilleure compréhension. En effet, cela permet de répondre aux besoins de Léna Albert. Parmi ses attentes, elle avait besoin de labelliser chaque vidéo et channel indépendamment avec une description.

Robot Python / Selenium

Figure 8 : Logo de Sélénium et de Python

Nous avons choisi de développer le robot avec Python, car nous maîtrisons plutôt bien ce langage et facile d'utilisation et de développement. Selenium est un outil gratuit de web scraping facile de prise en main, bénéficiant d'un Framework dans une extension Chrome et d'une librairie Python. La combinaison des deux, mêlant simplicité de prise en main et efficacité d'exécution, nous ont poussé à utiliser ces outils pour le robot.

API PLATFORM - Symfony

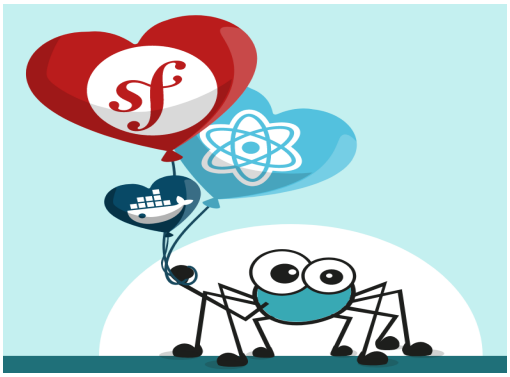


Figure 9 : Logo de Platform API

Nous avons choisi de coupler la partie Robot/Selenium avec la librairie API Platform de la distribution Symfony (framework français "Cocorico") qui permet de créer rapidement et simplement des API REST. En outre, la librairie permet d'avoir une page qui permet de nous récapituler toutes les méthodes (GET, POST, PUT, PATCH, DELETE) et de les tester à partir de curl. On s'est rapidement porté sur ce choix puisqu'une personne de notre groupe connaissait cette librairie. De plus, le fait que Symfony ait plusieurs bundles utiles comme RabbitMQ, nous a définitivement convaincus de son utilisation.

Architecture HATEOAS

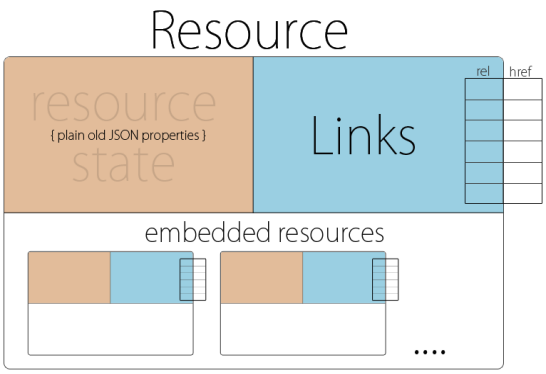


Figure 10 : Schéma de l'architecture Hateoas

Le choix de l'architecture HATEOAS (Hypermedia As The Engine of Application State) pour agencer nos API REST s'est justifié par la flexibilité d'utilisation, malgré tout dans un environnement contrôlé, qu'elle permet. En effet, les échanges restent sous les protocoles classiques REST, mais la source d'action et d'interaction avec le système est le contenu hypermédia (XML, JSON, …) et non plus la requête elle-même. Cela permet à l'API de se décharger de la compréhension de ce qu'elle transporte. Aussi, l'accès à l'application se fait par une URL fixe, chacune des possibilités futures d'interaction est retournée par le serveur.

RabbitMQ

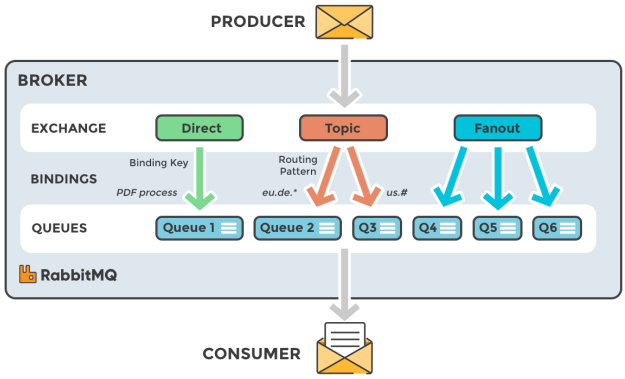


Figure 11 : Schéma du fonctionnement de RabbitMQ

Le choix de RabbitMQ comme système communiquant majeur, a été là pour pallier aux différents problèmes que nous aurions pu rencontrer. En effet, l'un problème de la communication par une API classique c'est que si un des deux côtés est indisponible pour une panne, un problème réseau ou autre, les données envoyées à l'API sont "perdues", là où RabbitMQ dispose d'un système de file d'attente sur un serveur indépendant. Il s'agit en réalité d'ajouter un serveur supplémentaire qui se charge de conserver les données le temps qu'elles soient traitées par l'un des consommateurs de la queue.

De plus, ce système permet de gérer une volumétrie importante, de nombreux messages peuvent être mis en attente dans la file, et plusieurs files peuvent être créés pour les différents types de messages.

Docker

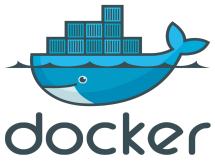


Figure 12 : Logo de Docker

Docker est un logiciel de conteneurisation qui permet la création et l'utilisation de conteneurs. La technologie Docker permet de traiter les conteneurs comme des machines virtuelles très légères et modulaires. En outre, ces conteneurs offrent une grande flexibilité. Cela nous permet de pouvoir créer, déployer, copier et déplacer un environnement à un autre aisément. Nous avons décidé de nous porter sur ce choix pour différentes raisons: tout d'abord, la modularité, en effet, l'approche de Docker en matière de conteneurisation repose sur la décomposition des applications et permet de mettre à jour une application sans forcément désactiver l'ensemble de cette dernière. De plus, la restauration est la fonction la plus intéressante puisque chaque image est composée de couches. Ainsi, si l'image ne nous convient pas , nous pouvons restaurer la version précédente. Enfin, le déploiement rapide est une force indiscutable de Docker. Avant il fallait quelques jours pour mettre opérationnel une machine, maintenant, en quelques secondes grâce aux conteneurs, cela est possible.

Architecture

1. Schéma

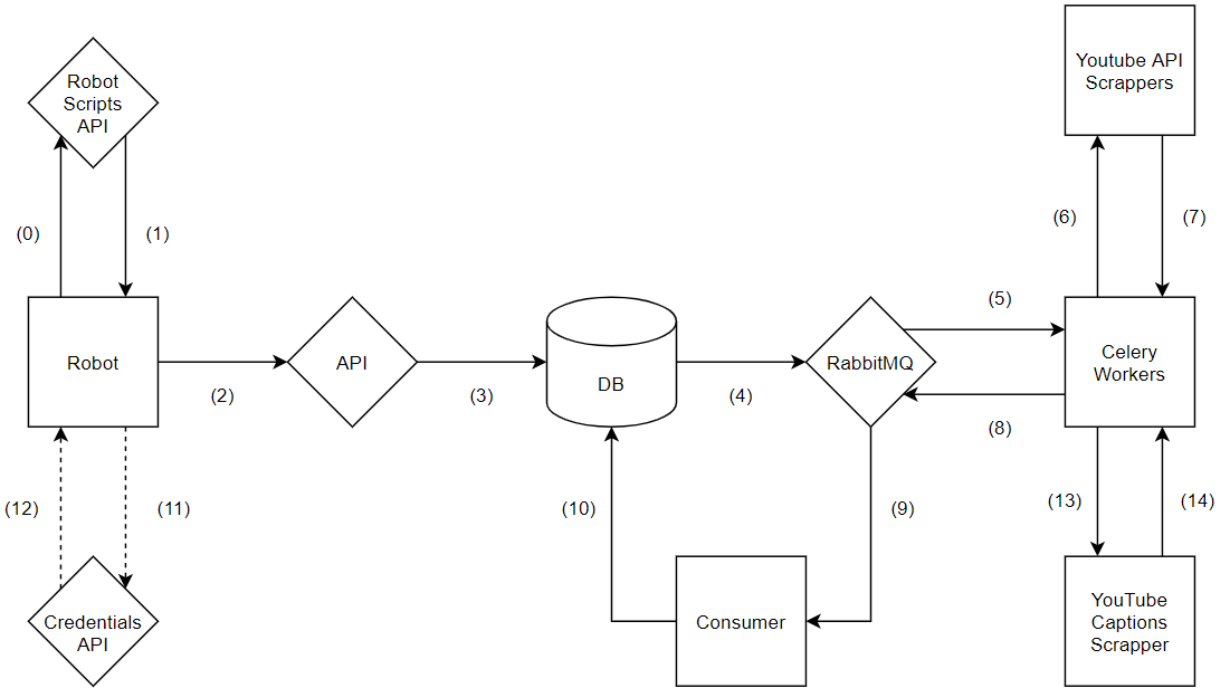


Figure 13 : Schéma global des interactions entre les différents composants

Dans un premier temps, un robot Python va demander à l'API Golang de lui générer un script JSON (0) pour qu'il puisse naviguer de manière automatisée, d'après les instructions reçues (1), dans un navigateur Google Chrome déployé avec Sélénium.

Dans le cas où le script JSON généré par l'API Golang demande au robot de se connecter à un compte Google, le robot va faire une requête à Credentials API (11), avec l'email qui est passé en paramètre, et récupérer le mot de passe (12) pour se connecter. Dans le cas où aucune adresse email est passée en paramètre, alors le robot s'authentifie avec un compte choisi au hasard.

Toutes les informations collectées par le robot sont envoyées en JSON vers une API PHP Symfony (2) qui va se charger de les insérer dans la base de données MySQL (3), ainsi que de l'envoi vers RabbitMQ des identifiants de vidéos (4). Ils seront par la suite consommés par des workers Python Celery (5) qui vont se charger de distribuer les tâches entre les robots de scrapping (6, 13).

D'un côté, les métadonnées des vidéos seront demandées à l'API de YouTube (11) qui seront retournées jusqu'aux workers (11, 7), de l'autre côté, des robots de scrapping iront récupérer les sous-titres directement sur les pages des vidéos via un navigateur Chrome automatisé par Selenium (14). Les workers Python Celery retournent ensuite ces informations dans une autre queue RabbitMQ (8) et les identifiants n'ayant pas pu être traités dans la queue initiale.

Les données récoltées sont ensuite consommées (9) par un consommer PHP Symfony, qui se charge d'isoler la part des données qui nous intéressent avant d'aller les insérer dans la base de données MySQL (10).

2. Questionnaire

1. *Votre architecture vous permettrait-t-elle de supporter facilement une autre langue utilisateur (français et anglais par exemple) ?*

Cette question ne s'applique pas pour l'instant car l'application de dispose pas encore d'une interface affichant autre chose que les données récoltées qui restent, elles, dans leur langage d'origine pour éviter de dénaturer le contenu, et donc, le résultat.

2. *Évaluez la difficulté du passage de votre application web à du mobile.*

Étant donné qu'il ne s'agit pour l'instant pas d'une application mais d'une succession de micro-services de de clients imbriqués, cette question ne s'applique pas. Cependant, dans le cas où une réalisation de ce genre aurait lieu, avec les nombreuses API entourant le projet ainsi que la base de données centrale, il serait assez simple de développer une solution web puis mobile.

3. *La modularité est un critère important de qualité des architectures (Un composant ou classe n'a qu'un seul objectif). Les composants de votre architecture respectent-ils ce principe de modularité ?*

Concernant le robot Python / Sélénium, dans une majeure partie oui, sauf par exemple pour la partie "watch" du robot qui scrappe YouTube : en effet, ici le robot visionne une vidéo, récupère les vidéos présentes sur la page mais c'est aussi ici que le robot va regarder pendant une certaine durée la vidéo, activer ou désactiver le défilement automatique, liker ou disliker la vidéo en question.

Le reste de l'infrastructure a initialement été développé avec en tête cet objectif de pouvoir utiliser les différents modules par une à plusieurs implémentations dans divers langages, ce qui a nécessité de bien séparer chaque partie. Le passage par des API permet aussi d'avoir un découpage plus précis.

4. *Des éléments de votre architecture seraient-ils réutilisables dans un autre projet ?*

Toute la partie robot et RabbitMQ peuvent être utilisées telles-quelles, avec de très légères modifications de configuration. Les workers Celery peuvent aussi être utilisés dans un autre projet similaire.

5. *Votre architecture permettrait-elle de changer de SGBD facilement ?*

Nous avons réfléchi en amont sur notre architecture afin qu'elle puisse permettre ce genre de changement en l'occurrence passer d'un SGBD de type SQL à l'autre aisément. En revanche, il sera plus compliqué de passer par des SGBD de type NoSQL auquelle la base à un fonctionnement plus permissive.

6. *Si vous deviez changer totalement la réalisation de l'interface graphique, y-aurait-il des composants ou classes inchangé(e)s ?*

Il n'y a pas d'interface graphique pour l'instant ; ainsi la question ne se pose pas. Donc s'il y avait un élément à changer sur cette dernière, ça serait déjà simplement son existence.

7. *Quels seraient les effets d'un éventuel changement dans le schéma de la base de données (ajout d'une nouvelle table ou nouvel attribut, changement du nom d'une table ou d'un attribut, changement sur un type d'attribut, sur une contrainte) dans l'application ?*

Tout dépend du changement qu'on souhaite opérer. Si c'est un ajout d'une nouvelle table, d'un changement de nom ou bien un changement de type d'attribut il faudra penser aux éventuels impacts sur la partie fonctionnelle. En revanche, si c'est un ajout d'un nouvel attribut dans une table, il faut faire attention à ne pas le mettre en "not null" sans quoi s'il y a des données à l'intérieur de cette table, il sera primordial de mettre une valeur par défaut pour n'avoir aucune erreur.

Bilan

1. Retour sur le travail réalisé

Ce projet a été un moyen de nous pousser hors de nos zones de confort en coopérant dans un travail de groupe. C'est une réalisation importante devant être délivrée à un commanditaire. Cet important défi a, dans l'ensemble, été relevé avec succès par notre groupe : nous avons délivré une architecture permettant de récupérer des données en parcourant des pages YouTube, suivant un script qui peut être modifié ; de les envoyer à une base de donnée qui va permettre de récolter des données complémentaire à l'aide d'un système de queue, et d'afficher ces données dans un dashboard, qui offre aussi la possibilité de mettre à jour certaines étiquettes.

Cela va sans dire mais le projet ne s'est pas déroulé sans accroc. Il a été difficile au début de trouver un bon rythme, entre le distanciel et les complications de la vie. Cependant, nous avons su rebondir et adopter de bonnes pratiques pour avancer efficacement et recevoir un feedback régulier, assurant que nous avancions dans la bonne direction. S'ajoute à cette première difficulté la nouveauté de certaines technologies, qui ont nécessité un temps d'apprentissage et d'adaptation, où la progression a été assez lente. Cependant, avec la pratique, nous avons commencé à mieux maîtriser nos outils ce qui nous a permis d'accélérer le processus de production.

Cependant, malgré toutes nos précautions, le projet n'est pas sans faille. Premièrement, il faut remarquer que nous tentons de récolter des données comme si nous étions un utilisateur humain. Cependant, il est très difficile de faire passer un programme informatique pour un être humain sur Internet : de nombreux outils sont mis en place pour identifier les sessions suspectes, qui peuvent être ensuite signalées comme telles, ce qui peut complètement fausser les données récoltées. À ce jour, notre robot est très peu équipé pour se faire passer pour un humain, et il nous est impossible de vérifier s'il est signalé comme tel lors de ses navigations.

Une seconde limite est un changement du code source de YouTube. Une telle modification rendrait obsolète le robot, et nous ne pouvons rien faire pour anticiper cette éventualité. Cependant, il faut garder en tête que le cœur de du robot ne serait pas impacté, et qu'un simple changement de sélecteur est nécessaire pour le corriger.

Une troisième limite se situe du côté de l'analyse et la récupération de plus de données, faite via l'API YouTube. En effet, nous effectuons une opération de lecture de commentaire et de lecture de vidéos (pour récupérer des métadonnées). Chaque opération individuellement coûte 1 crédit. Donc chaque vidéo coûte 2 crédits. Nous avons certes 10 000 crédits offerts quotidiennement par l'API YouTube, ce qui nous fait un total de 5 000 vidéos pouvant être analysées par jour. Cependant, même si ce nombre peut sembler important, chaque vidéo visionnée renvoie en fait une vingtaine de vidéos, et c'est sans compter sur les retours à la

page d'accueil, sur les visites de chaînes et lors des recherches avec des mots-clefs qui renvoient aussi une vingtaine de vidéos. Cela fait qu'à partir d'environ 250 instructions effectuées par le robot par jour, nous risquons de nous retrouver à court de crédits. Pour nuancer, cette limite est déjà prise en compte dans l'implémentation des queues : si jamais une ou plusieurs vidéos ne peuvent être traitées, elles restent dans la queue et seront traitées la prochaine fois (le lendemain dans notre cas). La nouvelle limite est donc la capacité de stockage des queues, ce qui revient à approximativement 10 Gb de données. Pour des petites chaînes de caractères, cette limite semble assez peu préoccupante.

Dans les axes d'évolution envisageables de notre délivrable, le robot et sa discrétion sur Internet est un grand champ d'amélioration possible, dans le but de le rendre le plus humain possible, pour récolter des données de grande qualité. Un autre axe serait d'améliorer le dashboard : avoir une solution simple pour visionner et mettre à jour les labels des vidéos avec une description. La dockerisation de tout le projet est aussi une possibilité, avec tout les changement que cela accompagne : la possibilité de pouvoir proposer des scripts sur une plateforme que le robot récupérerait lors de son exécution, pour avoir des collectes plus fines, et pourquoi pas un tableau de bord qui afficherait toutes les actions entreprises par le robot en temps réel.

2. Retour réflexif sur l'action

Sylvain Vocale

Questionnement: Comment les choix d'implémentations ont-ils eu un impact sur ma méthode de travail ?

Avant toute explication dans ma réflexion à la problématique, j'apprécie de poser quelques bases afin de rentrer dans une compréhension plus poussée.

Ma méthodologie de travail sur le choix des différentes implémentations lors du commencement d'un projet se porte principalement sur ce que je connais afin de perdre le moins de temps possible. Je me suis toujours dit: "qu'il fallait, dans chaque cas, arriver rapidement à l'objectif demandé en faisant abstraction des différents points obligatoires qui pourront être vus une fois le projet en phase terminal." De ce fait, je n'ai jamais poussé ma réflexion au-delà lors de nouveaux projets en comparant le meilleur choix d'implémentation.

Il est vrai que je suis souvent le décisionnaire de mes projets, que j'essaye d'imposer le choix technique car c'est une facilité pour moi. Mais une fois le projet finalisé, je m'aperçois moi-même de certaines choses. Le fait que mon application ne soit pas optimisée, que l'UX ne soit pas assez dynamique.

Après avoir réalisé ce projet, je constate que ma méthode de travail a sensiblement changée, non sur le point de vue relationnel mais sur les choix d'implémentations. Au début du projet, j'ai une fois de plus décidé en fonction des mes compétences que je connaissais le mieux. Mais, une nouvelle fois, je suis rentré dans ma "routine de travail" en faisant abstraction de mes camarades sur ce qu'il allait implémenter de leur côté.

Cette fois, la notion de travail en groupe m'a rappelé à l'ordre. Je n'étais pas seul dans ce projet et je n'avais pas la même liberté de choisir comment développer un module pour plusieurs raisons: chacun devait faire sa partie de développement et chacun avait ses forces et faiblesses. De plus, l'optimisation était un facteur clé que je ne pouvais mettre de côté comme je le fais régulièrement. De ces faits, j'ai dû me remettre en question, en me donnant des challenges supplémentaires afin d'avoir la meilleure implémentation et de faciliter l'imbrication entre les différents modules que chaque personne développait de son côté. Ce n'a pas été chose aisée, il m'a fallu me documenter sur différentes technologies que je laissais jusqu'alors de côté.

D'ailleurs, c'est à ce moment précis que j'ai repensé à mon cours de gestion de projet. Une partie était destinée à voir plus clair sur ses capacités et s'aider à identifier les obstacles et opportunités d'évolution. Il parle notamment de cycle de croissance et de renouvellement ou nous apprenons à placer nos différentes capacités dans ces deux entités. Cela m'a permis de me conforter dans ma mauvaise approche de travail que j'avais jusqu'à présent et de pouvoir repartir avec de nouveaux choix d'implémentations à l'aide du cycle de croissance qui exige du temps et de l'efforts pour découvrir la valeur de la technologie.

Cette nouvelle expérience m'a mis en exergue que la facilité à choisir des implémentations selon mes forces ne fut pas une bonne pratique et qu'il ne fallait pas se conforter dans ses habitudes en utilisant des œillères mais, au contraire, ce challenger et travailler en groupe en fonction des besoins des autres.

Pour conclure, j'ai abouti à une première idée: il est important de se reposer sur ses capacités afin d'avancer rapidement mais il ne faut pas commencer à penser que l'on perd du temps à apprendre de nouvelles compétences. Ce projet a été l'exemple parfait pour moi de me remettre dans le bon axe et m'aidera à l'avenir à me projeter de façon plus impliquée et pragmatique dans la différence de choix d'implémentations en prenant le temps de peser le pour et le contre sur telle technologie à utiliser au lieu de foncer tête-bêche sur mes acquis de compétences.

Pierre Rambert

Ce projet représente une charge de travail importante et sur une longue durée. Il représente un travail de longue haleine, en groupe, avec de nombreuses réunions dans une logique Agile pour avancer sur le projet. Tous ces échanges sur le travail, le délivrable et les méthodes de travail ont été pour moi une vraie première, et cela m'a beaucoup fait réfléchir sur le plan personnel. Ainsi, je me suis demandé comment ce projet a-t-il modifié ma façon de travailler ?

Il m'est bien sûr arrivé de réaliser des travaux de groupe. Mais étant une personne plutôt introvertie et pragmatique, j'ai toujours préféré opérer selon une méthode qui me semblait simple et efficace. En premier temps, il faut définir en groupe le cadre de travail de chacun. Il s'agit ici de diviser la charge de travail en fonctions des préférences et de la capacité de travail de chacun, fixer la quantité de travail qui doit être rendue, une date pour se retrouver et mettre en commun le tout, ainsi que les canaux de communications qui vont être utilisés pour s'échanger, se donner des nouvelles avant cette date de mise en commun définie. La deuxième partie est donc le travail personnel, suivant la liste qui a été convenue en première partie. A ce moment-là, tout le monde peut travailler de manière complètement indépendante, ce qui me convenait très bien. Ce que je trouvais bien avec cette organisation, c'est l'indépendance qu'elle offrait à tous les membres du groupe. Bien qu'étant un travail de groupe, ce qui veut dire qu'il est impératif que tout le monde se rassemble et se concerte sur les attentes et objectifs du projet, cela permettait d'offrir à chacun un temps pour soi, qu'il pouvait utiliser à sa guise : une véritable flexibilité d'organisation et de travail, qui permet à chacun d'avoir un moment où il est à l'aise avec son travail. Certains se rassemblent en petit groupe, d'autres donnent des nouvelles régulièrement pour savoir si le projet avance à un rythme régulier à travers ces canaux, et d'autres s'isolent jusqu'à la date limite. Enfin, en dernière partie, il s'agit de faire la mise en commun de ce qui a été réalisé. Fort des technologies du XXIème siècle et de l'utilisation de services comme Google Drive, cette partie est en fait partiellement faite pendant la seconde phase (chacun à pu de son côté étoffer le diaporama, le texte à rendre, et cetera), et cette dernière partie revient à conclure sur ce que chacun à fait et à s'assurer de la qualité de ce qui à été rendu. C'est avec cette méthode que j'ai affronté les travaux de groupe au collège et au

lycée, me permettant tantôt de m'isoler des individus avec qui je ne m'entendais pas bien, et de travailler plus avec mes amis s'ils faisaient partie du groupe.

Cependant, le projet de ce semestre ne ressemble en rien à ce que j'ai pu faire dans ma vie. Que ce soit dans le travail à fournir, l'investissement personnel, le rendu et les responsabilités : délivrer pour que quelqu'un d'autre puisse faire son travail, j'ai découvert un nouveau niveau dans le rendu de projet. Si mon ancienne vision n'est pas à jeter, elle était clairement à revoir pour pouvoir réussir. Il a été très important de commencer par définir quelles étaient les limites, les enjeux et les dates limites du projet. Cependant, au vu de la taille et de la difficulté à priori du projet, il était impossible de prévoir des limites claires. Il a fallu donc s'adapter au fur et à mesure que nous avancions. Ainsi, il était parfaitement logique d'avoir des contres-rendu réguliers, ou nous présentions ce que nous avions fait la semaine précédente, et nous expliquions ce que nous prévoyons pour la prochaine réunion ; et malgré des réunions toutes les semaines, il était impératif que nous puissions communiquer encore plus régulièrement, de manière moins formel avec une ou deux personnes du groupe. La communication, très régulière, passant par des réunions et des messages privés a été aussi différente sur ce qu'elle m'a permis de dire ou non. En effet, mon avis c'est fait désirer et entendre par le reste du groupe, tout comme j'attendais l'avis du reste des personnes que j'écoutais avec attention. Ces échanges d'idées ont été un vrai terreau cognitif et ont largement participé à renforcer ma motivation et mon zèle. S'ajoute à ces nouveautés l'ordonnancement des tâches : bien que j'y ai déjà été confronté avant, c'est bien parce que l'enjeux du projet est bien différent que je l'ai vécu différemment aussi. En effet, lors de la mise en commun, si des personnes n'ont pas encore finalisé leur partie, il est possible de se retrouver bloqué sur une partie prioritaire du travail. Il faut donc savoir s'adapter et s'atteler à des parties moins importantes du projet mais faisable dès à présent. Cette gymnastique dans l'organisation du travail a été formatrice dans mon appréciation de ce que représente le travail de groupe ; mêlant un travail personnel intense avec un partage très important et régulier de chacune des avancées (esprit Agile).

Bien sûr, il y a plein de choses que je peux améliorer. Je pense tout d'abord que je n'ai pas suffisamment utilisé les outils collaboratifs tels que GitHub et Trello par exemple, ne donnant pas assez de visibilité sur mon travail. Globalement, je pense que mon travail d'un point de vue technique est convenable, mais j'ai un large champ d'amélioration dans le partage de mon travail, bien plus sur le partage du code que sur l'explication de ce que je fais et de ce que je compte faire. Aussi, étant une première fois, mon code n'est pas du tout adapté à une utilisation autre que sur mon poste, ce qui freine encore plus sa diffusion. Il serait intéressant à l'avenir de prendre cette dimension en considération

En conclusion, ma vision du travail d'équipe à beaucoup évolué en peu de temps. Je suis passé d'une conception collégiale de la gestion de projet à une conception entrepreneuriale, telle qu'elle est utilisée en entreprise, et telle qu'elle est enseignée à un niveau académique. C'est toute une dimension du travail collaboratif qui s'est présenté à moi, que je découvre et que j'apprends avec un plaisir certain. Avec ce nouvel outil, je me sens plus

confiant dans ma gestion de projet et je suis désormais mieux armé pour pouvoir réaliser de nouveaux et plus grands projets.

Isaïa Guille

Ayant déjà eu au préalable des expériences dans la conception et la réalisation de projets informatiques, j'avais déjà une idée de comment structurer ces derniers, que ce soit sur le plan organisationnel comme technique. Cependant, tous ces projets avaient pour point commun d'être des éléments logiciels stand-alone développés dans un langage unique. L'approche prise ici était bien différente. Comment ce projet a-t-il changé ma vision des architectures informatiques ?

Avec mon cursus scolaire professionnalisant en Brevet de Technicien supérieur, il m'a notamment été demandé de réaliser de nombreux sites internet. La plupart de ces derniers étaient des projets de groupes qui étaient inclus à l'apprentissage d'un langage de programmation en particulier. La contrainte principale était donc de réaliser cette application dans ce langage donné. Compte tenu de cela, la question de comment faire communiquer les différents composants ne se posait pas : chaque développeur ajoutait ses fonctions, son code, ses composants logiciels directement au programme et le raccord se faisait au travers d'une simple instruction intégrée au contrôleur de l'application MVC.

Il en a été de même lors de mes deux stages en entreprise pendant cette période. Je devais réaliser intégralement l'application avec des composants ayant déjà une communication facile et sans trop de contraintes (e.g. : PHP et MySQL).

Avec ces projets aux contraintes similaires, je me suis encroûté dans un environnement familier sans jamais me poser la question "Que se passerait-il si mon collègue développeur et moi-même ne parlions pas les mêmes langages ?". J'ai supposé que je ne serais jamais confronté à une telle situation sans vraiment questionner mes acquis ; et quelle erreur cela fût.

En effet, cette problématique s'est posée au cours de ce projet commun. Ayant chacun eu un parcours différent dans le domaine de la programmation informatique, - contrairement à lors de mon BTS - chaque membre du groupe avait ses habitudes et aptitudes pour ses langages de prédilection, ses composants logiciels avec lesquels il était confortable. Cette situation inédite dans mon parcours a rapidement été bloquante : comment un logiciel Python peut communiquer avec une application PHP qui veut interroger un programme JavaScript ?

N'ayant pas un bagage technique très important mais plutôt une appétence à la gestion et au management, j'ai vite réalisé que ça allait devenir un réel problème dans le futur et que je me devais d'être informé sur les éventuelles solutions pour pouvoir au mieux résoudre ce genre de soucis qui, une fois sorti de cet environnement familier et protecteur, risque d'être un soucis régulier.

En prenant le temps d'y réfléchir, et grâce à l'expérience des autres membres du projet, j'ai remarqué que c'est en réalité une situation que j'ai déjà expérimentée au cours de cette année universitaire, grâce à l'alternance, et durant certains de mes projets personnels. Effectivement, je développe des scripts JavaScript pour un progiciel de gestion intégré durant

mon alternance, et ce dernier doit souvent faire appel à des applications développées dans d'autres langages et sur d'autres systèmes. La solution de prédilection la plus évidente était en réalité les API. Une fois ce constat fait, j'ai compris que grâce à ce système issu du Web 2.0 promouvant les échanges pour un Internet plus collaboratif, ce problème n'en était plus un.

De plus, un autre système similaire mais aux propriétés différentes m'a été introduit au cours de ce projet : les files d'attente de messages asynchrones. Elles répondent à la même problématique mais par une approche différente, l'approche non-synchrone apporte la possibilité, non seulement de faire communiquer différents composants logiciels entre eux, mais aussi de le faire même lorsque ces derniers sont éteints ou indisponibles.

Pour conclure, la vision API / files d'attente de messages asynchrones m'apporte une toute nouvelle dimension à la conception d'architecture de solutions informatiques. Grâce à ces systèmes, j'ai compris qu'un logiciel informatique n'avait pas une forme unique mais pouvait aussi être l'assemblage d'une succession de petits composants interagissant entre eux, ce qui renouvelle complètement l'approche que je pourrais avoir sur mes projets futurs. La préparation d'un service n'a plus uniquement une seule possibilité architecturale et en devient beaucoup plus permissive, ce qui rapproche aussi ces principes aux méthodes agiles et aux bonnes pratiques du développement, en proposant une succession de petits modules pouvant être développés séparément.

Annexes

Exemple de code généré par le script récupéré de l'API Golang


```
{
    "actions":[
        {
            "action": "settings",
            "options": {
                "autoPlay": false
            }
         },
        {
            "action": "search",
            "toSearch": "Qanon"
        },
        {
            "action": "watch",
            "index": 3,
            "watchContext": {
                "stopsAt": "5"
            }
        },
        {
            "action": "watch",
            "index": 1,
            "watchContext": {
                "stopsAt": "5",
                "social": "like"
            }
        }, {
            "action": "watch",
            "index": 2,
            "watchContext": {
                "stopsAt": "5",
                "social": "dislike"
            }
        },
        {
            "action": "goToChannel"
        },
        {
            "action": "watch",
            "index": 5,
            "watchContext": {
                "stopsAt": "5"
            }
        },
        {
            "action": "watch",
            "url": "https://www.youtube.com/watch?v=N3zujKdhH0U"
        }
    ]
}
```

```
[
    {
        "links": [
            {
                "rel": "all-sessions",
                "href": "https://test.netops.fr/api/sessions"
            },
            {
                "rel": "all-logs",
                "href": "https://test.netops.fr/api/logs"
            },
            {
                "rel": "all-captures",
                "href": "https://test.netops.fr/api/captures"
            },
            {
                "rel": "all-videos",
                "href": "https://test.netops.fr/api/videos"
            },
            {
                "rel": "all-channels",
                "href": "https://test.netops.fr/api/channels"
            },
            {
                "rel": "all-comments",
                "href": "https://test.netops.fr/api/comments"
            },
            {
                "rel": "all-captions",
                "href": "https://test.netops.fr/api/captions"
            },
            {
                "rel": "all-videos-label",
                "href": "https://test.netops.fr/api/videos/label"
            },
            {
                "rel": "all-channels/label",
                "href": "https://test.netops.fr/api/channels/label"
            },
            {
                "rel": "all-actions",
                "href": "https://test.netops.fr/api/actions"
            }
        ]
    }
]
```