

# **METHODE D'ACCES AU DONNEES**

## **Rapport**

### **Projet**

Christen, Dupraz & Mottier

# Table des matières

1. Description .....	3
2. Fonctionnalités .....	3
3. Modélisation des données .....	6
Provenance des données .....	7
Intégration des données.....	7
4. Explication des requêtes .....	7
5. Mise en place du bot .....	7
6. Problèmes rencontrés .....	8
7. Améliorations possibles .....	8

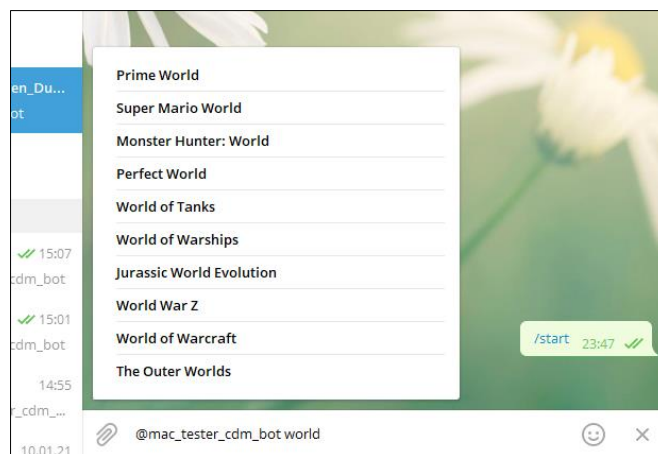
## 1. Description

Le but du projet est de créer un bot sur le client de messagerie Telegram. Ce bot aura pour but de pouvoir interagir avec la plateforme de streaming Twitch (<https://dev.twitch.tv/docs/api/>) qui dispose d'une API.

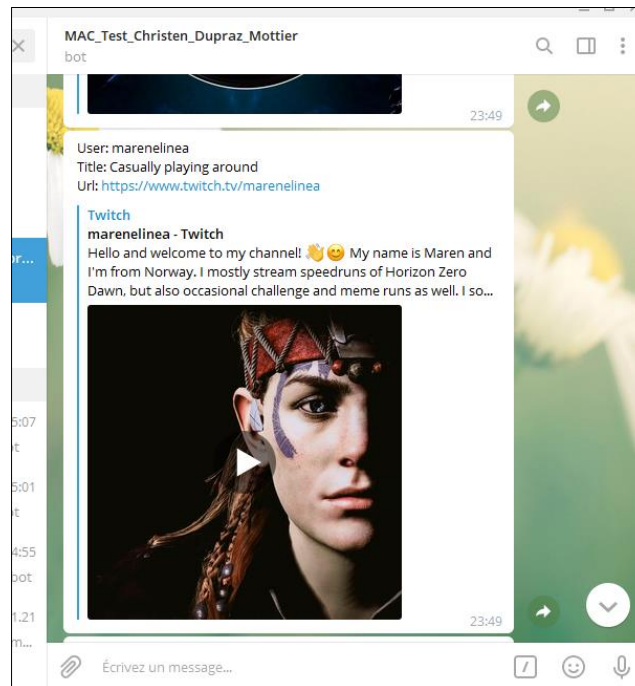
Le bot aura plusieurs fonctionnalités qui seront énoncées au point suivant.

## 2. Fonctionnalités

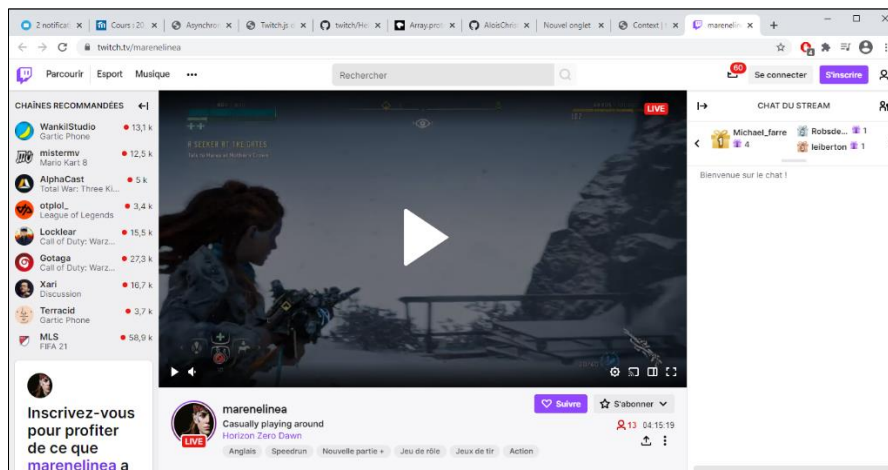
Le bot vous propose de liker les jeux disponibles dans le top 500 des jeux streamés sur Twitch. L'image ci-dessous montre un exemple de recherche de jeu et l'affichage instantané des résultats.



En entrant la commande « /recommendstreamer », le bot affiche une liste de streamers qui pourraient vous plaire, à la condition que vous ayez liké assez de jeux. La capture ci-dessous montre une proposition de stream d'Horizon Zero Dawn




La sélection d'un streamer vous proposera d'aller voir son stream, comme ci-dessous avec la proposition de stream précédente.



En entrant la commande « /recommendgame », le bot affiche une liste de jeux qui pourraient vous plaire en fonction de ce que vous avez liké, à la condition que vous ayez liké assez de jeux.

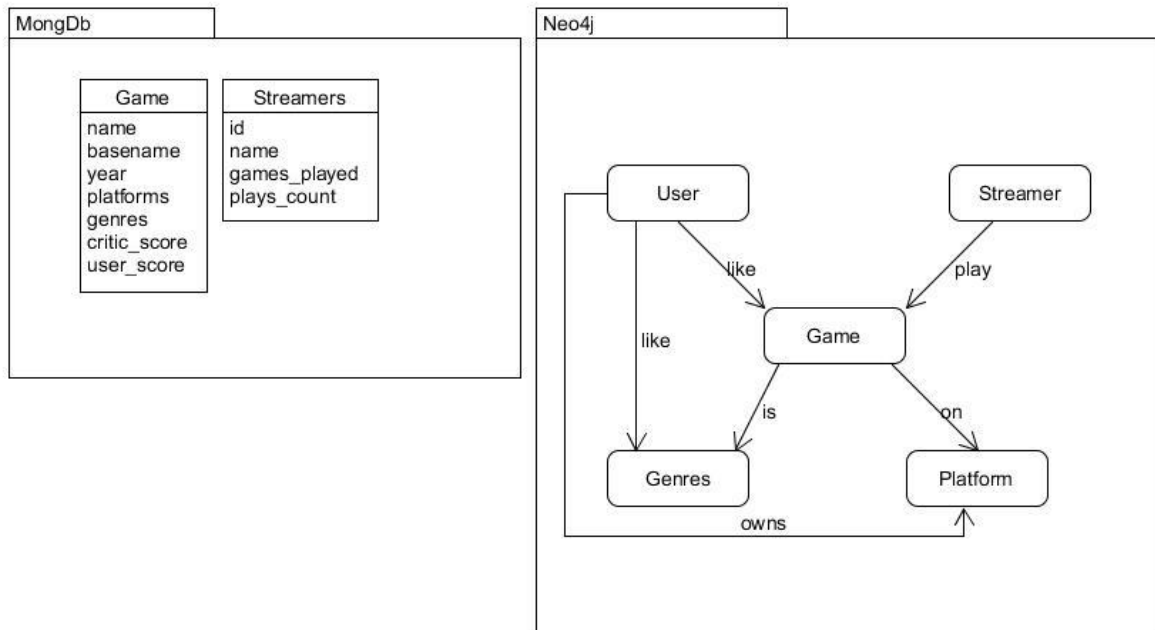
Title: Call of Duty: WWII  
Year: 2017  
Platforms : PS4,PC,XOne  
Genres: Shooter  
Twitch: [https://www.twitch.tv/directory/game/Call%20of%20Duty:  
%20WWII](https://www.twitch.tv/directory/game/Call%20of%20Duty:%20WWII)

**Twitch**  
**Call of Duty: WWII - Twitch**  
Watch Call of Duty: WWII channels streaming live on Twitch. Sign up or login to join the community and follow your favorite Call of Duty: WWII streamers!



08:53

### 3. Modélisation des données



Dans notre base MongoDB, nous avons deux collections :

- Games : cette collection contient les top jeux streamés sur Twitch. On peut y retrouver le nom du jeu, sa date de sortie, les plateformes sur lesquelles le jeu peut être joué, les genres qui lui sont attribués et son id Twitch.
- Streamers, le top 3 des streamers pour la première partie des jeux, et le top 1 des streamers sur la seconde partie des jeux. La collection est composée de l'id Twitch et du nom streamer et son basename (nom dans l'url).

Dans notre base Neo4j, nous avons 5 types de nodes :

- User
- Streamer
- Game
- Genre
- Platform

Et contient aussi 4 types de relations :

- LIKE : User → Game | User → Genre
- PLAYS\_TO: Streamer → Game
- BELONGS\_TO: Game → Genre
- PLAYED\_ON: Game → Platform
- OWNS : User → Platform

## Provenance des données

Les données insérées dans MangoDB pour les jeux proviennent de [Kaggle](#) et les données sur les streamers viennent directement de Twitch en passant par leur [API](#).

## Intégration des données

Nous intégrons les données des jeux sur MangoDB grâce au package « mangodb » pour NodeJS. Nous avons modifié nos données de base pour n'avoir que le top 500 des jeux streamés sur Twitch.

Pour intégrer les données des streamers, nous créons un CSV grâce à l'API de Twitch. Nous insérons seulement des streamers qui jouent à un jeu qui est déjà dans notre base.

## 4. Explication des requêtes

« /recommendstreamer » : Nous allons chercher 5 streamers, qui ont joué à au moins un jeu que l'utilisateur aura « like » auparavant. Les jeux doivent bien évidemment être différents et le score doit être plus haut que 3 étoiles. Si un streamer joue plus de fois au jeu « liké », mieux il sera positionné pour ressortir dans la recherche.

« /recommendgame » Nous allons chercher 5 jeux, qui ont un genre que l'utilisateur aura aimé par le biais d'un jeu. Les jeux ne sont pas des jeux déjà « like ». Le score du jeu doit être plus haut que 3 étoiles.

## 5. Mise en place du bot

1. Lancer le docker-compose qui se trouve à la racine du projet
2. Modifier votre environnement en conséquence
3. Lancer la commande « npm run install » pour installer toutes les dépendances du projet
4. Lancer la commande « npm run import » pour loader les datas en bases
5. Lancer la commande « npm start »
6. Le bot est prêt à l'emploi

## **6. Problèmes rencontrés**

Lors de l'insertion des données dans Neo4j, il arrive que nous soyons bloqués, car la base se lock. Il y a un deadlock entre deux insertions différentes. Malheureusement, nous n'avons pas réussi à résoudre ce bug.

Pour résoudre ce problème, il faut relancer le script d'insertion.

Les problèmes de synchronisation nous ont posé beaucoup de problème, ce qui nous a coûté énormément de temps.

L'API Twitch est assez pauvre en fonctions pour l'objectif qu'on s'était fixé. Nous devons faire beaucoup de requêtes pour pouvoir récupérer les données et comme nous sommes limités par le nombre de requête sur une certaine période, cela peut prendre beaucoup de temps.

## **7. Améliorations possibles**

En l'état, notre bot pourrait être amélioré de nombreuses façons. Tout d'abord, plus de jeux pourraient être stockés dans la base de données, sans se limitant aux 500 top jeux de Twitch. Il serait aussi possible de laisser l'utilisateur rentrer lui-même des jeux, ou des streamers (par exemple s'il se lance dans cette activité).

Une autre amélioration évidente serait de pouvoir liker également des streamers, et que la recommandation de streamers prenne en compte ces préférences. Ces mêmes recommandations pourraient de plus intégrer d'autres paramètres : année de sortie, développeur, langue du stream, etc.

Utilisation du « inline » pour rechercher autre chose que des jeux, et pouvoir choisir de rechercher un streamer, un genre ou une plateforme.