Documentation Technique

youtube.php (Frontend)

1. **Introduction :**
   * La page youtube.php est une interface utilisateur affichant des informations sur le nombre d'abonnés YouTube d'une chaîne spécifique.
   * Les données sont obtenues via l'API YouTube, avec des fonctionnalités d'historique et de gestion des requêtes.
2. **Contenu HTML :**
   * Titre, image de profil, nom de la chaîne et nombre d'abonnés sont affichés.
   * Des commentaires HTML précisent les éléments temporaires ou de débogage à retirer dans la version finale.
3. **Script JavaScript (youtube.php) :**
   * Importe la classe ***‘Compteur\_Histo’*** depuis le fichier externe **./Classes/Compteur\_Histo.js**
   * Définit les clés d'API, l'ID de la chaîne, et instancie la classe ***‘Compteur\_Histo’***.
4. **Classe JavaScript - Compteur\_Histo :**
   * Gère la récupération des données YouTube, les requêtes HTTP, la connexion WebSocket, et les interactions avec le backend.
   * Utilise deux clés d'API, bascule entre elles pour éviter les limitations.
   * Utilise une WebSocket pour envoyer des informations au serveur C++.
   * Effectue des mises à jour toutes les 5 secondes.
5. **Méthodes principales :**
   * ***‘updateData’*** : Orchestre la mise à jour des données en appelant d'autres méthodes séquentiellement.
   * ***‘getChannelInfo’*** : Utilise l'API YouTube pour récupérer des informations sur la chaîne.
   * ***‘getNbRequetesFromBack’***, ***‘getDateFromBack’*** : Récupèrent les informations depuis le backend via des requêtes asynchrones.
   * ***‘sendInfoToCpp’*** : Envoie des informations au serveur C++ via WebSocket.
   * ***‘checkAndUpdateApiKey’*** : Vérifie et bascule entre deux clés d'API.
   * ***‘resetAutoIncrementValue’*** : Réinitialise ou incrémente le nombre de requêtes en fonction du jour actuel.
6. **Documentation des méthodes :**
   * Chaque méthode est documentée dans le code pour expliquer sa fonctionnalité.

**Compteur\_Histo.js (Frontend)**

1. **Introduction :**
   * La classe ***‘Compteur\_Histo’*** est conçue pour gérer les statistiques d'une chaîne YouTube via l'API YouTube et le backend.
2. **Propriétés de classe :**
   * ***‘apiKey1’***, ***‘apiKey2’*** : Deux clés d'API YouTube.
   * ***‘apiKey’*** : Clé d'API actuelle stockée localement.
   * ***‘channelId’*** : ID de la chaîne YouTube.
   * ***‘socket’*** : Objet WebSocket pour la communication avec le serveur C++.
   * ***‘DateBDD’***, ***‘NBRequete’*** : Informations de la base de données initialisées dans le constructeur.
3. **Méthodes de classe :**
   * ***‘getNbRequetesFromBack’***, ***‘getDateFromBack’*** : Requêtes asynchrones pour récupérer des informations depuis le backend.
   * ***‘sendInfoToCpp’*** : Envoie des informations au serveur C++ via WebSocket.
   * ***‘checkAndUpdateApiKey’*** : Bascule entre deux clés d'API en fonction du nombre de requêtes.
   * ***‘resetAutoIncrementValue’*** : Réinitialise ou incrémente le nombre de requêtes en fonction du jour actuel.
   * ***‘getChannelInfo’*** : Utilise l'API YouTube pour récupérer des informations sur la chaîne.
   * ***‘PostToBack’*** : Envoie les données au backend.
   * ***‘updateData’*** : Méthode principale orchestrant la mise à jour des données.

**PostQuery.js (Backend) :**

1. **Introduction :**
   * Backend utilisant Express pour traiter les requêtes POST depuis le frontend.
   * Communique avec une base de données MySQL pour stocker les informations.
2. **Configuration Express :**
   * Utilisation du middleware ***‘body-parser’*** pour analyser les données JSON dans les requêtes POST.
   * Gestion des CORS pour autoriser les requêtes depuis n'importe quelle origine.
3. **Connexion à la Base de Données :**
   * Utilisation d'une fonction ***‘ConnexionBDD’*** pour établir la connexion à la base de données MySQL.
4. **Routes Express :**
   * **/new/Classes/Compteur\_Histo/PostDate** : Renvoie la dernière date enregistrée dans la base de données.
   * **/new/Classes/Compteur\_Histo/PostNBRequetes** : Renvoie le nombre de requêtes le plus élevé enregistré dans la base de données.
   * **/new/Classes/Compteur\_Histo** : Reçoit les données du frontend et les enregistre dans la base de données.
5. **Gestion des Requêtes MySQL :**
   * Utilisation de requêtes SQL pour récupérer les informations demandées.
   * Insertion des données reçues du frontend dans la table *"Compteur Histo"*.

1. **Gestion des Erreurs :**
   * Gestion des erreurs potentielles lors de la connexion à la base de données ou lors de l'exécution des requêtes.
2. **Démarrage du Serveur :**
   * Écoute sur le port 8080 pour les requêtes entrantes.