**Description de ton projet**

**MasterTagSystem** est un système conçu pour gérer des données JSON de manière efficace et permettre la création et la modification d'arborescences JSON complexes. Ce projet inclut le développement d'une architecture complète, impliquant un backend en C# pour le stockage et le traitement des données JSON, un frontend en Angular pour l'affichage et la modification d'une arborescence JSON, et Kafka pour la gestion de flux de données en temps réel. À terme, Mesos sera utilisé pour l'orchestration des ressources afin de garantir l'évolutivité de l’application, notamment dans le cadre d’un traitement à grande échelle.

**Objectifs du projet**

1. **Backend** : Créer un système robuste pour consommer, stocker et traiter des messages JSON dans MongoDB, avec une capacité d'évoluer vers une haute fréquence de traitement (jusqu’à des milliers de messages par seconde).
2. **Frontend** : Développer une interface Angular permettant de manipuler les messages JSON sous forme d’arborescence, avec la possibilité d’éditer et de structurer facilement les données.
3. **Kafka** : Assurer la transmission en temps réel des messages JSON vers le backend pour une insertion rapide dans MongoDB.
4. **Mesos** : Utiliser Mesos pour orchestrer les services backend, Kafka, et MongoDB, facilitant la gestion des ressources et l’extensibilité de l’application.

**Ce que tu as réalisé**

1. **Mise en place de MongoDB** :
   * Installation et configuration de MongoDB (localement) pour le stockage des données JSON.
   * Création d'une base de données CriteoProject et connexion réussie au backend C# pour le stockage des messages JSON.
2. **Développement du backend en C# (ASP.NET Core)** :
   * Création d’un service TagService pour l’interaction avec MongoDB, incluant des fonctions de validation et d’insertion des JSON.
   * Développement d’un contrôleur TagController pour gérer les requêtes d’API, recevoir les messages JSON et déclencher l’insertion dans MongoDB.
   * Tests initiaux pour s’assurer que le backend reçoit et stocke correctement les données JSON.
3. **Simulation de données et tests avec Kafka (en Python)** :
   * Création d’un script producerKafka.py pour simuler l’envoi de messages JSON en continu au backend, vérifiant la résilience de l’application.
   * Mise en place d'un envoi de messages toutes les 2 secondes pour tester le système de manière basique.

**Ce qu'il te reste à faire**

En intégrant Mesos et l’arborescence JSON, voici les étapes restantes dans l'ordre.

1. **Finaliser l'interface arborescente en Angular** :
   * **Objectif** : Développer une interface utilisateur sous forme d’arborescence pour afficher et manipuler les messages JSON.
   * **Actions** :
     + Afficher les données JSON sous forme d'arborescence pour une meilleure lisibilité.
     + Ajouter la possibilité d’éditer les nœuds JSON (ajout, modification, suppression).
     + Mettre en place une interface utilisateur simple et intuitive permettant de valider l’intégrité des modifications avant envoi au backend.
2. **Intégrer Kafka pour la transmission des données en temps réel** :
   * **Objectif** : Remplacer le script de simulation par une intégration complète de Kafka dans l’architecture.
   * **Actions** :
     + Configurer Kafka pour l’envoi de messages JSON en temps réel du producteur vers le backend.
     + Modifier le backend C# pour écouter Kafka et consommer les messages en temps réel.
     + Effectuer des tests pour s’assurer que les messages sont traités avec rapidité et exactitude.
3. **Orchestration avec Mesos** :
   * **Objectif** : Utiliser Mesos pour orchestrer les services de manière efficace, surtout dans un contexte de traitement de données à grande échelle.
   * **Actions** :
     + Configurer Mesos pour gérer les ressources backend et Kafka.
     + Orchestrer le backend, MongoDB et Kafka pour garantir leur fiabilité et leur scalabilité sous une charge importante.
     + Effectuer des tests de charge pour valider la capacité du système à traiter des milliers de messages JSON par seconde.
4. **Améliorer la qualité des données dans MongoDB** :
   * **Objectif** : Assurer la qualité et l'intégrité des données stockées.
   * **Actions** :
     + Mettre en place des validations plus complexes pour le format et la structure des JSON avant leur insertion.
     + Ajouter un processus de vérification pour détecter les utilisateurs ou données indésirables (frauduleuses ou de faible valeur).
5. **Tests finaux et documentation** :
   * **Objectif** : Vérifier l'ensemble des fonctionnalités et s'assurer que l'application est prête pour un environnement de production.
   * **Actions** :
     + Exécuter des tests de bout en bout sur chaque composant.
     + Documenter le code et les configurations pour faciliter la maintenance et la mise à l’échelle.
     + Préparer la solution pour un déploiement en production, incluant la gestion des configurations et les scripts d’installation.