HomePedia Rapport

DATE:	**15/07/2024**
Type de document	**Rapport sur le projet de spécialité Big Data : HomePedia**

INTRODUCTION

L'aventure de HomePedia dans le domaine de traitement de données en batch s'est avérée être un périple riche en apprentissages et en défis.

1.1 Objectif du Projet

L'objectif de ce projet est de fournir une solution complète pour la collecte, la transformation et la visualisation des données immobilières provenant de diverses sources. En intégrant des données extraites de sites web d'immobilier et de sources gouvernementales, le projet vise à offrir une plateforme centralisée pour analyser et explorer ces données.

1.2 Fonctionnalités principales

- Collecte automatisée de données immobilières à partir de sources variées
- Transformation et nettoyage des données brutes
- Stockage efficace dans des bases de données SQL et NoSQL
- Interface utilisateur intuitive pour la visualisation et l'exploration des données
- Recherche avancée et analyse des tendances immobilières

1. Organisation

1.1 Équipe de Développement:

Le projet a été mené par une équipe composée de trois développeurs expérimentés. Chaque développeur avait des responsabilités spécifiques dans la conception, le développement et le déploiement du système. Les rôles clés au sein de l'équipe de développement étaient les suivants :

- **Amine** : Responsable du de regroupement de données, ainsi que du développement des pipelines ETL pour stocker les données dans les bases de données SQL et NOSQL.
- **Idriss**: En charge du développement de l'interface utilisateur à l'aide d'Angular. S'est occupé de créer des visualisations de données conviviales à l'aide de Chart.js pour afficher les données récupérées.
- **Samy** : Responsable du Scrapping des données et de la coordination entre le backend et le frontend, s'assurant que les données étaient correctement transmises et affichées dans l'interface utilisateur.
- **Zakia**: Responsable de la gestion du projet.
- **Louis**: En charge du développement de l'interface utilisateur à l'aide d'Angular.

1.2 Équipe de Gestion de Projet:

L'équipe de gestion de projet comprenait un chef de projet. Les rôles clés de l'équipe de gestion de projet étaient les suivants :

- S'assurer la supervision générale du projet, y compris la planification, la coordination des équipes de développement et de gestion, la communication avec les parties prenantes et la résolution des problèmes éventuels.
- S'assurer que le produit final répondait aux besoins fonctionnels des utilisateurs. Il a travaillé en étroite collaboration avec l'équipe de développement pour garantir que les fonctionnalités nécessaires étaient correctement implémentées.

1.3 Communication et Collaboration:

Une communication claire et régulière entre les membres de l'équipe de développement et de gestion de projet a été essentielle pour garantir le succès du projet. Des réunions régulières ont été organisées pour discuter de l'avancement, résoudre les problèmes et prendre des décisions importantes. Un système de gestion de projet a été mis en place pour suivre les tâches, les échéances et les problèmes. En outre, des canaux de communication ont été établis pour permettre une collaboration efficace entre les membres de l'équipe.

Cette version utilise le passé pour décrire les responsabilités et les actions des équipes de développement et de gestion de projet une fois le projet réalisé.

1.4 Méthodologie de travail

Pour garantir une avancée efficace du projet, nous avons utilisé une méthode de travail **Agile Scrum.**

Nous avons donc dans un premier temps identifié les compétences et rôles nécessaires pour le projet et nous nous sommes répartis les tâches et rôle en fonction de nos compétences et pouvant collaborer efficacement.

Nous avons avancé avec un point toutes les deux semaines en physique pour partager l'avancée du projet et parler des points de blocage potentiel. Pour tout autre moyen de communication, teams a été le moyen le plus privilégié.

En prenant compte nos points forts de l'agilité que nous procure la méthode, nous avons pu nous aider sur les autres points dont on n'était pas forcément assigné au départ.

Les documents et tâches sont tous répertoriés sur notre espace Notion pour être le plus transparent possible pour montrer l'avancée du projet en fonction de notre planning.

2. Architecture du Système

2.1 Vue d'ensemble

Le système est composé de plusieurs composants interconnectés :

- Backend : FastAPI
- Frontend : Angular
- Bases de données : SQLite (SQL) et MongoDB (NoSQL)
- Pipelines ETL : Apache Spark

2.2 Flux de données

1. Extraction des données brutes (sites web immobiliers, sources gouvernementales)
2. Transformation et traitement des données avec Apache Spark
3. Chargement des données dans les bases SQLite et MongoDB
4. Exposition des données via l'API FastAPI
5. Visualisation et interaction utilisateur via l'interface Angular

3. Backend

3.1 Structure des routes API

3.1.1 Routes NoSQL (MongoDB)

- GET / : Récupère toutes les données immobilières
- GET /{commune} : Récupère les données par commune
- POST / : Ajoute de nouvelles données
- GET /test-db-connection : Teste la connexion à la base de données

3.1.2 Routes SQL (SQLite)

- GET /parville/{ville} : Récupère les données par ville
- GET /pardepartement/{departement} : Récupère les données par département
- GET /parregion/{region} : Récupère les données par région

3.2 Intégration avec les bases de données

Notre système utilise une approche hybride pour le stockage des données, tirant parti des forces de deux types de bases de données. Pour les données détaillées sur les propriétés immobilières, nous avons opté pour MongoDB, une base de données NoSQL. Ce choix permet une grande flexibilité dans la structure des données et une excellente performance pour les opérations de lecture et d'écriture à grande échelle. En parallèle, nous utilisons SQLite pour stocker les données agrégées par ville, département et région. SQLite, étant une base de données relationnelle légère, est parfaitement

adaptée pour gérer ces données structurées et permet des requêtes complexes efficaces sur ces informations agrégées.

3.3 Sécurité et performance

La sécurité et la performance sont au cœur de notre architecture backend. Pour garantir la fiabilité du système, nous avons mis en place un endpoint spécifique (/test-db-connection) permettant de tester la connexion à la base de données à tout moment. Cela facilite le diagnostic rapide des problèmes potentiels de connectivité. En ce qui concerne la sécurité des données, nous utilisons des modèles Pydantic pour la validation rigoureuse des données entrantes et sortantes. Cette approche nous permet de nous assurer que toutes les données traitées par notre API sont conformes aux structures attendues, réduisant ainsi les risques d'erreurs et de failles de sécurité. De plus, pour maintenir la qualité et la fiabilité de notre code, nous avons mis en place une suite de tests automatisés utilisant pytest. Ces tests couvrent les différentes fonctionnalités de notre API, assurant ainsi que toute modification du code n'introduit pas de régressions et que le système continue de fonctionner comme prévu.

3.4 Données utilisées dans le frontend

Les données suivantes seront principalement utilisées dans l'interface utilisateur :

1. Données immobilières détaillées (de MongoDB) :
   - Type de local
   - Commune
   - Prix moyen au m²
   - Valeur foncière moyenne
   - Surface bâtie moyenne
   - Prix médian au m²
   - Nombre de transactions
2. Données agrégées par zone géographique (de SQLite) :
   - Indicateurs de qualité de vie (sécurité, culture, animation, environnement, vie pratique)
   - Données démographiques (nombre d'habitants, densité, âge de la population)
   - Données économiques (population active, taux de chômage, revenu moyen)
   - Prix au m² moyen

Ces données permettront de créer des visualisations riches et interactives, telles que :

- Des cartes de chaleur des prix immobiliers
- Des graphiques comparatifs entre différentes zones géographiques
- Des tableaux de bord montrant les indicateurs clés pour une zone sélectionnée
- Des filtres interactifs basés sur divers critères (type de propriété, fourchette de prix, etc.)

4. Frontend

En ce qui concerne le frontend, nous avons utilisé AngularTs pour développer une interface utilisateur intuitive et réactive. Nous avons utilisé des composants Angular tels que les tableaux pour afficher les données de manière organisée et les charts avec Chart.js pour visualiser graphiquement les données sous forme de graphiques et de diagrammes. Nous avons opté pour Chart.js dans notre

projet de visualisation des données en raison de sa légèreté, de sa facilité d'utilisation et de sa compatibilité avec les navigateurs modernes et les appareils mobiles.

Nous avons aussi utilisé MapBox afin de construire des cartes interactives et fluides pour visualiser les régions, villes et départements et avoir une vue d'ensemble et de ces dernières.

Cette approche de visualisation des données en temps réel offre aux utilisateurs une vue claire et dynamique des tendances et des informations clés, facilitant ainsi la prise de décision et l'analyse des données dans un environnement convivial et interactif.

4.1 Composants principaux

- Tableau de bord principal
- Carte interactive des propriétés
- Visualisations graphiques (graphiques, diagrammes)

4.2 Services

- Service API pour communiquer avec le backend
- Service de gestion d'état (ex: NgRx) pour gérer l'état global de l'application

4.3 Modules

- Module de routage pour la navigation entre les différentes vues
- Utilisation d'Angular Material pour des UI plus conviviales
- Utilisation de chartjs pour la visualisation de données

5. Pipelines ETL

Notre pipeline ETL est conçu pour traiter et intégrer des données immobilières provenant de diverses sources. Ce processus est crucial pour transformer des données brutes en informations exploitables pour notre application d'analyse immobilière.

5.1 Sources de données

Nous utilisons deux principales sources de données, chacune apportant des informations uniques à notre projet :

- Données immobilières détaillées : Ces données proviennent de fichiers CSV fournis par le gouvernement. d'environ 8 Go de données brutes.

  Source:

- Données agrégées sur les villes, départements et régions issues de scrapping de sites tels que "https://www.bien-dans-ma-ville.fr". Ces informations, contenues dans des fichiers JSON, d'environ 66 Go de données brutes.

5.2 Processus d'extraction

L'extraction des données est adaptée à chaque source :

Pour les données immobilières, nous utilisons PySpark pour sa capacité à gérer efficacement de grands volumes de données. Le processus implique :

- La lecture et la combinaison de tous les fichiers CSV du dossier spécifié.
- L'utilisation d'options appropriées comme le délimiteur "|" et la prise en compte des en-têtes.

Pour les données agrégées, nous employons la bibliothèque json de Python pour lire le fichier JSON. Les données sont ensuite parsées en trois catégories distinctes : “par villes”, “par départements”, et “par régions”.

5.3 Processus de transformation

La transformation des données est une étape cruciale qui diffère selon le type de données :

- Pour les données immobilières, les principales étapes comprennent :

1. Renommage des colonnes (normalisation des noms des colonnes).
2. Conversion des dates et traitement des données numériques.
3. Filtrage des données incomplètes ou invalides.
4. Calculs dérivés (comme le prix au mètre carré).
5. Agrégation des données par type de local et commune, avec calcul de diverses statistiques.

- Pour les données agrégées, le processus est plus simple :

- Conversion des champs numériques en type “float”
- Gestion des valeurs manquantes

5.4 Processus de chargement

Le chargement des données transformées se fait dans deux systèmes différents :

1. Données immobilières :
   - Conversion du DataFrame Spark en DataFrame Pandas
   - Ajustement des formats de données pour la compatibilité
   - Insertion dans une collection MongoDB nommée 'transformed_data'
2. Données agrégées :
   - Création de DataFrames Pandas pour chaque catégorie
   - Insertion dans une base de données SQLite ('my_database.db') avec une table par catégorie

5.5 Défis et particularités

Notre pipeline ETL fait face à plusieurs défis intéressants :

- Gestion d'un volume important de données (66 Go) nécessitant l'utilisation de PySpark sur un cluster Spark.
- Nettoyage et uniformisation de formats de données variés
- Réalisation de calculs statistiques avancés sur les données immobilières
- Intégration de deux types de bases de données (MongoDB et SQLite) pour différents besoins

En conclusion, ce pipeline ETL transforme des données brutes volumineuses et hétérogènes en un ensemble structuré et enrichi, prêt à alimenter notre application d'analyse immobilière. Il illustre la complexité et la puissance des processus de traitement de données modernes, combinant différentes technologies pour répondre aux besoins spécifiques du projet.

6. Visualisation des données

Notre application offre une variété de visualisations interactives pour permettre aux utilisateurs d'explorer et d'analyser efficacement les données immobilières. Ces visualisations sont réparties sur deux pages principales, chacune se concentrant sur différents aspects des données.

7.1 Page 1 : Cartographie

La première page est centrée sur une représentation cartographique des données immobilières :

- Carte interactive de la France : Cette carte permet aux utilisateurs de visualiser les données immobilières par région, département ou ville.
- Fonctionnalités de la carte :
  - Zoom et déplacement pour explorer différentes zones géographiques
  - Code couleur pour représenter différentes métriques (ex: prix moyen au m², nombre de transactions)
  - Possibilité de sélectionner une région ou un département pour un zoom automatique

Cette visualisation cartographique offre une vue d'ensemble rapide et intuitive de la répartition géographique des données immobilières, permettant d'identifier facilement les tendances régionales et les zones d'intérêt.

7.2 Page 2 : Graphiques détaillés

La deuxième page propose une analyse plus approfondie à travers trois types de graphiques, Ces graphiques offrent une vue claire des proportions et sont particulièrement utiles pour comparer différentes catégories au sein d'un ensemble de données.

  - Répartition des types de biens dans une zone sélectionnée
  - Distribution des tranches de prix
  - Proportion des transactions par surface habitable
  - Comparaison du prix moyen au m² entre différentes villes ou régions
  - Nombre de transactions par mois ou par année
  - Répartition des biens par nombre de pièces
  - Évolution du prix moyen au m² sur une période donnée
  - Tendances du nombre de transactions au fil du temps

- Variation saisonnière des prix ou des volumes de vente

Fonctionnalités communes à tous les graphiques :

- Interactivité : Les utilisateurs peuvent cliquer sur des éléments spécifiques pour obtenir plus de détails.
- Personnalisation : Options pour ajuster la période, les catégories affichées, ou le type de données visualisées.
- Exportation : Possibilité de télécharger les graphiques ou les données sous-jacentes.

L'ensemble de ces visualisations permet aux utilisateurs d'explorer les données sous différents angles, de l'aperçu général offert par la carte aux analyses détaillées fournies par les graphiques variés. Cette approche multi-facettes facilite la compréhension des tendances du marché immobilier et aide à la prise de décision informée.

7. Axes d'amélioration

7.1. Récolter et traiter plus de données :

Augmenter la quantité de données collectées à partir de diverses sources pour enrichir notre base de données et améliorer la précision des analyses.

7.2 Mettre en place un cluster Spark avec plus d'instances workers :

Déployer un cluster Spark avec un nombre accru de workers afin de traiter plus rapidement les données récoltées et d'optimiser les performances du pipeline ETL.

7.3 Adopter une base de données comme PostgreSQL au lieu de SQLite :

Passer à une base de données plus robuste comme PostgreSQL pour mieux gérer une grande quantité de données et bénéficier des fonctionnalités avancées de gestion de données.

7.4 Ajouter d'autres pages dans l'application web :

Développer de nouvelles pages dans l'application web pour organiser et structurer les données de manière plus claire et séparée, facilitant ainsi l'accès et l'interprétation des informations par les utilisateurs.

Conclusion

La réalisation de ce projet dans le cadre de notre formation à Epitech, a été une expérience très enrichissante pour notre groupe. Nous avons perfectionné nos techniques de travaille en groupe, ainsi que les techniques de traitement des données à partir de scraping de données en collectant des

informations sur des sites, ce qui nous a permis de mieux comprendre les défis liés à l'extraction de grandes quantités de données non structurées et leur conversion en formats utilisables.

L'utilisation d'Apache Spark pour le nettoyage et l'agrégation des données nous a aidés à gérer efficacement les gros volumes de données brutes, nous permettant d'identifier et de résoudre les problèmes courants de qualité des données.

Travailler avec des bases de données SQL et NoSQL nous a également aidés à comprendre les forces et les limites de chaque type de base de données, et à choisir la technologie adaptée aux besoins spécifiques de notre projet.

Le développement de l'interface utilisateur pour la visualisation des données a renforcé notre compréhension des bonnes pratiques en matière de développement front-end, d'expérience utilisateur et de design d'interface.

La gestion de volumes de données aussi importants a été un véritable défi, nécessitant des optimisations constantes des outils et des infrastructures disponibles.

La performance de notre pipeline ETL était cruciale, et nous avons dû optimiser les jobs Spark et gérer efficacement les ressources.

Intégrer et harmoniser des données de différentes sources a souvent été complexe, demandant des scripts de transformation sophistiqués pour assurer la cohérence et l'exactitude des données.

Utiliser SQLite pour une grande quantité de données a montré ses limites en termes de performance et de scalabilité, soulignant l'importance de choisir des technologies adaptées.

En conclusion, ce projet nous a permis de renforcer nos compétences techniques et notre capacité à résoudre des problèmes complexes, nous donnant une compréhension plus profonde des réalités du traitement des Big Data et de la mise en œuvre de solutions analytiques à grande échelle, compétences qui seront extrêmement utiles dans nos futures carrières professionnelles.