Documentation du projet Hopital-Data-Management

Table des matières

1. Description du projet

Le projet **Hopital-Data-Management** est une solution complète pour le suivi, la visualisation et l'analyse des données médicales d'un hôpital (Donanam). Chaque service de l'hôpital possède sa propre collection de patients dans une base MongoDB. Le projet permet de :

- Gérer les flux de données entre services.
- Analyser les tendances de fréquentation et de traitement.
- Visualiser les statistiques par année et par service.
- Fournir un tableau de bord interactif avec Streamlit.
- Surveiller l'activité avec Prometheus et Grafana.

Objectifs principaux

- Enregistrement des patients dans chaque collection MongoDB correspondant à un service.
- Affichage de statistiques médicales : nombre de patients, âge moyen, note moyenne, durée moyenne de traitement.
- Visualisation interactive avec courbes, histogrammes et cartes de chaleur.
- Monitoring des données et de l’activité en temps réel.

---

2. Technologies utilisées

- **MongoDB** : Stockage des données médicales par service.
- **Python** : Scripts de traitement et génération de données.
- **Streamlit** : Application web interactive pour visualiser les statistiques.
- **Prometheus** : Collecte des métriques de performance.
- **Grafana** : Tableaux de bord pour les métriques Prometheus.
- **Docker** : Conteneurisation de tous les services.

---

3. Architecture du projet

```
/hopital-data-management
├── docker-compose.yml
├── Dockerfile
├── Dockerfile.streamlit
├── app/
│   ├── main.py
│   ├── db.py
│   ├── dashboard.py  ← Interface Streamlit
│   ├── generate_data.py
├── requirements.txt
```

Composants principaux

- **MongoDB ReplicaSet** : Pour la haute disponibilité.
- **Streamlit** : Pour l'affichage de statistiques.
- **Prometheus & Grafana** : Pour le monitoring.
- **Python Scripts** : Pour générer et traiter les données.

---

4. Configuration de l'environnement

Prérequis

- Docker & Docker Compose
- Python 3.10+
- Accès à un terminal Linux ou Windows avec WSL

Variables d’environnement

- `MONGO_URI` : URI MongoDB avec ReplicaSet

---

5. Exécution du projet

Démarrage rapide

```
docker-compose build
docker-compose up
```

- Accès à Streamlit : `http://localhost:8501`
- Accès à Grafana : `http://localhost:3000` (admin/admin)
- Accès à Prometheus : `http://localhost:9090`

Exécution manuelle (hors Docker)

1. Installer les dépendances :

```
pip install -r requirements.txt
```

2. Générer les données :

```
python app/generate_data.py
```

3. Lancer l’interface Streamlit :

```
streamlit run app/dashboard.py
```

4. (Optionnel) Lancer l’API Flask :

```
python app/main.py
```

---

6. Base de données MongoDB

Chaque collection représente un service :

```
"medecine_generale", "immunologie", "radiologie", "chirurgie",
"neurologie", "cardiologie", "ontologie", "pneumologie",
"odontologie", "pediatrie"
```

Exemple de document patient :

```
{
  "nom": "Jean Dupont",
  "age": 52,
  "date": "2025-03-15",
  "note": 8.5,
  "duree_traitement": 5.2
}
```

---

7. Génération et gestion des données

Le script `generate_data.py` permet de remplir la base avec des données aléatoires pour chaque service. Il simule :

- L’âge du patient
- La date d’admission
- La note moyenne de traitement
- La durée moyenne de traitement

---

8. Visualisation avec Streamlit

L’interface Streamlit (`dashboard.py`) permet :

- De choisir une année (2025 à 2028)
- D'afficher les statistiques globales sous forme de tableau
- D'afficher des graphiques interactifs :
    - Histogrammes
    - Courbes
    - Heatmaps (via Plotly)

Données affichées par service

- Nombre de patients
- Âge moyen

- Note moyenne
- Durée moyenne de traitement

---

9. Monitoring avec Prometheus et Grafana

- **Prometheus** collecte les métriques exportées par des endpoints Flask ou via un exporter personnalisé Python.
- **Grafana** lit ces métriques et affiche des tableaux de bord interactifs :
    - Évolution du nombre de patients
    - Moyennes par service ou par année

---

10. API Flask pour l'affichage des données (optionnel)

Une API Flask (`main.py`) peut être utilisée pour exposer des endpoints comme :

- `GET /patients`
- `GET /stats/{service}`
- `GET /trend`

---

11. Gestion des erreurs et performances

- Validation des connexions MongoDB avec gestion d’erreurs `try/except`
- Utilisation d’agrégations MongoDB pour éviter de charger tous les documents
- Indexation sur les champs `date` et `service`

---

12. Exemples d'utilisation

Visualiser les données d’une année

- Aller sur Streamlit → Sélectionner l’année → Lire les tableaux et les graphes

Afficher les tendances en temps réel

- Aller sur Grafana → Dashboard général → Visualisation des tendances

---

Auteur : Djeradé Golbé Parfait

Projet réalisé dans le cadre de la DSI de l’hôpital Donanam