Massyle Dekkar

Bilal Bouziane

Bienvenue dans le Manuel d'Utilisation du Projet

Bonjour et bienvenue dans ce manuel d'utilisation dédié au projet de génération de messages FHIR, à la transmission de messages via Kafka, ainsi qu'à leur indexation et visualisation à l'aide d'Elasticsearch et Kibana. Ce document a pour objectif de détailler la configuration et l'interaction des différents fichiers utilisés dans notre projet, afin de vous offrir une compréhension approfondie de son fonctionnement.

L'ensemble des informations essentielles relatives à ce projet est disponible dans notre fichier README.md, que vous pouvez consulter sur notre repository GitHub à l'adresse suivante : ChartMasterMind/FHIR.

Le fichier README.md explique les principales étapes nécessaires pour exécuter et tester les scripts sur votre environnement local. Vous y trouverez également une description des commandes Docker, Git et Linux requises.

Le présent manuel vise quant à lui à approfondir ces explications en détaillant :

- La structure et le rôle de chaque fichier dans le projet.
- Les interactions entre les différents fichiers.
- Les choix effectués concernant la génération et la sélection des données.
- Les raisons de la pertinence des données visualisées.
- L'objectif global de la visualisation mise en place.

Voici un bref contexte de notre projet extrait du README de notre repository :

Contexte :

La surveillance des patients grâce à leurs mesures de pression artérielle est essentielle pour détecter les cas nécessitant une prise en charge rapide.
En s'appuyant sur le standard **FHIR** (documentation officielle), ce projet vise à :

- Générer et analyser des données médicales standardisées.
- Détecter les anomalies critiques.
- Fournir une visualisation claire et efficace pour un suivi renforcé.

Fonctionnalités :

1. **Génération de messages FHIR** :
    - Création de fichiers JSON contenant des observations de pression artérielle (systolique et diastolique).
2. **Transmission avec Kafka** :
    - Publier les données générées sur un topic Kafka.
    - Récupérer les messages pour analyse avec un consumer Kafka.
3. **Analyse des données** :
    - Détecter les anomalies selon les règles suivantes :
        - **Systolique** : > 140 mmHg ou < 90 mmHg.
        - **Diastolique** : > 90 mmHg ou < 60 mmHg.
4. **Traitement des données** :
    - **Pression artérielle anormale** : Indexation dans Elasticsearch avec métadonnées.
    - **Pression artérielle normale** : Archivage local dans des fichiers JSON.
5. **Visualisation avec Kibana** :
    - Création de Dashboard pour observer :
        - La répartition des anomalies.
        - Les tendances des pressions artérielles.
        - Les cas critiques nécessitant une attention immédiate.

Objectifs du Projet :

Développer une solution complète pour analyser et traiter les données de pression artérielle afin d'améliorer le suivi médical des patients.

Objectifs Spécifiques :

1. Générer des messages FHIR au format JSON pour différents patients.
2. Publier ces messages sur Kafka et les consommer pour analyse.
3. Détecter les anomalies selon les seuils définis.
4. Indexer les anomalies dans Elasticsearch et visualiser les données dans Kibana.
5. Archiver les données normales localement.

Comme indiqué dans le fichier README.md, notre repository contient plusieurs fichiers, chacun ayant une importance cruciale.

Ce manuel vous guidera pas à pas dans la compréhension de chaque fichier, de ses interactions avec les autres composants, et de la logique adoptée pour leur construction.

SOMMAIRE

1. Message_FHIR_Project.py

Un message FHIR (Fast Healthcare Interoperability Resources) est un format standardisé utilisé pour échanger des informations de santé entre différents systèmes et applications. Ce format est largement adopté dans le domaine de la santé pour assurer l'interopérabilité des données médicales. Les messages FHIR sont conçus pour être simples, flexibles et évolutifs, afin de faciliter l'intégration de données provenant de diverses sources, tout en étant compatibles avec des technologies modernes comme le web et les API.

Le message FHIR se compose de ressources qui représentent des concepts cliniques ou administratifs spécifiques dans le domaine de la santé, telles que les informations sur le patient, les observations cliniques, les diagnostics, les prescriptions, etc. Ces ressources sont représentées sous forme de documents JSON ou XML, et elles peuvent être combinées pour former des messages complets.

Le script Message_FHIR_Project.py génère ces messages FHIR afin de simuler des enregistrements de données cliniques (par exemple, des mesures de pression artérielle) en suivant les normes FHIR. Chaque message FHIR généré par ce script contient des informations pertinentes sur un patient, telles que son ID, ses mesures de pression artérielle (SYS et DIA), et d'autres données relatives à son état de santé.

Ainsi, ce fichier est essentiel pour la simulation et l'envoi de messages FHIR dans le cadre de votre projet, où ces données peuvent être analysées, traitées et intégrées dans un système de gestion des informations de santé.

Maintenant revenons au script. Ce script contient 2 fonctions et 10 bibliothèques. Il génère aussi un fichier : dernière_date.txt.

a. dernière_date.txt

Un fichier nommé "dernière date" est créé ou mis à jour à chaque exécution ou réexécution du script *Message_FHIR_project*. Ce fichier stocke la dernière date utilisée pour générer une observation FHIR. Si aucune date n'est sauvegardée, le script démarre à partir d'une date par défaut. Ce choix a été fait pour garantir une génération réaliste et chronologique des observations, permettant ainsi de créer une nouvelle base d'analyse à chaque exécution. Cela évite que les données générées s'accumulent sur celles déjà existantes, ce qui aurait pu arriver si le script repartait systématiquement d'une date fixe (comme le 1er janvier 2021), entraînant une duplication incohérente des observations. La date est également convertie au fuseau horaire UTC pour assurer une cohérence temporelle, ce qui a également permis de résoudre des problèmes rencontrés lors de l'envoi des données sur Kafka.

b. Notre Modèle

À l'aide de la bibliothèque Faker, nous avons généré une liste de 100 patients fictifs. Chaque patient est associé à un nom, un identifiant unique (UUID), et un sexe déterminé aléatoirement à partir de son nom. Cette liste permet de prendre un échantillon aléatoire de 100 individus et d'assurer le suivi de leur tension artérielle dans le temps.

Cette approche est essentielle, car nous partons de l'hypothèse qu'un patient peut présenter des pressions artérielles anormales à différents moments. Cela nous permet de surveiller toutes les

occurrences de pressions anormales et, potentiellement, d'anticiper les traitements nécessaires en fonction des antécédents du patient. Nous pouvons également analyser la fréquence des pathologies artérielles spécifiques qu'un patient a pu subir au fil du temps.

Bien que le sexe ait été généré pour chaque patient, il n'a pas été inclus dans les observations FHIR ni dans le tableau de bord. En effet, des études confirment que le sexe n'a pas d'impact significatif sur la pression artérielle. Cependant, cette variable est conservée dans le code afin que vous puissiez, si besoin, l'ajouter aux observations FHIR et l'indexer dans Elasticsearch. Les instructions pour ce faire sont détaillées dans le README. L'inclusion du sexe est avant tout esthétique et vise à mieux référencer chaque patient pour un suivi plus détaillé.

Toutes ces informations sont organisées dans un dictionnaire associant chaque patient à son ID et à ses données personnelles, garantissant ainsi une cohérence et un réalisme dans la simulation du suivi des patients.

Pour les mesures de pression artérielle, systolique et diastolique, nous utilisons également la bibliothèque Faker. Rappelons que :

- **La pression systolique** (pression dans les artères après la pulsation du cœur) est considérée comme normale entre 80 mm Hg et 120 mm Hg.
- **La pression diastolique** (pression dans les artères au relâchement du cœur) est considérée comme normale entre 60 mm Hg et 90 mm Hg.
  Des valeurs en dehors de ces plages indiquent une tension artérielle anormale.

Les plages de valeurs générées par notre modèle sont :

- Pression systolique : entre 78 mm Hg et 190 mm Hg.
- Pression diastolique : entre 40 mm Hg et 130 mm Hg.

L'ensemble de ces valeurs viennent de source fiables http://unitsofmeasure.org)

La génération de la date est une partie essentielle de notre modèle. Cette variable a été conçue pour rendre la génération des messages FHIR aussi réaliste que possible, en simulant un envoi de messages en temps réel, ou du moins dans un temps accéléré. À chaque observation, une date aléatoire est générée, mais elle est toujours postérieure à la date de l'observation précédente. Cela permet de reproduire le fonctionnement réel, où les messages FHIR sont envoyés de manière dynamique et chronologique. Ainsi, bien que les dates soient générées aléatoirement, elles respectent toujours un ordre temporel logique, garantissant qu'aucune date ne précède celles déjà générées.

c. Le message FHIR

Nous avons également développé une fonction essentielle pour générer des messages FHIR pour différents patients : **generate_blood_pressure_observation()**. Nous avons utilisé le site http://terminology.hl7.org/CodeSystem/observation-category pour élaborer notre Message FHIR.

Cette fonction permet de créer un message FHIR contenant les informations suivantes :

- Le nom du patient.
- L'ID unique rattaché au patient.
- Les valeurs de sa pression diastolique et systolique

- La date à laquelle l'observation a eu lieu.

Le message ainsi généré est retourné au format **JSON**, ce qui est crucial pour l'intégration avec le producteur Kafka. Ce choix de format JSON garantit une compatibilité optimale avec les systèmes modernes de gestion et d'échange de données, tout en assurant une standardisation conforme aux normes FHIR.

Pour la conversion au format JSON, nous avons délibérément opté pour la fonction **dumps** au lieu de **dump**. Cette décision a été prise afin d'éviter de créer un nouveau fichier JSON à chaque itération, ce qui pourrait entraîner un encombrement inutile du stockage. En utilisant **dumps**, les messages FHIR sont directement générés en mémoire, ce qui permet un traitement plus léger et efficace, particulièrement lorsque le script est exécuté sur de longues périodes ou pour un grand nombre d'observations.

Cette approche garantit que nos messages FHIR restent dynamiques et adaptés aux besoins de simulation, tout en minimisant les contraintes liées à la gestion des fichiers.

d. Ajuster la période temporelle

Vous pouvez également ajuster la période temporelle générée par notre script **Message_FHIR.py**. Cela vous permet de personnaliser l'intervalle entre les observations, selon vos besoins spécifiques. Pour plus de détails sur le principe de génération des périodes temporelles et les étapes pour les ajuster, veuillez consulter le fichier **README**.

2. Importation des fichiers vers le producteur Kafka

*Point important, nous avons créé un fichier **Product** et **consumer** pour simuler le rôle du producteur / consommateur dans le système Kafka. Bien qu'il aurait été possible de centraliser tout le traitement dans un seul fichier, nous avons choisi de séparer les rôles de producteur et de consommateur. Cette séparation est essentielle dans le cadre de ce projet, car elle nous permet de mieux comprendre et simuler les différentes problématiques rencontrées par chaque collaborateur. D'un côté, le **producteur** est responsable de la génération et de l'envoi des messages FHIR, tandis que de l'autre, le **consommateur** prend en charge la réception, le traitement et l'analyse de ces messages. Cette distinction nous aide à mieux cerner les défis liés à la production des messages (comme la gestion des données envoyées) et ceux associés à la consommation et au traitement de ces messages (comme l'analyse et la distribution des informations reçues).*

À la fin de notre script, vous remarquerez qu'une fonction appelée **produce_to_kafka(observations)** est exécutée. Cette fonction, définie dans le fichier **product.py**, est importée dans notre script **Message_FHIR_Project.py**.

Le rôle de cette fonction est de transmettre tous les messages FHIR générés par le script au producteur Kafka. Ces messages sont ensuite envoyés à des consommateurs spécifiques via le système de messagerie Kafka. Kafka, en tant que système de streaming distribué, est particulièrement adapté pour gérer de grands volumes de données en temps réel tout en garantissant leur fiabilité. Il permet donc une transmission de donnée (messages) en temps réel pour de gros volume de données, très utile pour détecter des anomalies à tout moment. Kafka à besoins aussi de donnée en format JSON ce qui explique pourquoi nous avons converti nos message FHIR en donnée JSON.

Pour rendre cela possible, il a été nécessaire d'installer les dépendances nécessaires à l'aide de **Docker Compose**. Les étapes détaillées pour configurer Docker Compose se trouvent dans le fichier **README**.

Le Docker Compose est un outil qui permet de définir et de gérer des environnements multi-conteneurs Docker. Dans notre projet, il a été utilisé pour créer des conteneurs isolés, comme un conteneur pour Kafka et un autre pour Zookeeper (qui est essentiel pour le fonctionnement de Kafka), assurer aussi une interaction entre ces conteneurs, tout en maintenant un environnement de travail isolé et surtout de simplifier la gestion des dépendances et des bibliothèques requises.

Une fois Docker installé et le fichier **docker-compose.yml** télécharger, nous avons configuré notre environnement en lançant un conteneur Docker avec Kafka et ZooKeeper. Ensuite, nous avons spécifié dans notre configuration du producteur Kafka l'option producer = Producer({'bootstrap.servers': 'localhost:9092'}) pour indiquer à notre producteur Kafka de se connecter au serveur Kafka local qui tourne sur le port 9092. Nous avons également créé un **topic** spécifique sur Kafka, qui sert de canal pour l'envoi des messages FHIR générés. Ce **topic** permet de centraliser toutes les observations de pression artérielle générées, ce qui facilite leur traitement et analyse.

3. Consommation des fichier, traitements et indexation.

a. Consommations et traitements

Avant de continuer, il est important de définir le processus de **consommation des fichiers sous Kafka**. Kafka fonctionne selon un modèle producteur-consommateur. Le producteur, que nous avons configuré précédemment, envoie des messages à un **topic** Kafka. Ces messages peuvent être ensuite consommés par un ou plusieurs **consommateurs** qui les lisent à partir de ce **topic**.

Dans le cadre de notre projet, un consommateur Kafka est chargé de récupérer les messages FHIR envoyés par le producteur et de les traiter pour ensuite les analyser.

La gestion de cette consommation repose sur des **groupes de consommateurs**, permettant de répartir les messages entre plusieurs instances pour un traitement parallèle et plus rapide.

Concernant la configuration de notre fonction consumer nous avons spécifié plusieurs paramètres essentiels pour permettre la réception des messages depuis Kafka :

- **« bootstrap.servers » :** Ce paramètre définit l'adresse du serveur Kafka auquel se connecter. Dans notre cas, il pointe vers **localhost:9092**, ce qui signifie que Kafka est exécuté sur la même machine avec le port 9092.

- **« group.id »** : C'est l'identifiant du groupe de consommateurs. Ici, nous avons défini le groupe comme **« python-consumers »**. Cela permet à Kafka de gérer les offsets (qui sont enfaites les positions des messages dans le topic) pour ce groupe, afin de garantir que chaque message est consommé une seule fois par chaque membre du groupe.

- **« auto.offset.reset »** : Ce paramètre définit ce qui se passe lorsqu'il n'y a pas d'offset enregistré pour un consommateur (par exemple, lors de la première exécution). Ici, nous avons choisi **« earliest »**, ce qui signifie que le consommateur commencera à lire les messages à partir du tout premier message du topic.

Ensuite, avec la commande consumer.subscribe(['blood_pressure_topic_final_1']), nous avons inscrit notre consommateur au topic **'blood_pressure_topic_final_1'**. Cela signifie que notre consommateur va écouter ce topic pour recevoir les messages qui y sont publiés.

Concernant le traitement des données, nous avons crée une fonction de détection d'anomalie de pression artérielle appelé detect_anomaly (). Nous nous sommes appuyés sur un document fourni par notre professeur.

L'objectif du traitement des données était de conserver et d'indexer les valeurs anormales sur ElasticSearch et les valeurs normales dans un fichier JSON que l'on à nommer **normal_blood_pressure.json**. Pour cela nous avons créé une fonction save_normal_data () qui permet d'ajouter tous les message FHIR ayant des valeurs normales dans un même fichier JSON (pour éviter d'avoir une quantité astronomique de fichier).

A chaque fois qu'un nouveau message sera présent sur le topic Kafka blood_pressure_topic_final_1, il va être consommer puis traiter grâce à la fonction detect_anomaly().

b. Indexation sous ElasticSearch

L'indexation dans **Elasticsearch** fait référence au processus de stockage et d'organisation des données pour les rendre facilement accessibles et recherchables. Cela permet de structurer les données de manière à pouvoir les interroger efficacement et de les exploiter dans des outils comme **Kibana** pour la visualisation.

Dans notre projet, une fois les données traitées (notamment les observations de la pression artérielle), nous avons procéder à l'indexation des données a**normales** dans **Elasticsearch**. Cela signifie que toutes les mesures de pression artérielle jugées « anormales » (c'est-à-dire, les tensions systoliques et diastoliques qui dépassent les seuils normaux) sont envoyées à Elasticsearch, où elles sont stockées sous forme d'index.

Ces index sont ensuite utilisés pour créer des tableaux de bord sur **Kibana**. Kibana est un outil de visualisation qui permet de représenter graphiquement les données stockées dans Elasticsearch sous forme de graphiques, diagrammes et autres visualisations interactives.

Grace à l'indexation dans Elasticsearch, cela peut nous offrir plusieurs avantages :

- **Recherche rapide** : Grâce à l'indexation, nous pouvons rechercher et extraire des données de manière extrêmement rapide et efficace.
- **Facilité d'analyse** : Une fois les données indexées, elles peuvent être facilement agrégées, filtrées et analysées dans Kibana.
- **La Scalabilité** : Elasticsearch est conçu pour gérer de grandes quantités de données et permettre des recherches distribuées, ce qui est utile lorsque nous travaillons avec un grand volume de mesures de pression artérielle.

Pour indexer les données jugées anormale, nous consommons et traitons les message FHIR contenant les valeurs en anormale en extrayant les valeurs qui nous semblait les plus pertinente à intégrer dans notre Dashboard Kibana à savoir les pressions systoliques et diastoliques, le prénom et l'ID du patient, le type d'anomalie qui est renvoyé par la fonction detect_anomaly() et la variable date qui nous permettra de faire des analyses temporelles.

Le sexe aurait pu être intégrer à cette indexation à condition d'intégrer un champ « sex » dans la structure du message FHIR (au choix mais non conventionnelle).

Nous stockons les données dans un body appelé **anomaly_data**. Un **body** permet de structurer et d'envoyer les informations nécessaires à Elasticsearch. Dans notre cas, ce body contient toutes les données liées aux observations de pression artérielle, comme les valeurs systolique et diastolique, ainsi que le statut de l'observation

Une fois que les données sont indexées dans Elasticsearch, elles peuvent être récupérées et visualisées à partir de **Kibana**, à l'adresse du serveur Kibana, généralement **localhost:5601**. Pour y accéder, allez dans la section **Index Management** où vous pourrez créer un index.

Lors de la création de l'index, il est important de spécifier que vous avez une colonne **date**. Cela permet à Kibana de bien gérer les données temporelles et de les exploiter correctement dans les visualisations.

Une fois l'index créé, vous pourrez naviguer vers la section **Discover**, où vous pourrez consulter et explorer les données indexées. Vous pourrez ensuite créer des **tables** en sélectionnant les colonnes les plus pertinentes pour votre analyse, comme les mesures de pression artérielle, les dates, et autres informations pertinentes pour vos besoins d'analyse. Cela vous permettra d'identifier les tendances et d'ajouter des visualisations basées sur ces données.

4. Création de notre Dashboard sur Kibana

a. Dashboard générale

Le lien du Dashboard se trouve à cette adresse :
FHIR/DashBoard_Kibana/DashBoard_Apercu_des_données.png at main · ChartMasterMind/FHIR

Nous avons conçu notre **Dashboard** pour inclure toutes les mesures pertinentes dans le suivi des patients, en particulier concernant leurs anomalies artérielles. Ce Dashboard sans filtrage permet de donner un aperçu général sur la repartions des types d'anomalies, des données temporelles, des informations relatives aux patients.

Nous faisons l'hypothèse que nous travaillons pour un hôpital et que ce tableau de bord est mis à leur disposition pour améliorer le suivi de leurs patients. Concrètement, chaque observation présente sur ce tableau de bord indique qu'une mesure de la pression artérielle a été effectuée au sein de cet hôpital et que le traitement de cette donnée a également été réalisé dans le même établissement.

Tout d'abord, nous avons créé un graphique qui montre la répartition des types d'anomalies artérielles observées chez les patients. Ce graphique peut être très utile pour les hôpitaux, par exemple, afin d'allouer des ressources de manière plus ciblée, en se concentrant sur les pathologies artérielles les plus fréquentes.

Par exemple, en fournissant ce Dashboard aux hôpitaux, ils pourront avoir une vue d'ensemble des anomalies artérielles les plus courantes, telles que l'hypertension de stade 2 et les crises hypertensives. Cette information leur permettra de prioriser le traitement de ces pathologies spécifiques, d'ajuster leur gestion des ressources et d'augmenter la commande de traitements visant à abaisser la tension artérielle, comme des médicaments antihypertenseurs ou des traitements pour prévenir les crises hypertensives.

Ainsi, ce Dashboard soutient la prise de décision médicale en offrant une visualisation claire des données relatives aux anomalies artérielles, permettant aux professionnels de santé de mieux cibler leurs interventions.

Nous avons aussi créé des cartes pour pouvoir avoir un aperçu du nombre de patients ayant des tensions anormales, le nombre de mesure qui ont était prise en temps actuelle, et le nombre de personne atteint de différents types d'anomalies (hypertension de stade 2 et 1, hypotension, et crise hypertensive).

Nous avons lié hypertension de stade 2 et 1 car ces deux stades impliquent une hypertension chronique et avérée. Elle nécessite souvent les mêmes traitements à dose variable.

Une explication de ces cartes pour bien comprendre ce qu'elle signifie s'impose.

IMPORTANT: Dans ce Dashboard, nous avons exécuter nos scripts deux fois.

Étant donné que notre modèle de génération suppose qu'un patient peut être présent dans plusieurs messages FHIR, car cela reflète le fait qu'il peut se rendre à l'hôpital à plusieurs reprises au cours d'une période donnée en raison de fluctuations de sa pression artérielle (hausses ou baisses). Cette hypothèse permet aussi d'obtenir un suivi individuel avec des mesures SYS et DIA différentes à chaque génération pour simuler une évolution réaliste. Par conséquent, le nombre de patients ayant des tensions anormales sera toujours inférieur au nombre total de mesures prises pendant la période couverte par la génération (c'est-à-dire l'ensemble du temps généré par notre code).

Pour matérialiser ces données, nous avons donc fait un unique_counts des patients que l'on retrouver sur elasticsearch.

Cela veut dire que nous avons dans notre Dashboard 123 patients atteint de tension anormale parmi un groupe de 200 patients (car nous avons exécuter le code 2 fois donc nous avons générer une base de données contenant 200 patients (100 patients à chaque exécution du script Message_FHIR_Project.py).

Le nombre de mesure pour ces patients est de 870 fois. Il faut aussi prendre en compte le fait que la période temporelle dépend du nombre d'itération dans le script

Une boucle de 1000 itérations (comme c'est le cas ici) génère une période moyenne de 7 ans. Cela signifie qu'en moyenne, chaque patient a entre 1 et 2 mesures par an où une anomalie artérielle est détectée. Ce rythme est réaliste, car il correspond à la fréquence des visites à l'hôpital pour les personnes suivies pour de l'hypertension artérielle.

De même pour les valeurs liées aux crises hypertensives, à l'hypertension et à l'hypotension. Étant donné que les données SYS et DIA sont générées aléatoirement, un individu peut présenter, sur une période de 7 ans, plusieurs types d'anomalies (ou plusieurs occurrences d'un même type d'anomalie). Cela peut artificiellement gonfler les valeurs affichées dans les cartes concernant l'hypertension, les crises hypertensives et l'hypotension.

Par exemple, un individu peut être détecté en 7 ans avec 4 hypertensions et 2 hypotensions. Ainsi, les cartes qui affichent « le nombre de personnes ayant une hypertension de stade 1 et 2 » montreront une valeur gonflée. Dans ce cas, on compterait 1 patient avec une tension anormale, mais 4 occurrences d'hypertension et 2 occurrences d'hypotension.

La bonne interprétation des cartes telles que « le nombre de personnes ayant une hypertension de stade 1 et 2 », « le nombre de personnes ayant eu une crise hypertensive » et « le nombre de

personnes ayant une hypotension » serait donc le nombre total d'occurrences d'un type particulier d'anomalie détecté au fil du temps, et non le nombre de patients uniques.

Nous avons ensuite analysé les données temporelles en élaborant plusieurs graphiques à des fins d'analyse, notamment pour détecter d'éventuelles saisonnalités dans les données. Par exemple, nous pourrions formuler l'hypothèse que la tension artérielle augmente en été et vérifier cette hypothèse grâce à nos données.

Pour cela, nous avons créé un graphique qui montre le **nombre d'observations de patients détectés ou redétectés avec une anomalie artérielle, réparti par mois**. Cela permet de visualiser les mois durant lesquels l'hôpital a enregistré le plus de patients présentant des anomalies artérielles, aidant ainsi à identifier d'éventuelles tendances saisonnières ou périodes critiques.

Nous avons aussi créé un graphique qui montre le **nombre de patients ayant était mesurer avec des tensions artérielles anormales, réparti par jours.** Cela permet de visualiser les jours durant lesquels l'hôpital a enregistré le plus de patients présentant des anomalies artérielles, aidant ainsi à identifier sur des jours sont plus propices à faire augmenter la tension artérielle.

Par exemple, nous pourrions formuler l'hypothèse que la tension artérielle augmente le week-end du fait des sortie en boite de nuit ou des soirées très alcoolisés qui augmente fortement la tension artérielle.

Enfin, nous avons créé un tableau qui regroupe toutes les informations des patients pour un meilleur suivi par exemple. Il contient le nom de chaque patient, sont ID associé, la date et l'heure auquel ils ont été mesurer avec des pressions artérielles anormales, leurs pression SYS et DIA au moment de la mesure et leur type de pathologie artérielle.

Nous avons également développé un système de filtrage qui offre plusieurs possibilités pour affiner l'analyse des données. Il permet :

- **De filtrer par nom de patient afin de suivre l'évolution spécifique d'un individu.**
- **De sélectionner les patients atteints d'une certaine pathologie artérielle, comme l'hypertension ou l'hypotension.**
- **D'isoler les patients détectés avec une pression SYS ou DIA supérieure ou inférieure à une certaine valeur, pour repérer des cas critiques ou des anomalies spécifiques.**
- **De choisir une période temporelle précise, avec la possibilité de filtrer par jour, par mois ou par année, ou même de définir un intervalle personnalisé.**

b. Suivi du patient

Le lien du Dashboard se trouve à cette adresse :

FHIR/DashBoard_Kibana/DashBoard_Données_filtrées_pour_un_patient.png at main · ChartMasterMind/FHIR

Notre Dashboard est aussi concu pour pouvoir suivre individuellement les patients. Pour cela, nous devons sélectionner un patient particulier dans notre système de filtrage.

Chaque graphique nous permet d'avoir une indication sur le patient pour un meilleurs suivi.

Le graphique sur la répartition des types d'anomalies permet d'identifier les pathologies rencontrées par les patients.

- **Les cartes** montrent le nombre de fois qu'un patient a été détecté avec chaque type de pathologie, ce qui aide à quantifier la fréquence des anomalies.
- **Les graphiques temporels**, en particulier « le **nombre de patients ayant était mesurer avec des tensions artérielles anormales, réparti par jours »**, permettent de situer chaque mesure où un patient a été détecté avec une tension artérielle anormale. Cela aide à analyser si ces anomalies apparaissent à intervalles réguliers, rapprochés, ou de manière aléatoire (ce qui est le cas ici, puisque les données sont générées aléatoirement).

Nous avons également deux graphiques qui prennent tout leur sens lors du suivi des patients :

- **Les graphiques de pression artérielle SYS et DIA mesurées par intervention/patient** :
  Ces graphiques permettent de visualiser, dans le temps, l'évolution des pressions systolique (SYS) et diastolique (DIA) pour chaque patient.

Ils sont particulièrement utiles pour :

- Analyser si la pression artérielle d'un patient diminue ou augmente sous l'effet de certains médicaments.
- Identifier l'influence d'autres facteurs, comme les conditions météorologiques (été, hiver), les périodes de fêtes, ou d'autres événements.

Ces visualisations permettent ainsi d'offrir un meilleur suivi et une meilleure compréhension des variations de la tension artérielle au fil du temps.

Nous avons également, dans notre tableau **Patient ID**, toutes les informations relatives aux patients filtrés :

- Toutes les pressions SYS et DIA mesurées et détectées comme anormales.
- Les pathologies artérielles détectées.
- Enfin, la date à laquelle le patient a été mesuré avec ces pressions artérielles (bien que non visible sur la capture d'écran).

Ce tableau permet de centraliser les données essentielles pour un suivi précis et efficace de chaque patient.

5. Le fichier lancement.py

Pour lancer l'ensemble des scripts, il vous suffit d'exécuter le fichier lancement.py depuis votre terminal ou via Visual Studio Code. Ce fichier a pour fonction de démarrer simultanément les fichiers Message_FHIR_Project.py, producer.py et consumer.py, permettant ainsi la génération, la production et la consommation des messages en temps réel. Dès que les messages sont générés par le script Message_FHIR_Project.py, ils sont immédiatement envoyés via producer.py à Kafka et consommés en temps réel par consumer.py.

Cette méthode permet de simuler un flux continu de données, idéal pour tester le traitement des messages dans un environnement de production. Le tout fonctionne de manière fluide et intégrée pour analyser et traiter les données générées dans le cadre du suivi des patients. Une version Dockerisée de ce processus sera disponible prochainement, facilitant encore plus le déploiement et l'utilisation de l'application dans différents environnements.

6. Conclusion :

En conclusion, notre projet illustre l'intégration efficace de technologies modernes comme Apache Kafka, Elasticsearch et le standard FHIR pour le traitement et l'analyse des données médicales. Kafka a permis de gérer le flux de données en temps réel, facilitant le traitement des messages FHIR générés par notre script. Elasticsearch a ensuite assuré un stockage rapide et optimisé pour la recherche, rendant les données facilement accessibles et exploitables dans notre tableau de bord. Ce dernier, conçu avec soin, regroupe des visualisations claires et pertinentes, permettant aux professionnels de santé de suivre les patients, d'analyser les anomalies artérielles et d'identifier des tendances temporelles ou saisonnières. L'ensemble du travail repose sur une chaîne de traitement bien structurée, allant de la collecte des données à leur visualisation.

Le projet montre une bonne compréhension des technologies de streaming de données et de stockage scalable, combinées à des normes médicales comme FHIR, ce qui est essentiel pour des systèmes de suivi patient modernes. La solution est robuste, mais pourrait être enrichie en ajoutant des fonctionnalités comme des analyses prédictives pour anticiper les anomalies ou encore des alertes en temps réel. Néanmoins, le tableau de bord et la chaîne de traitement offrent déjà une solution pratique et bien adaptée aux besoins des hôpitaux pour optimiser la gestion des patients souffrant de pathologies artérielles.