**Architecture Requirements**

*Identify architecture requirements by analyzing constraints (technical, operational,regulatory).*

#### **I - Besoins Métier (BM)**

* **BM-1: Collecte de Données Multi-formats :** La plateforme doit permettre aux utilisateurs d'uploader de manière sécurisée leurs documents comptables aux formats **PDF, CSV et Excel (.xlsx)**.
* **BM-2: Traitement Automatisé :** Le système doit automatiquement extraire les données pertinentes de ces documents, les valider et les structurer pour analyse.
* **BM-3: Calcul d'indicateurs :** Le système doit calculer de manière fiable un set d'indicateurs financiers stratégiques.
* **BM-4: Visualisation :** La plateforme doit présenter ces indicateurs via des tableaux de bord interactifs et intuitifs.

### **II - Contraintes Techniques**

#### *Technologies Imposées*

Conformément au cahier des charges du projet, la stack suivante est imposée :

* **Conteneurisation :** Docker
* **Orchestration :** Kubernetes
* **Workflow Automation :** Apache Airflow
* **Monitoring :** Grafana

#### *Choix Technologiques de l'Équipe*

* **Fournisseur Cloud : Google Cloud Platform (GCP)**
  + **Justification :** GCP est choisi pour l'excellence de son service managé Kubernetes, **Google Kubernetes Engine (GKE)**, qui simplifie drastiquement les opérations de déploiement et de scaling. L'écosystème de GCP est nativement orienté données et IA, ce qui offre des perspectives d'évolution futures pour Quantis (ex: modèles de prédiction). Sa gestion des identités et des accès (IAM) est robuste et granulaire, un point essentiel pour la sécurité.
* **Langage et Framework : Python**
  + **Justification :** Python est le standard de facto pour la manipulation de données grâce à son écosystème riche (Pandas, NumPy, etc.).
* **Bases de Données : PostgreSQL & Redis**
  + **PostgreSQL (Base principale) :** Les données financières extraites sont hautement structurées et relationnelles (liées à une entreprise, une année, etc.). PostgreSQL est la référence des bases de données relationnelles open-source. Sa fiabilité, sa robustesse et ses performances sont idéales pour une application FinTech.
  + **Redis (Cache) :** Pour atteindre des temps de réponse très faibles sur les tableaux de bord, les indicateurs déjà calculés et fréquemment consultés seront mis en cache. Redis est une base de données en mémoire ultra-rapide, parfaite pour cette tâche. L'utilisation d'un cache est une technique d'optimisation fondamentale pour une architecture performante.

### **III - Exigences Opérationnelles**

* **Scalabilité :**
  + **Objectif initial :** Gérer 100 entreprises clientes durant la première année. L'architecture doit être pensée pour pouvoir scaler sans friction à 5000 clients sur 3 ans.
  + **Charge de travail :** Le système doit pouvoir traiter jusqu'à 200 documents par jour, avec des pics d'activité attendus en fin de mois (clôtures comptables).
* **Performance (SLO - Service Level Objectives) :**
  + **Temps de traitement :** 95% des documents importés doivent être traités (extraction, calculs) en **moins de 60 secondes**.
  + **Temps de réponse API :** Le 95ème centile (P95) du temps de réponse des endpoints API servant les tableaux de bord doit être **inférieur à 200ms**.
* **Disponibilité :**
  + **Uptime :** Le service exposé aux clients doit garantir une disponibilité de **99.9%**.
  + **Implications techniques :** Cela impose la mise en place de readiness et liveness probes sur Kubernetes, et le déploiement de chaque microservice critique (API, etc.) avec un minimum de **2 réplicas** (replicas: 2) pour assurer la redondance et la haute disponibilité.

### **IV - Contraintes Réglementaires et de Sécurité**

* **Confidentialité (RGPD) :**
  + **Cloisonnement des Données :** Un cloisonnement logique strict sera appliqué au niveau applicatif. **Absolument toutes les requêtes en base de données devront inclure un filtre sur l'identifiant du client (client\_id)**. Aucune donnée ne doit être accessible sans ce filtre pour prévenir toute fuite de données entre les comptes.
* **Sécurité :**
  + **Chiffrement :** Le chiffrement est obligatoire. **En transit**, tout le trafic sera protégé par TLS 1.2+ (HTTPS). **Au repos**, les données stockées dans PostgreSQL, Redis et Cloud Storage seront chiffrées par défaut par les mécanismes natifs de GCP.
  + **Authentification :** L'accès à l'API sera sécurisé par des **tokens JWT (JSON Web Tokens)** qui ont une durée de vie limitée.
  + **Stockage des documents :** Les fichiers uploadés seront stockés dans un bucket **Google Cloud Storage privé**. L'accès aux fichiers se fera exclusivement via des **URLs signées** à usage unique et à durée de vie très courte, générées par le backend. Le client final n'aura jamais d'accès direct au bucket. C'est un design pattern de sécurité essentiel.