## 🎯 ****Synthèse du Système avec Références en Gestion des Risques****

📌 **Objectif** : Déployer un **système de prédiction du risque fiscal** en intégrant **SQL, NLP (SpaCy), Machine Learning et RDF4J**. Ce système suit les principes de gestion des risques, en s’inspirant du **PMBOK pour la gestion de projet** et du **ISO 31000 pour la gestion des risques**.

📖 **Référence proposée : ISO 31000 – Management du Risque**  
ISO 31000 est une norme internationale qui définit un cadre et des lignes directrices pour la **gestion du risque**. Elle est utilisée dans plusieurs domaines, y compris la gestion fiscale et financière, pour évaluer et réduire les incertitudes.

### ****🛠️ Acteurs Principaux et Leur Interaction avec le Système****

| **🎭 Acteur** | **🔄 Interaction avec le Système** | **📖 Lien avec ISO 31000** |
| --- | --- | --- |
| **Contribuables (Entreprises, Individus)** | Soumettent leurs déclarations fiscales, le système évalue leur risque. | **Identification du risque** basé sur l’historique des paiements. |
| **Inspecteurs Fiscaux** | Accèdent aux prédictions pour prioriser les contrôles. | **Traitement et atténuation du risque** fiscal. |
| **Administrateurs Systèmes** | Gèrent le stockage et l’optimisation de RDF4J et Spring Boot. | **Gestion du risque technologique** et sécurité des données. |
| **Responsables Financiers & Décideurs** | Utilisent les résultats pour ajuster les politiques fiscales. | **Suivi et réévaluation des risques**. |

### ****🔄 Fonctionnement du Système : Inputs et Outputs****

| **⚙ Étape** | **📥 Input** | **📤 Output** | **📖 Lien avec ISO 31000** |
| --- | --- | --- | --- |
| **1.Collecte des Données** | Base SQL : Paiements, retards, amendes. | Jeu de données structuré. | **Contexte et identification du risque**. |
| **2.NLP avec SpaCy** | Textes des paiements et motifs de retard. | Extraction d’entités (Montant, Date, Entreprise). | **Analyse qualitative du risque**. |
| **3.Modélisation Machine Learning** | Données catégoriques, numériques et embeddings. | Score de risque fiscal. | **Évaluation et quantification du risque**. |
| **4.Explicabilité des Prédictions (SHAP)** | Score de risque du modèle ML. | Justification des prédictions. | **Transparence et traçabilité du risque**. |
| **5.Stockage RDF4J** | Résultats ML transformés en format RDF. | Interrogation des scores avec SPARQL. | **Suivi des risques et décisions fondées sur des données**. |
| **6.API REST (Spring Boot)** | Requêtes pour récupérer les prédictions RDF. | JSON/XML exposant les résultats RDF4J. | **Communication et reporting du risque**. |
| **7.Dashboard & Reporting** | Consultation des scores via Metabase. | Visualisation des tendances fiscales. | **Aide à la prise de décision**. |

### ****📌 Scénario d’Utilisation en Gestion des Risques****

🔹 **1. Un contribuable (Entreprise X) soumet sa déclaration fiscale.**  
🔹 **2. L’historique est analysé avec SpaCy et Machine Learning.**  
🔹 **3. Le modèle prédit un score de risque basé sur ISO 31000.**  
🔹 **4. Un inspecteur fiscal interroge RDF4J pour trouver les contribuables à haut risque.**  
🔹 **5. Le score est expliqué via SHAP et visible sur Metabase.**  
🔹 **6. Une action d’atténuation du risque est recommandée (contrôle, mise en conformité).**

### ****🛠️ Stack Technique Complète****

| **🏗 Technologie** | **🔍 Usage** | **📖 Lien avec ISO 31000** |
| --- | --- | --- |
| **PostgreSQL / MySQL** | Stockage des données fiscales. | **Base de référence pour l’évaluation du risque**. |
| **Python (Pandas, SpaCy, Word2Vec)** | Analyse NLP et extraction des entités. | **Détection et catégorisation du risque**. |
| **XGBoost / Random Forest** | Modèle de prédiction du risque. | **Modélisation du risque** pour les contribuables. |
| **SHAP** | Explication des décisions du modèle. | **Transparence et auditabilité**. |
| **Spring Boot** | API REST pour interagir avec RDF4J. | **Communication du risque aux parties prenantes**. |
| **RDF4J (GraphDB, OpenRDF)** | Stockage et interrogation SPARQL. | **Archivage et suivi des risques fiscaux**. |
| **Metabase / Grafana** | Visualisation et reporting. | **Aide à la prise de décision et gestion du risque**. |

### ****📌 Schéma de Flux des Données****

**css**

**CopierModifier**

**🔹 Base SQL → 🔹 NLP (SpaCy) → 🔹 Machine Learning (XGBoost) → 🔹 RDF4J (Stockage & SPARQL)**

**⬇ ⬇ ⬇**

**API REST ← Spring Boot ← Dashboard Metabase/Grafana**

📌 **Lien avec ISO 31000** : Ce flux suit **le cycle de gestion des risques** :  
1️⃣ **Identification et évaluation** des risques via SQL et Machine Learning.  
2️⃣ **Quantification et explication** via SHAP et RDF4J.  
3️⃣ **Communication et suivi** des risques via API et Dashboard.

### 🚀 ****Prochaine Étape : Déploiement de RDF4J avec ISO 31000****

📌 **Objectif** : Installer RDF4J et exécuter des requêtes SPARQL en intégrant une logique de gestion des risques basée sur ISO 31000.

✅ **1. Déploiement de RDF4J avec Docker**

**sh**

**CopierModifier**

**docker run -d -p 7200:7200 --name rdf4j eclipse/rdf4j-workbench**

📌 **Accès à RDF4J** via **http://localhost:7200**

✅ **2. Création d’un Repository pour Suivi du Risque**

* Nom : **fiscal\_risk\_management**
* Type : **Native Store**

✅ **3. Chargement d’une Ontologie de Risque (Turtle RDF)**

**ttl**

**CopierModifier**

**@prefix ex: <http://example.org/fiscal#> .**

**ex:Entreprise\_X a ex:Contribuable ;**

**ex:nom "Entreprise X" ;**

**ex:secteur "Technologie" ;**

**ex:montantDû 100000 ;**

**ex:retardsPaiement 5 ;**

**ex:scoreRisque 0.85 ;**

**ex:niveauRisque "Élevé" .**

📌 **Intégration avec ISO 31000** : L’attribut ex:niveauRisque permet de classifier les risques (Faible, Moyen, Élevé).

✅ **4. Exécution de Requêtes SPARQL pour Identifier les Risques Élevés**

**sparql**

**CopierModifier**

**PREFIX ex: <http://example.org/fiscal#>**

**SELECT ?contribuable ?score ?niveau WHERE {**

**?contribuable ex:scoreRisque ?score .**

**?contribuable ex:niveauRisque ?niveau .**

**FILTER (?score > 0.8)**

**}**

📌 **Sortie attendue** : Liste des entreprises en **risque fiscal élevé**.

✅ **5. Exposition de RDF4J via API REST avec Spring Boot**

**java**

**CopierModifier**

**@GetMapping("/risques-eleves")**

**public ResponseEntity<List<ContribuableDto>> getContribuablesRisqueEleve() {**

**String sparqlQuery = "PREFIX ex: <http://example.org/fiscal#> " +**

**"SELECT ?contribuable WHERE { " +**

**"?contribuable ex:scoreRisque ?score . FILTER(?score > 0.8) }";**

**List<ContribuableDto> result = rdf4jService.executeSparqlQuery(sparqlQuery);**

**return ResponseEntity.ok(result);**

**}**

📌 **Résultat** : Une API REST exposant les contribuables en risque élevé.

### ****🎯 Résultat Final : Une Gestion du Risque Optimisée avec RDF4J et ISO 31000****

📌 **Entrées :** Données SQL (paiements, retards).  
📌 **Traitement :** NLP, ML, stockage RDF.  
📌 **Sorties :** API REST, requêtes SPARQL, dashboards.  
📌 **Action :** **Optimisation de la gestion des risques fiscaux.**

🚀 **Prochaine Étape : un schéma détaillé montrant comment l'ISO 31000 structure le modèle RDF ?**