****

**Année Académique 2023-2024**

**Encadreur :**

Monsieur Oumar KY

**Collaborateurs du projet**

TRAORE Issouf

OUEDRAOGO A. ROCH

OUEDRAOGO Souleymane

MILLOGO O. Nicolas

**THEME :**

**Conception et Développement d’une Application RESTful avec Spring Boot et Talend Open Studio : Intégration de MySQL et Monitoring avec Prometheus. Cas spécifique de gestion des articles.**

**PROJET DE DEVELOPPEMENT A BASE DE COMPOSANT ET SERVICES WEB**

**Master informatique**

**INSTITUT BURKINABE DES ARTS ET METIERS**

**(IBAM)**

**UNIVERSITE JOSEPH KI-ZERBO**

**(UJKZ)**

# Sommaire

[Sommaire 2](#_Toc177508267)

[Table des illustrations 3](#_Toc177508268)

[Introduction 5](#_Toc177508269)

[1 ÉTUDE PREALABLE 5](#_Toc177508270)

[7.1 Contexte et justification 5](#_Toc177508271)

[7.2 Objectifs du projet 6](#_Toc177508272)

[7.3 Résultats attendus 7](#_Toc177508273)

[7.4 Méthode d'analyse 8](#_Toc177508274)

[7.5 Présentation de l'équipe de travail 8](#_Toc177508275)

[7.6 Présentation des technologies utilisées 9](#_Toc177508276)

[2 CONCEPTION ET ARCHITECTURE DU SYSTÈME 10](#_Toc177508284)

[1.1 Technologies de l'architecture 10](#_Toc177508286)

[1.2 Architecture du système 11](#_Toc177508287)

[3 CONFIGURATION ET PROCESSUS ETL AVEC TALEND OPEN STUDIO 13](#_Toc177508288)

[7.1 Présentation de Talend Open Studio et de son utilisation 13](#_Toc177508293)

[7.1 Installation et configuration de Talend Open Studio 15](#_Toc177508294)

[4 DÉVELOPPEMENT DE L'API RESTFUL AVEC SPRING BOOT 20](#_Toc177508295)

[7.1 Présentation de Spring Boot et de l'API RESTful 20](#_Toc177508297)

[7.2 Création de l'environnement et installation des dépendances 22](#_Toc177508298)

[7.3 Structure de l’application et organisation des routes 25](#_Toc177508299)

[7.4 Endpoints de l'API 28](#_Toc177508300)

[5 TESTS ET DOCUMENTATION 29](#_Toc177508301)

[Procédure d'installation de Postman 29](#_Toc177508302)

[31](#_Toc177508303)

[6. Scénarios de tests définis 33](#_Toc177508304)

[6.1. Requêtes GET 33](#_Toc177508305)

[6.2. Requêtes POST 34](#_Toc177508306)

[6.3. Requêtes PUT 34](#_Toc177508307)

[6.4. Requêtes DELETE 34](#_Toc177508308)

[6.5. Gestion des erreurs 34](#_Toc177508309)

[7 DÉPLOIEMENT ET CONCLUSION 34](#_Toc177508310)

[7.1 Déploiement de l'application 34](#_Toc177508311)

[Conclusion 37](#_Toc177508312)

[Table des matières 38](#_Toc177508313)

# Table des illustrations

[Figure 1:Architecture du système 13](#_Toc177582709)

[Figure 2: Page de téléchargement de talend 17](#_Toc177582710)

[Figure 3: Après installation 18](#_Toc177582711)

[Figure 4: Installation de Talend 18](#_Toc177582712)

[Figure 5: Création d'un nouveau projet 19](#_Toc177582713)

[Figure 6: Confirmation de la création du projet 19](#_Toc177582714)

[Figure 7: Parfait ! vous y êtes. 20](#_Toc177582715)

[Figure 8: Fichier pom.xml 24](#_Toc177582716)

[Figure 9: build.gradle 24](#_Toc177582717)

[Figure 10: Mvn Install 24](#_Toc177582718)

[Figure 11: Démarrage de springboot 25](#_Toc177582719)

[Figure 12: Modèle produit 27](#_Toc177582720)

[Figure 13: Site web officiel de Postman 31](#_Toc177582721)

[Figure 14: Postman pour windows 32](#_Toc177582722)

[Figure 15: Postman Windows 64 bits 32](#_Toc177582723)

[Figure 16: Parfait ! Vous y êtes avec Postman 33](#_Toc177582724)

[Figure 17: Requête GET sur la liste des produits 34](#_Toc177582725)

[Figure 18:Création d'un produit 34](#_Toc177582726)

[Figure 19: Edition d'un produit 35](#_Toc177582727)

[Figure 20:Le fichier docker-compose.yml 36](#_Toc177582728)

[Figure 21: Le fichier DockerFile 37](#_Toc177582729)

[Figure 22: Construction du build docker 37](#_Toc177582730)

[Figure 23: Lancement des services de docker-compose.yml 37](#_Toc177582731)

# Introduction

Dans un contexte technologique en constante évolution, la gestion et l'exploitation efficaces des données sont devenues des enjeux cruciaux pour les entreprises. Ce projet vise à concevoir et développer une application RESTful utilisant **Spring Boot** pour fournir une architecture backend performante et scalable. Cette application se distingue par son intégration avec **Talend Open Studio**, un puissant outil ETL (Extraction, Transformation, Chargement), permettant de faciliter l’ingestion et le traitement des données depuis une base de données **MySQL**.

En outre, afin d’assurer une surveillance proactive et continue de la santé et des performances de l’application, nous avons intégré un système de monitoring basé sur **Prometheus**, permettant de collecter et analyser en temps réel des métriques critiques. Ce choix technologique garantit à la fois une visibilité accrue sur le fonctionnement de l'application et une capacité à répondre rapidement à tout incident opérationnel.

L’objectif global de ce projet est de fournir une solution robuste et évolutive, tout en optimisant les processus de gestion des données et en assurant une surveillance continue grâce à des outils de pointe. Ce rapport documente l’ensemble des étapes suivies, des choix technologiques effectués, et des résultats obtenus tout au long du développement.

# ÉTUDE PREALABLE

## Contexte et justification

Dans le cadre de la transformation numérique, les entreprises sont confrontées à des volumes croissants de données, provenant de diverses sources et formats. La capacité à centraliser, traiter, et analyser ces données de manière efficace est devenue un facteur clé de compétitivité. Face à cette réalité, le besoin de solutions flexibles et robustes, capables de gérer à la fois l'intégration des données et la fourniture d'interfaces applicatives modernes, est devenu primordial.

C'est dans ce contexte que s'inscrit le projet de développement d'une application **RESTful** basée sur **Spring Boot**, destinée à offrir une architecture backend performante, sécurisée, et extensible. L'intégration avec **Talend Open Studio** permet d’automatiser les processus d’intégration de données depuis des bases de données relationnelles comme **MySQL**, garantissant ainsi une gestion fluide des flux de données critiques pour l’entreprise.

De plus, la montée en puissance des applications distribuées et microservices a rendu essentiel le déploiement d'outils de **monitoring** pour superviser l’état de santé et les performances des systèmes. Dans cette optique, **Prometheus** a été choisi pour son efficacité à surveiller les métriques de l'application en temps réel, permettant ainsi d'identifier rapidement les goulets d'étranglement et d'assurer une maintenance proactive.

Le projet répond donc à un double objectif : améliorer la gestion des données grâce à une intégration simplifiée et sécurisée, tout en assurant une surveillance proactive de l'application, garantissant sa disponibilité et sa performance à long terme.

## Objectifs du projet

L'objectif principal de ce projet est de développer une **application RESTful** performante et évolutive, basée sur **Spring Boot**, et d'intégrer un système de gestion des données efficace en utilisant **Talend Open Studio**. Plus précisément, le projet vise à :

1. **Créer une API RESTful robuste** avec **Spring Boot** pour permettre l’interaction fluide entre différentes parties prenantes et systèmes externes, en assurant une gestion centralisée et sécurisée des opérations.
2. **Mettre en place une base de données relationnelle** avec **MySQL**, assurant une gestion efficace et sécurisée des données applicatives, avec un schéma adapté aux besoins métiers de l'application.
3. **Intégrer un processus ETL** (Extraction, Transformation, Chargement) via **Talend Open Studio**, afin de permettre l'ingestion, la transformation et le chargement des données depuis et vers la base de données de manière automatisée et flexible.
4. **Assurer un monitoring en temps réel** de l’application à l’aide de **Prometheus**, afin de suivre les performances, détecter rapidement les anomalies, et garantir une disponibilité continue du système.
5. **Fournir une solution évolutive et scalable**, capable de répondre aux futurs besoins d’extension ou d’adaptation, tout en maintenant des standards de performance et de sécurité élevés.

En résumé, ce projet a pour ambition de fournir une solution complète, alliant efficacité dans la gestion des données, sécurité des transactions, et supervision en temps réel, pour répondre aux besoins actuels des entreprises dans un environnement numérique compétitif.

## Résultats attendus

La mise en œuvre de ce projet doit aboutir à plusieurs résultats concrets, qui marquent la réussite de l’application et la satisfaction des objectifs définis. Les résultats attendus sont les suivants :

1. **Développement d'une API RESTful fonctionnelle** : L'application doit fournir une API stable et performante, capable de gérer efficacement les requêtes (GET, POST, PUT, DELETE) des utilisateurs ou des systèmes tiers, tout en assurant une réponse rapide et sécurisée.
2. **Intégration fluide des données avec Talend Open Studio** : Le processus ETL mis en place avec Talend doit permettre une extraction, transformation et chargement automatisés des données entre la base de données MySQL et l’application, garantissant l'intégrité et la fiabilité des données.
3. **Base de données relationnelle opérationnelle** : Une base de données bien structurée et optimisée (MySQL) doit être fonctionnelle, permettant un stockage, une gestion, et une récupération rapide des données.
4. **Monitoring efficace avec Prometheus** : L’implémentation de Prometheus doit permettre une surveillance en temps réel des performances et des ressources utilisées par l’application, avec des alertes configurées pour anticiper les pannes ou dysfonctionnements.
5. **Système évolutif et sécurisé** : L'application doit être conçue pour pouvoir évoluer facilement en fonction des besoins futurs, tout en assurant des standards de sécurité élevés pour la protection des données sensibles et la gestion des accès.
6. **Documentation complète et tests concluants** : Une documentation claire doit être produite pour guider les utilisateurs et développeurs dans l’utilisation, l’intégration et le monitoring de l’application. Des tests complets doivent valider la fiabilité de l’application à chaque étape du développement.

En atteignant ces résultats, le projet fournira une solution technologique complète, répondant aux exigences d'efficacité, de sécurité, et de maintenabilité dans le cadre de la gestion et du traitement des données d’entreprise.

## Méthode d'analyse

La méthodologie adoptée pour ce projet repose sur une approche agile, permettant une flexibilité dans l'implémentation et l'amélioration continue des solutions techniques. Une première phase d’analyse des besoins fonctionnels et techniques a permis de définir les principales fonctionnalités de l’application, telles que la création d’une API RESTful, l’intégration de données via Talend Open Studio, et la mise en place d’un monitoring en temps réel avec Prometheus.

Le choix des technologies a été basé sur leur capacité à répondre aux besoins spécifiques du projet. **Spring Boot** a été retenu pour sa robustesse et sa capacité à développer rapidement des API, tandis que **Talend Open Studio** a été choisi pour son efficacité dans la gestion des processus ETL. **MySQL** ont été sélectionnés pour assurer la gestion des bases de données, et **Prometheus** pour son aptitude à surveiller et alerter sur les performances en temps réel.

## Présentation de l'équipe de travail

L'équipe de travail représente l'ensemble des acteurs impliqués dans la réalisation du projet. Elle est organisée en trois groupes : le comité de pilotage, l'équipe projet, et le groupe utilisateur.

* **Comité de pilotage** : Ce groupe a pour mission de superviser l’ensemble du projet. Il est responsable de la planification des étapes clés, de la validation des propositions soumises par l’équipe projet et de la prise de décisions stratégiques pour garantir l’atteinte des objectifs.
* **Équipe projet** : Constituée des personnes directement en charge de l'exécution du projet, cette équipe fait le lien entre le comité de pilotage et les utilisateurs. Les membres de cette équipe sont :
  + MILLOGO O Nicolas
  + TRAORE Issouf
  + OUEDRAOGO Souleymane
  + OUEDRAOGO A. Roch
* **Groupe utilisateur** : Composé de toutes les personnes physiques ou morales qui interagiront avec le système final, ce groupe inclut principalement les utilisateurs finaux chargés de la gestion des articles et des clients, notamment dans le cadre des opérations de vente.

## Présentation des technologies utilisées

Pour la réalisation de ce projet, plusieurs technologies ont été choisies en fonction de leurs capacités à répondre aux exigences de performance, de sécurité et de flexibilité. Voici un aperçu des principaux outils et technologies utilisés :

1. **Spring Boot**  
   Spring Boot est un Framework open-source basé sur Java, conçu pour faciliter le développement d’applications web robustes et scalables. Il permet de créer des services RESTful rapidement grâce à une configuration simplifiée et une gestion intégrée des dépendances.
2. **Talend Open Studio**  
   Talend Open Studio est un outil ETL (Extraction, Transformation, Chargement) qui permet de gérer les flux de données de manière flexible. Il est utilisé pour automatiser l'intégration des données depuis et vers la base de données, en garantissant la qualité des données traitées.
3. **MySQL**   
   Deux systèmes de gestion de bases de données relationnelles (SGBDR) sont envisagés pour ce projet :
   * **MySQL** : Une base de données populaire, connue pour sa rapidité et son efficacité, adaptée aux applications web de grande envergure.
4. **Prometheus**  
   Prometheus est un système de monitoring et d'alerte open-source, utilisé pour superviser les performances de l'application en temps réel. Il collecte les métriques des différents services et émet des alertes en cas de dépassement de seuils critiques, permettant ainsi une gestion proactive des incidents.
5. **Postman**  
   Postman est un outil populaire pour tester et documenter les API. Il a été utilisé pour valider le bon fonctionnement des endpoints RESTful, en simulant des requêtes HTTP et en vérifiant les réponses retournées par l’application.
6. **IntelliJ IDEA**  
   L’environnement de développement intégré (IDE) IntelliJ IDEA a été utilisé pour l’écriture du code Java et la gestion du projet Spring Boot. Cet outil offre de puissantes fonctionnalités de débogage, d’édition, et d’intégration continue, facilitant ainsi le développement et la maintenance de l’application.
7. **Docker**  
   Docker a été utilisé pour créer des conteneurs permettant de déployer l'application et ses dépendances de manière isolée et reproductible. Il assure une portabilité entre les environnements de développement, de test et de production, tout en optimisant les ressources.

# 



# CONCEPTION ET ARCHITECTURE DU SYSTÈME



## Technologies de l'architecture

L'architecture du projet repose sur une combinaison de technologies modernes, permettant de créer une application modulaire, évolutive et performante. Ces technologies ont été choisies en fonction de leur capacité à répondre aux besoins du projet, tout en assurant une interopérabilité entre les différents composants. Les principales technologies architecturales utilisées sont :

1. **Architecture REST ful avec SpringBoot**  
   L'architecture de l'application est basée sur le style d'architecture REST (Representational State Transfer), qui permet de structurer et d'organiser les échanges de données entre le serveur et les clients. Spring Boot est utilisé pour implémenter cette architecture RESTful, offrant un cadre léger pour la création d'API et une gestion efficace des requêtes HTTP (GET, POST, PUT, DELETE).
2. **Micro services et conteneurisation avec Docker**  
   Afin de garantir la scalabilité et la portabilité de l'application, chaque composant est conteneurisé avec **Docker**. Cette approche permet de déployer les différents services de manière indépendante, tout en assurant un environnement uniforme à travers les phases de développement, de test et de production. Chaque conteneur est isolé, ce qui facilite la gestion des dépendances et améliore la stabilité du système.
3. **Gestion des données avec MySQL**   
   La gestion des données est assurée par des bases de données relationnelles **MySQL**. Ces systèmes de gestion de bases de données offrent une capacité de gestion des transactions, une intégrité des données, et des performances optimisées pour les opérations de lecture et d'écriture.
4. **ETL avec Talend Open Studio**  
   Talend Open Studio est utilisé pour la gestion des processus ETL (Extraction, Transformation, Chargement), assurant une intégration fluide des données entre la base de données et les autres composants de l'application. Il automatise le flux de données en respectant les normes de qualité et de sécurité, garantissant ainsi des échanges de données fiables et continus.
5. **Surveillance et gestion des performances avec Prometheus**  
   Pour assurer un suivi en temps réel des performances de l’application, **Prometheus** est intégré à l'architecture. Il collecte des métriques depuis les différents services, permettant de détecter rapidement les anomalies ou les baisses de performance. Des alertes sont configurées pour permettre une réponse proactive en cas de problème.

## Architecture du système

L'architecture du système repose sur une approche modulaire et orientée services, permettant une gestion claire des responsabilités et une facilité d’évolution. L'application suit les principes d'une architecture RESTful et est constituée de plusieurs couches interconnectées, chacune ayant un rôle bien défini. Les principaux composants de cette architecture sont décrits ci-dessous :

1. **Couche de présentation (Frontend)**  
   Bien que le projet se concentre principalement sur la création d'une API RESTful, la couche de présentation peut être implémentée avec n'importe quelle interface utilisateur (UI), telle qu'une application web ou mobile. Cette couche interagit avec l'API pour récupérer les données et afficher les informations aux utilisateurs finaux. Elle consomme les services exposés par l'API en envoyant des requêtes HTTP et en recevant des réponses sous forme de JSON.
2. **Couche API (Backend avec Spring Boot)**  
   La couche API est le cœur de l'application, responsable de la gestion des requêtes et des réponses. Elle est construite avec **Spring Boot**, qui permet de créer rapidement des endpoints RESTful pour effectuer les opérations CRUD (Create, Read, Update, Delete) sur les articles et les clients. Cette couche comprend également des services de gestion des utilisateurs, des ventes, et des articles, tout en assurant la sécurisation des échanges via des mécanismes d'authentification et d'autorisation.
3. **Couche de persistance (Base de données)**  
   La gestion des données est assurée par une base de données relationnelle, **MySQL**, en fonction des besoins. Ces bases de données stockent toutes les informations relatives aux articles, clients, ventes, et utilisateurs. La couche de persistance est connectée à l'API grâce à **JPA/Hibernate**, ce qui permet de gérer efficacement les entités, les transactions et les relations entre les différentes tables de la base de données.
4. **Couche d'intégration de données (ETL avec Talend Open Studio)**  
   Pour assurer l'intégration et la transformation des données provenant de diverses sources, **Talend Open Studio** est utilisé comme plateforme ETL (Extraction, Transformation, Chargement). Cette couche permet de traiter les données avant leur insertion dans la base de données ou avant leur consommation par l'API, garantissant ainsi la qualité et la cohérence des données.
5. **Couche de monitoring (Prometheus)**  
   **Prometheus** est intégré dans l'architecture pour surveiller les performances de l'application et collecter des métriques. Cette couche permet de suivre l'état des différents services, la consommation des ressources, et la charge des serveurs en temps réel. En cas d'anomalie ou de seuil critique dépassé, des alertes sont envoyées, permettant aux administrateurs d'intervenir rapidement.
6. **Couche de sécurité**  
   La sécurité des données et des transactions est une priorité dans l’architecture du système. Des mécanismes tels que l’authentification par jetons JWT (JSON Web Token), le chiffrement des données sensibles et l’utilisation de certificats SSL/TLS pour les communications sécurisées sont mis en place. Cette couche s’assure que seules les personnes ou entités autorisées peuvent accéder aux ressources ou effectuer des actions sensibles.
7. **Conteneurisation et déploiement (Docker)**  
   Chaque composant de l'architecture est conteneurisé avec **Docker** pour assurer une portabilité et une isolation maximale entre les environnements de développement, de test, et de production. Cela permet également de déployer les services de manière indépendante, facilitant ainsi la gestion et la mise à jour de l'application.

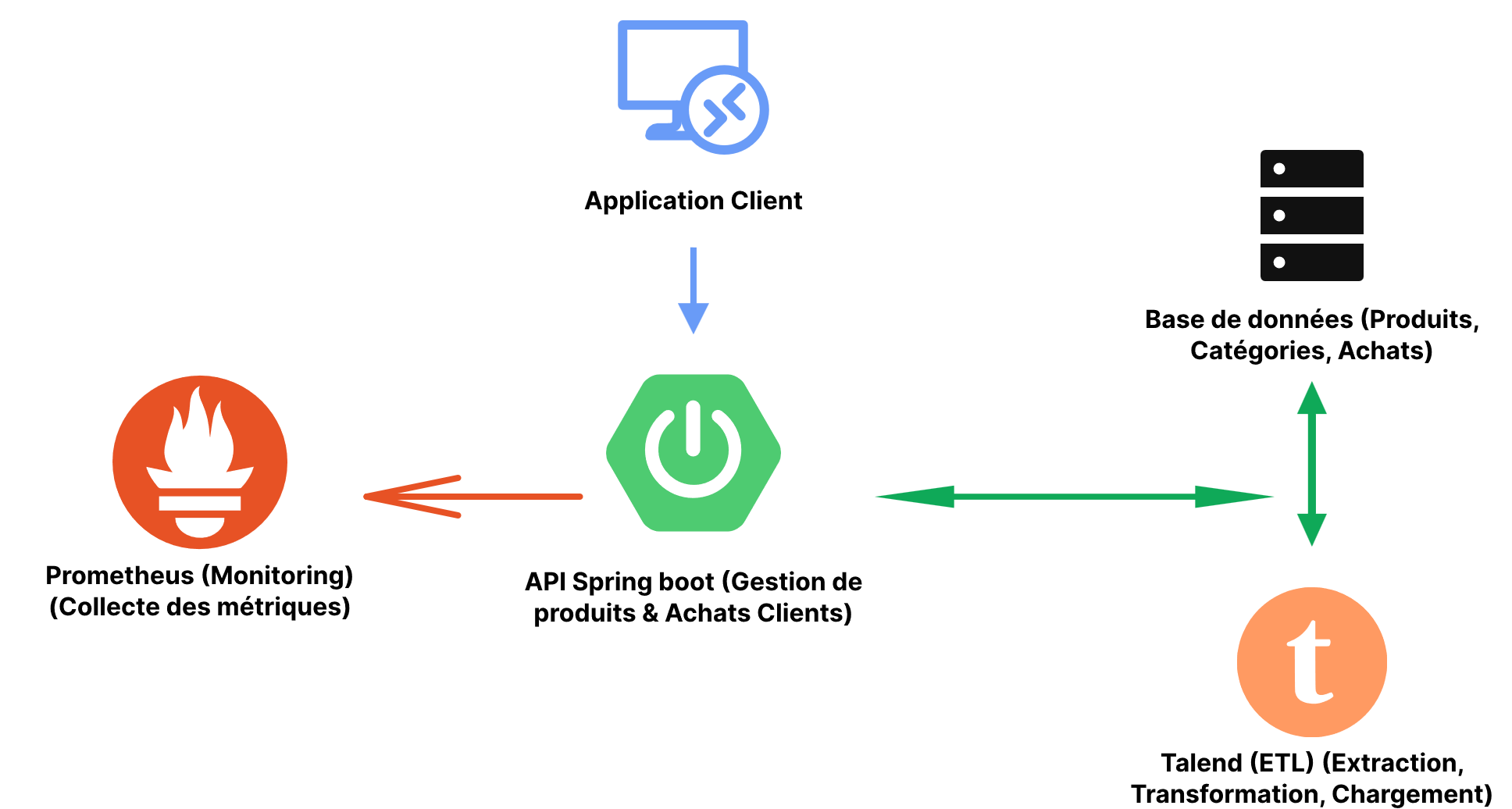


Figure 1:Architecture du système

# CONFIGURATION ET PROCESSUS ETL AVEC TALEND OPEN STUDIO



## Présentation de Talend Open Studio et de son utilisation

**Talend Open Studio** est une plateforme de gestion des données open-source qui facilite les processus d'Extraction, Transformation et Chargement (ETL). Conçu pour simplifier le traitement des données, Talend Open Studio offre un environnement graphique intuitif pour concevoir, exécuter et surveiller des flux de données complexes.

1. **Présentation de Talend Open Studio**

Talend Open Studio permet aux utilisateurs de créer des processus ETL en utilisant une interface de type glisser-déposer, ce qui simplifie la conception et l'exécution des workflows de données. L'outil supporte une large gamme de sources et de cibles de données, incluant des bases de données, des fichiers, des systèmes de messagerie et des API web. Il est particulièrement apprécié pour sa flexibilité et sa capacité à gérer des volumes importants de données tout en garantissant leur qualité.

1. **Utilisation de Talend Open Studio dans le projet**

Dans le cadre de ce projet, Talend Open Studio est utilisé pour automatiser les tâches d'intégration de données entre la base de données et les services RESTful. Voici les principales étapes de son utilisation :

1. **Conception des workflows ETL**  
   Les utilisateurs conçoivent des workflows pour extraire les données des sources (telles que des bases de données ou des fichiers), les transformer en fonction des besoins (comme la validation, le nettoyage, ou la transformation de format), et les charger dans les systèmes cibles (comme la base de données finale utilisée par l'application).
2. **Automatisation des processus**  
   Talend Open Studio permet de planifier et d'automatiser l'exécution des processus ETL. Cela assure une mise à jour régulière des données et une intégration fluide sans intervention manuelle, garantissant que les données dans l'application sont toujours à jour et cohérentes.
3. **Surveillance et gestion des erreurs**  
   L'outil offre des fonctionnalités de surveillance pour suivre l'état des processus ETL en temps réel. En cas d'erreur ou d'anomalie, des alertes peuvent être configurées pour permettre une intervention rapide et minimiser les interruptions de service.
4. **Documentation et maintenance**  
   Talend Open Studio facilite la documentation des processus ETL et offre des outils pour la maintenance continue. Les workflows peuvent être facilement modifiés et mis à jour pour s'adapter aux évolutions des besoins ou aux nouvelles sources de données.

En utilisant Talend Open Studio, le projet bénéficie d'une intégration efficace des données, d'une automatisation des flux de travail et d'une gestion simplifiée des processus ETL, tout en assurant la qualité et la cohérence des données utilisées par l'application.

## Installation et configuration de Talend Open Studio

**Talend Open Studio** est un outil de gestion des données qui nécessite une installation et une configuration adéquates pour être pleinement opérationnel. Cette section décrit les étapes nécessaires pour installer et configurer Talend Open Studio dans le cadre de votre projet.

**1. Installation de Talend Open Studio**

1. **Téléchargement de l’outil**
   * Accédez au site officiel de Talend (https://www.talend.com/download/) et téléchargez la version gratuite de Talend Open Studio. Choisissez la version appropriée en fonction de votre système d'exploitation (Windows, macOS, Linux).
2. **Décompression du fichier téléchargé**
   * Une fois le fichier téléchargé, décompressez l'archive ZIP dans un répertoire de votre choix. Le fichier ZIP contient l'ensemble des fichiers nécessaires pour l'exécution de Talend Open Studio.
3. **Exécution de l’application**
   * Naviguez vers le répertoire décompressé et lancez l'application en exécutant le fichier Talend-Studio-win-x.y.z.exe (pour Windows) ou Talend-Studio-linux-x.y.z.sh (pour Linux). Suivez les instructions à l'écran pour démarrer l'application.
4. **2. Configuration initiale**
5. **Création d’un espace de travail**
   * Lors du premier lancement, Talend Open Studio vous invite à sélectionner un espace de travail. Cet espace de travail est l'emplacement où tous vos projets et configurations seront stockés. Choisissez un répertoire approprié et confirmez.
6. **Configuration des préférences**
   * Après avoir créé un espace de travail, configurez les préférences de l'application selon vos besoins. Allez dans Window > Preferences pour accéder aux options de configuration. Vous pouvez définir des paramètres tels que les préférences de connexion, les paramètres de proxy, et les préférences de l'éditeur.
7. **3. Connexion à la base de données**
8. **Création de la connexion**
   * Ouvrez le Repository dans l'interface de Talend et faites un clic droit sur Metadata > Db Connections > Create connection. Suivez l’assistant pour configurer la connexion à votre base de données (MySQL). Fournissez les informations nécessaires telles que l'URL de la base de données, le nom d'utilisateur, et le mot de passe.
9. **Test de la connexion**
   * Après avoir saisi les informations de connexion, cliquez sur le bouton Test connection pour vérifier que Talend Open Studio peut se connecter à la base de données. Assurez-vous que le test réussit avant de procéder à l'étape suivante.
10. **Importation des métadonnées**
    * Une fois la connexion établie, importez les métadonnées de la base de données dans votre espace de travail. Cela inclut les tables, les vues et les schémas nécessaires pour la conception de vos processus ETL.

**4. Configuration des processus ETL**

1. **Création d’un nouveau projet**
   * Dans l’interface de Talend, créez un nouveau projet en allant dans File > New Project. Donnez un nom à votre projet et configurez les paramètres de base.
2. **Conception des workflows ETL**
   * Utilisez l’éditeur graphique de Talend pour concevoir vos processus ETL. Faites glisser et déposez les composants nécessaires pour l’extraction, la transformation et le chargement des données. Configurez chaque composant en fonction des besoins spécifiques de votre projet.
3. **Définition des tâches planifiées**
   * Configurez les tâches ETL pour qu’elles s’exécutent automatiquement à intervalles réguliers en utilisant le planificateur intégré de Talend. Cela permet de garantir une mise à jour continue des données sans intervention manuelle.

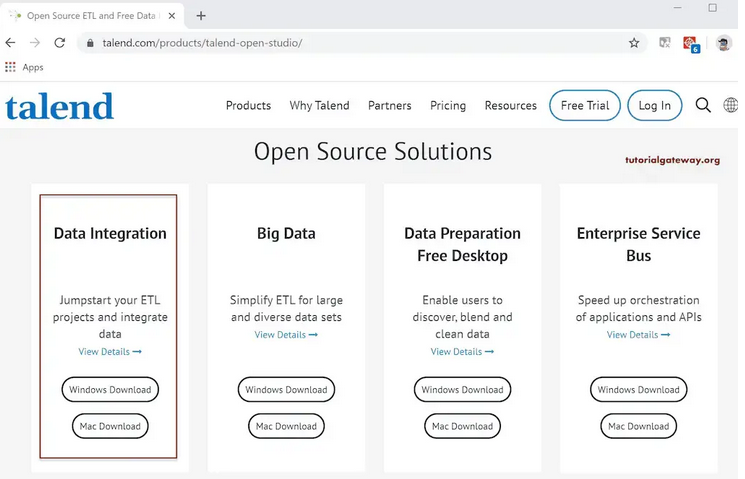


Figure 2: Page de téléchargement de talend

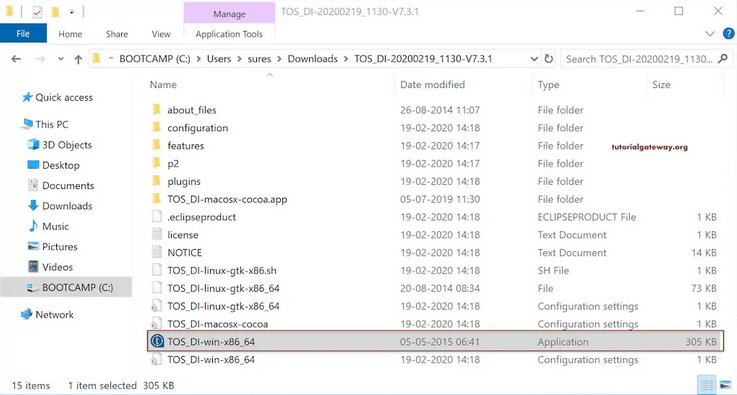


Figure 3: Après installation

Une image contenant texte, capture d’écran, Police, Page web

Description générée automatiquementFigure 4: Installation de Talend

Une image contenant texte, Police, logiciel, Page web

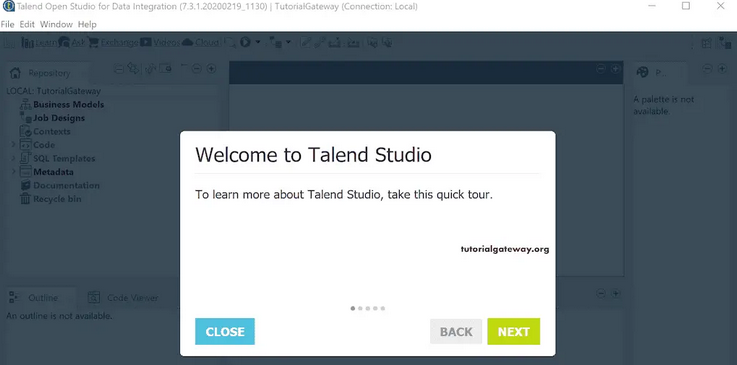
Description générée automatiquement

Figure 5: Création d'un nouveau projet

Une image contenant texte, capture d’écran, affichage, logiciel

Description générée automatiquement

Figure 6: Confirmation de la création du projet

Figure 7: Parfait ! vous y êtes.

# DÉVELOPPEMENT DE L'API RESTFUL AVEC SPRING BOOT



## Présentation de Spring Boot et de l'API RESTful

**Spring Boot** est un Framework open-source basé sur le Framework Spring, conçu pour simplifier le développement d'applications Java en offrant une configuration minimale et une mise en route rapide. Il se distingue par sa capacité à créer des applications autonomes et prêtes à l'emploi, en intégrant par défaut des configurations courantes et des dépendances essentielles.

1. **Caractéristiques principales de Spring Boot :**
   * **Configuration automatique** : Spring Boot fournit des configurations prédéfinies pour une variété de scénarios d'application, ce qui réduit le besoin de configuration manuelle.
   * **Démarrage rapide** : Grâce à ses "starters" (démarreurs), il est possible d'ajouter facilement des dépendances et des configurations spécifiques à un projet en un minimum de temps.
   * **Applications autonomes** : Les applications Spring Boot peuvent être exécutées directement à partir d'un JAR exécutable, sans avoir besoin d'un serveur d'application externe.
   * **Intégration des microservices** : Spring Boot facilite la création de microservices en intégrant des composants tels que Spring Cloud, qui permettent de gérer la configuration, la découverte de services, et plus encore.
2. **Avantages pour le projet :**
   * **Simplicité et rapidité de développement** : La configuration simplifiée et l’automatisation des tâches permettent de se concentrer sur le développement de la logique métier sans se soucier des détails de configuration.
   * **Robustesse et scalabilité** : La capacité à créer des services évolutifs et robustes répond aux exigences des applications modernes nécessitant une haute disponibilité et des performances optimales.

**Présentation de l'API RESTful**

Une API RESTful (Representational State Transfer) est un style d'architecture pour la conception de services web qui utilisent les protocoles HTTP pour la communication entre les clients et les serveurs. Les services RESTful sont conçus pour être simples, stateless, et exploitables sur Internet.

1. **Principes fondamentaux de REST :**
   * **Stateless** : Chaque requête HTTP du client au serveur doit contenir toutes les informations nécessaires pour comprendre et traiter la requête. Le serveur ne conserve aucune information sur l'état du client entre les requêtes.
   * **Méthodes HTTP** : Les API RESTful utilisent les méthodes HTTP standard pour effectuer des opérations sur les ressources :
     + **GET** : Récupération des ressources.
     + **POST** : Création de nouvelles ressources.
     + **PUT** : Mise à jour des ressources existantes.
     + **DELETE** : Suppression des ressources.
   * **Ressources** : Les données sont représentées sous forme de ressources accessibles via des URL uniques. Les ressources sont généralement retournées au format JSON ou XML.
2. **Avantages pour le projet :**
   * **Simplicité d’utilisation** : Les API RESTful sont faciles à comprendre et à utiliser grâce à leur structure claire et leurs méthodes HTTP bien définies.
   * **Interopérabilité** : REST est indépendant de la plateforme et du langage de programmation, ce qui permet une intégration facile avec différents types de clients (applications web, mobiles, etc.).
   * **Scalabilité** : L'architecture stateless et l'utilisation de méthodes HTTP standards facilitent la scalabilité horizontale, permettant de gérer une augmentation du nombre de requêtes sans perte de performance.
3. **Implémentation avec Spring Boot :**
   * **Création d'API RESTful** : Avec Spring Boot, la création d'API RESTful est simplifiée grâce à des annotations telles que @RestController, @RequestMapping, @GetMapping, @PostMapping, etc. Ces annotations permettent de définir les endpoints de l'API, de gérer les requêtes HTTP et de sérialiser les réponses en JSON ou XML.
   * **Gestion des erreurs** : Spring Boot facilite la gestion des erreurs avec des mécanismes tels que @ExceptionHandler pour traiter les exceptions et retourner des réponses d'erreur appropriées aux clients.

## Création de l'environnement et installation des dépendances

Pour développer une application avec Spring Boot, il est essentiel de configurer correctement l'environnement de développement et d'installer les dépendances nécessaires. Cette étape garantit que l'application fonctionne de manière fluide et que toutes les fonctionnalités requises sont disponibles.

**1. Création de l'Environnement de Développement**

**Installation de Java Development Kit (JDK) :**

* + Assurez-vous que le JDK est installé sur votre machine. Spring Boot nécessite Java 8 ou une version ultérieure.
  + Téléchargez la dernière version du JDK depuis le site officiel d'Oracle ou utilisez un JDK open source comme OpenJDK.
  + Configurez les variables d'environnement JAVA\_HOME pour pointer vers le répertoire d'installation du JDK.

**Installation d'un IDE :**

* + Utilisez un environnement de développement intégré (IDE) compatible avec Spring Boot, tel que **IntelliJ IDEA**, **Eclipse**, ou **VS Code** avec le plugin Java.
  + Configurez l'IDE pour travailler avec le projet Spring Boot en installant les plugins nécessaires, tels que le support Maven ou Gradle.

**Initialisation du Projet Spring Boot**

**Création du Projet :**

* + Utilisez **Spring Initializr** (<https://start.spring.io>) pour générer un projet Spring Boot. Sélectionnez les paramètres appropriés :
    - **Project** : Maven ou Gradle
    - **Language** : Java
    - **Spring Boot Version** : Choisissez la version stable recommandée.
    - **Dependencies** : Ajoutez les dépendances nécessaires telles que **Spring Web**, **Spring Data JPA**, **MySQL Driver**, **Spring Boot DevTools**, et **Lombok**.

1. **Téléchargement et Extraction :**
   * Téléchargez l'archive ZIP générée par Spring Initializr.
   * Décompressez le fichier ZIP et importez le projet dans votre IDE.

**3. Installation des Dépendances**

**Utilisation de Maven :**

* + Le fichier pom.xml contient les dépendances nécessaires pour le projet. Voici un exemple de dépendances essentielles :

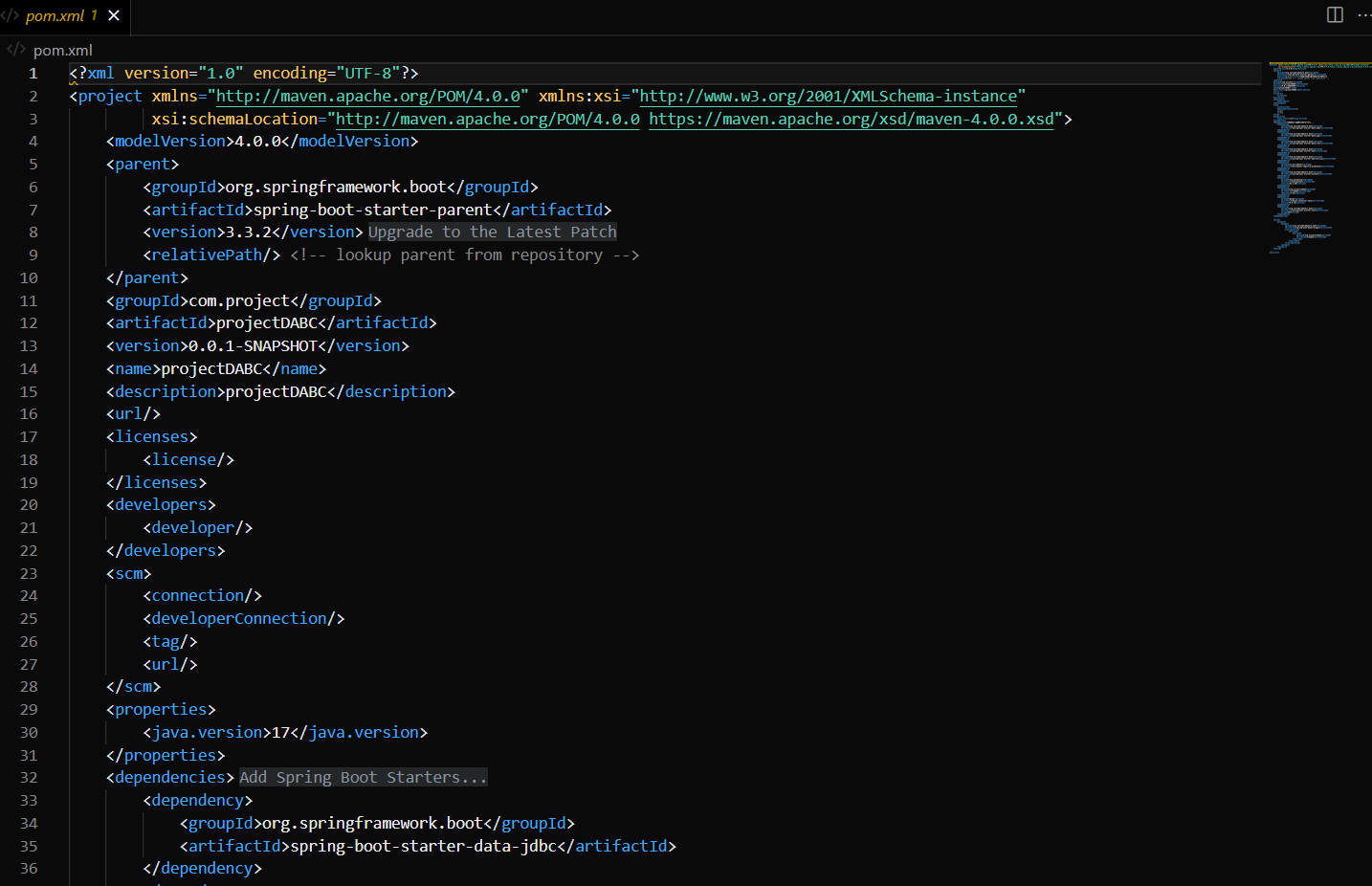


Figure 8: Fichier pom.xml

**Utilisation de Gradle :**

* Le fichier build.gradle contient les dépendances nécessaires pour le projet. Voici un exemple de configuration :

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Figure 9: build.gradle

Mvn install :



Figure 10: Mvn Install

Démarrage :

Une image contenant texte, capture d’écran, logiciel, Police

Description générée automatiquement

Figure 11: Démarrage de springboot

## Structure de l’application et organisation des routes

Pour garantir une organisation claire et efficace du projet, la structure de l'application suit les conventions de Spring Boot. La structure du projet est divisée en trois principaux répertoires : src/main, src/test, et src/main/resources. Cette section décrit la disposition des packages et les routes de l'API RESTful.

**1. Structure du Projet**

La structure du projet est organisée comme suit :

* **src/main/java** : Contient le code source principal de l'application.
  + **com.example.project** : Le package racine du projet. Remplacez example et project par les noms appropriés pour votre projet.
    - **controller** : Contient les classes qui gèrent les requêtes HTTP. Les contrôleurs définissent les endpoints de l'API et coordonnent les interactions entre les services et les clients.
      * Exemple : ArticleController.java
    - **service** : Contient les classes pour la logique métier de l'application. Les services manipulent les données et appliquent les règles métier.
      * Exemple : ArticleService.java
    - **repository** : Contient les interfaces pour l'accès aux données. Les repositories étendent Spring Data JPA pour fournir des opérations CRUD.
      * Exemple : ArticleRepository.java
    - **model** : Contient les classes représentant les entités de la base de données. Ces modèles sont mappés aux tables de la base de données via JPA.
      * Exemple : Article.java
    - **config** : Contient les classes de configuration pour les paramètres spécifiques de l'application, comme la configuration de la base de données et les aspects de sécurité.
      * Exemple : WebConfig.java
    - **exception** : Contient les classes définissant les exceptions personnalisées et les gestionnaires d'exceptions pour l'API.
      * Exemple : CustomExceptionHandler.java
* **src/test/java** : Contient les classes de test pour vérifier le bon fonctionnement des différentes parties de l'application. Les tests doivent couvrir les contrôleurs, services, et autres composants de l'application.
  + Exemple : ArticleControllerTests.java
* **src/main/resources** : Contient les fichiers de configuration et les ressources de l'application.
  + **application.properties** : Fichier de configuration pour les paramètres de l'application, comme la connexion à la base de données.

**2. Organisation des Routes**

Les routes de l'API RESTful sont organisées pour gérer les opérations CRUD sur les articles. Voici une vue d'ensemble des principales routes exposées par l'application :

* **/api/articles** : Point d'entrée principal pour les opérations liées aux articles.
  + **GET /api/articles** : Récupère la liste de tous les articles.
  + **GET /api/articles/{id}** : Récupère un article spécifique par son ID.
  + **POST /api/articles** : Crée un nouvel article. Le corps de la requête doit contenir les détails de l'article à créer.
  + **PUT /api/articles/{id}** : Met à jour les informations d'un article existant par son ID. Le corps de la requête doit contenir les nouvelles informations.
  + **DELETE /api/articles/{id}** : Supprime un article spécifique par son ID.

**3. Exemples de Code**

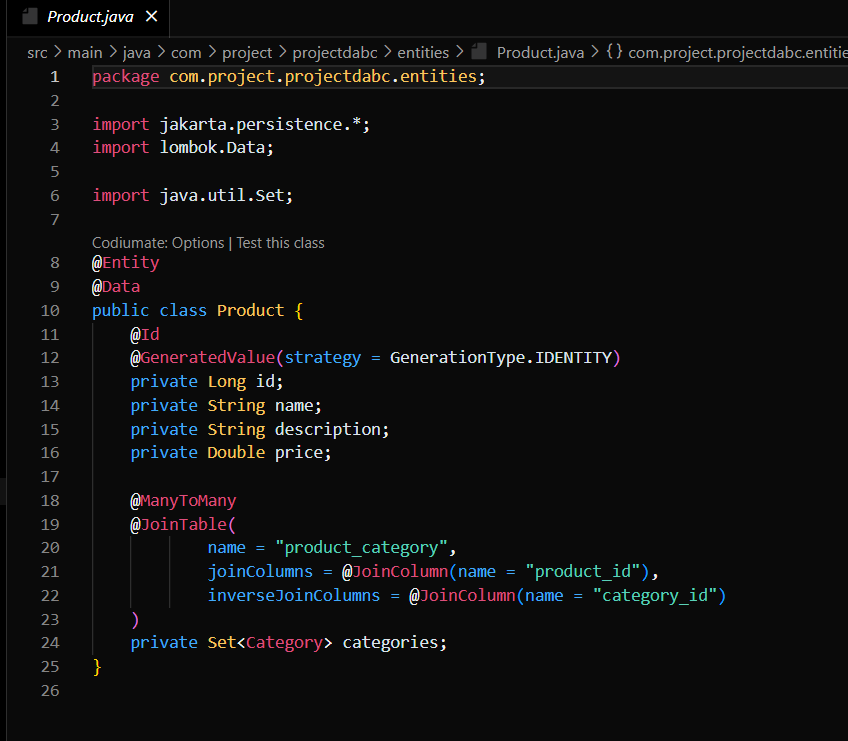
* **Model de produit :**
* 

Figure 12: Modèle produit

## Endpoints de l'API

Cette section décrit les endpoints exposés par l'API RESTful pour chaque modèle de l'application : Category, Customer, Order, et Product. Chaque modèle possède des endpoints pour effectuer les opérations CRUD (Créer, Lire, Mettre à jour, Supprimer). Les URLs sont conçues pour permettre une interaction facile avec les ressources de l'application.

**1. Modèle : Category**

* **GET /api/categories** : Récupère la liste de toutes les catégories.
* **GET /api/categories/{id}** : Récupère une catégorie spécifique par son ID.
* **POST /api/categories** : Crée une nouvelle catégorie. Le corps de la requête doit contenir les détails de la catégorie.
* **PUT /api/categories/{id}** : Met à jour les informations d'une catégorie existante par son ID. Le corps de la requête doit contenir les nouvelles informations.
* **DELETE /api/categories/{id}** : Supprime une catégorie spécifique par son ID.

**2. Modèle : Customer**

* **GET /api/customers** : Récupère la liste de tous les clients.
* **GET /api/customers/{id}** : Récupère un client spécifique par son ID.
* **POST /api/customers** : Crée un nouveau client. Le corps de la requête doit contenir les détails du client.
* **PUT /api/customers/{id}** : Met à jour les informations d'un client existant par son ID. Le corps de la requête doit contenir les nouvelles informations.
* **DELETE /api/customers/{id}** : Supprime un client spécifique par son ID.

**3. Modèle : Order**

* **GET /api/orders** : Récupère la liste de toutes les commandes.
* **GET /api/orders/{id}** : Récupère une commande spécifique par son ID.
* **POST /api/orders** : Crée une nouvelle commande. Le corps de la requête doit contenir les détails de la commande.
* **PUT /api/orders/{id}** : Met à jour les informations d'une commande existante par son ID. Le corps de la requête doit contenir les nouvelles informations.
* **DELETE /api/orders/{id}** : Supprime une commande spécifique par son ID.

**4.Modèle : Product**

* **GET /api/products** : Récupère la liste de tous les produits.
* **GET /api/products/{id}** : Récupère un produit spécifique par son ID.
* **POST /api/products** : Crée un nouveau produit. Le corps de la requête doit contenir les détails du produit.
* **PUT /api/products/{id}** : Met à jour les informations d'un produit existant par son ID. Le corps de la requête doit contenir les nouvelles informations.
* **DELETE /api/products/{id}** : Supprime un produit spécifique par son ID.

# TESTS ET DOCUMENTATION

Procédure d'installation de Postman

**Postman** est un outil populaire utilisé pour le test des API RESTful. Il permet aux développeurs de créer, exécuter, et automatiser des tests d'API de manière efficace, en offrant une interface utilisateur conviviale pour la gestion des requêtes HTTP et l'analyse des réponses.

**1. Téléchargement de Postman**

**Accéder au site officiel**

Rendez-vous sur le site web de Postman à l'adresse https://www.postman.com/downloads/.

**Choisir la version appropriée**

Sélectionnez la version correspondant à votre système d'exploitation (Windows, macOS, Linux). Postman propose également une version pour les systèmes basés sur les architectures ARM.

**Télécharger le fichier d'installation**

Cliquez sur le bouton de téléchargement pour obtenir le fichier d'installation de Postman. Le fichier sera téléchargé en format .exe pour Windows, .dmg pour macOS, ou .tar.gz pour Linux.

**2. Installation de Postman**

**Installation sur Windows**

**Exécuter le fichier** : Double-cliquez sur le fichier .exe téléchargé pour lancer l'installateur.

**Suivre les instructions** : Suivez les instructions à l'écran pour compléter l'installation. Vous pouvez choisir l'emplacement d'installation par défaut ou spécifier un autre répertoire.

**Lancer Postman** : Une fois l'installation terminée, vous pouvez lancer Postman depuis le menu Démarrer ou le raccourci créé sur le bureau.

**Installation sur macOS**

**Ouvrir le fichier** : Double-cliquez sur le fichier .dmg téléchargé pour monter l'image disque.

**Glisser-déposer l'application** : Faites glisser l'icône de Postman vers le dossier Applications pour l'installer.

**Lancer Postman** : Ouvrez le dossier Applications et double-cliquez sur l'icône de Postman pour démarrer l'application.

**Installation sur Linux**

**Extraire l'archive** : Décompressez le fichier .tar.gz téléchargé dans un répertoire de votre choix.

**Exécuter l'application** : Accédez au répertoire décompressé et exécutez le script postman pour lancer l'application. Vous pouvez également créer un raccourci pour faciliter l'accès à l'application.

**3. Configuration initiale**

**Création d'un compte**

Lors du premier lancement de Postman, vous êtes invité à créer un compte ou à vous connecter si vous en possédez déjà un. La création d'un compte vous permet de synchroniser vos collections et configurations à travers plusieurs appareils.

**Configuration des préférences**

Accédez aux paramètres en cliquant sur l’icône de roue dentée dans le coin supérieur droit. Configurez les préférences selon vos besoins, telles que les paramètres de proxy, les thèmes de l'interface, et les options de synchronisation.

**Importation des collections et environnement**

Vous pouvez importer des collections et des environnements existants en utilisant l'option d'importation dans l'interface de Postman. Cela vous permet de charger des configurations prédéfinies pour vos tests API.

Une image contenant texte, Page web, logiciel, Site web

Description générée automatiquement

Figure 13: Site web officiel de Postman

Une image contenant texte, logiciel, Page web, Site web

Description générée automatiquement

Figure 14: Postman pour windows

Une image contenant texte, Police, Page web, Site web

Description générée automatiquement

Figure 15: Postman Windows 64 bits

Une image contenant texte, capture d’écran, logiciel, Page web

Description générée automatiquement

Figure 16: Parfait ! Vous y êtes avec Postman

1. Scénarios de tests définis

Les scénarios de tests sont essentiels pour valider le bon fonctionnement des fonctionnalités de l'API RESTful développée dans le cadre de ce projet. Ils permettent de s'assurer que les opérations de création, de récupération, de mise à jour et de suppression des articles sont exécutées correctement et que les réponses du serveur sont conformes aux attentes. Cette section décrit les différents tests effectués à l'aide de Postman, en détaillant les configurations des requêtes, les résultats attendus, et les étapes pour capturer les réponses. Les tests incluent également la documentation des jobs ETL utilisés pour l'intégration des données, garantissant ainsi que le système fonctionne de manière fluide et cohérente.

* 1. Requêtes GET

Une image contenant texte, capture d’écran, nombre, logiciel

Description générée automatiquement

Figure 17: Requête GET sur la liste des produits

* 1. Requêtes POST

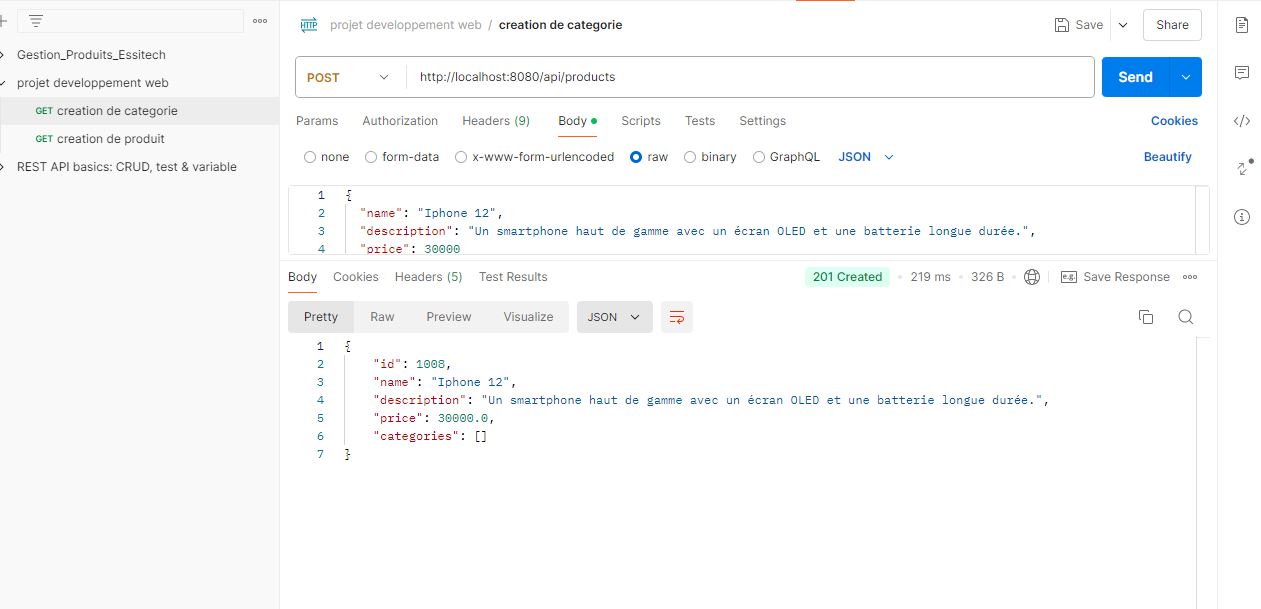


Figure 18:Création d'un produit

* 1. Requêtes PUT

Une image contenant texte, capture d’écran, logiciel, nombre

Description générée automatiquement

Figure 19: Edition d'un produit

# DÉPLOIEMENT ET CONCLUSION

## Déploiement de l'application

Le déploiement de l'application a été réalisé en utilisant Docker et Docker Compose pour assurer une configuration cohérente et une gestion simplifiée des conteneurs. Cette approche permet de créer un environnement isolé et reproductible pour l'exécution de l'application, incluant tous ses composants nécessaires, tels que le backend Spring Boot, la base de données, et les outils de monitoring.

1. Docker Compose

**Docker Compose** est un outil permettant de définir et de gérer des applications multi-conteneurs. Il utilise un fichier de configuration docker-compose.yml pour orchestrer le déploiement et la coordination des différents services de l'application.

* **Emplacement du fichier** : Le fichier docker-compose.yml se trouve à la racine du projet. Ce fichier définit les services nécessaires, les volumes, et les réseaux pour l'application.
* **Contenu typique** :

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

Figure 20:Le fichier docker-compose.yml

**2. Dockerfile**

Le **Dockerfile** est un script utilisé par Docker pour construire une image Docker de l'application. Il spécifie les instructions pour créer l'image, incluant l'installation des dépendances et la configuration de l'environnement d'exécution.

* **Emplacement du fichier** : Le Dockerfile se trouve également à la racine du projet.
* **Contenu typique** :

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

Figure 21: Le fichier DockerFile

**Instructions de Déploiement**

1. **Préparation des Images :**
   * Placez-vous dans le répertoire racine du projet contenant les fichiers docker-compose.yml et Dockerfile.
   * Construisez les images Docker à l'aide de Docker Compose :



Figure 22: Construction du build docker

1. **Démarrage des Services :**

* Lancez les services définis dans le fichier docker-compose.yml :

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Figure 23: Lancement des services de docker-compose.yml

1. **Vérification :**

* Accédez à l'application via http://localhost:8080 pour vérifier son bon fonctionnement.
* Utilisez l'interface de Prometheus à http://localhost:9090 pour surveiller les métriques et assurer la bonne santé de l'application.

# Conclusion

Ce rapport a détaillé la conception, le développement, et le déploiement d'une application RESTful intégrée à une base de données MySQL, avec un système de monitoring via Prometheus. En utilisant des outils modernes tels que Spring Boot, Talend Open Studio, et Docker, nous avons créé une solution robuste et adaptable pour la gestion des articles et des clients dans le cadre de ventes.

Le processus de développement a été structuré en plusieurs étapes clés, de l'analyse des besoins et de la conception de l'architecture à la mise en place des scénarios de tests et du déploiement. L'utilisation de Docker Compose a facilité la gestion des conteneurs, assurant ainsi une intégration fluide des différents composants de l'application. Les tests réalisés ont validé les fonctionnalités essentielles de l'API, garantissant leur performance et leur fiabilité dans des environnements de production.

En conclusion, ce projet illustre non seulement l'efficacité des technologies employées mais aussi l'importance d'une approche méthodique pour assurer le succès d'une application complexe. La mise en œuvre de solutions de monitoring et l'utilisation de pratiques de déploiement modernes témoignent de notre engagement envers la qualité et la performance. Cette expérience offre une base solide pour de futurs projets, en mettant en évidence les meilleures pratiques et les outils indispensables pour le développement d'applications d'envergure.

# Table des matières

[Sommaire 2](#_Toc177508205)

[Table des illustrations 4](#_Toc177508206)

[Introduction 5](#_Toc177508207)

[1 ÉTUDE PREALABLE 5](#_Toc177508208)

[7.1 Contexte et justification 5](#_Toc177508209)

[7.2 Objectifs du projet 6](#_Toc177508210)

[7.3 Résultats attendus 7](#_Toc177508211)

[7.4 Méthode d'analyse 8](#_Toc177508212)

[7.5 Présentation de l'équipe de travail 8](#_Toc177508213)

[7.6 Présentation des technologies utilisées 9](#_Toc177508214)

[2 CONCEPTION ET ARCHITECTURE DU SYSTÈME 10](#_Toc177508222)

[1.1 Technologies de l'architecture 10](#_Toc177508224)

[1.2 Architecture du système 11](#_Toc177508225)

[3 CONFIGURATION ET PROCESSUS ETL AVEC TALEND OPEN STUDIO 13](#_Toc177508226)

[7.1 Présentation de Talend Open Studio et de son utilisation 13](#_Toc177508231)

[7.1 Installation et configuration de Talend Open Studio 15](#_Toc177508232)

[4 DÉVELOPPEMENT DE L'API RESTFUL AVEC SPRING BOOT 20](#_Toc177508233)

[7.1 Présentation de Spring Boot et de l'API RESTful 20](#_Toc177508235)

[7.2 Création de l'environnement et installation des dépendances 22](#_Toc177508236)

[7.3 Structure de l’application et organisation des routes 25](#_Toc177508237)

[7.4 Endpoints de l'API 28](#_Toc177508238)

[5 TESTS ET DOCUMENTATION 29](#_Toc177508239)

[Procédure d'installation de Postman 29](#_Toc177508240)

[31](#_Toc177508241)

[6. Scénarios de tests définis 33](#_Toc177508242)

[6.1. Requêtes GET 33](#_Toc177508243)

[6.2. Requêtes POST 34](#_Toc177508244)

[6.3. Requêtes PUT 34](#_Toc177508245)

[6.4. Requêtes DELETE 34](#_Toc177508246)

[6.5. Gestion des erreurs 34](#_Toc177508247)

[7 DÉPLOIEMENT ET CONCLUSION 34](#_Toc177508248)

[7.1 Déploiement de l'application 34](#_Toc177508249)

[Conclusion 37](#_Toc177508250)

[Table des matières 38](#_Toc177508251)