Documentation Technique

Ce document présente une vue d'ensemble de l'architecture et du fonctionnement de l'application MySurvey. Il se décompose en deux parties principales :

1. La documentation technique de l'API REST, qui détaille la conception, les couches logicielles et les principaux endpoints exposés.
2. La documentation de la pipeline CI/CD, qui décrit l'automatisation de l'intégration, des tests et du déploiement dans un environnement public.

1. Documentation Technique de l'API REST

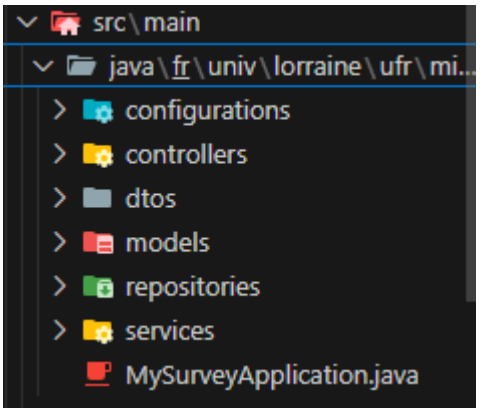
1.1. Introduction

L'API REST de MySurvey permet de gérer des sondages de dates en ligne. Elle offre des fonctionnalités pour créer, modifier, consulter et supprimer des sondages, ainsi que pour gérer la participation des utilisateurs (avec leurs choix de disponibilité) et l'ajout de commentaires. Développée en Java 11 avec Spring Boot, l'API s'appuie sur une architecture en couches facilitant la modularité et la maintenance.

1.2. Architecture Générale

L'API est organisée en plusieurs couches complémentaires :

- **Models**
  Représentent les entités persistées (sondages, participants, commentaires, dates, participations) en utilisant JPA.
- **DTOs**
  (Data Transfer Objects) : Permettent le transfert de données entre le client et le serveur, sans exposer la structure interne des entités.
- **Repositories**
  Interfaces reposant sur Spring Data JPA qui fournissent des opérations CRUD et des requêtes spécifiques (notamment pour le calcul des meilleures dates).
- **Services**
  Contiennent la logique métier de l'application en orchestrant les validations et la persistance des données. Chaque service (pour les sondages, participants, commentaires ou dates) interagit avec les repositories pour effectuer les traitements requis.
- **Controllers**
  Exposent les endpoints REST et coordonnent la conversion entre les entités et les DTOs à l'aide d'un `ModelMapper`. Ils reçoivent les requêtes HTTP et renvoient les réponses au format JSON avec le statut approprié.



1.3. Fonctionnalités et Endpoints

L'API propose plusieurs fonctionnalités regroupées autour de quatre domaines principaux :

- **Sondages**
    - **Création :** Un endpoint reçoit un objet JSON (correspondant à un DTO) pour créer un nouveau sondage en spécifiant le nom, la description, la date de fin, le statut de clôture et l'identifiant du créateur.
    - **Consultation, Mise à Jour et Suppression** : Des endpoints dédiés permettent de récupérer les détails d'un sondage, de le modifier ou de le supprimer.
    - **Gestion des Dates et Commentaires :** Des sous-endpoints permettent d'ajouter des dates ou des commentaires, et de consulter les participations associées.
- **Participants**
    - Les endpoints permettent de créer, consulter, modifier et supprimer des participants, ainsi que de récupérer la liste complète des participants enregistrés.
- **Commentaires**
    - Les utilisateurs peuvent ajouter ou modifier des commentaires associés à un sondage via des endpoints dédiés.
- **Participation et Calcul des Meilleures Dates**
    - Les participants indiquent leur disponibilité (choix : DISPONIBLE, INDISPONIBLE, PEUT-ÊTRE) pour chaque date proposée.
    - Des requêtes spécifiques (définies au niveau des repositories) permettent de déterminer la date optimale et la date potentiellement optimale en se basant sur le nombre de réponses.

1.4. Processus de Traitement d’une Requête

1. **Réception :**
   Un controller reçoit une requête HTTP (GET, POST, PUT ou DELETE) et convertit le payload JSON en DTO grâce à `ModelMapper`.

2. **Traitement :**
   Le DTO est mappé sur une entité. La couche service effectue alors les validations et la logique métier, en interagissant avec les repositories pour la persistance des données.

3. **Réponse :**
   Le résultat est reconverti en DTO et renvoyé au client sous forme de réponse JSON avec un statut HTTP adapté (200, 201, etc.).

2. Documentation de la Pipeline CI/CD

2.1. Objectif et Contexte

La pipeline CI/CD vise à automatiser l’intégration, la vérification de la qualité du code et le déploiement de l’application MySurvey dans un environnement public. Elle garantit la stabilité et la sécurité du projet grâce à l’exécution systématique de différents types de tests.

2.2. Déclencheurs

La pipeline est déclenchée automatiquement dans les situations suivantes :

- **Push** sur la branche `main`
- **Pull Request** visant la branche `main`

2.3. Environnement et Configuration

- **Plateforme d'exécution** : Ubuntu-latest.
- **Service de base de données** : PostgreSQL 13 est déployé en tant que service, avec les variables d’environnement définies pour l’utilisateur, le mot de passe et le nom de la base.
- **Outils** :
  - Java 11 via Maven pour la compilation et le build.
  - Maven est également utilisé pour l’exécution des tests et la génération des rapports.

2.4. Étapes de la Pipeline

La pipeline, définie dans le fichier `prod-logiciel\.github\workflows\ci-cd-pipeline.yml`, se compose des étapes suivantes :

1. **Checkout du Code**
   Utilisation de `actions/checkout@v3` pour récupérer le code source.

2. **Configuration de l'Environnement Java**
   Installation de JDK 11 via `actions/setup-java@v3`.

3. **Linter et Formatter** :
   - Utilisation de **SonarLint** intégré dans **IntelliJ** pour détecter les problèmes de formatage et améliorer la lisibilité du code.
   - Application des règles de codage et refactorisation des fichiers si nécessaire.

4. **Mise en Cache des Dépendances Maven**
   Mise en cache du répertoire `~/.m2/repository` avec `actions/cache@v2` pour accélérer les builds.

5. **Build de l'Application**
   Exécution de la commande `mvn clean install` pour compiler l'application et générer les artefacts.

6. **Exécution des Tests Unitaires**
   Lancement des tests unitaires avec `mvn test -Dtest=**/*Test.java`.

7. **Attente du Démarrage de la Base de Données**
   Une pause (sleep 10) permet de s'assurer que PostgreSQL est opérationnel avant les tests d'intégration.

8. **Exécution des Tests d'Intégration**
   Les tests d'intégration sont exécutés avec `mvn verify -DskipUnitTests -Dtest=**/*IntegrationTest.java`.

9. **Exécution des Tests End-to-End (E2E)**
   Les tests E2E (le cas échéant) sont lancés via `mvn verify -De2eTests -Dtest=**/*E2eTest.java`.

10. **Tests de Vulnérabilité**
    L'exécution d'OWASP Dependency Check (`mvn org.owasp:dependency-check-maven:check`) permet d'identifier les vulnérabilités dans les dépendances.

11. **Génération du Rapport de Couverture de Tests**
    L'outil Jacoco est utilisé pour générer un rapport de couverture de tests (via `mvn jacoco:report`).

12. **Déploiement en Production**
    Si la branche est `main`, une étape de déploiement est déclenchée pour déployer l'application dans l'environnement public. Cette étape peut être personnalisée avec

des commandes spécifiques au déploiement.

2.5. Conclusion de la Pipeline CI/CD

Cette pipeline CI/CD assure une intégration continue complète en automatisant :

- La compilation et le build de l'application.
- L'exécution de tests unitaires, d'intégration, end-to-end et de vulnérabilité.
- La génération de rapports de couverture.
- Le déploiement automatisé dans l'environnement public.

Cette approche permet de réduire les risques liés aux déploiements manuels et garantit la qualité et la sécurité du code tout au long du cycle de développement.

Conclusion

Le projet MySurvey s'appuie sur une architecture en couches bien définie pour l'API REST, favorisant ainsi une séparation claire entre la logique métier, l'accès aux données et l'interface utilisateur. Parallèlement, la pipeline CI/CD assure un processus d'intégration et de déploiement continu, en automatisant la compilation, les tests et le déploiement de l'application.

En combinant ces deux aspects, MySurvey offre une solution évolutive, sécurisée et fiable pour la gestion de sondages en ligne, tout en garantissant une haute qualité de code et une livraison rapide des fonctionnalités.