Cahier des Charges Technique - Application 1Civil.it

1. Architecture Générale

L'application **1Civil.it** sera composée de deux modules principaux :

- **Application mobile** pour les utilisateurs citoyens.
- **Application desktop** pour les administrateurs (services municipaux, police, associations).

Les deux applications consommeront les mêmes API, qui seront implémentées en Java avec le framework Spring Boot. Les API seront déployées en microservices et interagiront via RabbitMQ pour assurer une communication fluide et asynchrone entre les services.

Technologies utilisées :

- **Backend API** : Java, Spring Boot, Hibernate, Data JPA, Lombok, Spring Security, RabbitMQ.
- **Base de données** : PostgreSQL.
- **Tests unitaires** : JUnit, Mockito.
- **Sécurité** : Spring Security / OAuth2 / Keycloak.
- **Frontend Desktop** : Angular.
- **Frontend Mobile** : Flutter.

2. Backend - API

2.1. Microservices

Les microservices seront développés en utilisant le framework Spring Boot et communiqueront entre eux via RabbitMQ. Chaque microservice disposera de sa propre base de données PostgreSQL.

2.1.1. Microservice Utilisateur

- **Fonctionnalités** :
    - Gestion des utilisateurs (inscription, connexion, récupération de mot de passe).
    - Validation des pièces d'identité.
    - Gestion des rôles (citoyen, administrateur).
- **Endpoints** :
    - `POST /api/utilisateurs/register` : Inscription d’un utilisateur.
    - `POST /api/utilisateurs/login` : Connexion d’un utilisateur.

- `GET /api/utilisateurs/{id}` : Récupération des informations d'un utilisateur.
- `PUT /api/utilisateurs/{id}` : Mise à jour des informations d'un utilisateur.

2.1.2. Microservice Signalement

- **Fonctionnalités** :
  - Gestion des signalements (création, mise à jour, suppression).
  - Stockage des photos de signalement et des informations de localisation.
  - Envoi des informations aux administrateurs via RabbitMQ.
- **Endpoints** :
  - `POST /api/signalements` : Création d'un nouveau signalement.
  - `GET /api/signalements` : Récupération de la liste des signalements.
  - `GET /api/signalements/{id}` : Détails d'un signalement.
  - `PUT /api/signalements/{id}` : Mise à jour d'un signalement.

2.1.3. Microservice Traitement des Signalements

- **Fonctionnalités** :
  - Traitement et suivi des signalements par les administrateurs.
  - Mise à jour des statuts des signalements (en cours, résolu, rejeté).
- **Endpoints** :
  - `GET /api/traitements` : Liste des signalements à traiter.
  - `PUT /api/traitements/{id}` : Mise à jour du statut d'un signalement.
  - `POST /api/traitements/commentaire` : Ajouter un commentaire à un signalement.

2.1.4. Microservice Statistiques et Prédictions

- **Fonctionnalités** :
  - Calcul des statistiques sur les signalements.
  - Prédictions des zones à risque et des périodes critiques.
- **Endpoints** :
  - `GET /api/statistiques` : Récupération des statistiques globales.
  - `GET /api/statistiques/predictions` : Récupération des prédictions basées sur les données historiques.

2.2. Sécurité

- Utilisation de **Spring Security** pour la gestion de l'authentification et de l'autorisation.
- Intégration de **OAuth2** ou **Keycloak** pour une gestion centralisée des identités.
- Chiffrement des mots de passe avec **BCrypt**.
- Gestion des permissions par rôle (citoyen, admin) sur les différents endpoints.

2.3. Base de données

- Base de données **PostgreSQL** pour chaque microservice.
- Modélisation des données en fonction des besoins spécifiques de chaque service, avec des relations entre tables bien définies (ex : relation entre utilisateur et signalement).

2.4. Tests

- **Tests unitaires** avec **JUnit** et **Mockito** pour chaque microservice.
- Tests d'intégration pour vérifier les interactions entre les services.
- Mock des services externes avec **Mockito** pour simuler des réponses d'API ou de RabbitMQ.

3. Frontend Desktop - Angular

3.1. Structure Générale

- **Framework** : Angular.
- **Composants principaux** :
    - Navbar : Couleur #4682B4 avec logo en haut à gauche.
    - Footer : Couleur sombre #2F4F4F avec informations de copyright et contacts.
    - Dashboard : Tableau interactif avec la liste des dénonciations, filtrage, et recherche.
    - Carte interactive : Affichage des signalements sur une carte.
    - Page de contact : Formulaire de contact pour communiquer avec les administrateurs.
    - Page de statistiques et de prédictions : Graphiques et tableaux des statistiques et prévisions.

3.2. Dashboard de Signalements

- Liste des signalements sous forme de tableau.
- Colonnes : ID signalement, date, type d'incivilité, statut, actions.
- **Lien de localisation** : Affichage de la localisation sur une carte (Google Maps ou Leaflet).
- **Actions** : Modifier le statut du signalement, ajouter un commentaire.

3.3. Page de Statistiques et Prédictions

- Graphiques interactifs (Barres, Lignes, Camemberts).
- Affichage des prédictions sur les incivilités (zones à risque, périodes critiques).

3.4. UI/UX

- **Palette de couleurs** :
  - Navbar : `#4682B4`
  - Footer : `#2F4F4F`
  - Page : `#F8F8FF`
- Design sobre, facile à naviguer, et responsive.

4. Frontend Mobile - Flutter

4.1. Structure Générale

- **Framework** : Flutter.
- **Composants principaux** :
  - Navbar : Couleur `#50C878` avec logo en haut à gauche.
  - Footer : Couleur sombre `#2F4F4F`.
  - Page principale : Bouton central pour accéder à la fonctionnalité de dénonciation.
  - Carrousel : Explication des fonctionnalités et engagement de l'application lors de la première utilisation.
  - Bulles d'indication : Guides contextuels pour la première utilisation.

4.2. Fonctionnalité de Dénonciation

- **Bouton principal** au centre de l'interface pour commencer un signalement.
- Sélection du type d'incivilité via un menu déroulant.
- Accès à l'appareil photo pour capturer le délit.
- Champ de commentaire pour ajouter des informations supplémentaires.
- Confirmation et envoi du signalement.

4.3. UI/UX

- **Palette de couleurs** :
  - Navbar : `#50C878`
  - Footer : `#2F4F4F`
  - Page : `#FFFFF0`
- Design fluide et minimaliste pour une navigation intuitive.
- Carrousel explicatif et bulles d'aide pour la première utilisation.

5. Interconnexion des Applications

5.1. Communication Backend-Frontend

- Utilisation de **REST API** pour les échanges de données entre le frontend et le backend.
- Utilisation de WebSockets ou Server-Sent Events (SSE) pour les mises à jour en temps réel des signalements sur le dashboard de l'application desktop.

5.2. RabbitMQ

- Utilisation de **RabbitMQ** pour la communication asynchrone entre microservices.
- Échange des informations critiques comme les nouvelles dénonciations ou les mises à jour de statut.

6. Système de Login et Sign Up

6.1. Inscription et Connexion

- Formulaire d'inscription avec validation des champs (nom, prénom, email, mot de passe, pièce d'identité).
- Connexion avec email et mot de passe.
- Option "Se souvenir de moi" pour mémoriser la session de l'utilisateur.

6.2. Gestion des Sessions

- Utilisation de tokens JWT pour l'authentification.
- Stockage sécurisé des tokens dans le local storage ou le secure storage (selon plateforme).

7. Déploiement et Infrastructure

7.1. Infrastructure Cloud

- Déploiement des microservices sur un cloud provider (AWS, GCP, Azure).
- Utilisation de Docker et Kubernetes pour la gestion des conteneurs et le déploiement.

7.2. CI/CD

- Intégration continue avec Jenkins ou GitLab CI.
- Pipeline de déploiement automatisé avec tests et validation.

7.3. Surveillance et Logging

- Surveillance des performances avec Prometheus et Grafana.
- Centralisation des logs avec ELK Stack (Elasticsearch, Logstash, Kibana).