Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE

MINISTÈRE CHARGÉ
DE L'EMPLOI

DOSSIER PROFESSIONNEL (DP)

Nom de naissance ▸ Marzak
Nom d'usage ▸ Marzak
Prénom ▸ Ali
Adresse ▸ 1 Rue des Chevesnes, 67540 Ostwald

Titre professionnel visé

Concepteur & Développeur d'Application niveau 6

MODALITE D'ACCES :

☒ Parcours de formation
☐ Validation des Acquis de l'Expérience (VAE)

Présentation du dossier

Le dossier professionnel (DP) constitue un élément du système de validation du titre professionnel. **Ce titre est délivré par le Ministère chargé de l'emploi.**

Le DP appartient au candidat. Il le conserve, l'actualise durant son parcours et le présente **obligatoirement à chaque session d'examen**.

Pour rédiger le DP, le candidat peut être aidé par un formateur ou par un accompagnateur VAE.

Il est consulté par le jury au moment de la session d'examen.

Pour prendre sa décision, le jury dispose :

1. des résultats de la mise en situation professionnelle complétés, éventuellement, du questionnaire professionnel ou de l'entretien professionnel ou de l'entretien technique ou du questionnement à partir de productions.
2. du **Dossier Professionnel** (DP) dans lequel le candidat a consigné les preuves de sa pratique professionnelle.
3. des résultats des évaluations passées en cours de formation lorsque le candidat évalué est issu d'un parcours de formation
4. de l'entretien final (dans le cadre de la session titre).

[Arrêté du 22 décembre 2015, relatif aux conditions de délivrance des titres professionnels du ministère chargé de l'Emploi]

Ce dossier comporte :

- pour chaque activité-type du titre visé, un à trois exemples de pratique professionnelle ;
- un tableau à renseigner si le candidat souhaite porter à la connaissance du jury la détention d'un titre, d'un diplôme, d'un certificat de qualification professionnelle (CQP) ou des attestations de formation ;
- une déclaration sur l'honneur à compléter et à signer ;
- des documents illustrant la pratique professionnelle du candidat (facultatif)
- des annexes, si nécessaire.

Pour compléter ce dossier, le candidat dispose d'un site web en accès libre sur le site.

http://travail-emploi.gouv.fr/titres-professionnels

Sommaire

Exemples de pratique professionnelle

EXEMPLES DE PRATIQUE PROFESSIONNELLE

Activité-type 1 Développer une application sécurisée

Exemple n°1 ▸ ***Installation de l'environnement et Gestion de projet***

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Description :
Dans le cadre de mon projet **FaceSwap** (microservices), il était essentiel pour moi de mettre en place un environnement standardisé, stable et cohérent.

Réalisation :

- Environnement conteneurisé sous Docker avec un fichier **Docker Compose** (Voir Annexe 1) pour le lancement de tous mes services et applications.
- Utilisation de **Git** pour le Versionnage de mes applications avec une convention de nommage des branches pour assurer une cohérence et une traçabilité (Voir Annexe 2).
- Mise en place d'une convention de nommage des tags pour les **Commits Git** pour une meilleure lecture et gestion du versionnage de code (Voir Annexe 3).
- Mise en place d'un **tableau de projet Trello** pour la gestion de projet avec la Méthode Agile et plus précisément la méthode Scrum (Voir Annexe 4).

Sécurité :
Tous les secrets sont isolés dans un **fichier .env non versionné** présent seulement en local pour éviter toute fuite.

2. Précisez les moyens utilisés :

- Outils de Gestion de projet : Trello
- Outils de Versionnage de code : Git
- Outils d'hébergement de code : Github
- Moteur Docker : Docker Desktop (Engine)

3. Avec qui avez-vous travaillé ?

J'ai réalisé ce projet en autonomie.

DOSSIER PROFESSIONNEL (DP)

4. Contexte

Nom de l'entreprise, organisme ou association ▸ *M2I Formation*

Chantier, atelier, service ▸ *M2I Formation CDA*

Période d'exercice ▸ Du : *01/10/2025* au : *28/01/2026*

5. Informations complémentaires *(facultatif)*

Activité-type 1 Développer une application sécurisée

Exemple n°2 ▸ *Cliquez ici pour entrer l'intitulé de l'exemple*

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Description :
Dans le projet **OurEvents**, l'objectif est de développer une interface utilisateur qui soit fluide, respectant les normes et sécurisé.

Réalisation :

- Développement du Front avec **Next JS** qui est un Méta Framework utilisant la librairie **React**
- Interface **Responsive Design** avec l'utilisation de Tailwind CSS et ces breakpoints (Voir Annexe 5)
- Utilisation de librairie **ReactJS** pour un front plus interactif et dynamique avec l'utilisation du langage JavaScript. Meilleure organisation avec un système de **"Component" réutilisable**
- Utilisation de **"Tailwind CSS"** pour une Interface Utilisateur (UI) beaucoup plus agréable, une application optimisée avec les classes utilitaires css et un responsive design beaucoup plus simple et efficace à mettre en place
- La combinaison NextJs et Tailwind CSS améliore considérablement **l'expérience utilisateur (UX)**.
- Le Framework NextJs permet aussi une **meilleure SEO** auprès des moteurs de recherches grâce à son optimisation
- Mise en place de bonne pratique pour la gestion des images permettant un meilleur référencement (SEO) : Utilisation de **l'attribut "alt"**
- Respect des **standards W3C**
- Application de transitions sur les éléments interactifs (boutons, liens, images) pour améliorer l'expérience utilisateur (UX)

Sécurité :
ReactJs échappe automatiquement les données lors de son affichage via les balises react

2. Précisez les moyens utilisés :

- Editeur de code : VS Code
- Framework : NextJs, Tailwind CSS
- Documentation : NextJs, Tailwind
- Standards : W3C, WCAG

3. Avec qui avez-vous travaillé ?

J'ai réalisé ce projet en autonomie, sous la supervision de mes formateurs.

4. Contexte

Nom de l'entreprise, organisme ou association ▸ *M2I Formation*

Chantier, atelier, service ▸ M2I Formation CDA

Période d'exercice ▸ Du : *01/10/2025* au : *15/10/2025*

5. Informations complémentaires *(facultatif)*

Activité-type 1 Développer une application sécurisée

Exemple n°3 ▸ ***Développer des composants métier***

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Description :
Pour l'application **FaceSwap**, l'enjeu était de garantir une sécurité rigoureuse, une communication fluide avec les services et un code qui soit facilement maintenable.

Réalisation :

- Mise en place d'un service de **Gateway** pour la redirection vers les services
- Mise en place d'un service **Discovery** permettant l'enregistrement d'un service à son lancement dans l'annuaire grâce à **Netflix Eureka** (Voir Annexe 6)
- Mise en place d'un **Rate Limiter** dans le service Gateway (Voir Annexe 7)
- Mise en place d'un **CORS** (Voir Annexe 8)
- Utilisation de **Token JWT** (AccessToken et Refresh Token) pour l'authentification (Voir Annexe 9)
- **Hachage du mot de passe** en utilisant l'algorithme **Bcrypt**
- Utilisation de **Spring Data JPA** (**Hibernate**) pour la partie SQL qui permet d'utiliser des requêtes préparées qui protègent nativement l'application contre **les injections SQL**
- Utilisation des **annotations** proposées par Spring boot qui sont optimisées et suivent les bonnes pratiques de codage grâce **aux Design Pattern utilisés**
- Mise en place d'un code qui respecte les **principes SOLID** pour faciliter une maintenance
- **Centralisation des Exceptions** dans les services (Voir Annexe 10)
- **Standardisations des réponses** avec des **DTOs** (Voir Annexe 11)

2. Précisez les moyens utilisés :

- Editeur de code : Eclipse IDE for Java Developers
- Framework : Spring boot
- Dépendances : Spring Security et Spring Cloud
- Outils de dépendance : Maven

3. Avec qui avez-vous travaillé ?

J'ai réalisé ce projet en autonomie.

4. Contexte

Nom de l'entreprise, organisme ou association ▸ *M2I Formation*

Chantier, atelier, service ▸ M2I Formation CDA

Période d'exercice ▸ Du : *01/10/2025* au : *28/01/2026*

5. Informations complémentaires *(facultatif)*

Activité-type 2

Concevoir et développer une application sécurisée organisée en couches

Exemple n° 1 ▸ ***Analyser les besoins et maquetter une application***

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Description :

Dans le projet **OurEvents**, il était essentiel de rédiger un cahier des charges dans lequel nous **analysons les besoins** pour ensuite les traduire en **spécification technique**.
Cette étape est très importante puisqu'il s'agit des fondations de notre conception permettant ainsi d'éviter des coûts suite à des erreurs de conception.
Une fois que ces besoins sont décrits, nous passons à la réalisation du maquettage avec l'outil **Figma** qui est un logiciel de design UX/UI.

Réalisation :

- Mise en place d'un **cahier des charges** qui fournit les besoins en termes de Charte graphique, d'accessibilité et de sécurité.
- Conception d'un **diagramme UML de cas d'utilisation** (Voir Annexe 12)
- Réalisation d'un **Zoning** pour le maquettage permettant de définir la structure du Front (Voir Annexe 13)
- Réalisation d'un **Wireframe** permettant une visualisation du parcours utilisateur et de fonctionnalités essentiels de notre application (Voir Annexe 14)
- Réalisation d'une **Maquette** dans laquelle on définit **l'identité visuelle** de l'application à partir de la charte graphique que l'on a récupéré lors de la conception du cahier des charges (Voir Annexe 15)
- Mise en place d'une **conception Design Responsive** avec une version Mobile et une version desktop de nos maquettes
- Conception en utilisant **l'approche Mobile First**, c'est-à-dire qu'on pense l'application en version mobile d'abord. Elle permet d'optimiser le **référencement naturel** (SEO)

2. Précisez les moyens utilisés :

- Figma pour la réalisation du maquettage
- Draw.io pour la réalisation des diagrammes

3. Avec qui avez-vous travaillé ?

J'ai réalisé ce projet en autonomie, sous la supervision de mes formateurs.

4. Contexte

Nom de l'entreprise, organisme ou association ▸ *M2I Formation*

Chantier, atelier, service ▸ M2I Formation CDA

Période d'exercice ▸ Du : *01/10/2025* au : *15/10/2025*

5. Informations complémentaires *(facultatif)*

Activité-type 2

Concevoir et développer une application sécurisée organisée en couches

Exemple n° 2 ▸ ***Définir l'architecture logicielle d'une application***

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Description :

Dans mon application **FaceSwap**, le choix de l'architecture logicielle a été très important pour garantir de bonnes performances, mais aussi une bonne sécurité. Pour ma part, j'étais parti sur une application avec une **architecture microservices** ce qui a des avantages, mais aussi des inconvénients.

J'ai décidé donc de partir sur une **application distribuée** (Voir Annexe 16) avec l'utilisation du Framework **Angular** pour la partie Front End et le Framework **Spring boot** pour la partie Back End. Le Back End suivra donc une architecture microservices.

Réalisation :

- Réalisation d'une architecture microservices où j'ai séparé l'application en différents services :
  - Service qui gère l'authentification et les utilisateurs
  - Service qui gère la Gateway
  - Service qui gère les abonnements et les paiements
  - Service qui gère l'envoi des Emails
  - Service qui gère la fonctionnalité principal FaceSwap

- A l'intérieur de chaque microservice avec Spring boot, j'applique une **architecture n-Tiers** pour une meilleure organisation, stabilité et séparation du code en se basant sur les bonnes pratiques de codage :
  - Package Contrôleur qui gère les Endpoints d'un service
  - Package Service qui gère toute la logique métier
  - Package Repository (DAO) qui permet l'accès aux données
  - Package DTO qui gère le transfert de données
- Pour la partie Front End avec Angular, je poursuis vers une **architecture modulaire orientée fonctionnalités**.
- Réalisation d'un **diagramme UML de classe** qui permet la modélisation d'un système dans une programmation orientée objet (Voir Annexe 17)
- Réalisation d'un **diagramme UML de séquence** qui permet de renseigner sur le comportement des composants dans un système (Voir Annexe 18)

2. Précisez les moyens utilisés :

- Editeur de code pour le Front End : VS Code
- Editeur de code pour le Back End : Eclipse IDE for Java Developers
- Draw.io pour la réalisation des diagrammes
- Documentation : Angular

3. Avec qui avez-vous travaillé ?

J'ai réalisé ce projet en autonomie.

4. Contexte

Nom de l'entreprise, organisme ou association ▸ *M2I Formation*

Chantier, atelier, service ▸ M2I Formation CDA

Période d'exercice ▸ Du : *01/10/2025* au : *28/01/2026*

5. Informations complémentaires *(facultatif)*

Activité-type 2

Concevoir et développer une application sécurisée organisée en couches

Exemple n° 3 ▸ ***Conception de la Base de Données et Accès aux Données***

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Description :

Dans l'application **OurEvents**, je devais récupérer des données, les stockées et pouvoir les récupérer. Pour cela, nous devons mettre en place un **système de gestion de bases de données** (SGBD) et gérer toute la conception de la Base de données pour éviter des conflits de données par la suite avec des formats différents.

Réalisation :

- Choix et mise en place d'un SGBD relationnel avec l'utilisation de l'image Docker **Mysql**
- Mise en place de PhyMyAdmin (via Docker) qui s'occupe d'administrer la base de données
- Mise en place d'un **dictionnaire de données** (Voir Annexe 19) dans lequel on listera toutes les données dont nous avons besoin avec leurs caractéristiques
- Conception du **MCD** (Modèle Conceptuel de données) (Voir Annexe 20)
- Conception du **MLD** (Modèle Logique de données) (Voir Annexe 21)
- Conception du **MPD** (Modèle Physique de données) (Voir Annexe 22)
- Accès aux données dans le code grâce à **Doctrine** qui est l'ORM de Symfony qui joue le rôle de **couche d'abstraction** à la base de données

2. Précisez les moyens utilisés :

- Outils pour les images : Docker Desktop (Engine)
- Draw.io pour la réalisation des diagrammes

3. Avec qui avez-vous travaillé ?

J'ai réalisé ce projet en autonomie, sous la supervision de mes formateurs.

4. Contexte

Nom de l'entreprise, organisme ou association ▸ *M2I Formation*

Chantier, atelier, service ▸ M2I Formation CDA

Période d'exercice ▸ Du : *01/10/2025* au : *15/10/2025*

5. Informations complémentaires *(facultatif)*

Activité-type 3 — Préparer le déploiement d'une application sécurisée

Exemple n° 1 ▸ ***Préparer et exécuter les plans de tests d'une application***

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Description :
Dans le cadre du projet **FaceSwap**, pour garantir la robustesse et la non régression lors du développement, j'ai mis en place un **plan de tests** comprenant des Tests **unitaires** et des Tests **d'intégration** au niveau de l'application Microservices en Java.

Réalisation :
Pour les tests, j'ai utilisé la dépendance pour les tests que proposent Spring boot qui contient tous les outils nécessaires, dont **Junit 5** et **Mockito**.

Les Tests appliquent un **Pattern AAA** (Arrange, Act et Assert) permettant de séparer les responsabilités dans un Test et de garantir une lisibilité et une maintenance optimale. (Voir Annexe 23)

Nous avons les tests unitaires qui vont nous permettre de valider la **logique métier de manière isolée** de chacun des composants dans chacun de nos microservices.

Nous avons les tests d'intégrations qui va nous permettre de valider un **ensemble de composants**
Pour pousser la validation, j'ai intégré **SonarQube** (logiciel tiers) dans mon pipeline CI/CD pour analyser la **qualité du code** à chaque push et pour détecter les **vulnérabilités de sécurité** (OWASP) mais aussi détecter les mauvaises pratiques.

2. Précisez les moyens utilisés :

- Outils de dépendance : Maven
- Framework : Spring boot
- Dépendance : Spring boot Test (JUnit 5 et Mockito)
- Analyse code : SonarCloud
- Documentation : Spring boot

3. Avec qui avez-vous travaillé ?

J'ai réalisé ce projet en autonomie.

4. Contexte

Nom de l'entreprise, organisme ou association ▸ *M2I Formation*

Chantier, atelier, service ▸ M2I Formation CDA

Période d'exercice ▸ Du : *01/10/2025* au : *28/01/2026*

5. Informations complémentaires *(facultatif)*

Activité-type 3 — Préparer le déploiement d'une application sécurisée

Exemple n° 2 ▸ ***Préparer et documenter le déploiement d'une application***

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Descrption :
Le fait que l'application **FaceSwap** ait une architecture distribuée (microservices) nécessite une documentation rigoureuse pour permettre de comprendre le fonctionnement de l'application ce qui facilitera la phase de développement.

Réalisation technique :
Le framework Spring boot fournit la dépendance **SpringDoc/OpenApi** que j'ai implémenté dans mes services et qui permet de générer une **documentation dynamique** de manière automatique.
Cette dépendance contient **Swagger UI** qui permet de visualiser les endpoints (Voir Annexe 24) avec toutes les informations qui y sont liés et de pouvoir les **tester en temps réel** sans avoir besoin d'outils externes à l'application.

2. Précisez les moyens utilisés :

- Editeur de code : Eclipse IDE for Java Developers
- Outils de dépendance : Maven
- Framework : Spring boot
- Dépendance : SpringDoc / OpenApi
- Documentation : Spring boot

3. Avec qui avez-vous travaillé ?

J'ai réalisé ce projet en autonomie.

4. Contexte

Nom de l'entreprise, organisme ou association ▸ *M2I Formation*

Chantier, atelier, service ▸ M2I Formation CDA

Période d'exercice ▸ Du : *01/10/2025* au : *28/01/2026*

5. Informations complémentaires *(facultatif)*

Activité-type 3 — Préparer le déploiement d'une application sécurisée

Exemple n° 3 ▸ ***Contribuer à la mise en production dans une démarche DevOps***

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Description :

Etant parti pour une méthode Agile pour mon projet **FaceSwap**, il était essentiel d'automatiser le cycle de vie de mon application en partant sur **une approche DevOps**. Cette approche aura pour but de réduire considérablement les erreurs que l'on puisse faire manuellement lors du cycle de vie de l'application ou encore des erreurs commis dans le code lors du développement. Cela permet donc d'accélérer les différentes phases de cycle de vie comme le développement via les différentes branches ou encore la mise en production dans un environnement dédié. De plus, après un déploiement, il est très important de gérer la maintenance et pour cela avoir une **Stack de Monitoring** est indispensable.

Réalisation :

Pour cela, j'ai mis en place un **pipeline CI/CD** (Voir Annexe 25 & 26) complète avec l'aide de **Github Actions** qui s'organise en 3 phases :

- **Analyse + Build & Test :** On analyse le code avec **SonarQube** puis Compilation du code avec Maven + lancement et vérification des tests

- **Conteneurisation :** Création d'images Docker optimisées via **Dockerfile** pour chaque microservice avec une logique **multi stage** qui prend aussi en compte les caches Docker :

  - **Builder :** Phase de compilation sans les Tests qui sont effectués durant le pipeline CI.
  - **Runner :** Phase d'exécution qui récupère le fichier JAR du Builder et lance l'image Docker et donc charge l'application.

Durant le pipeline CD, on optimise le build de l'image avec **Docker BuildX** pour permettre de réduire considérablement les temps de déploiement notamment lors de la reconstruction d'image grâce au système de **mise en cache Docker**.

- Déploiement : Stockage de l'image dans le **registre GitHub Container Registry (GHCR)** permettant un stockage centralisé et sécurisé avec une mise en place de tag intelligent selon la branche.

J'ai aussi mis en place une **Stack de Monitoring "Grafana Observability"** basée sur :

- **Prometheus** qui va permettre de récupérer les **métriques** de chacun de nos services
- **Tempo** qui va gérer les traces distribuées et donc permettre de suivre une requête de bout en bout pour avoir une **meilleure traçabilité** notamment dans une application microservices.
- **Loki** qui est un système de **centralisation des logs** permettant aux services de produire leurs propres logs

- **Grafana** qui aura pour rôle de **centraliser les 3 éléments précédents** dans un seul Dashboard avec notamment des graphes pour mieux visualiser les informations obtenues.

Sécurité :

- Mise en place d'un **utilisateur non root** avec transmission des permissions lors de la construction de l'image Docker.
- Audit de sécurité avec **Trivy** permettant de scanner l'image avant le déploiement.

2. Précisez les moyens utilisés :

- Editeur de code : Eclipse IDE for Java Developers
- Outils de dépendance : Maven
- Framework : Spring boot
- Logiciel Docker : Docker Desktop & Docker Hub
- Logiciel Pipeline DevOps : Github Actions
- Image Monitoring : Grafana, Prometheus, Tempo et Loki
- Documentation : Spring boot, Grafana, Prometheus

3. Avec qui avez-vous travaillé ?

J'ai réalisé ce projet en autonomie.

4. Contexte

Nom de l'entreprise, organisme ou association ▸ *M2I Formation*

Chantier, atelier, service ▸ M2I Formation CDA

Période d'exercice ▸ Du : *01/10/2025* au : *28/01/2026*

5. Informations complémentaires *(facultatif)*

Titres, diplômes, CQP, attestations de formation

(facultatif)

Intitulé	Autorité ou organisme	Date
Cliquez ici.	Cliquez ici pour taper du texte.	Cliquez ici pour sélectionner une date.

Déclaration sur l'honneur

Je soussigné(e) [prénom et nom] MARZAK Ali ,

déclare sur l'honneur que les renseignements fournis dans ce dossier sont exacts et que je suis l'auteur(e) des réalisations jointes.

Fait à Strasbourg le 26/01/2026

pour faire valoir ce que de droit.

Signature :

Documents illustrant la pratique professionnelle

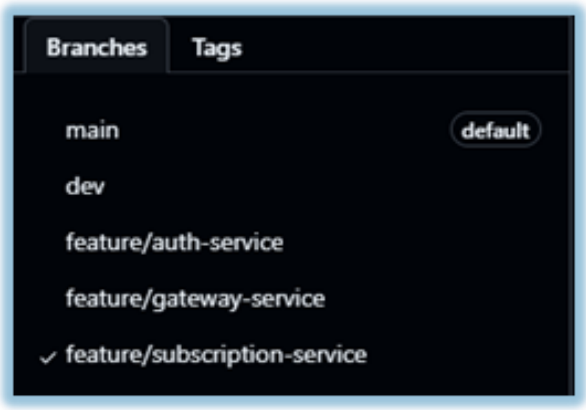
(facultatif)

Intitulé
Cliquez ici pour taper du texte.

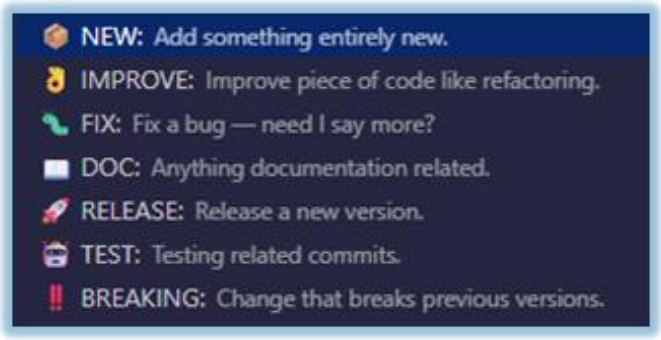
ANNEXES

```yaml
services:
  discovery-service:
    build:
      context: ./discovery-service
      dockerfile: Dockerfile
    container_name: discovery-service
    environment:
      - SPRING_PROFILES_ACTIVE=${SPRING_PROFILES_ACTIVE}
      - EUREKA_CLIENT_REGISTER_WITH_EUREKA=false
      - EUREKA_CLIENT_FETCH_REGISTRY=false
      - OTEL_EXPORTER_OTLP_ENDPOINT=${OTEL_EXPORTER_OTLP_ENDPOINT}
      - OTEL_SERVICE_NAME=discovery-service
      - OTEL_METRICS_EXPORTER=none
      - OTEL_TRACES_EXPORTER=otlp
    ports:
      - "8761:8761"
    networks:
      - default
      - observability
    env_file:
      - .env
    deploy:
      resources:
        limits:
          cpus: '0.5'   # Limite l'usage CPU à 50% d'un CPU
          memory: 512M  # Limite l'usage mémoire à 512MB
        reservations:
          cpus: '0.25'  # Réserve au moins 25% d'un CPU
          memory: 256M  # Réserve au moins 256MB de mémoire
  gateway-service:
    build:
      context: ./gateway-service
      dockerfile: Dockerfile
    container_name: gateway-service
    environment:
      - FRONTEND_URL=${FRONTEND_URL}
      - SPRING_PROFILES_ACTIVE=${SPRING_PROFILES_ACTIVE}
      - SPRING_APPLICATION_NAME=gateway-service
      - EUREKA_CLIENT_SERVICEURL_DEFAULTZONE=http://discovery-service:${DISCOVERY_PORT}/eureka/
      - EUREKA_INSTANCE_PREFER_IP_ADDRESS=true
      - JWT_SECRET=${JWT_SECRET}
      - SPRING_DATA_REDIS_HOST=${REDIS_HOST}
      - SPRING_DATA_REDIS_PORT=${REDIS_PORT}
      - SPRING_DATA_REDIS_TIMEOUT=${REDIS_TIMEOUT}
      - PROMETHEUS_PORT=8080
      - LOKI_URL=${LOKI_URL}
      - OTEL_EXPORTER_OTLP_ENDPOINT=${OTEL_EXPORTER_OTLP_ENDPOINT}
      - OTEL_SERVICE_NAME=gateway-service
      - OTEL_METRICS_EXPORTER=none
      - OTEL_TRACES_EXPORTER=otlp
    ports:
      - "${GATEWAY_PORT}:8888"
    networks:
      - default
      - observability
    depends_on:
      discovery-service:
        condition: service_started
      redis-gateway:
        condition: service_healthy
    env_file:
      - .env
    deploy:
      resources:
        limits:
          cpus: '0.5'   # Limite l'usage CPU à 50% d'un CPU
          memory: 512M  # Limite l'usage mémoire à 512MB
        reservations:
          cpus: '0.25'  # Réserve au moins 25% d'un CPU
          memory: 256M  # Réserve au moins 256MB de mémoire
```

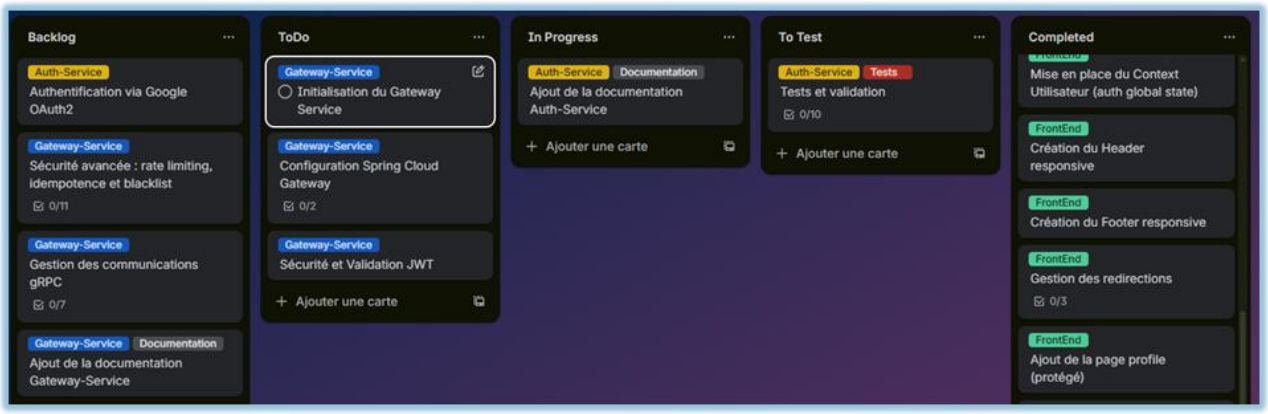
Annexe 1 Docker Compose de l'application FaceSwap



Annexe 2 Convention de nommage des branches git



Annexe 3 Convention de nommage des Commits



Annexe 4 Tableau Trello FaceSwap

Breakpoint prefix	Minimum width	CSS
`sm`	40rem *(640px)*	`@media (width >= 40rem) { ... }`
`md`	48rem *(768px)*	`@media (width >= 48rem) { ... }`
`lg`	64rem *(1024px)*	`@media (width >= 64rem) { ... }`
`xl`	80rem *(1280px)*	`@media (width >= 80rem) { ... }`
`2xl`	96rem *(1536px)*	`@media (width >= 96rem) { ... }`

Annexe 5 Breakpoints proposés par Tailwind CSS

```java
@SpringBootApplication
@EnableEurekaServer
public class DiscoveryServiceApplication {

    public static void main(String[] args) {
        SpringApplication.run(DiscoveryServiceApplication.class, args);
    }

}
```

Annexe 6 Lancement du Service Discovery

```yaml
spring:
  data:
    redis:
      host: ${SPRING_DATA_REDIS_HOST:gateway_redis}
      port: ${SPRING_DATA_REDIS_PORT:6379}
      timeout: ${SPRING_DATA_REDIS_TIMEOUT:2000ms}
  cloud:
    gateway:
      routes:
        - id: auth_login
          uri: lb://auth-service
          predicates:
            - Path=/api/v1/auth/authenticate
            - Method=POST
          filters:
            - name: RequestRateLimiter
              args:
                key-resolver: "#{@ipKeyResolver}"
                redis-rate-limiter.replenishRate: 3
                redis-rate-limiter.burstCapacity: 5
                redis-rate-limiter.requestedTokens: 1
            - name: CircuitBreaker
              args:
                name: authLoginCircuitBreaker
                fallbackUri: forward:/fallback/auth
        - id: auth_register
          uri: lb://auth-service
          predicates:
            - Path=/api/v1/auth/register
            - Method=POST
          filters:
            - name: RequestRateLimiter
              args:
                key-resolver: "#{@ipKeyResolver}"
                redis-rate-limiter.replenishRate: 2
                redis-rate-limiter.burstCapacity: 4
                redis-rate-limiter.requestedTokens: 1
            - name: CircuitBreaker
              args:
                name: authRegisterCircuitBreaker
                fallbackUri: forward:/fallback/auth
        - id: auth_logout
```

Annexe 7 Rate Limiter du Gateway

```yaml
spring:
  cloud:
    gateway:
      globalcors:
        cors-configurations:
          '[/**]':
            allowedOrigins:
              - "http://localhost:3000"
              - ${FRONTEND_URL:http://localhost:4200}
            allowedMethods:
              - GET
              - POST
              - PUT
              - DELETE
              - OPTIONS
              - PATCH
            allowedHeaders:
              - "*"
            exposedHeaders:
              - Authorization
              - Content-Type
            allowCredentials: true
            maxAge: 3600
```

Annexe 8 Cors application FaceSwap

```java
@Service
public class JwtService {
    @Value("${application.security.jwt.secret-key}")
    private String secretKey;
    @Value("${application.security.jwt.expiration}")
    private long jwtExpiration;
    @Value("${application.security.jwt.refresh-token.expiration}")
    private long refreshExpiration;

    public String extractUsername(String token) {
        return extractClaim(token, Claims::getSubject);
    }

    public <T> T extractClaim(String token, Function<Claims, T> claimsResolver) {
        final Claims claims = extractAllClaims(token);
        return claimsResolver.apply(claims);
    }

    public String generateToken(UserDetails userDetails) {
        return generateToken(new HashMap<>(), userDetails);
    }

    public String generateToken(Map<String, Object> extraClaims, UserDetails userDetails) {
        return buildToken(extraClaims, userDetails, jwtExpiration);
    }

    public String generateRefreshToken(UserDetails userDetails) {
        return buildToken(new HashMap<>(), userDetails, refreshExpiration);
    }
    private String buildToken(Map<String, Object> extraClaims, UserDetails userDetails, long expirationInSeconds) {
        long expirationInMillis = expirationInSeconds * 1000;
        return Jwts
            .builder()
            .setClaims(extraClaims)
            .setSubject(userDetails.getUsername())
            .setIssuedAt(new Date(System.currentTimeMillis()))
            .setExpiration(new Date(System.currentTimeMillis() + expirationInMillis))
            .signWith(getSignInKey(), SignatureAlgorithm.HS256)
            .compact();
    }
    public boolean isTokenValid(String token, UserDetails userDetails) {
        final String username = extractUsername(token);
        return (username.equals(userDetails.getUsername())) && !isTokenExpired(token);
    }
    private boolean isTokenExpired(String token) {
        return extractExpiration(token).before(new Date());
    }
    private Date extractExpiration(String token) {
        return extractClaim(token, Claims::getExpiration);
    }
    private Claims extractAllClaims(String token) {
        return Jwts
            .parserBuilder()
            .setSigningKey(getSignInKey())
            .build()
            .parseClaimsJws(token)
            .getBody();
    }
    private Key getSignInKey() {
        byte[] keyBytes = Decoders.BASE64.decode(secretKey);
        return Keys.hmacShaKeyFor(keyBytes);
    }
}
```

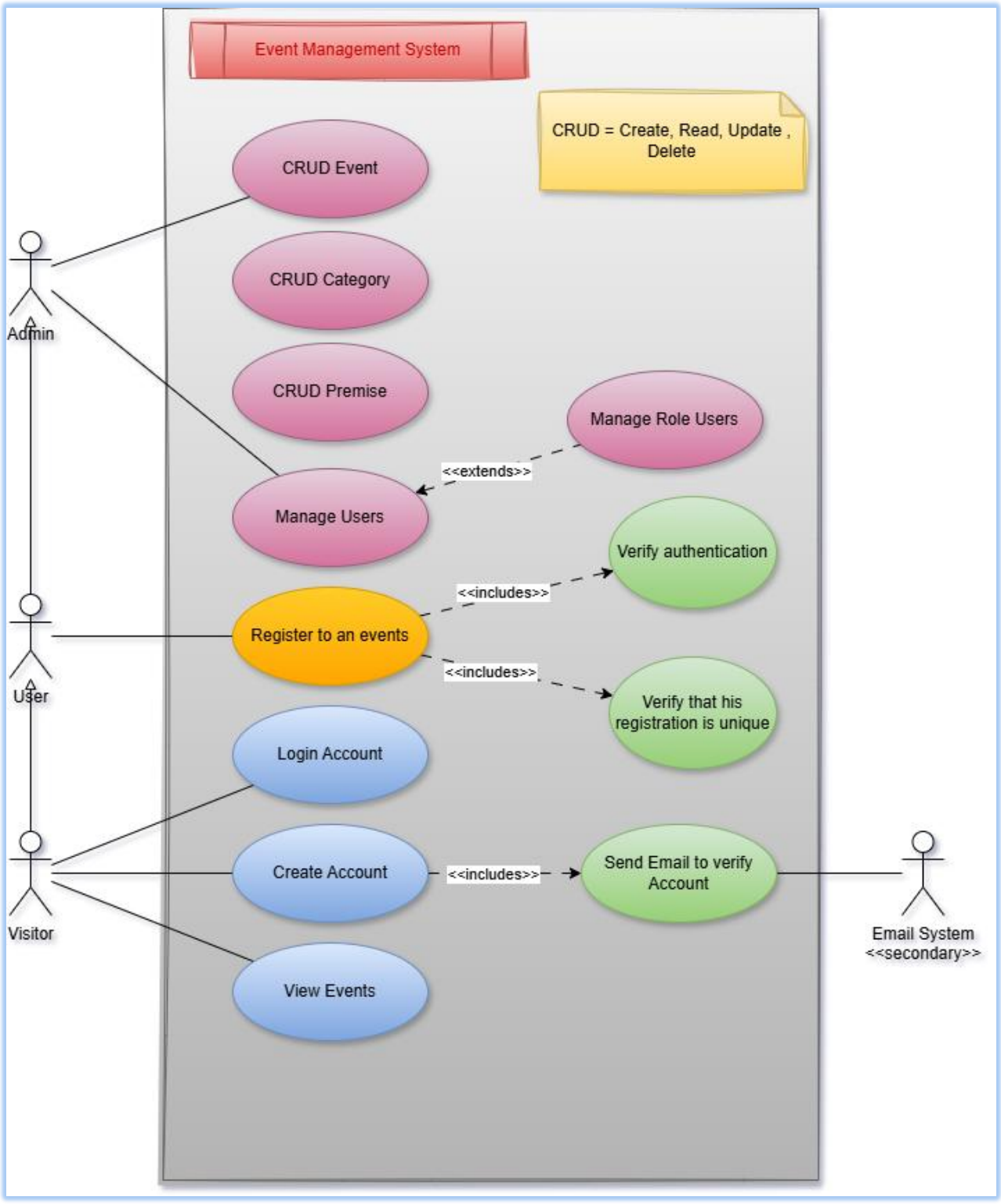
Annexe 9 Service de gestion JWT

```java
@Slf4j
@RestControllerAdvice
public class GlobalExceptionHandler {
    @ExceptionHandler(IllegalArgumentException.class)
    public ResponseEntity<ApiResponse<Void>> handleIllegalArgument(IllegalArgumentException exception, HttpServletRequest request)
    {
        return buildResponse(HttpStatus.BAD_REQUEST, "VALIDATION_ERROR", exception.getMessage(), request);
    }

    @ExceptionHandler(IllegalStateException.class)
    public ResponseEntity<ApiResponse<Void>> handleIllegalState(IllegalStateException exception, HttpServletRequest request)
    {
        return buildResponse(HttpStatus.FORBIDDEN, "BUSINESS_RULE_VIOLATION", exception.getMessage(), request);
    }

    @ExceptionHandler(SecurityException.class)
    public ResponseEntity<ApiResponse<Void>> handleSecurity(SecurityException exception, HttpServletRequest request)
    {
        return buildResponse(HttpStatus.UNAUTHORIZED, "AUTHENTICATION_ERROR", exception.getMessage(), request);
    }

    @ExceptionHandler(Exception.class)
    public ResponseEntity<ApiResponse<Void>> handleGeneral(Exception exception, HttpServletRequest request)
    {
        log.error("Erreur non gérée", exception);
        return buildResponse(HttpStatus.INTERNAL_SERVER_ERROR, "INTERNAL_ERROR", "Une erreur interne est survenue", request);
    }

    @ExceptionHandler(MethodArgumentNotValidException.class)
    public ResponseEntity<ApiResponse<Void>> handleValidationException(MethodArgumentNotValidException exception, HttpServletRequest
request)
    {
        String errorMessage = exception.getBindingResult()
                .getFieldErrors()
                .stream()
                .map(FieldError::getDefaultMessage)
                .findFirst()
                .orElse("Requête invalide");

        return buildResponse(HttpStatus.BAD_REQUEST, "VALIDATION_ERROR", errorMessage, request);
    }

    private ResponseEntity<ApiResponse<Void>> buildResponse(HttpStatus status, String code, String message, HttpServletRequest request) {
        ApiResponse<Void> response = ApiResponse.<Void>builder()
                .timestamp(LocalDateTime.now().toString())
                .status(status.value())
                .success(false)
                .code(code)
                .message(message)
                .path(request.getRequestURI())
                .build();
        return new ResponseEntity<>(response, status);
    }
}
```

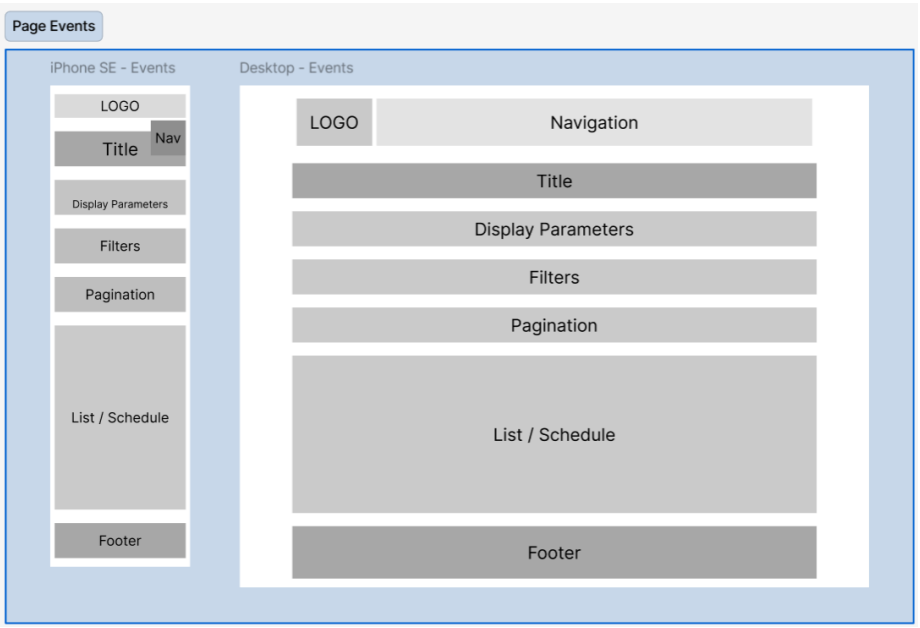
Annexe 10 Gestion des Exceptions

```java
return ApiResponse.<RegisterResponse>builder()
                .status(201)
                .success(true)
                .message("Inscription réussie ! Un email de vérification vous a été envoyé.")
                .data(response)
                .build();
```

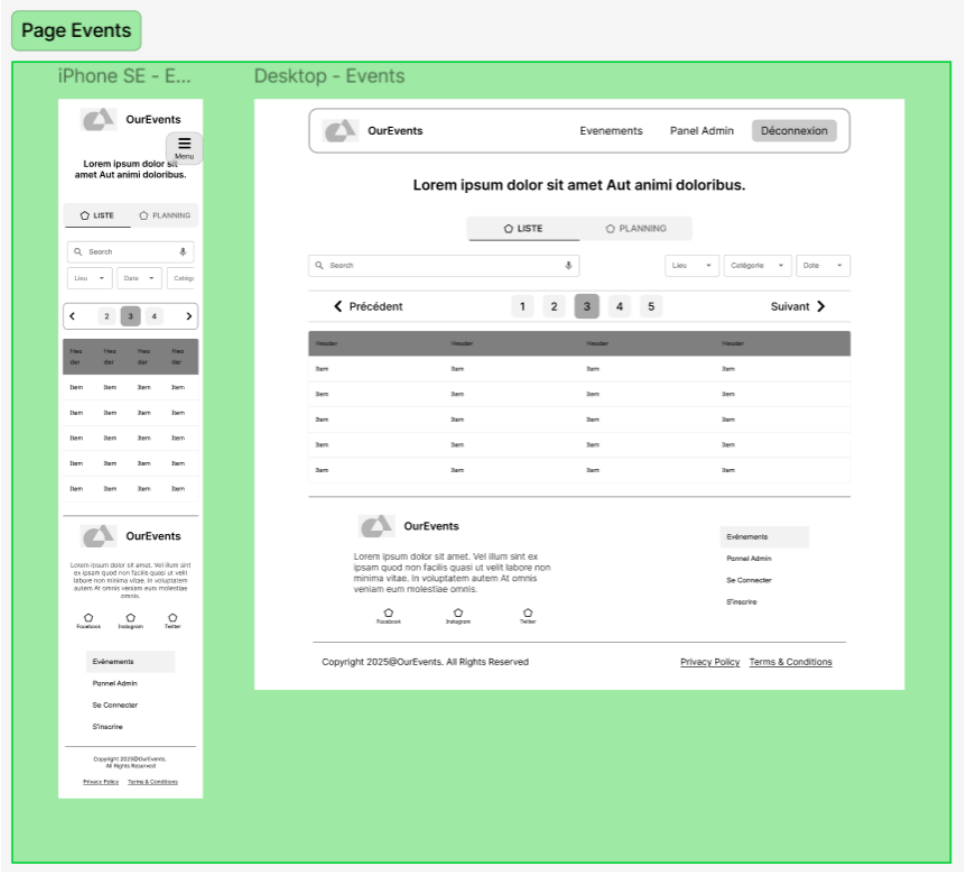
Annexe 11 Standardisation des réponses



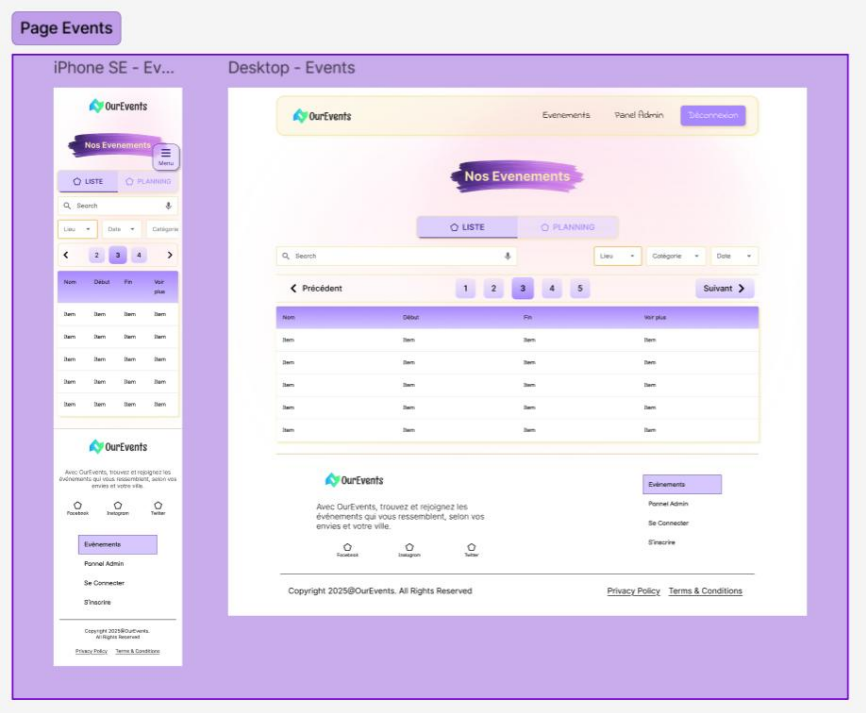
Annexe 12 Diagramme UML UseCase



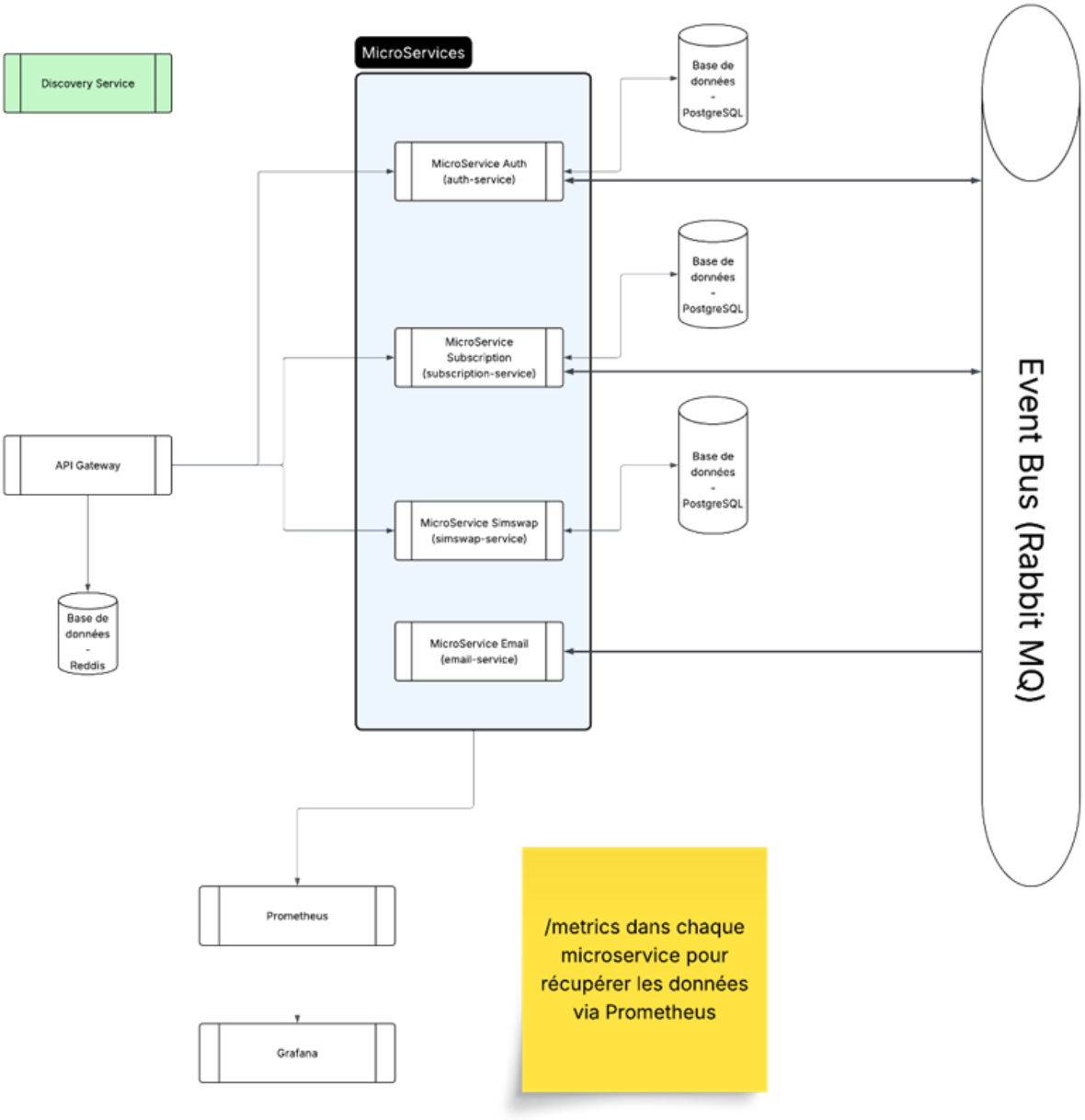
Annexe 13 Zoning du projet OurEvents



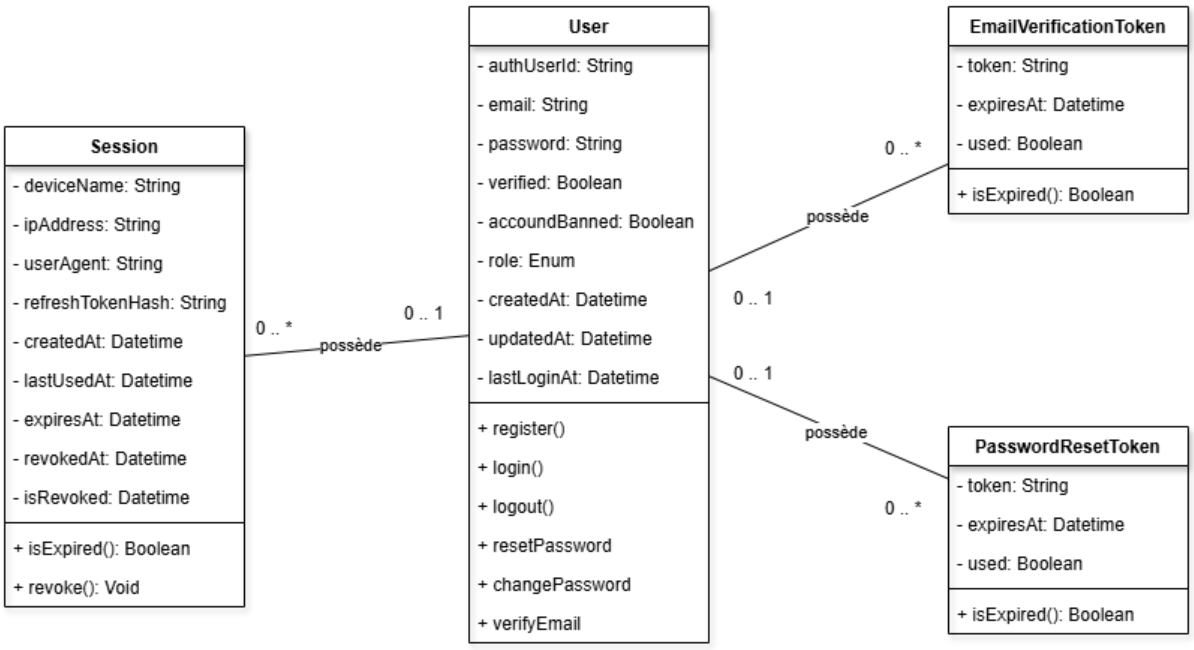
Annexe 14 Wireframe du projet OurEvents



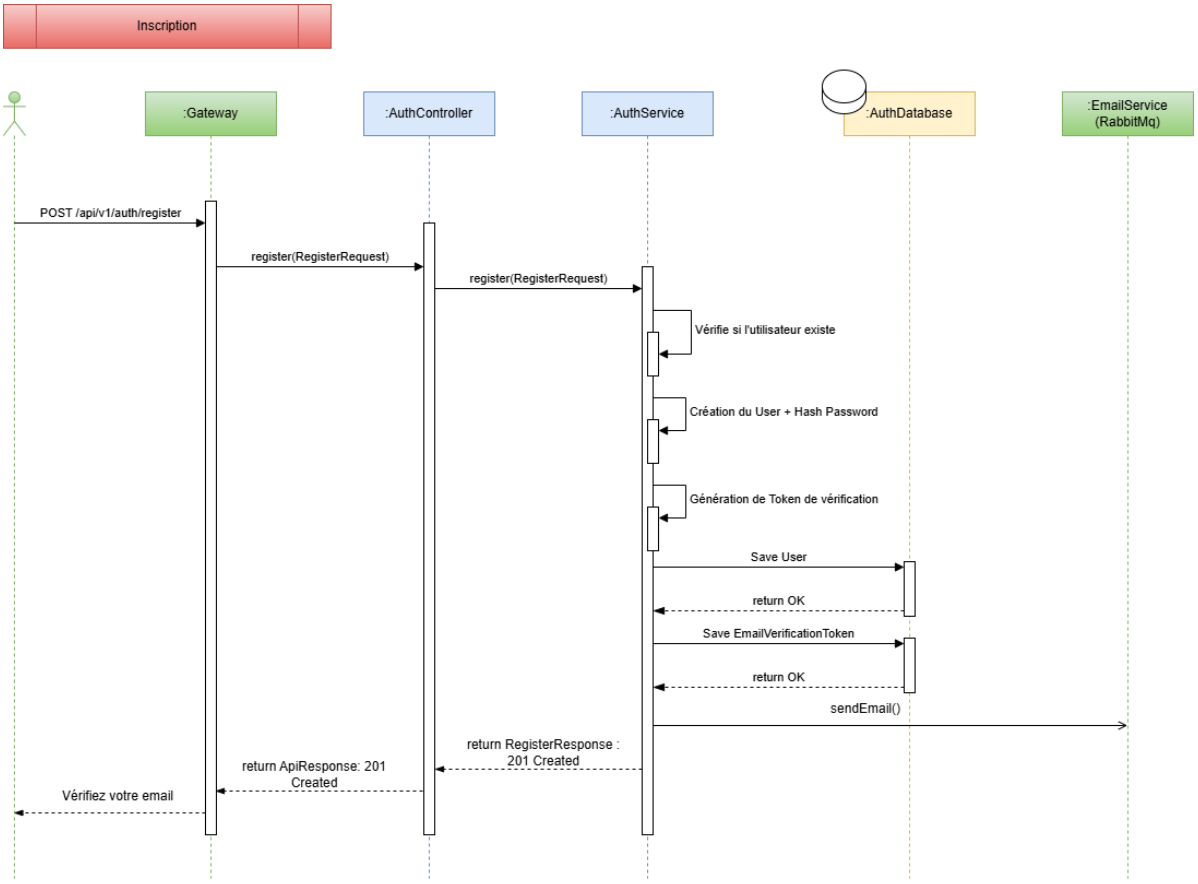
Annexe 15 Maquette du projet OurEvents



Annexe 16 Architecture distribuée



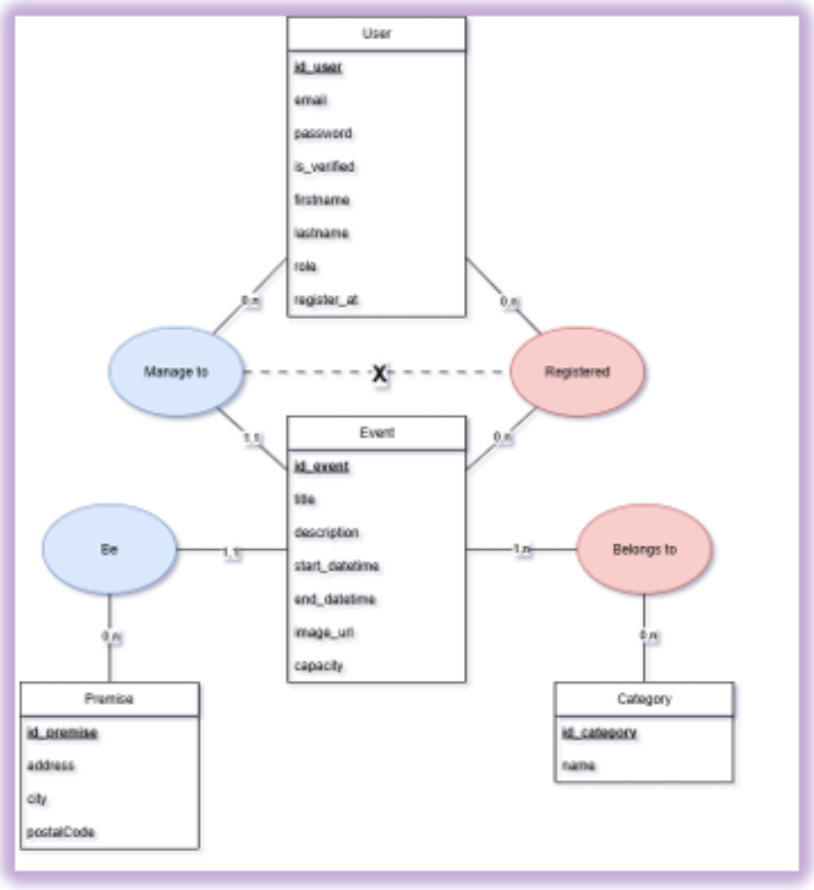
Annexe 17 Diagramme de classe UML



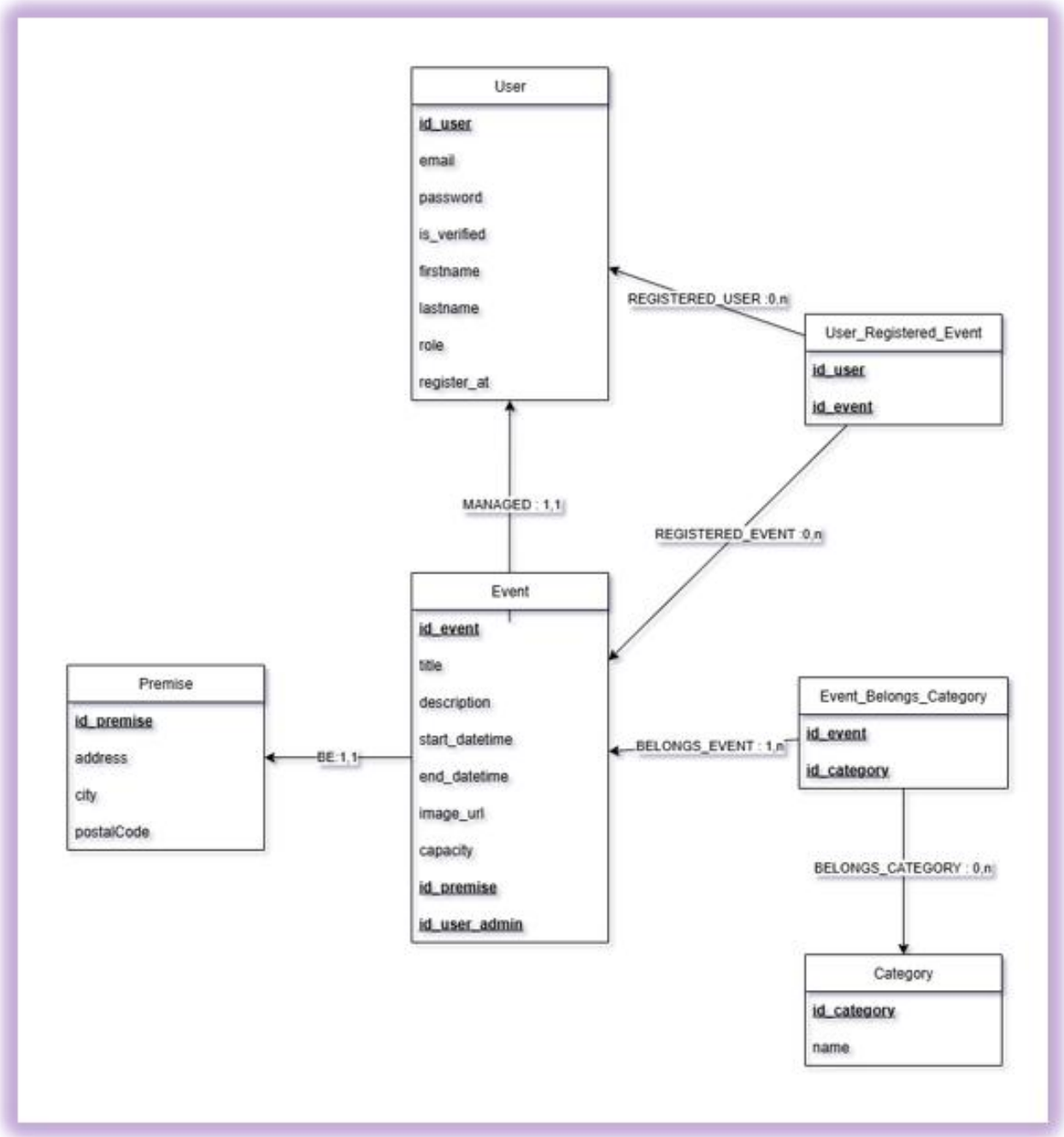
Annexe 18 Diagramme de séquence UML

Code mnémonique	Désignation	Type	Taille	Attributs	Remarques
id_user	Identifiant numérique d'un utilisateur	N		unsigned	Clé primaire AUTO INCREMENT
email	Email d'un utilisateur	A	100		NOT NULL, UNIQUE
password	Mot de passe d'un utilisateur	A	255		NOT NULL
is_verified	Boolean d'un compte vérifié	Booléen			DEFAULT(FALSE)
first_name	Prenom d'un utilisateur	A	100		NOT NULL
last_name	Nom de famille d'un utilisateur	A	100		NOT NULL
roles	Roles d'un utilisateur	JSON			DEFAULT(["ROLE_USER"])
createdAt	Date de création d'un compte utilisateur	DateTime			
id_event	Identifiant numérique d'un évènement	N		unsigned	Clé primaire AUTO INCREMENT
title	Titre d'un évenement	A	255		NOT NULL, UNIQUE
description	Description d'un événement	AN			
start_datetime	Début d'un évenement	DateTime			NOT NULL
end_datetime	Fin d'un évenement	DateTime			NOT NULL
image_url	URL d'une image d'un évenement	AN	255		NOT NULL
capacity	Capacité maximale d'un événement	N			> 0
id_premise	Identifiant numérique d'un local	N		unsigned	Clé étrangère vers Premise
id_user_admin	Identifiant numérique d'un utilisateur admin	N		unsigned	Clé étrangère vers User
id_premise	Identifiant numérique d'un local	N		unsigned	Clé primaire AUTO INCREMENT
address	Adresse du local	A	255		NOT NULL
city	Ville du local	A	100		NOT NULL
postal_code	Code postal du local	AN	50		NOT NULL
id_category	Identifiant numérique d'une catégorie	N		unsigned	Clé primaire AUTO INCREMENT
name	Nom d'une catégorie	A	100		NOT NULL, UNIQUE

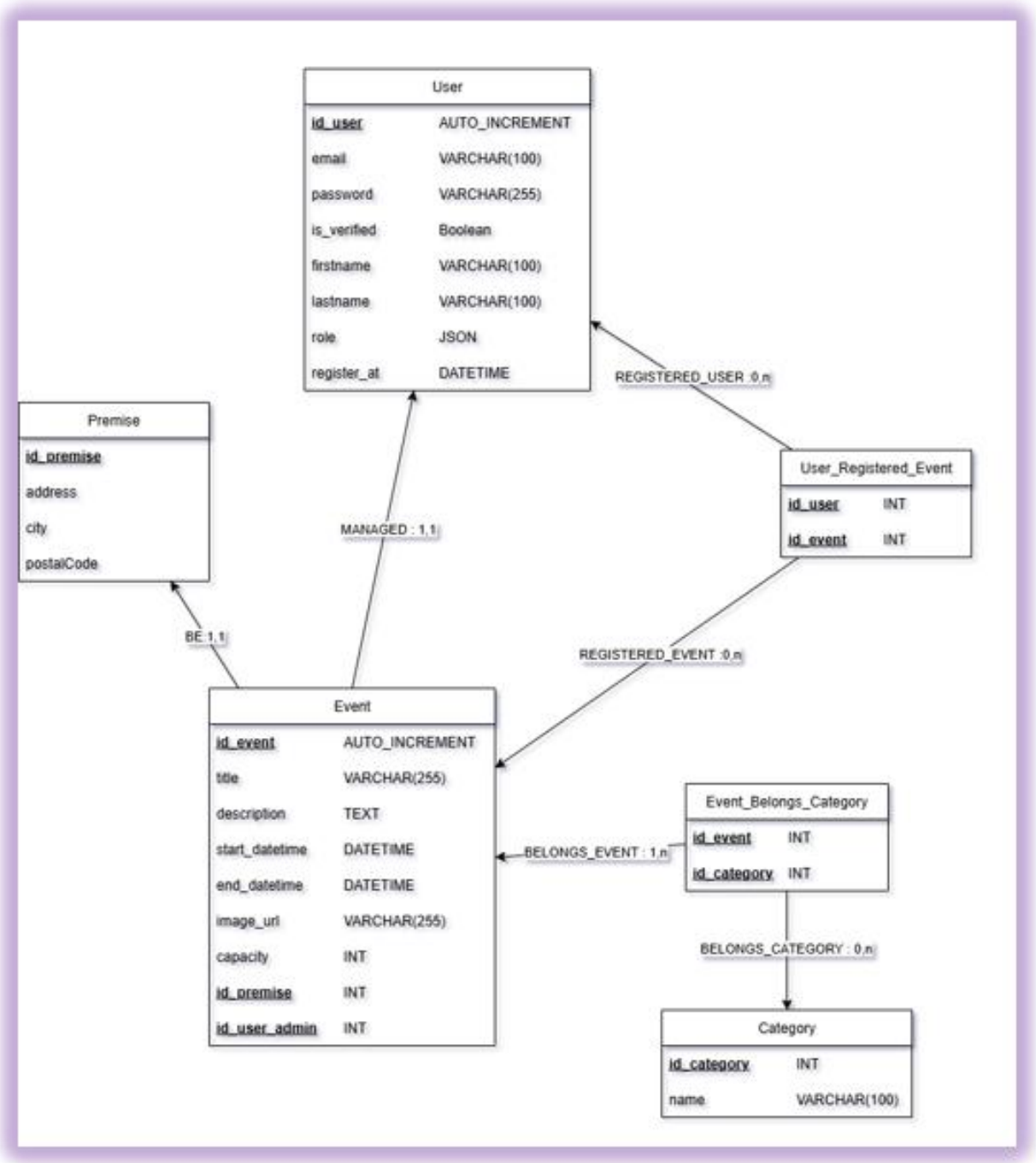
Annexe 19 Dictionnaire des données OurEvents



Annexe 20 Modèle Conceptuel de données



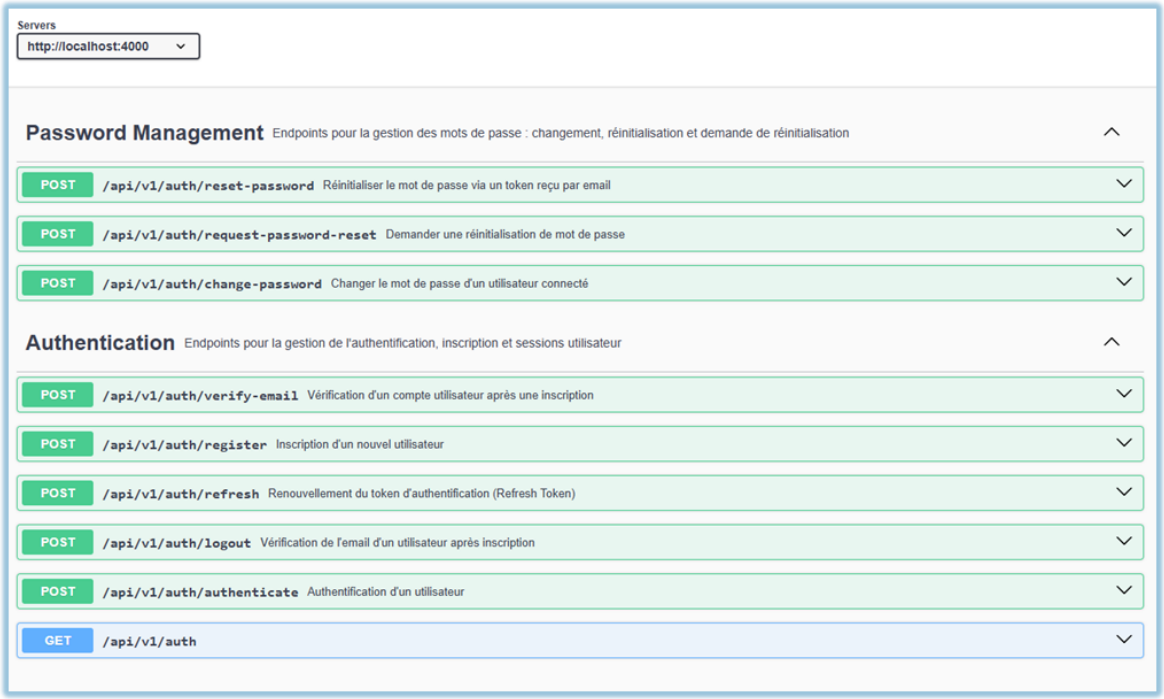
Annexe 21 Modèle Logique de données



Annexe 22 Modèle Physique de données

```java
@Test
void register_ShouldCreateNewUser_WhenEmailIsNotUsed() {

    // ARRANGE : Préparation des données
    when(userRepository.findByEmail(registerRequest.getEmail()))
            .thenReturn(Optional.empty());

    when(passwordEncoder.encode(registerRequest.getPassword()))
            .thenReturn("encodedPassword");

    when(userRepository.save(any(User.class)))
            .thenReturn(testUser);

    when(emailVerificationRepository.save(any(EmailVerificationToken.class)))
            .thenReturn(new EmailVerificationToken());

    // ACT : Exécution du service à tester
    ApiResponse<RegisterResponse> response = authService.register(registerRequest);

    // ASSERT : Vérification des résultats
    assertNotNull(response);
    assertTrue(response.isSuccess());
    assertEquals(201, response.getStatus());
    assertEquals("Inscription réussie", response.getMessage());
    assertNotNull(response.getData());
    assertEquals(registerRequest.getEmail(), response.getData().getEmail());

    // Vérification que les méthodes ont bien été appelées
    verify(userRepository, times(1)).findByEmail(registerRequest.getEmail());
    verify(userRepository, times(1)).save(any(User.class));
    verify(emailVerificationRepository, times(1)).save(any(EmailVerificationToken.class));
}
```

Annexe 23 Test unitaire Register



Annexe 24 Documentation OpenAPI/Swagger du service Auth

```yaml
name: Reusable CI Workflow

on:
    workflow_call:
        inputs:
            service-path:
                required: true
                type: string
        secrets:
          SONAR_TOKEN:
            required: true
            description: "SonarCloud authentication token"

jobs:
    ci:
        runs-on: ubuntu-latest
        steps:
            - name: Checkout code
              uses: actions/checkout@v4

            - name: Set up JDK 25
              uses: actions/setup-java@v4
              with:
                  distribution: 'temurin'
                  java-version: '25'
                  cache: 'maven'

            - name: Cache SonarQube packages
              uses: actions/cache@v4
              with:
                path: ~/.sonar/cache
                key: ${{ runner.os }}-sonar
                restore-keys: ${{ runner.os }}-sonar

            - name: Execute Test and Code Quality Analysis
              env:
                SONAR_TOKEN: ${{ secrets.SONAR_TOKEN }}
              run: |
                cd ${{ inputs.service-path }}
                mvn verify org.sonarsource.scanner.maven:sonar-maven-plugin:sonar jacoco:check \
                    --file pom.xml \
                    -DskipTests \
                    -Dsonar.projectKey=Mzk-Ali_app_ms_simswap \
                    -Dsonar.host.url=https://sonarcloud.io \
                    -Dsonar.token=$SONAR_TOKEN
            - name: Build and Test Service
              run: |
                cd ${{ inputs.service-path }}
                mvn clean verify -DskipTests
```

Annexe 25 Pipeline CI réutilisable

```yaml
name: Reusable CD Workflow

on:
    workflow_call:
        inputs:
            service:
                required: true
                type: string
            image-tag:
                required: true
                type: string
                default: 'latest'
            latest-tag:
                required: false
                type: string

jobs:
    cd:
        runs-on: ubuntu-latest
        steps:
            - name: Checkout code
              uses: actions/checkout@v4

            - name: Set up Docker Buildx
              uses: docker/setup-buildx-action@v4

            - name: Cache Docker layers
              uses: actions/cache@v4
              with:
                path: /tmp/.buildx-cache
                key: ${{ runner.os }}-docker-${{ inputs.service }}-${{ inputs.image-tag }}
                restore-keys: |
                  ${{ runner.os }}-docker-${{ inputs.service }}-

            - name: Build Docker Image with Cache
              run: |
                cd ${{ inputs.service }}
                docker buildx build \
                    --cache-from=type=local,src=/tmp/.buildx-cache \
                    --cache-to=type=local,dest=/tmp/.buildx-cache-new \
                    -tag ghcr.io/${{ github.repository_owner }}/${{ inputs.service }}:${{ inputs.image-tag }} \
                    --load .

            - name: Login to GitHub Container Registry
              uses: docker/login-action@v3
              with:
                  registry: ghcr.io
                  username: ${{ github.repository_owner }}
                  password: ${{ secrets.GITHUB_TOKEN }}

            - name: Scan Image with Trivy
              uses: aquasecurity/trivy-action@master
              with:
                image-ref: ghcr.io/${{ github.repository_owner }}/${{ inputs.service }}:${{ inputs.image-tag }}
                format: 'table'
                ignore-unfixed: true

            - name: Push Docker Image
              run: |
                docker push ghcr.io/${{ github.repository_owner }}/${{ inputs.service }}:${{ inputs.image-tag }}

            - name: Tag and push as latest-tag
              if: ${{ inputs.latest-tag != '' }}
              run: |
                echo "Retagging image as '${{ inputs.latest-tag }}'"
                docker tag ghcr.io/${{ github.repository_owner }}/${{ inputs.service }}:${{ inputs.image-tag }} \
                           ghcr.io/${{ github.repository_owner }}/${{ inputs.service }}:${{ inputs.latest-tag }}

                docker push ghcr.io/${{ github.repository_owner }}/${{ inputs.service }}:${{ inputs.latest-tag }}

            - name: Deploy to development environment
              run: |
                echo "Deploying ${{ inputs.service }} with tag ${{ inputs.image-tag }} to development environment"
```

Annexe 26 Pipeline CD réutilisable