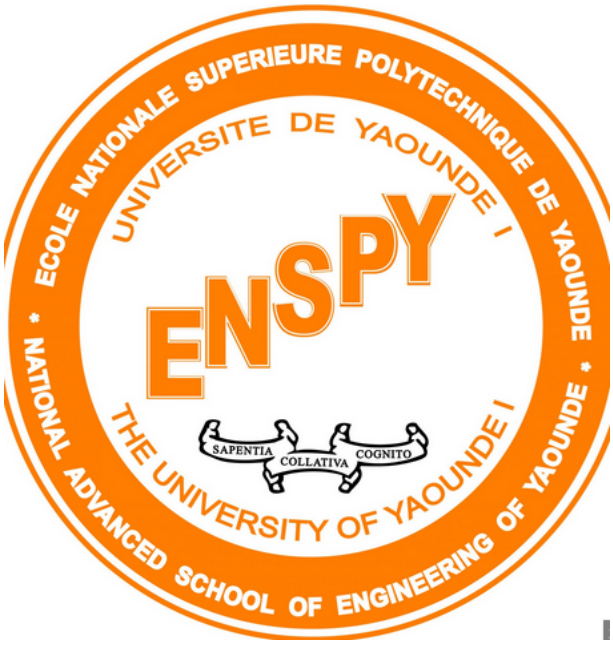

Rapport de Développement

Conception d'une application de Gestion d'Événements

Fait par : AKOULOUZE MANY Eva Félicia Meg
Encadrant : Dr. KUNGNE
Date : 26 mai 2025

Table des matières

1 Introduction

Ce rapport présente le développement d'une application de gestion d'événements. Celle-ci permet à des organisateurs de créer et gérer des événements, et à des participants de ceux-ci de s'y inscrire. Pour sa mise en place, il est question pour nous de nous baser sur la programmation orientée objet avec Java, et implémente des concepts tels que l'héritage, les interfaces, les collections, et le patron Singleton.

2 Objectifs du projet

— Permettre la création d'événements (conférences, concerts).
— Gérer les inscriptions et suppressions de participants.
— Gérer les organisateurs et les événements qu'ils organisent.
— Implémenter une interface de notification. .

3 Modélisation des Classes

Voici un aperçu des principales classes et de leurs responsabilités :

3.1 Classe abstraite : Evenement

— Attributs : id, nom, date, lieu, capaciteMax.
— Méthodes : ajouterParticipant(), annuler(), afficherDetails().

3.2 Sous-classes

Conference
— Attributs spécifiques : theme, intervenants (List<Intervenant>).
Concert
— Attributs spécifiques : artiste, genreMusical.

3.3 Participant

— Attributs : id, nom, email.

3.4 Organisateur (hérite de Participant)

— Attribut : evenementsOrganises (List<Evenement>).

3.5 NotificationService (interface)

— Méthode : envoyerNotification(String message).

3.6 GestionEvenements (Singleton)

— Attribut : evenements (Map<String, Evenement>).
— Méthodes : ajouterEvenement(), supprimerEvenement(), rechercherEvenement().

4 Diagramme de contexte

Ce diagramme permet principalement de permettre de visualiser clairement qui utilise l'application et comment ils interagissent avec le système de gestion d'événements, sans entrer dans les détails techniques internes. Dans notre cas, notre diagramme ce présente comme suit :

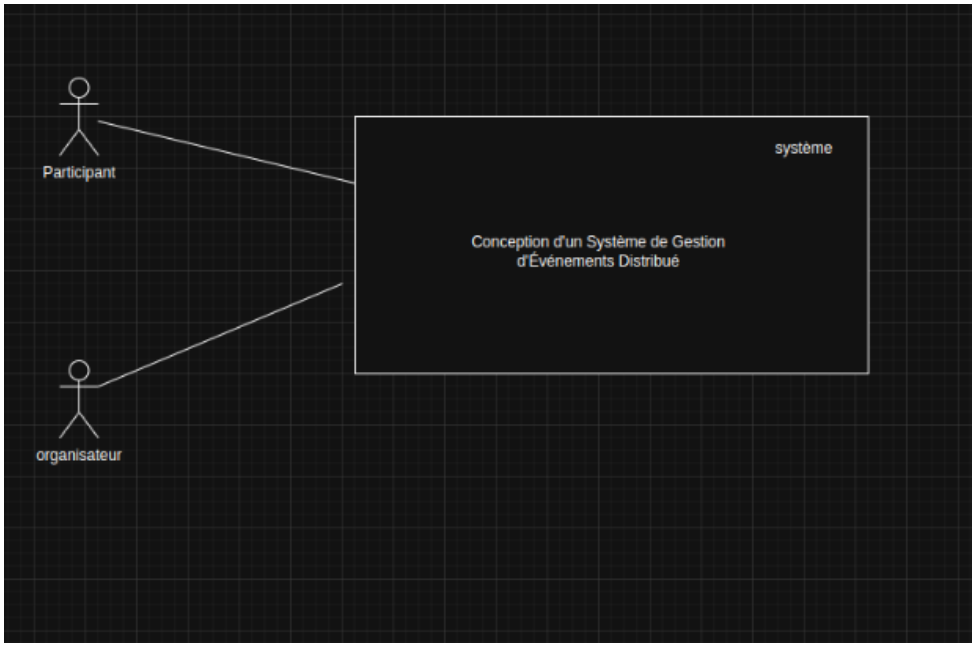

FIGURE 1 – Diagramme de contexte

5 Diagramme de package

Ce diagramme joue un rôle fondamental dans l'organisation logique et la structuration modulaire de ton système. Dans notre cas, notre diagramme ce présente comme suit :

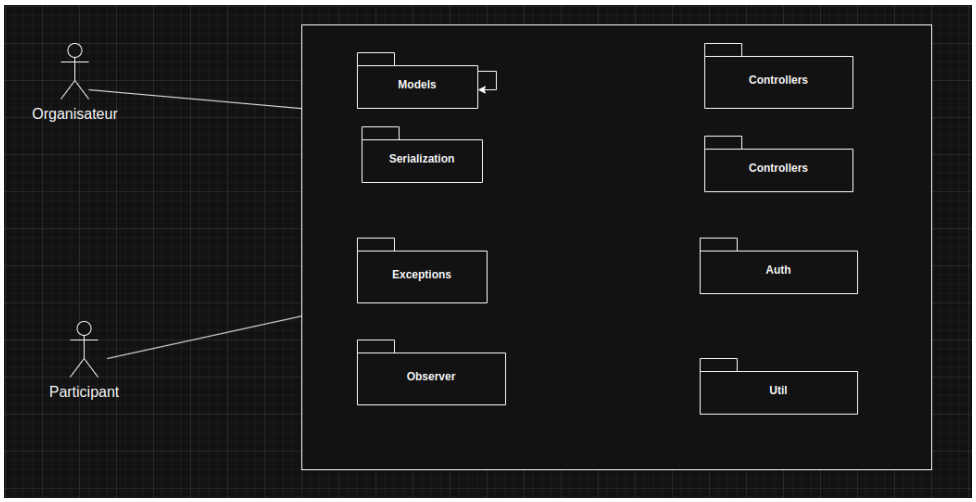

FIGURE 2 – Diagramme de package

6 Diagramme de classe

Ce diagramme formalise la structure des objets métier, et sert de plan pour l'implémentation du système. C'est à la fois un outil de modélisation et de communication.. Dans notre cas, notre diagramme ce présente comme suit :

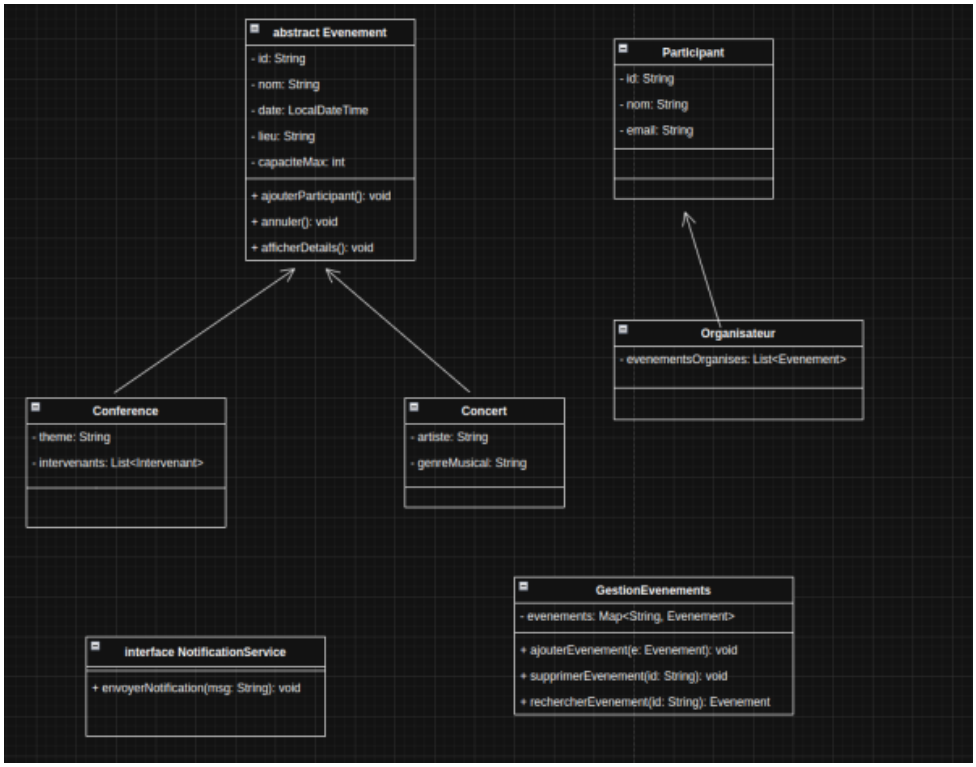

FIGURE 3 – Diagramme de classe

7 Diagramme de cas d'utilisation

Ce diagramme permet d'identifier les fonctionnalités principales du système du point de vue des utilisateurs. Il décrit les interactions entre les acteurs (comme l'organisateur ou le participant) et le système à travers des scénarios fonctionnels.. Dans notre cas, notre diagramme ce présente comme suit :

FIGURE 4 – Diagramme de cas d'utilisation

8 Architecture et Design

8.1 Concepts utilisés

- **Héritage** : pour différencier les types d'événements.
- **Interface** : NotificationService permet d'implémenter différents canaux (mail, SMS...).
- **Singleton** : GestionEvenements assure une instance globale pour gérer les événements.
- **Collections Java** : List, Map utilisées pour gérer les données dynamiquement.

8.2 Structure du projet (Packages proposés)

- **modele** : classes métiers (Evenement, Participant, Organisateur...)
- **services** : NotificationService, GestionEvenements.
- **main / ui** : point d'entrée, interface console ou GUI.

9 Fonctionnalités de l'application

- Création et suppression d'un événement par un organisateur.

— Inscription et désinscription à un événement par un participant.
— Affichage des détails d'un événement.
— Notification des participants.
— Recherche d'événements par identifiant.

10 Conclusion

Ce projet illustre efficacement l'application des principes fondamentaux de la programmation orientée objet pour concevoir un système structuré, modulaire et maintenable. Grâce à cette architecture, l'application est aisément extensible, que ce soit par l'intégration d'une base de données ou le développement d'une interface graphique, ouvrant ainsi la voie à une solution logicielle complète et évolutive.

Fin du rapport