Ecole Nationale Supérieure
d'Informatique et d'Analyse
des Systèmes - RABAT

Projet
Plateformes de Développement

Plateforme de Dons

Réalisé par :
BANOUQ Mohammed Yassine
CHADIFI Ayoub
ELOKRI Amine

Encadré par :
Pr. EL HAMLAOUI Mahmoud

Année Universitaire 2022-2023

Résumé

Le présent document résume notre travail académique, réalisé dans le cadre du projet de Plates-formes de développement intitulé "Mise en place d'une plateforme de dons", pour le module d'Industrialisation logicielle. Notre projet, baptisé "Yaa Khayr", poursuit l'objectif ambitieux de concevoir une application web sophistiquée permettant de collecter des dons en ligne pour des organismes de bienfaisance divers.

Cette solution novatrice facilite les dons grâce à un processus de communication direct et bilatéral entre les donateurs et les organisations de bienfaisance, en offrant une plateforme technologique de qualité supérieure pour soutenir cette noble cause. Ainsi, toute personne souhaitant contribuer peut le faire en partageant ses connaissances et son expertise et en faisant un don à l'organisation de son choix.

Notre mission, ambitieuse, est de concevoir, d'implémenter et de tester une plateforme de dons permettant la gestion, le contrôle et la validation des données collectées. L'objectif ultime est de rendre la plateforme "Yaa Khayr" la plus compétitive, efficace et adaptée aux besoins du marché. Nous avons travaillé avec acharnement tout au long du développement de notre projet, en nous appuyant principalement sur les services proposés par JEE pour créer une solution technologique de pointe dans le domaine des dons en ligne.

Abstract

This document summarizes our academic work carried out as part of the Platforms Development project entitled "Establishment of an online donation platform" for the Software Industrialization module. Our project, named "Yaa Khayr", aims to develop a sophisticated web application for collecting online donations for various charitable organizations.

This innovative solution facilitates donations through a direct and bilateral communication process between donors and charitable organizations, offering a superior quality technological platform to support this noble cause. Thus, anyone willing to contribute can do so by sharing their knowledge and expertise and making a donation to the organization of their choice.

Our ambitious mission is to design, implement, and test a donation platform that enables the management, control, and validation of collected data. The ultimate goal is to make the "Yaa Khayr" platform the most competitive, efficient, and suitable for the market needs. We worked tirelessly throughout the development of our project, relying mainly on the services offered by JEE to create a state-of-the-art technological solution in the field of online donations.

Table des Matières

Table des figures

Liste des abreviations

DAO *Data Access Object*

DTO *Data Transfer Objects*

JEE *Java Entreprise Edition*

IDE *Integrated Development Environment*

MLD *Modèle logique des données*

MVC *Modèle-Vue-Contrôleur*

MySQL *Structured Query Language*

SGBDR *système de gestion de base de données relationnellese*

Introduction

Le monde actuel est confronté à de nombreux défis sociaux, économiques et environnementaux, et le besoin d'aide et de solidarité est plus important que jamais. C'est dans ce contexte que le projet YaKhayr a été développé, dans le but de fournir une plateforme en *Java Entreprise Edition* (JEE) pour les personnes ayant besoin d'aide et pour ceux qui souhaitent apporter leur contribution en faisant des donations.

Le présent rapport détaille les différentes étapes de développement du projet YaKhayr, depuis l'analyse des besoins jusqu'à la mise en production. Dans ce rapport, nous décrirons les fonctionnalités de l'application, ainsi que les outils et technologies utilisés pour sa réalisation.

Ce rapport détaillera les différentes étapes de développement du projet YaKhay, organisé en trois chapitres :

• Le premier chapitre aborde le contexte général du projet. Il présente une étude préalable du sujet , une description de ce dernier puis la planification adoptée pour le réalisé.

• Le deuxième chapitre concerne la spécification des besoins et aborde l'analyse du sujet en suivant la démarche de modélisation et en respectant le cahier de charge fourni.

• Enfin, le dernier chapitre aborde la réalisation de notre travail et les technologies adoptée.

En somme, ce rapport présentera une vue d'ensemble complète du développement de l'application YaKhayr et servira de guide pour ceux qui souhaitent comprendre les différentes étapes de développement d'une application web en JEE.

Chapitre 1

Contexte général du projet

1.1 Introduction

Ce chapitre se concentre sur la présentation du contexte du projet, y compris la problématique qui a conduit à la conception et à la réalisation de cette application web, les objectifs à atteindre et le plan de gestion adopté pour assembler les différentes composantes du projet.

1.2 Problématique

La problématique du projet YaKhayr est liée aux difficultés rencontrées par les personnes nécessitant des donations ou de l'aide. En effet, ces personnes peuvent avoir du mal à trouver des donateurs ou des bienfaiteurs, ou peuvent être confrontées à des difficultés administratives ou logistiques pour obtenir l'aide dont elles ont besoin.

De plus, les donateurs et les bienfaiteurs peuvent également être confrontés à des défis tels que l'identification de personnes dignes de confiance et l'assurance que leur don sera utilisé de manière responsable.

Dans ce contexte, le projet YaKhayr vise à fournir une plateforme en ligne sécurisée et efficace pour faciliter les donations et les demandes d'aide, en mettant en relation les donateurs et les bénéficiaires de manière transparente et équitable.

1.3 Objectifs

Les objectifs du projet YaKhayr sont les suivants :

Faciliter les donations et les demandes d'aide : La plateforme vise à mettre en relation les donateurs et les bénéficiaires de manière efficace et transparente, en fournissant des fonctionnalités telles que la publication d'annonces, la recherche d'annonces et la création de comptes utilisateurs.

Assurer la transparence et la responsabilité : Le projet YaKhayr vise à garantir la transparence et la responsabilité dans le processus de donation et d'aide. Les bénéficiaires devront soumettre des informations vérifiables pour leurs besoins et les donateurs pourront suivre l'utilisation de leurs dons.

Favoriser la confiance entre les donateurs et les bénéficiaires : La plateforme YaKhayr vise à établir une relation de confiance entre les donateurs et les bénéficiaires en garantissant la vérification des annonces publiées et des informations des utilisateurs.

Contribuer à la résolution des problèmes sociaux et économiques : En facilitant les donations et les demandes d'aide, le projet YaKhayr vise à contribuer à la résolution des problèmes sociaux et économiques auxquels sont confrontées les personnes dans le besoin.

Fournir une solution durable et évolutive : Le projet YaKhayr vise à fournir une solution durable et évolutive qui répondra aux besoins des utilisateurs à long terme, en continuant à offrir de nouvelles fonctionnalités et en améliorant l'expérience utilisateur.

1.4 Contraintes

1.4.1 Les contraintes techniques

Le projet doit être développé en utilisant les technologies JEE (Java Enterprise Edition), et doit être compatible avec différents navigateurs web. De plus, la plateforme doit être sécurisée

pour garantir la protection des données et la prévention des attaques de pirates informatiques.

Les contraintes temporelles : Le projet doit être livré dans les délais impartis pour répondre aux besoins des utilisateurs et éviter les retards qui peuvent impacter la qualité de la solution.

1.4.2 Les contraintes légales

Le projet doit respecter les lois et les réglementations en vigueur concernant la collecte et l'utilisation de données personnelles.

1.4.3 Les contraintes liées aux utilisateurs

Le projet doit être facile à utiliser et convivial pour répondre aux besoins des utilisateurs, tout en fournissant une solution qui est accessible à tous, y compris aux personnes ayant des besoins spécifiques.

1.5 Solution

Les technologies JEE présentent une alternative solide et fiable pour la création d'applications web. Les divers outils et frameworks open-source proposés par l'environnement JEE peuvent s'avérer utiles pour simplifier le processus de développement et faciliter l'intégration de fonctionnalités multiples.

Il convient d'apporter une attention particulière à l'ergonomie de la plateforme ainsi qu'à l'accessibilité de son interface utilisateur. En effet, les exigences spécifiques des utilisateurs doivent être soigneusement prises en considération afin de garantir une accessibilité universelle de la solution.

1.6 Conduite du projet

1.6.1 Phases de conduite

L'élaboration de notre projet a suivi un processus en plusieurs étapes, qui a débuté par une étude théorique préliminaire. Nous avons ensuite procédé à une analyse des besoins et spécifications, consistant en une dissociation des diverses métriques du projet, afin de pouvoir établir un cahier des charges complet et rigoureux.

La phase suivante a été la conception du système, qui a consisté à définir les modalités concrètes de réalisation du projet, en répondant notamment à la question fondamentale du "comment" : comment répondre aux spécifications établies en analysant le système futur, à travers notamment la prise en compte des données, des fonctions logicielles, de la représentation du comportement et de la hiérarchisation.

La familiarisation avec la technologie utilisée a également constitué une étape primordiale de notre processus, dans la mesure où chaque projet présente sa spécificité. Il était donc essentiel de se documenter sur les outils d'implémentation, le langage de programmation web et les frameworks à utiliser.

Nous avons ensuite procédé à l'implémentation du système, après avoir spécifié l'ensemble des paramètres primordiaux pour une compréhension détaillée du système à réaliser. Cette étape a permis de passer de la théorie à la réalité, en procédant à la mise en place effective du projet.

Enfin, nous avons effectué les réglages des paramètres globaux du projet et procédé à son exportation. Cela a impliqué de réaliser des ajustements sur les interfaces utilisateur, de vérifier la corrélation et la cohérence du site, de guider l'utilisateur et de contrôler ses actions, d'assurer l'adaptabilité de l'application, de gérer les erreurs, de garantir l'homogénéité et la cohérence, et enfin de s'assurer de la compatibilité du projet.

1.6.2 Cycle de vie

Le modèle en V été le meilleur choix pour le cycle de vie de notre projet. Cela afin d'assurer une gestion efficace et rigoureuse. Ce modèle met en avant la nécessité de tirer des enseigne-

ments de chaque phase montante afin de nourrir et améliorer les phases en aval.

En effet, en cas de détection de défauts, les phases de la partie ascendante sont chargées de fournir les informations nécessaires pour améliorer le logiciel. Pour cela, chaque phase du projet démarre par une réunion de lancement, particulièrement importante pour les projets sensibles.

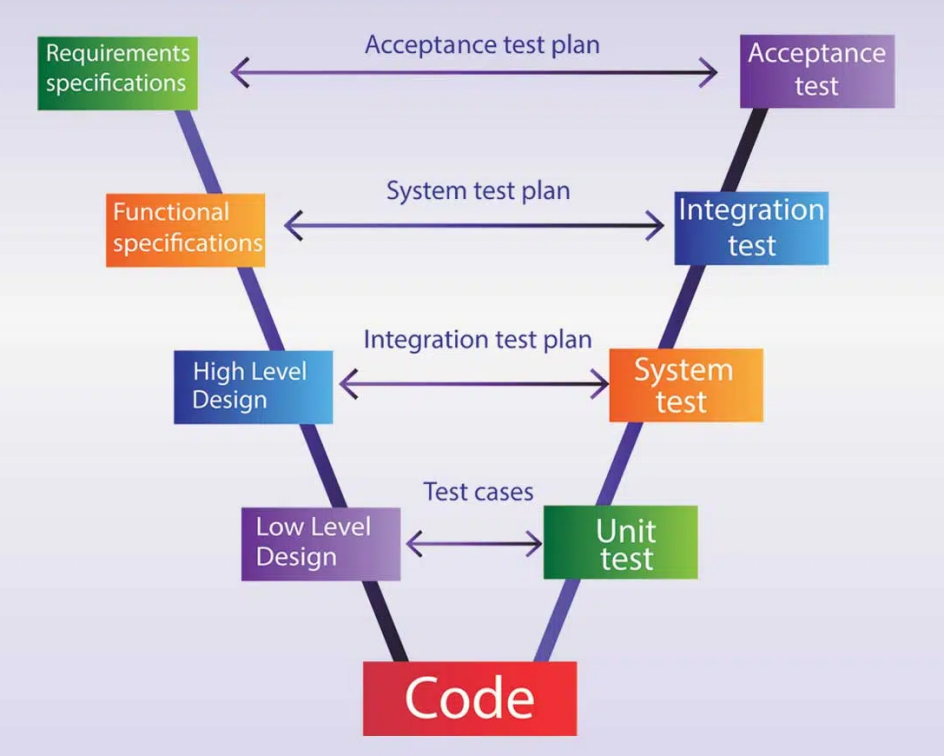

FIGURE 1.1 – V Model.

Par ailleurs, le modèle en V souligne l'importance de préparer et anticiper les attentes des futures phases, dès les étapes descendantes. Ainsi, les attentes des tests de validation sont définies dès les spécifications, alors que celles des tests unitaires sont précisées lors de la conception.

En somme, suivre le modèle en V permet d'assurer une gestion méthodique et structurée de notre projet.

1.7 Conclusion

Au fil de ce chapitre, nous avons abordé le cadre global de notre projet en commençant par présenter le sujet, en découlant la problématique qui en découle. Nous avons ensuite exposé la solution envisagée ainsi que les objectifs et les contraintes associées. Dans le chapitre à venir, nous allons nous pencher sur l'analyse et la conception de notre système avec davantage de détails.

Chapitre 2

Analyse et Conception

2.1 Introduction

Au sein de ce chapitre, nous allons mettre en lumière les besoins fonctionnels qui sous-tendent la conception de notre application. Dans un premier temps, nous aborderons l'analyse du système. Dans un second temps, nous nous concentrerons sur la phase de conception en détaillant la démarche de modélisation.

2.2 Analyse des besoins

2.2.1 Besoins fonctionnels

Les besoins fonctionnels, tels que seront décrits , décrivent les fonctionnalités attendues du système. Ils se concentrent sur ce que le système doit être capable de faire.

Voici les principales fonctionnalités identifiées lors de l'analyse :

La création de profil : Les utilisateurs doivent pouvoir créer un profil sur la plateforme pour pouvoir publier des annonces de dons ou de besoins.

La publication d'annonces : Les utilisateurs doivent pouvoir publier des annonces de dons

ou de besoins, en fournissant des informations détaillées sur les articles proposés ou les besoins exprimés.

La recherche d'annonces : Les utilisateurs doivent pouvoir rechercher des annonces en fonction de différents critères, tels que la catégorie d'articles, la localisation, etc.

La gestion des annonces : Les utilisateurs doivent pouvoir gérer leurs annonces, par exemple en les supprimant ou en les modifiant.

Commentaires : Les utilisateurs doivent pouvoir communiquer entre eux via les commentaires pour avoir plus d'informations

La gestion des utilisateurs : Les administrateurs de la plateforme doivent pouvoir gérer les comptes des utilisateurs, en supprimant les comptes inactifs ou en suspendant les comptes qui ne respectent pas les conditions d'utilisation de la plateforme.

La gestion des notifications : Les utilisateurs de la plateforme doivent pouvoir avoir les notifications d'articles qu'ils leurs appartiennent.

L'authentification et l'autorisation : La plateforme doit garantir l'authentification et l'autorisation des utilisateurs pour garantir la sécurité et la confidentialité des données.

2.2.2 Besoins non fonctionnels

Les besoins non fonctionnels, en revanche, décrivent les contraintes auxquelles le système doit répondre. Ils se concentrent sur la manière dont le système doit fonctionner. Voici les principaux besoins non fonctionnels identifiés lors de l'analyse du projet :

La fiabilité : Le système doit être fiable et fonctionner de manière cohérente. Les données doivent être stockées de manière fiable et les utilisateurs doivent être en mesure de publier des annonces sans risque de perte de données.

La convivialité : Le système doit être facile à utiliser pour les utilisateurs, avec une interface intuitive et ergonomique.

Les performances : Le système doit être performant et réactif, en garantissant des temps de réponse rapides pour les requêtes des utilisateurs.

La compatibilité : Le système doit être compatible avec différents navigateurs Web et différents types de terminaux (ordinateurs, smartphones, tablettes).

La modularité : Le système doit être conçu de manière modulaire, en permettant l'ajout ou la suppression de fonctionnalités en fonction des besoins futurs.

En résumé, l'analyse des besoins de YaKhayr comprend à la fois des besoins fonctionnels et des besoins non fonctionnels, qui doivent être pris en compte dans la conception de la plateforme.

2.3 Démarche de modélisation

UML (Unified Modeling Language) est un language de modélisation, il présente 9 diagrammes principaux. Ces diagrammes sont utilisés dans les différents processus de développement du système depuis la découverte du besoin du système jusqu'a l'analyse :

Dans ce travail nous utiliserons seulement les diagrammes suivants pour la modélisation de notre projet :

— Construction du diagramme de cas d'utilisation ;

— Construction du diagramme de classe ;

2.3.1 Diagramme de cas d'utilisation

Le diagramme de cas d'utilisation est est l'un des diagrammes fonctionnels. Il permet de recueillir, d'analyser et d'organiser les besoins. Il constitue alors la première étape d'une modélisation UML.

Tout d'abord, il y a deux types d'utilisateurs : les "Associations" et les "Individus", qui ont chacun leurs fonctionnalités spécifiques. Ils sont représentés dans le diagramme par des acteurs reliés aux cas d'utilisation. Les deux heritent de l'acteur "User".

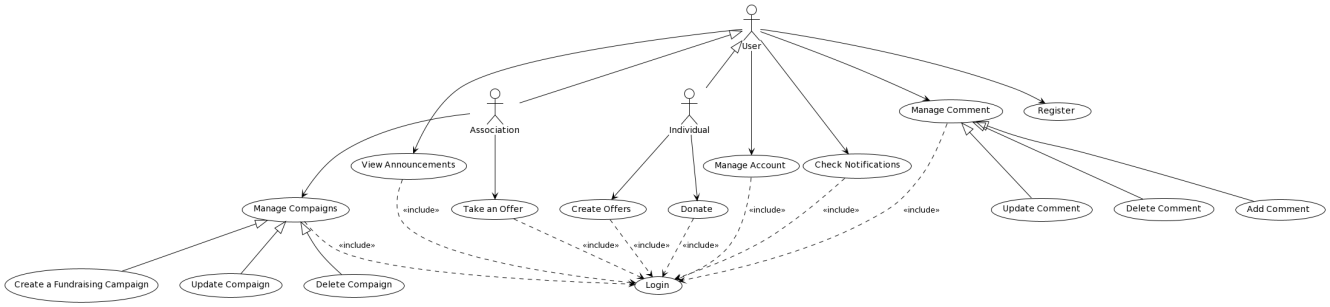

FIGURE 2.1 – Diagramme de cas d'utilisation .

Cas d'utilisation de l'utilisateur :

— S'inscrire (Register) : Permet à un utilisateur de créer un compte sur le système.

— Gérer des commentaires (Manage Comment) : Permet à un utilisateur de gérer les commentaires qu'il a publiés sur le système, en ajoutant, mettant à jour ou supprimant des commentaires.

— Gérer son compte (Manage Account) : Permet à un utilisateur de gérer les informations de son compte utilisateur, comme le nom, l'adresse email, etc.

— Vérifier les notifications (Check Notifications) : Permet à un utilisateur de voir les notifications qu'il a reçues, telles que les messages reçus, les commentaires ajoutés, etc.

— Voir les annonces (View Announcements) : Permet à un utilisateur de voir les annonces publiées sur le système.

Cas d'utilisation de l'association :

— Gérer des campagnes (Manage Campaigns) : Permet à une association de créer et de gérer des campagnes de collecte de fonds sur le système.

— Prendre une offre (Take an Offer) : Permet à une association d'accepter des offres de dons proposées par des individus sur le système.

— Gérer les publications de besoins (Manage Need Posts) : Permet à une association de gérer les publications de besoins qu'elle a publiées sur le système, en ajoutant, mettant à jour ou supprimant des publications.

Cas d'utilisation de l'individu :

— Faire un don (Donate) : Permet à un individu de donner de l'argent à une association sur

le système.

— Créer des offres (Create Offers) : Permet à un individu de proposer des offres de dons à une association sur le système.

2.3.2 Diagramme de classe

Le diagramme de classe est un outil UML qui permet de modéliser la structure du système en termes d'objets, de classes, d'attributs et de méthodes. Il permet de représenter les différentes entités du système et les relations entre elles.

Pour YaKhayr, le diagramme de classe inclut les classes suivantes :

— Utilisateur : Cette classe représente les utilisateurs du système, qu'ils soient des individus ou des associations.

— Association : Cette classe représente les associations. Elles peuvent créer des campagnes de collecte de dons.

— Don : représente les annonces publiées sur la plateforme.

— Offert : représente les dons effectués par les utilisateurs de la plateforme.

— Campagne : représente les besoins exprimés par les Campagnes de la plateforme.

— Individu : représente les utilisateurs de la plateforme qui peuvent effectuer un don.

— Commentaire : Cette classe représente les commentaires qui peuvent être ajoutés, mis à jour ou supprimés par les utilisateurs.

— Notification : Cette classe représente les notifications que les utilisateurs peuvent consulter.

— Campagne : Cette classe représente les campagnes que les associations peuvent gérer, poster, mettre à jour ou supprimer.

Les relations entre ces classes sont les suivantes :

— Association et Individu sont des utilisateurs.

— Un Don peut être publiée par un utilisateur.

— Un don peut être effectué par un Individu sur une annonce.

— Un besoin peut être exprimé par une association.

— Une association peut créer des campagnes de collecte de dons.

— Une association peut prendre des dons.

— Un utilisateur peut commenter un don.

— Un utilisateur est notifié.

— Un individu peut poser un offre.

— Un individu peut donner a une campagne de dons.

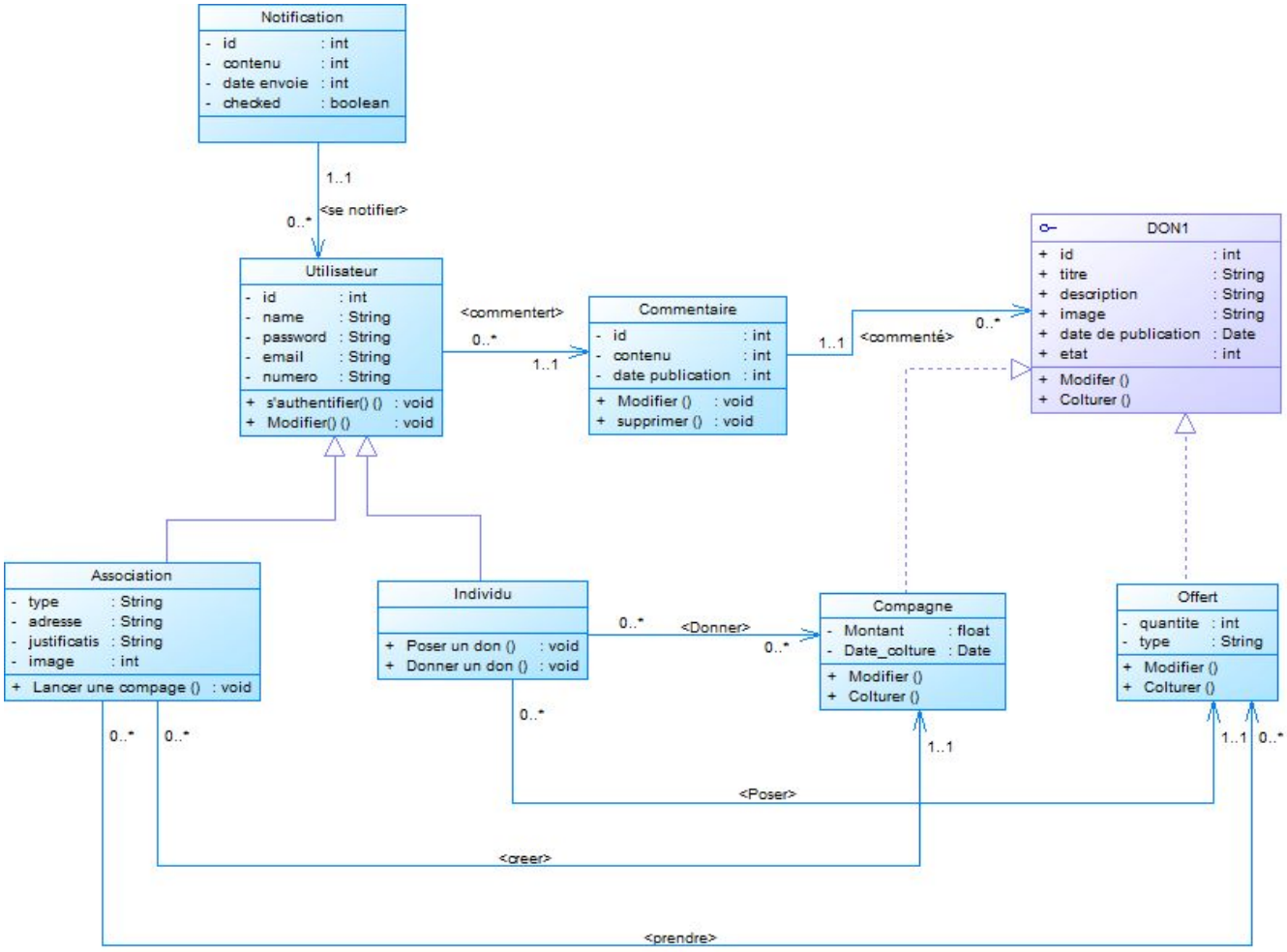

FIGURE 2.2 – Diagramme de classe .

A titre d'information, on a utilisé le design pattern Factory Method pour la création d'annonces (Don) de deux types différents : Offre (Classe Offert) et Demande (Classe Campagne). L'utilisation du design pattern Factory Method permet de créer des objets de types différents sans avoir à modifier le code existant de la classe AnnonceFactory, tout en respectant le principe

d'encapsulation et de modularité.

2.4 Spécifications techniques

Lors de la phase de spécifications techniques, l'architecture logicielle du projet doit être définie. Pour notre plateforme, l'architecture adoptée est une architecture client/serveur avec une architecture *Modèle-Vue-Contrôleur* (MVC) .

2.4.1 Spring Boot Flow Architecture

Spring Boot utilise tous les modules de Spring comme Spring MVC, Spring Data, etc. L'architecture de Spring Boot est la même que celle de Spring MVC.

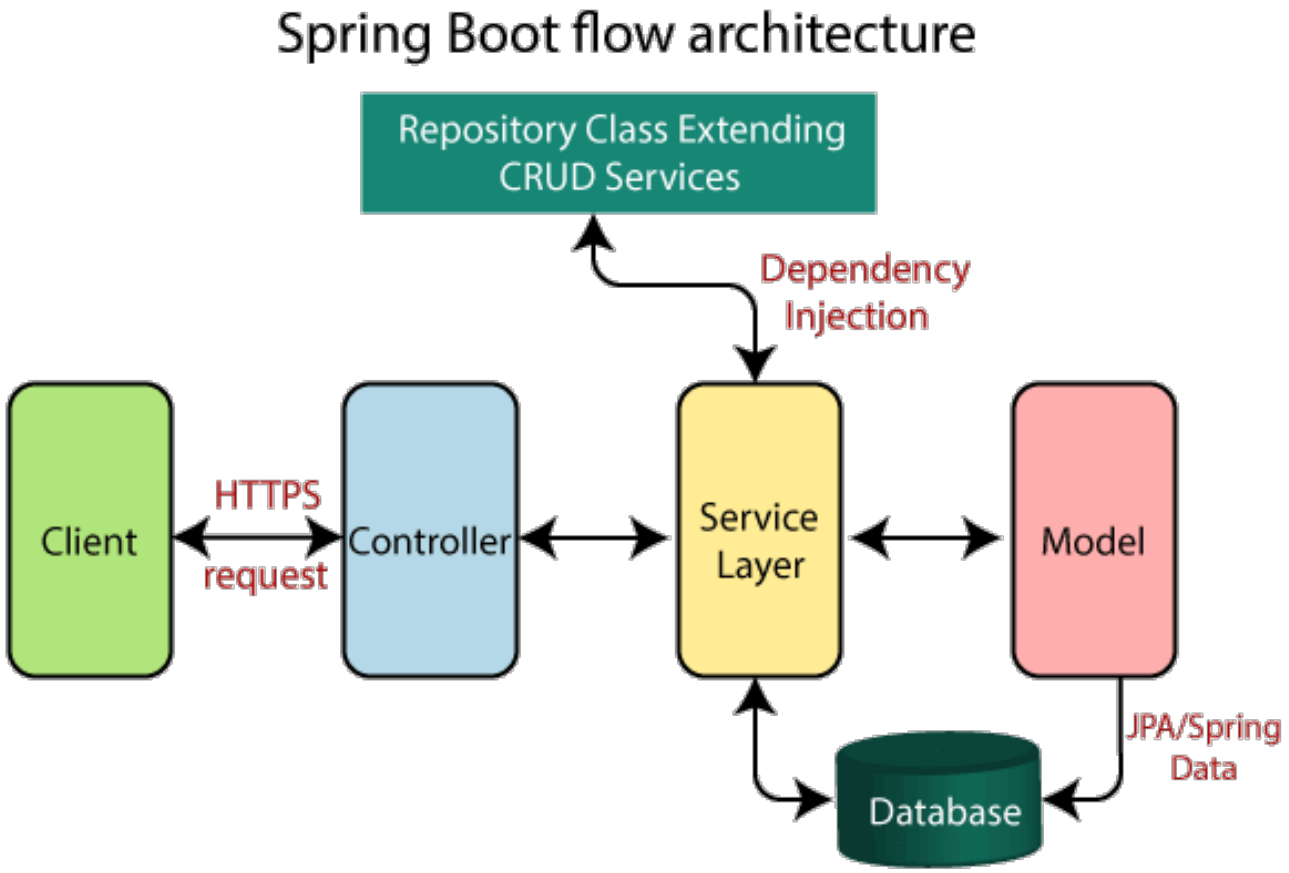

FIGURE 2.3 – Spring Boot Flow Architecture.

Il crée une couche d'accès aux données et effectue des opérations CRUD. Le client effectue des requêtes HTTP (PUT ou GET). La requête est envoyée au contrôleur, qui la mappe et la gère. Ensuite, il appelle la logique de service si nécessaire. Dans la couche de service, toute la logique métier est effectuée. Elle effectue la logique sur les données qui sont mappées sur JPA avec des classes de modèle. Une page JSP est renvoyée à l'utilisateur si aucune erreur ne s'est produite [1].

Dans le cas de notre plateforme de collecte de dons, lorsqu'un client envoie une requête HTTP (GET, PUT, POST, etc.), celle-ci est envoyée au Contrôleur qui la traite en la mappant et en appelant la logique serveur si nécessaire. La logique métier est ensuite effectuée dans la couche de Service où Spring Boot effectue toutes les opérations sur les données de la base de données, qui sont mappées à la classe de modèle Spring Boot via la bibliothèque Java Persistence (JPA). Enfin, le contrôleur renvoie une page JSP en tant que réponse.

2.4.2 L'architecture MVC

L'architecture MVC est une architecture logicielle qui permet de séparer les différentes composantes d'une application. Cette architecture se compose de trois éléments principaux :

1. Le modèle : représente les données et la logique de l'application.
2. La vue : représente l'interface utilisateur de l'application.
3. Le contrôleur : représente la logique de contrôle de l'application, en recevant les demandes des utilisateurs et en les traitant.

Dans le cas de YaKhayr, l'architecture MVC est utilisée pour séparer la logique de l'application, l'interface utilisateur et la gestion des demandes des utilisateurs. Les différents éléments du système pourraient être organisés comme suit :

1. Le modèle : représente les données de l'application et la logique qui leur est associée. Cette partie de l'application peut inclure une base de données qui stocke les informations sur les utilisateurs, les annonces, les dons, les besoins, etc.

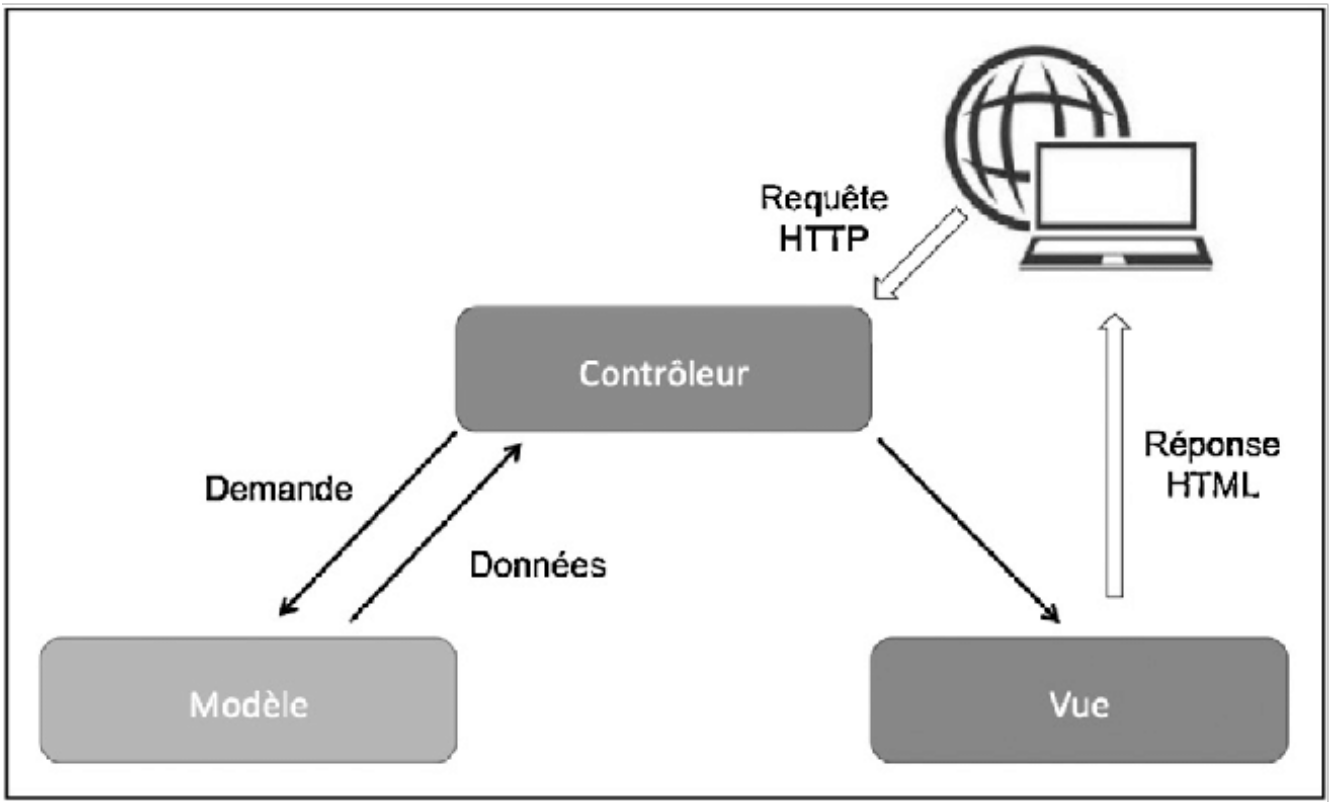

FIGURE 2.4 – Architecture MVC.

2. La vue : représente l'interface utilisateur de l'application. Cette partie de l'application peut inclure les pages web qui affichent les annonces, les formulaires de création d'annonces, les pages de profil utilisateur, etc.

3. Le contrôleur : représente la logique de contrôle de l'application, en recevant les demandes des utilisateurs et en les traitant. Cette partie de l'application peut inclure les scripts qui gèrent les interactions utilisateur (création d'annonces, dons, besoins, etc.) et communiquent avec le modèle pour récupérer ou enregistrer les données.

En résumé, l'architecture client/serveur avec l'architecture MVC permettent de séparer les différentes parties de l'application et de simplifier la gestion du projet. Les différentes parties du système peuvent être développées indépendamment les unes des autres, ce qui facilite la maintenance et l'évolutivité de l'application.

2.4.3 DAO et DTO Patterns

DAO Pattern

Le *Data Access Object* (DAO) est une interface qui définit les méthodes pour effectuer des opérations CRUD sur une entité donnée. Cette interface est implémentée par une classe concrète qui effectue les opérations réelles sur la base de données.

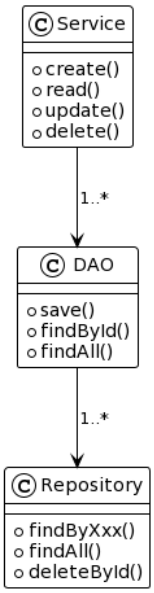

FIGURE 2.5 – Schema DAO.

Dans ce schéma, la couche de service appelle les méthodes de la couche DAO pour interagir avec la base de données. La couche DAO contient les méthodes qui effectuent les opérations CRUD (create, read, update, delete) et d'autres opérations sur la base de données en utilisant la couche Repository, qui agit comme une interface entre l'application et la base de données. Cette couche utilise des requêtes SQL ou JPQL pour interagir avec la base de données.

Le DAO utilise le design pattern Factory pour créer des instances de la couche Repository, et le *Data Transfer Objects* (DTO) pour transférer les données entre les différentes couches de l'application.

DTO Pattern

DTO est un modèle de conception qui permet de transférer des données entre les différentes couches d'une application. Dans le contexte de notre application, les DTO peuvent être utilisés pour représenter les données des entités de notre modèle de domaine (par exemple, les données d'un Utilisateur) de manière plus adaptée pour la couche de présentation (par exemple, pour les vues HTML). Cela peut aider à réduire la complexité de l'objet transféré et à éviter la surcharge de données inutiles.

Pour créer des DTO pour nos entités, il faut créez une classe DTO correspondant à l'entité, avec des attributs correspondants aux champs pertinents de l'entité. Ajoutez des méthodes pour obtenir et définir les valeurs des attributs (getter et setter). Dans les services qui retournent des entités, convertissez l'entité en DTO avant de la renvoyer à la couche de présentation. Dans les contrôleurs, on va les DTO au lieu des entités pour passer les données à la vue.

2.5 Conclusion

D'après ce qu'on a essayer de travailler dans ce chapitre, il nous reste qu'aborder la réalisation du projet.

Chapitre 3

Réalisation et mise en oeuvre

3.1 Introduction

Dans ce chapitre nous allons voir successivement les outils utilisés et la présentation de la réalisation du projet.

3.2 Outils utilisés

Pour réaliser notre projet, nous avons fait appel à plusieurs outils.

3.2.1 IntelliJ IDEA

IntelliJ IDEA est un environnement de développement intégré *Integrated Development Environment* (IDE) pour les langages de programmation Java, Kotlin, Groovy et Scala. Il est développé par JetBrains et est disponible en deux versions : une version communautaire gratuite et une version professionnelle payante qui offre des fonctionnalités supplémentaires.

IntelliJ IDEA offre des fonctionnalités telles que la complétion de code, la refactorisation, la détection d'erreurs de syntaxe et de logique, le débogage, la gestion de versions, la création de tests automatisés, la conception d'interfaces graphiques et la prise en charge de différents frameworks et technologies de développement web. Il est largement utilisé par les développeurs

FIGURE 3.1 – Logo IntelliJ IDEA.

Java pour la création d'applications web, de logiciels d'entreprise, de jeux vidéo, etc.

3.2.2 JUnit

JUnit est un framework open-source de tests unitaires pour la plateforme Java. Il est utilisé pour écrire et exécuter des tests automatisés pour des applications Java et permet aux développeurs de s'assurer que leur code fonctionne correctement en vérifiant que chaque méthode ou fonction du code produit les résultats attendus.

FIGURE 3.2 – Logo JUnit.

3.2.3 Docker

Docker est une plateforme de virtualisation de conteneurs qui permet aux développeurs de créer et d'exécuter des applications dans des environnements isolés. Contrairement à la virtualisation traditionnelle, Docker utilise une approche de virtualisation de niveau système d'exploitation, ce qui permet de partager les ressources de l'OS hôte entre plusieurs conteneurs.

Les conteneurs Docker contiennent tous les éléments nécessaires pour exécuter une application, tels que les bibliothèques, les dépendances et les fichiers de configuration. Docker permet de créer des applications portables et reproductibles qui peuvent être exécutées sur n'importe quelle machine disposant d'un environnement Docker.

FIGURE 3.3 – Logo Docker.

3.2.4 Jenkins

Jenkins est un outil open-source d'intégration et de déploiement continus (CI/CD) qui permet aux développeurs de tester, construire et déployer automatiquement des applications. Jenkins offre une interface web pour créer des tâches d'automatisation, telles que la compilation de code, les tests unitaires, la génération de rapports, le déploiement et la notification de résultats. Il s'intègre avec une large gamme d'outils de développement, notamment Git, GitHub, Docker et JIRA. Jenkins est hautement personnalisable grâce à une bibliothèque de plugins qui permettent d'ajouter des fonctionnalités supplémentaires. Jenkins est largement utilisé dans les projets de développement de logiciels pour améliorer la qualité et la rapidité des déploiements d'applications.

FIGURE 3.4 – Logo Jenkins.

3.2.5 Git et Github

Git est un système de contrôle de version de code source distribué, utilisé pour suivre les modifications apportées à leur code source au fil du temps. Git permet de travailler sur des versions différentes du code source en même temps, de fusionner les modifications apportées par différents contributeurs et de restaurer des versions précédentes du code source si nécessaire.

FIGURE 3.5 – Logo git.

GitHub, quant à lui, est un service en ligne qui héberge des projets Git et fournit des fonctionnalités sociales pour les développeurs. Les utilisateurs peuvent télécharger et partager des projets, contribuer à des projets en ouvrant des demandes de tirage (pull requests), signaler des problèmes et collaborer avec d'autres développeurs en travaillant sur le même code source. GitHub fournit également des fonctionnalités de suivi des problèmes, de gestion de projets et d'intégration continue pour les projets hébergés sur sa plateforme.

FIGURE 3.6 – Logo GitHUb.

En somme, Git est un système de contrôle de version de code source distribué utilisé pour suivre les modifications apportées au code source, tandis que GitHub est un service en ligne qui héberge des projets Git et fournit des fonctionnalités sociales pour les développeurs.

3.2.6 Spring Framework

Spring est un framework open-source pour le développement d'applications Java. Il fournit une plateforme complète pour la création d'applications d'entreprise, en offrant une large gamme de fonctionnalités et de services prêts à l'emploi. Spring permet aux développeurs de construire des applications évolutives, hautement performantes et robustes.

FIGURE 3.7 – Logo Spring-Framework.

Le framework Spring est largement utilisé dans l'industrie pour le développement d'applications d'entreprise et est considéré comme l'un des frameworks Java les plus populaires. Spring permet aux développeurs de se concentrer sur la logique métier de leurs applications plutôt que sur des détails de bas niveau tels que la gestion des transactions, la sécurité et la configuration de l'infrastructure.

3.2.7 Angular

Angular est un framework open-source développé par Google pour la création d'applications Web dynamiques et interactives. Il permet aux développeurs de créer des applications Web à l'aide de TypeScript, un langage de programmation orienté objet basé sur JavaScript. Angular offre une architecture basée sur des composants, une injection de dépendances, une liaison de données bidirectionnelle, ainsi que des fonctionnalités pour la gestion des formulaires, la validation et la gestion des événements. Avec Angular, les développeurs peuvent créer des applications Web performantes et évolutives qui peuvent être déployées sur différentes plateformes, y compris les ordinateurs de bureau et les appareils mobiles.

FIGURE 3.8 – Logo Angular.

3.2.8 Tomcat

Tomcat est un serveur web open-source Java qui permet d'exécuter des applications web Java. Il prend en charge les protocoles HTTP et HTTPS et fournit une interface Java Servlet pour gérer les demandes de pages web dynamiques et les applets Java côté serveur. Tomcat est utilisé principalement pour exécuter des applications web Java basées sur les technologies telles que JavaServer Pages (JSP), Java Servlets, Java Expression Language (EL) et Java WebSocket. Il peut être utilisé comme serveur web autonome ou intégré à un serveur d'applications Java EE pour fournir des services web avancés. Tomcat est largement utilisé dans les environnements de développement et de production pour fournir des applications web robustes et évolutives.

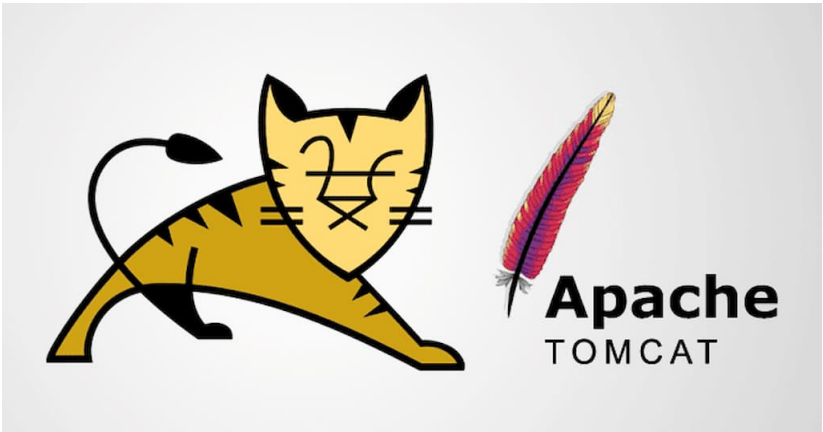

FIGURE 3.9 – Logo Tomcat.

3.2.9 MySQL

Structured Query Language (MySQL) est un *système de gestion de base de données relationnellese* (SGBDR) open-source très populaire. Il est utilisé pour stocker et gérer des données structurées dans des applications Web, des systèmes d'information, des applications mobiles et d'autres types d'applications logicielles. MySQL est écrit en langage C et C++ et est distribué sous une licence libre et gratuite. Il est compatible avec différents systèmes d'exploitation tels que Windows, Linux et Mac OS X. MySQL est largement utilisé dans l'industrie en raison de sa facilité d'utilisation, de sa rapidité et de sa capacité à gérer de grands volumes de données.

FIGURE 3.10 – Logo MySQL .

3.3 La réalisation

Externalisation de la base de données :

Dans le cadre de notre projet , nous avons pris la décision d'externaliser notre base de données afin de garantir une meilleure disponibilité et une meilleure gestion des données.

L'externalisation de notre base de données MySQL a été effectuée en utilisant le service de base de données en ligne FreeDB. Nous avons choisi cette option car elle offre une solution gratuite et facile à utiliser pour héberger notre base de données. Pour accéder à la base de données, nous avons utilisé un driver JDBC fourni par MySQL.

Les informations de connexion à notre base de données sont les suivantes :

- Host : sql.freedb.tech
- Port : 3306
- Nom de la base de données : freedb_YAAKHAYR
- Nom d'utilisateur de la base de données : freedb_jeeteam
- Mot de passe de la base de données : GCA3V*XANZZ5n9

Nous avons configuré notre application pour utiliser ces informations de connexion pour se connecter à notre base de données hébergée sur FreeDB. Cela nous a permis de séparer notre infrastructure de base de données de notre application, offrant ainsi une plus grande flexibilité et une meilleure évolutivité.

Front-end :

La partie front-end concerne principalement l'interface utilisateur et la présentation des données. Nous avons utilisé principalement les technologies web telles que HTML, CSS et JavaScript. Nous avons utilisé également JSF pour générer les pages dynamiquement[2].

Back-end :

La partie back-end se concentre sur la gestion des données et les traitements côté serveur. Nous avons utilisé principalement JEE, Spring et Hibernate pour concevoir cette partie. Nous avons également appliqué les différents design patterns tels que DAO, DTO, Factory pour garantir une architecture propre et maintenable [3].

Tests unitaires et test d'intégration :

nous avons réalisé des tests unitaires pour chacun de nos services, ainsi qu'un test d'intégration pour s'assurer que toutes les composantes de notre application fonctionnent correctement ensemble.

Les tests unitaires ont été effectués en utilisant JUnit, une bibliothèque Java pour écrire des tests unitaires automatisés. Chaque service a été testé pour s'assurer qu'il renvoie les résultats

attendus pour différentes entrées. Les tests unitaires nous ont permis de détecter rapidement les erreurs et de les corriger avant de déployer l'application.

En ce qui concerne le test d'intégration, nous avons créé un environnement de test séparé pour s'assurer que l'application fonctionne correctement dans son ensemble. Nous avons vérifié que les différentes parties de l'application communiquent correctement entre elles et que les données sont stockées et récupérées de la base de données de manière cohérente. Nous avons utilisé JUnit pour réaliser ce test.

Les tests unitaires et d'intégration ont été automatisés pour être exécutés lors de chaque déploiement de l'application, nous garantissant ainsi que toutes les fonctionnalités de l'application restent opérationnelles.

Guide d'utilisations d'APIs :

Nous avons essayé de concevoir un guide d'utilisations d'APIs qui est dans la figure suivante :

URL http://localhost:8080/	Description	Type of Request	Request	Respond
utilisateurs/allusers	retourne tous les utilisateurs	GET	-	[{id,name,password,email,numero}]
utilisateurs/add	Ajouter un utilisateur et retourne l'utilisateur ajoute	POST	{name,password,email,numero}	{id,name,password,email,numero}
utilisateurs/login/email={email}&&password={password}	Verifier s'il existe un utilisateur avec ces donnees si oui il retourne un utilisateur sinon il retourne une reponse 404 (NOT FOUND)	GET	-	{id,name,password,email,numero}
utilisateurs/delete/{id}	Supprimer un utilisateur par son id	DELET	-	-
utilisateurs/add	Ajouter un utilisateur	POST	-	{id,name,password,email, numero,infos}
utilisateurs/update/{id}	Modifier un utilisateur,il retourne l'utilisateur avec les nouvelles modifications	PUT	{name,password,email,numero}	{id,name,password,email, numero,infos}
individu/allindividu	retourne tous les individus	GET	-	[{id,name,password,email, numero,infos}]
individu/{id}	retourne un individu par son id s'il n'existe pas retourne reponse 404(NOT FOUND)	GET	-	{id,name,password,email, numero,infos}
individu/add	Ajouter un utilisateur et retourne l'individu ajoute	POST	{name,password,email,numero,infos}	{id,name,password, email,numero,infos}
individu/myoffers	retourne tous les offers cree par un individu	GET	-	{id,title,description,image, date_publication,etat,quantite, type,individuld}
association/id/{id}	retourne un association par son id s'elle n'existe pas retourne reponse 404(NOT FOUND)	GET	-	{id,name,password,email, numero,type,adresse, justificatifs,image}
association/allassociation	retourne tous les associations	GET	-	[{id,name,password,email, numero,type,adresse, justificatifs,image}]
association/add	Ajouter une association et retourne la nouvelle association	POST	{name,password,email,numero,type,adresse, justificatifs,image}	[{id,name,password,email, numero,type,adresse, justificatifs,image}]
association/update/{id}	modifier une association et retourne l'association modifiee	PUT	{name,password,email,numero,type,adresse, justificatifs,image}	[{id,name,password,email, numero,type,adresse, justificatifs,image}]
association/delet/{id}	Supprimer une association	DELET	-	-
Don/alldon	retourne tous les dons(offre,compagne)	GET	-	{id,title,description,image, date_publication,etat}
Don/id/{id}	retourne un Don par son id senon 404	GET	-	{id,title,description,image, date_publication,etat}
offers/id/{}	retourne une offre par son id senon 404	GET	-	{id,title,description,image, date_publication,etat,quantite, type,individuld}
offers/alloffers	retourne tous les offers	GET	-	[{id,title,description,image, date_publication,etat,quantite, type,individuld}]
offers/add	Ajouter une offre et retourne la nouvelle offere	POST	{title,description,image,date_publication, etat,quantite,type,individuld}	{id,title,description,image, date_publication,etat,quantite, type,individuld}
offers/add	Ajouter une offre et retourne la nouvelle offere	POST	{title,description,image,date_publication, etat,quantite,type,individuld}	{id,title,description,image, date_publication,etat,quantite, type,individuld}
offers/delet/{id}	Supprimer une offre	DELET	-	-
offers/update/{id}	Modifier une offre et retourne l'offere modifiee	PUT	{title,description,image,date_publication, etat,quantite,type,individuld}	{id,title,description,image, date_publication,etat,quantite, type,individuld}
offers/update/{id}	Modifier une offre et retourne l'offere modifiee	PUT	{title,description,image,date_publication, etat,quantite,type,individuld}	{id,title,description,image, date_publication,etat,quantite, type,individuld}

FIGURE 3.11 – guide utilisation d'APIs .

offers/alltypes	Retourner les types des offrs	GET	-	List<String> tous les types des offres
offers/nottaking	Retourner les offrs disponible	GET	-	[{id,title,description,image, date_publication,etat,quantite, type,individuId}]
offers/byindividu	Retourner les offrs creer par un individu	GET	{id,name,password,email,numero,infos}	[{id,title,description,image, date_publication,etat,quantite, type,individuId}]
compagne/id/{id}	retourne une compagne par son id senon 404	GET	-	{id,title,description,image, date_publication,etat,montant, date_colture,associationId}
compagne/id/{id}	retourne une compagne par son id senon 404	GET	-	{id,title,description,image, date_publication,etat,montant, date_colture,associationId}
compagne/allcompagne	retourne tous les compagnes	GET	-	[{id,title,description,image, date_publication,etat,montant, date_colture,associationId}]
compagne/add	Ajouter une compagne et retourner compagne ajoutee	POST	{title,description,image,date_publication, etat,montant,date_colture,associationId}	[{id,title,description,image, date_publication,etat,montant, date_colture,associationId}]
compagne/delet/{id}	Supprimer une compagne	Delet	-	-
compagne/update/{id}	Modifier une compagne et retourner compagne modifiee	PUT	{title,description,image,date_publication, etat,montant,date_colture,associationId}	{id,title,description,image, date_publication,etat,montant, date_colture,associationId}
compagne/colturer/{id}	Colturer une compagne	GET	-	-
compagne/notcolture	retourner tous les compagnes non fermees	GET	-	[{id,title,description,image, date_publication,etat,montant, date_colture,associationId}]
compagne/findbyassociation	retourner tous les compagnes crees par une association	GET	-	[{id,title,description,image, date_publication,etat,montant, date_colture,associationId}]
donner/alldonner	Tous les donners (donner: individu fait un don a une compagne)	GET	-	{id_donner,id_donneur, id_compagne,montant}
donner/findbyindividu/{id}	les compagnes qu'un indifividu fait le donner	GET	-	[{id,title,description,image, date_publication,etat,montant, date_colture,associationId}]
donner/findbycompagne/{id}	tous les individus quit font donner a une compagne	GET	-	[{id,name,password, email,numero,infos}]
donner/add	Ajouter un donner et retourner le nouveau donner	POSt	{id_donneur,id_compagne,montant}	{id_donner,id_donneur, id_compagne,montant}
prendre/add	Ajouter un prendre et retourne le nouveau prendre prendre: une association prend une offre	POSt	{id_association,id_offert,quantite}	{id_prendre,id_association, id_offert,quantite}
prendre/takenbyassociation /{id}	Les offres qui sont prends par une association	GET	-	[{id,title,description,image, date_publication,etat,quantite, type,individuId}]
prendre/offerttakenby /{id_offert}	les associations qui ont prends une offre	GET	-	[{id,name,password,email, numero,type,adresse, justificatifs,image}]

FIGURE 3.12 – guide utilisation d'APIs -suite- .

Conclusion

Au terme de ce projet, nous nous sommes intéressés à consolider et améliorer nos connaissances théoriques et pratiques acquises pendant ce semestre. Ainsi, il y avait plusieurs approches pour être développé, on a essayé de laisser notre touche de créativité, de travailler en harmonie, de s'adapter, et de trouver des solutions efficaces qui respectent la contrainte de temps et d'espace pour être toujours dans l'optimum.

Ce présent travail, étant réalisé par un humain, il n'est pas exempté d'imperfection. Ainsi, les remarques et suggestions pour son amélioration sont donc les bienvenues.

Bibliographie

[1] Spring boot architecture.

https://www.javatpoint.com/spring-boot-architecture, 2011-2021.

[2] Pr Mahmoud EL Hamlaoui. Cours industrialisation logicielle - plateformes de développement, 2023.

[3] Eric Freeman et Kathy Sierra. Design patterns : Tête la première.

http://bliaudet.free.fr/IMG/pdf/DPTLP.pdf, maj. 2010.

Titre : Plateforme des dons "YaKhayr"

Résumé : Le présent document opère une synthèse de notre travail académique intitulé "Mise en place d'une plateforme de dons". Notre projet, dénommé "YaKhayr", a pour but de mettre en place une application web sophistiquée destinée à recueillir des dons en ligne pour divers organismes de bienfaisance.
Cette initiative innovante facilite les dons en permettant une communication directe et bilatérale entre les donateurs et les organisations de bienfaisance grâce à une plateforme technologique de qualité supérieure. Ainsi, tout individu désireux de contribuer à cette noble cause peut le faire en partageant ses connaissances et son expertise tout en effectuant un don à l'organisation de son choix.