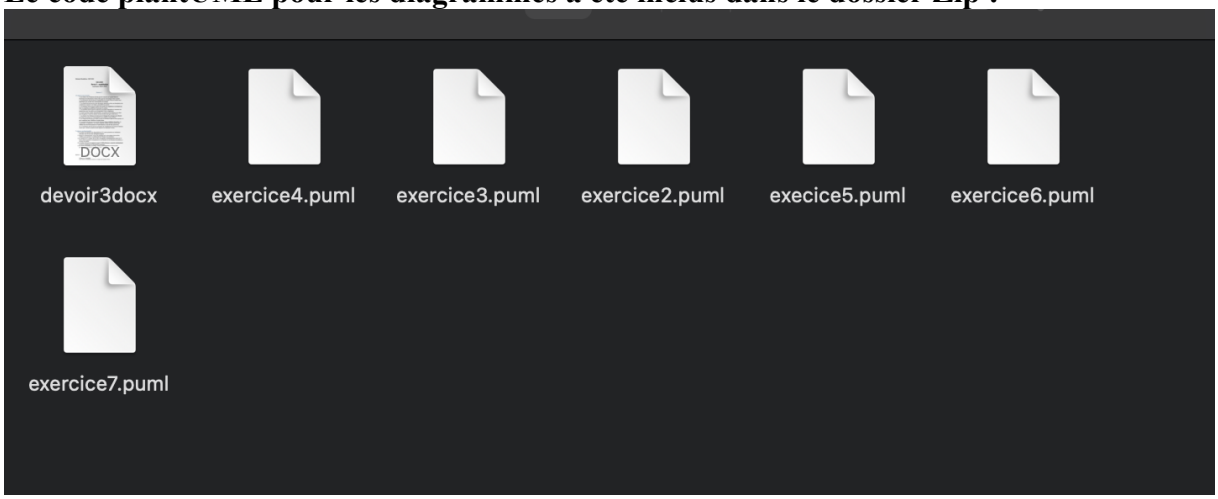
Mohamed Boudabbous- 300376202

SEG2505
Devoir 3 : modélisation
Automne 2024-2025

Le code plantUML pour les diagrammes a été inclus dans le dossier Zip :



Exercice 1 :

10 exigences fonctionnelles :

1.Les clients et les freelances doivent pouvoir s'inscrire sur la plate-forme en fournissant des informations de base telles que leur nom unique et leur courriel.
2. Les utilisateurs doivent pouvoir s'authentifier sur la plate-forme en entrant leurs identifiants pour accéder aux fonctionnalités du système.
3. Les clients doivent pouvoir créer des projets, spécifier un titre, une description, des compétences requises et un budget, puis publier ces projets.
4. Les freelances doivent pouvoir gérer leurs profils de compétences en utilisant une liste de compétences prédéfinies fournie par le système.
5. Les freelances doivent pouvoir parcourir les projets disponibles et soumettre des propositions pour les projets qui correspondent à leurs compétences.
6. Lorsqu'un client accepte une proposition, un contrat qui lie freelance et le client doit être généré, il inclut la mission, la date de début ainsi que la date de fin.
7. Les clients et les freelances doivent pouvoir échanger des messages pour discuter des détails des projets par l'intermédiaire d'un système de chat sécurisé.
8. Les clients doivent pouvoir évaluer et noter les freelances après la fin des projets, ce qui va améliorer leur visibilité sur la plate-forme.
9. Les projets doivent pouvoir traverser plusieurs étapes prédéfinies (brouillon, en recrutement, en réalisation, en révision, suspendu, annulé, terminé, archivé) qui reflètent leur état d'avancement et d'achèvement au cours de leur cycle de vie.
10. Les freelances doivent pouvoir soumettre des contributions sous forme d'artefacts (nom, type, version), qui peuvent être approuvé ou rejeté par le client.

5 exigences non fonctionnelles :

1. Le système doit garantir une disponibilité de 95 % pour permettre aux utilisateurs d'accéder aux services sans interruption majeure.

2. Toutes les communications, tel que les échanges par le chat intégré, doivent être chiffrées pour garantir la confidentialité des données des utilisateurs.
3. Les données de ce système doivent être sauvegardées automatiquement toutes les 10 heures afin de garantir la récupération de ces données en cas de panne du système ou de perte de donnée.
4. Le système doit pouvoir supporter jusqu'à 10 000 utilisateurs connectés simultanément en tout en maintenant des performances optimales.
5. Les actions principales du système, comme l'authentification ou l'envoi d'une proposition, doivent s'exécuter en moins de 1 secondes

Exercice 2 **(exercice2.puml)***:*

liste de cas d'utilisation UML pour le système :

1.Sign Up (S'inscrire)
Les clients et freelances créent un compte sur la plate-forme.

2.Log In (S'authentifier)
Les utilisateurs se connectent pour avoir l'accès aux fonctionnalités.

3.Create and Manage a Profile (Créer et gérer un profil)
Les freelances gèrent leurs profils de compétences.

4.Create a Project (Créer un projet)
Les clients ajoutent les informations de base du projet.

5.Publish a Project (Publier un projet)
Les clients rendent un projet visible aux freelances.

6.Browse Projects (Parcourir les projets)
Les freelances découvrent et explorent les projets disponibles.

7.Submit a Proposal (Soumettre une proposition)
Les freelances proposent leurs services pour des projets en soumettant une demande.

8.Manage Proposals (Gérer les propositions)
Les clients examinent et sélectionnent les propositions des freelances.

9.Establish a Contract (Établir un contrat)
Un contrat qui lie un client et un freelance pour un projet est créé.

10.Submit a Contribution (Soumettre une contribution)
Les freelances soumettent des artefacts pour les projets en cours.

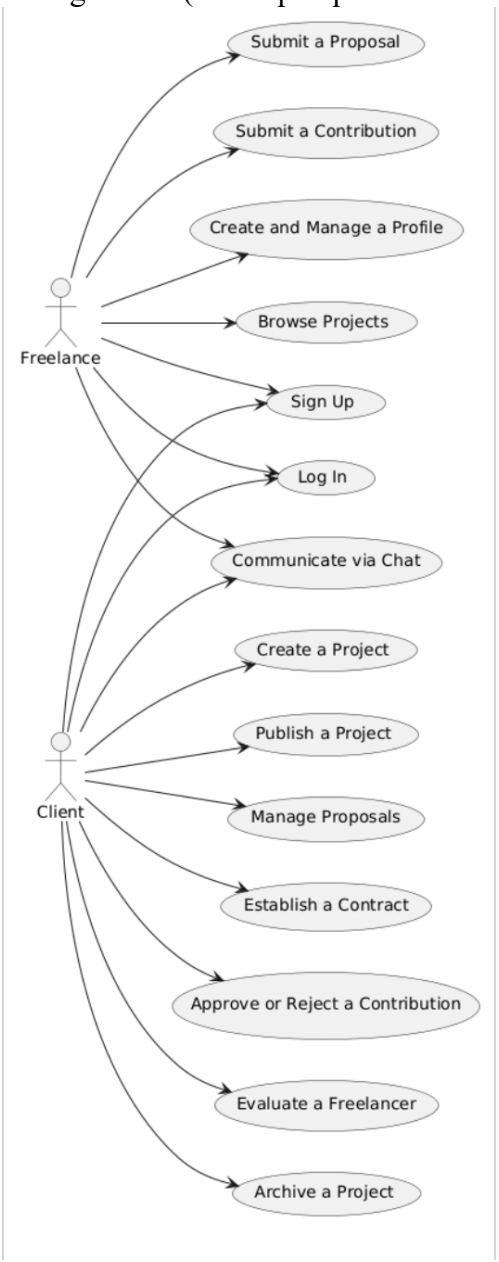
11.Approve or Reject a Contribution (Valider ou rejeter une contribution)
Les clients approuvent ou rejettent les contributions soumises.

12.Communicate via Chat (Communiquer via le chat)
Les clients et freelances s'échangent des messages sur les projets.

13.Evaluate a Freelancer (Évaluer un freelance)
Les clients évaluent les freelances à la fin d'un projet.

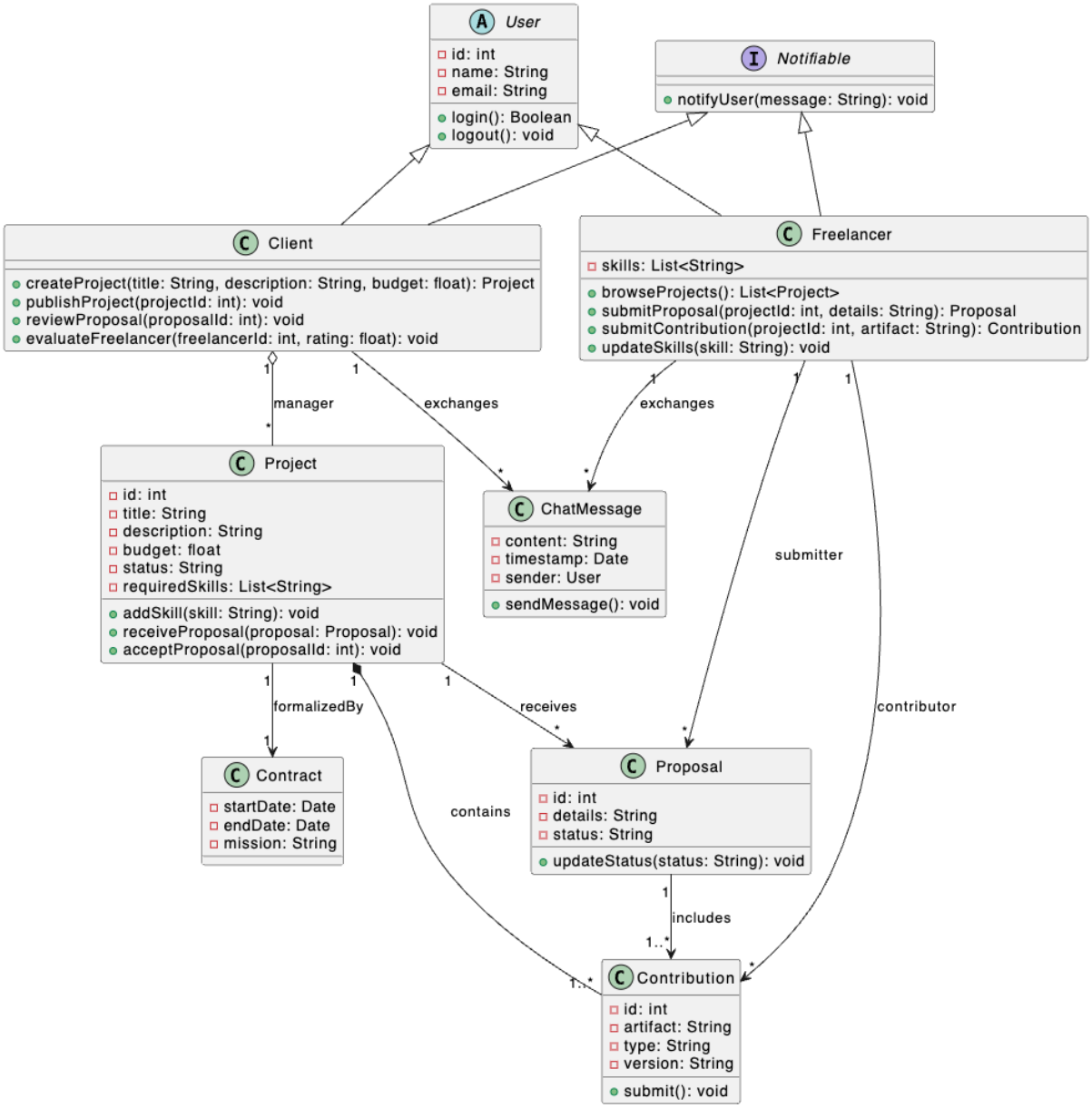
14.Archive a Project (Archiver un projet)
Les projets terminés sont archivés pour des consultations futures.

Voci le diagramme (une copie .puml est fornie dans le dossier)



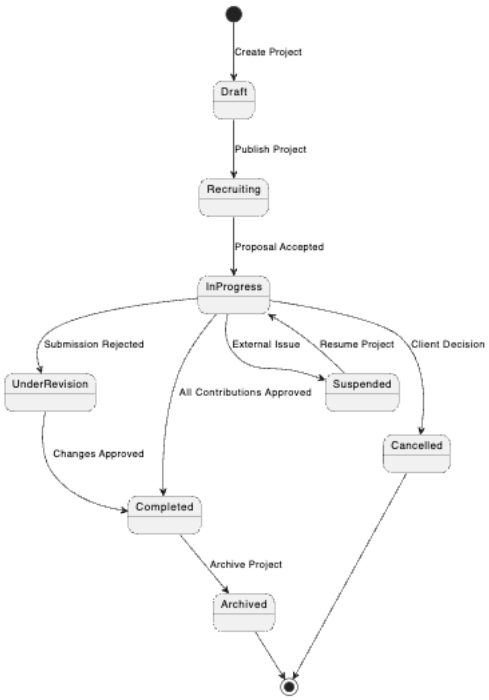
Exercice 3 :

Le diagramme UML modélise les entités et interactions clés de la plateforme FreelanceHub. La classe abstraite Userdéfinit les attributs communs et actions de base, avec deux spécialisations : Client, qui peut créer, publier et gérer des projets, et Freelancer, qui peut soumettre des propositions, gérer ses compétences et fournir des contributions. Les projets, représentés par Project, reçoivent des Proposals soumises par des freelances et peuvent inclure des Contributions validées. La classe Contract formalise la relation entre un projet et un freelance, tandis que ChatMessage modélise les échanges entre utilisateurs. Enfin, l'interface Notifiable permet de gérer les notifications, offrant une vue claire des rôles et relations dans le système.

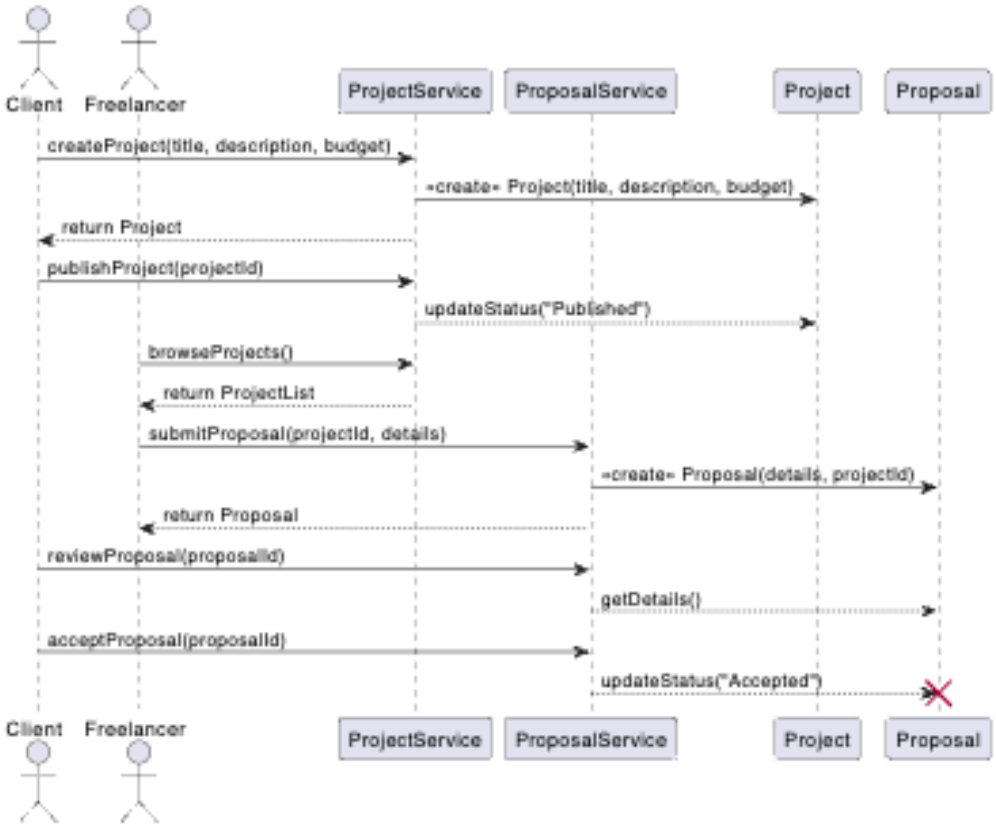


Exercice 4 (exercice4puml):

Un projet traverse plusieurs états au cours de son cycle de vie. Initialement, il est dans l'état Draft (Brouillon), où il est créé mais non publié. Une fois publié, il passe à l'état Recruiting (En cours de recrutement), devenant visible aux freelances qui peuvent soumettre des propositions. Après la sélection des freelances, le projet entre dans l'état In Progress (En cours de réalisation), où le travail commence. Si des modifications sont nécessaires après soumission, le projet passe à l'état Under Revision (En révision). En cas de problème, il peut être temporairement mis en Suspended (Suspendu). Si le client décide d'interrompre définitivement le projet, il passe à l'état Cancelled (Annulé). Lorsqu'il est terminé avec toutes les contributions approuvées, il atteint l'état Completed (Terminé). Enfin, un projet achevé et finalisé est conservé en Archived (Archivé) pour référence future.

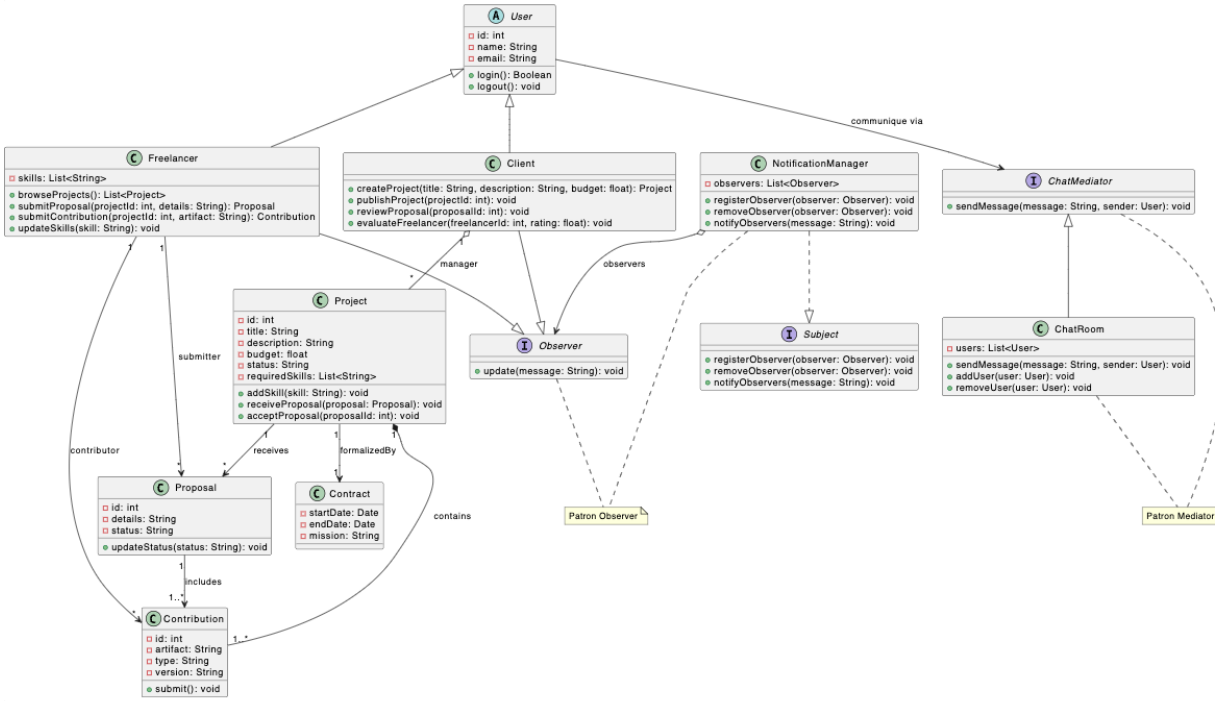


Exercice 5 : (exercice5.puml)



Le Client commence par créer un projet via le service ProjectService, qui génère un objet Project. Une fois publié, le projet devient accessible aux freelances via l'action browseProjects(). Un Freelancer soumet ensuite une proposition via ProposalService, qui crée un objet Proposal. Enfin, le client examine et accepte la proposition, ce qui met à jour son état à "Accepted", suivi de la destruction de l'objet Proposal. Le diagramme met en évidence les interactions principales entre acteurs et services, en restant au niveau "modèle de domaine".

Exercice 6 : (exercice6.puml)

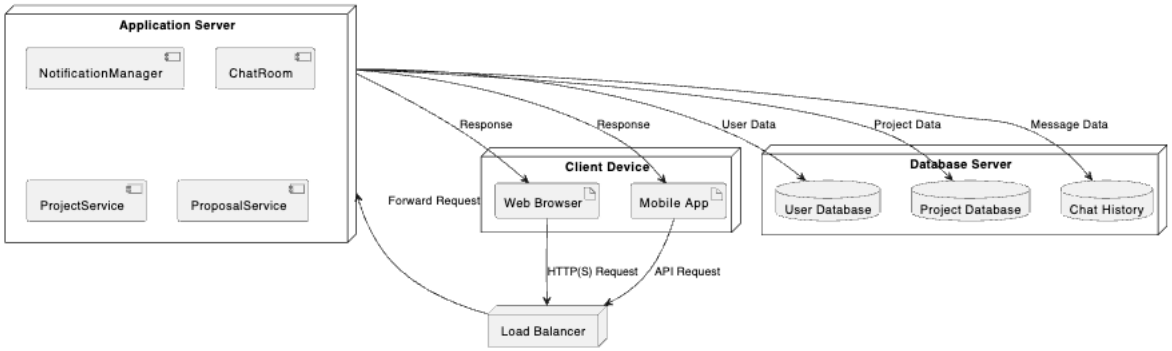


Ce diagramme UML représente les principales entités du système et leurs interactions. La classe abstraite User est la base pour Client et Freelancer, permettant la connexion et déconnexion. Les clients peuvent créer, publier des projets, examiner des propositions, et évaluer des freelances, tandis que les freelances gèrent leurs compétences, soumettent des propositions et des contributions. Les projets, modélisés par Project, incluent des informations essentielles et sont associés à des propositions et contributions.

Le système utilise deux patrons de conception : Observer, via NotificationManager, pour notifier dynamiquement les utilisateurs, et Mediator, avec ChatRoom, pour centraliser la communication. Les relations, incluant l'agrégation et la composition, structurent efficacement les interactions, offrant une architecture claire.

Des classes et interfaces supplémentaires pour implémenter efficacement les patrons de conception Observer et Mediator afin de renforcer la modularité et la maintenabilité du système

Exercice 7 :



Ce diagramme de déploiement UML représente l'infrastructure du système. Les utilisateurs accèdent au système via un navigateur web ou une application mobile, leurs requêtes étant d'abord traitées par un Load Balancer qui redirige vers un Application Server. Ce dernier héberge les composants principaux telque NotificationManager, ChatRoom, ProjectService, et ProposalService, qui sont responsables des fonctionnalités logiques du systeme. Les données des utilisateurs, des projets et des messages sont stockées dans un Database Server, ce dernier est structuré en bases spécifiques : pour les utilisateurs, les projets et l'historique des communications. Cette architecture centralise la gestion et optimise la communication entre les composants.