**Sujet de TD/TP 2 : Modélisation UML des cas d'utilisation et des classes**

**Dans ce TD/TP, vous allez approfondir votre compréhension de la modélisation UML en vous basant sur la plateforme WAVE. Votre mission consiste à :**

**1. Analyser les cas d'utilisation identifiés précédemment pour chaque acteur et en décrire les interactions avec le système.**

**2. Créer un diagramme de cas d'utilisation représentant les fonctionnalités principales du système, incluant les acteurs et leurs interactions avec les cas d'utilisation.**

**3. Élaborer un diagramme de classes qui décrit la structure du système, en définissant les classes, les attributs, les méthodes et les relations entre les classes.**

**4. Définir les relations d'héritage, d'association et d'agrégation dans le diagramme de classes pour illustrer les interactions entre les différentes entités du système.**

**5. Discuter des implications de la modélisation UML sur le développement du système d'information de WAVE, en prenant en compte les enjeux, défis, et objectifs de la plateforme.**

**Réfléchissez également à la manière dont ces diagrammes peuvent servir de base pour la conception technique et le développement futur du système d'information.**

**1. Analyse des cas d'utilisation et interactions avec le système**

Chaque acteur du système WAVE interagit avec différentes fonctionnalités. Ton objectif est d’identifier les actions que chaque acteur effectue et comment elles sont liées au système.

**Administrateur** :

 Gérer les comptes utilisateurs (création, modification, suppression).

* Configurer les paramètres de sécurité.
* Gérer les accès et les permissions.

 **Client (utilisateur final)** :

* Accéder aux projets.
* Consulter les rapports.
* Gérer son profil (modifier informations, mot de passe).

 **Développeur** :

* Consulter et modifier les tickets (bugs, nouvelles fonctionnalités).
* Soumettre des mises à jour ou des correctifs.

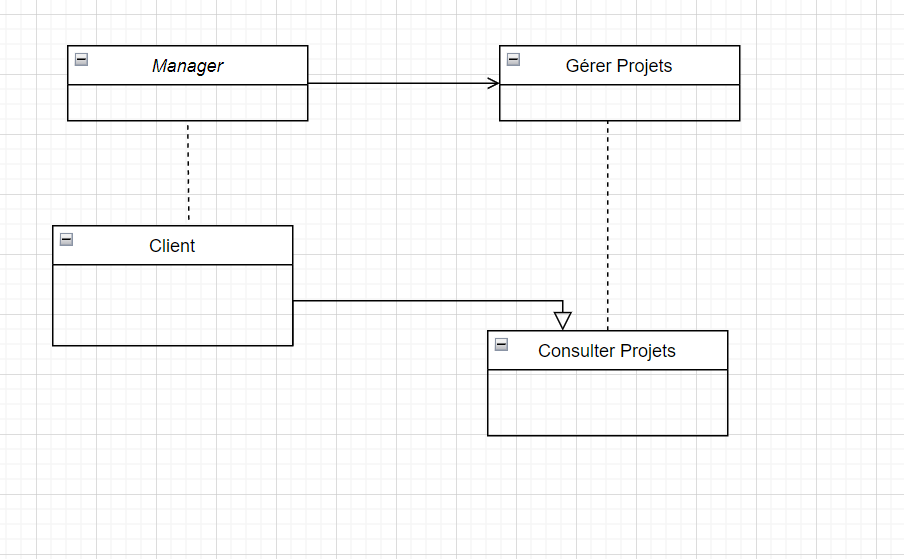
 **Manager de projet** :

* Créer et gérer des projets.
* Suivre les avancées (statuts, délais).
* Gérer les équipes projet (attribution de tâches).

2. **Diagramme de cas d'utilisation**

Les **cas d'utilisation** incluraient :

* Gérer Projets (pour le Manager de projet).
* Consulter Projets (pour le Client).
* Gérer Tickets (pour le Développeur).

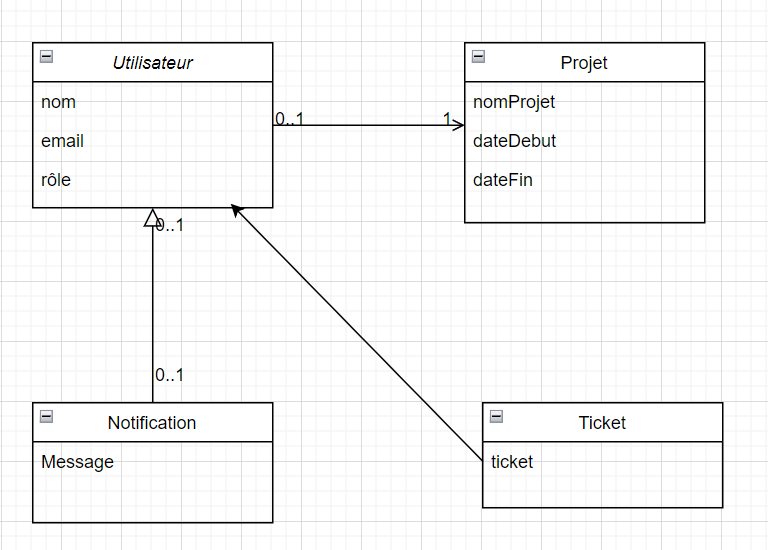
****

**3. Diagramme de classes**

Le **diagramme de classes** montre la structure du système, en se concentrant sur les classes (entités), leurs attributs, méthodes, et leurs relations.

Pour la plateforme WAVE, les classes principales pourraient inclure :

* **Utilisateur** (classe parent pour tous les types d'utilisateurs).
* **Projet** (chaque projet géré sur la plateforme).
* **Ticket** (rapport de bugs ou fonctionnalités).
* **Notification** (informations envoyées aux utilisateurs).



### 4. ****Définir les relations d'héritage, d'association, et d'agrégation****

* **Héritage** : Utilisé lorsque certaines classes partagent des propriétés communes. Par exemple, les classes Admin, Client, et Manager de Projet peuvent hériter de la classe Utilisateur (relation avec un triangle ouvert).
* **Association** : Relation simple entre deux classes. Par exemple, un Utilisateur peut être associé à plusieurs Projets et chaque Projet est associé à un ou plusieurs Tickets.
* **Agrégation** : Lorsque certaines classes sont constituées d’autres classes mais ne dépendent pas entièrement de leur existence. Par exemple, un Projet est composé de plusieurs Tickets, mais un Ticket peut exister en dehors d’un projet.

**5. Implications de la modélisation UML sur le développement du système**

* **Enjeux** :
  + Simplifier la communication entre les développeurs, les chefs de projet et les parties prenantes en utilisant des représentations visuelles compréhensibles.
  + Clarifier les fonctionnalités et les relations avant le codage, ce qui permet d'éviter des erreurs de conception coûteuses.
* **Défis** :
  + Gérer la complexité croissante du système à mesure que de nouvelles fonctionnalités sont ajoutées.
  + S'assurer que le modèle UML reflète bien les besoins fonctionnels et techniques.
* **Objectifs** :
  + Créer une documentation visuelle précise et complète qui servira de base pour la conception technique et le développement.
  + Faciliter l'identification des classes réutilisables et les interactions potentielles entre les entités.