

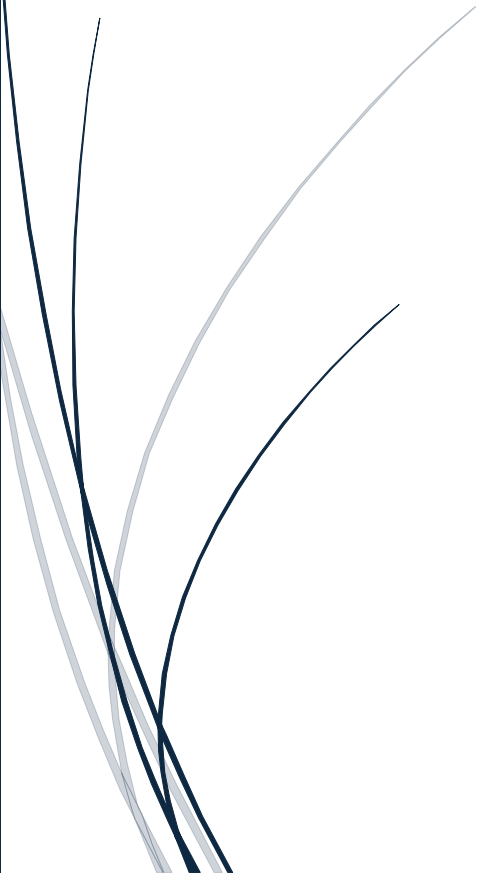
2024-08-01

# Projet : Timelog

NOMS ET PRÉNOMS DES MEMBRES :

AHUELIE WILFRIED

NYAMA KOUMBA AUDE GUYSLIAH MAELISSE



Abd-Ali Jamal  
INF1163-01 Modélisation et conception orientée  
objet

# Introduction

Dans le monde professionnel d'aujourd'hui, la gestion efficace du temps et des ressources est devenue un enjeu crucial pour les entreprises. Le projet TimeLog répond à ce besoin en offrant une solution innovante pour suivre et analyser le temps consacré par les employés à divers projets et activités. En considérant l'importance croissante de la gestion du temps dans les entreprises modernes, comment un système comme TimeLog pourrait-il évoluer pour répondre aux défis futurs du travail, tels que l'augmentation du télétravail et la flexibilité des horaires ? Nous examinerons les principales entités du système, telles que les employés, les projets et les activités, ainsi que leurs relations. Nous verrons comment les diagrammes de classe et de séquence ont été utilisés pour représenter la structure et le comportement du système. Nous décrirons le processus de transformation des modèles UML en code Java fonctionnel à l'aide de l'outil Modelio.

## A-Modèle du domaine

### 1. Système Timelog :

- C'est la classe principale qui englobe tout le système.
- Elle a un attribut "admin" de type Admin.
- Elle a un attribut "NPE" de type Integer.
- Elle contient des listes pour les employés, projets et activités.
- Elle possède une classe de paie (payrollinfo).
- Elle possède une interface de paie (payrollInterface).

### 2. Employé :

- Attributs : id, nom, poste, NAS, ListActivite, ListTauxHoraire, dateDepart, dateEmbauche.
- Un employé peut être associé à plusieurs activités et projets.

### 3. Admin :

- Attributs : username, id, dateDebut, dateFin.
- Il y a une relation entre Admin et Système Timelog (probablement pour la gestion du système).

### 4. Projet :

- Attributs : id, nom, dateDebut, dateFin, heures\_budgetees.
- Un projet peut avoir plusieurs activités.

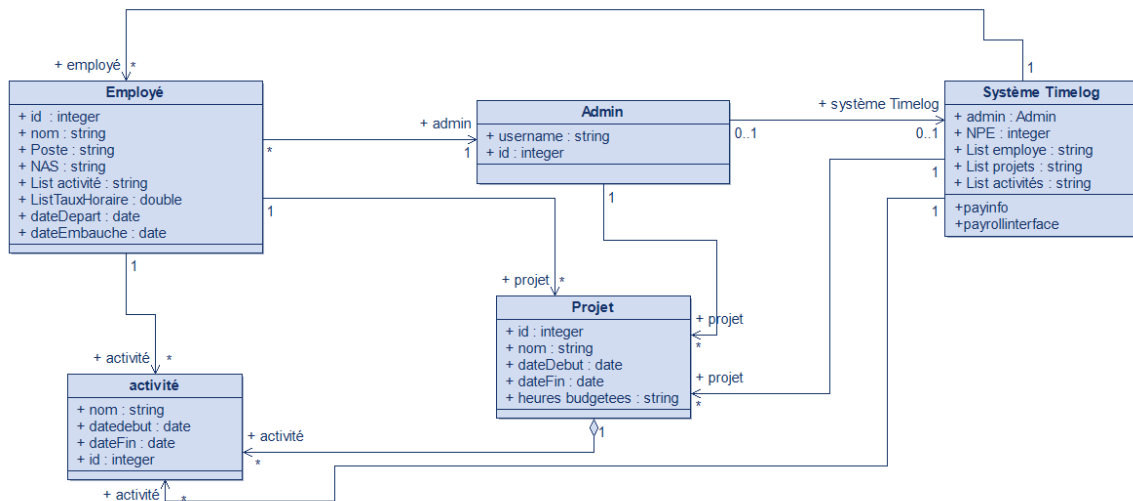
- Il est lié au Système Timelog.

### **5. Activité :**

- Attributs : nom, dateDebut, dateFin, id.

- Une activité est associée à un projet et peut être réalisée par un employé.

**NB :** Dans notre projet, nous avons choisi de ne pas utiliser le terme "**discipline**" de manière explicite. Au lieu de cela, nous avons opté pour l'utilisation directe du nom de chaque activité. Cette décision a été prise pour éviter toute confusion potentielle, étant donné que, selon l'énoncé du projet, il n'y a pas de distinction significative entre une discipline et une activité.



### **6-Relations :**

- Un Système Timelog a un Admin

- Un Système Timelog peut avoir plusieurs Employés, Projets et Activités.

- Un Employé peut participer à plusieurs Projets et Activités.

- Un Projet peut avoir plusieurs Activités.

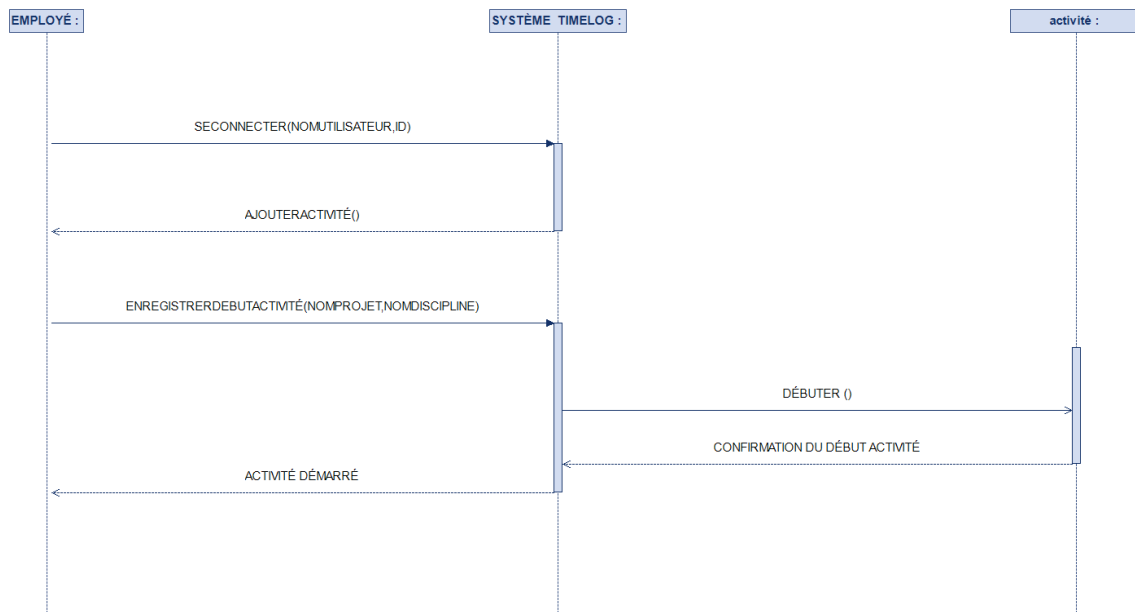
- Une Activité appartient à un seul Projet.

Ce modèle semble conçu pour suivre le temps passé par les employés sur différents projets et activités, probablement dans un contexte de gestion de ressources humaines ou de facturation de temps.

### **B- Diagrammes uml**

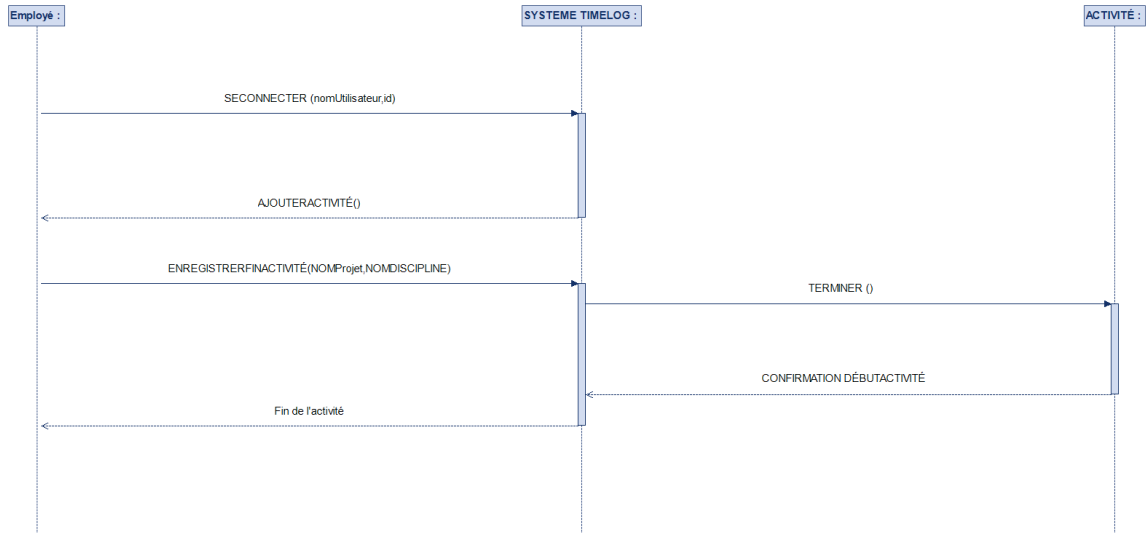
### ●Diagramme de séquences de débiter une activité

Ce diagramme de séquence offre une représentation graphique détaillée des interactions complexes qui se déroulent entre les diverses composantes du système lors du lancement d'une nouvelle activité. Il dépeint avec précision l'enchaînement chronologique des actions, les flux de données échangés, ainsi que les mécanismes de communication entre les acteurs et les modules du système.



### ●Diagramme de séquences de terminé une activité

Ce diagramme de séquence illustre les interactions entre les différents éléments impliqués dans le processus de terminer une activité. Il met en évidence l'enchaînement des actions et des échanges d'informations nécessaires pour clôturer une activité dans le système. Cette représentation visuelle permet de comprendre clairement la chronologie et la logique des opérations effectuées lors de la finalisation d'une activité, montrant ainsi les relations entre les acteurs et les composants du système dans ce contexte spécifique.

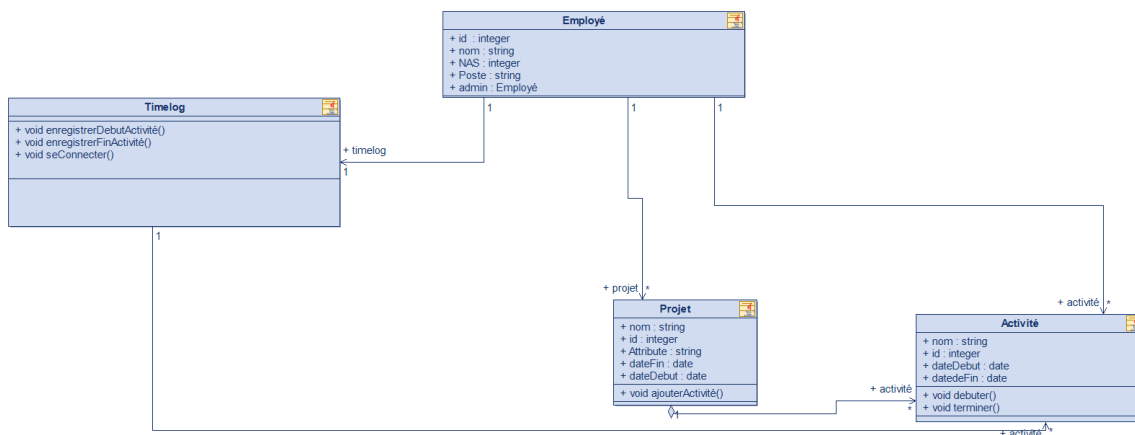


## ●Diagramme de classe

Ce diagramme représente de manière détaillée les scénarios des deux cas d'utilisation : "débuter une activité" et "terminer une activité". Il met en lumière, de façon claire et précise, les opérations décrites dans les deux Diagrammes de Séquence Système (DSS) mentionnés précédemment.

Ce schéma est une extension du modèle du domaine, enrichi par les spécificités des scénarios étudiés. Il joue un rôle crucial dans notre processus de développement, car c'est à partir de ce diagramme que nous allons générer le code de l'application.

Cette représentation visuelle nous permet de visualiser concrètement les interactions entre les différents éléments du système, facilitant ainsi la transition entre la conception et l'implémentation du projet.



## C-GÉNÉRATION DU CODE MODELIO

La génération de code sur Modelio nécessite de suivre un cheminement précis, composé de plusieurs étapes bien définies. Ces étapes, présentées ci-dessous, forment un processus structuré pour assurer une génération de code efficace et cohérente.

### **1. Création du diagramme de classe :**

- Ouvrez un projet dans Modelio.
- Choisissez le type de diagramme en sélectionnant "**Diagramme de classe**".
- Utilisez les outils situés à gauche de l'interface pour modéliser votre diagramme. Ces outils vous permettront de créer des classes, d'ajouter des méthodes, des attributs et de définir les associations entre les différentes classes.

### **2. Configuration de Java Designer :**

- Cliquez sur "**Configuration**" dans le menu de Modelio.
- Sélectionnez l'option "**Module**".
- Cliquez sur "**Ajouter**", puis choisissez l'extension **pour le langage Java**.

### **3. Préparation des classes pour la génération de code :**

- Pour chaque classe dont vous souhaitez générer le code, ajoutez l'étiquette "**Java**".
- Faites un clic droit sur la classe concernée.
- Dans le menu contextuel, sélectionnez "**Java**".

### **4. Activation de la génération de code pour chaque classe :**

- Sélectionnez la classe désirée.
- Cochez la case "**Élément Java**" dans les propriétés de la classe.
- Cliquez ensuite sur "**UML**" pour lancer la génération du code.

### **5. Visualisation du code généré :**

- Une fois la génération terminée, ouvrez l'**éditeur de code** intégré à Modelio.
- Le code Java correspondant à votre diagramme de classe s'affichera dans cet éditeur.

Il est important de noter que cette méthode permet de transformer votre modèle UML en code Java fonctionnel. Cela facilite grandement le passage de la conception à l'implémentation, en assurant une correspondance directe entre votre diagramme de classe et le code source. N'hésitez pas à ajuster votre diagramme de classe si nécessaire avant de générer le code. Cela vous permettra d'obtenir une structure de code qui reflète précisément votre conception. De plus, Modelio offre la possibilité de synchroniser les modifications apportées au diagramme avec le code généré, ce qui est particulièrement utile lors de la phase de développement itératif.

## **Conclusion**

La modélisation UML et la génération de code à l'aide de Modelio facilitent la transition entre la conception et la mise en œuvre de projets TimeLog. Ce processus garantit un mappage direct entre les modèles UML et le code Java, garantissant ainsi une structure de code bien définie qui répond aux spécifications de votre projet. En suivant ces étapes, nous avons construit une base solide pour développer un système TimeLog qui nous permet de gérer efficacement les projets et les activités de vos employés.