**Énoncé du projet : Développement d’un RaidPlanner pour Final Fantasy XIV :**

**Contexte :**

Les raids dans **Final Fantasy XIV (FF14)** sont des activités de groupe où la coordination entre les joueurs est essentielle. La planification de ces raids, l'attribution des rôles, la gestion des disponibilités et la communication des stratégies peuvent devenir un véritable casse-tête sans un outil dédié. Ce projet vise à développer une application web de planification de raids pour FF14, en utilisant **ASP.NET Core Web API** pour la gestion de la logique côté serveur et **Angular** pour la création d'une interface utilisateur réactive et moderne.

**Objectifs du projet :**

* **Backend (ASP.NET Core Web API)** : Fournir une API RESTful permettant la gestion des utilisateurs, des raids, des rôles, des disponibilités et des stratégies.
* **Frontend (Angular)** : Développer une interface utilisateur dynamique qui permet aux joueurs de planifier, gérer et suivre leurs raids en toute simplicité.
* **Authentification et autorisation** : Intégrer un système d'authentification et de gestion des utilisateurs (par exemple, avec JWT).
* **Notifications et rappels** : Envoyer des notifications aux membres de l’équipe pour les informer des mises à jour des raids et des rappels avant les événements.

**Fonctionnalités principales :**

1. **Gestion des utilisateurs (Authentification et autorisation)** :
   * Les utilisateurs peuvent s’inscrire, se connecter, et gérer leurs profils.
   * Les joueurs peuvent se voir attribuer des rôles spécifiques (chef de raid, membre).
   * Utilisation de **JWT (JSON Web Tokens)** pour sécuriser les sessions.
2. **Création et gestion des raids** :
   * Les chefs de raid peuvent créer des raids en spécifiant la date, l’heure et les membres de l’équipe.
   * Les joueurs peuvent se joindre ou se désister des raids existants.
   * Attribution des rôles de manière manuelle ou automatique (Tank, DPS, Healer).
3. **Gestion des disponibilités** :
   * Les joueurs peuvent spécifier leurs disponibilités pour chaque raid.
   * Les chefs de raid peuvent voir rapidement qui est disponible pour un raid donné.
4. **Planification et calendrier** :
   * Un calendrier interactif permet de voir tous les raids programmés et les disponibilités des joueurs.
   * Les membres peuvent voir l’ensemble des raids à venir et leurs rôles dans chacun d'eux.
5. **Gestion des stratégies** :
   * Les joueurs peuvent partager des stratégies et des guides pour chaque rencontre de raid.
   * L’interface permettra d’ajouter des notes, des images ou des vidéos pour chaque boss ou mécaniques spécifiques.
6. **Notifications** :
   * Envoi de notifications par **WebSocket** ou par **email** pour rappeler aux joueurs les raids à venir ou les changements dans la planification.
7. **Suivi des performances** :
   * Statistiques sur les performances des joueurs dans les raids précédents (par exemple, DPS, Heals, efficacité des rôles, etc.).
   * Tableau de bord dynamique avec graphiques pour visualiser l'évolution des performances des membres.
8. **Chat intégré (optionnel)** :
   * Un système de chat en temps réel pour la communication entre les membres du raid.
   * Support des messages privés et des canaux de discussion.

**Technologies suggérées :**

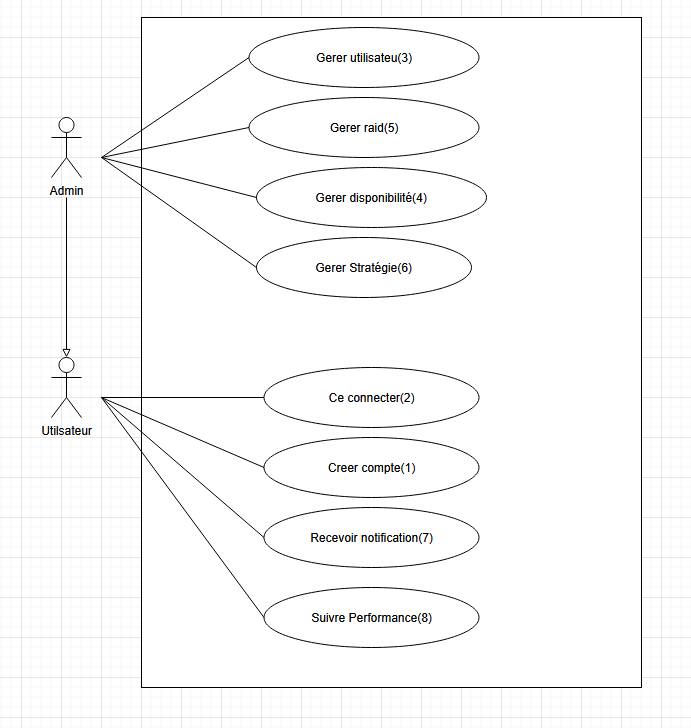
* **Backend (ASP.NET Core Web API)** :
  + **ASP.NET Core** pour créer l'API RESTful.
  + **Entity Framework Core** pour la gestion de la base de données (par exemple, SQL Server ou PostgreSQL). Packet nuget : Microsoft.EntityFrameworkCore, Microsoft.EntityFrameworkCore.SqlServer, Microsoft.EntityFrameworkCore.Tools
  + **JWT (JSON Web Token)** pour l'authentification.
  + **SignalR** pour les notifications en temps réel (par exemple, pour les rappels de raid ou les mises à jour).
* **Frontend (Angular)** :
  + **Angular** pour la création de l'interface utilisateur réactive.
  + **RxJS** pour la gestion des flux de données asynchrones.
  + **Bootstrap/Booswatch** ou **Angular Material** pour le design et la mise en page de l’application.
  + **Angular Services** pour la gestion des appels API.
* **Base de données** :
  + **SQL Server** ou **PostgreSQL** pour la persistance des données (utilisateurs, raids, rôles, statistiques).
  + **Redis** (optionnel) pour les sessions ou la gestion du cache.

**Livrables attendus :**

* **API Backend (ASP.NET Core Web API)** : API fonctionnelle et sécurisée, permettant la gestion des utilisateurs, des raids, des disponibilités et des stratégies.
* **Frontend (Angular)** : Application web réactive permettant la gestion complète des raids, des utilisateurs et des notifications.
* **Documentation technique et utilisateur** : Explication de l’architecture, des API, et des fonctionnalités pour les développeurs et les utilisateurs finaux.
* **Code source** : Le code source complet, disponible sur une plateforme de versioning comme GitHub.

**Durée estimée du projet :** semaines

**Use case diagram :**



Signature :

**Priorité :**

1.Creer compte

2.Se connecter

3.Gerer utilisateur

4.Gerer disponibilité

5.Gerer raid

6.Gerer stratégie

7.Recevoir notification

8.Suivre performance

**Use case high level :**

**1. Créer un compte (Inscription)**

* **Acteur principal** : Utilisateur (Joueur)
* **Description** : L'utilisateur peut s'inscrire à l'application en créant un compte avec un nom d'utilisateur, un email, et un mot de passe.

**2. Se connecter**

* **Acteur principal** : Utilisateur (Joueur)
* **Description** : Permet à un utilisateur de se connecter à son compte en utilisant son email et son mot de passe

**3. Gérer utilisateur (Profil)**

* **Acteur principal** : Utilisateur (Joueur)
* **Description** : Permet à l'utilisateur de gérer son profil (changer son mot de passe, mettre à jour ses informations).

**4. Gérer disponibilité**

* **Acteur principal** : Utilisateur (Joueur)
* **Description** : Permet à l'utilisateur de définir ses disponibilités pour un raid (jours et heures).

**5. Gérer raid**

* **Acteur principal** : Chef de raid
* **Description** : Permet au chef de raid de créer un raid, définir la date, les membres et les rôles.

**6. Gérer stratégie**

* **Acteur principal** : Chef de raid, Utilisateur (Joueur)
* **Description** : Permet au chef de raid et aux joueurs de consulter et partager des stratégies pour chaque boss.

**7. Recevoir notification**

* **Acteur principal** : Utilisateur (Joueur)
* **Description** : Permet à l'utilisateur de recevoir des notifications pour des rappels de raid ou des changements dans l'organisation des raids.

**8. Suivre performance**

* **Acteur principal** : Utilisateur (Joueur), Chef de raid
* **Description** : Permet à l'utilisateur de suivre ses performances dans les raids passés (DPS, Heals, rôle dans le raid, etc.).

**Use case 1 : Créer un compte (Inscription) :**

### **Use Cases détaillés - Low Level**

**Acteur principal** : Utilisateur

**Description** : Un nouvel utilisateur veut s'inscrire dans l'application en créant un compte avec un nom d'utilisateur, un email et un mot de passe.

**Pré-conditions** :

* L'utilisateur n'a pas encore de compte.

**Post-conditions** :

* L'utilisateur est enregistré dans la base de données.
* Un email de confirmation peut être envoyé pour vérifier l'email de l'utilisateur (optionnel).

**Scenario principal :**

|  |  |  |
| --- | --- | --- |
| Etapes | Acteurs | Système |
| 1 | L’use case commence quand l’utilisateur clique sur l’option « Créer un compte » |  |
| 2 |  | Le système affiche une page avec un formulaire de création de compte (username, email, Password) |
| 3 | L’utilisateur entre son **nom d'utilisateur**, son **email**, et son **mot de passe**. |  |
| 4 |  | Le système affiche une page avec les informations de l’utilisateur. En dessous il affiche un bouton modifier. Il affiche aussi un bouton valider. |
| 5 | L’utilisateur clique sur valider |  |
| 6 |  | Le système valide les informations, enregistre dans la base de données et redirige l'utilisateur vers la page de connexion. |

**Scenario alternatif :**

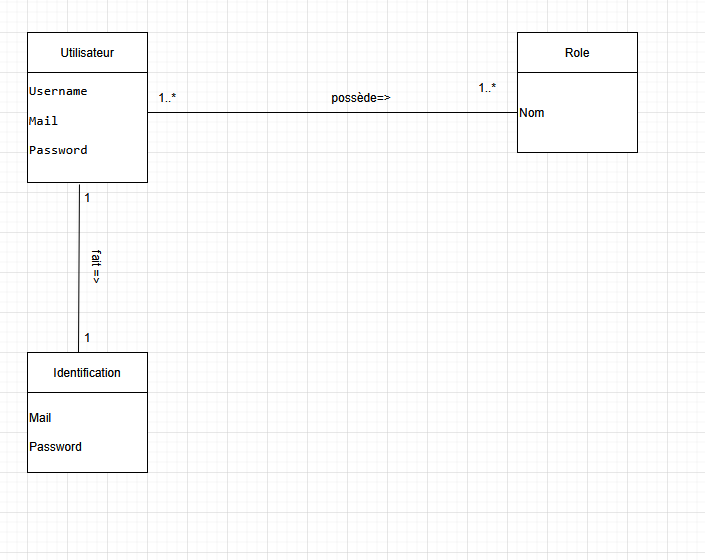
|  |  |  |
| --- | --- | --- |
| Etapes | Acteurs | Système |
| 5.a.1 | L’utilisateur clique sur Modifier |  |
| 5.a.2 |  | Le système affiche une page avec les informations de l’utilisateur et permet de les modifié |
| 5.a.3 | L’utilisateur modifie les information |  |
| 5.a.4 |  | Le système affiche une page avec les informations de l’utilisateur. En dessous il affiche un bouton modifier. Il affiche aussi un bouton valider. |
| 5.a.5 | L’utilisateur clique sur valider |  |
| 5.a.6 |  | Le système valide les informations, enregistre dans la base de données et redirige l'utilisateur vers la page de connexion. |

**Scenario d’exception :**

|  |  |  |
| --- | --- | --- |
| Etapes | Acteurs | Système |
| 3.a.1 | L'utilisateur ne renseigne pas de nom d'utilisateur |  |
| 3.a.2 |  | Le système affiche Un message d'erreur s'affiche : **"Le nom d'utilisateur est requis."** |
| 3.b.1 | L'utilisateur entre un email au format incorrect |  |
| 3.b.2 |  | Le système affiche Un message d'erreur s'affiche : **"L'adresse email n'est pas valide."** |
| 3.c.1 | L'utilisateur tente d'utiliser une adresse email déjà enregistrée dans la base de données. |  |
| 3.c.2 |  | Le système affiche Un message d'erreur s'affiche : **"Cette adresse email est déjà utilisée."** |
| 3.d.1 | L'utilisateur ne renseigne pas de mot de passe. |  |
| 3.d.2 |  | Le système affiche Un message d'erreur s'affiche : "**Le mot de passe est requis."** |

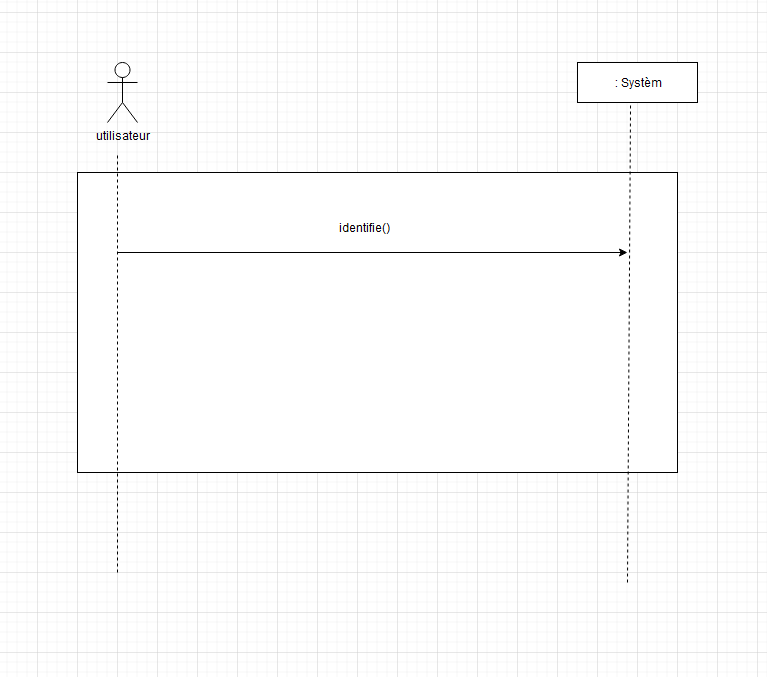
Signature :

**Modèle conceptuel :**

****

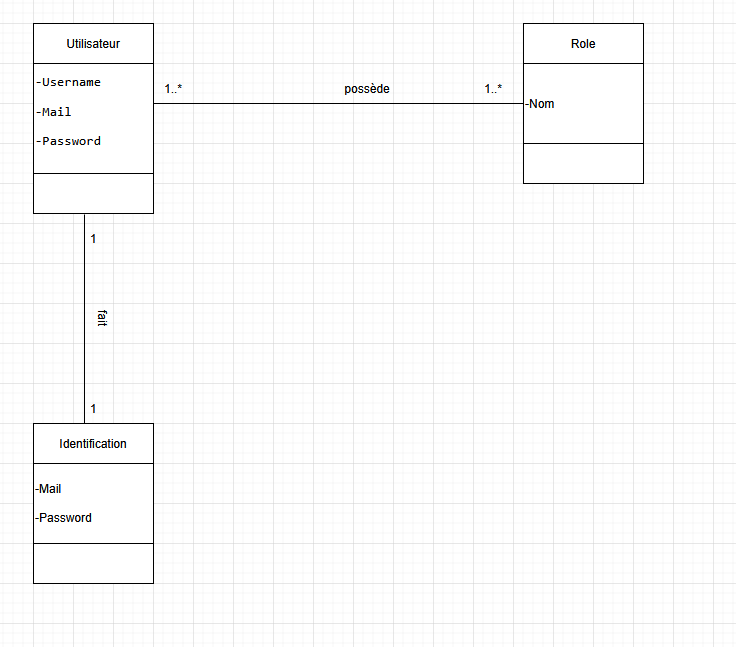
Signature :

**Système de Séquence Diagram (SSD) :**

**** Signature :

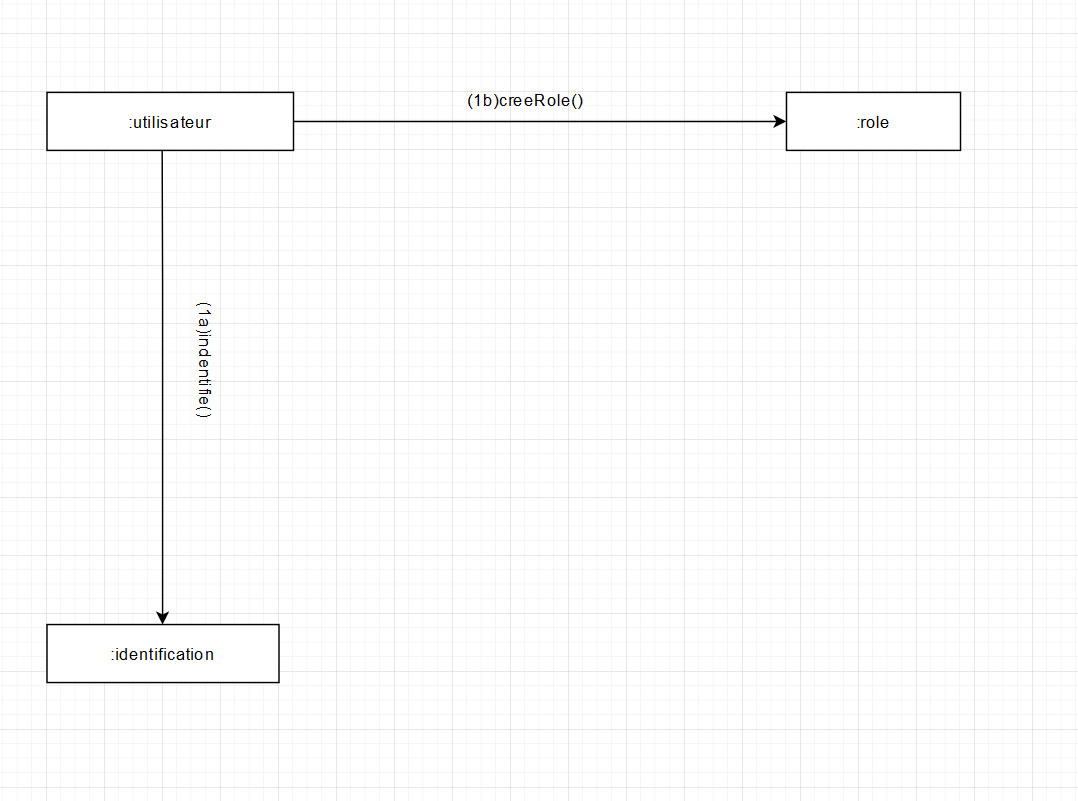
**Conception :**

**Class Diagram incomplet :**

****

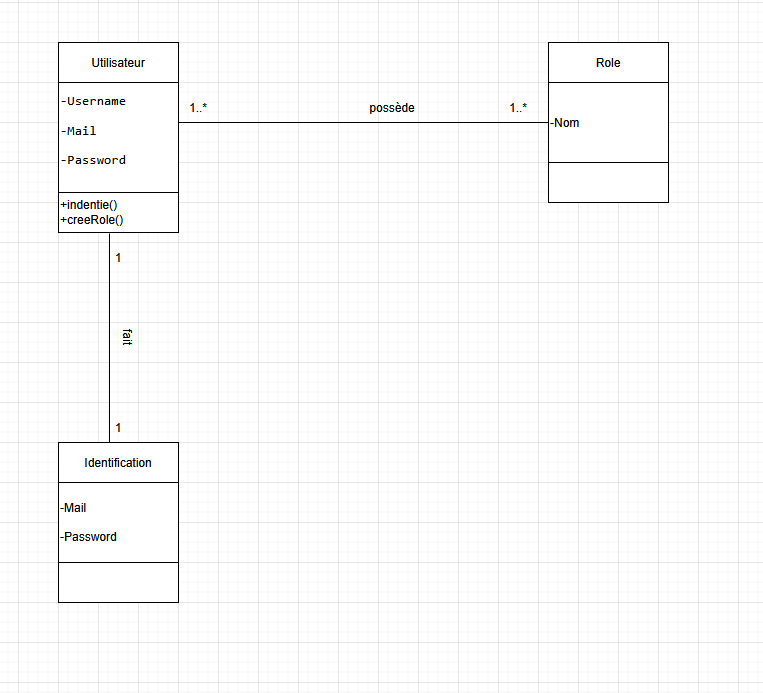
Signature :

**Collaboration Diagram :**

****

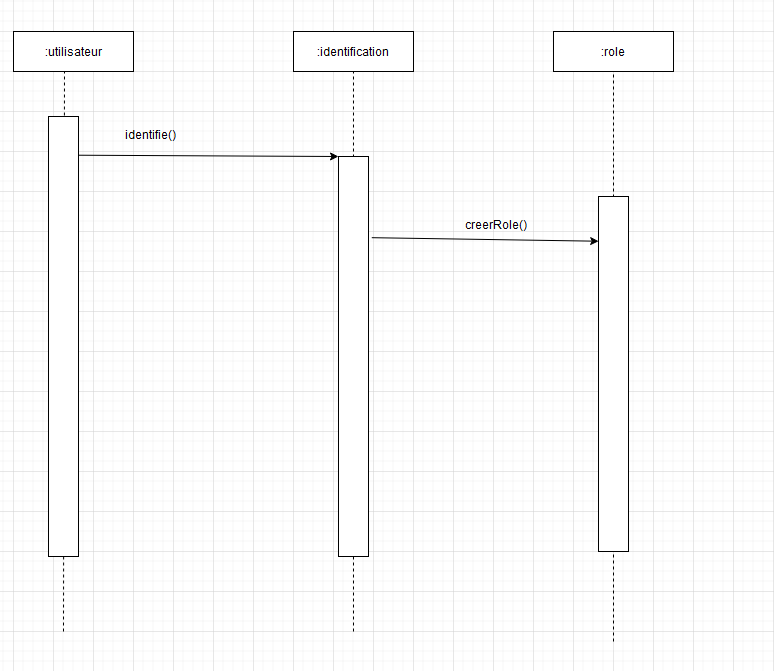
Signature :

**Class Diagram complet :**

****

Signature :

**Séquence Diagram :**



Signature :

**Use case 2 : Se connecter :**

**Acteur principal** : Utilisateur (Joueur) : l’utilisateur de se connecter à son compte en utilisant son email et son mot de passe.

**Préconditions :** L’utilisateur dispose déjà d'un compte enregistré avec un email et un mot de passe

**Postconditions :** L’utilisateur est connecté à son compte et redirigé vers son tableau de bord ou page d’accueil

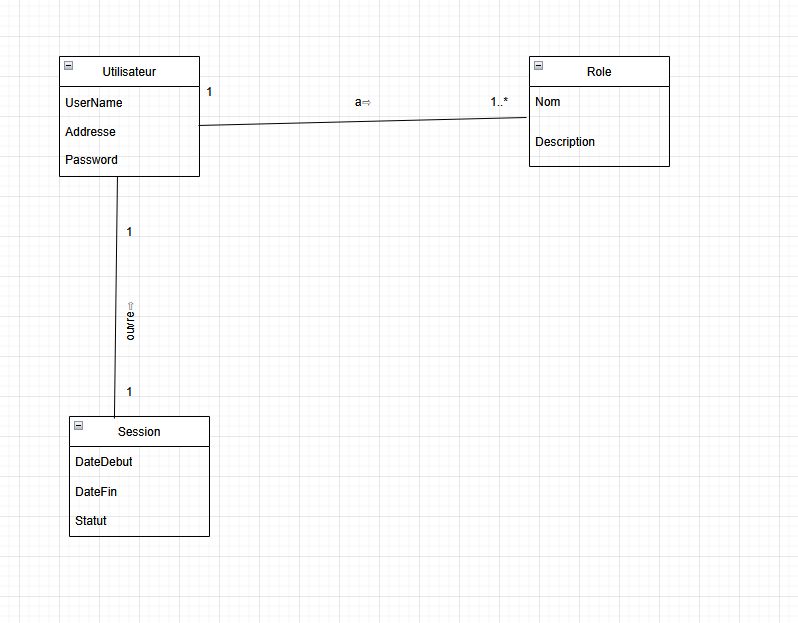
**Scénario principal :**

|  |  |  |
| --- | --- | --- |
| Etapes | Acteurs | Système |
| 1 | L’use case commence quand l’utilisateur clique sur l’option « Se connecter » |  |
| 2 |  | Le système affiche un formulaire de connexion (email + mot de passe) il affiche en dessous un bouton « se connecter » |
| **3** | L'utilisateur saisit son email et son mot de passe, il clique ensuite sur le bouton. |  |
| 4 |  | Le système authentifie l'utilisateur en vérifiant que l'email existe dans la base de données et que le mot de passe hashé correspond, avant de le rediriger vers sa page d'accueil. |

**Scénarios d’exception :**

|  |  |  |
| --- | --- | --- |
| Etapes | Acteurs | Système |
| 1 |  | Les identifiants sont incorrects : Le système affiche un message d’erreur : "Email ou mot de passe  Incorrect. |
| 2 |  | L’utilisateur oublie de remplir un champ : Le système affiche un message : "Tous les champs sont  Obligatoires." |
| **3** |  | Le compte est désactivé où  Verrouillé :Le système affiche un message : "Votre compte est désactivé. Veuillez contacter un administrateur." |

**Modèle conceptuel :**



Signature :

**Système de Séquence Diagram (SSD) :**

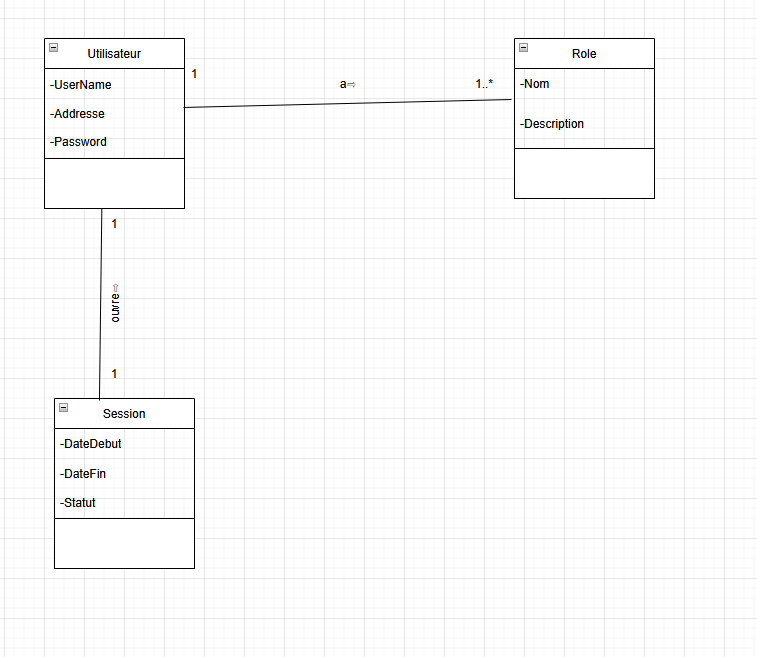
**Une image contenant diagramme, texte, ligne, Rectangle

Le contenu généré par l’IA peut être incorrect.**

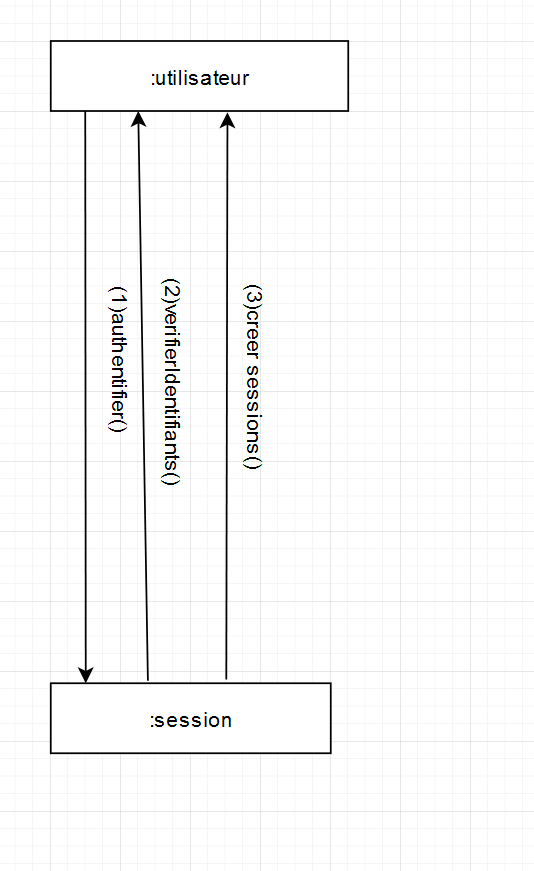
Signature :

**Conception :**

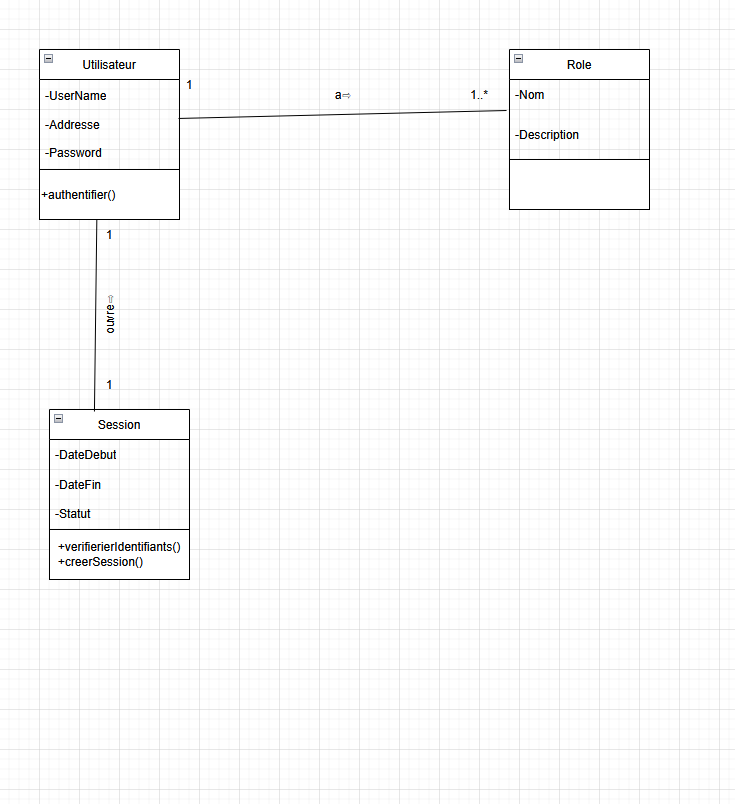
**Class Diagram incomplet :**

****

**Collaboration Diagram :**



**Class Diagram complet :**



**Séquence Diagram :**

