Sujet de projet : Dressage.
Les auteurs : Lefi Manel && Kheriji Rahma.
Date : Saisi le 08/04/2023.

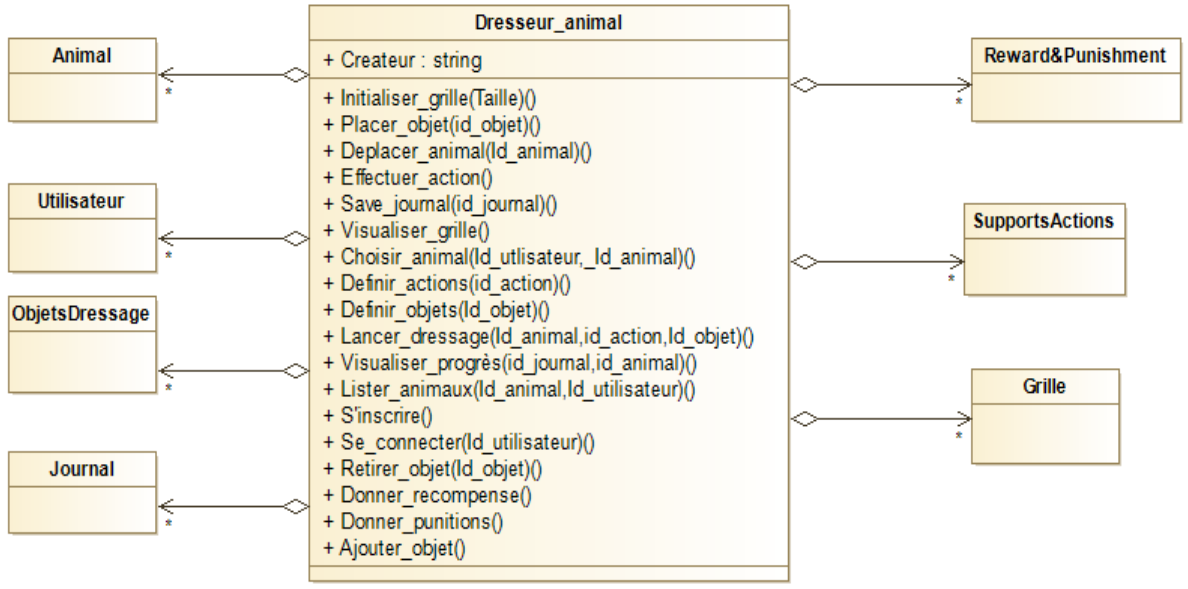
Liste des classes:

Classes	Attributs	Description	Types	Opérations
★ Grille	-Taille -Animal -Objet	-Cette classe représente l'environnement de grille N x N cases dans lequel l'animal évolue. Elle contient des informations sur la taille de la grille et les objets qui sont placés sur la grille (animal , objets de dressage).	-Entier -Animal -ObjetsDressage	init(taille) deplacerAnimal() supprimerObjet() getObjet_RP() getObjet_Act()
★ Animal	-Id_animal -Etat_d -Catégorie -Race -Nom -Couleur -Journal	-Cette classe interagit d'une façon directe avec la grille.Elle contient toutes les informations qui concerne l'animal tels que son état actuel, son nom, ainsi que des méthodes pour permettre à l'animal d'effectuer des actions et de tenir un journal	-Chaîne -Booléen -Chaîne -Chaîne -Chaîne -Chaîne -Journal	init(taille) getJournal(id_journal)
★ Utilisateur	-Id_utilisateur -Etat_c -Nom -Prénom -Âge -Date_inscrit	-Cette classe dispose toutes les informations qui concernent l'utilisateur qui est en train de dresser son animal en appliquant les actions de son choix.	-Chaîne -Booléen -Chaîne -Chaîne -Entier -Chaîne	seConnecter() s'inscrire() dresserAnimal() choisirObjets() accesJournal()
★ Journal	-Id_journal -Id_utilisateur -Id_animal -Actions -Récompenses Punitions -Date_debut -Date_fin -Etat_initial -Etat_final	-La classe journal regroupe toutes les informations sur l'animal qui est en train d'être dressé (son progrès,ses actions,ses récompenses,ses punitions).	-Chaîne -Chaîne -Chaîne -Chaîne -Reward&Punishment -Reward&Punishment -Date -Date -Chaîne -Chaîne	getUtilisateur() getAnimal() donnerActions() durée(date_debut,date_fin) changerEtat(etat_init,etat_fin)

★ ObjetsDressage	-Id_objet -Catégorie -Nom_objet	-Cette classe contient tous les objets que l'utilisateur peut sélectionner pour dresser son animal(carotte\ cerceaux).	-Chaîne -Chaîne -Chaîne	getNom() getCategorie()
★ Reward& Punishment	-Reward -Punishment -Description -Objet_RP	-Cette classe regroupe l'ensemble des récompenses et punitions définies par l'utilisateur.	-Chaîne -Chaîne -Chaîne -ObjetsDressage	getReward() getPunishment()
★ Supports Actions	-Id_action -Description -Objet_Act	-La classe SupportsActions contient	-Chaîne -Chaîne -ObjetsDressage	getDescription()
★ Utilisateur_inscrit	-Login -Password -Nb_animaux	-Cette classe contient les données des utilisateurs qui sont déjà inscrits (utilisateur type 1).	-Chaîne -Chaîne -Entier	
★ Utilisateur_visiteur	-Pages_consultés	-Cette classe concerne les utilisateurs qui ne sont pas encore inscrits dans le système (utilisateur type 0).	-Chaîne	
★ Animal_dressé	-Parcours	-La classe Animal_dressé contient les informations des animaux déjà dressés (animal type 1).	-Chaîne	
★ Animal_non_dressé	-Caractère	-Cette classe contient des informations sur l'état de l'animal, ainsi que des méthodes pour permettre à l'utilisateur d'interagir avec l'animal et de commencer à le dresser (animal type 0).	-Chaîne	

Façade :

- La classe "Dresseur d'animal" pourrait contenir des méthodes pour permettre à l'utilisateur de visualiser l'état actuel de l'environnement de grille et de l'animal, et pour afficher les informations stockées dans le journal. En utilisant cette façade, l'utilisateur pourrait interagir avec le système de manière simplifiée et intuitive, sans avoir besoin de connaître tous les détails de mise en œuvre du système.



Les relations:

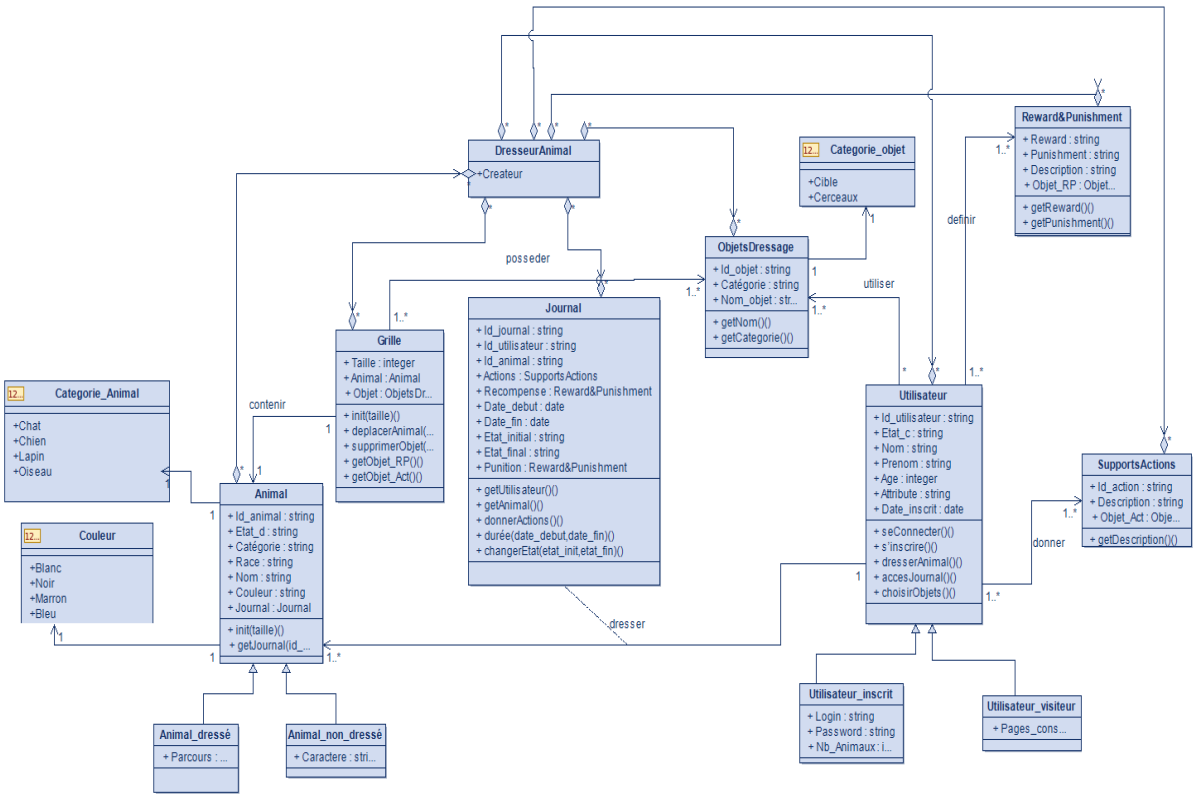
Pour mieux expliquer notre diagramme de classe on a opter pour ces 6 relations :

- La relation “dresser” : - La classe “Animal” peut avoir une relation d'agrégation avec la classe “Utilisateur” cela signifie que chaque utilisateur peut dresser un ou plusieurs animaux mais un animal peut être dressé par un seul utilisateur .
- La relation “contenir” : - La classe “Animal” peut avoir une relation d'agrégation avec la classe “grille” cela signifie que chaque animal peut trouver dans une seule grille et une grille peut contenir un et un seul animal.
- La relation “posséder”: - La classe “grille” peut avoir une relation d'agrégation avec la classe “ObjetsDressage” cela signifie qu'une grille peut contenir au moins un objet et un objet peut être trouvé dans plusieurs grilles.
- La relation “utiliser” : -La classe “utilisateur” peut avoir une relation d'agrégation avec la classe “ObjetsDressage” cela signifie que l’utilisateur peut utiliser plusieurs objets

pour dresser son animal et chaque objet est utilisé par 0 ou plusieurs utilisateurs à la fois.

- La relation “définir” : -La classe “utilisateur” peut avoir une relation d'agrégation avec la classe “Reward&Punishment” cela signifie que l’utilisateur peut définir un ou plusieurs récompenses\punitions et chaque récompenses\punitions est définie par 1 ou plusieurs utilisateurs.
- La relation “donner” : -La classe “utilisateur” peut avoir une relation d'agrégation avec la classe “SupportsActions” cela signifie que l’utilisateur peut donner un ou plusieurs actions et chaque action est définie par 1 ou plusieurs utilisateurs.

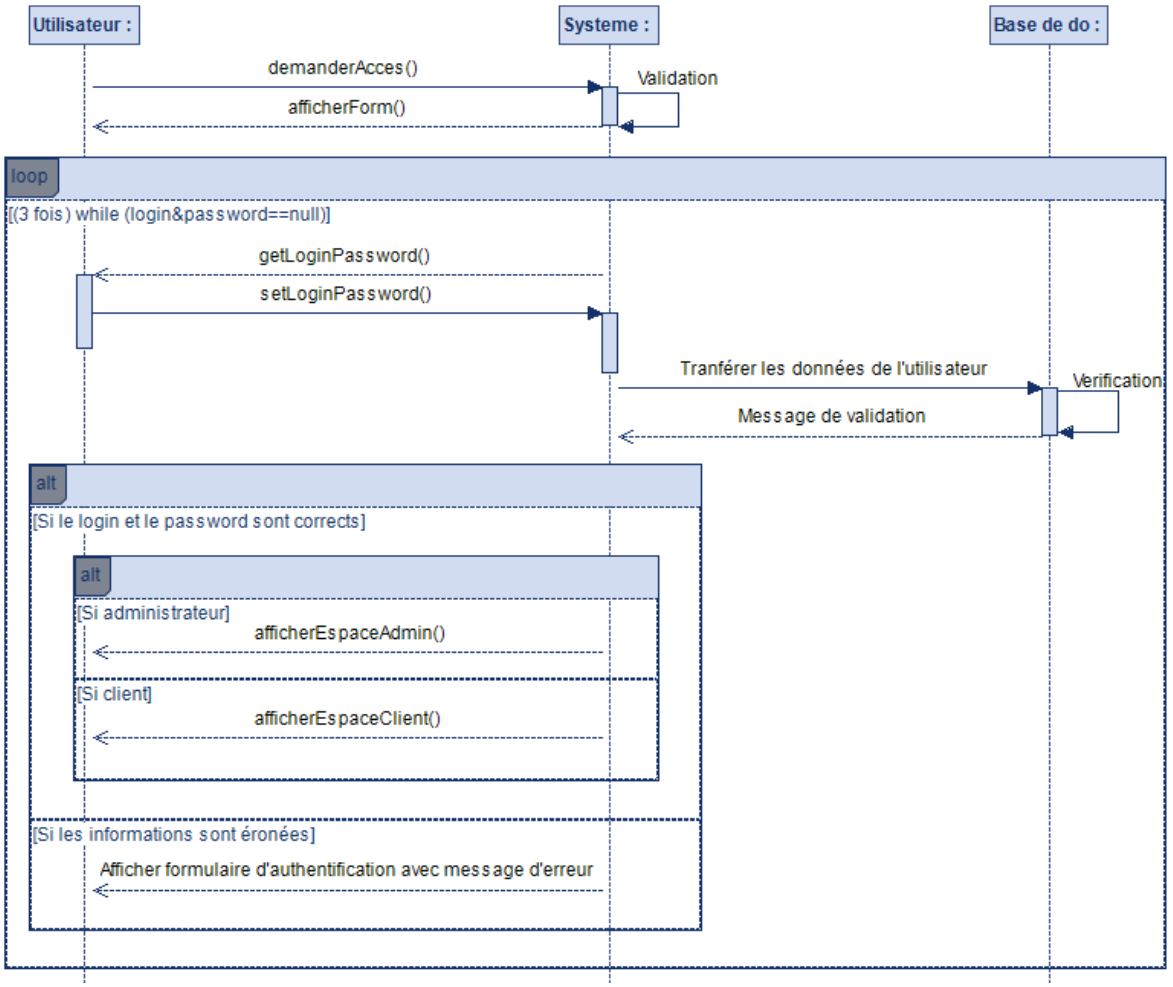
Diagramme de classe :



Diagrammes de séquence:

1) Diagramme de séquence du use case “S’authentifier”:

Description : 1- L'utilisateur ouvre l'interface de connexion du système et demande un accès au formulaire. 2- Le système affiche un formulaire de connexion comprenant deux champs : un champ pour le login et un champ pour le mot de passe. 3- L'utilisateur saisit son login et son mot de passe et répète cette action jusqu’à ce que les données soient compatibles au format demandé. 4- Le système vérifie les identifiants de l'utilisateur avec la base de données. 5- Si les identifiants sont valides, l'utilisateur est authentifié et redirigé vers la page d'accueil du système (soit espace admin soit espace client) .Si les identifiants sont invalides, un message d'erreur est affiché à l'utilisateur et il doit réessayer de se connecter en fournissant des identifiants valides.



2) Diagramme de séquence du use case “Définir les comportements désirés”:

Scénario principal :

1- L’utilisateur s’authentifie par son login et password. 2- L'utilisateur sélectionne un animal pour lequel il souhaite définir les comportements désirés. 3- Le système affiche un formulaire de comportements possibles pour l'animal que l’utilisateur doit remplir à son choix. 4- L’utilisateur valide ses actions donnés. 5- Les comportements désirés de l'animal sont enregistrés dans le système de dressage.

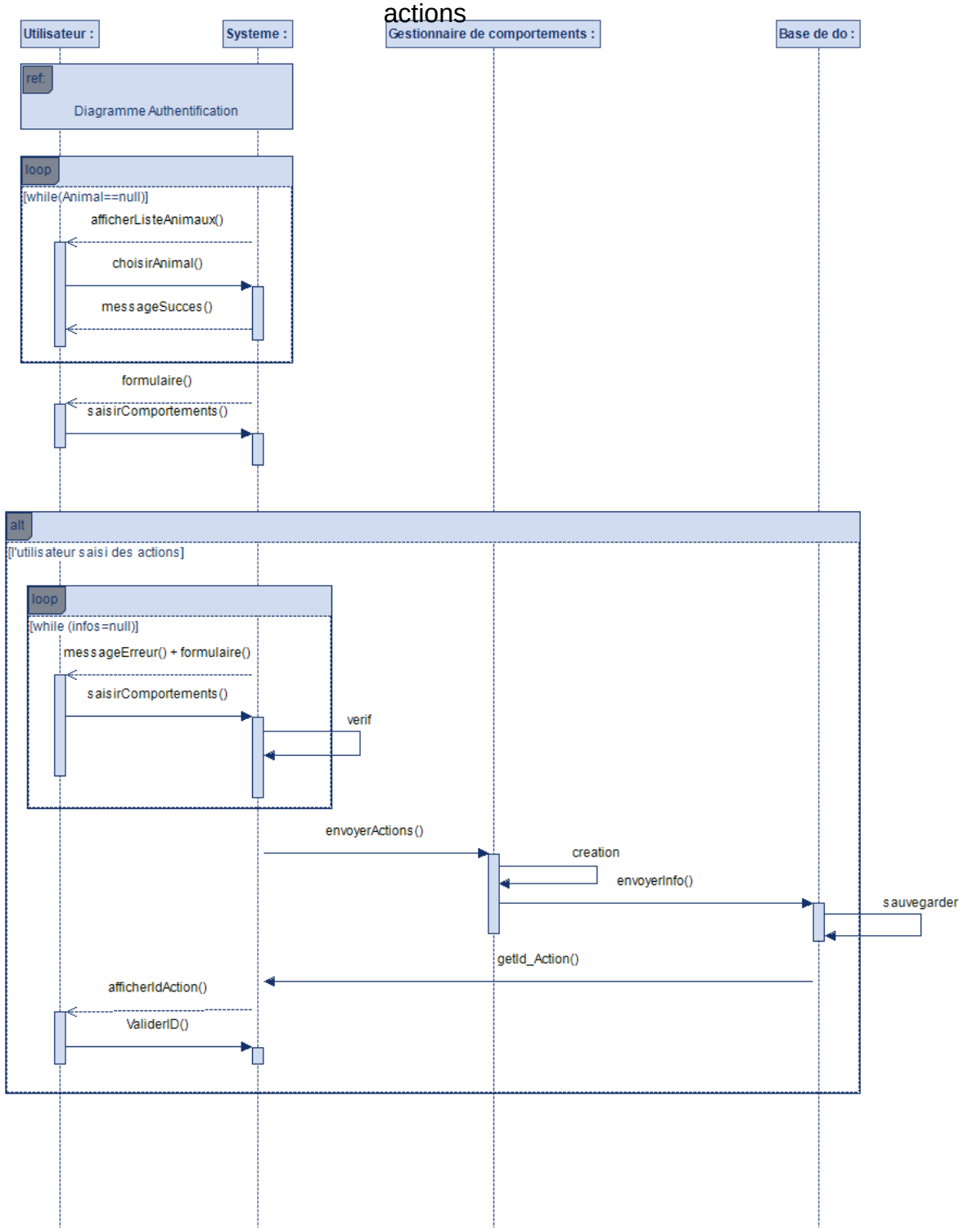
Scénarios alternatifs :

- Si l'animal sélectionné n'existe pas dans le système, un message d'erreur est affiché à l'utilisateur et il doit sélectionner un animal existant dans le formulaire.
- Si la liste de comportements saisie par l'utilisateur est mal formée, un message d'erreur est affiché à l'utilisateur et il doit respecter des conditions valides.

ANIMAL

à effacer

actions



3) Diagramme de séquence du use case “Enregistrer le journal de l’animal”:

Scénario principal :

1- L'utilisateur se connecte au système en fournissant son login et son mot de passe. 2- Si les identifiants sont valides, le système demande l’id de l’animal pour référencer son journal. 3- L'utilisateur enregistre le journal en fournissant les informations sur les actions, récompenses et punitions. 4- Si les informations fournies sont valides, le système met à jour le type de l'animal à 1, enregistre le journal et affiche un message de succès à l'utilisateur. 5- Le journal est enregistré et consultable à tout moment.

Scénarios alternatifs:

- Si les identifiants sont invalides, le système affiche un message d'erreur à l'utilisateur.
- Si l’id de l’animal est introuvable dans la liste, le système affiche un message d'erreur à l'utilisateur.
- Si les informations fournies sont invalides, le système affiche un message d'erreur à l'utilisateur.