

RAPPORT PROJET GL

Application de rôliste

Réalisé par :

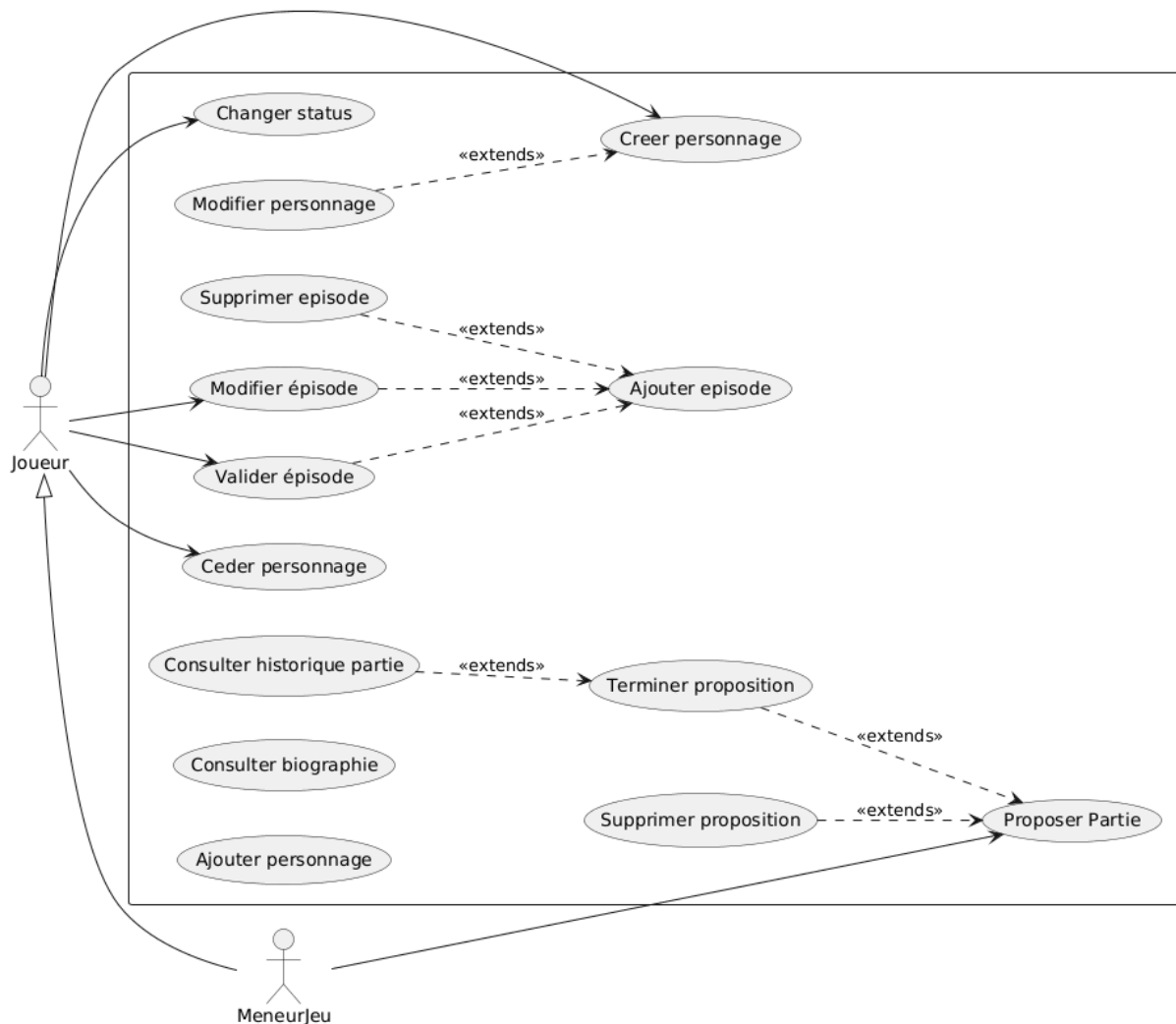
❖ *DOUANLA MELI Scott*

❖ *NGUEGANG KOWO Laura Stefy*

DOCUMENT D'ANALYSE.....	3
1. Diagramme de cas d'utilisation.....	3
Diagramme de séquence système : Créer personnage.....	4
Diagramme de séquence système : Ajouter Episode.....	5
Diagramme de séquence système : Valider épisode.....	6
Diagramme de séquence système : Proposer partie.....	6
2. Diagramme de classe d'analyse.....	7
DOCUMENT DE CONCEPTION.....	8
1. Implémentation du MVC.....	8
2. Diagramme de classe logicielle.....	9
A. Package modèle.....	9
B. Package Vue.....	10
C. Package Controller.....	11
D. Relation entre les différents package.....	12
L'objet SQLiteManager fait office de point d'entrée unique pour la gestion de la base de données. Fournisseur de connexion, Il encapsule la logique d'ouverture et de fermeture des sessions JDBC, garantissant que chaque DAO utilise une instance de connexion valide et optimisée.....	17
3. Diagramme de séquence technique.....	18
• Ajouter Episode.....	18
• Ajouter Personnage.....	18
• Proposer partie.....	19
4. Diagramme d'état transition.....	20
• Etat transition Episode.....	20
• État transition partie.....	21
• État transition épisode.....	21
MANUEL UTILISATEUR.....	22
1. Présentation.....	22
2. Démarrage rapide.....	22
3. Guide des fonctionnalités.....	23
A. Gestion des parties.....	24
B. Gestion des personnages.....	27
C. Gestion de la biographie des personnages.....	28
D. Focus sur les meneurs de jeu.....	30
BILAN DES TECHNOLOGIES DE MODÉLISATION UTILISÉES.....	31

DOCUMENT D'ANALYSE

1. Diagramme de cas d'utilisation



Description textuelle des cas d'utilisation:

- Créer personnage

Ce cas d'utilisation permet au Joueur de créer un nouveau personnage au sein du système. Le Joueur renseigne les informations nécessaires à la création, afin de pouvoir participer à une partie avec ce personnage.

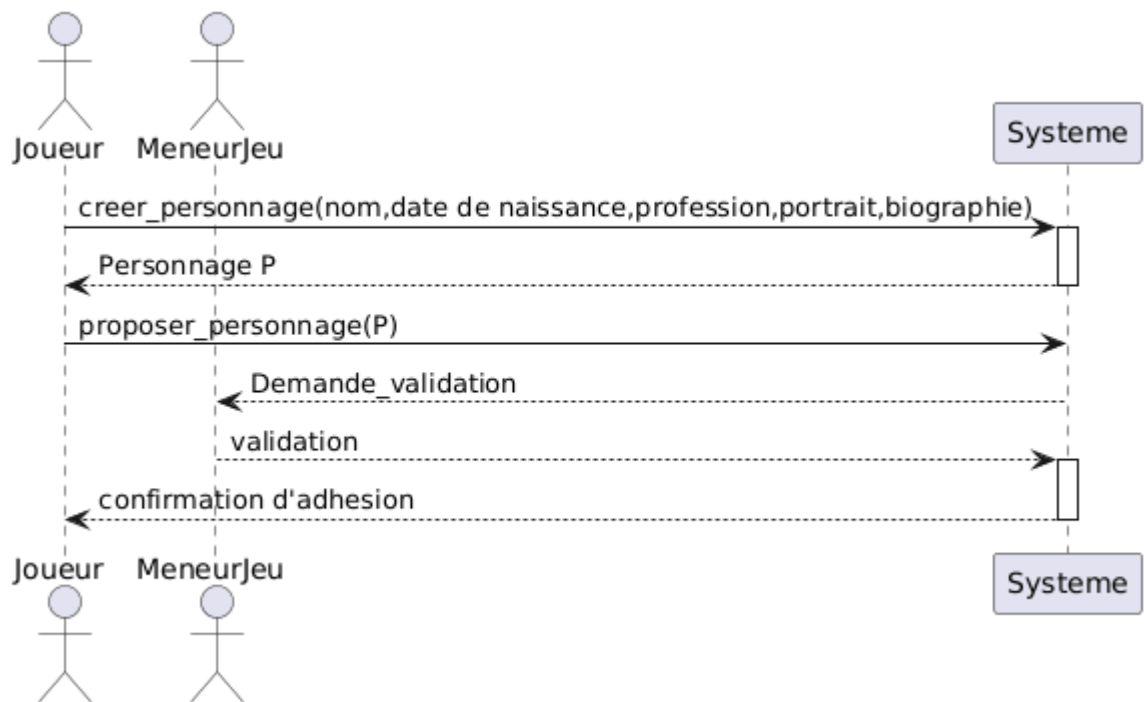


Diagramme de séquence système : Créer personnage

- Modifier personnage
Ce cas d'utilisation permet au Joueur de modifier les caractéristiques d'un personnage existant. Il s'agit d'une extension du cas d'utilisation *Créer personnage*, permettant d'ajuster ou de corriger les informations du personnage après sa création.
- Céder personnage
Ce cas d'utilisation permet au Joueur de céder la propriété d'un personnage à un autre joueur, conformément aux règles définies par le système.
- Changer statut
Ce cas d'utilisation permet au Joueur de modifier le statut de son personnage, par exemple pour indiquer sa disponibilité ou son état dans la partie.
- Ajouter épisode
Ce cas d'utilisation permet d'ajouter un nouvel épisode à une partie. Il constitue une fonctionnalité centrale sur laquelle reposent plusieurs autres actions liées à la gestion narrative.

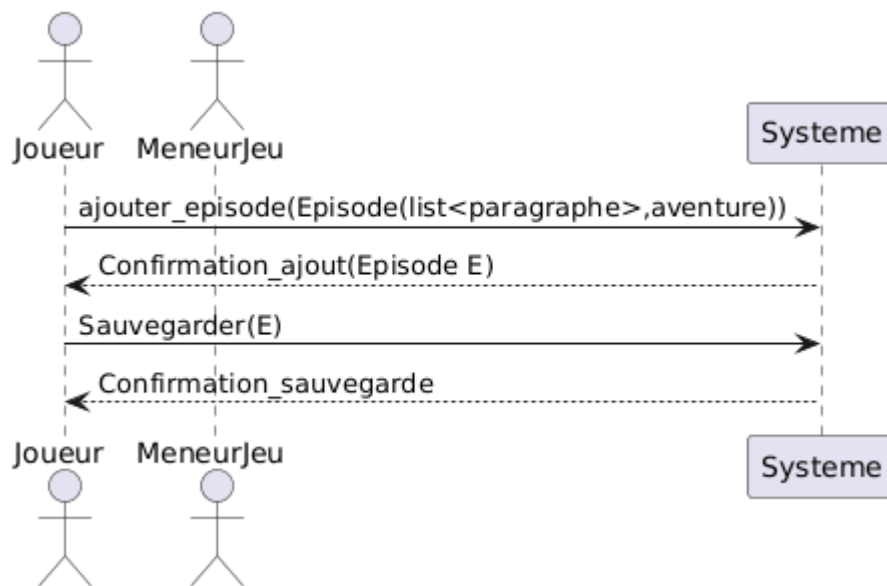


Diagramme de séquence système : Ajouter Episode

- Modifier épisode
Ce cas d'utilisation permet au Joueur de modifier un épisode existant. Il étend le cas d'utilisation *Ajouter épisode* en autorisant la modification du contenu narratif.
- Valider épisode
Ce cas d'utilisation permet au Joueur de valider un épisode afin de confirmer son intégration définitive dans le déroulement de la partie. Il s'agit également d'une extension du cas d'utilisation *Ajouter épisode*.

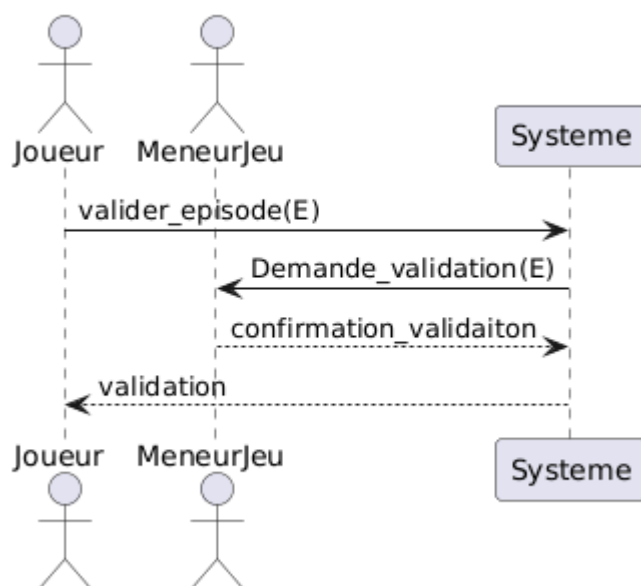


Diagramme de séquence système : Valider épisode

- Supprimer épisode
Ce cas d'utilisation permet de supprimer un épisode précédemment ajouté à une partie. Il étend le cas d'utilisation *Ajouter épisode* afin de gérer les corrections ou suppressions nécessaires.
- Consulter biographie
Ce cas d'utilisation permet au Joueur de consulter la biographie d'un personnage afin d'obtenir des informations détaillées sur son historique et ses caractéristiques.
- Proposer partie
Ce cas d'utilisation permet au Meneur de Jeu de proposer une nouvelle partie aux joueurs. Il s'agit de l'action initiale de création et d'organisation d'une partie.

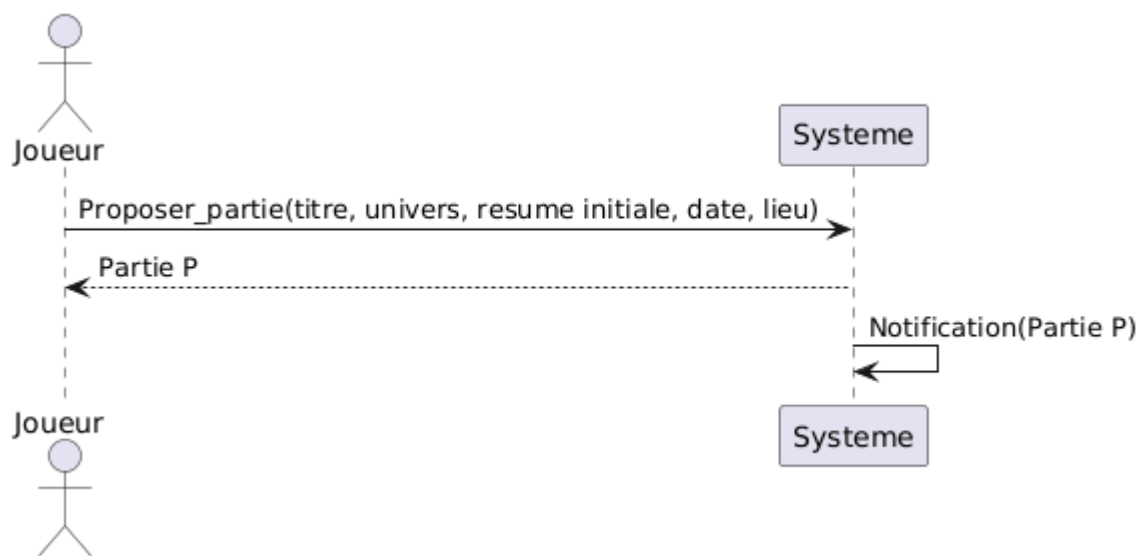


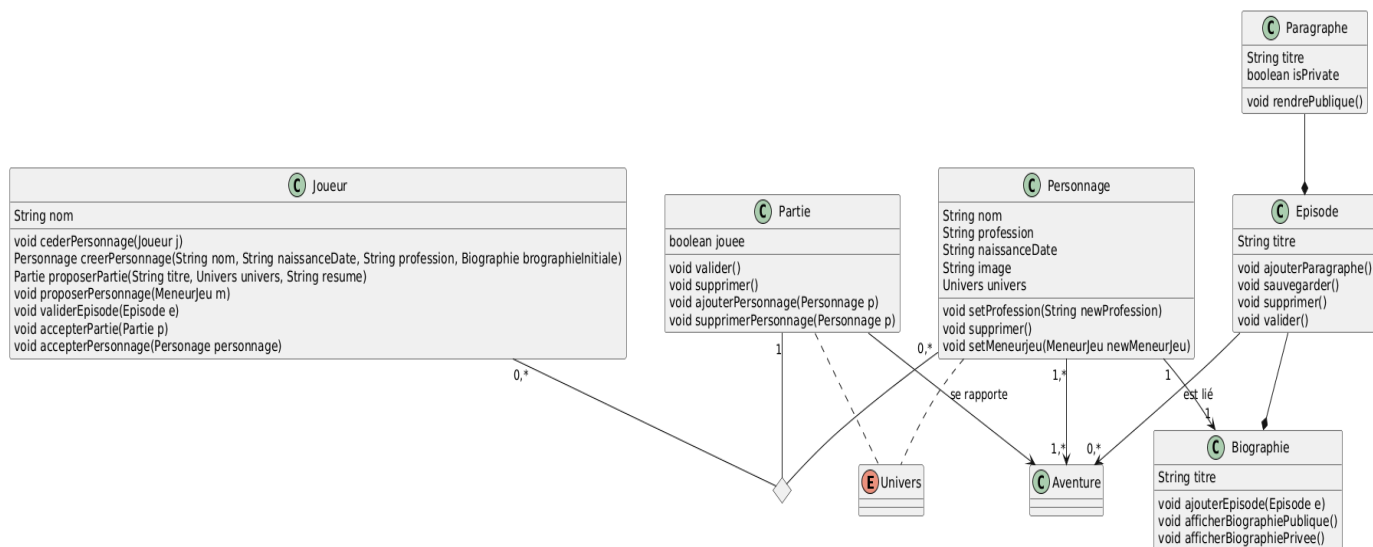
Diagramme de séquence système : Proposer partie

- Supprimer proposition
Ce cas d'utilisation permet au Meneur de Jeu de supprimer une proposition de partie existante. Il étend le cas d'utilisation *Proposer partie*.
- Terminer proposition
Ce cas d'utilisation permet au Meneur de Jeu de mettre fin à une

proposition de partie. Il étend également le cas d'utilisation *Proposer partie*.

- Consulter historique partie
Ce cas d'utilisation permet de consulter l'historique d'une partie terminée. Il s'agit d'une extension du cas d'utilisation *Terminer proposition*, permettant de retracer les événements passés.

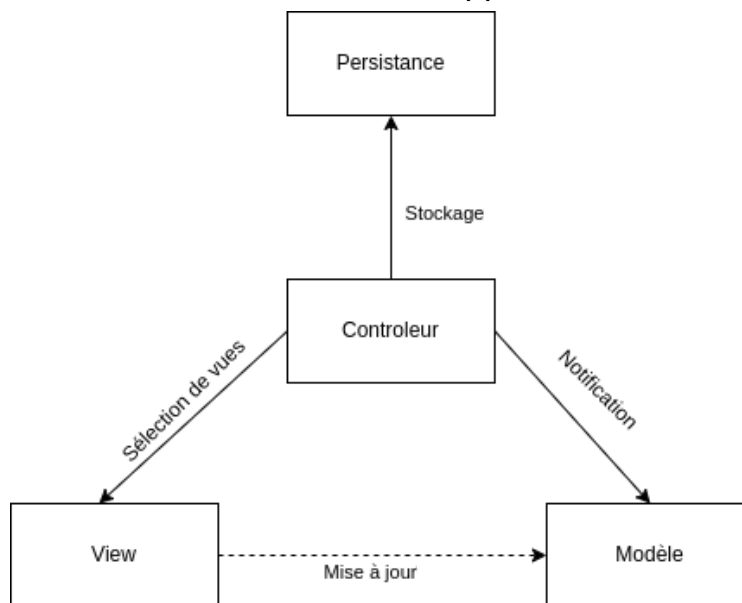
2. Diagramme de classe d'analyse



DOCUMENT DE CONCEPTION

1. Implémentation du MVC

L'architecture MVC de notre application est de la forme :

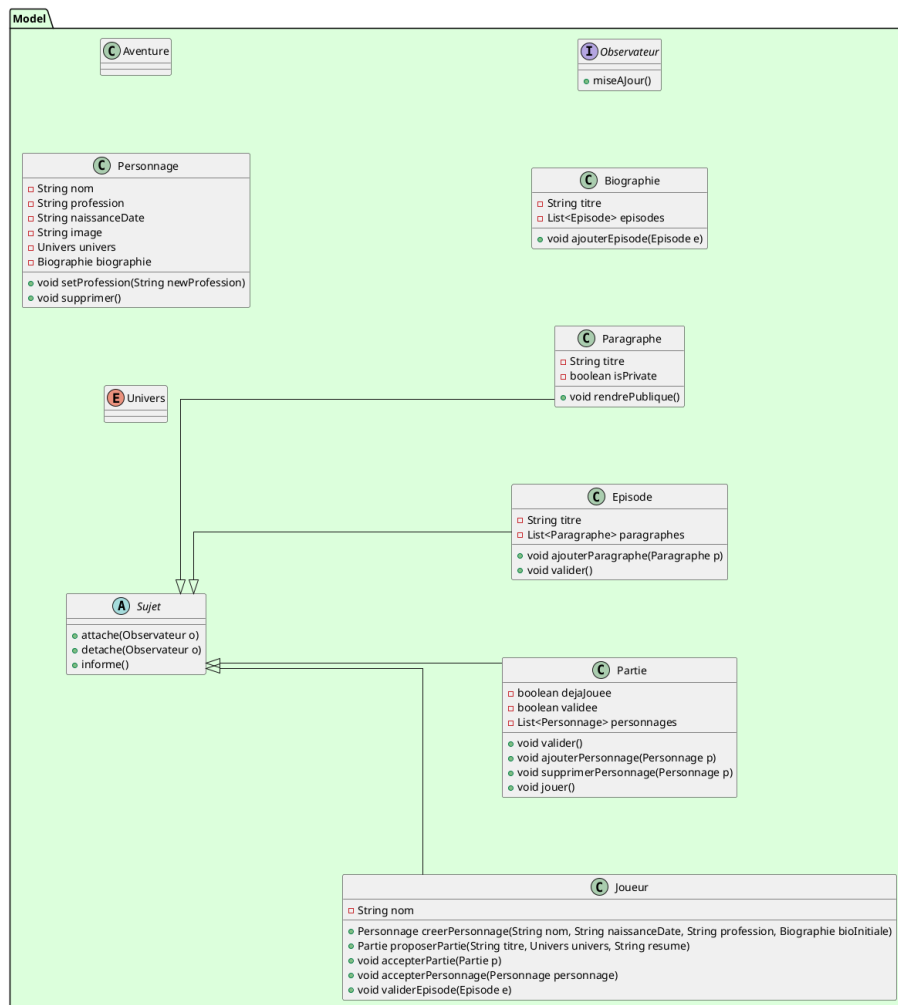


Ainsi, l'application repose sur un découpage rigoureux en quatre pôles de responsabilité, optimisant la maintenance et l'évolution du système :

- **Le Contrôleur comme Chef d'Orchestre** : Contrairement à un modèle MVC classique où la vue peut interagir avec le modèle, ici le Contrôleur centralise toute la logique applicative. Il assure la **sélection des vues**, notifie le **Modèle** des changements d'état et gère le **stockage** via la couche de persistance.
- **Synchronisation par le Pattern Observateur** : Pour maintenir un couplage faible tout en assurant une interface réactive, la **Vue** s'appuie sur le patron de conception **Observateur**. Elle s'abonne aux changements du **Modèle** pour déclencher une mise à jour automatique de l'affichage dès qu'une donnée métier évolue.
- **Isolation de la Persistance** : L'intégration explicite d'un bloc "Persistance" (DAOs) au-dessus du Contrôleur souligne que la gestion du stockage est traitée comme une ressource externe pilotée exclusivement par la logique de contrôle, garantissant que le **Modèle** reste pur et indépendant de toute technologie de base de données.

2. Diagramme de classe logicielle

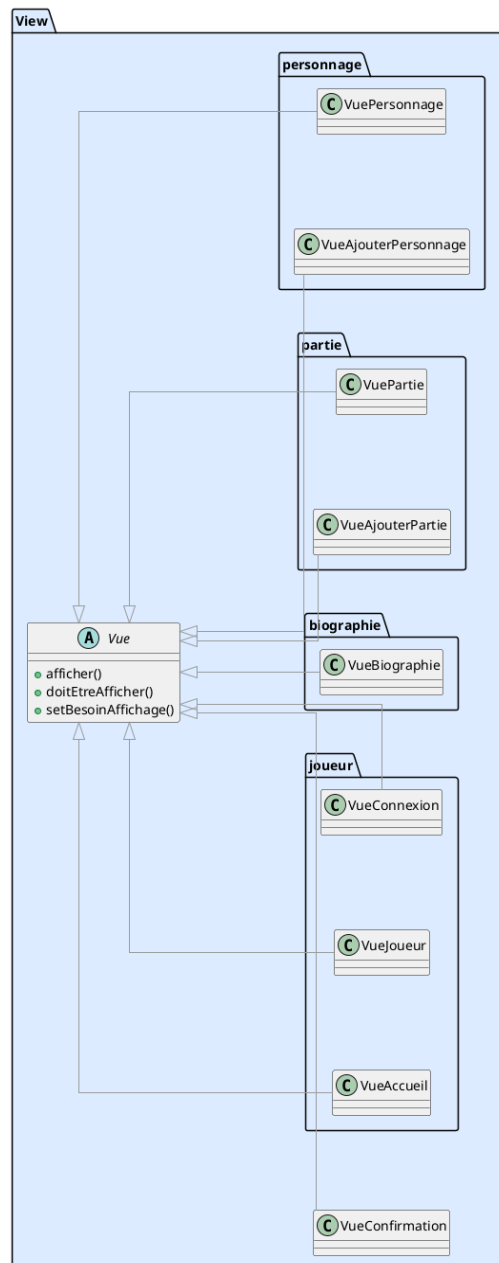
A. Package modèle



La logique de conception repose sur une architecture découplée, centrée sur le patron de conception Observateur. En faisant hériter l'ensemble des entités métier (Joueur, Partie, Personnage, etc.) de la classe abstraite Sujet, le modèle garantit une synchronisation automatique et transparente avec l'interface utilisateur (l'Observateur). Cette approche permet de notifier les vues de tout changement d'état du modèle sans que les entités métier n'aient connaissance des détails de l'affichage. Cette structure favorise une séparation stricte des responsabilités (principe du MVC), où le modèle gère exclusivement la cohérence des données de jeu de rôle (univers, biographies,

épisodes) tout en offrant une réactivité en temps réel aux actions de l'utilisateur.

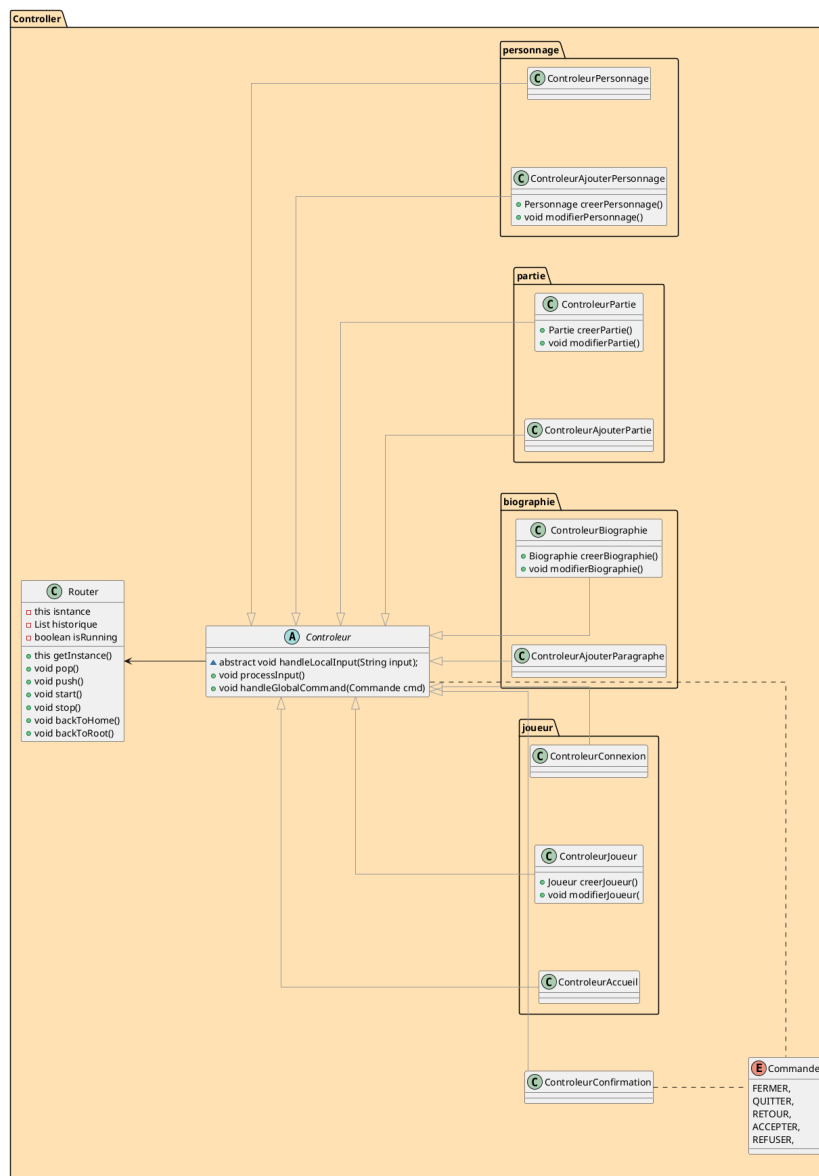
B. Package Vue



La logique de conception de la Vue s'inscrit dans une dynamique réactive pilotée par le patron de conception Observateur. En héritant d'une base commune, toutes les vues concrètes sont prêtes à réagir aux notifications provenant du Modèle. La présence d'un état interne de "besoin d'affichage" permet de transformer une interface console traditionnellement statique en un système dynamique capable d'optimiser ses cycles de rendu.

Cette structure permet de séparer strictement l'affichage pur du traitement des entrées, laissant aux contrôleurs le soin de manipuler ces vues via les méthodes publiques héritées, assurant ainsi une transition fluide entre les différents menus de l'univers de jeu.

C. Package Controller

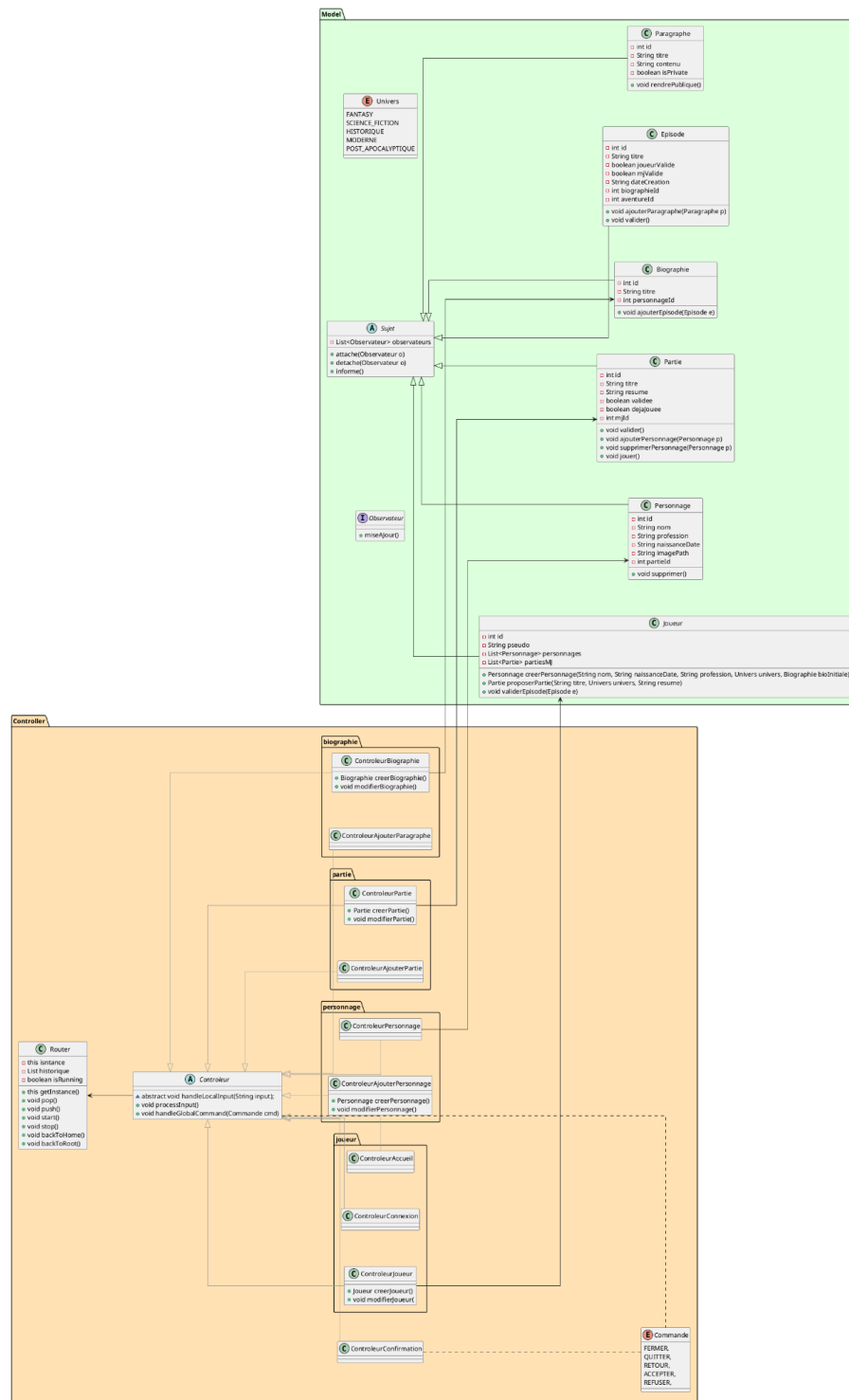


L'architecture de contrôle de l'application repose sur une synergie entre le patron de conception **Médiateur** (via le Router qui est un singleton) et une structure de **Machine à États** (via les contrôleurs spécialisés). La classe abstraite **Controleur** sert de socle commun, offrant une interface standardisée pour le traitement des flux d'entrée grâce à la méthode `processInput()` et la gestion unifiée des intentions utilisateurs via l'énumération **Commande**.

D. Relation entre les différents package

- **Contrôleur → Modèle**

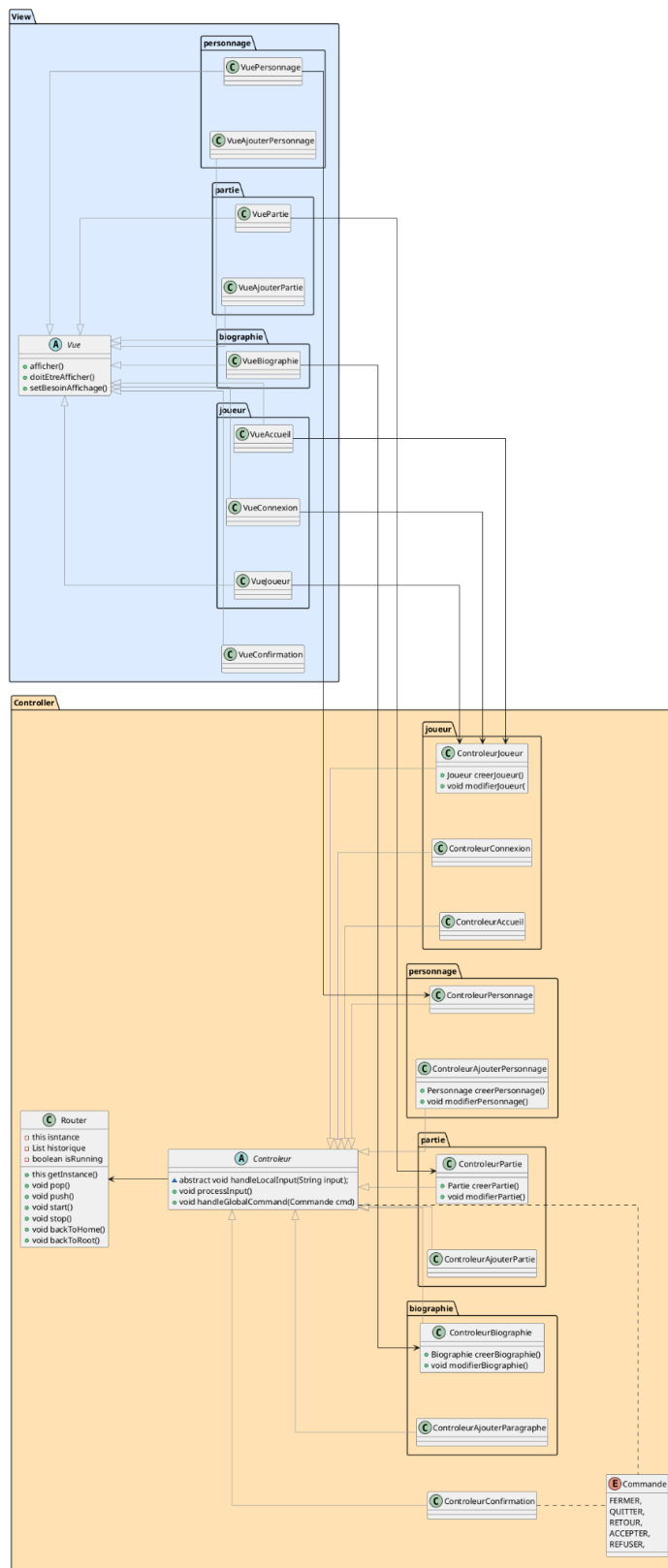
La logique de conception de l'application repose sur un **couplage structuré** entre la couche de contrôle et le modèle métier, facilitant la gestion de l'état et la persistance. Chaque contrôleur spécialisé (ex: `ContrôleurPartie`, `ContrôleurBiographie`) agit comme un médiateur qui traduit les intentions de l'utilisateur en modifications directes sur les objets du domaine. Cette architecture est renforcée par le patron de conception **Observateur**, où chaque entité du modèle (`Partie`, `Joueur`, `Episode`, etc.) hérite de la classe `Sujet`. Ce mécanisme permet aux entités métier de notifier les changements d'état de manière autonome, garantissant que toute action de modification initiée par un contrôleur entraîne une mise à jour cohérente et automatique des vues associées sans créer de dépendances circulaires. L'organisation modulaire en packages thématiques assure ainsi une séparation claire des responsabilités, où les contrôleurs gèrent le flux opérationnel tandis que le modèle encapsule les règles métier et la diffusion des données.



● Contrôleur → Vue

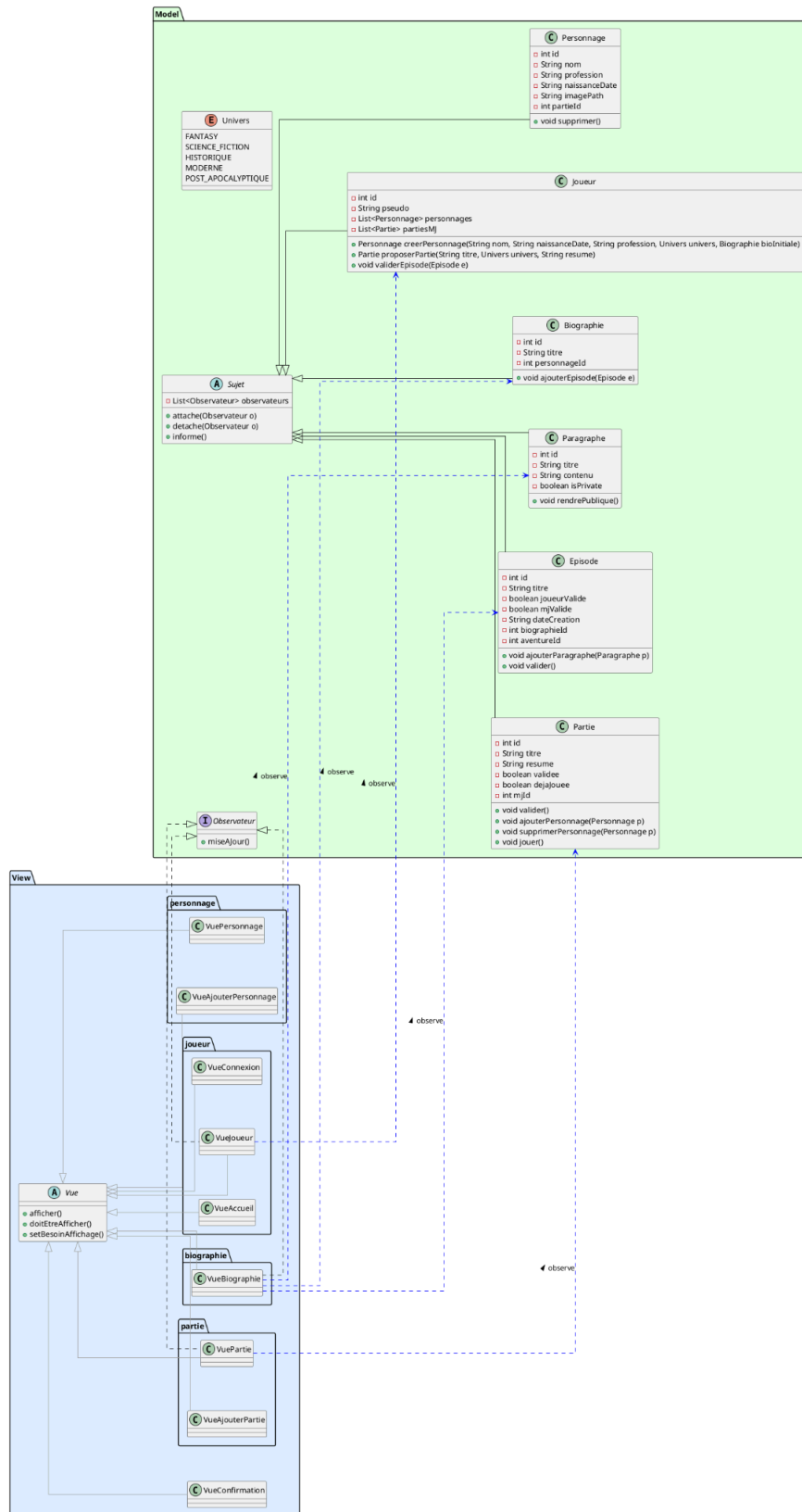
Ici est mis en évidence un couplage fonctionnel direct : **chaque vue délègue la gestion des événements à son contrôleur associé** (ex: VuePartie → ContrôleurPartie).

1. **Le Contrôleur pilote la Vue** : Il décide quand la vue doit être rafraîchie (par exemple, après une navigation réussie) en manipulant l'état besoinAffichage de celle-ci.
2. **La Vue reste passive** : Elle se contente de restituer l'information du modèle, tandis que le contrôleur orchestre les changements d'états et les appels à la persistance.

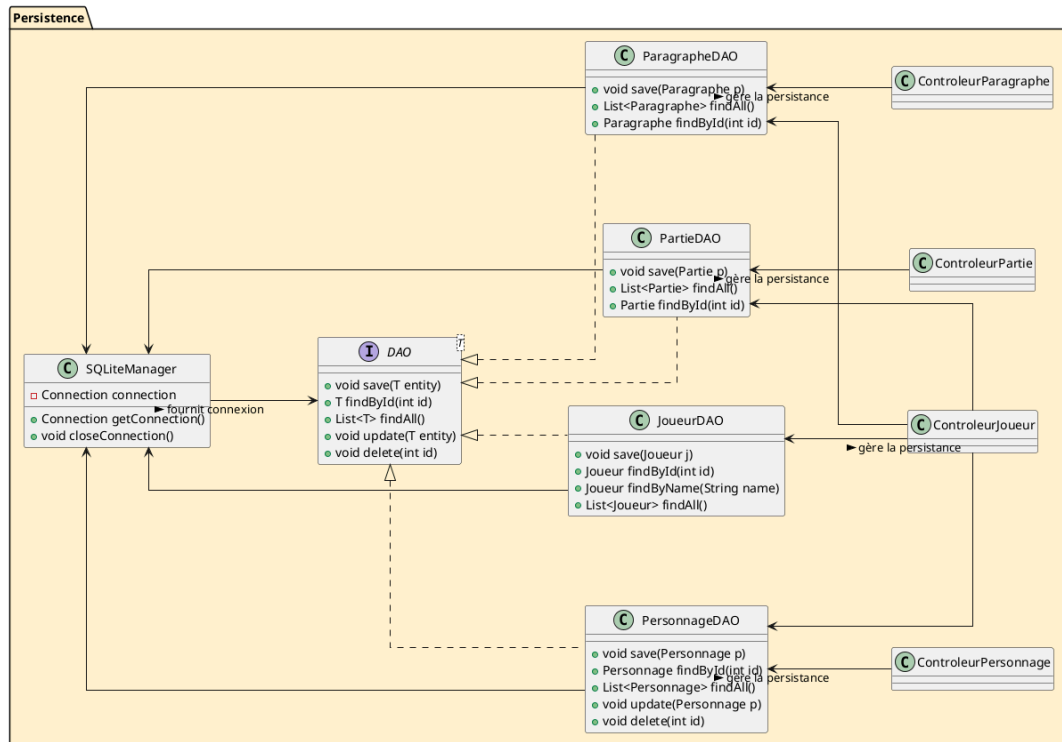


• Vue .-> Modèle

La relation entre les packages Model et View repose sur une implémentation rigoureuse du patron de conception Observateur, garantissant un couplage faible et une synchronisation automatique des données.



Point sur la persistance

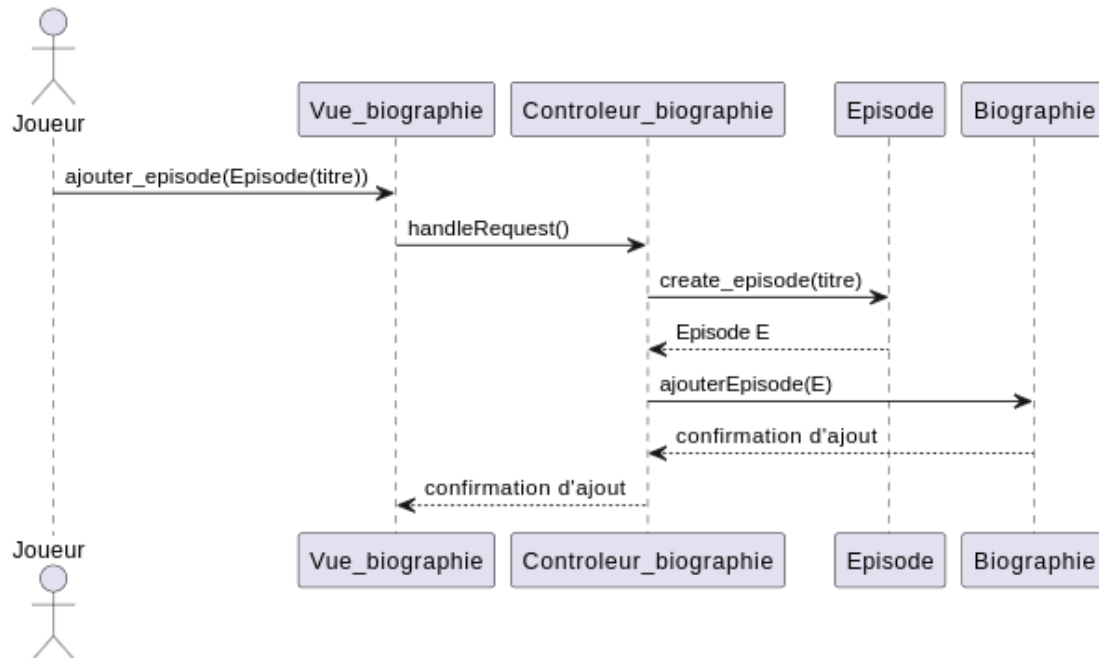


La couche de persistance de l'application est structurée selon le **Data Access Object (DAO) Pattern**, permettant une séparation stricte entre la logique métier et les requêtes SQL. Cette architecture facilite la maintenance et l'évolution du support de stockage (SQLite) sans impacter les contrôleurs ou les modèles.

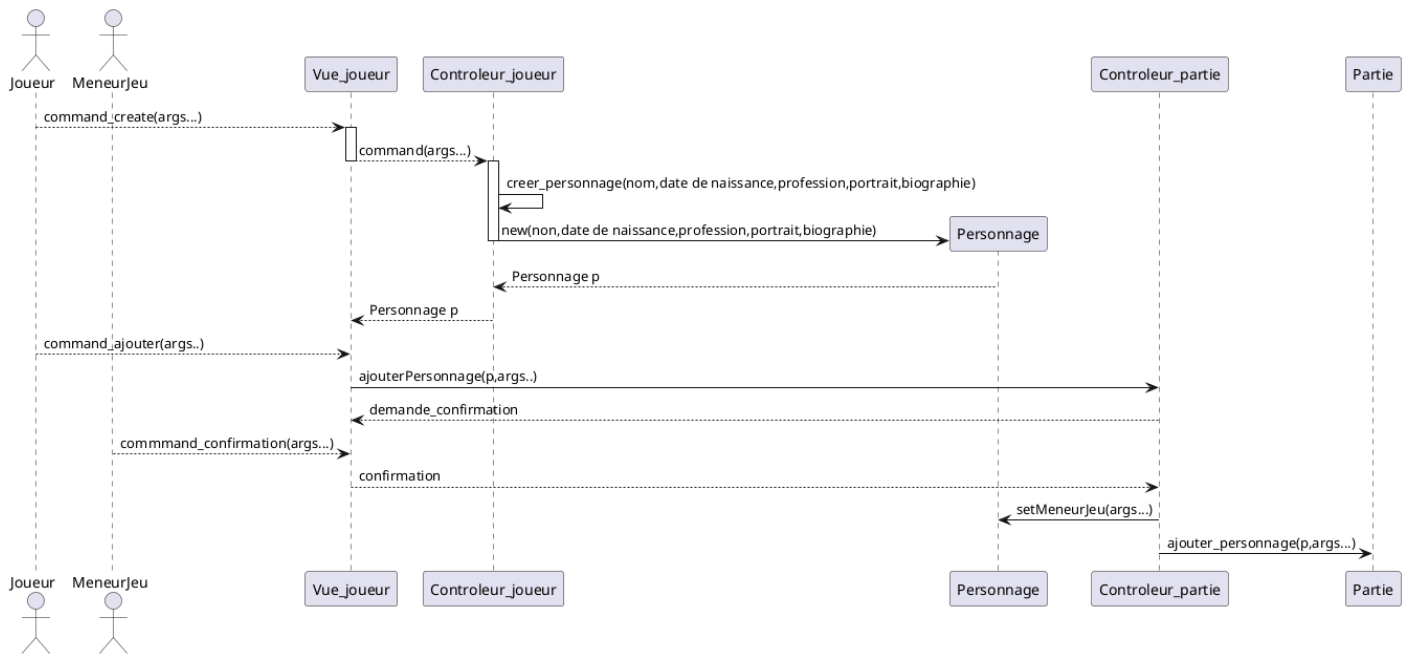
L'objet **SQLiteManager** fait office de point d'entrée unique pour la gestion de la base de données. Fournisseur de connexion, Il encapsule la logique d'ouverture et de fermeture des sessions JDBC, garantissant que chaque DAO utilise une instance de connexion valide et optimisée.

3. Diagramme de séquence technique

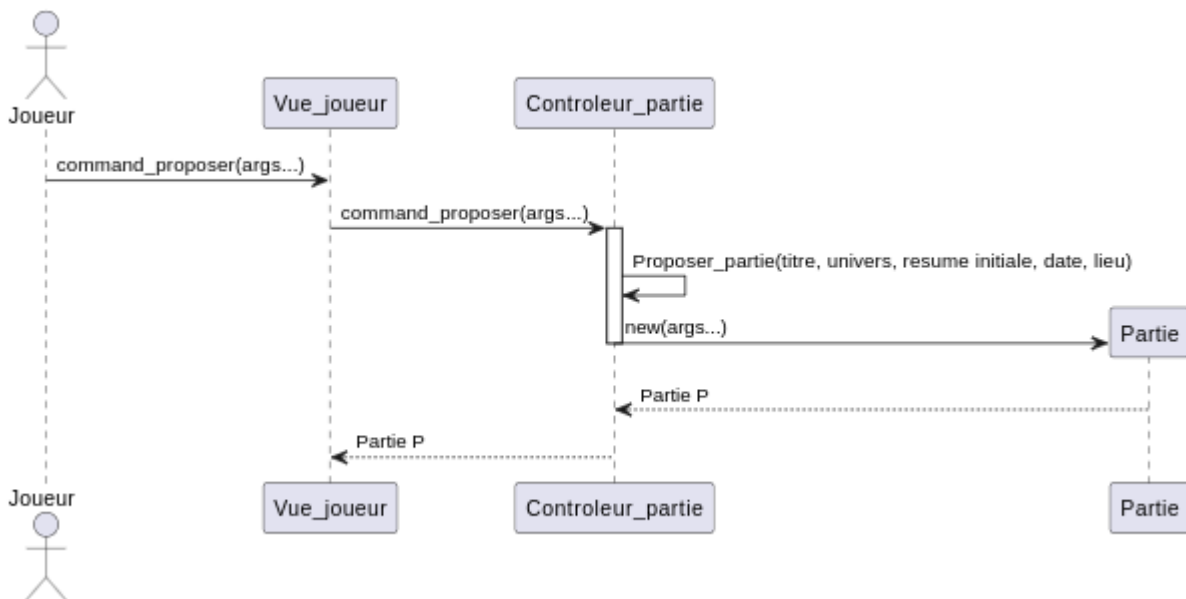
- Ajouter Episode



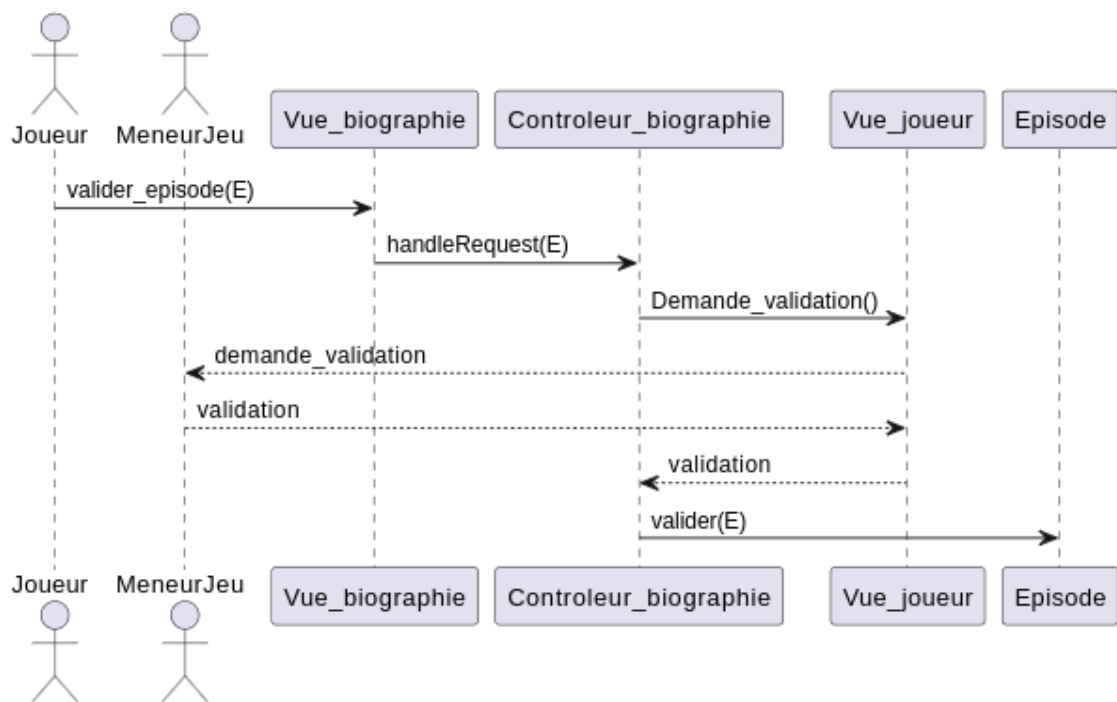
- Ajouter Personnage



- Proposer partie

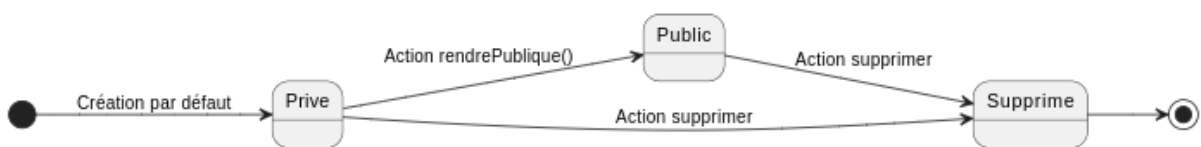


- Valider épisode

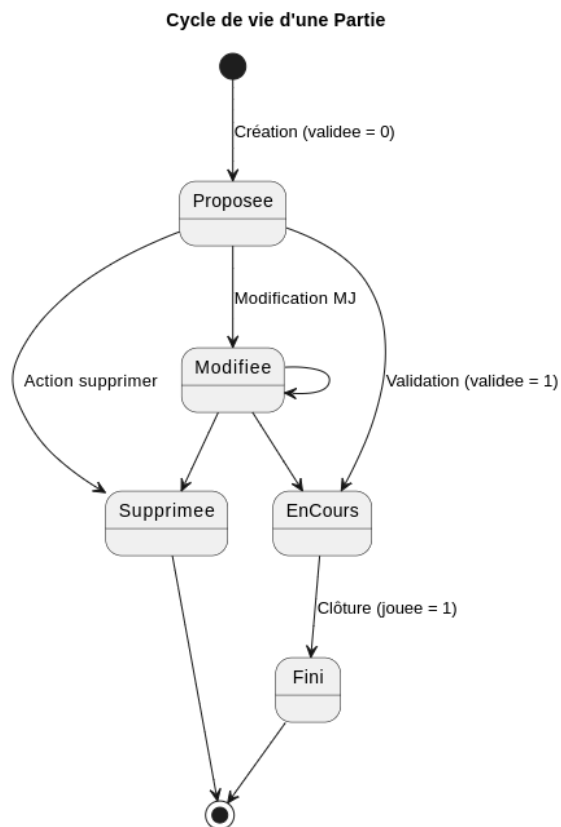


4. Diagramme d'état transition

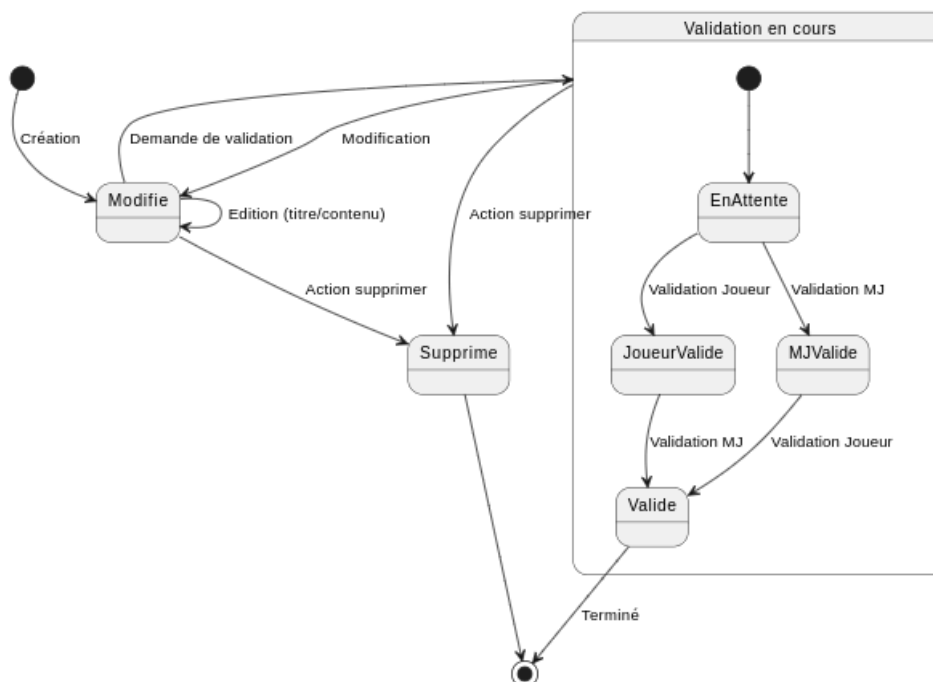
- Etat transition Episode



- État transition partie



- État transition épisode



MANUEL UTILISATEUR

1. Présentation

La Plateforme de Gestion de Parties de Jeu de Rôle est une application en ligne de commande permettant aux joueurs et maîtres du jeu de :

- Créer et gérer des personnages
- Organiser des parties de jeu de rôle
- Rédiger et consulter des biographies de personnages

2. Démarrage rapide

Lancer l'application grâce à la commande `mvn compile exec:java`

NB :

- Vous devez être à la racine du projet (au niveau du `pom.xml`)
- Plusieurs instances de l'application peuvent être lancées simultanément sur le même appareil pour les tests. Veillez cependant à commenter les lignes de code pour la création et le peuplement de la base de donnée de test (voir `README.md`)

Au démarrage, vous verrez cet écran d'accueil :

```
===== Bienvenue sur notre plateforme de roliste =====
-----
COMMANDES GLOBALES : Ces commandes fonctionnent tout le long du programme
- x : pour fermer l'application
- q : pour se deconnecter l'application -> revenir sur la page d'accueil
- h : pour revenir à la page du joueur
- r : pour revenir à la page précédente
-----
- Pour vous connecter taper 1
- Pour inscrire un nouveau joueur taper 2
```

Pour le test de l'application, les joueurs existants dans la base de donnée fournies sont :

- Alice
- Bob
- Charlie
- David
- Eve

- Frank
- Grace
- Heidi
- Ivan
- Judy

La navigation le long de l'application est dirigée. Veuillez donc à respecter les consignes données.

Une gestion des erreurs (à une certaine mesure) est cependant faite dans l'application.

```
-----
- Pour vous connecter taper 1
- Pour inscrire un nouveau joueur taper 2

> 3
Commande inconnue. Veuillez choisir une commande valide
>
> █
```

Lors de toutes les opérations sensibles (modification, suppression), une confirmation vous sera demandée :

```
- Pour supprimer un épisode
  xe,numéro de l'épisode

> xe,101
Êtes vous sûr de vouloir supprimer cet épisode ?
- o : pour confirmer
- n : pour annuler
> n
```

3. Guide des fonctionnalités

Lorsque vous vous connecter à l'application, un ensemble de fonctionnalité vous sont proposés :

```

===== BIENVENUE Alice =====
>>>>>>>>> Liste des parties dans lequel vous avez un personnage
Partie | Partie id | Personnage | Personnage id
Station Delta | 2 | Héros 1 | 1
Le Secret du Roi | 3 | Héros 2 | 2
Braquage à l'italienne | 4 | Héros 3 | 3
Terres Brûlées | 5 | Héros 4 | 4
>>>>>>>>> Liste des parties dans lequel vous êtes meneur de jeu
Partie | Partie id
Le Grimoire Perdu | 1
>>>>>>>>> Liste des parties des propositions de partie
lkdfosjoeofneofnoenf | 34 | 2
ydeefief | 35 | 4
Entrez :
- Pour afficher une partie en particulier
  1
- Pour afficher un personnage en particulier
  2
- Pour creer une partie
  3
- Pour creer un personnage
  4
- Pour rejoindre une partie
  5,id de la partie,id du personnage

```

Vous avez accès aux parties dans lesquelles vous avez un personnage participant, vous êtes meneur de jeu et les autres propositions de partie.

A. Gestion des parties

A.1. Affichage

Selon que vous soyez meneur de jeu de la partie ou pas, différentes options vous sont présentées.

- Vous n'êtes pas meneur de jeu

```

> 1

Entrez l'id de la partie
>2
===== BIENVENUE sur la partie: Station Delta =====
Situation initiale de la partie:
null
>>>>>>> Participants de la partie
Personnage | Personnage id
Héros 1 | 1
Héros 22 | 22
Héros 26 | 26
Héros 30 | 30
Héros 34 | 34
Entrez :
- Pour accéder à un personnage
  go,id du personnage

```

- Vous êtes meneur de jeu

```

> 1

Entrez l'id de la partie
>1
===== BIENVENUE sur la partie: Le Grimoire Perdu =====
Situation initiale de la partie:
null
>>>>>>> Participants de la partie
Personnage | Personnage id
Héros 21 | 21
Héros 25 | 25
Héros 29 | 29
Héros 33 | 33
Héros 37 | 37
>>>>>>> Personnages en attente de validation
Personnage | Personnage id
Aucun personnage en attente de validation.
Entrez :
- Pour accéder à un personnage
  go,id du personnage
- Pour marquer la partie comme terminée
  t,resumé de la partie
- Pour accepter un personnage
  ac,id du personnage
- Pour supprimer un personnage
  xp,id du personnage

```

A.2. Creation

Dans l'interface du joueur, vous avez la possibilité de créer une partie en entrant "3". Un ensemble d'informations vous est demandé successivement. À la fin vous retourner à la vue du joueur et vous pouvez visualiser vous partie nouvelle créée.

```

> 3
Entrez le titre de votre aventure:
>
> test
Entrez le resume de la situation initiale:
>
> partie de test
Entrez l'id de l'Univers :
>
> 3
Entrez l'id de l'Univers :
>
partie creee
===== BIENVENUE Alice =====
>>>>>>>> Liste des parties dans lequel vous avez un personnage
Partie | Partie id | Personnage | Personnage id
Station Delta | 2 | Héros 1 | 1
Le Secret du Roi | 3 | Héros 2 | 2
Braquage à l'italienne | 4 | Héros 3 | 3
Terres Brûlées | 5 | Héros 4 | 4
>>>>>>>> Liste des parties dans lequel vous êtes meneur de jeu
Partie | Partie id
Le Grimoire Perdu | 1
>>>>>>>> Liste des parties des propositions de partie
lkdfosjoeofneofnoenf | 34 | 2
ydeefief | 35 | 4
test | 6 | 3
Entrez :
- Pour afficher une partie en particulier
  1
- Pour afficher un personnage en particulier

```

Votre nouvelle partie est bien évidemment sans participant.

```

===== BIENVENUE sur la partie: test =====
Situation initiale de la partie:
partie de test
>>>>>>>> Participants de la partie
Personnage | Personnage id
>>>>>>>> Personnages en attente de validation
Personnage | Personnage id
Entrez :
- Pour accéder à un personnage
  go,id du personnage
- Pour marquer la partie comme terminée
  t,resumé de la partie
- Pour accepter un personnage
  ac,id du personnage
- Pour supprimer un personnage
  xp,id du personnage

```

B. Gestion des personnages

B.1. Affichage

Vous avez accès à un personnage soit directement dans l'interface du joueur soit dans celle d'une partie. En fonction que vous soyez propriétaire du personnage, son meneur de jeu ou un joueur lambda pour le personnage, l'affichage est différent.

```
> 1

  Entrez l'id de la partie
  >2
===== BIENVENUE sur la partie: Station Delta =====
Situation initiale de la partie:
  null
>>>>>>> Participants de la partie
  Personnage | Personnage id
Héros 1 | 1
Héros 22 | 22
Héros 26 | 26
Héros 30 | 30
Héros 34 | 34
Entrez :
  - Pour accéder à un personnage
    go,id du personnage

> go,22
===== DETAILS DU PERSONNAGE Héros 22 =====

>>>>>>>>> MENEUR DE JEU : Bob

>>>>>>>>> PARTIE ASSOCIEE : Station Delta

Entrez (en respectant le format le cas échéant):

- Pour afficher la biographie du personnage
  bio_pub

- Pour accéder à la page de la partie relative à ce personnage
  go_partie

> █
```

```
===== DETAILS DU PERSONNAGE Héros 1 =====

>>>>>>>>>> MENEUR DE JEU : Bob

>>>>>>>>>> PARTIE ASSOCIEE : Station Delta

Entrez (en respectant le format le cas échéant):

- Pour transférer ce personnage à un autre joueur
  t,id du joueur à qui l'on veut transférer le personnage

- Pour changer de le meneur de jeu de ce joueur
  c,id du meneur de jeu

- Pour afficher la biographie publique du personnage
  bio_pub

- Pour affihcer la biographie intégrale (publique + privée) du personnage
  bio

- Pour accéder à la page de la partie relative à ce personnage
  go_partie

> |
```

B.2. Creation

Idem que la création d'une partie

```
> 4
Veillez entrer le nom du personnage:
>
> tom
Veillez entrer la profession du personnage:
>
> dev
Veillez entrer la date de naissance du personnage:
>
> 2025
Veillez entrer l'ID de l'univers du personnage:
>
> 1
Veillez entrer le chemin de l'image du personnage (optionnel):
>
>
Veillez entrer le contenu de la biographie initiale:
>
> nouveau personnage
Veillez entrer l'ID de la partie associée:
>
> 3
Personnage créé avec succès !
===== BIENVENUE Alice =====
```

C. Gestion de la biographie des personnages

C.1. Affichage

Chaque personnage a une biographie publique (visible par tous) et une biographie privée (visible par son meneur de jeu et son propriétaire)

[illegible]

Les épisodes non validé n'apparaissent pas dans la biographie publique du joueur; de même que les paragraphes privés

```
> bio
```

BIOGRAPHIE INTEGRALE

```
>>>>>>>>>>>>>>>> Episode 1 -- Status : Validé  
Titre -- Chapitre 1 : Les Origines de Héros 1  
  
**** PARAGRAPHE 882 -- Type : Publique  
Titre -- Section 1  
Détails du récit numéro 1  
  
**** PARAGRAPHE 883 -- Type : Privée  
Titre -- Section 2  
Détails du récit numéro 2  
  
**** PARAGRAPHE 884 -- Type : Publique  
Titre -- Section 3  
Détails du récit numéro 3  
  
>>>>>>>>>>>>>>>> Episode 101 -- Status : En cours d'écriture  
Titre -- Chapitre 2 : L'Aventure continue pour Héros 1  
  
**** PARAGRAPHE 1002 -- Type : Publique  
Titre -- Section 1  
Détails du récit numéro 1  
  
**** PARAGRAPHE 1003 -- Type : Privée  
Titre -- Section 2  
Détails du récit numéro 2  
  
**** PARAGRAPHE 1004 -- Type : Publique  
Titre -- Section 3  
Détails du récit numéro 3
```

C.2. Modification

Plusieurs modifications sont possible sur la biographie des personnages uniquement par son propriétaire et son meneur de jeu

```
Entrez (en respectant le format):
```

- Pour ajouter un paragraphe à un épisode existant
a, numéro de l'épisode, titre du paragraphe, contenu du paragraphe, privée? (1 pour oui 0 sinon)
ex : a,2,la rentree,le premier jour de classe,0
- Pour ajouter creer un épisode (avec son premier paragraphe)
a,0,titre de l'episode, titre du paragraphe, contenu du paragraphe, privée? (1 pour oui 0 sinon)
ex : a,0,le lycée,la rentree,le premier jour de classe,1
- Pour rendre un paragraphe public
p, numéro du paragraphe
ex : p,10
- Pour modifier épisode :
 - * Le titre d'un épisode
me, t, numéro de l'épisode, nouveau titre de l'épisode
 - * Le titre d'un paragraphe
mp, t, numéro du paragraphe, nouveau titre du paragraphe
 - * Le contenu d'un paragraphe
mp, c, numéro du paragraphe, nouveau contenu du paragraphe
- Pour valider un épisode
ve, numéro de l'épisode
- Pour supprimer paragraphe
xp, numéro du paragraphe
- Pour supprimer un épisode
xe,numéro de l'épisode

```
> a,0,le lycée2,la nouvelle rentrée,le nouveau prof,0
Ce personnage n'a aucune aventure à laquelle vous pouvez lié ce nouvel épisode
Aventure crée avec succès
```

```
>>>>>>>>>>> Episode 142 -- Status : En cours d'écriture  
Titre -- le lycée2  
  
**** PARAGRAPHE 1363 -- Type : Publique  
Titre -- la nouvelle rentrée  
le nouveau prof
```

- Validation des personnages voulant participer à une partie dont ils sont

BILAN DES TECHNOLOGIES DE MODÉLISATION UTILISÉES

Pour la phase de modélisation du projet, la technologie utilisée est **PlantUML**, qui permet de décrire des diagrammes à l'aide d'un langage textuel. Les diagrammes ont été réalisés à l'aide de **l'éditeur PlantUML en ligne** ainsi que de **l'extension PlantUML sur Visual Studio Code**, ce qui a facilité l'intégration de la modélisation au processus de développement. Néanmoins, plusieurs difficultés ont été rencontrées, notamment des problèmes de compilation occasionnels avec l'extension VS Code, nécessitant le recours à l'éditeur en ligne pour vérifier et corriger les diagrammes que l'on arrivait pas à faire compiler. De plus, l'extension ne prenait pas en charge l'ensemble des extensions de fichiers utilisées, ce qui a parfois limité son usage, en effet les fichiers d'extension **.plantuml** posait dans certains cas des soucis à la compilation, ce qui nous a obligé à passer souvent par l'extension **.wsd**. Enfin, la prise en main du langage **PlantUML** a représenté un défi initial, en raison de sa syntaxe spécifique et de la nécessité de respecter rigoureusement les règles du langage pour obtenir des diagrammes valides.