Université Jean-Monnet

Télécom Saint-ÉTIENNE

Groupe 2
Filière : Informatique-Réseau

Document téchnique : Projet Informatique

Réalisé par :

BENZHA Marieme
EL ATMIOUI Nassim
FUSERO Clement
MARDOUME Boutaina
MEDIENE Yanis

Encadré par :

M. REMY Girodon
M. ZAGARRIO Romain
M. ZEGHDALLOU Mehdi

Année universitaire : 2024 - 2025

1 Déscription du projet

L'application est conçue pour la gestion et l'organisation des collections de bandes dessinées. Elle permet aux utilisateurs de découvrir des comics tout en offrant des fonctionnalités permettant de gérer efficacement leur propre bibliothèque de comics.
Les principales fonctionnalités de l'application incluent :

- Recherche de comics : Les utilisateurs peuvent rechercher des comics par titre et rechercher des personnages.
- Gestion de la bibliothèque personnelle : L'application permet d'ajouter, supprimer et trier les comics dans la bibliothèque personnelle de l'utilisateur.
- Suivi de l'état de lecture : Les utilisateurs peuvent suivre l'état de lecture de leurs comics, en les marquant comme "non commencé", "en cours" ou "terminé".
- Marquage des achats : Les utilisateurs peuvent indiquer si un comic de leur bibliothèque a été acheté ou non.
- Recommandations personnalisées : Basées sur les comics présents dans la bibliothèque de l'utilisateur et ses préférences, l'application propose trois types de recommandations :
    1. personnalisé : Recommandations fondées sur les comics que l'utilisateur a complétés.
    2. Basées sur la bibliothèque de l'utilisateur : Suggestions de comics similaires ou complémentaires à ceux déjà présents dans la bibliothèque.
    3. Recommandations générales : Présentation des nouveautés et des comics populaires, indépendamment de la bibliothèque de l'utilisateur.

L'application offre également deux types d'utilisateurs :

1. Utilisateur inscrit : Un utilisateur ayant créé un compte en s'inscrivant et se connectant . Cet utilisateur peut ajouter des comics à sa bibliothèque, suivre son état de lecture, marquer ses achats et bénéficier des recommandations personnalisées basées sur sa bibliothèque et ses préférences.

2. Utilisateur invité : Un utilisateur qui n'a pas créé de compte. Cet utilisateur peut rechercher et découvrir des comics, mais ne peut pas ajouter de comics à sa bibliothèque personnelle ni bénéficier de recommandations personnalisées basées sur sa bibliothèque.

L'objectif principal de notre projet est d'offrir une expérience fluide et intuitive aux passionnés de comics, leur permettant non seulement de gérer efficacement leur collection, mais aussi de découvrir de nouveaux titres en fonction de leurs goûts et de leurs préférences

.

2 Technologies et Outils Utilisés

Voici un aperçu des principales technologies et outils utilisés dans ce projet :
Langages de Programmation

— Java : Le langage principal utilisé pour développer l'application. Java permet de créer une application multiplateforme robuste et sécurisée, idéale pour la gestion des collections de comics.

Interface Utilisateur

— Java Swing : Bibliothèque Java utilisée pour créer l'interface graphique de l'application. Java Swing permet de concevoir une interface utilisateur riche et interactive avec des composants comme des boutons, des champs de texte, des panneaux et d'autres éléments graphiques. Elle est idéale pour des applications de bureau et permet une personnalisation facile de l'apparence.
— FlatLaf : est une bibliothèque moderne de look-and-feel pour Swing, offrant un design plat, épuré et élégant. Elle améliore considérablement l'esthétique des interfaces Swing en leur donnant un style plus contemporain.
— AWT : est l'une des bibliothèques natives de Java pour la création d'interfaces graphiques. Elle fournit des outils de base pour développer des applications avec des composants graphiques simples et des interactions utilisateur.

Base de Données

- MySQL : Système de gestion de base de données relationnelle utilisé pour stocker les informations des utilisateurs, des comics, des séries, et d'autres données associées. MySQL est choisi pour sa fiabilité et sa capacité à exécuter des requêtes
- SQL efficaces sur de grandes quantités de données. SQL : Des requêtes SQL sont utilisées pour interagir avec la base de données MySQL.

Outils de Développement

- Eclipse, VSCode et IntelliJ IDEA : IDE (Environnement de Développement Intégré) utilisés pour écrire et organiser le code source de l'application. Ces IDE facilitent le développement Java avec des outils pour le débogage et la gestion de projets.
- Maven : Outil de gestion de projet utilisé pour automatiser les tâches de compilation, de gestion des dépendances et d'exécution de l'application. Maven garantit une structure cohérente et facilite l'intégration de bibliothèques tierces.

Sécurité

- bcrypt : Utilisé pour hacher les mots de passe des utilisateurs afin de garantir une sécurité optimale lors de leur stockage dans la base de données. Cette méthode empêche la récupération directe des mots de passe, renforçant ainsi la sécurité des informations personnelles des utilisateurs.

API Externe

- Comic Vine API : API externe utilisée pour récupérer des informations détaillées sur les comics, les personnages et les séries.

Outils de Test

- JUnit : Framework utilisé pour la réalisation de tests unitaires dans l'application. JUnit permet de vérifier que les composants du système fonctionnent correctement.
- JUnit Jupiter : Utilisé pour les tests avec la version la plus récente de JUnit (version 5), qui propose de nouvelles fonctionnalités et une meilleure flexibilité dans l'écriture des tests.

Dépendances et Gestion des Dépendances

— MySQL Connector Java : Connexion à la base de données MySQL depuis l'application Java.

— JSON (org.json) : Utilisé pour la gestion et la manipulation des données JSON, notamment pour les interactions avec l'API Comic Vine et la gestion des données en réponse à des requêtes API.

— FlatLaf : Librairie pour le look and feel moderne de l'interface graphique Java.

Contrôle de Version

— Git : Outil de gestion de version pour suivre les changements dans le code et faciliter la collaboration avec d'autres développeurs.

— GitLab : Plateforme utilisée pour héberger le code source, gérer les versions et suivre les problèmes liés au développement de l'application. GitLab permet une gestion collaborative du projet et le suivi des modifications.

3 Architecture du Projet

L'architecture du projet est basée sur une approche modulaire et orientée objet, permettant de séparer clairement les différentes responsabilités au sein de l'application. Le projet suit le modèle MVC (Model-View-Controller) et utilise une architecture de type DAO (Data Access Object) pour l'interaction avec la base de données. Voici une vue d'ensemble des composants principaux du projet.

3.1 Schéma de l'Architecture

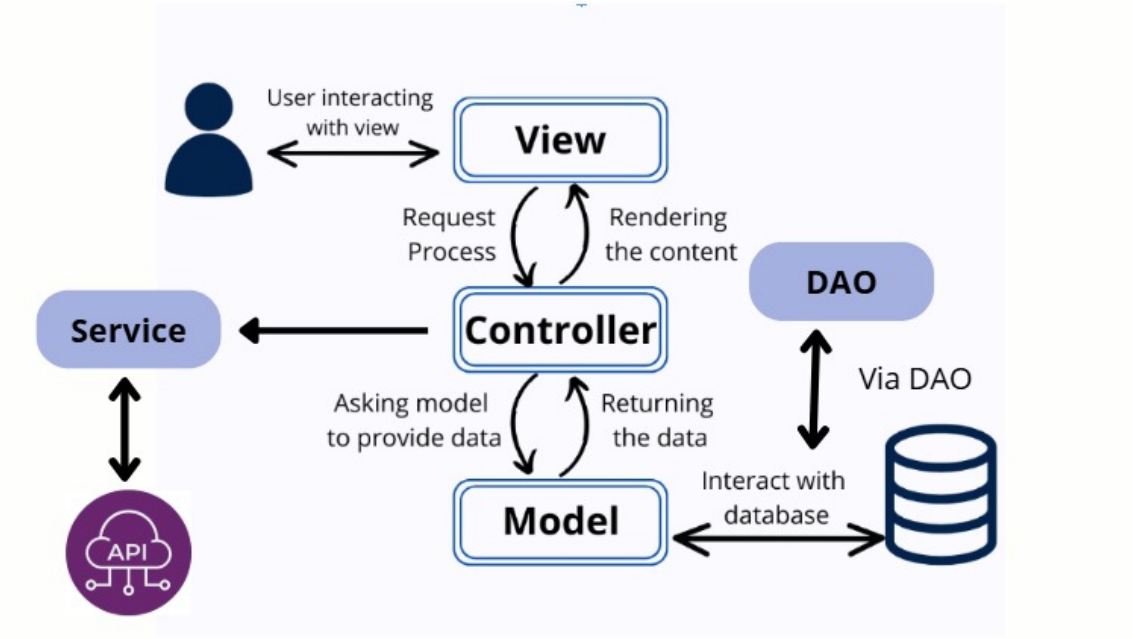

FIGURE 2 – Architecture du projet

3.2 Structure de l'Application

1. src/main/java/tse/info2

   — controller : Contient les classes responsables de la gestion des interactions avec l'utilisateur. Les contrôleurs sont chargés de recevoir les requêtes, de les traiter et de retourner les réponses appropriées.

   — model : Définit les entités de l'application. Ces classes représentent les objets métiers tels que Personnage, Comics, User, Issue, etc., qui sont utilisés dans toute l'application.

   — database : Contient les classes DAO (Data Access Object) qui gèrent l'accès aux données de la base de données. Chaque classe DAO est responsable de la gestion des entités spécifiques dans la base de données, par exemple, PersonnageDAO, ComicObjectDAO, UserDAO, etc. Ces classes effectuent les opérations CRUD (création, lecture, mise à jour, suppression) sur les tables de la base de données.

   — service : Contient les classes de services, responsables de la logique métier. Ces services traitent les données en provenance des DAO et les pré-

parent pour l'affichage ou les autres opérations nécessaires. Exemples de services : LibraryService.java, AuthenticationService.java, RecommendationService.java, FollowUpService.java. session : Gère la gestion de la session utilisateur .

- util : Contient des classes utilitaires pour la configuration de l'API ou d'autres fonctionnalités de l'application, comme ApiClient.java et ApiConfig.java.
- view : Contient les classes responsables de l'affichage de l'interface utilisateur, incluant les composants graphiques Java Swing.

2. src/main/resources

   - application.properties : Fichier de configuration pour les paramètres d'application comme la connexion à la base de données et les configurations d'API.
   - Logos et images : contenant des ressources graphiques utilisées dans l'interface utilisateur.

3. src/test/java/tse/info2

   - Contient les tests unitaires qui vérifient le bon fonctionnement des différentes fonctionnalités de l'application. Ces tests sont réalisés pour chaque classe métier et service, garantissant que les fonctionnalités sont correctes avant le déploiement.

3.3 Architecture DAO (Data Access Object)

Le projet utilise le modèle DAO pour séparer la logique d'accès aux données de la logique métier. Chaque entité (comme Personnage, Comics, etc.) dispose d'un

DAO dédié qui gère toutes les interactions avec la base de données pour cette entité.
Exemple de DAO : La classe PersonnageDAO gère toutes les opérations liées à la table Personnage dans la base de données. Cela inclut la recherche d'un personnage, son insertion, sa mise à jour et sa suppression.
Fonctionnement des DAO

- Méthodes CRUD : Chaque DAO contient des méthodes pour effectuer des opérations CRUD sur les données.
  - findOrSave : Méthode pour vérifier si un élément existe déjà dans la base de données, sinon, l'insérer.
  - update : Méthode pour mettre à jour un élément dans la base de données.
  - delete : Méthode pour supprimer un élément de la base de données.
- retrieve : Méthode pour récupérer des objets de la base de données

3.4 Sécurité et Authentification

L'application gère les utilisateurs avec un système d'authentification basé sur des sessions. L'authentification est gérée dans le package session où les informations d'identification de l'utilisateur sont vérifiées avant d'accorder l'accès à certaines fonctionnalités.

4 Architecture de l'Interface Utilisateur

L'interface graphique de l'application est développée en utilisant Java Swing, une bibliothèque Java dédiée à la création d'interfaces utilisateur riches et interactives.
Structure des Composants Swing
L'application utilise principalement une seule fenêtre qui se met à jour dynamiquement en fonction des actions de l'utilisateur. Plutôt que d'ouvrir plusieurs fenêtres, l'application affiche différentes sections dans une même fenêtre, chaque section étant chargée ou modifiée selon les actions de l'utilisateur.
Voici un aperçu des principales classes de l'interface graphique dans le package

tse.info2.view :

1. MainFrame.java : La fenêtre principale de l'application. Elle est lancée par Main.java et gère l'affichage dynamique de tous les autres panneaux dans le contentPanel. Elle permet de charger et afficher les différents panneaux de manière fluide sans ouvrir de nouvelles fenêtres.
2. ComicHomePage.java : Page d'accueil où l'utilisateur peut découvrir les recommendations. IssueDetailsPanel.java : Panneau qui s'affiche lorsqu'un utilisateur sélectionne un comic spécifique pour en voir les détails. Ce panneau affiche les informations sur le comic sélectionné.
3. LibraryPage.java : La bibliothèque personnelle de l'utilisateur où il peut consulter, trier , filtrer ou supprimer des comics de la bibliothèque.
4. LoginForm.java : Formulaire d'identification de l'utilisateur, permettant à l'utilisateur de se connecter à l'application.
5. PersonnageDetailsPanel.java : Affiche les détails d'un personnage spécifique, si l'utilisateur effectue une recherche sur un personnage.
6. SearchPage.java : Page permettant à l'utilisateur de rechercher des comics ou des personnages. Elle affiche les résultats de recherche et permet de sélectionner un comic ou un personnage.
7. SignupForm.java : Formulaire d'inscription permettant à l'utilisateur de créer un compte pour accéder à l'application.

Exemple de scenario de l'application :

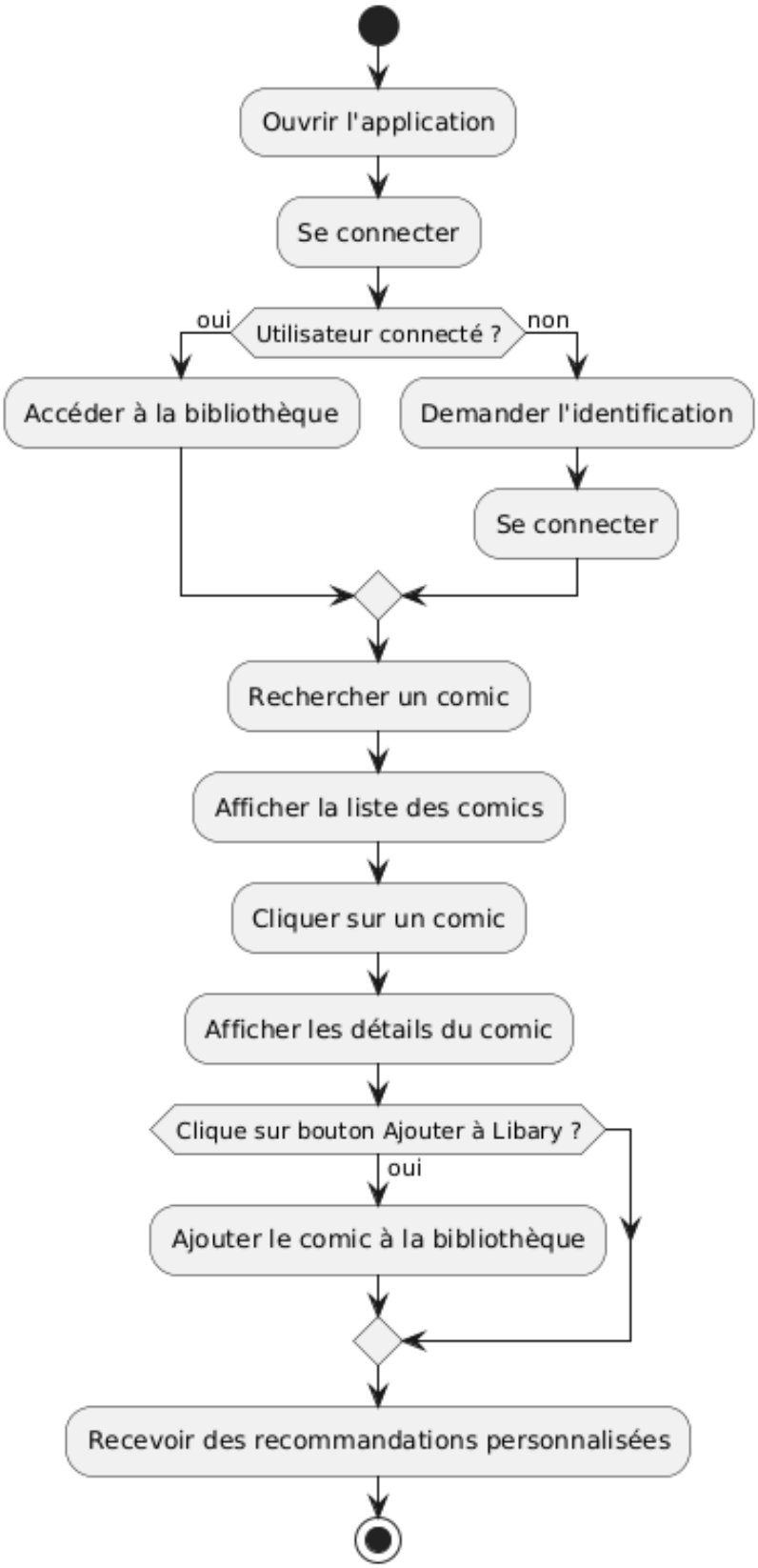

FIGURE 3 – Diagrame d'activité

5 Base de Données

La base de données comicsdb est utilisée pour stocker toutes les informations relatives aux utilisateurs, aux comics, aux personnages, aux auteurs et autres entités associées. Cette base de données est gérée par MySQL, un système de gestion de base de données relationnelle (SGBDR) réputé pour sa fiabilité, sa rapidité et sa capacité à gérer de grandes quantités de données de manière efficace. La structure relationnelle de la base de données permet de maintenir des relations cohérentes entre les différentes entités du projet.

5.1 Structure de la Base de Données

La base de données comicsdb est composée de plusieurs tables qui stockent les informations relatives aux utilisateurs, aux comics et à leurs éléments associés.

1. users : Contient les informations des utilisateurs (idUser, username, password).
2. Volume : Informations sur les volumes de comics (idVolume, titre, description, apidetailurl).
3. Issue : Détails des numéros de comics, avec référence au volume (idIssue, idVolume, numero, titre, datePublication, imageCouverture).
4. UserIssue : Lien entre les utilisateurs et les issues, incluant les favoris, l'avancement et l'achat (idUser, idIssue, Favoris, Avancement, Acheter).
5. Personnage : Détails des personnages (idPersonnage, nom, image, apidetailurl).
6. IssuePersonnage : Lien entre les issues et les personnages (idIssue, idPersonnage).
7. Power : Informations sur les pouvoirs des personnages (idPower, nom, apidetailurl).
8. PersoPower : Lien entre les personnages et leurs pouvoirs (idPersonnage, idPower).
9. Location : Lieux associés aux comics (idLocation, name, image, apidetailurl).

10. IssueLocation : Lien entre les issues et les lieux (idIssue, idLocation).
11. Genre : Genres des comics (idGenre, name, image, apidetailurl).
12. IssueGenre : Lien entre les issues et leurs genres (idIssue, idGenre).
13. ComicObject : Objets associés aux comics (idComicObject, name, image, apidetailurl).
14. IssueComicObject : Lien entre les issues et les objets (idIssue, idComicObject).
15. Auteur : Informations sur les auteurs des comics (idAuteur, nom, role, image, apidetailurl).
16. IssueAuteur : Lien entre les issues et les auteurs (idIssue, idAuteur).
17. Team : Équipes présentes dans les comics (idTeam, name, image, apidetailurl).
18. IssueTeam : Lien entre les issues et les équipes (idIssue, idTeam).
19. StoryArc : Informations sur les arcs narratifs (idStoryArc, image, apidetailurl).
20. IssueStoryArc : Lien entre les issues et les arcs narratifs (idIssue, idStoryArc).

Chaque table est liée par des clés étrangères permettant de maintenir des relations cohérentes entre les différentes entités, ce qui facilite la gestion des données dans l'application.

5.2 Justification du Choix de la Base de Données

Les informations sur les issues de la bibliothèque (tables Volume, Personnage, Auteurs, Genre, etc.) sont stockées dans la base de données pour plusieurs raisons :

— Réduction des appels API : En stockant les données localement, nous limitons les appels à l'API externe, qui sont limités.

— Meilleure réactivité : La récupération des informations des issues dans la bibliothèque depuis la base de données est plus rapide que de faire des appels API à chaque consultation, offrant ainsi une expérience utilisateur plus fluide.

— Recommandations personnalisées : Les données locales permettent de créer des recommandations basées sur la bibliothèque de l'utilisateur.

— Contrôle des données : Stocker les informations dans la base de données permet une gestion plus flexible et un meilleur contrôle des données utilisées dans l'application.

6 Modèle Conceptuel de Données (MCD)

Le MCD représente les entités et leurs relations dans la base de données. Les entités principales sont :

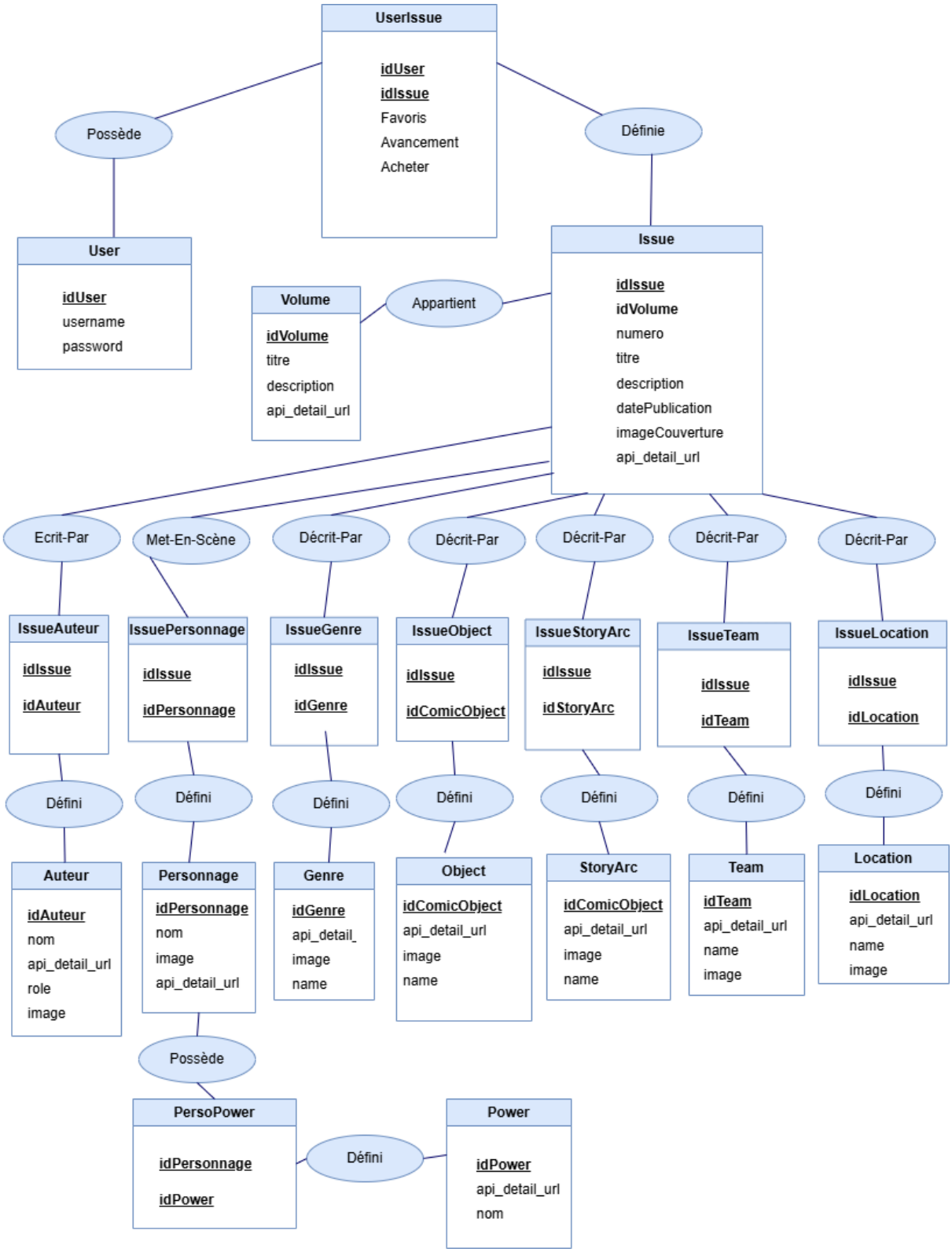

FIGURE 4 – Model Conceptuel des Données

7 Algorithme de Recommandation

L'algorithme de recommandation repose sur trois types principaux de recommandations, qui sont :

- Recommandations Générales Basées sur la Bibliothèque de l'Utilisateur
- Recommandations de Suivi pour les Séries Terminées
- Recommandations Basées sur les Nouveautés et les savec un Grand Nombre d'Issues

1. 1. Recommandations Générales Basées sur la Bibliothèque de l'Utilisateur
   L'objectif de cette recommandation est de suggérer des comics que l'utilisateur pourrait aimer, basées sur les volumes qu'il possède déjà dans sa bibliothèque.
   - Étape 1 : Récupération des issues de l'utilisateur L'algorithme commence par récupérer toutes les issues que l'utilisateur possède, via la table UserIssue.
   - Étape 2 : Identification des volumes À partir des issues, l'algorithme identifie les volumes correspondants pour comprendre les séries que l'utilisateur a déjà commencées.
   - Étape 3 : Recherche de nouvelles issues Il recommande des issues provenant des volumes que l'utilisateur possède déjà, mais qu'il n'a pas encore dans sa bibliothèque, permettant ainsi de découvrir de nouveaux comics dans des séries qu'il a déjà entamées.
   - Étape 4 : Limitation du nombre de recommandations Le nombre total de recommandations est limité à un maximum (par exemple, 5 recommandations).
2. Recommandations de Suivi pour les Séries Terminées
   Ce type de recommandation est destiné à aider l'utilisateur à poursuivre les séries qu'il a déjà terminées, en lui suggérant les prochains numéros à lire.
   - Étape 1 : Identification des séries terminées L'algorithme commence par identifier les issues terminées par l'utilisateur (avec Avancement = 'Completed' dans la table UserIssue).

- Étape 2 : Regroupement par volume Les issues terminées sont regroupées par volume, afin de savoir quelles séries l'utilisateur a déjà lues.
- Étape 3 : Recherche du prochain numéro disponible Pour chaque volume, l'algorithme cherche le dernier numéro consécutif que l'utilisateur a lu et recommande le prochain numéro disponible.
- Étape 4 : Limitation du nombre de recommandations Le nombre de recommandations est limité, par exemple à 5, pour éviter une surcharge d'informations.

3. Recommandations Basées sur les Nouveautés et les Issues avec un Grand Nombre d'Issues Cette fonctionnalité propose des recommandations de comics en fonction de deux critères principaux :

   - Issues récentes : En utilisant l'API Comic Vine, la méthode getLatestIssues récupère les comics récemment publiés, triés par date de mise à jour et numéro d'issue. Nous appliquons un filtre récupérer aussi les issues ayant une revue de personnel (hasstaffreview :true), garantissant ainsi que les recommandations concernent des comics de qualité. Les résultats sont ensuite limités à 5 comics uniquement.
   - Issues populaires : Les comics avec un grand nombre d'exemplaires ou d'intérêt de la part des utilisateurs sont également prioritaires. Cela aide à recommander des comics qui suscitent beaucoup d'attention dans la communauté.

   En combinant ces critères, l'application propose des suggestions pertinentes et actuelles aux utilisateurs.

Conclusion

En conclusion, ce projet de gestion et d'organisation de collections de bandes dessinées offre une solution innovante et intuitive aux amateurs de comics. En s'appuyant sur une architecture bien structurée (modèle MVC et DAO) et des technologies robustes comme Java, MySQL, et Swing, l'application garantit une expérience utilisateur fluide et personnalisée. Les fonctionnalités avancées, telles que les recommandations adaptées aux préférences des utilisateurs et la gestion complète des bibliothèques personnelles, répondent efficacement aux besoins variés des passionnés de comics.

La mise en œuvre des algorithmes de recommandation et l'intégration de l'API Comic Vine ajoutent une valeur significative au projet, en enrichissant les contenus proposés et en optimisant la découverte de nouveaux titres. De plus, l'utilisation d'une base de données locale améliore les performances tout en réduisant la dépendance aux services externes, garantissant ainsi une meilleure réactivité de l'application.

Enfin, grâce à une interface utilisateur moderne et à une attention particulière portée à la sécurité (hachage des mots de passe, gestion des sessions), ce projet allie esthétique, fonctionnalité et fiabilité. Il constitue une base solide pour des évolutions futures, telles que l'ajout de fonctionnalités sociales ou la migration vers une application mobile, élargissant ainsi son impact dans le domaine de la gestion de collections numériques.