

**Département d’informatique et mathématiques**

**8TRD157 – Base de données avancées**

Travail pratique #3 : Conception du serveur d'application et du client

|  |  |
| --- | --- |
| **Cours** | 8TRD157 – Base de données avancées |
| **Étudiant 1** | Valentin BAILLEUL |
| **Étudiant 2** | Alexandre COX |

|  |  |
| --- | --- |
| **Rapport** | **(45 pts)** |
| **Manuel utilisateur** | **/ 5** |
| **Stratégie de test** | **/ 5** |
| **Patrons de conception** | **/ 5** |
| **Validation des contraintes et de l’accès** | **/ 5** |
| **Code de mappage et justification** | **/ 10** |
| **Question 1** | **/ 5** |
| **Question 2** | **/ 5** |
| **Autres items** | **/ 5** |
|  |  |
| **Code source** | **(30 pts)** |
| **Code Hibernate** | **/ 10** |
| **Code d’application Java** | **/ 20** |
| **Qualité du code** |  |
|  |  |
| **Fonctionnalité** | **(25 pts)** |
| **Correction interactive** | **/ 25** |
| **Total** | **/ 100** |

Avant toute chose, veuillez créer un utilisateur « main » ainsi qu’un mot de passe « main » dans la base de données en lui donnant tous les privilèges pour le bon déroulement du manuel d’utilisateur (vous pouvez nous contacter par mail pour plus e renseignement concernant celui-ci).

# Manuel utilisateur

[Décrivez brièvement, pour chaque cas d’utilisation à implémenter, une procédure permettant d’utiliser ou de tester votre application. Par exemple, pour la recherche de films : quels sont les paramètres à entrer, sous quelle forme doivent être ces paramètres, etc. ?]

Pour commencer, rendez-vous sur la page « ConnectionUser » sur votre navigateur Web. Ensuite, choisissez un client dans le fichier « data.sql » (voici un client pour vous simplifier la tâche, username : [jeanty.final@videotron.ca](mailto:jeanty.final@videotron.ca) et password : 8956230f). Ensuite, vous allez vous retrouver sur une page qui propose de rechercher ou louer un film, il vous suffit de cliquer sur le bouton que vous voulez :

* Si vous avez choisi la recherche : Vous vous trouverez sur une page permettant de rechercher un film. Vous avez le droit d’ajouter n’importe quel filtre pour pouvoir effectuer votre recherche. Pour ajouter un filtre, il vous suffit de remplir le champ souhaité puis de cliquer sur le bouton « soumettre » qui lui correspond (ici, ajoutez « Clavier » dans le filtre Acteur par exemple). Ensuite, une fois les filtres ajoutés, vous devrez cliquer sur le bouton de recherche final pour obtenir un résultat.
* Si vous avez choisi de louer : inscrivez le nom du film que vous souhaitez louer (ici Intouchable) puis cliquez sur le bouton « louer ». Une fois cela réalisé, la transaction a été faite et vous allez être redirigé vers la page d’accueil qui vous propose les deux opérations possibles.

# Stratégie de test

[Décrivez la stratégie de test employée pour valider la fonctionnalité et les règles d’affaires de votre application. Par exemple : un simulateur de transactions, tests unitaires, tests d’utilisabilité, etc.]

Dans un premier temps, nous avons créé différentes « Web Pages » pour tester notre application Web. En effet nous avons utilisé les Servlets Java pour l’interface de notre application.

En ce qui concerne la base de données, nous avons effectué régulièrement des vérifications directement dans Oracle pour s’assurer que la location s’organisait correctement.

Tous les tests effectués sur l’application ont été fait manuellement, vous trouverez plus de détails concernant ceux-ci dans la partie « 4 Validation des contraintes ».

# Patrons de conception employés

[Décrivez les patrons utilisez lors de la conception de votre application en précisant le rôle de ces patrons. Par exemple : patron de façade, courtier BD ou DAO, DTO, etc.]

Nous avons décidé, pour ce TP3, d’utiliser les courtier BD (vous pouvez les trouver dans le dossier « src/java/courtier/ »).

# Validation des contraintes

[Décrivez la stratégie employée **dans votre application** pour valider les règles d’affaires. Par exemple : validation au niveau des données (TRIGGER) ou validation au niveau de l’application. Soyez spécifique.]

Nous avons effectué différents tests sur l’application pour que les règles d’affaires soient valides. Tout d’abord, nous avons créé un client qui avait le forfait « Intermédiaire » directement dans un fichier sql. Nous avons ensuite décidé de louer plus de cinq films pour savoir comment l’application réagissait : une erreur est donc survenue sur la sixième location et la transaction ne sait pas faite (cela a donc été une réussite dès le premier test).

Ensuite, nous avons voulu tester le nombre de copie de film et lorsque nous essayons de loué un film qui n’avait plus de copie, il n’y avait aucune réaction du côté de l’interface Web mais la transaction ne sait pas faite du côté de la base de données (le trigger fonctionnait correctement cependant il fallait prévenir le client que le film n’était plus disponible).

Pour finir, nous avons décidé exceptionnellement pour ce TP qu’un employé pouvait également se connecter sur l’interface client (cependant, il doit utiliser son matricule pour le bon fonctionnement de l’application). Comme les trois cas d’utilisations ne précisaient pas ce que l’employé pouvait faire, nous avons donc décidé de donner les mêmes fonctionnalités que celles des clients.

# Validation de l’accès et sécurité du système

[Décrivez la stratégie employée pour valider l’accès au système et pour assurer la confidentialité des données. Par exemple : accès contrôlé par l’application ou par la BD, encryption des mots de passe, etc.]

# Modifications au schéma

[Décrivez brièvement les modifications apportées aux schémas conceptuel et relationnel faits au laboratoire précédent, et expliquez pourquoi ces changements ont été faits.]

Pour simplifier le système de location de film, nous avons décidé de rajouter la classe « FilmCopie », qui permet donc d’avoir un identifiant de copie avec un identifiant de film qui lui correspond.

Pour le reste du schéma, rien n’a changé.

# Difficultés rencontrées avec Hibernate

[Décrivez brièvement les difficultés rencontrées lors du mappage de votre modèle objet au schéma de la BD avec Hibernate. Par exemple, stratégie de spécialisation non-conforme, clés à plusieurs colonnes, etc.]

Nous avons réussi à mapper notre modèle automatiquement, la seule difficulté que nous avons rencontrée était la disparition de certains fichiers lors de la création du « POJOS ». En effet, lorsque nous avons essayé de créer celui-ci, il nous été impossible d’insérer le fichier « Hibernate Reverse Engineering Wizard » car il était introuvable. Après plusieurs heures de recherche, nous avons décidé de recréer le projet au complet et le problème avait disparu.

# Code de mappages Hibernate pour les films, acteurs et réalisateurs

[Donnez le code XML des fichiers Hibernate servant à mapper les classes modélisant les films, les acteurs, les réalisateurs ainsi que leurs relations.]

Le code de mappages Hibernate pour les films, acteurs et réalisateurs (ainsi que toute les autres classes) se trouve dans le dossier « src/java/MovieRental ».

# Planification des tâches

[Décrivez brièvement comment le travail a été divisé dans votre équipe. Estimez, pour chaque tâche de l’énoncé, le pourcentage du travail effectué par chacun des membres de votre équipe.]

# Question théorique 1

[Donnez deux avantages d’utiliser un framework ORM comme Hibernate et précisez dans quel(s) contexte(s) ceux-ci NE SONT PAS utiles.]

Voici les deux avantages d’utiliser un framework ORM :

1. Réduit la quantité de code qui doit être écrit et permet une homogénéité avec le reste du code pour les langages orientés objets.

2. Très bonne méthode de programmation pour la traduction des données entre les systèmes de bases de données relationnelles et des langages de programmation orientés objet.

Malheureusement, les frameworks ORM induisent une couche logicielle supplémentaire, ils sont donc inutiles lorsque nous voulons privilégier la performance du système.

# Question théorique 2

[À quoi sert le patron de façade dans une application à plusieurs couches ?]

Le patron de conception “façade” a pour but de cacher une conception complexes difficile à comprendre en proposant une interface beaucoup plus simplifiée.

Dans une application à plusieurs couches, un patron de façade fournit donc une interface très simple et cela pour n’importe quel nombre de couche ajoutées ce qui permet une simplification dans l’utilisation et la compréhension de l’application.