Département de génie logiciel et des T.I.

RAPPORT DE LABORATOIRE

Numéro du laboratoire	**02**
Étudiant(s) et code permanent	Alexandre Laroche LARA12078907 Carl-Henri Codio CODC11157808 Esdras Metayer METE21058607 DIALLO Amadou Oury DIAA26018801 Sana Bouhadida BOUS27518509
Numéro d'équipe	**06**
Cours	**GTI660**
Groupe	**01**
Chargés(es) de laboratoire	**Mirna Awad**
Date	12 mars 2019

TABLE DES MATIÈRES

TABLE DES MATIÈRES 2

INTRODUCTION 3

ANALYSE 4

- **Deux prototypes papiers** 5
  - **Prototype des informations client** 5
  - **Prototype de la fenêtre de recherche** 6

CONCEPTION 6

- **Requêtes en algèbre relationnelle** 9
  - **Recherche par film** 9
  - **Recherche par personne** 9

IMPLÉMENTATION et ALGORITHMES 10

- **Conception** 10
- **Description des classes java créées** 10
- **Algorithme principal** 13
- **Structure du fichier resultat.xml** 14

DISCUSSION 14

- **Les améliorations possibles** 15

Manuel de l'utilisateur - Location de film 16

- **Démarrage de l’application** 16
- **Utilisation de l’application** 16

CONCLUSION 17

RÉFÉRENCES 18

1. INTRODUCTION

Le présent laboratoire s'inscrit dans le cadre du cours base de données multimédia et la continuation du premier laboratoire qui consiste à développer un système de base de données de location de films. Le but de ce laboratoire consiste à développer un système de base de données respectant une architecture trois tiers. Ce système proposera à l'utilisateur d'interroger, de modifier et d'interagir avec une base de données multimédia. Plus précisément, il s'agit d'un système de gestion de téléchargement de films.

Pour une bonne conception, l'application doit respecter les exigences décrites dans l'énoncé. C'est à dire, l'utilisation adéquate de plusieurs patrons de conception, ainsi que les fonctionnalités suivantes fenêtre de connexion, barre de menu, barre d'état de connexion, une fenêtre d'aide pour ne citer que cela. Ainsi, lorsque le résultat de l'exécution d'une requête est volumineux, la navigation de l'utilisateur doit être limitée entre menu précédent et suivant. L'utilisateur doit avoir la possibilité de sauvegarder le résultat des recherches dans un fichier XML.

Les prochaines lignes présentent l'analyse de l'ensemble des exigences; la conception qui inclura le diagramme d'architecture applicative et le diagramme de classe avec deux requêtes sous forme d'algèbre relationnelle; une section d'implémentation décrivant les différentes classes faisant partie de l'application ainsi que les relations entre elles, l'algorithme principal en pseudo code et la structure du fichier resultat.xml; ainsi qu'une courte discussion et une conclusion.

2. ANALYSE

L'application multimédia qui sera proposée à l'issue de ce laboratoire doit permettre d'interagir et de communiquer avec la base de données conçue au premier laboratoire afin de répondre au mieux aux exigences des utilisateurs.

Ainsi, il s'agit de développer une application interactive pour la consultation de contenu multimédia. L'application doit proposer une interface conviviale qui permettra à l'utilisateur d'interroger, de modifier et d'interagir avec la base de données et d'effectuer un ensemble de requêtes.
Une première interface à développer doit permettre à l'utilisateur de se connecter à l'application en entrant ses informations d'identification (le login et le mot de passe).
Une deuxième interface doit permettre à un client de modifier et de mettre à jour son profil à savoir ses informations de connexion, ses informations personnelles, son adresse, ainsi que les informations reliées à sa carte de crédit.

Une interface doit permettre à un client d'effectuer une recherche de film selon plusieurs critères soit le titre, un intervalle d'année, un intervalle de durée, la langue, la personne et le genre. La même interface donnera aussi la possibilité de rechercher un scénariste, un réalisateur ou un acteur. Ainsi, le client peut sélectionner un film, pour ensuite l'ajouter ou le retirer de son profil.

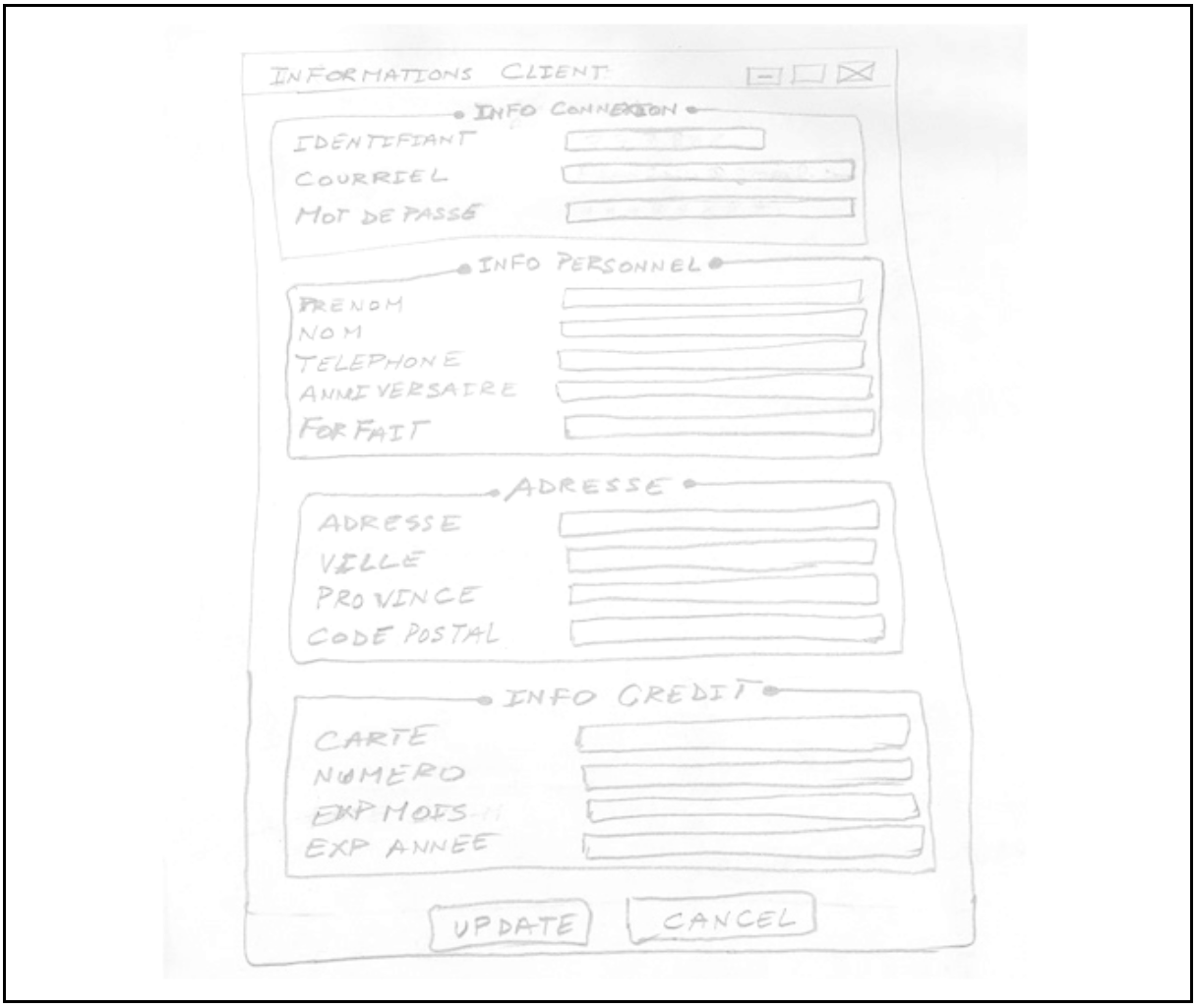
Une interface doit permettre au client de visualiser les détails sur une personne trouvée après une recherche, aussi de visualiser la photo de cette même personne.

D'autres éléments doivent aussi être développés comme une barre d'état avec un composant graphique pour illustrer l'état de la connexion avec la base de données (fermée, en cours, indéterminée).
Aussi, une barre de menu doit être intégrée à l'interface graphique afin de permettre à l'utilisateur de naviguer facilement entre les différentes fonctionnalités ainsi qu'une fenêtre d'aide comportant les raccourcis des différentes fonctionnalités de l'application.

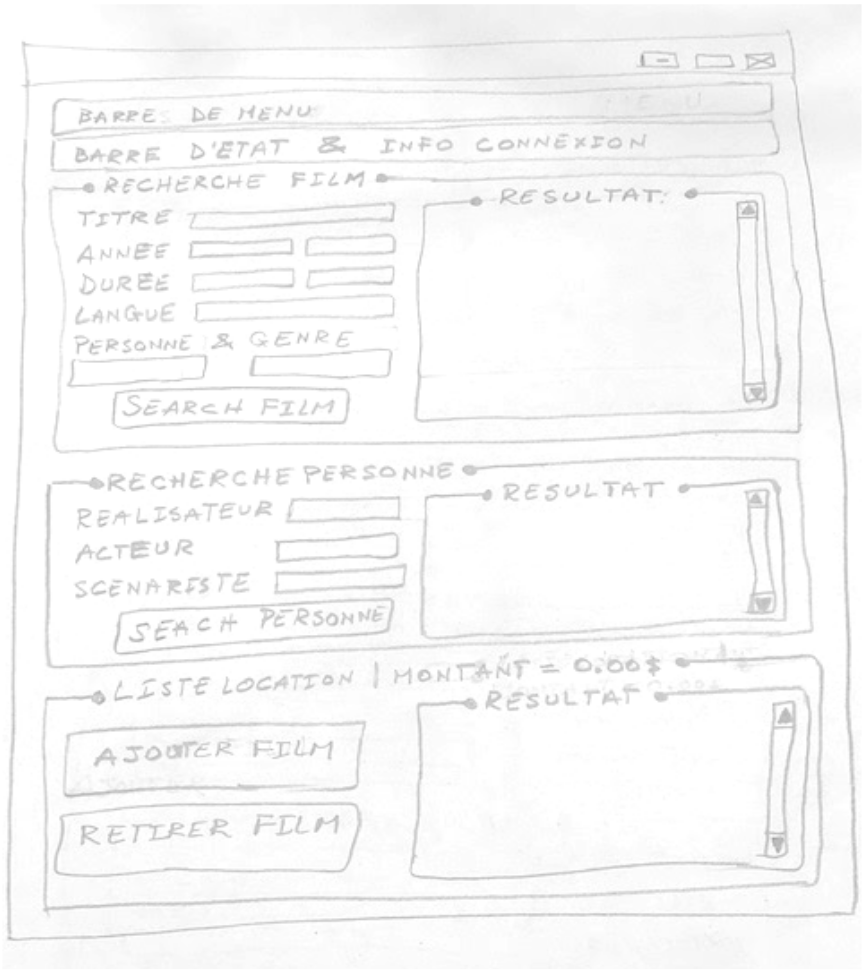
Deux prototypes papiers

Les prototypes sur papier sont très utiles pour l'architecture du contenu, la conception d'interface utilisateur et la simulation des besoins utilisateurs ou des tests. Ainsi, deux prototypes papiers ont été rapidement gribouillés pour les écrans des informations client et les recherches afin de faciliter la navigation des utilisateurs de l'application. Ces prototypes sont sujets à modification lors de l'implémentation afin de répondre le plus possible aux exigences.

Prototype des informations client



Prototype de la fenêtre de recherche



3. CONCEPTION

Solution aux exigences du laboratoire

L'architecture ci-dessous présente à haut niveau les différents packages nécessaires afin de transformer l'application du premier laboratoire qui était du deux tiers vers une structure trois tiers. Un ***Package Client*** regroupe toutes les classes d'affichage ainsi que celles qui sont utilisées pour la définition des objets d'affichage et l'exécution de l'interface utilisateur. Un ***Package Serveur*** englobe toutes les classes qui font le traitement les demandes utilisateurs, comme une connexion (login), la recherche d'un film ou d'une personne. Finalement, une ***base de données Oracle*** pour la persistance de toutes les données utiles à la location de films. Le diagramme de classe sera utilisé pour décrire toutes les classes utilisées lors de l'implémentation de l'application.

Architecture trois tiers de l'application

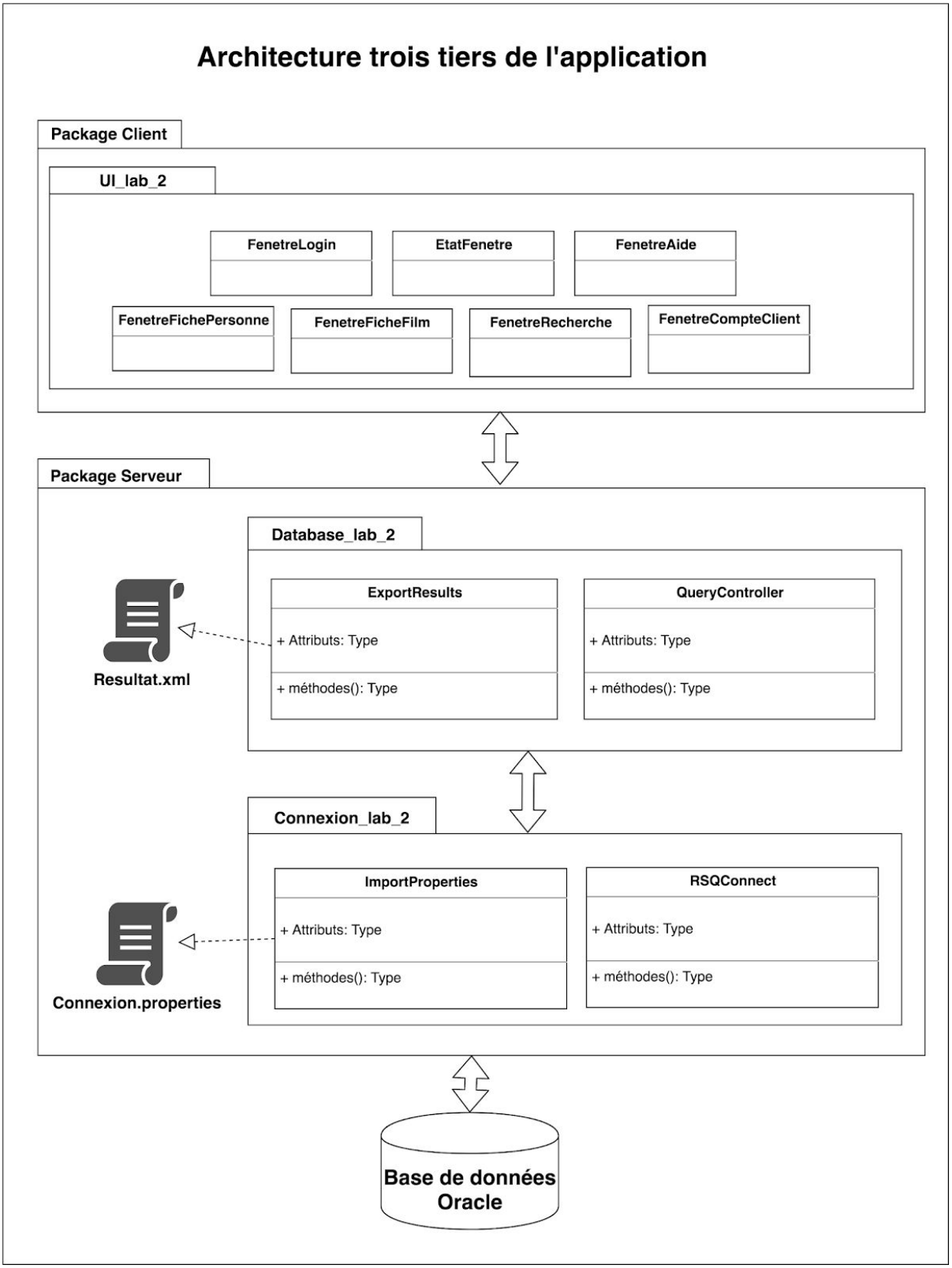
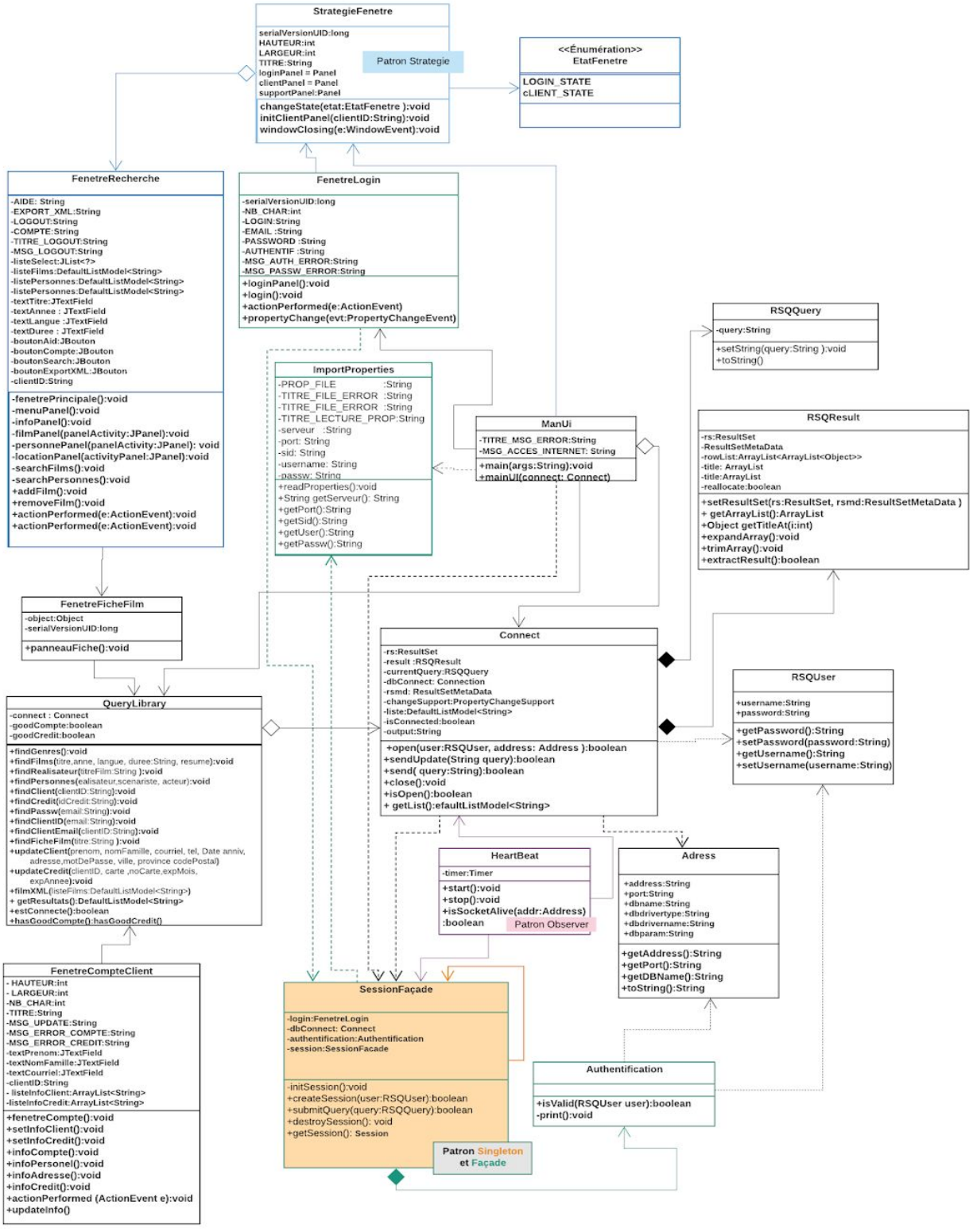


Diagramme de classe

Le diagramme de classe présenté ci-dessous est une représentation de toutes les classes de l'application. Une copie du fichier (DigrammeDeClasseUml.pdf) est ajoutée en annexe pour la remise.

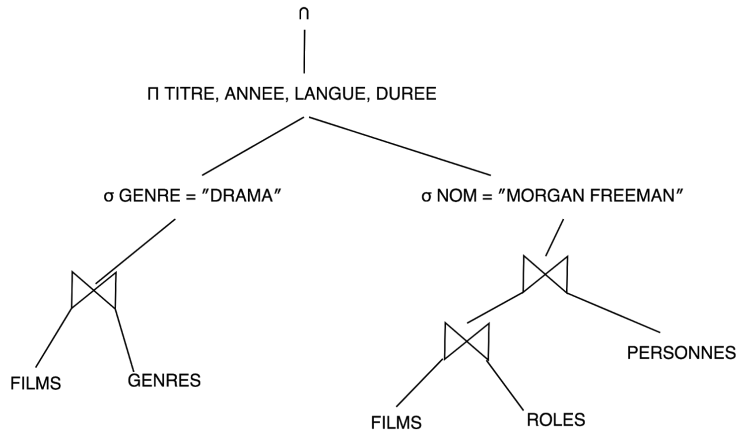


Requêtes en algèbre relationnelle

Recherche par film

La présente requête permet la recherche des films ayant le genre drama et dont l'un des rôles est joué par 'Morgan Freeman':

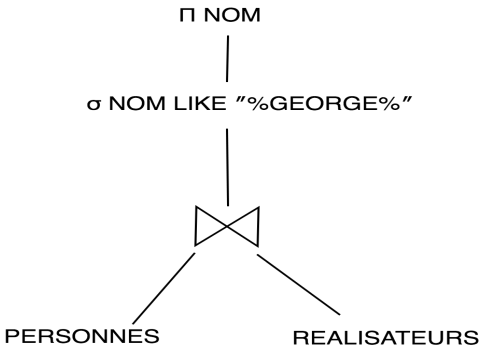
$$\Pi_{\text{TITRE, ANNEE, LANGUE, DUREE}} (\sigma_{\text{GENRE = "Drama"}} (\text{FILMS} \bowtie \text{GENRES})) \cap \Pi_{\text{TITRE, ANNEE, LANGUE, DUREE}} (\sigma_{\text{NOM = "MORGAN FREEMAN"}} (\text{FILMS} \bowtie \text{ROLES} \bowtie \text{PERSONNES}))$$



Recherche par personne

La présente requête permet la recherche des réalisateurs dont le nom contient le mot 'George':

$$\Pi_{\text{NOM}} (\sigma_{\text{NOM LIKE "\%GEORGE\%"}} (\text{PERSONNES} \bowtie \text{REALISATEURS}))$$



4. IMPLÉMENTATION et ALGORITHMES

Conception

Vue que la technologie utilisée dans la réalisation de l'application est java pour le front-end ainsi que le back-end. Il est tout à fait naturel d'utiliser les patrons ***Façade*** pour cacher toute la complexité du back-end tout en fournissant les fonctionnalitées nécessaires à l'interface. De plus, le patron ***Singleton*** est utilisé afin de créer un objet unique pour toutes les connexions à la base de données. Le patron ***Stratégie*** est aussi utilisé afin de créer un objet au niveau de l'interface pouvant effectuer plusieurs traitements différents.

Description des classes java créées

Classe	**QueryLibrary**
Description	Cette classe contient l'ensemble des requêtes SQL permettant d'interroger la base de données lorsqu'un utilisateur lance une recherche ou souhaite afficher les informations d'un client ou d'une personne ou sur un film en particulier..

Classe	**RSQResult**
Description	Cette classe définit un objet ***result*** permettant de regrouper le résultat d'une recherche sur la base de données.

Classe	**ExportResults**
Description	Cette classe est utilisé pour bâtir le fichier XML d'exportation via la méthode ***buildFileXML()***

Classe	**SessionFacade**
Description	Cette classe implémente le pattern Singleton ***SessionFacade session = new SessionFacade()*** afin de rendre unique chacune des sessions fait par un utilisateur et la façade afin de cacher toute la complexité de back-end à l'interface utilisateur

Classe	**FenetreLogin**
Description	Cette classe affiche la fenêtre de connexion aux utilisateurs ainsi que l'état de connexion de l'application. Le but est d'informer l'utilisateur en temps réel si une connexion est possible ou non.

Classe	**FenetreCompteClient**
Description	Cette classe utilise le constructeur *FenetreCompteClient(String _clientID)* afin de construire la page d'affichage des informations pour le client actuellement connecté.

Classe	**StrategieFenetre**
Description	Cette classe crée la fenêtre de base de l'application. Elle implémente le pattern stratégie pour changer l'état de la fenêtre à afficher. Elle gère l'état de la fenêtre Login et la fenêtre principale de recherche Films et Personnes.

Classe	**EtatFenetre (Énumération)**
Description	Cette classe contient les variables d'état de connexion et des recherches LOGIN_STATE, SEARCH_STATE

Classe	**FenetreFicheFilm**
Description	Cette classe permet l'affichage des informations sur un film en particulier à l'intérieur d'une session utilisateur en utilisant la méthode panneauFiche()

Classe	**MainUI**
Description	Cette classe représente le point d'entrée du programme, c'est toujours à partir de celle-ci que l'apps se démarre.

Classe	**Connect**
Description	Classe est un singleton. Elle crée l'objet connectDB qui rend unique le point d'interaction entre le back-end et la base de données.

Classe	**FenetreRecherche**
Description	Cette classe affiche l'écran principale de l'application "fenetrePrincipale()". Elle permet de faire la recherche sur les films et personnes jusqu'à la location.

Classe	**FenetreFichePersonne**
Description	Cette classe affiche les informations sur une personne à la fois FenetreFichePersonne(Object object)

Classe	**Address**
Description	Cette classe contient toutes les données nécessaires à la connexion vers le système de gestion de base de données ORACLE (port, nom de la BD, adresse du serveur, driver, type de BD)

Classe	**RSQUser**
Description	Cette classe définit le modèle du compte utilisateur d'un client

Classe	**HeartBeat**
Description	Cette classe crée un socket avec les mêmes informations de connexion de la base de données initiales et appel la méthode *isSocketAlive(Address address)* en boucle pour vérifier si la connexion est toujours ouverte.

Classe	**ImportProperties**
Description	Cette classe fait la lecture des informations de connexion vers la base de données se trouvant dans le fichier ***connexion.properties*** qui est à l'exterieur de l'application

Classe	**Authentification**
Description	Dans Cette classe nous avons défini une méthode qui renvoie une valeur booléenne indiquant que l'utilisateur est connecté ou non

Algorithme principal

Le pseudocode suivant décrit le mécanisme mis en place dans l'application lorsqu'un usager démarre une recherche. Afin de faciliter l'écriture du pseudocode la recherche d'un film par son titre est utilisé pour décrire l'algorithme.

Rechercher un film par son titre

Début classe **FenetreRecherche**
  Déclarer DefaultListeModel<String>: listefilms
  Déclarer DefaultListeModel<String>: historique
  Fonction ***searchFilms***
    Nettoyer ***listefilms***
    Capturer historique recherche films
    Appel : fonction ***findFilmsByTitre*** de la classe ***QueryController***
    Pour i allant de 0 à la longueur de la liste des résultats
      ajouter resultat dans ***listefilms***
  Fin fonction
Fin

Début classe ***QueryController***
  Fonction ***findFilmsByTitle*** (chaîne de caractères: titre)
    Déclarer chaîne de caractères : requête
    requête ← SELECT TITRE FROM FILMS WHERE TITRE LIKE '% titre %'
    Appel : fonction ***send*** de la classe ***RSQConnect(paramètre : requête)***
  Fin fonction
Fin

Debut classe ***RSQConnect***
  Déclarer objet ***RSQresult***: result
  Déclarer variable logique : isConnected
  Fonction logique ***send*** (chaîne de caractères : query)
    Si ***isConnected*** est vrai
      Déclarer objet ***PreparedStatement***: pstmt
      Executer la requete en utilisant le pstmt
      Appel : la fonction result.***setResultSet***
      Creer liste de recherche : liste
        Si result.***extractResult*** est vrai
          Pour i allant de 0 au nombre de ligne dans le résultat
            Pour i allant de 0 au nombre de colonne dans le résultat

```
                    Ajouter le résultat dans : liste
                Retourner vrai
            Sinon retourner faux
    Retourner faux
Fin
```

Structure du fichier resultat.xml

Le fichier resultat.xml est utilisé pour enregistrer l'exportation des recherches liées à un usager. Le fichier utilise l'encodage UTF-8 et tous les attributs utilisés dans ce fichier sont référencés dans un DTD (Définition de Type de Document) externe.

Exemple d'exportation de deux films

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<films>
    <film id="1">
        <!--Détail sur le film>
        … …. …. … … …
        … …. …. … … ...
    </film>
    <film id="2">
        <!--Détail sur le film>
        … …. …. … … …
        … …. …. … … ...
    </film>
</films>
```

5. DISCUSSION

Les points forts et les points faibles

Étant donné que dans les premiers travaux du premier laboratoire, l'équipe s'est concentrée sur une bonne analyse de la problématique et que les forces et faiblesses de chacun des membres de l'équipe étaient connues, les décisions sur la répartition des tâches étaient assez faciles à faire pour ce laboratoire.

Au niveau de l'application, la possibilité de faire des recherches simples ou complexes constitue le principal point fort de l'application. Des onglets sont disponibles pour des recherches normales et avancées. Un des points faibles de l'application se situe au niveau

de la conception ou pour chacune des requêtes une méthode a été développée. L'utilisation du patron décorateur aurait pu facilement corriger cette situation.

Les motivations pour la conception et le design

La structure de l'application respecte l'architecture MVC, les patrons de conception Façade, Stratégie, Observateur et Singleton ont été choisis. Le patron de conception façade permet de masquer la complexité d'utilisation en fournissant moins de fonctionnalités. Dans le travail de développement, la façade a été implémentée afin de présenter une interface simple à l'utilisateur. Et donc il présente une interface entre la classe session et le classe client lors de la connexion au système. Le patron Singleton fait l'interconnexion avec la BD.

Les choix de l'implémentation

Les mêmes outils du premier laboratoire ont été utilisés dans la modélisation du diagramme de classe, le programme est entièrement codé via la plateforme Java. Tel que décrit dans le fichier du lab2 : Les autres langages comme le PHP présente un inconvénient majeur : l'impossibilité de faire le prochain laboratoire dans ce langage; quant au Python (incluant Django), il a l'inconvénient de n'être pas le langage le plus rapide; avec le C# (Dotnet), les librairies de vision par ordinateurs et de traitement d'images ne sont pas faciles à intégrer.

Ajout(s) personnel(s)

Un Heartbeat est mis en place pour écouter la connexion en temps réel.
Les fiches Personne et film sont disponibles via l'application.
Les Patrons de conception : Façade, Stratégie, Observateur, Singleton ont pu être adoptés et appliqués dans le code.

Pour un champ 'ANNÉE', la saisie des lettres ou plus de quatre (4) chiffres ne sont pas permis. Des restrictions sur l'utilisation de tous les champs de texte sont appliquées.

L'application inclut plusieurs modes de recherche: normale et spécialisée. Elle présente aussi une interface utilisateur facile à utiliser afin de faciliter la recherche.

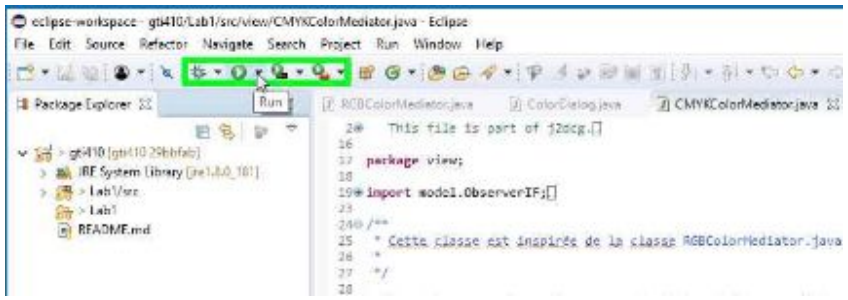
Les améliorations possibles

Il pourrait être utile d'ajouter des tests unitaires pour valider la fiabilité des résultats, un scrollbar dans la fenêtre pour faciliter la navigation de l'utilisateur et utiliser davantage d'interface ou d'héritage pour mieux optimiser le code.

6. Manuel de l'utilisateur - Location de film

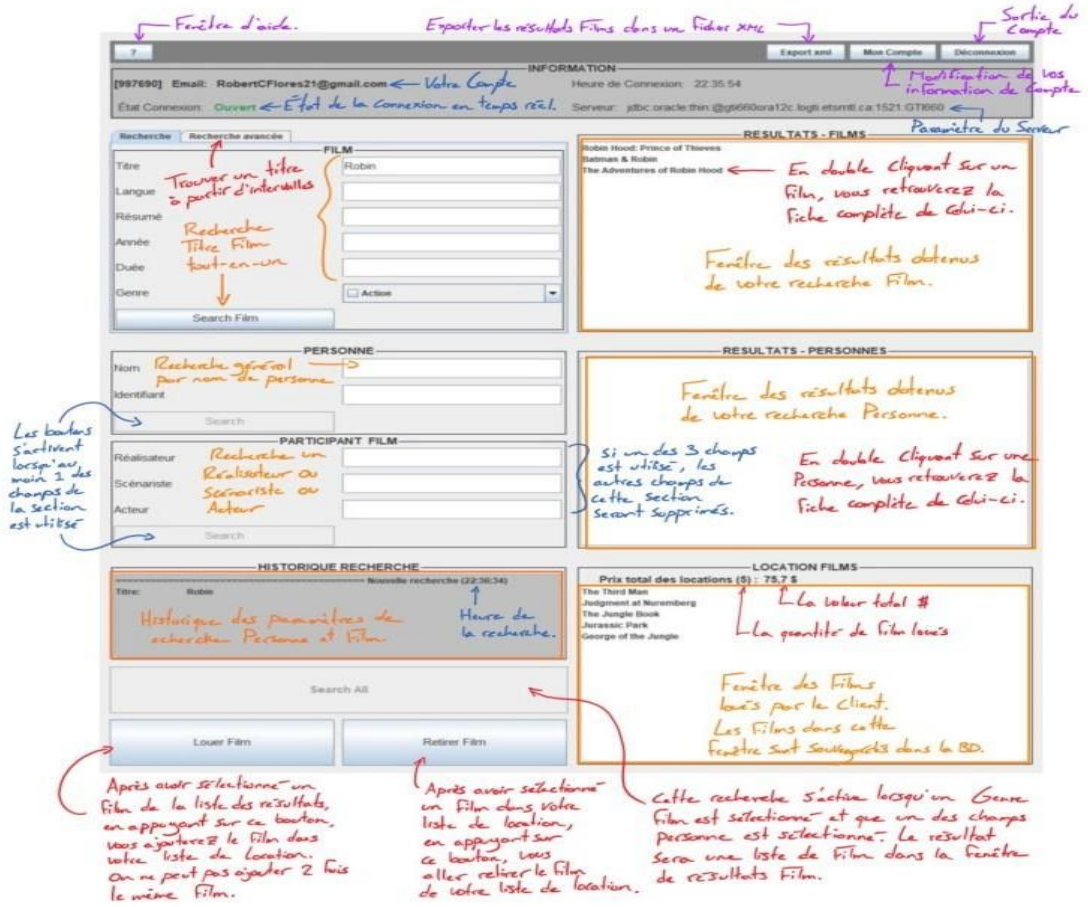
Démarrage de l'application

À partir du logiciel Eclipse

	**Dans la barre d'outils d'eclipse** cliquez gauche sur la flèche à côté du bouton **play** 

Utilisation de l'application

L'application étant démarré, il reste à faire les manipulations sur les fonctionnalités …



7. CONCLUSION

En résumé, l'objectif de ce laboratoire était de concevoir une application de gestion de base des données multimédia en respectant l'architecture trois tiers. Le point de départ était d'analyser les exigences pour déterminer les besoins spécifiques de l'application comme les différentes fonctionnalités de l'interface et l'utilisation des patrons de conception.
Par la suite, la conception a permis d'avoir une visibilité sur la composition de l'application à travers l'architecture, le diagramme de classes et d'apporter des solutions telles que le passage de l'architecture deux tiers vers le trois tiers.
Puis, l'implémentation consacre la réalisation de la conception, l'intégration du module de connexion et les interfaces les plus significatives de l'application.

En ce qui concerne l'amélioration, l'application respecte les exigences de qualité telles que la performance, la convivialité, la fiabilité et la maintenabilité par exemple le code est bien commentées et modulaires par conséquent il est facile à maintenir et extensible.

Lors de l'implémentation, plusieurs difficultés ont été surmontées, le premier choix de la technologie était spring boot, très vite un problème de conversion des données s'est posé et cette piste a été abandonnée pour revenir en java. En plus, les compétences des membres de l'équipe sont diverses, tout le monde n'est pas à l'aise avec la programmation, mais chacun a apporté sa contribution en fonction de ses compétences d'où la réussite du laboratoire.

8. RÉFÉRENCES

https://sgbd.developpez.com/tutoriels/cours-complet-bdd-sql/?page=algebre-relationnelle#LVI-E