

Programmation Web Serveur 2

Cahier des charges

Table des matières

Table des matières	2
1. Introduction	2
1. Contexte	3
2. Objectifs et présentation du produit	3
3. Guide de lecture du document	3
2. Spécifications	3
1. Spécifications fonctionnelles	3
a) Maquettes de l'interface	5
b) Enchaînement des vues	10
2. Spécifications techniques	11
a) Modèle de données	12
b) Modèle physique de données	13
3. Plan de développement	15
1. Classes et méthodes	15
3. Contraintes	21
1. Contraintes de délai	21
2. Contraintes matérielles	21
3. Contraintes fonctionnelles	21
4. Déroulement du projet	22
5. Références & Ressources	23
5.1 Outils	23
5.2 Ressources des spécifications techniques et fonctionnelles	23

1. Introduction

1. Contexte

Le projet consiste à développer une application graphique pour gérer une vidéothèque. Il s'agit donc d'un logiciel qui s'installe sur un poste informatique de la vidéothèque avec pour seul utilisateur le gérant de cette vidéothèque.

2. Objectifs et présentation du produit

L'application communique avec une base de données qui stocke toutes les informations relatives aux films loués, aux clients, etc. Il s'agit d'une application graphique qui dispose de trois vues globales qui permettent de visualiser les listes des films, des clients ainsi que des emprunts. Elle dispose également d'une vue globale qui permet de faire des opérations de filtre sur les films proposés par la vidéothèque.

3. Guide de lecture du document

Le code et les fichiers relatifs à la conception du logiciel sont déposés et mis à jour sur le github du projet :

<https://github.com/GentilBastien/Videotheque>

Le prototype du logiciel (créé sous figma) est disponible et exécutable à ce lien :

<https://www.figma.com/proto/VbHhCBmmqn3BtQ4LhKaMHj/Videotheque?node-id=448%3A2186&starting-point-node-id=448%3A2186>

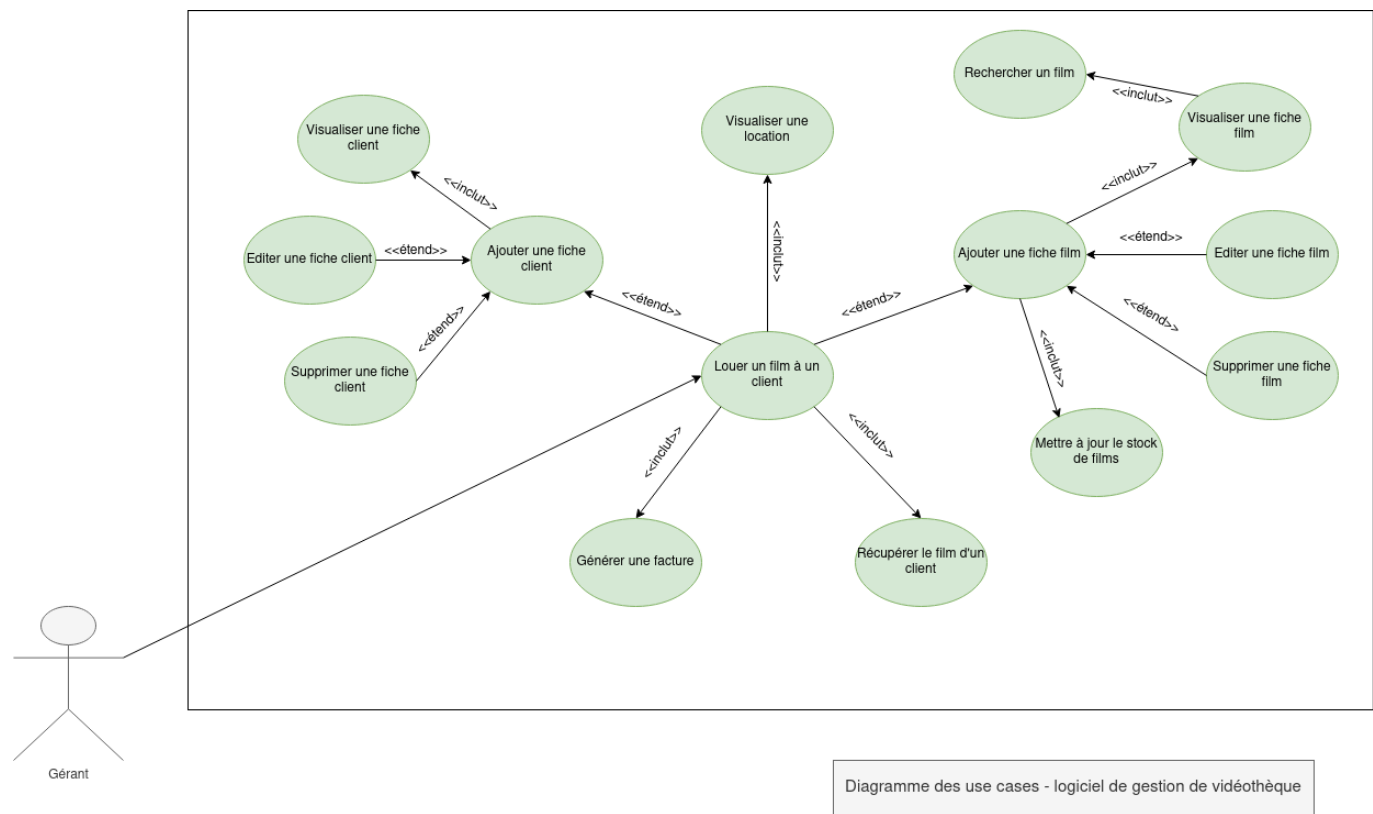
2. Spécifications

1. Spécifications fonctionnelles

Les spécifications qui apparaissent en gras sont les spécifications demandées explicitement dans le cahier des charges du client.

- **Les films doivent être recherchables par nom du film, réalisateur, année de sortie et par catégorie ;**
- **Les résultats de filtrage sur les films sont triés en fonction du nombre de fois qu'ils ont été loués ;**
- **Le stock de film doit pouvoir être mis à jour au fur et à mesure que le gérant de la vidéothèque reçoit des films**
- Afficher le nombre de copies de films actuellement en commande par la vidéothèque ;
- **Connaître les locations en cours ainsi que les dates de retour ;**
- Éditer l'état d'une location et mettre à jour le stock lorsque le gérant affirme qu'un client a retourné un exemplaire de film ;
- **Connaître les films loués par un client ;**
- **Connaître les clients qui ont loué un film ;**
- Les informations relatives à un client sont éditables ;
- Les informations relatives à un film sont éditables ;
- Les informations relatives à une location sont éditables ;
- **Faire la location de films aux clients ;**
- **Editer une facture à un client ;**
- Les films étant loués physiquement, le gérant peut scanner un exemplaire de film pour retrouver son code-barre et ajouter une nouvelle location à un client au lieu d'éditer manuellement toutes les informations relatives à l'ajout d'une nouvelle location.

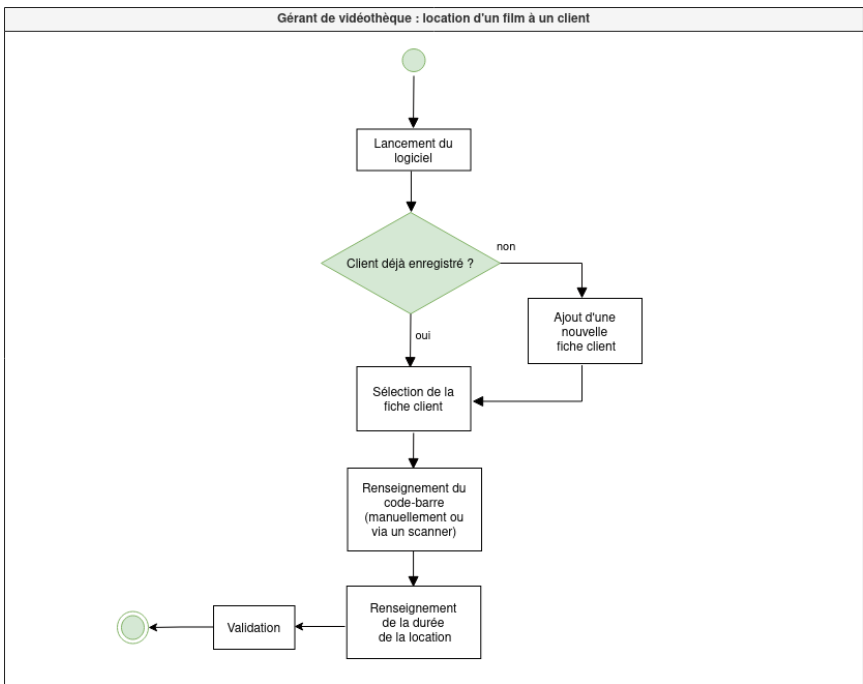
Ces spécifications fonctionnelles sont en outre représentées dans le diagramme de use-cases ci-dessous (disponible en annexe) :



Notre logiciel servant avant tout à gérer les locations, l'action consistant à louer un film à un client est ainsi au centre de toutes les actions possibles. En effet, afin de pouvoir louer un film à un client, il faut avoir ajouté d'une part une fiche film, et d'autre part une fiche client. De là découlent toutes les actions :

- Pour une location : sa visualisation, la génération de la facture associée ainsi que la récupération d'un film loué
- Pour une fiche client : sa visualisation, son édition et sa suppression
- Pour une fiche film : sa visualisation, son édition, sa suppression, sa visualisation, et par inclusion sa recherche

Le diagramme d'activité ci-dessous présente le processus permettant de louer un film à un client, également disponible en annexe :



Scénario : un client entre dans la vidéothèque et choisit un DVD. Il se rend en caisse, et le tend au gérant. À cette étape, le logiciel est déjà lancé. Si le client n'est pas inscrit auprès de la vidéothèque (dans le cas d'un nouveau client), le gérant doit créer sa fiche. Il sélectionne ainsi la fiche associée à ce client, puis renseigne le code-barre et la durée de la location, ce qui lui permet ainsi de valider l'opération.

En nous appuyant sur ces diagrammes, nous pouvons ainsi créer un prototype qui répond à toutes nos exigences. Ce que nous présentons dans les prochaines sections ci-dessous.

a) Maquettes de l'interface

Nous présentons dans cette section le prototype de notre interface. Pour rappel, le prototype dynamique est accessibles depuis ce lien :


<https://www.figma.com/proto/VbHhCBmmqn3BtQ4LhKaMHj/Videothèque?node-id=448%3A2186&starting-point-node-id=448%3A2186>

Le nom des vues correspond entre ce document et la présentation Figma.

Vue de prototype "Page principale"

Lorsque l'utilisateur lance l'application, il arrive sur cette vue. Elle permet notamment d'accéder facilement à la liste de tous les films proposés par la vidéothèque. Les films sont toujours triés du plus loué au moins loué. Un bloc permet de filtrer les films par nom de film, réalisateur, acteur(s), date de sortie, catégorie(s).

Vue de prototype "Film"




Clients

Films

Ajouter fiche

Locations



Description

Le jour de ses onze ans, Harry Potter, un orphelin élevé par un oncle et une tante qui le détestent, voit son existence bouleversée. Un géant vient le chercher pour l'emmener à Poudlard, une école de sorcellerie l'héber en balai, yler des sorts, combattre les trolls : Harry se révèle un sorcier doué. Mais un mystère entoure sa naissance et l'effroyable V., le magi dont personne n'ose prononcer le nom.

Acteurs principaux

Daniel Radcliffe, Emma Watson, Alan Rickman

Réalisateur(s)

Chris Columbus

Langues & sous-titres

Français, Anglais

Année sortie

2001

Durée

129min

Genre

Aventure Action Sci-Fi

Prix location

10€

État du stock

En stock 3

En prêt 1

En commande 0

Locations

Loué par	Date début location	Date fin location	État
Jean	15/05/2010	30/05/2010	En rayon
Marcel	12/08/2010	18/08/2010	En rayon
Michel	03/07/2010	27/07/2010	En rayon

Supprimer fiche

Editer la fiche

Gérer les copies

Code barre	Nom film	Durée	Genre	Prix location	État	En stock	En prêt	En commande
9790123456785	Le Seigneur des Anneaux : La communauté de...	210min	Aventure>Action;Sci-Fi	6€	En rayon	12	1	0
9880123856885	Harry Potter à l'école des sorciers	129min	Aventure>Action;Sci-Fi	10€	En rayon	3	1	0
9000123856005	Star Wars Episode 1 : La Menace Fantôme	122min	Aventure>Action;Sci-Fi	11€	Rupture stock	0	8	0
9844423856453	Harry Potter et le Magicien d'Oz	114min	Aventure>Action;Sci-Fi	4€	En rayon	20	1	0
9812345859985	L'aile ou la Cuisse	71min	Aventure>Action;Sci-Fi	8€	Rupture stock	0	12	0
9440123456441	Bac Nord	92min	Aventure>Action;Sci-Fi	10€	En rayon	23	0	0
9760177756885	Linea Verde	154min	Aventure>Action;Sci-Fi	10€	Rupture stock Commandés	0	0	6
9990987655684	Harry Potter à l'école	189min	Aventure>Action;Sci-Fi	11€	En rayon Commandés	30	0	4
9010145585666	Star Wars Episode 3 : La revanche des Sith	200min	Aventure>Action;Sci-Fi	19€	En rayon	14	3	0
9843573853475	Kaïssé se moque d'un caissier	3min	Aventure>Action;Sci-Fi	1€	En rayon	14	6	0
9009112385880	Avatar 2 : la voie de l'eau	569min	Aventure>Action;Sci-Fi	15€	En rayon	11	5	0
9333412835635	Taken 3	115min	Aventure>Action;Sci-Fi	16€	En rayon	7	1	0

La vue film se divise horizontalement en deux parties. La partie haute présente les caractéristiques du film actuellement sélectionné. La partie basse est la liste de tous les films dans la bibliothèque. Cliquer sur une ligne de ce tableau permet de mettre à jour la partie haute. L'état d'un film est soit "En rayon" ou "Rupture stock" en fonction du nombre de copies restantes en stock. Un état "Commandés" est rajouté à côté des films dont des copies ont été commandées par le gérant de la vidéothèque. La partie haute présente également une section "Locations" qui indique toutes les locations en cours pour ce film.

Vue de prototype "Supprimer un film"

Lorsque l'on clique sur "Supprimer fiche", une fenêtre modale de confirmation apparaît. Si l'utilisateur clique sur valider, le film est retiré de la base de données ainsi que toutes les locations relatives à ce film (en cascade).

Vue de prototype “Ajouter un film”

Si l'utilisateur clique sur “Ajouter fiche” dans le menu, un formulaire apparaît où tous les champs doivent être remplis. Cliquer sur Valider permet d'ajouter une nouvelle fiche avec ces caractéristiques dans la base de données et de la sélectionner dans la vue.

Vue de prototype “Éditer un film”

Clients

Films

Ajouter fiche

Locations

Harry Potter à l'école des sorciers

Description

Le jour de ses onze ans, Harry Potter, un orphelin élevé par un oncle et une tante qui le détestent, voit son existence bouleversée. Un géant vient le chercher pour l'emmener à Poudlard, une école de sorcellerie ! Voler en balai, jeter des sorts, combattre les trolls : Harry se révèle un sorcier doué. Mais un mystère entoure sa naissance et l'effroyable V, le mage dont personne n'ose prononcer le nom.

Acteurs principaux

Daniel Radcliffe, Emma Watson, Alan Rickman

Réalisateur(s)

Chris Columbus

Langues & sous-titres

Français, Anglais

Année sortie

2001

Durée

129min

Genre

Aventure Action Sci-Fi

Prix location

10€

Locations

Loué par	Date début location	Date fin location	Etat
Jean	15/05/2010	30/05/2010	En rayon
Marcel	12/08/2010	18/08/2010	En rayon
Michel	03/07/2010	27/07/2010	En rayon

Editer la fiche “Harry Potter à l'école des sorcier”

Description

Le jour de ses onze ans, Harry Potter, un orphelin élevé par un oncle et une tante qui le détestent, voit son existence bouleversée. Un géant vient le chercher pour l'emmener à Poudlard, une école de sorcellerie ! Voler en balai, jeter des sorts, combattre les trolls : Harry se révèle un sorcier doué. Mais un mystère entoure sa naissance et l'effroyable V, le mage dont personne n'ose prononcer le nom.

Acteurs principaux

Daniel Radcliffe, Emma Watson, Alan Rickman

Réalisateur(s)

Chris Columbus

Langues & sous-titres

Anglais, Français

Année sortie

2001

Durée

129

Genre

Aventure, Action, Sci-Fi

Prix location

10

Chemin photo

/img/hero/harry_potter_1.png

Supprimer fiche

Editer la fiche

Gérer

Code barre	Nom film	En stock	En prêt	En commande
9790123456785	Le Seigneur des Anneaux : Les deux tours	12	1	0
9880123856885	Harry Potter à l'école des sorciers	3	1	0
9000123856005	Star Wars Episode 1 : La Menace Fantôme	0	8	0
9844423856453	Harry Potter et le Magicien d'Oz	20	1	0
9812345859985	L'aile ou la Cuisse	0	12	0
9440123456441	Bac Nord	23	0	0
9760177756885	Linea Verde	0	0	6
9990987655684	Harry Potter à l'école	30	0	4
9010145585666	Star Wars Episode 3 : La revanche des Sith	14	3	0
9843573853475	Kaïssé se moque d'un caissier	14	6	0
9009112385880	Avatar 2 : la voie de l'eau	11	5	0
9333412835635	Taken 3	7	1	0

Lorsque l'utilisateur clique sur “Éditer la fiche”, un formulaire apparaît où tous les champs sont pré-remplis avec les caractéristiques actuelles de la fiche. Cliquer sur Valider permet de mettre à jour la fiche actuelle ainsi que ses informations dans la base de données.

Vue de prototype “Gérer les copies d'un film”

Clients

Films

Ajouter fiche

Locations

Harry Potter à l'école des sorciers

Description

Le jour de ses onze ans, Harry Potter, un orphelin élevé par un oncle et une tante qui le détestent, voit son existence bouleversée. Un géant vient le chercher pour l'emmener à Poudlard, une école de sorcellerie ! Voler en balai, jeter des sorts, combattre les trolls : Harry se révèle un sorcier doué. Mais un mystère entoure sa naissance et l'effroyable V, le mage dont personne n'ose prononcer le nom.

Acteurs principaux

Daniel Radcliffe, Emma Watson, Alan Rickman

Réalisateur(s)

Chris Columbus

Langues & sous-titres

Français, Anglais

Année sortie

2001

Durée

129min

Genre

Aventure Action Sci-Fi

Prix location

10€

Locations

Loué par	Date début location	Date fin location	Etat
Jean	15/05/2010	30/05/2010	En rayon
Marcel	12/08/2010	18/08/2010	En rayon
Michel	03/07/2010	27/07/2010	En rayon

Supprimer fiche

Editer la fiche

Gérer

Gérer les copies de “Harry Potter à l'école des sorcier”

Actuellement en stock

3

Actuellement en prêt

1

Actuellement en commande

0

Commande reçue

Valider

Annuler

Code barre	Nom film	En stock	En prêt	En commande
9790123456785	Le Seigneur des Anneaux : Les deux tours	12	1	0
9880123856885	Harry Potter à l'école des sorciers	3	1	0
9000123856005	Star Wars Episode 1 : La Menace Fantôme	0	8	0
9844423856453	Harry Potter et le Magicien d'Oz	20	1	0
9812345859985	L'aile ou la Cuisse	0	12	0
9440123456441	Bac Nord	23	0	0
9760177756885	Linea Verde	0	0	6
9990987655684	Harry Potter à l'école	30	0	4
9010145585666	Star Wars Episode 3 : La revanche des Sith	14	3	0
9843573853475	Kaïssé se moque d'un caissier	14	6	0
9009112385880	Avatar 2 : la voie de l'eau	11	5	0
9333412835635	Taken 3	7	1	0

Lorsque l'utilisateur clique sur "Gérer les copies", un formulaire apparaît où tous les champs sont pré-remplis avec les données actuelles du stock pour la fiche sélectionnée. Cliquer sur le bouton "Commande reçue" permet de transférer l'entier dans "Actuellement en commande" au stock. Par exemple, si la fiche actuelle a 5 copies actuellement en commande, cliquer sur "Commande reçue" va mettre l'attribut "en commande" à 0, et incrémenter l'attribut du stock de 5. Cliquer sur Valider permet de quitter en confirmant les changements, "Annuler" pour le cas inverse.

Vue de prototype "Client"

Clients

Ajouter clients

Films

Locations

Roissey Pierre

Numéro client
489528

Nom
Roissey

Prénom
Pierre

Téléphone
06 99 45 12 14

Mail
exemple.nom at gmail.com

Adresse
38000 Grenoble 3 rue Joseph

Supprimer client

Editer client

Gérer facture

Louer un film

Historique des locations

Film id	Nom film	Date début location	Date fin location	Etat
9799503366226	Batman	15/05/2010	30/05/2010	Reçu
9256456366709	Superman	12/08/2010	18/08/2010	Reçu
9545503348955	Iron-man	03/07/2010	27/07/2010	En location
9090912345656	Batman 2	15/09/2010	23/09/2010	En location En retard

Num. client	Nom	Prénom	Téléphone	Mail	Adresse	Locations
654542	Dupont	Dupont	06 78 98 52 12	exemple.nom at gmail.com	38000 Grenoble 3 rue Joseph	0
251651	Marcel	Marcel	06 89 52 12 45	exemple.nom at gmail.com	38000 Grenoble 3 rue Joseph	2
895412	Laurent	Laurent	07 20 54 78 55	exemple.nom at gmail.com	38000 Grenoble 3 rue Joseph	0
489528	Roissey	Pierre	06 99 45 12 14	exemple.nom at gmail.com	38000 Grenoble 3 rue Joseph	1
369741	Charles	Charles	06 52 21 14 45	exemple.nom at gmail.com	38000 Grenoble 3 rue Joseph	0
898952	Maxime	Maxime	09 35 32 45 41	exemple.nom at gmail.com	38000 Grenoble 3 rue Joseph	1
120200	Lucie	Lucie	09 89 51 02 21	exemple.nom at gmail.com	38000 Grenoble 3 rue Joseph	1
011987	Enzo	Enzo	00 58 88 95 66	exemple.nom at gmail.com	38000 Grenoble 3 rue Joseph	0
778203	Baptiste	Baptiste	06 01 11 87 10	exemple.nom at gmail.com	38000 Grenoble 3 rue Joseph	5
998996	André	André	06 98 78 74 00	exemple.nom at gmail.com	38000 Grenoble 3 rue Joseph	0
744544	Charlotte	Charlotte	06 98 45 48 47	exemple.nom at gmail.com	38000 Grenoble 3 rue Joseph	0
158745	Julie	Julie	06 87 45 78 10	exemple.nom at gmail.com	38000 Grenoble 3 rue Joseph	0

La vue client se divise horizontalement en deux parties. La partie haute présente les informations du clients actuellement sélectionné. La partie basse est la liste de tous les clients de la vidéothèque. Cliquer sur une ligne de ce tableau permet de mettre à jour la partie haute. La partie haute présente également une section "Historiques des locations" qui indique toutes les locations en cours pour ce client.

Vue de prototype "Supprimer un Client"

Lorsque l'on clique sur "Supprimer client", une fenêtre modale de confirmation apparaît. Si l'utilisateur clique sur valider, le client est retiré de la base de données ainsi que toutes ses locations en cours s'il en a.

Vue de prototype "Ajouter un Client"

Si l'utilisateur clique sur "Ajouter clients" dans le menu, un formulaire apparaît où tous les champs doivent être remplis. Cliquer sur Valider permet d'ajouter une nouvelle fiche client avec ces caractéristiques dans la base de données et de la sélectionner dans la vue.

Vue de prototype "Éditer un Client"

Lorsque l'utilisateur clique sur "Éditer client", un formulaire apparaît où tous les champs sont pré-remplis avec les caractéristiques actuelles du client sélectionné. Cliquer sur Valider permet de mettre à jour la fiche client actuelle ainsi que ses informations dans la base de données.

Vue de prototype “Gérer facture”

Clients

Ajouter clients

Films

Locations

Roissy Pierre

Numéro client

489528

Nom

Roissy

Prénom

Pierre

Téléphone

06 99 45 12 14

Mail

exemple.nom at gmail.com

Adresse

38000 Grenoble 3 rue Joseph

Supprimer client

Editer client

Gérer f

Num. client	Nom	Prénom
654542	Dupont	Dupont
251651	Marcel	Marcel
895412	Laurent	Laurent
489528	Roissy	Pierre
369741	Charles	Charles
898952	Maxime	Maxime
120200	Lucie	Lucie
011987	Enzo	Enzo
778203	Baptiste	Baptiste
998996	André	André
744544	Charlotte	Charlotte
158745	Julie	Julie

Facture de Roissy Pierre

Historique des locations

Film id	Nom film	Date début location	Date fin location	État
9799503366236	Batman	15/05/2010	30/05/2010	Rendu
9256456366709	Superman	12/08/2010	18/08/2010	Rendu
9545503348955	Iron man	03/07/2010	27/07/2010	En location
9090912345656	Batman 2	15/09/2010	23/09/2010	En location

Afficher facture

Imprimer facture

Envoyer facture par mail

Annuler

Lorsque l'utilisateur clique sur le bouton “Gérer facture”, un tableau représentant l'historique des achats de ce client apparaît. En fonction des achats sélectionnés, la facture se met à jour. L'utilisateur peut faire différentes manipulation avec cette facture, comme l'afficher, l'imprimer, l'envoyer par mail au client (si son mail est renseigné).

Vue de prototype “Louer un film” (1/2)

Clients

Ajouter clients

Films

Locations

Roissy Pierre

Numéro client

489528

Nom

Roissy

Prénom

Pierre

Téléphone

06 99 45 12 14

Mail

exemple.nom at gmail.com

Adresse

38000 Grenoble 3 rue Joseph

Supprimer client

Editer client

Gérer f

Num. client	Nom	Prénom
654542	Dupont	Dupont
251651	Marcel	Marcel
895412	Laurent	Laurent
489528	Roissy	Pierre
369741	Charles	Charles
898952	Maxime	Maxime
120200	Lucie	Lucie
011987	Enzo	Enzo
778203	Baptiste	Baptiste
998996	André	André
744544	Charlotte	Charlotte
158745	Julie	Julie

Location de film pour Roissy Pierre

Scannez ou renseignez le code barre du film...

Code barre film

Annuler

Valider

Lorsque l'utilisateur clique sur le bouton “Louer un film”, une fenêtre modale apparaît où il est demandé un code barre de film. Celui-ci peut être renseigné via un lecteur de code-barre ou bien manuellement par l'utilisateur. Cliquer sur valider permet de continuer le remplissage du formulaire.

Vue de prototype “Louer un film” (2/2)

Clients

Ajouter clients

Films

Locations

Roissy Pierre

Supprimer client

Editer client

Gérer film

Numéro client

489528

Nom

Roissy

Prénom

Pierre

Téléphone

06 99 45 12 14

Mail

exemple.nom.at@gmail.com

Adresse

38000 Grenoble 3 rue Joseph

Historique des locations

Film id	Nom film	Date début location	Date fin location	État
9799503366236	Batman	15/05/2010	30/05/2010	Rendu
9256456366709	Superman	12/08/2010	18/08/2010	Rendu
9545503348955	Iron-man	03/07/2010	27/07/2010	En location
9090912345656	Batman 2	15/09/2010	23/09/2010	En location En retard

Location de film pour Roissy Pierre

Code barre film

9309567348955

Nom film

Le Seigneur des Anneaux : La communauté de l'anneau

Prix à payer

13€

Date début location

25/09/2022

Temps de location

1 semaine

Date fin location

31/05/2010

Valider

Annuler

Num. client	Nom	Prénom
654542	Dupont	Dupont
251651	Marcel	Marcel
895412	Laurent	Laurent
489528	Roissy	Pierre
369741	Charles	Charles
898952	Maxime	Maxime
120200	Lucie	Lucie
011987	Enzo	Enzo
778203	Baptiste	Baptiste
998996	André	André
744544	Charlotte	Charlotte
158745	Julie	Julie

En fonction du code barre fourni, plusieurs informations liées au film apparaissent comme le nom du film et le prix de la location. À cet instant, l'utilisateur doit uniquement préciser une durée de location en semaine, ce qui mettra à jour le prix (qui est indiqué à la semaine) et la date de fin de location. La date de début de location se définit automatiquement à la date d'aujourd'hui, mais reste éditable pour laisser une certaine flexibilité au gérant. Cliquer sur Valider permet d'enregistrer la nouvelle location.

Vue de prototype “Location”

Clients

Films

Locations

Détail de location

Num. location

4896321

Code barre film

CNDSJZDFFCLO

Nom film

Le Seigneur des Anneaux : La communauté de l'anneau

Num. client

489528

Nom client

Roissy Pierre

Date début location

12/05/2010

Date fin location

12/05/2010

Annuler location

Editer location

Mettre à jour location

Prix payé

13€

État de la location

En location

Num. location	Code barre film	Num.client	Date début location	Date fin location	Prix payé	État
45621	9790123456785	45348	12/05/2010	12/05/2010	13€	Rendu
4896321	9880123856885	83258312	12/05/2010	12/05/2010	10€	Rendu
78341	9000123856005	4831208	12/05/2010	12/05/2010	7€	En location En retard
45348	9844423856453	5204652	12/05/2010	12/05/2010	5€	Rendu
124853	9812345859985	0453874	12/05/2010	12/05/2010	9€	En location En retard
45341	9440123456441	1507813	12/05/2010	12/05/2010	11€	Rendu
89424783	9760177756885	3074145	12/05/2010	12/05/2010	11€	En location En retard
4342470	9990987655684	9785450	12/05/2010	12/05/2010	5€	En location
47084806	9010145585666	8014204	12/05/2010	12/05/2010	15€	En location
602100170	9843573853475	750420786	12/05/2010	12/05/2010	27€	Rendu
078750645	9009112385880	6486210	12/05/2010	12/05/2010	19€	En location
0949756564	9333412835635	48363148	12/05/2010	12/05/2010	19€	Rendu

La vue location se divise horizontalement en deux parties. La partie haute présente les informations de la location actuellement sélectionnée. La partie basse est l'historique de toutes les locations ayant eu lieu dans la vidéothèque. Cliquer sur une ligne de

ce tableau permet de mettre à jour la partie haute. La partie haute présente également une section détaillée pour l'état en cours de la location.

Vue de prototype “Supprimer une Location”

Lorsque l'on clique sur “Annuler location”, une fenêtre modale de confirmation apparaît. Si l'utilisateur clique sur valider, la location est annulée et la base de données est mise à jour en conséquence.

Vue de prototype “Éditer une Location”

Lorsque l'utilisateur clique sur “Éditer location”, un formulaire apparaît où tous les champs sont pré-remplis avec les caractéristiques actuelles de la location sélectionnée. Cliquer sur Valider permet de mettre à jour la location ainsi que ses informations dans la base de données.

Vue de prototype “Mettre à jour une Location”

Clients

Films

Locations

Détail de location

Num. location
4896321

Code barre film
CNDSJ2D0FFCLO

Nom film
Le Seigneur des Anneaux : La communauté de l'anneau

Num. client
489528

Nom client
Roissy Pierre

Date début location
12/05/2010

Date fin location
12/05/2010

Prix payé
13€

État de la location
En location

Annuler location

Editer location

Mettre à jour la location

Mettre à jour la location n°4896321

État de la location

☒ En location

☐ Rendu

Valider

Annuler

Num. location	Code barre film	N				
45621	9790123456785	45...				
4896321	9880123856885	83258312	12/05/2010	12/05/2010		Rendu
78341	9000123856005	4831208	12/05/2010	12/05/2010		En retard
45348	9844423856453	5204652	12/05/2010	12/05/2010		Rendu
124853	9812345859985	0453874	12/05/2010	12/05/2010		En retard
45341	9440123456441	1507813	12/05/2010	12/05/2010		Rendu
89424783	9760177756885	3074145	12/05/2010	12/05/2010		En location En retard
4342470	9990987655684	9785450	12/05/2010	12/05/2010		En location
47084806	9010145585666	8014204	12/05/2010	12/05/2010		En location
602100170	9843573853475	750420786	12/05/2010	12/05/2010		Rendu
078750645	9009112385880	6486210	12/05/2010	12/05/2010		En location
0949756564	9333412835635	48363148	12/05/2010	12/05/2010		Rendu

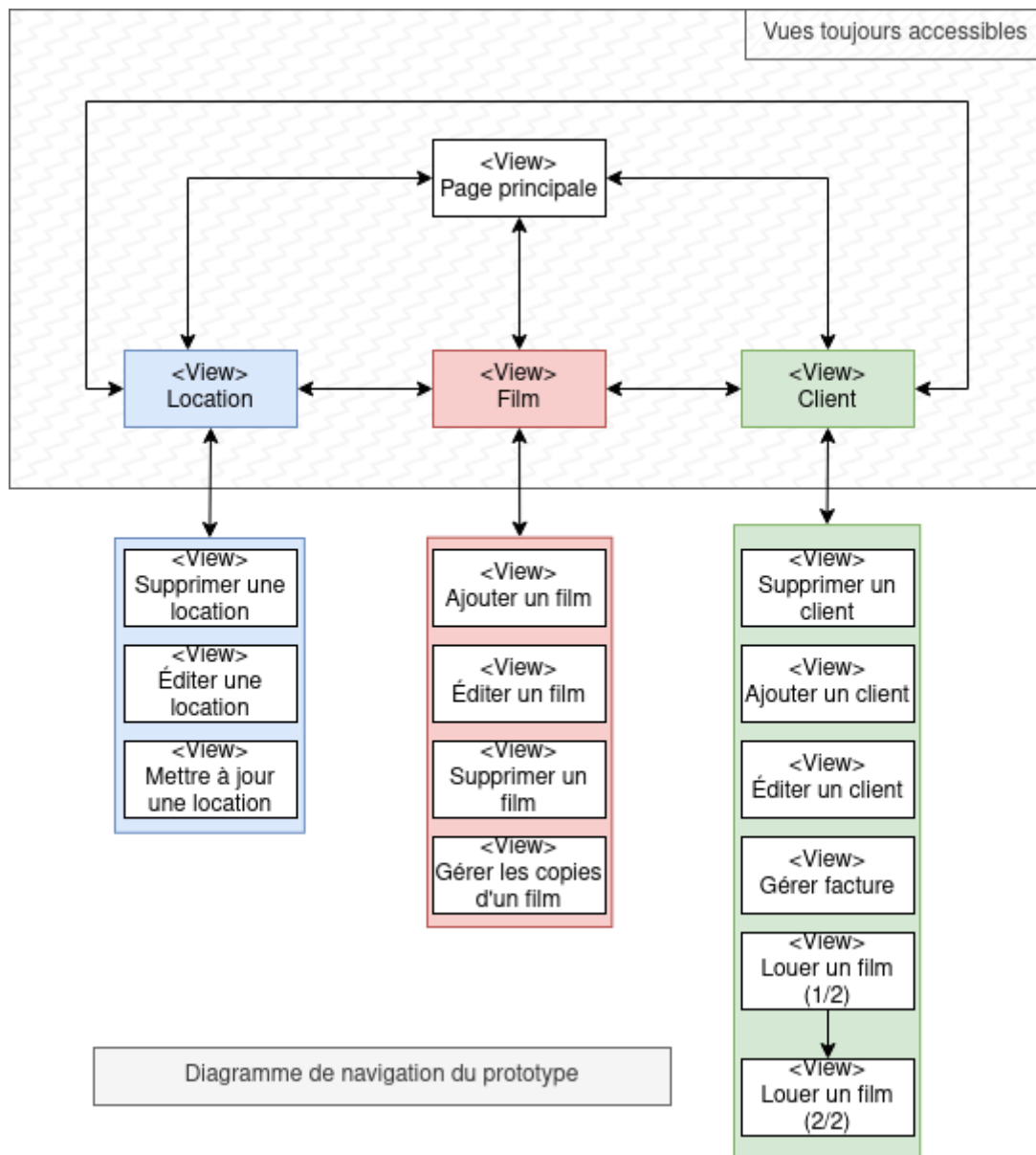
Lorsque l'utilisateur clique sur “Mettre à jour location”, un formulaire de radio boutons apparaît avec tous les états possibles d'une location. L'état qui caractérisait la location est préalablement sélectionné. Cela permet notamment au gérant d'indiquer dans sa base de données qu'un client a rendu un film qu'il avait loué.

b) Enchaînement des vues

Toutes les vues disposent d'un menu sur la gauche qui permet de naviguer rapidement vers les 4 vues principales de l'application :

- la vue d'accueil
- la vue films
- la vue clients
- la vue locations

Les flèches indiquent que l'on peut passer d'une vue à l'autre. Les vues modales (ou bloquantes) sont les vues qui requièrent une action de l'utilisateur (validation, annulation...) et qui bloquent la navigation. Elles sont regroupées entre elles et reliées uniquement à la vue correspondante. Le code couleur permet de signifier cela.



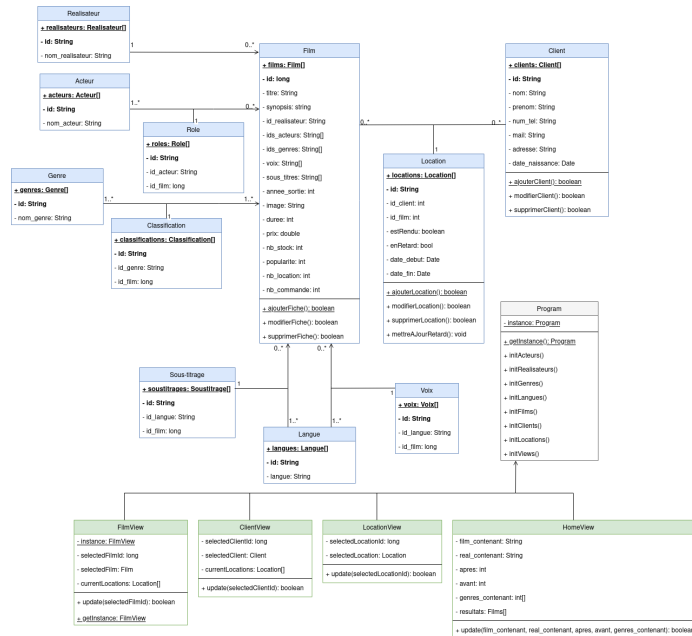
2. Spécifications techniques

- L'application est développée sur .NET en utilisant le langage C# et le patron de conception MVVM. Il consiste à utiliser un modèle pour chaque vue (appelé ModelView), qui met à jour tous les attributs nécessaires à la création d'une vue. Lors d'événements, la vue appelle les méthodes du contrôleur, ces méthodes se chargent de mettre à jour le ModelView et de renvoyer une nouvelle instance de la vue.
- La base de données utilisée est SQLServer Express LocalDB pour sa simplicité et sa rapidité et puisqu'il s'agit d'une BDD avec laquelle on communique localement et par un seul utilisateur (le gérant qui utilise l'application). On n'a donc aucun problème de modification concurrente (plusieurs clients qui tentent de lire/modifier un enregistrement dans la base). Les types de données énoncés ci-dessus sont ainsi en adéquation avec les normes de SQL Server.
- Nous utilisons Entity Framework pour générer un modèle qui s'interface avec la base de données. Le modèle est stocké dans un fichier EDMX. Nous utilisons une méthode "Base First" pour que le modèle colle à la base de données au lancement de l'application.
- Les classes du modèle générées par EntityFramework utilisent linQ pour faire des requêtes dans la base.

- Nous utilisons WPF pour l'interface graphique puisqu'il s'inscrit parfaitement avec le modèle MVVM. WPF utilise le langage XAML (Extensible Application Markup Language) pour décrire la composition des éléments graphiques d'une vue.

a) Modèle de données

Le modèle de données se situe en annexe (sur le git) pour une meilleure lisibilité.



Le modèle de données suit le modèle MVVM comme indiqué plus haut. La partie du modèle qui communique avec la BDD correspond aux classes qui sont nommées en association à une table de cette BDD (au singulier). Par conséquent, les classes colorées en bleu sur le schéma (Client, Film, Location, ...) permettent principalement de faire des requêtes dans la base de données. Le but est de faire une interface entre BDD et modèle pour permettre de ne manipuler que des objets C# tout en modifiant la BDD. Les attributs de ces classes correspondent aux colonnes des tables associées.

Les classes en vert sont les "ModelView". Elles correspondent au modèle d'une vue. Elles disposent de tous les attributs nécessaires à la création de la vue auxquelles elles sont associées. Par exemple, la classe ClientView possède tous les attributs qui vont être utilisés par la vue client. Les classes de ModelView ne sont instanciées qu'une seule fois au lancement de l'application ; elles disposent d'une seule méthode update qui permet de mettre à jour tous les attributs de la classe. Elles possèdent par conséquent le pattern Singleton.

Enfin, la classe Program dispose d'une entrée pour exécuter le programme. Elle instancie les ModelView (une unique fois) et construit les entités en fonction des tables.

b) Modèle physique de données

Le schéma est disponible en entier en annexe. Nous nous proposons d'en détailler certaines tables ici. Les formats de données correspondent tous aux normes de SQL Server.

Ce modèle comporte 11 tables, à savoir :

Table **Films**

Chaque entrée de la table renvoie à un film sur son support DVD comportant un code-barre unique (de type bigint), qui sera alors sa clé primaire.

Les autres champs contiennent les informations sur le film (titre, synopsis, année de sortie, durée en minutes, un chemin vers l'image associée, un réalisateur : clé étrangère faisant référence à un id de type uniqueidentifier de la table réalisateurs), ainsi que sur les DVD associés à ce film (nombres d'exemplaires en stock, loués, les commandes que l'on a passées au fournisseur en attente de réception, ainsi que le prix de location).

Il existe plusieurs copies d'un même film, qui possèdent par conséquent le même code-barre

Films	
PK	code_barre bigint NOT NULL
FK1	realisateur uniqueidentifier NOT NULL
	titre text NOT NULL
	synopsis text NOT NULL
	annee int
	duree int
	image text
	stock_total int NOT NULL
	exemplaires_loues int
	commandes int
	prix float

Table **Réalisateurs**

Cette table standardise les noms des réalisateurs des films mis en location, et ainsi éviter les doublons. Ainsi, la recherche par réalisateur sera possible plus efficace. La clé primaire est un uniqueidentifier, faisant référence au champ réalisateur sous forme textuelle (chaîne de caractères) dans la table Films. Mis à part cet identifiant, la table ne contient que le nom du réalisateur.

Table **Clients**

Contient les informations personnelles et coordonnées des clients. Chaque client possède un uniqueidentifier qui le singularise.

Table **Locations**

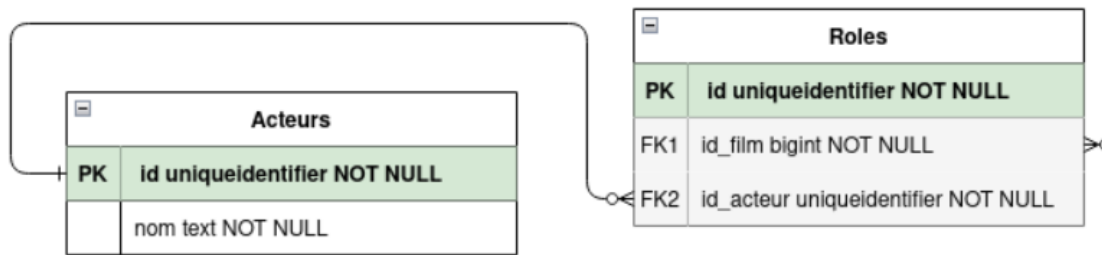
Contient les informations de location de chaque DVD par un client. La clé primaire est un uniqueidentifiant liant un client et un film (ie un DVD de ce film). La date de début et la date de fin sont renseignées, ainsi que l'état de la location, si elle est rendue ou non.

Locations	
PK	id uniqueidentifier NOT NULL
FK1	id_client uniqueidentifier NOT NULL
FK2	id_film bigint NOT NULL
	rendu bit NOT NULL
	date_debut date NOT NULL
	date_fin date NOT NULL

Tables **Rôles, Acteurs**

La table Rôles permet de faire le lien entre un film et un acteur. Cela a été pensé dans le but d'éviter les redondances d'acteurs, et ainsi faciliter une recherche par acteur, mais aussi afin de renseigner plusieurs acteurs par film.

Ainsi, la table acteurs ne contient que des noms d'acteurs (uniques), et la table Rôles permet de faire la liaison entre celle-ci et la table Films via un id_acteur et un id_film.

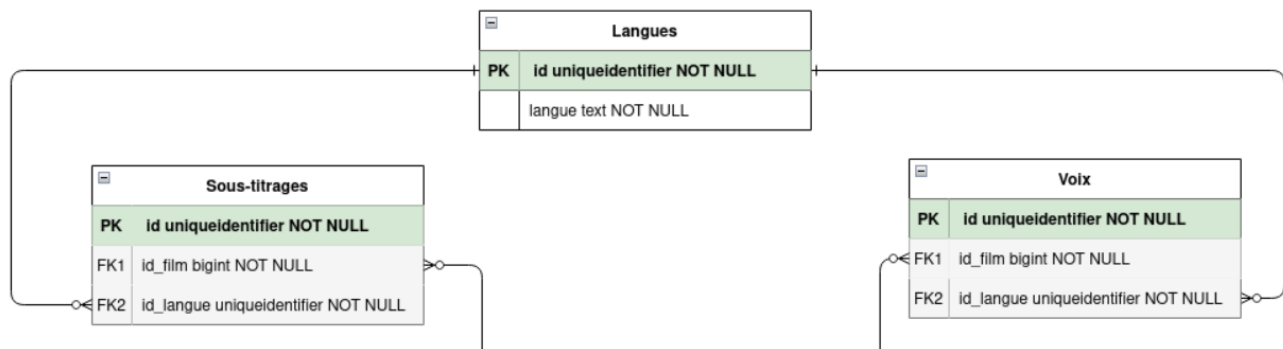


Tables **Classifications, Genres**

Une classification est l'association d'un film à un genre pré-établi. Ainsi, de manière analogue aux tables Rôles et Acteurs, un film peut avoir plusieurs genres et un genre peut être associé à plusieurs films. Nous pouvons pourrions ainsi discriminer les films par genres, ainsi que par rôle et genre (ex: comédies avec Jim Carrey).

Tables **Sous-titrages, Voix et Langues**

Ces tables sont analogues, dans le sens où elles lient un DVD à plusieurs langues écrites / orales. Ces deux tables fonctionnent ainsi à la manière des tables Rôles et Classifications. Elles renvoient à la même table Langues (exemple : un film peut posséder la voix française, les sous-titres en français, ainsi que les sous-titres en espagnol)



Concernant les types de données :

- N'étant pas limités en ressources, nous laissons toutes les données textuelles en type text, sans placer de contraintes sur les nombres de caractères.
- Pour le champ rendu de la table Locations, nous utilisons un type bit NOT NULL, ce qui renvoie à un booléen strict (n'ayant pas la possibilité d'être NULL, il est toujours initialisé à 0).
- Le type uniqueidentifiant nous contraindra à générer un nouvel identifiant à chaque fois que l'on écrira dans les tables concernées.
- Une fiche de film dispose d'une ou plusieurs copies. Toutes les copies possèdent donc toutes le même code barre.

3. Plan de développement

1. Classes et méthodes

Nous présentons dans cette partie la liste des classes et des méthodes développées, ainsi qu'une documentation les concernant.

Classe Film

Décrit les caractéristiques d'un film de la vidéothèque. Celui-ci peut exister en plusieurs exemplaires, et le gérant peut en commander davantage. Le code-barre du film est un numéro unique qui différencie chaque film. Par conséquent, deux exemplaires d'un même film possèdent le même code-barre. La popularité d'un film désigne le nombre de fois qu'il a été loué par un client. Ce nombre ne fait que s'incrémenter et est utilisé pour trier les films sur la page d'accueil du logiciel. L'état d'un film est l'un des états suivants : en rayon / rupture stock / en rayon et commandés / rupture stock et commandés.

Attributs de Film

+ films: Film[] désigne le dictionnaire de films tel que représenté dans la base de données. Il s'agit d'un attribut statique puisqu'il n'est pas pertinent pour l'instance courante d'un film.
- id: long désigne le numéro code-barre du film de l'instance courante.
- titre: String désigne le titre du film de l'instance courante.
- synopsis: String désigne le résumé du film de l'instance courante.
- id_realisateur: String désigne l'id du réalisateur du film de l'instance courante.
- ids_acteurs: String[] désigne la liste des ids des acteurs du film de l'instance courante.
- ids_genres: String[] désigne la liste des ids des genres/catégories du film de l'instance courante.
- voix: String[] désigne la liste des ids des voix disponibles du film de l'instance courante.
- sous_titres: String[] désigne la liste des ids des sous-titres disponibles du film de l'instance courante.
- annee_sortie: int désigne l'année de sortie du film de l'instance courante. On le modélise par un nombre entier pour faciliter plus tard les calculs de tri par ordre du film le plus récent.
- image: String désigne le chemin qui mène à l'image .png pour la fiche du film de l'instance courante.
- duree: int désigne la durée en minutes du film de l'instance courante.
- prix: double désigne le prix à la location du film de l'instance courante.
- nb_stock: int désigne le nombre de copies/d'exemplaires disponibles dans la vidéothèque du film de l'instance courante.
- popularite: int désigne le nombre de fois que le film de l'instance courante a été loué par des clients.
- nb_location: int désigne le nombre de locations en cours du film de l'instance courante.
- nb_commande: int désigne le nombre de copies/exemplaires de ce film en cours de commande. C'est-à-dire les copies commandées mais pas encore arrivées dans le stock.

Méthodes de Film

+ ajouterFiche(): boolean permet d'ajouter une nouvelle fiche de film à la vidéothèque. Cette opération ajoute un nouvel objet Film au dictionnaire de films ainsi que dans la table Films de la base de données. Les attributs de ce film sont récupérés via le formulaire rempli par le gérant de la vidéothèque. Il s'agit d'une méthode statique puisqu'elle n'est pas pertinente pour l'instance courante d'un film en particulier.
+ modifierFiche(): boolean permet de modifier les attributs du film de l'instance courante avec les données du formulaire rempli par le gérant de la vidéothèque. Cela modifie également les attributs de ce film dans la base de données.
+ supprimerFiche(): boolean permet de retirer définitivement le film de l'instance courante du modèle et de la base de données.

Classe **Client**

Décrit les caractéristiques d'un client de la vidéothèque. La location de film se fait à un client. Les factures sont également propres à un client.

Attributs de **Client**

+ **clients: Client[]** désigne le dictionnaire de clients tel que représenté dans la base de données. Il s'agit d'un attribut statique puisqu'il n'est pas pertinent pour l'instance courante d'un client.

- **id: int** désigne l'unique identifiant/id du client de l'instance courante.

- **nom: String** désigne le nom du client de l'instance courante.

- **prenom: String** désigne le prénom du client de l'instance courante.

- **num_tel: String** désigne le numéro de téléphone du client de l'instance courante.

- **mail: String** désigne le courriel du client de l'instance courante.

- **adresse: String** désigne l'adresse du client de l'instance courante.

- **date_naissance: Date** désigne la date de naissance du client de l'instance courante.

Méthodes de **Client**

+ **ajouterClient(): boolean** permet d'ajouter un nouveau client à la vidéothèque. Cette opération ajoute un nouvel objet Client au dictionnaire de clients ainsi que dans la table Client de la base de données. Les attributs de ce client sont récupérés via le formulaire rempli par le gérant de la vidéothèque. Il s'agit d'une méthode statique puisqu'elle n'est pas pertinente pour l'instance courante d'un client en particulier.

+ **modifierClient(): boolean** permet de modifier les attributs du client de l'instance courante avec les données du formulaire rempli par le gérant de la vidéothèque. Cela modifie également les attributs de ce client dans la base de données.

+ **supprimerClient(): boolean** permet de retirer définitivement le client de l'instance courante du modèle et de la base de données.

Classe d'association **Location**

Décrit les caractéristiques de la location d'un film par un client dans la vidéothèque. L'état d'une location est l'un des états suivants : rendu / en location / en location et en retard. Lorsqu'un client rend un film, c'est le gérant de la vidéothèque qui doit mettre à jour l'état de la location. Lorsqu'un client ne rend pas un film dans les délais (la date actuelle a dépassé la date de fin de location) c'est le logiciel qui met à jour automatiquement l'état de la location.

Attributs de **Location**

+ locations: Location[] désigne le dictionnaire de locations tel que représenté dans la base de données. Il s'agit d'un attribut statique puisqu'il n'est pas pertinent pour l'instance courante d'une location.

- id: int désigne l'unique identifiant/id de la location de l'instance courante.

- id_client: int désigne l'id du client propriétaire de la location de l'instance courante.

- id_film: int désigne l'id du film loué par le client propriétaire de la location de l'instance courante.

- estRendu: boolean indique si le client a finalisé la location en cours.

- enRetard: bool indique si la location en cours est en retard.

- date_debut: Date désigne la date de début de location.

- date_fin: Date désigne la date de début de location.

Méthodes de **Location**

+ ajouterLocation(): boolean permet d'ajouter une nouvelle location à l'historique de location. Cette opération ajoute un nouvel objet Location au dictionnaire de locations ainsi que dans la table Locations de la base de données. Les attributs de cette location sont récupérés lors de l'ajout d'une location à un client par le gérant de la vidéothèque. Il s'agit d'une méthode statique puisqu'elle n'est pas pertinente pour l'instance courante d'une location en particulier.

+ modifierLocation(): boolean permet de modifier les attributs de la location de l'instance courante avec les données du formulaire rempli par le gérant de la vidéothèque. Cela modifie également les attributs de cette location dans la base de données.

+ supprimerLocation(): boolean permet de retirer définitivement la location de l'instance courante du modèle et de la base de données.

Classe **FilmView**

Il n'existe qu'une instance de la classe FilmView, utilisant le design pattern singleton. Cette instance est initialisée au lancement de l'application. La classe permet de faire tous les calculs nécessaires, en récupérant les données du modèle, pour afficher correctement la vue des films. Elle est mise à jour dans le contrôleur juste avant de renvoyer la vue grâce à sa méthode update.

Attributs de **FilmView**

- **instance: FilmView** désigne l'unique instance de la classe FilmView, avec une visibilité restreinte.
- **selectedFilmId: long** désigne l'id du film actuellement sélectionné sur la vue.
- **selectedFilm: Film** désigne le film actuellement sélectionné sur la vue.
- **currentLocations: Location[]** désigne la liste des locations du film actuellement sélectionné sur la vue.

Méthodes de **FilmView**

- + **update(long selectedFilmId): boolean** permet de mettre à jour les attributs de FilmView à partir d'un nouvel id de film sélectionné.
- + **getInstance: FilmView** permet d'accéder à l'unique instance de FilmView.

Classe **ClientView**

Il n'existe qu'une instance de la classe ClientView, utilisant le design pattern singleton. Cette instance est initialisée au lancement de l'application. La classe permet de faire tous les calculs nécessaires, en récupérant les données du modèle, pour afficher correctement la vue des clients. Elle est mise à jour dans le contrôleur juste avant de renvoyer la vue grâce à sa méthode update.

Attributs de **ClientView**

- **instance: ClientView** désigne l'unique instance de la classe ClientView, avec une visibilité restreinte.
- **selectedClientId: long** désigne l'id du client actuellement sélectionné sur la vue.
- **selectedClient: Client** désigne le client actuellement sélectionné sur la vue.
- **currentLocations: Location[]** désigne la liste des locations du client actuellement sélectionné sur la vue.

Méthodes de **ClientView**

- + **update(int selectedClientId): boolean** permet de mettre à jour les attributs de ClientView à partir d'un nouvel id de client sélectionné.
- + **getInstance: ClientView** permet d'accéder à l'unique instance de ClientView.

Classe **LocationView**

Il n'existe qu'une instance de la classe **LocationView**, utilisant le design pattern singleton. Cette instance est initialisée au lancement de l'application. La classe permet de faire tous les calculs nécessaires, en récupérant les données du modèle, pour afficher correctement la vue des locations. Elle est mise à jour dans le contrôleur juste avant de renvoyer la vue grâce à sa méthode **update**.

Attributs de **LocationView**

- **instance: LocationView** désigne l'unique instance de la classe **LocationView**, avec une visibilité restreinte.

- **selectedLocationId: long** désigne l'id de la location actuellement sélectionné sur la vue.

- **selectedLocation: Location** désigne la location actuellement sélectionnée sur la vue.

Méthodes de **LocationView**

+ **update(int selectedLocationId): boolean** permet de mettre à jour les attributs de **LocationView** à partir d'un nouvel id de location sélectionné.

+ **getInstance: LocationView** permet d'accéder à l'unique instance de **LocationView**.

Classe **HomeView**

Il n'existe qu'une instance de la classe **HomeView**, utilisant le design pattern singleton. Cette instance est initialisée au lancement de l'application. La classe permet de faire tous les calculs nécessaires, en récupérant les données du modèle, pour afficher correctement la vue principale de l'application. Elle est mise à jour dans le contrôleur juste avant de renvoyer la vue grâce à sa méthode **update**.

Attributs de **HomeView**

- **instance: HomeView** désigne l'unique instance de la classe **HomeView**, avec une visibilité restreinte.

- **film_contenant: String** désigne le filtre utilisé sur les titres des films retournés. Si l'attribut vaut null, alors aucune recherche n'est faite sur les titres de films, là où la chaîne vide autorise tous les films.

- **real_contenant: String** désigne le filtre utilisé sur les noms de réalisateur des films retournés. Si l'attribut vaut null, alors aucune recherche n'est faite sur les réalisateurs des films, là où la chaîne vide autorise tous les films.

- **apres: int** désigne le filtre utilisé sur la date de sortie minimale des films retournés. Si l'attribut vaut null, alors aucune recherche n'est faite sur les dates minimum de sortie des films.

- **avant: int** désigne le filtre utilisé sur la date de sortie maximale des films retournés. Si l'attribut vaut null, alors aucune recherche n'est faite sur les dates maximum de sortie des films.

- **genres_contenant: int[]** désigne la liste des ids des genres que doivent contenir les films retournés.

- **resultats: Films[]** désigne les films qui résultent de la recherche par filtrage.

Méthodes de **HomeView**

+ **update(film_contenant, real_contenant, apres, avant, genres_contenant): boolean** permet de mettre à jour les attributs de **HomeView** à partir des paramètres de filtrage.

+ **getInstance: HomeView** permet d'accéder à l'unique instance de **HomeView**.

Classe **Program**

Il n'existe qu'une instance de la classe Program, utilisant le design pattern singleton. Il s'agit du premier appel au lancement de l'application. Cela consiste à initialiser à faire deux opérations :

- initialiser un objet Film, Client, Location, etc pour chaque enregistrement dans les tables correspondantes. Par exemple, si trois enregistrements de films sont présents dans la table Film de la base de données, alors le modèle va lire la base de données et instancier trois objets de type Film avec les bons attributs.
- initialiser tous les dictionnaires de toutes les classes à partir de la base de données. Pour reprendre l'exemple précédent, l'attribut statique du dictionnaire des films devra contenir trois éléments de type film qui pointent vers les trois instances qui viennent d'être créées.

Attributs de **Program**

- **instance: Program** désigne l'unique instance de la classe Program, avec une visibilité restreinte.

Méthodes de **Program**

+ **getInstance(): Program** permet d'accéder à l'unique instance de Program.

+ **initActeurs()** permet de lire la table Acteurs de la base de données et d'instancier tous les objets Acteur. Ajoute également les objets Acteurs au dictionnaire d'acteurs.

+ **initRealisateurs()** permet de lire la table Realisateurs de la base de données et d'instancier tous les objets Realisateur. Ajoute également les objets Realisateur au dictionnaire de réalisateurs.

+ **initGenres()** permet de lire la table Genres de la base de données et d'instancier tous les objets Genre. Ajoute également les objets Genre au dictionnaire de genres.

+ **initLangues()** permet de lire la table Langues de la base de données et d'instancier tous les objets Langue. Ajoute également les objets Langue au dictionnaire de langues.

+ **initFilms()** permet de lire la table Films de la base de données et d'instancier tous les objets Film. Ajoute également les objets Film au dictionnaire de films.

+ **initClients()** permet de lire la table Clients de la base de données et d'instancier tous les objets Client. Ajoute également les objets Client au dictionnaire de clients.

+ **initLocations()** permet de lire la table Locations de la base de données et d'instancier tous les objets Location. Ajoute également les objets Location au dictionnaire des locations.

+ **initViews()** permet d'initialiser tous les singletons des vues au lancement de l'application. Le premier id de la liste des clients, films, locations est choisi pour initialiser les vues.

Il reste les classes d'associations Role, Classification, Sous-titrage, Langue, Voix qui se construisent de la même manière :

- on retrouve un attribut Collection qui liste tous les objets instanciés de ce type ;
- un attribut id String qui est la clé primaire de la table représentée dans la classe ;
- des attributs id_ * String qui font référence à une clé primaire d'une autre table ;
- enfin, possiblement, un attribut String qui décrit l'enregistrement dans la table (ex: "français", "jamel debbouze", "action", ...).

3. Contraintes

1. Contraintes de délai

Livrable 1 pour le mercredi 12 octobre minuit.

Livrable 2, logiciel et manuels pour le lundi 12 décembre minuit.

2. Contraintes matérielles

Une machine virtuelle Windows pour exécuter Visual Studio, l'un de nous deux étant sous linux.

La livraison du logiciel se fera sous la forme d'une archive .zip contenant entre autres l'exécutable et des manuels d'installation et d'utilisation adaptés à cette situation.

3. Contraintes fonctionnelles

L'utilisateur doit modifier par lui-même la base de données via l'interface pour les opérations suivantes :

- mettre à jour le stock de ses films (lorsqu'une commande est arrivée) ;
- mettre à jour un emprunt (lorsqu'un client rend un film) ;
- ajouter/supprimer un client, une fiche de film

Il ne devra à aucun moment être confronté au code, ni à des messages d'erreurs non explicites et pour lesquels il n'a aucun contrôle. Voici les types de message d'erreur auxquels l'utilisateur peut être confronté :

- "le code barre ne pointe vers aucun film de la table"
- "le numéro de client n'est associé à aucun client de la table"
- "le film — est actuellement en rupture de stock"
- "la date de fin de location précède la date de début de location"

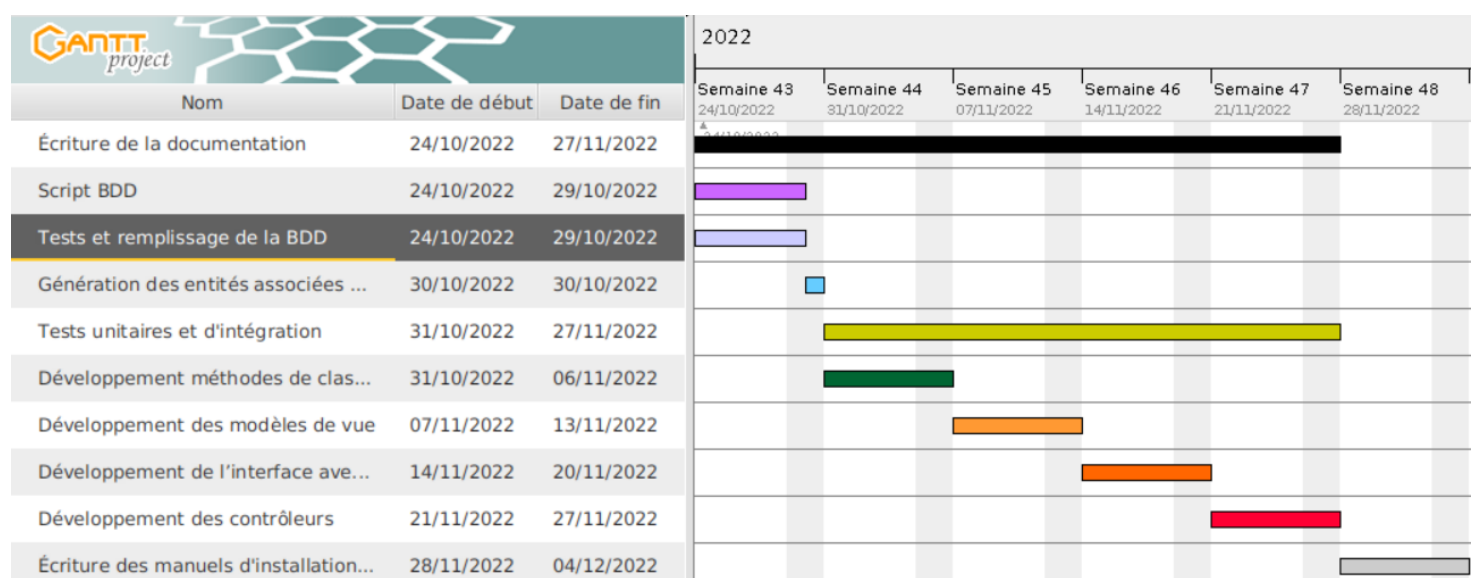
4. Déroulement du projet

La première phase du projet est une phase de conception. Il s'agit de rendre ce premier livrable vers mi-octobre qui spécifie en détail les fonctionnalités et les technologies utilisées pour le développement d'un logiciel. La deuxième phase consiste à implémenter le logiciel et faire des tests.

En nous basant sur le travail réalisé au cours de la rédaction de ce livrable, nous pouvons expliciter les tâches, ainsi que leur agencement et leur durée :

- L'écriture de la documentation se fera tout au long du projet
- Les tests unitaires et d'intégration (hors tests de la BDD) interviendront à partir de la **Semaine 2**, et seront effectués tout au long du projet
- **Semaine 1 (24/10 - 30/10)** : Script SQL de la base de données, tests et remplissage de cette dernière par un jeu de données factice, et génération d'entités via Entity Framework (EF).
- **Semaine 2 (31/10 - 06/11)** : Développement des méthodes de classes pour les modèles générés par EF en utilisant linq, afin de faire correspondre nos objets c# aux classes Acteur, Réalisateur, Genre, Langue, Film, Client, ainsi qu'aux classes d'association Rôle, Classification, Voix, Sous-titrage et Location.
- **Semaine 3 (07/11 - 13/11)** : Développement des modèles de vue selon les classes FilmView, ClientView, LocationView et HomeView du modèle de données
- **Semaine 4 (14/11 - 20/11)** : Développement de l'interface en utilisant WPF
- **Semaine 5 (21/11 - 27/11)** : Développement des contrôleurs
- **Semaine 6 (28/11 - 04/12)** : Écriture des manuels d'installation et d'utilisation, puis livraison du logiciel avec sa documentation et son plan de tests.

Nous suivrons l'avancée du logiciel à partir du diagramme de Gantt ci-dessous.



5. Références & Ressources

Toutes les annexes de ce document sont au format pdf sur le git du projet : <https://github.com/GentilBastien/Videothèque>

5.1 Outils

<https://app.diagrams.net/>

Afin de réaliser nos diagrammes

<https://www.figma.com/>

Afin de construire notre prototype

5.2 Ressources des spécifications techniques et fonctionnelles

<https://learn.microsoft.com/fr-fr/visualstudio/get-started/csharp/?view=vs-2022>

Documentation microsoft pour le langage C#

<https://learn.microsoft.com/en-us/sql/database-engine/configure-windows/sql-server-express-localdb?view=sql-server-ver16>

Documentation microsoft pour SQL Server Express LocalDB

<https://learn.microsoft.com/fr-fr/ef/ef6/modeling/designer/workflows/database-first>

Indications sur la méthode Database-First

<https://learn.microsoft.com/en-us/windows/communitytoolkit/mvvm/introduction>

Introduction au design pattern MVVM

<https://learn.microsoft.com/fr-fr/visualstudio/designers/getting-started-with-wpf?view=vs-2022>

Introduction à WPF