

## Rapport C# Application de locations de jeux vidéo

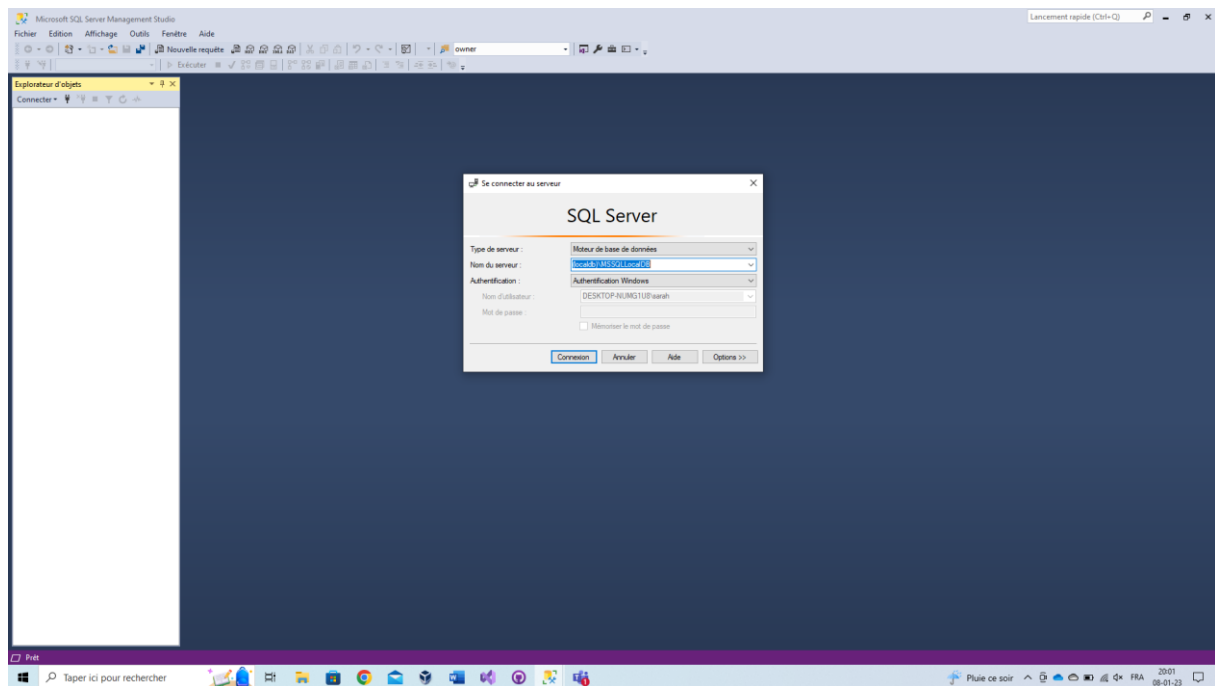
### GameSwitch

#### Mise en place de l'application

1. Ouvrir MSSQL
2. Se connecter en local

Pour se connecter en local, il faut entrer une chaîne de connexion qui est généralement celle-ci :

(localdb)\MSSQLLocalDB



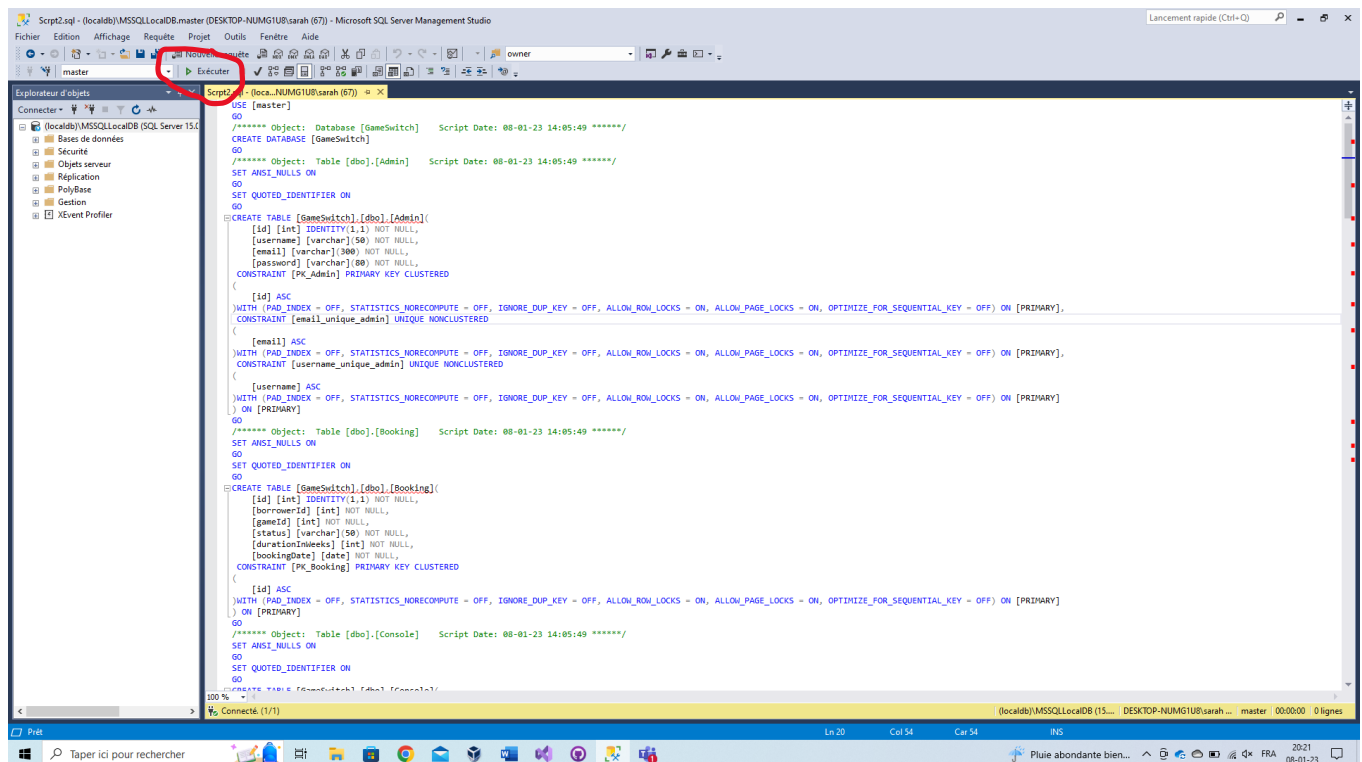
Cliquer sur connexion, on est normalement connecté en local.

3. Vérifier qu'il n'y a aucune base de données nommée GameSwitch, s'il y en a une, la supprimer.
4. Exécuter le script de création de la Base de données, des tables et d'insertion des données :

Le script se trouve à la racine lors du dézippage du projet : scrpt2.sql

Aller dans fichier dans le menu, ouvrir, fichier, choisir le fichier là où il est sur l'ordinateur.

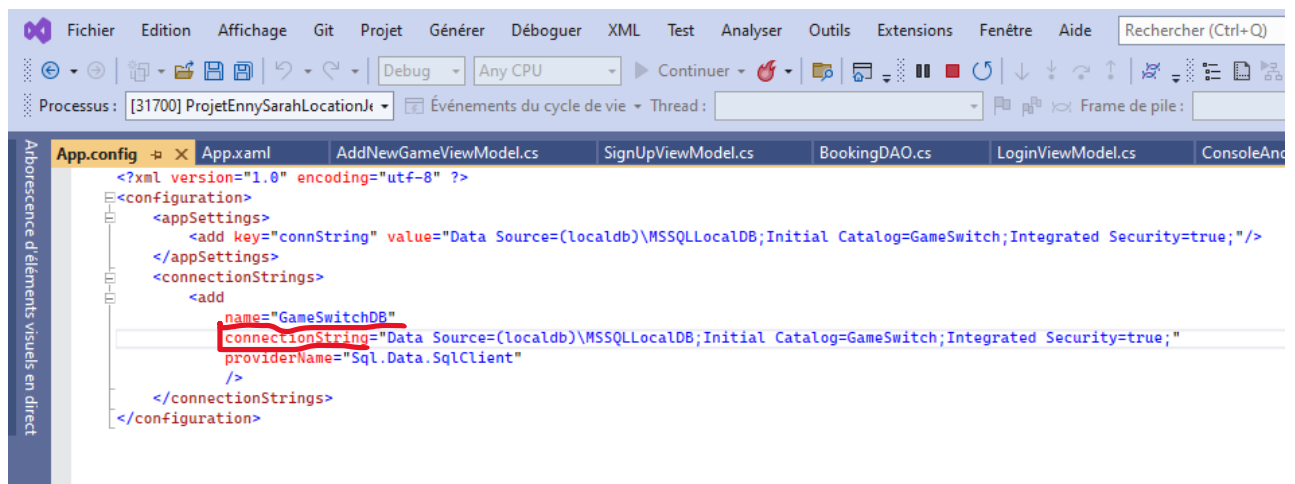
Le fichier va s'ouvrir, cliquer ensuite sur exécuter pour exécuter le script



La base de données devrait normalement être créée et les données insérées.

Si la chaîne de connexion à la base de données est différente, il faut la changer.

La chaîne de connexion se trouve dans le fichier App.config, il faut juste changer la chaîne de caractères qui se trouve dans connectionString



Une fois la mise en place de la base de données effectuée, on peut lancer l'application.

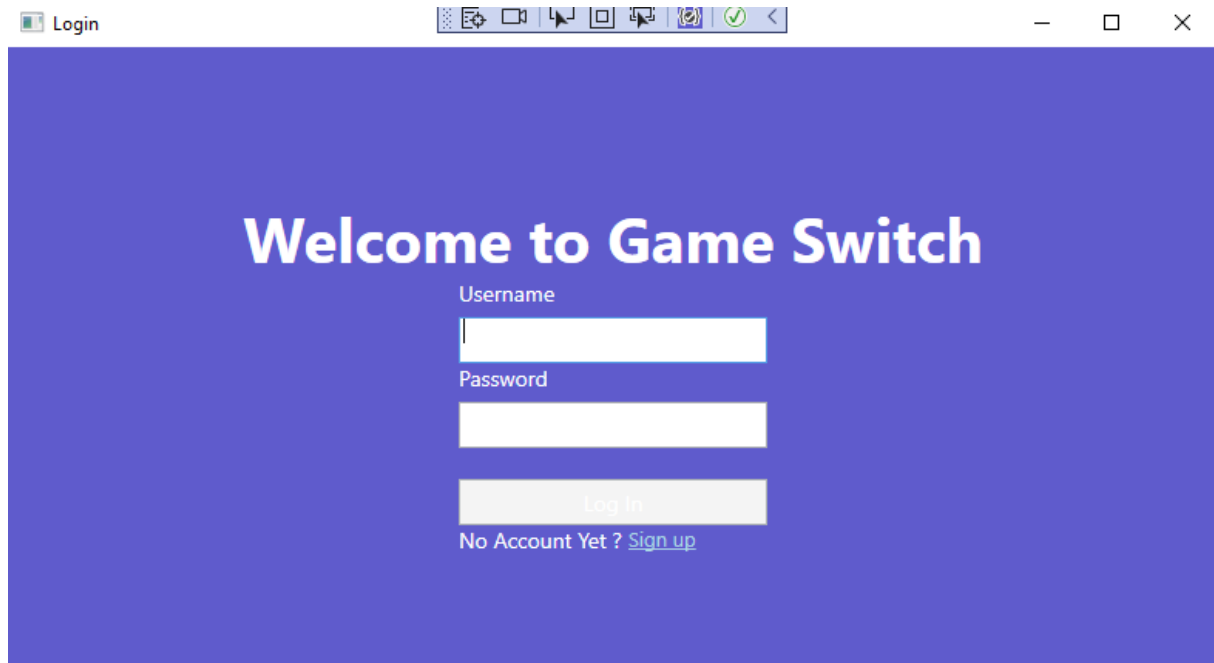
Pour lancer l'application il faut soit ouvrir la solution donnée dans notre zip, soit la cloner depuis github.

Voici le lien github : <https://github.com/sara985/ProjetEnnySarahLocationJeux>

En lançant l'application, on arrive sur une page de login. Cette page est pour les admins et pour les utilisateurs.

Il y a déjà deux utilisateurs et un admin de créé. Les utilisateurs ont pour username : sarah et enny. Leur mot de passe est le même : condorcet

L'administrateur a également le même mot de passe et son username est admin.



The screenshot shows a web browser window with the title 'Login'. The browser's address bar and toolbar are visible at the top. The main content area has a solid blue background. In the center, the text 'Welcome to Game Switch' is displayed in a large, white, sans-serif font. Below this text, there is a login form with three white input fields. The first field is labeled 'Username' and the second is labeled 'Password'. Below these fields is a white button with the text 'Log In'. At the bottom of the form, there is a link that says 'No Account Yet ? [Sign up](#)'.

Trois options sont possibles :

- Cliquer sur signup pour aller à la page d'inscription et s'inscrire
- Entrer son username et son mot de passe afin de se connecter en tant qu'utilisateur.
- Entrer son username et son mot de passe afin de se connecter en tant qu'administrateur

La première me semble assez simple donc je ne vais pas expliquer plus que ça.

#### 1. User

On entre le bon username et le bon mot de passe, on arrive ensuite sur la page my account.

PlayerMainWindow

My Account  
Catalog  
My Games

Hello Sarah [Sign out](#)

Your Current Data:

Number Of Credits

You are currently renting

Number of Games Taken (Borrowed)

You can Modify your Data below

Email  [Modify](#)

Your Username :  [Modify](#)

Your Password :  [Modify](#)

[Delete your Account](#)

Cette page permet de se déconnecter, de mettre à jour son mail, son username et son mot de passe ainsi que d'effacer son compte.

La page Catalog permet de voir tous les jeux vidéo répertoriés dans le catalogue. Ce n'est pas forcément parce qu'un jeu s'y trouve qu'il possède une copie. Si le jeu ne possède aucune copie, il sera impossible de louer/booker. Si le joueur n'a pas assez de crédit sur son compte, le bouton book me/rent me sera grisé. Le bouton book me/ rent me change selon si le jeu vidéo sélectionné a des copies disponibles ou pas. Si un jeu a au moins une copie disponible, il sera possible de le louer directement, sinon il faudra effectuer une réservation. Le bouton I own this game permet de déclarer qu'on possède ce jeu. Il y a trois critères de recherche, un pour chercher selon le nom du jeu, un autre pour chercher selon la console et on peut éventuellement choisir la version. Le bouton Reset permet d'effacer les critères de recherche.

PlayerMainWindow

My Account  
Catalog  
My Games

Hello Sarah [Sign out](#)

[Book Me](#) [I own this game](#) Search  Console  Version  [Reset](#)

**Zelda**  
4 Credits  
2006  
XBOX 360

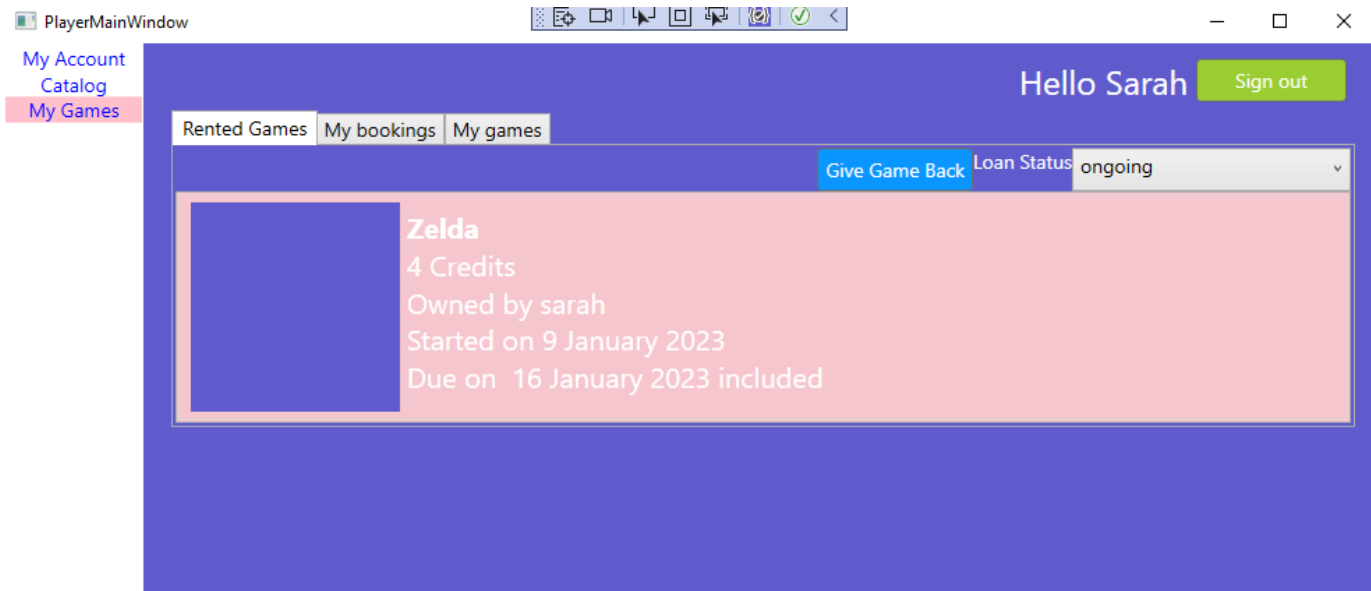
**MarioKart**  
3 Credits  
2008  
XBOX One

**Kingsman**  
4 Credits  
2012  
PlayStation 4

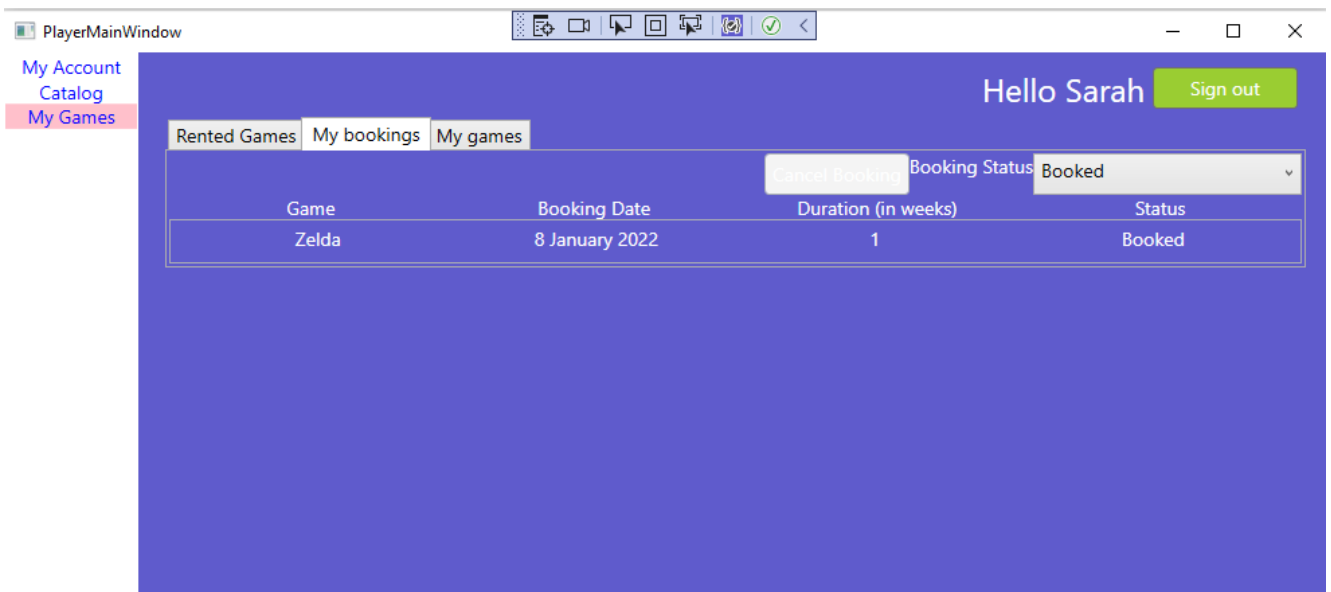
**Sonic Colors**  
4 Credits  
2021  
PlayStation 5

Et dernière page, la page My Games. Cette page contient 3 onglets et donc pas mal de fonctionnalités.

Le premier onglet Rented Games permet de voir les jeux qu'on loue et ceux qu'on a loués. Il y a un bouton Give game back pour rendre le jeu. Si il y a une réservation pour ce jeu elle commencera automatiquement.



Deuxième onglet, My bookings.



Dans cet onglet, on peut voir les réservations et les trier selon qu'elles soient devenues une réservation (Booked), En attente d'acceptation (Waiting) ou annulées (Canceled). On peut annuler une réservation en attente.

Et le dernier onglet, My games.

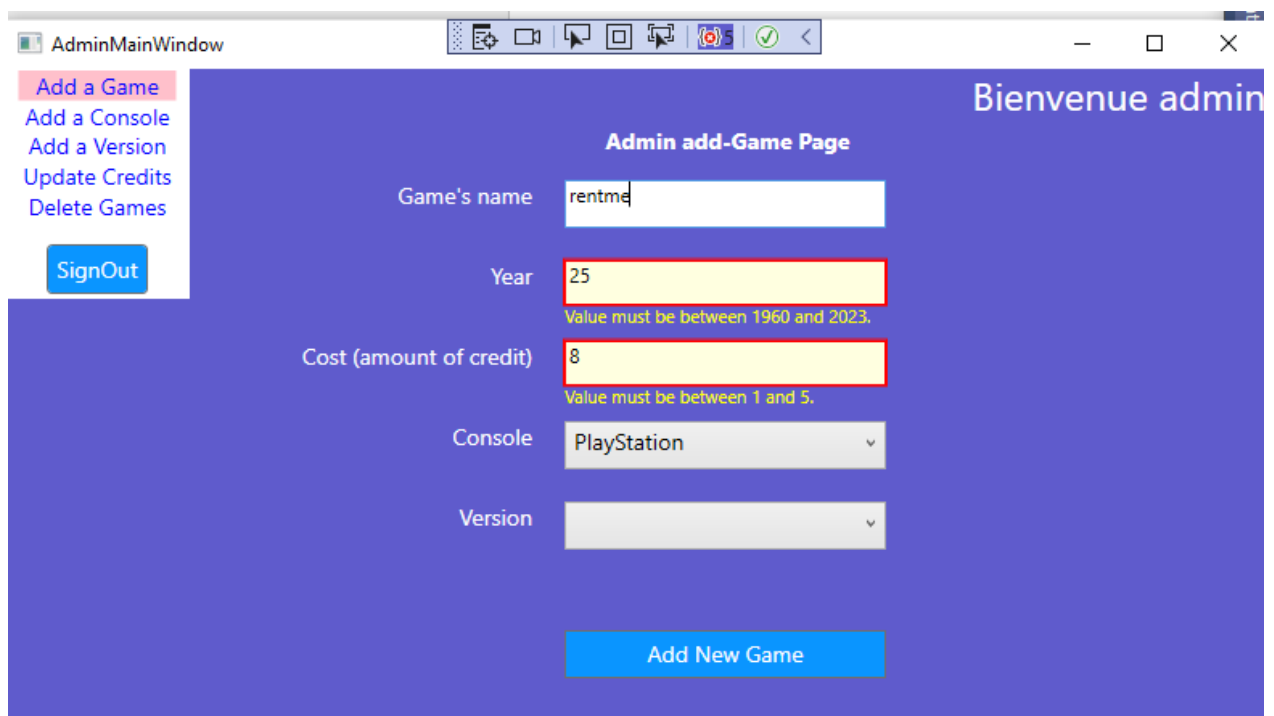


Cet onglet liste les jeux que l'on possède et nous permet de s'en débarrasser à condition qu'il n'ait pas été loués.

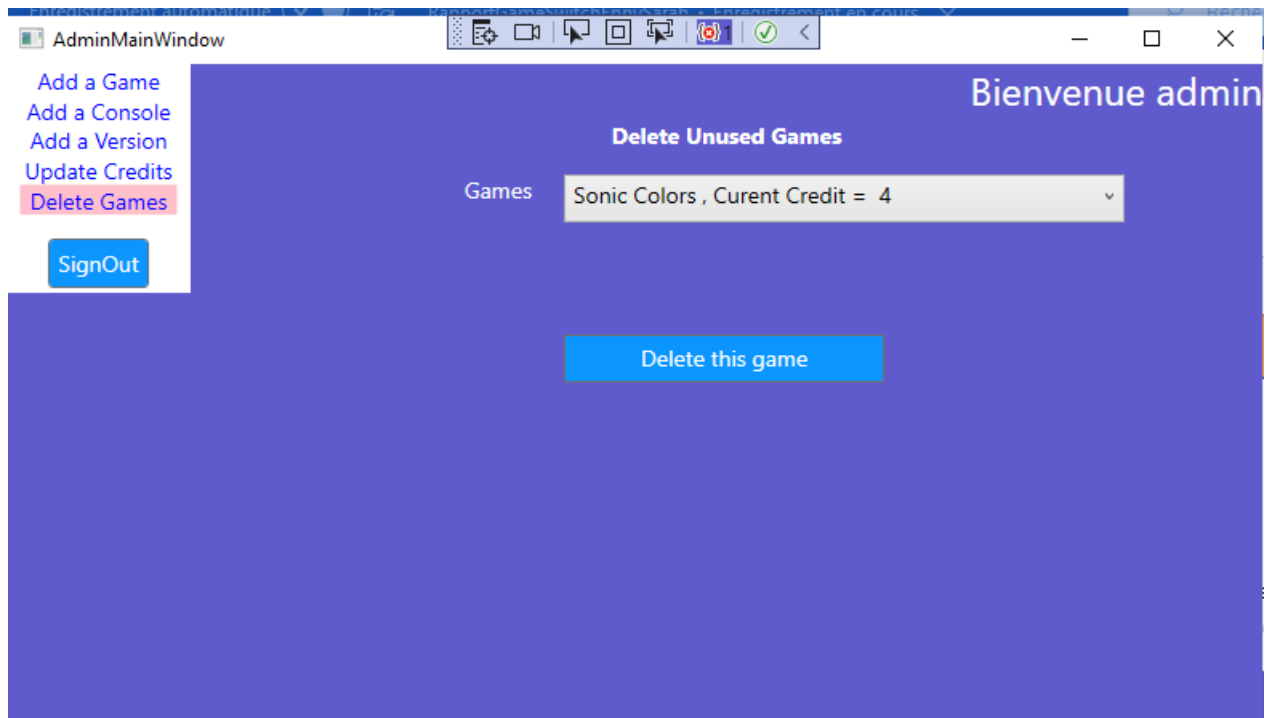
## 2. Admin

La deuxième partie de cette application est la partie admin.

La première page permet d'ajouter un jeu. Quelques contrôles sont présents.



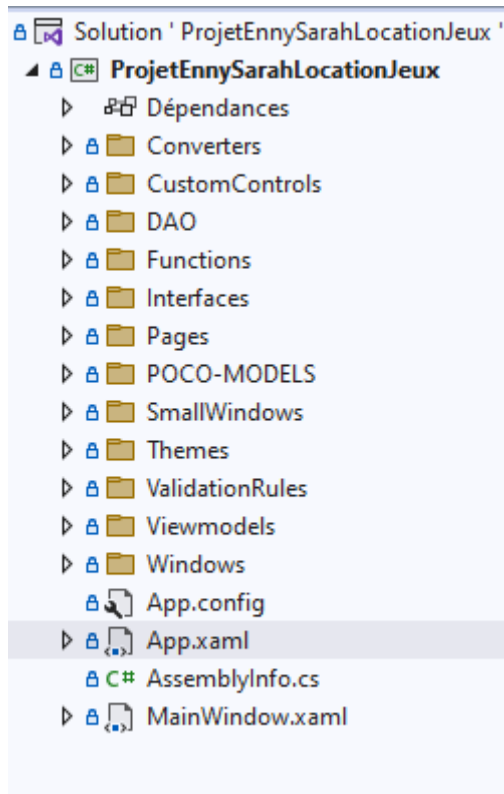
Les onglets add a console, add a version et update credit sont assez explicites.



L'onglet Delete games permet de supprimer les jeux inutilisés, c'est-à-dire, les jeux qui n'ont aucune copie. Il faut sélectionner un jeu dans la liste s'il y en a un

## Architecture

Ce projet utilise le modèle MVVM (Model View ViewModel). Ce modèle nous a été expliqué en classe. Il sépare les données de l'application (le modèle), la vue qui affiche les données et la logique de l'application (le view model). Le view model est responsable de fournir les données nécessaires à la vue et de gérer les interactions utilisateur, tandis que la vue s'occupe simplement de l'affichage de



ces données et du déclenchement d'actions sur le viewmodel. Cela permet une meilleure séparation des préoccupations et une réutilisation plus facile du code.

Comme on peut le voir sur l'image ci-contre, l'application est divisée en plusieurs dossiers.

Converters comprend des classes pour convertir une valeur en un attribut xaml. Ils sont utilisés pour convertir par exemple une liste vide en Visibility.Collapsed. Ils sont surtout utilisés pour montrer ou non un TextBlock comme « Vous n'avez pas de réservation ». On ne va pas le montrer s'il y a une réservation.

Les CustomControls sont des contrôles qu'on peut créer soi-même, ils peuvent ou non hériter d'un contrôle de base (TextBox par exemple). Ils peuvent également être faits de toute pièce comme pour le NavButton ici. On peut alors l'utiliser comme un contrôle normal.

```
<customControls:NavButton ItemName="My
Account" NavLink="/Pages/MyAccountPage.xaml"/>
```

Je passe maintenant au dossier Themes, ce dossier est créé automatiquement lors de la création d'une classe de contrôle personnalisé, afin de lui donner une apparence. Dans ce projet, ce dossier a été créé en même temps que la classe NavButton.

Le dossier POCO-MODELS contient toutes les classes métiers de l'application (Administrator, Booking, ConsoleAndVersion, Copy, Loan, Player, User, VideoGame) ainsi qu'un enum : Status qui définit les statuts possibles pour une réservation (En attente, Annulé ou booké). Il est à noter que Player et Administrator héritent tous deux de la classe abstraite User.

Je reviens maintenant un peu plus haut afin de parler des classes DAO. DAO signifie Data Access Object. C'est un design pattern qui permet de séparer les couches de l'application, notamment la couche liée à la persistance des données du reste de l'application. Si on passe de SQL Server à oracle, il n'y aura ainsi que ce dossier à modifier. Il contient donc différentes classes pour chaque objet qui en a besoin (Player, Admin, Booking, VideoGame, ConsoleAndVersion, Copy, Loan)

Le dossier Functions contient des fonctions globales, il n'y en a qu'une qui est InitCap qui remplace la fonction INITCAP() d'oracle qui n'est donc pas disponible sur SQL server. Elle est utilisée pour mettre le nom et prénom avec une majuscule avant de l'insérer dans la base de données.

Le dossier Interfaces contient des fichiers interfaces, pour ce projet il y en a deux. IDAOInterface qui définit les fonctions de base à implémenter dans chaque DAO et ICloseWindows qui indique qu'une fenêtre peut se refermer à l'aide d'une commande dans le viewmodel.

Le dossier Pages contient toutes les pages de l'application.

Le dossier SmallWindows contient les petites fenêtres qui sont ouvertes lors de la réservation ou de la location d'un jeu pour savoir combien de semaine l'utilisateur aimerait louer/réserver le jeu. Il peut choisir de une à quatre semaines.

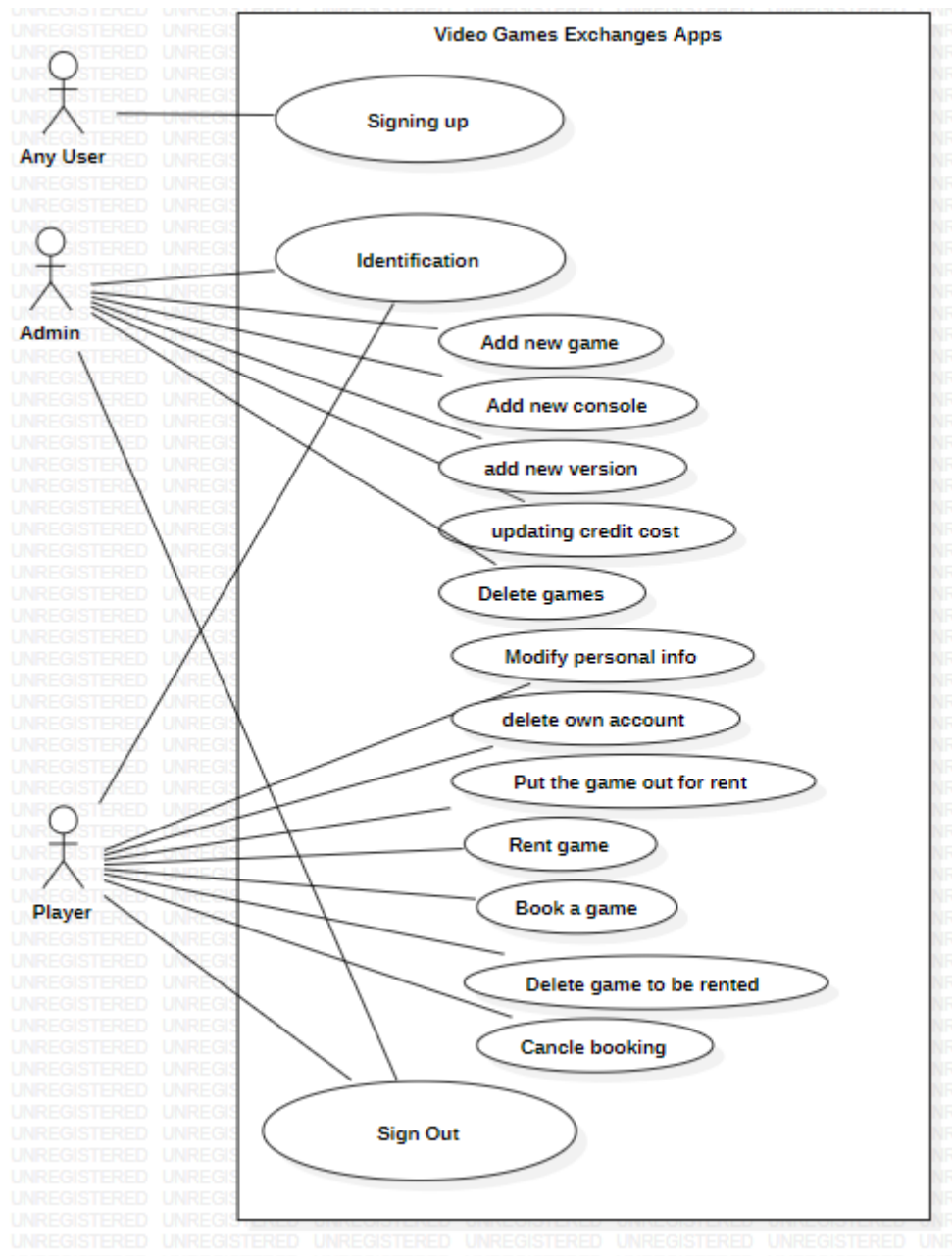
Le dossier ValidationRules contient les règles de validation pour un champ précis, c'est à lier à un contrôle modifiable par l'utilisateur. (TextBox, DatePicker)

Le dossier Viewmodels reprend les différents viewmodels. Un viewmodel est une classe qui contient les données et les comportements qui sont nécessaires pour alimenter une vue ou une interface utilisateur dans une application. Le viewmodel est généralement utilisé pour fournir une couche de liaison entre les données de l'application et la vue, en permettant à la vue de se concentrer sur l'affichage des données et non sur la logique de l'application. Le viewmodel est souvent utilisé avec le modèle de conception MVVM (Model-View-ViewModel), où il joue le rôle de "modèle de vue".

Le dossier Windows contient les fenêtres principales : LoginWindow, SignupWindow, PLayerWindow et AdminWindow.

## Diagramme de Use Case





## Use Case Description/Scenario of Video-Games Exchanges Desktop Apps

Identifier :

UC-1 : USER -SIGNUP

Actor : New User

Precondition : Login Page Ready, Signup Page Ready, Database Connected

Nominal Scenario :

1. User Open the Apps

2. Login Page is Opened, and a link for signup as well
3. User click on the SignUp link
4. Directed to SignUp Page
5. User Enter all the necessary data
6. Click on the save button
7. A message appear informing that now the user can Login
8. Back to Login Page

Alternative Scenario:

5. User doesn't complete the form correctly
6. User click the save button
7. Stay in the same page, Error Message launched

Identifier:

UC-2: USER -LOGIN

Actor : User (Any User)

Precondition : Login Page Ready, Database Connected, The User is registered in the Data-Base

Nominal Scenario :

1. User Open the Apps
2. User Enter the correct Username and Password
3. User directed to Home page (Admin or Player)

Alternative Scenario:

2. User Enter incorrect Username or (and) Password
3. Not directed to other page while a message Appear mentioning "invalid"

Identifier:

UC-2: ADMIN-PAGE – Add New Game

Actor : Administrator

Precondition : Administrator Login correctly, Database Connected

Nominal Scenario :

1. Admin Seeing the Home Page with Navigation Menu On the Left side
2. Press the Menu : "Add Game"
3. Admin input the data correctly
4. Admin Press the button save new game

5. New Game Added to the database

Alternative Scenario A:

3. Admin enter the same Name of the game
4. Click the button save
5. The Data is not saved, an error message thrown

Alternative Scenario B:

3. Admin don't fill all the field
4. click the button "save new game"
5. The Data is not saved, an error message thrown

Identifier:

UC-3: ADMIN-PAGE – ADD A CONSOLE

Actor : Administrator

Precondition : Administrator Login correctly, Database Connected

Nominal Scenario :

1. Admin navigated to Main Page
2. Click on "Add a Console"
3. A little form to add console appear
4. Admin input the name of new console
5. Click on the button save
6. New console added to Database

Alternative Scenario:

4. Admin enter the console name that already existed/ leave it empty
5. Click on the save button
6. No new console added to database, The Error message thrown

Identifier:

UC-4: ADMIN-PAGE – ADD A VERSION

Actor : Administrator

Precondition : Administrator Login correctly, Database Connected, List of console in database is not empty

Nominal Scenario :

1. Admin navigated to Main Page

2. Click on Add Version
3. A form to add new version appear
4. Fill in the form
5. Click on save button

Alternative Scenario:

4. Admin didn't fill the form correctly
5. Click on save button
6. The Version not added to database, an error message thrown

Identifier:

UC-5: ADMIN-PAGE – MODIFY CREDIT OF A GAME

Actor : Administrator

Precondition : Administrator Login correctly, Database Connected, List of console, Version and Games are not empty

Nominal Scenario :

1. Admin navigated to Main Page
2. Click on Update Credits
3. A little form appear
4. Admin choose the game and input the new credit value
5. Click on the button update credit
6. The credit updated in the database

Alternative Scenario:

4. Admin didn't fill the form correctly
5. Click on save button
6. The credit is not modified, an error message thrown

Identifier:

UC-6: ADMIN-PAGE – DELETE A GAME

Actor : Administrator

Precondition : Administrator Login correctly, Database Connected, List of console, Version and Games are not empty

Nominal Scenario :

1. Admin navigated to Main Page
2. Click on Delete A Game

3. There will be a list of games with no registered copy
4. Admin choose a game from the list to be deleted
5. Click on the button Delete
6. The Chosen Game deleted from the Database

Alternative Scenario A:

3. The list is empty, so Admin can not delete any game

Alternative Scenario B:

4. Admin Didn't chose any game
5. Click on the button Delete
6. An Error message Appear, no game deleted in database.

Identifier:

UC-7: ADMIN-PAGE – SIGN OUT

Actor : Administrator

Precondition : Administrator Login correctly, Database Connected, List of console, Version and Games are not empty

Nominal Scenario :

1. Admin navigated to Main Page
2. Click on the button Sign out
3. The Saved data of current Admin cut off
4. Directed to Login Page

Identifier:

UC-8: PLAYER-PAGE – MY ACCOUNT

Actor : Registered Player

Precondition : A player Login correctly, Database Connected

Nominal Scenario :

1. After Login, The player directed to the Player main window
2. Clickable Navigation Menu on the left of the page proposed
3. Player click on MY Account
4. Directed to MY ACOUNT PAGE
5. The data of the number of credit, number of person renting his game, and number of his game are being rented displayed (read only data).

6. The other information : email, username and password are shown in the same page and editable to allow player modify his data, with the modify buttons.
7. Player change any of these information
8. Click on any of the modify button related to the modified information
9. A textbox appeared confirming the update.
10. On the same page, there is Delete Account Button
11. Player click on Delete Button
12. A Message box appear asking confirmation to be deleted
13. Player confirming
14. The Player account removed with the messagebox appear confirming delete.

Alternative scenario:

1. Player click the delete button (he has some games that are rented or he's booking a game or borrowing a game;
2. Error Message in Message Box mentioning that he might have a copyrented or own a game that currently borrowed.

Identifier:

UC-9: PLAYER-PAGE – CATALOG

Actor : Registered Player

Precondition : A player Login correctly, Database Connected, List of Consoles, Versions, and Games are existed in the database

Nominal Scenario :

1. After Login, The player directed to the Player main window
2. Player click on Catalog
3. There will be all copies of games displayed in the catalog
4. If needed, player can type the specific game's name he/she's looking for in the search box
5. Or (and) Player can choose also the Console type from the dropdown list
6. (Or then) go even more specific by choose the Version-of-chosen-Console from a dropdown list
7. Click on the Search button Or Click on Reset Button to reset to default display list which is the full list of all the copies
8. The related game displayed in case of clicking on the Search button, or the full list of copies displayed on case of clicking on the Reset button
9. Click on one of the game
10. Click on one of the button: I own this game

11. A textbox appear asking the confirmation if the player own this game
12. Player click on Yes
13. The Game then added to the list and available to be rented/borrowed
14. The Player choosing one of the game
15. Click on the button Rent Me
16. A Message Box appear asking for how long he would like to rent: with 1 week as the default duration
17. Player choose the number of week from dropdown list
18. There is to Button Option to go forward either Rent Button or Cancel Button
19. Player click on Rent button
20. Another Message Box Appear as the confirmation Of renting

Alternative Scenario A:

9. The Player click on the Game that existed in the list of the games that he own
10. Player click on the button : I own this game
11. The Message Box appear mention that he/she already own this game
12. The list of owning the game is not added nor updated

Alternative Scenario B:

9. The Player choosing one of the game that he's already(currently) renting
10. Click on the button Rent Me
11. A Message Box appearing, mentioned that he/she's currently renting this game
12. There is no update done to his/her renting list of games

Alternative Scenario C:

9. The Player choosing one of the game while his credit is on minus
10. Warning Message mentioning that he's not able to rent a game displayed on the page
11. The button Rent me is inactivated

Identifier:

UC-10: PLAYER-PAGE – MY GAMES

Actor : Registered Player

Precondition : A player Login correctly, Database Connected, List of Consoles, Versions, and Games are existed in the database

Nominal Scenario :

1. After Login, The player directed to the Player main window

2. Player click on My Games
3. My Games Page displayed with 3 main menus
4. Player click on first menu: "Rented games"
5. There is a dropdown list to choose Ongoing / finished (Ongoing as default)
6. Player choose Ongoing
7. The list of ongoing rented games displayed
8. Player chose on one of the game
9. Click the button Give Game Back
10. A message Box confirming that he gave back the game and informing the balance of his credit
11. Player click on second menu : My Bookings
12. List of his bookings are displayed
13. There is a dropdown list to choose three different status: Waiting, Cancel, and Booked (Waiting as default)
14. The player choose the Waiting status
15. Click on one of the list
16. Click on Cancel Booking
17. A Message Box appeared confirming the cancelation
18. The game is removed from the list in the waiting status, and the List of Canceled status added.
19. Player click on Third Menu: My Games
20. The list of the games displayed
21. Player click on one of the game
22. Player click on Delete Button
23. A Message appear for the confirmation that the game is deleted.
24. Game is deleted from the list of game this player own

Alternative Scenario:

21. Player click on one of the game (this game is borrowed by another player)
22. Player click on Delete Button
23. A Message displayed mention that the game cannot be deleted, and suggestion to contact the administrator.
24. Nothing change in the list of the games this player own.



Identifier:

UC-11: PLAYER-PAGE – SIGN OUT

Actor : Registered Player

Precondition : A player Login correctly

Nominal Scenario :

1. After Login, The player directed to the Player main window
2. Player click on Sign Out button
3. A message box mentioned that the User session will be closed and directed to Login Page.

## Diagramme de classes

