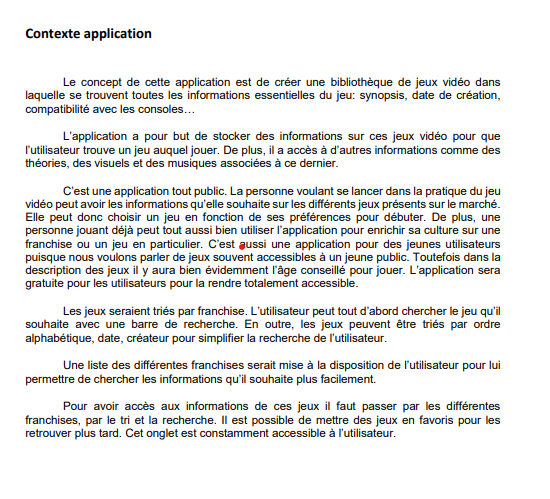
**Preuves**

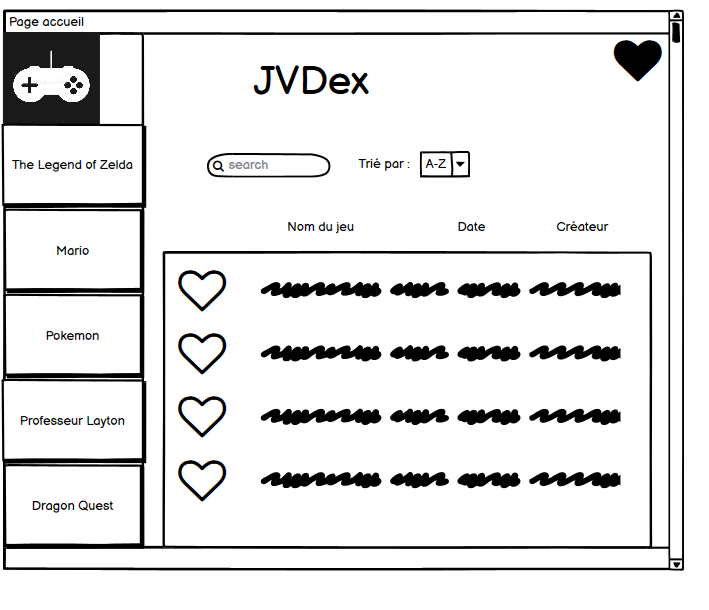
**IHM : Interface Homme-Machine (XAML, WPF)**

***Je sais décrire le contexte de mon application pour qu’il soit compréhensible par tout le monde.***

******

Voici l’explication du contexte de l’application, dans le fichier *documentation\_appliPartie1.pdf*

***Je sais dessiner des sketchs pour concevoir les fenêtres de mon application.***

******

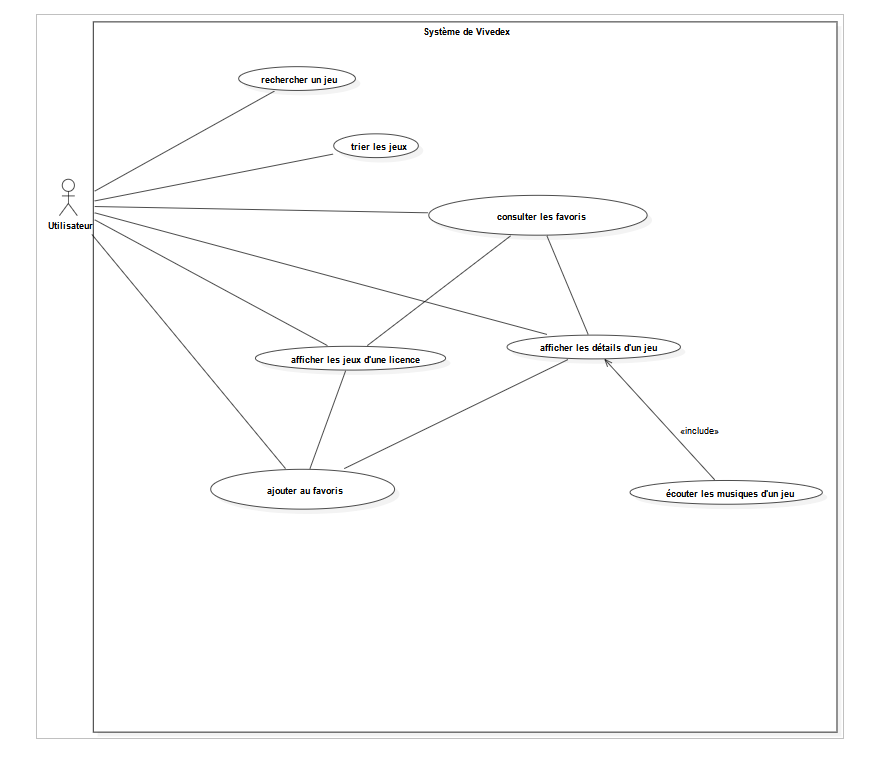
Exemple d’un des sketchs sur notre vue principale.

***Je sais enchaîner mes sketchs au sein d’un story-board.***

Nous avons fait un storyboard expliqué pour montrer l’enchaînement possible des différents sketcks. Cela se trouve dans *documentation\_appliPartie1.pdf*



***Je sais concevoir un diagramme de cas d’utilisation qui représente les fonctionnalités de mon application.***

******

***Je sais concevoir un diagramme de classes qui représente mon application.***

Le diagramme est dans le fichier descriptionDiagramme2.pdf

***Je sais réaliser un diagramme de paquetages qui illustre bien l’isolation entre les parties de mon application.***

Le diagramme est dans le fichier descriptionDiagramme2.pdf

***Je sais décrire mes deux diagrammes en mettant en valeur et en justifiant les éléments essentiels.***

Le diagramme est dans le fichier descriptionDiagramme2.pdf

***Je sais concevoir une application ergonomique.***

Le diagramme est dans le fichier descriptionDiagramme2.pdf

***Je sais concevoir une application avec une prise en compte de l’accessibilité.***

Le diagramme est dans le fichier descriptionDiagramme2.pdf

***Je sais mettre en avant dans mon diagramme de classes la persistance de mon application.***

Le diagramme est dans le fichier descriptionPersistance.pdf

***Je sais mettre en avant dans mon diagramme de classes ma partie personnelle.***

Le diagramme est dans le fichier descriptionPersistance.pdf

***Je sais mettre en avant dans mon diagramme de paquetages la persistance de mon application.***

Le diagramme est dans le fichier descriptionPersistance.pdf

**Conception et Programmation Orientées Objets (C#, .NET)**

***Je maîtrise les bases de la Programmation C# (classes, structures, instances…) :***

Nous avons créé de nombreuses classes dans plusieurs projets différents, par exemple *InformationsJeu* que l’on utilise énormément dans notre programme :

Une image contenant texte

Description générée automatiquement

On instancie un *InformationsJeu* dans la classe *Jeu*(ligne 26):

Une image contenant texte

Description générée automatiquement

Nous avons aussi créé des Enum, par exemple *Plateformes*:

Une image contenant texte

Description générée automatiquement

***Je sais utiliser l’abstraction à bon escient (héritage, interfaces, polymorphisme) :***

Pour la persistance, de l’application, nous devions sérialiser la liste de jeux et le dictionnaire de franchise contenues dans *StockApp*. Pour cela, pour avons créé l’interface *IPersistanceStockApp*. Ainsi toutes les classes l’implémentant pourrais être utilisé comme « stockage » de données.

Une image contenant texte, intérieur, capture d’écran

Description générée automatiquement

Pour faire les tests, nous avons donc utiliser la classe *Stub*, qui implémente *IPersistanceStockApp*. La méthode *ChargeDonnees()* possédant des données utilisées seulement pour les tests :

Une image contenant texte

Description générée automatiquement

Mais pour le déploiement, nous avions besoin d’une persistance dans un fichier, c’est pourquoi nous avons fait la classe *PersistanceFichierTexte*, qui comme son nom l’indique, utilise un fichier texte pour charger et sauvegarder les données :

Une image contenant texte

Description générée automatiquement

*(ici on ne voit que la méthode ChargeDonnees(), mais il y en d’autres, cf Data/PersistanceFichierTexte)*

Ainsi, dans app, on peut changer facilement de « mode » de persistance (on pourrait en rajouter d’autre dans le futur : xml, bases de données…) tout en ne changeant pas le type statique de la propriété *Pers* d’*App.* Il suffit d’instancier la classe de persistance que l’on souhaite :

Une image contenant texte, moniteur, écran, noir

Description générée automatiquement

***Je sais gérer des collections simples ( tableaux, listes…) :***

Dans la classe *Jeu*, on a utilisé des collections très simples pour stocker les *Theories, les Musiques et les Visuels*. Nous avons préféré utiliser des *Set* car on ne veut pas de doublons et on n’a pas besoins qu’elles soient triées :

Une image contenant texte

Description générée automatiquement

***Je sais gérer des collections avancées(dictionnaires) :***

Pour stocker les *Franchise*, nous avons utilisé un dictionnaire. Chaque clé est donc une *Franchise* et chaque valeur associée est une *List<Jeu>*, qui sont donc les jeux d’une franchise.



***Je sais contrôler l’encapsulation au sein de mon application :***

Pour encapsuler nos données, nous avons utilisé des Propriété, qui sont donc publiques, d’où on peut récupérer les valeurs stockées dans les champs privés associé.

Exemple : Propriété publique *TousLesJeux* => Champ privé *tousLesJeux*

Une image contenant texte

Description générée automatiquement

***Je sais tester mon application :***

Nous avons un projet entier consacré aux tests de notre application, *ConsoleTest.* Cependant nous n’avons fait que des tests fonctionnels :

Une image contenant texte

Description générée automatiquement

*(ici il n’y en a que 3, pour les autres : cf ConsoleTest/Program)*

***Je sais utiliser LINQ :***

Nous avons, dans notre *Manager*, la liste de tous les jeux disponibles dans l’application. Cette liste est reliée à une propriété calculée qui permet d’effectuer la recherche de jeux dans la liste, ainsi que le tri. Pour faire cela on utilise LINQ, ce qui facilite grandement les choses :

Une image contenant texte

Description générée automatiquement

***Je sais gérer les évènements :***

Nous avons utilisé l’évènement *PropertyChanged* pour notifier lorsqu’une propriété avait changé, surtout dans le *Manager :*

Une image contenant texte

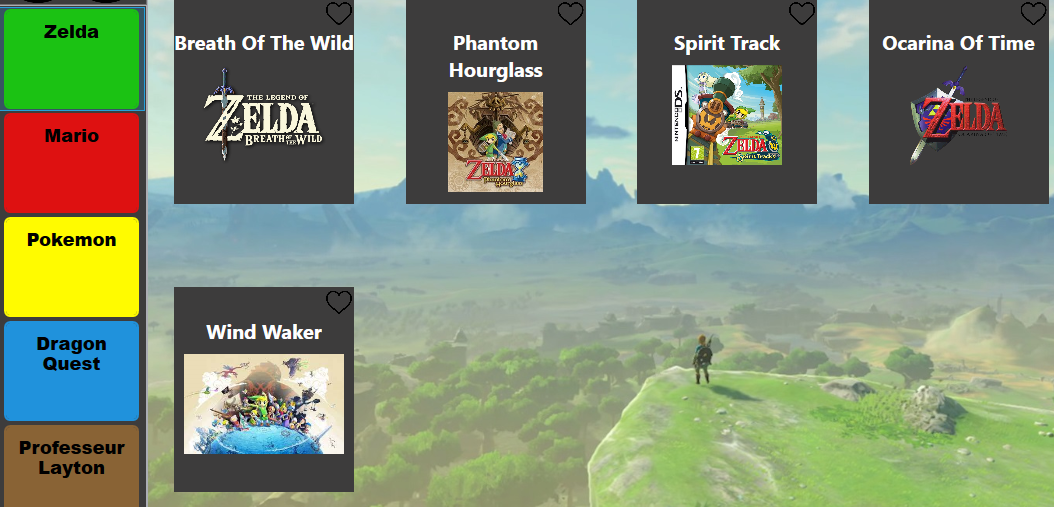
Description générée automatiquement

***Je sais choisir mes layouts à bon escient :***

Nous avons dû choisir nos layouts précisément pour que notre application ressemble à ce que nous avions prévu. C’est pourquoi, dans notre fenêtre *MainWindow* nous avons une *Grid.* En effet, nous voulions deux colonnes pour afficher le master (les franchises) à gauche de la fenêtre et le détail à droite (les jeux) :

Une image contenant texte

Description générée automatiquement



***Je sais choisir mes composants à bon escient :***

Nous avons utilisé de nombreux composants dans nos UserControl, TextBox pour la recherche, ComboBox pour le tri…

Une image contenant texte, capture d’écran, écran

Description générée automatiquement

Une image contenant texte, capture d’écran, moniteur, écran

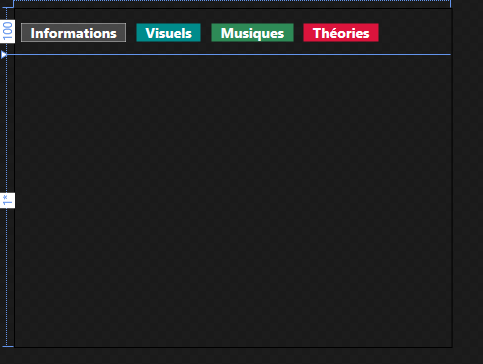
Description générée automatiquement

*(il y en a beaucoup d’autre dans tout le projet ApplicationGraphique)*

***Je sais créer mon propre composant :***

Nous avons dû créer énormément de UserControl comme nous n’avions qu’une vue, les principaux sont *UserControlMain, UserControlFranchise et UserControlVueJeu :*

Une image contenant texte, noir, capture d’écran

Description générée automatiquement

*(Tous les autres se trouvent aussi dans ApplicationGraphique/Nos\_UC/)*

***Je sais intercepter les évènements de la vue :***

Que ce soit pour la navigation entre les fragments, la recherche de jeu, le tri, l’écoute d’une musique, etc., nous avons besoin d’intercepter les évènements de la vue presque partout dans *ApplicationGraphique*, en voici plusieurs exemples :

Une image contenant texte

Description générée automatiquement

*Code behind UserControlMainWindow*

Une image contenant texte

Description générée automatiquement

*Code Behind UserControlMusique4ItemsControlMusique*

Une image contenant texte

Description générée automatiquement

*Code Behind MainWindow*

***Je sais gérer le DataBinding sur mon master :***

Pour afficher les franchises à gauche de l’application, il nous a fallu bind notre *ListBoxFranchises* au dictionnaire *ToutesLesFranchises* du *Manager*. Pour récupérer ses franchises on se bind donc sur ToutesLesFranchises.Keys (étant la liste des clés du dictionnaire) :

Une image contenant texte

Description générée automatiquement

*Ce qui donne ça avec nos données :*

Une image contenant texte

Description générée automatiquement

*(Voir aussi le binding de ApplicationGraphique/UserControlFranchise)*

***Je sais gérer le DataBinding sur mon detail :***

Le detail de notre application est l’affichage du jeu sélectionné. Ainsi la vue d’un *Jeu* est bind sur *JeuSelected* de *Manager*. Cependant, cette vue est séparée en onglets, que sont *UserControlInformations, UserControlVisuels, UserControlMusiques et UserControlTheories.*

Exemple : *UserControlInformations*

Une image contenant texte

Description générée automatiquement

*Design avec un jeu :*

Une image contenant texte

Description générée automatiquement

***Je sais gérer le DataBinding et les Dependency Property sur mes UserControl :***

Le DataBinding sur les *UserControl* a été effectué beaucoup de fois, puisque nous en possédons un grand nombre, cependant, un seul *UserControl* possède des *Dependency Property : UserControlJeu,* qui est utilisé dans *UserControlFranchise* pour afficher chaque *Jeu*.

Ainsi le binding de fais sur les *Dependency Property,* et non sur celles du *Jeu.*

Une image contenant texte, intérieur, capture d’écran, écran

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

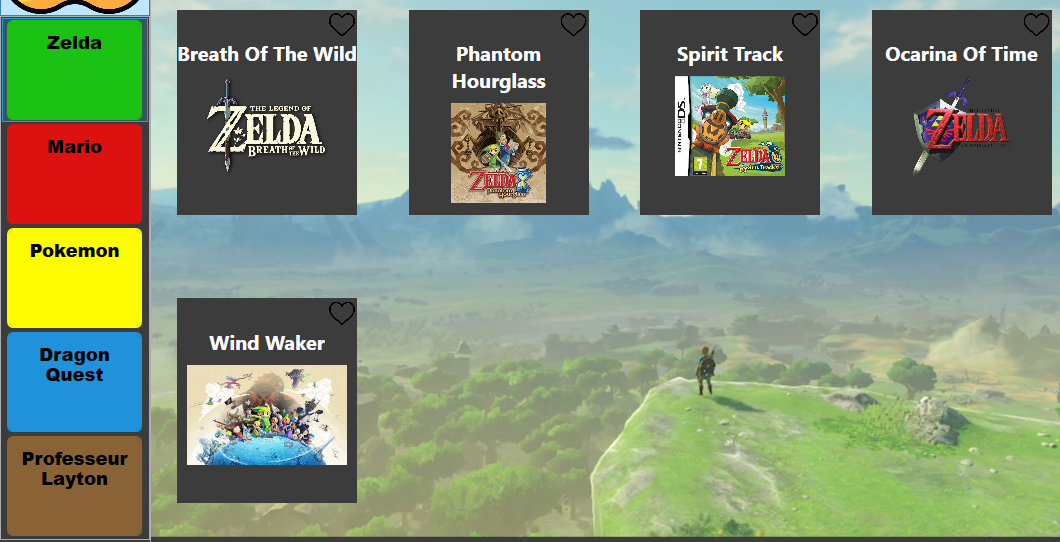
***Je sais développer un Master/Detail :***

Nous avons tout fait dans le but de produire un master détail. Même s’ il y a plusieurs masters, in fine le detail reste le jeu sélectionné. Pour y arriver on peut donc passer par la liste de jeux, grâce, par exemple, à la recherche, ou alors depuis une franchise, ou encore depuis les favoris.

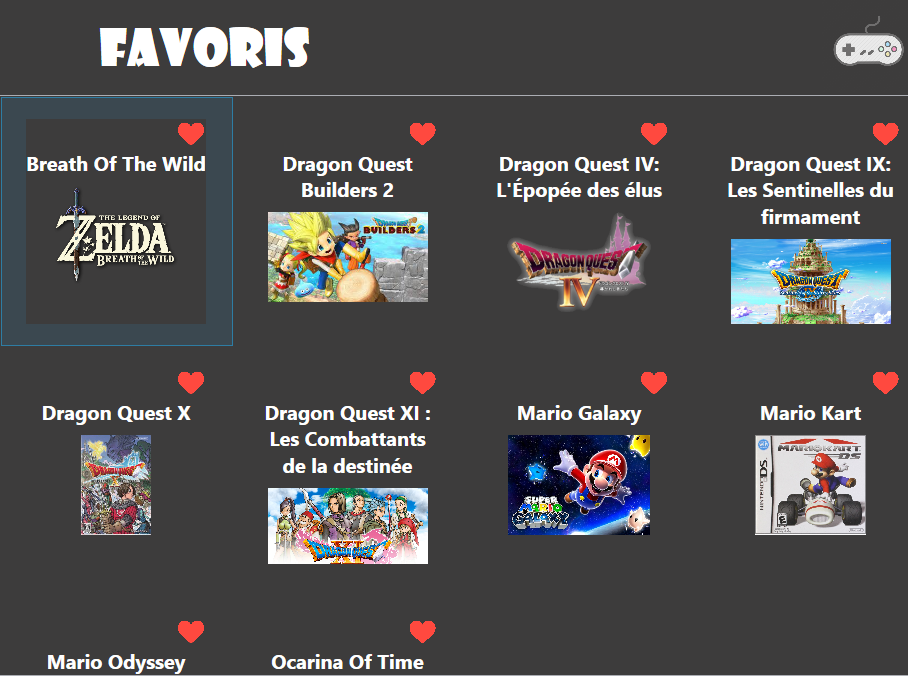
Une image contenant texte

Description générée automatiquement

*Master 1 : Liste des jeux de UserControlMainWindow*



*Master 2 : Franchise sélectionnée (ici => Zelda )*



*Master 3 : Onglet des favoris*

Une image contenant texte

Description générée automatiquement

*Detail : Jeu sélectionné (ici => Ocarina Of Time)*

***Je sais réaliser une vidéo de 1 à 3 minutes qui montre la démo de mon application.***

Notre vidéo se trouve dans le répertoire document.

***Je sais coder la persistance au sein de mon application***

Dans notre application seulement les jeux en favoris sont persistés. Quand l’utilisateur quitte l’application, il retrouve ses favoris. On utilise des méthodes *ChargerDonnees()* et *SauvegardeDonnees().*

***Je sais coder une fonctionnalité qui m’est personnelle***

Nous avons codé un lecteur de musique. En plus de cela nous avons une méthode de tri et de recherche personnelle.

***Je sais coder la persistance au sein de mon application.***

Nous avons codé un lecteur de musique. En plus de cela nous avons une méthode de tri et de recherche personnelle.

***Je sais utiliser SVN.***

******

***Je sais développer une application qui compile***

Notre application compile, nous n’avons aucune erreur.

***Je sais développer une application fonctionnelle***

Notre application compile, nous n’avons aucune erreur.

***Je sais mettre à disposition un outil pour déployer mon application***

***Le fichier est JVDexbin/JVDex\_Setup.msi***

******