

Contexte application

Le concept de cette application est de créer une bibliothèque de jeux vidéo dans laquelle se trouvent toutes les informations essentielles du jeu: synopsis, date de création, compatibilité avec les consoles...

L'application a pour but de stocker des informations sur ces jeux vidéo pour que l'utilisateur trouve un jeu auquel jouer. De plus, il a accès à d'autres informations comme des théories, des visuels et des musiques associées à ce dernier.

C'est une application tout public. La personne voulant se lancer dans la pratique du jeu vidéo peut avoir les informations qu'elle souhaite sur les différents jeux présents sur le marché. Elle peut donc choisir un jeu en fonction de ses préférences pour débiter. De plus, une personne jouant déjà peut tout aussi bien utiliser l'application pour enrichir sa culture sur une franchise ou un jeu en particulier. C'est aussi une application pour des jeunes utilisateurs puisque nous voulons parler de jeux souvent accessibles à un jeune public. Toutefois dans la description des jeux il y aura bien évidemment l'âge conseillé pour jouer. L'application sera gratuite pour les utilisateurs pour la rendre totalement accessible.

Les jeux seraient triés par franchise. L'utilisateur peut tout d'abord chercher le jeu qu'il souhaite avec une barre de recherche. En outre, les jeux peuvent être triés par ordre alphabétique, date, créateur pour simplifier la recherche de l'utilisateur.

Une liste des différentes franchises serait mise à la disposition de l'utilisateur pour lui permettre de chercher les informations qu'il souhaite plus facilement.

Pour avoir accès aux informations de ces jeux il faut passer par les différentes franchises, par le tri et la recherche. Il est possible de mettre des jeux en favoris pour les retrouver plus tard. Cet onglet est constamment accessible à l'utilisateur.

Persona



Intitulé de poste
lycéenne

Âge
Moins de 18 ans

Niveau d'études
Inférieur au baccalauréat

Réseaux sociaux



Secteur d'activité
étude

Ava

Moyen de communication préféré

réseaux sociaux

Outils nécessaires au quotidien

- Logiciel de traitement de texte
- excel

Supérieur hiérarchique

professeurs

Objectifs

avoir son baccalauréat

Besoin

"J'aime beaucoup passer du temps avec mes amis. Dernièrement ils me parlent de jeux vidéos. Je ne joue pas du tout. J'aimerais discuter de ce sujet avec eux pour leur plaisir. Je ne sais pas comment faire pour me renseigner sur le sujet "



Intitulé de poste
technicien

Âge
Entre 35 et 44 ans

Niveau d'études
Licence ou diplôme équivalent

Réseaux sociaux



Secteur d'activité
Technologie

Taille de l'entreprise
501 à 1 000 salariés

George

Moyen de communication préféré

- face à face
- SMS
- mail

Outils nécessaires au quotidien

- Logiciel de planning des salariés
- Logiciel de reporting
- Logiciel de traitement de texte
- Tableaux de bord de données commerciales

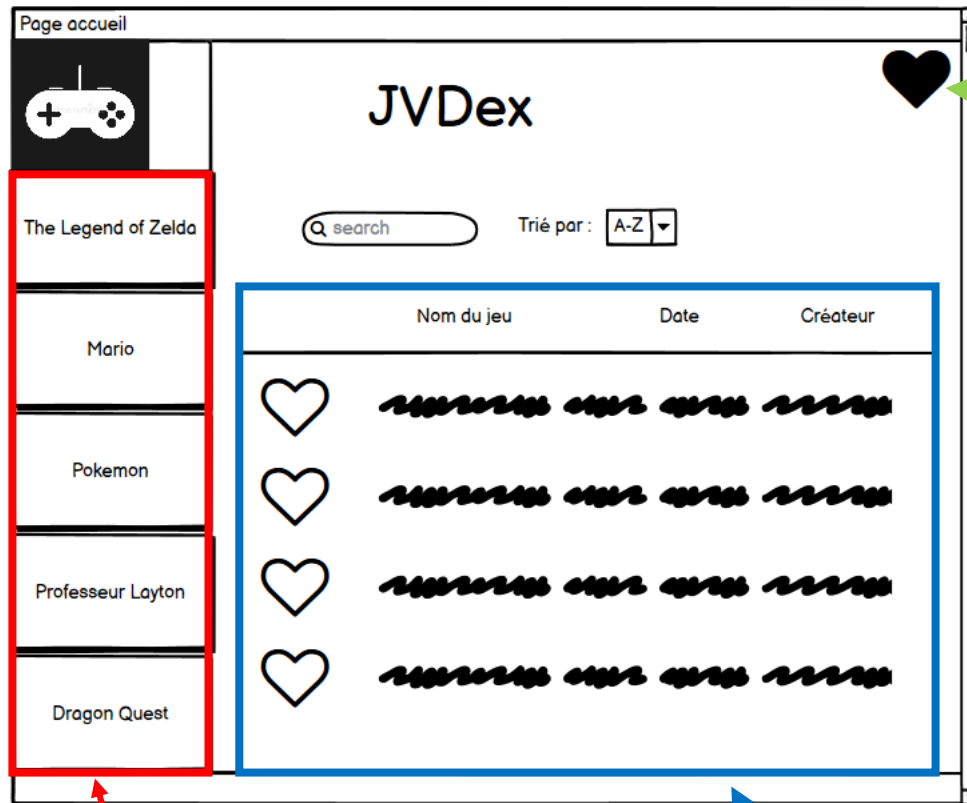
Principaux défis

- Communications et relations avec les clients
- Communication interne
- Gestion des projets et organisation
- Résolution des problèmes et prise de décisions

Saisissez un texte

"Je jouais beaucoup durant mon adolescence. Pris d'une certaine nostalgie, j'aimerais de nouveau jouer aux jeux vidéo. Seulement je ne suis pas trop à jour sur les derniers jeux sortis ces dernières années. J'ai tellement hâte de me replonger dans cet univers!"

Description des sketches



Cette icône donne l'accès à l'onglets des Favoris. Il se trouve sur toutes les vues de l'application. Il suffit à l'utilisateur de cliquer dessus pour avoir accès à ses favoris

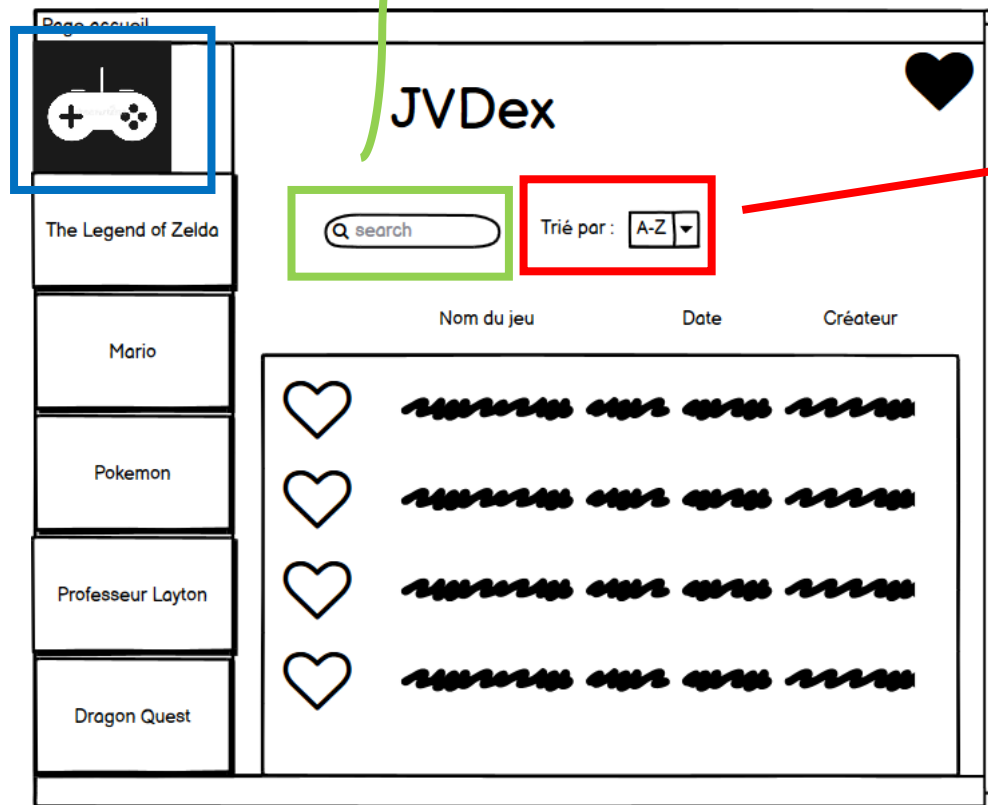
Liste des franchises (boutons) présentes dans le logiciel.

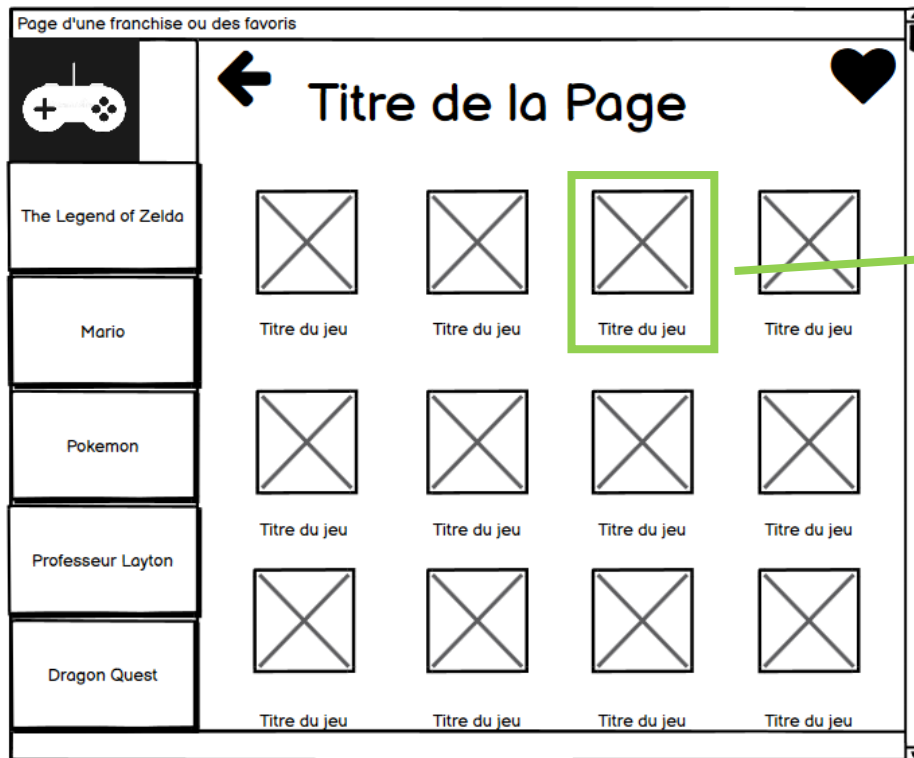
Liste des jeux cliquables triée selon le choix de l'utilisateur.

Bouton qui permet de retourner à la page d'accueil. Il fait aussi office de logo de l'application.

Permet à l'utilisateur de faire une recherche d'un jeu pour avoir ses informations plus rapidement.

Permet à l'utilisateur de trier les jeux selon son critère : par nom de jeux, date de créations, développeurs...





L'utilisateur choisit le jeu dont il veut consulter les informations. Il clique alors sur le jeu en question. Cela le redirige vers la vue des informations du jeu.

La vue des favoris et des franchises sont les mêmes.




Cette icône permet de faire retour à l'onglet d'avant. Il a la même fonctionnalité sur toutes les vues de l'application

Image du jeu

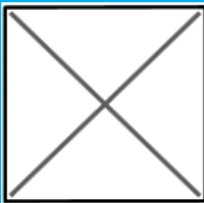
Informations du jeu


Histoire du jeu


Page d'un jeu (informations)




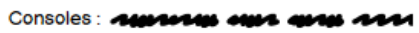
JeuVisuelsMusiqueThéories





Nom : 

Créateur : 

Date de création : 

Consoles : 

Genre : 

Limite d'âge : 

Synopsis

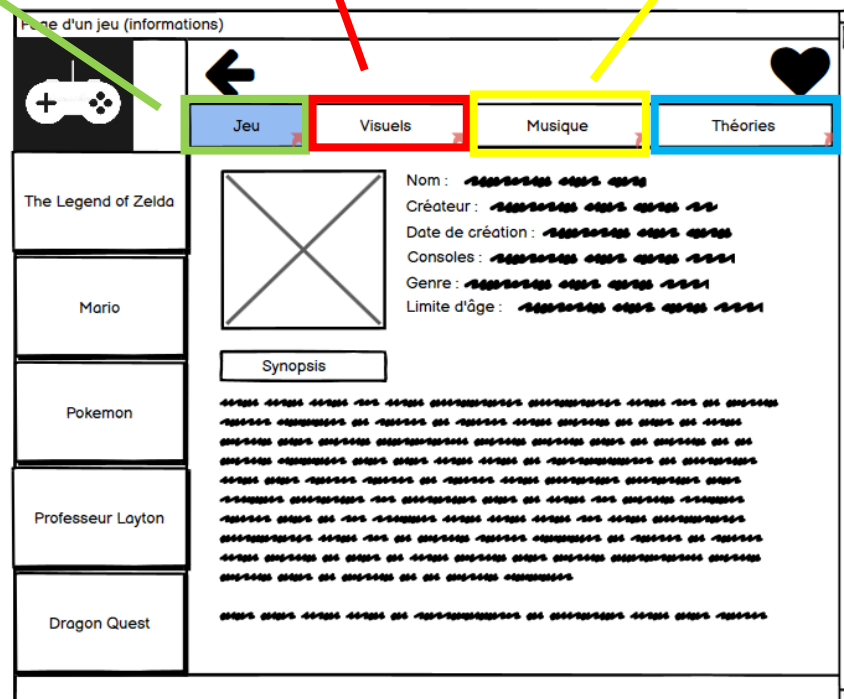


Bouton qui donne accès aux informations du jeu

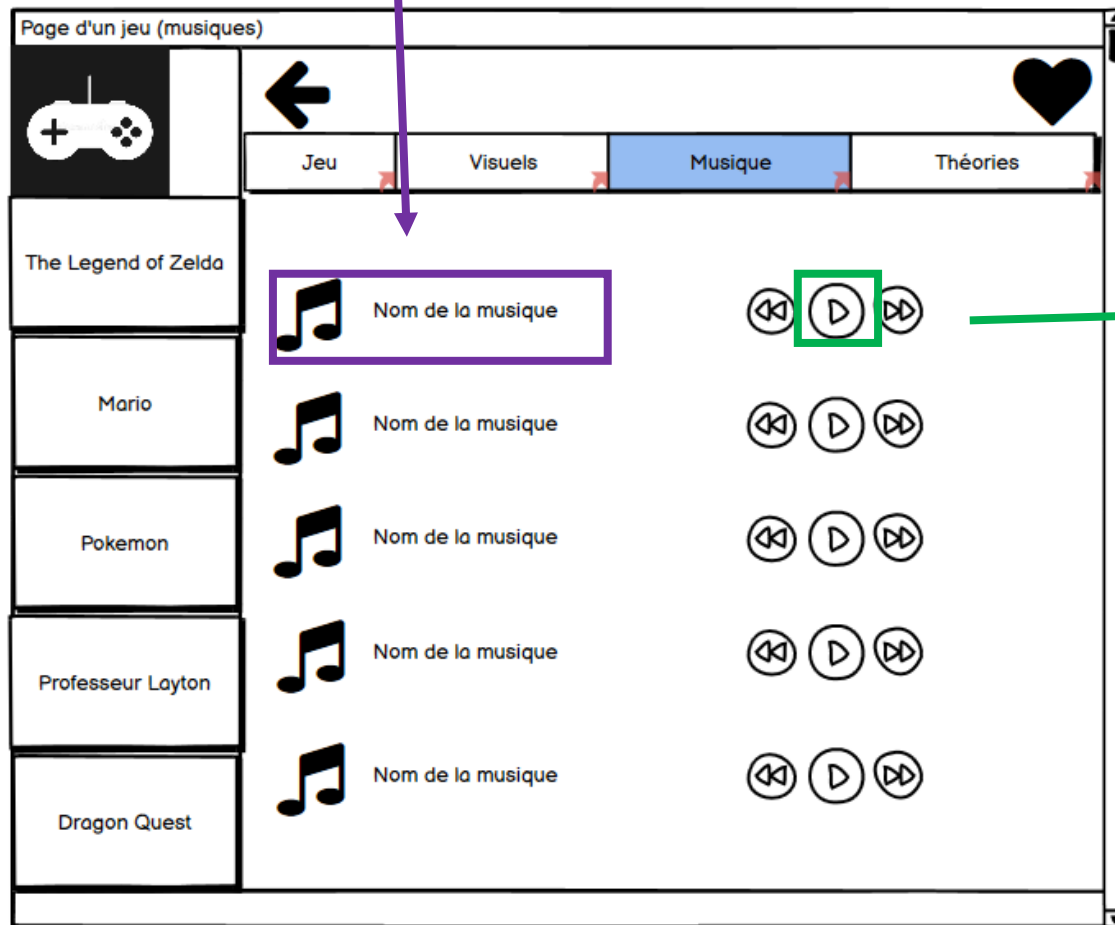
Bouton qui donne accès aux images du jeu

Bouton qui donne accès aux musiques du jeu

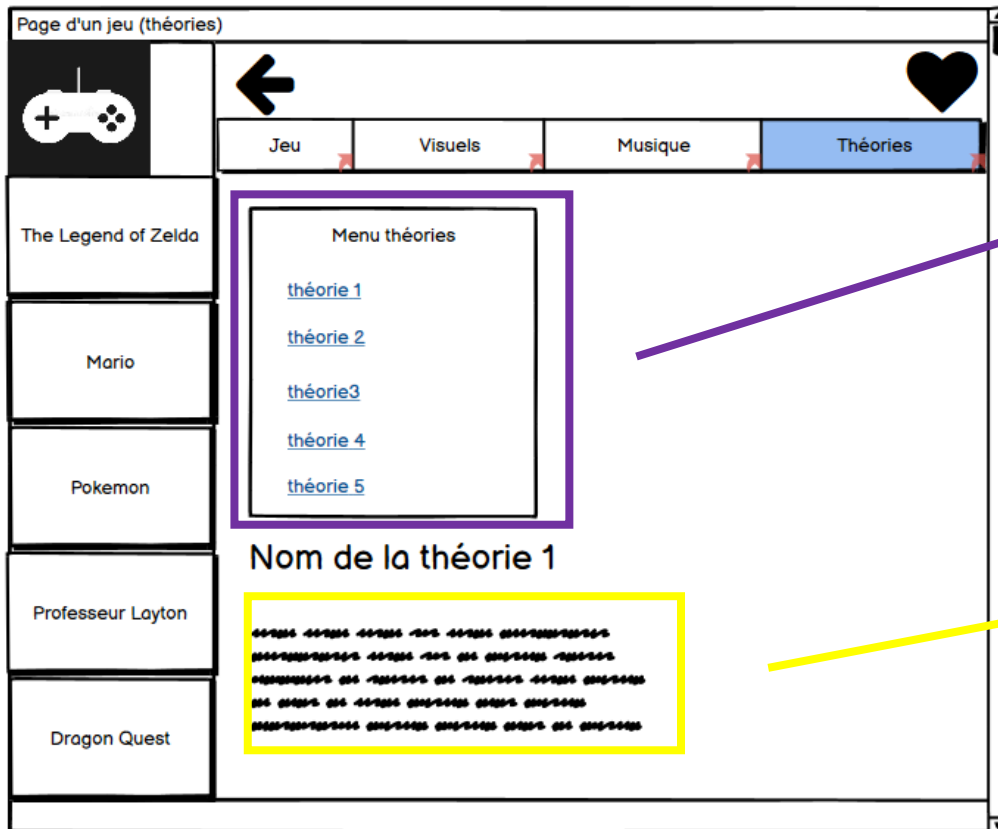
Bouton qui donne accès aux théories du jeu



Nom de la musique et logo



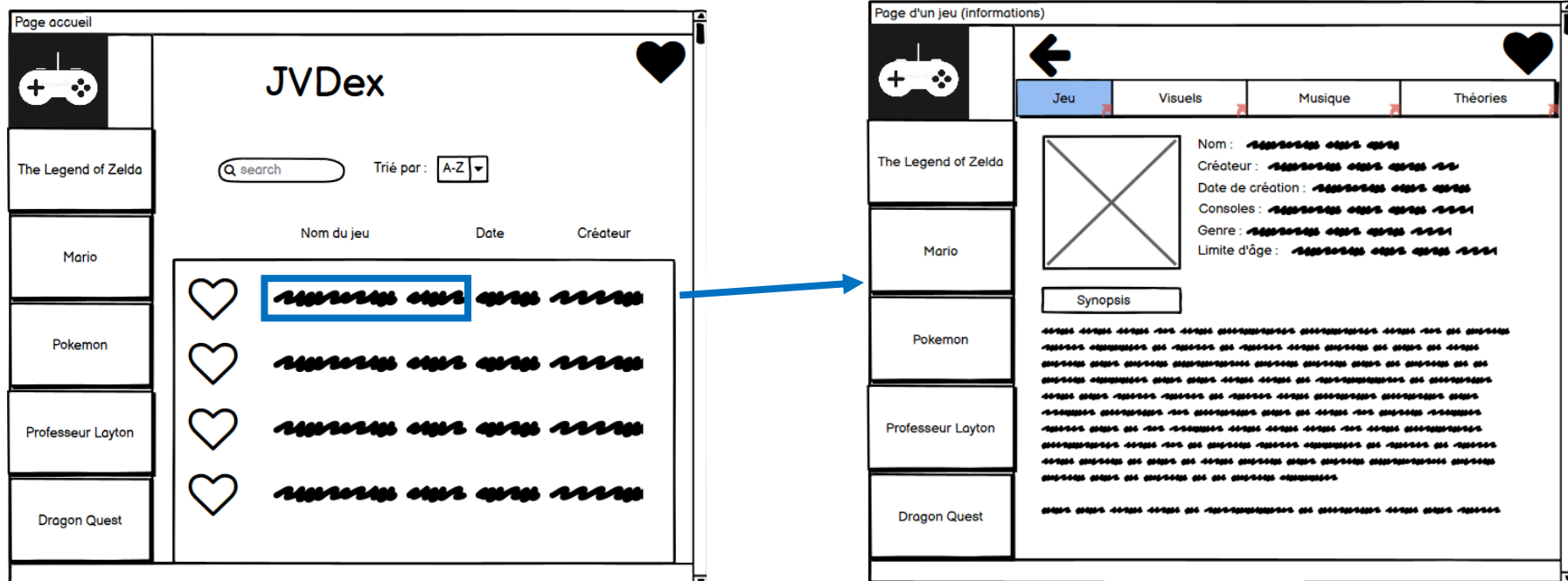
Ce bouton permet
de lancer l'écoute
de la musique



Le menu des théories permet à l'utilisateur d'avoir la liste des différentes théories. De plus il est possible de cliquer sur l'un des noms ce qui nous emmène directement sur la théorie choisie.

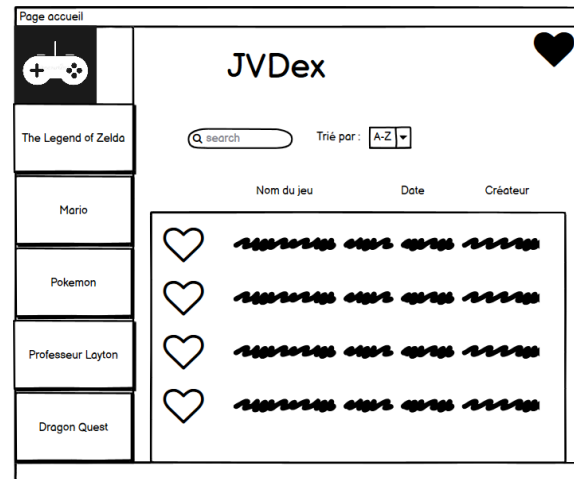
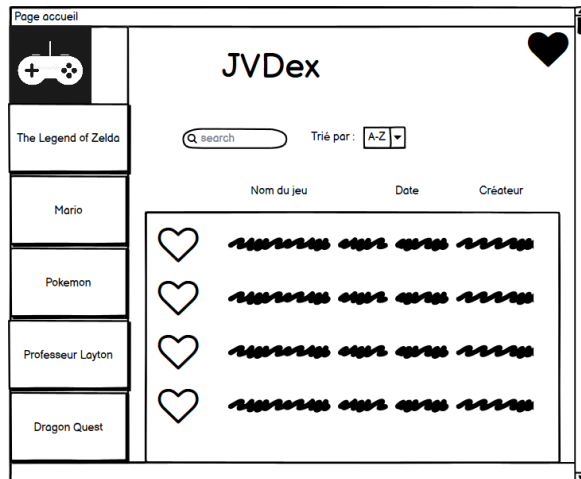
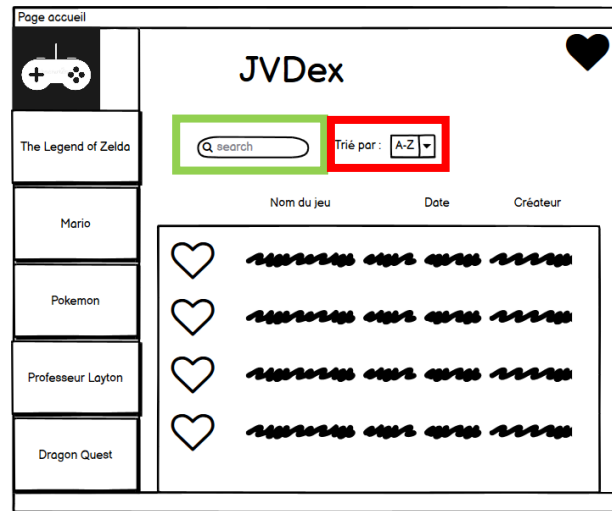
Explication de la théorie

Storyboard

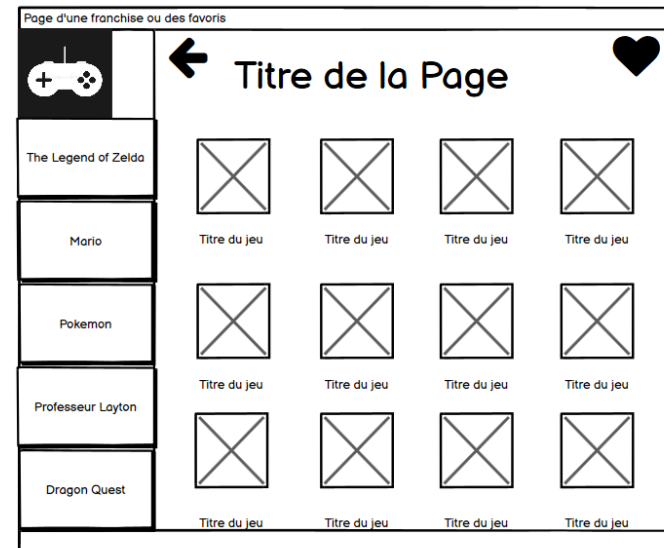
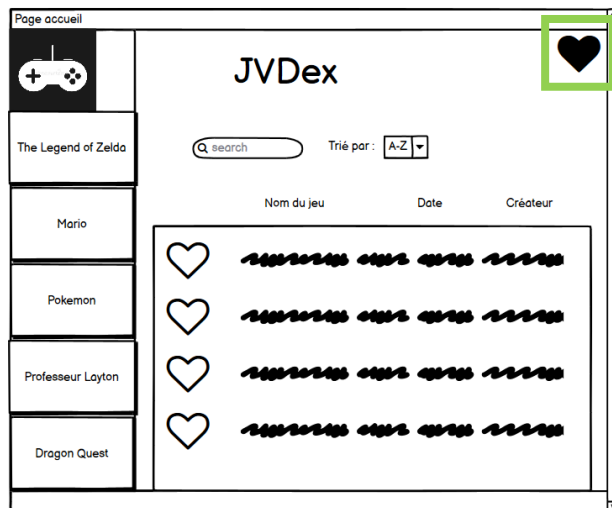


Lorsque l'on clique sur le nom du jeu, on accède aux informations détaillées de ce dernier.

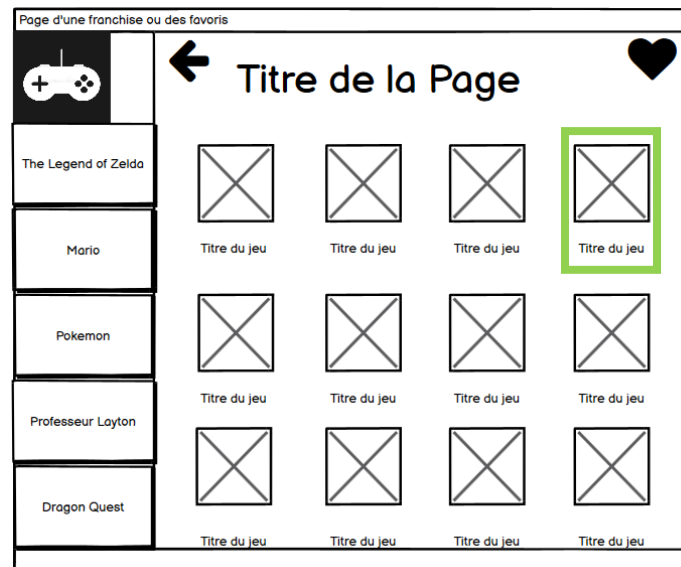
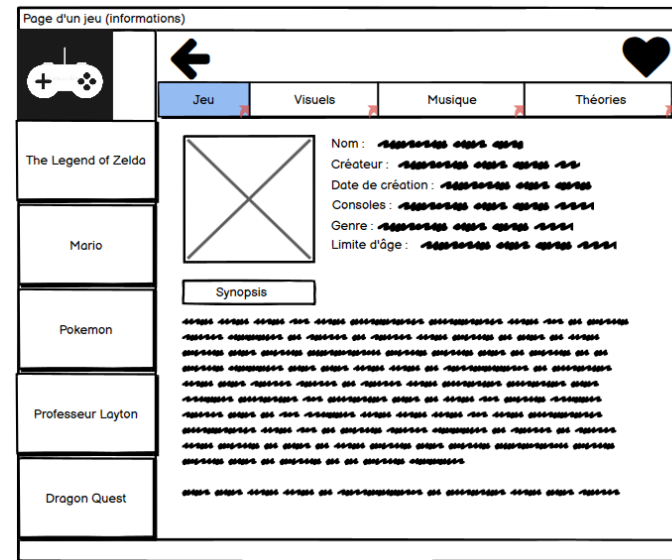
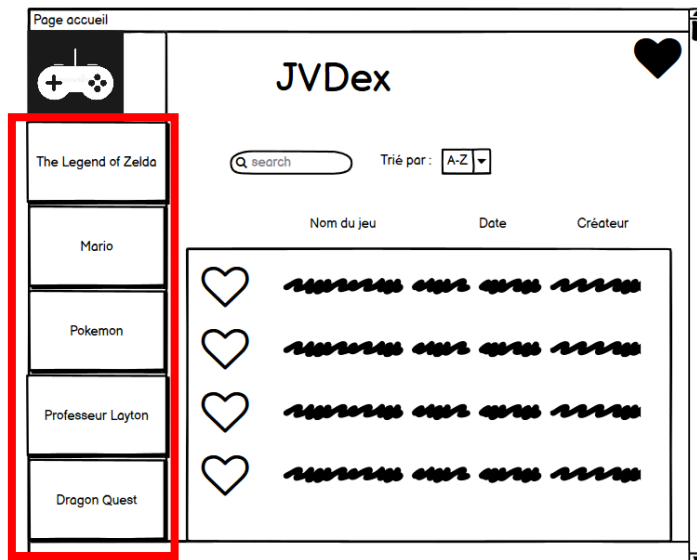
Une recherche peut aboutir à une liste de plusieurs jeux puisque c'est possible de chercher avec le nom du jeu, un créateur spécifique par exemple.



Un tri conduit à une vue comme celle-ci avec les jeux triés.

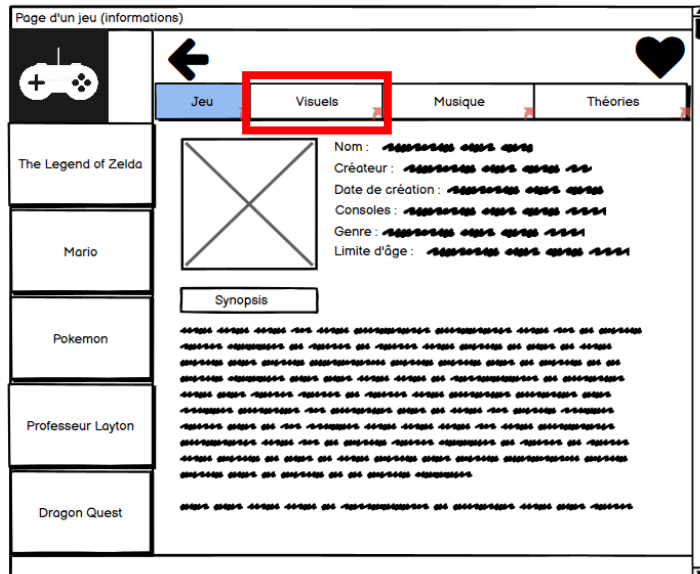
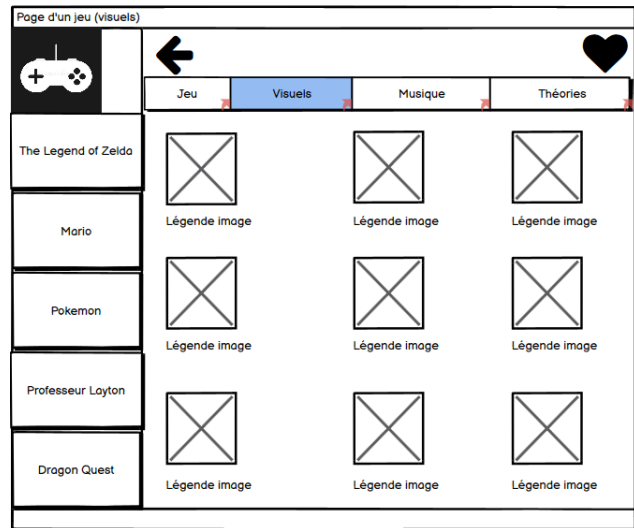


Cliquer sur l'icône des Favoris conduit l'utilisateur vers une vue avec la liste de ses favoris.



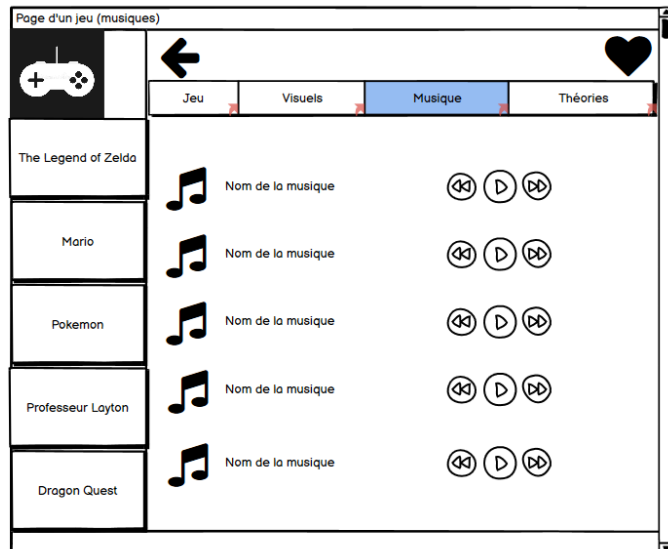
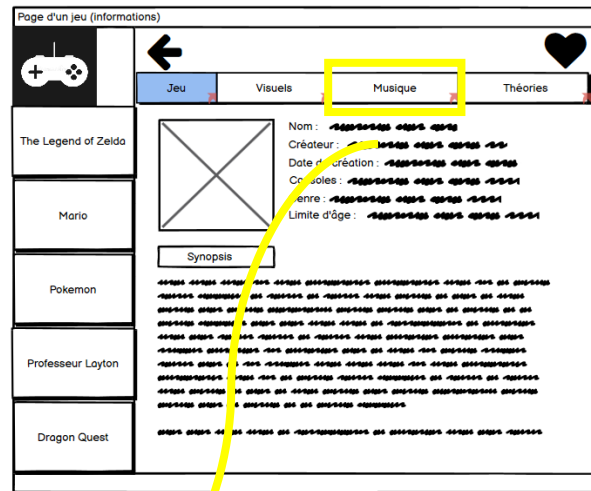
Pour avoir accès aux informations d'un jeu on peut aussi passer par les franchises. L'utilisateur clique sur l'une des 5 franchises proposées. Il est alors sur une nouvelle vue avec la liste des différents jeux appartenant à cette franchise.

L'utilisateur choisit le jeu dont il veut consulter les informations. Il clique alors sur le jeu en question. Cela le redirige vers la vue des informations du jeu.

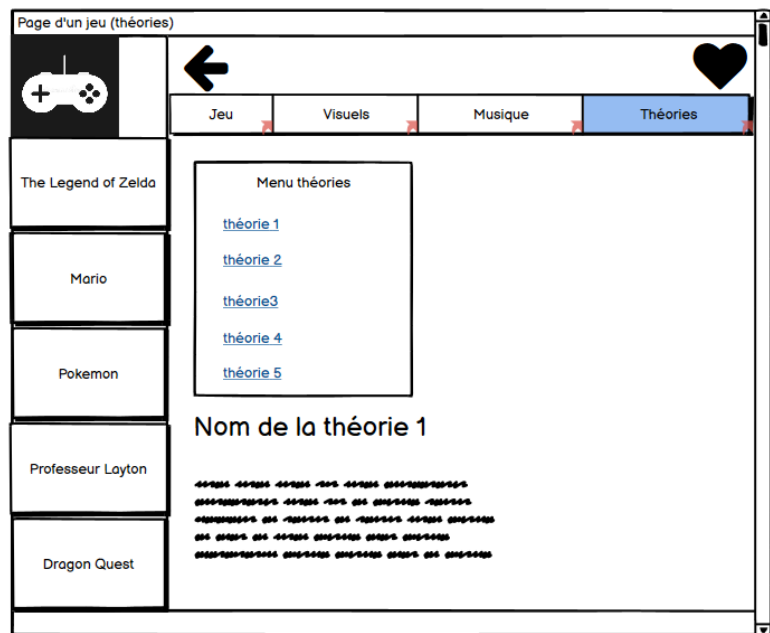
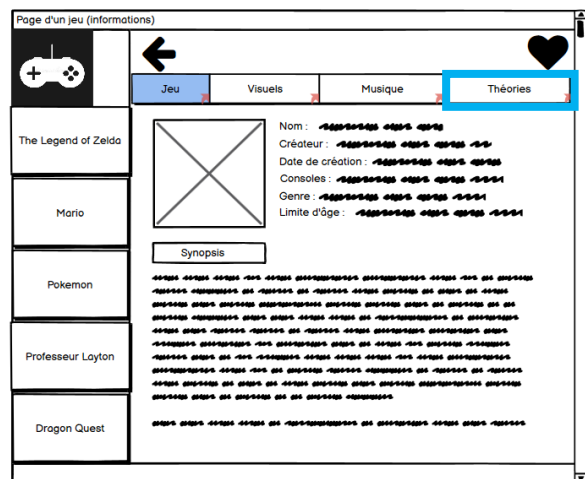


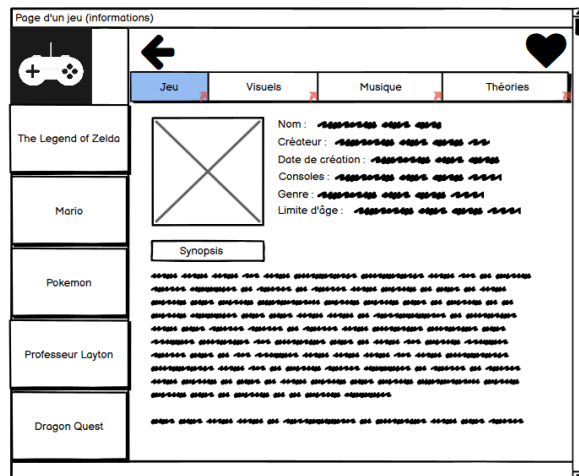
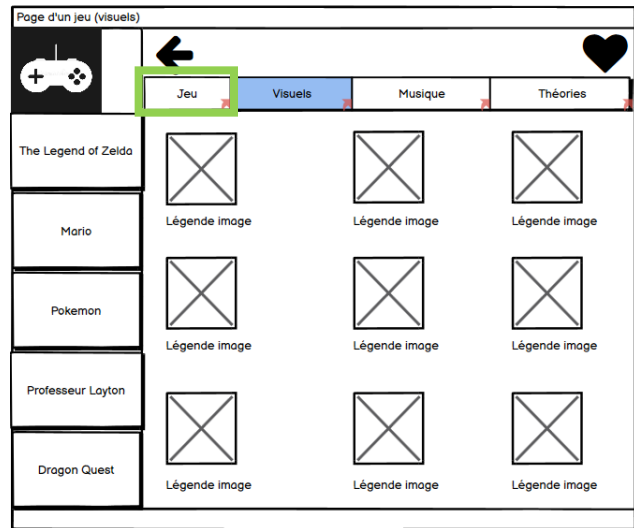
Cliquer sur le bouton
« Visuels » donne accès à la
galerie d'images du jeu

Cliquer sur le bouton
« Musique » donne accès à
toutes les musiques du jeu mis à
disposition dans l'application.



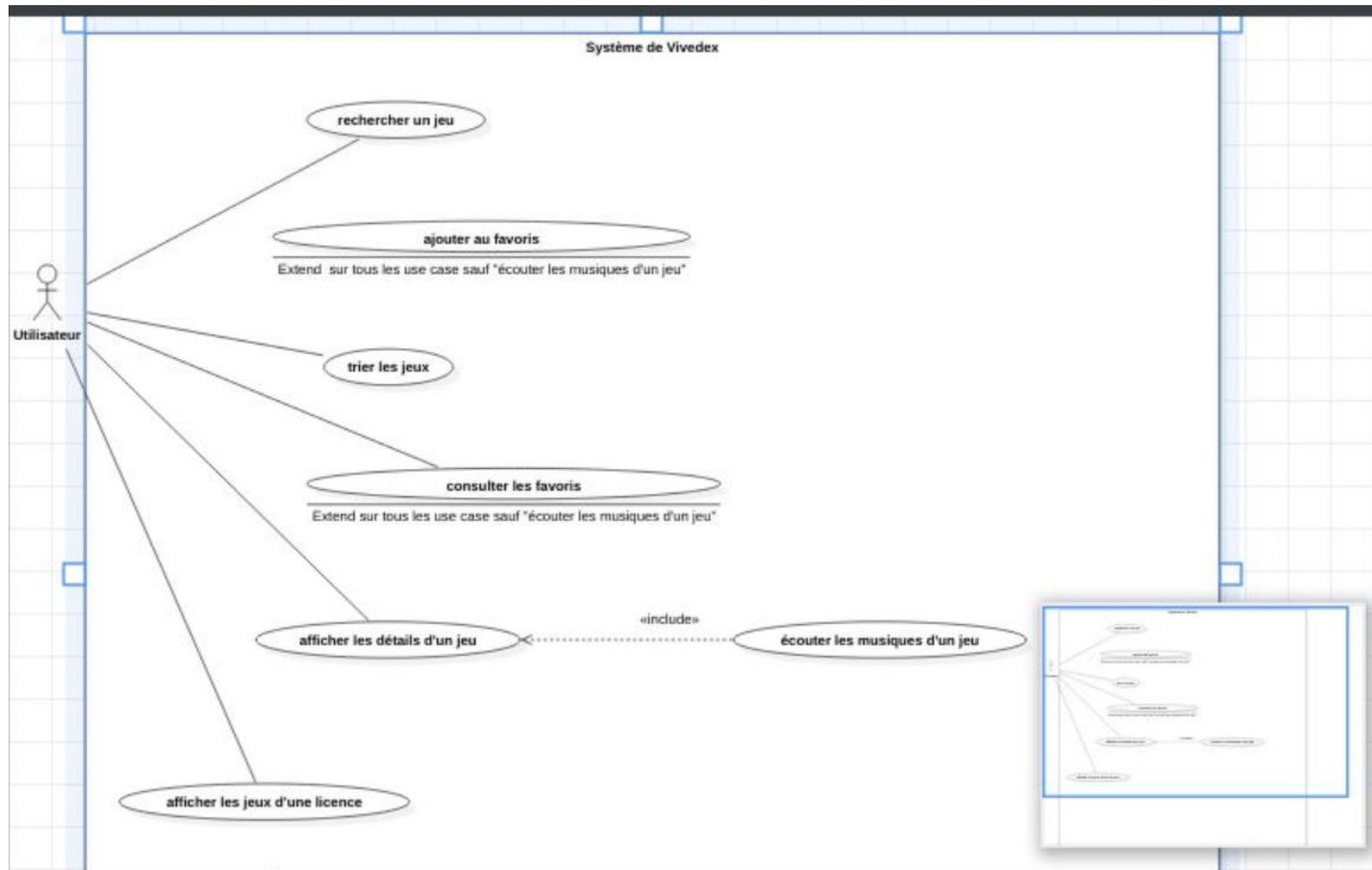
Cliquer sur le bouton
« Théorie » permet à
l'utilisateur de consulter
les différentes théories du
jeu.





Cliquer sur le bouton « Jeu » donne accès aux informations élémentaires du jeu

Diagramme de cas d'utilisation



Description du diagramme de cas d'utilisation

Nom: rechercher un jeu

Objectif: pouvoir rechercher un jeu spécifique

Acteurs: utilisateurs

Cas inclus: /

Conditions initiales: l'utilisateur doit ouvrir l'application

Scénario nominal: 1: écrire le nom du jeu que l'utilisateur recherche

1.1: le jeu est existant

1.2: affichage du jeu

Scénario alternatif: 2: le jeu rechercher n'est pas existant dans l'application

2.1: affichage message d'erreur

2.2: retour à la page principale

Condition de fin: 1) la recherche aboutit alors affichage jeu

2) retour page principal

Nom: trier les jeux

Objectif: trier les jeux pour les rechercher plus facilement et répondre aux critères des utilisateurs

Acteurs: utilisateurs

Cas inclus: /

Conditions initiales: être sur la page d'accueil

Scénario nominal: 1: sélectionner le type de tri voulu

1.1: affichage des jeux en fonction du tri

Scénario alternatif :

Condition de fin: 1) affichage des jeux trié

Nom: afficher le détail d'un jeu

Objectif: afficher les informations d'un jeu

Acteurs: utilisateurs

Cas inclus: /

Conditions initiales: ouvrir l'application

Scénario nominal:

1: sélectionner le jeu voulu

1.1: possibilité d'aller sur différents onglets:

-informations

-visuels

-musique

-théories

Scénario alternatif:

2: faire une recherche

2.1: sélectionner le jeu voulu

2.2: possibilité d'aller sur différents onglets:

-informations

-visuels

-musique

-théories

3: faire un tri

3.1: sélectionner le jeu voulu

3.2: possibilité d'aller sur différents onglets:

-informations

-visuels

-musique

-théories

4: sélectionner une franchise de jeu

4.1: sélectionner le jeu voulu

4.2: possibilité d'aller sur différents onglets: -informations

-visuels

-musique

-théories

5) aller dans les favoris

5.1: sélectionner le jeu voulu

5.2: possibilité d'aller sur différents onglets:

-informations

-visuels

-musique

-théories

Condition de fin:

1) affichage des informations

2) affichage des informations

- 3) affichage des informations
- 4) affichage des informations
- 5) affichage des informations

Nom: ajouter aux favoris

Objectif: mettre certains jeux en favoris pour que l'utilisateur puisse avoir accès à l'information de ses jeux préférés plus rapidement

Acteurs: utilisateur

Cas inclus: /

Conditions initiales: ouvrir l'application

Scénario nominal: 1: sélectionner un jeu
1.1: cliquer sur le petit icône "favoris" en forme de cœur

Scénario alternatif: 2: ouvrir le détail d'un jeu
2.1: cliquer sur le petit icône "favoris" en forme de cœur
3: appuyer sur une franchise
3.1: cliquer sur le petit icône "favoris" en forme de cœur
4: faire une recherche
4.1: cliquer sur le petit icône "favoris" en forme de cœur
5: faire un tri
5.1: cliquer sur le petit icône "favoris" en forme de cœur

Condition de fin: 1) jeu dans les favoris
2) jeu dans les favoris
3) jeu dans les favoris
4) jeu dans les favoris
5) jeu dans les favoris

Notes:

- Étendu sur "Afficher un jeu" et "Afficher les détails d'un jeu »
- L'ajout est possible même si le détail du jeu en question n'est pas affiché (Exemple : sur la page d'accueil, les jeux associés à la recherche peuvent tout de même être ajoutés aux favoris via le cœur à gauche de leur nom)

Nom: consulter les favoris

Objectif: permet à l'utilisateur de consulter les jeux qu'il a mis en favoris

Acteurs: utilisateur

Cas inclus: /

Condition initiale: ouvrir l'application

Scénario initiale: 1: cliquer sur l'icône
1.1: affichage de la liste des jeux

Scénario alternatif: /

Condition de fin: 1) affichage des favoris

Note: Étendu sur "Afficher un jeu" et "Afficher les détails d'un jeu »

Nom: afficher les jeux d'une licence

Objectif: afficher la liste des jeux d'une franchise

Acteurs:

Cas inclus: /

Conditions initiales: être sur la page principale

Scénario nominal: 1: sélectionner une franchise
1.1: cliquer sur la franchise choisie
1.2: affichage des jeux

Scénario alternatif:

Condition de fin: 1) affichage des jeux

Nom: écouter une musique

Objectif: écouter les musiques d'un jeu spécifique pour prendre connaissance de l'univers du jeu

Acteurs: utilisateur

Cas inclus: afficher les détails d'un jeu

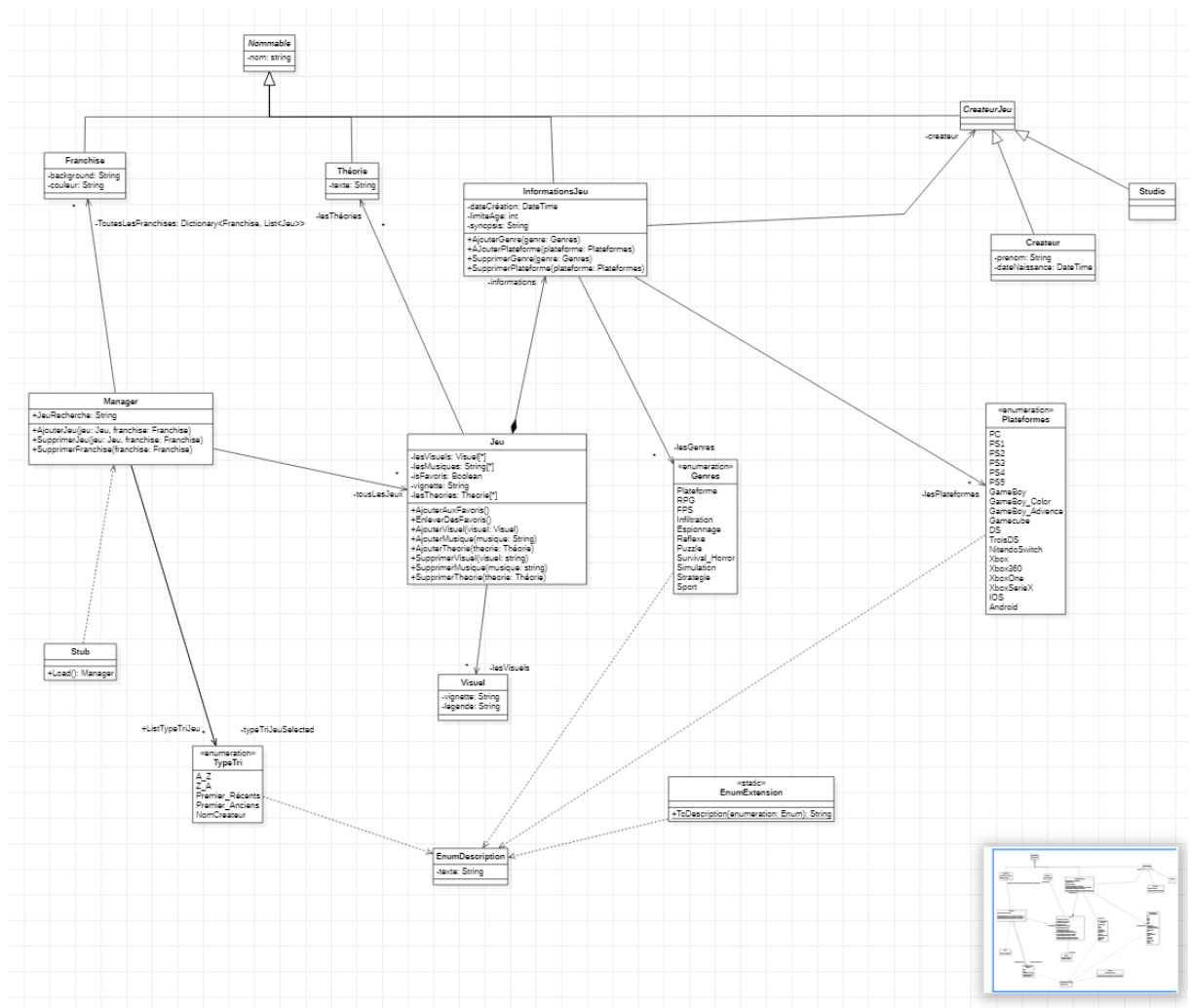
Conditions initiales: avoir affiché le détail des jeux

Scénario nominal: 1: être dans l'onglet musique
1.1: cliquer sur l'icône de lecture de la musique

Scénario alternatif: 2: mettre sur pause
2.1 : remettre la musique
3: arrêter la musique

Condition de fin: 1) fin de la musique
2) fin de la musique
3) arrêt volontaire de la musique en court d'écoute

Description des classes



Dans notre diagramme de classe nous utilisons une classe abstraite nommée *Nommable*. Celle-ci nous permet de faire de l'héritage. En effet les classes *InformationsJeu*, *Théorie*, *CreateurJeu* et *Franchise* dérivent de la classe *Nommable*. Celle-ci ne contient qu'un attribut *nom* de type *string*.

La classe *CreateurJeu* est aussi une classe abstraite, mère de *Createur* et *Studio*. En effet cela rendrait le code plus réutilisable de faire des classes séparées. Si nous voulions ajouter des informations sur le créateur et studio il est possible de le faire.

La classe *informationsJeu* est une classe qui a pour attribut une *dateCreation*, *limiteAge* et un *synopsis*. Cette classe contient une liste de Genre et de Plateforme. La classe dépend des enum Genres et Plateformes. Il est donc possible d'ajouter et de supprimer des genres ou des plateformes avec les classes *ajouterGenres()*, *ajouterPlateforme()*, *supprimerGenre()*, *supprimerPlateforme()*. Ces méthodes prennent en paramètre soit un genre, de type Genre, ou alors une plateforme, de type Plateforme. La classe *Jeu* va dépendre aussi de la classe *InformationJeu*. Les instances de la classe *InformationsJeu* existent seulement quand la classe *Jeu* existe. La vignette correspond à l'image du jeu.

Nous avons dû faire les classe EnumDescription et EnumExtension pour pouvoir écrire les enum comme nous le voulions. Nous avons associé aux noms des enum un autre texte qui sera lui afficher sur la console et donc manipulé par l'utilisateur.

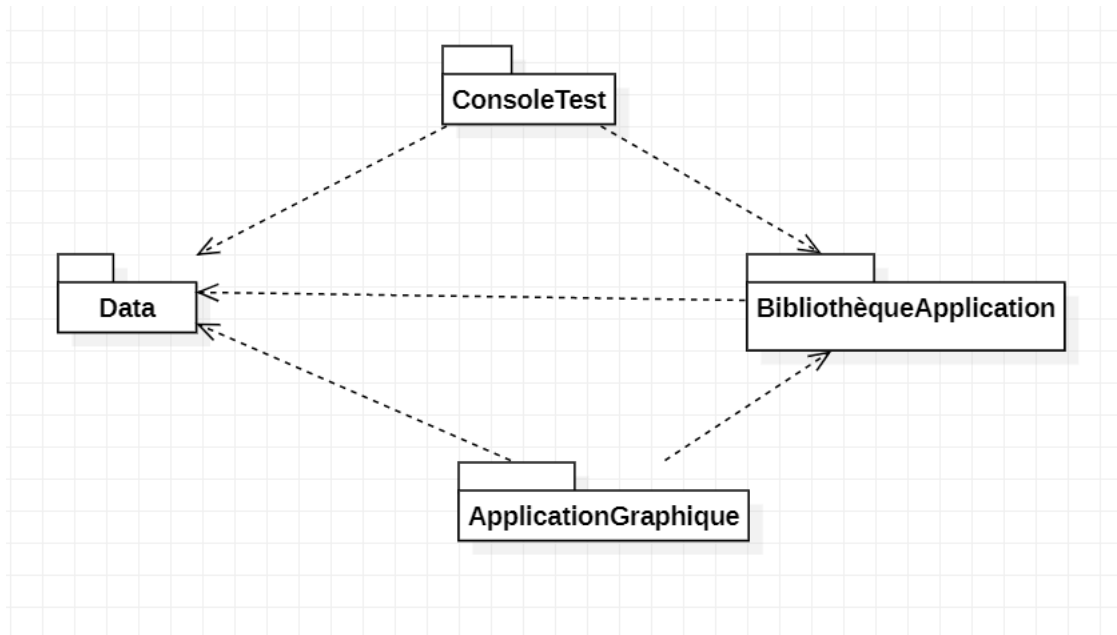
Le booléen IsFavoris permet de déterminer si le jeu est mis en favoris ou non, quand il est true le jeu est dans les favoris. Les méthodes qui rend un jeu favori ou non sont *ajouterAuxFavoris()* et *enleverDesFavoris()* qui ne prennent rien en paramètre. Cette classe contient trois listes : une liste de visuels, de musique et de théorie. Il n'y a pas de nombre défini de ces éléments dans la liste. Pour faciliter la manipulation des éléments, nous avons préféré faire une classe Théorie et Visuel. La classe Jeu dépend des deux classes Visuels et Théorie. Nous avons des méthodes qui permettent d'ajouter des visuels (*ajouterVisuels()*), des musiques (*ajouterMusique()*) et des théories (*ajouterTheorie()*). Ces méthodes prennent en paramètre l'élément à ajouter. Nous avons aussi des méthodes permettant de les supprimer : *supprimerVisuel()*, *supprimerMusique()*, *supprimerTheorie()*. Ces méthodes, comme les méthodes d'ajout, prennent en paramètre l'élément à supprimer.

Nous avons aussi une classe *Franchise* qui a pour attribut un *background* de type string et une *couleur*. Ces attributs nous permettent de relier le code à la vue.

Notre application contient une classe *Manager* qui permet de gérer un dictionnaire de franchises. En effet nous avons décidé de faire un dictionnaire qui a pour clef les franchises et pour valeur une liste de jeux correspondant à la franchise. Cette classe contient une propriété calculée sur la liste des jeux permettant de faire un tri sur la liste ou alors une recherche. Pour associer les différents types de tri nous avons fait un enum *TypeTri* contenant les différents types de tri que l'utilisateur pourra choisir sur la vue. La méthode *ajouterJeu()* vérifie si le jeu n'est pas déjà existant avant de l'ajouter. Pour cela on va parcourir toutes les listes de jeux. S'il est déjà existant, on ne l'ajoute pas. Si ce n'est pas le cas on ajoute la franchise correspondante (appel de la méthode *ajouterFranchise()*) et on ajoute le jeu dans la liste des jeux correspondant à la franchise. Il est possible de supprimer un jeu dans une franchise grâce à la méthode *supprimerJeu()*. Elle ne supprime seulement le jeu mis en paramètre. Pour supprimer une franchise il faut utiliser la méthode *supprimerFranchise()* qui prend la franchise à supprimer. En supprimant la franchise, la liste des jeux associées à la franchise est elle aussi supprimée.

Pour finir nous avons une classe *Stub* qui permet de stocker des informations qu'on utilise pour faire des tests fonctionnels. Cette classe contient une méthode *load()* qui permet de retourner un manager qui a les données. Cela nous sert à avoir ces données dans l'application.

Description des package



Le package nommé *BibliothèqueApplication* contient toute nos classes de notre application. C'est ici que se trouve tous les méthodes principales de notre application. Ce package dépend du package

Data. Ce dernier contient notre classe *Stub*. Elle nous permet de gérer des données qui seront utilisées dans l'application tout comme dans les tests fonctionnels. Ce package ne dépend d'aucun autre package.

Le *ConsoleTest* est le package comportant tous nos tests. Il nous a fallu faire des tests pour vérifier si nos méthodes fonctionnaient correctement. C'est ici qu'ils se font. Il doit dépendre de *Data* pour pouvoir avoir les données pour réaliser les tests. Elle a aussi besoin de dépendre de la *BibliothèqueApplication* pour pouvoir utiliser les méthodes des différentes classes.

Pour finir *ApplicationGraphique* contient toute la partie graphique, xamel, de l'application. Elle doit dépendre de *Data* pour pouvoir utiliser les données et les afficher dans les vues. Elle utilise aussi les méthodes des classes dans *BibliothèqueApplication*.

Description de l'architecture

Les différentes classes de mon package *BibliothèqueApplication* dépendent les unes des autres. En effet la classe *Jeu*, qui est l'une des classes principales, dépend de la classe *InformationsJeu* parce qu'un jeu a besoin d'avoir des informations tel que le nom, la date de création ou encore un synopsis. Un jeu comporte aussi des théories et des visuels. Nous avons donc fait deux classes différentes nommées *Théorie* et *Visuel*. Cela permet de rendre le programme plus optimisé puisque c'est possible de modifier les théories et les visuels plus facilement.

Notre classe qui gère toutes les classes est notre classe *Manager*. C'est elle qui lie le code et la vue. C'est dans cette classe que nous utilisons nos propriétés pour le binding. Notre dictionnaire

de Franchise, avec une liste comme de jeu comme value, se fait ici. C'est comme cela que nous relierons les franchises et les jeux.

Nos classes filles de Nomnable ont toutes des noms et nous trouvons plus judicieux de faire des classes qui hériteraient de nommable. De plus nous faisons de l'héritage aussi entre les CreateurJeu, qui est la classe mère, Createur et Studio qui sont les classes filles. Il était plus simple de faire comme cela si nous voulions ajouter des informations sur les studios de production des jeux. En effet cela nous permet d'avoir une possible évolution pour notre application. Pour le moment nous ne voulons pas vraiment ajouter ce genre d'information mais il est possible de le faire à l'avenir.

Pour la partie graphique nous avons décidé de faire seulement une seule vue. En effet nous avons remarqué que si nous faisons plusieurs vues cela dupliquerait du code. Une seule nous suffit. Nous utiliserons donc plusieurs User Control et faire de la navigation entre eux pour rendre notre application fonctionnelle.

Diagramme de séquence

Le diagramme de séquence montre comment on ajoute un jeu à la liste des jeux en fonction d'une franchise.

