
Documentation Technique de Caiman

Version 1.1.0

Lorenzo Bauduccio

juin 10, 2021

table des matières

1	Introduction	1
1.1	Contexte	1
1.2	Motivations	1
1.3	Résumé	2
1.4	Abstract	2
1.5	Légalité du projet	3
1.6	Installation	3
1.7	Poster	5
2	Cahier des charges	7
2.1	Objectifs	7
2.2	Spécification	7
2.3	Limites du Projet	9
2.4	Calendrier	9
2.5	Matériel	10
3	Étude d'opportunité	11
3.1	RetroPie	11
3.2	RetroArch	12
3.3	RomStation	13
3.4	Steam	14
3.5	Conclusion de l'analyse de l'existant	14
4	Analyse fonctionnelle	15
4.1	Base de données	15
4.2	Site web Caiman	17
4.3	Application Caiman C#	32
4.4	Interface utilisateur utilisable à la manette	39
4.5	Serveur Debian 10	42
5	Analyse organique	43
5.1	Technologies utilisées	43
5.2	Architecture du projet	44

5.3	Description technique : Site web	47
5.4	Description technique : API	52
5.5	Description technique : Application Caiman	58
5.6	Description technique : Interface graphique	77
6	Tests	87
6.1	Test de l'interface graphique	87
6.2	Test de Caiman	88
7	Planning	91
7.1	Planning prévisionnel	92
7.2	Planning effectif	94
8	Conclusion et perspectives	97
8.1	Problème rencontrés	97
8.2	Expérience acquise durant le travail	98
8.3	Améliorations envisageables	98
8.4	Conclusion Globale	99
9	Sources	101
10	Remerciement	103
11	Logbook	105
11.1	19.04.2021	105
11.2	20.04.2021	107
11.3	21.04.2021	107
11.4	22.04.2021	108
11.5	23.04.2021	109
11.6	26.04.2021	110
11.7	27.04.2021	112
11.8	28.04.2021	113
11.9	29.04.2021	113
11.10	30.04.2021	114
11.11	notes pour le premier rendu	115
11.12	03.05.2021	115
11.13	04.05.2021	119
11.14	05.05.2021	119
11.15	06.05.2021	121
11.16	07.05.2021	122
11.17	10.05.2021	123
11.18	11.05.2021	124
11.19	12.05.2021	125
11.20	13.05.2021	126
11.21	14.05.2021	126
11.22	17.05.2021	127
11.23	18.05.2021	128
11.24	19.05.2021	128
11.25	20.05.2021	129

11.26	21.05.2021	130
11.27	24.05.2021	131
11.28	25.05.2021	131
11.29	26.05.2021	132
11.30	27.05.2021	133
11.31	28.05.2021	133
11.32	30.05.2021	134
11.33	31.05.2021	135
11.34	01.06.2021	135
11.35	02.06.2021	135
11.36	03.06.2021	136
11.37	04.06.2021	136
11.38	07.06.2021	137
11.39	08.06.2021	137
11.40	09.06.2021	137
11.41	10.06.2021	138

CHAPITRE 1

Introduction

1.1 Contexte

Caiman a été réalisé dans le cadre du travail de diplôme de technicien deuxième année. La réussite du travail de diplôme est l'une des conditions pour recevoir son titre de technicien en informatique.

1.2 Motivations

Le choix de ce travail de diplôme découle de mon envie de faire un travail de diplôme qui serait intéressant, ludique et en lien avec les jeux vidéo. La réalisation d'un jeu vidéo étant trop compliquée pour un travail de diplôme, j'ai décidé de créer une application à mi-chemin entre un launcher de jeu comme Steam ou GOG et un gestionnaire de VM.

J'utilise depuis des années des émulateurs et je me heurte souvent aux mêmes problèmes, perte de sauvegarde dû à une réinstallation de windows, problème pour ripper mes jeux, téléchargement de chaque émulateur un par un. Donc, J'ai eu l'idée de créer Caiman pour pallier ces différents problèmes.

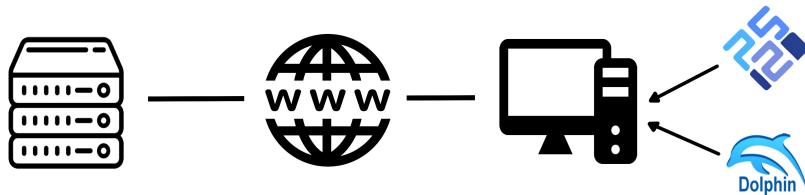
Un autre aspect important pour moi est la simplicité. Je voulais que l'utilisation de l'application soit le plus simple possible, une personne sans connaissance particulière en informatique doit pouvoir utiliser Caiman sans difficulté. Une autre spécification est que l'utilisation de Caiman puisse se faire à la manette. Pour créer une interface ergonomique, je me suis basé sur l'interface Big Picture de Steam qui a justement été créé pour être utilisé à la manette.

1.3 Résumé

Ce document décrit le processus de conception de mon projet Caiman. Caiman est inspiré par des applications comme Retropie, RetroArch ou Steam. RetroPie est une distribution pour raspberry comprenant des émulateurs pour d'anciennes consoles de jeu. RetroArch est lui une application servant de frontend pour émulateur, RetroArch est disponible sur plusieurs plateformes(Windows, macOS, Xbox).

Caiman est un projet comprenant deux parties distinctes. La première partie consiste en un site web PHP permettant de se créer un compte et de s'identifier. Le site web permet de consulter des informations sur les jeux présent sur le store ainsi que de pouvoir administrer les sites et le store.

La deuxième partie du projet est une application C# inspirée par le mode Big Picture de steam. Caiman sert de frontend pour différents émulateurs (Dolphin, PCSX2,etc). L'application permet de télécharger les différents jeux qui ont été ajoutés au store prévu pour le projet. Les jeux pris en charge sont ajoutés par des administrateurs via le site internet de Caiman. L'application permet de synchroniser les sauvegardes de l'utilisateur et cela peut importe sur quel pc, il lance l'application.



1.4 Abstract

This document describes the design process of my project Caiman. Caiman is inspired by applications like Retropie, RetroArch or Steam. RetroPie is a raspberry distribution including emulators for old game consoles. RetroArch is a frontend application for emulators, RetroArch is available on several platforms (Windows, macOS, Xbox).

Caiman is a project with two distinct parts. The first part consists of a PHP website allowing the visitor to create an account and to identify himself. The website allows you to consult information about the games on the store and to administer the site and the store.

The second part of the project is a C# application inspired by the Big Picture mode of Steam. Caiman is used as a frontend for different emulators (Dolphin, PCSX2, etc). The application allows you to download the different games that have been added to the store provided for the

project. Supported games are added by administrators via the Caiman website. The application allows synchronizing the user's save, no matter on which PC the user runs the application.

1.5 Légalité du projet

1.5.1 Caiman

Je pense qu'il est nécessaire d'aborder le sujet de la légalité de Caiman. Selon le droit Suisse, il n'est pas légal de mettre à disposition des fichiers sous droit d'auteur. Il est par contre légal de télécharger des films, séries ou jeux à condition de les utiliser dans un but privé. Caiman ne répondant pas à ces descriptions l'utilisation de l'application n'est pas conforme au droit Suisse.

1.5.2 Utilisation d'émulateurs

L'utilisation d'émulateurs n'est pas illégale en soi, en sachant que le code des émulateurs n'est pas la propriété de Sony ou de Nintendo. Par contre, il y a un petit point technique à détailler. L'émulateur PCSX2 doit pour fonctionner utiliser un BIOS de console de Playstation 2. Le BIOS de la Playstation 2 étant soumis à des droits d'auteur, il n'est en théorie pas légal de le distribuer en Suisse.

Je tiens donc à spécifier que Caiman est développé dans un seul but pédagogique. Il n'a aucune vocation à être distribué après la fin de mon travail de diplôme. L'application sera supprimée du serveur de l'école à la fin de mon travail. Par contre, il serait légal de distribuer le projet si je supprime la fonctionnalité de téléchargement de jeu.

1.6 Installation

1.6.1 Installation de Caiman utilisateur

Pour utiliser Caiman, il faut le télécharger depuis caiman.cfpt.info. Quand on télécharge l'application, l'utilisateur télécharge un fichier caiman.zip contenant l'application ainsi que les émulateurs, il n'y a donc rien d'autre à télécharger.

Quand l'utilisateur décomprime le fichier caiman.zip, il se retrouve avec un dossier Caiman contenant deux dossiers. Un contenant les émulateurs et un autre contenant les fichiers de Caiman.



Pour pouvoir exécuter Caiman, il faut se rendre dans le dossier "\bin\Debug\" et ensuite l'utilisateur va trouver le fichier "Caiman.exe". Ce fichier permet de lancer l'application.

Caiman est donc une application "portable", c'est-à-dire qu'elle ne nécessite pas d'une installation pour pouvoir être lancée.

1.6.2 Installation de Caiman serveur

Pour pouvoir déployer le site et l'API de caiman, il faut que le serveur ait les services suivant :

- Apache2
- PHP 7.4 (minimum)
- mysql

Pour la base de données, il faut l'importer dans mysql sans faire de manipulation particulière.

Le site web et l'api doivent se trouver dans le même dossier, en sachant qu'il a des chemins relatifs entre les deux dossiers. ils ne peuvent donc pas être séparés pour l'instant.

Pour pouvoir uploader des jeux, il faut augmenter la taille que le PHP peut recevoir par formulaire, il faut également le configurer pour recevoir minimum 8GB. Pour finir, Il faut autoriser l'écriture dans les dossiers des sauvegardes et des jeux.

1.7 Poster



CHAPITRE 2

Cahier des charges

2.1 Objectifs

Caiman est une application pour Windows regroupant plusieurs émulateurs de console de jeu. L'utilisateur peut exécuter les jeux depuis Caiman.

Caiman a pour but d'être simple d'utilisation, aucune connaissance n'est requise pour l'utiliser. Caiman permet une utilisation complète à la manette ou au clavier/souris. L'avantage de Caiman est qu'il ne requiert aucune configuration. Caiman est utilisable instantanément par l'utilisateur, contrairement d'un émulateur traditionnel qui requiert une configuration relativement compliquée.

Le téléchargement des jeux se fait directement depuis Caiman, un serveur nommé le "Bunker" contient les fichiers des différents jeux.

Un site web permet de créer un compte pour accéder à l'application , de voir les informations des joueurs ainsi que de télécharger Caiman.

2.2 Spécification

2.2.1 Emulateurs et contrôles

Les différents émulateurs qui permettent d'exécuter des jeux dans Caiman sont :

- PCSX2 (playstation 2)
- Dolphin (gamecube / wii)

L'utilisateur doit utiliser une manette de Xbox pour utiliser Caiman à la manette (une seule manette est requise pour jouer mais plusieurs peuvent être connectées, pour jouer en multijoueurs local).

Caiman est utilisable à la manette à l'exception des formulaires (création de compte, importation de jeux).

2.2.2 Création d'un compte utilisateur

L'utilisateur a l'obligation de créer un compte pour pouvoir utiliser Caiman.

L'utilisateur peut créer un compte directement depuis Caiman s'il n'en possède pas.

2.2.3 Interface

Dans l'interface, un système de catégories existe pour permettre de se retrouver plus aisément dans la liste des jeux à télécharger. Les différentes catégories sont créées par un administrateur.

Exemple de catégories :

- jeux Mario
- jeux Zelda
- jeux d'aventure
- jeux de réflexion
- jeux multijoueur
- jeux favoris
- etc..

L'utilisateur peut ajouter depuis Caiman des jeux favoris pour pouvoir les retrouver plus facilement. (une catégorie "favoris" est visible contenant les jeux favoris de l'utilisateur authentifié)

2.2.4 Paramètre graphiques

L'utilisateur a la possibilité de modifier certains paramètres d'émulation :

- La définition des jeux (définition native, proche de 1080p, proche de 4K).
- La langue des machines.
- Si les jeux doivent être lancés en plein écran ou non.
- Format d'écran 16/9 ou 4/3.

2.2.5 Gestion des sauvegardes

Il existe plusieurs cas où les sauvegardes de l'utilisateur se synchronisent :

1. Le joueur sauvegarde sa partie directement dans un jeu depuis Caiman. La sauvegarde que l'utilisateur vient de créer est envoyée sur le Bunker et remplace la version présente.
2. Quand le joueur lance Caiman, une vérification est faite pour savoir si une version plus récente des sauvegardes du joueur sont présentes, si c'est le cas, elles sont téléchargées sur la machine du joueur.

2.2.6 Spécifications du “Bunker”

Le Bunker contient les fichiers des différents jeux au format (.ISO).

Le Bunker contient les fichiers de sauvegardes des utilisateurs. Ces sauvegardes seront envoyées de la machine de l'utilisateur vers le Bunker quand l'utilisateur sauvegarde sa partie depuis un jeu.

Les données des utilisateurs (email, jeux favoris, heures de jeux) sont stockées dans une base de données.

2.2.7 Spécification site web (site présent dans le Bunker)

Sur le site web, il est possible de créer un compte en spécifiant un mail et un mot de passe.

L'interface web permet de rechercher des utilisateurs pour voir leur profil.

L'affichage d'un profil permet de voir les jeux favoris d'un utilisateur ainsi que son nombre d'heures de jeu sur chaque jeu.

Le site web permet à un administrateur d'ajouter des jeux et de modifier les informations des jeux.

Le site web permet de modifier les informations de compte d'un utilisateur.

Un utilisateur peut faire la demande de réinitialiser son mot de passe.

2.2.8 Installation

Caiman est téléchargeable depuis le site web. L'installateur contient les émulateurs, il n'y a donc rien à télécharger d'autre.

2.3 Limites du Projet

Caiman sera limité aux jeux que j'ai pu ajouter moi même ainsi qu'aux émulateurs que j'aurais importé, la totalité des consoles de jeu ne sera donc pas prise en charge.

2.4 Calendrier

Le début du travail de diplôme est fixé au lundi 19 avril et le rendu est fixé au 11 juin à 12h00.

Le travail est à exécuter soit en présentiel et en télétravail du lundi au vendredi et cela 8h00 par jour.

2.5 Matériel

J'ai à ma disposition pour la réalisation de Caiman :

- Mon pc pour le développement
- Intel i7-7700 3.6 GHz
- 32 GB de ram
- GTX 1060 3GB
- Un serveur pour héberger le site et les fichiers du store
- Une manette de Xbox Series

CHAPITRE 3

Étude d'opportunité

3.1 RetroPie



RetroPie est une distribution Linux prévue pour Raspberry et PC. C'est un frontend pour jouer aux jeux d'arcade, d'anciennes consoles et de jeux pc. Retropie est à la différence de Caiman un OS à part entière et chaque jeu doit être ajouté indépendamment par l'utilisateur sur le Raspberry/PC.

3.2 RetroArch



RetroArch est un frontend pour émulateurs, moteur de jeu et media players. Il embarque un très grand nombre d'émulateurs et a l'avantage d'être disponible sur un très grand nombre de plateformes.

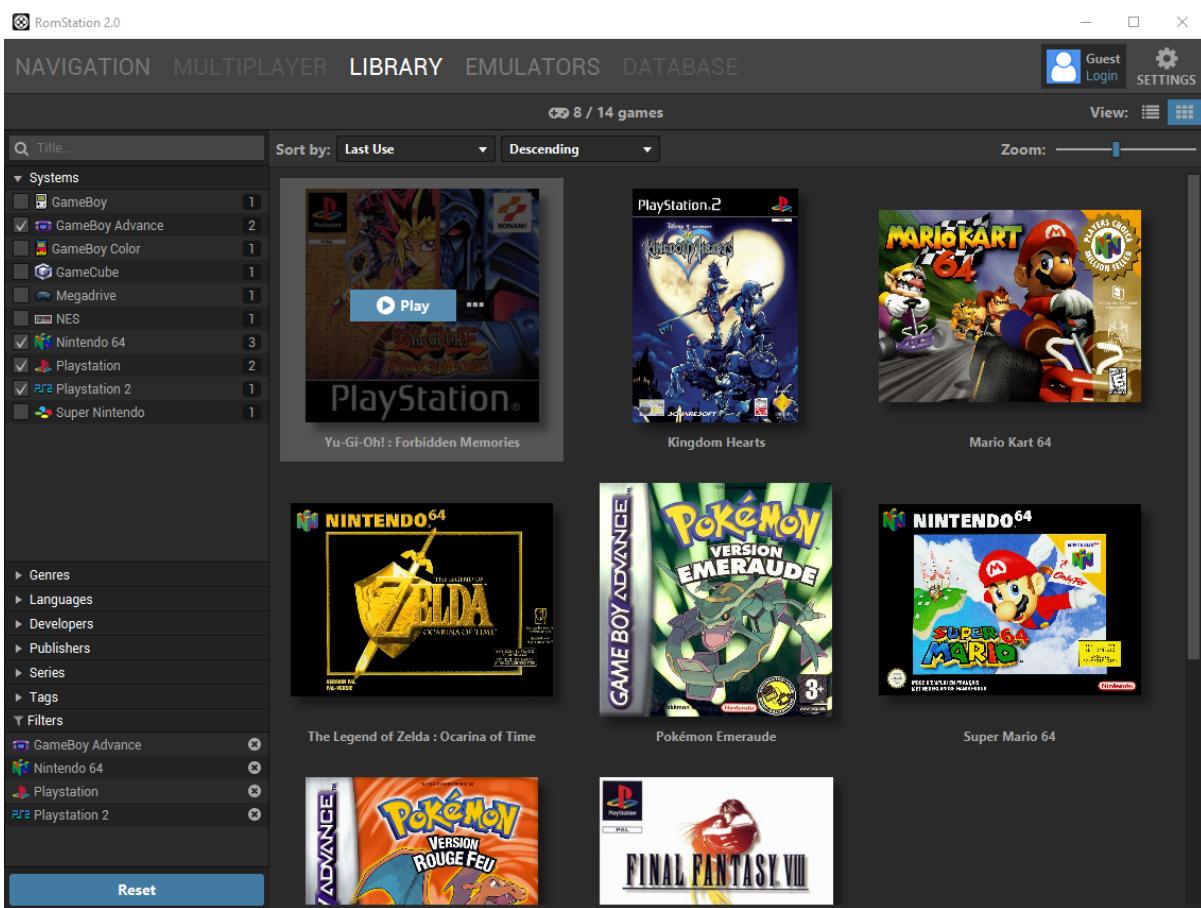
RetroArch possède des fonctionnalités très utiles comme des Shaders intégré, le jeux en réseau, le calcul du temps entre les frames, et d'autres. C'est sûrement le Frontend pour émulateur le plus complet disponible.

L'interface de RetroArch est inspirée par le XMB de la Playstation 3 et de la PSP, ce qui le rend compatible avec une manette.

RetroArch est disponible sur :

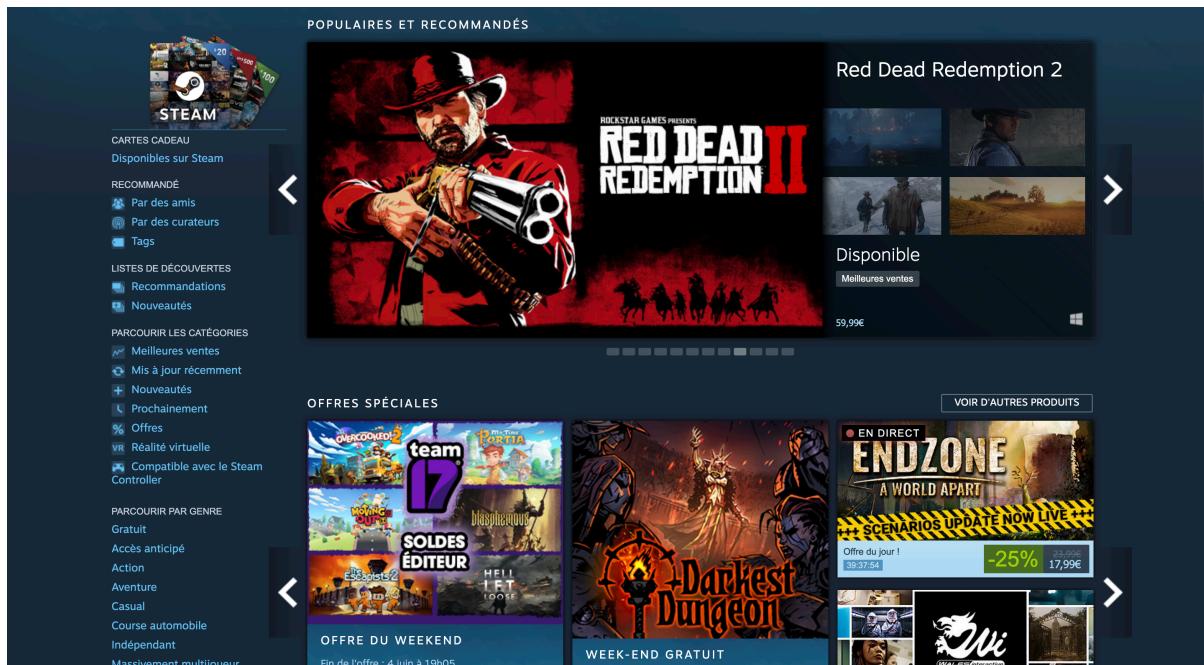
- Windows
- Linux
- Raspberry
- Android
- Apple macOS (ARM/x64)
- Apple macOS/OSX
- IOS/Apple TV
- Xbox Series / One
- OpenDingux
- PSVita
- PSP
- PS2
- PS3
- PS4
- Switch
- WII U
- WII
- Gamecube
- 3DS / 2DS

3.3 RomStation



RomStation est un frontend pour émulateur facilitant l'émulation de nombreuses consoles. RomStation a la différence de RetroPie ou RetroArch, l'utilisateur a la possibilité de télécharger des jeux directement depuis l'application. L'application a une formule payante qui permet d'améliorer la vitesse des téléchargements des jeux. RomStation est disponible sur Windows (x64)(x86), macOS.

3.4 Steam



Steam est une plateforme de distribution de jeux et d'applications en ligne. Steam est spécialisé dans les jeux vidéo contrairement aux autres projets que j'ai cité ci-dessus. C'est une application commerciale à but lucratif.

Steam n'est pas directement liée à l'utilisation d'émulateurs, mais j'ai décidé d'en parler car je m'inspire de son interface. Elle est créée spécifiquement pour une utilisation à la manette et la gestion que fais steam dès sauvegardes des utilisateurs.

3.5 Conclusion de l'analyse de l'existant

Il existe un grand nombre de frontend pour émulateur, ils permettent de simplifier l'utilisation pour l'utilisateur mais je n'en ai pas trouvé qui essaie d'atteindre le même but que moi. Le but de Caiman est de permettre la simplification de l'utilisation d'émulateurs et de permettre de synchroniser des sauvegardes tout en fournissant la possibilité de télécharger des jeux.

Steam représente un niveau de finition dans son interface BigPicture que j'aimerais atteindre. La synchronisation des sauvegardes sur steam est peut-être la plus performante, si l'on compare avec d'autres boutiques comme l'Epic Game Store ou Origin.

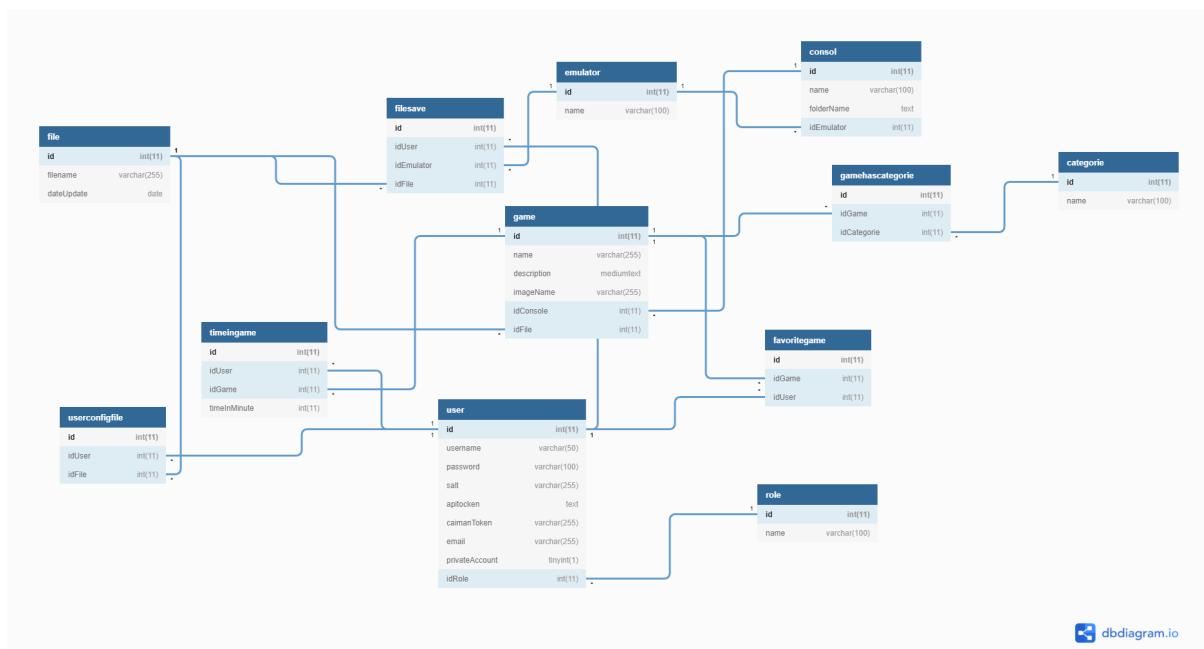
CHAPITRE 4

Analyse fonctionnelle

4.1 Base de données

4.1.1 Schéma

Le schéma suivant représente la structure de la base de données. La base de données est commune au site web et à l'application Caiman. Les tables sont principalement utilisées pour stocker les informations des utilisateurs et des différents jeux.



dbdiagram.io

4.1.2 Table categorie

Cette table est utilisée pour stocker les différentes catégories auxquelles les jeux peuvent appartenir.

4.1.3 Table console

Cette table sert à stocker les différentes consoles prises en charge par l'application Caiman. Chaque console doit être reliée à un émulateur.

4.1.4 Table emulator

La table émulateur sert à lister les différents émulateurs disponibles pour l'application, il est possible qu'un émulateur soit compatible avec plusieurs consoles.

4.1.5 Table favoritegame

Sert à lister les jeux favoris des utilisateurs.

4.1.6 Table file

La table file sert à lister le nom d'un fichier, cela peut être un fichier .iso d'un jeu, un fichier de sauvegarde ou un fichier de configuration utilisateur.

4.1.7 Table filesave

Sert faire le lien entre un émulateur, un utilisateur, et un fichier de sauvegarde. Alors elle permet de connaître les sauvegardes pour un émulateur particulier d'un utilisateur.

4.1.8 Table game

Table qui liste les jeux disponibles. Chaque jeu a plusieurs informations : un nom, une description, une console. La console permet à l'application de savoir quel émulateur doit être utilisé.

4.1.9 Table gamehascategorie

Sert à assigner des catégories aux jeux.

4.1.10 Table rôle

Sert à lister les différents rôles (utilisateur, administrateur), elle ne sert que pour le site web sachant que toute la partie administration est sur le site.

4.1.11 Table timeingame

Sert à stocker le temps de jeu en minutes de chaque utilisateur. Le temps de jeu est spécifique à chaque jeu. Il est mis à jour directement depuis l'application Caiman.

4.1.12 Table user

Cette table sert à stocker les informations de compte de chaque utilisateur de Caiman. Le compte est commun au site web et à l'application. Le mot de passe de l'utilisateur est crypté avec les fonctions de sécurité de php.

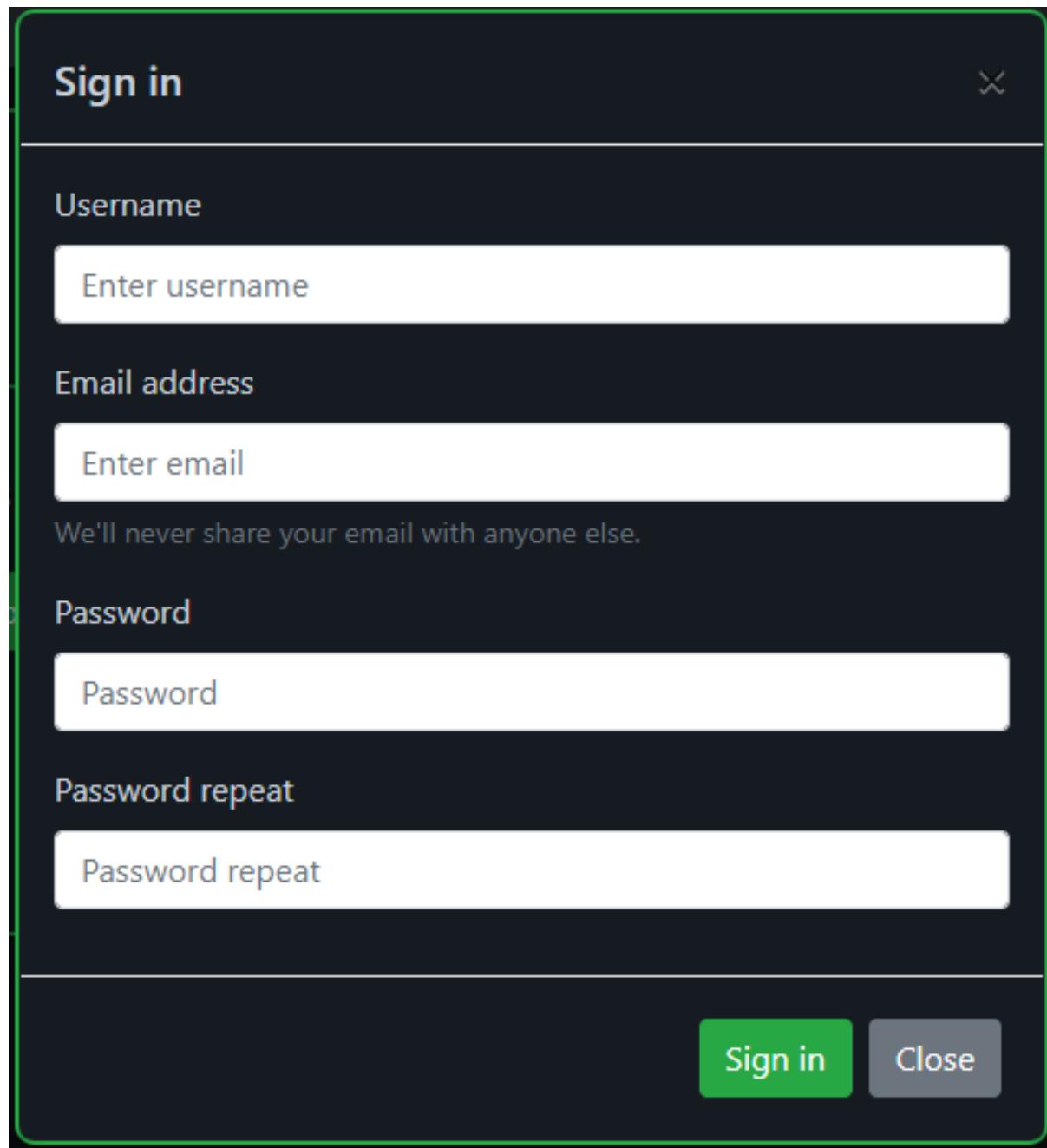
4.1.13 Table userconfigfile

Cette table sert à faire le lien entre un fichier de configuration et un utilisateur. Le fichier de configuration sert à connaître la configuration.

4.2 Site web Caiman

4.2.1 Crédation de compte

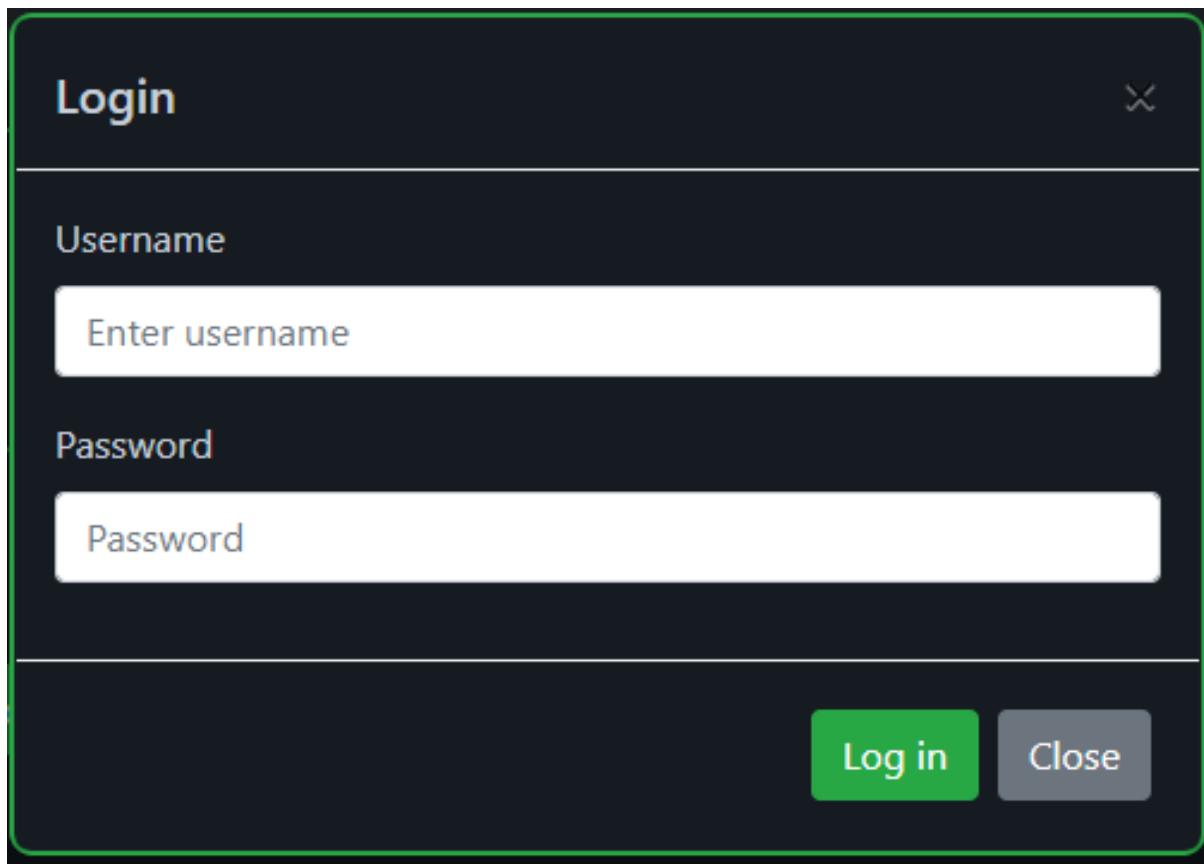
L'utilisateur du site a la possibilité de créer un compte qui sera commun au site et à l'application. La création de compte nécessite de renseigner son email, de donner un nom d'utilisateur ainsi que un mot de passe.



4.2.2 connexion

La connexion à son compte utilisateur permet de modifier nos informations de compte et d'ajouter ou de supprimer des jeux à la liste de favoris.

Si l'utilisateur oublie son mot de passe, il a la possibilité de le réinitialiser. L'utilisateur qui se décide de réinitialiser son mot de passe reçoit un mail contenant un lien de réinitialisation.



4.2.3 modification des informations d'un utilisateur

Un utilisateur connecté a la possibilité de modifier ces informations :

- mot de passe
- liste de jeux favoris
- la visibilité de son profil pour les autres utilisateurs

Caiman Games Download Users Search games Dashboard admin Dashboard Logout

User's Informations

Username: lorenzo1227
Email: united4player@gmail.com
Your account is visible for other users

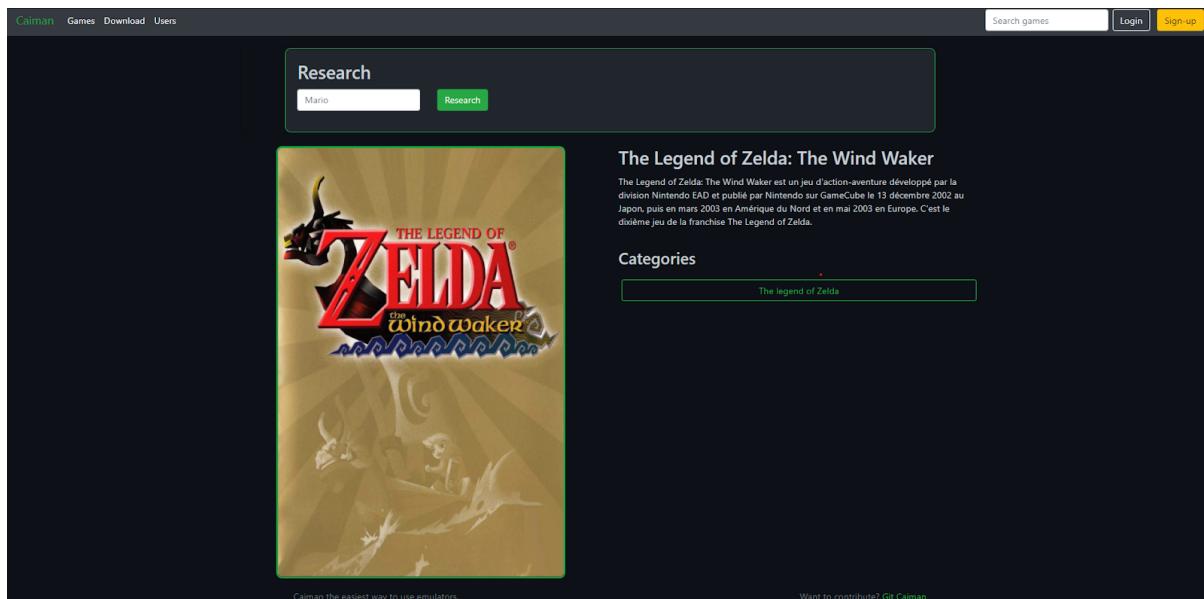
Update password Update if account is private

4.2.4 affichage des jeux

Tous les utilisateurs ont la possibilité d'afficher la liste de jeux disponible. Il n'y a pas de restriction particulière.

Les informations disponibles pour chaque jeu sont les suivantes :

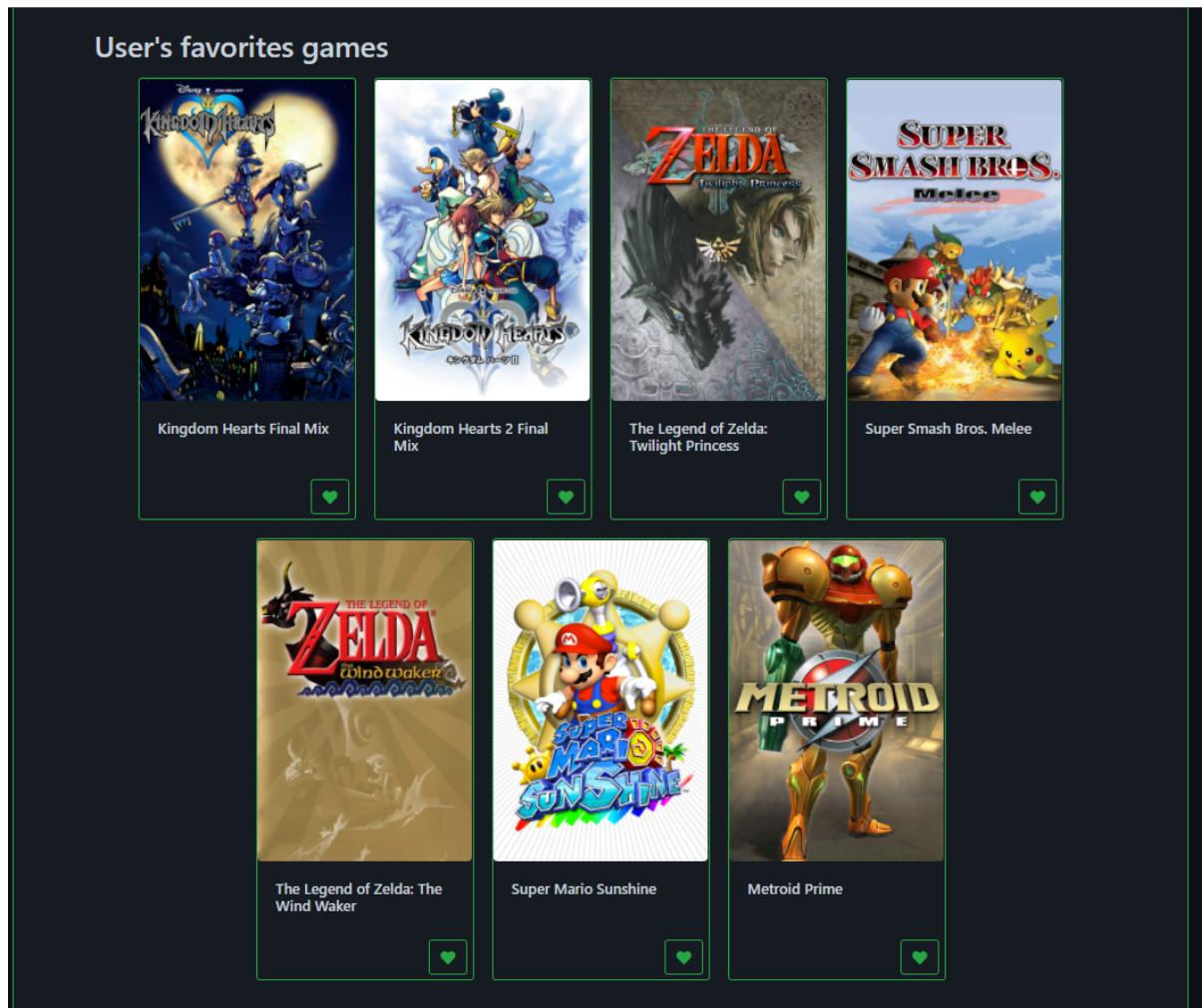
- Nom
- Description
- Catégories du jeu

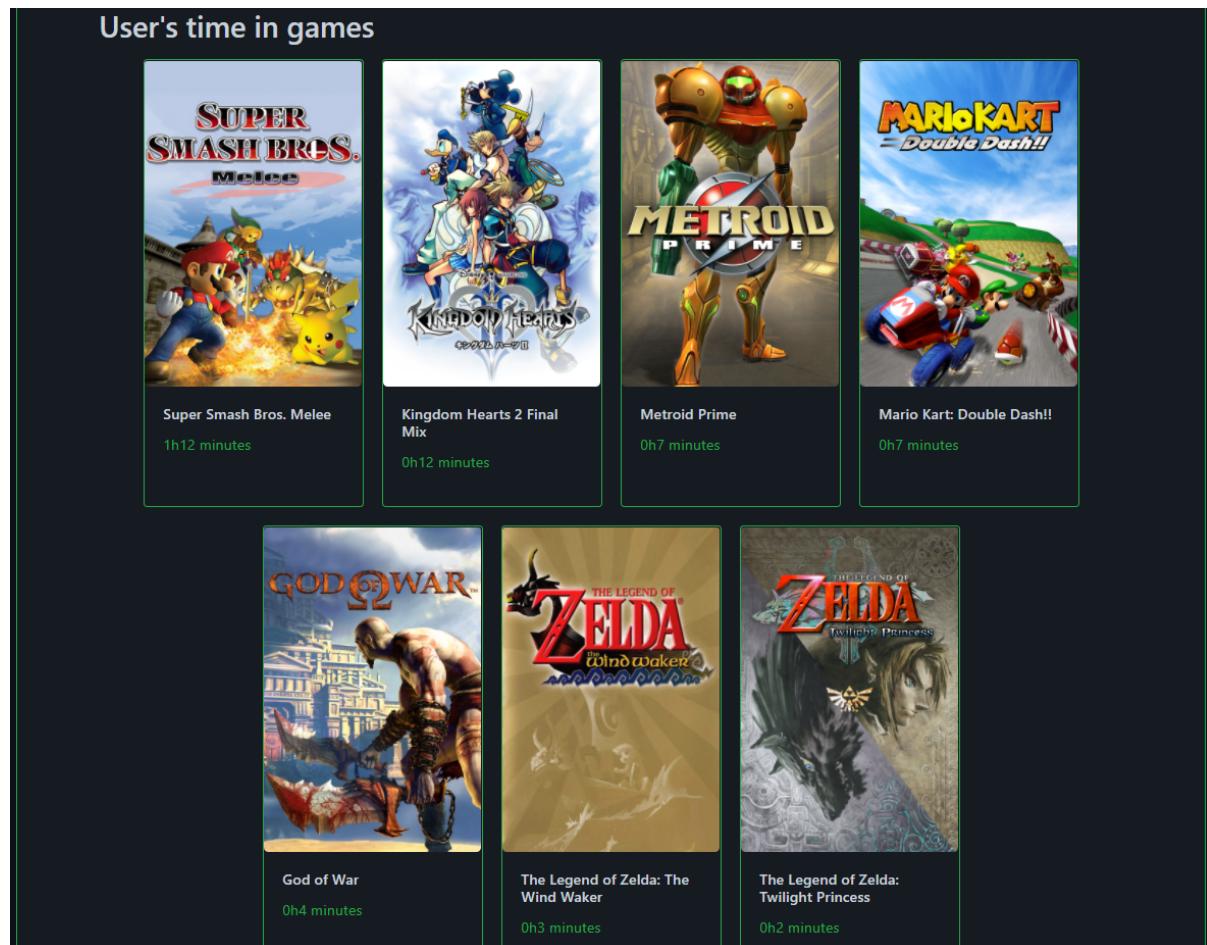


4.2.5 affichage d'un profil utilisateur

Il est possible de consulter la page personnelle d'un utilisateur si celui-ci a rendu son compte public. Les informations disponibles sont celle-ci :

- Nom d'utilisateur
- Jeux favoris
- Nombres d'heures de jeux sur chaque jeu





4.2.6 ajout d'un jeu à la base de données / sur le Bunker pour le fichier .ISO

L'ajout d'un jeu se fait grâce à un formulaire, plusieurs champs sont à renseigner :

- le nom du jeu
- une description
- une image
- la console du jeu qui est uploadé
- le nom que va porter le jeu sur le Bunker
- le fichier de base du jeu

L'ajout à la base va créer deux entrées. Une dans la table Game et une autre dans la table file. Après avoir été ajouté depuis le site web, le jeu devient accessible depuis l'application Caiman et le site web pour de la consultation.

Add game

Game's name

Game's description

Game's image name.
Example: Super Mario Sunshine -> SUPER_MARIO_SUNSHINE.png

Image File
 Aucun fichier choisi

Console

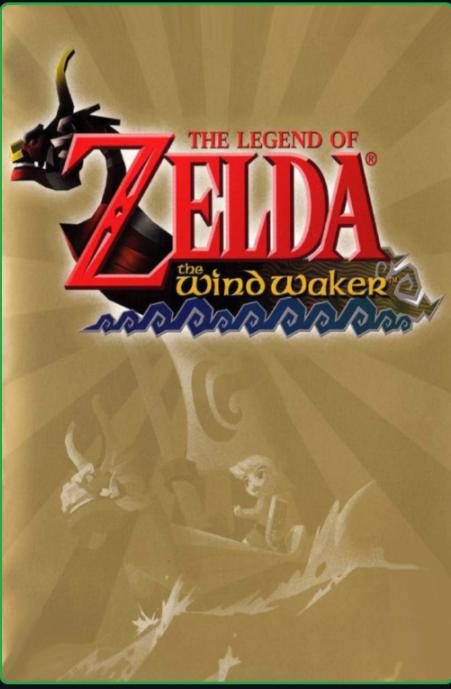
Game's file name.
Example: Super Mario Sunshine -> SUPER_MARIO_SUNSHINE.iso

Game file
 Aucun fichier choisi

4.2.7 modification d'un jeu

La modification d'un jeu ne peut être faite que par un administrateur. Les modifications possibles sont les suivantes :

- nom
- description
- console
- catégories



The Legend of Zelda: The Wind Waker

The Legend of Zelda: The Wind Waker est un jeu d'action-aventure développé par la division Nintendo EAD et publié par Nintendo sur GameCube le 13 décembre 2002 au Japon, puis en mars 2003 en Amérique du Nord et en mai 2003 en Europe. C'est le dixième jeu de la franchise The Legend of Zelda.

[Remove favorite](#)

[Update game](#)

[Update/add categories](#)

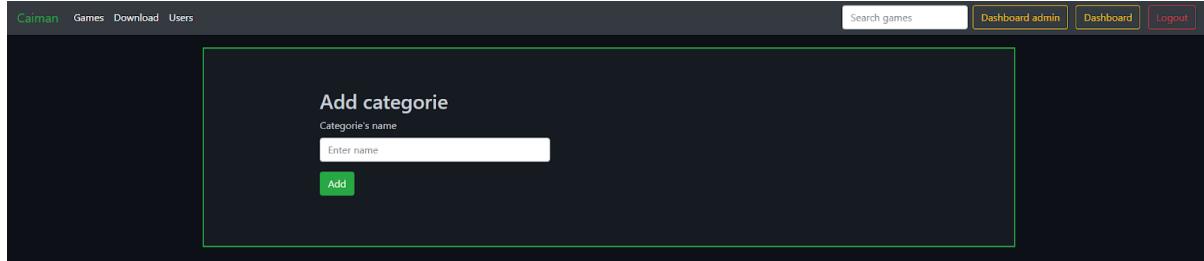
Categories

[The legend of Zelda](#)

4.2.8 Administration

Les administrateurs ont la possibilité de faire plusieurs choses. Donc, je vais les lister ici, il n'est pas nécessaire de les détailler.

- modifier un jeu
- ajouter des catégories
- ajouter ou supprimer des catégories à un jeu



Caiman Games Download Users Search games Dashboard admin Dashboard Logout

Add categorie

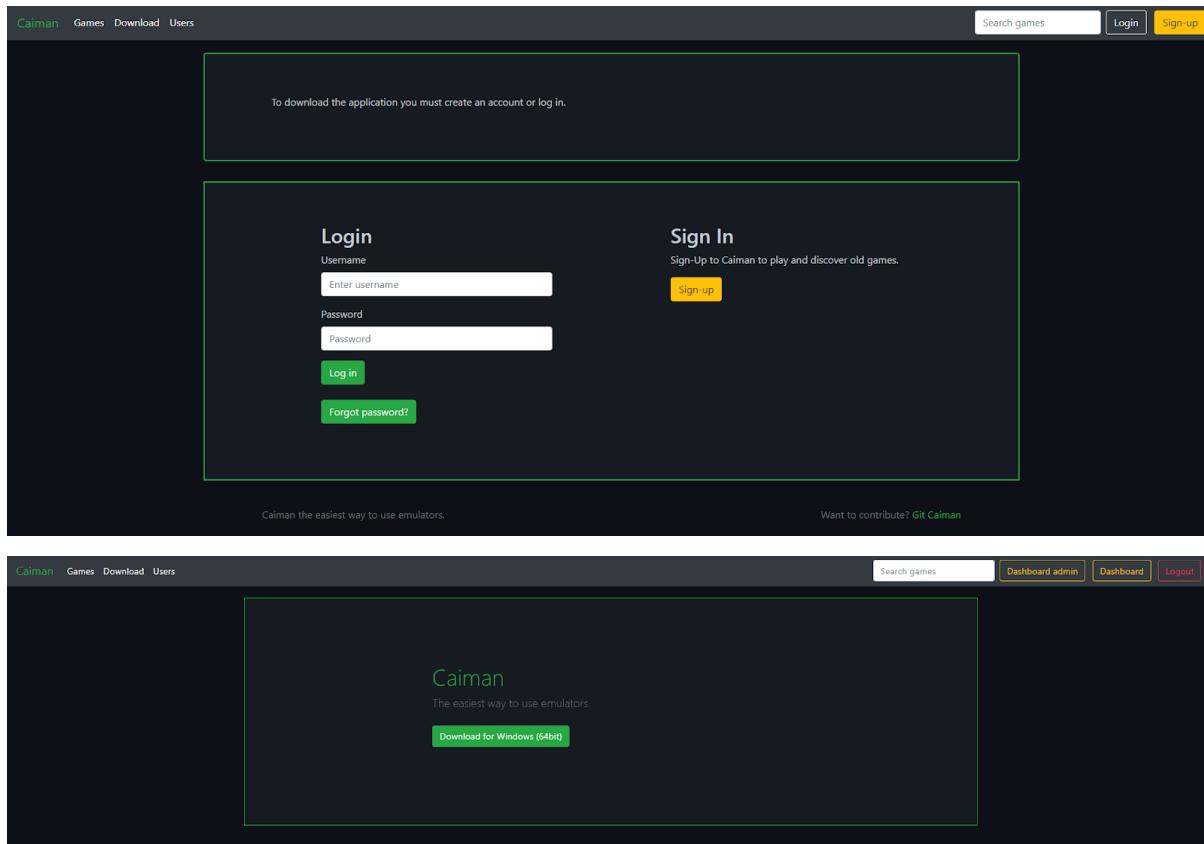
Categorie's name

Enter name

Add

4.2.9 Téléchargement

L'un des intérêts du site est de pouvoir télécharger l'application Caiman. Le téléchargement de cette application nécessite d'être authentifié sur le site. Si un invité se rend sur la page de téléchargement sans être authentifié, une invitation lui sera faite de s'authentifier ou de créer un compte.



4.2.10 Fonctionnalités manquante

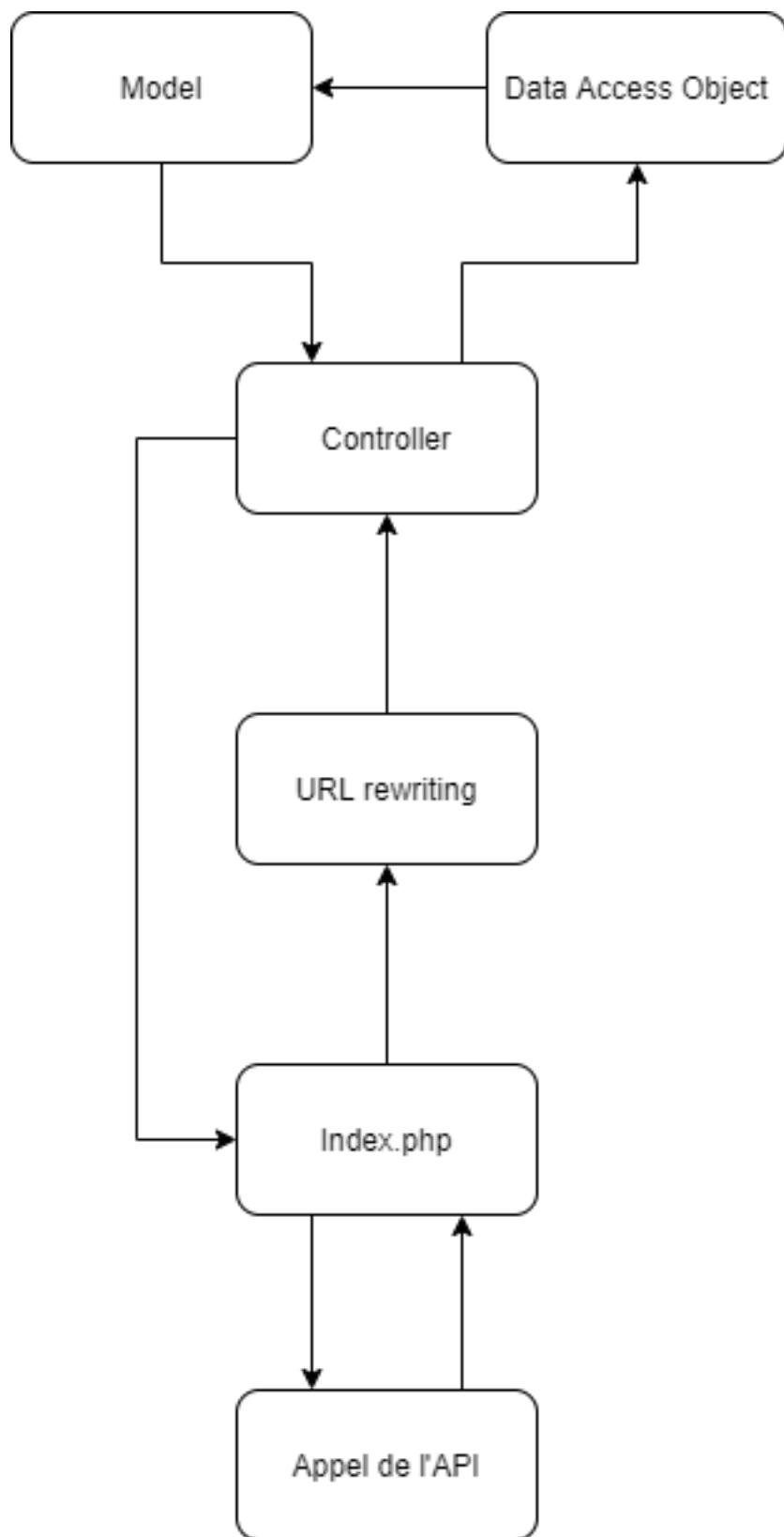
Malheureusement dû au temps imposé j'ai dû faire des choix organisationnel. J'ai donc décidé de me focaliser sur L'application Caiman au lieu du site internet ce qui explique que certaines fonctionnalités ne soient pas finalisées ou mises en place. Je vais lister les ce qui n'a pas été fini complètement ou abandonné.

- Modification complète des données des jeux
- Modification d'une catégorie
- Réinitialisation de mot de passe
- Création de compte administrateur

4.2.11 API Caiman

L'API sert à pouvoir accéder à la base de données depuis l'application Caiman. Je vais détailler les endpoint et la structure de l'API.

4.2.12 Structure de l'API



Pour expliquer la structure de l'API, je vais expliquer étape par étape comment un appel se passe.

1. L'utilisateur envoie une requête à la page index.php de api.caiman.cfpt.info.

2. La requête est réceptionnée par index.php. L'url est ensuite traité par le .htaccess pour savoir où doit envoyer à quel controller.
3. Le contrôleur décide selon les informations reçues quelle méthode il doit exécuter.
4. Le DAO est appelé et va rechercher dans la base de données les données demandé
5. Le DAO crée la réponse grâce au model.
6. La réponse est envoyée à l'utilisateur par l'intermédiaire de la page index.php

4.2.13 Categories

4.2.14 GET

Permet de recevoir la liste des catégories disponibles.

Les informations reçues sont les suivantes :

- id
- nom

4.2.15 Games

4.2.16 GET

Retourne la liste des jeux.

Les informations reçues sont les suivantes :

- id
- description
- nom de l'image
- id de la console
- id du fichier du jeu

4.2.17 GET(?byName)

Retourne la liste des jeux qui dans le nom contient ce que l'utilisateur a demandé.

Les informations reçues sont les suivantes :

- id
- description
- nom de l'image
- id de la console
- id du fichier du jeu

4.2.18 GET(?byCategory)

Retourne la liste des jeux qui appartiennent à une catégorie.

Il faut spécifier l'id de la catégorie qui est demandée.

Les informations reçues sont les suivantes :

- id
- description
- nom de l'image
- id de la console
- id du fichier du jeu

4.2.19 GET(?byFavoriteUser)

Retourne la liste des jeux favoris d'un utilisateur.

Il faut spécifier l'id de l'utilisateur.

Les informations reçues sont les suivantes :

- id
- description
- nom de l'image
- id de la console
- id du fichier du jeu

4.2.20 GET(?byUserTime)

Retourne la liste des jeux auxquels un joueur a joué.

Il faut spécifier l'id de l'utilisateur.

Les informations reçues sont les suivantes :

- id
- description
- nom de l'image
- id de la console
- id du fichier du jeu
- nombre de minutes en jeu

4.2.21 GET(?gameFileName)

Retourne le nom du fichier d'un jeu.

Les informations reçues sont les suivantes :

- filename

4.2.22 GET(?gameConsole)

Retourne la console d'un jeu.

Les informations reçues sont les suivantes :

- name
- folderName

4.2.23 GET(?idGame&apiKey)

Retourne le fichier d'un jeu.

Les informations reçues sont les suivantes :

- fichier.iso

4.2.24 GET(?idGameTime&idUser)

Retourne le temps de jeu sur un jeu.

Les informations reçues sont les suivantes :

- minutes

4.2.25 GET(?idEmulator&idUser&apiKey)

Retourne un fichier zip contenant les sauvegardes d'un joueur pour un émulateur particulier

Les informations reçues sont les suivantes :

- fichier.zip

4.2.26 POST(?idEmulator&idUser&apiKey)

Upload un fichier zip contenant les sauvegardes d'un joueur pour un émulateur particulier

4.2.27 POST(idGameAdd&idUser)

Ajouter un jeu en favoris pour un utilisateur particulier

4.2.28 POST(idGameRemove&idUser)

Supprime un jeu en favoris pour un utilisateur particulier

4.2.29 POST(idGameCheck&idUser)

Vérifie si un jeu est déjà en favoris et retourne un booléen

4.2.30 POST(idGameTimeAdd&idUser)

Ajouter une minute de jeu à un jeu particulier pour un utilisateur

4.2.31 Users

4.2.32 GET(sans paramètres)

Retourne la liste des utilisateurs.

Les informations reçues sont les suivantes :

- id
- username

4.2.33 POST(avec apitoken)

Retourne un utilisateur en particulier

Les informations reçues sont les suivantes :

- id
- username
- password
- salt
- apitoken
- caimanToken
- email
- idRole

4.2.34 User/connection

4.2.35 POST(?username, password)

Permet de vérifier les informations de connexion d'un utilisateur

Les informations reçues sont les suivantes :

- id
- username
- password
- salt
- apitoken
- caimanToken
- email
- idRole

4.2.36 POST(caimanToken)

Permet de recevoir les informations d'un utilisateur grâce à un token généré à chaque connexion.

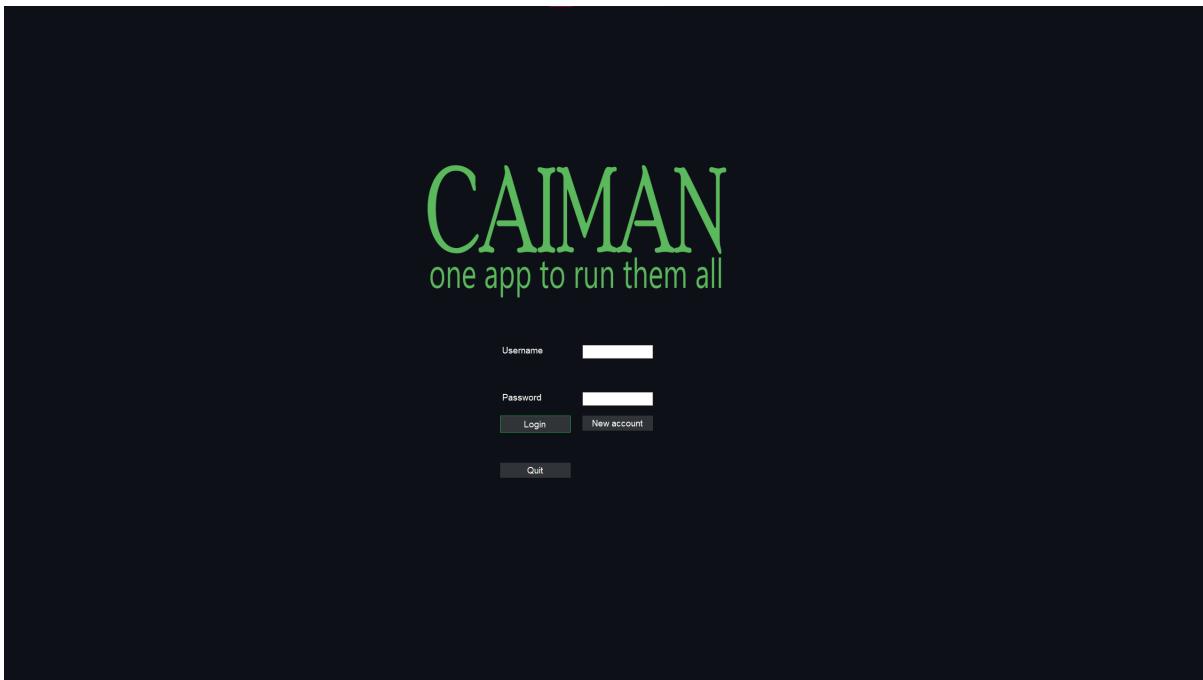
Les informations reçues sont les suivantes :

- id
- username
- password
- salt
- apitoken
- caimanToken
- email
- idRole

4.3 Application Caiman C#

4.3.1 Connexion

L'utilisateur de Caiman doit obligatoirement être connecté pour pouvoir utiliser caiman. La connexion se fait avec le nom d'utilisateur et le mot de passe. Une fois la connexion valable, elle est maintenue tant que l'utilisateur ne se déconnecte pas ou tant que l'utilisateur ne s'est pas connecté sur un autre ordinateur.

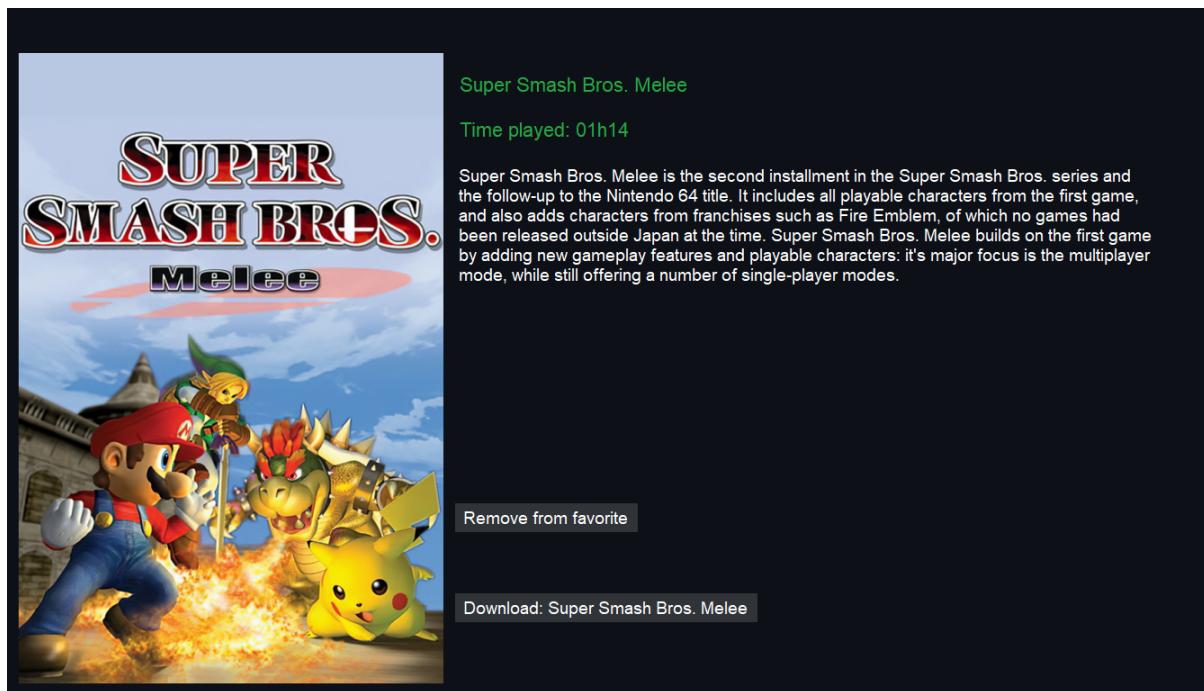


4.3.2 Inscription

L'inscription n'est pas disponible sur caiman mais un bouton est disponible pour être redirigé sur le site web de Caiman.

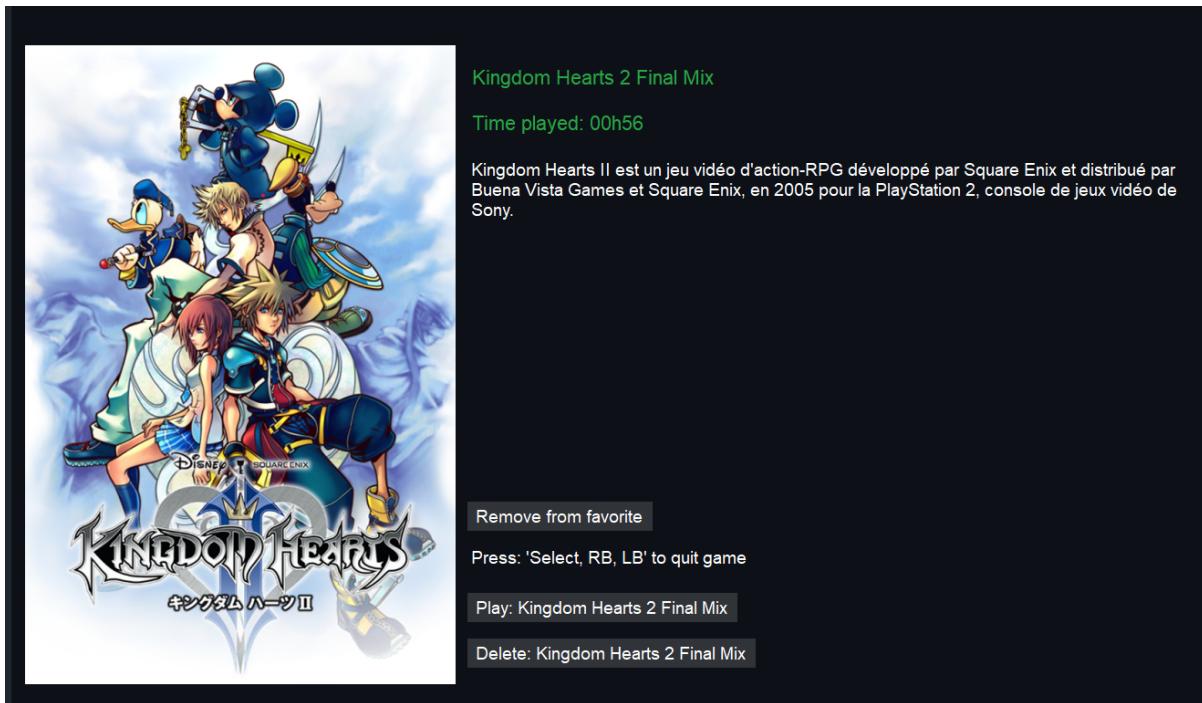
4.3.3 Téléchargement de jeu

L'utilisateur a la possibilité de télécharger des jeux. Les jeux disponibles dans l'application sont ajoutés depuis le site internet de Caiman. Les téléchargements des jeux se font les un après les autres.



4.3.4 Lancement d'un jeu

Caiman inclut deux émulateurs Dolphin et PCSX2. Ces deux émulateurs permettent d'exécuter des jeux de Gamecube et Wii pour Dolphin et de Playstation 2 pour PCSX2. Pour lancer un jeu, il suffit de le télécharger, puis de cliquer sur Play. Il n'est donc pas nécessaire de lancer soit même un émulateur.



4.3.5 Synchronisation des sauvegarde entre le pc client et le Bunker

Les sauvegardes de l'utilisateur sont synchronisées entre les différents pc qu'il utilise. La synchronisation se fait à la connexion, les sauvegardes sont stockées sur les serveurs de Caiman. La copie des sauvegardes des utilisateurs est envoyée automatiquement donc l'utilisateur n'a pas de manipulation à faire. L'envoie se passe dès que l'utilisateur sauvegarde, cela permet d'éviter de perte de sauvegardes si un problème se produit durant le moment où le joueur est en train de jouer.

4.3.6 Modification de la configuration utilisateur

L'utilisateur a la possibilité de modifier plusieurs paramètres graphiques. Il a la possibilité de choisir en trois mode de configuration global :

- Original
- 1080p
- 4K

C'est différent mode modifie l'antialiasing et la définition native de l'émulateur qui va être utilisé.

L'utilisateur a aussi la possibilité de choisir entre lancer le jeu plein écran ou non et il peut choisir si le jeu doit être en 16/9 ou en 4/3.

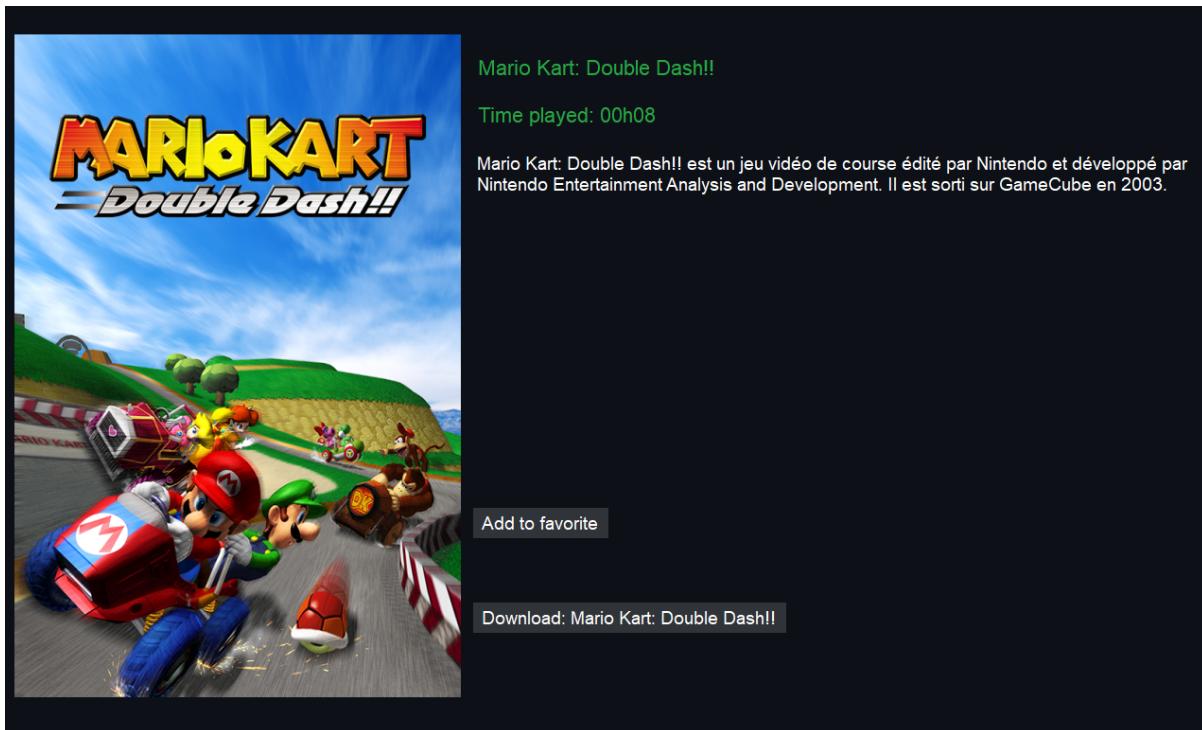


4.3.7 Application de la configuration utilisateur

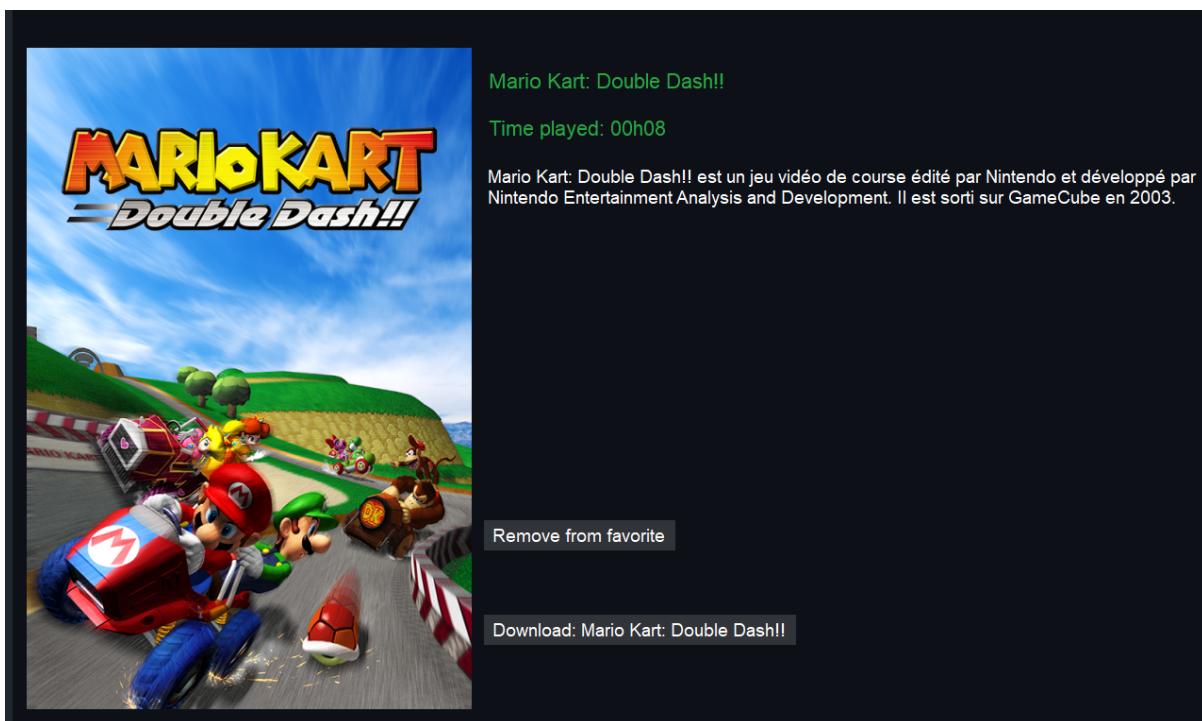
La configuration est appliquée avant de lancer un jeu, c'est à dire que la configuration n'est pas appliquée si l'utilisateur est déjà en jeu.

4.3.8 Ajout/suppression de jeu en favoris

L'utilisateur a la possibilité de modifier ses jeux favoris directement depuis caiman.

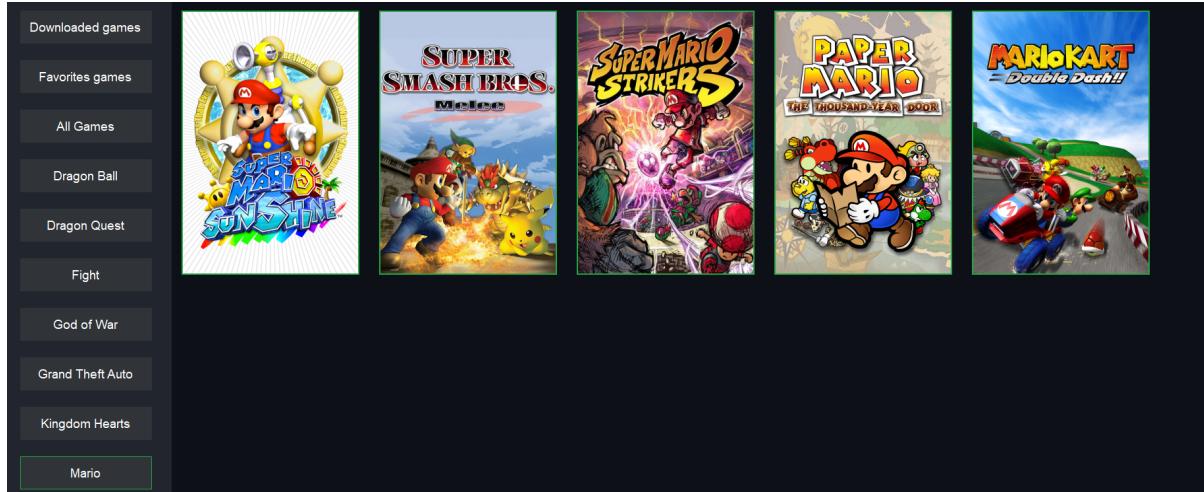


Suppression de jeu des favoris



4.3.9 Affichage de jeux par catégories

L'utilisateur a la possibilité d'afficher les jeux par catégories. Les catégories sont créées manuellement par un administrateur. Il y a certaines catégories « spéciales », par exemple les jeux favoris et les jeux uniques téléchargés par chaque utilisateur.



4.3.10 Nombre d'heures de jeu

Le nombre de minutes de jeu d'un utilisateur est mis à jour à chaque minute de jeu. le nombre de minutes de jeu est visible sur la page de chaque jeu si l'utilisateur n'a pas encore joué.



4.3.11 Gestion des manettes

Pour Caiman, les manettes supportées sont les manettes pour xbox qui fonctionne avec Xinput. Il est possible d'utiliser d'autres manettes pour cela, il faut passer par un programme qui va convertir les inputs de la manette non compatible en input de manette xbox.

Pour gérer les déplacements, j'utilise les touches “haut, bas, gauche, droite”, le stick gauche, la validation se fait avec la touche “A” et le retour arrière avec la touche “B”.

4.4 Interface utilisateur utilisable à la manette

4.4.1 Réception des input des manettes connecté

L'utilisateur de Caiman a la possibilité de pouvoir utiliser l'application au clavier souris mais aussi à la manette. Pour ce faire, j'ai utilisé le paquet nuGet "SharpDX.XInput". Ce paquet me permet de connaître les manettes connectées au pc ainsi que les touches appuyées par l'utilisateur.

La seule manette qui peut se déplacer dans l'application est la manette 1. Pour connaître les boutons de la manette, j'utilise la fonction getInput(). Cette fonction me permet de connaître les touches qui sont pressées à un instant T. Je vais chercher les inputs toutes les 2ms pour être sûr de ne pas louper d'inputs.

Les inputs sont ensuite traités par l'interface de l'application qui décide quoi en faire selon le contexte.

4.4.2 Transformation des input de la manette en événement

Les inputs de la manette sont analysés par la form principale de Caiman. Selon la ou les touches qui sont pressées, l'application exécute des actions différentes. Par exemple, quand la touche "A" est pressée, alors le programme envoie la touche ENTER à l'application ce qui me permet de cliquer sur des boutons.

Pour gérer les événements, je fais un test pour savoir si l'utilisateur utilise l'application ou non . Si l'application n'est pas focus par l'utilisateur, seul une partie des actions sont possibles pour éviter que des actions inattendues puissent arriver alors que l'application n'est plus visible.

4.4.3 Structure de l'affichage

L'affichage est constitué d'un "XboxMainForm". Il sert à contenir tous les autres panels, il est aussi chargé de la gestion des inputs de la manette de l'utilisateur 1.

Un "XboxMainForm" contient une liste de XboxUserController qui elles contiennent différentes choses comme des boutons des images ou des labels.

Le XboxMainForm est aussi responsable de la gestion des demandes de l'utilisateur, par exemple si l'utilisateur veut afficher l'accueil de Caiman, il va lui passer un objet contenant sa demande. Il va donc traiter les demandes et afficher les panels selon les besoins de l'utilisateur.

4.4.4 Déplacement dans un panel de l'application

L'application est conçue avec des "panels", c'est-à-dire une liste de listes de controls. Cette liste de controls est propre à chaque panel. Les panels contiennent aussi une variable position_x et position_y qui permettent de connaître le control actuellement sélectionné par l'utilisateur. Quand l'utilisateur décide de se déplacer, il demande au panel de modifier ses variables x et y. Avant de valider ce changement, le panel regarde si le déplacement demandé par l'utilisateur est possible ou non.

Il existe 3 possibilités :

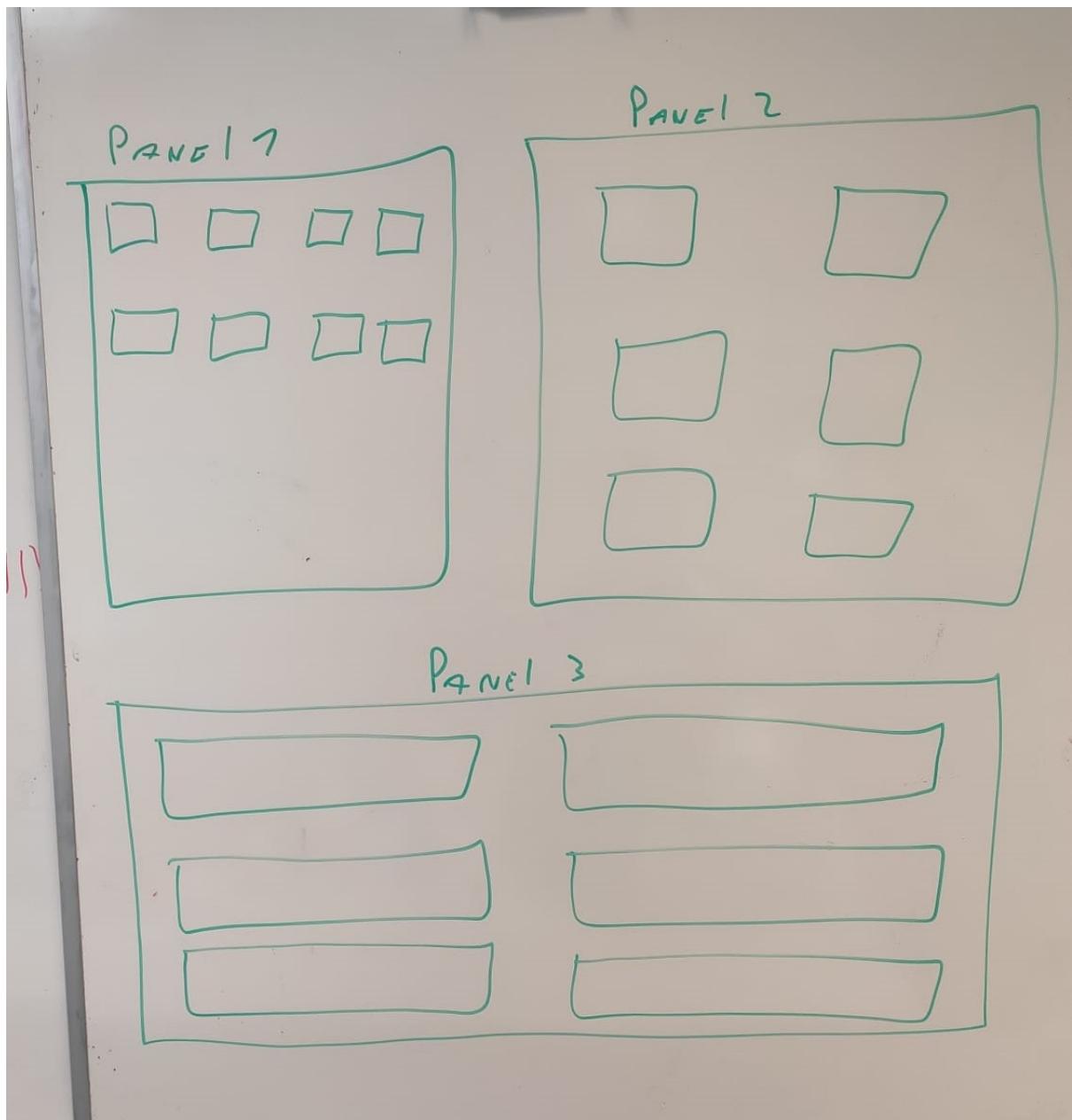
1. Le déplacement est possible, alors la position sur l'axe x,y est modifiée.
2. Le déplacement est impossible car il n'y a rien à l'emplacement demandé. Dans ce cas, le panel va décider de bouger le curseur sur un des emplacements possibles.
3. L'utilisateur est à la fin du panel et "sors du panel" dans ce cas il va se diriger dans un autre panel s'il y en a un dans la direction demandée.

4.4.5 Déplacement de panel en panel

Chaque "panel" possède un pointeur sur le panel du haut, du bas, de droite et de gauche.

Ces pointers ne sont pas forcément utilisés, ils ont le droit d'être nul.

Si on prend l'exemple suivant :



Nous avons 3 panels différents qui contiennent chacun plusieurs controls.

Le panel 1 possède donc deux pointeurs différents, un sur le panel 2 et un autre sur le panel 3.
trop confus

Si l'utilisateur se trouve en bas du panel 1 et qu'il décide de se déplacer encore plus bas, il ne pourra pas car aucune case n'est disponible dans ce panel. C'est pourquoi une vérification sera faite pour savoir si un panel est indiqué comme le panel "down", si tel est le cas le focus va changer de panel.

Un autre cas possible est que l'utilisateur va peut-être décider de retourner sur le panel 1. La contrainte est de savoir où doit pointer le panel haut du panel 3. Actuellement, un seul panel peut être défini par côté, mais la solution est de créer de petits panels pour éviter que ces situations arrivent.

4.5 Serveur Debian 10

J'ai utilisé un serveur sous Debian10 pour héberger le site web de caiman, l'API et les fichiers des différents jeux.

4.5.1 Configuration de PHP

J'ai dû installer PHP pour le site web et l'API. Donc, je vais détailler la configuration que j'ai faite pour Caiman.

- version 7.4
- max_input_time = 600
- max_execution_time = 300
- post_max_size = 8G
- upload_max_filesize = 10G

J'ai dû grandement augmenter la taille d'upload pour pouvoir uploader les fichiers des jeux sur le serveur.

4.5.2 Modification des droits des dossiers

J'ai modifié les droits d'écriture pour le dossier où les images sont stockées et les dossiers où sont stockés les fichiers des jeux. Le dossier où sont stockées les sauvegardes des utilisateurs a lui aussi eu ces droits modifiés.

Chaque émulateur qui est supporté par caiman a un dossier où les jeux compatibles sont stockés. Le dossier pour les sauvegardes contient les sauvegardes pour chaque émulateur. Il n'est donc pas nécessaire de créer plusieurs sous-dossiers.

4.5.3 Services installé

J'ai aussi installer un service FTP pour pouvoir transférer les fichiers sur le serveur plus facilement que par ssh. J'ai aussi installé mySQL et apache2.

4.5.4 Crédit de virtual host

Pour simplifier l'accès au site de Caiman et à l'API, les deux ne possèdent pas la même URL. Pour cela, j'ai dû configurer un sous-domaine pour l'API. Le nom de domaine de "base" est "caiman.cfpt.info" mais j'ai créé le sous-domaine "api.caiman.cfpt.info" pour les requêtes à l'API.

Le chemin où pointe "caiman.cfpt.info" est "/var/www/caimanWeb/www" et le chemin pour "api.caiman.cfpt.info" est "/var/www/caimanAPI/public". Pour créer les deux domaines j'ai dû modifier le fichier "/etc/apache2/sites-enabled/000-default.conf". et ajouter les deux chemin et leur nom.

CHAPITRE 5

Analyse organique

5.1 Technologies utilisées

5.1.1 HTML

5.1.2 CSS

5.1.3 PHP

J'ai utilisé le php pour le site web de Caiman ainsi que pour l'API.

5.1.4 C#

L'application Caiman a été développé en C#, j'ai utilisé les paquets suivants :

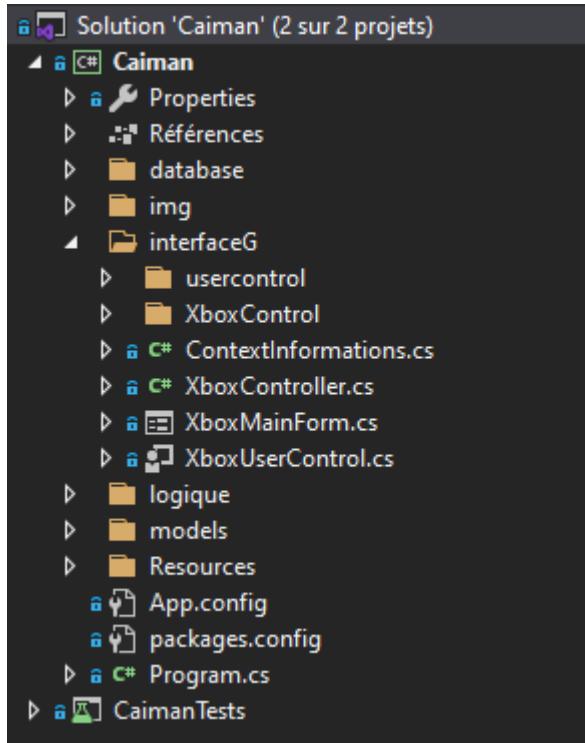
- Microsoft.AspNet.WEbApi.Client
- Newtonsoft.Json
- SharpDX

5.2 Architecture du projet

Le projet se divise en 3 parties : l'application Caiman pour windows, le site web et l'API. Chaque partie du projet contient une documentation doxygen qui lui est propre.

5.2.1 Caiman

Je vais expliquer ci-dessous l'arborescence des fichiers de l'application Caiman et comment les émulateurs y sont intégrés.



Je vais expliquer ce que contiennent les différents dossiers.

Database

Contient les classes servant à appeler la base de données.

interfaceG

Ce dossier contient les fichiers de base de l'affichage.

interfaceG\usercontrol

Ce dossier contient les différents panels qui sont utilisés dans Caiman pour créer l'affichage.

interfaceG\XboxControl

Ce dossier contient une version modifiée de plusieurs contrôles de base de Windows Form.

logique

Ce dossier contient les classes servant à gérer les émulateurs et les sauvegardes et plus généralement toute la logique de l'application.

models

Ce dossier contient les modèles pour simplifier l'interaction entre la base de données et Caiman.

emulators

Le dossier est bien présent mais n'est pas visible sur la photo. Ce dossier contient tous les fichiers des émulateurs.

5.2.2 Site web

Le site web contient en plus des fichiers du site les fichiers des jeux et des sauvegardes des utilisateurs.

 games	26.05.2021 08:33	Dossier de fichiers
 release	26.05.2021 15:57	Dossier de fichiers
 saves	30.05.2021 15:43	Dossier de fichiers
 www	30.05.2021 14:12	Dossier de fichiers
 HOST.php	26.05.2021 08:33	Fichier source PHP 1 Ko

games

Ce dossier contient dans un sous dossier spécifique à chaque émulateur les fichiers des jeux.

release

Ce dossier contient le fichier .zip de la version téléchargeable de Caiman depuis le site.

saves

Ce dossier contient les fichiers de sauvegardes des utilisateurs.

www

Ce dossier contient les fichiers du site web de Caiman.

 common	26.05.2021 08:33	Dossier de fichiers
 controllers	30.05.2021 13:31	Dossier de fichiers
 css	26.05.2021 08:33	Dossier de fichiers
 img	26.05.2021 08:33	Dossier de fichiers
 models	30.05.2021 13:31	Dossier de fichiers
 debug.log	30.05.2021 14:49	Document texte 1 Ko
 Doxyfile	26.05.2021 08:33	Fichier 117 Ko
 index.php	26.05.2021 08:33	Fichier source PHP 2 Ko
 test.php	26.05.2021 08:33	Fichier source PHP 2 Ko

common

Ce dossier contient les fichiers de base de la vue comme le footer ou le header.

controllers

Ce dossier contient les différents controllers de l'application.

css

contient les fichiers de css

img

contient les images des jeux

models

contient les fichiers pour simplifier l'accès à la base de données

5.2.3 API

 app	11.05.2021 07:57	Dossier de fichiers
 public	19.05.2021 17:29	Dossier de fichiers

Le dossier de base de l'api contient deux dossiers importants .

app

 Controllers	26.05.2021 08:33	Dossier de fichiers
 DataAccessObject	30.05.2021 13:43	Dossier de fichiers
 Models	30.05.2021 13:50	Dossier de fichiers
 System	13.05.2021 12:10	Dossier de fichiers

Controllers

Ce dossier contient les différents controllers de l'API.

DataAccessObject

Ce dossier contient les différents fichiers servant à exécuter des requêtes à la base de données.

Models

Ce dossier contient les modèles utilisés par l'API.

System

Ce dossier contient des fichiers servant à se connecter à la base de données.

public

Je ne vais pas développer davantage sur le contenu du dossier, en sachant que le détail des différents endpoint est disponible dans la documentation. Néanmoins, ce dossier comporte les points d'accès de l'API.

5.3 Description technique : Site web

5.3.1 Crédit de compte

Caiman propose aux utilisateurs de se créer un compte. Se compte est commun au site internet et à l'application. Pour pouvoir se créer un compte, il faut renseigner plusieurs champs.

- un nom d'utilisateur
- un mail
- un mot de passe
- une validation du mot de passe

Dans l'état actuel de l'application, l'email n'est pas utilisé.

```
// check if email already used
$sqlrequestEmail = "SELECT * FROM user WHERE email = :search_
˓→email ";
$this->psCheckEmail = $this->dbh->prepare($sqlrequestEmail);
$this->psCheckEmail->setFetchMode(PDO::FETCH_ASSOC);

// check if username already used
$sqlRequestUsername = "SELECT * FROM user WHERE username =_
˓→:search_username ";
$this->psCheckUsername = $this->dbh->prepare(
˓→$sqlRequestUsername);
$this->psCheckUsername->setFetchMode(PDO::FETCH_ASSOC);

$sqlInsert = "INSERT INTO user (username, password, email,_
˓→salt, apitocken)

VALUES (:insert_username, :insert_password, :insert_email,_
˓→:insert_salt, :insert_apiTocken)";
$this->psInsert = $this->dbh->prepare($sqlInsert);
$this->psInsert->setFetchMode(PDO::FETCH_ASSOC);
```

5.3.2 Connexion

Pour se connecter au site web de caiman, il faut renseigner son nom d'utilisateur et son mot de passe.

```
$this->psLogin->execute(array(':search_username' => $this->search_
˓→username));
    $result = $this->psLogin->fetchAll();
    if ($result != null) {
        if (md5($result[0]["salt"] . $password_verify) ==
˓→$result[0]["password"]) {
            $returnArray = $result;
            $_SESSION['error'] = "Welcome back: ".
˓→$result[0]['username'];
        } else
        {
            $_SESSION['error'] = "Invalid log in";
        }
    }
}
```

5.3.3 Fonctionnalité disponible une fois connecté

Quand un utilisateur se connecte à l'application, il a accès à de nouvelles fonctionnalités. L'utilisateur a maintenant la possibilité de modifier son mot de passe, ajouter/supprimer des jeux de ses favoris et de modifier la visibilité de son profil (si le profil est visible, il sera affiché dans la liste des utilisateurs).

```
public function updatePassword(string $newPassword, string
˓→$newPasswordRepeat, string $oldPassword)
{
    $dbh = new PDO('mysql:host=' . HOST . ';dbname=' . DBNAME,
˓→USER, PASSWORD, array(
        PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES utf8",
        PDO::ATTR_PERSISTENT => true
    ));

    $hasBeenUpdated = 1;
    if (md5($_SESSION['user']->salt . $oldPassword) == $this->
˓→getUserPassword()) {

        if ($newPasswordRepeat == $newPassword) {
            try {
                $salt = rand(1,10000);
                $sqlUpdatePassword = "UPDATE user SET password =
˓→:update_password, salt = :update_salt WHERE id = :id_user";
                $psUpdatePassword = $dbh->prepare(
˓→$sqlUpdatePassword);
                $psUpdatePassword->execute(array(':update_-
˓→password' => md5($salt . $newPassword), ':id_user' => $this->idUser,
˓→':update_salt' => $salt));
            }
        }
    }
}
```

(suite sur la page suivante)

(suite de la page précédente)

```

        $hasBeenUpdated = 0;
    } catch (PDOException $e) {
        print "Erreur !: " . $e->getMessage() . "<br>";
        die();
    }
} else {
    $hasBeenUpdated = 2;
}
} else {
    $hasBeenUpdated = 4;
}

return $hasBeenUpdated;
}

```

5.3.4 Ajout de jeu

L'ajout de jeu ne peut se faire que par le site et seulement par les administrateurs du site.

Pour pouvoir ajouter un jeu, il faut au préalable avoir sur le disque dur de l'administrateur un fichier .iso du jeu, il faut aussi envoyer une image. Cette image sera affichée sur le site web et dans l'application Caiman.

Pour pouvoir envoyer le formulaire, il faut remplir les champs suivants :

- nom du jeu
- description
- nom de l'image(le nom doit respecter un certain format)
- le fichier de l'image
- la console sur lequel le jeu tourne
- le nom du fichier du jeu(le nom doit respecter un certain format)
- le fichier du jeu

Le serveur est configuré pour recevoir des fichiers jusqu'à 8GB. Donc pour les fichiers de Gamecube qui ne peuvent faire que 1.4GB et les fichiers de Playstation 2 qui peuvent faire jusqu'à 7 GB, il n'y a pas de problèmes.

```

public function uploadGame($gameFileName, $consoleId)
{
    $uploadIsValid = false;
    $target_dir = "../games/" . $this->getConsoleFolderName(
    ↴$consoleId) . "/";
    $target_file = basename($_FILES["fileGame"]["name"]);
    $uploadOk = 1;
    $fileType = strtolower(pathinfo($target_file, PATHINFO_
    ↴EXTENSION));

```

(suite sur la page suivante)

(suite de la page précédente)

```
//rename file
$newfilename = $gameFileName ;

// Check if file already exists
if (file_exists($target_file)) {
    echo "Sorry, file already exists.";
    $uploadOk = 0;
}
if ($uploadOk == 0) {
    echo "Sorry, your file was not uploaded.";
    // if everything is ok, try to upload file
} else {
    if (move_uploaded_file($_FILES["fileGame"]["tmp_name"],
→$target_dir . $newfilename)) {
        $uploadIsValid = true;
    } else {
        //Sorry, there was an error uploading your file
    }
}
return $uploadIsValid;
```

5.3.5 Ajout / suppression de catégories au jeux

Chaque jeu peut appartenir à une ou plusieurs catégories. Seuls les administrateurs ont le droit d'ajouter des catégories aux jeux.

Pour pouvoir modifier les catégories d'un jeu, il faut qu'un administrateur aille sur la page du jeu qui doit être modifiée.

```
public function addCategorieToGame(int $idGame, int $idCategorie)
{
    $result = null;
    try {
        $this->psCheckIfGameHasCategorie->execute(array(
→':insert_idCategorie' => $idCategorie, ':insert_idGame' =>
→$idGame));
        $result = $this->psCheckIfGameHasCategorie->fetchAll();
    } catch (PDOException $e) {
        print "Erreur !: " . $e->getMessage() . "<br>";
        die();
    }
    if ($result == null) {
        try {
            $this->psAddCategorieToGame->execute(array(':insert_
→idCategorie' => $idCategorie, ':insert_idGame' => $idGame));
        } catch (PDOException $e) {
```

(suite sur la page suivante)

(suite de la page précédente)

```
        print "Erreur !: " . $e->getMessage() . "<br>";
        die();
    }
}
```

5.3.6 Ajout de catégories

La liste des catégories est définie par les administrateurs, ils ont la possibilité d'en ajouter depuis le menu d'administration. Pour ajouter une catégorie, il suffit de spécifier le nom que va porter la nouvelle catégorie.

```
public function addCategorie(string $categorieName)
{
    try {
        $this->psAddCategorie->execute(array(':categorie_name' => $categorieName));
        $result = $this->psAddCategorie->fetchAll();
    } catch (PDOException $e) {
        print "Erreur !: " . $e->getMessage() . "<br>";
        die();
    }
    return $result;
}
```

5.3.7 Téléchargement de Caiman

Le téléchargement de Caiman se fait sous forme de fichier .zip. Pour pouvoir télécharger l'application, il faut être authentifié. Actuellement, pour pouvoir modifier le fichier caiman.zip que l'utilisateur télécharge, il faut passer par le FTP. Le site web ne permet pas de le mettre à jour.

```
public function downloadCaiman()
{
    $zipFile = "../release/Caiman.zip";
    $file_name = basename($zipFile);
    header("Content-Type: application/zip");
    header("Content-Disposition: attachment; filename=$file_name");
    header("Content-Length: " . filesize($zipFile));
    readfile($zipFile);
    exit;
}
```

(suite sur la page suivante)

(suite de la page précédente)

{
}

5.3.8 Modification des jeux

Actuellement, il est possible de modifier le nom et la description d'un jeu.

5.4 Description technique : API

5.4.1 Téléchargement de jeu

Les jeux sont sous forme de fichier .iso dans le dossier “caimanWeb\games\”. Par contre, ce n'est pas tout pour simplifier, il faut bien séparer les jeux des différentes consoles. J'ai décidé de créer des dossiers par émulateurs. C'est pourquoi il va y avoir des sous-dossiers “GamecubeWii\” et “Playstation2\”. Chaque jeu est stocké avec le nom que l'administrateur lui aura donné quand il l'a ajouté depuis le site web.

Le chemin sur le serveur web pour accéder au fichier des jeux n'est pas public. Par conséquent, il a donc fallu que je trouve une solution pour que n'importe qui ne puisse pas télécharger un jeu. Pour ce faire, j'ai créé une route dans mon API qui prend en paramètres l'id du jeu et l'apiKey de l'utilisateur qui veut télécharger le jeu.

La fonction getURL(idGame, apiKey) permet de recevoir le lien de téléchargement pour un jeu. Avant de valider le téléchargement, l'API vérifie que l'apiKey qui lui a été donné est valide. Pour savoir si l'apiKey est valide, j'utilise la fonction DAOUser->Find(apikey) qui me retourne l'utilisateur en lien avec cette apiKey. Si l'apiKey est valide, je vais donc devoir reconstituer le chemin vers le dossier où est le jeu stocké.

Pour connaître le nom du dossier où se trouve le fichier, il faut savoir de quelle console est le jeu. Pour connaître cette information, je dois rechercher un jeu grâce à son id. Quand j'ai l'id de la console en lien avec le jeu, je dois encore faire une recherche pour savoir quel est son nom de dossier dans la base de données. Alors j'utilise la fonction DAOConsole->find(idConsole) pour connaître son nom de dossier.

Désormais que nous avons le dossier dans lequel se trouve le fichier du jeu, il faut maintenant connaître son nom de fichier. Pour cela, j'utilise la fonction DAOFile->find(file). l'id du fichier est connu grâce à l'appel la fonction DAOGame->find().

À présent que nous avons toutes les parties du chemin, il est possible de construire le chemin complet en concaténant toutes les parties. Ensuite pour renvoyer le jeu à l'utilisateur, j'utilise les fonctions fopen(path,’rb’) et fpassthru(fopen).

```
public function getURL(int $idGame, string $apikey)
{
```

(suite sur la page suivante)

(suite de la page précédente)

```

$user = $this->DAOUser->find($apikey);
$fullpath = "";
if (is_null($user)) {
    return ResponseController::notFoundResponse();
} else {
    $game = $this->DAOGame->find($idGame);
    $file = $this->DAOFile->find($game->idFile);
    $console = $this->DAOConsole->find($game->idConsole);
    $fullpath = "../../../../../caimanWeb/games/" . $console->
    ↪folderName . "/" . $file->filename;

    if (file_exists($fullpath)) {
        header('Content-Description: File Transfer');
        header('Content-Type: application/octet-stream');
        header('Content-Disposition: attachment; filename=' ._
    ↪. basename($fullpath));
        header('Content-Transfer-Encoding: binary');
        header('Expires: 0');
        header('Cache-Control: must-revalidate, post-
    ↪check=0, pre-check=0');
        header('Pragma: public');
        header('Content-Length: ' . filesize($fullpath));
        $fp = fopen($fullpath, 'rb');
        fpassthru($fp);
        exit;
    }
}

return ResponseController::successfulRequest($fullpath);
}

```

5.4.2 Téléchargement de sauvegarde

Les sauvegardes des utilisateurs sont stockées sous forme de fichiers .zip. L'intérêt d'utiliser des fichiers .zip est la taille et le fait qu'un fichier puisse contenir toutes les sauvegardes instantanément. Les fichiers de sauvegardes se trouvent dans le dossier “\CaimanWeb\saves\”.

Le nom attribué au fichier est décidé au premier envoi de sauvegarde, le nom est le md5 du microtime de l'heure d'envoie.

Pour recevoir le fichier, l'API possède une route qui prend les paramètres suivants :

- idEmulator
- iduser
- apikey

Cette route va envoyer le fichier demandé. Comme pour les fichiers des jeux, les sauvegardes sont dans un dossier privé du serveur. La fonction getURLsave(idEmulator, Iduser,apikey) se charge de renvoyer les fichiers à l'utilisateur. Pour faire ça il faut connaître le nom du fichier qui doit être

envoyé. Il faut donc utiliser la fonction DAOFileSave->find(idemulator, idUser) pour connaître les informations du fichier.

Désormais que le nom du fichier est connu, il est possible de retourner le fichier grâce à la fonction fopen(path,'rb') et fpassthru(fopen) pour renvoyer le fichier à l'utilisateur.

```
public function getURLSave(int $idEmulator, int $idUser, string
→$apikey)
{
    $user = $this->DAOUser->find($apikey);
    $fullpath = "";
    if (is_null($user)) {
        return ResponseController::notFoundResponse();
    } else {
        $fileSave = $this->DAOFileSave->find($idEmulator,
→$idUser);

        if (is_null($fileSave)) {
            return ResponseController::notFoundResponse();
            exit;
        }
        $file = $this->DAOFile->find($fileSave->idFile);
        $fullpath = "../../../../../caimanWeb/saves/" . $file->
→filename;

        if (file_exists($fullpath)) {
            header('Content-Description: File Transfer');
            header('Content-Type: application/zip');
            header('Content-Disposition: attachment; filename=' .
→. basename($fullpath));
            header('Content-Transfer-Encoding: binary');
            header('Expires: 0');
            header('Cache-Control: must-revalidate, post-
→check=0, pre-check=0');
            header('Pragma: public');
            header('Content-Length: ' . filesize($fullpath));
            $fp = fopen($fullpath, 'rb');
            fpassthru($fp);
            exit;
        }
    }

    return ResponseController::successfulRequest();
}
```

Dans l'état actuel de l'applicationCaiman, il est impossible de télécharger seulement la sauvegarde d'un seul jeu, la structure des sauvegardes des émulateurs ne laisse pas vraiment le choix.

5.4.3 Upload de sauvegarde

Pour enregistrer les sauvegardes, il y a une route dans l'API qui prend en paramètres :

- idEmulator
- iduser
- apikey
- file

Les fichiers de sauvegarde concernent un émulateur à chaque fois, donc il faut le spécifier à l'envoi. Il faut aussi spécifier l'utilisateur à qui appartiennent ces sauvegardes, il faut aussi transmettre une apiKey valide et finalement le fichier à envoyer.

Quand l'API reçoit ces informations, il faut déjà vérifier si l'apiKey reçu est bien valide. Si elle est valide, il faut ensuite vérifier si l'utilisateur a déjà créé une sauvegarde pour cet émulateur. Pour effectuer cette vérification, il faut utiliser la fonction DAOFileSave->find(idEmulator,idUser) qui retourne les informations du fichier s'il existe.

```
public function AddSave($idEmulator, $idUser, $apiKey, $file)
{
    if (is_null($idEmulator) || is_null($idUser) || is_null(
        $apiKey)) {
        return ResponseController::notFoundResponse();
        exit;
    }
    $user = $this->DAOUser->find($apiKey);

    if ($user == null) {
        return ResponseController::notFoundResponse();
        exit;
    }
    $isNewFile = false;
    $newfilename = $this->DAOFileSave->FindFileName($idEmulator,
        $idUser);

    if ($newfilename == null) {
        $newfilename = md5(microtime());
        $newfilename = $newfilename . ".zip";
        $isNewFile = true;
    }
    $target_dir = "../../../../../caimanWeb/saves/";

    if (move_uploaded_file($file, $target_dir . $newfilename)) {
        if ($isNewFile) {
            $this->DAOFileSave->AddFileSave($idEmulator,
                $idUser, $newfilename);
        }
    } else {
        return ResponseController::uploadFailed();
        exit;
    }
}
```

(suite sur la page suivante)

(suite de la page précédente)

```
        return ResponseController::successfulRequest();  
    }
```

Sauvegarde déjà présente

Si la sauvegarde d'un émulateur est déjà présente, il faut utiliser la fonction move_uploaded_file(file,target_dir). Tout d'abord il faut connaître le nom du fichier qui est attribué à la sauvegarde. Pour faire cela, il faut appeler la fonction DAOFileSave->findFilename(idEmulator,idUser) celle-ci va renvoyer le nom du fichier. Il suffit maintenant d'écraser le fichier présent sur le serveur par le nouveau.

```
public function findFileName(int $idEmulator, int $idUser)  
{  
  
    $statement = "  
SELECT f.filename filesave FROM `filesave` as fs  
    LEFT JOIN file as f  
    ON fs.idFile = f.id  
    WHERE idUser = :ID_user AND idEmulator = :ID_  
→emulator;";  
    try {  
        $statement = $this->db->prepare($statement);  
        $statement->bindParam(':ID_user', $idUser, \PDO::PARAM_  
→INT);  
        $statement->bindParam(':ID_emulator', $idEmulator, \  
→PDO::PARAM_INT);  
        $statement->execute();  
  
        $file = new File();  
  
        if ($statement->rowCount() == 1) {  
            $result = $statement->fetch(\PDO::FETCH_ASSOC);  
            $file->filename = $result['filesave'];  
        }  
        return $file->filename;  
    } catch (\PDOException $e) {  
        exit($e->getMessage());  
    }  
}
```

Sauvegarde pas encore présente

S'il n'y a aucune sauvegarde pour l'émulateur en question et l'utilisateur en question. Il faut commencer par donner un nom au fichier. Ce nom est constitué du MD5 du microtime() et de l'extension ".zip". Quand le nom pour le fichier a été créé, il faut uploader le fichier dans le dossier de sauvegarde grâce à la fonction move_uploaded_file(file,path).

Quand l'upload est validé, un ajout de fichier de sauvegarde va se faire grâce à la fonction DAOFileSave->AddFileSave(idEmulator,idUser,newfilename). Cette fonction va créer une entrée dans la base de données pour le nouveau fichier.

```
public function AddFileSave($idEmulator, $idUser, $newFileName)
{
    $statementFile = "
    INSERT INTO file
    (filename, dateUpdate)
    VALUES
    (:FILENAME, NOW());
    try {
        $statementFile = $this->db->prepare($statementFile);
        $statementFile->bindParam(':FILENAME', $newFileName, \
        \PDO::PARAM_STR);
        $statementFile->execute();
        //when create file is done
        $statementFileSave = "
        INSERT INTO filesave
        ($idUser, idEmulator,idFile)
        VALUES
        (:ID_user,:ID_emulator, :ID_file)";
        try {
            $statementFileSave = $this->db->prepare(
            \$statementFileSave);
            $statementFileSave->bindParam(':ID_user', $idUser, \
            \PDO::PARAM_INT);
            $statementFileSave->bindParam(':ID_emulator',
            \$idEmulator, \PDO::PARAM_INT);
            $lastInsertId = $this->db->lastInsertId();
            $statementFileSave->bindParam(':ID_file',
            \$lastInsertId, \PDO::PARAM_INT);
            $statementFileSave->execute();
        } catch (\PDOException $e) {
            exit($e->getMessage());
        }
    } catch (\PDOException $e) {
        exit($e->getMessage());
    }
}
```

5.4.4 Mise à jour du caimanToken

Le caimanToken sert à pouvoir se connecter sans mot de passe à l'application Caiman. Se token doit être changée à chaque fois que l'utilisateur se connecte, que ce soit en se connectant avec un mot de passe ou par le caimanToken justement.

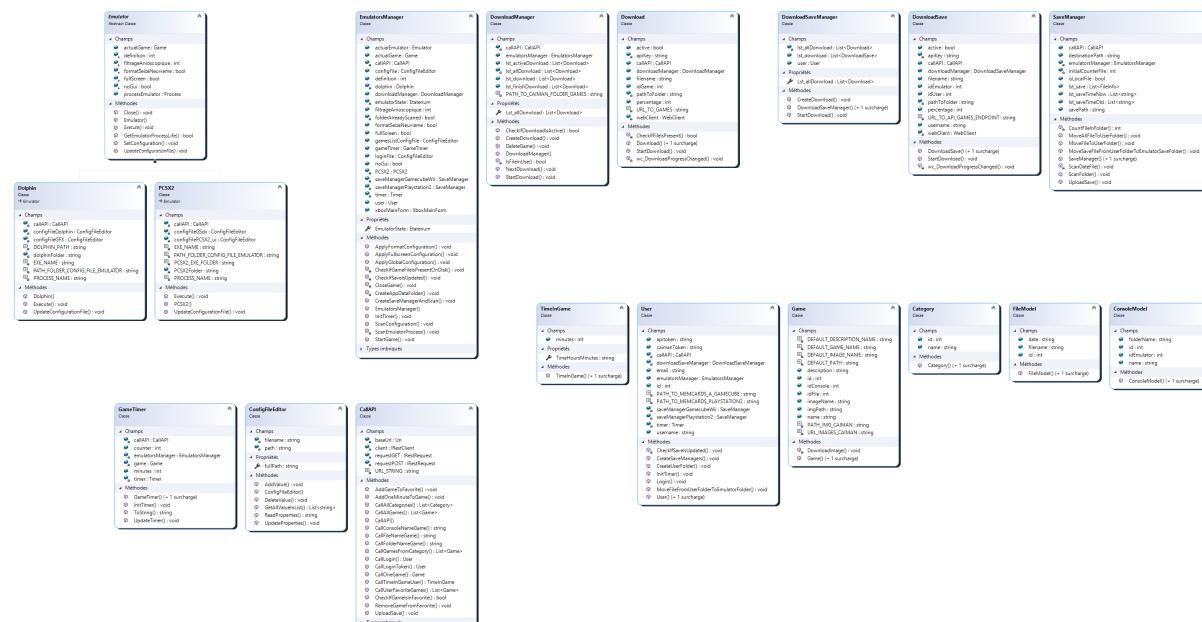
Pour ce faire, j'ai créé une fonction DAOUser->updateCaimanToken(apiToken) qui sert à modifier le caimanToken d'un utilisateur dans la base de données. Le caimanToken est un md5 du microtime actuel. Par conséquent, il est presque impossible que deux utilisateurs aient le même.

```
public function updateCaimanToken(string $apitocken)
{
    $statement = "
UPDATE user
SET caimanToken = :CAIMAN_TOKEN
WHERE apitocken = :API_TOCKEN;";

    try {
        $statement = $this->db->prepare($statement);
        $statement->bindParam(':API_TOCKEN', $apitocken);
        $caimanTocken = md5(microtime());
        $statement->bindParam(':CAIMAN_TOKEN', $caimanTocken);
        $statement->execute();
    } catch (\PDOException $e) {
        exit($e->getMessage());
    }
}
```

5.5 Description technique : Application Caiman

Diagramme de classe



5.5.1 Cration de dossier

Structure des dossiers

Caiman necessite de crer certains dossiers et fichiers pour pouvoir fonctionner correctement. Ces dossiers se trouvent dans le dossier “%appdata%\Roaming\Caiman”

 Caiman	28.05.2021 13:44	Dossier de fichiers
 img	28.05.2021 15:20	Dossier de fichiers
 users	28.05.2021 13:44	Dossier de fichiers

Caiman

Ce dossier contient les fichiers de configuration graphique, la liste des jeux telechargs et le dernier CaimanToken reu.

img

Ce dossier contient les differentes images des jeux, ces images sont telecharges sur le site internet de caiman.

users

Ce dossier contient les dossiers personnels de chaque utilisateur de l’application qui s’est connect sur le pc. Il possede les fichiers de sauvegarde de l’utilisateur en question.

Cration des dossiers

Dossier de base

Les dossiers pour caiman doivent obligatoirement tre cres. Donc au debut du lancement de l’application Caiman, il y a toujours une verification pour savoir si les dossiers sont bien prents. Si c’est le premier lancement de Caiman ou si les dossiers ont te supprims, ils vont tre cres.

```
private void CreateAppDataFolder()
{
    var appDataPath = Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData);
    var basePath = Path.Combine(appDataPath, @"Caiman\");
    var caimanConfigPath = Path.Combine(appDataPath, @
    @"Caiman\Caiman\");
    var imgPath = Path.Combine(appDataPath, @"Caiman\img\");
    var gamesPath = Path.Combine(appDataPath, @"Caiman\
    \Caiman\games.ini");
    var configPath = Path.Combine(appDataPath, @"Caiman\
    \Caiman\config.ini");
    var loginPath = Path.Combine(appDataPath, @"Caiman\
    \Caiman\login.ini");

    if (!Directory.Exists(basePath))
    {
```

(suite sur la page suivante)

(suite de la page précédente)

```
        Directory.CreateDirectory(basePath);
    }
    if (!Directory.Exists(imgPath))
    {
        Directory.CreateDirectory(imgPath);
    }
    if (!Directory.Exists(caimanConfigPath))
    {
        Directory.CreateDirectory(caimanConfigPath);
    }
    if (!Directory.Exists(@"C:\Caiman\Playstation2"))
    {
        Directory.CreateDirectory(@"C:\Caiman\Playstation2\
→");
    }
    if (!Directory.Exists(@"C:\Caiman\GamecubeWii\")) →
    {
        Directory.CreateDirectory(@"C:\Caiman\GamecubeWii\
→");
    }
    if (!File.Exists(loginPath))
    {
        using (StreamWriter sw = File.CreateText(loginPath))
        {
            sw.WriteLine("token = 0");
        }
        loginFile = new ConfigFileEditor(caimanConfigPath,
→"login.ini");
    }
    if (!File.Exists(gamesPath))
    {
        using (StreamWriter sw = File.CreateText(gamesPath))
        {

        }
        configFile = new ConfigFileEditor(caimanConfigPath,
→"config.ini");
    }
    if (!File.Exists(configPath))
    {
        //if the file do not exist create it with with the
→default emulators value
        using (StreamWriter sw = File.
→CreateText(configPath))
        {
            sw.WriteLine("configuration = 1080");
            sw.WriteLine("fullscreen = true");
            sw.WriteLine("definition = 3");
            sw.WriteLine("formatSeizeNeuvieme = true");
        }
    }
}
```

(suite sur la page suivante)

(suite de la page précédente)

```
        sw.WriteLine("filtrageAnioscopique = 3");
    }
    configFile = new ConfigFileEditor(caimanConfigPath,
→"config.ini");

}
```

Dossier des utilisateurs

La création des dossiers des utilisateurs pour gérer correctement les sauvegardes. Ces dossiers sont créés à la première connexion de l'utilisateur.

```
public void CreateUserFolder()
{
    var appDataPath = Environment.GetFolderPath(Environment.
→SpecialFolder.ApplicationData);
    var userPath = Path.Combine(appDataPath, @"Caiman\users\
→" + username + @"\");
    var savePath = Path.Combine(userPath, @"Save\");
    var savePathPlaystation = Path.Combine(savePath, @
→"Playstation2\");
    var savePathGamecubeWii = Path.Combine(savePath, @
→"GamecubeWii\");

    if (!Directory.Exists(userPath))
    {
        Directory.CreateDirectory(userPath);
    }
    if (!Directory.Exists(savePath))
    {
        Directory.CreateDirectory(savePath);
    }
    if (!Directory.Exists(savePathPlaystation))
    {
        Directory.CreateDirectory(savePathPlaystation);
    }
    if (!Directory.Exists(savePathGamecubeWii))
    {
        Directory.CreateDirectory(savePathGamecubeWii);
    }
}
```

5.5.2 Vérification des jeux présent sur le disque

Pour connaître les jeux qui ont été téléchargés, il existe un fichier comprenant les ids des jeux. Ce fichier se trouve dans le dossier “appdata” de Caiman. Il est donc commun à tous les utilisateurs. Ce fichier est mis à jour quand un utilisateur a fini de télécharger un jeu ou quand il décide d'en supprimer un.

```
private void CheckIfGameFileIsPresentOnDisk()
{
    List<string> lst_idGames = gamesListConfigFile.
    ↪ GetAllValueInList();

    foreach (var idGameString in lst_idGames)
    {
        if (idGameString != "")
        {
            int idGame = Convert.ToInt32(idGameString);
            if (!File.Exists(@"C:\Caiman\" + callAPI.
    ↪ CallFolderNameGame(idGame) + @"\" + callAPI.
    ↪ CallFileNameGame(idGame)))
            {
                gamesListConfigFile.
    ↪ DeleteValue(idGameString);
            }
        }
    }
}
```

Si l'utilisateur décide de supprimer un fichier sans passer par caiman il pourrait y avoir un souci alors pour pallier à ce problème une vérification est faite. Au lancement de l'application, si un fichier est manquant, alors le fichier qui contient les id des jeux sera mis à jour.

5.5.3 Paramètres graphique

Liste des paramètres graphique

L'utilisateur a la possibilité de modifier plusieurs paramètres :

La configuration global de l'émulateur

- Original
- 1080p
- 4K

Si le jeu doit se lancer en mode plein écran

- true
- false

Si le jeu doit se lancer en 16/9

- true
- false

Les différents modes graphiques modifient ces paramètres :

- le filtrage anisotropique
- l'upscale de la définition

Le filtrage anisotropique permet entre autres de diminuer l'effet de crénage, ainsi qu'à améliorer l'affichage des textures vue depuis un angle de vue extrême.

L'upscale de la définition permet d'augmenter la définition native du rendu. Par exemple, la définition de base de la Playstation 2 sur le jeu Kingdom Heart(512x448). Cette définition est donc la définition que l'utilisateur va avoir en "Original", mais s'il opte pour le paramètre 1080p, il va avoir une définition native de 1536x1344. En mode 4K le rendu sera en définition 4096x3584.

Fichiers de configuration PCSX2

Pour l'émulateur PCSX2, il y a deux fichiers de configuration qui doivent être modifiés.

Les fichiers se trouvent dans le dossier "PCSX2\inis\".

Le fichier "GSDX.ini" permet de modifier les paramètres de d'upscale et de filtrage anisotropique.

Le fichier "PCSX2_ui.ini" permet de modifier l'affichage en plein écran et le format d'affichage.

Le paramètre de format d'écran dans le fichier de configuration de PCSX n'est pas un booléen mais du texte je dois avant l'écriture dans le fichier le convertir.

Pareil pour le paramètre de mode plein écran

```
public override void UpdateConfigurationFile()
{
    configFileGSDx.UpdateProperties("upscale_multiplier", ↵
→definition.ToString());
    configFileGSDx.UpdateProperties("MaxAnisotropy", ↵
→filtrageAnioscopique.ToString());

    // the format is not true or false so i have to format ↵
→it
    if (fullScreen)
    {
        configFilePCSX2_ui.UpdateProperties(
→"DefaultToFullscreen", "enabled");
    }
    else
    {
        configFilePCSX2_ui.UpdateProperties(
→"DefaultToFullscreen", "disabled");
    }

    // the format is not true or false so i have to format ↵
→it
    if (formatSeizeNeuvieme)
    {
        configFilePCSX2_ui.UpdateProperties("AspectRatio",
→"16:9");
    }
}
```

(suite sur la page suivante)

(suite de la page précédente)

```
        }
    else
    {
        configFilePCSX2_ui.UpdateProperties("AspectRatio",
→"4:3");
    }
}
```

Fichiers de configuration Dolphin

Pour l'émulateur Dolphin, il y a deux fichiers de configuration qui doivent être modifiés.

Les fichiers se trouvent dans le dossier “Dolphin\User\Config”.

Le fichier “GFX.ini” permet de modifier les paramètres de d’upscale et de filtrage anisotropique.

Le fichier “Dolphin.ini” permet de modifier l'affichage en plein écran et le format d'affichage.

Le paramètre de format d'écran dans le fichier de configuration de Dolphin n'est pas un booléen mais un nombre. Donc avant l'écriture dans le fichier je dois le convertir.

```
public override void UpdateConfigurationFile()
{
    configFileDolphin.UpdateProperties("Fullscreen",_
→fullScreen.ToString());
    configFileGFX.UpdateProperties("InternalResolution",_
→definition.ToString());
    configFileGFX.UpdateProperties("MaxAnisotropy",_
→filtrageAnioscopique.ToString());

    if (formatSeizeNeuvieme)
    {
        configFileGFX.UpdateProperties("AspectRatio", "1");
    }
    else
    {
        configFileGFX.UpdateProperties("AspectRatio", "2");
    }
}
```

Sauvegarde des paramètres graphique

Les paramètres graphiques définis par l'utilisateur se trouvent dans le dossier “app-data\Caiman\Caiman\config.ini”.

Les paramètres sont mis à jour grâce à la classe ConfigFileEditor, cette classe permet la manipulation de fichier “.ini” que ce soit la lecture, la modification, la suppression, la création de propriétés.

```
public void ApplyGlobalConfiguration(string configuration)
{
    switch (configuration)
    {
        case "original":
            configFile.UpdateProperties("definition", "1");
            configFile.UpdateProperties(
                "filtrageAnioscopique", "1");
            configFile.UpdateProperties("configuration",
                "original");
            ScanConfiguration();
            break;
        case "1080":
            configFile.UpdateProperties("definition", "3");
            configFile.UpdateProperties(
                "filtrageAnioscopique", "4");
            configFile.UpdateProperties("configuration",
                "1080p");
            ScanConfiguration();
            break;
        case "4K":
            configFile.UpdateProperties("definition", "8");
            configFile.UpdateProperties(
                "filtrageAnioscopique", "4");
            configFile.UpdateProperties("configuration", "4K");
            ScanConfiguration();
            break;
        default:
            break;
    }
}
```

5.5.4 Téléchargement d'images

Le téléchargement des images se fait à la création d'un objet de type Game. Le téléchargement se fait à partir du site web de Caiman "caiman.cfpt.info". Les fichiers sont téléchargés dans le dossier appdata. L'intérêt de stocker les images dans ce dossier est de créer un système de cache qui sera accessible à tous les utilisateurs. Le téléchargement se fait grâce à un WebClient. Avant tout téléchargement de fichier, Caiman vérifie si le fichier n'est pas déjà existant, ce qui permet d'éviter de télécharger plusieurs fois la même image.

```
private void DownloadImage()
{
    if (!File.Exists(imgPath + imageName))
    {
        using (WebClient client = new WebClient())
        {
            client.DownloadFile(new Uri(URL_IMAGES_CAIMAN +_
→imageName), (imgPath + imageName));
        }
    }
}
```

5.5.5 Exécution de jeu

Choix du jeu à lancer

Le choix du jeu se fait à partir de la page de visualisation des détails d'un jeu. Quand l'utilisateur clique sur le bouton "Play" l'id du jeu qui doit être exécuté est envoyé à la fonction EmulatorManager.StartGame(idGame).

La fonction va ensuite faire un appel à l'API pour connaître de quelle console est le jeu qui doit être exécuté et selon la console un émulateur différent sera utilisé.

Emulateur embarqué avec Caiman

Caiman embarque deux émulateurs PCSX2 et Dolphin ces émulateurs servent respectivement à l'exécution de jeux de Playstation 2 et de Gamecube/Wii. L'exécution n'est pas la même selon l'émulateur, alors je vais détailler pour chacun. Avant l'exécution d'un jeu, l'application des paramètres graphiques est faite mais je ne vais pas le détailler ici.

Exécution avec PCSX2

En premier lieu, je vais recréer le chemin qui mène au fichier qui doit être exécuté par exemple “C :\Caiman\Playstation2\DRAGON_QUEST_VIII.iso”. Par la suite, je vais créer une variable pour les différents paramètres, dans la version de caiman actuel on ne peut pas changer ces paramètres se sera donc toujours “–nogui –portable”.

Le premier paramètre fait en sorte que l’émulateur n’affiche pas d’interface graphique, seule la fenêtre d’exécution du jeu est visible. Le deuxième paramètre sert à dire à PCSX2 que les fichiers de configurations qui doivent être utilisés sont ceux du dossier de l’émulateur, et non ceux du dossier créé pour chaque utilisateur windows par PCSX2.

Pour lancer l’exécution, je vais lancer le processus de PCSX2.exe avec en premier paramètre le chemin de l’iso à exécuter et après les paramètres cités précédemment.

```

public override void Execute(int idGame)
{
    string path = @"C:\Caiman\" + callAPI.
    ↪CallFolderNameGame(idGame) + @"\";
    string filename = callAPI.CallFileNameGame(idGame);
    UpdateConfigurationFile();
    int process = Process.GetProcessesByName(PROCESS_NAME).
    ↪Length;

    if (process == 0)
    {
        //param pour ne pas afficher l'interface graphique -
        ↪-portable --nogui
        string param = " --nogui --portable";

        processEmulator = Process.Start(PCSX2Folder + EXE_
        ↪NAME, path + filename + param);

    }
    else
    {
        Close();
        Execute(idGame);
    }
}

```

Exécution avec Dolphin

Ensuite, je vais recréer le chemin qui mène au fichier qui doit être exécuté par exemple “C :\Caiman\GamecubeWii\METROID_PRIME.iso”. Je vais également créer une variable pour les différents paramètres, dans la version de caiman actuel on ne peut pas changer ces paramètres, il sera toujours “–batch”.

Le paramètre “–batch” fait en sorte de ne pas afficher l’interface de Dolphin, et donc d’avoir seulement la fenêtre d’exécution du jeu.

Pour lancer l’exécution, je vais lancer le processus de PCSX2.exe avec en premier paramètre le chemin de l’iso à exécuter et après le paramètre cité précédemment.

```
public override void Execute(int idGame)
{
    string path = @"C:\Caiman\" + callAPI.
    ↪CallFolderNameGame(idGame) + @"\";
    string filename = callAPI.CallFileNameGame(idGame);

    UpdateConfigurationFile();
    int process = Process.GetProcessesByName(PROCESS_NAME).
    ↪Length;

    if (process == 0)
    {
        //param pour ne pas mettre de gui et en fullscreen -
        ↪-portable
        string param = " --batch";

        processEmulator = Process.Start(dolphinFolder + EXE_-
        ↪NAME, param + " --exec \"\" + path + filename);

    }
    else
    {
        Close();
        Execute(idGame);
    }
}
```

Information complémentaire

Il n'est pas possible de lancer un jeu alors qu'un autre est toujours en cours.

5.5.6 Affichage et calcul du temps de jeu

Le temps de jeu est comptabilisé dans la base de données à la minute près.

Affichage dans le détail d'un jeu

Quand un utilisateur de Caiman se rend sur la page de détails d'un jeu, il verra son nombre d'heures et de minutes de jeu. Pour récupérer cette information, je fais un appel à la base de données .Call-TimeInGameUser(idGame,idUser) cette appel va me rendre un objet TimeInGame qui va permettre de formater la réponse de l'API et afficher les heures et le minutes sous le format "10h10". Si l'utilisateur n'a pas jouer au jeu, il verra afficher "Time played : 00h00".

```
public string TimeHoursMinutes
{
    get
    {
        string time = "";
        int hours = this.minutes / 60;
        int minutesInt = this.minutes % 60;
        if (minutesInt == 60)
        {
            hours++;
            minutesInt = 0;
        }
        string minutesString = minutesInt.ToString();
        string hoursString = "";
        if (minutesInt < 10)
        {
            minutesString = "0" + minutesInt;
        }
        if (hours < 10)
        {
            hoursString = "0" + hours;
        }
        time = hoursString + " h " + minutesString;
        return time;
    }
}
```

Affichage du temps de jeu actuel

Quand un utilisateur lance un jeu, un objet GameTimer va être créé, cet objet sert à compter le temps de jeu de la session actuelle et à mettre à jour la base de données.

La classe GameTimer va initialiser un timer pour rafraîchir l'affichage de la navbar avec la valeur adéquate. Ce timer se rafraîchit toutes les secondes. Celui-ci va également appeler la fonction UpdateTimer, UpdateTimer va incrémenter le nombre de secondes de jeu et toutes les 60 secondes, il va faire un appel à l'API pour incrémenter le temps de jeu de l'utilisateur dans la base de données. L'incrémentation se fait donc toutes les minutes, cela permet d'être assez précis dans le décompte de temps de jeu.

```
public void InitTimer()
{
    timer = new Timer();
    timer.Tick += new EventHandler(UpdateTimer);

    timer.Interval = 1000;
    timer.Start();
}

public void UpdateTimer(object sender, EventArgs e)
{
    counter++;

    if (counter == 60)
    {
        minutes++;
        counter = 0;

        callAPI.AddOneMinuteToGame(game.id,
←emulatorsManager.user.id);
    }
}
```

L'affichage de temps de jeu se fait dans la navbar en haut à gauche, Les informations affichées sont les suivantes :

- Nom du jeu en cours
- heures et minutes de jeu de la session actuel sous le format 10m50

5.5.7 Téléchargement de jeux

Type de fichiers

Les fichiers utilisés par les émulateurs sont des fichiers .iso, que ce soit par PCSX2 ou Dolphin.

Choix du fichier à télécharger

Le fichier à télécharger se fait sur la page de détails d'un jeu, si le jeu n'a pas déjà été téléchargé. Un bouton "download" sera présent, ce bouton va envoyer l'id du jeu qui doit être téléchargé.

```
if (!File.Exists(gamePath))
{
    XboxButton btn_download = new XboxButton(
→"download", game.id, 0, 0);
    btn_download.Text = "Download: " + game.name;
    btn_download.Location = new System.Drawing.
→Point(500, 650);
    btn_download.Click += new System.
→EventHandler(bouton_Click);
    lstControls[rowCounter].Add(btn_download);
    Controls.Add(btn_download);
    rowCounter++;
    lstControls.Add(new List<Control>());
}
```

Ajout d'un jeu à la liste de téléchargement

Les téléchargements des jeux sont gérés par la classe "DownloadManager", cette classe contient 3 listes de téléchargement différentes :

- lst_download
- lst_activeDownload
- lst_finishDownload

La première contient les téléchargements en attente, la deuxième le téléchargement en cours et la troisième la liste des téléchargements terminés.

Quand un téléchargement est créé, il est automatiquement ajouté à la liste de téléchargement en attente.

Lancement d'un téléchargement

À l'ajout d'un téléchargement, la fonction StartDownload() est appelée. Cette fonction vérifie si dans la liste de téléchargement il sera lancé. Si aucun téléchargement n'est en cours, la fonction NextDownload() va être lancée.

La fonction NextDownload() sert à savoir si un téléchargement est en cours. Si un téléchargement est en cours, il sera déplacé dans la liste des téléchargements finis, si aucun téléchargement n'est en cours le premier téléchargement de la liste en attente est lancé

Téléchargement d'un fichier

Quand la fonction download.StartDownload() est lancée, le WebClient se lance également. Ce WebClient télécharge le fichier en appelant l'API de caiman en passant en paramètres l'id du jeu qui doit être téléchargé et l'apiKey de l'utilisateur.

```
public void StartDownload()
{
    if (!CheckIfFileIsPresent())
    {
        webClient = new WebClient();
        Uri uri = new Uri(URL_TO_GAMES + "?idGame=" + idGame +
        "&apiKey=" + apiKey);
        webClient.DownloadProgressChanged += wc_
        DownloadProgressChanged;
        webClient.DownloadFileAsync(uri, pathToFolder + "temp.
        " + filename);
        active = true;
    }
}
```

Le fichier sera téléchargé dans le dossier spécifique à l'émulateur, par exemple si un jeu pour l'émulateur PCSX2 est téléchargé alors le jeu sera téléchargé dans le dossier

"C :\Caiman\Playstation2". Par contre, le fichier n'est pas téléchargé avec le nom final, le préfix "temp." est ajouté. Le préfixe est ajouté pour éviter que si un utilisateur quitte Caiman pendant un téléchargement, il ne se retrouve pas avec un fichier incomplet au prochain lancement de l'application.

A la fin du téléchargement d'un fichier, la fonction DownloadManager.NextDownload() est appelée. L'id du jeu est ajouté à la liste des jeux téléchargés, et le fichier est renommé avec le nom correct.

```
public void NextDownload()
{
    if (lst_activeDownload.Count > 0)
    {
        lst_finishDownload.Add(lst_activeDownload[0]);
        lst_activeDownload.RemoveAt(0);
    }
}
```

(suite sur la page suivante)

(suite de la page précédente)

```

        }

        if (lst_download.Count () == 1)
        {
            lst_activeDownload.Add(lst_download[0]);
            lst_download.RemoveAt(0);
            StartDownload();
        }

    }
}

```

5.5.8 Synchronisation des sauvegardes

Structure des sauvegardes

Les fichiers de sauvegardes des émulateurs sont faits ainsi :

PCSX2 :

Les fichiers de sauvegardes pour PCSX2 sont 2 fichiers de 8MB, comme les memory cards étaient à l'époque de la Playstation 2. Ces deux fichiers contiennent les différentes sauvegardes des jeux.

emulators > PCSX2 > memcards			
Nom	Modifié le	Type	Taille
Mcd001.ps2	01.06.2021 08:12	Fichier PS2	8 448 Ko
Mcd002.ps2	01.06.2021 08:12	Fichier PS2	8 448 Ko

Dolphin :

Les fichiers de sauvegardes de l'émulateur Dolphin ne concernent qu'un seul jeu à la fois.

User > GC > EUR > Card A			
Nom	Modifié le	Type	Taille
01-G2MP-MetroidPrime2.gci	01.06.2021 08:12	Fichier GCI	25 Ko
01-G8MP-mariost_save_file.gci	01.06.2021 08:12	Fichier GCI	137 Ko
01-GM8P-MetroidPrime.gci	01.06.2021 08:12	Fichier GCI	25 Ko

La solution pour structurer les sauvegardes des différents utilisateurs est celle-ci :

Chaque utilisateur possède un dossier personnel dans le dossier appdata de Caiman. Ce dossier contient le fichier de configuration graphique de l'application et un dossier où ces sauvegardes sont présentes.

Le dossier de l'utilisateur lorenzo1227 :

Caiman > users > lorenzo1227				Rechercher dans : lorenzo1227
Nom	Modifié le	Type	Taille	
Save	30.05.2021 20:59	Dossier de fichiers		
config.ini	30.05.2021 20:46	Fichier source INI	0 Ko	

Le dossier contenant les sauvegardes de l'utilisateur lorenzo1227 :

Caiman > users > lorenzo1227 > Save				Rechercher dans : Save
Nom	Modifié le	Type	Taille	
GamecubeWii	01.06.2021 08:26	Dossier de fichiers		
Playstation2	01.06.2021 08:26	Dossier de fichiers		
GamecubeWii.zip	01.06.2021 08:26	Archive WinRAR ZIP	15 Ko	
Playstation2.zip	01.06.2021 08:26	Archive WinRAR ZIP	82 Ko	

Upload des sauvegardes

Pour pouvoir synchroniser les sauvegardes entre les différents PCs d'un utilisateur, j'ai commencé par savoir si l'un des fichiers présents dans les dossiers des différents émulateurs a été mis à jour. Pour ce faire, j'ai un timer qui va faire une vérification sur la dernière date de modification du fichier. S'il s'avère qu'une modification a été faite dans l'un des dossiers de sauvegardes, je copie l'intégralité du dossier de l'émulateur vers le dossier appdata de l'utilisateur.

```
private void ScanDateFile()
{
    int counter = 0;
    lst_saveTimeOld.Clear();
    lst_saveTimeOld.AddRange(lst_saveTimeNow);
    lst_saveTimeNow.Clear();
    //if a file has been added or delete sync the folder
    if (initialCounterFile != CountFileInFolder())
    {
        if (isLocalFile)
        {
            initialCounterFile = CountFileInFolder();
            MoveAllFileToUserFolder();
        }
    }
    foreach (FileInfo save in lst_save)
    {

        save.Refresh();

        lst_saveTimeNow.Add(save.LastWriteTime.Ticks.
        ↪ToString());
    }
}
```

(suite sur la page suivante)

(suite de la page précédente)

```

if (lst_saveTimeOld.Count () > 0)
{
    try
    {
        if (lst_saveTimeOld[counter] != lst_
→saveTimeNow[counter])
        {
            //if the file has been modified since_
→the last time move or sync the folder
            if (lst_saveTimeOld[counter] != "")
            {
                if (isLocalFile)
                {
                    MoveFileToUserFolder(save);
                }
                else
                {
                    UploadSave();
                }
            }
            counter++;
        }
        catch { }
    }

}
}

public void MoveAllFileToUserFolder()
{
    DirectoryInfo d = new DirectoryInfo(savePath);
    FileInfo[] Files = d.GetFiles();
    foreach (FileInfo file in Files)
    {
        try
        {

            File.Copy(file.FullName, Path.
→Combine(destinationPath, file.Name), true);
        }
        catch (Exception)
        {
        }
    }
}

```

Si je trouve une différence alors cela veut dire que le fichier a été modifié. Quand un fichier a été

modifié je crée une copie de ce fichier dans le dossier “username/save/nom_de_emulateur/”.

Quand un fichier a été mis à jour, supprimé ou créé dans le dossier de sauvegarde de l’utilisateur, je zip les fichiers présents dans le dossier pour les envoyer sur le serveur de Caiman.

```
public void UploadSave()
{
    var appDataPath = Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData);
    var savePathZipDolphin = "";
    var savePathZipPCSX2 = "";
    var savePath = Path.Combine(appDataPath, @"Caiman\users\" + emulatorsManager.user.username + @"\Save\");
    savePathZipDolphin = Path.Combine(appDataPath, @"Caiman\users\" + emulatorsManager.user.username + @"\Save\GamecubeWii\");
    savePathZipPCSX2 = Path.Combine(appDataPath, @"Caiman\users\" + emulatorsManager.user.username + @"\Save\Playstation2\");
    //zip the save of the emulators
    ZipFile.CreateFromDirectory(savePathZipPCSX2, savePath + "tempPCSX2.zip");
    ZipFile.CreateFromDirectory(savePathZipDolphin, savePath + "tempDolphin.zip");
    //call the api to create a copy of the save to the Bunker
    callAPI.UploadSave(1, emulatorsManager.user.id, emulatorsManager.user.apitoken, savePath + "tempDolphin.zip");
    callAPI.UploadSave(2, emulatorsManager.user.id, emulatorsManager.user.apitoken, savePath + "tempPCSX2.zip");
    //delete the temporary zip file
    File.Delete(savePath + "tempPCSX2.zip");
    File.Delete(savePath + "tempDolphin.zip");

}
```

Download des sauvegardes présente sur le serveur de Caiman

Le lancement du téléchargement des sauvegardes est lancé à la connexion de l’utilisateur. Un appel à l’API est fait pour les deux émulateurs :

```
public void CreateDownload(int idEmulator, string apiKey)
{
    var appDataPath = Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData);
    var savePath = Path.Combine(appDataPath, @"Caiman\users\" + user.username + @"\Save\");
```

(suite sur la page suivante)

(suite de la page précédente)

```

if (lst_download == null)
{
    lst_download = new List<DownloadSave>();
}
lst_download.Add(new DownloadSave(savePath, idEmulator,
→user.id, apiKey,user.username,this));
}

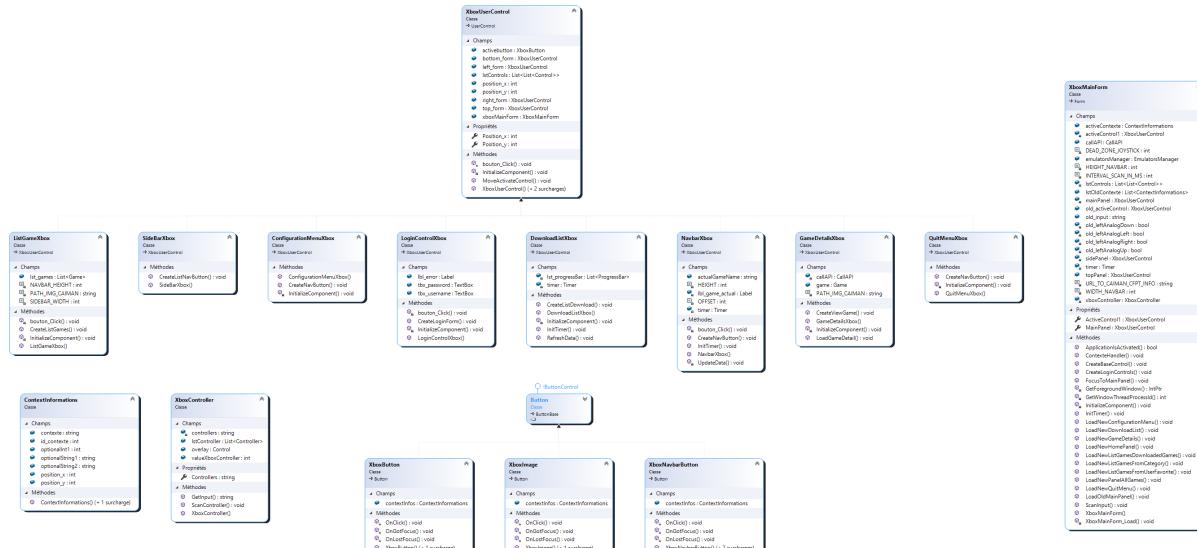
```

Ces appels à l'API vont télécharger des fichiers .zip contenant les sauvegardes des différents émulateurs. Ces deux fichiers vont être décompressé dans le dossier spécifique de chaque émulateur. Avant de pouvoir décompresser le dossier, je dois au préalable supprimer le contenu des dossiers de destination (Si je dois supprimer le contenu des dossiers c'est parce que la classe Zipfile de C# sous la version 5.0 ne peut pas "override" le contenu d'un dossier).

Quand les fichiers zip sont décompressés le contenu est envoyé dans les dossiers des différents émulateurs.

5.6 Description technique : Interface graphique

Diagramme de classe



5.6.1 Organisation des boutons

Pour faire en sorte de pouvoir se déplacer de bouton en bouton, il m'a fallu trouver une stratégie pour organiser les boutons. Pour structurer la liste des boutons, j'ai finalement décidé de créer une liste de liste de boutons. La liste principale sert de rangé et la "sous liste" sert de colonne. Cela permet de connaître plus aisément l'emplacement des boutons qu'une simple liste à une seule dimension.

5.6.2 Gestion des inputs

Interaction avec les manettes

L'application Caiman prend en charge les manettes reconnues nativement par windows (Xinput). Pour pouvoir communiquer avec la manette, j'ai utilisé le paquet NuGet "SharpDX", il me permet de connaître les informations et les inputs des manette connectés au pc.

Pour simplifier l'interaction avec la manette de l'utilisateur, j'ai créé une classe "XboxController.cs", Cette classe me permet de connaître les manettes connectées et de savoir à un instant T les inputs de chaque manette.

Transformation d'inputs en événement.

Pour simplifier la navigation, seule la manette une a le droit de se déplacer dans l'application de Caiman. Pour connaître les inputs, je les rafraîchis toutes les 10ms, cela permet de ne pas avoir de latence ou de louper des inputs.

Gestion des boutons

Quand la fonction ScanInput reçoit un string contenant les inputs de la manette 1. Ce string est envoyé dans un switch qui va faire une comparaison entre l'input précédent de la manette et l'input actuel. Si l'input actuel est différent de l'ancien alors cela veut dire que l'utilisateur a pressé un bouton puis il l'a soit relâché soit fait une autre combinaison de touches.

Le code suivant n'est pas complet !

```
public void ScanInput(object sender, EventArgs e)
{
    // if the user 1 controller is connected
    if (xboxController.lstController[0].IsConnected)
    {
        string input = xboxController.lstController[0].
        ↪GetState().Gamepad.Buttons.ToString();
        int inputAnalogLeftX = xboxController.
        ↪lstController[0].GetState().Gamepad.LeftThumbX;
        int inputAnalogLeftY = xboxController.
        ↪lstController[0].GetState().Gamepad.LeftThumbY;
```

```
if (input == "A" && old_input != "A")
{
    SendKeys.Send("{ENTER}");
}
```

Gestion du joystick gauche

La réception des inputs pour le joystick n'est pas comme pour les boutons, en sachant que les joysticks sont analogiques, je reçois une valeur entre -32000 et +32000. Je vais recevoir ces valeurs pour l'axe X et l'axe Y. Pour savoir si l'utilisateur a poussé le joystick dans une direction, j'ai défini une "deadzone" dans laquelle je considère qu'"aucune direction n'est définie par l'utilisateur de la manette. J'ai défini cette zone à 20000, j'ai fait des tests pour savoir où je trouvais que la "deadzone" devait s'arrêter. Puis j'ai fait comme pour les boutons, j'ai fait une comparaison avec la valeur précédente pour savoir si le joystick vient d'être poussé.

```
bool leftAnalogUp = false;
bool leftAnalogDown = false;
bool leftAnalogLeft = false;
bool leftAnalogRight = false;

if (inputAnalogLeftX > DEAD_ZONE_JOYSTICK)
{
    leftAnalogRight = true;
}

if (inputAnalogLeftX < -DEAD_ZONE_JOYSTICK)
{
    leftAnalogLeft = true;
}

if (inputAnalogLeftY > DEAD_ZONE_JOYSTICK)
{
    leftAnalogUp = true;
}

if (inputAnalogLeftY < -DEAD_ZONE_JOYSTICK)
{
    leftAnalogDown = true;
}
```

5.6.3 Découpage de l'interface

L'interface de l'application Caiman est divisée en trois parties hormis pour la connexion.

Navbar

La première est la navbar qui contient les différents boutons de navigation et les informations sur le jeu en cours. Tous les boutons sont créés dynamiquement, le placement s'adapte à l'écran principal de l'utilisateur, mais s'il a un écran vraiment trop petit, les informations sur le jeu en cours et les boutons risquent de s'entremêler.

Pour afficher le jeu en cours, le navigateur est obligé de se rafraîchir et de se redessiner toutes les secondes pour afficher le temps de jeu. Pour savoir si un jeu est en cours, la navbar regarde si le GameTimer est lancé. Si le GameTimer est lancé c'est qu'un jeu est en cours, dans ce cas il va afficher le nom et le temps de jeu.

```
private void UpdateData(object sender, EventArgs e)
{
    if (xboxMainForm.emulatorsManager.gameTimer != null)
    {
        if (actualGameName != "")
        {
            lbl_game_actual.Text = " Now playing: " +_
actualGameName + " " + xboxMainForm.emulatorsManager.gameTimer.-
ToString();
        }
        else
        {
            lbl_game_actual.Text = "";
        }
    }
    else
    {
        lbl_game_actual.Text = "";
    }
}
```

Sidebar

La sidebar contient des boutons qui permettent d'afficher différentes listes de jeux. Ces boutons sont en partie créés dynamiquement et en partie reçus de l'API. Les premiers boutons sont ceux qui concernent l'utilisateur, on va les avoir dans l'ordre :

- Downloaded games
- Favorites games
- All games

Ces catégories ne sont pas créées dynamiquement.

Par contre, les suivants sont reçus de l'API, ils concernent les différentes catégories disponibles pour Caiman. À l'appui de ces boutons, la liste des jeux va être chargée et le main panel va afficher la liste des jeux reçus par l'API.

```

public void CreateListNavButton()
{
    List<string> lst-navbar = new List<string>();
    List<Category> lst-category = xboxMainForm.callAPI.
→CallAllCategories();
    lst-navbar.Add("Downloaded games");
    lst-navbar.Add("Favorites games");
    lst-navbar.Add("All games");

    foreach (var item in lst-category)
    {
        lst-navbar.Add(item.name);
    }
    for (int i = 0; i < lst-navbar.Count; i++)
    {
        lstControls.Add(new List<Control>());
        lstControls[i].Add(new XboxButton());
    }

    //update the buttons infos
    for (int a_row = 0; a_row <= (lst-navbar.Count -1); a_
→row++)
    {

        List<string> lstString = new List<string>();
        XboxButton tempButton = new XboxButton("side", a_
→row, a_row, 0);
        lstControls[a_row][0] = tempButton;
        lstControls[a_row][0].Text = lst-navbar[a_row];
        lstControls[a_row][0].Location = new System.Drawing.
→Point(0 * 100 + 20, a_row * 75 + 15);
        lstControls[a_row][0].Width = 200;
        lstControls[a_row][0].Height = 50;
        lstControls[a_row][0].Name = "btn_" + lst-navbar[a_
→row];

        Controls.Add(lstControls[a_row][0]);
        lstControls[a_row][0].Click += new System.
→EventHandler(bouton_Click);

    }

    //set the action of button
    XboxButton downloadedGames =_
→(XboxButton)lstControls[0][0];

```

(suite sur la page suivante)

(suite de la page précédente)

```
downloadedGames.contextInfos.contexte = "downloadedGames";
};

XboxButton userFavoritesGames =
(XboxButton)lstControls[1][0];
userFavoritesGames.contextInfos.contexte = "favorite";
XboxButton allGames = (XboxButton)lstControls[2][0];
allGames.contextInfos.contexte = "home";

for (int i = 0; i < lst_category.Count; i++)
{
    XboxButton tempButton =
(XboxButton)lstControls[(i+3)][0];
    tempButton.contextInfos.contexte = "category";
    tempButton.contextInfos.id_contexte = (lst_
category[i].id);
}

}
```

Main panel

Néanmoins, le main panel ne peut pas vraiment être décrit, c'est un panel contextuel qui va être défini par les besoins de l'utilisateur.

5.6.4 Affichage d'une liste de jeu

Pour l'affichage des listes des jeux, j'ai essayé de faire en sorte que le panel soit le plus flexible possible. le panel demande juste une liste de jeux, il n'y a pas de panel fait pour les jeux favoris ou les jeux déjà présents sur le disque.

Le panel utilise la fonction CreateListGames() pour dessiner les différents boutons de lien vers les détails des jeux. Pour que certaines images ne dépasse pas l'écran et qu'elles soient coupées, la fonction calcule le nombre d'images qu'elle peut afficher par ligne. Le calcul se fait selon la définition de l'écran de l'utilisateur. Les images affichées sont celles qui sont dans le dossier appdata de Caiman.

```
public void CreateListGames()
{
    string imgPath = Path.Combine(Environment.
GetFolderPath(Environment.SpecialFolder.ApplicationData), PATH_
IMG_CAIMAN);
    XboxImage tempXboxImage = new XboxImage();
```

(suite sur la page suivante)

(suite de la page précédente)

```

int max_rank = Width / (315);
int tempPos_x = 0;
int tempPos_y = 0;
lstControls.Add(new List<Control>());

foreach (var game in lst_games)
{
    if (tempPos_x == max_rank)
    {
        lstControls.Add(new List<Control>());
        tempPos_y++;
        tempPos_x = 0;
    }

    lstControls[tempPos_y].Add(new XboxImage());
    Image img = new Bitmap((imgPath+ game.imageName));
    XboxImage tempButton = new XboxImage("game", img,
    ↪game.id, tempPos_x, tempPos_y);
    lstControls[tempPos_y][tempPos_x] = tempButton;
    lstControls[tempPos_y][tempPos_x].Location = new_
    ↪System.Drawing.Point(tempPos_x * 300 + 15, tempPos_y * 430 + 15);

    Controls.Add(lstControls[tempPos_y][tempPos_x]);
    tempButton.Click += new System.EventHandler(bouton_
    ↪Click);

    tempPos_x++;
}
}

```

5.6.5 Gestion des erreurs de déplacement

Le déplacement à la manette se fait par déplacement haut,bas,gauche,droite. Donc, J'ai décidé de faire en sorte que si l'utilisateur se déplace dans une case qui n'existe pas, ils soient soit déplacés dans l'une des cases disponibles proche de la case demandée, autrement rien ne se passe.

Pour cela, j'ai dû décider arbitrairement ce qui se passait. si un mouvement illégal était fait. J'ai essayé de reproduire les déplacements utilisés dans l'interface de la console Xbox. Il y a certains cas qui ne sont pas forcément bien gérés. Il serait possible de faire un déplacement en calculant la position actuelle et en se basant sur la position en pixel. Cette alternative pourrait être intéressante.

```

public void MoveActivateControl(string destination = "")
{
    //top = 1
}

```

(suite sur la page suivante)

(suite de la page précédente)

```
//right = 2
//down = 3
//left = 4
if (destination == "down")
{
    if (lstControls[position_y] [position_x] == null)
    {
        int x = position_x;
        int y = position_y;

        int y_right_not_disponible;
        while (x < lstControls[position_y].Count () && ↴
lstControls[position_y] [x] == null)
        {
            if (x > lstControls[position_x-1].Count ())
            {
                break;
            }
            x++;
        }
        if (x == lstControls[position_y].Count ())
        {
            y_right_not_disponible = y;

            while (y_right_not_disponible>0)
            {
                y_right_not_disponible--;
                if (lstControls[y_right_not_ ↴
disponible] [position_x] != null)
                {
                    position_y = y_right_not_disponible;
                    lstControls[ (position_y) ] [position_ ↴
x] .Focus ();
                    return;
                }
            }
        }
        lstControls[ (position_y) ] [x] .Focus ();
        Position_x = x;
    }
    else
    {
        lstControls[ (position_y) ] [position_x] .Focus ();
    }
}
else
{
    if (position_x < lstControls[position_y].Count ())
    {
```

(suite sur la page suivante)

(suite de la page précédente)

```
    if (lstControls[position_y] [position_x] == null)
    {
        int x = position_x - 1;
        int y = position_y;
        while (lstControls[position_y] [x] == null)
        {
            if (x < 0)
            {

                break;
            }
            x--;
        }
        lstControls[ (position_y) ] [x].Focus();
        Position_x = x;
    }
    else
    {
        lstControls[ (position_y) ] [position_x] .
→Focus();
    }
}
else {
    position_y--;
}
}
```

5.6.6 Passage de la souris à la manette

Durant mes tests de l'application, je me suis rendu compte que le fait de passer de la manette à la souris et inversement, posait un problème. Quand je passais de l'un à l'autre la position du curseur n'était pas sauvegardée. Quand je passais de la souris à la manette, le curseur se retrouvait à une position erronée. Pour pouvoir pallier au problème, chaque bouton a une position définie et quand l'utilisateur clique dessus ,un événement se déclenche et notifie la forme que le bouton actuel a changé.

```
protected override void OnClick(EventArgs e)
{
    base.OnClick(e);

    //tell to the topMainForm which control is active
    XboxUserControl xboxUserControl = (XboxUserControl)this.
    ↪Parent;

    if (xboxUserControl != null)
```

(suite sur la page suivante)

(suite de la page précédente)

```
    {
        xboxUserControl.position_x = this.contextInfos.position_
    ↵x;
        xboxUserControl.position_y = this.contextInfos.position_
    ↵y;
    }

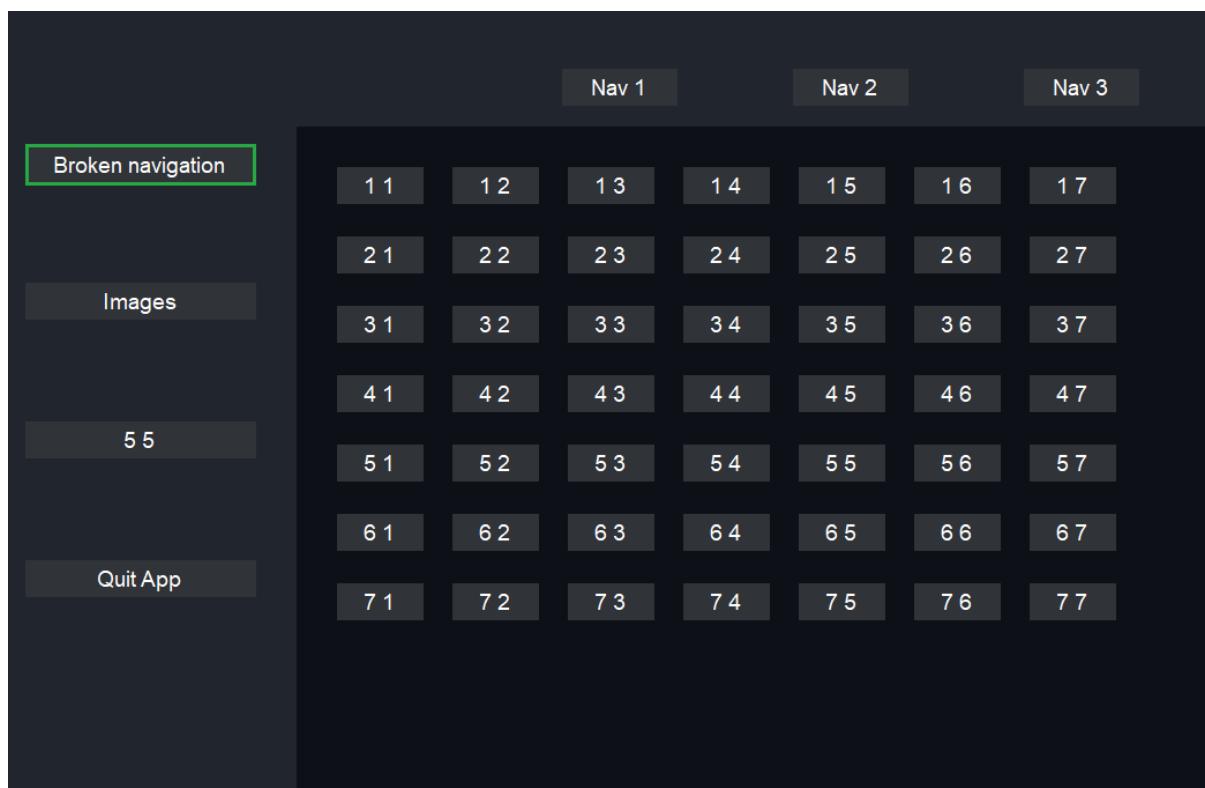
    if ((XboxMainForm)this.TopLevelControl != null)
    {
        XboxMainForm top MainForm = (XboxMainForm)this.
    ↵TopLevelControl;
        top MainForm.ActiveControl1 = xboxUserControl;
    }
}
```

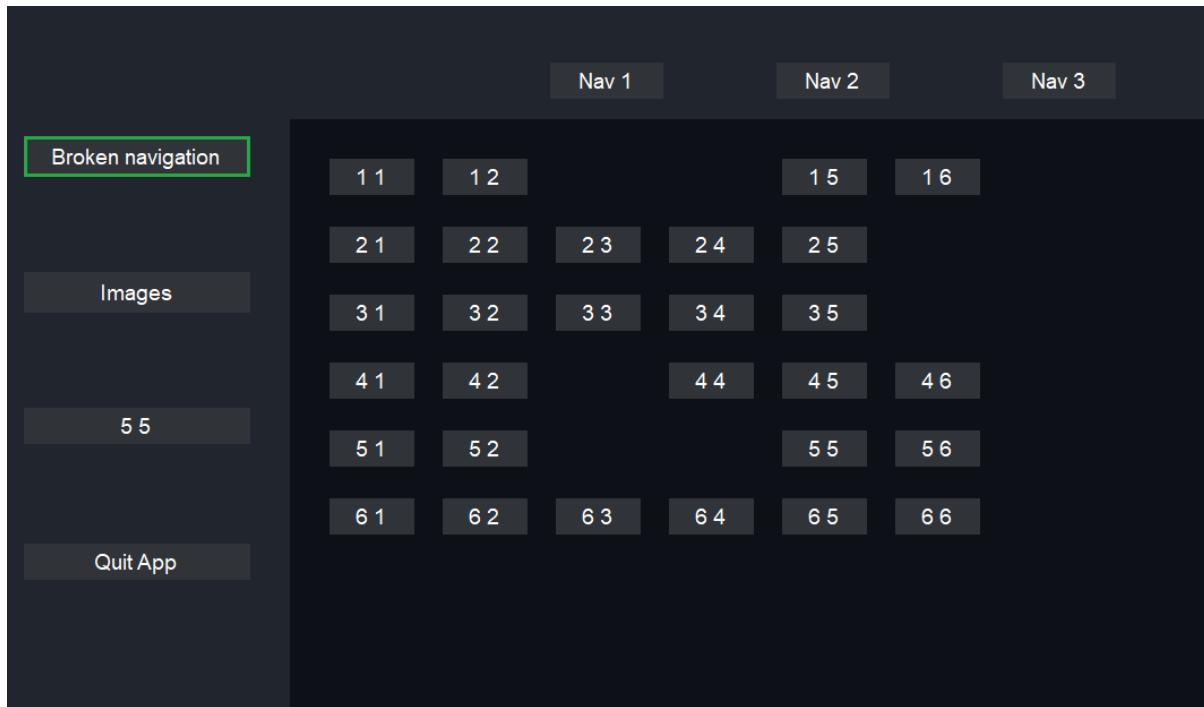
CHAPITRE 6

Tests

6.1 Test de l'interface graphique

Pour pouvoir tester mon interface graphique, j'ai créé un projet contenant des différentes vues pour pouvoir essayer le déplacement et le changement de vue.





Ce programme de test m'a permis de me rendre compte des différentes difficultés que j'ai pu avoir, ainsi que de pouvoir les corriger avant de passer à la partie développement de l'application.

Le but de ces tests étaient de ne pas avoir à me soucier de l'interface pendant que je devais développer la gestion des jeux et des émulateurs.

6.2 Test de Caiman

6.2.1 Test globaux

Pour pouvoir faire des tests, j'ai d'abord pensé à utiliser les tests de Visual Studio, malheureusement cela n'est pas vraiment applicable à Caiman. Alors, je n'ai donc testé que quelques classes de cette manière. Pour quand même essayer de trouver le plus de bogues possibles, j'ai demandé à plusieurs personnes de mon entourage d'essayer mon application pour me faire remonter les différents soucis qu'ils ont pu avoir.

Grâce aux différents retours, j'ai pu connaître un bon nombre de bogues liés à l'interface utilisable à la manettes et aux différents soucis d'ergonomie de Caiman. J'ai pu fournir à différents moments du développement des versions Alpha de Caiman, cela me permettait d'avoir des retours le plus régulièrement possible pour éviter d'avoir trop de contraintes à régler à la fin du développement.

6.2.2 Test synchronisation des sauvegardes

Pour tester la synchronisation des sauvegardes, j'ai créé un "protocole". Ce protocole consiste à utiliser les sauvegardes du jeu "Metroid Prime" pour tester les sauvegardes des jeux de gamecube. Metroid Prime nous laisse créer un plusieurs fichiers de sauvegarde directement au lancement du jeu, je me sers donc de ces fichiers pour savoir si la sauvegarde est bien présente.

Pour tester les sauvegardes des jeux de Playstation 2, j'ai utilisé le jeu Dragon Quest : VIII, il a l'avantage de pouvoir créer un fichier de sauvegarde avec un nom spécifique. Donc, J'ai créé un fichier de sauvegarde avec mon prénom et un autre avec un nom complètement étrange. Si je ne trouve pas les sauvegardes quand je lance le jeu, alors je sais que la synchronisation n'a pas marché.

CHAPITRE 7

Planning

7.1 Planning prévisionnel

Lorenzo Bauduccio

Caiman planning V1.0

Planning prévisionnelle Caiman 15/03/2021

Jour	SPRINT 3	SPRINT 3	18/05/2021	19/05/2021	20/05/2021	21/05/2021	24/05/2021	25/05/2021	26/05/2021	27/05/2021	28/05/2021	31/05/2021	SPRINT 4	01/06/2021	02/06/2021	03/06/2021	04/06/2021	05/06/2021	06/06/2021	07/06/2021	08/06/2021	09/06/2021	10/06/2021	11/06/2021	
Général																									
Création du projet																									
Analysé																									
Design de la base de données																									
Installation du Bunker																									
Installation de la machine de développement																									
Configuration de Dolphin																									
Installation de l'émulateur																									
Configuration des manettes et des inputs																									
Gestion de l'iso qui est exécuté																									
Gestion des paramètres graphiques																									
Configuration de PSX2																									
Installation de l'émulateur																									
Configuration des manettes et des inputs																									
Gestion de l'iso qui est exécuté																									
Gestion des paramètres graphiques																									
Caiman																									
Création de compte																									
Connexion utilisateur																									
Execution d'un jeu																									
Ajout de jeux en favoris																									
Modification des paramètres d'émulation																									
Execution d'un jeu																									
Choix de l'utilisateur à utiliser																									
Application des paramètres d'émulation																									
Téléchargement de jeu																									
Vérification des jeux qui sont présent sur le pc																									
Téléchargement des jeux vers le pc																									
Affichage des téléchargements/opération																									
Synchronisation des sauvegardes																									
Comparaison des sauvegardes entre le pc utilisateur et le Bunker																									
Téléchargement des sauvegardes vers Caiman																									
Téléchargement des sauvegardes vers le Bunker																									
Site web																									
Création du site																									
Création de compte																									
Création interface																									
Affichage des info utilisateur																									
Interface																									
Création d'une interface																									
Utilisation de la maquette pour contrôler Caiman																									
Autres :																									
Test / résolution des bogues																									
Documentation																									
Journal de bord																									
Rapport																									

Planning: Caiman

	19/04/2021	20/04/2021	21/04/2021	22/04/2021	23/04/2021	24/04/2021	25/04/2021	26/04/2021	27/04/2021	28/04/2021	29/04/2021	30/04/2021	SPRINT 2	03/05/2021	04/05/2021	05/05/2021	06/05/2021	07/05/2021	08/05/2021	09/05/2021	10/05/2021	11/05/2021	12/05/2021	13/05/2021	14/05/2021	17/05/2021
Général:																										
Création du projet.																										
Analysé																										
Design de la base de données																										
Installation du Bunker																										
Installation de la machine de développement																										
Configuration de Dolphin																										
Installation de l'emulateur																										
Configuration des manettes et des inputs																										
Gestion de l'iso qui est exécuté																										
Gestion des paramètres graphiques																										
Configuration de PSX2																										
Installation de l'emulateur																										
Configuration des manettes et des inputs																										
Gestion de l'iso qui est exécuté																										
Gestion des paramètres graphiques																										
Caiman																										
Création de compte																										
Connexion utilisateur																										
Execution d'un jeu																										
Ajout de jeux en favoris																										
Modification des paramètres d'émulation																										
Execution d'un jeu																										
Choix de l'emulateur à utiliser																										
Application des paramètres d'émulation																										
Téléchargement de jeu																										
Vérification des jeux qui sont présent sur le pc																										
Téléchargement des jeux vers le pc																										
Affichage des téléchargements/gestion																										
Synchronisation des sauvegardes																										
Comparaison des sauvegardes entre le pc utilisateur et le Bunker																										
Téléchargement des sauvegardes vers Caiman																										
Téléchargement des sauvegardes vers le Bunker																										
Site web																										
Création du site																										
Création de compte																										
Création interface																										
Affichage des infos utilisateur																										
Téléchargement de Caiman																										
Interface																										
Création d'une interface																										
Utilisation de la manette pour contrôler Caiman																										
Autres :																										
Test / résolution des bogues																										
Documentation																										
Journal de bord																										
Rapport																										

Début sprint 1 : 10/04/2021
 Durée sprint 1 : 10 jours
 Date fin sprint 1 : 17/04/2021
 Début sprint 2 : 17/04/2021
 Durée sprint 2 : 10 jours
 Date fin sprint 2 : 27/04/2021

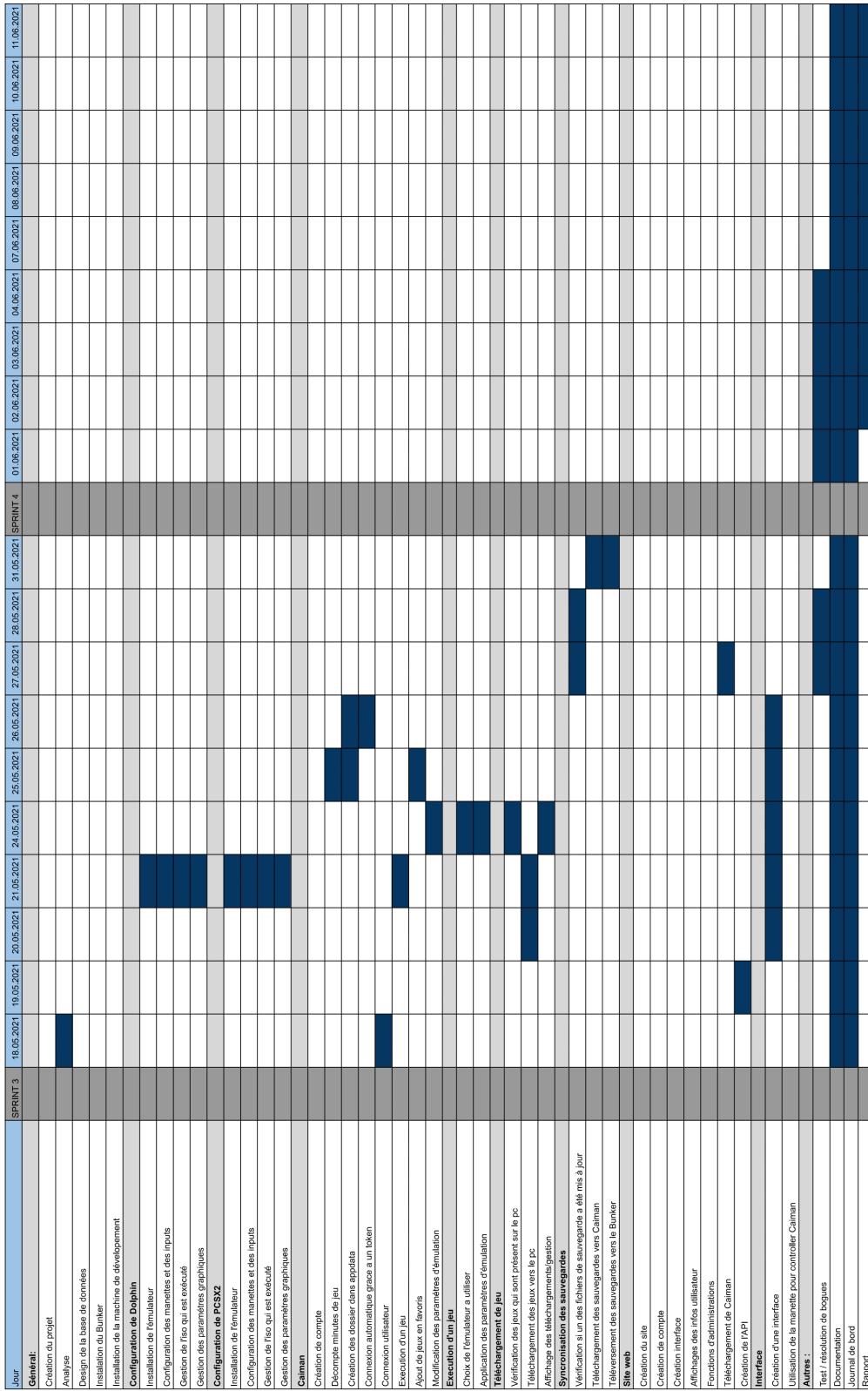
Release sprint 1 : 10/04/2021
 Release sprint 2 : 17/04/2021

7.2 Planning effectif

Lorenzo Bauduccio

Caiman planning V1.0

Planning effectif Caiman



Planning effectif Caiman

Caiman planning V1.0

Lorenzo Bauduccio

Jour	SPRINT 1	19.04.2021	20.04.2021	21.04.2021	22.04.2021	23.04.2021	26.04.2021	27.04.2021	28.04.2021	29.04.2021	30.04.2021	SPRINT 2	03.05.2021	04.05.2021	05.05.2021	06.05.2021	07.05.2021	10.05.2021	11.05.2021	12.05.2021	14.05.2021	17.05.2021		
Général:																								
Création du projet																								
Analysse																								
Design de la base de données																								
Installation du Bunker																								
Installation de la machine de développement																								
Configuration de Dolphin																								
Installation de l'hémuleur																								
Configuration des manettes et des inputs																								
Gestion de l'iso qui est toxique																								
Gestion des paramètres graphiques																								
Configuration de PSX2																								
Installation de l'hémuleur																								
Configuration des manettes et des inputs																								
Gestion de l'iso qui est exécutable																								
Gestion des paramètres graphiques																								
Caiman																								
Création de compte																								
Décompte minutes de jeu																								
Création des dossier dans appdata																								
Connexion automatique grâce à un token																								
Connexion utilisateur																								
Execution d'un jeu																								
Apport de jeu en favoris																								
Modification des paramètres d'émulation																								
Execution d'un jeu																								
Choix de l'hémuleur à utiliser																								
Appliquer des paramètres d'émulation																								
Téléchargement de jeu																								
Verification des jeux qui sont présent sur le pc																								
Téléchargement des jeux vers le pc																								
Archivage des téléchargements/gestion																								
Synchronisation des sauvegardes																								
Verfication si un des fichier de sauvegarde a été mis à jour																								
Téléchargement des sauvegardes vers Caiman																								
Téléversement des sauvegardes vers le Bunker																								
Site web																								
Création du site																								
Création de compte																								
Création interface																								
Affichages des info's utilisateur																								
Fonctions d'administrations																								
Téléchargement de Caiman																								
Création de l'API																								
Interface																								
Création d'une interface																								
Utilisation de la maquette pour contrôler Caiman																								
Autres :																								
Test/résolution de bogues																								
Documentation																								
Journal de bord																								
Rapport																								

CHAPITRE 8

Conclusion et perspectives

8.1 Problème rencontrés

Durant ce travail, j'ai dû surmonter plusieurs difficultés, que ce soit des choses que je ne connaissais pas, ou bien des inconnues techniques.

Je pense que la plus grosse contrainte que j'ai eu, est que j'ai dû créer une API que je n'avais pas prévu à la base. Quand j'ai commencé l'application Caiman, j'ai discuté avec M.Maréchal et M.Schmid, les deux m'ont conseillé de créer une API pour accéder à la base de données depuis Caiman. Donc, J'ai pris une semaine complète pour créer mon API. Originellement, ce temps passé sur l'API n'était pas prévu mais je pense que cela m'a fait gagner du temps finalement.

Au niveau du code tout s'est relativement bien passé. L'un des seuls points où j'ai eu des difficultés est le téléchargement de jeux. Que ce soit du côté serveur ou client, je n'avais jamais fais de l'upload / téléchargement de fichiers. J'ai été conseillé par M.Schmid, ce qui m'a bien aidé.

Un autre point qui a été compliqué c'est la façon dont j'allais distribuer mon application. L'outil inclus dans Visual Studio n'étant pas vraiment mis à jour par Microsoft, j'ai du chercher de mon côté comment créer un programme d'installation, mais je n'ai pas vraiment trouvé. Finalement, M.Schmid m'a aidé mais tout n'était pas réglé pour autant. Le problème était que quand j'installais Caiman, certains fichiers des émulateurs n'étaient plus là donc je ne pouvais plus exécuter de jeu.

Le plus gros défi a été de créer une interface graphique. Durant notre formation nous ne sommes pas formés pour cela. Alors, j'ai dû beaucoup me documenter pour le projet. Finalement, je pense avoir une interface agréable et de surcroît utilisable avec une manette.

8.2 Expérience acquise durant le travail

J'ai énormément appris de choses durant ce travail, en particulier sur le développement en C#, et sur le fonctionnement des émulateurs. Je n'avais jamais créé d'application aussi complexe. Pour faire fonctionner mon application, j'ai dû faire énormément de choses différentes. J'ai apprécié mettre toutes ces choses en relation du développement C#, du développement de site web, la création d'une API et la configuration d'un serveur Debian. J'ai donc pu créer une application de A à Z (sans compter les émulateurs bien sur) qui au final fonctionnent bien.

8.3 Améliorations envisageables

8.3.1 Caiman C#

J'ai eu une grande frustration durant ce travail, le fait de ne pas avoir 2 mois de plus. Chaque jour sur lequel j'ai travaillé sur l'application Caiman, j'avais énormément d'idées qui me venaient en tête. Les principales étaient. :

- Ajouter d'autres émulateurs
- Synchroniser les paramètres de caiman entre les différents pc de l'utilisateur
- Améliorer l'interface
- Ajouter un système d'amis pour voir plus facilement ce à quoi jouent les autres joueurs
- Créer un meilleur tri dans les jeux
- Faire la liste des jeux les plus téléchargés
- Avoir une meilleure gestion des téléchargements
- Prendre en compte plus de manettes (PS4,Switch,etc)

Vous l'avez compris, cette liste est sans fin. Mais je savais pertinemment avant même de commencer ce travail que je ne pourrais pas être vraiment satisfait du résultat final.

J'aurais aussi aimé pouvoir créer un installateur pour Caiman, mais quand j'ai essayé j'ai eu des soucis avec les fichiers des émulateurs. J'ai donc finalement dû y renoncer à cause d'un manque de temps.

8.3.2 API

L'API n'étant pas prévu à la base, elle a été une charge de travail supplémentaire mais qui finalement m'a fait gagner du temps. Des améliorations sont possibles, par exemple je pense que mes endpoints pourraient être améliorés et surtout plus nombreux. Par exemple, je pourrais créer un endpoint pour les téléchargements des jeux et des fichiers de sauvegarde.

8.4 Conclusion Globale

J'ai eu un réel plaisir à créer Caiman. Ce fut une expérience vraiment intéressante pour moi. Durant ma formation, je n'avais jamais pu créer de "vrai" projet en C#, alors quand j'ai dû choisir le sujet de mon travail, j'ai tout de suite sauté sur l'occasion d'en faire.

J'ai eu la chance de pouvoir travailler sur ce projet en sachant qu'il était atypique. Je voulais vraiment pour mon diplôme travailler sur un sujet qui me plaît.

Pour conclure, je pense que Caiman est une réussite. Mon but principal était qu'une personne qui ne connaisse rien aux jeux vidéo et en particulier aux émulateurs puisse jouer à un jeu en moins de 5 minutes, ce qui est le cas. Voilà pourquoi je pense que mon projet est réussi.

CHAPITRE 9

Sources

<https://stackoverflow.com/questions/30013448/c-sharp-create-dir-in-appdata/30013494>

<https://www.digitalocean.com/community/tutorials/how-to-install-the-apache-web-server-on-debian-10>

<https://linuxize.com/post/how-to-install-php-on-debian-10/>

<https://stackoverflow.com/questions/307688/how-to-download-a-file-from-a-url-in-c>

<https://www.tutorialspoint.com/check-if-a-file-exists-in-chash>

<https://stackoverflow.com/questions/29728829/cannot-download-file-using-fpassthru>

<https://stackoverflow.com/questions/26010606/how-to-make-winform-scrollable-in-c-sharp>

<https://stackoverflow.com/questions/8255533/how-to-add-new-line-into-txt-file>

<https://stackoverflow.com/questions/837488/how-can-i-get-the-applications-path-in-a-net-console-application>

<https://stackoverflow.com/questions/2104099/c-sharp-if-then-directives-for-debug-vs-release>

https://www.reddit.com/r/DolphinEmulator/comments/gtj0es/dolphin_for_windows_no_gui/

(pour les images des jeux)

<https://ggapp.io/>

<https://stackoverflow.com/questions/8103860/move-uploaded-file-gives-failed-to-open-stream-permission-denied>

<https://www.c-sharpcorner.com/UploadFile/dbeniwal321/how-to-delete-a-file-in-C-Sharp/>

<https://stackoverflow.com/questions/1288718/how-to-delete-all-files-and-folders-in-a-directory>

<https://www.c-sharpcorner.com/UploadFile/mahesh/create-a-text-file-in-C-Sharp/>

<https://stackoverflow.com/questions/4580263/how-to-open-in-default-browser-in-c-sharp>

<https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/file-system/how-to-copy-delete-and-move-files-and-folders>

<https://docs.microsoft.com/en-us/dotnet/standard/io/how-to-compress-and-extract-files>

https://www.reddit.com/r/DolphinEmulator/comments/gtj0es/dolphin_for_windows_no_gui/

<http://institutions.ville-geneve.ch/fr/bm/interroge/archives-questions-reponses/detail/question/est-il-legal-de-telecharger-des-films-sur-internet-br/>

<https://css.comonsoft.com/tutoriels/installation-certificat-lets-encrypt-ssl-debian.htm>

CHAPITRE 10

Remerciement

Pour commencer, je tenais en particulier à remercier les personnes qui ont pris le temps de me faire des retours sur Caiman. Sans les retours que j'ai reçus de leur part, Caiman ne serait pas aussi bien peaufiné qu'il l'est actuellement. Ensuite, j'aimerais remercier M.Maréchal et M.Schmid pour leurs conseils et leurs retours tout le long de mon travail de diplôme. Pour finir, j'aimerais remercier les autres élèves de ma classe avec qui j'ai passé de très bon moment et qui m'ont aussi donné des conseils pertinents. Je tiens en particulier à remercier M.Borel-Jaquet qui m'a énormément aidé avec la structure de l'API.

CHAPITRE 11

Logbook

11.1 19.04.2021

11.1.1 8h05

Entretiens avec M. Garcia

11.1.2 9h05

Copie de mon disque pour Gawen

11.1.3 9h20

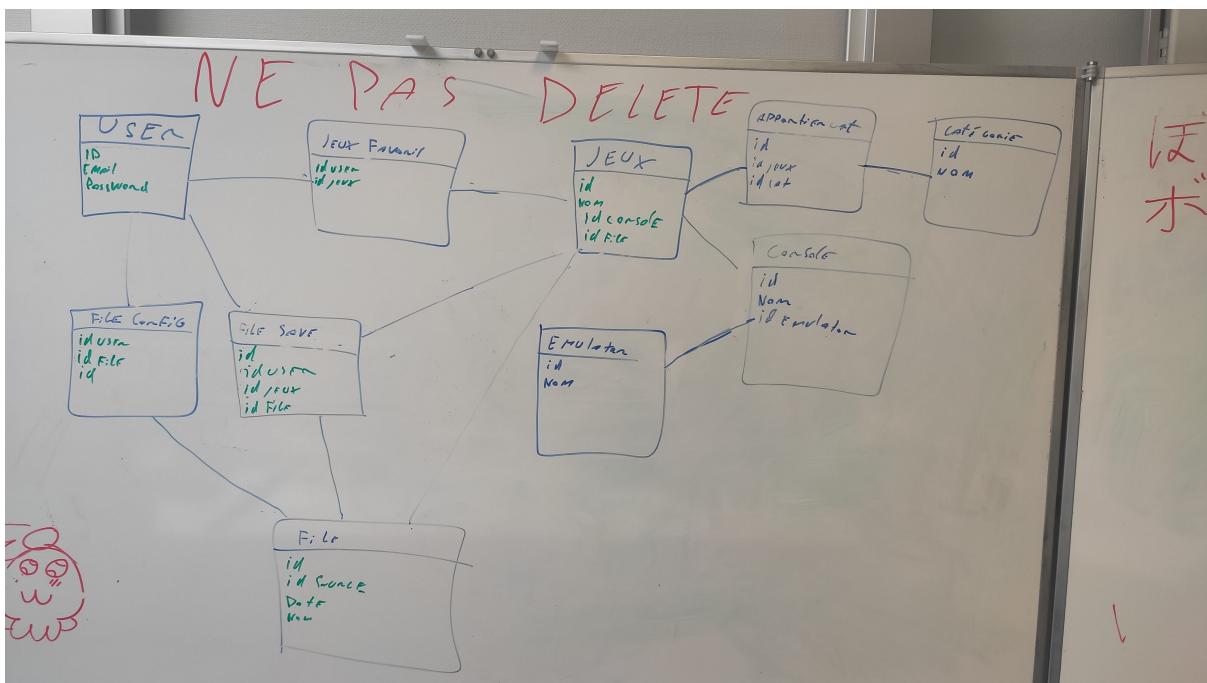
Création du git

11.1.4 9h30

je réfléchi à ce par quoi je vais commencer j'hésite entre commencé entre le site web ou les téléchargement

11.1.5 9h40

Modélisation de la BDD



11.1.6 10h30

Installation de laragon

11.1.7 10h40

Création de la base de données

11.1.8 12h40

Création de la structure du site web

11.1.9 14h35

Création des différentes pages et mise en place de bootstrap

11.1.10 résumé

j'ai créer la base de donnée et le site web

11.2 20.04.2021

11.2.1 8h05

je continue à créer le site web je crée les formulaires pour la connexion

11.2.2 15h20

J'ai impliqué la création de compte et la connexion, le mail doit être unique et le username aussi.
j'ai selon les indication de M. Schmid utilisé les fonctions
password_hash et password_verify de php.

11.2.3 15h30

j'ajoute des donnée a la main dans la bdd

11.2.4 15h56

aide de M.Schmid pour du sql

11.3 21.04.2021

11.3.1 8h10

Modification de la structure du site et ajout de l'update de mots de passe

11.3.2 13h00

l'affichage des jeux est disponible ainsi qu'une recherche sur les jeux grâce à leurs noms.

11.3.3 13h20

Ajout de champs dans la table game

- description
- imageName

11.3.4 15h00

l'affichage de la recherche et du détail d'un jeu fonctionne mais n'est pas beau.

11.4 22.04.2021

11.4.1 8h05

Ajout de l'affichage des catégories de chaque jeu

11.4.2 9h00

Affichage des jeux qui appartiennent à une catégorie.

11.4.3 10h20

modification de l'interface de recherche

11.4.4 notes personnelles

- je dois ajouter une table pour savoir le nombre d'heure de jeu de chaque utilisateurs
- je dois ajouter une gestion des messages d'erreurs.
- je peux ajouter une photo de profil

11.4.5 11h0

modification de l'interface de connexion et de d'inscription

11.4.6 11h50

suppression de la page de création de compte

11.4.7 12h40

Ajout d'un jeux en favoris

11.4.8 15h00

suppression d'un jeu en favoris

11.4.9 a faire demain

- l'affichage des card dans le dashboard n'est pas bon
-

11.5 23.04.2021

11.5.1 8h05

modification de la l'interface du Dashboard

11.5.2 8h40

Ajout d'un champ dans la table utilisateur pour spécifier si l'utilisateur est privé ou non si l'utilisateur n'est pas privé tout le monde va pouvoir voir son profils

11.5.3 9h30

la modification du paramètre pour savoir si le compte est privé ou non

11.5.4 12h20

test de Git Hook

11.5.5 13h30

modification de l'interface theme blanc -> dark

11.5.6 15h20

Création de la page de téléchargement et mise en place de du téléchargement de Caiman depuis le site

11.5.7 notes pour la prochaine fois

- Je dois créer la partie Administrateur du site
 - Je dois créer une fonctionnalité qui me permet de gérer les messages d'erreurs
 - Je dois sécuriser l'accès aux pages
 - Je dois sécuriser les différents formulaires
 - Je dois me renseigner comment uploader des gros fichiers depuis un poste clients
 - Je dois changer de navbar
-

11.6 26.04.2021

11.6.1 8h05

notes personnelles :

- Je dois ajouter la possibilité d'afficher la page d'un utilisateur
- Je dois corriger mon script d'export de base de données

11.6.2 8h10

Création de la page dédiée aux administrateurs.

11.6.3 8h30

Ajout de catégorie

11.6.4 9h00

Ajout de jeu

11.6.5 notes personnels

j'ai regardé plusieurs méthodes pour envoyer un fichier depuis un formulaire en php. Pour l'instant j'utilise les fonctions de base de php et elle fonctionne donc je vais faire des tests une fois le site uploadé sur le Bunker.

11.6.6 11h00

le fichier .iso est uploadé avec le bon nom mais pas encore dans la base de données

11.6.7 15h00

Le jeu est bien ajouté avec le bon nom ainsi que la bonne image.

11.6.8 15h45

il est maintenant possible de mettre à jour le nom, la description ou la console d'un jeu.

11.6.9 15h50

modification de la structure du git

11.7 27.04.2021

11.7.1 8h05

ajout/ suppression de catégories à un jeu

11.7.2 9h20

modification mineur de l'interface

11.7.3 10h05

recherche d'un profil utilisateur

11.7.4 12h15

la recherche et l'affichage d'un profil utilisateur est fonctionnel

11.7.5 13h00

modification de l'interface pour que les jeux s'affiche correctement

11.7.6 13h30

Le site est fonctionnel mais il manque des détail comme les message d'erreur et les droit sur les pages

11.7.7 notes personnelles

pour finir le site il me reste les choses suivante à faire :

- Sécuriser les pages
- afficher des messages d'erreur
- suppression de catégories
- suppression de jeu
- mot de passe oublié
- commenter mon code
- pagination pour les recherches

11.7.8 13h40

documentation

11.8 28.04.2021

11.8.1 8h05

documentation

11.8.2 10h40

correction d'un bogue sur le nombre d'heure de jeu et triage des jeux par heure de jeu

11.8.3 11h00

gestion des droit d'accès au page

11.8.4 13h10

modification de la navbar

11.8.5 14h00

gestion des erreurs de du login

11.9 29.04.2021

11.9.1 8h30

Documentation

11.9.2 10h20

Mr. Garcia m'a aidé à mettre en place ma documentation doxygen

11.9.3 11h00

documentation

11.10 30.04.2021

11.10.1 8h05

Correction du logbook

11.10.2 8h20

supression de code inutile sur le site et mis a jour de la documentation

11.10.3 9h30

ajout de contrainte dans la base de données

11.10.4 10h45

Création du planning effectif

11.10.5 13h00

Configuration de debian

user tfp : FTPdiplomant password ftp : SuperCfpt@

J'ai pus installer apache, php, et mysql mais je n'arrive pas à me connecter en ftp. A la connexion je bloque sur l'erreur : "Impossible de récupérer le contenu du dossier". Malgré l'aide de Mr.Schmidt je n'ai toujours pas réussi.

11.10.6 15h20

Je n'arrive toujours pas à me connecter au ftp, je ne sais pas si le problème viens de ma configuration ou du firewall

11.11 notes pour le premier rendu

J'ai durant ces deux premières semaines, créé le site internet de Caiman. Le site n'est pas fini à 100% mais les fonctionnalités de base sont toutes implémentées. Les fonctionnalités actuelles permettent de faire toutes les choses nécessaires au fonctionnement de l'application. J'ai pris plus de temps que prévu à réaliser le site mais durant la création de mon planning prévisionnel j'ai fait des erreurs, j'ai par exemple oublié de planifier la création des fonctionnalités d'administration (Ajout de jeux, ajout de catégories, assignation de catégories à un jeu, upload de jeux, etc).

La documentation du site n'est pas forcément touffue mais le site n'est pas particulièrement complexe, il m'a pris du temps dû au nombre de tables à gérer. Il reste à mettre en place la récupération de mot de passe mais j'ai décidé de passer à l'interface graphique de l'application C# dès la semaine prochaine. Étant donné que le projet est la partie la plus importante de l'application, je ne vais pas configurer la récupération de mot de passe maintenant.

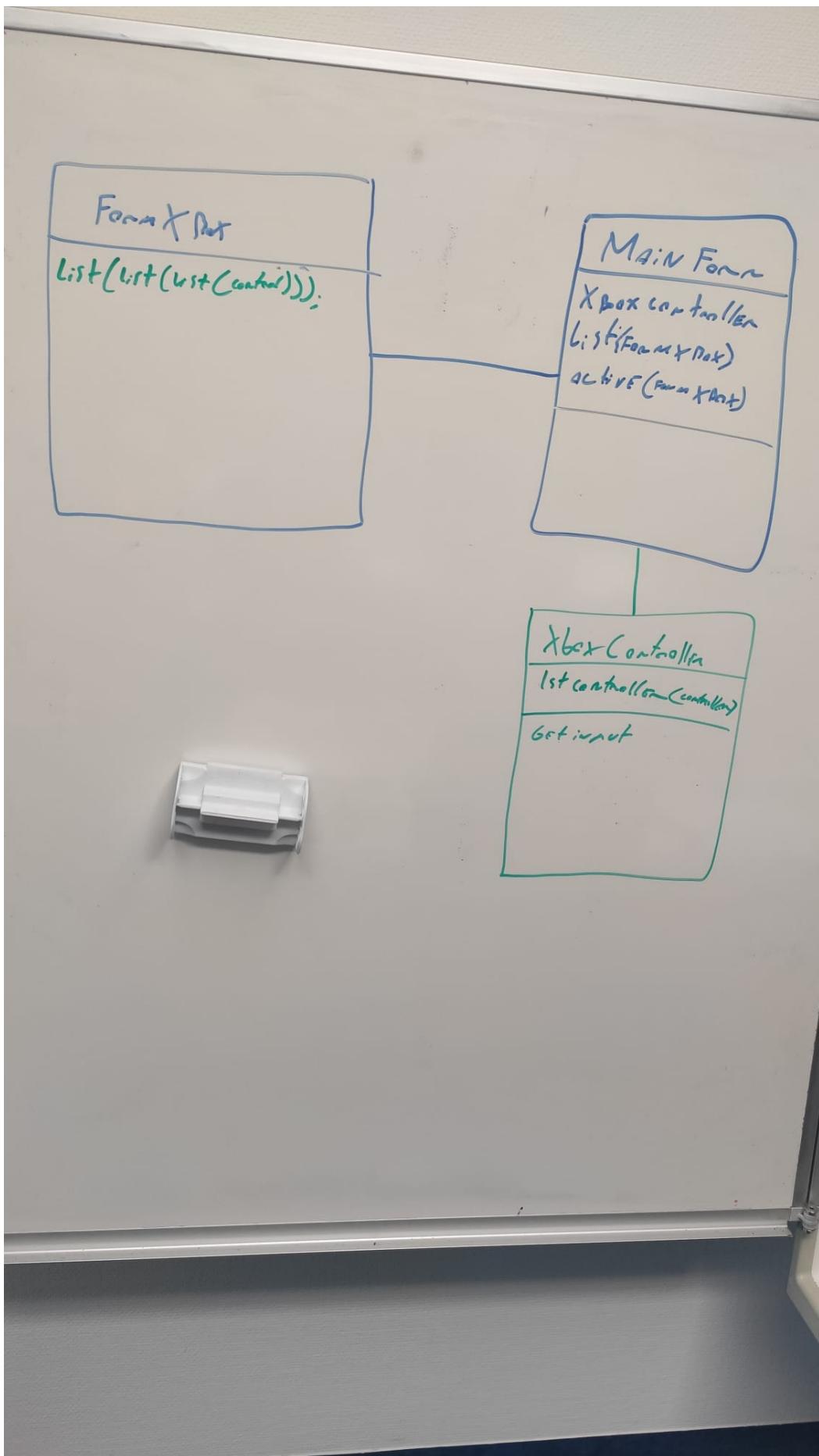
11.12 03.05.2021

11.12.1 8h10

Réflexion sur l'interface graphique et création du projet

11.12.2 8h15

Importation de la classe XboxController.cs que j'ai créé précédemment.



La table MainForm contient un XboxController() cette classe permet de connaître les manettes connectés au pc et de recevoir leur inputs.

Elle contient aussi une liste de form, ces formes sont les différentes fenêtres de l'application (si possible j'aimerai faire que seul une fenêtre soit active).

Je fais des test avec les usercontrols

11.12.3 10h00

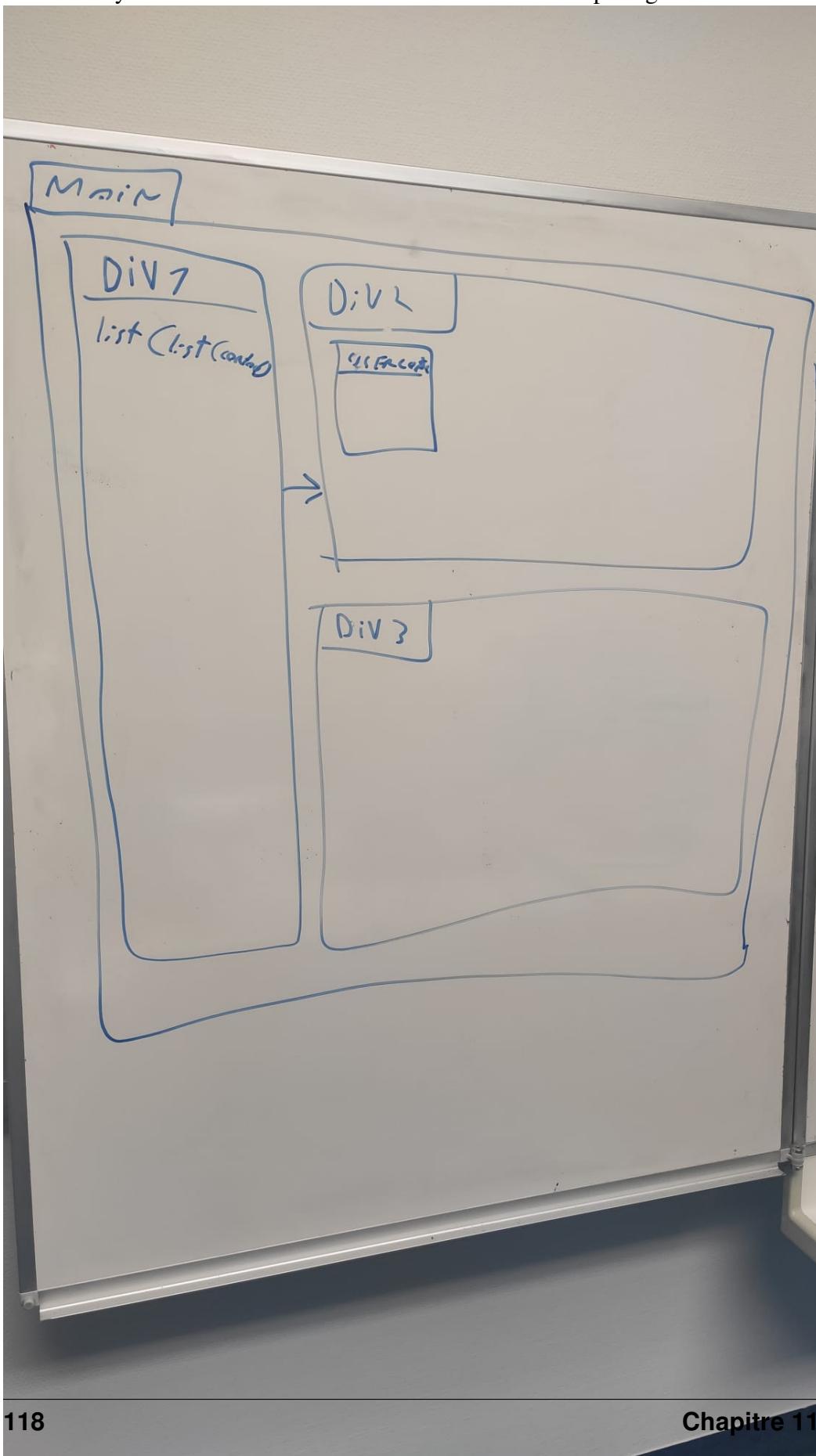
je n'arrive pas à afficher dynamiquement quand une action est faite.

11.12.4 12h00

Mon problème venait du fait que je n'initialisais pas l "usercontrol.

11.12.5 13h00

Je vais essayer de me baser sur la structure des div en html pour gérer le contenu de l'affichage.



11.12.6 14h00

Finalisation de la configuration du serveur

11.13 04.05.2021

11.13.1 8h05

Je continue à faire des test pour pouvoir bouger le focus d'une "div" à une autre

11.13.2 11h00

Je continue à faire des test mais j'ai apporté des modifications :

le MainForm ne contient pas une liste de list de control, ce n'est pas nécessaire sachant que le lien entre les control est seulement connu des sous contrôle

dorénavant chaque sous control possède une liste de listes de control pour pouvoir se déplacer.

quand on déplace la position maximum dans un sous control il y a deux possibilité. : un control est disponible dans la direction souhaité rien n'est disponible Si un control est disponible alors la main form est informée qu'elle doit changer d'activeForm pour pouvoir se déplacer dans de bonne condition.

11.13.3 12h05

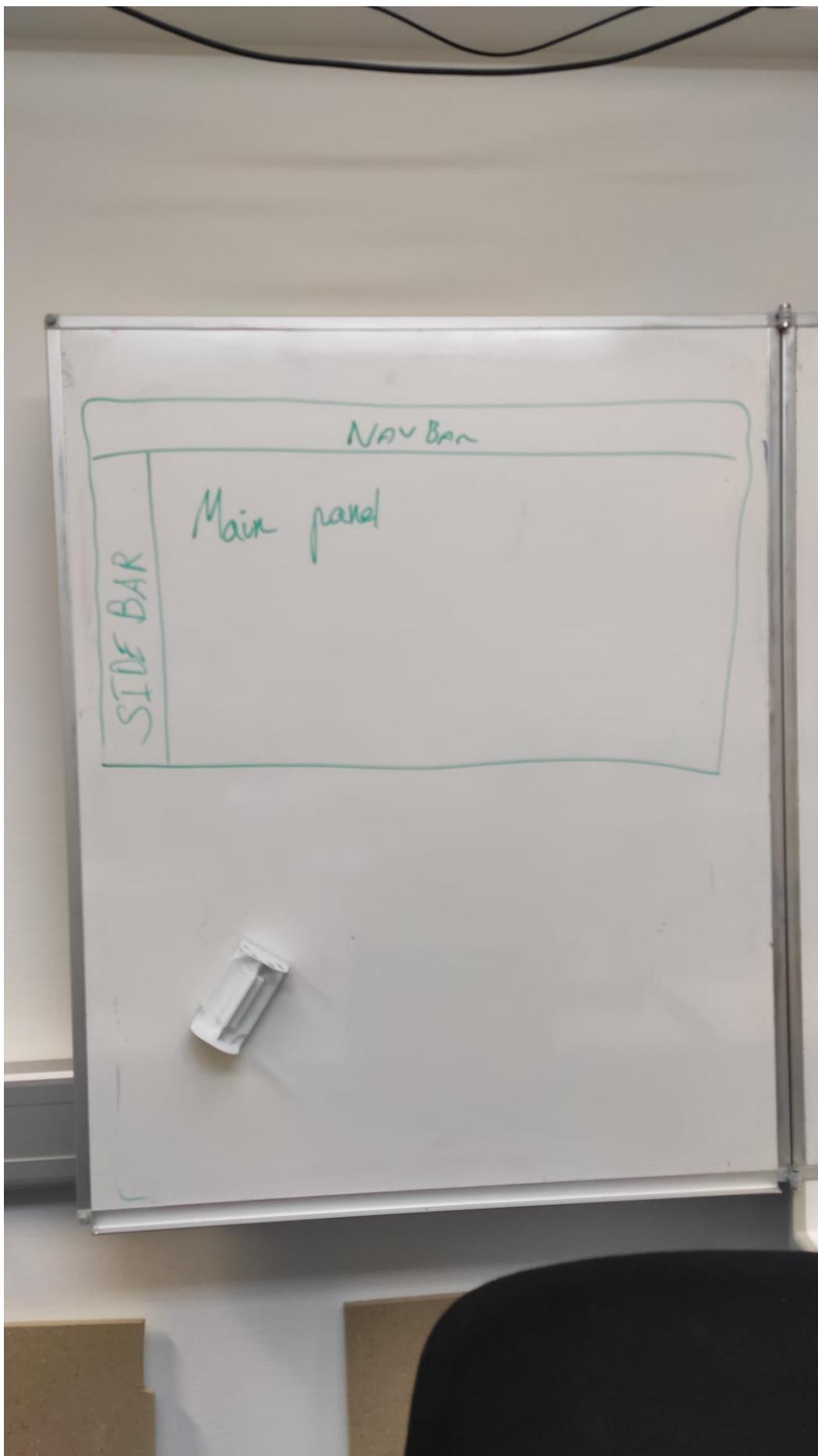
Le déplacement dans chaque form fonctionne et l'on peut passer d'une form à une autre.

11.14 05.05.2021

11.14.1 8h05

Je continue à améliorer le fonctionnement de l'interface.

J'ai décidé de diviser l'interface en 3 "partie" la navbar la sidebar le main contenu



11.14.2 10h50

J'ai créé une nouvelle classe ButtonContext.cs, elle contient les paramètres qui doivent être passer a la forme pour quel sache l'action à exécuter.

Il est maintenant possible de cliquer sur un bouton et la manette va reprendre la ou l'utilisateur a cliqué.

11.14.3 12h40

Je permet le déplacement grâce au joystick gauche

11.14.4 13h00

J'essaie de faire en sorte que je puisse revenir en arrière dans les fenêtre

11.14.5 15h00

J'ai essayé de sauver dans une liste les choses précédemment affichée mais j'ai des soucis avec les liens entre les différents panel.

Je vais essayer de recréer les anciens panel a chaque fois au lieu de les recharger.

11.15 06.05.2021

11.15.1 8h05

Je continue a faire en sorte que je puisse retourner en arrière grâce à la touche "B"

11.15.2 9h00

Il est maintenant possible de revenir en arrière dans la navigation

11.15.3 10h00

si il y a des “trou” dans la navigation le curseur le contourne

11.15.4 10h40

J'essaie d'afficher des images dans l'application

11.15.5 13h00

J'ai discuté avec Mr Maréchal de mon git. Pour pouvoir appliquer correctement mon gitignore j'ai dus supprimer les fichier du git

11.15.6 14h00

J'ai fais des recherche sur la publication de projet et j'ai corrigé des bugs

11.16 07.05.2021

11.16.1 8h05

J'ai essayé le paquet “Microsoft Visual Studio Installer Projects” que m'a conseillé M. Schmid. L'installation marche mais certains dossiers de PCSX2 et de Dolphin ne sont pas inclus dans l'installation.

11.16.2 10h00

Je commente les classe que j'ai créé et je supprimer les fonctions qui ne sont pas utilisé

11.16.3 12h40

documentation + Création d'une release pour pouvoir essayer l'interface

Je déplace la documentation du projet web au projet desktop la seul documentation qui reste dans le projet web est celle qui concerne son propre code (doxygen)

11.16.4 notes personnelles

Je dois modifier la connexion a la base de données pour passer des fonctions de cryptographie de PHP au mds. Le problème ce que je ne peux pas me connecter depuis le c# avec les fonctions de cryptographie de php.

11.16.5 15h30

J'ai mis à jour ma documentation du projet.

J'ai essayé d'ajouter des tâches à mon git mais j'ai eu des soucis pour la signature du projet donc je remet ça à plus tard.

11.17 10.05.2021

11.17.1 8h05

Je modifie la création de mot de passe et la connexion pour qu'un utilisateur puisse se connecter depuis le site web et Caiman.

J'utilisais les fonctions de php (password_hash et password_verify) mais je passe a du md5 + salt. La raison est que je ne pouvais pas utiliser ces fonctions pour me connecter depuis l'application c#.

11.17.2 10h10

je vais essayer de créer un login en C#

Pour ce faire, je commence par créer une classe “AccessDatabase.cs” pour communiquer avec la base. Pour stocker la la connexionString je l'ai mise dans les settings de l'application.

11.17.3 10h45

Pour pouvoir me connecter a la base de donnée je dois passer par le port 1433 mais il est fermé dans le firewall du coup je dois essayer de me connecter par le port 433

11.17.4 13h30

Je commence à créer la “vrai” interface je commence par les menu de configurations et le menu pour quitter l’application.

11.17.5 14h00

Après une discussion avec M. Scmid j’ai décidé de faire une api.

11.17.6 14h30

J’ai demandé à M.Borel de l’aide pour la structure de mon api, il a pu m’indiquer une structure correcte mais elle est très verbeuse donc je vais sûrement avoir pour minimum 3 jours à faire mon API.

11.18 11.05.2021

11.18.1 8h00

Je continue mon api, je commence par essayer la structure de M.Borel sur une seule table pour essayer puis je vais faire les autres.

11.18.2 11h40

Je peux maintenant faire des requêtes mais j’ai un souci avec les headers. Le header Authorization n’est pas correctement reçu Je pense que vu le temps que faire une API prend je fais faire en sorte que l’API ne soit que faite pour Caiman. Je vais donc seulement créer les requêtes nécessaires

11.18.3 13h00

Je vais lister les requêtes qui me seront potentiellement utile pour Caiman

liste des jeux recherche de jeu affichage des informations d’un jeu recherche des informations d’un utilisateur recherche des jeux avec un nombre qui ont été joué par un utilisateur particulier recherche d’un jeu selon sa catégorie jeux favoris d’un utilisateur recherche des jeux qui ont été joué connexion d’un utilisateur Création de compte réception d’un fichier de sauvegarde réception d’un fichier de configuration

11.18.4 13h40

Je fais un test d'appel a l'API depuis Caiman

11.18.5 14h20

Pour pouvoir utiliser correctement les appels à l'api je dois utiliser des objets que je rempli avec chaque appel.

Par exemple, si je veux recevoir les informations d'un jeu je créer un objet jeu a qui je vais attribuer les données que je viens de recevoir.

Il me faut donc créer une classe pour l'utilisateur et une classe pour les jeux.

11.18.6 15h50

je dois aussi pouvoir connaître la liste des catégories grâce à l'API

11.18.7 16h00

Il est maintenant possible de récupérer les catégories

11.19 12.05.2021

11.19.1 8h05

Documentation et suppression de code inutile

11.19.2 13h00

J'ai discuté avec M.Smid il m'a donner des conseil sur pour mon api

11.19.3 14h30

J'ai changé les routes de mon api exemple : /games/userFavorites/8 => /games/?byUserFavorites=8

11.19.4 15h30

J'ai essayé de mettre l'api en ligne mais quand j'arrive sur une page j'ai une erreur 500.

11.20 13.05.2021

11.20.1 11h00

L'erreur 500 que j'avais était liée à une version de PHP, le serveur avait une version de php 7.3 alors que mon API a besoin d'une version PHP minimum en 7.4. J'ai donc mis à jour la version présente sur le serveur.

11.20.2 12h00

J'ai corrigé différentes erreurs liées à l'api par exemple quand on envoyait que le username et pas de password pour se connecter une erreur apparaissait.

11.20.3 13h00

J'ai une erreur sur le serveur quand je veux faire une requête d'utilisateur avec sa clé d'API j'ai une erreur 404.

Je vais peut être passer la recherche d'api en POST et non en GET

11.21 14.05.2021

11.21.1 8h00

J'ai toujours une erreur quand je veux récupérer les informations d'un utilisateur grâce à son apitoken, l'erreur n'est présente que sur le caiman.cfpt.info. Pour corriger l'erreur je vais passer par du post pour éviter de perdre trop de temps.

11.21.2 09h22

j'ai push sur le serveur une version de l'API qui fonctionne bien, je vais maintenant continuer ma documentation

11.21.3 09h30

je me rend compte qu j'ai une requête qui ne marche pas

11.21.4 11h30

J'ai dû modifier le .htaccess pour pouvoir appliquer les rewriteRules. Le fichier n'était pas pris en compte sur le serveur je l'ai donc mis dans le dossier ou point le virtual host.

11.21.5 12h00

documentation

11.22 17.05.2021

11.22.1 8h05

Création d'une ébauche de diagram de classe sur le tableau.

J'ai repris en grande partie le diagramme de mon POC en y ajoutant une gestion des sauvegardes et des téléchargement.

11.22.2 9h00

Création du schéma sous UMLetino

Première version du diagramme de la logique de Caiman.

11.22.3 13h20

J'essaie de réfléchir à la manière de télécharger les sauvegardes et la façon dont je dois stocker les jeux que l'utilisateur a téléchargé.

Je pense que je vais faire en sorte de vérifier si les jeux qui devraient être présents sur le disque de l'utilisateur le sont réellement.

Je vais faire en sorte que l'utilisateur ait la possibilité de pouvoir télécharger les jeux dans un dossier spécifique. Il va devoir au premier lancement de l'application spécifier où l'installation doit se faire. (c'est peut être contraire à mon but de faire une application simple d'utilisation).

Je dois aussi savoir si le disque de l'utilisateur n'a pas la place requise pour télécharger le jeu demandé.

11.22.4 14h35

Discussion avec M.Maréchal

11.23 18.05.2021

11.23.1 8h05

Je vais créer une page de connexion pour l'utilisateur

11.23.2 10h00

La connection marche, je vais maintenant télécharger les images des jeux

11.23.3 12h00

Je commence à faire en sorte que quand un nouveau jeu est reçu depuis l'API son image est automatiquement téléchargé et mise dans le dossier %appdata% de l'application

11.23.4 13h30

J'affiche dans la sidebar le nom des catégories

11.23.5 15h00

Quand je clique sur une catégorie les jeux de la catégorie choisie sont afficher.

11.24 19.05.2021

11.24.1 8h05

je fais en sorte d'afficher tous les jeux quand je lance l'application.

11.24.2 9h00

je vais maintenant faire en sorte de pouvoir voir les détails d'un jeu.

11.24.3 9h30

J'ai un problème avec les image bouton le clic n'est pas pris en compte

11.24.4 11h00

j'ai corriger le soucis des boutons et je modifie un petit peu l'interface

11.24.5 13h00

je commence l'affichage des détails d'un jeu

11.24.6 14h00

je fais des recherche sur la façon de télécharger un jeu

11.25 20.05.2021

11.25.1 8h05

Je crée une route pour pouvoir recevoir le lien de téléchargement d'un jeu.

11.25.2 09h15

Je me rend compte que pour créer la route de téléchargement je dois :

Avoir le lien du site Avoir le lien pour la console avoir le nom du fichier

Pour avec ces différents éléments je dois modifier mon API pour pouvoir faire tout ça.

11.25.3 11h00

J'arrive à créer l'URL mais le fichier n'est pas accessible en dehors du serveur donc je ne sais pas vraiment quoi faire

11.25.4 12h50

Correction d'un problème où l'on pouvait se déplacer dans une case qui n'existe pas donc l'application plantait.

11.25.5 14h00

Je commence à faire en sorte que je puisse créer la route pour télécharger le fichier des jeux

11.26 21.05.2021

11.26.1 9h00

Je voulais télécharger un jeu en passant par le webClient en C# mais ma route est en POST dans mon API. Malheureusement je ne peux pas fournir de paramètres en POST avec la fonction downloadFileAsync donc je dois passer ma route en GET

11.26.2 11h00

j'ai changé ma route et j'en ai ajouté pour pouvoir connaître les dossiers où je dois stocker les jeux et le nom du fichier d'un jeu.

11.26.3 16h00

J'ai réussi à afficher la liste des téléchargements et maintenant je suis en train de faire en sorte de pouvoir ajouter un jeu au favoris. j'ai réglé des soucis de droit d'accès au fichier qui sont en cours de lecture.

11.27 24.05.2021

11.27.1 15h15

les téléchargement se font maintenant l'un après l'autre

11.27.2 16h35

Affichage des jeux téléchargé

11.28 25.05.2021

11.28.1 8h10

je dois modifier l'API pour pouvoir manipuler les jeux favoris des utilisateur

11.28.2 10h25

il est maintenant possible d'ajouter et de supprimer un jeu des favoris. je vais maintenant faire en sorte de pouvoir exécuter des jeux depuis caiman.

11.28.3 10h30

Je regarde le code que j'ai fais pour le poc pour voir si je peux reprendre des parties de codes.

11.28.4 14h00

J'ai un soucis avec le lancement des jeux de PS2

11.28.5 14h25

Le problème venait des paramètres j'avais oublié une espace entre le nom du fichier à exécuter et les paramètres

11.28.6 15h50

J'ai pu ajouter les émulateurs et appliquer certains paramètres graphique.

11.29 26.05.2021

11.29.1 8h05

Je dois modifier mon api pour pouvoir générer un token à chaque connexion pour ne pas avoir à se connecter à chaque lancement de l'application.

11.29.2 10h30

J'ai modifié l'API, je vais maintenant passer a Caiman.

11.29.3 11h00

Il est maintenant possible de se connecter automatiquement , je vais maintenant modifier un petit peu l'aspect graphique de Caiman.

11.29.4 13h00

je veux afficher le jeu qui est actuellement utilisé par l'utilisateur

11.29.5 15h30

Il est possible de voir le jeu actuellement lancé par l'utilisateur et peut importe comment l'utilisateur quite le jeu Caiman va être au courant

11.29.6 18h00

Affichage du temps de jeu de la session

11.29.7 21h00

Le nombre de minutes de jeu a comptabilisé

11.30 27.05.2021

11.30.1 8h10

J'ai remarqué que je ne pouvais pas upload de jeu actuellement je vais essayer de comprendre pourquoi.

11.30.2 09h00

J'ai résolu le problème je devais ajouter les droit d'écriture dans mon debian Par contre hier soir j'ai oublié de push donc je ne peux pas continuer Caiman ce matin

11.30.3 10h05

Je profite de ne pas pouvoir coder pour faire des tests, ajouter des jeux et faire de la doc

11.30.4 23h30

J'ai réussi à scanner un dossier pour savoir si il a y eu une modification.

11.31 28.05.2021

11.31.1 9h30

j'ai réussi à faire en sorte que les sauvegardes de l'utilisateur qui se connecte sont déplacé dans le dossier des sauvegardes des émulateurs.

11.31.2 10h00

Je réfléchi à comment faire pour synchroniser les sauvegardes des utilisateurs. J'ai eu l'idée de créer un sous répertoire pour chaque utilisateur sur le serveur

###10h30

J'ai réfléchi et finalement je pense envoyer des fichier zip contenant toutes les sauvegardes d'un utilisateur et cela pour chaque console. Cela a l'avantage de réduire la taille des fichier et de simplifier l'upload et le téléchargement

11.31.3 14h00

J'ai eu un bogue ou les sauvegarde n'était pas envoyé dans le bon dossier

11.31.4 14h35

Documentation

11.32 30.05.2021

11.32.1 13h30

Modification de l'API pour pouvoir télécharger un fichier de sauvegarde

11.32.2 16h12

Je fais en sorte que les sauvegardes se synchronisent sur caiman

11.32.3 19h43

j'ai ENFIN réussi à synchroniser les sauvegardes de Gamecube mais j'ai un soucis avec celle de ps2

11.33 31.05.2021

11.33.1 8h05

Je commente le code de Caiman

11.33.2 14h00

Discussion avec M.Maréchal et ajout de tests pour la classe GameTimer

11.34 01.06.2021

11.34.1 8h05

Ajout de tests pour la classe TimeInGame

11.34.2 8h20

Je veux faire en sorte que Caiman s'exécute sur l'écran principale du pc de l'utilisateur

11.34.3 8h30

J'ai désactiver le .gitIgnore par ce que cela créait des soucis avec les dossier des émulateurs

11.35 02.06.2021

11.35.1 8h05

Commentaire du code de l'API

11.35.2 10h30

documentation

11.35.3 15h30

J'ai corrigé le fait que un compte pouvait être créé sans remplir tous les champ

11.36 03.06.2021

11.36.1 8h10

J'ai eu un problème de conflit avec mon git j'ai du re télécharger tout le projet mais maintenant c'est bon.

11.36.2 8h34

Je continue la documentation

11.37 04.06.2021

11.37.1 8h05

Documentation

11.37.2 13h15

La documentation avance bien je vais donc créer les diagrammes de classe pour le C#.

11.38 07.06.2021

11.38.1 8h10

Recherche sur comment exporter ma documentation en pdf et je continue ma doc.

11.38.2 11h20

Je vais essayer d'installer un certificat ssl sur le serveur debian

11.38.3 13h30

Je n'arrive pas à installer certbot donc je laisse ça en attente

11.38.4 14h00

Documentation

11.39 08.06.2021

11.39.1 8h10

Documentation

11.40 09.06.2021

11.40.1 8h10

Documentation

11.41 10.06.2021

11.41.1 8h10

Export des fichiers sources en pdf et vérification global