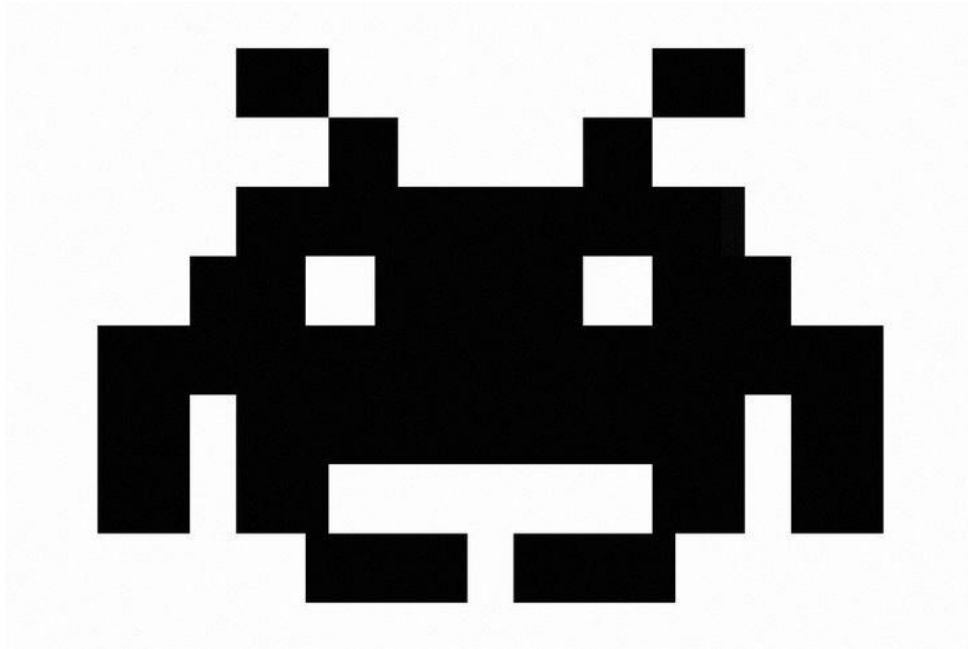


Projet Space Invaders



Auteur : Tiago Rodrigues Sousa

Chef de projet : Mathieu Meylan / Xavier Carrel / Aurélie
Curchod

Du 11/09/2023 au 03/05/2023

Table des matières

1	POO.....	3
1.1	Introduction.....	3
1.2	Analyse fonctionnelle.....	3
1.2.1	Player mouvement	3
1.2.2	Enemy mouvement	3
1.2.3	Player shoot.....	4
1.2.4	Change the program structure	4
1.2.5	make menu	4
1.2.6	Score display.....	5
1.2.7	infinite apparition enemies	5
1.2.8	limit ammo	5
1.2.9	Finish game.....	5
1.2.10	dataBase connection	6
1.3	Analyse technique	6
1.3.1	Diagramme de classe.....	6
1.3.2	Explication	8
1.4	Test unitaire.....	12
1.5	Conclusion	12
2	UX	13
2.1	Introduction.....	13
2.2	Analyse	13
2.2.1	Conception centrée utilisateur.....	13
2.2.2	Palette graphique	14
2.2.3	Eco-conception.....	14
2.2.4	Accessibilité	15
2.3	Conception	15
2.4	Test	18
2.5	Conclusion	18
3	DB	19
3.1	Introduction.....	19
3.2	Importer les données et le schéma de base de données.....	19
3.3	Gestion des utilisateurs	20

3.4	Requêtes de sélection	23
3.4.1	Explication quelques commandes	23
3.4.2	Requêtes.....	24
3.5	Index	27
3.6	Sauvegarde/Restore	27
3.6.1	Sauvegarde	27
3.6.2	Restaurer	27
3.7	Intégration de la base de données dans le code.....	28
3.8	Conclusion	28
4	ChatGPT	28
4.1	POO.....	28
4.2	UX	28
4.3	DB	28
5	Conclusion générale	29

1 POO

1.1 Introduction

Lors du projet space invaders, il a été demandé de faire un jeu comme space invaders en programmation orienté objet. Cette section va expliquer la partie code du projet. Le cahier des charges exigeant que le programme puisse faire au minium :

- Un vaisseau avec possibilité de tir et déplacement
- 10 ennemis qui descendent sur l'axe vertical

Il a également été demandé d'utiliser IceScrum pour la gestion de projet.

1.2 Analyse fonctionnelle

1.2.1 Player mouvement

As a player I want to go left and right with a player	
Tests d'acceptance :	
Go to the left	In the program when i press "a" the player go to the left
Go to the right	In the program when i press "d" the player go to the right
Border left	In the program when i press "a" and the player is on the border left the player stop mouvement
Border right	In the program when i press "d" and the player is on the border right the player doesn't move

1.2.2 Enemy mouvement

As a player I want to see my enemy move	
Tests d'acceptance:	
Ennemie show	When the game start Enemies are displayed (see maquette 1)
Move to the down	When enemies are in the border left or right enemies move down (see maquette 3)
Move to the left or right	If enemies are the border right or left and go down enemies move to the left or right
Move right at the start	when game start enemies go to the right (see maquette 2)

1.2.3 Player shoot

As a player I want to can shoot with de player In order to kill enemies	
Tests d'acceptance :	
player shoot	In the program when the player press "space" a ammo is shot
If ammo don't touch enemies	In the program If ammo reaches the border top ammo disappear
If ammo touch enemies	In the program if ammo touch enemies ammo and enemies disappear
ammo move	in the program with an ammo shot the ammo moves by itself straight up

1.2.4 Change the program structure

As a programmer I want to change the program structure to separate the different elements of the program	
Tests d'acceptance:	
Program separate	In the structure of the program when i see the structure the program is separate in 4 project (model, display, console, storage)

1.2.5 make menu

As a player I want a menu In order to navigate in game	
Tests d'acceptance:	
display menu	In the desktop when i start the program he display a menu (see maquette 1)
go down	In the main menu when i press down arrow Cursor go down
go up	In the main menu when i press up arrow Cursor go up
border top/down	In the main menu when cursor is in border top/down and i press up/down arrow Cursor don't move
go highscore/rules/option	In the main menu when cursor is in highscore/rules/option and press Enter he display a highscore/rules/option page
quit	In the main menu when cursor is in quit and press Enter He quit the program
play	In the main menu when cursor is in play and press Enter He start the game

1.2.6 Score display

As a player I want to see score when i play	
Tests d'acceptance:	
display score	In the game when the game starts the score is displayed and it's equal at 0 (see maquette 1)
add score	In the game when player kill an enemy the score add 10 points

1.2.7 infinit apparation enemies

As a player I want to more enemies appear In order to add more difficult	
Tests d'acceptance:	
new enemies appear	In the game Every 8 seconds between 5 and 10 enemies appear randomly at the top middle of screen
all enemies go down	In the game when ONE enemies touch the border left or right ALL enemies go down
change direction	In the game when enemies go down they change a direction at opposite

1.2.8 limit ammo

As a player I want to limit a number of ammo In order to have more difficult and not spam	
Tests d'acceptance:	
display number ammo	In the program When game starts he display a number ammo under the score
lose ammo	In the game when the player shoot the number of ammo goes down by 1
give ammo	In the game when the player kills 5 enemies the number of ammo goes up by 3
run out of ammo	In the game with 1 ammo left when the player shoots and does not earn new ammos the player loses game

1.2.9 Finish game

As a player I want to have end in the game	
Tests d'acceptance :	
Lose game	In the main program if the enemies are on the same line as the player the console is cleared and he displays a game over screen (see maquette 1)
Press Enter	In the game over screen When player press enter This returns to the main menu
Enter a name	In the game over screen When player enter a random name The name is displayed at the screen next to "Enter a name"

1.2.10 dataBase connection

As a devlopper I want to do the dataBase connection

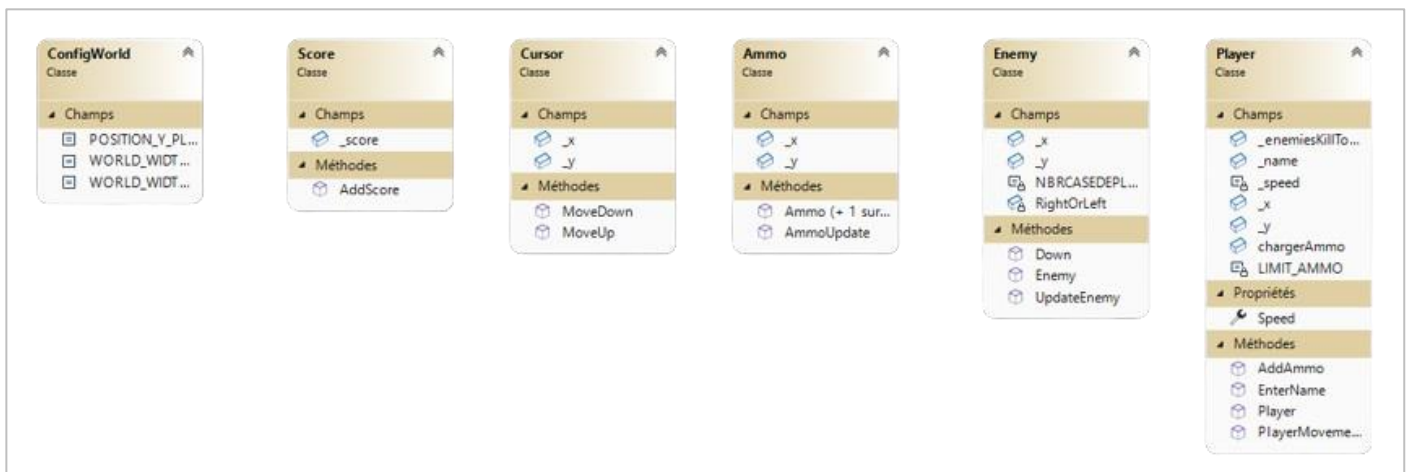
Tests d'acceptance:

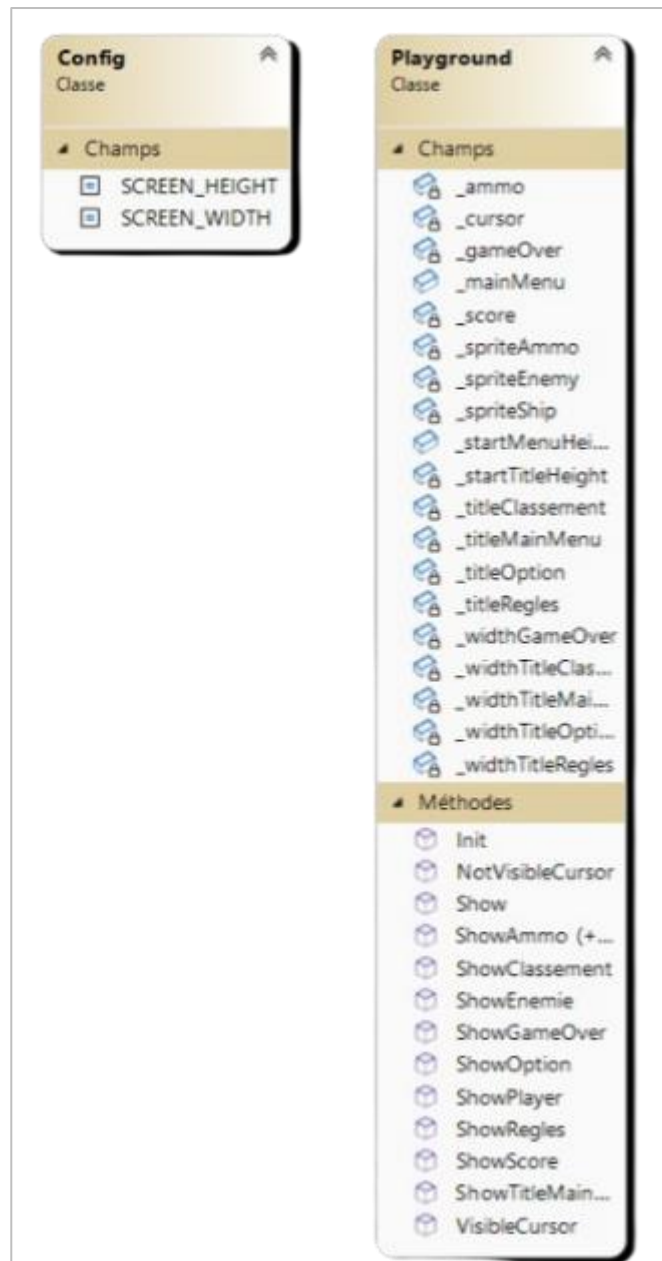
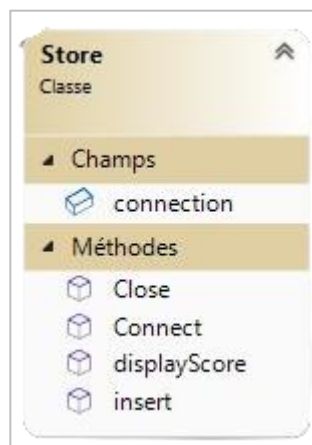
open connection	In the main progam when the highscore page is loaded the database connection is open
Select scores	In the main progam when the highscore page is loaded a select query is made to get all games data: - the player name - player Score The data is sorted on time ascending, only the first 5 players are selected
Insert player score	In the main program when the player lose the game her score is save in the database score
close connection	In the main progam When the highscore page is unloaded the database connection is closed

1.3 Analyse technique

1.3.1 Diagramme de classe

Model



Display*Store*

1.3.2 Explication

1.3.2.1 *Model.Ammo Class Reference*

Public Member Functions

- **Ammo** (int x)
Constructeur lorsque le joueur tire la balle prend la même position que le joueur.
- **Ammo** ()
Constructeur pour mettre des balles dans le chargeur du joueur.
- void **AmmoUpdate** ()
met à jour les balles

Public Attributes

- int **_x**
- int **_y** = ConfigWorld.POSITION_Y_PLAYER_FOR_START

Constructor & Destructor Documentation

Model.Ammo.Ammo (int x)

Constructeur lorsque le joueur tire la balle prend la même position que le joueur.

Parameters

x	
---	--

Display.Config Class Reference

Static Public Attributes

- const int **SCREEN_HEIGHT** = 40
- const int **SCREEN_WIDTH** = 150

1.3.2.2 *Model.ConfigWorld Class Reference*

Static Public Attributes

- const int **WORLD_WIDTH_LEFT** = 25
- const int **WORLD_WIDTH_RIGHT** = 125
- const int **POSITION_Y_PLAYER_FOR_START** = 30

1.3.2.3 *Model.Cursor Class Reference*

Public Member Functions

- void **MoveDown** ()
Met à jour la position verticale lorsqu'il descend
- void **MoveUp** ()
Met à jour la position verticale lorsqu'il monte

Public Attributes

- int **_x**
- int **_y** = 0

1.3.2.4 *Model.Enemy Class Reference*

Public Member Functions

- **Enemy** (int x)
Constructeur pour décider ou commence l'ennemie sur la ligne horizontal
- void **UpdateEnemy** ()
Va déplacer l'ennemie à gauche ou à droite d'un certain nombre de case
- void **Down** ()
Fais descendre l'ennemie de 2 case (hauteur de l'ennemi) et va changer la direction x de l'ennemie, c'est à dire s'il va à gaucher ou à droite

Public Attributes

- int **_x**
- int **_y**

Constructor & Destructor Documentation

Model.Enemy.Enemy (int x)

Constructeur pour décider de la hauteur des ennemies.

Parameters

x	
---	--

1.3.2.5 Model.Player Class Reference

Public Member Functions

- **Player ()**
Constructeur qui va mettre le nombre maximal de munitions dans le chargeur du joueur
- void **AddAmmo** (int numberAmmo)
Va ajouter des balles dans le chargeur
- void **PlayerMovementUpdate** (int move)
Va mettre à joueur la position du joueur
- void **EnterName** (string pseudo)
le joueur met un pseudo

Public Attributes

- int **_x** = 75
- int **_y** = ConfigWorld.POSITION_Y_PLAYER_FOR_START
- int **_enemiesKillToGiveAmmo** = 0
- string **_name** = ""
- List< **Ammo** > **chargerAmmo** = new List<Ammo>()

Properties

- int **Speed** [get]

Member Function Documentation

void Model.Player.AddAmmo (int numberAmmo)

Va ajouter des balles dans le chargeur

Parameters

<i>numberAmmo</i>	
-------------------	--

void Model.Player.EnterName (string pseudo)

Le joueur met un pseudo

Parameters

<i>pseudo</i>	
---------------	--

void Model.Player.PlayerMovementUpdate (int move)

Va mettre à joueur la position du joueur

Parameters

<i>move</i>	
-------------	--

1.3.2.6 *Model.Score Class Reference*

Public Member Functions

- void **AddScore** ()
Rajoute 10 de score à chaque ennemi tué

Public Attributes

- int **_score**

1.3.2.7 *Storage.Store Class Reference*

Public Member Functions

- void **Connect** ()
Tente de se connecter à une base de données MySQL.
- void **Close** ()
Ferme la connexion à la base de données.
- void **displayScore** (int widthWorld, int height)
Va chercher les pseudos et les scores des 5 meilleurs joueurs à la base de données.
- void **insert** (**Player** player, **Score** score, int widthWorld, int height)
Va insérer un score avec le pseudo dans la base de données.

Public Attributes

- MySqlConnection **connection**

Member Function Documentation

void Storage.Store.displayScore (int widthWorld, int height)

Va chercher les pseudos et les scores des 5 meilleurs joueurs à la base de données.

Parameters

<i>widthWorld</i>	
<i>height</i>	

void Storage.Store.insert (Player player, Score score, int widthWorld, int height)

Va insérer un score avec le pseudo dans la base de données.

Parameters

<i>player</i>	
<i>score</i>	

1.4 Test unitaire

Nom test	Classe	Méthode testée	Description	Condition réussite
AddAmmoTest	Player	AddAmmo	Ajoute des balles dans le chargeur du joueur	Nombre de balle ajoutée = nombre de balle dans le chargeur
PlayerMovement UpdateTest	Player	PlayerMovement Update	Le joueur se déplace d'un vers la droite ou la gauche	X du joueur augmente ou descend d'un
DownTest	Enemy	Down	L'ennemi descend de 2 qui correspond à sa taille	Y de l'ennemi augmente de 2
UpdateEnemyTest	Enemy	UpdateEnemy	L'ennemi se déplace d'un vers la droite ou la gauche	X de l'ennemi augmente ou descend d'un
AddScoreTest	Score	AddScore	Ajoute 10 au score	Si le score augmente de 10
AmmoUpdateTest	Ammo	AmmoUpdate	Test le déplacement vertical de la balle	Si la balle se déplace de -1 à la vertical

1.5 Conclusion

Ce projet m'a permis d'approfondir mes connaissances en POO. Le projet a rempli tous les critères dans le cahier des charges. Certain chose pourrait être améliorer notamment en ce qui concerne la modifiabilité de certaine valeur et l'indépendance de certaine classe. Aussi il y a la menue option et le menu règles qui peuvent être implémenter.

L'utilisation d'IceScrum a été compliqué au début car nous n'avons pas commencer tout de suite à l'utiliser. Le fait d'avoir un seul Scrum master pour tous les élèves et assez compliqué car techniquement nous ne pouvions pas commencer tant que l'user story n'avait pas été validé.

Le fait que le projet soit un jeu a été assez motivant. Le projet en général était assez amusant et instructive.

2 UX

2.1 Introduction

Lors du projet space invaders, un menu pour le jeu a été demandé. Il devait les contraintes suivantes :

- Option permettant de jouer solo ou de jouer en multijoueur
- Plusieurs palettes graphiques
- Plusieurs designs pour les ennemis
- Une page highscore
- Le choix de gameplay

Les différentes maquettes sont disponibles dans le dossier UX mais également via ce lien :

<https://www.figma.com/file/GPEciM51zBgYw1dANUbxZJ/Untitled?type=design&node-id=0-1&mode=design&t=GekVAswL9YVfJqn-0>

2.2 Analyse

2.2.1 Conception centrée utilisateur

La conception c'est centré sur 2 catégories de joueur, le joueur occasionnel et le joueur régulier. 2 persona ont été créés en leur donnant les informations suivantes :

- Informations personnelles
- Description/Graphique
- Ce qu'il aime
- Ce qu'il déteste

Les persona ont été générés grâce à chatGPT. Les voici :

Martin Dubois



Age : 19
Étudiant en informatique
Résidence : Angleterre
Passion : Speedrun

Description

Martin Dubois est un jeune de 19 ans résidant en Angleterre, passionné de jeux vidéo et de speedruns. Né le 15 mars 2004 à Manchester, il a découvert sa passion pour les jeux vidéo dès son plus jeune âge. À 12 ans, il a plongé dans le monde des speedruns, où il a affiné ses compétences et a même établi des records personnels impressionnants, principalement dans "Super Mario 64". Martin est également actif sur YouTube et Twitch, où il partage ses sessions de jeu en direct. En plus de ses exploits virtuels, il poursuit des études universitaires en programmation de jeux vidéo.

Son objectif est de réaliser des exploits de plus en plus grand et qui sont intéressants pour sa communauté.

Aime :

- Le jeu doit être en anglais car la communauté est anglaise et il veut que tout le monde comprenne ce qu'il se passe
- Le jeu doit être esthétique pour ne pas perdre l'attention de ses spectateurs
- Le jeu doit être challengeant pour lui car c'est un compétiteur dans l'âme

Déteste :

- Trop de texte il déteste que le jeu soit coupé dans l'action
- Le jeu soit chronophage sans challenge

Zoé Moreau



Age : 20
Etudiante en Art
Résidence : Suisse Lausanne

Description

Zoé Moreau, une jeune artiste de 16 ans vivant à Lausanne, est étudiante en art. Malheureusement, elle souffre de problèmes d'articulation de la main qui compliquent sa pratique artistique. Malgré ces difficultés, Zoé est déterminée à poursuivre sa passion pour l'art et à surmonter ses défis personnels. En dehors de son art, Zoé est une joueuse casual qui aime jouer à des jeux vidéo avec ses amis. Elle préfère les jeux simples et amusants qui lui permettent de se détendre et de passer du temps de qualité avec ses amis.

Aime :

- le fait de pouvoir changer les touches permet de soulager ses mains
- Zoé adore jouer avec ses amis, donc un jeu avec une composante multijoueur conviviale où elle peut se connecter et jouer facilement avec eux serait idéal.

Déteste :

- Elle ne serait pas intéressée par un jeu très complexe avec de nombreuses commandes, mécanismes et règles à apprendre, car elle préfère une expérience de jeu simple.
- Les jeux qui n'ont pas une identité visuelle assez forte risque de fortement lui déplaire

Ce choix a été fait pour avoir 2 types de joueur qui sont opposé dans leur manière de consommer et donc de toucher un maximum de personnes.

2.2.2 Palette graphique

Concernant les couleurs les couleurs choisies, elles s'inspirent du vrai jeu vidéo avec un fond noir et un texte blanc tout en rajoutant en fond, un background avec des étoiles qui rappelle l'espace. Il y a également du vert lorsque on sélectionne quelque chose dans le menu. Cela rajoute un côté rétro au jeu. Il y également un thème clair qui va inverser les couleurs sauf le vert. Voici le code hexadécimal les couleurs utilisés : noir = #000000, blanc = #FFFFFF, vert = #3EBD02.



2.2.3 Eco-conception

Pour l'éco-conception, plusieurs choses ont été fait en se basant sur les 115 règles de bonne conception :

- Un thème sombre a été fait pour limiter la consommation d'énergie des écrans
- Peu d'animation ont été intégré pour limiter la consommation
- Le parcours de l'utilisateur a été optimiser pour éviter une perte de temps dans les menus

2.2.4 Accessibilité

Pour l'accessibilité, il y a plusieurs choses. Le jeu est disponible en français et en anglais, le but étant de toucher un public plus large en le rendant disponible aux anglophones.

Il est possible de configurer les touches pour toutes les personnes ayant des problèmes d'articulation aux mains. Cela leur permet de soulager leur main en se mettant dans une configuration adaptée.

Le jeu est entièrement disponible au clavier.

Il y a un thème clair qui permet aux personnes ayant du mal à voir avec le thème sombre de pouvoir un autre thème qui permet de mieux voir.

Il y a une possibilité d'enlever le fond qui peut gêner la vision du jeu.

Il y a plusieurs difficultés permettant à des gens moins expérimentés de quand même pouvoir jouer sans frustration. Cela permet également aux gens voulant du défi de ne pas quitter le jeu car il est trop facile.

2.3 Conception

Lors de la conception, deux designs ont été fait. Un qui va être intégrer et l'autre qui ne sera pas intégrer.

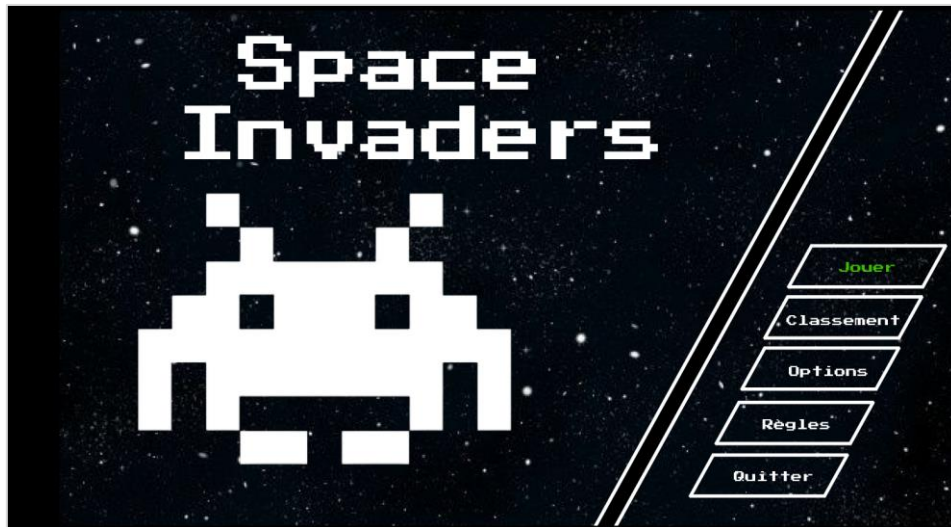
Pour le menu qui a été intégrer, des contraintes techniques dut au fait que le programme est fait en mode console ont dut être pris compte. Pour cette raison, le menu est très simpliste et n'ai que composé d'un fond noir et du texte blanc. Seulement un menu principal, un menu avec des règles et un menu des options et de configurations de touches ont été désignée. L'inspiration principale vient surtout des ancien jeux arcades (Pac-Man, Space Invaders).



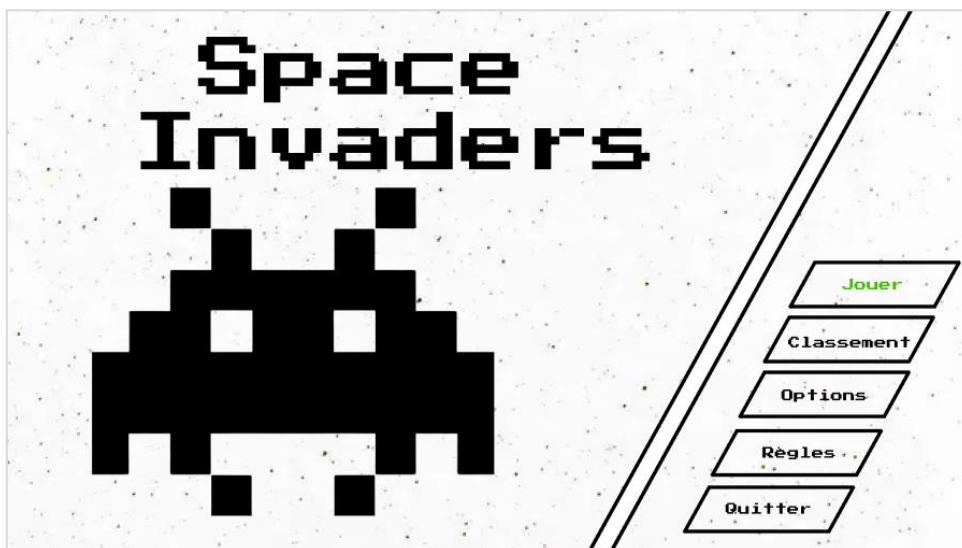
Pour le menu qui ne sera pas intégrer, les couleurs choisies reste fidèles au vrai jeu avec un fond noir du texte blanc et le curseur du joueur en vert. Voici les différentes pages :

- Menu principal
- Menu avec les différentes options et un autre pour la configuration des touches
- Menu avec les règles
- Menu avec le classement des 5 meilleurs joueurs
- Menu pour sélectionner 1 ou 2 joueurs
- Menu pour sélectionner si local ou en ligne (pour 2 joueurs)
- Écran d'attente d'adversaire
- Menu de sélection de vaisseau

Voici un exemple (menu principale) :



Il existe un thème clair pour ceux qui préfèrent.



Cela donne un côté rétro au jeu. Le design du menu principal a été inspiré par un jeu (Iconoclasts), cela donne un côté moderne au jeu.



Il est possible de faire plusieurs choses dans le menu options



Voici les différentes options :

- Changer la langue en anglais
- Changer le volume
- Changer le design des ennemis pour plus de variétés
- Changer de thème (sombre ou clair)
- Afficher ou non le fond avec les étoiles pour permettre une meilleure lisibilité

2.4 Test

Voici les tests d'utilisabilité :

Menu	Description	Précondi tion	Étape	Donnée	Résultat attendu
Classement	Le but est de voir le classement des 5 meilleurs joueurs local.	Avoir l'applicati on installé	Dans le menu, cliquez sur classement	Aucun	Classement des 5 meilleurs joueurs
Lancer une partie en ligne	Le but est de lancer une partie en ligne	Avoir l'applicati on installé, être connecté à internet	Dans le menu, cliquez sur Jouer Cliquez sur 2P Cliquez sur En ligne	Aucun	Une page avec marqué "Recherche d'un autre joueur"
Sélectionner un vaisseau	Le but est de sélectionner un des vaisseau débloqué	Avoir l'applicati on installé	Dans le menu, cliquez sur Jouer Cliquez sur 1P Cliquez sur Facile Cliquez sur un vaisseau débloqué	Aucun	Une image 3D du vaisseau sélectionné avec ses stats et son nom
Changer la langue	Le but est de changer la langue	Avoir l'applicati on installé	Dans le menu, cliquez sur option Cliquez sur changer la langue	Aucun	Cela va changer le jeu en une autre langue (FR -> ENG, ENG -> FR)
Changer de thème	Le but est de changer de thème	Avoir l'applicati on installé	Dans le menu, cliquez sur option Cliquez sur thème	Aucun	Cela va le thème sombre en clair et vice versa

2.5 Conclusion

Ce projet a permis d'utiliser la théorie dans un cas concret. Pour ma part j'ai eu beaucoup de mal pour la partie design. Pour le reste cela était assez divertissant de faire une maquette sur un jeu. Toute les demandes dans le cahier des charges ont été respectées. Si je devais refaire le projet, j'aurais essayé de faire la maquette d'en un design plus démarqué et moins simpliste/minimaliste. La partie prototype était assez amusante.

3 DB

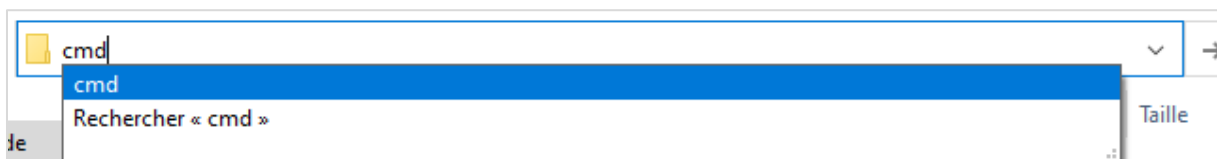
3.1 Introduction

Dans le cadre du projet space invaders, des demandes concernant la base de données du jeu ont été demandées. Les voici :

- Importer une base de données et faire une sauvegarde
- Faire la gestion d'utilisateur et de rôles
- Expliquer 10 requêtes
- Expliquer l'intérêt des index et s'il était utile d'en avoir un

3.2 Importer les données et le schéma de base de données

Lancez le terminal (conseil : allez dans l'endroit où il y a la base de données est dans le chemin d'accès mettez cmd) :



Puis mettez la commande suivante :



```
docker exec -i id_docker mysql -uroot -proot < db_space_invaders.sql
```

Docker = met en lien avec l'application docker

exec = exécuter

-i = identifiant docker

Id_docker = identifiant du container

mysql = met en lien avec MySQL

-u = utilisateur MySQL

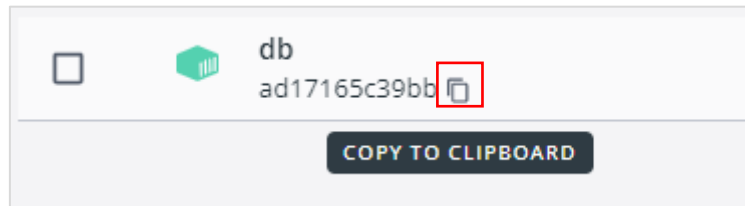
-p = mot de passe

root = nom et mot de passe admin dans MySQL par défaut

< = importer vers MySQL

db_space_inavaders.sql = chemin d'accès à la base de données

(PS : Pour copier l'identifiant docker, il suffit d'aller dans docker, puis le menu des containers et de cliquer sur l'icône pour copier)



Normalement cela devrait avoir implémenter la base de données sur le serveur mysql.

3.3 Gestion des utilisateurs

Tout d'abord nous allons créer 3 utilisateurs : Alice (administratrice de jeu), Bob (joueur), Charly (gestionnaire de la boutique)

Pour créer les utilisateurs, nous allons utiliser la commande ci-dessous :



```
CREATE USER 'bob'@'localhost' IDENTIFIED BY 'root';
```

'bob' = nom de l'utilisateur

'localhost' = L'host depuis lequel il a le droit de se connecter

'root' = mot de passe

Je vais le faire pour les 2 autres utilisateurs.



```
CREATE USER 'alice'@'localhost' IDENTIFIED BY 'root';  
CREATE USER 'charly'@'localhost' IDENTIFIED BY 'root';
```

On va vérifier si les utilisateurs ont été créés, pour cela il faudra sélectionner la base de données MySQL puis faire une requête.

```
USE mysql;  
SELECT user from user;
```

Normalement vous devriez voir les différents noms des utilisateurs.

Maintenant nous allons créer les rôles pour les différents besoins (administratrice de jeu, joueur, gestionnaire de la boutique). Le but d'un rôle est de pouvoir donner ce rôle à beaucoup des utilisateurs sans avoir à redonner des rôles à chaque fois.

Donc pour créer un rôle, il suffit de faire la commande ci-dessous :

```
CREATE ROLE 'r_player';  
CREATE ROLE 'r_admin';  
CREATE ROLE 'r_shopkeeper';
```

'r_player' = nom du rôle

Ensuite on va donner aux rôles des privilèges, pour ça nous allons faire la commande suivante :

```
GRANT ALL PRIVILEGES ON *.* TO 'r_admin' WITH GRANT OPTION;
```

```
GRANT SELECT ON db_space_invaders.t_arme TO 'r_player';  
GRANT INSERT, SELECT ON db_space_invaders.t_commande TO 'r_player';  
GRANT SELECT ON db_space_invaders.t_detail_commande TO 'r_player';
```

```
GRANT INSERT, DELETE, UPDATE, SELECT ON db_space_invaders.t_arme TO 'r_shopkeeper';  
GRANT SELECT ON db_space_invaders.t_joueur TO 'r_shopkeeper';  
GRANT SELECT ON db_space_invaders.t_commande TO 'r_shopkeeper';
```

(Ici on va rajouter un droit sur la table joueur car il en a besoin pour créer une commande)

Explication de la requête :


GRANT = indique qu'on va donner chose

SELECT, ALTER, DELETE, ... = les permissions données

ON db_space_invaders.t_joueur = base de donnée.table (* = tout)

TO 'r_shopkeeper' = indiquer qui est concerné par cette commande (par exemple : r_shopkeeper)

Maintenant on va donner les rôles aux utilisateurs, pour ça on va utiliser la commande suivante :



```
GRANT 'r_shopkeeper' TO 'charly'@'localhost';  
GRANT 'r_admin' TO 'alice'@'localhost';  
GRANT 'r_player' TO 'bob'@'localhost';
```

Cela va donner les rôles aux différents utilisateurs (à noter qu'on peut rajouter plusieurs utilisateurs à la fois).

3.4 Requêtes de sélection

3.4.1 Explication quelques commandes

SELECT = permet de sélectionner des colonnes

FROM = indique depuis quelle table on fait le select

WHERE = permet de mettre une condition aux données

ORDER BY = permet de mettre en ordre les résultats (par défaut croissant mais peut être décroissant grâce au mot DESC)

LIMIT = permet de mettre un nombre défini de résultat

MAX = sélectionne le plus grand

MIN = sélectionne le plus petit

AVG = Fais la moyenne

AS = permet de renommer un champ (si on l'utilise, on doit utiliser le surnom si la méta donnée réapparaît)

COUNT = compte le nombre de données lorsqu'il y a un group by

GROUP BY = permet de regrouper par ligne qui ont la même donnée

HAVING = permet de mettre une condition au regroupement, ne peut être utilisé qu'après des GROUP BY

INNER JOIN (ou JOIN) = permet de joindre 2 tables grâce aux clés primaires (id) et aux clés étrangères (fk) qu'on va regrouper avec le ON et puis regroupe uniquement ceux qui sont en commun

LEFT JOIN = permet de joindre 2 tables grâce aux clés primaires (id) et aux clés étrangères (fk) qu'on va regrouper avec le ON et puis regroupe tout ce qui sont en commun puis toutes les informations de la table qui est sur le FROM même s'il n'est pas sur l'autre table

RIGHT JOIN = la même chose que LEFT JOIN mais au lieu de mettre toutes les informations de la table qui est sur le FROM, ce sera la table avec laquelle on va relier (après le RIGHT JOIN) qui aura toutes les informations même s'il n'apparaît pas sur l'autre table

SUM() = va faire la somme de tous les chiffres

DISTINCT = permet d'éviter les doublons

3.4.2 Requêtes

Voici les différentes requêtes :

Requête n°1 :



```
SELECT * FROM t_joueur ORDER BY jouNombrePoints DESC LIMIT 5;
```

Cela va tout (*) sélectionner [SELECT] de la table *t_joueur* [FROM], par ordre [ORDER BY] décroissant [DESC] par rapport au nombre de point et se limitant [LIMIT] au 5 premier résultat.


Requête n°2 :



```
SELECT MAX(armPrix) AS PrixMaximum, MIN(armPrix) AS PrixMinimum, AVG(armPrix) AS PrixMoyen FROM t_arme;
```

Cela va sélectionner l'arme la plus chère [MAX()] et renommer la colonne *PrixMaximum* grâce au [AS], l'arme la moins chère [MIN()] et renommer la colonne *PrixMinimum* et le prix moyen des armes [AVG()] et renommer la colonne *PrixMoyen* depuis la table *t_arme*.


Requête n°3 :



```
SELECT fkJoueur AS IdJoueur, COUNT(idCommande) AS NombreCommandes FROM t_commande GROUP BY IdJoueur ORDER BY NombreCommandes DESC;
```

Cela va sélectionner l'id des joueurs en renommant la colonne *idJoueur*, le nombre de commande [COUNT()] et renommer la colonne *NombreCommande* depuis la table *t_commande*. Puis les grouper (group by) par l'id des joueurs et les ordonner par le nombre de commande décroissant.

Requête n°4 :



```
SELECT fkJoueur AS IdJoueur, COUNT(idCommande) AS NombreCommandes FROM t_commande GROUP BY IdJoueur HAVING NombreCommandes > 2;
```

Cela va sélectionner l'id des joueurs en renommant la colonne *idJoueur*, le nombre de commande et renommer la colonne *NombreCommande* depuis la table *t_commande*. Puis les grouper par l'id des joueurs qui ont plus de 2 commandes [HAVING].

Requête n°5 :



```
SELECT t1.jouPseudo, t2.idCommande, t4.armNom FROM t_joueur AS t1
INNER JOIN t_commande AS t2 ON t2.fkJoueur = t1.idJoueur
INNER JOIN t_detail_commande AS t3 ON t3.fkCommande = t2.idCommande
INNER JOIN t_arme AS t4 ON t4.idArme = t3.fkArme;
```

Cela va sélectionner le nom du pseudo (table : *t_joueur*), l'id de la commande (table : *t_commande*) et le nom de l'arme (table : *t_arme*). Puis on va joindre la table *t_joueur* et la table *t_commande* via l'id du joueur [**INNER JOIN**], on va joindre la table *t_detail_commande* et la table *t_commande* via l'id des commandes et on va joindre la table *t_detail_commande* et la table *t_arme* via l'id de l'arme.

L'**INNER JOIN** a été choisi car il regroupe uniquement ce qui est en commun car certaines personnes n'ont pas acheté d'arme.

Requête n°6 :



```
SELECT t3.fkJoueur AS idJoueur, SUM(t2.detQuantiteCommande * t1.armPrix) AS TotalDepense FROM t_arme AS t1
INNER JOIN t_detail_commande AS t2 ON t2.idArme = t1.fkArme
INNER JOIN t_commande AS t3 ON t2.fkCommande = t3.idCommande
GROUP BY idJoueur ORDER BY TotalDepense DESC LIMIT 10;
```

Cela va sélectionner l'id des joueurs (table : *t_commande*), la somme de tous les calculs prix de l'arme (table : *t_arme*) * la quantité (table : *t_detail_commande*), tout ça regroupé par joueur. On va joindre *t_detail_commande* à *t_arme* puis on va regrouper *t_commande* avec *t_detail_commande* pour avoir accès à l'id des joueurs et au nombre de quantité de commande.

On va regrouper le tout par joueur, mettre en ordre décroissant par le total dépensé et ne mettre que les 10 premières réponses.

Requête n°7 :



```
SELECT t1.jouPseudo, t2.idCommande FROM t_joueur AS t1
LEFT JOIN t_commande AS t2 ON t2.fkJoueur = t1.idJoueur;
```

On va sélectionner le pseudo des joueurs (table : *t_joueur*) et leur commande (table : *t_commande*). On va relier les joueurs à leur commande via l'id des joueurs. Ici on fait un **LEFT JOIN** pour avoir accès à tous les joueurs même s'ils n'ont pas passées de commandes.

Requête n°8 :

```
SELECT t1.jouPseudo, t2.idCommande FROM t_joueur AS t1
RIGHT JOIN t_commande AS t2 ON t2.fkJoueur = t1.idJoueur;
```

On va sélectionner le pseudo des joueurs (table : *t_joueur*) et leur commande (table : *t_commande*). On va relier les joueurs à leur commande via l'id des joueurs. Ici on fait un *RIGHT JOIN* pour avoir accès à toutes les commandes même si le joueur n'existe plus.

Requête n°9 :

```
SELECT t1.idJoueur, SUM(t3.detQuantiteCommande) FROM t_joueur AS t1
LEFT JOIN t_commande AS t2 ON t2.fkJoueur = t1.idJoueur
LEFT JOIN t_detail_commande AS t3 ON t2.idCommande = t3.fkCommande
GROUP BY t1.idJoueur;
```

On va sélectionner l'id des joueurs (table : *t_joueur*), la somme [**SUM()**] de ce qu'ils ont commandé (table : *t_detail_commande*). Puis on va joindre la table des joueurs et des commandes pour savoir quel joueur a fait quelle commande puis la table des commandes avec celle des détails des commandes pour avoir la quantité de chaque commande. Le tout regrouper par joueur. On utilise des *LEFT JOIN* pour avoir accès à tous les joueurs même s'ils n'ont rien commandé.

Requête n°10 :

```
SELECT t1.idJoueur, t1.jouPseudo, COUNT(DISTINCT t3.fkArme) FROM t_joueur AS t1
INNER JOIN t_commande AS t2 ON t2.fkJoueur = t1.idJoueur
INNER JOIN t_detail_commande AS t3 ON t2.idCommande = t3.fkCommande
GROUP BY t1.idJoueur HAVING COUNT(DISTINCT t3.fkArme) > 3;
```

On va sélectionner l'id (table : *t_joueur*), le pseudo (table : *t_detail_commande*) et on va compter le nombre d'arme différents dans la liste de commande de chaque joueur. Le *DISTINCT* permet d'éviter que 2 commandes avec la même arme fait par le même joueur ne soit compter 2 fois. Puis on va joindre la table des joueurs et des commandes pour savoir quel joueur a fait quelle commande puis la table des commandes avec celle des détails des commandes pour savoir quelle arme est commandé. Le tout regrouper par joueur et on va trier par le nombre d'arme différente que chaque joueur a commandé, on va uniquement garder ceux qui ont commandé plus de 3 armes différentes.

3.5 Index

- 1) Les 3 index qui sont créés automatiquement sont pour les colonnes id (clé primaire), fk (clé étrangère), et les colonnes qui sont uniques. Ils sont créés automatiquement car les id et les fk sont souvent utilisés pour les requêtes ce qui permet de les accélérer. Pareil pour les champs uniques.
- 2) Le but d'un index est d'accélérer les requêtes selectes en accélérant la navigation dans les colonnes. Pour cela il crée un arbre B-Tree. Cela permet d'aller plus vite dans les requêtes selectes mais en contrepartie cela va prendre plus de mémoire et va rendre les insertions, modifications et les suppressions plus lentes.
- 3) Toutes les tables sont souvent mises à jour sauf la table arme. Cela pourrait être intéressant de mettre un index composite dans l'ordre suivant : armNom, armPrix, armDescription, armForce et armNombreCoups. Cela serait intéressant dans le cas où on veut souvent savoir des informations sur les armes, par exemple pour voir l'inventaire ou bien s'il y a un shop ce qui nécessiterait de souvent faire un select. L'index composite serait utile car lorsque on voit une arme on affiche souvent les informations la concernant et ce ne que rarement qu'on update la table.

3.6 Sauvegarde/Restore

3.6.1 Sauvegarde

Voici la commande pour faire une sauvegarde :



```
docker exec -id id_docker mysqldump --databases -uroot -proot  
db_space_invaders > db_space_invaders.sql
```

docker exec -i id_docker = cette partie sert à utiliser le docker où il y a la base de données (remplacé id_docker par l'id de votre docker)

mysqldump = mysqldump est un utilitaire permettant de sauvegarder et de restaurer des bases de données MySQL

--databases = va rajouter dans le fichier un code qui crée la base de données, cela évite d'avoir un problème lors de la restauration car il faut lui donner une base de données ou la créer

-uroot -proot = permet de se connecter à la base de données (-u = utilisateur, -p = mot de passe)

db_space_invaders = nom de la base de données qu'on aimerait sauvegarder

> db_space_invaders.sql = nom du fichier dans le lequel on va stocker la base de données

3.6.2 Restaurer

Voici la commande pour restaurer une base de données :



```
docker exec -i id_docker mysql -uroot -proot < db_space_invaders.sql
```

docker exec -i id_docker = cette partie sert à utiliser le docker où il y a la base de données (remplacé id_docker par l'id de votre docker)

mysql = va mettre un lien avec MySQL

-uroot -proot = permet de se connecter à la base de données (-u = utilisateur, -p = mot de passe)

< db_space_invaders.sql = chemin d'accès vers la sauvegarde

3.7 Intégration de la base de données dans le code

Le code est disponible dans le dossier POO, SpicyConso et le SpicyConso.sln.

Pour la connexion au serveur MySQL, voici ce qu'il faut :

- De l'adresse IP du serveur (server), ici ça sera localhost (127.0.0.1) car le serveur est stocké sur la même machine que le programme
- Un *user name* (userID) qui sera root car la connexion avec d'autres utilisateurs n'a pas marché
- Le mot de passe du *user* (pwd) qui ici sera root car il n'a pas été changé
- Le nom de la base de données (database) qui sera *db_space_invaders*
- Le port (port), celui-ci normalement n'a pas besoin d'être précisé mais le serveur MySQL est stocké sur Docker est utilisé le port 6033, si nous voulons accéder à la base de données MySQL nous devons passer par ce port

Nous allons mettre toutes ces informations dans une variable (connectString) pour pouvoir se connecter à la base de données

3.8 Conclusion

Cette partie a permis de renforcer ma connaissance en termes de requêtes MySQL et m'a permis de découvrir comment intégrer la base de données dans du code c#.

Toutes les demandes du cahier des charges ont été respecté. Si le projet était à refaire, je changerais la manière dans le rapport écrit pour que ce soit mieux synthétisé et surtout d'avoir les bonnes pratiques dès le début car il m'est arrivé plusieurs fois de recommencer certaine partie car je n'avais pas fait directement les bonnes pratiques, par exemple : mettre en fond blanc les images pour économiser de l'encre, mettre certain mot en gras ou italique, etc. Il y a également un problème en ce qui concerne la connexion car je n'ai pas réussi à me connecter avec un autre *user* que root.

4 ChatGPT

4.1 POO

Je ne l'ai utilisé qu'une seule fois pour trouver une erreur dans le code, il ne m'a pas du tout aidé. L'erreur concernait une sortie de boucle qui ne faisait pas ce que je voulais, finalement la solution a été trouvé après débogage de la boucle.

4.2 UX

Je l'ai utilisé pour la création de Persona.

4.3 DB

Je ne l'ai pas utilisé.

5 Conclusion générale

Ce projet m'a permis de renforcer mes connaissances générales en informatique le tout de manière assez ludique. Je pense que la partie gestion de projet est assez compliquée à gérer car nous n'avons pas directement commencer à l'utiliser. Et le fait de n'avoir qu'un seul Scrum Master par projet pour 15 élèves est assez compliquer surtout si on doit attendre qu'il valide les User Story.