Projet Spicy Invader

UX, OO, DB

Ethan Schafstall

28.08.2023-03.11.2023

**Table des matières**

[Expérience utilisateur (UX) 1](#_Toc149846735)

[Introduction 1](#_Toc149846736)

[Analyse 1](#_Toc149846737)

[Conception 3](#_Toc149846738)

[Évaluation 5](#_Toc149846739)

[Programmation orientée objet (OO) 5](#_Toc149846740)

[Introduction 5](#_Toc149846741)

[Analyse fonctionnelle 5](#_Toc149846742)

[Analyse technique 5](#_Toc149846743)

[Tests Unitaire 5](#_Toc149846744)

[ChatGPT 5](#_Toc149846745)

[Conclusion 6](#_Toc149846746)

[Base de données (DB) 6](#_Toc149846747)

[Introduction 6](#_Toc149846748)

[Importation données et schéma de base de données 6](#_Toc149846749)

[Gestions des utilisateurs 6](#_Toc149846750)

[Requêtes de sélection 7](#_Toc149846751)

[Création des index 10](#_Toc149846752)

[Backup/Restore 10](#_Toc149846753)

# **Expérience utilisateur (UX)**

## **Introduction**

L'**UI** et l'**UX** sont parmi les éléments les plus importants dans le développement d'un jeu ou de tout autre produit. Même avec les meilleures technologies, si l'utilisateur a du mal à comprendre, naviguer et a généralement une mauvaise expérience, alors tout le travail est une défaite pour le **PO** (Product Owner) et les développeurs. Il est donc important d'identifier les besoins des utilisateurs pour que le produit soit une réussite.

## **Analyse**

### **L’Inclusivité/l’accessibilité**

Dans le domaine des médias, il y a un problème d’inclusion de groupes marginalisées, toujours en temps moderne. Que ça soit le manque de représentation de certaines ethnies dans le cinéma. Le manque d’options pour navigation pour personnes avec handicapes. Les personnes de différentes zones géographiques, différentes langues, ainsi que plein d’autres.

### **L’Inclusivilité : Représentation des femmes**

Dans le contexte de l’informatique et des jeux vidéo, c’est le manque de reconnaissance des femmes en tant que clientèle potentielle, et utilisateurs actifs des produits. L’idée qu’il y a qu’il y a seulement les males (ados surtout), et que les filles/femmes ne joue pas au jeux vidéo est un mythe qui percés malgré les études de haute éducation, et les études de marché faites dans les dernières+ 20 ans.

C’est pendant que phase de conception d’un produit ou, en tant que développer qu’il y a une responsabilité d’identifier les clients/utilisateurs des produits et de leur offrir une meilleure expérience.

### **L’accessibilité : Localisation**

Dans un monde globalisé, Il y a une importance à fournir de différentes options (langue, monnaie, horaire, etc.). Dans le contexte de jeux, avoir plusieurs options de langues pour pouvoir naviguer les menus est nécessaire, mais si le jeu en lui-même a très peu d’éléments de lecture.

### **L’accessibilité : Déficience/Handicap visuelle**

Pour toutes formes de “l’Entertainment“, surtout toutes media en rapport avec un écran, il faut avoir des adaptations pour personnes avec déficiences ou handicaps.

Dans le groupe de déficience/ et handicap visuelle, il y a :

* Personnes avec daltonisme
* Personnes âgées
* Personnes myopes ou hypermétropes
* Personnes avec TSA, ou autres troubles néologiques

Le choix d’une fonte qui soit facilement lisible a de différentes distances/tailles, avec choix de couleurs approprié.

### **Éco-conception**

L'éco-conception est un concept très important en tant que développeur, car les applications vont toujours consommer des ressources énergétiques à chaque usage. Il est donc important d'optimiser au maximum et d'utiliser le plus de techniques pour éviter l'utilisation de ressources inutilement.

Pour un jeu, il y a plusieurs moyens :

* Le choix de la palette de couleurs pour le jeu/UI.
* Le nombre d'objets à l'écran pendant le gameplay.
* La fréquence de la connectivité à la base de données.

Autant d'autres peuvent avoir un meilleur impact écologique, avec le minimum de sacrifice pour l'expérience du joueur.

### **Innovation**

Pour donner créer une meilleure expérience pour l’utilisateur c’est important d’etre innovatif.

L’implémentation de idées qui créer de nouvelles/améliore le produit ou service.

Pour les jeux il y a pleins de méthodes pour le rendre plus innovatif :

Le gameplay

Le design des personnages

Les options de personnalisation

Pour créer une meilleure expérience pour l’utilisateur, il est important d’être innovatif. L’implémentation d'idées qui créent de nouvelles améliorations pour le produit ou service.

Pour les jeux, il existe de nombreuses méthodes pour les rendre plus innovatifs :

* Le gameplay
* Le design des personnages
* Les options de personnalisation

Ce sont quelques-unes des méthodes où la créativité des développeurs joue un rôle clé, tout comme la capacité à repousser les limites de la technologie utiliser pour offrir des expériences uniques.

## **Conception**

### **Persona**

Une persona est censée être une représentation fiable et réaliste du public clé pour un produit, donc c'était l'opportunité d'utiliser des femmes.

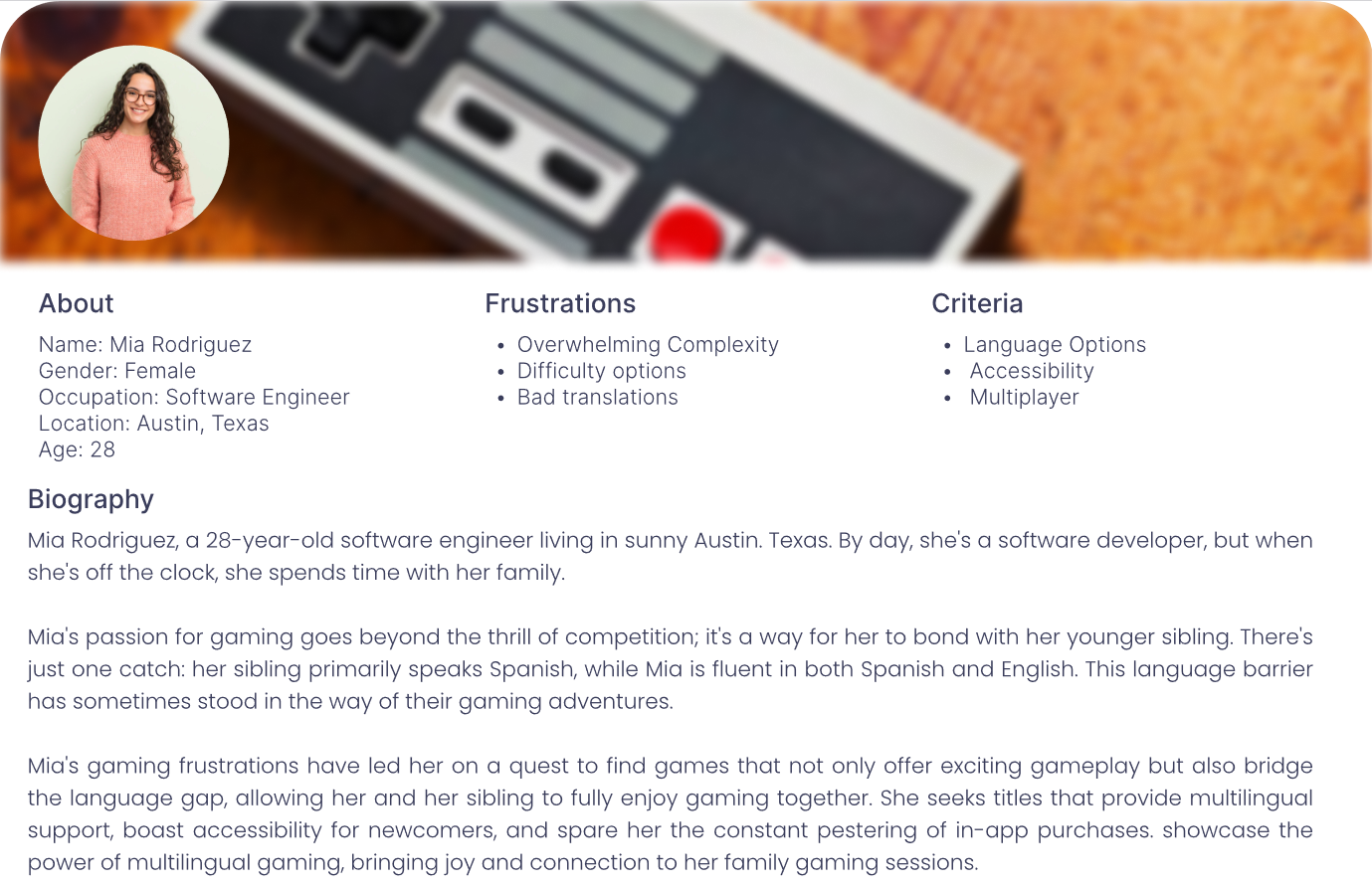
Les deux personas crée ce sont des femmes, chacune avec une profession, un âge de plus de 25 ans, et une biographie qui ne correspond pas typiquement au stéréotype du jeune "gamer" masculin, mais qui est plus représentative du public actuel qui s'intéresse aux jeux.

### **Persona : Mia Rodriguez**

**Mia Rodriguez**, est une jeune développeuse du Texas, mais avec des origines hispaniques. Elle est très bilingue en anglais et espagnol et est très proche de sa famille.

Avec Mme **Rodriguez**, il était important de se focaliser sur la notion de localisation et du partage. Elle utilise les jeux comme moyen de se connecter à sa jeune frangine, qui ne parle pas que l'espagnol, et veut qu'elle puisse en profiter autant que sa grande sœur.

Donc ce persona met en évidence l'importance de proposer différentes options linguistiques, pour que tout le monde puisse utiliser, comprendre et naviguer dans le jeu facilement. Elle montre aussi que toutes les personnes qui vont jouer au jeu n'ont pas forcément de l'expérience à jouer aux jeux vidéo, que ce soit des personnes âgées, des jeunes comme dans le cas de sa sœur, ou autres. Il est important de simplifier le plus possible pour le rendre accessible.

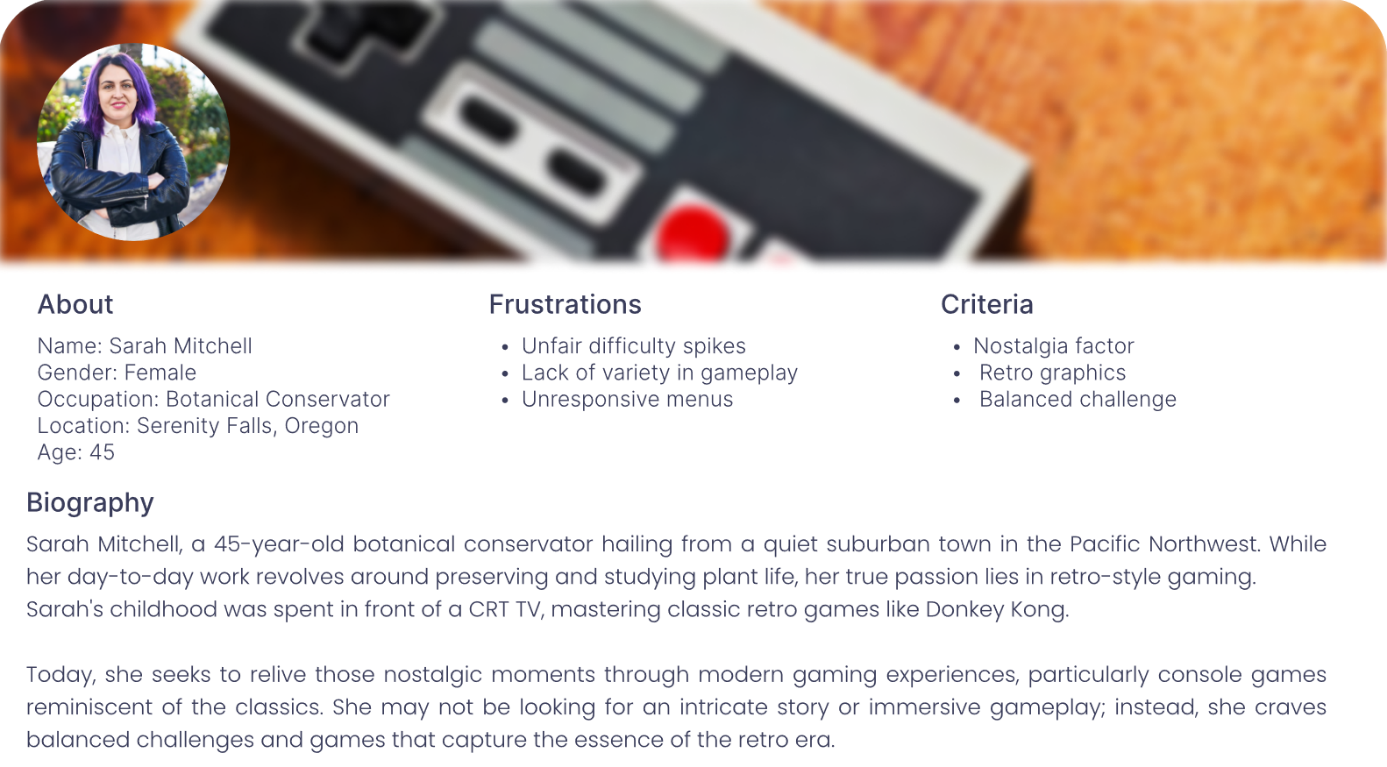


### **Persona : Sarah Mitchell**

**Sarah Mitchell** est une conservatrice botanique âgée de 45 ans, originaire d'une petite ville en Oregon.

**Mitchell** représente la clientèle qui a grandi durant l'âge d'or des jeux, où les gens passaient leurs week-ends à l'arcade ou à jouer sur leur Atari 2600. Ceux qui ne s'identifient pas avec les jeux modernes, ou peut-être préfèrent juste d'autres options.

Cette persona met en évidence l'importance de la nostalgie pour de nombreuses personnes qui recherchent les mêmes expériences qu'elles ont eues pendant leur enfance: des temps plus simples pour les jeux en ce qui concerne les graphismes, le gameplay, la musique et le choix des jeux. Pas trop difficiles, mais un défi correct.

****

### **Maquette**

### **Maquette : Basse Fidélité**

### **Maquette : Haut Fidélité**

## **Évaluation**

Le projet s'est globalement bien déroulé avec les différentes phases, et les travaux rendus correspondent aux critères du cahier des charges. La phase analytique a permis l'identification de potentiels problèmes, besoins, et améliorations qui pourraient être abordés.

La phase de conception a ensuite permis la mise en œuvre des différentes solutions et choix de design. Le choix des personas, de la palette de couleurs, des maquettes, etc. Tout cet ensemble a servi à créer de la valeur pour le plus grand nombre d'utilisateurs. Mais aussi à promouvoir les idées d’accessibilité et d’inclusivité

# **Programmation orientée objet (OO)**

## **Introduction**

La partie programmation orientée objet du projet comprends tout le code et fonctionnalité, du jeu, avec sa complexité, c’est nécessaire d’utiliser le classes et objets.

Comme la majorité des jeux, il utilise un design pattern de **MVC** (Model-View-Controller), attribué l’un de trois responsabilités à chaque classe.

Le **Model** est tous qui concerne les données et la logique de notre jeu, dont les classes d’ennemies, du vaisseau joueur, etc.

Le **View** est pour toute affichage graphique de l’interface et des modèles.

Le **Controller** c’est pour gérer toutes logique concernant les actions effectuées par l’utilisateur

## **Analyse fonctionnelle**

### **Move the player ship horizontally**

|  |
| --- |
| As a player I want to move the player ship horizontally In order to avoid enemy projectiles |
| Tests d'acceptance:   |  |  | | --- | --- | | Move left | \_\*Given\*\_ during gameplay \_\*When\*\_ I press leftarrow or a key \_\*Then\*\_ my ship moves left | | move right | \_\*Given\*\_ during gameplay \_\*When\*\_ I press rightarrow or d key \_\*Then\*\_ my ship moves right | |

### **implement vector based movement.**

|  |
| --- |
| As a développer I want to implement vector based movement. In order to streamline movement based classes. |
| Tests d'acceptance:   |  |  | | --- | --- | | Creating an instance of the Vector class | Given: The Vector class is available. When: I create a new Vector instance with values (3, 4). Then: The Vector instance should have x and y values set to 3 and 4, respectively. | | Adding two vectors together. | Given: Two Vector instances are available: Vector A with values (2, 3) and Vector B with values (1, -1). When: I add Vector A and Vector B together. Then: The result should be a new Vector instance with values (3, 2). | | Updating a class to include Vector-based movement. | Given: A class named "MoveableEntity" exists, representing a game object. When: I update the "MoveableEntity" class to include Vector-based movement by adding a "velocity" property of type Vector. Then: The "MoveableEntity" class should now have a "velocity" attribut, allowing objects to move using vectors. | | Moving a game object using Vector-based movement. | Given: A "MoveableEntity" instance with position (5, 5) and a "velocity" of (2, 2). When: I update the game state to move the object according to its velocity and move direction for one time step. Then: The "MoveableEntity" should have a new position of (7-/+2, 7-/+2). | |

### **Projectile enemy interation**

|  |
| --- |
| As a user I want projectile enemy interation In order to make the gameplay interactive |
| Tests d'acceptance:   |  |  | | --- | --- | | Player projectile hits Enemy | \_\*Given\*\_ I'm playing the game \_\*When\*\_ one of the player's projectiles hit an enemy \_\*Then\*\_ I get a visual confirmation that the enemy was hit | | Player projectile destroys Enemy | \_\*Given\*\_ I'm playing the game \_\*When\*\_ one of the player's projectiles hit an enemy with lower healthpoints than projectile damage \_\*Then\*\_ I get a visual confirmation that the enemy was destroyed | | Enemy projectile destroys Player | \_\*Given\*\_ I'm playing the game \_\*When\*\_ one of the Enemy's projectiles hit the player ship \_\*Then\*\_ I should get a visual confirmation that the player ship was destroyed | | Remove destroyed Enemies | \_\*Given\*\_ I'm playing the game \_\*When\*\_ after I've destroyed an enemy \_\*Then\*\_ the destroyed enemy does not appear on the screen | |

### **Create UML**

|  |
| --- |
| As a développer I want to a UML diagram In order to understand/visualize my programs classes and relations. |
| Tests d'acceptance:   |  |  | | --- | --- | | UML conventions | \_\*Given\*\_ The file docs/invaders.drawio (ou quelque soit le nom de votre fichier) \_\*When\*\_ I open it \_\*Then\*\_ the diagram is displayed, the UML conventions are respected | | Contains all classes, attributs, and functions. | \*Given\*\_ The file docs/invaders.drawio (ou quelque soit le nom de votre fichier) \_\*When\*\_ I open it \_\*Then\*\_ all the classes, their attributs and functions from the projet solution are in the UML | |

## **Analyse technique**

### **Diagramme de classe**

### **Explications**

## **Tests Unitaire**

## **ChatGPT**

Dans le contexte de ce projet **ChatGPT** a été utiliser à plusieurs stades de développement.

Aux premiers stades du projet, dans la phase de conception du programme, les classes, etc. **ChatGPT** est utiliser pour mieux comprendre les fonctionnalités, et interactions de l’héritage en C#. Les classes **SmartEntity**, **MoveableEntity**, **Projectile**, ce sont exemplaires de l’utilisation correct d’héritage en C#. Ces structures dans le code ont été conçu, en parti, grâce aux explications, et exemplaires fournis par **ChatGPT.**

Durant les stades de développement/programmation du jeu, **ChatGPT** est utiliser pour le renommage de classes, variables, méthodes, pour rendre le code plus compréhensible sans le besoin de commentaires pour chaque ligne de code.

Pendant cette phase de développement **ChatGPT** est aussi utiliser dans l’optimisation de code, pour avoir une meilleure performance, ainsi pour éviter de la redondance.

## **Conclusion**

# **Base de données (DB)**

## **Introduction**

La fonctionnalité de **SQL** et l'utilisation d'une base de données dans ce jeu servent à offrir une meilleure expérience pour l'utilisateur, ainsi que de pouvoir récolter des données pertinentes. Ce chapitre comprend la partie technique de la base de données du jeu, les différentes requêtes pour la récupération de données, la gestion des utilisateurs/rôles, les index, ainsi que la protection de notre base de données via les sauvegardes et les restaurations.

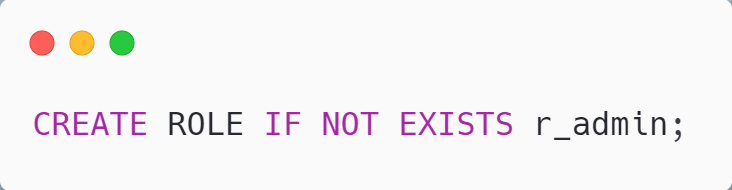
## **Importation données et schéma de base de données**

## **Gestions des utilisateurs**

La gestion des utilisateurs et des rôles est une étape importante pour définir et limité l’accès de certains donnés.

### **Administrateur de jeu**

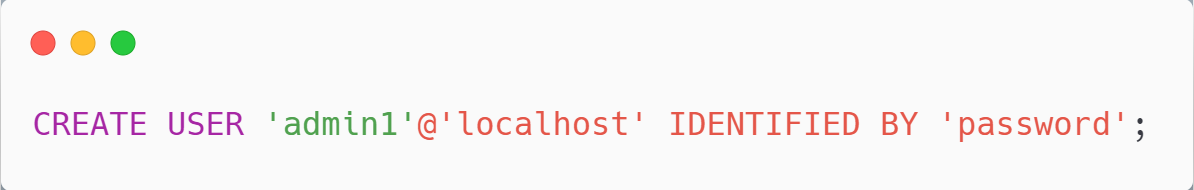
Création du rôle **r\_admin** avec le command **CREATE**.



Donne les privilèges au rôle **r\_admin** avec command **GRANT**. **WITH GRANT OPTION** et **\*.\*** pour donner tous les privilèges au rôle, donc créer un rôle "super user"



La création d’un user "admin" avec **CREATE USER**, nommée admin1

****

L’ajoute du user admin1 au rôle **r\_admin** avec la commande **GRANT ROLE**.

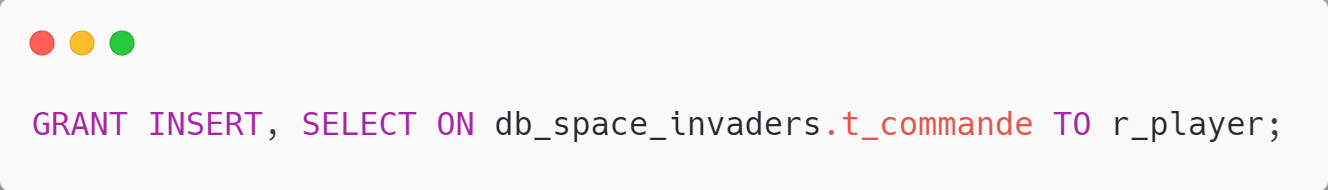
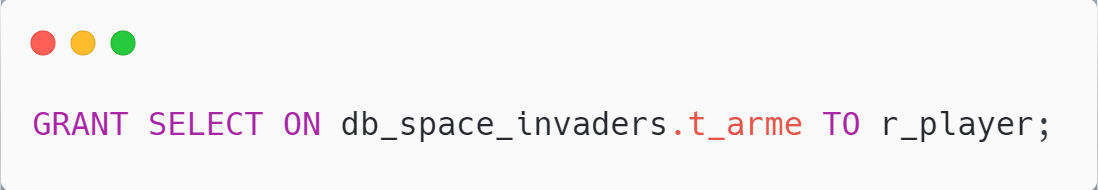


### **Joueur**

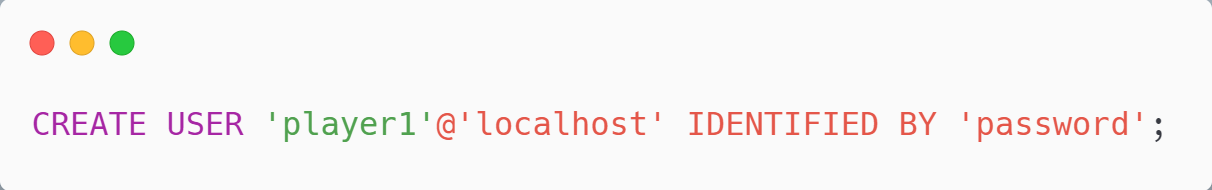
Création du rôle **r\_player** avec le command **CREATE**.



Donne les privilèges au rôle r\_player avec command **GRANT** **SELECT ON db\_space\_invaders.t\_arme** car le joueur n’a que le droit de récupérer des données sur les armes. **GRANT** **INSERT, SELECT ON db\_space\_invaders.t\_commade** car le jour à le droit d’insérer, et récupérer des donnes dans la table commande. Pour que le joueur puisse voir les infos sur les armes, leurs prix, etc. Et ensuite effectuer des commandes.



La création d’un user "joueur" avec **CREATE USER**, nommée player1

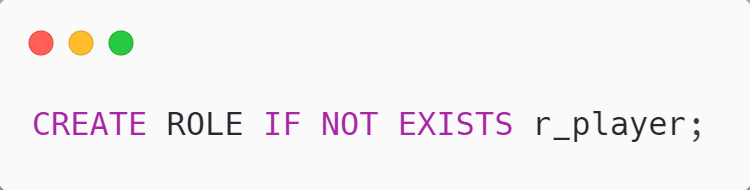


L’ajoute du user player1 au rôle **r\_player** avec la commande **GRANT ROLE**.

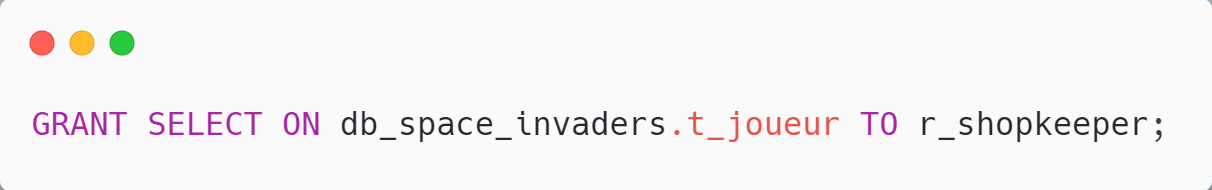


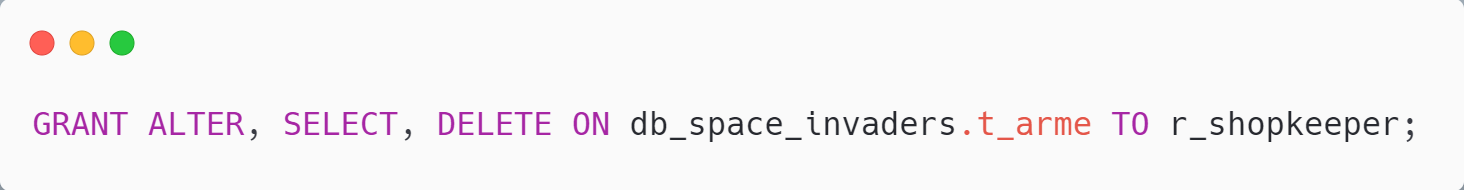
### **Gestionnaire de la boutique**

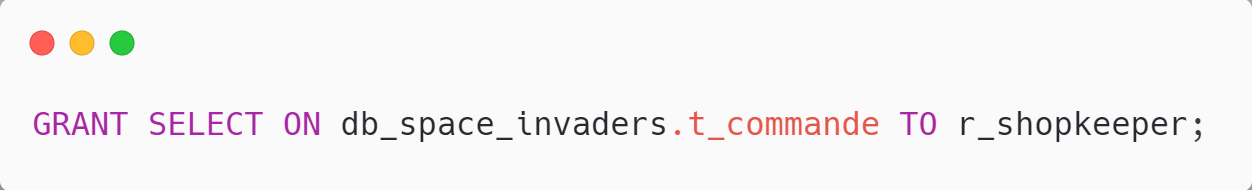
Création du rôle **r\_shopkeeper** avec le command **CREATE**.



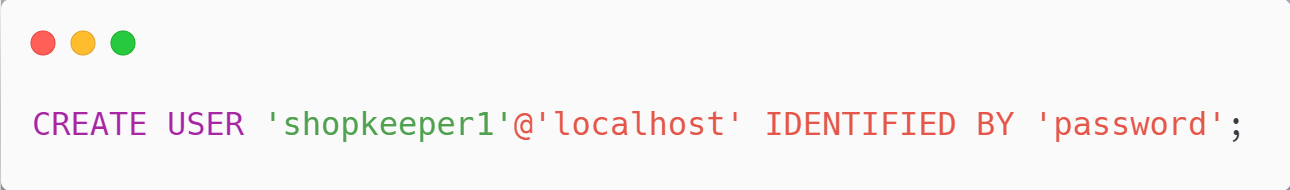
Donne les privilèges au rôle **r\_shopkeeper** avec command **GRANT** **SELECT ON db\_space\_invaders.t\_joueur** car le shopkeeper n’a que le droit de récupérer des données sur les joueurs. **GRANT** **ALTER, SELECT, DELETE ON db\_space\_invaders.t\_arme** pour pouvoir ajouter de nouvelles armes/modifier les prix. Et **GRANT SELECT ON db\_space\_invaders.t\_commade** pour voire toutes commandes faites.



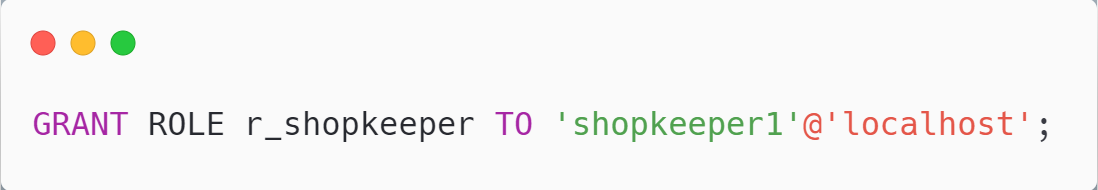




La création d’un user "shopkeeper" avec **CREATE USER**, nommée shopkeeper1



L’ajoute du user shopkeeper1 au rôle **r\_shopkeeper** avec la commande **GRANT ROLE**.



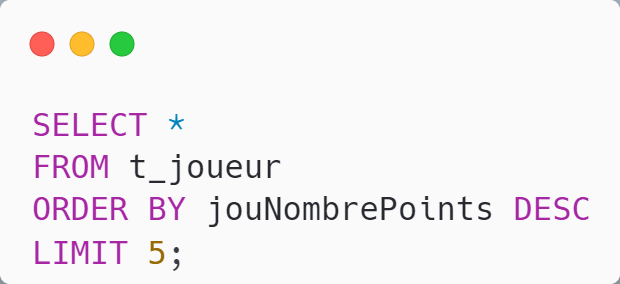
## **Requêtes de sélection**

Les requêtes de select **SQL** servent à récupérer des donnés depuis la base de données du jeu, pour ensuite afficher/ou en servir pour l’UX, ainsi que le fonctionnement du jeu.

### **Requête n°1**

Cette requête récupère toutes les colonnes de **t\_joueur**, les ordonner parce leur nombre de points en décroissant, et limitée à cinq.

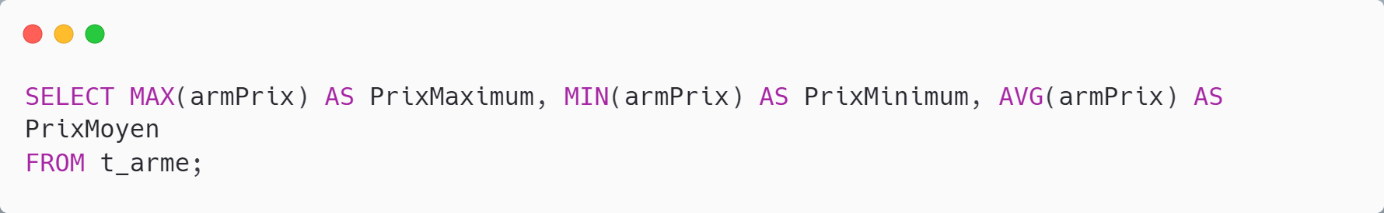
Donc elle montre les top cinq joueurs parce nombre de point décroissant va être afficher.



### **Requête n°2**

Cette requête récupère le **MAX**, **MIN**, et **AVG** du tableaux **t\_arme**.

Donc elle récupère l’arme qui coute le plus cher, le moins cher, et le prix moyenne d’une arme.



### **Requête n°3**

Cette requête récupère le **COUNT** de **idCommande** et le **fkJoueur** du tableaux **t\_idCommande**, groupé par le **fkJoueur**, et ordonner par le **COUNT** de **idCommande DESC.**

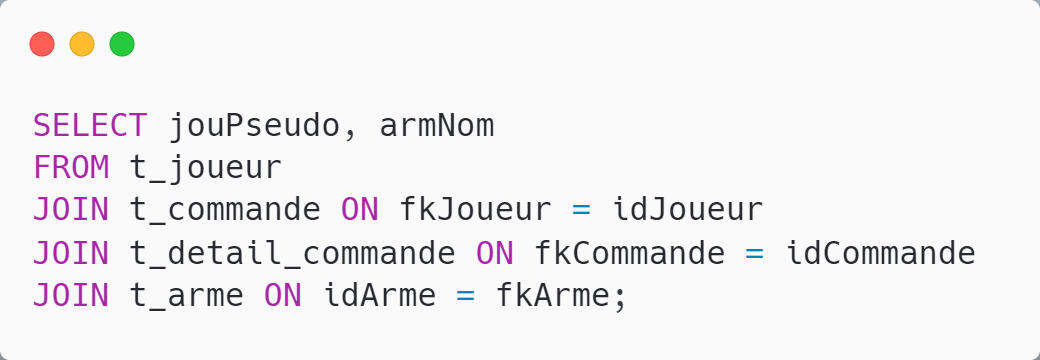
Donc elle récupère le id de chaque joueur et le nombre de commandes qu’ils ont effecteur, ordonner par le **COUNT DESC** pour les organiser par du joueur avec les plus au jouer avec le moins de commandes.



### **Requête n°4**

Cette requête récupère le **COUNT** **idCommande** et le **fkJoueur** du tableaux **t\_commande**, groupé par le **fkJoueur**, et groupée par le fkJoueur **HAVING** **COUNT** de **idCommande** plus que 2.

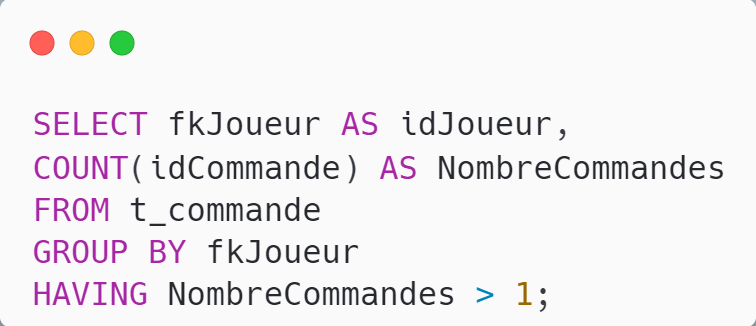
Seulement les joueurs qui ont passé plus que 2 commandes vont être afficher.



### **Requête n°5**

Cette requête récupère le **jouPseudo** de **t\_joueur**, le **armNom** de **t\_arme**, et comNumeroCommade de **t\_joueur**. Des **JOIN** sont utiliser pour faire les liaisons entre les différents tableux à travers de id et fk.

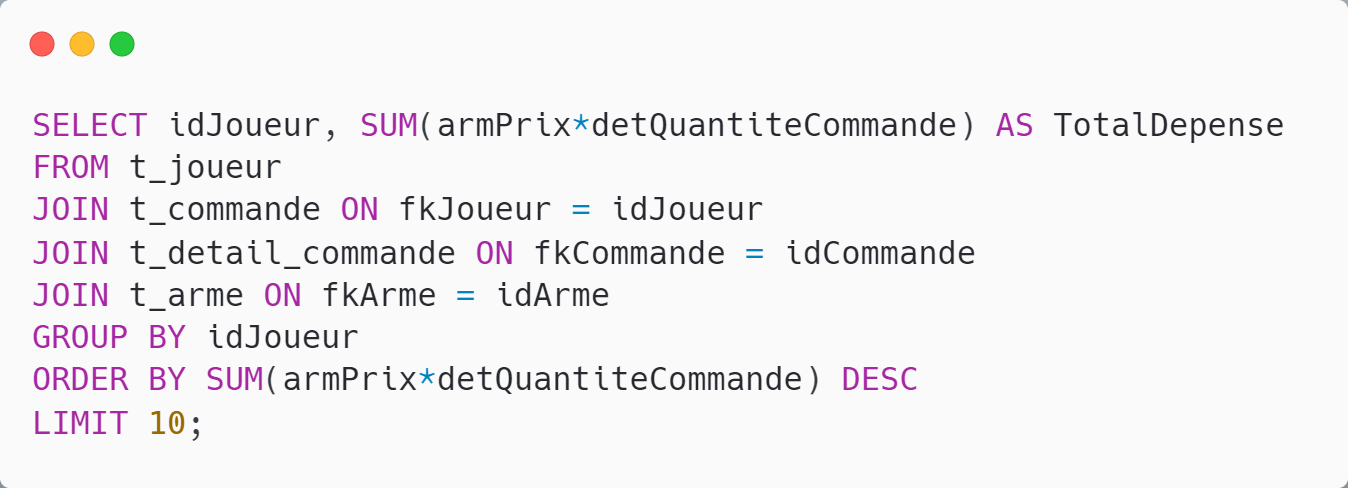
Cette requête affiche les pseudos et les armes qu’ils ont acheté dans chaque commande.



### **Requête n°6**

Cette requête récupère le **idJoueur** de **t\_joueur**, le **SUM armPrix \* detQuantiteCommande** de **t\_arme** et **t\_detail\_commande.** Des **JOIN** sont utiliser pour faire les liaisons entre les différents tableaux à travers de id et fk. C’est groupé par le **idJoueur**, ordonnée par **SUM armPrix \* detQuantiteCommande DESC**, et **LIMIT** 10.

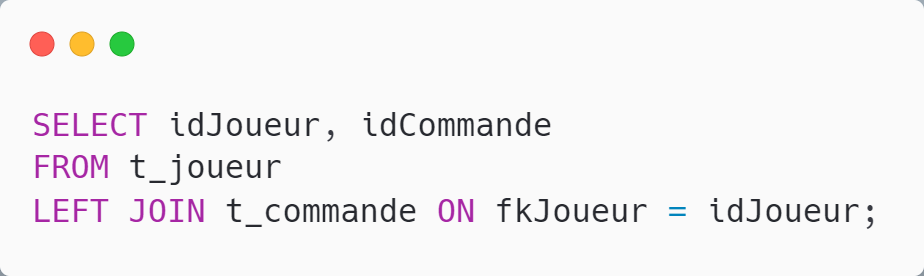
Cette requête affiche les identifient des joueurs et le total qu’ils ont dépensé, avec seulement le top 10 qui ont dépensé le plus en ordre décroissant.



### **Requête n°7**

Cette requête récupère **idJoueur** de **t\_joueur**, et le **idCommande** de **t\_commande**. **t\_commande** est relier à **t\_joueur** via un **LEFT JOIN**.

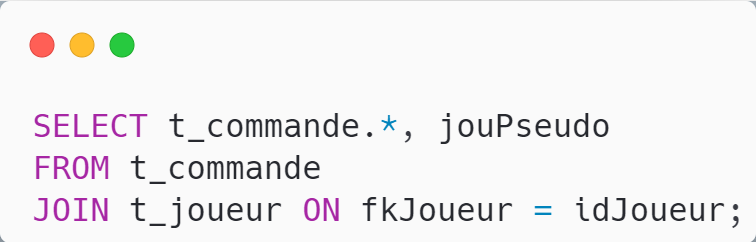
Cette requête affiche tous les joueurs, et leurs commandes, ainsi que ceux qui n’ont pas passé de commandes.



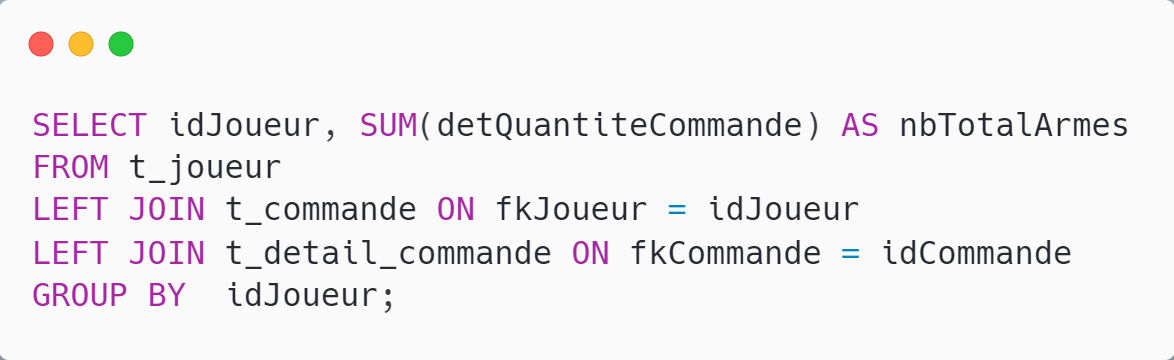
### **Requête n°8**

Cette requête récupère toutes les commandes par joueur,

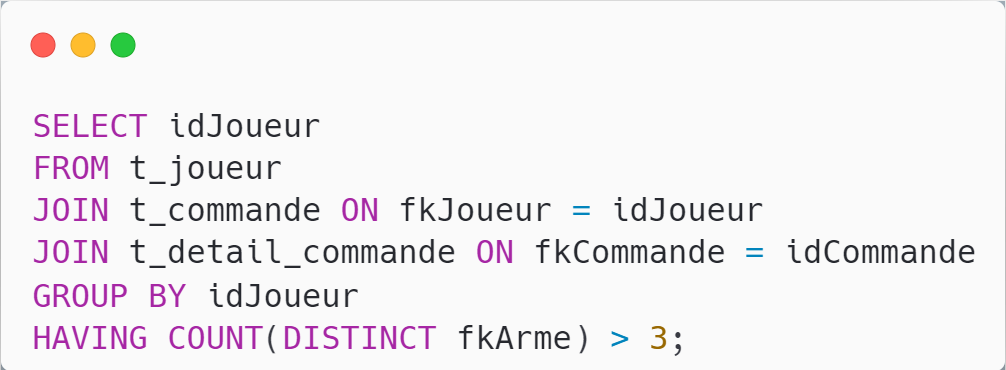
Cette requête affiche chaque colonne de **t\_commande** et la Pseudo de joueur qui là effectuer.



### **Requête n°9**



### **Requête n°10**



## **Création des index**

### **3.5.1.**

MySQL va automatiquement créer des indexes sur les colonnes uniques, les clés primaires, et les clés étrangères.

### **3.5.2.**

Un index va permettre d’optimiser le requête **SELECT** pour les effectuer plus, en utilisant un B+ tree structure

### **3.5.3.**

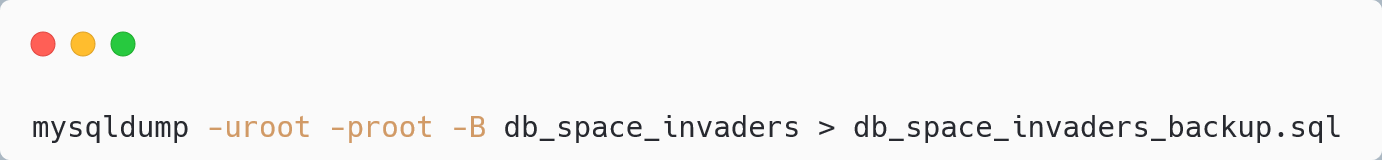
Dans le contexte de la structure de base de données, si le jamais avant un système de commandes et de boutique.

Ça aurait été pertinent de mettre un index sur le champ 'comDate' de la table 't\_commande' car lorsqu'une requête vise à recouper des données en fonction de ce champ (avant une certaine date, entre deux dates spécifiques), cela peut prendre du temps. Par exemple, lorsqu'on veut récupérer toutes les commandes effectuées avant le 31 décembre 2023, MySQL devra parcourir tous les jours de tous les mois avant d'atteindre la date du 31 décembre 2023. Ce n'est pas très efficace et cela prendrait plus de temps que nécessaire.

## **Backup/Restore**

### **Backup**

Cette commande **mysqldump**, qui va être utiliser dans une **CMD**, va servir à créer une sauvegarde de notre base de donnes, sa structure, et ses données. Le **-B** est utiliser pour que le fichier backup contient un **DROP nom\_du\_db** et **USE** **nom\_du\_db** avant et après le **CREATE** **nom\_du\_db**. Sinon l’exportation du dump, ne marcherais pas sans avoir déjà créer la **DB** avant.



### **Restore**

Cette commande serre à créer une nouvelle base de données avec structure et donnes a partir d’un fichier de backup.sql créer avec la commande **mysqldump.** La commande dois être exécuter depuis le répertoire, ou se trouve le fichier backup, sinon la lien du fichier dump dois être mis dans la commande.

