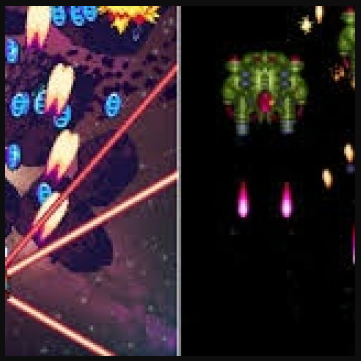
P\_DB106 Shoot me UP



Sena Cruz – CID2A

Vennes B13

24 périodes

Antoine Mveng

Table des matières

[1 Spécifications 3](#_Toc308526316)

[1.1 Titre 3](#_Toc308526317)

[1.2 Description 3](#_Toc308526318)

[1.3 Matériel et logiciels à disposition 3](#_Toc308526319)

[1.4 Prérequis 3](#_Toc308526320)

[1.5 Cahier des charges 3](#_Toc308526321)

[1.5.1 Objectifs et portée du projet (objectifs SMART) 3](#_Toc308526322)

[1.5.2 Caractéristiques des utilisateurs et impacts 3](#_Toc308526323)

[1.5.3 Fonctionnalités requises (du point de vue de l’utilisateur) 3](#_Toc308526324)

[1.5.4 Contraintes 3](#_Toc308526325)

[1.5.5 Travail à réaliser par l'apprenti 4](#_Toc308526326)

[1.5.6 Si le temps le permet … 4](#_Toc308526327)

[1.5.7 Méthodes de validation des solutions 4](#_Toc308526328)

[1.6 Les points suivants seront évalués 4](#_Toc308526329)

[1.7 Validation et conditions de réussite 4](#_Toc308526330)

[2 Planification Initiale 4](#_Toc308526331)

[3 Analyse 4](#_Toc308526332)

[3.1 Opportunités 4](#_Toc308526333)

[3.2 Document d’analyse et conception 4](#_Toc308526334)

[3.3 Conception des tests 5](#_Toc308526335)

[3.4 Planification détaillée 5](#_Toc308526336)

[4 Réalisation 5](#_Toc308526337)

[4.1 Dossier de Réalisation 5](#_Toc308526338)

[4.2 Modifications 5](#_Toc308526339)

[5 Tests 5](#_Toc308526340)

[5.1 Dossier des tests 5](#_Toc308526341)

[6 Conclusion 5](#_Toc308526342)

[6.1 Bilan des fonctionnalités demandées 5](#_Toc308526343)

[6.2 Bilan de la planification 5](#_Toc308526344)

[6.3 Bilan personnel 5](#_Toc308526345)

[7 Divers 6](#_Toc308526346)

[7.1 Journal de travail 6](#_Toc308526347)

[7.2 Bibliographie 6](#_Toc308526348)

[7.3 Webographie 6](#_Toc308526349)

[8 Annexes 6](#_Toc308526350)

# Spécifications

## Titre

P\_DB106 Space Invaders

## Description

Le but du projet P\_DB106 consiste à utiliser nos connaissances acquises au module 106 en requêtes SQL afin de former plusieurs requêtes permettant d’effectuer une série d’actions par rapport à la base de données « db\_space\_invaders.sql ». Les actions en question effectuent la création d’utilisateurs, la création et l’affectation de leur rôles, puis une série de sélections de données.

## Cahier des charges

Le cahier des charges se trouve sous le projet P\_DB106 sur marketplace.

# Planification Initiale

La planification du projet a été réalisée avec Icescrum.

# Réalisation

## Explication de chaque requête

### Requête n°1 :



Cette requête permet de sélectionner les cinq meilleurs joueurs de la base de données en termes de score. Pour cela on commence par sélectionner tous les éléments de la table ‘t\_joueur’ avec “SELECT \* FROM `t\_joueur`“, ensuite, il faut ordonner les éléments qui seront retournés par leur nombre de points, ce qui est fait en utilisant la commande “ORDER BY“. Il faut également préciser le sens de l’ordre, c’est pour cela qu’on inclut “DESC“, qui défini l’ordre en décroissant. Finalement, on ajoute “LIMIT 5“ afin de limiter le résultat uniquement aux 5 meilleurs joueurs.

### Requête n°2 :

A black screen with white text

Description automatically generated

Cette requête rend le prix maximum, moyen et minimum de chaque arme. On commence celle-ci en utilisant les fonctions “MAX“, “MIN“ et “AVG“ afin de sélectionner soit le prix maximum, minimum ou moyen de l’arme, en s’assurant d’utiliser “AS“ pour définir le nom de la valeur qui nous sera retournée.

### Requête n°3 :

A screen shot of a computer program

Description automatically generated

Cette requête permet de trouver le nombre total de commandes passées par chaque joueur. La clause DISTINCT garantit que l'on ne sélectionne qu'une seule instance de chaque joueur (fkJoueur), tandis que COUNT(idCommande) compte le nombre de commandes effectuées. La clause GROUP BY idJoueur regroupe les commandes par joueur, et l'ordre des résultats est décroissant (ORDER BY NombreCommandes DESC) pour afficher en premier les joueurs ayant le plus de commandes.

### Requête n°4 :

A black screen with white text

Description automatically generated

Cette requête est similaire à la précédente, mais elle inclut une clause HAVING pour filtrer les résultats et ne conserver que les joueurs ayant passé plus de 2 commandes. Cela permet de ne lister que les joueurs qui sont plus actifs, c'est-à-dire ceux ayant effectué plusieurs achats.

### Requête n°5 :

A computer screen shot of a computer code

Description automatically generated

Cette requête récupère le pseudo des joueurs (jouPseudo) et le nom des armes (armNom) associées aux commandes. Elle utilise des jointures entre t\_detail\_commande, t\_joueur et t\_arme. Cependant, il semble qu'il y ait une incohérence dans la clause JOIN, car t\_detail\_commande.fkCommande devrait correspondre à t\_commande.idCommande, et les armes devraient être jointes par fkArme. Cette requête risque de ne pas retourner le résultat attendu.

### Requête n°6 :

A computer screen shot of a code

Description automatically generated

Cette requête calcule le total dépensé par chaque joueur (SUM(armPrix)) en joignant les tables t\_detail\_commande, t\_joueur et t\_arme. Les résultats sont regroupés par idJoueur, triés par montant total dépensé (ORDER BY TotalDepense DESC) et limités aux 10 joueurs ayant dépensé le plus (LIMIT 10). Il semble que les conditions de jointure devraient être revues pour garantir la cohérence des relations.

### Requête n°7 :

A black background with white text

Description automatically generated

Cette requête effectue une jointure externe droite (RIGHT JOIN) pour récupérer tous les joueurs (t\_joueur) même ceux qui n'ont pas passé de commande (t\_commande). Elle retourne les pseudos des joueurs (jouPseudo) et l'identifiant des commandes (idCommande).

### Requête n°8 :

A computer screen shot of a computer screen

Description automatically generated

Inversement de la requête précédente, cette requête effectue une jointure externe droite pour afficher toutes les commandes (t\_commande) même si certains joueurs n'ont pas de pseudo associé (cas de commandes passées sans joueur associé ou anonymes).

### Requête n°9 :

A computer screen shot of a code

Description automatically generated

Cette requête détermine la quantité totale de produits commandés par chaque joueur (SUM(detQuantiteCommande)). Elle utilise une jointure externe gauche (LEFT JOIN) pour inclure tous les joueurs, même ceux qui n'ont pas passé de commande. Les résultats sont regroupés par jouPseudo et triés par la quantité totale commandée de manière décroissante.

### Requête n°10 :

A computer screen shot of white text

Description automatically generated

Cette requête compte le nombre de commandes par joueur (COUNT(idCommande)) et utilise la clause HAVING pour ne retenir que les joueurs ayant commandé plus de 3 types d'armes distinctes (COUNT(DISTINCT fkArme) > 3). Cela met en avant les joueurs ayant une diversité dans leurs achats d'armes.

## Explication de chaque utilisateur

### Administrateur

A black background with white text

Description automatically generated

Le rôle Admin a tous les privilèges sur toutes les tables et bases de données (ON \*.\*), y compris le droit de déléguer des privilèges grâce à WITH GRANT OPTION. L'utilisateur admin est créé et se voit attribuer ce rôle par GRANT 'Admin' TO 'admin'@'%';.

### Joueur

A black background with white text

Description automatically generated

Le rôle joueur a le droid d’utiliser SELECT sur la table t\_arme pour pouvoir consulter les informations sur les armes. Il peut également INSERT et SELECT sur la table t\_commande pour pouvoir passer des commandes et les consulter.

### Gestionnaire de la boutique

A screen shot of a computer

Description automatically generated

Le gestionnaire peut SELECT sur la table t\_joueur pour voir les informations sur les joueurs. Il a tous les privilèges sur la table t\_arme pour gérer les armes et SELECT sur la table t\_commande.

## Création des index

### En étudiant le dump MySQL db\_space\_invaders.sql vous constaterez que vous ne trouvez pas le mot clé INDEX.

1. Pourtant certains index existent déjà. Pourquoi ?

Certains index existent déjà en raison des clés primaires et étrangères créées automatiquement par MySQL pour optimiser la structure.

1. Quels sont les avantages et les inconvénients des index ?

Les index accélèrent les recherches mais consomment de l'espace et ralentissent les opérations d'écriture.

1. Sur quel champ (de quelle table), cela pourrait être pertinent d’ajouter un index ? Justifier votre réponse.

Un index sur comNumeroCommande dans la table t\_commande serait pertinent car ce champ est souvent utilisé pour rechercher des commandes spécifiques. En ajoutant un index, les requêtes qui filtrent ou recherchent par ce numéro seraient considérablement accélérées, réduisant le temps nécessaire pour parcourir l'ensemble de la table. De plus, cela améliorerait l'efficacité des opérations de reporting et de suivi des commandes, contribuant à une meilleure performance globale de l'application.