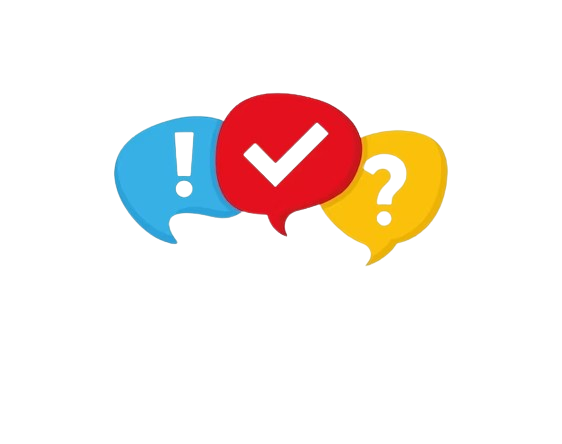
QuizMaster

Projet TPI du 02.05.2024 au 03.06.2024



Chef de projet : Helder Costa Lopes

Experts : Benjamin Wolf

Roger Malherbe

ETML – Alexandre Fernandes – CIN4B

Table des matières

[1 Glossaire 7](#_Toc168296600)

[2 Analyse préliminaire 10](#_Toc168296601)

[*2.1* Introduction 10](#_Toc168296602)

[2.1.1 Titre 10](#_Toc168296603)

[2.1.2 Description 10](#_Toc168296604)

[2.1.3 Contexte 10](#_Toc168296605)

[*2.2* Objectifs 10](#_Toc168296606)

[2.3 Planification initiale 11](#_Toc168296607)

[3 Analyse / Conception 12](#_Toc168296608)

[3.1 Concept 12](#_Toc168296609)

[3.1.1 Méthodologie de gestion de projet : en cascade 12](#_Toc168296610)

[*3.1.2* Maquettes graphiques 13](#_Toc168296611)

[3.1.3 Modèle Conceptuel des Données : 17](#_Toc168296612)

[3.1.4 Modèle Logique des Données 18](#_Toc168296613)

[*3.2* Stratégie de test 19](#_Toc168296614)

[*3.3* Risques techniques 19](#_Toc168296615)

[*3.4* Planification 20](#_Toc168296616)

[3.5 Dossier de conception 21](#_Toc168296617)

[3.5.1 Matériel Hardware 21](#_Toc168296618)

[3.5.2 Application / Logiciels / Outils 21](#_Toc168296619)

[3.5.3 Site web 22](#_Toc168296620)

[3.5.4 Suivi de projet 23](#_Toc168296621)

[3.5.5 Base de données : MariaDB 23](#_Toc168296622)

[3.5.6 Architecture du programme 26](#_Toc168296623)

[3.5.7 Dépendance externe 27](#_Toc168296624)

[3.5.8 Utilisation schématisée 27](#_Toc168296625)

[4 Réalisation 28](#_Toc168296626)

[4.1 Liste des fichiers : Répertoire 28](#_Toc168296627)

[4.2 Liste des fichiers : Code source 29](#_Toc168296628)

[4.3 Base de données : Comment fonctionnent les requêtes 30](#_Toc168296629)

[4.4 Base de données : Authentification 31](#_Toc168296630)

[4.4.1 Requêtes préparées utilisées 31](#_Toc168296631)

[4.4.2 Connexion 31](#_Toc168296632)

[4.4.3 Inscription 33](#_Toc168296633)

[4.4.4 Déconnexion 35](#_Toc168296634)

[4.5 Base de données : Affichage tous les quiz 36](#_Toc168296635)

[4.5.1 Requêtes préparées utilisées 36](#_Toc168296636)

[4.5.2 Affichage des quiz 36](#_Toc168296637)

[4.6 Base de données : Affichage quiz spécifique 37](#_Toc168296638)

[4.6.1 Requêtes préparées utilisées 37](#_Toc168296639)

[4.6.2 Affichage du quiz 37](#_Toc168296640)

[4.7 Base de données : Traitement du quiz 38](#_Toc168296641)

[4.7.1 Requêtes préparées utilisées 38](#_Toc168296642)

[4.7.2 Traitement 38](#_Toc168296643)

[4.7.3 Affichage des réponses 39](#_Toc168296644)

[4.8 Base de données : Création quiz 40](#_Toc168296645)

[4.8.1 Requêtes préparées utilisées 40](#_Toc168296646)

[4.8.2 Traitement et création du quiz 41](#_Toc168296647)

[4.9 Base de données : Suppression quiz 43](#_Toc168296648)

[4.9.1 Requêtes préparées utilisées 43](#_Toc168296649)

[4.10 Base de données : Affichage classement 45](#_Toc168296650)

[4.10.1 Requêtes préparées utilisées 45](#_Toc168296651)

[4.10.2 Affichage du classement 45](#_Toc168296652)

[4.11 Base de données : Gestion des utilisateurs 47](#_Toc168296653)

[4.11.1 Requêtes préparées utilisées 47](#_Toc168296654)

[4.11.2 Affichages de tous les utilisateurs 48](#_Toc168296655)

[4.11.3 Modification 48](#_Toc168296656)

[4.11.4 Suppression 49](#_Toc168296657)

[4.12 Cookies 51](#_Toc168296658)

[4.13 Base de données : résultat visuel (PHPMyAdmin) 51](#_Toc168296659)

[4.14 Versions du projet 53](#_Toc168296660)

[4.15 Description des tests effectués 54](#_Toc168296661)

[4.16 Test unitaire 58](#_Toc168296662)

[4.17 Erreurs restantes 59](#_Toc168296663)

[*4.18* Liste des documents fournis 59](#_Toc168296664)

[5 Conclusions 60](#_Toc168296665)

[5.1 Bilan des fonctionnalités demandées 60](#_Toc168296666)

[5.2 Bilan personnel 61](#_Toc168296667)

[5.3 Bilan de la planification 61](#_Toc168296668)

[6 Annexes 62](#_Toc168296669)

[6.1 Résumé du projet QuizMaster 62](#_Toc168296670)

[6.2 Webographie 62](#_Toc168296671)

[*6.3* Table des illustrations 64](#_Toc168296672)

[*6.4* Dossier des tests 65](#_Toc168296673)

[*6.5* Archives du projet 65](#_Toc168296674)

[6.6 Planification – Journal de travail 66](#_Toc168296675)

[68](#_Toc168296676)

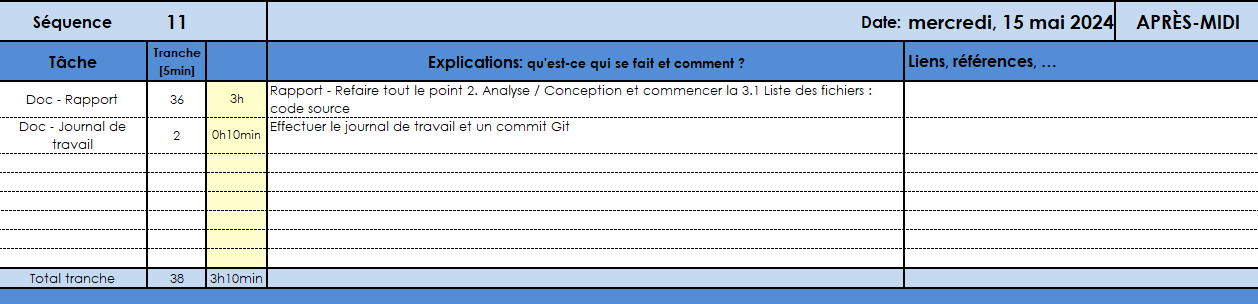
[69](#_Toc168296677)

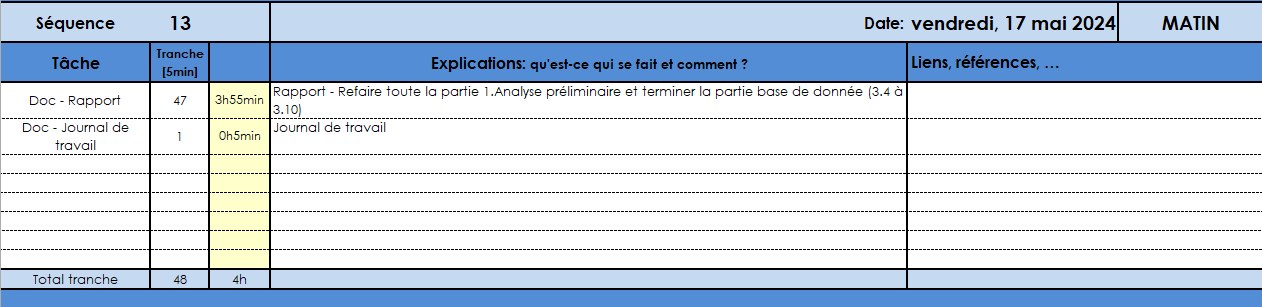
[70](#_Toc168296678)

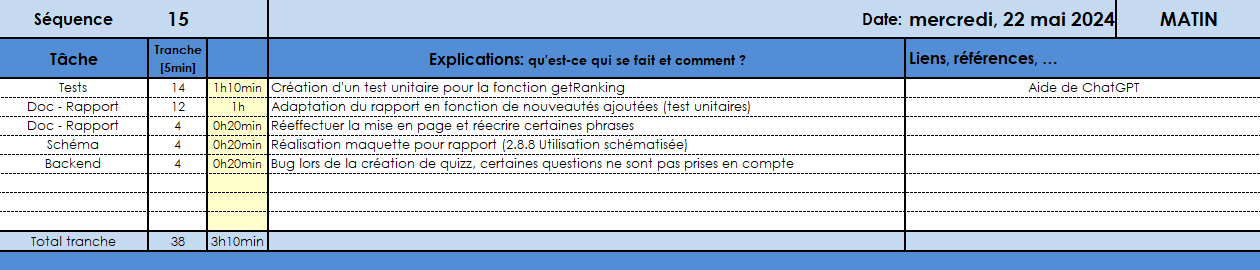
[71](#_Toc168296679)

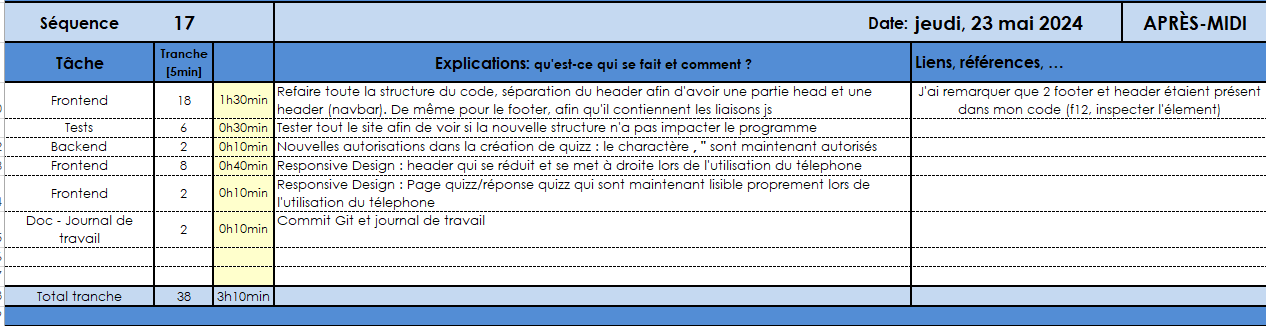
[72](#_Toc168296680)

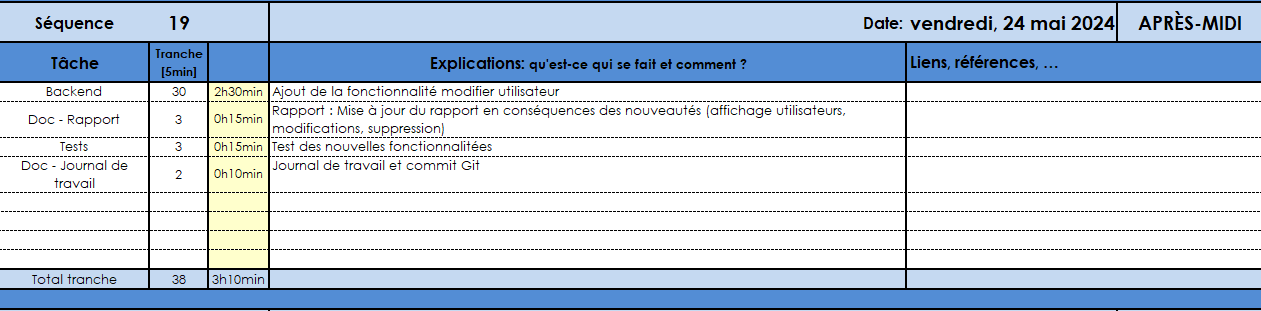
[6.7 Journal de travail 73](#_Toc168296681)

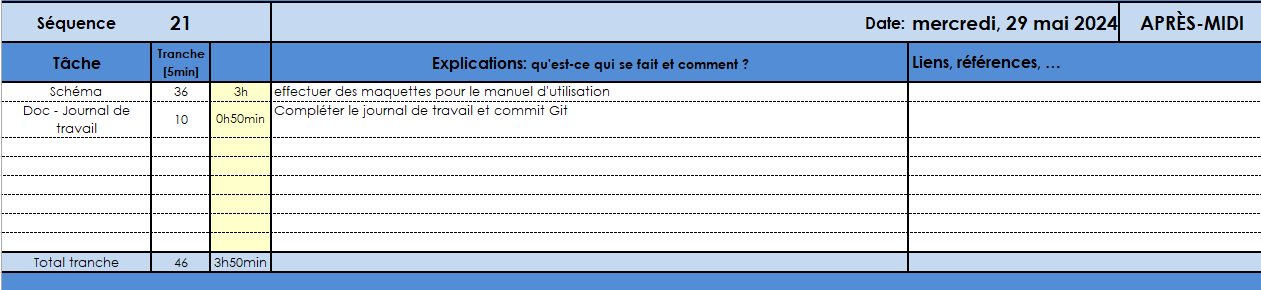
[ 78](#_Toc168296682)

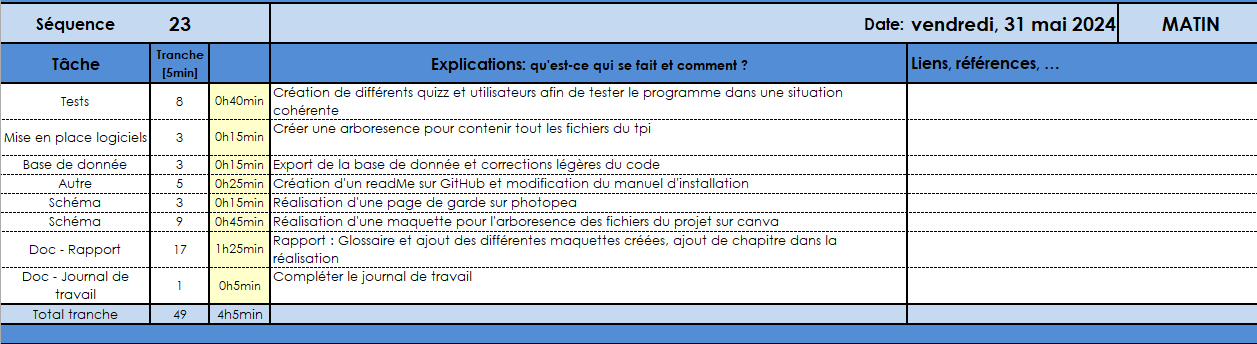
[ 79](#_Toc168296683)

[ 80](#_Toc168296684)

[ 81](#_Toc168296685)

[ 82](#_Toc168296686)

[ 83](#_Toc168296687)

[ 84](#_Toc168296688)

[*6.8* Manuel d'Installation 86](#_Toc168296689)

[6.9 Manuel d'Utilisation 87](#_Toc168296690)

[6.9.1 S’inscrire 87](#_Toc168296691)

[6.9.2 Se connecter 87](#_Toc168296692)

[6.9.3 Naviguer entre les différentes pages 88](#_Toc168296693)

[6.9.4 Utilisation de quiz 88](#_Toc168296694)

[6.9.5 Création d’un quizz (administrateur) 90](#_Toc168296695)

[6.9.6 Gestion des utilisateurs (administrateur) 90](#_Toc168296696)

[6.9.7 Consulter le classement général et son score personnel 91](#_Toc168296697)

# Glossaire

A

**Authentification**

Processus de vérification de l'identité d'un utilisateur, souvent en comparant un mot de passe saisi avec un mot de passe stocké dans une base de données.

B

**Bootstrap**

Collection d'outils pour créer des sites et applications web, comprenant des codes HTML, CSS et des extensions JavaScript.

C

**Connexion PDO**

Utilisation de PHP Data Objects pour se connecter à une base de données, permettant une interface uniforme pour accéder à plusieurs types de bases de données.

**CSS (Cascading Style Sheets)**

Langage utilisé pour décrire la présentation des documents HTML ou XML.

**Commit**

Enregistrement d'un ensemble de modifications dans un système de contrôle de version comme Git, permettant de suivre et de gérer l'évolution du code source.

F

**Flexbox**

Module de mise en page CSS permettant un agencement plus flexible des éléments sur une page web.

G

**GitHub**

Plateforme d'hébergement de code source utilisant le système de contrôle de version Git.

J

**jQuery**

Bibliothèque JavaScript qui simplifie la manipulation des documents HTML, la gestion des événements, l'animation et les interactions Ajax.

M

**MCD (Modèle Conceptuel des Données)**

Représentation graphique de la structure d'une base de données, décrivant les entités et les relations entre elles.

**MLD (Modèle Logique des Données)**

Transformation du MCD en une structure logique adaptée à la mise en œuvre dans un SGBD (Système de Gestion de Base de Données).

**MySQL**

Système de gestion de base de données relationnelle open source.

P

**PHP (Hypertext Preprocessor)**

Langage de script utilisé principalement pour le développement web et la création de pages web dynamiques.

**PHPMyAdmin**

Application web pour gérer les bases de données MySQL et MariaDB.

R

**Requêtes préparées**

Requêtes SQL où les paramètres sont fournis séparément de la commande SQL, augmentant ainsi la sécurité en protégeant contre les injections SQL.

**Responsive Design**

Approche de la conception web qui permet à un site de s'adapter à différentes tailles d'écran et appareils.

S

**SHA256**

Algorithme de hachage cryptographique produisant un condensé de 256 bits, souvent utilisé pour sécuriser les mots de passe.

**Suite Microsoft 365**

Ensemble d'applications et de services de productivité basés sur le cloud, incluant Word, Excel, PowerPoint et d'autres outils.

T

**TPI (Travail Pratique Individuel)**

Projet individuel réalisé par un étudiant pour valider ses compétences et obtenir son diplôme.

V

**Visual Studio Code**

Editeur de code source développé par Microsoft, offrant des fonctionnalités comme le débogage, la complétion de code et le contrôle de version intégré.

# Analyse préliminaire

## Introduction

### Titre

QuizMaster

### Description

Ce projet a pour but de créer un système de quiz interactif en PHP. Les utilisateurs pourront faire différents quiz, répondre aux questions et voir leurs résultats tout de suite. L’application aura une interface utilisateur simple, une base de données MYSQL pour garder les données des quiz et une authentification des utilisateurs pour distinguer les administrateurs des utilisateurs ordinaires.

Les fonctionnalités principales seront la création de quiz, le défilement des questions, le calcul du score et l’affichage du score.

On pourrait comparer cela à un « [Kahoot[[1]](#footnote-1)](https://kahoot.com/)» pour l’ETML.

### Contexte

Ce projet a été réalisé durant mon TPI. Celui-ci permettra de valider mon CFC. Le thème développement web et le chef de projet ont été choisi par moi-même puis le choix fut confirmer par le chef de projet Helder Manuel Costa Lopes.

## Objectifs

Ce projet a pour objectif de :

* La présence des différentes fonctionnalités tel que :
  + Gestion de la base de données MySQL
    - Création base de données
    - Création tables utilisateurs, questions, réponses, quiz
    - Insertion, récupération et mise à jour des données
  + Création de toutes les interfaces utilisateurs nécessaires
  + Intégration PHP-MYSQL
    - Connexion base de données
    - Exécution requêtes
    - Récupération et affichage des données
  + Authentification des utilisateurs (connexion, inscription, déconnexion)
  + Calcul du score des quizz et mise à jour du score
  + Affichage des scores à la fin du quiz avec les réponses correctes et fausses
    - Implémenter un classement des utilisateurs
  + Sécurité des données (hachage pour données sensibles)
  + Procédure de test
  + Design responsive
  + Gestion des erreurs
* Un schéma MCD/MLD et une sauvegarde régulière

## Planification initiale

Cette planification initiale permet de visionner de manière très large comment je pense devoir répartir mon temps pour la réalisation de ce projet.

Documentation :

* Ecriture du rapport et journal de travail

Analyse/Conception :

* Ecriture de la planification, réalisation de schéma

Réalisation projet :

* Mise en place des logiciels, réalisation du programme (base de données, backend, frontend)

Test :

* Réalisation de test du code

**Annexe :** [Planification](#_Planification_–_Journal)

# Analyse / Conception

## Concept

### Méthodologie de gestion de projet : en cascade

Pour la réalisation de ce projet, la méthode de travail est la suivante : en cascade.

****

Figure 1 Modèle : en cascade

**Exigences :** Les exigences correspondent à la rencontre entre l’équipe de projet et le client pour comprendre les besoins et attentes concernant le produit final. Les informations permettent de définir le périmètre du projet et les bases du développement.  
Dans notre cas, une rencontre avec le chef de projet et le premier expert a été effectuée.

**Analyse :** L’analyse permet de reprendre les informations obtenues lors de la phase d’exigences afin de les analyser et être sûr qu’elles soient claires, complètes et cohérentes. Des décisions peuvent déjà être prise sur cette étape concernant par exemple une planification initiale ou le choix des ressources nécessaire ou même des échéances concernant le projet.  
Dans notre cas, cela a été effectué juste avant la signature du cahier des charges durant l’entrevue avec le premier expert.

**Conception :** La conception permet de réaliser une conception détaillée du projet, comme la conception d’une interface utilisateur, spécification des fonctionnalités, une planification détaillée.

Dans notre cas, cela a été effectué lors de la semaine suivant le début du projet, lors des deux premier jours une planification détaillée, suivant ces jours, des maquettes web et de base de données ont été effectuées.

**Mise en œuvre :** Lors de la mise en œuvre, l’équipe commence le code du projet en se basant sur les conceptions précédentes. Les fonctionnalités sont développées en utilisant dans la majorité des cas, des conventions.  
Dans notre cas cela est effectué en utilisant les conventions ETML tout au fil du temps impartit pour le TPI.

**Validation :** La validation permet, une fois le développement terminé, de vérifier que le logiciel corresponde aux exigences du client. Cela est effectué à l’aide de test unitaires par exemple.  
Dans notre cas, des tests sont effectués selon la stratégie de test cité plus bas, et la présentation de l’avancé au chef de projet

**Mise en service :** La mise en service correspond au déploiement dans son environnement. Une fois le développement aboutit, les tests validés.

Dans notre cas, cela correspond à la livraison du projet à la fin du TPI

### Maquettes graphiques

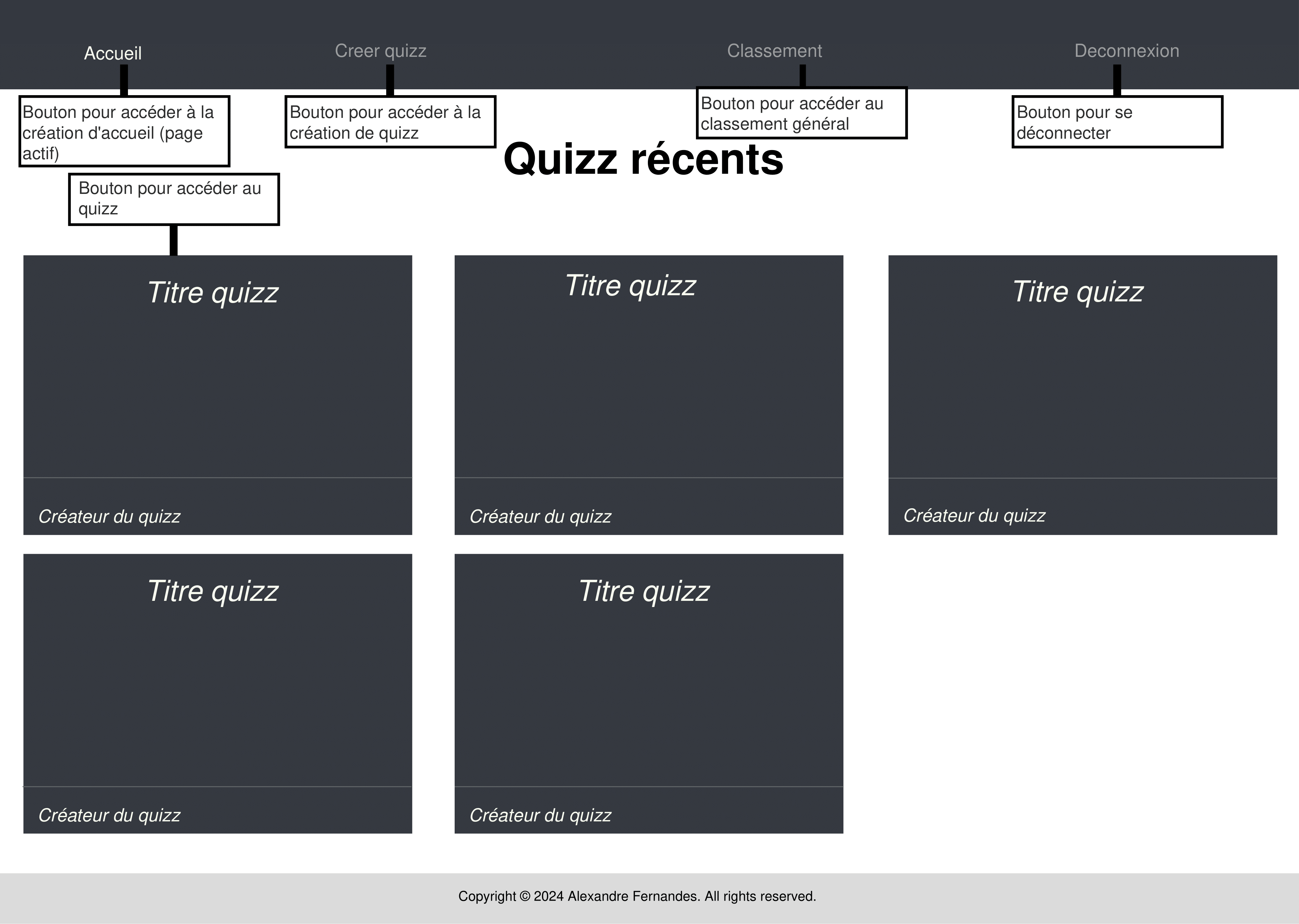


Figure 2 Maquette : Page d'accueil

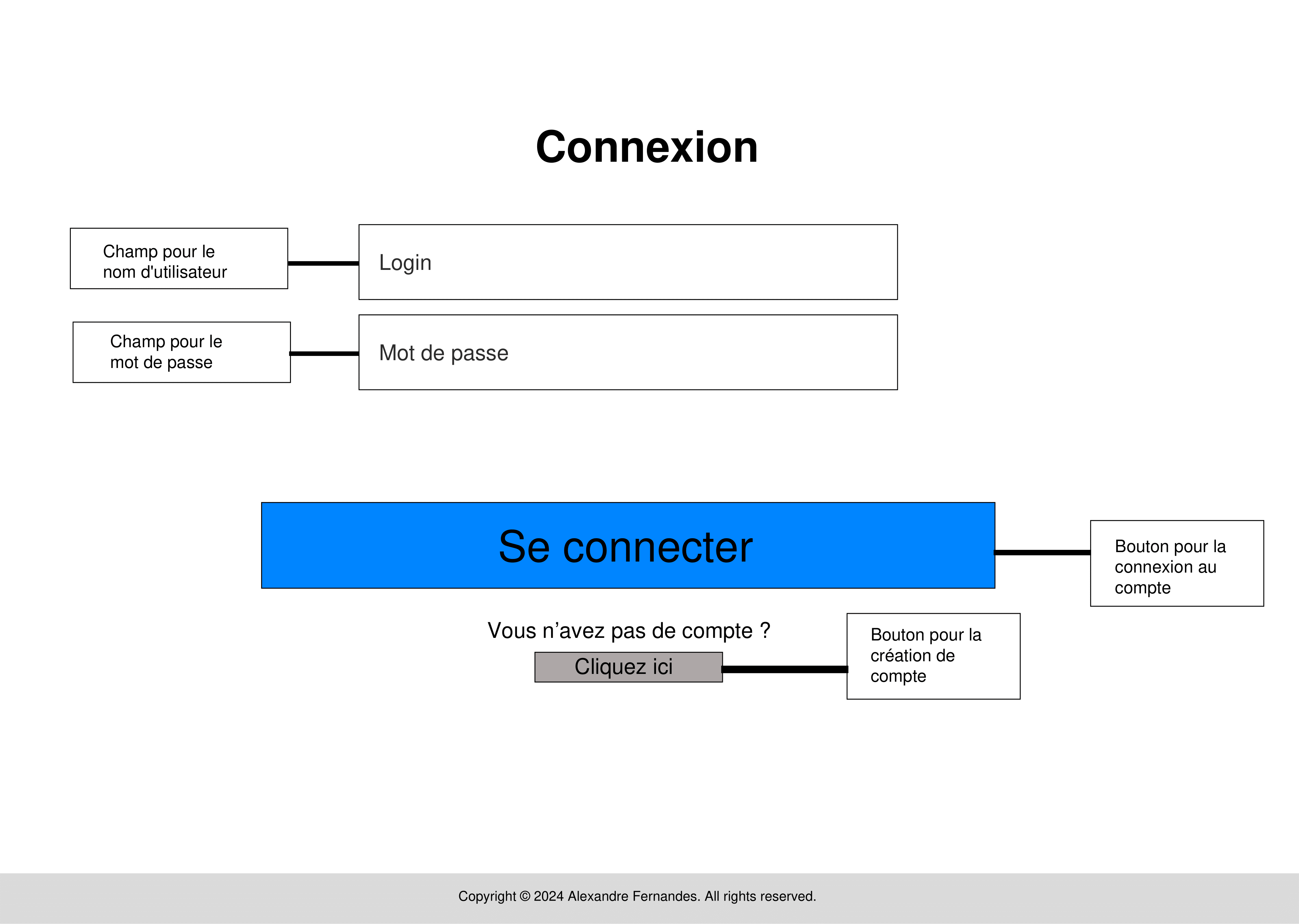


Figure 3 Maquette : Page de connexion

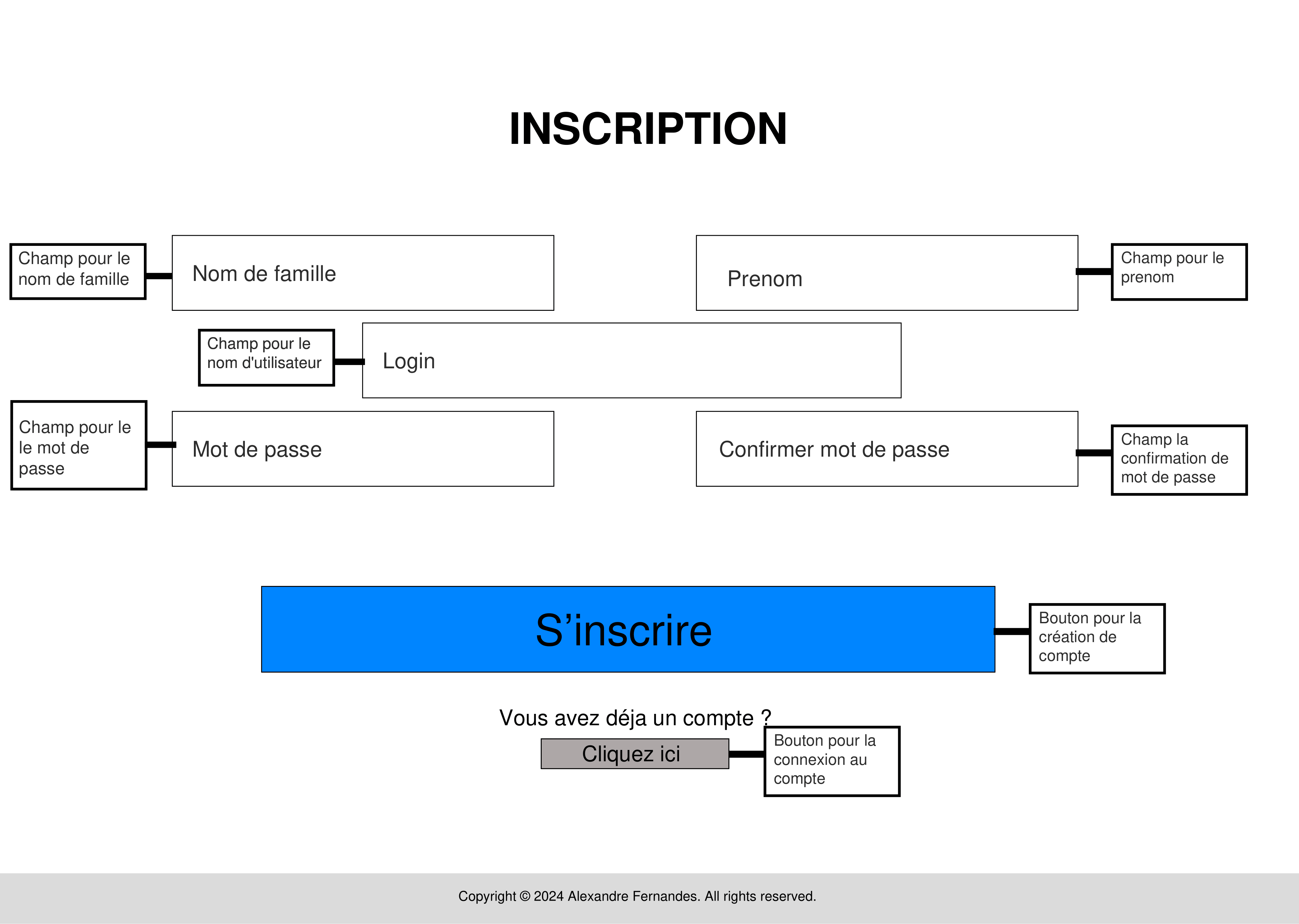


Figure 4 Maquette : Page d'inscription

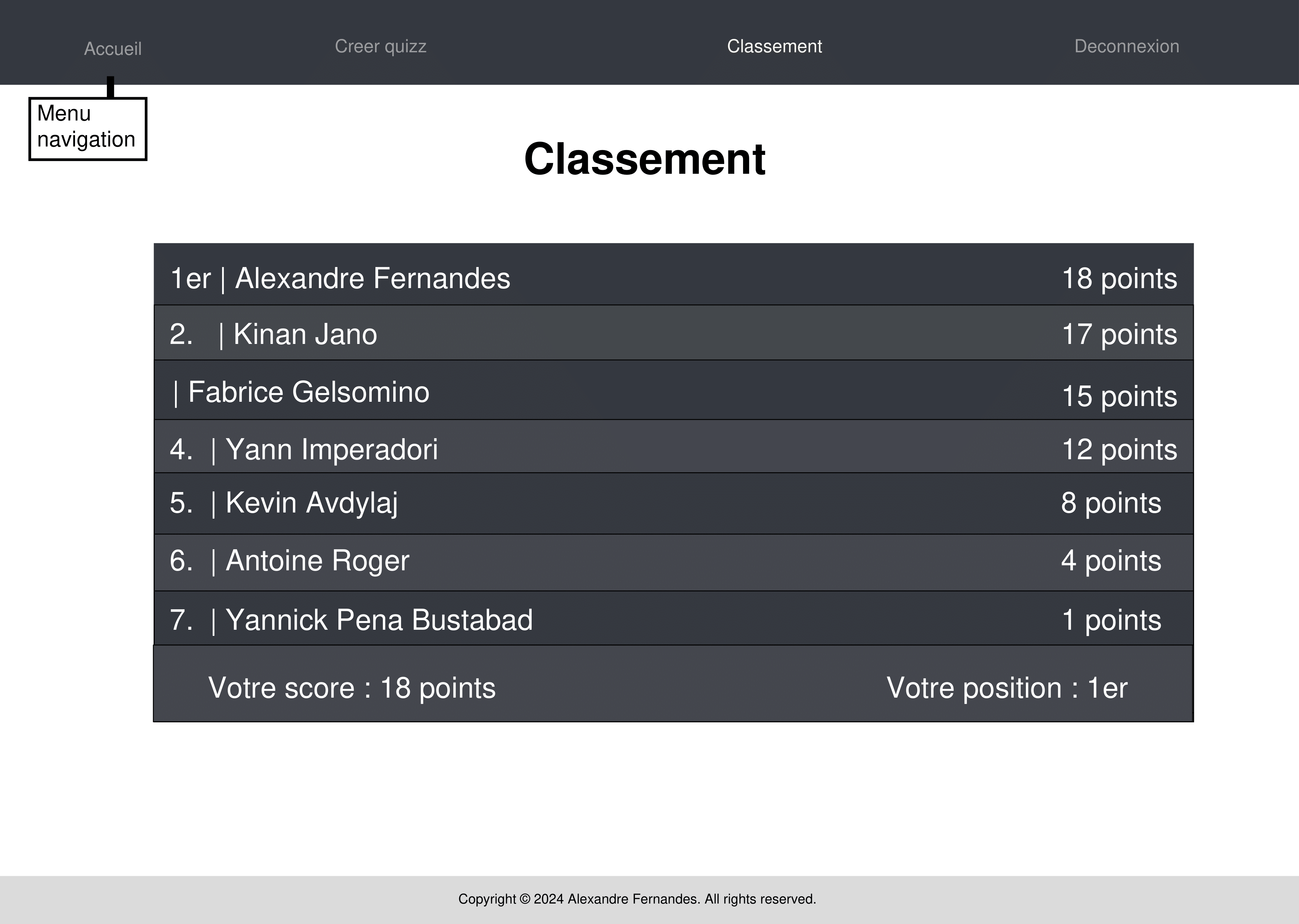


Figure 5 Maquette : Page de Classement

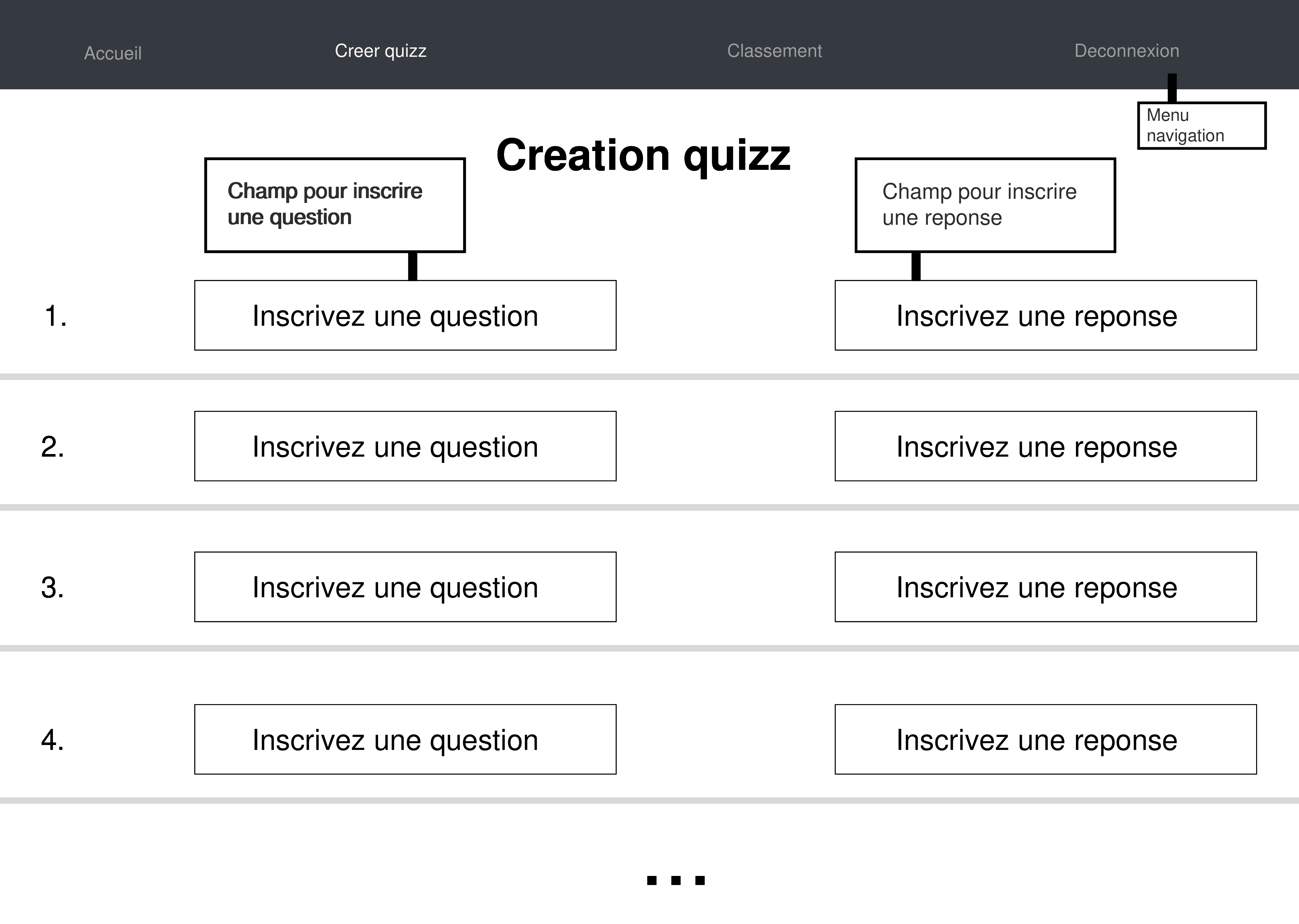


Figure 6 Maquette : Page de Création Quizz

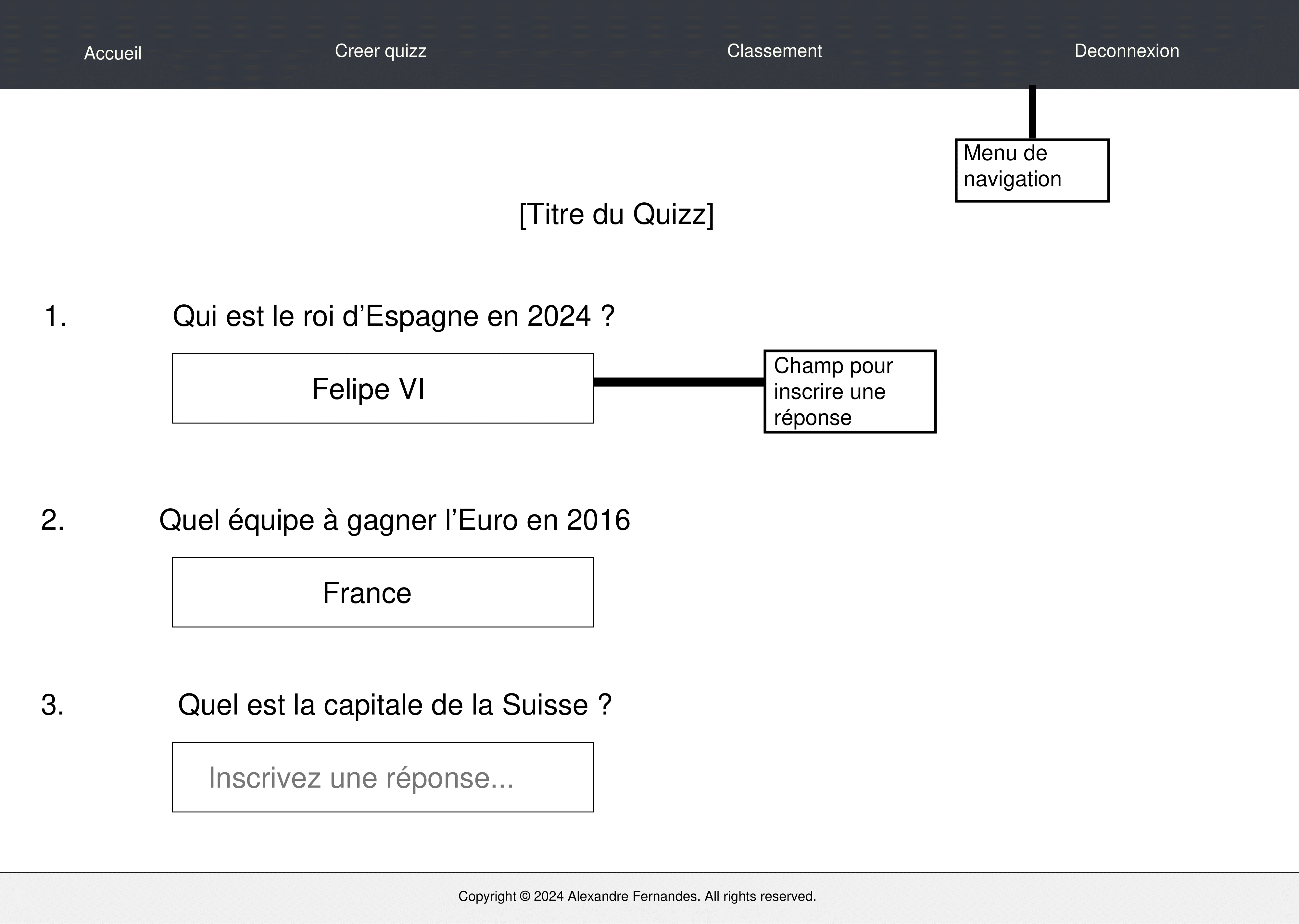


Figure 7 Maquette : Page de Quizz



Figure 8 Maquette : Page Réponse de Quizz

### Modèle Conceptuel des Données :

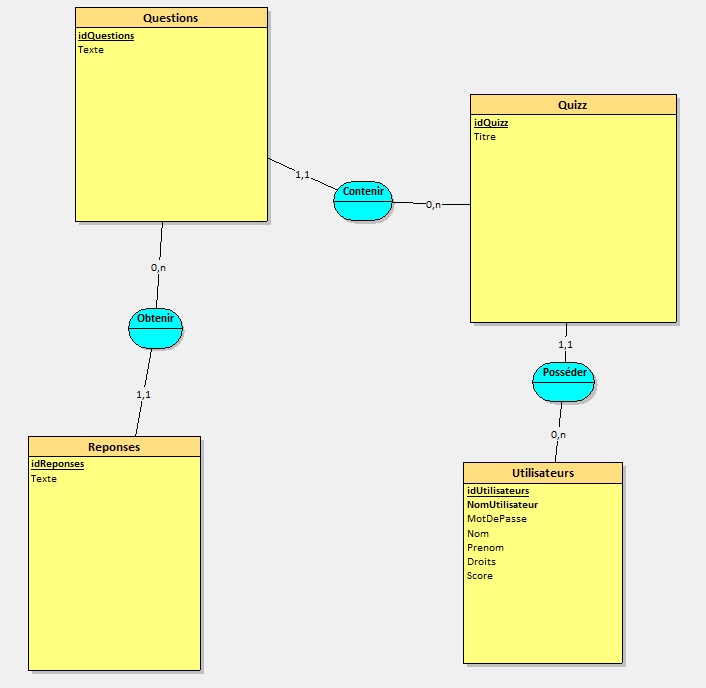


Figure 9 Modèle conceptuel des données

### Modèle Logique des Données

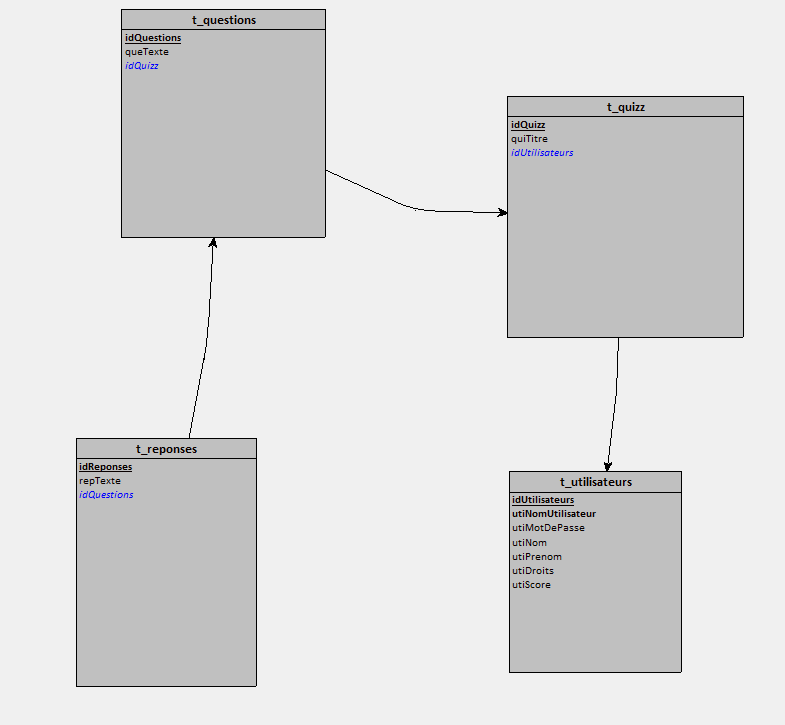


Figure 10 Modèle logique des données

## Stratégie de test

* Test unitaires à l’aide de fonctions PHP et requêtes SQL
* Test d’une fonction sous diverses conditions, par exemple exécuter une requête avec différents paramètres possibles (numéro, lettre, etc…)
* Test « manuel » en utilisant l’application moi-même ou par une personne externe, en cherchant les potentiels situations sensibles (changer code html, rafraichir à un moment pas prévu, insérer des données pas attendues afin de voir si l’application tient la route)

## Risques techniques

Durant ce projet je ne vois pas de potentiel problèmes que je risque de rencontrer au niveau du développement sur site.

Cependant ceci est le premier projet qui sera conçu pour être utilisé par un nombre conséquent d’utilisateurs simultanément, il est possible que certains problèmes inattendus surviennent lors des tests avec plusieurs utilisateurs.

## Planification

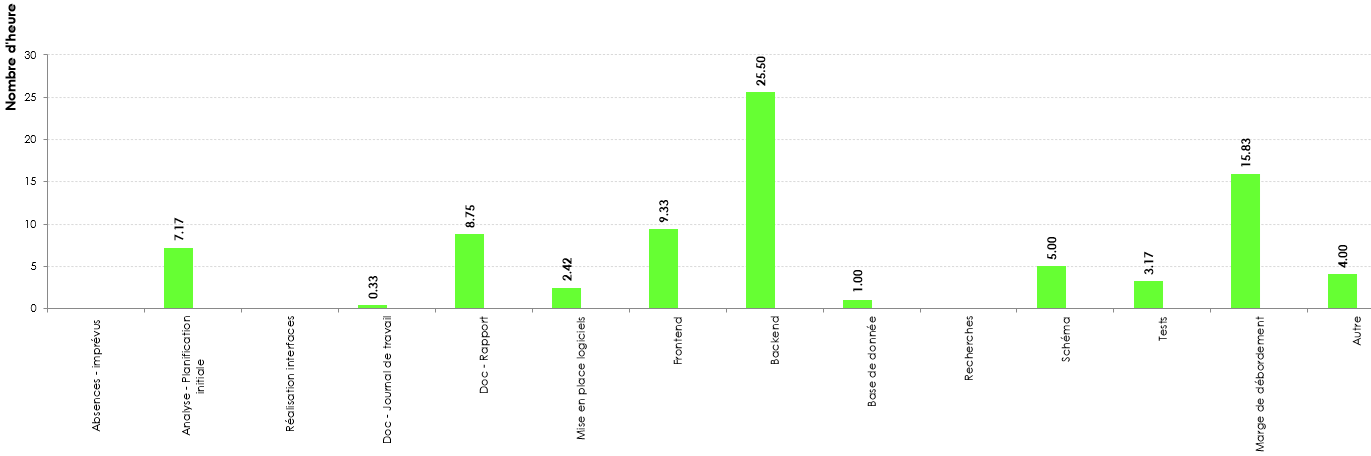
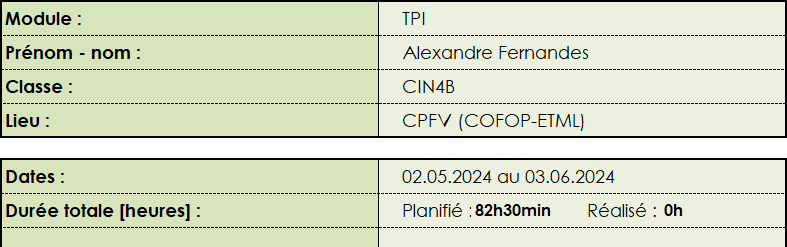


Figure 11 Graphique : Bilan de la planification initiale

## Dossier de conception

### Matériel Hardware

* Ordinateur ETML
  + Windows 10 Éducation 22H2 x64
  + 2.50 GHz Intel Core i7-11700 CPU
  + 32 GO RAM

### Application / Logiciels / Outils

* PHPMyAdmin 6.0
  + PhpMyAdmin est une application Web de gestion pour les systèmes de gestion de base de données MySQL et MariaDB
  + Durant ce projet il a été utile pour simplifier la gestion de la base de données MariaDB
* PHP 8.3.4
  + PHP : Hypertext Preprocessor, est un langage de programmation libre, principalement utilisé pour produire des pages Web dynamiques via un serveur web, mais pouvant également fonctionner comme n'importe quel langage interprété de façon locale.
  + Un des codes utiliser durant mon projet, il a par exemple été utiliser pour la communication entre la page web et la base de données
* MariaDB server 11.3.2
  + MariaDB est un système de gestion de base de données.
  + Durant le projet il m’a permis de gérer ma base de données
* Visual Studio Code
  + Visual Studio Code est un éditeur de code extensible développé par Microsoft pour Windows, Linux et macOS. Les fonctionnalités incluent la prise en charge du débogage, la mise en évidence de la syntaxe, la complétion intelligente du code, les snippets, la refactorisation du code et Git intégrer.
  + Il m’a été utile afin de programmer mon code
* jQuery 7.3.1
  + jQuery est une bibliothèque JavaScript libre et multiplateforme créée pour faciliter l'écriture de scripts côté client dans le code HTML des pages web
  + Un des langages de programmation utilisé dans mon code, il m’est utile afin de traiter des données et faire des retours d’erreurs
* GitHub Desktop 3.3.17
  + Société d’hébergement de développement
  + Me permet d’héberger mon projet et avoir un système de version
  + <https://github.com/Medellin05001/TPI-QuizMaster> <== Mon dépôt
* Bootstrap 4.5.2
  + Bootstrap est une collection d'outils utiles à la création du design de sites et d'applications web. C'est un ensemble qui contient des codes HTML et CSS, des formulaires, boutons, outils de navigation et autres éléments interactifs, ainsi que des extensions JavaScript en option
  + M’est utile afin de gagner du temps en évitant de refaire des interfaces existantes et la mise en page
* Suite Microsoft 365
  + Suite de logiciels utiles à la réalisation du projet tel que la documentation et la communication
* Looping
  + Looping est un logiciel de modélisation conceptuelle de données entièrement gratuit et libre d'utilisation.
  + M’a été utile pour réaliser le MCD MLD
* Photopea
  + Photopea est un logiciel propriétaire de retouche et de traitement d'images matricielles et vectorielles
  + Ce logiciel en ligne m’a permis de modifier des images tel que mes maquettes web
* Figma
  + Figma est un éditeur de graphiques vectoriels et un outil de prototypage
  + Cet outil m’a été pratique lors de la réalisation des maquettes

### Site web

* Design
  + Simple & Friendly afin de faciliter l’utilisation et les éléments inutiles
* Animation css/javascript
  + Minimaliste voire inexistants afin de ne pas impacté les performances web
* Priorité
  + Propreté du code la simplicité/optimisation afin qu’il puisse être repris facilement dans le futur
* Base de données et code
  + Un serveur MariaDB, PHP et PHPMyAdmin seront exécutés localement à l’aide de l’invite de commande pour faire tourner l’application web durant le TPI
* Sauvegardes
  + Un commit Git sera effectué chaque fin de journée de travail

### Suivi de projet

Afin de garantir un suivi et une transmission du projet correct, diverses mesures ont été mises en place.

Les fichiers du programme en lui-même sera déposer dans un [répertoire git](https://github.com/Medellin05001/TPI-QuizMaster)[[2]](#footnote-2) à chaque fin de journée de travail.

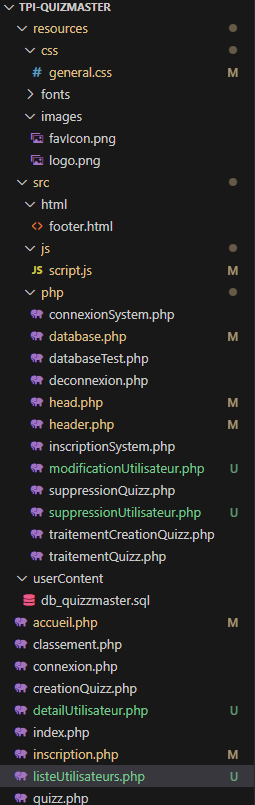
Chaque jour, je tiendrai un journal de travail qui contiendra également le planning initial dans un fichier Excel. Ce journal de travail détaillera toutes les tâches de la journée par tranche de 5 minutes et sera rempli en même temps que je réaliserai ces tâches. J'enverrai ce fichier par e-mail chaque fin d'après-midi aux experts et au chef de projet.

### Base de données : MariaDB

* **Table t\_utilisateurs :**
  + (INT 11) idUtilisateurs : Clé primaire, identifiant unique pour chaque utilisateur.
  + (VARCHAR 20) utiNomUtilisateur : Nom d'utilisateur, unique pour chaque utilisateur.
  + (VARCHAR 64) utiMotDePasse : Mot de passe de l'utilisateur (sécurisé par hachage).
  + (VARCHAR 50) utiNom : Nom de l'utilisateur.
  + (VARCHAR 35) utiPrenom : Prénom de l'utilisateur.
  + (VARCHAR 20) utiDroits : Les droits de l'utilisateur, par exemple, "utilisateur", "admin"
  + (INT 11) utiScore : Le score de l'utilisateur dans le contexte du quizz.
* **Table t\_quizz :**
  + (INT 11) idQuizz : Clé primaire, identifiant unique pour chaque quizz.
  + (VARCHAR 50) quiTitre : Titre du quizz.
  + (INT 11) fkUtilisateurs : Clé étrangère faisant référence à l'ID de l'utilisateur qui a créé le quizz.
* **Table t\_questions :**
  + (INT 11) idQuestions : Clé primaire, identifiant unique pour chaque question.
  + (VARCHAR 255) queTexte : Texte de la question.
  + (INT 11) fkQuizz : Clé étrangère faisant référence à l'ID du quizz auquel cette question est associée.
* **Table t\_reponses :**
  + (INT 11) idReponses : Clé primaire, identifiant unique pour chaque réponse.
  + (VARCHAR 255) repTexte : Texte de la réponse.
  + (INT 11) fkQuestions : Clé étrangère faisant référence à l'ID de la question à laquelle cette réponse est associée.
* Requêtes :
  + **Authentification** : Quand les données de connexion sont insérées, le code envoie une requête qui permet de vérifier si l’utilisateur existe, s’il existe on compare le mot de passe dans a base de donnée (haché) et le mot de passe inséré par l’utilisateur (haché en backend par le code) afin d’approuver la connexion s’ils correspondent sinon il refuse la connexion
  + **Classement** : Une requête obtient tous les joueurs en fonction de leurs nombres de points dans l’ordre croissants de leurs points en leurs attribuant une position dans une variable temporaire lors de la requête. Si deux joueurs ont le même score, c’est le joueur qui a le compte le plus récent qui est pris en priorité (ID plus petit)
  + **Affichage des quiz** : Sur la page d’accueil, une requête est exécutée afin d’afficher tous les quiz de la base de données (du plus récent au moins récent)
  + **Récupération information quiz spécifique** : Lors de la sélection d’un quiz, le code envoie une requête qui permet d’obtenir toutes les informations du quiz en prenant compte de toutes les tables (t\_quizz, t\_question, t\_reponse)
  + **Ajout score** : Quand un utilisateur finit un quiz, son score est calculé. S’il est supérieur à 0 le score s’additionne à celui actuel.
  + **Création quiz** : Quand un administrateur créé un quiz, le code envoie 4 requêtes.
    - Création quiz (t\_quizz)
    - Création question (t\_question)
    - Création réponse (t\_reponse)
    - Une requête sera chargée de récupérer l’id de l’élément qui vient d’être créé afin de la transmettre la clé étrangère
  + **Suppression quiz** : Quand un administrateur supprime un quizz, le code envoie 3 requêtes.
    - Suppression réponses (t\_reponses)
    - Suppression questions (t\_question)
    - Supprression quiz (t\_quizz)
    - Tout cela est englober dans un transactions afin de tout annuler s’il y a une erreur
  + **Affichage des utilisateurs** : Quand un administrateur est sur la page de la liste des utilisateurs exécutée afin d’afficher tous les utilisateurs de la base de données. 2 boutons sont également présents (afin de modifier et supprimer l’utilisateur)
  + **Modification utilisateur** : Lorsque l’utilisateur insère les nouvelles données de l’utilisateur, une requête est exécutée qui met à jour la base de données au niveau des données de l’utilisateurs
  + **Suppression utilisateur** : Quand un administrateur supprime un utilisateur en appuyant sur le bouton prévu pour, 4 requêtes sont envoyées
    - Suppression réponses de l’utilisateur (t\_reponses)
    - Suppression questions l’utilisateur (t\_question)
    - Suppression quiz l’utilisateur (t\_quizz)
    - Suppression de l’utilisateur (t\_utilisateurs)
    - Tout cela est englober dans un transactions afin de tout annuler s’il y a une erreur

### Architecture du programme

Durant ce projet j’ai réalisé une architecture spécifique qui consiste à séparer les pages visibles par l’utilisateur et les « pages systèmes ».



Ressources : Mise en page des interfaces

Figure 12 Architecture : programme entier

PHP

JS

HTML

« Frontend » Interfaces utilisateurs, ce que l’utilisateur pourra voir

userContent : Contient tout ce qui est contenu des utilisateurs tel que la base de donnée

Fonts

Images

CSS

Src : « Backend », pratiquement toutes le traitement d’informations, fonctionnalités et requêtes se passe dedans, l’utilisateur ne verra pas ces pages

### Dépendance externe

Durant ce projet diverses liaisons externes au code ont été utilisées, afin d’avoir une vue globale sur ceux-ci, ils sont listés ici :

* Bootstrap : Mise en page facilitée, utilisé dans toutes les interfaces utilisateurs, utilisation de bout de code HTML/CSS et javascript déjà disponible gratuitement sur internet afin d’accélérer la création d’interfaces
* Flexbox : Aide à la mise en page des éléments lors du CSS, comme pour l’alignement des éléments
* jQuery : Bibliothèque javascript qui permet d’utiliser des syntaxes simplifiées et des fonction déjà prêtes et optimisées

### Utilisation schématisée

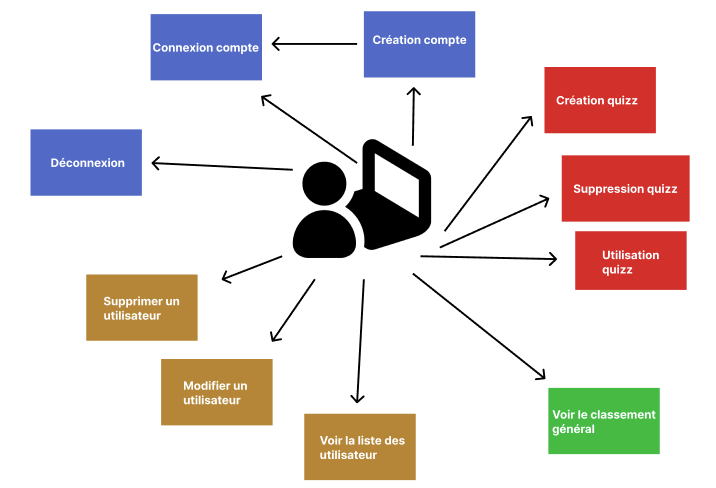


Figure 13 Maquette : Utilisation possibles

# Réalisation

## Liste des fichiers : Répertoire

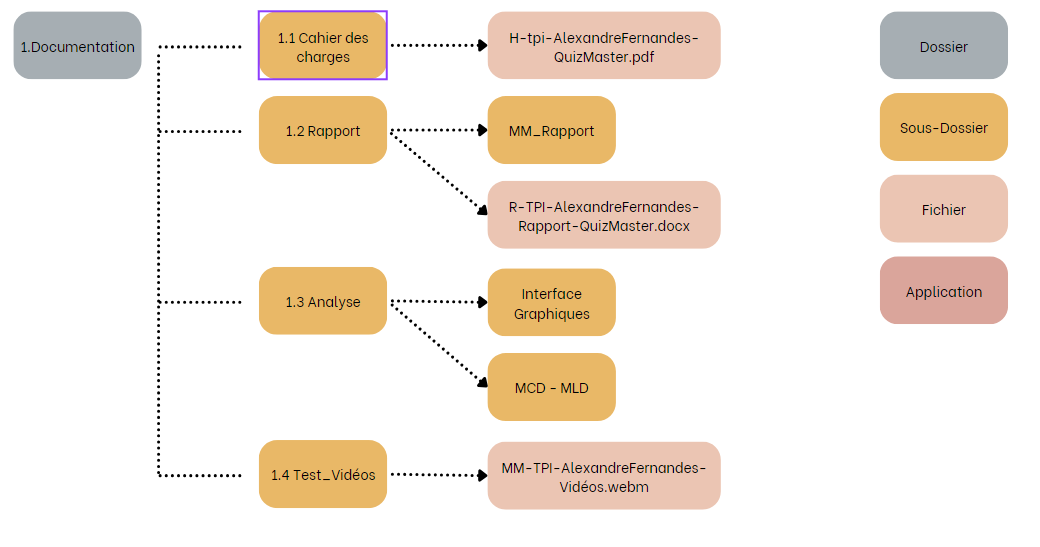


Figure 14 Répertoire : Projet (1)

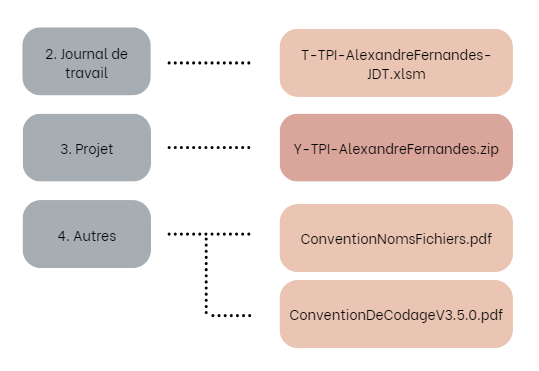
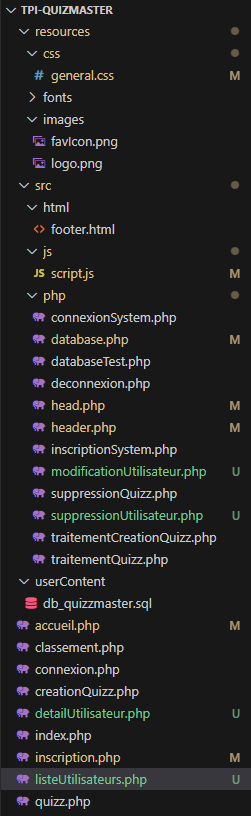


Figure 15 Répertoire : Projet (2)

## Liste des fichiers : Code source



*Interface utilisateur : Utilisation quizz*

*Interface utilisateur : Liste des utilisateurs*

*Interface utilisateur : Inscription*

*Index (routeur)*

*Interface utilisateur (admin) : Information d’un utilisateur*

*Interface utilisateur (admin) : Création quiz*

*Interface utilisateur : Connexion*

*Interface utilisateur : Classement*

*Interface utilisateur : Accueil*

*Contient toutes les requêtes SQL du programme*

*Base de données*

*Code pour modifier un utilisateur*

*Code pour supprimer un quizz*

*Code pour supprimer un utilisateur*

*Traitement des données et requêtes lors de la création d’un quizz*

*Traitement des données et requêtes lors de la création d’un compte*

*La navBar est entièrement codée ici*

*Différentes liaisons, cookies, session start sont stockées ici*

*Système qui permet de déconnecter l’utilisateur*

*Tests Unitaires*

*Traitement des données et requêtes lors de la connexion*

**« php » Contient tous les fichier php qui contribuent au traitement de données**

*les différents scripts sont stockés dedans*

**« js » Contient tous les fichier script du programme**

*footer présent en bas de page*

**« html » Contient toutes les pages uniquement html du site**

**« src » traitement d’informations, fonctionnalités et requêtes**

*logo utilisé pour au niveau de l’onglet*

*logo utilisé pour le header*

**« images » Contient toutes les images/logo**

**« fonts » Contient toutes les polices d’écritures (vide)**

*css utilisé pour l’ensemble du site web*

**« userContent » Données utilisateurs**

*Traitement des données et requêtes lors de l’envoie d’un quizz par un utilisateur*

*« bleu » fichier*

**« rouge » Dossier**

**« css » Contient tous les fichiers css**

Figure 16 Répertoire : Programme

**« resource » Mise en page des interfaces**

## Base de données : Comment fonctionnent les requêtes



Figure 17 SQL : Requêtes préparées

Afin de faire fonctionner les requêtes de façon sécuriser, une méthode a été mis en place.

\_\_construct () : Initialise la connexion à la base de données en PDO, permet de se connecter à une base de données MySQL. Elle est appelée à chaque nouvelle création d’instance de la classe, notamment dans le header.

querySimpleExecute ($query) : Permet de prendre une requête en paramètre, de l’exécuter, et de retourner son résultat obtenu. Elle est utilisée pour exécuter les requêtes sans paramètres

QueryPrepareExecute ($query, $binds) : Prend la requête SQL, la lie avec les paramètres puis l’exécute. Elle est utilisée pour exécuter des requêtes préparées avec des paramètres dynamiques, afin de protéger des injections SQL.  
En préparent les requêtes, les requêtes seront traitées comme des valeurs littérales et non du code SQL

formatData ($req) : Permet de formater les données sous formes de tableau, le code extrait les résultats sous forme de tableau associatif dans lequel les clés deviennent des noms de colonnes et les valeurs deviennent les données.

Cela me permet de manipuler les données plus facilement de l’exécution de la requête : $utilisateur[0]['utiNom']

## Base de données : Authentification

Les premières fonctions qui furent implémentées furent celles liées à l'authentification. Cela inclut la connexion, la création de compte (inscription) et la déconnexion.

### Requêtes préparées utilisées

Requête qui permet de sélectionner un utilisateur dans la base de données, utile pour vérifier si le nom d’utilisateur est disponible et récupérer les informations d’un utilisateur spécifique.

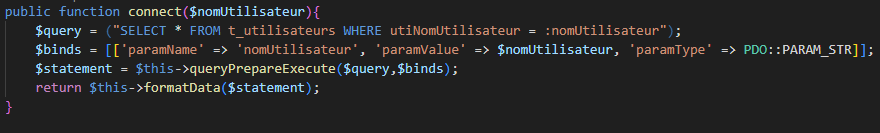


Figure 18 SQL : Connexion utilisateur

Cette requête SQL permet créer un nouvel utilisateur avec les paramètres reçus.

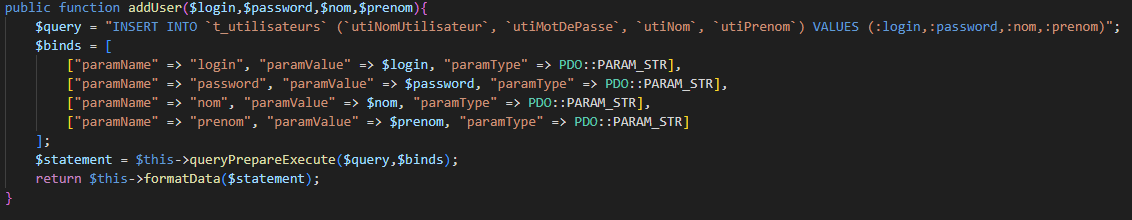


Figure 19 SQL : Création d’utilisateur

### Connexion

L’interface utilisateur qui permet la connexion se passe sur la page « connexion.php », un formulaire est présent afin de réceptionner et envoyer les informations que l’utilisateur inscrit.

Les données sont envoyées sur la page « script.js », plus précisément dans la fonction « $("#connectSubmit").click ».

Le rôle de cette fonction est de récupérer les informations et vérifier si elles correspondent aux deux restrictions imposées dans le code puis retourner à l’utilisateur le résultat :

* Input « nom d’utilisateur » vide 🡺 Veuillez entrer un nom d’uti…
* Input « mot de passe » vide 🡺 Veuillez entrer un mot de passe

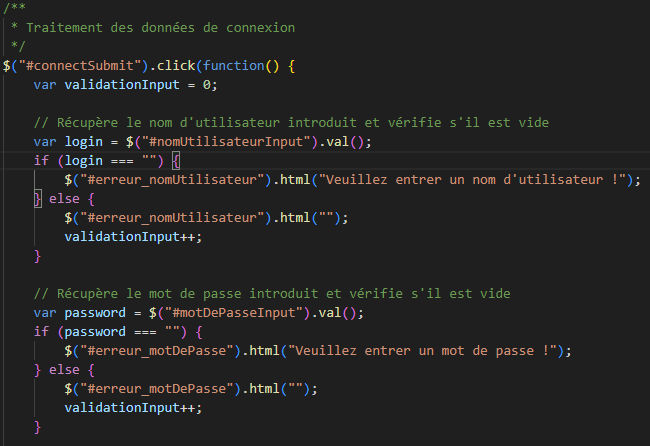


Figure 20 jQuery : Restriction Input connexion

Si toutes les conditions sont respectées les informations sont renvoyées en POST et les informations en paramètre sur la page « connexionSystem.php » à l’aide d’ajax. Ajax permet d’envoyer les informations mais également d’en recevoir afin que l’utilisateur reçoive un retour (connexion réussie ou ratée).

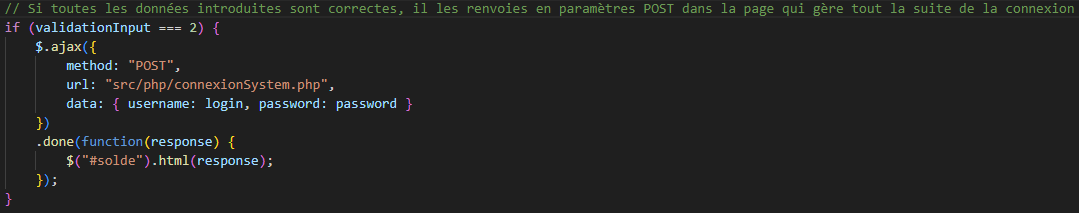


Figure 21 Ajax : Renvoie de page connexion en POST

ConnexionSystem.php récupère les informations envoyées en post (par script.js).

La page vérifie si les données sont bien présentes stock les données de la base de données concernant le nom d’utilisateur entrer

Cela est réaliser par une requête SQL « connect »

Le mot de passe récupérer en POST sera hacher (sha256) afin de le vérifier avec celui de la variable qui contient les informations de la base de données

La connexion est approuvée ou refusée en fonction du résultat

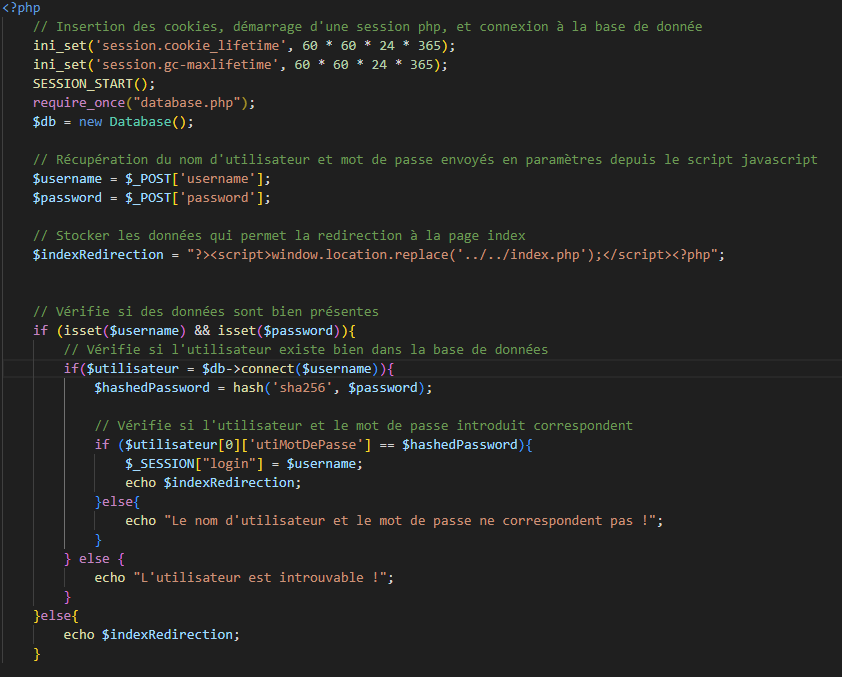


Figure 22 PHP : Traitement des données de connexion

### Inscription

L’utilisateur va pouvoir se créer un compte sur la page « inscription.php ». Comme pour la page connexion, l’utilisateur se retrouve avec un formulaire à remplir puis à envoyer afin de procéder à l’inscription.

Le processus est similaire à celui de la connexion mais avec plus de paramètres, les données sont premièrement envoyées sur la page « script.js » sur la fonction $("#registerSubmit").click

Cette fonction va récupérer les informations et vérifie si elles sont conformes aux exigences du programme. La page retourne des erreurs dans le formulaire afin d’indiquer à l’utilisateur ses erreurs.

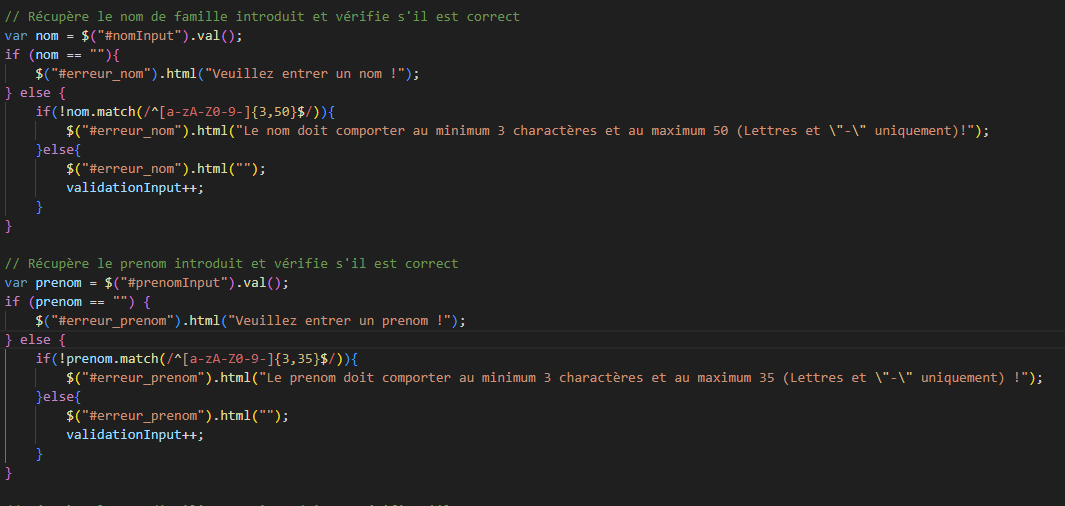


Figure 23 jQuery: Restriction Input inscription

Liste des restrictions pour l’inscription :

* Nom de famille : Uniquement lettres, minimum 3 et maximum 50
* Prénom : Uniquement lettres, minimum 3 et maximum 35
* Nom d’utilisateur : Lettres et chiffres, minimum 3 et maximum 20
* Mot de passe : Uniquement lettres et chiffres, minimum 8 et maximum 64

Si toutes les conditions sont respectées les informations sont renvoyées en POST et les informations en paramètre sur la page « inscriptionSystem.php » à l’aide d’ajax. Ajax permet d’envoyer les informations mais également d’en recevoir afin que l’utilisateur reçoive un retour (connexion réussie ou ratée).

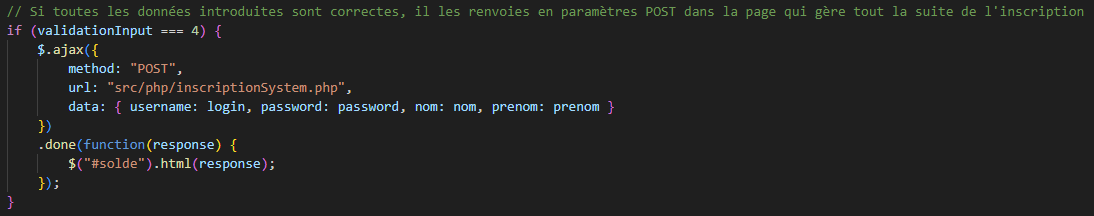


Figure 24 Ajax : Renvoie de page inscription en POST

InscriptionSystem.php récupère les informations envoyées en post (par script.js)

La page vérifie si les données sont bien présentes. Le code vérifie si l’utilisateur est déjà présent dans la base de données (requête SQL « connect ».

Si le nom d’utilisateur est disponible, le mot de passe introduit dans le formulaire est haché (sha256) puis un compte est créé avec toutes les données inscrites dans le formulaire à l’aide de la requête SQL « addUser »

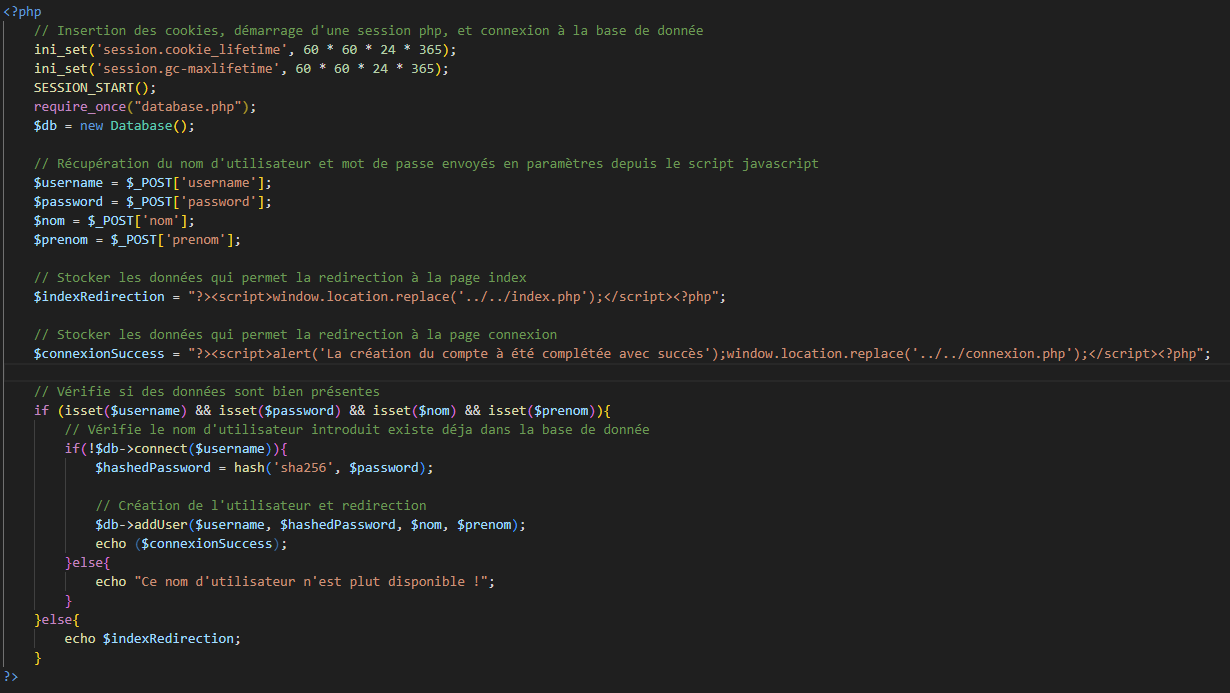


Figure 25 PHP : Traitement des données d’inscription

### Déconnexion

Afin d’effectuer la connexion, j’ai inclus dans le header un texte « Déconnexion » qui permet de se déconnecter en cliquant dessous tout simplement.

Quand l’utilisateur clique dessus il est renvoyé vers la page deconnexion.php qui détruit la session active tout simplement.



Figure 26 PHP : déconnexion utilisateur

## Base de données : Affichage tous les quiz

### Requêtes préparées utilisées

Avec cette requête, on pet récupère les informations du créateur du quiz et les informations du quiz



Figure 27 SQL : obtenir tous les quiz

### Affichage des quiz

Quand l’utilisateur est sur la page d’accueil, le code récupère tous les quizz afin de les stocker dans une variable



Figure 28 Code : stocker tous les quiz

Une boucle foreach va parcourir la variable afin d’afficher le titre du quiz et son créateur. Cette boucle est affichée sous forme de carte pour l’utilisateur.

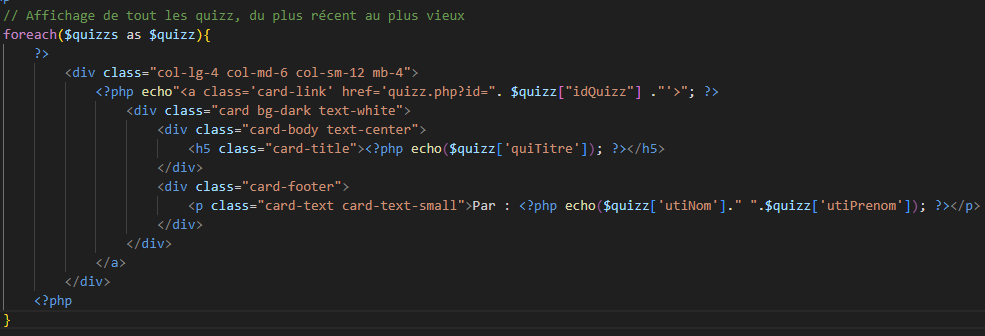


Figure 29 PHP : Affichage de tous les quiz

## Base de données : Affichage quiz spécifique

### Requêtes préparées utilisées

Cette requête permet d’obtenir l’entièreté des informations d’un quiz spécifique (Question, Réponse, Titre du Quiz). Dans cette requête on relie toutes les tables.

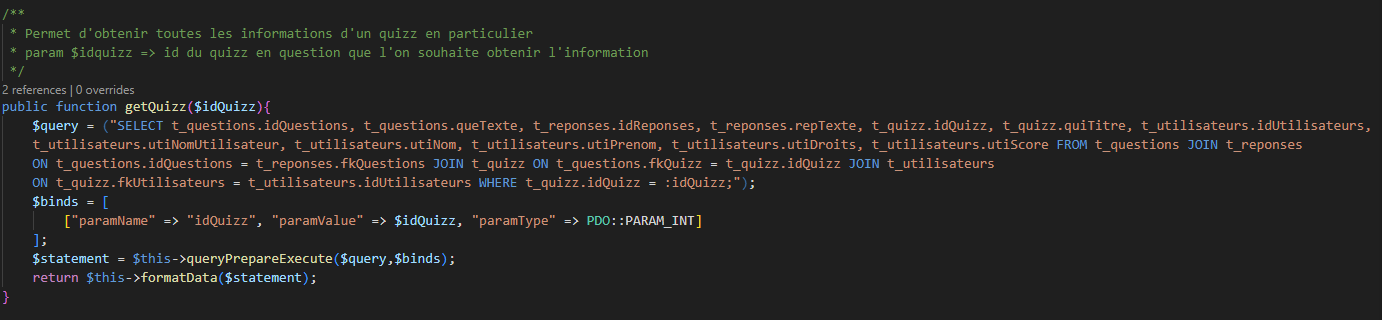


Figure 30 SQL : obtenir un quiz spécifique

### Affichage du quiz

Avec cette ligne de code, on peut accéder obtenir un lien vers la page quizz comportant un paramètre dans le lien : l’id du quiz.



Figure 31 PHP : accès à la page d'un quiz spécifique

Afin d’afficher les informations du quiz, on stocke tout dans une variable.



Figure 32 PHP : stock les informations du quiz

Pour afficher toutes les questions, on utilise une boucle foreach qui va parcourir toutes les questions en les affichants.



Figure 33 PHP : Affichage du quiz

## Base de données : Traitement du quiz

### Requêtes préparées utilisées

Cette requête permet tout simplement de mettre à jour le score de l’utilisateur après l’envoi d’un nouveau quizz, afin d’ajouter ses nouveaux points.

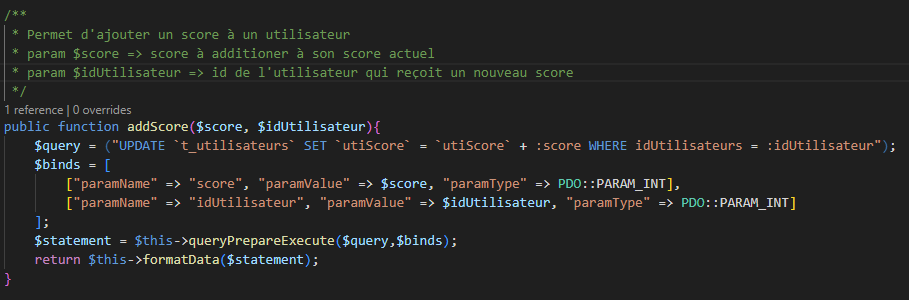


Figure 34 SQL : update le score de l’utilisateur

### Traitement

Quand un quiz est envoyé, le code va se charger de récupérer les informations et de les envoyées dans la page « traitementQuizz.php »

Cette page va récupérer les réponses que l’utilisateur à entrer et les stocker dans une variable

Elle va également récupérer les réponses de la base de données (réponses correctes) afin de les stocker dans une seconde variable

Le code va se charger de comparer les deux tableaux, quand une réponse est correcte, il gagne un point. Quand une réponse est fausse, il perd un point.

*Sachant qu’il ne peut pas avoir un score en dessous de 0 points.*

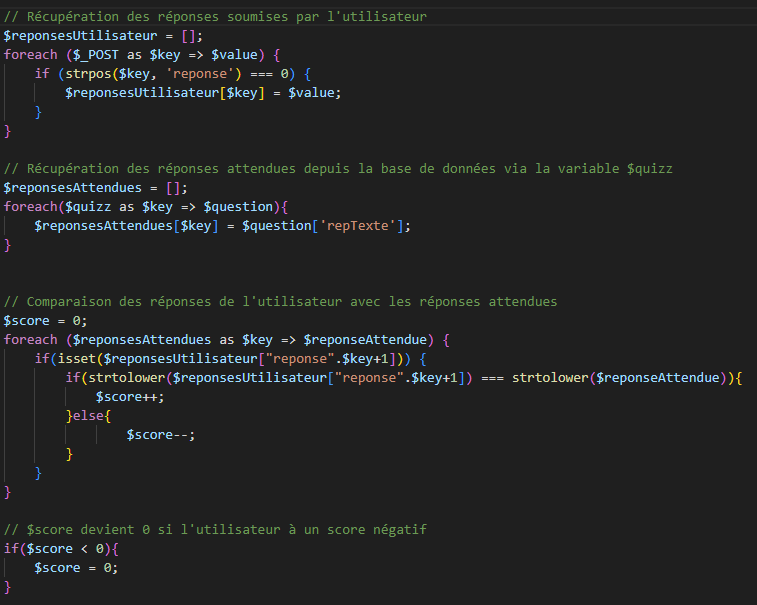


Figure 35 Code : traitement des réponses d'un quiz

### Affichage des réponses

Pour afficher les réponses, le code stocke les données de « traitementQuizz.php » dans une variable session afin de pouvoir les réutilisées facilement dans la page de quizz.

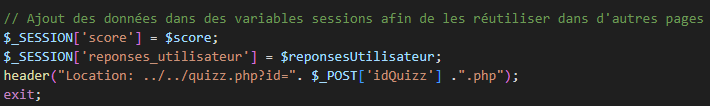


Figure 36 PHP : stock résultat utilisateur

Pour résumer : quand l’utilisateur envoie une réponse, il est donc renvoyé vers « traitementQuizz.php » puis renvoyer instantanément vers le quizz ou il était.

Cependant, la page quizz a été séparé en deux parties afin d’éviter d’éparpiller le code en recréant une nouvelle page. Une partie pour effectuer le quiz, et une partie pour afficher les réponses une fois le quiz envoyé

La question est affichée avec sa réponse, et la réponse de l’utilisateur également.

Le score est mis à jour dans la base de données au niveau de l’utilisateur

Un total est affiché en bas de page puis le nouveau score général de l’utilisateur.



Figure 37 PHP : Affichage réponses utilisateurs

## Base de données : Création quiz

### Requêtes préparées utilisées

Les 3 requêtes qui suivent permettent une nouvelle insertion dans la table quizz, réponses, questions afin de faire la création d’un nouveau quiz

Quiz :

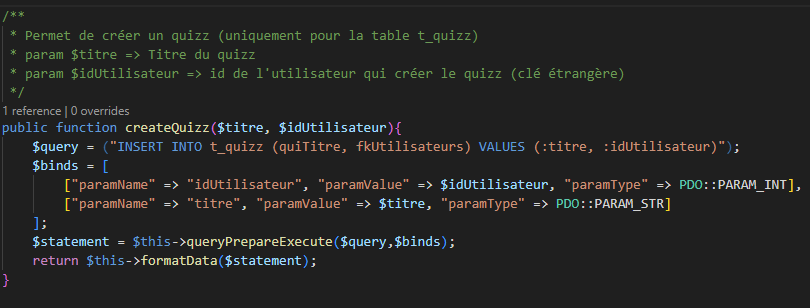


Figure 38 SQL : insertion t\_quizz

Question :

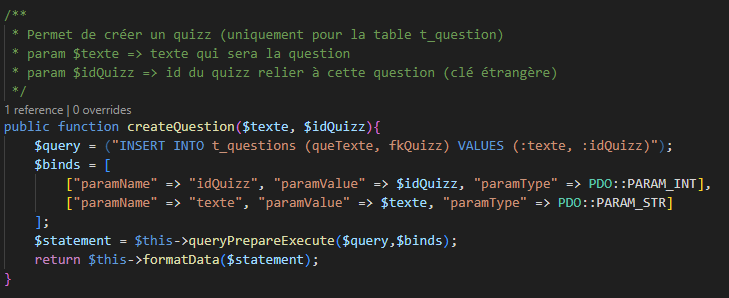


Figure 39 SQL : insertion t\_questions

Reponse :

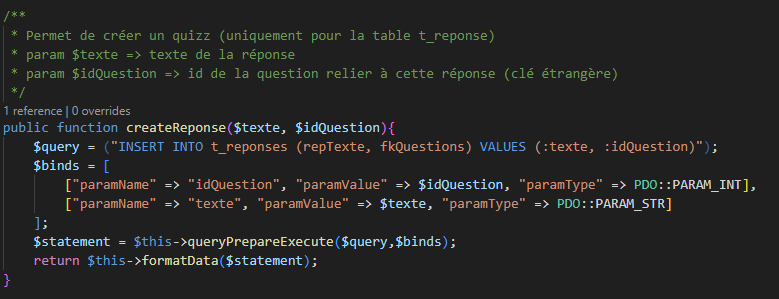


Figure 40 SQL : insertion t\_reponses

Cette requête permet d’obtenir le dernier ID créer dans la base de données afin de faire la création complète du quizz en faisant la gestion des clés étrangères.

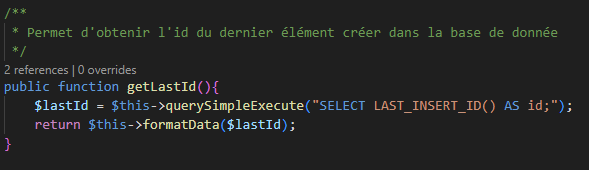


Figure 41 SQL : Obtenir dernier id créer dans la base de données

### Traitement et création du quiz

Quand un administrateur souhaite ajouter un nouveau quiz au système il doit passer par la page « CreationQuizz.php ».

Un simple formulaire sera présent et enverra toutes les données à la page « traitementCreationQuizz.php »

La page de traitement va vérifier à chaque input de réponse/question :

* Si la question est associée à une réponse
* Si la réponse est associée à une question
* Si les deux inputs ne sont pas vides

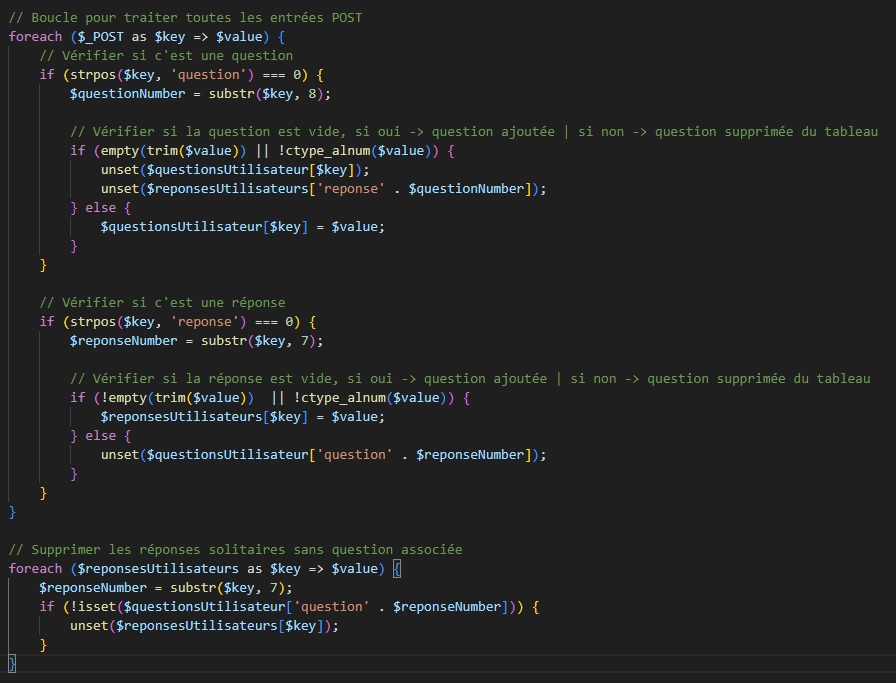


Figure 42 PHP : Traitement données création d’un quiz

Quand ce test est validé la question et la réponse sont ajoutés à un tableau séparément

Si le total de question/réponse est supérieur à 0, le quizz est créé

La création du quizz se compose de 3 parties distinctes :

* Insertion table t\_quizz
  + Stock dans une variable l’id du nouvel élément
* Insertion table t\_questions (boucle foreach pour insérer toutes les questions)
  + Stock dans une variable l’id du nouvel élément
* Insertions table t\_reponses (boucle foreach pour insérer toutes les questions)
  + Stock dans une variable l’id du nouvel élément



Figure 43 PHP : Création d’un quizz

## Base de données : Suppression quiz

### Requêtes préparées utilisées

Afin de garantir une suppression complète, beginTransation() permet de regrouper toutes les transactions afin de les considérer comme un groupe

Une fois toutes les opérations effectuées $this->connector->commit() sera effectué et donc la suppression sera effectué.

Mais si un soucis est détecter dans une seule des transaction, rollBack() va permettre de tout annuler dans le groupe, comme si rien n’a été effectuer depuis le début.



Figure 44 SQL : suppression quizz

Lorsque l’administrateur appuie sur le bouton « suppression quiz » d’un quizz, une fonction js est appeler. Cette fonction va permettre d’envoyer un message de confirmation à l’administrateur afin qu’il soit sûr de son action



Figure 45 PHP : renvoie vers suppression d'un quizz

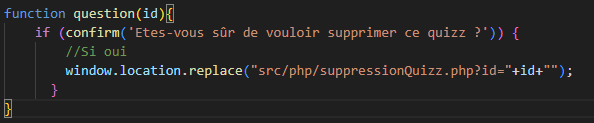


Figure 46 JS : message de confirmation de suppression

Quelques vérifications sont effectuées avant que le quizz ne soie supprimer :

* Si un id est présent dans l’URL
* Si l’utilisateur accédant à l’URL est connecté
* Si l’utilisateur accédant à l’URL est administrateur

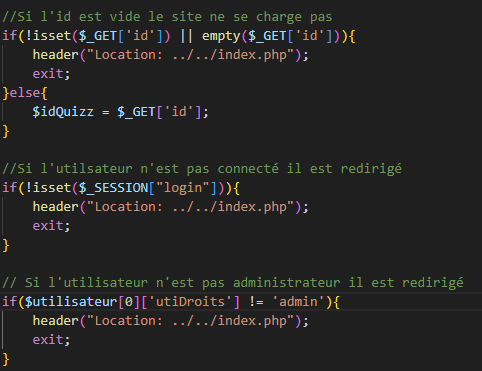


Figure 47 PHP : Sécurité avant suppression quizz

Une dernière vérification est effectuée avant de supprimer le quizz, une requête va être effectuée afin de vérifier si le quizz existe, si oui il est supprimé, sinon il y a un message d’erreur



Figure 48 PHP : Suppression Quizz

## Base de données : Affichage classement

### Requêtes préparées utilisées

Cette requête permet d’obtenir le classement de tous les joueurs, une variable est créée afin de déterminer sa position.

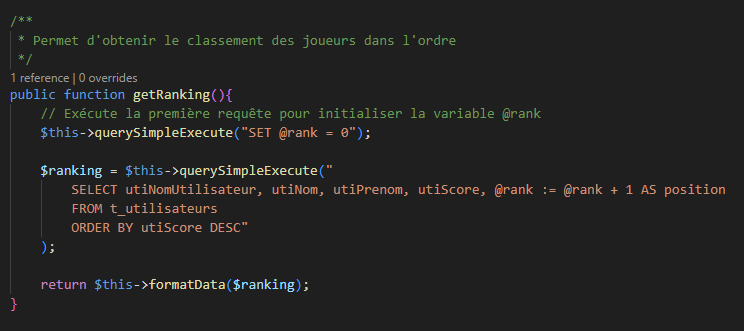


Figure 49 SQL : Obtenir le classement général

### Affichage du classement

Pour consulter le classement qui affiche les 7 joueurs ayant obtenus le plus de points, et visualiser sa propre position, l’utilisateur peut se rendre sur la page « classement.php »

Une simple variable contenant les utilisateurs dans l’ordre décroissant au niveau des points est créer puis est lue par une boucle foreach afin d’afficher toutes ses informations sous forme de tableau.

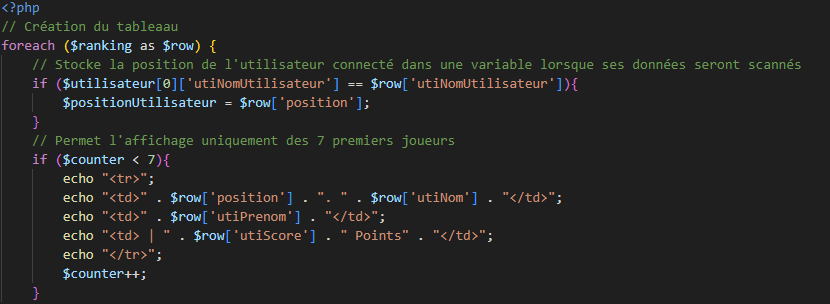


Figure 50 PHP : Afficher les 7 premiers utilisateurs du classement

Un bout de code permet de tenir informer l’utilisateur de sa position et ses points

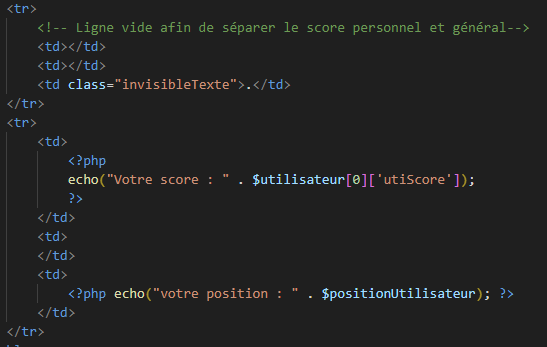


Figure 51 PHP : Affichage des points/position de l'utilisateur connecté

## Base de données : Gestion des utilisateurs

### Requêtes préparées utilisées

Cette requête permet d’obtenir toutes les informations de tous les utilisateurs

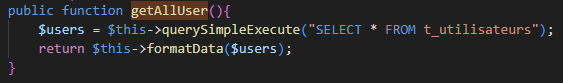


Figure 52 SQL : Obtenir tous les utilisateurs

Cette requête modifie les données de l’utilisateur en fonction de ce qu’il reçoit en paramètre

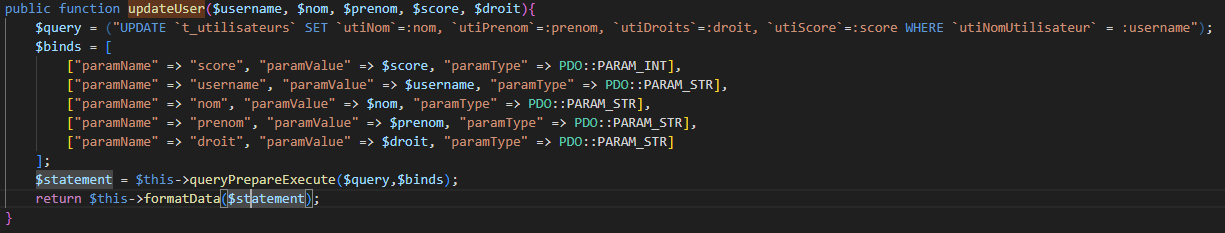


Figure 53 SQl : Modifier un utilisateur

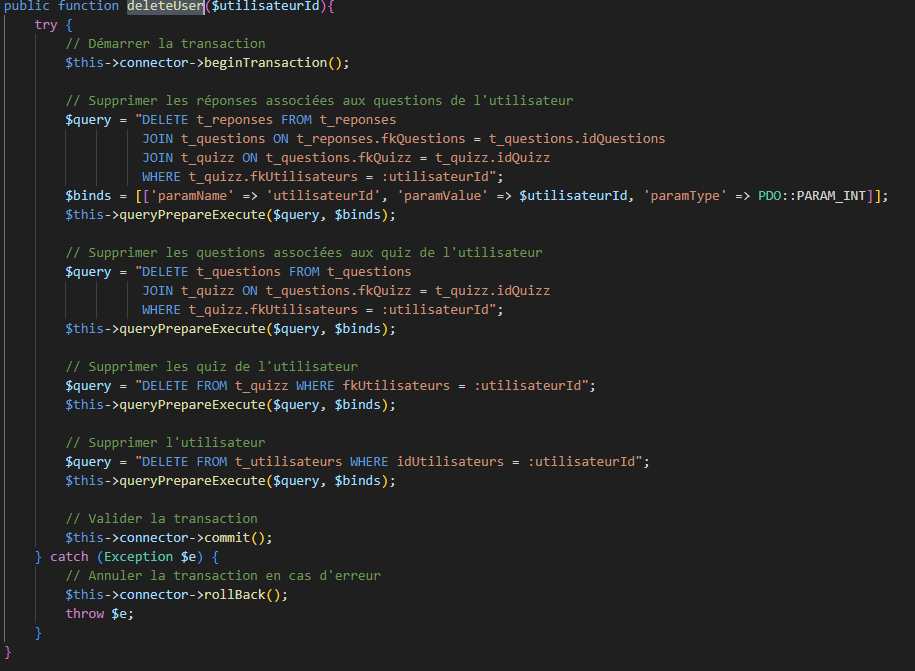
Cette requête l’utilisateur qu’l reçoit en paramètre

Figure 54 SQL : supprimer un utilisateur

### Affichages de tous les utilisateurs

Pour afficher tous les utilisateurs sous forme de tableau, on récupère les informations et on les stocks dans une variable



Figure 55 PHP : Récupérer les infos des utilisateurs

Après les avoir stocker, on parcourt le tableau avec une boucle foreach afin de les afficher à l’écran de l’utilisateur



Figure 56 PHP : Afficher les infos des utilisateurs

### Modification

Lors de l’affichage de utilisateurs on affiche un bouton sous forme d’icône de crayon qui renvoie à une page « detailUtilisateur.php » avec l’id de l’utilisateur en paramètre

Cette page récupère l’id de l’utilisateur et affiche les données de celui-ci sous forme de formulaire.



Figure 57 PHP : Récupération des données de l'utilisateur à modifier

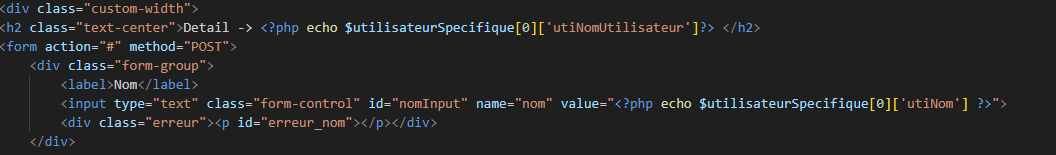
Dans ce début de tableau on peut voir un exemple dans la manière que les données sont affichées, un input avec les données à l’intérieur afin de facilité la modification

Figure 58 PHP : Formulaire de modification d'utilisateur

Lorsque les données sont complétées et le bouton submit est pressé, un code jQuery prend le relais en vérifiant les données de la même manière que l’inscription afin de vérifier si les données suivent les restrictions du site web.

Puis, sur la page « modificationUtilisateur.php » une requête updateUser est exécutée avec les nouvelles informations en paramètres, ce qui effectue la modification dans la base de données

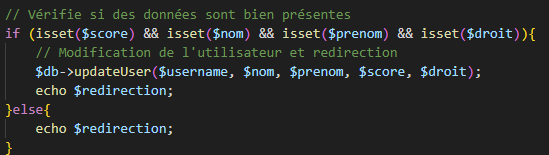


Figure 59 PHP : Modification des données de l'utilisateur

### Suppression

Lors de l’affichage de utilisateurs on affiche un bouton sous forme d’icône de poubelle crayon qui, lors du clique, un script javascript est appelé afin de demander une confirmation

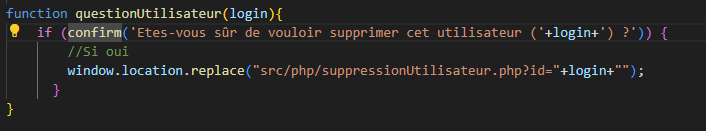


Figure 60 JS : Confirmation de suppression d'utilisateur

Si la confirmation est effectuée, l’utilisateur est renvoyé vers la page suppressionUtilisateur avec l’id en paramètre URL.

La page récupère l’id passé en paramètre dans le lien afin de récupérer les informations de l’utilisateurs et vérifier certaines sécurités.

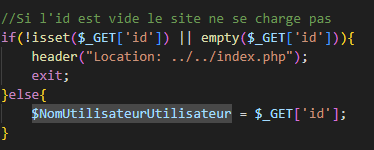


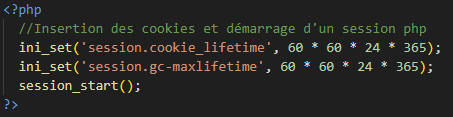
Figure 61 PHP : Récupération de l'id dans l'URL



Figure 62 PHP : suppression de l'utilisateur

## Cookies

Des cookies sont utilisés sur toutes les pages afin que l’utilisateur n’aie pas besoin de se reconnecter à chaque fois. La durée de ce cookie est d’une 365 jours.



## Base de données : résultat visuel (PHPMyAdmin)

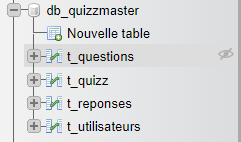


Figure 63 PhpMyAdmin : Arborescence

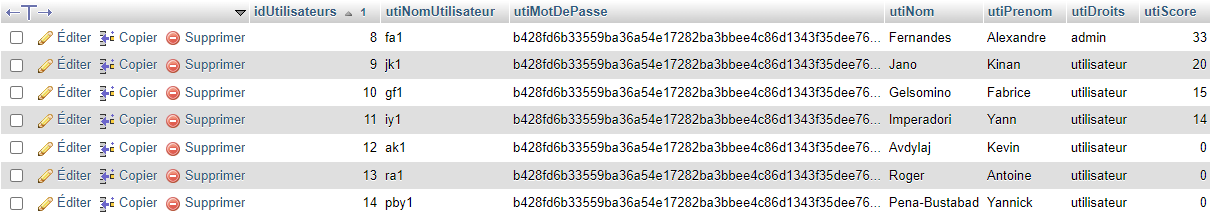


Figure 64 PhpMyAdmin : Table utilisateur

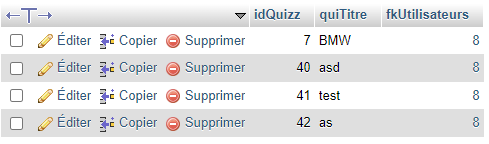


Figure 65 PhpMyAdmin : Table quizz

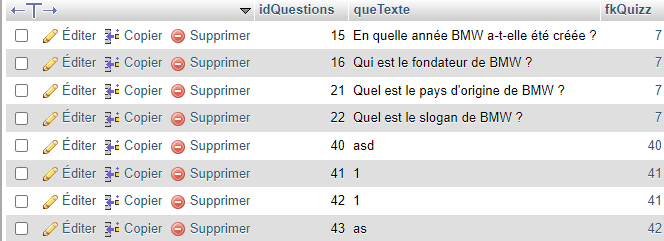


Figure 66 PhpMyAdmin : Table question



Figure 67 PhpMyAdmin : Table réponse

## Versions du projet

Toutes les versions du projet GitHub dans mon dépôt. Afin de consulter l’historique des versions depuis mon début il suffit de cliquer sur le bouton « Commits » de mon répertoire et de sélectionner celui voulu.

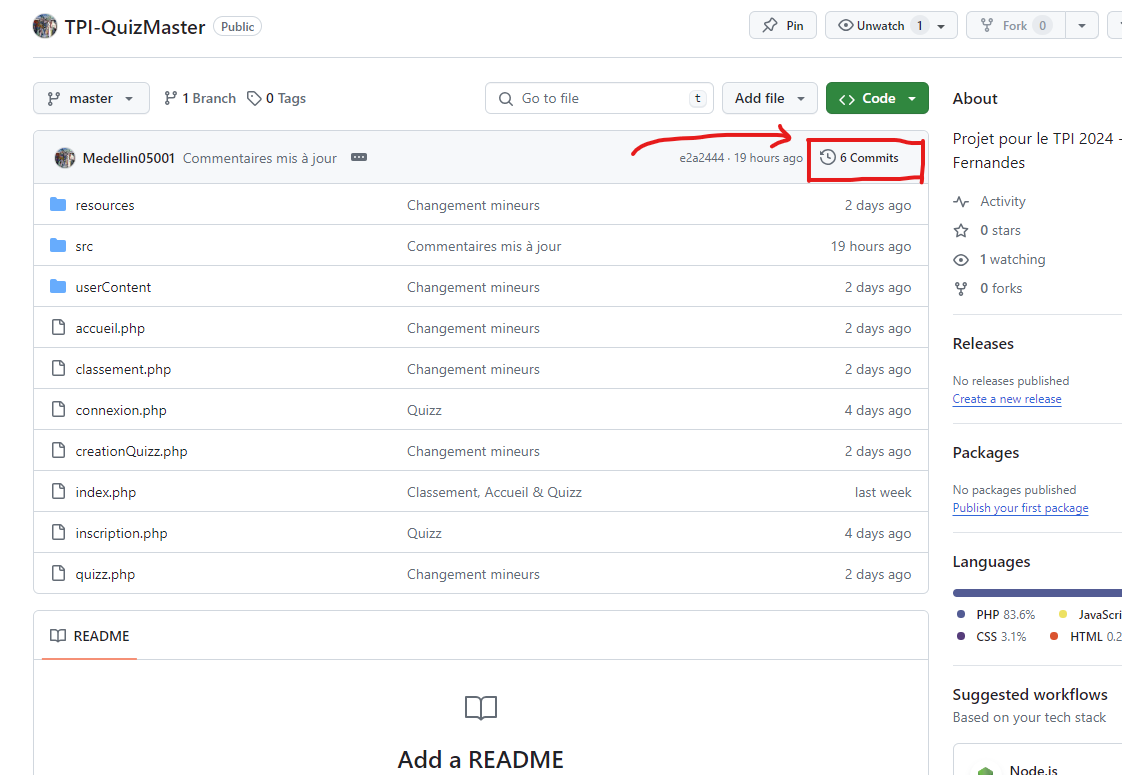


Figure 68 GitHub : Accès au commit



Figure 69 GitHub : Choix du commit

## Description des tests effectués

**Code couleur :**

|  |  |  |
| --- | --- | --- |
| **Non validé** | **Partiellement validé** | **Validé** |

**Description catégorie :**

|  |  |  |
| --- | --- | --- |
| **Fonctionnalité** | **Retour** | **Résultat** |
| Cette case décrira la fonctionnalité que l’on veut juger | Cette case démontrera ce qui est attendu de la fonctionnalité de manière résumé | Cette case contiendra la couleur indiquant le résultat du test |

|  |  |  |
| --- | --- | --- |
| Fonctionnalité | Retour | Résultat |
| Page connexion   * Bouton connexion utilisateur | Le bouton permet de rediriger l’utilisateur ayant inscrit les informations de compte correct sur la page d’accueil en étant connecté à son compte |  |
| * Bouton création compte | Le bouton redirige correctement sur la page d’inscription afin que l’utilisateur puisse se créer un compte par la suite |  |
| * Retour d’erreur lors de la saisie d’information | Quand l’utilisateur saisit une donnée pas accepté par le site, une erreur est retournée sous un petit texte rouge en dessous du champ en question. L’erreur retourne correctement ce qui ne convient pas |  |
|  |  |  |
| Page création d’un compte   * Bouton création utilisateur | Le bouton permet de rediriger l’utilisateur ayant inscrit des informations correctes sur la page de connexion, le compte voulu est créé également. |  |
| * Bouton connexion compte | Le bouton redirige correctement sur la page de connexion afin que l’utilisateur puisse se connecter par la suite |  |
| * Retour d’erreur lors de la saisie d’information | Quand l’utilisateur saisit une donnée pas accepté par le site, une erreur est retournée sous un petit texte rouge en dessous du champ en question. L’erreur retourne correctement ce qui ne convient pas |  |
|  |  |  |
| Page d’accueil   * Affichage de tous les quizz (plus récents d’abord) | Les quizz sont tous affichés sur la page d’accueil (plus récent au moins récent). Les informations tel que le titre et le nom/prénom du créateur sont bien affichés également |  |
| * Quizz cliquable | Lors du clique sur un quizz, l’utilisateur est bien redirigé vers la page quizz.php?id=…  Le lien contient bien l’id du quizz sur lequel l’utilisateur aurait cliqué |  |
| * Supprimer Quizz | Lorsque l’utilisateur clique sur supprimer quizz, le quizz sélectionner est correctement supprimer |  |
|  |  |  |
| Page Quizz (utilisation)   * Envoie du formulaire | Le bouton soumettre permet d’envoyer les données saisies par l’utilisateur |  |
| * Affichage des réponses | Le formulaire se transforme en « tableau de réponse » en affichant :  La question | la réponse correcte | la réponse de l’utilisateur |  |
| * Calcul du score | Le score de l’utilisateur est calculé de la manière suivante : réponse correcte = +1 | réponse incorrecte = -1 | l’utilisateur ne peut pas avoir moins que 0. |  |
| * Affichage du résumer du score | Un total est affiché (2/6) avec le nouveau score de classement de l’utilisateur |  |
|  |  |  |
| Page création de quiz   * Bouton ajouter une question | Le bouton ajouter de manière correct une paire d’input (question & réponse) |  |
| * Bouton création quiz | Le bouton permet de créer un nouveau quizz à l’aide des informations ajoutées dans les différents champs, tout en prenant compte des inattendus qui ont été ajoutés |  |
| * Prise en compte des champs complets | Une vérification est effectuée afin de supprimer toutes les paires incomplètes (question sans réponse associée et vice-versa) et les paires totalement vides. |  |
| * Création des quizz conformes aux règles | Le code vérifie si le nom du quizz comporte maximum 50 caractères et contient uniquement lettre(accent)/chiffres.  De même pour les questions/réponses mais avec maximum 255 caractères.  Le quizz doit comporter minimum 1 question/réponse valable |  |
|  |  |  |
| Page classement   * Affichage des 7 premiers utilisateurs | Un tableau affichant les 7 meilleurs utilisateurs est affiché (Nom|Prenom|Points) |  |
| * Affichage score de l’utilisateur | La page affiche le score de l’utilisateur et sa position par rapport aux autres joueurs |  |
| Liste utilisateurs   * Affichage des utilisateurs | Un tableau affiche les informations de tous les utilisateurs de la base de données. Cette page est uniquement accessible par les administrateurs |  |
| * Modification utilisateur | Un bouton permet de modifier certaines données de l’utilisateur. |  |
| * Supprimer utilisateur | Un bouton permet de supprimer un utilisateur sélectionner ainsi que tous les quizz qu’il a créés. L’administrateur ne peux pas se supprimer lui même |  |
| Header   * Accéder aux différentes pages | Lors du clique d’un élément sur le header, on est correctement redirigé vers la page en question (ex : classement) |  |
| * Information sur la page active | Le texte de l’élément correspond à la page sur la quel nous somme devient blanc afin d’indiquer sur quel page l’utilisateur est (ex : Accueil) |  |
| * Déconnexion | Le bouton texte déconnexion, déconnecte correctement l’utilisateur de sa session et le renvoie vers la page index |  |
|  |  |  |
| Index   * Redirection de l’utilisateur | La page redirige correctement l’utilisateur en fonction de son état (connecté ou non) |  |
|  |  |  |
| Toutes les pages   * Responsive Design | Toutes les pages sont adaptées pour accueillir mobile/tablette/ordinateur |  |

## 

## Test unitaire

|  |  |  |
| --- | --- | --- |
| Test Unitaire   * Get Ranking | Le test unitaire créer 3 utilisateurs différents avec des score différents, le classement donné par le test unitaire est correct |  |

Ce test unitaire est séparé en 3 parties :

* Intervention manuelle de l’administrateur : afin que le test unitaire puisse être effectué, l’administrateur doit vider la table t\_utilisateurs puis rendre public les fonctions querySimpleExecute et queryPrepareExecute du fichier database.php le temps du test unitaire
* Le code va insérer dans la base de données divers utilisateurs avec des score différents



Figure 70 Test Unitaire : préparation test getRanking

* Le code va vérifier si la fonction getRanking à correctement sortit les utilisateurs dans l’ordre.

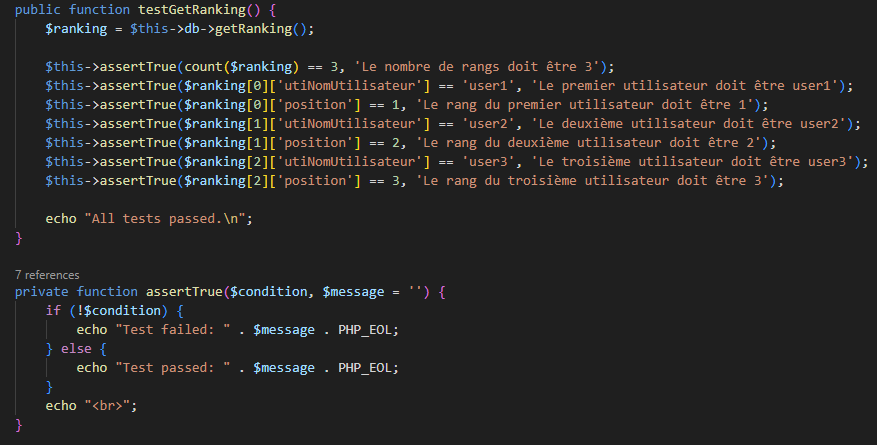


Figure 71 Test Unitaire: test getRanking

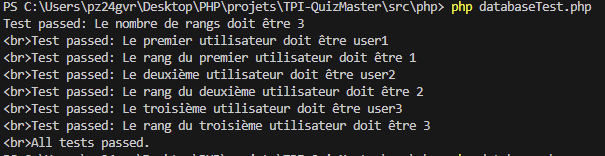


Figure 72 Test Unitaire : résultat

## Erreurs restantes

À ma connaissance, le code ne comporte aucune erreur. Des tests ponctuels ont été réalisés lors de la mise en place de chaque fonctionnalité, ce qui m’a permis de maximiser les chances de détecter et corriger tout bug ou erreur éventuel.

## Liste des documents fournis

* Cahier des charges – 01.05.2024
* Planification initiale – 03.05.2024
* Journal de travail – 03.06.2024
* Maquettes de l’application – 03.06.2024
* Rapport du projet – 03.06.2024
* Abstract/résumé du projet – 03.06.2024
* Normes de codages ETML V.3.5.0 – 03.06.2024
* Code source de l’application – 03.06.2024
* Manuel de l’application - 03.06.2024

# Conclusions

## Bilan des fonctionnalités demandées

Toutes les fonctionnalités demandées dans le cahier des charges ont été implémentées et fonctionnent correctement.

Cependant afin d’améliorer l’expérience utilisateur (administrateur) j’ai ajouté plusieurs fonctionnalités que je trouvais indispensable. Tel que la suppression de quizz et utilisateur, la modification d’information d’utilisateur et l’affichage de tous les utilisateurs. Ces fonctionnalités permettent grandement de faciliter la gestion du site et évite de devoir se perdre énormément de temps dans la base de données directement.

Si je devais améliorer cette application web, j’envisagerais de :

* + Ajouter des types de questions
    - Exemple : choix entre plusieurs questions prédéfinies, choix multiple, etc…
  + Modifications de quiz
    - Une fois qu’un quizz est créé il est actuellement impossible de le modifier, sauf si on le supprime et on le recrée

## Bilan personnel

Durant ce projet, j’ai pu réaliser à quoi pourrait ressembler un projet venant d’un client.

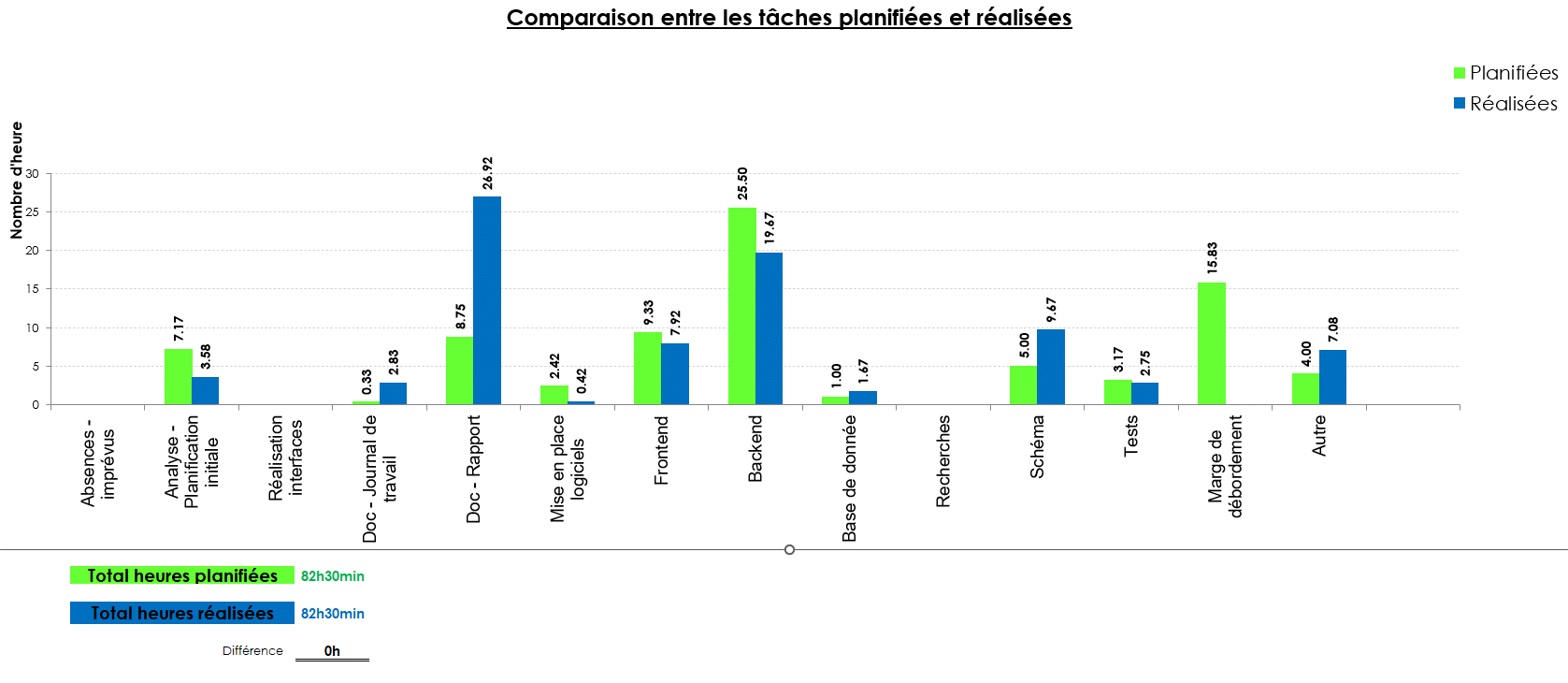
J’ai rencontré des difficultés avec la liaison à jQuery et le responsive Design au niveau de la navbar. Faire face à ce problème a été une chance car j’ai pu apprendre à correctement structurer mon code. Les liens entre les différentes pages rentraient en conflit ce qui empêchait la page contenant mon jQuery et bootstrap de fonctionner.

J’ai apprécié travailler sur ce projet, premièrement le développement web est le domaine que je préfère, cela a grandement continuer à me motiver tout du long. Ensuite ce projet m’a permis de m’améliorer dans la façon dont je démarre un projet web au niveau de la structuration du code mais également d’apprendre à implémenter des sauvegardes Git dans mes projets.

En dernier point, le point négatif qui me concerne, ce serait la tenue du journal de travail. J’ai eu du mal à noter mes activités pendant que je les faisait ce qui, en les remplissant donc toutes en fin de journées, cela m’a empêcher d’être précis. Cependant le projet m’a permis d’apprendre à le compléter de manière efficace et correcte.

## Bilan de la planification

Durant le projet, je pense avoir relativement bien suivi la planification. Si la planification devait être refaire, j’aurais planifié plus de temps pour la documentation.



# Annexes

## Résumé du projet QuizMaster

**Situation de départ**

Dans le cadre de ma formation d'informaticien à l'ETML visant l'obtention d'un CFC, j'ai développé une application web de gestion de quiz afin de valider les compétences acquises au cours de mes quatre années d'études. Ce projet consiste à créer une application web qui permet de consulter des quiz avec un système de classement en fonction des points obtenus. L’utilisateur pourra compléter un quizz et obtenir des points. Une interface administrateur permettra de gérer toutes les données.

**Mise en œuvre**

Pour mener à bien ce projet, j'ai adopté la méthodologie de travail dite « en cascade ». J'ai segmenté le projet en différentes étapes afin de le planifier de manière optimale : l'analyse du cahier des charges, la planification initiale, la conception de l'application web et des maquettes, le choix de MariaDB pour la base de données, la réalisation du MCD et du MLD, l'étude des besoins des utilisateurs, et enfin l'élaboration d'une stratégie de test. Pour le suivi de mon projet, j'ai utilisé un dépôt Git via l'application GitHub Desktop, me permettant de fournir aux experts des mises à jour régulières illustrant l'avancement de mon projet. J'ai également tenu un journal de travail à jour.

Pour finaliser, j'ai mené ce projet en utilisant Visual Studio Code connecté à GitHub pour la gestion des commits. Les tests ont été effectués directement dans mon navigateur web à l'aide de l'outil d'inspection d'éléments.

**Résultat**

L’application est entièrement opérationnelle ; toutes les fonctionnalités et les objectifs du cahier des charges ont été réalisés. Le design graphique de l’application web reste très basique et peut être amélioré. Pendant le développement, j'ai intégré un système de gestion des utilisateurs et des quiz, ajoutant ainsi une fonctionnalité significative.

Le répertoire Git :

<https://github.com/Medellin05001/TPI-QuizMaster>

## Webographie

* <https://www.figma.com/> - Site internet utilisé pour la réalisation des maquettes graphiques.
* <https://canva.com> – Site internet utilisé pour la réalisation de logo graphique
* <https://photopea.com> – Site internet utilisé pour la réalisation de retouche graphique des maquettes
* <https://stackoverflow.com/> - Forum orienté développement de programme et code
* <https://www.developpez.com/> - Forum orienté développement de programme et code
* <https://getbootstrap.com/> - Page officiel Bootstrap, contient les informations nécessaires à son utilisation
* <https://chatgpt.com> – Intelligence artificiel
* <https://validator.w3.org/> - Permet d’évaluer la structure du code html

## Table des illustrations

[Figure 1 Modèle : en cascade 12](#_Toc168296698)

[Figure 2 Maquette : Page d'accueil 13](#_Toc168296699)

[Figure 3 Maquette : Page de connexion 14](#_Toc168296700)

[Figure 4 Maquette : Page d'inscription 14](#_Toc168296701)

[Figure 5 Maquette : Page de Classement 15](#_Toc168296702)

[Figure 6 Maquette : Page de Création Quizz 15](#_Toc168296703)

[Figure 7 Maquette : Page de Quizz 16](#_Toc168296704)

[Figure 8 Maquette : Page Réponse de Quizz 16](#_Toc168296705)

[Figure 9 Modèle conceptuel des données 17](#_Toc168296706)

[Figure 10 Modèle logique des données 18](#_Toc168296707)

[Figure 11 Graphique : Bilan de la planification initiale 20](#_Toc168296708)

[Figure 12 Architecture : programme entier 26](file:///C:\Users\pz24gvr\Desktop\TPI_QuizzMaster_AlexandreFernandes\1.%20Documentation\1.2%20Rapport\R-TPI-AlexandreFernandes-Rapport-QuizMaster.docx#_Toc168296709)

[Figure 13 Maquette : Utilisation possibles 27](#_Toc168296710)

[Figure 14 Répertoire : Projet (1) 28](file:///C:\Users\pz24gvr\Desktop\TPI_QuizzMaster_AlexandreFernandes\1.%20Documentation\1.2%20Rapport\R-TPI-AlexandreFernandes-Rapport-QuizMaster.docx#_Toc168296711)

[Figure 15 Répertoire : Projet (2) 28](file:///C:\Users\pz24gvr\Desktop\TPI_QuizzMaster_AlexandreFernandes\1.%20Documentation\1.2%20Rapport\R-TPI-AlexandreFernandes-Rapport-QuizMaster.docx#_Toc168296712)

[Figure 16 Répertoire : Programme 29](file:///C:\Users\pz24gvr\Desktop\TPI_QuizzMaster_AlexandreFernandes\1.%20Documentation\1.2%20Rapport\R-TPI-AlexandreFernandes-Rapport-QuizMaster.docx#_Toc168296713)

[Figure 17 SQL : Requêtes préparées 30](#_Toc168296714)

[Figure 18 SQL : Connexion utilisateur 31](#_Toc168296715)

[Figure 19 SQL : Création d’utilisateur 31](#_Toc168296716)

[Figure 20 jQuery : Restriction Input connexion 32](#_Toc168296717)

[Figure 21 Ajax : Renvoie de page connexion en POST 32](#_Toc168296718)

[Figure 22 PHP : Traitement des données de connexion 33](#_Toc168296719)

[Figure 23 jQuery: Restriction Input inscription 33](#_Toc168296720)

[Figure 24 Ajax : Renvoie de page inscription en POST 34](#_Toc168296721)

[Figure 25 PHP : Traitement des données d’inscription 35](#_Toc168296722)

[Figure 26 PHP : déconnexion utilisateur 35](#_Toc168296723)

[Figure 27 SQL : obtenir tous les quiz 36](#_Toc168296724)

[Figure 28 Code : stocker tous les quiz 36](#_Toc168296725)

[Figure 29 PHP : Affichage de tous les quiz 36](#_Toc168296726)

[Figure 30 SQL : obtenir un quiz spécifique 37](#_Toc168296727)

[Figure 31 PHP : accès à la page d'un quiz spécifique 37](#_Toc168296728)

[Figure 32 PHP : stock les informations du quiz 37](#_Toc168296729)

[Figure 33 PHP : Affichage du quiz 37](#_Toc168296730)

[Figure 34 SQL : update le score de l’utilisateur 38](#_Toc168296731)

[Figure 35 Code : traitement des réponses d'un quiz 38](#_Toc168296732)

[Figure 36 PHP : stock résultat utilisateur 39](#_Toc168296733)

[Figure 37 PHP : Affichage réponses utilisateurs 39](#_Toc168296734)

[Figure 38 SQL : insertion t\_quizz 40](#_Toc168296735)

[Figure 39 SQL : insertion t\_questions 40](#_Toc168296736)

[Figure 40 SQL : insertion t\_reponses 40](#_Toc168296737)

[Figure 41 SQL : Obtenir dernier id créer dans la base de données 41](#_Toc168296738)

[Figure 42 PHP : Traitement données création d’un quiz 41](#_Toc168296739)

[Figure 43 PHP : Création d’un quizz 42](#_Toc168296740)

[Figure 44 SQL : suppression quizz 43](#_Toc168296741)

[Figure 45 PHP : renvoie vers suppression d'un quizz 43](#_Toc168296742)

[Figure 46 JS : message de confirmation de suppression 43](#_Toc168296743)

[Figure 47 PHP : Sécurité avant suppression quizz 44](#_Toc168296744)

[Figure 48 PHP : Suppression Quizz 44](#_Toc168296745)

[Figure 49 SQL : Obtenir le classement général 45](#_Toc168296746)

[Figure 50 PHP : Afficher les 7 premiers utilisateurs du classement 45](#_Toc168296747)

[Figure 51 PHP : Affichage des points/position de l'utilisateur connecté 46](#_Toc168296748)

[Figure 52 SQL : Obtenir tous les utilisateurs 47](#_Toc168296749)

[Figure 53 SQl : Modifier un utilisateur 47](#_Toc168296750)

[Figure 54 SQL : supprimer un utilisateur 47](#_Toc168296751)

[Figure 55 PHP : Récupérer les infos des utilisateurs 48](#_Toc168296752)

[Figure 56 PHP : Afficher les infos des utilisateurs 48](#_Toc168296753)

[Figure 57 PHP : Récupération des données de l'utilisateur à modifier 48](#_Toc168296754)

[Figure 58 PHP : Formulaire de modification d'utilisateur 48](#_Toc168296755)

[Figure 59 PHP : Modification des données de l'utilisateur 49](#_Toc168296756)

[Figure 60 JS : Confirmation de suppression d'utilisateur 49](#_Toc168296757)

[Figure 61 PHP : Récupération de l'id dans l'URL 49](#_Toc168296758)

[Figure 62 PHP : suppression de l'utilisateur 50](#_Toc168296759)

[Figure 63 PhpMyAdmin : Arborescence 51](#_Toc168296760)

[Figure 64 PhpMyAdmin : Table utilisateur 51](#_Toc168296761)

[Figure 65 PhpMyAdmin : Table quizz 51](#_Toc168296762)

[Figure 66 PhpMyAdmin : Table question 52](#_Toc168296763)

[Figure 67 PhpMyAdmin : Table réponse 52](#_Toc168296764)

[Figure 68 GitHub : Accès au commit 53](#_Toc168296765)

[Figure 69 GitHub : Choix du commit 53](#_Toc168296766)

[Figure 70 Test Unitaire : préparation test getRanking 58](#_Toc168296767)

[Figure 71 Test Unitaire: test getRanking 59](#_Toc168296768)

[Figure 72 Test Unitaire : résultat 59](#_Toc168296769)

[Figure 73 Manuel d'utilisation : inscription 87](file:///C:\Users\pz24gvr\Desktop\TPI_QuizzMaster_AlexandreFernandes\1.%20Documentation\1.2%20Rapport\R-TPI-AlexandreFernandes-Rapport-QuizMaster.docx#_Toc168296770)

[Figure 74 Manuel d'utilisation : connexion 87](file:///C:\Users\pz24gvr\Desktop\TPI_QuizzMaster_AlexandreFernandes\1.%20Documentation\1.2%20Rapport\R-TPI-AlexandreFernandes-Rapport-QuizMaster.docx#_Toc168296771)

[Figure 75 Manuel d'utilisation : header 88](#_Toc168296772)

[Figure 76 Manuel d'utilisation : liste des quiz 88](#_Toc168296773)

[Figure 77 Manuel d'utilisation : utilisation quiz 89](#_Toc168296774)

[Figure 78 Manuel d'utilisation : réponse quiz 89](#_Toc168296775)

[Figure 79 Manuel d'utilisation : création quiz 90](#_Toc168296776)

[Figure 80 Manuel d'utilisation : consulter liste des utilisateurs 90](#_Toc168296777)

[Figure 81 Manuel d'utilisation : Modification utilisateur 91](#_Toc168296778)

[Figure 82 Manuel d'utilisation : Classement général 91](#_Toc168296779)

## Dossier des tests

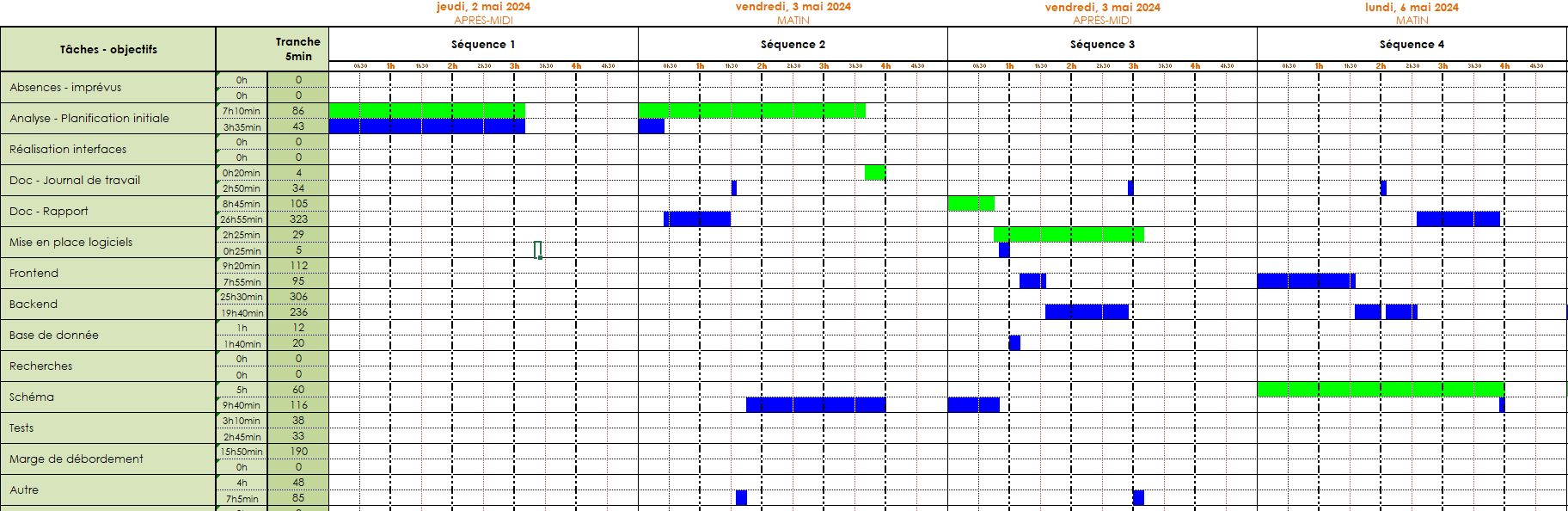
Dans le répertoire du projet, sous « 1. Documentation », vous pouvez accéder au sous-dossier intitulé « 1.4 Test\_Vidéos ». Ce dernier contient des tests effectués sous forme de vidéos courtes ou de captures d’écran de l’application.

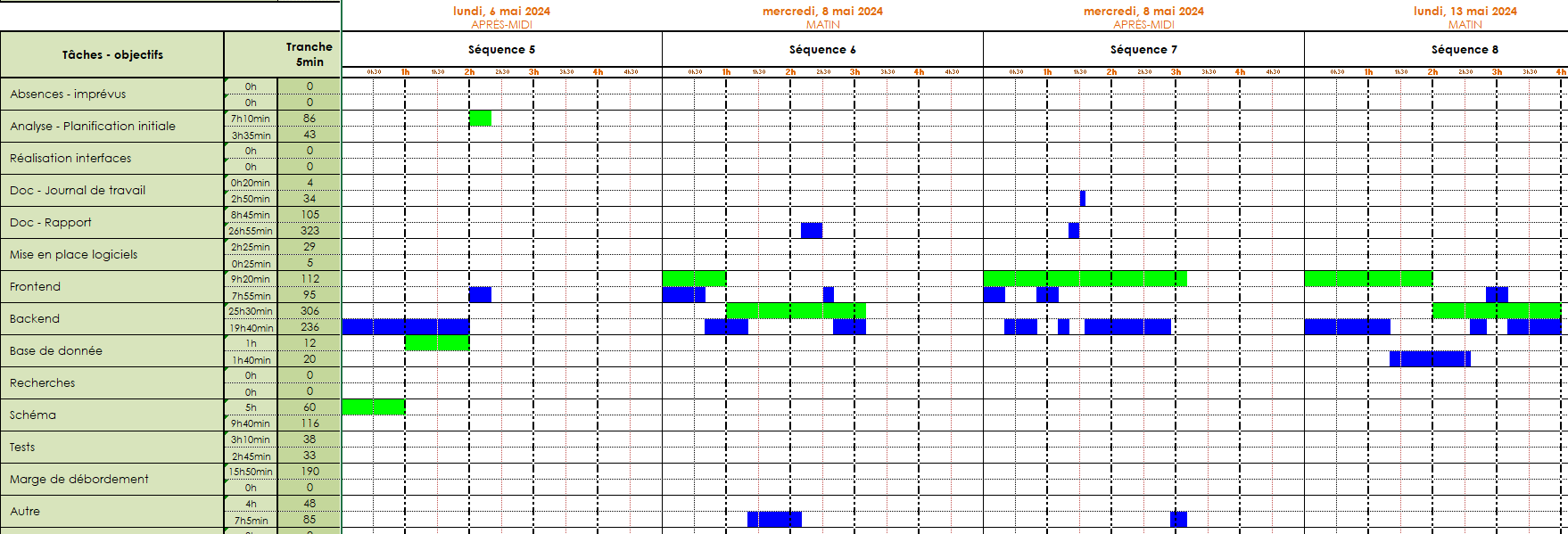
## Archives du projet

Toutes les archives du projet se retrouvent sur GitHub, plus précisément sur le dépôt suivant : <https://github.com/Medellin05001/TPI-QuizMaster>

## 

## Planification – Journal de travail





## 

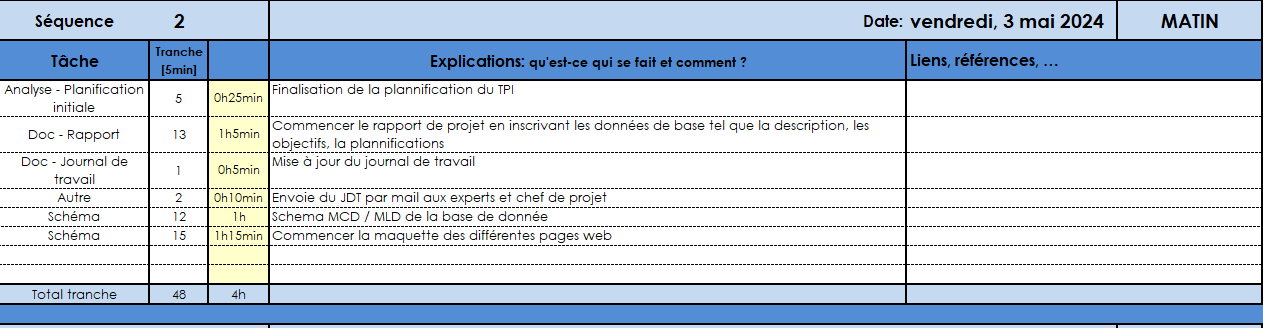
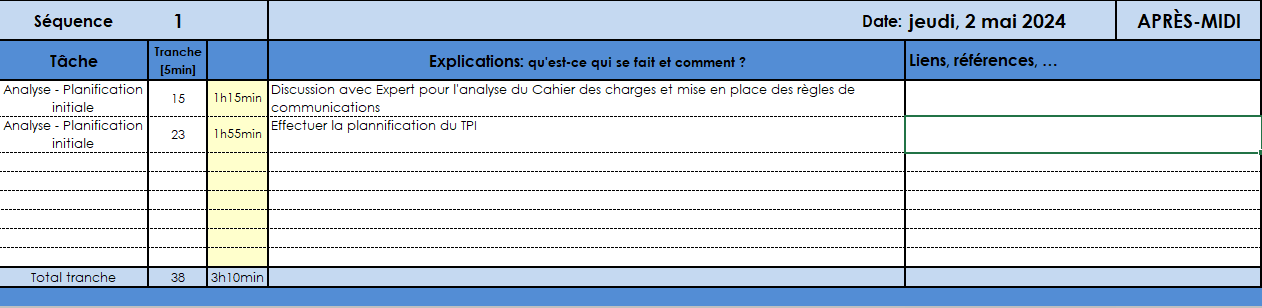
## 

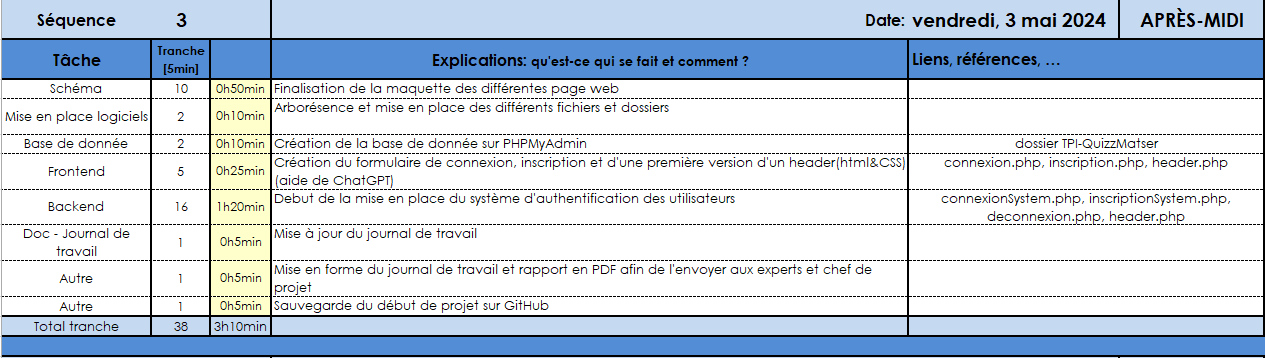
## 

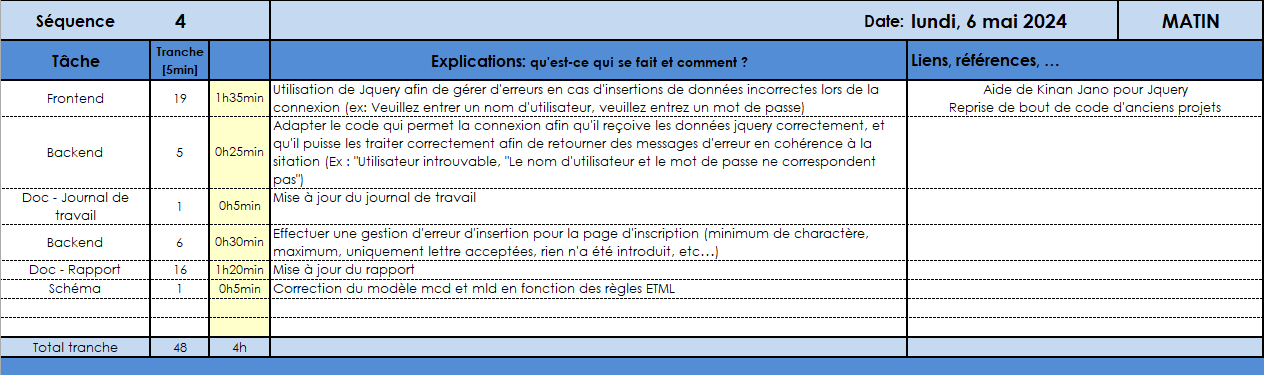
## 

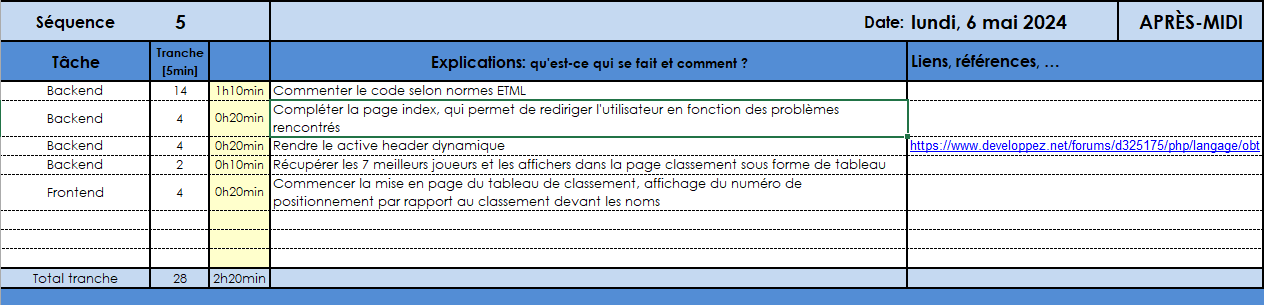
## 

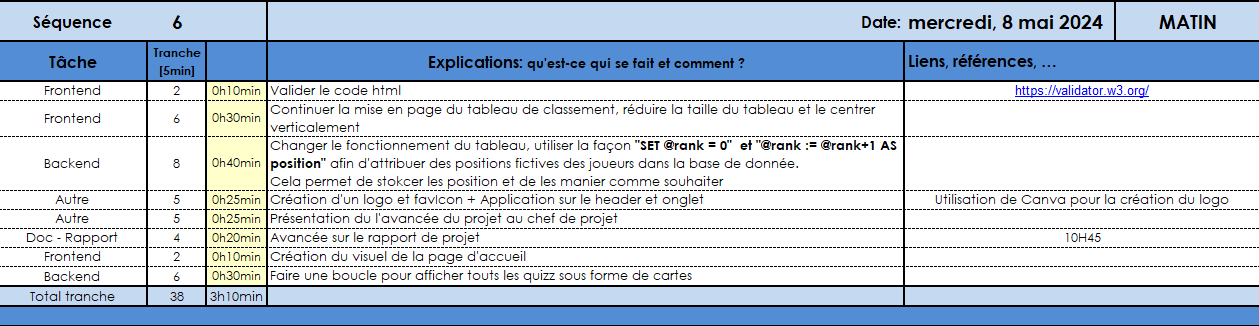
## Journal de travail



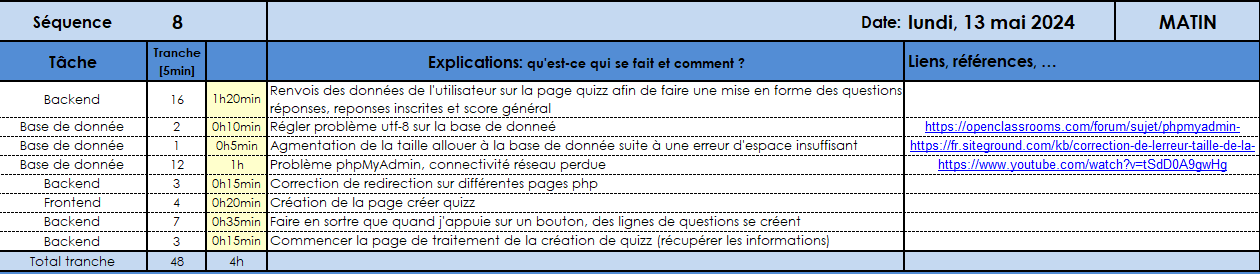


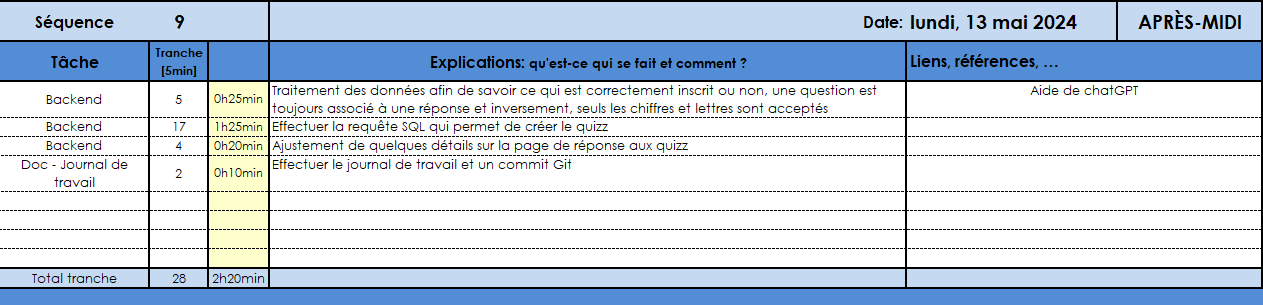


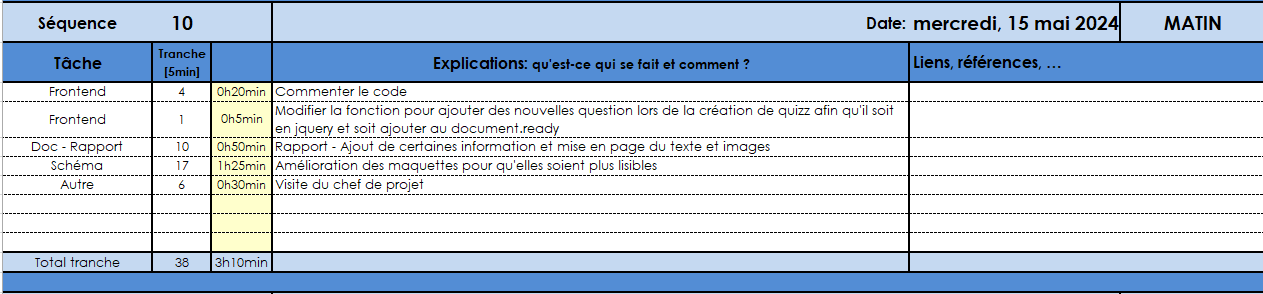




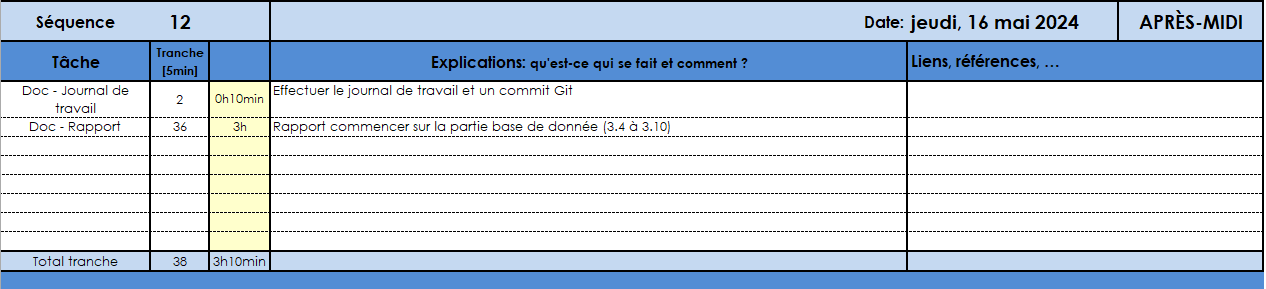




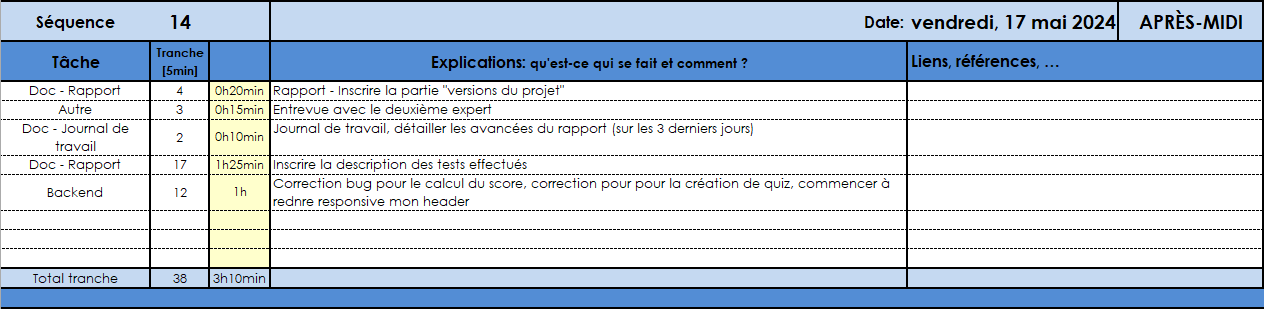




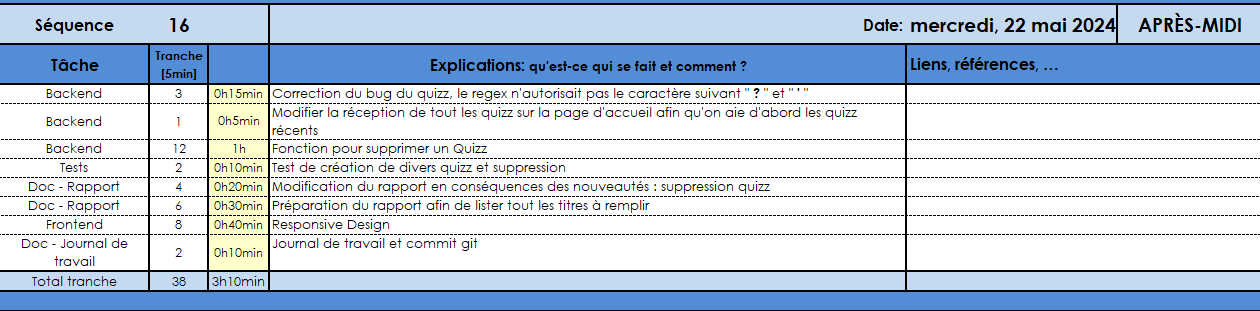
## 



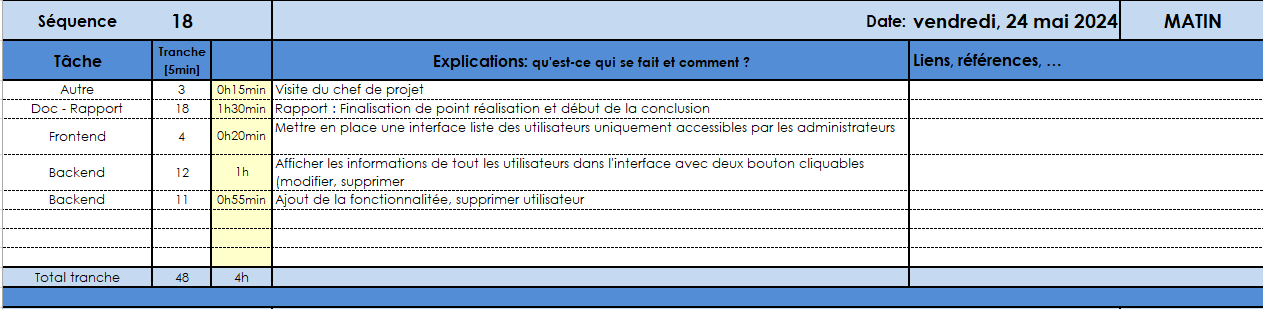
## 



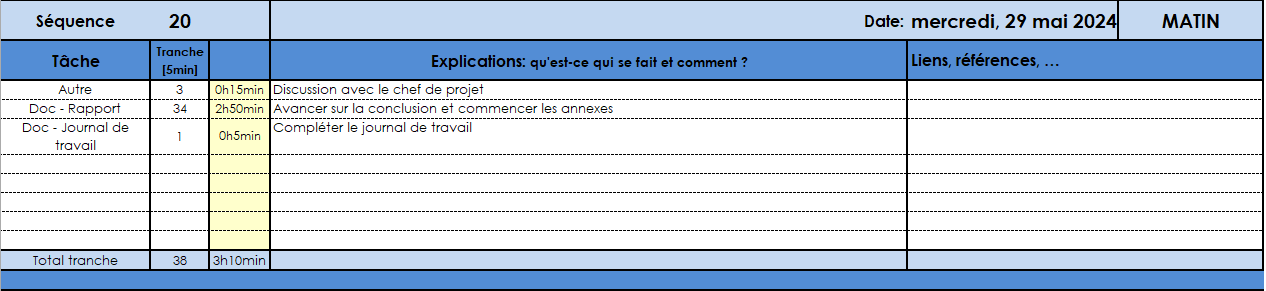
## 



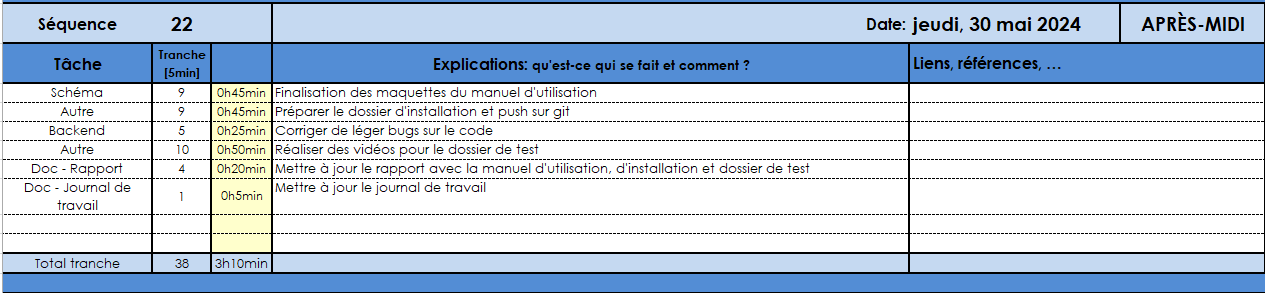
## 



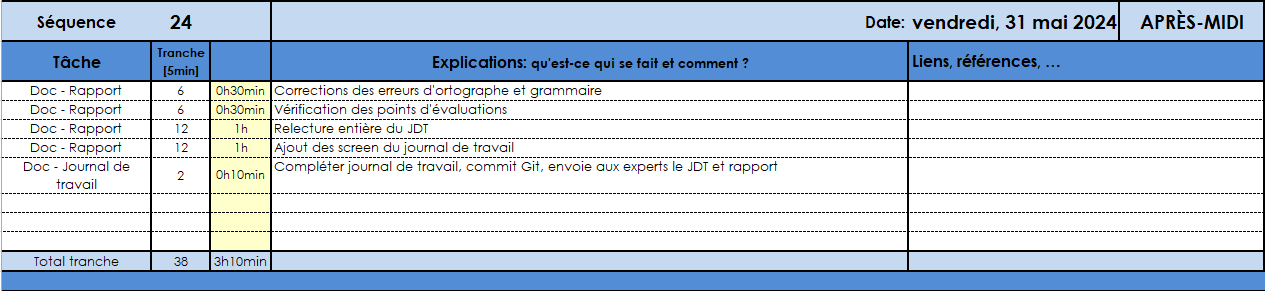
## 

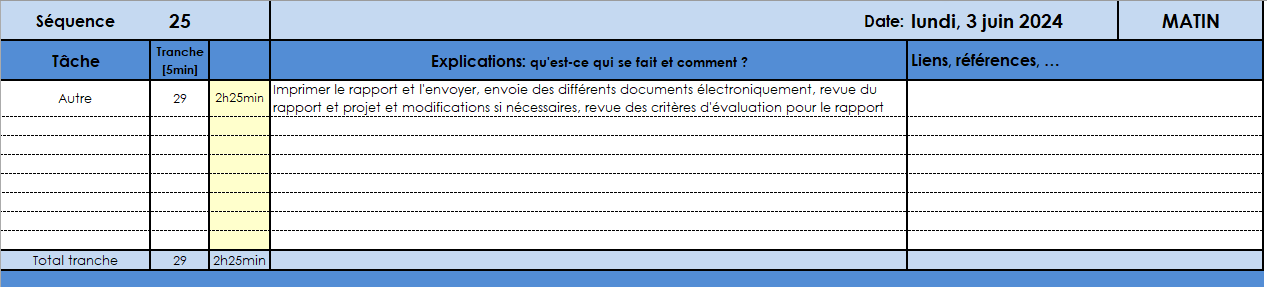


## 



## 





## Manuel d'Installation

Décompressez le dossier se trouvant dans « 3. Projet », ouvrez le fichier START pour démarrer les différents serveurs.

Pour importer les données, il suffit d’accéder à la page PhpmyAdmin (localhost :8080) (user : root, password : root)

Cliquez en haut à gauche sur « Nouvelle base de donnée » puis dans l’onglet importer choisissez le fichier .sql qui se trouve dans le dossier de code puis « PHP/Projets/TPI-QuizMaster/userContent »

L’adresse du projet d’après la configuration devrait être : « localhost :80 »

Login admin du projet :

* Utilisateur : Admin
* Mot de passe : Test1234

## Manuel d'Utilisation

### S’inscrire



Figure 73 Manuel d'utilisation : inscription

### Se connecter

Figure 74 Manuel d'utilisation : connexion

### Naviguer entre les différentes pages

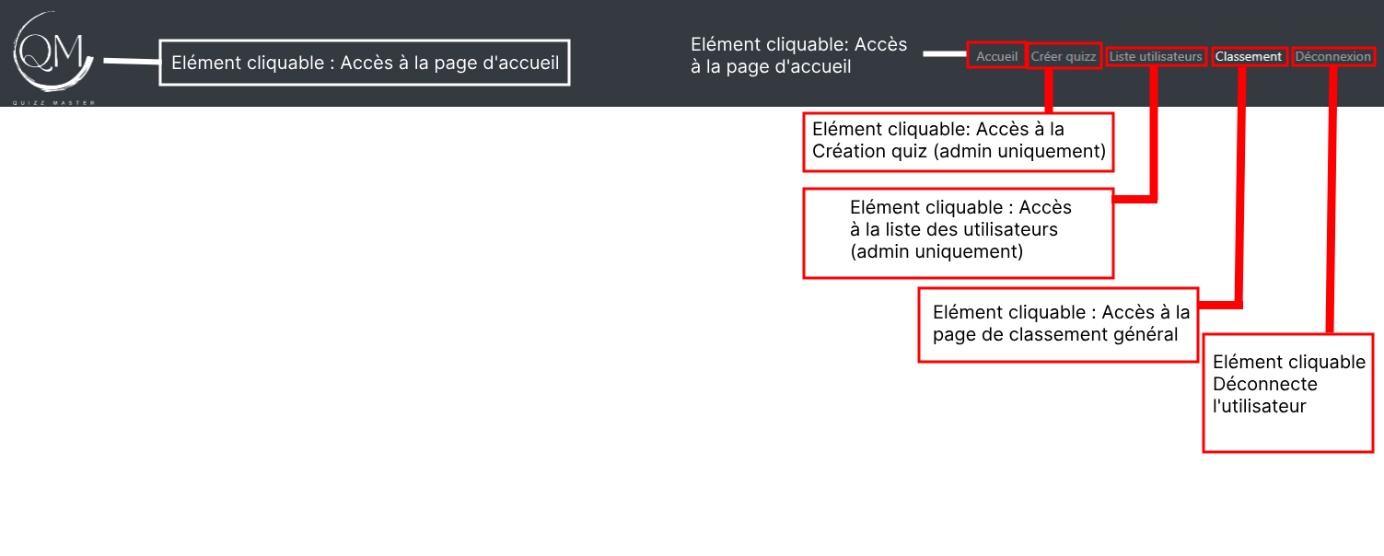


Figure 75 Manuel d'utilisation : header

### Utilisation de quiz

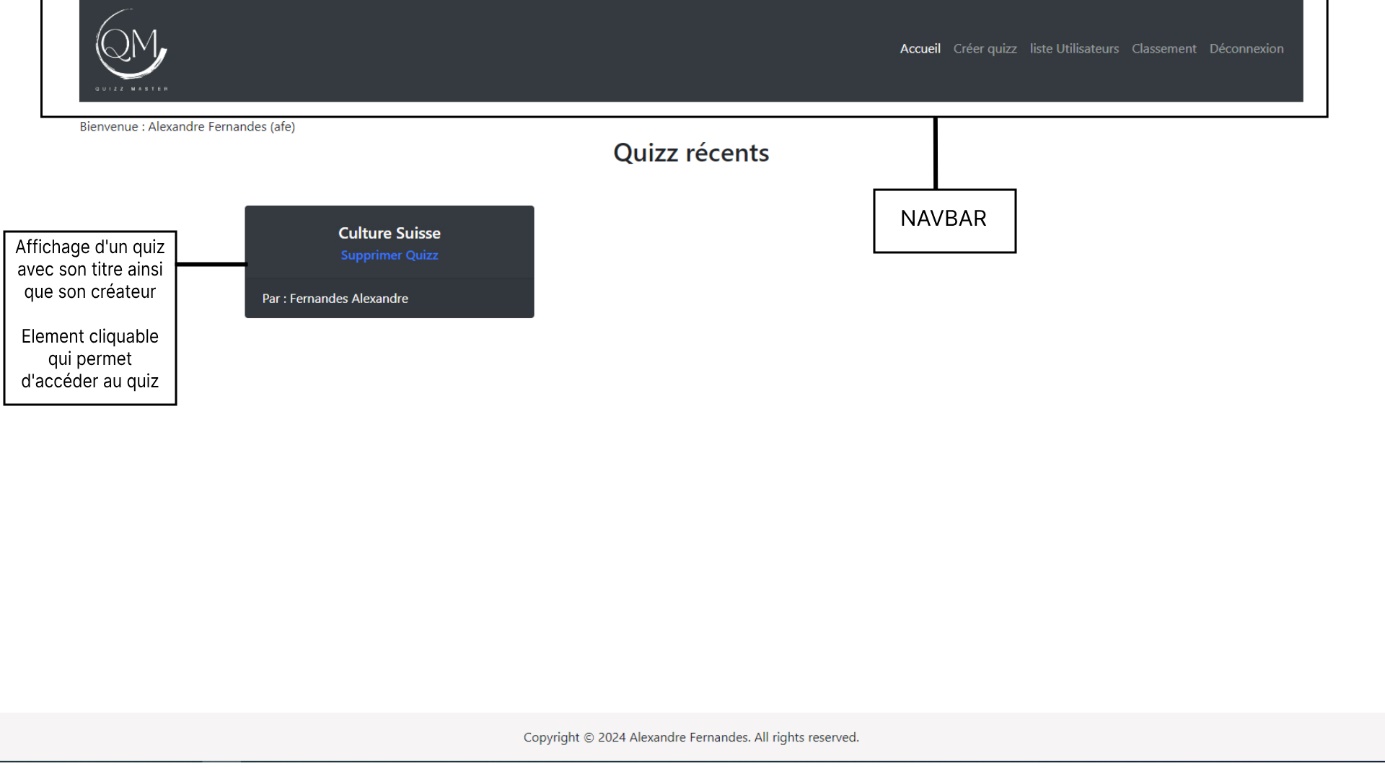


Figure 76 Manuel d'utilisation : liste des quiz

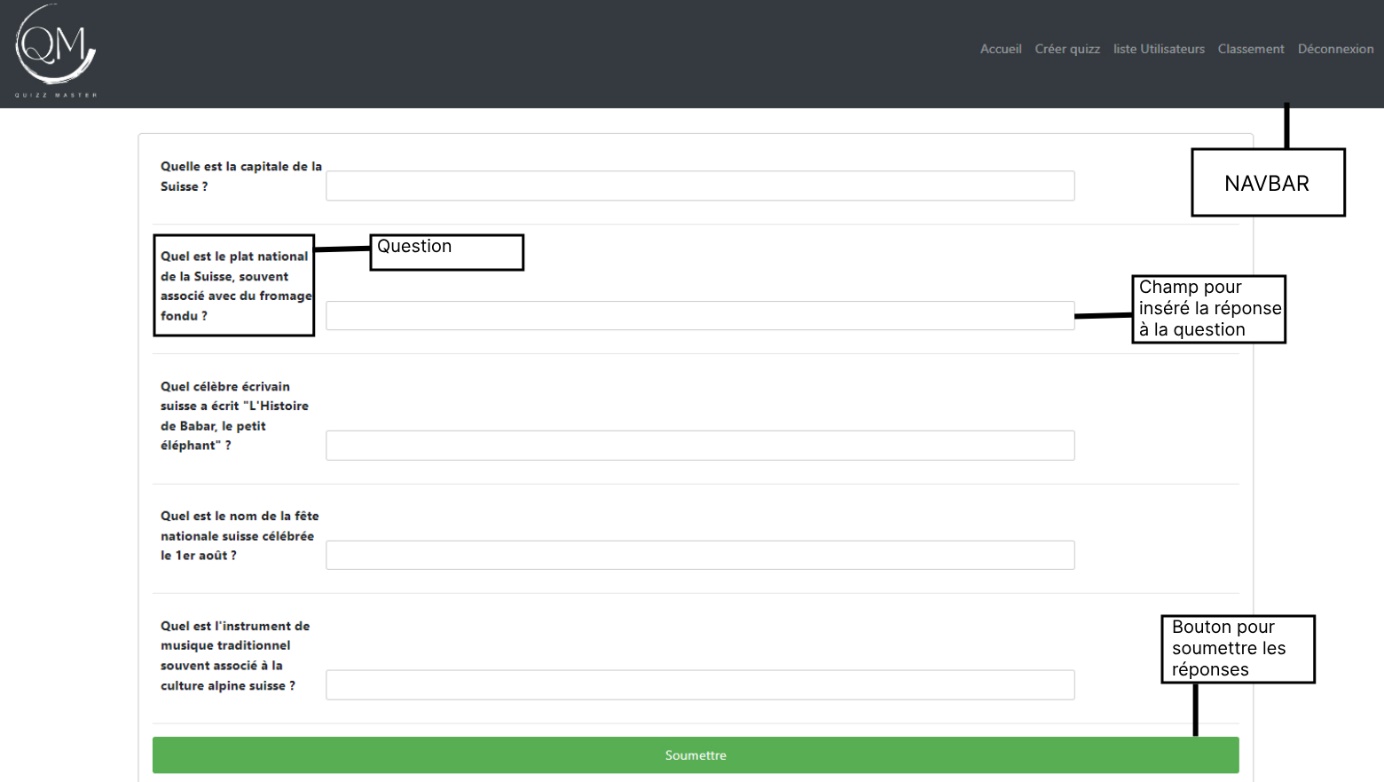


Figure 77 Manuel d'utilisation : utilisation quiz

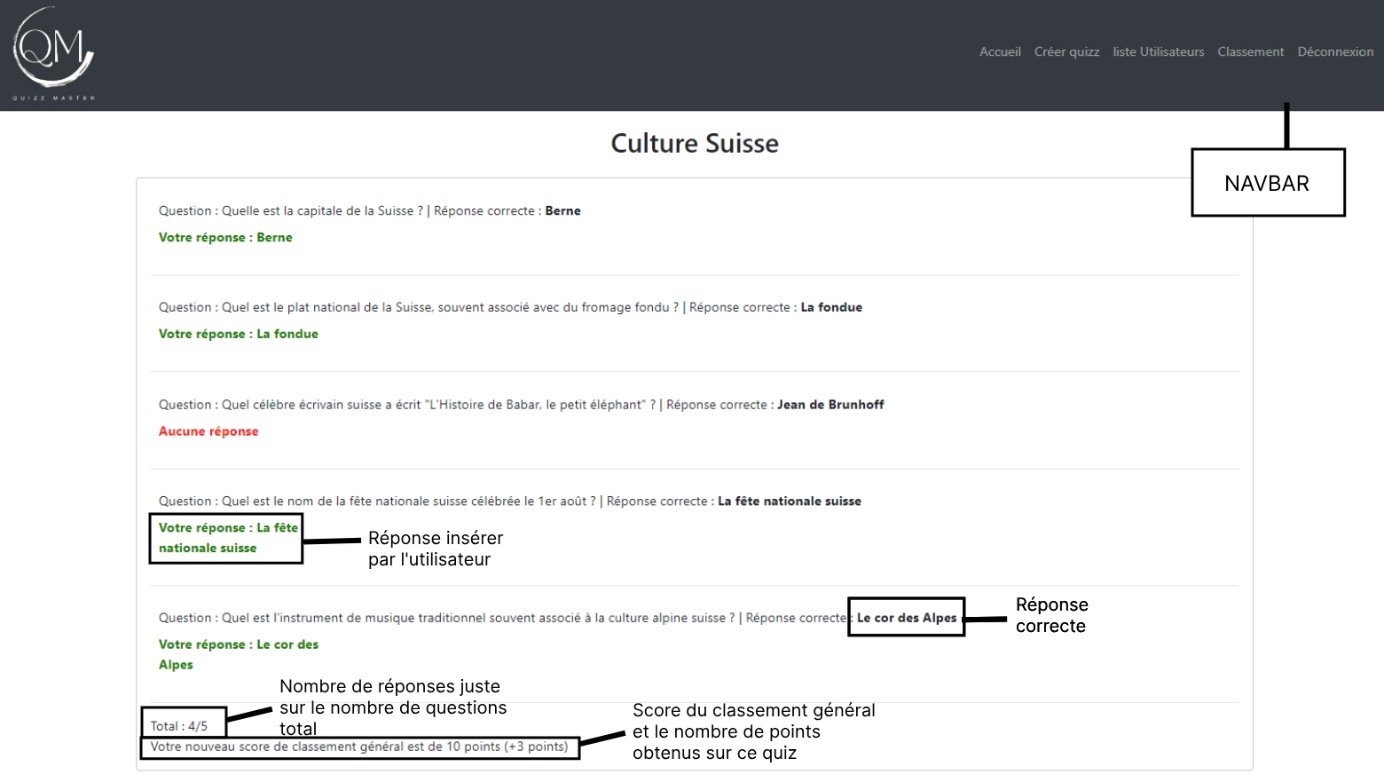


Figure 78 Manuel d'utilisation : réponse quiz

### Création d’un quizz (administrateur)

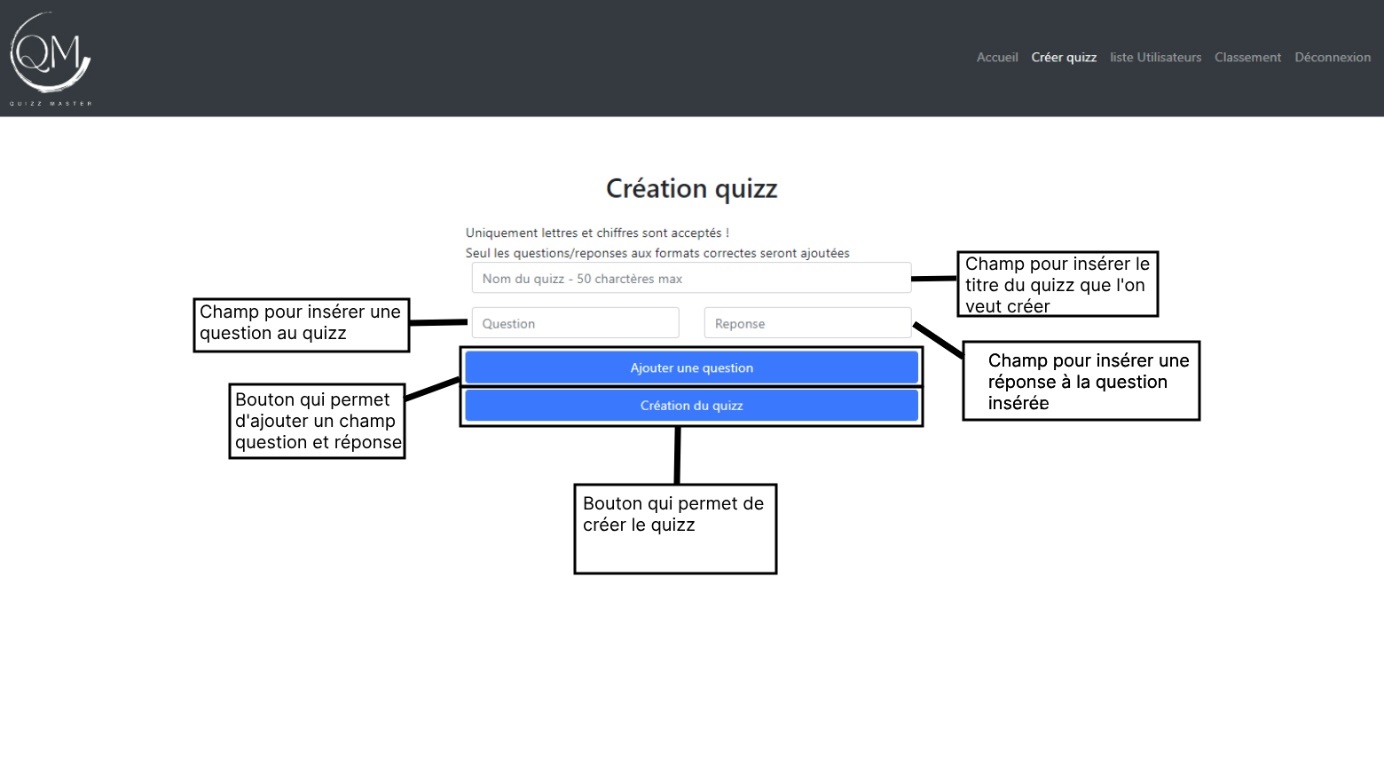


Figure 79 Manuel d'utilisation : création quiz

### Gestion des utilisateurs (administrateur)

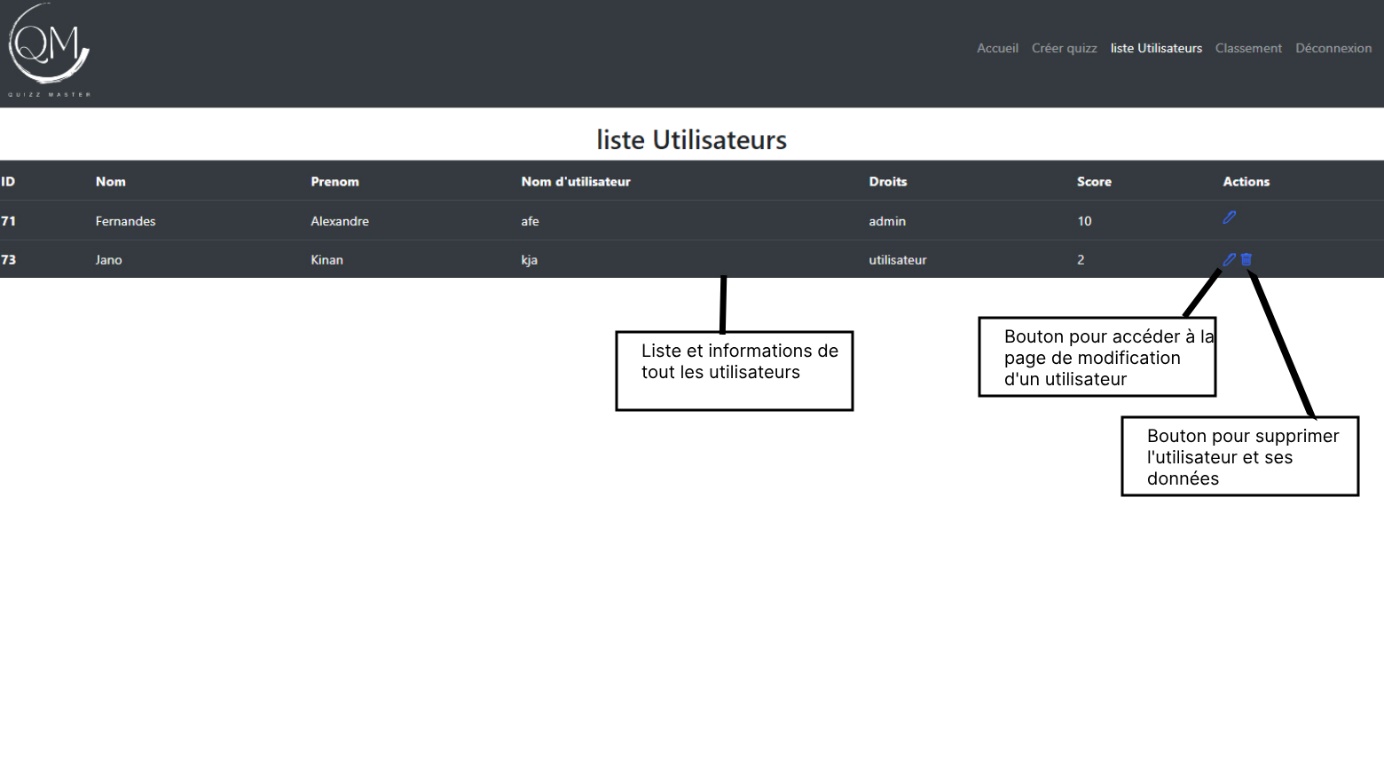


Figure 80 Manuel d'utilisation : consulter liste des utilisateurs

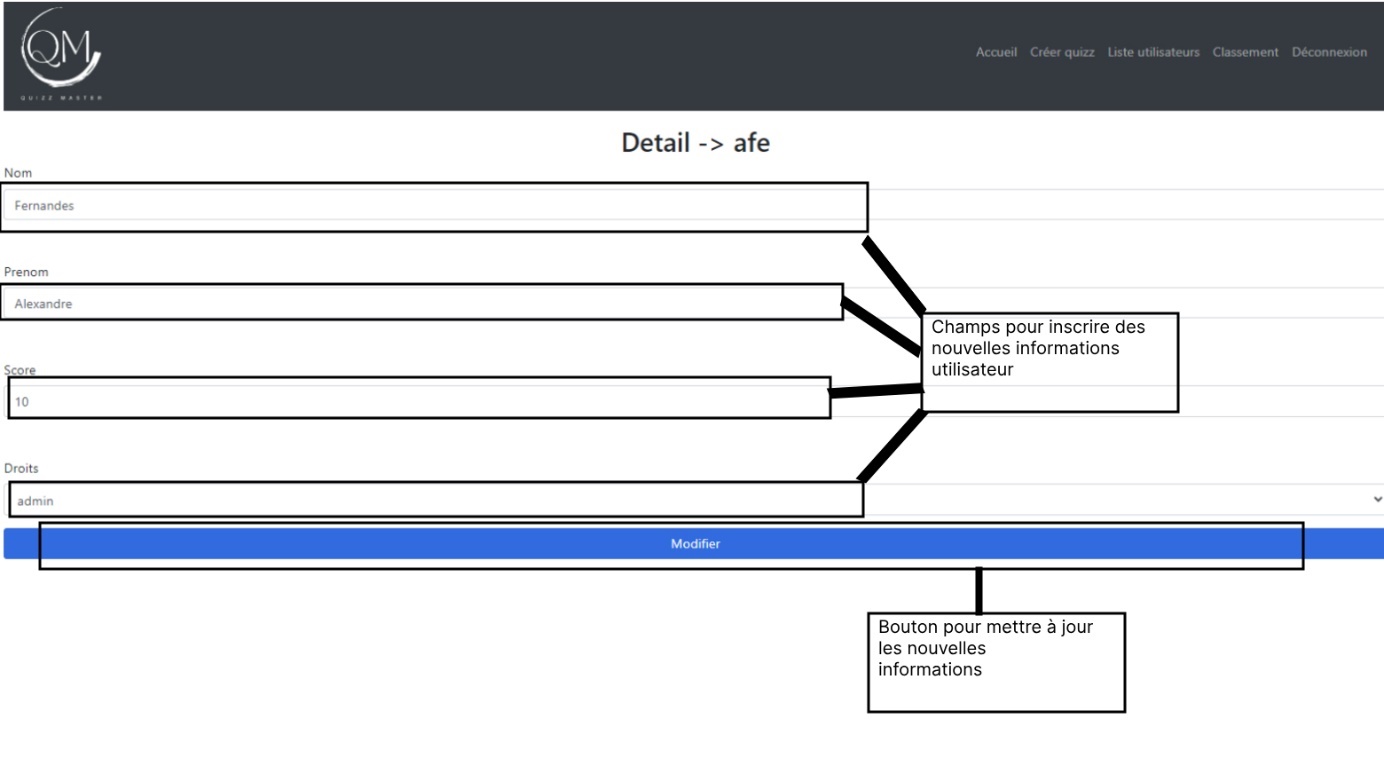


Figure 81 Manuel d'utilisation : Modification utilisateur

### Consulter le classement général et son score personnel

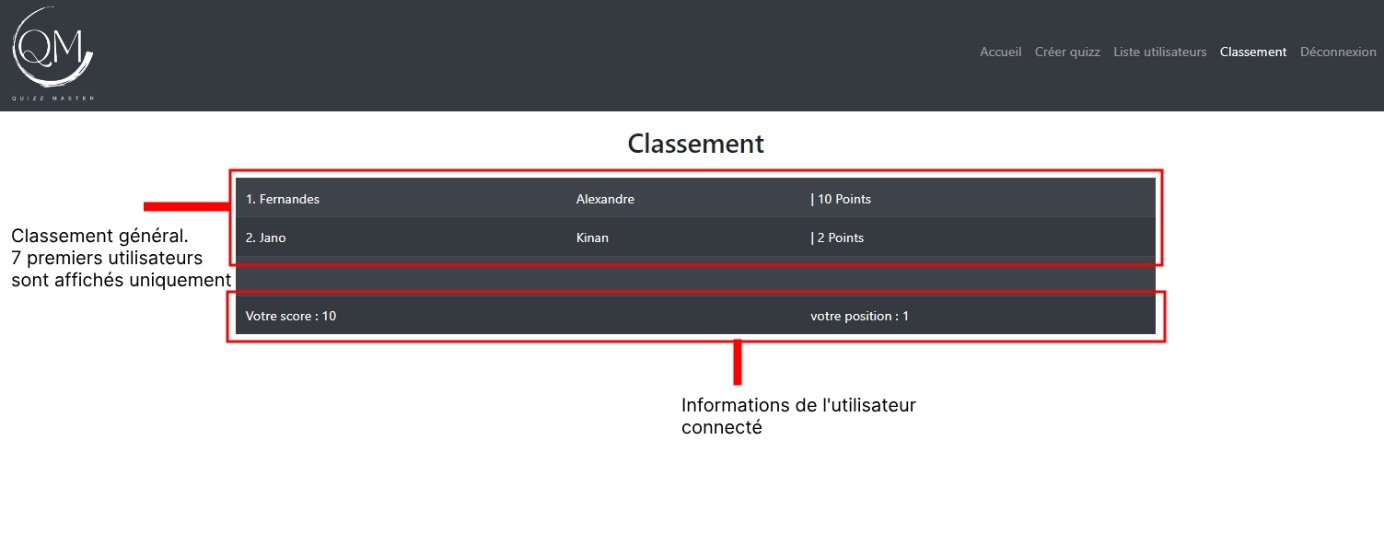


Figure 82 Manuel d'utilisation : Classement général

1. <https://kahoot.com/> [↑](#footnote-ref-1)
2. <https://github.com/Medellin05001/TPI-QuizMaster> [↑](#footnote-ref-2)