



DOCUMENTATION TECHNIQUE

Billy Nguyen – I.FA-P3B

CFPT - Travail Pratique Individuel

Table des matières

[1. Gestion des versions 3](#_Toc516817225)

[2. Introduction 3](#_Toc516817226)

[2.1 Qu’est-ce Blindtest ? 3](#_Toc516817227)

[2.2 Pourquoi le développement Web ? 3](#_Toc516817228)

[3. Cahier des charges 3](#_Toc516817229)

[3.1. Titre du projet : 3](#_Toc516817230)

[3.2. Objectifs du projet : 3](#_Toc516817231)

[3.3. Description détaillée 4](#_Toc516817232)

[3.3.1. Environnement et remarques générales 4](#_Toc516817233)

[3.3.2. Définition du contenu et des fonctionnalités 4](#_Toc516817234)

[3.3.3. Gestion du site 4](#_Toc516817235)

[4. Analyse de l’existant 4](#_Toc516817236)

[4.1 Pururin.eu 4](#_Toc516817237)

[4.1.1. Avantage : 5](#_Toc516817238)

[4.1.2. Désavantages : 5](#_Toc516817239)

[5. Analyse fonctionnelle 5](#_Toc516817240)

[5.1 Description générale 5](#_Toc516817241)

[5.2 Accueil 5](#_Toc516817242)

[5.3 Inscription 5](#_Toc516817243)

[5.4 Profil 6](#_Toc516817244)

[5.5 Paramètres 6](#_Toc516817245)

[5.6 Jouer 7](#_Toc516817246)

[5.7 Liste des musiques (admin) 8](#_Toc516817247)

[6. Analyse organique 9](#_Toc516817248)

[6.1 Modèle Conceptuel de Données 9](#_Toc516817249)

[6.1.1 Utilisateur 10](#_Toc516817250)

[6.1.2 Paramètres 10](#_Toc516817251)

[6.1.3 Score 10](#_Toc516817252)

[6.1.4 Musique 10](#_Toc516817253)

[6.1.5 Partie 10](#_Toc516817254)

[6.2 Base de données 10](#_Toc516817255)

[6.3 Classe PHP Data Objects (PDO) 10](#_Toc516817256)

[6.4 Création / Ajout 11](#_Toc516817257)

[6.4.1 Function Create\_user ($name, $nickname, $email, $pwd, $profilepic) 11](#_Toc516817258)

[6.4.2 Function Add\_music ($music\_title, $music\_author, $music\_style\_id) 11](#_Toc516817259)

[6.4.3 Function Add\_file\_cover($music\_id, $music\_file, $music\_cover) 11](#_Toc516817260)

[6.4.4 Function Save\_parameters($question\_time, $questions\_number, $user\_id, $music\_style\_id) 11](#_Toc516817261)

[6.4.5 Function Add\_score($score, $score\_question, $user\_id) 11](#_Toc516817262)

[6.4.6 Function Create\_game() 11](#_Toc516817263)

[6.4.7 Function Add\_game($game\_id, $user\_id, $music\_id) 11](#_Toc516817264)

[6.5 Récupération des données 12](#_Toc516817265)

[6.5.1 Function Check\_music($music\_title) 12](#_Toc516817266)

[6.5.2 Function Get\_all\_music\_random($game\_id, $music\_style\_id) 12](#_Toc516817267)

[6.6 Suppression 12](#_Toc516817268)

[6.6.1 Function Delete\_music($music\_id, $music\_cover, $music\_file) 12](#_Toc516817269)

[6.7 Javascript 12](#_Toc516817270)

[6.8 Site web 12](#_Toc516817271)

[6.9 Navigation 12](#_Toc516817272)

[7. Installation 13](#_Toc516817273)

[8. Amélioration possible 13](#_Toc516817274)

[9. Rapport de tests 13](#_Toc516817275)

[10. Conclusion 16](#_Toc516817276)

[10.1 Objectif 17](#_Toc516817277)

[10.2 Problèmes rencontrés 17](#_Toc516817278)

[11. Annexes 17](#_Toc516817279)

[11.1 Bibliographie 17](#_Toc516817280)

[11.1.1 PHP 17](#_Toc516817281)

[11.1.2 SQL 17](#_Toc516817282)

[11.1.3 Bootstrap 17](#_Toc516817283)

[11.1.4 Wamp 17](#_Toc516817284)

[11.1.5 W3School 18](#_Toc516817285)

[11.1.6 Stackoverflow 18](#_Toc516817286)

[11.2 Aides reçues 18](#_Toc516817287)

[11.3 Remerciement 18](#_Toc516817288)

[12. Planning 19](#_Toc516817289)

[12.1 Planning théorique 19](#_Toc516817290)

[12.2 Planning effectif 19](#_Toc516817291)

[13. Table des illustrations 17](#_Toc516817292)

# Gestion des versions

|  |  |  |
| --- | --- | --- |
| N° version | Nouveauté | Date |
| 0.1 | Création des différentes sections. | 06.06.18 |
| 0.2 | Ajout du cahier des charges et l’analyse de l’existant. | 07.06.18 |
| 0.3 | Ajout de l’analyse fonctionnelle. | 11.06.18 |
| 0.4 | Ajout partiel de l’analyse organique. | 12.06.18 |
| 0.5 | Ajout de la description des fonctions dans l’analyse organique. | 13.06.18 |
| 0.6 | Ajout de l’amélioration et conclusion. | 14.06.18 |
| 0.7 | Ajout d’explications concernant la base de données | 15.06.18 |

# Introduction

Dans le cadre de la formation professionnelle, il m’est demandé de réaliser un travail pratique individuel (TPI). Ce travail permet de montrer tout notre potentiel acquis au fil de l’apprentissage.

Au début, j’avais pour idée de faire un site de référencement d’articles liés à la technologie. Par exemple un site comme [frandroid.com](http://www.frandroid.com/). J’ai donc proposé mon idée à ma responsable de TPI et malheureusement, ce n’était pas possible. Elle m’a donc donné un autre projet que voici : Blindtest.

## Qu’est-ce Blindtest ?

C’est tout simplement un site où chaque utilisateur pourra s’il le souhaite, tester ses connaissances en matière de musique. Ledit utilisateur écoute un morceau et durant le temps imparti pour chaque question, devra trouver la bonne réponse.

## Pourquoi le développement Web ?

Au début de la formation, quand on avait eu le module 101 qui était le développement web HTML / CSS, j’ai beaucoup aimé. De ce fait, en 2e année, on a commencé à apprendre le PHP, cependant, à ce moment-là, j’ai eu une grosse perte de motivation. J’avais donc du mal à comprendre ce langage, mais je n’étais pas le seul dans ce cas, une partie de ma classe aussi. Puis c’est finalement vers cette fin d’année que j’ai compris le PHP et j’ai décidé donc de faire un projet web et non C#. J’ai beaucoup de mal avec le C# et ce mal s’est empiré en dernière année, ce n’était donc pas une bonne idée pour moi de faire mon TPI en C#.

# Cahier des charges

## Titre du projet :

Blindtest

## Objectifs du projet :

* L’objectif du projet est de développer un quizz sur de la musique en général. Un utilisateur pourra tester ses connaissances soit sur une chanson soit sur une image du chanteur / groupe.
* Le projet utilisera une base de données liée à un site web, avec le nom des chanteurs, une description, des images ainsi que les musiques ainsi qu’une catégorie de musique.

## Description détaillée

### Environnement et remarques générales

* Le site web sera développé en PHP / HTML5 / CSS3 / MySQL et utilisera une base de données (dont le modèle est fourni)
* Le design n’est pas une priorité, il devra être conçu de façon simple et pouvant être modifié rapidement. Il peut être fait à l’avance, mais le mentionner dans le journal de bord.
* Le site sera conçu à l’aide de fonctions pour simplifier la maintenance et son évolution.

### Définition du contenu et des fonctionnalités

*Identification et compte*

2 types de comptes : utilisateur et administrateur.

1. Utilisateur
   * L’utilisateur devra s’inscrire via le formulaire ad hoc (voir mind map suivant), il aura le statut 0 par défaut. Il sera obligé d’être inscrit pour pouvoir jouer au quizz.
   * Il aura la possibilité de modifier ses données et d’afficher ses scores et de tester ses connaissances.
2. L’administrateur
   * L’administrateur aura le statut à 1, il aura tous les droits.
   * L’administrateur aura la possibilité d’uploader des images et des sons, qui seront stockés dans un dossier image et son et le nom de l’image et du son seront enregistré dans la base.
   * Il aura la possibilité d’ajouter, de modifier et supprimer des images et des musiques (la suppression et la modification s’effectue étalement dans le dossier).

### Gestion du site

*Utilisateur inscrit  
Administrateur*

* L’administrateur aura la possibilité de gérer l’ensemble de la base (CRUD).

*Public*

* L’utilisateur lambda aura la possibilité de s’inscrire pour jouer.

# Analyse de l’existant

## [Pururin.eu](http://www.pururin.eu/)

Mon TPI ressemble beaucoup à une version très simplifiée de Pururin. Cependant, la conception de mon projet n’est basée sur aucun des sites de blindtest disponibles sur internet.

Ce site est un blindtest bien plus complet basé sur de la musique de jeu et dessins anime japonais. La différence entre ce site et mon projet est que le jeu est dynamique et qu’il est jouable en multijoueur.

Le site trie tous ses morceaux en plusieurs catégories où il est possible de chercher dans une page.

Les utilisateurs ont la possibilité de proposer un morceau que les admins valideront si le morceau est conforme.

Pururin propose bien plus de fonctions que Blindtest et est bien plus complet. Il est possible de voir le score de tous les joueurs. Le site propose différents filtres dans la page profil, par exemple en quelle langue l’utilisateur souhaite écouter la musique et jouer.

### Avantage :

* Multijoueur.
* Possibilité de proposer des morceaux.
* Un système de filtre plus complet.
* Des paramètres plus précis dans le profil.
* Affiche les personnes connectées.
* Possibilité de commenter.

### Désavantages :

* Obligation de jouer avec des joueurs.
* Site lourd en javascript : les ordinateurs plantent assez facilement s’ils sont faibles / anciens.
* Site pas très intuitif.

# Analyse fonctionnelle

## Description générale

L’application est un blindtest dont le but est d’écouter un morceau et de deviner de quel morceau il s’agit.

## Accueil

Sur la page d’accueil, on a la possibilité de se connecter ou de se créer un compte. Le lien se trouve sur la barre de navigation. A la connexion, l’utilisateur est envoyé à la page de profil.

|  |
| --- |
| Figure - Accueil |

## Inscription

La page d’inscription permet de se créer un compte utilisateur. L’utilisateur a la possibilité de choisir ou non une image de profil.

|  |
| --- |
| Figure - Page d’inscription |

## Profil

Dans la page profil se trouve le nom, le pseudo, le score et l’image de profil de l’utilisateur. Il a la possibilité de changer son image de profil.

|  |  |
| --- | --- |
| |  | | --- | | Figure - Page de profil | |

## Paramètres

L’utilisateur a la possibilité de choisir ses paramètres tels que le temps (en seconde) de chaque question, le nombre de questions et le style de musique.

|  |
| --- |
| Figure - Page des paramètres |

## Jouer

Sur la page jouer est affiché les paramètres que l’utilisateur a au préalable choisis. S’il ne l’a pas fait, les paramètres par défaut seront utilisés. Il y a un résumé des paramètres que l’utilisateur a choisi et si ses paramètres lui conviennent, il peut appuyer sur « commencer ».

Il verra ensuite le nombre de question qu’il lui reste à faire, les 4 propositions (chacune d’entre elles est une musique), une barre qui permet de régler le volume, le temps qu’il lui reste avant que le compte à rebours atteigne 0 et le bouton « valider.

S’il choisit une bonne réponse, il gagnera un point, sinon, la question passe à la suivante.

Quand il aura répondu à toutes les questions. Le jeu affichera une page de résultats lui indiquant son nombre de point(s) sur le nombre de questions choisi.

|  |
| --- |
| Figure - Page de début de jeu |
| Figure - En jeu |

## Liste des musiques (admin)

Sur cette page s’affiche l’intégralité des musiques de la base de données. L’admin a plusieurs possibilités sur cette page :

* Modifier les chansons. Les données prises en compte dans la modification sont : Le titre, l’auteur, la chanson et la pochette d’album.
* Supprimer les chansons.
* Ajouter des chansons. Il faudra un titre, un auteur, un morceau et une pochette d’album.

|  |
| --- |
| Figure - Liste des musiques |
| Figure - Ajout / modification des musiques |
|  |
| Figure - Ajout d'un style de musique |

# Analyse organique

## Modèle Conceptuel de Données

|  |
| --- |
| Figure - MCD |

La base de données est en 5 tables. La table « users » est la plus importante, étant donné que quasiment toute l’application passe par celle-ci. Une fois le compte créé, l’utilisateur doit configurer ses paramètres et à chaque fois qu’il joue, un score lui sera attribué. Les musiques sont indépendantes de l’utilisateur. La table « game » sert à déterminer quel joueur est en train de jouer et s’il a répondu juste à une musique. La table sert aussi à avoir des statistiques comme par exemple le nombre de parties que l’utilisateur a fait.

La base de données est différentes du schéma initial. Pour simplifier le code et les différentes requêtes, j’ai changé la relation entre la table « music » et « music\_style ». De plus, la table « game » a été ajoutée pour pouvoir faire un lien entre la musique jouée et l’utilisateur durant une partie. Cette table sert aussi à faire des statistiques comme par exemple le nombre de partie que l’utilisateur a fait.

### Utilisateur

Cette table contient le nom, le pseudo, l’email et le mot de passe de l’utilisateur. L’image de profil est optionnelle, donc sa valeur par défaut est donc « null ». Quant au statut de l’utilisateur, il est par défaut à 0, c’est-à-dire utilisateur normal. 1 voudrait dire admin.

### Paramètres

Cette table enregistre le temps pour chaque question, le nombre de questions et le style de musique.

### Score

Cette table enregistre le score de l’utilisateur au moment où il a fini et le nombre de questions pour pouvoir savoir s’il a besoin de s’améliorer ou non. Par ex : « 3 points sur 5 questions. » L’utilisateur a donc répondu juste à 3 questions sur 5, ce qui correct.

### Musique

Cette table permet de stocker les musiques. On a le titre du morceau, son auteur, le nom du fichier et le nom de la pochette d’album. Ces 2 derniers champs enregistrent les fichiers pour pouvoir les utiliser plus tard.

### Partie

Cette table stocke les réponses justes de l’utilisateur durant une partie, elle sert à exclure les musiques qui ont été répondues juste. game\_id n’est pas en auto incrément.

## Base de données

La base de données utilisée est MySQL version 5.7.21 intégrée avec wamp. L’accès à la base de données dans l’application se fait grâce à une fonction nommée PDO.

## Classe PHP Data Objects (PDO)

|  |
| --- |
| Figure - PDO |

Une variable $dbc est créée et elle sert d’intermédiaire avec la base de données. On lui donne une valeur qui comprend 4 variables globales :

* DB\_HOST : représente l’adresse à laquelle la base se trouve. Localhost avec wamp.
* DB\_NAME : représente le nom de la base de données.
* DB\_USER : représente le nom de l’utilisateur de la base de données.
* DB\_PASSWORD : représente le mot de passe de l’utilisateur de la base de données.

S’il y a une erreur à la connexion, la fonction nous l’indiquera. Toute requête SQL passera par cette fonction.

## Création / Ajout

### Function Create\_user ($name, $nickname, $email, $pwd, $profilepic)

Cette fonction permet de créer un utilisateur grâce à son nom, pseudo, email et mot de passe. L’image de profil est optionnelle.

|  |
| --- |
| INSERT INTO `users`(`user\_name`, `user\_nickname`, `user\_email`, `user\_password`, `user\_profilepic`, `user\_status`) VALUES (:name, :nickname, :email, :pwd, :profilepic, 0) |

Le statut par défaut est 0, donc utilisateur.

### Function Add\_music ($music\_title, $music\_author, $music\_style\_id)

Cette fonction ajoute les musiques dans la base, mais seulement le titre, l’auteur et le style de musique. L’ajout total dans la table se fait en deux fois avec la fonction suivante. Une vérification se fait pour voir si la musique existe déjà.

|  |
| --- |
| INSERT INTO `music`(`music\_title`, `music\_author`, `music\_style\_id`) VALUES (:music\_title, :music\_author, :music\_style\_id) |

### Function Add\_file\_cover($music\_id, $music\_file, $music\_cover)

Cette fonction ajoute la deuxième partie de la musique, c’est-à-dire le fichier et sa pochette d’album, s’il y en a une.

|  |
| --- |
| UPDATE `music` SET `music\_file`= :music\_file, `music\_cover`= :music\_cover WHERE `music\_id` = :music\_id |

### Function Save\_parameters($question\_time, $questions\_number, $user\_id, $music\_style\_id)

Cette fonction sert à ajouter / sauvegarder les paramètres en fonction de l’utilisateur. Elle est appelée la première fois à la création du compte puis plus tard, dans la page paramètres si l’utilisateur les change.

|  |
| --- |
| INSERT INTO / UPDATE `parameters` SET parameters\_time`=:question\_time, `parameters\_questions\_number`=:questions\_number, `music\_style\_id`=:music\_style\_id WHERE `user\_id` =:user\_id |

### Function Add\_score($score, $score\_question, $user\_id)

Cette fonction ajoute le score de l’utilisateur. Il prend aussi en compte le nombre de questions.

|  |
| --- |
| INSERT INTO `score`(`score`, `score\_questions\_number`, `user\_id`) VALUES (:score, :score\_question, :user\_id) |

### Function Create\_game()

Cette fonction crée la partie. Coalesce sert à récupérer la première valeur non nulle de la table. Étant donné que game\_id n’est pas en auto incrément, la sous-requête va retourner 0, puis on fait une addition de 1 pour avoir l’id 1. Si une deuxième partie est lancée, la requête va récupérer la première valeur la plus haute, donc 1 et faire l’additionner à 1, ce qui donne 2, ainsi de suite.

|  |
| --- |
| SELECT COALESCE(MAX(`game\_id`), 0) + 1 as `game\_id` FROM `game` |

### Function Add\_game($game\_id, $user\_id, $music\_id)

Cette fonction ajoute une réponse juste à la partie et à l’utilisateur. C’est-à-dire qu’à chaque fois que l’utilisateur répond juste à une question, cette fonction est appelée. Ce qui permettra à la fonction Get\_all\_music\_random($game\_id, $music\_style\_id) d’exclure les bonnes réponses.

|  |
| --- |
| INSERT INTO `game`(`game\_id`, `user\_id`, `music\_id`) VALUES (:game\_id, :user\_id, :music\_id) |

## Récupération des données

### Function Check\_music($music\_title)

Cette fonction permet quand l’utilisateur coche une réponse, de la vérifier. Cette vérification sert à savoir si l’utilisateur a répondu juste, si oui, la musique sera alors exclue de la prochaine question.

|  |
| --- |
| SELECT `music\_title` WHERE `music\_title` = :music\_title |

### Function Get\_all\_music\_random($game\_id, $music\_style\_id)

Cette fonction permet de récupérer 4 musiques aléatoire de la base de données où les questions répondues justes sont exclues. Si l’utilisateur choisit dans les paramètres « tous » les styles de musiques, une autre requête est appelée. Sinon, la fonction retournera un 4 musiques sous forme de tableau du style de musique choisi.

|  |
| --- |
| SELECT \* FROM `music` WHERE `music\_id` NOT IN(SELECT `music\_id` FROM `game` WHERE `game\_id` = :game\_id) AND `music\_style\_id` IN (SELECT `music\_style\_id` FROM `music` WHERE `music\_style\_id` = :music\_style\_id) ORDER BY RAND() LIMIT 4 |
|  |
| SELECT \* FROM `music` WHERE `music\_id` NOT IN(SELECT `music\_id` FROM `game` WHERE `game\_id` = :game\_id) ORDER BY RAND() LIMIT 4 |

## Suppression

### Function Delete\_music($music\_id, $music\_cover, $music\_file)

Cette fonction supprime de la base l’enregistrement lié à l’ID. Mais elle supprime aussi les fichiers auxquels la musique est liée, c’est-à-dire le fichier « .mp3 » et la pochette d’album s’il y en a une.

|  |
| --- |
| DELETE FROM `music` WHERE `music\_id` = :music\_id |

## Javascript

Durant le développement, l’utilisation du javascript était nécessaire pour trois choses :

La première était d’avoir un « slidebar » qui permet de régler des valeurs. Pour ici, le temps en seconde de chaque question et le volume.

La deuxième était pour masquer le lecteur audio pour éviter que l’utilisateur puisse mettre pause durant la partie.

La troisième utilisation était pour faire un compte à rebours qui permet de faire recharger la page dès que le temps est écoulé.

## Site web

Le site a été développé de manière classique, c’est-à-dire qu’aucun « modèle » n’a été utilisé. Ex : MVC.

La page principale se nomme « index.php » et c’est par-là que tout utilisateur passe. Sur cette page se trouve le login ou un lien vers la page d’inscription. Les pages où l’utilisateur doit être connecté pour y accéder sont protégées, c’est-à-dire que si l’URL est entrée manuellement, le site nous redirigera automatiquement à la page d’accueil.

## Navigation

La navigation du site se fait grâce à une barre de navigation.

La barre de navigation se présente comme voici si l’on n’est pas connecté :

|  |
| --- |
|  |

Si l’on est connecté :

|  |
| --- |
|  |

Et si on est admin :

|  |
| --- |
|  |

# Installation

Pour installer le site, il faudra installer un logiciel « plateforme de développement Web », par exemple easyPHP, Laragon ou bien Wamp. Une fois l’installation faite, mettre les fichiers sources dans le dossier web de l’application et lancer l’application. Il faudra importer la base de données trouvable dans le dossier « db ».

Pour le développement de l’application, aucun compte PhpMyAdmin n’a été créé, il n’est donc pas nécessaire d’en créer un.

# Amélioration possible

Le projet a été développé dans le cadre du TPI sur une durée de deux semaines. Durant ces deux semaines, beaucoup de concessions ont dû être faites. Ce sont donc pour une partie, ces concessions qui seraient des améliorations possibles. Par exemple :

* Le fait de pouvoir modifier le pseudo de l’utilisateur (seule l’image de profil est modifiable).
* Mot de passe oublié.
* Le filtre dans la page « liste des musiques » de l’admin. (par exemple : trier les morceaux par titre).
* Rendre le jeu jouable à plusieurs.
* Meilleure sécurité de l’application.
* Ajout d’une gestion d’utilisateurs.
* L’aspect visuel du site.

# Rapport de tests

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Date** | **Scénario** | **Résultat attendu** | **Résultat** | **Date testé** | **Conforme** |
| **1.** | 06.06.18 | L’utilisateur rentre ses informations sur la page login et clique sur « Se connecter ». | L’utilisateur est connecté et renvoyé à son profil. | L’utilisateur est connecté et renvoyé à son profil. | 06.06.18 | OK |
| **2.** | 06.06.18 | L’utilisateur rentre un mauvais mot de passe et tente de se connecter. | La page lui renvoie un message d’erreur. | La page lui renvoie un message d’erreur. | 07.06.18 | OK |
| **3.** | 06.06.18 | L’utilisateur s’inscrit en rentrant les informations avec ou sans image. | L’utilisateur est inscrit et renvoyé à la page d’accueil. | L’utilisateur est inscrit et renvoyé à la page d’accueil. | 06.06.18 | OK |
| **4.** | 06.06.18 | L’utilisateur oubli de rentrer un champ. | Un message s’affiche pour lui dire que tel champ ne peut pas être vide. | Un message s’affiche pour lui dire que tel champ ne peut pas être vide. | 06.06.18 | OK |
| **5.** | 07.06.18 | L’utilisateur appuie sur « valider » dans son profil sans mettre d’image. | Rien ne se passe. | Rien ne se passe. | 07.06.18 | OK |
| **6.** | 07.06.18 | L’utilisateur mets une image dans le « file input » et appuie sur « valider ». | L’image est remplacée par la nouvelle et dans le répertoire, l’ancienne image est supprimée. | L’image est remplacée par la nouvelle et dans le répertoire, l’ancienne image est supprimée. | 07.06.18 | OK |
| **7.** | 07.06.18 | L’utilisateur appuie sur « se déconnecter ». | L’utilisateur est déconnecté et renvoyé à la page d’accueil. | L’utilisateur est déconnecté et renvoyé à la page d’accueil. | 07.06.18 | OK |
| **8.** | 07.06.18 | L’utilisateur se connecte en tant qu’admin. | L’onglet CRUD apparaît. | L’onglet CRUD apparaît. | 07.06.18 | OK |
| **9.** | 07.06.18 | L’admin tente d’ajouter une musique sans remplir les champs. | Un message lui indique que les champs doivent être remplis. | Un message lui indique que les champs doivent être remplis. | 07.06.18 | OK |
| **10.** | 07.06.18 | L’admin tente de mettre un fichier autre qu’audio dans le champ dédié ou le champ de la pochette d’album. | Il ne peut pas, le boîte de dialogue ne reconnaît pas le fichier et donc, ne l’affiche pas. | Il ne peut pas, le boîte de dialogue ne reconnaît pas le fichier et donc, ne l’affiche pas. | 07.06.18 | OK |
| **11.** | 07.06.18 | L’admin ajoute une musique sans mettre de pochette d’album (image). | La musique est ajoutée et le champ image est vide. | La musique est ajoutée et le champ image est vide. | 07.06.18 | OK |
| **12.** | 07.06.18 | L’admin va dans la page où s’affiche toutes les musiques dans la base.  Les images s’affichent correctement, même quand il n’y a pas de données dans la base. | L’image s’affiche correctement. S’il n’y a pas d’image, une image « no\_cover » s’affichera à la place. | L’image s’affiche correctement. S’il n’y a pas d’image, une image « no\_cover » s’affichera à la place. | 07.06.18 | OK |
| **13.** | 07.06.18 | L’admin supprime une musique. | La musique est supprimée et ses fichiers avec. | La musique est supprimée et ses fichiers avec. | 07.06.18 | OK |
| **14.** | 07.06.18 | L’admin modifie une musique. | La musique est bien modifiée dans la base de données et les fichiers sont remplacés. | La musique est bien modifiée dans la base de données et les fichiers sont remplacés. | 14.06.18 | OK |
| **15.** | 07.06.18 | L’utilisateur enregistre ses paramètres. | Les paramètres sont enregistrés dans la base de données. | Les paramètres sont enregistrés dans la base de données. | 14.06.18 | OK |
| **16.** | 07.06.18 | La page profil affiche bien les scores de l’utilisateur. | Les scores sont affichés. | Les scores sont affichés. | 14.06.18 | OK |
| **17.** | 07.06.18 | L’utilisateur appuie sur « commencer » dans la page « jouer ». | La page le redirige vers la page de jeu. Affiche 4 possibilités et lance une musique. | La page le redirige vers la page de jeu. Affiche 4 possibilités et lance une musique. | 07.06.18 | OK |
| **18.** | 07.06.18 | L’admin tente d’upload un fichier plus lourd que 16 MB. | La page se recharge et rien ne se passe. | La page se recharge et rien ne se passe. | 07.06.18 | OK |
| **19.** | 07.06.18 | L’admin modifie uniquement la musique. | Seule la musique est modifiée. | Seule la musique est modifiée. | 14.06.18 | OK |
| **20.** | 07.06.18 | L’admin modifie le titre et/ou l’auteur d’une musique sans charger une autre musique, ni une autre pochette. | La musique ne change pas ni la pochette, mais le titre et/ou l’auteur change. | La musique ne change pas ni la pochette, mais le titre et/ou l’auteur change. | 14.06.18 | OK |
| **21.** | 07.06.18 | L’admin modifie uniquement la pochette. | Seule la pochette change. | Seule la pochette change. | 14.06.18 | OK |
| **22.** | 08.06.18 | L’utilisateur va sur la page « paramètres ». | La page affiche les paramètres de l’utilisateur. | La page affiche les paramètres de l’utilisateur. | 14.06.18 | OK |
| **23.** | 08.06.18 | L’utilisateur laisse le compte à rebours s’écouler. | La page se recharge et la question suivante est affichée.. | La page se recharge et la question suivante est affichée.. | 14.06.18 | OK |
| **24.** | 08.06.18 | L’utilisateur choisit une mauvaise réponse. | La page se recharge et la question suivante est affichée. | La page se recharge et la question suivante est affichée. | 14.06.18 | OK |
| **25.** | 08.06.18 | L’utilisateur choisit la bonne réponse. | La page se recharge et la question suivante s’affiche. La musique « bonne réponse » ne sera plus réaffichée. | La « bonne réponse » peut réapparaître. | 14.06.18 | KO |
| **26.** | 08.06.18 | L’utilisateur appuie sur le bouton sans cocher de réponse. | La question suivante se lance. | La question suivante se lance. | 14.06.18 | OK |

# Conclusion

Ce projet était pour moi une belle expérience. Ce n’était pas de tout repos, beaucoup de stress, mais c’est ce stress qui m’a permis de travailler à un rythme régulier et de ne pas m’égarer. J’ai pu m’organiser sans qu’on me dise de prioriser une étape à une autre.

Ce projet m’a beaucoup apporté sur la gestion de mon travail, sur l’ordre et la priorité des choses. J’ai aussi amélioré mes connaissances en développement web PHP et SQL.

J’ai appris à simplifier mon code en modifiant la base de données, grâce à ça, j’ai appris une fonction SQL « coalesce » qui permet de récupérer la première valeur non nulle. C’est pour éviter d’avoir une table en auto-incrément.

Grâce à ce projet, j’ai appris à travailler en solo, j’ai sollicité très peu d’aide et j’en suis entièrement satisfait du résultat.

## Objectif

Mon objectif initial était de remplir le cahier des charges. Ensuite, faire ce que ma responsable demandait. Puis après, s’il me restait du temps, de peaufiner l’application et la sécuriser.

## Problèmes rencontrés

Au début du TPI, je n’avais pas compris le fonctionnement du schéma de la base de données que j’ai reçu. De ce fait, j’ai demandé à ma responsable s’il était possible de modifier ma base de données. Autorisation qu’elle m’accorda. J’ai ensuite modifié ma base de données et grâce à une remarque d’une de mes professeurs, j’ai finalement compris le fonctionnement du schéma initial. J’ai de ce fait, encore modifié la base de données, ce qui implique aussi la modification du code pour l’adapter à la nouvelle base. Cela m’a fait perdre beaucoup de temps, mais grâce à tout ça, j’ai pu mieux comprendre le fonctionnement des bases de données.

Un deuxième problème est rencontré à la fin du développement. Une des requêtes qui permet de récupérer les données depuis la base de données sauf les questions répondues juste ne fonctionne pas. C’est-à-dire que la requête de base doit récupérer toutes les musiques de la base. Ensuite, elle filtre pour qu’à chaque fois qu’une question a été répondue juste, elle soit exclue, cependant, PHP n’applique pas ce filtre. Uniquement PHP, la requête en SQL est juste et fonctionne.

# Annexes

## Table des illustrations

[Figure 1 - Accueil 5](#_Toc516817304)

[Figure 2- Page d’inscription 6](#_Toc516817305)

[Figure 3- Page de profil 6](#_Toc516817306)

[Figure 4- Page des paramètres 7](#_Toc516817307)

[Figure 5- Page de début de jeu 7](#_Toc516817308)

[Figure 6 - En jeu 8](#_Toc516817309)

[Figure 7 - Liste des musiques 8](#_Toc516817310)

[Figure 8 - Ajout / modification des musiques 9](#_Toc516817311)

[Figure 9 - Ajout d'un style de musique 9](#_Toc516817312)

[Figure 10 - MCD 9](#_Toc516817313)

[Figure 11 - PDO 10](#_Toc516817314)

## Bibliographie

### PHP

<http://php.net/>

### SQL

<https://dev.mysql.com/>

### Bootstrap

<https://getbootstrap.com/>

### Wamp

<http://www.wampserver.com/>

### W3School

<https://www.w3schools.com/>

### Stackoverflow

<https://stackoverflow.com/>

## Aides reçues

* Mme Terrier
* M. Comminot
* M. Garchery
* M. Bergeret
* Mme Weber
* M. Vanini
* M. Foti

## Remerciement

Je tenais à remercier Mme Terrier pour m’avoir expliqué le fonctionnement du schéma initial. M. Comminot et M. Garchery pour m’avoir écouté quand j’ai eu mon problème de requête SQL en PHP et d’avoir tenté de m’aider. M. Bergeret pour avoir corrigé mon MCD et ma base de données. Et Mme Weber pour avoir été très compréhensive sur mes idées farfelues et d’avoir été ma responsable TPI.

Je remercie aussi M. Vanini et M. Foti pour avoir porté de l’intérêt à mon projet et pour m’avoir aiguillé sur certains points.

# Planning

## Planning théorique

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 04.06 | 05.06 | 06.06 | 07.06 | 08.06 | 11.06 | 12.06 | 13.06 | 14.06 | 15.06 |
| Mise en place immersion |  |  |  |  |  |  |  |  |  |  |
| Base de données |  |  |  |  |  |  |  |  |  |  |
| Modélisation du site (template et structure) |  |  |  |  |  |  |  |  |  |  |
| Différents formulaires |  |  |  |  |  |  |  |  |  |  |
| Page de paramètres |  |  |  |  |  |  |  |  |  |  |
| Quizz |  |  |  |  |  |  |  |  |  |  |
| Test |  |  |  |  |  |  |  |  |  |  |
| Documentation |  |  |  |  |  |  |  |  |  |  |
| Journal de bord |  |  |  |  |  |  |  |  |  |  |

## Planning effectif

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 04.06 | 05.06 | 06.06 | 07.06 | 08.06 | 11.06 | 12.06 | 13.06 | 14.06 | 15.06 |
| Mise en place immersion |  |  |  |  |  |  |  |  |  |  |
| Base de données |  |  |  |  |  |  |  |  |  |  |
| Modélisation du site (template et structure) |  |  |  |  |  |  |  |  |  |  |
| Différents formulaires |  |  |  |  |  |  |  |  |  |  |
| Page de paramètres |  |  |  |  |  |  |  |  |  |  |
| Quizz |  |  |  |  |  |  |  |  |  |  |
| Test |  |  |  |  |  |  |  |  |  |  |
| Documentation |  |  |  |  |  |  |  |  |  |  |
| Journal de bord |  |  |  |  |  |  |  |  |  |  |