



TPI  
Gregory Mbayo

## Table des matières

---

1	Analyse préliminaire .....	4
1.1	Introduction.....	4
1.2	Contexte .....	4
1.3	Objectifs.....	4
1.3.1	Points techniques évalués spécifique au projet .....	4
1.4	Planification initiale .....	6
2	Analyse / Conception.....	8
2.1	Méthodologie de travail.....	8
2.2	Stratégie de sauvegarde.....	8
2.3	Public visé .....	8
2.4	Concept .....	9
2.4.1	Maquette graphique.....	9
2.4.2	Modèle conceptuel de données .....	17
2.4.3	Modèle logique des données .....	19
2.4.4	Modèle physique des données .....	21
2.5	Stratégie de test.....	22
2.6	Risques techniques .....	23
2.7	Planification .....	23
2.7.1	Dates et Horaires de travail .....	23
2.7.2	Déroulement du travail.....	23
2.7.3	Planification détaillée .....	24
2.8	Dossier de conception .....	28
2.8.1	Matériel à disposition .....	28
2.9	Glossaire .....	28
3	Réalisation.....	30
3.1	Dossier de réalisation .....	30
3.1.1	Gestion de la base de données .....	30
3.1.2	MVC.....	32
3.1.3	Connexion à la base de données .....	35
3.1.4	Authentification depuis la page login .....	35
3.1.5	Script de hachage de mot de passe .....	37
3.1.6	Page du choix de l'équipement.....	39
3.1.7	Page de réservation.....	40
3.1.8	Page Mes réservations .....	44
3.1.9	Déconnexion.....	47
3.1.10	Méthode renderView.....	47
3.1.11	Amélioration .....	48
3.1.12	Problèmes rencontrés.....	50
3.1.13	Git .....	51
3.2	Description des tests effectués .....	53
3.2.1	Test de l'authentification .....	53
3.2.2	Test d'authentification avec des identifiants erronés .....	53
3.2.3	Test de hachage des mots de passe .....	54
3.2.4	Test de choix de l'équipement .....	55
3.2.5	Test de la réservation de l'équipement .....	55
3.2.6	Test d'affichage de mes réservations .....	56
3.2.7	Test d'annulation d'une réservation .....	56

---

3.2.8	Test de déconnexion .....	57
3.2.9	Test de statut terminé .....	57
3.2.10	Test du calendrier interactif.....	58
3.2.11	Test de l'affichage des images .....	58
3.3	Erreurs restantes .....	59
3.4	Liste des documents fournis .....	59
4	Conclusions .....	59
4.1	Bilan des fonctionnalités .....	59
4.2	Bilan personnel.....	60
4.2.1	Points positifs.....	60
4.2.2	Points négatifs .....	60
4.3	Bilan de la planification .....	60
4.3.1	Bilan du suivi de la maquette graphique .....	60
4.4	Si le projet était à refaire .....	61
4.5	Suites possibles au projet.....	61
5	Annexes.....	61
5.1	Table des illustrations .....	61
5.2	Résumé du rapport du TPI / version succincte de la documentation .....	63
5.2.1	Situation de départ.....	63
5.2.2	Mise en œuvre.....	63
5.2.3	Résultat .....	63
5.3	Journal de travail .....	64
	.....	66
	Voici le journal de travail de la semaine 4 .....	67
	Voici le journal de travail de la semaine 5 .....	68
5.4	Archives du projet.....	69

## 1 Analyse préliminaire

### 1.1 Introduction

Le but de ce projet est de réaliser une application web permettant aux membres d'un club de sport de réserver des créneaux pour les terrains ou les équipements sportifs.

### 1.2 Contexte

Un club sportif souhaite mettre à disposition ses ressources (terrains ou équipements) aux membres sans pour autant disposer de solutions informatiques complexes ou coûteuses. Ce projet vise à développer une application web simple, fonctionnelle et intuitive, permettant aux membres d'un club de réserver une ressource disponible, à une date et un horaire précis.

Les ressources seront prédéfinies dans la base de données. Aucun espace administrateur n'est nécessaire : les membres pourront uniquement consulter et réserver ce qui est déjà disponible.

### 1.3 Objectifs

Voici la liste des livrables à la fin du TPI :

- Une planification initiale.
- Un rapport de projet contenant entre autres un plan de tests manuels.
- Un journal de travail.
- Script(s) SQL (structure + utilisateurs + ressources).
- Code source complet disponible dans un repository GIT.
- README.md Clair et détaillé (explication du contenu de repository et liens vers toutes les ressources qui s'y trouvent).

Voici la liste des fonctionnalités à mettre en place durant ce projet :

- Connexion à l'application via des comptes prédéfinis.
- Consultation des ressources disponibles (terrains équipements).
- Réervation d'un créneau horaire pour une ressource.
- Affichage des réservations passées et à venir par l'utilisateur.
- Annulation d'une réservation encore à venir.

#### 1.3.1 Points techniques évalués spécifique au projet

La grille d'évaluation définit les critères généraux selon lesquels le travail du candidat sera évalué (documentation, journal de travail, respect des normes, qualité, ...)

En plus de cela, le travail sera évalué sur les 7 points spécifiques suivants (Point A14 à A20) :

- Mise en place d'une structure MVC : séparation claire entre les modèles (accès BDD), les vues (affichage HTML) et les contrôleurs(logique métier).

- Connexion sécurisée à la base de données via PDO et utilisateur MySQL dédié (pas d'utilisation du compte *root*)
- Authentification fonctionnelle avec gestion de session et espace personnel accessible après connexion.
- Affichage dynamique des ressources disponibles : seules les ressources non réservées pour la date / heure choisies sont proposées.
- Réservation dynamique d'une ressource : enregistrement conditionnel à la disponibilité réelle lors de la soumission.
- Liste des réservations pour l'utilisateur connecté avec statut(confirmée/annulée)
- Annulation d'une réservation à venir par l'utilisateur connecté : mise à jour du statut de la réservation dans la base de données.

**Remarque :**

Le recours à des outils en ligne d'intelligence artificielle (ex. : Chat GPT) doit être mentionné et ne peut servir que d'inspiration à la réalisation. En cas d'abus, l'évaluation du TPI en tiendra compte.

## 1.4 Planification initiale

Voici la planification initiale du projet par semaine :

Tâches - objectifs	Nb 1/4 heure	S 1
Absence - Imprévus	0	
	0	
Analyse	49	
	0	
Implémentation	147	
	0	
Tests	23	
	0	
Documentation	122	
	0	
<b>Total planifié</b>	<b>341</b>	
<b>Total réalisé</b>	<b>0</b>	

Figure 1 : planification initiale, semaine 1

Tâches - objectifs	Nb 1/4 heure	S 2
Absence - Imprévus	0	
	0	
Analyse	49	
	0	
Implémentation	147	
	0	
Tests	23	
	0	
Documentation	122	
	0	
<b>Total planifié</b>	<b>341</b>	
<b>Total réalisé</b>	<b>0</b>	

Figure 2 : planification initiale, semaine 2

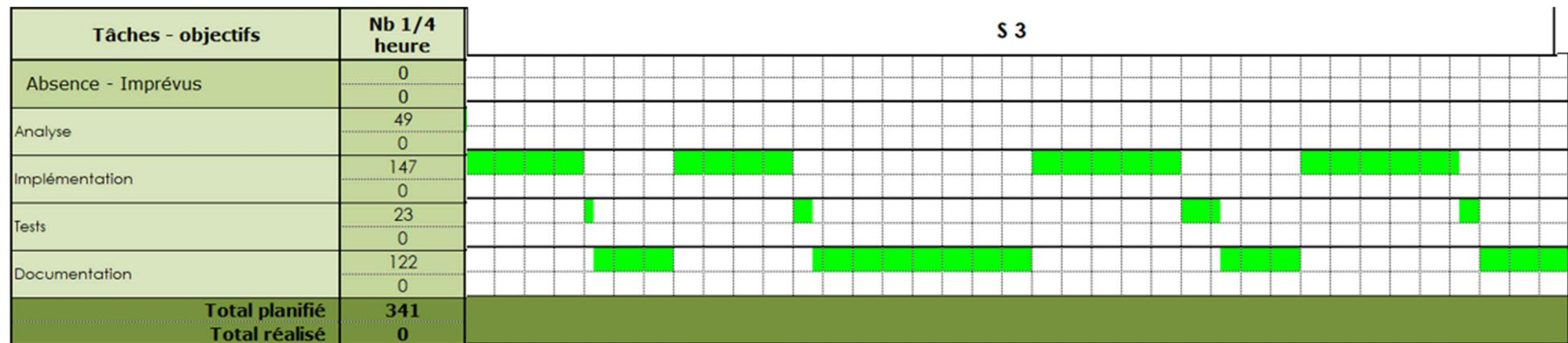


Figure 3 : planification initiale, semaine 3

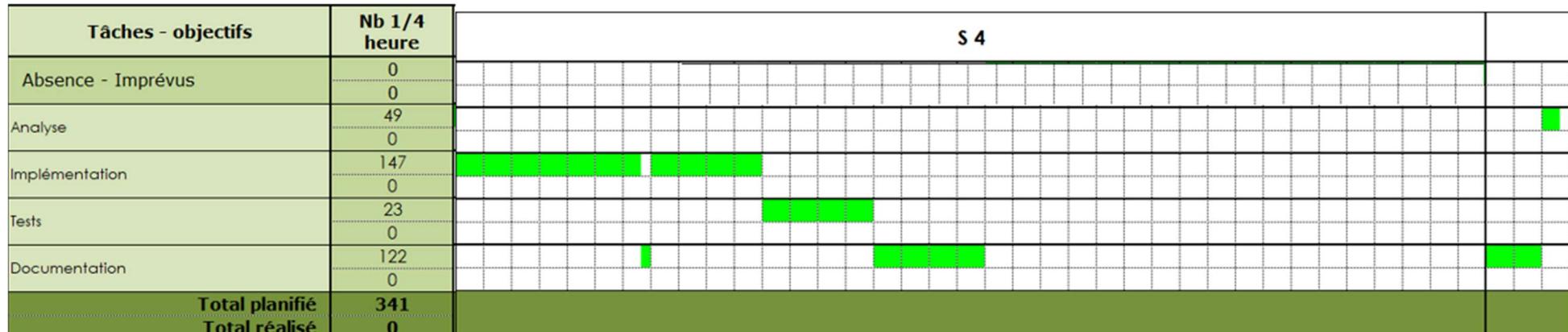


Figure 4 : planification initiale, semaine 4 et 5

## 2 Analyse / Conception

### 2.1 Méthodologie de travail

Pour ce TPI, je souhaite appliquer une méthodologie de travail en cascade. En effet, cette méthode me paraît être la plus adaptée pour le projet. Elle se décompose en 6 étapes :

#### 1. La définition des besoins.

Il s'agit de recueillir les attentes et exigences du commanditaire et définit l'orientation du reste du projet.

#### 2. L'analyse des besoins

Les exigences sont analysées pour en faire des objectifs concrets.

#### 3. La conception

Permet de découper et de planifier les tâches pour le bon fonctionnement du projet.

#### 4. La mise en œuvre

Réalisation des tâches de travail.

#### 5. La validation

Le produit terminé est contrôlé. Si des modifications sont nécessaires, elles sont effectuées puis le produit est validé.

#### 6. La mise en service

Le produit est mis en service.

### 2.2 Stratégie de sauvegarde

Afin de ne pas subir d'incident de perte de données, j'ai décidé de prévoir un enregistrement de mon travail sur le disque dur du poste de travail avant chaque pause et d'en copier l'intégralité sur mon disque dur externe ainsi que de le mettre en ligne sur le Git à la fin de chaque demi-journée afin d'assurer des backups. De cette façon, je suis à l'abri de perdre plus d'une demi-journée de travail.

### 2.3 Public visé

Le public cible pour ce projet sont les membres du club sportif souhaitant réserver des terrains et de l'équipement.

## 2.4 Concept

### 2.4.1 Maquette graphique

Pour la conception de ma maquette graphique, j'ai décidé d'utiliser le logiciel Pencil Project que j'ai eu l'occasion à maintes reprises d'utiliser durant mes projets ou cours à l'ETML.

La maquette se décompose en 7 pages :

1. Page de connexion
2. Page d'espace personnel
3. Page du choix de ressource
4. Page de réservation
5. Page de validation
6. Page d'historique des réservations
7. Page d'annulation de réservations

Voici la page de connexion :

veuillez vous connecter

Email

Mot de passe

Se connecter

Footer

*Figure 5 : page de connexion*

Voici la page d'espace personnel :



Figure 6 : page d'espace personnel

Page du choix des ressources :

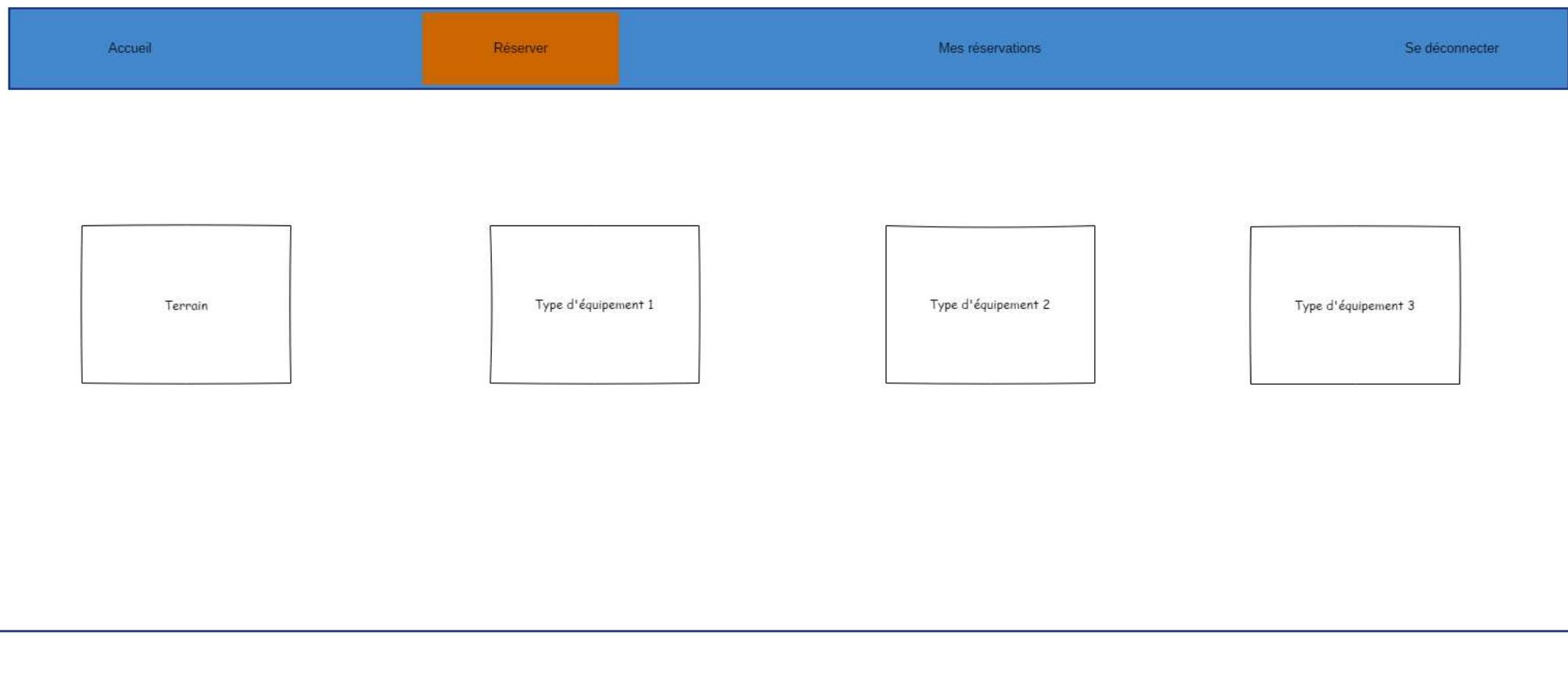


Figure 7 : page du choix des ressources

Page de réservation :

Accueil
Réserver
Mes réservations
Se déconnecter

## Terrains

Choisir la date de réservation

M	T	W	T	F	Sa	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	1	2	3	4
5	6	7	8	9	10	11

Valider

Informations

1

Terrain de foot

Valider

2

Terrain de foot

Valider

7

Terrain de foot

Valider

Retour

Footer

*Figure 8 : Page de réservation*

Page de validation :

The screenshot shows a web page with a blue header bar. From left to right, the header contains: "Accueil" (white text on blue background), "Reserver" (white text on orange background), "Mes réservations" (white text on blue background), and "Se déconnecter" (white text on blue background). Below the header, the word "Terrains" is centered in bold black font. A large rectangular box in the center contains the text: "Votre réservation de \"nom de la ressource\" n° \"numero de la ressource\" pour le \"date\" à \"heure\" à été validée". At the bottom right of the page, there is a "Retour" button.

Footer

*Figure 9 : page de valuidation*

Page d'historique des réservations :

terrain n°	Informations	date de réservation	Statut
4	Terrain de foot	06.06.2025	Confirmée <button>annuler</button>
6	Terrain de foot	22.05.2025	Annulée
8	Terrain de foot	12.02.2025	Confirmée

Footer

Figure 10 : page d'historique des réservations

---

Page d'annulation des réservations :

The screenshot shows a web application interface. At the top, there is a blue header bar with four items: "Accueil", "Réserver", "Mes réservations" (which is highlighted in orange), and "Se déconnecter". Below the header, the title "Historique des commandes" is centered. A large rectangular box contains the message: "Votre réservation de \"nom de la ressource\" n° \"numero de la ressource\" pour le \"date\" à \"heure\" à été Annulée". In the bottom right corner of the main content area, there is a small button labeled "Retour".

---

Footer

---

## 2.4.2 Modèle conceptuel de données

Pour la création du modèle conceptuel de données, j'ai choisi le logiciel looping avec lequel j'ai déjà eu l'occasion de pratiquer. Looping est un logiciel libre d'utilisation qui permet la conception de MCD, MLD et permet la génération de script SQL à partir de celui-ci. À la suite d'une réflexion, j'ai décidé de modifier la table RESERVATION en ajoutant l'attribut *status* qui permettra de définir si la réservation est : en cours, passée ou annulée. J'ai effectué la modification avec la commande suivante :

```
ALTER TABLE t_reservation
ADD status enum('active', 'terminée', 'annulée', '');
```

*Vous trouverez le mcd original en annexe.*

Voici le MCD réalisé sur Looping :



Figure 12 : MCD

#### 2.4.2.1 Les tables

- **La table USER**

Cette table contient les données des membres du club de sport.

**user\_id** est l'identifiant de la table. Son type est un **Counter**. Il est **NOT NULL** car chaque membre doit posséder un identifiant.

**firstname** est l'attribut qui comporte le prénom du membre. C'est un **VARCHAR** d'une longueur de **255**, car cela est suffisant pour un prénom. Il est **NOT NULL**.

**lastname** est l'attribut qui comporte le nom du membre. C'est un **VARCHAR** d'une longueur de **255**, car cela est suffisant pour un nom. Il est **NOT NULL**.

**password** est l'attribut qui comporte le mot de passe du membre. C'est un **VARCHAR** d'une longueur de **255**, car une fois le mot de passe hashé, il fera toujours la même longueur de 60 caractères. Cependant si dans le futur l'algorithme change il se peut que le nombre de caractères soit plus grand. Il est **NOT NULL**.

**email** est l'attribut qui comporte l'adresse mail du membre. C'est un **VARCHAR** d'une longueur de **320** car la longueur maximale que peut atteindre une adresse mail est de 320 caractères. Il est **NOT NULL**.

- **La table Reservation**

Cette table contient les données des réservations.

**reservation\_id** est l'identifiant de la table. Son type est un **Counter**. Il est **NOT NULL** car chaque réservation doit posséder un identifiant.

**date** est l'attribut qui comporte la date de la réservation. C'est un **DATE**. Il est **NOT NULL**.

**hour** est l'attribut qui comporte l'heure de la réservation. C'est un **TIME**. Il est **NOT NULL**.

**status** est l'attribut qui comporte le statut de la réservation. C'est un **ENUM**. Il est **NOT NULL**.

- **La table Resource**

Cette table contient les données des ressources du club de sport.

**type** est l'attribut qui comporte le type de ressource. C'est un **VARCHAR** d'une longueur de **255**. Il est **NOT NULL**.

**name** est l'attribut qui comporte le nom de la ressource. C'est un **VARCHAR** d'une longueur de **255**. Il est **NOT NULL**.

**available** est l'attribut qui définit si la ressource est disponible ou pas. C'est un **booléen**. Il est **NOT NULL**.

#### **2.4.2.2 Les associations**

L'association **HAVE** :

Un membre peut avoir 0 à N réservation alors qu'une réservation ne peut être associée à un seul membre.

L'association **CONTAIN** :

Une réservation ne peut contenir qu'une seule ressource tandis qu'à l'inverse, une ressource peut être dans 0 à N réservations.

#### **2.4.3 Modèle logique des données**

Le modèle logique des données a été réalisé avec looping. Les clés étrangères ont été renommées pour être conformes aux conventions de codage de l'ETML.

Voici le MLD :

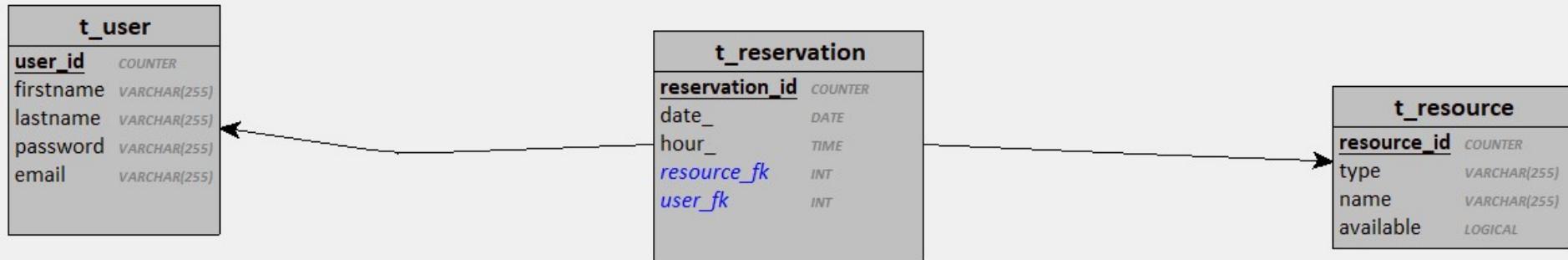


Figure 13 : MLD

#### 2.4.4 Modèle physique des données

Voici le modèle physique des données (MPD) :

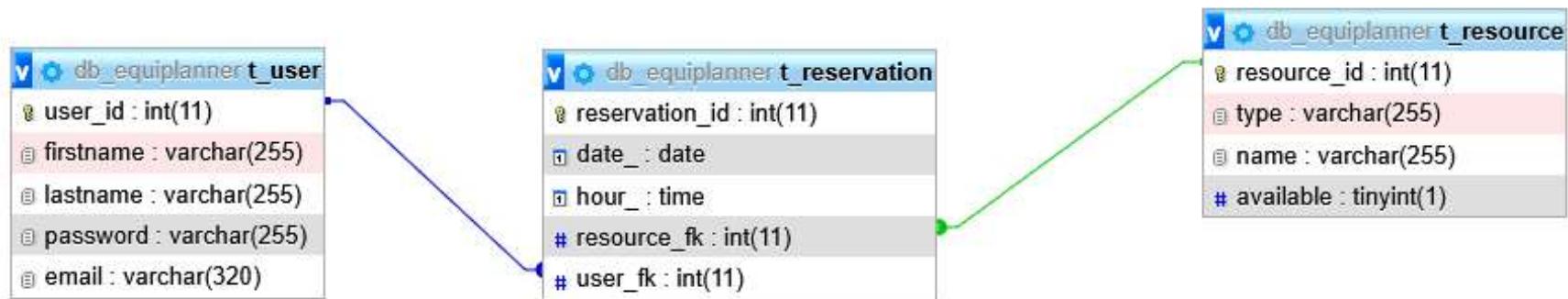


Figure 14 : MPD

## 2.5 Stratégie de test

Pour tester les fonctionnalités de mon projet, j'ai décidé de créer des cas avec scénario. Chaque étape des scénarios seront spécifiées et les résultats attendus seront définis à l'aide d'un tableau. Avec cette méthode, je peux clairement et aisément définir la réussite ou l'échec d'un test.

En cas d'échec, les modifications sont apportées et le test est réitéré

Voici un exemple de test :

Nom du test :	Exemple	
Identifiant :	Exemple-01	
Objectif :	Se connecter à son compte	
Prérequis :	-	
Cas de test :		
Etape	Action	Résultat attendu
1	Cliquer sur le bouton connexion	La page de connexion s'ouvre demandant l'adresse mail et le mot de passe
2	Entrer ses informations	L'adresse mail est affichée tandis que le mot de passe est caché
3	Cliquer sur le bouton se connecter	La page home du membre s'ouvre

## 2.6 Risques techniques

Le risque principal que je perçois serait de négliger la documentation au profit de la réalisation. Je devrais donc me tenir au maximum à ma planification initiale.

Un autre risque présent est le fait que je n'ai jamais eu l'occasion de vraiment expérimenter le *JavaScript* dans un projet.

## 2.7 Planification

### 2.7.1 Dates et Horaires de travail

Le TPI débute le jeudi 8 mai 2025 à 08h00 et se termine le lundi 2 juin 2025 à 10h00.

Les horaires de travail sont :

### 2.7.2 Déroulement du travail

Le projet se déroule sur un total de 90 heures réparties en 20% d'analyse, 40% d'implémentation, 10% de tests et 30% de documentation.

Lors du premier jour du TPI, une rencontre avec l'expert n°1 et le chef de projet s'effectue pour la prise de connaissance du cahier des charges et la validation de celui-ci.

Au milieu du projet, l'expert n°2 rend une visite au candidat afin de contrôler le travail effectué jusqu'à présent et de répondre aux éventuelles questions du candidat. Une fois arrivé au terme du projet, le candidat remet son travail aux experts et au chef de projet.

### 2.7.3 Planification détaillée

Pour la réalisation de ma planification détaillée lors du premier jour, j'ai choisi d'utiliser le modèle Excel de l'ETML. Celui-ci permet de découper les tâches en quart d'heure ce qui permet d'avoir une grande maniabilité de planification.

Voici la planification détaillée :

Semaine		1		
Tâche	Durée [1/4 h.]	Heures	Explications: qu'est-ce qui se fait et comment ?	Liens, références, ...
<b>LUNDI 05.05.25</b>				
	21			
<b>MARDI 06.05.25</b>				
	24			
<b>MERCREDI 07.05.25</b>				
	12			
<b>JEUDI 08.05.25</b>				
Analyse	6	1h30	Début du TPI, Séance avec l'expert n°1 et le chef de projet pour débuter le tpi, Analyse du cahier des charges dans son entièreté, analyse des outils/logiciel essentiel pour ce TPI.	
Analyse	15	3h45	Réalisation de la planification initiale du TPI, Réalisation du journal de travail de la journée, Envoi de la planification initiale aux experts et au chef de projet par mail.	
Analyse	6	1h30	Réalisation de la maquette graphique des différentes vue du site web.	
<b>VENDREDI 09.05.25</b>				
Documentation	15	3h45	Création du rapport, réalisation de la partie analyse de celui-ci. Avancement du rapport dans la partie réalisation de la maquette graphique et du mcd. Réalisation du journal de travail et envoie des documents aux experts et au chef de projet.	
Analyse	12	3h00	Conception du MCD de la base de donnée avec le logiciel Looping.	
Total semaine	111	13h30	<b>Max. 111</b>	

Figure 15 : planification détaillée, semaine 1

<b>Semaine 2</b>				
<b>Tâche</b>	<b>Durée [1/4 h.]</b>	<b>Heures</b>	<b>Explications: qu'est-ce qui se fait et comment ?</b>	<b>Liens, références, ...</b>
<b>LUNDI 12.05.25</b>				
Implémentation	16	4h00	Création de la base de donnée sur phpMyAdmin, création du Git.	
Documentation	5	1h15	Avancement du rapport dans la partie réalisation sur la création de la base de donnée. Réalisation du journal de travail.	
<b>MARDI 13.05.25</b>				
Analyse	8	2h00	Analyse de l'architecture MVC qui sera implémentée.	
Implémentation	12	3h00	Ajout des données prédefinis à la base de donnée. Conception de l'architecture MVC.	
Documentation	4	1h	Avancement du rapport dans la partie réalisation sur l'ajout des données prédefinis ainsi que la partie MVC. Réalisation du journal de travail et envoie des documents aux experts et au chef de projet.	
<b>MERCREDI 14.05.25</b>				
Documentation	12	3h00	Avancement du rapport. Réalisation du Journal de travail.	
<b>JEUDI 15.05.25</b>				
Implémentation	24	6h00	Implémentation du système d'authentification et de l'espace personnel sur le site web, Implémentation de l'affichage dynamique des ressources disponibles.	
Tests	2	00h30	Test de la fonctionnalité du système d'authenfication.	
Documentation	1	00h15	Réalisation du journal de travail de la journée.	
<b>VENDREDI 16.05.25</b>				
Documentation	19	4h45	Avancement du rapport dans la partie réalisation sur les fonctionnalité ajoutée et dans la partie test. Réalisation du journal de travil et envoie des documents aux experts et au chef de projet.	
Implémentation	8	2h00	commencement de l'implémentation de la réservervation dynamique.	
<b>Total semaine</b>	<b>111</b>	<b>27h45</b>	<b>Max. 111</b>	

Figure 16 : planification détaillée, semaine 2

Semaine 3				
Tâche	Durée [1/4 h.]	Heures	Explications: qu'est-ce qui se fait et comment ?	Liens, références, ...
<b>LUNDI 19.05.25</b>				
Implémentation	12	3h00	Continuation de l'implémentation de la réservation dynamique.	
Tests	1	00h15	Test de la fonctionnalité réservation dynamique.	
Documentation	8	2h00	Avancement du rapport dans la partie réalisation de fonctionnalité réservation dynamique et tests.	
<b>MARDI 20.05.25</b>				
Implémentation	12	3h00	Implémentation des listes des réservations pour l'utilisateur connecté.	
Tests	2	00h30	Tests de la liste des réservations.	
Documentation	10	2h30	Avancement du rapport dans la partie réalisation sur la liste des réservations et tests. Réalisation du journal de travail et envoi des documents aux experts et au chef de projet.	
<b>MERCREDI 21.05.25</b>				
Documentation	12	3h00	Avancement du rapport et réalisation du journal de travail.	
<b>JEUDI 22.05.25</b>				
Implémentation	15	3h45	Implémentation de l'annulation d'une réservation à venir par un utilisateur.	
Tests	4	1h00	Test de toutes les fonctionnalités.	
Documentation	8	2h00	Avancement du rapport de la partie réalisation sur la fonctionnalité d'annulation d'une réservation et des tests. Réalisation du journal de travail.	
<b>VENDREDI 23.05.25</b>				
Implémentation	16	4h00	Réalisation du fichier README.md dans Git.	
Tests	2	00h30	Test du fichier README.md.	
Documentation	9	2h15	Avancement du rapport dans la partie Réalisation sur le fichier README.md. Réalisation du Journal de travail et envoi des documents aux experts et au chef de projet.	
Total semaine	111	27h45	<b>Max. 111</b>	

Figure 17 : planification détaillée, semaine 3

Figure 18 : planification détaillée, semaine 4 et 5

## 2.8 Dossier de conception

### 2.8.1 Matériel à disposition

- 1 poste de travail tournant sous Windows 10 (configuration standard ETML)
- Suite Office 365
- Logiciel Looping
- Logiciel Visual Studio Code
- Logiciel Pencil
- Serveur local UWAMP(apache, PHP,MySQL)
- GIT+plateforme GitHub

## 2.9 Glossaire

- **Attribut**

L'attribut est une caractéristique d'une entité susceptible d'être enregistrée dans une base de données. Comme exemple, un utilisateur (l'entité), son prénom, nom et adresse (ses attributs).

- **Association**

Une association définit le lien qui relie les différentes entités. Par exemple l'association entre un client et sa commande.

- **Base de données**

Une Base de données permet de stocker de manières structurées ou à l'état brut.

- **Cardinalité**

La cardinalité dans une association entre une entité A et B définit le nombre de A pour lesquels un B existe et inversement pour l'autre sens. La cardinalité peut aller de 0 à 1, 0 à N, 1 à N ou N à N. par exemple, une réservation ne peut avoir que 1 utilisateur tandis que 1 utilisateur peut avoir 0 à N réservation.

- **Clé primaire**

La clé primaire est l'attribut de la base données dont le contenu est unique et change à chaque enregistrement de la table.

- **Clé étrangère**

La clé étrangère est l'attribut qui contient la référence à une donnée connexe (la clé primaire).

- **CSS**

*Cascading Style Sheets* est un langage de programmation qui décrit la présentation de documents HTML. Il est utilisé lors de la conception de site web.

- **Entité**  
Une entité est un sujet en rapport avec le thème de la base de données et concernant lequel des données sont enregistrées. Exemple, un utilisateur, une réservation.
- **Hachage**  
Le hachage d'un mot de passe est un processus de transformation d'un mot de passe en chaîne de caractères unique et fixe, à l'aide d'un algorithme de hachage cryptographique.
- **HTML**  
Hypertext Markup Language est un langage permettant d'écrire de l'hypertexte et de structurer une page web. Il est souvent utilisé en concordance avec les langages CSS et JavaScript.
- **JavaScript**  
JavaScript est un langage de programmation utilisé dans le développement web. Il permet de rendre les pages web interactives. Il est fréquemment utilisé avec les langages HTML et CSS.
- **Modèle conceptuel de données (MCD)**  
Le MCD est la représentation visuelle simplifiée de la totalité des données d'une base de données. Le MCD permet une compréhension facilitée du type et des relations entre les données.
- **Modèle logique de données (MLD)**  
Le modèle logique de données est la représentation textuelle du MCD. Il permet d'implémenter la base de données en transcrivant le MCD en instruction SQL.
- **Modèle physique de données (MPD)**  
Le MPD permet d'avoir une représentation graphique de la base de données.
- **MySQL**  
MySQL est un SGBD. Il permet la gestion de la base de données.
- **PHP**  
Hypertext Preprocessor est un langage principalement utilisé pour le développement de site web dynamique via un serveur web, mais est également fonctionnel en utilisation local.
- **Pop-up**  
Fenêtre qui s'ouvre automatiquement sur un site web.
- **SGBD**  
SGBD signifie système de gestion de base de données. C'est un logiciel permettant de stocker, partager et gérer les données dans une base de

---

données en garantissant la qualité, la pérennité et la confidentialité des informations.

- **SQL**

Structured Query Language est langage permettant d'exploiter des bases de données. Il permet de rechercher, de modifier, d'ajouter et de supprimer des données dans une base de données.

- **Table**

Une table, dans une base de données, est un ensemble de données organisées sous forme d'un tableau dans lequel, les colonnes correspondent à des catégories d'information et les lignes à des enregistrements.

### 3 Réalisation

#### 3.1 Dossier de réalisation

##### 3.1.1 **Gestion de la base de données**

###### 3.1.1.1 UWamp

**UWamp** est la version portable de **Wamp**.

UWamp permet de créer un environnement local qui me permet de développer mon application web dans un environnement contrôlé UWamp utilise *MYSQL* pour gérer la base de données, le serveur *Apache* pour gérer les requêtes web et *PHP* pour les scripts.

Cela permet de réaliser des sites web dynamiques et de les tester en local en toute sécurité.

Pour la création de la base de données que j'ai nommée *db\_EquiPlanner*, j'ai utilisé le code SQL proposé par looping suite à la création du MCD/MLD. Je l'ai directement inséré dans l'onglet SQL de *phpMyAdmin*. J'ai dû modifier le type de donnée pour les identifiants pour indiquer que les **COUNTER** étaient des **INT AUTO\_INCREMENT** pour chaque table ainsi que dans la table **RESOURCE** l'attribut **AVAILABLE** a été spécifié que c'était un **BOOLEAN**. Une fois ces étapes réalisées, ma base de données était créée avec l'intégralité de ses tables.

###### 3.1.1.2 Connexion à la base de données

Un des points techniques évalué durant ce projet est d'avoir une connexion sécurisée à la base de données via PDO et utilisateur MySQL dédié. J'ai donc créé un utilisateur pour ne pas utiliser le compte *root*.. De cette manière, l'utilisateur ne possède pas les droits admin sur la base de données et cela réduit grandement le risque de sécurité.

Voici l'utilisateur :

---

Utilisateur : mbayogr  
Mot de passe : M12345

### 3.1.1.3 Créations des membres du club

Une fois la base de données créée, j'ai pu créer les utilisateurs. Pour cette étape, je me suis rendu dans la table *USER* sous l'onglet insérer. Depuis cet onglet j'ai pu rentrer les informations spécifiques aux membres qui sont :

- Le prénom
- Le nom
- L'adresse mail
- Le mot de passe

Inutile de rentrer leur identifiant étant donné que l'attribut **user\_id** est en **AUTO\_INCREMENT**.

Le pattern pour le mot de passe et l'adresse mail des utilisateurs est le suivant :

Pour le mot de passe :

Leurs initiales en majuscules suivi de 12345

Pour l'adresse mail :

Leur prénom commençant par une majuscule . leur nom commençant par une majuscule @ equi.com.

Exemple pour l'utilisateur John Doe :

Adresse mail = [John.Doe@equi.com](mailto:John.Doe@equi.com)

Mot de passe = JD12345

### 3.1.1.4 Ajout des ressources

Pour l'ajout des ressources, je me suis rendu dans la table *RESOURCE* depuis laquelle j'ai pu définir le **type** des ressources (ex. terrain, ballon) le **nom** des ressources et leur **disponibilité** (oui / non).

Exemple pour un ballon de foot :

Type = ballon

Nom = Ballon de foot

Disponibilité = oui

### 3.1.2 MVC

#### 3.1.2.1 Explication MVC

MVC veut dire **Modèle Vue Contrôleur**. C'est une architecture logicielle qui est utilisée lorsque l'on souhaite séparer les différentes responsabilités d'une application web, ce qui permet finalement d'avoir une organisation du code optimale.

- **Modèle (Model)**

Le modèle gère les données et la logique métier de l'application web.

- Il gère la connexion et interagit avec la base de données
- Il contient les fonctions pour récupérer, enregistrer et modifier les données.

- **Vue (View)**

La vue est responsable de l'affichage de l'application. C'est la partie visible par l'utilisateur.

- Elle affiche les données envoyées par le contrôleur.
- Elle ne contient aucune logique métier.

- **Contrôleur (Controller)**

Le contrôleur fait les liens entre le modèle et la vue.

- Il récupère les interactions de l'utilisateur.
- Il appelle les méthodes du modèle pour réaliser le traitement des données.
- Il sélectionne la bonne vue à afficher avec les données nécessaires.

#### 3.1.2.2 Pourquoi utiliser MVC ?

- La **lisibilité** : Chaque partie joue un rôle prédéfini.
- La **maintenance** facilitée : il est possible de modifier l'interface sans toucher à la logique.
- La **réutilisabilité** : Les modèles et les vues peuvent être réutilisés dans plusieurs parties de l'application web.

### 3.1.2.3 Architecture MVC dans mon projet

Mettre en place une structure MVC est un des points techniques de ce projet.  
Voici donc mon architecture MVC imaginée pour ce projet :

/EquiPlanner

|----/controllers  
|    |-MainController.php

|----/models  
|    |-database.php  
|    |-reservation.php  
|    |-resource.php

|----/views  
|    |-login.php  
|    |-reserve.php  
|    |-history.php  
|    |-ressourceChoice.php  
|    |-home.php

|----index.php

- *Contrôleur*
  - **mainController** est le contrôleur principal qui gère la logique de navigation entre les vues en appelant le modèle correspondant.
- *Modèle*
  - **database.php** gère la connexion à la base de données.
  - **reservation.php** gère les fonctions des réservations (ajout, suppression).
  - **resource.php** gère les ressources disponibles pour la réservation.
- *Vue*
  - **login.php** est la page de connexion.
  - **reserve.php** est la page pour faire les réservations.
  - **resourceChoice.php** est la page pour choisir la ressource souhaitée.
  - **history.php** est la page depuis laquelle l'utilisateur peut voir ses réservations passées, à venir et annulées.
  - **home.php** est la page d'accueil.
- **index.php** est le point d'entrée du site. Il redirige sur le login.

### 3.1.2.4 Schéma MVC

Voici le schéma de mon MVC :

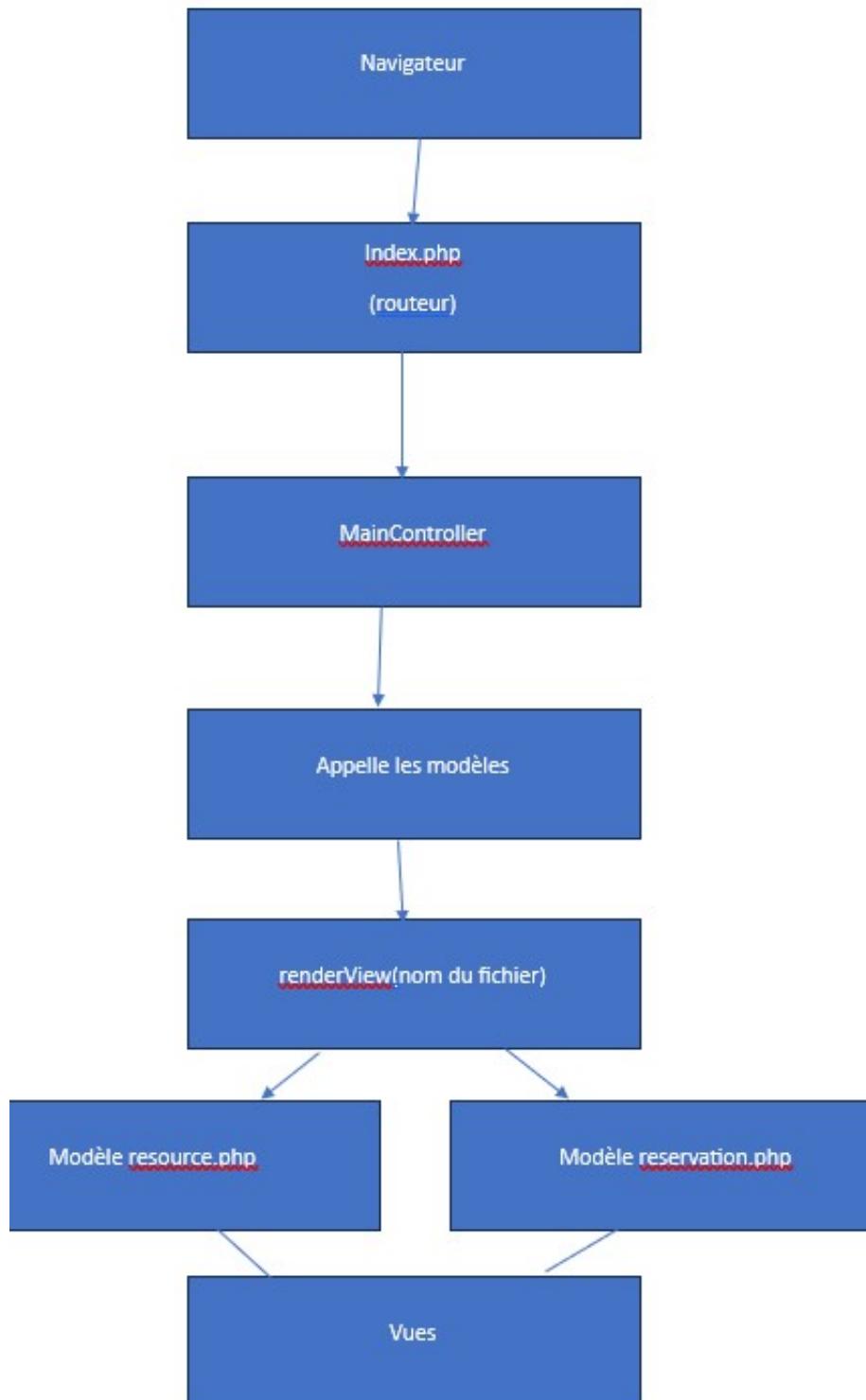


Figure 19 : schéma MVC

### 3.1.3 Connexion à la base de données

Le fichier *database.php* permet de centraliser la gestion de connexion à la base de données. Il permet d'éviter de devoir répéter le code de connexion dans les différents fichiers.

Le fichier contient une classe nommée *Database* contenant elle-même une méthode statique nommée *getConnection()*. Cette méthode retourne l'objet PDO (PHP Data Objects) configuré pour se connecter à la base de données MySQL.

### 3.1.4 Authentification depuis la page login

Le système de login permet d'identifier un membre du club de sport à l'aide de son adresse mail et de son mot de passe prédéfinis, afin de lui donner accès à la page d'accueil (*home.php*) du site web qui est personnalisée.

#### 3.1.4.1 Etapes du processus de login

- **Affichage du formulaire de connexion (*index.php*)**

La page *index* redirige vers la page *login* qui contient le formulaire de connexion. L'utilisateur doit saisir son adresse mail et son mot de passe qui sont envoyés en POST au contrôleur par l'action de cliquer sur le bouton *se connecter*.

Voici le formulaire de connexion :

The screenshot shows a simple login form titled "Connexion". It features two text input fields: one for "Email" and one for "Mot de passe". Each input field is preceded by its respective label. Below the input fields is a large, dark grey rectangular button with the word "Connexion" written in white. The entire form is set against a light gray background.

Figure 20 : formulaire de connexion

- **Envoie des identifiants au contrôleur (*mainController.php*) et Vérification des identifiants dans la base de données**

Le contrôleur récupère les identifiants, interroge la base de données et vérifie le mot de passe.

- **Création de la session si l'authentification est réussie**

Les données de session qui sont l'identifiant de l'utilisateur et son prénom sont stockées après vérification pour permettre de l'identifier sur d'autres pages.

- **Redirection sur la page d'accueil du site web(*home.php*)**

Une ces étapes passées, l'utilisateur à accès à la page d'accueil qui a un accès limité aux utilisateurs connectés.

- **Affichage d'un message d'erreur en cas d'échec de l'authentification**

Au commencement, j'avais décidé qu'en cas d'échec de l'authentification un message d'erreur était affiché indiquant : *Nom d'utilisateur ou mot de passe incorrect*. Ainsi qu'un bouton retour mais j'ai finalement décidé d'intégrer un script *javascript* à ma page pour qu'une pop-up soit affiché en cas d'erreur.

Voici la pop-up :



Figure 21 : pop-up d'échec d'authentification

o

---

voici le script *javascript* :

```
<script>
function closeModal() {
    const modal = document.getElementById("loginError");
    if(modal) {
        modal.style.display = "none";
    }
}
</script>
```

Figure 22: script de popup erreur

### 3.1.4.2 Sécurité

- **Requêtes préparées**

Les requêtes SQL étant préparées cela permet d'éviter les injections SQL

- **Mot de passe hashé**

Les mots de passe sont stockés avec *password\_hash()* et vérifiés avec *password\_verify()*.

### 3.1.5 Script de hachage de mot de passe

Le script *hashpassword.php* a pour but de sécuriser les mots de passe existants dans la base de données en les convertissant en versions hachées via l'algorithme *Bcrypt* (par défaut avec *password\_hash()* en php).

Le voici :

```

<?php
require_once 'models/database.php';

try {
    // Connexion à la base de données
    $db = Database::getConnection();

    // Récupération de tous les utilisateurs
    $sql = "SELECT user_id, password FROM t_user";
    $result = $db->query($sql);

    while ($row = $result->fetch(PDO::FETCH_ASSOC)) {
        $userId = $row['user_id'];
        $plainPassword = $row['password'];
        // Vérifie si le mot de passe est déjà haché
        $hashInfo = password_get_info($plainPassword);
        if (strlen($plainPassword)<60) {
            $hashedPassword = password_hash($plainPassword, PASSWORD_DEFAULT);

            // Mise à jour en base de données
            $updateSql = "UPDATE t_user SET password = :password WHERE user_id = :id";
            $updateStmt = $db->prepare($updateSql);
            $updateStmt->execute([
                'password' => $hashedPassword,
                'id' => $userId
            ]);

            echo "Mot de passe de l'utilisateur ID $userId haché avec succès.<br>";
        } else {
            echo "Utilisateur ID $userId : mot de passe déjà haché, ignoré.<br>";
        }
    }
} catch (PDOException $e) {
    echo "Erreur : " . $e->getMessage();
}

```

Figure 23 : script de hachage des mots de passe

### 3.1.5.1 Etapes du script

- Connexion à la base de données via la classe *Database*.
- Récupération de tous les utilisateurs avec leurs mots de passe.
- Vérification que le mot de passe ne soit pas déjà haché
- Hachage du mot de passe avec *password\_hash()*.
- Mise à jour du champ *password* dans la base de données.
- Affichage de la confirmation utilisateur par utilisateur.

### 3.1.6 Page du choix de l'équipement

Lorsque l'utilisateur clique sur l'onglet *réserver*, la page *resourceChoice.php* s'affiche. Elle contient les différents types d'équipement mis à disposition pour l'utilisateur.

En voici un aperçu

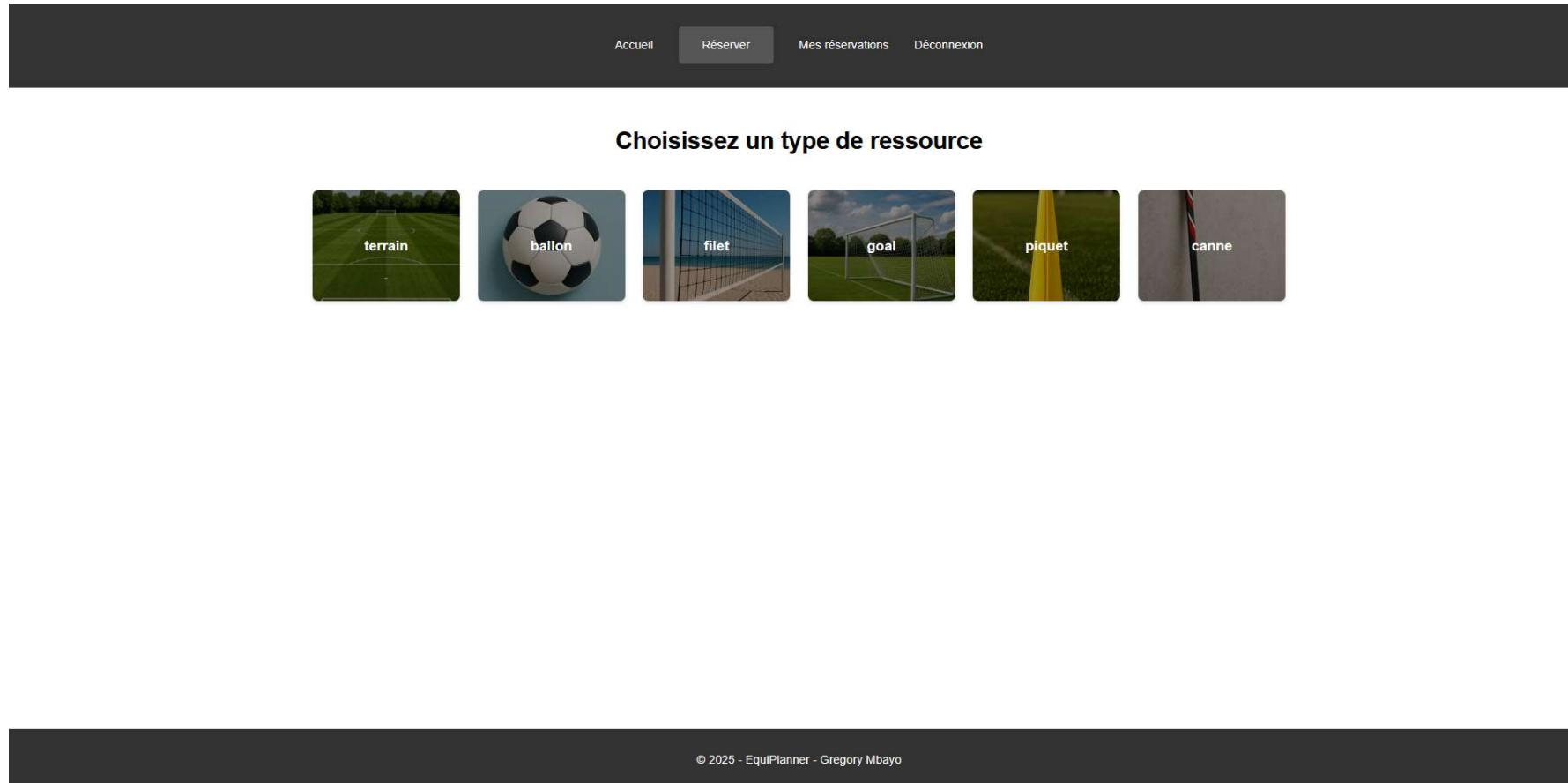


Figure 24 : page du choix de l'équipement

Depuis cette interface, l'utilisateur à la possibilité de sectionner l'équipement qu'il souhaite. Ce qui l'amènera sur la page *reserve.php*, qui permet d'effectuer la réservation.

L'utilisateur a la possibilité de naviguer sur les autres pages depuis le menu bandeau.

### 3.1.6.1 Affichage des équipements

Les ressources sont récupérées depuis la base de données via une requête SQL qui se trouve dans la méthode *getAllType* dans la classe *ResourceModel* du modèle *Resource.php*.

Voici la méthode contenant la requête SQL :

```
public function getAllTypes() {
    try {
        $stmt = $this->db->prepare("SELECT DISTINCT type FROM t_resource");
        $stmt->execute();
        return $stmt->fetchAll(PDO::FETCH_ASSOC);
    } catch(PDOException $e) {
        echo "Erreur SQL : " . $e->getMessage();
        return [];
    }
}
```

Figure 25 : méthode *getAllTypes*

Dans la requête, j'utilise un **SELECT DISTINCT** ce qui permet d'éviter la redondance étant donné que plusieurs équipements peuvent avoir le même type. Exemple un ballon de foot et un ballon de volley sont tous les deux du type "ballon"

### 3.1.7 Page de réservation

Une fois l'équipement choisi, l'utilisateur arrive sur la page de réservation. Sur cette page, l'utilisateur a accès à un calendrier interactif pour les dates ainsi qu'un choix d'heure, ce qui lui permet de filtrer et d'afficher uniquement les ressources disponibles au moment souhaité.

Voici l'affichage de la page :

The screenshot shows a web application interface for resource reservations. At the top, there is a dark header bar with navigation links: Accueil, Réserver (which is highlighted in blue), Mes réservations, and Déconnexion. Below the header, the main title is "Ressources disponibles pour : ballon". There is a search/filter section with fields for Date (jj . mm . aaaa) and Heure (-- : --), and a "Filtrer" button. Below this, six resource items are listed in a grid:

- Ballon de foot (Available): Includes a date/time selector and a "Réserver" button.
- Ballon de volley (Available): Includes a date/time selector and a "Réserver" button.
- Ballon de foot (Available): Includes a date/time selector and a "Réserver" button.
- Boule de pétanque (Available): Includes a date/time selector and a "Réserver" button.
- Boule de pétanque (Available): Includes a date/time selector and a "Réserver" button.
- Boule de pétanque (Available): Includes a date/time selector and a "Réserver" button.

At the bottom of the page, there is a dark footer bar with the copyright notice: © 2025 - EquiPlanner - Gregory Mbayo.

Figure 26 : page de réservation

### 3.1.7.1 Affichage des ressources par date

Pour afficher uniquement les ressources du bon type et disponibles à la bonne date, j'ai créé la méthode `getAvailableResources` dans la classe `ResourceModel` du fichier `Ressource.php` qui exécute une requête SQL que voici :

```
public function getAvailableResources($type, $date, $time) {
    try {
        $stmt = $this->db->prepare("
            SELECT r.* FROM t_resource r
            WHERE r.type = :type
            AND r.available = 1
            AND r.resource_id NOT IN (
                SELECT resource_fk FROM t_reservation
                WHERE date_ = :date AND hour_ = :time AND status = 'active'
            )
        ");
        $stmt->execute([
            'type' => $type,
            'date' => $date,
            'time' => $time
        ]);
        return $stmt->fetchAll(PDO::FETCH_ASSOC);
    } catch (PDOException $e) {
        return [];
    }
}
```

Figure 27 : méthode `getAvailableResources`

Comme dit précédemment, la méthode sert à récupérer les ressources disponibles selon :

- Le type d'équipement (`$type`)
- La date (`$date`)
- L'heure (`$time`)

#### Explication :

- **T\_resource** est la table contenant toutes les ressources.
- **Type = :type** filtre uniquement les ressources du type sélectionné précédemment.
- **NOT IN (...)** permet d'exclure les ressources déjà réservées pour cette date et heure.

#### Seconde requête :

Elle retourne l'identifiant des ressources qui sont déjà réservées pour cette date et heure.

### 3.1.7.2 Message de confirmation

Une fois que l'utilisateur clique sur le bouton *réserver* de la ressource souhaitée, une pop-up réalisée en *JavaScript* s'affiche à l'écran avec un message de confirmation.

La voici :

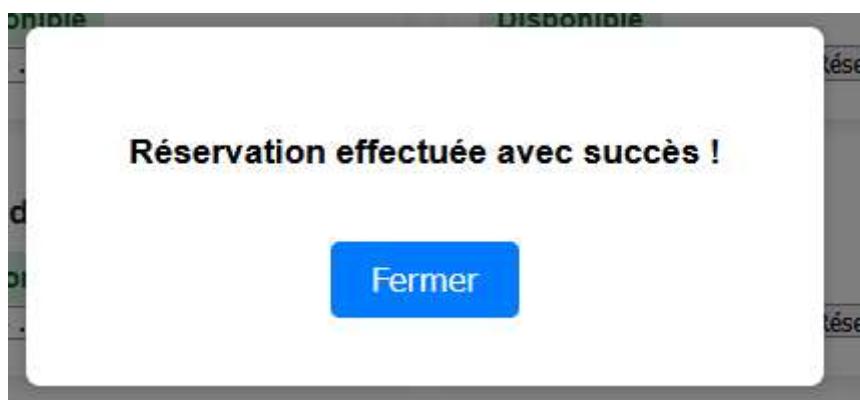


Figure 28 : pop-up de confirmation de réservation

Voici le script *JavaScript* :

```
<script>
document.addEventListener('DOMContentLoaded', function() {
  const modal = document.getElementById('confirmationModal');
  if (modal) {
    const closeBtn = document.getElementById('closeModalBtn');
    closeBtn.addEventListener('click', () => [
      modal.style.display = 'none';
      if (history.replaceState) {
        const url = new URL(window.location);
        url.searchParams.delete('success');
        window.history.replaceState({}, document.title, url.toString());
      }
    ]);
  }
});
</script>
```

Figure 29 : script de la pop-up de confirmation de réservation.

### 3.1.8 Page Mes réservations

L'utilisateur a la possibilité de voir toutes ses réservations personnelles. Pour ce faire, il doit se rendre sur la page "Mes réservations". Depuis cette page, l'utilisateur peut voir toutes ses réservations, peu importe le statut de celles-ci. Il a aussi la possibilité d'annuler les réservations à venir depuis cette page via le bouton *annuler* qui est affiché seulement pour les réservations dont la date n'est pas passée.

Voici la page :

Ressource	Date	Heure	Statut	Action
Ballon de foot	2025-05-27	12:09:00	active	<button>Annuler</button>
Boule de pétanque	2025-05-27	12:09:00	annulée	
Ballon de volley	2025-05-22	12:00:00	terminée	
Goal de foot (7m)	2025-05-21	20:00:00	terminée	
Terrain de unihockey	2025-05-20	23:00:00	terminée	
Terrain de foot	2025-05-01	10:00:00	terminée	

© 2025 - EquiPlanner - Gregory Mbayo

Figure 30 : page Mes réservations

### 3.1.8.1 Affichage des réservations de l'utilisateur

Pour afficher les réservations de l'utilisateur connecté, j'ai ajouté la méthode `getUserReservations` dans la classe `ReservationModel` du fichier `Reservation.php`.

Voici la méthode `getUserReservations` :

```
public function getUserReservations($userId) {
    try {
        $stmt = $this->db->prepare("
            SELECT r.*, t.name AS resource_name
            FROM t_reservation r
            JOIN t_resource t ON r.resource_fk = t.resource_id
            WHERE r.user_fk = :user_id
            ORDER BY r.date DESC, r.hour DESC
        ");
        $stmt->execute(['user_id' => $userId]);
        return $stmt->fetchAll(PDO::FETCH_ASSOC);
    } catch (PDOException $e) {
        return [];
    }
}
```

Figure 31 : méthode `getUserReservations`

Cette méthode est composée d'une requête SQL.

- `.r*` permet de sélectionner toutes les colonnes de la table `t_reservation`.
- `t.name AS ressource_name` permet de récupérer le nom de la ressource liée à chaque réservation.
- `JOIN` fait une jointure avec `t_resource` pour obtenir les informations de la ressource de la réservation.
- `WHERE r.user_fk = :user_id` permet de filtrer les réservations seulement de l'utilisateur connecté.
- `ORDER BY` permet de trier les résultats par ordre chronologique.

Finalement la méthode retourne un tableau contenant :

- L'identifiant de la réservation
- La date
- L'heure
- Le statut
- Le nom de la ressource liée

### 3.1.8.2 Annulation d'une réservation

L'utilisateur a la possibilité d'annuler une réservation s'il le souhaite. Pour réaliser cette action, l'utilisateur doit cliquer sur le bouton "**Annuler**" qui se trouve dans la colonne *Action* du tableau. Ce bouton n'est affiché que si le statut de la réservation est "*active*". Une fois cette action réalisée, le statut de la réservation devient "*annulée*" et un message réalisé en *JavaScript* apparaît durant 4 secondes confirmant l'annulation de la réservation.

Voici le message de confirmation de l'annulation :



Figure 32 : pop-up de confirmation d'annulation de la réservation

Voici la méthode *cancelReservationById* qui se trouve dans la classe *ReservationModel* dans le fichier *Reservation.php* :

```
public function cancelReservationById($reservationId) {
    try {
        $stmt = $this->db->prepare("UPDATE t_reservation SET status = 'annulée' WHERE reservation_id = :id");
        $stmt->execute(['id' => $reservationId]);

        $stmt2 = $this->db->prepare("UPDATE t_resource
            SET available = 1
            WHERE resource_id = (
                SELECT resource_fk FROM t_reservation WHERE reservation_id = :id
            )
        ");
        $stmt2->execute(['id' => $reservationId]);

    } catch (PDOException $e) {
        echo "Erreur lors de l'annulation : " . $e->getMessage();
    }
}
```

Figure 33 : méthode *cancelReservationById*

Cette méthode est composée d'une requête SQL qui permet de **modifier** et non pas **supprimer** la réservation.

- **UPDATE t\_reservation** permet de modifier une ligne dans la table *t\_reservation*.
- **SET status = 'annulée'** permet de changer le statut de la réservation en *annulée*.
- **WHERE reservation\_id = :id** permet de modifier uniquement la ligne correspondante à l'identifiant.

---

Une seconde requête SQL permet de modifier l'état de disponibilité de la ressource liée à la réservation.

- **UPDATE t\_resource** permet de modifier une ligne dans la table *t\_resource*
- **SET available = 1** permet de définir que la ressource est disponible.
- **WHERE resource\_id = (...)** permet de sélectionner l'identifiant de la ressource à la bonne ligne de la table *t\_reservation*.

### 3.1.9 Déconnexion

Pour se déconnecter, l'utilisateur doit cliquer sur l'onglet *déconnexion* dans le menu. Cela redirigera l'utilisateur sur la page de connexion et détruira la session. Pour cela, j'ai créé la méthode *logout()* dans la classe *MainController* du fichier *MainController.php*.

Voici la méthode *logout* :

```
public function logout() {  
    session_unset();  
    session_destroy();  
  
    header('Location: index.php?page=login');  
    exit();  
}
```

Figure 34 : méthode *logout*

- **session\_unset()** ; permet d'effacer toutes les variables enregistrées dans la session.
- **session\_destroy()** ; permet de supprimer la session
- **header(...)** ; redirige l'utilisateur vers la page de connexion.

### 3.1.10 Méthode *renderView*

La méthode *renderView* de la classe *MainController* dans le fichier *MainController.php* sert à afficher les vues tout en transmettant les données nécessaires à la page.

La voici :

```
public function renderView($view, $data = []) {  
    extract($data);  
    require_once __DIR__ . '/../views/' . $view . '.php';  
}
```

### 3.1.10.1 Les paramètres

- **\$view** est le nom de la page souhaitée à afficher.
- **\$data** est un tableau associatif contenant les données à afficher sur la page.

### 3.1.10.2 Fonctionnement

- **extract(\$data);** c'est une fonction PHP permettant de convertir chaque clé du tableau (\$data) en variable ce qui permet à la vue d'utiliser directement les variables.
- **require\_once \_\_DIR\_\_ . '/../views' . \$view . '.php';** permet d'inclure le fichier de la vue situé dans le dossier *views* du répertoire de projet

## 3.1.11 Amélioration

### 3.1.11.1 Horloge dynamique

Sur la page *Mes réservations* j'ai décidé d'ajouter une horloge affichant la date ainsi que l'heure actuelle. Ceci permet à l'utilisateur de se situer dans le temps pour ne pas oublier ses réservations. En voici une image :

## Mes réservations

Nous sommes le 28/05/2025 à 08:38

Figure 36 : horloge dynamique

Le temps s'actualise toutes les 60 secondes.  
Cette fonctionnalité a été réalisée avec un script *JavaScript*.

### 3.1.11.2 Images

J'ai décidé d'ajouter des images représentatives des différentes catégories après une discussion avec le chef de projet car cela rend le site plus "user friendly".

Les images ont été générées avec l'aide de chatGPT.  
J'ai décidé de ne pas les intégrer dans la base de données car je ne souhaitais pas avoir à la modifier à la fin de mon projet. J'ai donc ajouté les images dans un dossier *images* et indiqué le chemin dans le code. J'ai aussi mis en place une image par défaut pour que dans le cas où une ressource serait ajoutée au site, elle puisse tout de même posséder une image.

### 3.1.11.3 Affichage du calendrier interactif

A l'origine, le calendrier interactif permettait de sélectionner des dates dans le passé pour faire une réservation. Néanmoins, une réservation faite avec une date antérieure à la date actuelle obtenait immédiatement le statut *terminée*.

Mais pour optimiser le site web, j'ai décidé de faire en sorte que dans le calendrier, les dates déjà passées soient grises et bloquées. L'utilisateur ne peut pas non plus entrer une heure inférieure à l'heure actuelle sous peine de recevoir un message le lui indiquant.

Voici le calendrier :



Figure 37 : Calendrier interactif

Voici le message d'erreur si l'utilisateur essaie d'entrer une heure inférieure :

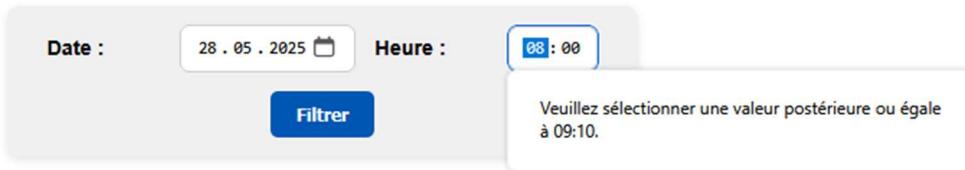


Figure 38 : Message d'erreur de l'heure invalide

Si l'utilisateur arrive tout de même à choisir une date antérieure à la date actuelle en passant par le calendrier disponible sur la ressource directement, un message d'erreur est affiché lors de la réservation. Le voici :



Figure 39 : message d'erreur pour la réservation dans le passé

### 3.1.12 Problèmes rencontrés

#### 3.1.12.1 Erreur de connexion

##### 3.1.12.1.1 Contexte

Lors de ce projet un problème est survenu en voici le contexte :

Le lendemain, après avoir mis en place la pop-up d'erreur d'authentification, il m'était impossible de me reconnecter. La page de login ne s'affichait tout simplement pas. À la place de celle-ci un message venant directement du navigateur indiquant *La page n'est pas redirigée correctement* était affiché.

##### 3.1.12.1.2 Tentatives de solution

J'ai commencé par essayer de me connecter sur un autre navigateur mais le problème subsistait. J'ai donc commencé mes recherches sur internet. La majorité de mes recherches indiquait qu'il fallait supprimer les cookies et vider le cache pour résoudre ce problème, cependant rien de tout cela ne changeait la nature du message.

##### 3.1.12.1.3 Solution finale

J'ai fini par trouver que cela pouvait provenir également d'une boucle infinie de redirection.

J'ai donc cherché dans le code implémenté la veille et j'ai trouvé d'où l'erreur venait. C'était bel et bien une boucle infinie de redirection entre la page *login* et une condition qui renvoyait vers la page *login*.

L'utilisateur était redirigé vers la page *login* après un échec de connexion, mais celle-ci était protégée par une vérification de session qui provoquait une redirection vers la

---

page de *login* à l'infini. J'ai supprimé le code en trop qui était resté de ma première tentative d'affichage d'erreur pour ne laisser place qu'au code de la pop-up. Pour résumer, au lieu de rediriger vers la page de connexion après un échec, la vue est réaffichée avec un message d'erreur.

### 3.1.13 Git

#### 3.1.13.1 Explication Git

Git est un logiciel de gestion des versions décentralisé. Ce logiciel est libre et il est gratuit. Il fut créé en 2005 par Linus Torvalds.

Git permet de stocker un ensemble de fichiers tout en conservant les indications chronologiques des modifications de ces fichiers.

#### 3.1.13.2 Fonctionnement

**Structure d'un dépôt Git.**

**Voici la structure d'un dépôt Git :**

- **Le répertoire git :**  
Il contient l'historique des versions et les métadonnées.
- **Les fichiers de travail :**  
Ce sont les fichiers actuels du projet.
- **La staging area :**  
C'est une zone intermédiaire où l'on prépare les modifications avant de les valider.
- **Le commit :**  
C'est la validation des modifications apportées, qui enregistre un état du projet à un moment donné.

#### 3.1.13.3 Pourquoi utiliser Git

- **La traçabilité :**  
Lorsque l'on travaille avec git, chaque modification apportée au projet est enregistrée avec un identifiant, une date, un auteur et un message du commit. Cela permet de suivre l'évolution du projet ainsi que de revenir facilement à une version antérieure en cas de nécessité.
- **La collaboration :**  
Git facilite grandement le travail en équipe. Plusieurs personnes peuvent travailler ensemble sur le même projet en simultané sans risquer d'altérer le travail effectué par les autres.

- **Les sauvegardes :**

Chaque copie du dépôt contient l'ensemble de l'historique des versions du projet, ce qui assure une sécurité face à la perte de données.

- **Expérimentation sans risque :**

Grâce au système de branches, il est possible de tester de nouvelle fonctionnalité sans pour autant mettre en péril le projet principal.

- **Les plateformes collaboratives**

Git s'utilise généralement avec des plateformes comme *GitHub*, *GitLab* ou *Bitbucket* qui permettent de :

- Héberger des dépôts en ligne
- Gérer les demandes de fusion (pull request)
- Suivre les tickets et tâches
- Automatiser les tests

Git est devenu un outil incontournable dans le développement logiciel grâce à sa puissance, sa flexibilité et son modèle distribué. Il permet une gestion rigoureuse des versions, favorise la collaboration et offre une grande sécurité pour les projets, qu'ils soient personnels ou professionnels.

### **3.2 Description des tests effectués**

#### **3.2.1 Test de l'authentification**

Nom du test :	Test d'authentification	
Identifiant :	T-01	
Objectif :	Se connecter à son compte	
Prérequis :	Avoir son adresse mail et son mot de passe	
Cas de test :		
Etape	Action	Résultat attendu
1	Entrer son adresse mail dans le champ prévu à cet effet	L'adresse mail est affiché en caractères lisible
2	Entrer son mot de passe dans le champ prévu à cet effet	Le mot de passe n'est pas lisible
3	Cliquer sur le bouton se connecter	La page d'accueil s'affiche avec un message personnalisé.

Le test est réussi.

#### **3.2.2 Test d'authentification avec des identifiants erronés**

Nom du test :	Test d'authentification avec des identifiants erronés	
Identifiant :	T-02	
Objectif :	Avoir un message d'erreur	
Prérequis :	-	
Cas de test :		
Etape	Action	Résultat attendu
1	Entrer une adresse mail erronée dans le champ prévu à cet effet	L'adresse mail est affiché en caractères lisible.
2	Entrer un mot de passe erroné dans le champ prévu à cet effet	Le mot de passe n'est pas lisible.

3	Cliquer sur le bouton se connecter	Un message d'erreur indiquant que les identifiants sont incorrects s'affiche.
---	------------------------------------	---

Le test est réussi.

### 3.2.3 Test de hachage des mots de passe

Nom du test :	Test de hachage de mot de passe	
Identifiant :	T-03	
Objectif :	Hacher les mots de passe de la DB	
Prérequis :	Avoir le script de hachage et les utilisateurs avec mot de passe clair dans la DB	
Cas de test :		
Etape	Action	Résultat attendu
1	Lancer le script	Un message pour chaque utilisateur indique que les mots de passe ont été haché.
2	Vérifier sur phpMyAdmin	Les mots de passe sont bien hachés.
3	Ajouter un nouvel utilisateur avec un mot de passe clair	L'utilisateur est créé et son mot de passe est lisible en clair.
4	Relancer le script	Le message pour les utilisateurs ayant déjà le mot de passe haché indique que le hache n'a pas été changé tandis que pour le nouvel utilisateur le message indique que le mot de passe a été haché.
5	Vérifier sur phpMyAdmin	Les mots de passe sont bien hachés.

Le test est réussi.

### 3.2.4 Test de choix de l'équipement

Nom du test :	Test de choix de l'équipement	
Identifiant :	T-04	
Objectif :	Pouvoir choisir le type d'équipement à réserver	
Prérequis :	Être connecter sur le site web	
Cas de test :		
Etape	Action	Résultat attendu
1	Cliquer sur la page <i>réserver</i>	Une page avec les différents types de ressource s'ouvre.
2	Cliquer sur la ressource souhaitée	Une nouvelle page s'affiche depuis l'utilisateur peut réserver uniquement ce type de ressource.

Le test est réussi.

### 3.2.5 Test de la réservation de l'équipement

Nom du test :	Test de la réservation	
Identifiant :	T-05	
Objectif :	Pouvoir choisir le type d'équipement à réserver	
Prérequis :	Être connecter sur le site web	
Cas de test :		
Etape	Action	Résultat attendu
1	Sélectionner la date et l'heure souhaitée	Seules les ressources disponibles sont affichées.
2	Sélectionné la ressource souhaitée	L'utilisateur clique sur la ressource souhaitée.
3	Valider la réservation	Une pop-up apparaît confirmant la réservation.

Le test est réussi.

### 3.2.6 Test d'affichage de mes réservations

Nom du test :	Test de l'affichage de la page <i>mes réservations</i>	
Identifiant :	T-06	
Objectif :	Pouvoir voir toutes les réservations effectuées par l'utilisateur connecté	
Prérequis :	Être connecter sur le site web	
Cas de test :		
Etape	Action	Résultat attendu
1	Cliquer sur la page <i>Mes réservations</i>	Toutes les réservations sont affichées dans un tableau avec le type d'équipement, la date l'heure et le statut.

Le test est réussi.

### 3.2.7 Test d'annulation d'une réservation

Nom du test :	Test d'annulation d'une réservation	
Identifiant :	T-07	
Objectif :	Pouvoir annuler une réservatuion	
Prérequis :	Être connecter sur le site web	
Cas de test :		
Etape	Action	Résultat attendu
1	Cliquer sur la page <i>Mes réservations</i> .	Toutes les réservations sont affichées dans un tableau avec le type d'équipement, la date l'heure et le statut.
2	Checker que les réservations active ont un bouton annulé	Le bouton annulé est bien disponible pour les réservations avec le statut. "active" dans la colonne "action".
3	Cliquer sur le bouton annuler.	Une confirmation est demandée.

4	Confirmé.	Un message de confirmation est affiché / la réservation obtient le statut "annulée" et le bouton 'annuler' n'est plus disponible.
---	-----------	---

Le test est réussi.

### 3.2.8 Test de déconnexion

Nom du test :	Test de déconnexion	
Identifiant :	T-08	
Objectif :	Se déconnecter	
Prérequis :	Être connecter sur le site web	
Cas de test :		
Etape	Action	Résultat attendu
1	Cliquer sur l'onglet "déconnexion" du menu	L'utilisateur est renvoyé sur la page de connexion.

### 3.2.9 Test de statut terminé

Nom du test :	Test de statut terminé	
Identifiant :	T-09	
Objectif :	Le statut change une fois la date et l'heure dépassé	
Prérequis :	Être connecter sur le site web	
Cas de test :		
Etape	Action	Résultat attendu
1	Faire une réservation	La réservation est correctement enregistrée avec le statut active.
2	Attendre que l'heure et la date soit dépassé de 1h	Le statut de la réservation est terminée.

Le test est réussi.

### 3.2.10 Test du calendrier interactif

Nom du test :	Test du calendrier interactif	
Identifiant :	T-10	
Objectif :	Sélectionner une date "Valide" dans le calendrier.	
Prérequis :	Être connecter sur le site web et être sur la page reserve.php	
Cas de test :		
Etape	Action	Résultat attendu
1	Cliquer sur le calendrier interactif.	Le calendrier s'affiche et les dates antécédent la date actuelle sont griseée.
2	Essayer de sélectionner une date déjà passée.	Il est impossible de les sélectionner
3	Sélectionné une date antérieure ou égale à la date actuelle	La date est sélectionnée.
4	Entrer une heure déjà passée.	Un message indiquant qu'il faut sélectionner une heure antérieure à l'heure actuelle est affiché.
5	Entrer une heure valide.	L'heure est affichée
6	Valider le filtre.	Les résultats disponibles sont affiché.

### 3.2.11 Test de l'affichage des images

Nom du test :	Test de l'affichage des images	
Identifiant :	T-11	
Objectif :	Avoir les images affichées sur la page de choix de l'équipement.	
Prérequis :	Être connecter sur le site web	
Cas de test :		
Etape	Action	Résultat attendu

1	Se rendre sur la page de choix de l'équipement.	Les différents équipements sont affichés avec leur images respective.
2	Supprimé une image dans le dossier <i>images</i> et recharger la page web.	La ressource avec l'image supprimée affiche l'image par défaut.

Le test est réussi.

### 3.3 Erreurs restantes

Une erreur que j'ai remarqué à la fin de mon projet est que l'utilisateur peut réserver deux fois le même équipement à la même heure en faisant une manipulation particulière. En effet, une fois le filtre de date et heure effectué, l'utilisateur peut réserver sa ressource. Une fois celle-ci réservée, si il refait un filtre pour une date différente, et qu'il change encore une fois la date et l'heure dans le calendrier propre à la ressource et non du principal, il peut la réserver à nouveau.

#### Solution :

La solution que j'ai trouvée est de supprimer le deuxième calendrier unique de chaque ressource pour empêcher à l'utilisateur de l'utiliser. De cette manière l'utilisateur serait bridé et ne pourra pas de double réservation.

Je n'ai pas appliqué cette correction car je l'ai remarqué à la fin de mon projet et je ne souhaitais pas modifier mon code pour ne pas risquer de provoquer d'autres problèmes de plus grandes gravités.

### 3.4 Liste des documents fournis

- Rapport V.2.0
- Planification V.1.0
- Journal de travail V.2.0

## 4 Conclusions

### 4.1 Bilan des fonctionnalités

Voici la liste des fonctionnalités demandé dans le cahier des charges et implémentées durant ce projet :

- ✓ Mise en place d'une structure MVC : Séparation claire entre les modèles (accès à BDD), les vues (affichage HTML) et les contrôleurs (logique métier).
- ✓ Connexion sécurisée à la base de données via PDO et utilisateur MySQL dédié (pas d'utilisation du compte root).

- 
- ✓ Authentification fonctionnelle avec gestion de session et espace personnel accessible après connexion.
  - ✓ Affichage dynamique des ressources disponibles : seules les ressources non réservées pour la date/heure choisie sont proposées.
  - ✓ Réservation dynamique d'une ressource : enregistrement conditionnel à la disponibilité réelle lors de la soumission.
  - ✓ Liste des réservations pour l'utilisateur connecté avec statut (confirmée/annulée)
  - ✓ Annulation d'une réservation à venir par l'utilisateur connecté : mise à jour du statut de la réservation dans la base de données.

## 4.2 Bilan personnel

Personnellement, j'ai trouvé ce projet très intéressant. Lors de la conception de la maquette graphique du site web et de la réalisation du MCD de la base de données, j'ai apprécié le fait de devoir penser à toutes les fonctionnalités que j'allais implémenter par la suite dans le site. Cette partie m'a beaucoup aidé à me représenter la "structure MVC" du site web que j'allais adopter et déjà avoir une idée des principales requêtes SQL vers la base de données.

### 4.2.1 Points positifs

J'ai apprécié le fait de pouvoir implémenter du *JavaScript* dans ce projet.

### 4.2.2 Points négatifs

J'ai été confronté à un problème lors de la réalisation des réservations. J'ai alors dû analyser les différentes possibilités pour continuer le projet. J'ai donc décidé de modifier mon MCD à ce moment-là pour ajouter un statut aux réservations car j'ai remarqué qu'il manquait une donnée cruciale pour le fonctionnement du site.

## 4.3 Bilan de la planification

Lors de la réalisation de ce projet, j'ai essayé de respecter au maximum ma planification initiale, bien que cela n'ait pas toujours été chose aisée. Afin de ne pas être surpris par un manque de temps, j'ai décidé de modifier l'ordre de certaines tâches durant la dernière semaine, mais je pense pouvoir dire que dans l'ensemble, la planification initiale a été respectée.

### 4.3.1 Bilan du suivi de la maquette graphique

J'ai pu constater que certaines pages du site web ont légèrement dévié de la version imaginée lors de la conception de la maquette graphique. Cela peut s'expliquer par le fait que certaines fonctionnalités ont été ajoutées par la suite. Cependant aucun changement majeur comme l'ajout d'une page n'a été ajouté.

#### **4.4 Si le projet était à refaire**

Si je devais recommencer le projet depuis la phase T1 tout en ayant acquis l'expérience de celui-ci, je passerai plus de temps d'analyse sur quelles fonctions sont nécessaires et dans quels fichiers elles se trouveront. En ayant connaissance des différentes tâches du projet, je changerais également la planification. Je fractionnerais les tâches déjà présentes en des tâches plus spécifiques pour faciliter la conduite du fil conducteur.

#### **4.5 Suites possibles au projet**

Plusieurs améliorations sont envisagées pour la suite du projet. En voici certaines :

- **Possibilité de modification des informations personnelle sur la page d'accueil.**  
L'utilisateur pourrait avoir la possibilité de modifier son mot de passe et/ou son adresse mail depuis la page d'accueil qui est actuellement vide hormis le message de bienvenu.
- **Ajout d'un compte administrateur.**  
Ajouter un compte administrateur qui aurait accès à toutes les réservations et pourrait les modifier.
- **Intégration des images à la base de données**  
Pour optimiser le site, il serait préférable d'intégrer les images à la base de données.
- **Ajout d'images**  
Ajouter des images pour représenter chaque ressources mise à disposition.
- **Réservations regroupées**  
L'utilisateur pourrait effectuer une réservation qui regrouperait plusieurs types d'équipement au lieu de devoir faire 10 réservations pour avoir 10 ballons.
- **Mise en production du site.**  
Si le temps le permet, j'aimerais mettre le site web en production sur le domaine de l'ETML.

### **5 Annexes**

#### **5.1 Table des illustrations**

Figure 1 : planification initiale, semaine 1 .....	6
Figure 2 : planification initiale, semaine 2 .....	6
Figure 3 : planification initiale, semaine 3 .....	7

---

Figure 4 : planification initiale, semaine 4 et 5 .....	7
Figure 5 : page de connexion .....	10
Figure 6 : page d'espace personnel.....	11
Figure 7 : page du choix des ressources.....	12
Figure 8 : Page de réservation .....	13
Figure 9 : page de valuidation .....	14
Figure 10 : page d'historique des réservations .....	15
Figure 11 : page d'annulation des réservations .....	16
Figure 12 : MCD .....	17
Figure 13 : MLD .....	20
Figure 14 : MPD .....	21
Figure 15 : planification détaillée, semaine 1.....	24
Figure 16 : planification détaillée, semaine 2.....	25
Figure 17 : planification détaillée, semaine 3.....	26
Figure 18 : planification détaillée, semaine 4 et 5.....	27
Figure 19 : schéma MVC .....	34
Figure 20 : formulaire de connexion .....	35
Figure 21 : pop-up d'echec d'authentification .....	36
Figure 22: script de popup erreur .....	37
Figure 23 : script de hachage des mots de passe .....	38
Figure 24 : page du choix de l'équipement.....	39
Figure 25 : méthode getAllTypes.....	40
Figure 26 : page de réservation.....	41
Figure 27 : méthode getAvailableRessources .....	42
Figure 28 : pop-up de confirmation de réservation .....	43
Figure 29 : script de la pop-up de confirmation de réservation.....	43
Figure 30 : page Mes réservations .....	44
Figure 31 : méthode getUserReservations .....	45
Figure 32 : pop-up de confirmation d'annulation de la réservation .....	46
Figure 33 : méthode cancelReservationByld .....	46
Figure 34 : méthode logout.....	47
Figure 35 : méthode renderView .....	47
Figure 36 : horloge dynamique .....	48
Figure 37 : Calendrier interactif .....	49
Figure 38 : Message d'erreur de l'heure invalide .....	49
Figure 39 : message d'erreur pour la réservation dans le passé .....	50
Figure 40: journal de travail, semaine 1.....	64
Figure 41 : journal de travail, semaine 2.....	65
Figure 42 : journal de travail, semaine 3.....	66
Figure 43 : journal de travail, semaine.....	67
Figure 44 : journal de travail, semaine 5.....	68
Figure 45 : MCD original.....	69

## **5.2 Résumé du rapport du TPI / version succincte de la documentation**

### **5.2.1 Situation de départ**

Un club sportif souhaite mettre à disposition ses ressources (terrains ou équipements) aux membres sans disposer de solutions informatiques complexes ou coûteuses. Ce projet vise à développer une application web simple, fonctionnelle et intuitive, permettant aux membres d'un club de réserver une ressource disponible, à une date et à un horaire précis.

Les ressources seront prédéfinies dans la base de données. Aucun espace administrateur n'est nécessaire : les membres pourront uniquement consulter et réserver ce qui est déjà disponible.

### **5.2.2 Mise en œuvre**

J'ai débuté le projet par une partie d'analyse pour pouvoir réaliser au mieux ma planification initiale. Une fois ma planification réalisée, je me suis mis à réaliser la maquette graphique de mon site web sur le logiciel *Pencil*. Une fois la maquette graphique réalisée, je pouvais passer à la conception du *MCD*. Pour cette conception, j'ai utilisé l'outil *Looping* qui m'a permis de produire le code *SQL* que j'ai utilisé pour créer la base de données dans *PhpMyAdmin*. J'ai ajouté des données de tests dans la base de données pour m'assurer que le site fonctionnera en situation réelle. À la suite de cela je disposais de tous pour commencer le développement. J'ai donc poursuivi en créant le site web et en y implémentant les différentes fonctionnalités. À chaque fonctionnalité implémentée dans le site web, je réalisais un test pour m'assurer du bon fonctionnement de celle-ci. En parallèle, je réalisais mon journal de travail et mon rapport que j'envoyais au chef de projet ainsi qu'aux deux experts deux fois par semaine.

### **5.2.3 Résultat**

A la fin de mon projet, je me retrouve avec un site web permettant la réservation d'équipements aux membres d'un club de sport.

Le site web permet à un utilisateur de faire une ou plusieurs réservations d'équipements à une date et une heure choisie. L'utilisateur a aussi la possibilité de voir l'historique de toutes ses réservations, et d'annuler les réservations à venir.

### **5.3 Journal de travail**

Voici le journal de travail de la semaine 1 :

Semaine 1			Début: Jeudi 8 mai 2025
Tâche	Durée [1/4 h.]	heures	Explications: qu'est-ce qui se fait et comment ?
			LUNDI 05.05.25
	21		
			MARDI 06.05.25
	24		
			MERCREDI 07.05.25
	12		
			JEUDI 08.05.25
Analyse	6	1h30	Début du TPI, Séance avec l'expert n°1 et le chef de projet pour débuter le tpi, Analyse du cahier des charges dans son entièreté, analyse des outils/logiciel essentiel pour ce TPI.
Analyse	15	3h45	Réalisation de la planification initiale du TPI, Réalisation du journal de travail de la journée, Envoi de la planification initiale aux experts et au chef de projet par mail.
Analyse	6	1h30	Réalisation de la maquette graphique des différentes vue du site web.
			VENDREDI 09.05.25
Documentation	17	4h15	Création du rapport, réalisation de la partie analyse de celui-ci. Avancement du rapport dans la partie réalisation de la maquette graphique et du mcd. Réalisation du journal de travail et envoie des documents aux experts et au chef de projet.
Analyse	10	2h00	Continuation de la maquette graphique du site web. Conception du MCD de la base de donnée avec le logiciel Looping.
Total semaine	111		Max. 111

Figure 40: journal de travail, semaine 1

Voici le journal de travail de la semaine 2

Semaine	2			Début: #VALEUR!
Tâche	Durée [1/4 h.]		Explications: qu'est-ce qui se fait et comment ?	Liens, références, ...
<b>LUNDI 12.05.25</b>				
Analyse	2	00h30	Discussion avec le chef de projet.	
Documentation	13	3h15	Avancement du rapport dans la partie analyse(mcd,mld et mpd). Avancement dans la partie réalisation sur la base de donnée et réalisation du journal de travail.	
Implémentation	6	1h30	Création de la base de donnée dans phpmyadmin et création du git.	
<b>MARDI 13.05.25</b>				
Analyse	8	2h00	Analyse de l'architecture MVC qui sera implémentée.	
Implémentation	8	2h00	Commencement de la structure MVC et ajout de données prédefinis dans la base de données.	
Documentation	8	2h00	Avancement du rapport dans la partie réalisation du MVC et ajout des des données prédefinis dans la base de données. Réalisation de journal de travail et envoie des documents aux experts.	
<b>MERCREDI 14.05.25</b>				
Documentation	12	3h00	Avancement du rapport dans la partie du glossaire et dans l partie réalisation. Réalisation du Journal de travail.	
<b>JEUDI 15.05.25</b>				
Implémentation	23	5h45	Implémentation du système d'authentification et de l'espace personnel sur le site web, Implémentation de l'affichage dynamique des ressources disponibles. Crédit à un script php pour hasher les mots de passe des utilisateurs.	
Tests	2	00h30	Test de la fonctionnalité du système d'authentification.	
Documentation	1	00h15	Réalisation du journal de travail de la journée.	
Analyse	1	00h15	Discussion avec le chef de projet.	
<b>VENDREDI 16.05.25</b>				
Implémentation	12	3h00	commencement de l'implémentation de la réservation dynamique.	
Documentation	15	3h45	Avancement du rapport dans la partie réalisation sur les fonctionnalité ajoutée et dans la partie test. Réalisation du journal de travil et envoie des documents aux experts et au chef de projet.	
Total semaine	111	27h45	<b>Max. 111</b>	

Figure 41 : journal de travail, semaine 2

Voici le journal de travail de la semaine 3 :

Semaine 3			Début: Lundi 19 mai 2025
Tâche	Durée [1/4 h.]	Explications: qu'est-ce qui se fait et comment ?	Liens, références, ...
LUNDI 19.05.25			
Implémentation	15	3h45 Continuation de l'implémentation de la réservation dynamique.	
Analyse	2	00h30 Visite de l'expert n°2. Discussion avec le chef de projet sur l'avancement global du projet et sur comment améliorer l'accueil du site, le message d'erreur d'échec d'authentification	
Documentation	4	1h00 Avancement du rapport dans la partie réalisation de la réservation ainsi que dans la partie test. Réalisation du journal de travail de la journée.	
MARDI 20.05.25			
Implémentation	20	6h45 Continuation de l'implémentation de la réservation dynamique car cette tache ma pris plus de temps que prévu et commencement de la liste des réservations pour l'utilisateur connecté (il reste un problème d'affichage mais la liste des réservation par utilisateur est complète)	
Tests	3	00h45 Test de la fonctionnalité réservation dynamique.	
Documentation	4	00h45 Avancement du rapport dans la partie réalisation de la réservation ainsi que dans la partie test. Réalisation du journal de travail de la journée. Envoie des document aux experts	
MERCREDI 21.05.25			
Implémentation	8	00h45 Correction d'un problème qui empêchait de se connecter au site et réalisation de la page mes réservations	Résolution du problème p.48 du rapport
Documentation	1	0h15 réalisation du journal de travail.	
JEUDI 22.05.25			
Documentation	19	4h45 Avancement du rapport dans la partie réalisation (page de choix de l'équipement, page de réservation et page mes réservations), réalisation du journal de travail.	
Analyse	1	00h15 Réunion le chef de projet, démonstration de l'avancement et discussion sur les tâches restantes	
Implémentation	5	1h15 Implémentation de la fonctionnalité permettant à l'utilisateur d'anuler une réservation	
Tests	2	00h30 Test de la page mes réservations	
VENDREDI 23.05.25			
Documentation	20	5h00 Avancement du rapport dans la partie réalisation en général et la partie Test. Réalisation du journal de travail de la journée. Envoie des document aux experts.	
Tests	3	00h45 Test générale du site pour contrôler qu'il n'y ait pas d'erreur de fonctionnalité et comment pouvoir jouter des améliorations	
Implémentation	4	1h00 Recherches et commencement du fichier Readme.md	
Total semaine	111	27h45 Max. 111	

Figure 42 : journal de travail, semaine 3

Voici le journal de travail de la semaine 4 :

Semaine 4			Début: Lundi 26.05.2025
Tâche	Durée [1/4 h.]	Explications: qu'est-ce qui se fait et comment ?	Liens, références, ...
<b>LUNDI 26.05.25</b>			
Documentation	16	4h00	Avancement du rapport de manière générale en particulier dans la partie réalisation. Réalisation du journal de travail de la journée.
Implémentation	4	1h00	Ajout de l'horloge dynamique qui indique la date et l'heure dans la page mes réservations.
Analyse	1	00h15	Discussion avec le chef de projet sur comment améliorer le site pour le rendre plus userfriendly
<b>MARDI 27.05.25</b>			
Documentation	1	0h15	Réalisation du journal de travail de la journée. Envoie des document aux experts.
Implémentation	20	6h45	Ajout d'image pour le choix des ressource pour rendre le site plus userfriendly, ajout d'une fonction qui empêche l'utilisateur de choisir une date dans le passé + correction du code /ajout de commentaire.
Tests	3	0h45	Test de toutes les fonctionnalités du site web
<b>MERCREDI 28.05.25</b>			
Documentation	8	3h00	Avancement du rapport, partie améliorations, conclusion tests et réalisation du journal de travail.
Implémentation	4	1h00	Modification du fichier readme.
<b>JEUDI 29.05.25</b>			
	27		Ascencion
<b>VENDREDI 30.05.25</b>			
	27		Ascencion
Total semaine	111		Max. 111

Figure 43 : journal de travail, semaine

Voici le journal de travail de la semaine 5 :

Semaine 5			Début: Lundi 02.06.2025
Tâche	Durée [1/4 h.]	Explications: qu'est-ce qui se fait et comment ?	Liens, références, ...
LUNDI 02.06.25			
Documentation	4	1h00	Relecture du rapport + ajout des dernières modifications. Réalisation du journal de travail, envoie du travail final aux expert et chef de projet.
Implémentation	4	1h00	Ajout du travail final au git + modification du fichier readme
Total semaine	111	Max. 111	

Figure 44 : journal de travail, semaine 5

## 5.4 Archives du projet

Voici le mcd original :

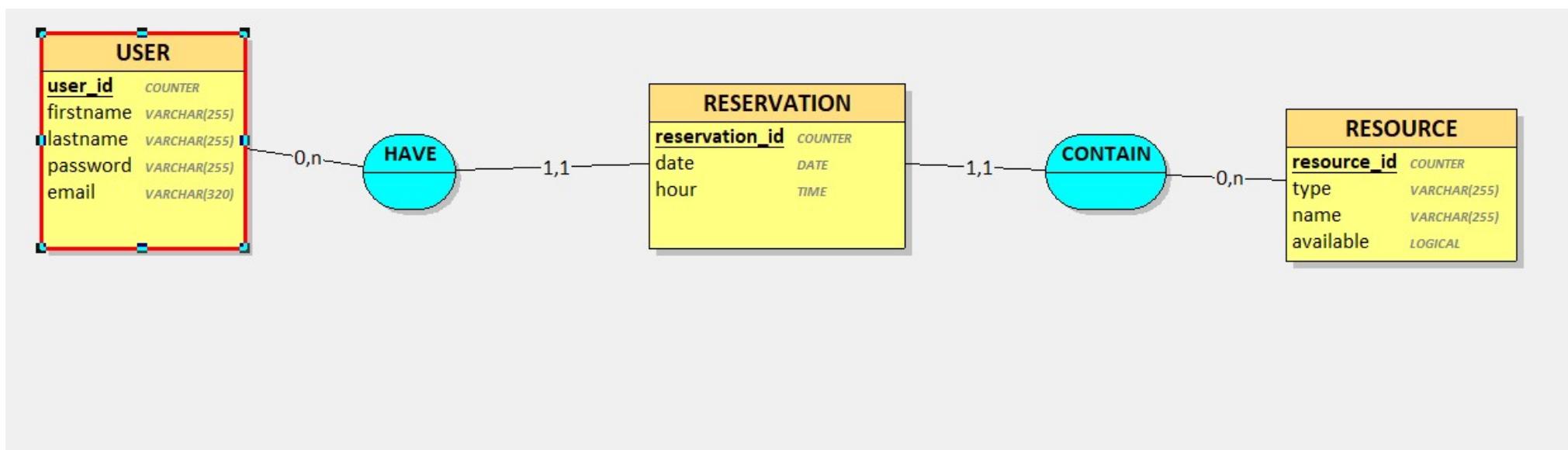


Figure 45 : MCD original