



KuriMediation

(Gestionnaire de cas de médiations)

Chef de projet : M. Dimitrios Lymberis

Expert 1 : M. Nicolas Borboën

Expert 2 : M. Bernard Oberson

Sujet : Développement WEB

Framework : Laravel Breeze – Livewire

Durée : 88 heures

Table des matières

1	Introduction.....	3
1.1	Titre	3
1.2	Introduction.....	3
1.3	Matériels à disposition	3
1.4	Prérequis	3
1.5	Livrables	3
2	Analyse / Conception.....	4
2.1	Objectifs.....	4
2.2	Planification initiale	5
2.3	Méthodologie de travail.....	6
2.4	Environnement	6
2.4.1	Laravel.....	6
2.4.2	MVC.....	6
2.4.3	Eloquent (ORM).....	7
2.4.4	Breeze	7
2.4.5	Tailwind CSS	7
2.4.6	Livewire	7
2.4.7	UwAmp	7
2.5	Conception	8
2.5.1	Schéma fonctionnel	8
2.5.2	Modélisation de la base de données (Méthode MERISE)	9
2.5.3	Justification des types de valeurs dans la modélisation.....	10
2.5.4	Maquettes	11
2.6	Stratégie de test.....	19
2.6.1	Model Factories	19
2.6.2	Seeders	19
2.7	Risques techniques	19
3	Réalisation.....	20
3.1	Mise en place de UwAmp	20
3.2	Installation et mise en place de Laravel 11	20
3.2.1	Connexion à la base de données grâce au fichier « .env »	20
3.3	Mise en place de Breeze	21
3.5	Migrations	22
3.5.1	Table « users »	22
3.5.2	Table « meetings »	23
3.5.3	Table « aftercares »	23
3.5.4	Table « documents »	24
3.5.5	Table « types »	24
3.6	Mise en place des contrôleurs	25
3.7	Mise en place des routes	25
3.7.1	web.php	25
3.7.2	auth.php.....	28
3.7.3	console.php	28
3.8	Authentification	29
3.9	Gestion d'entretiens.....	30
3.9.1	Enregistrement d'un entretien.....	30

3.9.2	Document PDF	33
3.9.3	Statistiques	33
3.10	Gestion des suivis.....	34
3.11	Insertion de documents PDF liés à un entretien	34
3.12	Affichage sous format graphique des statistiques	34
3.13	Export de la page statistique	34
3.14	Exactitudes des valeurs statistiques	34
3.15	Description des tests effectués.....	34
3.15.1	Test de fonctionnement du site.....	34
3.15.2	Test de migrations	34
3.16	Erreurs restantes	34
3.17	Liste des documents fournis	34
4	Conclusions	36
5	Annexes.....	37
5.1	Résumé du rapport du TPI / version succincte de la documentation	37
5.2	Sources – Bibliographie.....	37
5.3	Journal de travail	37
5.4	Manuel d'Installation	37
5.5	Manuel d'Utilisation.....	37
5.6	Archives du projet.....	37

1 Introduction

1.1 Titre

Réalisation d'une application web qui a pour but de gérer les cas de médiations au sein d'un établissement scolaire.

1.2 Introduction

Ce projet consiste à mettre en place une application Web permettant de gérer les cas de médiation. Cette application est destinée aux médiateurs souhaitant effectuer un bilan en fin d'année du temps consacré aux médiations en répertoriant non seulement le nombre d'heures mais également le nombre de cas de médiation selon leurs types.

Le but de cette application est de simplifier et optimiser le processus de suivi des entretiens au sein de l'établissement scolaire pour les membres de l'équipe santé (conseillers d'orientation, médiateurs, psychologues, infirmières, etc..).

1.3 Matériels à disposition

- Serveur hébergé par M. Lymberis
- Visual Code avec des extensions facilitant le développement.
- PHP 8.3
- Composer
- UwAmp pour le serveur de base de données de développement en local
- Connexion wifi.

1.4 Prérequis

- Connaître les notions de la Programmation Orienté Objet
- Savoir faire des requêtes SQL
- Savoir coder en langage PHP.
- Avoir suivi les cours des modules ICH-ICT et réalisé différents projets durant la formation d'informaticien CFC à l'ETML (Modules ICT 101, 104, 105, 133 et 151).
- Savoir générer des graphiques avec Highcharts ou D3.js, chart.js.

1.5 Livrables

- Un rapport de projet
- Un journal de travail
- Le code source avec la base de données.

2 Analyse / Conception

Dans ce projet, plusieurs fonctionnalités de bases apprises durant la formation seront à mettre en place. Des opérations CRUD devront être mises en place pour gérer les entretiens ainsi que les suivis, créer des utilisateurs qui auront le droit ou non d'accéder à des pages administratives, générer des graphiques exportables au format PDF basés sur les données récupérées d'entretiens et de suivis.

Après avoir validé l'application, elle sera déployée sur le serveur WEB mis en place par Monsieur Lymberis.

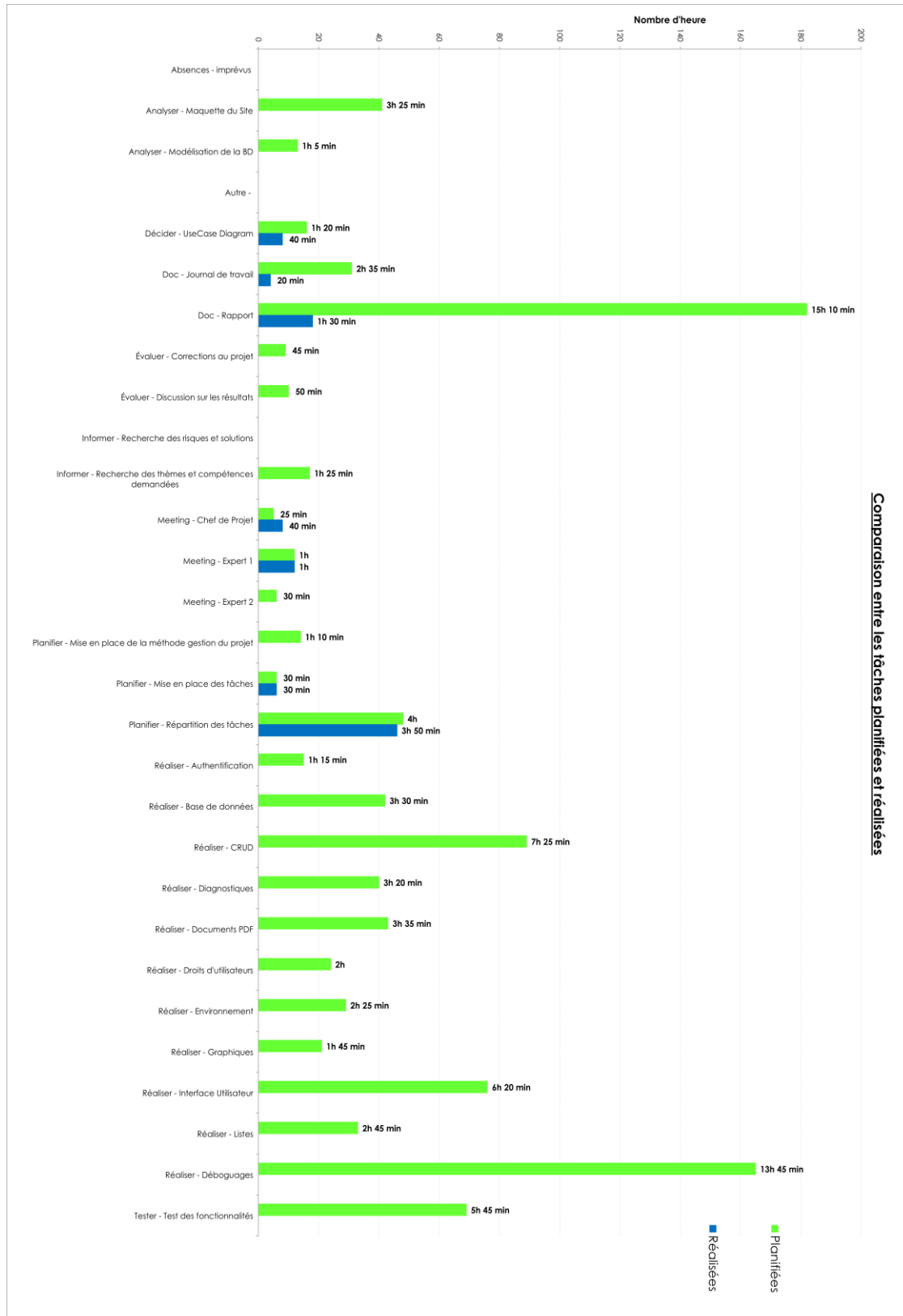
2.1 Objectifs

Ce sous-chapitre est consacré aux objectifs demandés. C'est-à-dire, les fonctionnalités vues avec le maître de stage :

- Opérations CRUD pour les entretiens, les suivis et les années.
- Liste d'entretiens et de ses suivis.
- Filtrage d'entretiens par date, ordre alphabétique ou ordre anti-alphabétique.
- Graphiques représentant la somme de cas de médiations par types, du temps consacré aux entretiens par année, de médiations au total. (Camembert, Histogramme, Barres).
- Téléchargement/Téléversement de fichiers en format PDF.
- Exactitude des valeurs entrées dans les formulaires.
- Ajout/Modification/Suppression de documents liés à un entretien.
- Affichage d'erreurs.

2.2 Planification initiale

Ce chapitre montre la planification du projet. Celui-ci peut être découpé en tâches qui seront planifiées. Il s'agit de la première planification du projet, celle-ci devra être revue après l'analyse. Cette planification sera présentée sous la forme d'un diagramme.



2.3 Méthodologie de travail

Afin de réaliser ce projet de TPI, la méthode qui sera utilisée est la méthode des six pas qui est une méthode très convenable dans le cadre où une limite de temps est imposée.

2.4 Environnement

2.4.1 Laravel

Laravel¹ est un framework PHP open-source, destiné au développement d'applications web en suivant l'architecture MVC (Model – View - Controller) créée par Taylor Otwell. Ce framework fait partie d'un des frameworks PHP les plus populaires grâce à sa simplicité d'utilisation, ses diverses fonctionnalités intégrées et une syntaxe très simpliste. De plus, il évolue à constamment et possède une communauté très active, ce qui fait que la documentation et l'aide en cas de soucis est fortement disponible.

Dans le cas de ce projet, Laravel 11 sera utilisé.

2.4.2 MVC

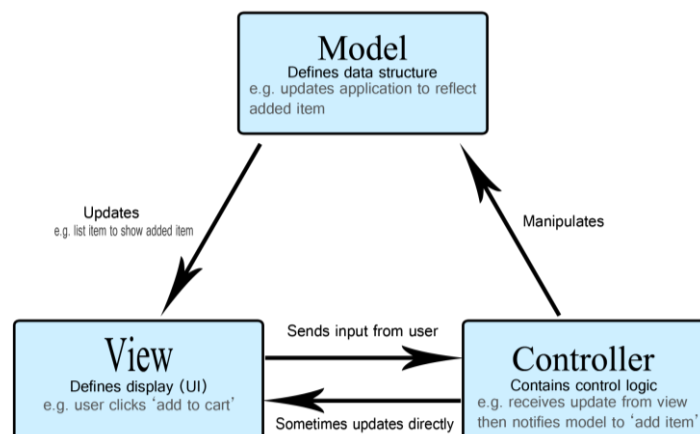
Le modèle MVC² (Model – View – Controller) représente l'architecture centrale utilisé surtout en Laravel. Il vise à simplifier le travail en divisant la charge du travail d'une application web.

Le modèle (Model) représente la partie logique, c'est-à-dire, tout ce qui concerne, la gestion, la récupération, la manipulation ainsi que la sauvegarde de données.

La vue (View) représente la partie visuelle, c'est-à-dire, l'affichage de la page web ainsi que les données. Elle concerne principalement l'interface utilisateur.

Le contrôleur (Controller) représente l'intermédiaire entre le modèle et la vue. Par exemple pour la création d'un utilisateur. Il reçoit les entrées à travers des formulaires puis les traite en interagissant avec le modèle puis, renvoie les résultats à la vue pour qu'ils puissent être affichés.

Cette division des tâches facilite énormément le développement, l'évolution et la maintenance des applications.



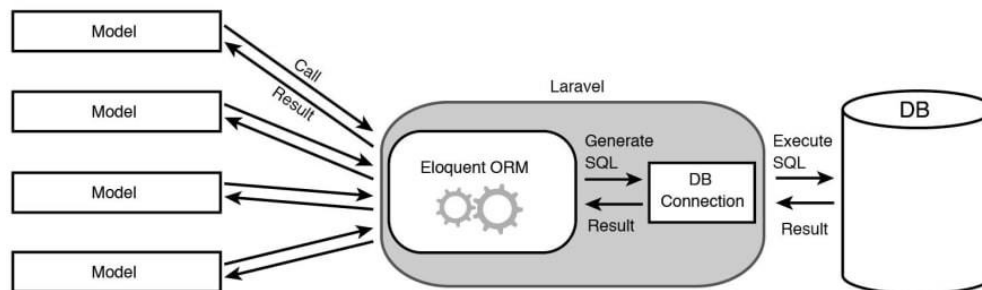
¹ Documentation off

¹ Représentation du fonctionnement du MVC

² Documentation du MVC : <https://openclassrooms.com/fr/courses/4670706>

2.4.3 Eloquent (ORM)

Eloquent³ est un ORM (Object-Relational Mapping) intégré à Laravel. C'est un programme qui se place entre une application web et une base de données afin de permettre aux développeurs de travailler avec les données sous formes d'objet à la place d'écrire des requêtes SQL.



2 Fonctionnement du Eloquent ORM

2.4.4 Breeze

Breeze⁴ ou Laravel Breeze est un composant supplémentaire de Laravel qui offre une implémentation très simpliste et minimaliste de fonctionnalités d'authentification. Il a pour but de servir de point de départ pour les applications web qui ont besoins de fonctionnalités de bases telles que l'enregistrement, la connexion, la modification de données utilisateurs, la suppression de compte, la vérification d'email etc...

Breeze inclut déjà Tailwind CSS qui est un framework CSS rendant la stylisation de vues plus facile et rapide.

2.4.5 Tailwind CSS

Tailwind CSS⁵ est comme mentionné ci-dessus, un framework CSS qui permet de rendre plus facile la stylisation de vue en évitant de passer par un fichier css.

2.4.6 Livewire

Livewire⁶ est un framework full-stack pour Laravel. Cela facilite la création d'interfaces utilisateur dynamiques directement dans Laravel sans avoir nécessairement besoin de connaissances en JavaScript. Par exemple, à la place de rediriger à chaque appui de boutons vers une autre page contenant un formulaire, avec Livewire, c'est possible d'afficher ou masquer le formulaire sur la page.

2.4.7 UwAmp

UwAmp⁷ est un ensemble de logiciels (Uniform, Windows, Apache, MySQL, PHP) dont le but est de créer un environnement de développement web local sur Windows. Il inclut des outils comme Apache, MySQL et PHP.

³ Documentation sur Eloquent : <https://laravel.com/docs/11.x/eloquent>

⁴ Documentation sur Breeze : <https://laravel.com/docs/11.x/starter-kits#laravel-breeze>

⁵ Documentation sur Tailwind CSS : <https://tailwindui.com>

⁶ Documentation sur Livewire : <https://laravel-livewire.com>

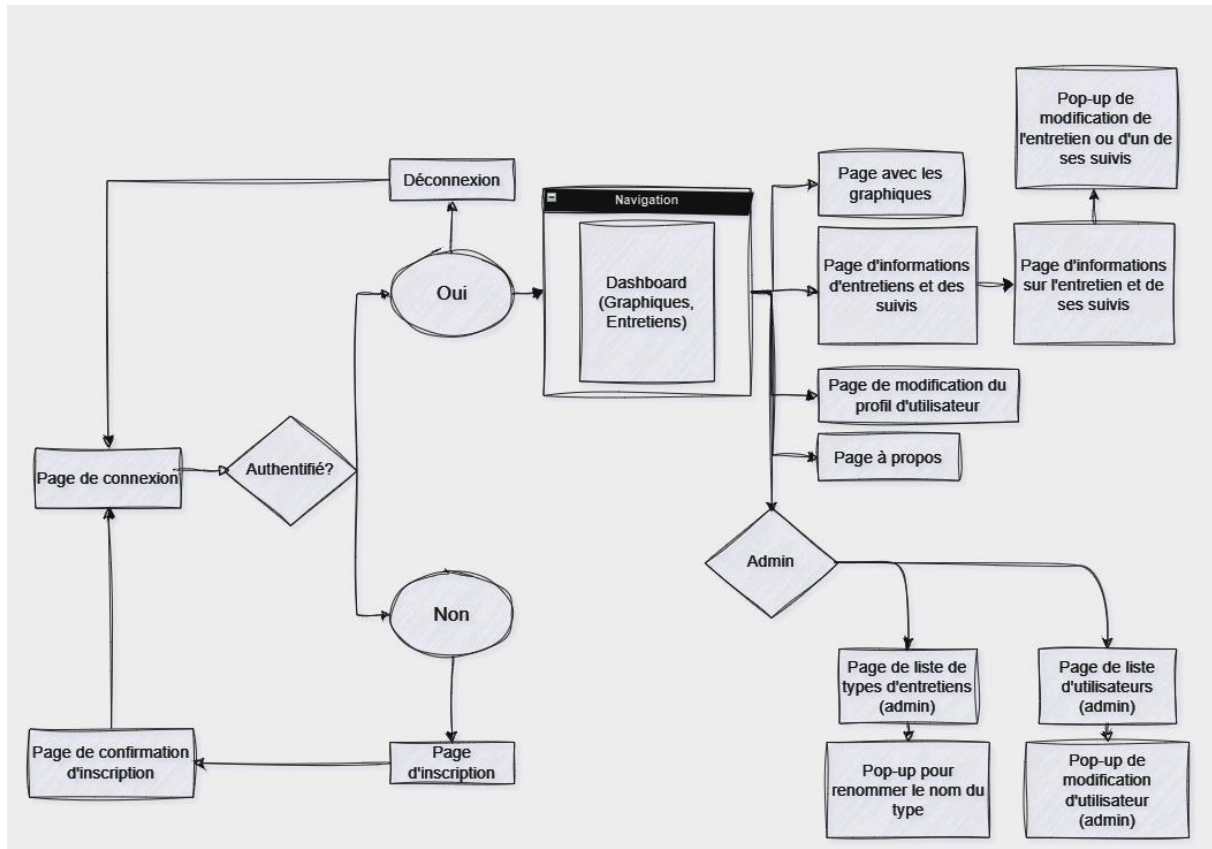
⁷ Documentation sur UwAmp : <https://www.uwamp.com/fr/>

2.5 Conception

Ce chapitre expliquera comment le site fonctionnera à l'aide d'un schéma de principe, des modèles MCD et MLD et des maquettes du site web.

2.5.1 Schéma fonctionnel

Le schéma ci-dessous représente le fonctionnement du site web.



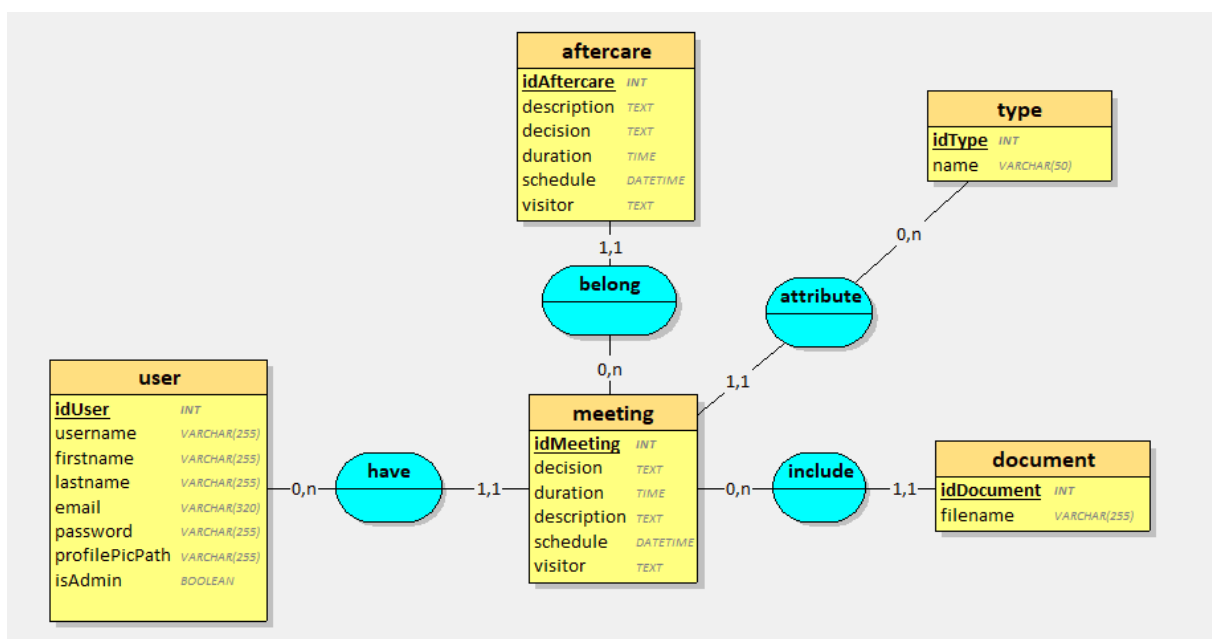
3 Schéma fonctionnel du site

2.5.2 Modélisation de la base de données (Méthode MERISE)

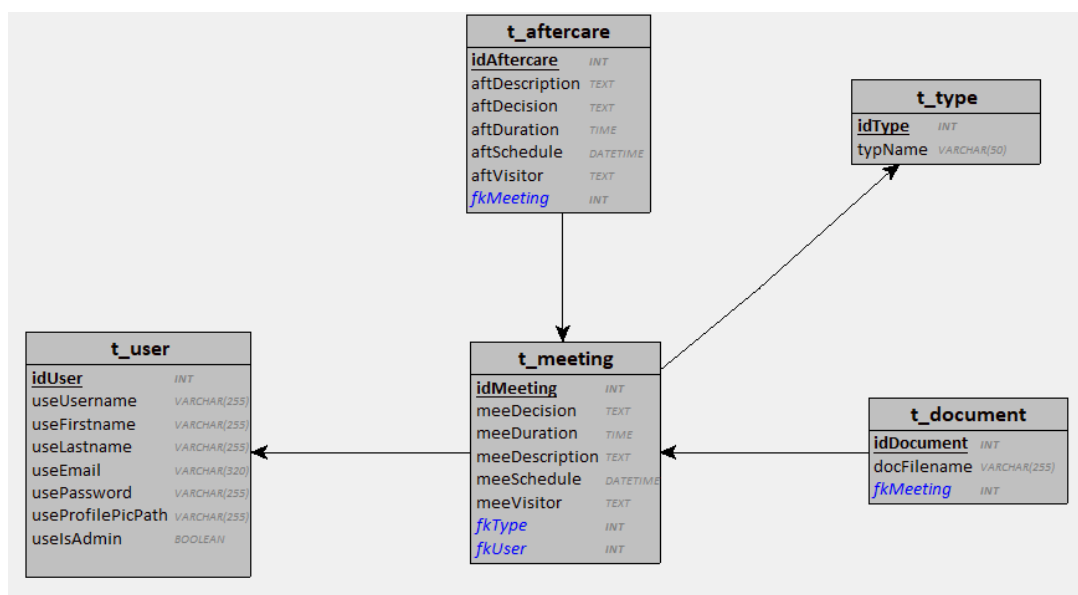
Les MCD et le MLD ci-dessous représentent la structure de la base de données utilisée dans le projet. Ces deux modèles ci-dessous respectent les conventions de nommage de l'ETML.

Cependant, pour une raison de compatibilité avec le framework utilisé, c'est-à-dire, Laravel, les noms des tables et des colonnes ont été adaptés à ses conventions de nommage.

Les entités « year » et « visitor » ont été supprimés car l'année pourrait tout simplement être récupérée dans les propriétés « schedule », puis, « visitor » est remplacé par un attribut dans les entités « meeting » et « aftercare »



4 MCD de l'application (Normes ETML)



5 MLD de l'application (Normes ETML)

2.5.3 Justification des types de valeurs dans la modélisation

Le nom de chaque table et attributs dans le MCD ont été définis en fonction des normes de codage imposées par l'ETML. Cependant dans le MLD, ce sont les normes de Laravel⁸ qui ont été utilisées. La raison de cette différence est qu'il y a des contraintes de compatibilités avec le framework.

2.5.3.1 Entité « USER »

L'entité « USER » est destinée aux utilisateurs. Les types de données et les longueurs des attributs sont définies en se basant sur la table d'utilisateur par défaut de Laravel. La longueur totale d'une adresse électronique est fixée à 320 caractères en raison de la norme RFC 3696⁹.

2.5.3.2 Type « VARCHAR(255) »

La plupart des attributs avec le type VARCHAR() ont une longueur de 255 caractères en raison de la valeur par défaut de Laravel.

2.5.3.3 Type « VARCHAR(300) »

Cette longueur d'attribut est réservée aux emails car elle respecte la norme RFC 3696 comme mentionné dans le chapitre 2.5.3.1.

2.5.3.4 Type « TEXT »

Les attributs ayant ce type est dû au fait que les données entrées seront majoritairement longues. Dans le cas du projet, ce sont des descriptions et ils atteindront une longueur assez grande rapidement.

⁸ Conventions de nommage de Laravel : <https://webdevetc.com/blog/laravel-naming-conventions>

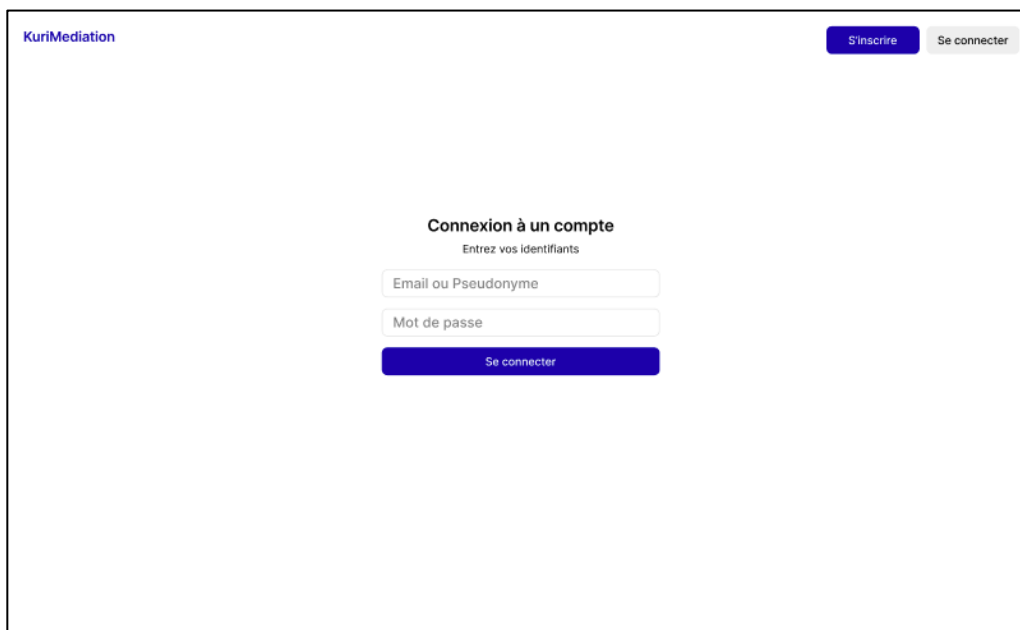
⁹ Norme RFC 3696 : <https://datatracker.ietf.org/doc/html/rfc3696>

2.5.4 Maquettes

Les maquettes suivantes servent à avoir une vision de l'apparence de l'application web.

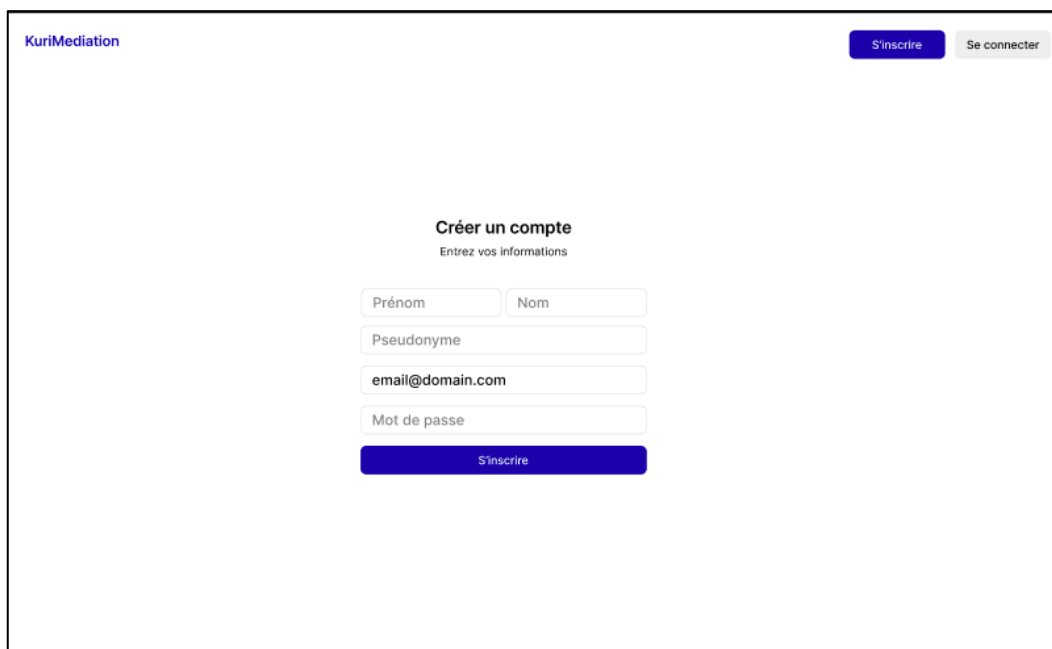
2.5.4.1 Desktop

2.5.4.1.1 Pages de connexion/inscription



Maquette de la page de connexion (Page 6). L'interface est blanche avec un logo 'KuriMediation' en haut à gauche. En haut à droite, il y a deux boutons : 'S'inscrire' (bleu) et 'Se connecter' (gris). Au centre, le titre 'Connexion à un compte' est suivi de 'Entrez vos identifiants'. Il y a deux champs de saisie : 'Email ou Pseudonyme' et 'Mot de passe'. En dessous, un bouton 'Se connecter' est en bleu.

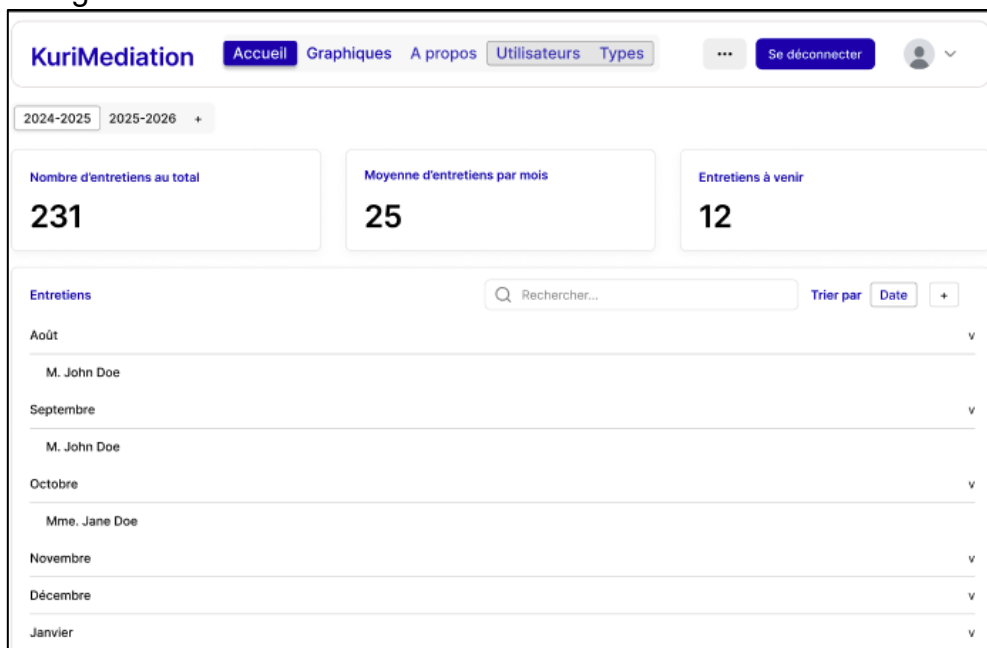
6 Page de connexion



Maquette de la page d'inscription (Page 7). L'interface est blanche avec un logo 'KuriMediation' en haut à gauche. En haut à droite, il y a deux boutons : 'S'inscrire' (bleu) et 'Se connecter' (gris). Au centre, le titre 'Créer un compte' est suivi de 'Entrez vos informations'. Il y a cinq champs de saisie : 'Prénom', 'Nom', 'Pseudonyme', 'email@domain.com' (avec un exemple de format), et 'Mot de passe'. En dessous, un bouton 'S'inscrire' est en bleu.

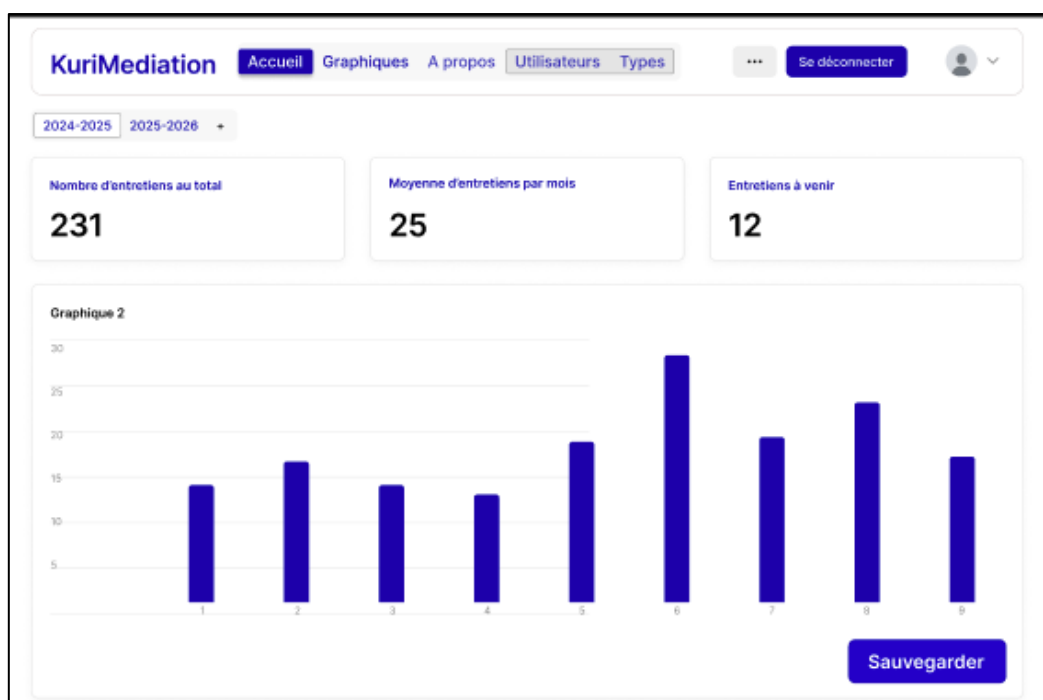
7 Page d'inscription

2.5.4.1.2 Page d'accueil



8 Page d'accueil

2.5.4.1.3 Page de graphiques



9 Page de graphiques

2.5.4.1.4 Page de modification du profil

KuriMediation

Modifier le profil

JojoDoe

Pseudonyme
JojoDoe

Prénom
John

Nom
Doe

Email
joindoe@hotmail.com

Changer le mot de passe

Mot de passe actuel

Nouveau mot de passe

Confirmer le nouveau mot de passe

Confirmer

Sauvegarder Annuler

10 Page de modification du profil

2.5.4.1.5 Page d'information sur l'entretien

KuriMediation Accueil Graphiques A propos

Se déconnecter

Entretien FIN4

Date : 13 septembre 2024 Type : Harcèlement

Entretien

Intervenants
John Doe, Jane Doe, Harry Potter

Durée en min: minutes

Description
Harry Potter subit d'harcèlement par John et Jane Doe.

Décision
John et Jane Doe seront convoqués avec leurs parents le 14 septembre afin de discuter à propos de l'harcèlement.

Ajouter

Sauvegarder

Suivis

Suivi du 17 septembre 2024

Intervenants : Harry Potter, John Doe, Jane Doe

Description : Harry Potter est revenu après avoir reçu une bombe à eau à la figure.

Décision : John et Jane sont en heures de colle pendant 4h.

Durée : 1h16

Suivi du 26 septembre 2024

Ajouter

Supprimer

11 Page d'informations sur un entretien

2.5.4.1.6 Formulaire d'ajout d'un nouvel entretien

The screenshot shows the 'Ajouter un nouvel entretien' (Add new appointment) form. The form is titled 'Ajouter un nouvel entretien' with the subtitle 'Entrez les informations'. It contains the following fields: 'Type intervention' (a dropdown menu), 'Titre' (text input), 'Durée en minutes' (text input), 'Date' (text input), 'Nom des intervenants' (text input), 'Description' (text area), and 'Décisions' (text area). At the bottom, there are two buttons: 'Ajouter' (Add) and 'Annuler' (Cancel). The form is displayed over a calendar interface for the month of September 2024.

12 Formulaire d'ajout d'un nouvel entretien

2.5.4.1.7 Formulaire d'ajout d'un nouveau suivi

The screenshot shows the 'Ajouter un suivi' (Add new follow-up) form. The form is titled 'Ajouter un suivi' with the subtitle 'Entrez les informations'. It contains the following fields: 'Durée en minutes' (text input), 'Date' (text input), 'Nom des intervenants' (text input), 'Description' (text area), and 'Décisions' (text area). At the bottom, there are two buttons: 'Ajouter' (Add) and 'Annuler' (Cancel). The form is displayed over a calendar interface for the month of September 2024. The background shows a table with columns for 'Entretien' and 'Date', and a row for 'Entretien FIN4' with a date of '13 septembre 2024' and a type of 'Harcèlement'.

13 Formulaire d'ajout d'un suivi

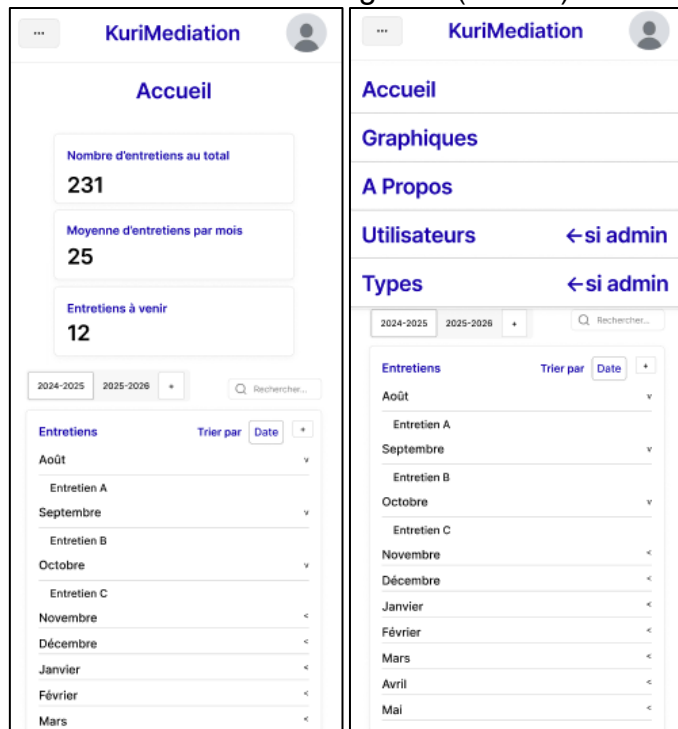
2.5.4.2 Mobile

2.5.4.2.1 Page de connexion/inscription (Mobile)

KuriMediation	KuriMediation
<p>Créer un compte</p> <p>Entrez votre email pour vous inscrire</p> <div><input type="text" value="Prénom"/> <input type="text" value="Nom"/></div> <div><input type="text" value="Pseudonyme"/></div> <div><input type="text" value="email@domain.com"/></div> <div><input type="password" value="Mot de passe"/></div> <div><input type="button" value="Se connecter"/></div> <div><input type="button" value="S'inscrire"/></div> <p><small>En appuyant sur Continuer, vous acceptez nos Conditions de Services et Politique de confidentialité</small></p>	<p>Se connecter</p> <p>Entrez vos identifiants</p> <div><input type="text" value="Email ou Pseudonyme"/></div> <div><input type="password" value="Mot de passe"/></div> <div><input type="button" value="Se connecter"/></div> <div><input type="button" value="S'inscrire"/></div> <p><small>En appuyant sur Continuer, vous acceptez nos Conditions de Services et Politique de confidentialité</small></p>

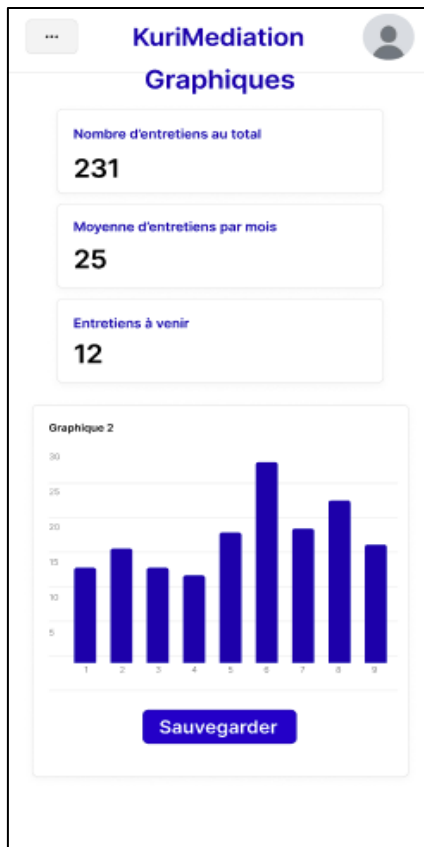
14 Pages d'inscription et de connexion (Mobile)

2.5.4.2.2 Page d'accueil avec barre de navigation (Mobile)



15 Pages d'accueil/Barre de navigation (Mobile)

2.5.4.2.3 Page de graphiques (Mobile)



16 Graphiques (Mobile)

2.5.4.2.4 Page de modification du profil (Mobile)

...

KuriMediation

Modifier le profil

JojoDoe

Pseudonyme

JojoDoe

Prénom

John

Nom

Doe

Email

johndoe@hotmail.com

Changer de mot de passe

Mot de passe actuel

Nouveau mot de passe

Confirmer le nouveau mot de passe

Confirmer

Sauvegarder

Annuler

17 Modification du profil (Mobile)

2.5.4.2.5 Page d'informations sur l'entretien (Mobile)

...

KuriMediation

Entretien FIN4

Entretien

Date : 13 septembre 2024

Type : Harcèlement

Durée en min: minutes

Intervenants

John Doe, Jane Doe, Harry Potter

Description

Harry Potter subit d'harcèlement par John et Jane Doe.

Décisions

Harry Potter subit d'harcèlement par John et Jane Doe.

• preuve1.pdf

• preuve2.pdf

• preuve3.pdf

Ajouter

Suivis

Ajouter

Suivi du 17 septembre 2024

Intervenants : Harry Potter, John & Jane Doe

Description : Harry Potter est revenu après avoir reçu une bombe à eau à la figure.

Décision : John et Jane sont en heures de

Sauvegarder

Supprimer

18 Information sur l'entretien (Mobile)

2.5.4.2.6 Formulaire d'ajout d'un nouvel entretien (Mobile)

The screenshot shows the 'KuriMediation' mobile app interface. At the top, there is a header bar with a menu icon, the app name 'KuriMediation', and a user profile icon. Below the header, the word 'Accueil' (Home) is displayed. The main content area is titled 'Ajouter un nouvel entretien' (Add a new interview) with the subtitle 'Entrez les informations' (Enter the information). The form contains several input fields: 'Type intervention' (a dropdown menu), 'Titre' (text), 'Durée en minutes' (text), 'Date' (text), and 'Nom des intervenants' (text). Below these are two larger text areas for 'Description' and 'Décisions'. At the bottom of the form, there are two buttons: 'Ajouter' (Add) in blue and 'Annuler' (Cancel) in black.

19 Formulaire Entretien (Mobile)

2.5.4.2.7 Formulaire d'ajout de suivi (Mobile)

The screenshot shows the 'KuriMediation' mobile app interface. At the top, there is a header bar with a menu icon, the app name 'KuriMediation', and a user profile icon. Below the header, the word 'Entretien FIN4' is displayed. The main content area is titled 'Entretien' with the subtitle 'Date : 13 septembre 2024'. Below this, there is a section for 'Type : Harcèlement' and 'Durée en min : minutes'. The main content area is titled 'Ajouter un nouvel entretien' (Add a new interview) with the subtitle 'Entrez les informations' (Enter the information). The form contains several input fields: 'Type intervention' (a dropdown menu), 'Titre' (text), 'Durée en minutes' (text), 'Date' (text), and 'Nom des intervenants' (text). Below these are two larger text areas for 'Description' and 'Décisions'. At the bottom of the form, there are two buttons: 'Ajouter' (Add) in blue and 'Annuler' (Cancel) in black.

20 Formulaire Suivi (Mobile)

2.6 Stratégie de test

2.6.1 Model Factories

Pour tester quelques tables dans la base de données, l'outil intégré dans Laravel appelé « Model Factories » qui consiste à définir modèle de création exemplaire repris du modèle Eloquent. Puis, c'est aux Seeders qui initient et remplissent la base de données.

Grâce à cela, on peut vérifier que les migrations sont bonnes et en plus de cela, en générant un grand nombre de données, on peut aussi tester le fonctionnement et la stabilité de l'application web.

2.6.2 Seeders

Comme expliqué ci-dessus, les seeders permettent d'automatiser les insertions de données dans la base de données. Combiné avec les « Model Factories », ils facilitent le développement, les tests ainsi que les démonstrations de l'application.

2.7 Risques techniques

L'intégration quelques éléments tels que l'importation et l'exportation de documents PDF, ainsi que la conversion de pages en PDF, pourraient ralentir le développement. Cela est dû au manque de maîtrise de ces technologies et à un apprentissage inachevé.

3 Réalisation

Ce chapitre représente toute la partie « pratique » de ce projet. A travers ce chapitre, le fonctionnement du code de l'application web « KuriMediation » sera expliqué ainsi que justifié. D'abord, la mise en place de l'environnement de travail sera expliquée puis chaque fonctionnalité sera expliquée.

3.1 Mise en place de UwAmp

Afin d'installer et mettre en place UwAmp, il faut tenir en compte les prérequis, c'est-à-dire, dans notre cas, avoir installé Visual C++ Redistributable 2017 (à partir de PHP 7.2).

Après avoir fait cela, on peut accéder à PHPMyAdmin en allant sur « localhost/mysql/ » depuis le navigateur. Pour créer la base de données, en faisant les migrations sur Laravel, une invitation proposera de la créer si ce n'est pas fait.

3.2 Installation et mise en place de Laravel 11

Afin d'installer Laravel, il est conseillé d'utiliser l'outil « Composer » qui est un outil en ligne de commande. Il suffit de lancer une commande dans un terminal sur VSCode afin d'installer tous les fichiers nécessaires au fonctionnement.

Cette commande permet de créer un projet Laravel : « composer create-project laravel/laravel KuriMediation »

3.2.1 Connexion à la base de données grâce au fichier « .env »

Le fichier « .env » est tout simplement le fichier de configuration de l'application web. Elle contient toutes les valeurs de configuration importantes à la connexion à la base de données.

Cependant, par défaut, le fichier « .env » se nommait « .env.example » car elle contient les informations confidentielles comme le nom de la base de données, l'identifiant, le mot de passe et autres. Il faut donc le renommer et insérer les données. Cela est dû au fait que le fichier fait partie de la liste des fichiers qui sont ignorés par Git.

Afin de pouvoir se connecter à la base de données depuis le projet, il faut remplir quelques champs dans le fichier comme dans l'image sur la prochaine page.

```
APP_NAME=KuriMediation
APP_ENV=local
APP_KEY=base64:/cRepdh51M6S934iy3oozSn8Dj5IPzWd44bHh84YUaM=
APP_DEBUG=true
APP_TIMEZONE=UTC
APP_URL=http://localhost

APP_LOCALE=fr
APP_FALLBACK_LOCALE=fr
APP_FAKER_LOCALE=en_US

APP_MAINTENANCE_DRIVER=file
APP_MAINTENANCE_STORE=database

BCRYPT_ROUNDS=12

LOG_CHANNEL=stack
LOG_STACK=single
LOG_DEPRECATIONS_CHANNEL=null
LOG_LEVEL=debug

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=db_kuriMediation
DB_USERNAME=root
DB_PASSWORD=root
```

21 Fichier de configuration .env

3.3 Mise en place de Breeze

Breeze servira de fournir un point de départ avec une page de bienvenue, une page de connexion et une page d'enregistrement.

Pour installer Breeze, il faut lancer la commande dans le terminal depuis le répertoire racine du projet Laravel : « `composer require laravel/breeze --dev` »

Le terminal proposera quel type de package Breeze souhaité. La version utilisé dans le projet est « `Laravel Breeze with Livewire scaffolding` ». Ce qui signifie que la bibliothèque « `Livewire` » est également ajoutée.

Après avoir lancé la commande, il faut configurer l'authentification en lançant la commande : « `php artisan breeze:install` »

Pour finaliser, il faut installer « `npm` » ensuite le lancer en lançant la commande : « `npm install && npm run dev` ». Il est à mémoriser que la commande « `npm run dev` » devra être lancée à chaque fois pour compiler les ressources frontales et avoir un affichage correct.

Comme cité dans le chapitre 2.4.4, Tailwind CSS est inclus et préconfiguré à l'installation de Breeze.

3.5 Migrations

Pour créer une migration, la commande « php artisan make:migration *nom de la migration* » est lancée.

En raison de compatibilité avec Laravel, le nom des tables et des attributs suit la convention de Laravel.

Les différents sous-chapitres qui suivent représentent les tables créées pour les migrations ainsi que des explications de la longueur des attributs.

3.5.1 Table « users »



```
1 Schema::create('users', function (Blueprint $table) {
2     $table->id();
3
4     $table->string('username');
5     $table->string('firstname');
6     $table->string('lastname');
7     $table->string('email', 320)->unique();
8     $table->string('password');
9     $table->string('profile_pic_path')->default('img/defaultprofilepic.jpg');
10    $table->boolean('isAdmin')->default(false);
11
12    $table->rememberToken();
13    $table->timestamps();
14 });
```

22 Migration de la table "users"

La table « users » est importante car elle contient les informations de l'utilisateur tels que l'email et le mot de passe (password).

L'attribut « email » a une longueur maximum de 320 caractères basé sur la convention RFC 3696 comme mentionné dans le chapitre 2.5.3.1.

L'attribut « isAdmin » sert à déterminer si l'utilisateur est administrateur ou pas. Ce rôle permet d'avoir accès aux pages de modifications d'utilisateurs et de types d'entretiens.

3.5.2 Table « meetings »

```
1 Schema::create('meetings', function (Blueprint $table) {
2     $table->id();
3
4     $table->string('name');
5     $table->text('decision');
6     $table->text('description');
7     $table->integer('duration');
8     $table->dateTime('schedule');
9     $table->text('visitor');
10
11
12     $table->unsignedBigInteger('user_id');
13     $table->foreign('user_id')->references('id')->on('users')->onDelete('cascade');
14     $table->unsignedBigInteger('type_id');
15     $table->foreign('type_id')->references('id')->on('types')->onDelete('cascade');
16
17     $table->timestamps();
18 });
```

23 Migration de la table "meetings"

La table « meetings » est une table sur laquelle se base le sujet du projet. Elle contient les différentes données récupérées durant un entretien. L'attribut « visitor » est de type « TEXT » car elle contient les personnes concernées par le cas de médiations et comme convenu avec M. Lymberis, il est plus simple de faire ainsi que de faire plusieurs attributs pour chaque intervenant.

3.5.3 Table « aftercares »

```
1 Schema::create('aftercares', function (Blueprint $table) {
2     $table->id();
3
4     $table->text('decision');
5     $table->text('description');
6     $table->integer('duration');
7     $table->dateTime('schedule');
8     $table->text('visitor');
9
10     $table->unsignedBigInteger('meeting_id');
11     $table->foreign('meeting_id')->references('id')->on('meetings')->onDelete('cascade');
12
13     $table->timestamps();
14 });
```

24 Migration de la table "aftercares"

La table « aftercares » est similaire à la table « meetings ». Ce qui change est le fait qu'il n'y ait plus de nom car elle appartient à un entretien.

3.5.4 Table « documents »

```
1 Schema::create('documents', function (Blueprint $table) {
2     $table->id();
3
4     $table->string('filename');
5
6     $table->unsignedBigInteger('meeting_id');
7     $table->foreign('meeting_id')->references('id')->on('meetings')->onDelete('cascade');
8
9     $table->timestamps();
10 });
```

25 Migration de la table "documents"

La table « documents » contiendra le nom du fichier PDF ajouté par l'utilisateur ainsi que l'identifiant de l'entretien. Elle permet d'ajouter ou supprimer un document PDF stocké dans le répertoire « public/pdf/*id de l'entretien* ».

3.5.5 Table « types »

```
1 Schema::create('types', function (Blueprint $table) {
2     $table->id();
3
4     $table->string('name');
5
6     $table->timestamps();
7 });
```

26 Migration de la table "types"

La table « types » contient juste un string qui aura comme valeur, le nom du type d'entretien.

3.6 Mise en place des contrôleurs

Dans le projet, il n'y a que des contrôleurs de ressources ont été utilisés. Cependant, il y a aussi des contrôleurs de bases. La différence entre ces deux sont que le contrôleur de ressources contient déjà un modèle contenant les différentes opérations CRUD suivant une convention standard et que le routage est plus simple. Tandis que le contrôleur de base est plus flexible quand on n'a pas besoin d'opérations CRUD.

La mise en place des contrôleurs se fait en lançant la commande : « php artisan make:controller SampleController » ou bien si c'est un contrôleur de ressources il faut ajouter « --resource » à la fin de la commande. Après la commande lancée, un fichier avec le nom du contrôleur sera créé dans le répertoire « app/http/Controllers ».

3.7 Mise en place des routes

Le routage de Laravel fonctionne comme un système central qui gère les requêtes HTTP pour ensuite les diriger vers les contrôleurs ou bien directement vers des vues.

Comme mentionné dans le chapitre précédant, il y a des routes pour les contrôleurs normaux et contrôleurs de ressources.

Pour faire un lien entre les routes et les contrôleurs. Il faut indiquer la classe de la route ainsi que la méthode. Tandis que pour les contrôleurs de ressources, il ne faut pas spécifier la méthode du contrôleur, ainsi que le nom de la méthode à utiliser.

Dans une application Laravel, les routes par défaut sont définies dans plusieurs fichiers. Ces fichiers sont « web.php » qui gère les routes web (cookies, sessions, etc...) et « console.php » qui gère les commandes Artisan.

Cependant, vu que mon projet Laravel est fait avec Breeze. Ce qui fait qu'un autre fichier appelé « auth.php » est ajouté dans le répertoire « routes » (à ne pas confondre avec le fichier auth.php dans le répertoire « config »). Ce fichier sert à gérer les routes d'authentification. Et dans ces routes-là, elles sont séparées en 2 types d'utilisateurs. Ceux qui sont authentifiés (auth) et ceux qui ne sont pas authentifiés (guest).

3.7.1 web.php

Dans ce fichier se trouvent mes différentes routes pour mes contrôleurs (MeetingController, AftercareController et GraphicController).

3.7.1.1 Authentication

```
// Routes for non-authenticated users
Route::middleware(['guest'])->group(function () {
    Route::get('/', function () {
        if (Auth::check()) {
            return redirect()->route('home');
        } else {
            return redirect()->route('login');
        }
    });

    // Route for the login page
    Route::get('/login', function () {
        return view('livewire.pages.auth.login');
    })->name('login');
});
```

27 Fichier web.php

Cette image ci-dessus représente la partie qui redirige vers soit la page d'accueil soit la page de connexion si l'utilisateur n'est pas authentifié.

3.7.1.2 MeetingController

```
// Route for displaying the homepage WITHOUT/WITH existing meetings
Route::get('/home/{year}', [MeetingController::class, 'index'])
    ->name('meeting.index')
    ->middleware(['auth']);
Route::get('/home/{year?}', [MeetingController::class, 'index'])
    ->name('meeting.index')
    ->middleware(['auth']);
Route::post('/meeting/update/{meetingId}', [MeetingController::class, 'update'])
    ->name('meeting.update')
    ->middleware(['auth']);
Route::get('/meeting/destroy/{meetingId}', [MeetingController::class, 'destroy'])
    ->name('meeting.destroy')
    ->middleware(['auth']);
Route::resource('meeting', MeetingController::class)
    ->except('index', 'update', 'destroy');
```

28 Routes pour le MeetingController

Dans la route « meeting.index » le paramètre « year ? » contient un point d'interrogation car il n'est pas obligatoire. Si la valeur est assignée, elle affiche la page avec les valeurs. Dans le cas contraire, elle affiche une page qui ne contient zéro valeur. Cependant une route « /home » a été ajoutée pour des problèmes dans mon contrôleur.

Le contrôleur « MeetingController » est un contrôleur de ressources mais comporte aussi d'autres routes afin qu'ils prennent plus de paramètres que sur le modèle de base. Ce qui explique le « ->except('index', 'update', 'destroy'); ».

La raison pourquoi il y a des autres routes que la route de ressources est pour la mise en forme de l'URL. Car avec la ressource l'url finirait avec « /meeting/*id de l'entretien » alors qu'avec la route personnalisée elle finit comme dans l'image, en « /home/{year?} ».

3.7.1.3 AftercareController

```
// Routes for aftercare resource and store, destroy methods
Route::post('/aftercare/store/{meetingId}', [AftercareController::class, 'store'])
    ->name('aftercare.store')
    ->middleware(['auth']);
Route::get('/aftercare/edit/{meetingId}', [AftercareController::class, 'edit'])
    ->name('aftercare.edit')
    ->middleware(['auth']);
Route::put('/aftercare/update/{meetingId}', [AftercareController::class, 'update'])
    ->name('aftercare.update')
    ->middleware(['auth']);
Route::get('/aftercare/destroy/{aftercareId}', [AftercareController::class, 'destroy'])
    ->name('aftercare.destroy')
    ->middleware(['auth']);
Route::resource('aftercare', AftercareController::class)
    ->except('store', 'edit', 'update', 'destroy');
```

29 Routes pour AftercareController

Comme précisé avec « MeetingController », les routes personnalisées sont que pour l'esthétique de l'URL.

3.7.1.4 GraphicsController

```
// Route that shows the graphics page
Route::get('/graphics/{year?}', [GraphicController::class, 'index'])
    ->name('graphic.index')
    ->middleware(['auth']);
```

30 Route pour GraphicController

3.7.1.5 DocumentController

```
// Routes for the uploading, displaying and deleting the document
Route::post('/document/upload/{meetingId}', [DocumentController::class, 'upload'])
    ->name('document.upload')
    ->middleware(['auth']);
Route::get('/document/show/{meetingId}/{fileName}', [DocumentController::class, 'show'])
    ->name('document.show')
    ->middleware(['auth']);
Route::get('/document/destroy/{meetingId}/{fileName}', [DocumentController::class, 'destroy'])
    ->name('document.destroy')
    ->middleware(['auth']);
```

31 Routes pour DocumentController

Pour ce contrôleur, il est le seul à avoir une route avec un nom différent car elle conçue spécialement pour router à la fonction « upload » qui permet de pourvoir téléverser un fichier PDF au stockage du projet ainsi que d'insérer les données dans la table « documents ».

3.7.2 auth.php

```
Route::middleware('guest')->group(function () {
    Volt::route('register', 'pages.auth.register')
        ->name('register');

    Volt::route('login', 'pages.auth.login')
        ->name('login');

    Volt::route('forgot-password', 'pages.auth.forgot-password')
        ->name('password.request');

    Volt::route('reset-password/{token}', 'pages.auth.reset-password')
        ->name('password.reset');
});

Route::middleware('auth')->group(function () {
    Volt::route('verify-email', 'pages.auth.verify-email')
        ->name('verification.notice');

    Route::get('verify-email/{id}/{hash}', VerifyEmailController::class)
        ->middleware(['signed', 'throttle:6,1'])
        ->name('verification.verify');

    Volt::route('confirm-password', 'pages.auth.confirm-password')
        ->name('password.confirm');
});
```

32 Routes dans le fichier "auth.php"

Ce fichier contient les différentes routes préconfigurées pour les fonctionnalités incluses dans Breeze telles que l'inscription, la connexion, la réinitialisation du mot de passe, la vérification de l'email et plus. Les routes sont groupées dans des « middlewares »

Un middleware un composant intermédiaire qui permet de gérer les requêtes HTTP avant qu'elles n'atteignent les autres routes. Dans le cas de ce projet, cela permet de séparer ce que les utilisateurs authentifiés et non-authentifiés peuvent faire.

3.7.3 console.php

```
Artisan::command('inspire', function () {
    $this->comment(Inspiring::quote());
})->purpose('Display an inspiring quote')->hourly();
```

33 Commandes dans le fichiers "console.php"

Comme mentionné avant, le fichier « console.php » sert à créer des commandes artisan personnalisées.

3.8 Authentification

Comme précisé dans au début du rapport, l'application exploite Laravel avec Breeze. Au moment de l'installation et de la mise en place de Breeze, tout est déjà préconfiguré. Cependant, afin d'aligner les attributs aux besoins de la table « users », plusieurs attributs ont été ajoutées/changées.

```
Schema::create('users', function (Blueprint $table) {
    $table->id();
    $table->string('name');
    $table->string('email')->unique();
    $table->timestamp('email_verified_at')->nullable();
    $table->string('password');
    $table->rememberToken();
    $table->timestamps();
});
```

34 Migrations de la table "users" par défaut

```
1 Schema::create('users', function (Blueprint $table) {
2     $table->id();
3
4     $table->string('username');
5     $table->string('firstname');
6     $table->string('lastname');
7     $table->string('email', 320)->unique();
8     $table->string('password');
9     $table->string('profile_pic_path')->default('img/defaultprofilepic.jpg');
10    $table->boolean('isAdmin')->default(false);
11
12    $table->rememberToken();
13    $table->timestamps();
14 });
```

35 Migration de la table "users" après changements

Comme montrés dans les deux images ci-dessus, les changements sont :

- Le prénom et le nom ont été ajoutés.
- La longueur maximale de caractères pour l'email est de 320.
- La booléenne « isAdmin » permet de définir si l'utilisateur est administrateur.
- Le champ « profile_pic_path » a été ajouté initialement afin de permettre à l'utilisateur de changer mais la fonctionnalité n'a pas été faite.

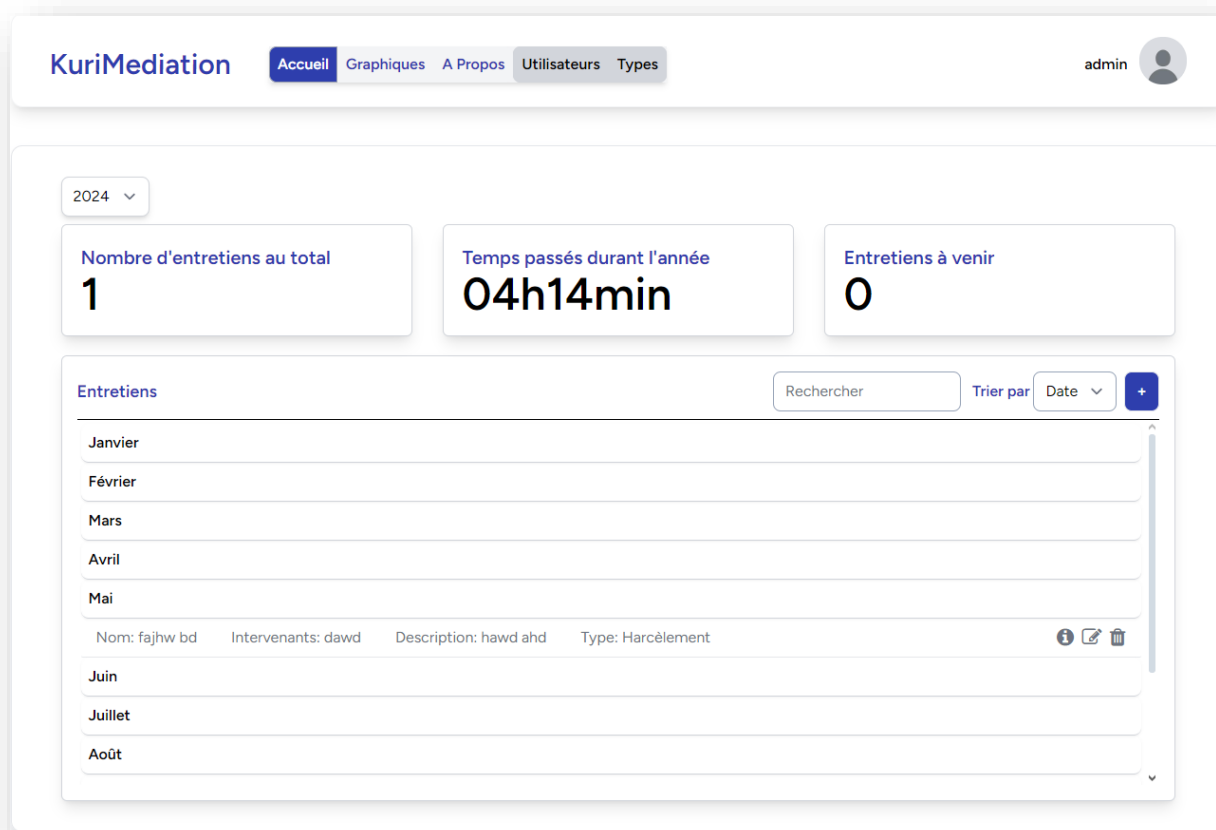
Il ne faut pas oublier que quand on ajoute des attributs il faut aussi aller dans le modèle et ajouter l'attribut dans « \$fillable ». Car, cela permet de remplir de façon simple et rapide plusieurs attributs d'un modèle en utilisant un tableau de données. Si cette étape n'est pas exécutée, les attributs ajoutés ne seront pas pris en compte.

3.9 Gestion d'entretiens

Les entretiens représentent le sujet principal du projet. Ils permettent de mettre en communication les personnes ayant le besoin de s'exprimer avec les médiateurs.

3.9.1 Enregistrement d'un entretien

La page principale contient une liste d'entretiens triés par mois ainsi que trois statistiques représentants : le nombre d'entretiens au total par année, le temps passé au total pour les entretiens et ses suivis et le nombre d'entretiens à venir.



36 Page principale du site

A partir de cette page, un bouton bleu permet d'afficher le formulaire pour enregistrer un nouvel entretien. Avant d'expliquer le fonctionnement du formulaire, le fonctionnement du bouton sera d'abord expliqué.

37 Formulaire pour l'enregistrement d'un entretien

Le bouton bleu est un fichier « livewire » ayant une variable booléenne qui change de valeur à chaque fois que le bouton est pressé. Un « if » vérifiera alors, sa valeur.

```
new class extends Component
{
    public $showForm = false;
    public $types;

    public function toggleForm()
    {
        $this->showForm = !$this->showForm;
    }

    public function mount(){
        $this->types = Type::all();
    }
}; ?>
<div>
    {{-- Button that toggles the visibility of the form --}}
    <button wire:click.prevent="toggleForm" class="ml-2 py-2 px-3 rounded-lg border text-white font-extrabold
    +
    </button>
    @if ($showForm)
    {{-- Le reste du code... --}}
    </div>
```

38 Fonctionnement de la fonction « toggleForm »

Dans le formulaire, on peut constater qu'il manque « la décision » et « la durée » de l'entretien. Cela est tout à normal car le but est que l'entretien puisse être planifié avant, puis la prise de décision et la durée seront remplies à la fin de l'entretien.

```
// Function that creates a new meeting with its parameters
public function store(Request $request){
    // Instanciate a new meeting
    $meeting = new Meeting();
    // Validate the data
    $validatedData = $request->validate([
        'name' => 'required|string|max:255',
        'description' => 'required|string|max:255',
        'schedule' => 'required|date',
        'type_id' => 'required|exists:types,id',
        'visitor' => 'required|string|max:255',
    ]);
    // Define the missing data
    $validatedData['duration'] = 0;
    $validatedData['decision'] = '';
    $validatedData['user_id'] = Auth::id();

    // Inserts the validated data and creates the meeting
    $meeting->create($validatedData);

    // Redirect to the homepage
    return redirect()->route('meeting.index');
}
```

39 Fonction store() dans MeetingController

Les requêtes sont validées d'abord en vérifiant si le champ est obligatoire, son type et sa longueur maximale. Puis, les valeurs non remplies telles que la durée et la décision sont à leur valeur minimale mais pas vides.

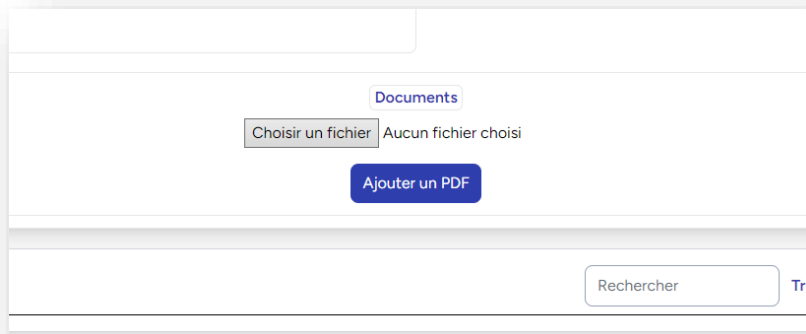
Après avoir créé l'entretien, il est possible d'afficher la page d'information de l'entretien, la page de modification ou bien supprimer l'entretien avec les boutons sur la page principale.



40 Boutons pour gérer l'entretien

3.9.2 Document PDF

Durant ou après un entretien, il arrive souvent qu'on fasse remplir un formulaire par le ou les visiteurs/visiteuses. Afin de pouvoir garder le document dans un endroit sûr et facile à récupérer, il est possible d'insérer des documents PDF. Le fonctionnement sera expliqué dans les prochains points.



41 Bouton pour ajouter un document PDF

3.9.3 Statistiques

Après avoir créé des entretiens il est possible de récupérer les données et en faire des statistiques.

3.9.3.1 Fonction countUserTimeSpent

Cette fonction permet de calculer et retourner le temps consacré aux entretiens et les suivis.

```
/**
 * Function that returns the total amount of time spent for meetings and its aftercares
 * for the user based on the chosen year
 */
public function countUserTimespent($year){
    $total = 0; // Define the total
    $meetings = Meeting::where('user_id', Auth::id()) // Get all meetings by the year
        ->whereYear('schedule', $year)
        ->get();
    // Sum the values into $total
    foreach($meetings as $meeting){
        $total += $meeting->duration;
        $meetingsAftercares = Aftercare::where('meeting_id', $meeting->id)->get();
        foreach($meetingsAftercares as $aftercare){
            $total += $aftercare->duration;
        }
    }
    $hours = floor($total / 60); // Get the lowest value
    $minutes = $total % 60; // Get the minutes from the hours
    return sprintf('%02dh%02dmin', $hours, $minutes);
}
```

41 Fonction countUserTimespent()

3.10 Gestion des suivis

3.11 Insertion de documents PDF liés à un entretien

3.12 Affichage sous format graphique des statistiques

3.13 Export de la page statistique

3.14 Exactitudes des valeurs statistiques

3.15 Description des tests effectués

3.15.1 Test de fonctionnement du site

Afin de vérifier si le serveur se lance, se connecte à la base de données et affiche bien la page par défaut de Laravel, la commande « php artisan serve » est lancée.

Vu que le projet contient aussi le package Breeze qui utilise le framework de CSS « Tailwind CSS », il est important de lancer la commande « npm run dev » sur un deuxième terminal.

3.15.2 Test de migrations

Pour s'assurer que les valeurs des attributs sont correctes, il faut tester en faisant les migrations.

3.16 Erreurs restantes

S'il reste encore des erreurs:

- *Description détaillée*
- *Conséquences sur l'utilisation du produit*
- *Actions envisagées ou possibles*

3.17 Liste des documents fournis

Lister les documents fournis au client avec votre produit, en indiquant les numéros de versions

- *le rapport de projet*
- *le manuel d'Installation (en annexe)*
- *le manuel d'Utilisation avec des exemples graphiques (en annexe)*
- *autres...*

4 Conclusions

Développez en tous cas les points suivants :

- *Objectifs atteints / non-atteints*
- *Points positifs / négatifs*
- *Difficultés particulières*
- *Suites possibles pour le projet (évolutions & améliorations)*

5 Annexes

5.1 Résumé du rapport du TPI / version succincte de la documentation

5.2 Sources – Bibliographie

Liste des livres utilisés (Titre, auteur, date), des sites Internet (URL) consultés, des articles (Revue, date, titre, auteur)... Et de toutes les aides externes (noms)

5.3 Journal de travail

Date	Durée	Activité	Remarques

5.4 Manuel d'Installation

5.5 Manuel d'Utilisation

5.6 Archives du projet

Media, ... dans une fourre en plastique