|  |
| --- |
| 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](#_Toc167794762)

[1.1 Titre 3](#_Toc167794763)

[1.2 Introduction 3](#_Toc167794764)

[1.3 Matériels à disposition 3](#_Toc167794765)

[1.4 Prérequis 3](#_Toc167794766)

[1.5 Livrables 3](#_Toc167794767)

[2 Analyse / Conception 4](#_Toc167794768)

[2.1 Objectifs 4](#_Toc167794769)

[2.2 Planification initiale 5](#_Toc167794770)

[2.3 Méthodologie de travail 6](#_Toc167794771)

[2.4 Environnement 6](#_Toc167794772)

[2.4.1 Laravel 6](#_Toc167794773)

[2.4.2 MVC 6](#_Toc167794774)

[2.4.3 Eloquent (ORM) 7](#_Toc167794775)

[2.4.4 Breeze 7](#_Toc167794776)

[2.4.5 Tailwind CSS 7](#_Toc167794777)

[2.4.6 Livewire 7](#_Toc167794778)

[2.4.7 UwAmp 7](#_Toc167794779)

[2.5 Conception 8](#_Toc167794780)

[2.5.1 Concept 8](#_Toc167794781)

[2.5.2 Modélisation de la base de données (Méthode MERISE) 9](#_Toc167794782)

[2.5.3 Justification des types de valeurs dans la modélisation 10](#_Toc167794783)

[2.5.4 Maquettes 11](#_Toc167794784)

[2.6 Stratégie de test 19](#_Toc167794785)

[2.6.1 Model Factories 19](#_Toc167794786)

[2.6.2 Seeders 19](#_Toc167794787)

[2.7 Risques techniques 19](#_Toc167794788)

[3 Réalisation 20](#_Toc167794789)

[3.1 Mise en place de UwAmp 20](#_Toc167794790)

[3.2 Installation et mise en place de Laravel 11 20](#_Toc167794791)

[3.2.1 Connexion à la base de données grâce au fichier « .env » 20](#_Toc167794792)

[3.3 Mise en place de Breeze 21](#_Toc167794793)

[3.4 Migrations 21](#_Toc167794794)

[3.4.1 Table « users » 21](#_Toc167794795)

[3.4.2 Table « meetings » 22](#_Toc167794796)

[3.4.3 Table « aftercares » 22](#_Toc167794797)

[3.5 Description des tests effectués 22](#_Toc167794798)

[3.5.1 Test de fonctionnement du site 22](#_Toc167794799)

[3.5.2 Test de migrations 23](#_Toc167794800)

[3.6 Erreurs restantes 23](#_Toc167794801)

[3.7 Liste des documents fournis 23](#_Toc167794802)

[4 Conclusions 24](#_Toc167794803)

[5 Annexes 25](#_Toc167794804)

[5.1 Résumé du rapport du TPI / version succincte de la documentation 25](#_Toc167794805)

[5.2 Sources – Bibliographie 25](#_Toc167794806)

[5.3 Journal de travail 25](#_Toc167794807)

[5.4 Manuel d'Installation 25](#_Toc167794808)

[5.5 Manuel d'Utilisation 25](#_Toc167794809)

[5.6 Archives du projet 25](#_Toc167794810)

# Introduction

## Titre

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

## 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..).

## Matériels à disposition

* Serveur hebergé 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.

## 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.

## Livrables

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

# 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.

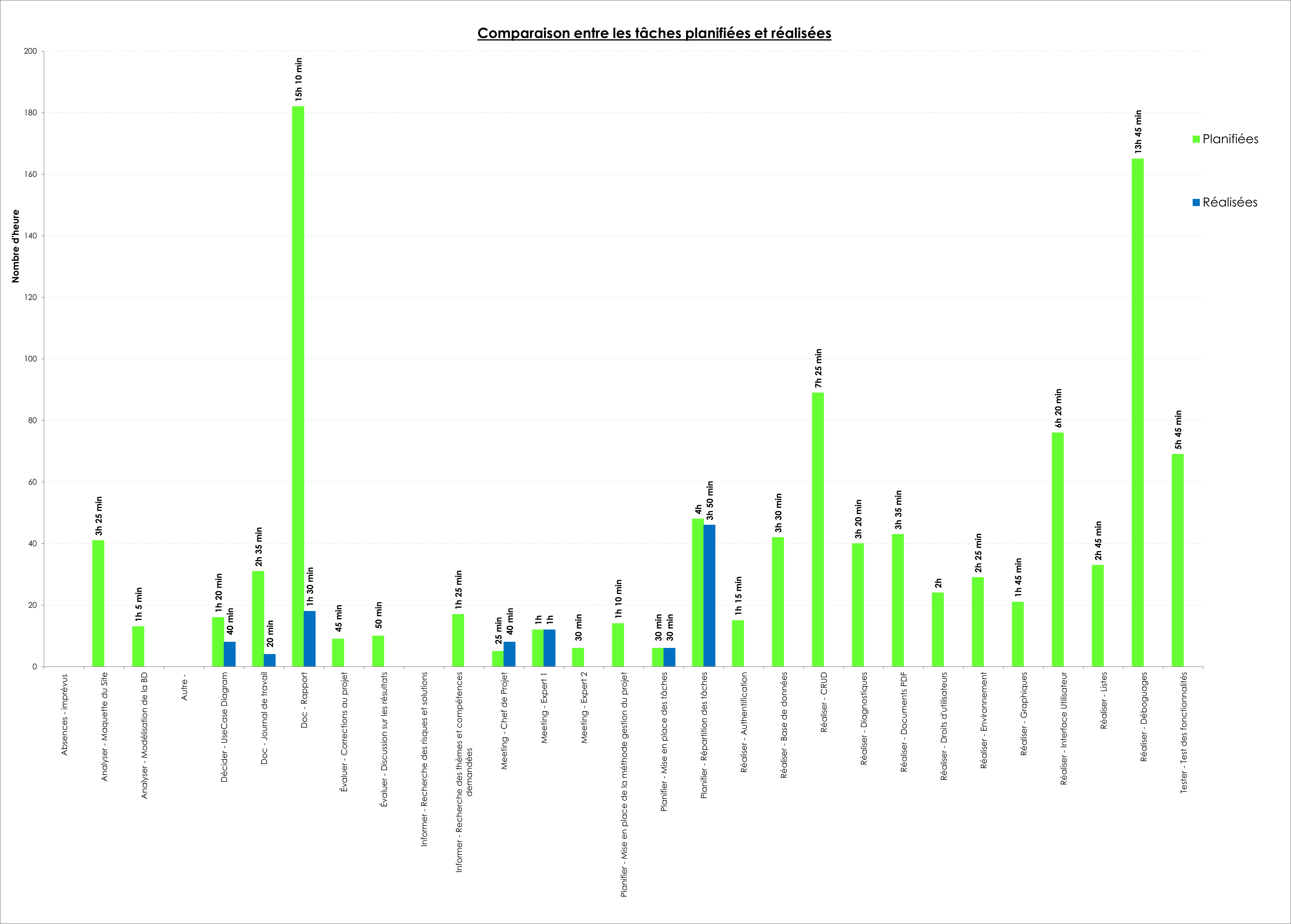
Après avoir validé l’application, elle sera déployée sur le serveur WEB mis en place par Monsieur Lymberis.

## 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.

## 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.

## 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.

## Environnement

### Laravel

Laravel[[1]](#footnote-1) est un framework PHP open-source, destiné au développement d’applications web en suivant l’architecture MVC (Model – View - Controller) crée 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.

### MVC

Le modèle MVC[[2]](#footnote-2) (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 controlleur (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 traites en interagissant avec le modèle puis, renvoie les résultats à la vue pour qu’ils puissent être affichés.

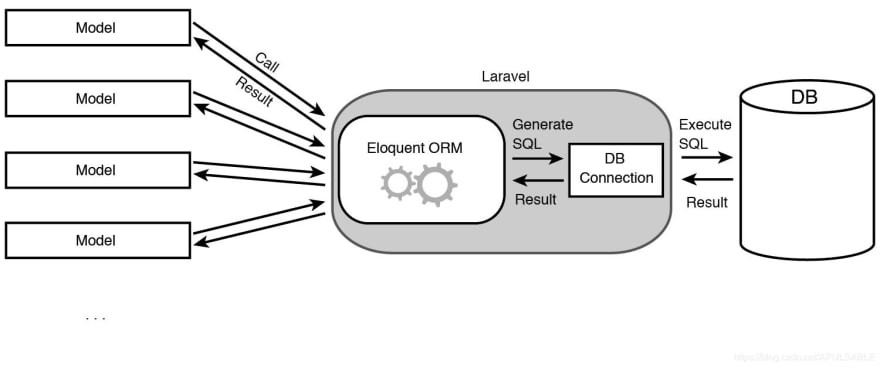
Cette division des tâches facilite énormément le développement, l’évolution et la maintenance des applications.



1 Représentation du fonctionnement du MVC

### Eloquent (ORM)

Eloquent[[3]](#footnote-3) est un ORM (Object-Relatinal 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

### Breeze

Breeze [[4]](#footnote-4)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…

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

### Tailwind CSS

Tailwind CSS[[5]](#footnote-5) 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.

### Livewire

Livewire[[6]](#footnote-6) 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.

### UwAmp

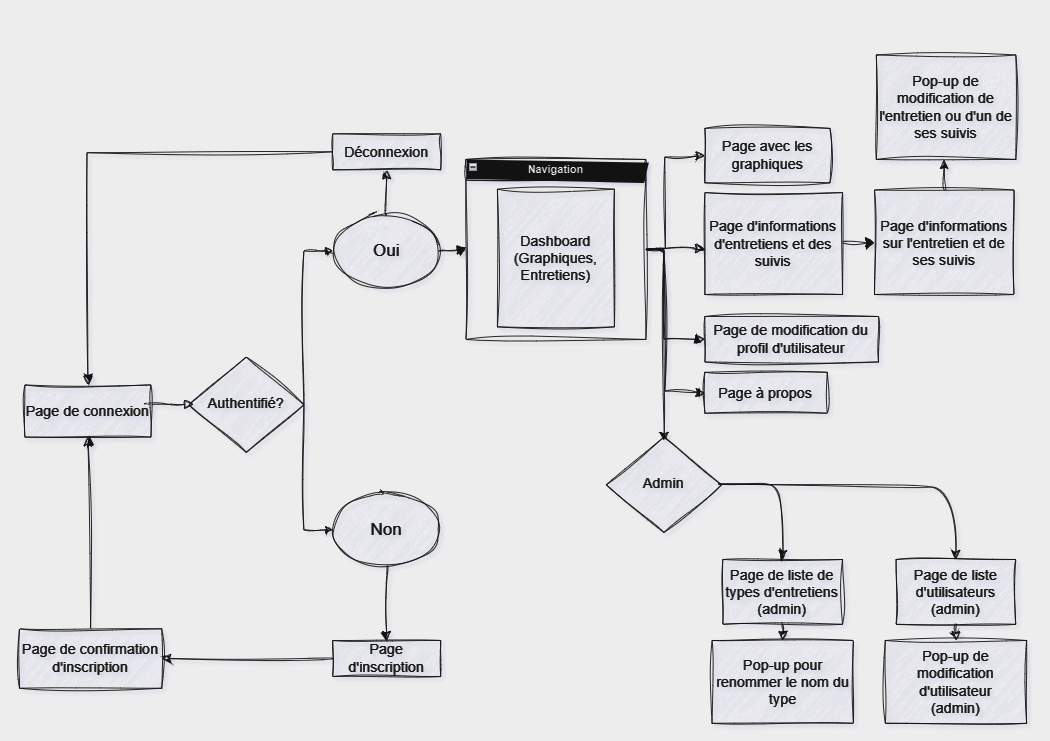
UwAmp[[7]](#footnote-7) est un ensemble de logiciels (Uniform, Windows, Apache, MySQL, PHP) dont le but est de créer un environnement de développement web local sur Windows. Il inclut des outils comme Apache, MySQL et PHP.

## 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.

### Concept

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



3 Schéma fonctionnel du site

### 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.

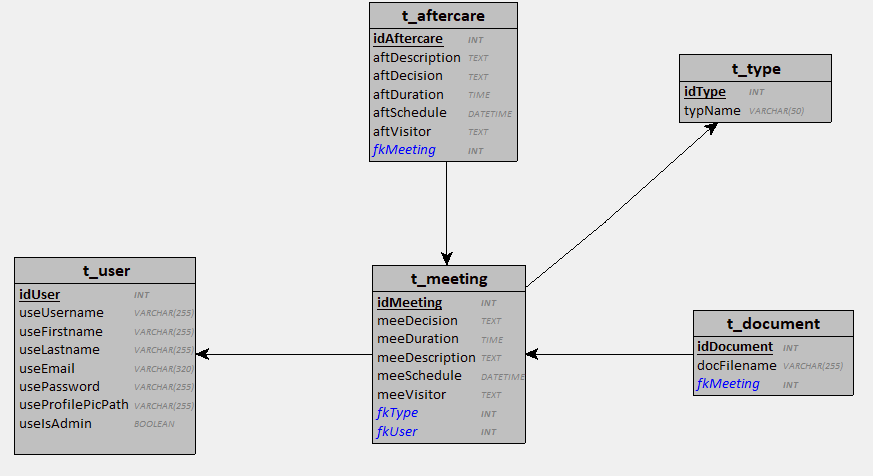
Cependant, pour une raison de compatibilité avec le framework utilisé, c’est-à-dire, Laravel, les noms des tables et des colonnes ont été adaptés à ses conventions de nommage.

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 »

Une image contenant texte, capture d’écran, diagramme, Police

Description générée automatiquement

4 MCD de l'application (Normes ETML)



5 MLD de l’application (Normes ETML)

### 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 Laravel[[8]](#footnote-8) qui ont été utilisées. La raison de cette différence est qu’il y a des contraintes de compatibilités avec le framework.

#### 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 charactères en raison de la norme RFC 3696[[9]](#footnote-9).

#### Type « VARCHAR(255) »

Les attributs avec le type VARCHAR() ont une longueur de 255 charactères en raison de la valeur par défaut de Laravel.

#### 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.

#### 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.

### Maquettes

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

#### Desktop

##### Une image contenant texte, capture d’écran, conception Description générée automatiquementPages de connexion/inscription

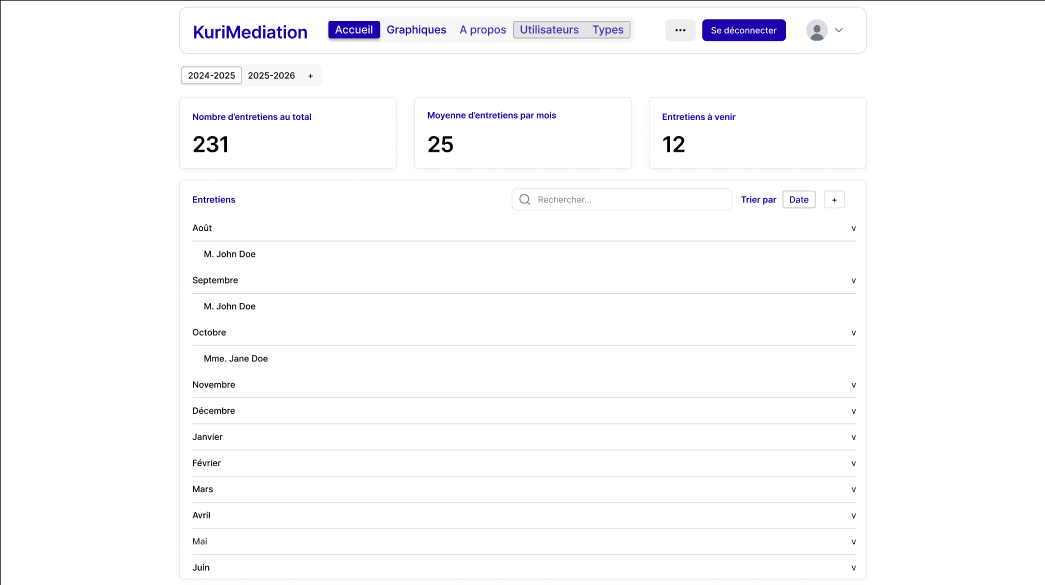
6 Page de connexion

Une image contenant texte, capture d’écran, conception

Description générée automatiquement

7 Page d'inscription

##### Page d’accueil



8 Page d'accueil

##### Une image contenant texte, nombre, logiciel, Police Description générée automatiquementPage de graphiques

9 Page de graphiques

##### Page de modification du profil

Une image contenant texte, capture d’écran, nombre

Description générée automatiquement

10 Page de modification du profil

##### Page d’information sur l’entretien

Une image contenant texte, logiciel, Page web, capture d’écran

Description générée automatiquement

11 Page d'informations sur un entretien

##### Formulaire d’ajout d’un nouvel entretien

Une image contenant texte, capture d’écran, logiciel, nombre

Description générée automatiquement

12 Formulaire d'ajout d'un nouvel entretien

##### Formulaire d’ajout d’un nouveau suivi

Une image contenant texte, logiciel, capture d’écran, Page web

Description générée automatiquement

13 Formulaire d'ajout d'un suivi

#### Mobile

##### Page de connexion/inscription (Mobile)

*Une image contenant texte, capture d’écran, conception

Description générée automatiquementUne image contenant texte, capture d’écran, conception

Description générée automatiquement*

14 Pages d'inscription et de connexion (Mobile)

##### Page d’accueil avec barre de navigation (Mobile)

*Une image contenant texte, capture d’écran, nombre, logiciel

Description générée automatiquementUne image contenant texte, capture d’écran, nombre, logiciel

Description générée automatiquement*

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

##### Page de graphiques (Mobile)

Une image contenant texte, capture d’écran, Police, logiciel

Description générée automatiquement

16 Graphiques (Mobile)

##### Page de modification du profil (Mobile)

##### Une image contenant texte, capture d’écran, logiciel, conception Description générée automatiquement

17 Modification du profil (Mobile)

##### Page d’informations sur l’entretien (Mobile)

Une image contenant texte, capture d’écran, logiciel, Icône d’ordinateur

Description générée automatiquement

18 Information sur l'entretien (Mobile)

##### Formulaire d’ajout d’un nouvel entretien (Mobile)

Une image contenant texte, capture d’écran, logiciel, conception

Description générée automatiquement

19 Formulaire Entretien (Mobile)

##### Formulaire d’ajout de suivi (Mobile)

Une image contenant texte, capture d’écran, logiciel, Système d’exploitation

Description générée automatiquement

20 Formulaire Suivi (Mobile)

## Stratégie de test

### 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.

### 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.

## 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.

# Réalisation

## 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.

## 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 »

### 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.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

21 Fichier de configuration .env

## 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 »

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.

## 

## 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 créent pour les migrations ainsi que des explications de la longueur des attributs.

### Table « users »



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 charactères 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.

### Table « meetings »

Une image contenant texte, capture d’écran, logiciel, affichage

Description générée automatiquement

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.

### Table « aftercares »

Une image contenant texte, capture d’écran, logiciel, Police

Description générée automatiquement

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.

### Table « documents »

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

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’ajout ou supprimer un document PDF stocké dans le répertoire « public/pdf/\*id de l’entretien\* ».

### Table « types »



26 Migration de la table "types"

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

## 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 ».

## 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.

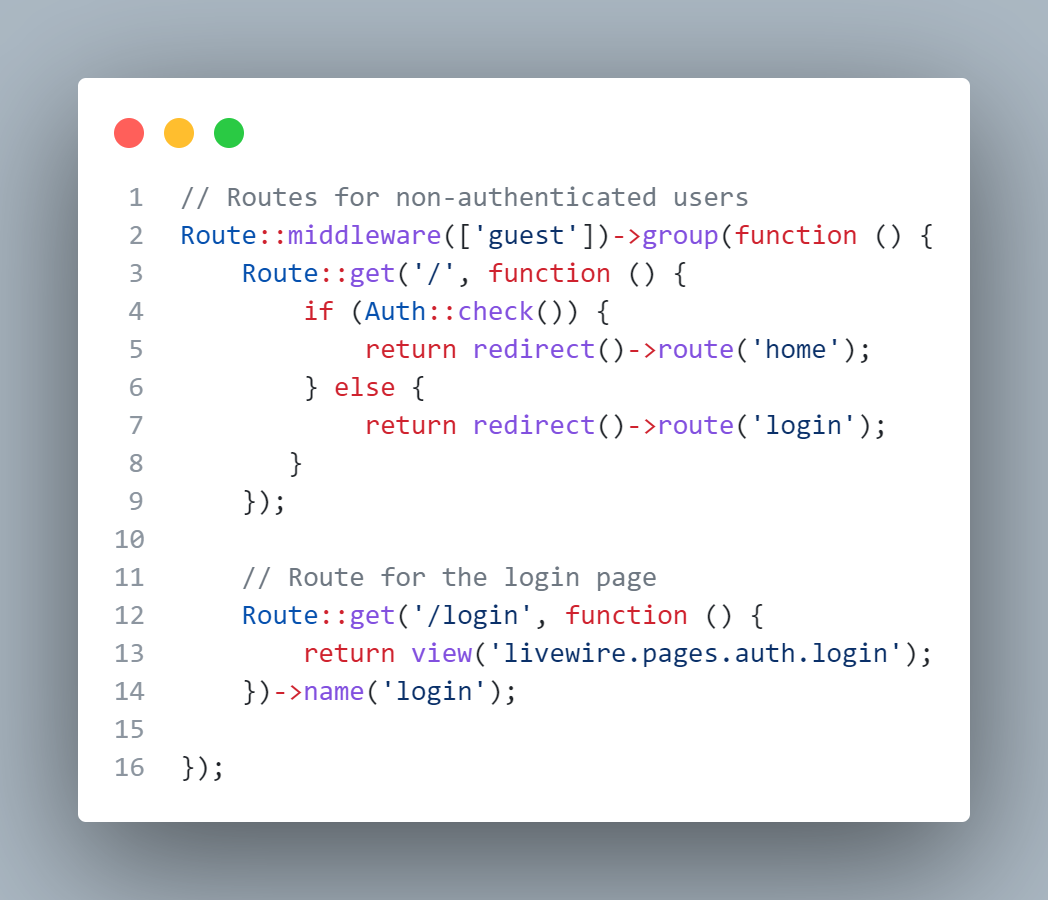
Dans une application Laravel, les routes par défaut sont définies dans plusieurs fichiers. Ces fichiers sont « web.php » qui gère les routes web (cookies, sessions, etc…) et « console.php » qui gère les commandes Artisan.

Cependant, vu que mon projet Laravel est fait avec Breeze. Ce qui fait qu’un autre fichier appelé « auth.php » est ajouté. 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 authentifés (guest).

### Web.php

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

#### Authentification

****

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.

#### MeetingController

Une image contenant texte, capture d’écran, logiciel, Police

Description générée automatiquement

28 Route pour le MeetingController

La route « meeting.index » apparaît deux fois car l’une affiche la page sans aucune donnée donc elle ne reprend pas la variable « year » et l’autre affiche la page avec les valeurs.

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.

#### AftercareController

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

#### GraphicsController

Une image contenant texte, capture d’écran, carte de visite, Police

Description générée automatiquement

#### DocumentController

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

## Description des tests effectués

### 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.

### Test de migrations

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

## Erreurs restantes

*S'il reste encore des erreurs:*

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

## 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…*

# Conclusions

*Développez en tous cas les points suivants :*

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

# Annexes

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

## 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)*

## Journal de travail

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Durée** | **Activité** | **Remarques** |
|  |  |  |  |
|  |  |  |  |

## 

## Manuel d'Installation

## Manuel d'Utilisation

## Archives du projet

*Media, … dans une fourre en plastique*

1. Documentation officielle de Laravel : [https://laravel.com/docs/11.x](https://laravel.com/docs/11.x/installation#why-laravel) [↑](#footnote-ref-1)
2. Documentation du MVC : [https://openclassrooms.com/fr/courses/4670706](https://openclassrooms.com/fr/courses/4670706-adoptez-une-architecture-mvc-en-php/7847928-decouvrez-comment-fonctionne-une-architecture-mvc) [↑](#footnote-ref-2)
3. Documentation sur Eloquent : <https://laravel.com/docs/11.x/eloquent> [↑](#footnote-ref-3)
4. Documentation sur Breeze : <https://laravel.com/docs/11.x/starter-kits#laravel-breeze> [↑](#footnote-ref-4)
5. Documentation sur Tailwind CSS : <https://tailwindui.com> [↑](#footnote-ref-5)
6. Documentation sur Livewire : <https://laravel-livewire.com> [↑](#footnote-ref-6)
7. Documentation sur UwAmp : <https://www.uwamp.com/fr/> [↑](#footnote-ref-7)
8. Conventions de nommage de Laravel : <https://webdevetc.com/blog/laravel-naming-conventions> [↑](#footnote-ref-8)
9. Norme RFC 3696 : <https://datatracker.ietf.org/doc/html/rfc3696> [↑](#footnote-ref-9)