**< Info-Endo >**

***Dossier de projet - Titre Professionnel***

***Développeur Web et Web Mobile***



Tiphaine PONSONNET

Formation DWWM - La Plateforme

Octobre 2025

**Sommaire**

**1. Présentation personnelle .......................... 5**

1.1 Mon parcours professionnel

1.2 Formation multimédia et reconversion

1.3 Motivation pour le développement web

**2. Le projet Info-Endo .......................... 7**

2.1 Présentation du projet ..........................

2.2 Compétences couvertes ..........................

2.3 Environnement de travail et technologies ..........................

2.4 Cahier des charges ..........................

**3. Front-End** ..........................

3.1 Maquettage

3.2 Extrait de code statique

3.3 Extrait de code dynamique

3.4 Sécurité (coté client)

**4. Back-End** ..........................

4.1 Conception base de données.........................

4.2 Base de données..........................

4.3 Migration ..........................

4.4 Composant d’accès aux données (orme loquent +extrait de code)............

4.5 Composant métier ..........................

4.6 Sécurité (côté serveur)

**5. Test et déploiement..........................**

5.1 Test ..........................

5.2 Déploiement ..........................

5.3 Sécurisation des données ..........................

**6. Veille**..........................

6.1 Erreur..........................

6.2 Recherche ..........................

6.3 Solution ..........................

**7. Conclusion** ..........................

1. **Présentation personnelle**
   1. **Mon parcours professionnel**

Je m'appelle Tiphaine Ponsonnet. Après avoir obtenu mon CAP Coiffure en 2016, j’ai exercé ce métier pendant plusieurs années. Cette expérience m’a permis de développer un sens du relationnel, une rigueur professionnelle ainsi qu’une capacité d’adaptation importante. Toutefois, j’ai ressenti le besoin d’évoluer vers un domaine plus technique et créatif, en adéquation avec mes nouvelles aspirations.

* 1. **Formation multimédia et reconversion**

## En 2022, j’ai entrepris une formation multimédia au Groupe Si2A, afin d’acquérir des compétences dans la création numérique. J’y ai découvert les bases du web et des outils essentiels tels que WordPress, Photoshop et Canva. Cette formation a été un véritable point de départ, m’ouvrant les portes vers l’univers du développement web.

* 1. **Motivation pour le développement web**

En janvier 2024, motivée par une volonté de me rapprocher de ma mère et de bénéficier de meilleures opportunités, j’ai déménagé dans le sud de la France. C’est dans ce contexte que j’ai intégré en juin 2024 la formation Développeur Web et Web Mobile à La Plateforme, pour une durée de 16 mois.

1. **Le projet < Info-Endo >**
   1. **Présentation du projet**

Dans le cadre de ma formation au titre professionnel Développeur Web et Web mobile, j’ai réalisé un projet final baptisé Info-Endo, une application web dédiée à l’endométriose. Ce projet a une signification particulière pour moi puisque je suis atteinte et diagnostiquée de cette maladie depuis 2019. Cette expérience personnelle m’a donné la volonté d’aborder un sujet médical encore trop peu médiatisé, en proposant un espace d’information utile, sécurisé et ouvert à tous.

L’objectif principal d’Info-Endo est de centraliser des ressources fiables et accessibles sur l’endométriose afin d’informer, de soutenir et de valoriser la parole des personnes concernées. A travers une interface moderne, dynamique et personnalisée, chacun peut accéder à des articles, témoignages, chiffres clés et localiser facilement les centres spécialisés.

Info-Endo s’adresse à la fois aux patientes, à leurs proches et aux professionnels de santé, pour favoriser la compréhension, encourager le dialogue et améliorer l’accès à l’information sur ce sujet de santé publique.

Comme j’ai mené ce projet seule, j’ai priorisé le **développement des fonctionnalités** essentielles afin d’obtenir rapidement une version **utilisable et présentable**.

Cette approche, inspirée du concept de **MVP (Minimum Viable Product)**, m’a permis de concentrer mes efforts sur le cœur de l’application :

* assurer sa fiabilité,
* sa sécurité
* et sa stabilité avant d’y intégrer des options secondaires.

Ce projet m’a permis d’appliquer l’ensemble des compétences acquis lors de ma formation :

* Développer les fonctionnalités centrales de manière fiable et pérenne,
* Mettre en place une interactivité efficace et accessible,
* Organiser les données de façon claire et cohérente,
* Assurer la sécurité des échanges et des comptes utilisateurs,
* Offrir une expérience adaptée sur tous supports.
  1. **Compétences couvertes**

La réalisation d’Info-Endo m’a permis de mobiliser et de développer un large panel de compétences techniques, organisationnelles et humaines tout au long du projet.

Ce travail m’a donné l’opportunité de mettre en pratique l’ensemble de notions clés abordées au cours de la formation Développeur Web et Web Mobile :

**Développement back-end et gestion de données**

* Structurer une application grâce au Framework Laravel (routes, contrôleurs, modèles, vues).
* Concevoir, créer et interroger une base de données relationnelle avec Eloquent ORM.
* Implémenter un système d’authentification sécurisé et gérer les droits d’accès utilisateurs
* Gérer la validation, le stockage et le traitement sécurisé des contenus.

**Développement front-end et expérience utilisateur**

* Concevoir une interface responsive, claire et accessibles avec **Tailwind** **CSS** et **Blade.**
* Dynamiser l’affichage avec Alpines.js et intégrer des éléments interactifs variés, tels que graphiques, cartes, ou encore un calendrier de suivi personnalisé.

Appliquer les principes d’une architecture **MVC** pour garantir la maintenabilité de l’interface, c'est-à-dire séparer clairement les responsabilités entres la gestion des données **(Modèle)**, la présentation **(Vue)** et la logique métier **(Contrôleur).**

**Intégration d’API et de librairies tierces**

* Utiliser **Chart.js** pour représenter dynamiquement des données de santé
* Implémenter **Leaflet.js** afin de proposer une carte interactive des centres spécialisés.

**Organisation et gestion de projet**

* Développer de l’autonomie dans la résolution de problèmes techniques.
* Adapter mes choix aux besoins réels des utilisateurs cibles (écoute, empathie, accessibilité).
* Maintenir une rigueur et organisation quotidienne indispensables au travail en solo.

Ce projet m’a ainsi permis d’approfondir non seulement mes compétences techniques, mais aussi de renforcer mon sens de l’initiative et ma capacité a mener a bien un projet de bout en bout . Bien que le projet ne soit pas encore totalement achevé, j’ai encore de nombreuses idées pour l’améliorer et le faire évoluer.

**2.3 Environnement de travail et technologies**

Pour réaliser **Info-Endo**, j’ai choisi un environnement technique moderne robuste, qui me permettait de travailler efficacement tout en respectant les standards actuels du développement web

**Langages principaux**

* **PHP 8.1+** pour le back-end, avec Laravel 10.45.1, un Framework puissant offrant une structure solide et sûre pour développer une application modulaire.
* **HTML5**, **CSS3** et **JavaScript ES6** pour la construction du front-end, permettant une interface riche, accessible et performante.

**Framework et Bibliothèques**

* **Laravel**: en plus de faciliter le développement avec son architecture monolithique et l’utilisation du modèle architectural MVC, il offre un écosystème complet (Eloquent ORM, Artisan CLI, middleware, sécurité intégrée).
* **Blade**: moteur de templates Laravel, permettant d’écrire des vues dynamiques en php de manière simple et efficace.
* **Tailwind CSS 4.1.8**: Framework CSS utilitaire, rapide à mettre en place pour un design responsive et moderne, avec une excellente personnalisation.
* **Alpine.js**: JavaScript minimaliste utilisé pour rendre l’interface interactive de manière légère, notamment pour le système de modales, filtres et calendrier.
* **Chart.js**: pour la visualisation dynamique des données statistiques.
* **Leaflet.js**: permets d’intégrer des cartes interactives avec des marqueurs personnalisés pour localiser les centres spécialisés.

**Outils de développement et workflow**

* **Visual Studio Code**: éditeur de code choisi pour sa légèreté et ses nombreuses extensions dédiées à PHP et Laravel
* **Vite 6.3.5**: blunder rapide assurant la compilation assurant la compilation et le rafraichissement efficace des fichiers CSS et JavaScript.
* **Node.js 18+**: nécessaire pour exécuter Vite et gérer les dépendances front-end.
* **Composer**: gestionnaire de paquet PHP essentiel pour installer et maintenir les dépendances Laravel.
* **Git et Github**: outils indispensables pour le versionnement, le suivi des modifications et l’hébergement du code source..
* **Artisan CLI**: interface en ligne de commande Laravel facilitant la génération automatique de code, migrations et le lancement du serveur de développement.

**Base de données**

* **SQLite** en local pour sa simplicité et sa légèreté pendant le développement.
* Compatibilité avec **MySQL/MariaDB** prévue pour un éventuel déploiement plus large.
* Gestion via **Eloquent ORM**, qui permet des manipulations simples, propres et sécurisées des données relationnelles.

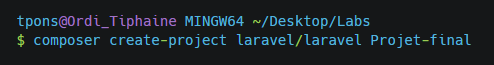
Cet ensemble d’outils et technologie m’a donné une base solide pour développer un projet complet, organisé et maintenable, tout en assurant des performances et une expérience utilisateur optimal

**Installation**

* Prérequi : PHP, Composer, Node.js

**Installation de laravel**

**Installer Laravel ( dans le terminal on tape cette cmd)**



**Installer les dépendances front-end (Vite intégré)**

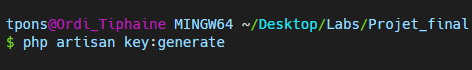
**Dans le dossier projet**

**

**Configurer l’environnement**

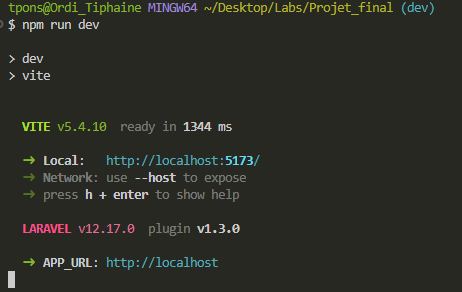
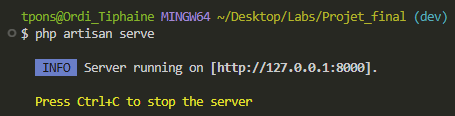


**Généré la clé d’application**



**Et adapter les variables dans env. selon les besoins (BDD, URLs, ect.)**

**Lancer le serveur de dev et Vite**



Arch**itecture générale : *Le modèle MVC***

Info-Endo est basé sur le **pattern MVC** *(Modèle – Vue – Contrôleur)*, standard dans le Framework **Laravel.**

*Ce principe permet une séparation claire des responsabilités :* ^

|  |  |
| --- | --- |
| Modèle | Gère la structure des données les relations, les règles métier et toute la logique liée à la base de données |
| Vue | Gère l’affichage et l’interface utilisateur avec Blade pour générer dynamiquement les HTML et Tailwind CSS pour le design |
| Contrôleur | Fait le lien entre le modèle et la vue, traite les requêtes, applique la logique métier et décide quelle affichage présenter |

*Ce choix garanti :*

* Une meilleure organisation du code
* Une plus grande facilité pour corriger ou faire évoluer les fonctionnalités
* Un respect des bonnes pratiques de développement moderne

*Front –end :*

|  |  |
| --- | --- |
| Composant | Rôle |
| Blade | * Moteur de templates PHP pour un rendu dynamique des vues |
| Tailwind CSS | * Framework CSS utilitaires pour un design responsive et moderne |
| Alpine.js | * Ajout d’interactivité légère côté client (modales, filtres) |
| Chart.js | * Visualisation dynamique des statistiques |
| Leaflet.js | * Intégration d’une carte interactive |

*Point clés :*

* Interface responsive et adaptable a tous supports
* Interaction fluide grâce a Alpine.js sans alourdir le chargement
* Visualisation claire et attractive des données via Chart.js Leaflet.js

*Back-****end****:*

Le back-end repose sur **Laravel**, qui organise le code en plusieurs couches bien distinctes pour garantir la maintenabilité et la robustesse.

La gestion des utilisateurs est assurée par **Laravel Breeze**, permettant un système sécurisé et simple.

L’utilisation **d’Eloquent ORM** facilite la manipulation des données tout en assurant leur intégrité.

|  |  |
| --- | --- |
| Composant | Rôle et Fonctionnalité |
| Laravel | * Framework PHP structurant le projet selon de modèle MVC |
| Routes | * Gestions des URL et des accès |
| Contrôleurs | * Traitement des requêtes et logique métier |
| Eloquent ORM | * Interaction ave la base de données et gestion des données |
| Laravel Breeze | * Authentification et gestion des utilisateurs |

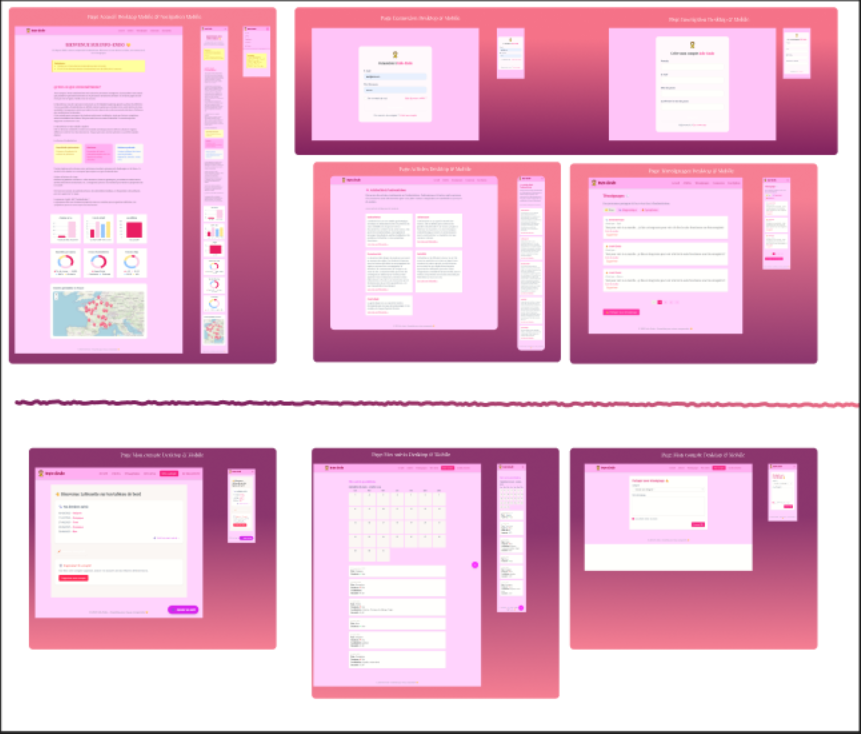
* 1. **Cahier des charges**

Pour mieux visualiser le cahier des charges d Info-Endo, je vous le présente ci-dessous, sous forme de tableau synthétique, qui regroupe les fonctionnalités, contraintes techniques et besoins utilisateurs, en tenant compte de mes ressources et du temps disponible.

Tableau 1 :

|  |  |  |
| --- | --- | --- |
| Catégorie | Description | Remarques |
| Objectifs fonctionnels | * Blog, * Témoignages, * Statistiques dynamiques (Chart.js), * Carte interactive (Leaflet.js), * Espace utilisateur sécurisé | Priorisation des fonctionnalités MVP pour garantir un produit fiable réalisé en autonomie |
| Contraintes techniques | * Framework Laravel, * Base de données SQLite compatible MySQL, * Sécurité avec Laravel Breeze et validation, * Responsive design | Nécessité de fiabilité et maintenabilité du code |
| Critères  qualité | * Ergonomie et accessibilité, * Performance rapide, * Architecture MVC claire, * Interactivité front-end avec Alpine.js | Approche modulaire pour évoluer facilement |
| Environnement de travail | * Utilisation de XAMPP, * Composer, * Node.js, * Git/GitHub, * Vite | Outils professionnels pour développement agile |
| Développement MVP | * Priorisation des fonctionnalités minimales viables garantissant un résultat rapide et fiable avec extension possible | Choix stratégique adapté au travail en solo |

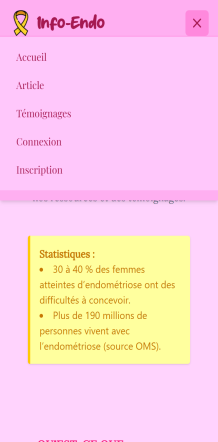
1. **Front-End**

Cette partie présente les différences fonctionnalités réalisées, illustrées par des captures d’écran de l’application Info-Endo.

**3.1 Maquettage**

**Interface d’accueil et navigation**

L’interface d’accueil propose une vue d’ensemble immédiate du projet Info-Endo, avec un accès centralisé à toutes les fonctionnalités : navigation vers le blog, la carte, les témoignages ou le suivi personnalisé lorsqu’on est connecté. La page d’accueil présente les grandes thématiques via un menu latéral pour desktop et adaptatif pour mobile, assurant une expérience fluide quel que soit le support. L’ensemble du design sobre et illustré, a été réalisé avec Tailwind CSS pour garantir accessibilité modernité

Desktop Mobile Navigation Mobile

Système de blog (articles)

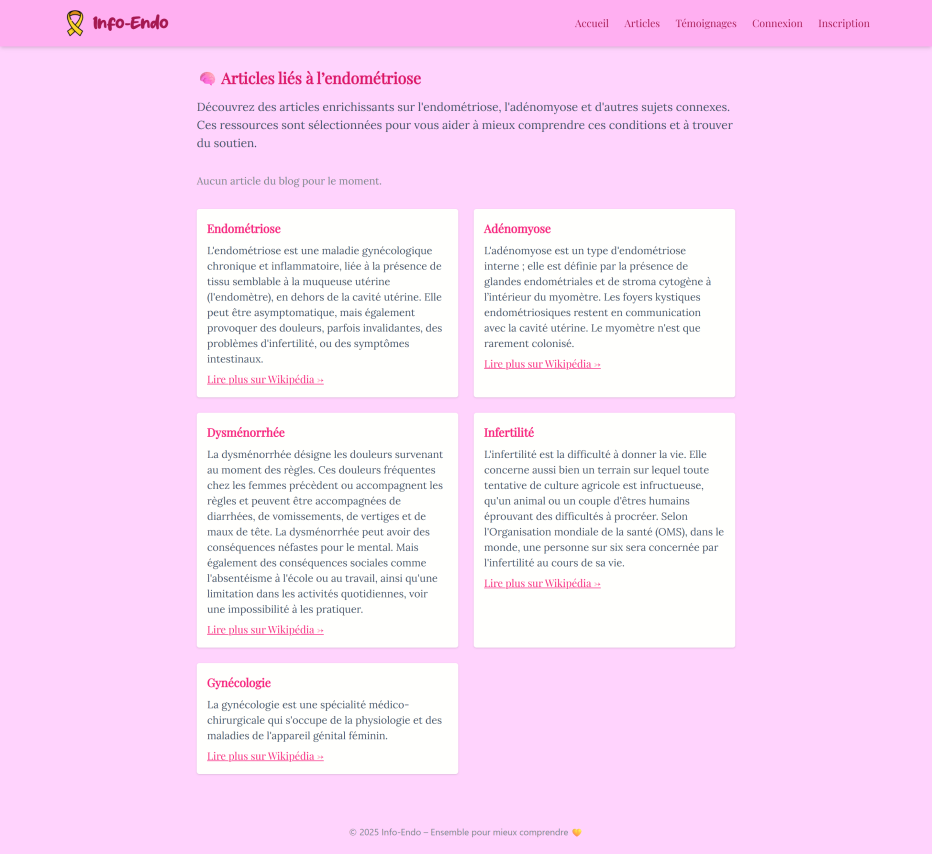
Le module d’articles offre aujourd’hui un flux d’actualités fiables issues de Wikipédia consultable par tous sur la page Articles. Cette a base déjà accessible et régulièrement actualisé, constitue un premier socle d’information autour de l’endométriose.

***A terme, le système évoluera pour inclure :***

* L’import automatique d’articles via des API santé spécialisées
* La possibilité pour les utilisatrices de proposer des articles, soumis à validation par un administrateur avant publication.





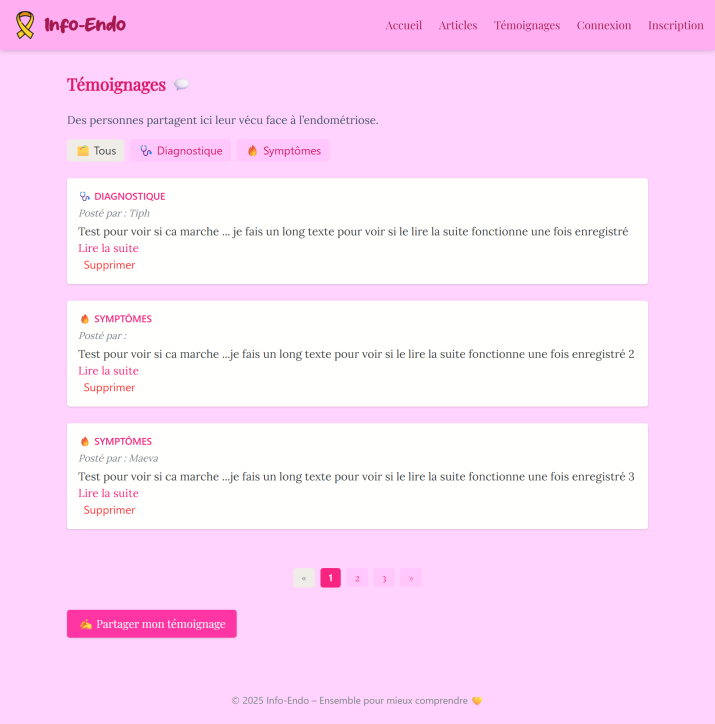


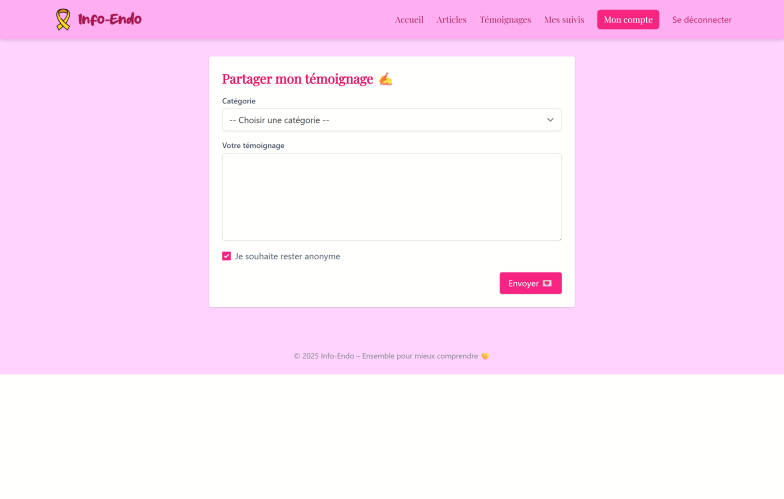
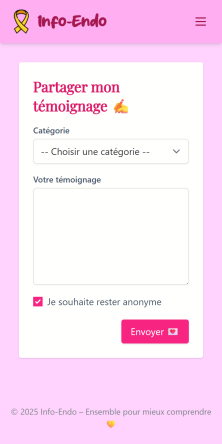
**Gestion des témoignages**

L’espace témoignages permets aux patientes (ou proches) de partager leur expérience autour de l’endométriose. Les contributions peuvent être publiques ou anonymes, et sont stockées dans la table ***post\_temoignages***.

*Le système repose sur* :

* Un formulaire guidé avec validation dynamique des champs (Laravel + Alpine.js),
* Une modération avant publication afin de garantir le respect et la sécurité,
* Un système de boutons pour filtrer les témoignages par diagnostique ou symptôme,
* Une pagination permettant d’alléger la présentation des contenus,
* Un affichage sous forme de liste, consultable par ordre chronologique ou pertinence.

*A termes, un système de filtres (par thèmes ou mots clés) pourra être ajouté pour facilité la lecture et valoriser les témoignages.*



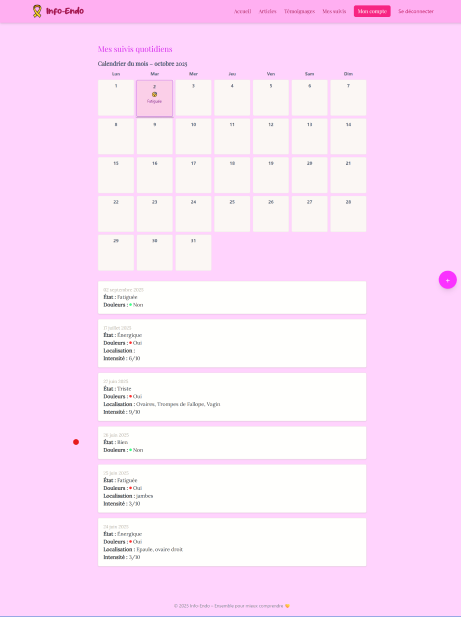
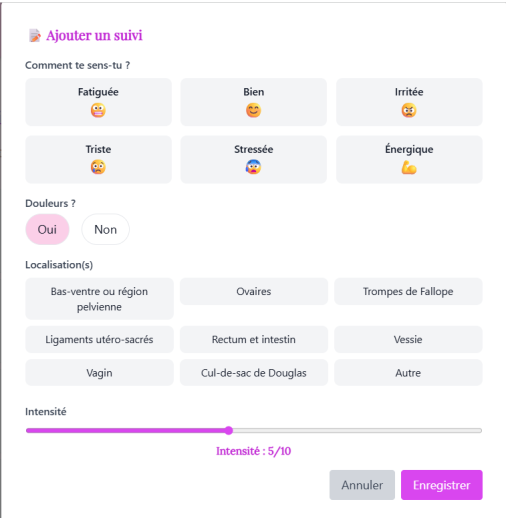


**Gestion des suivis personnalisés**

La page Suivi permet aux utilisatrices de saisir et consulter facilement l’évolution de leurs symptômes et douleurs. Ce suivi personnalisé facilite la compréhension des variations de la maladie et optimise l’accompagnement médical.

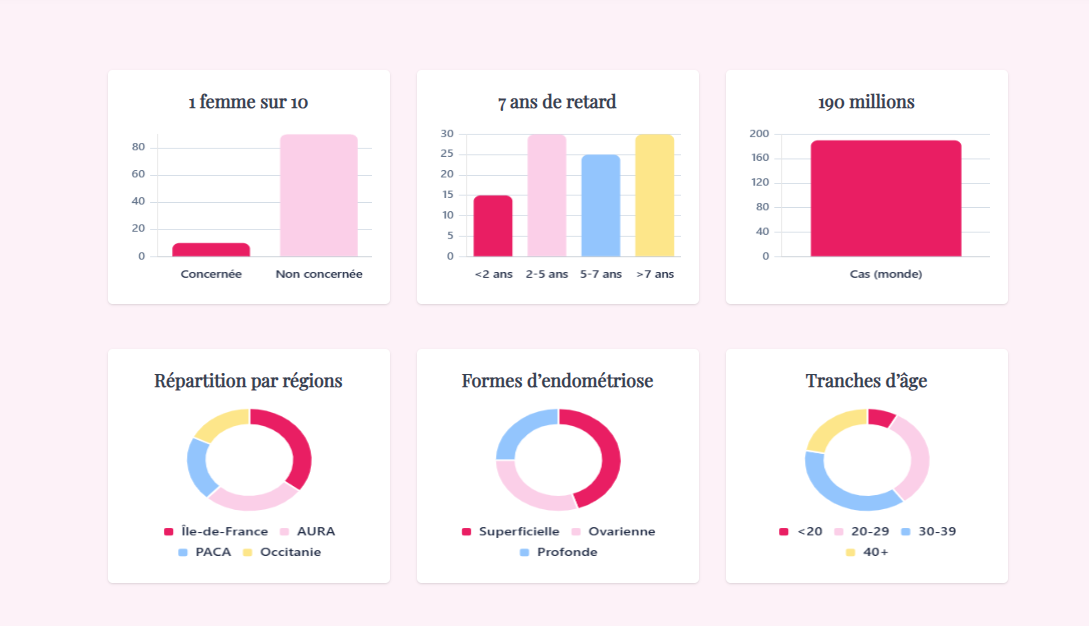
L’interface intuitive propose une visualisation simple de l’historique des données saisies, renforçant l’implication des utilisatrices dans leur parcours soin.



** Statistiques dynamiques Chart.js**

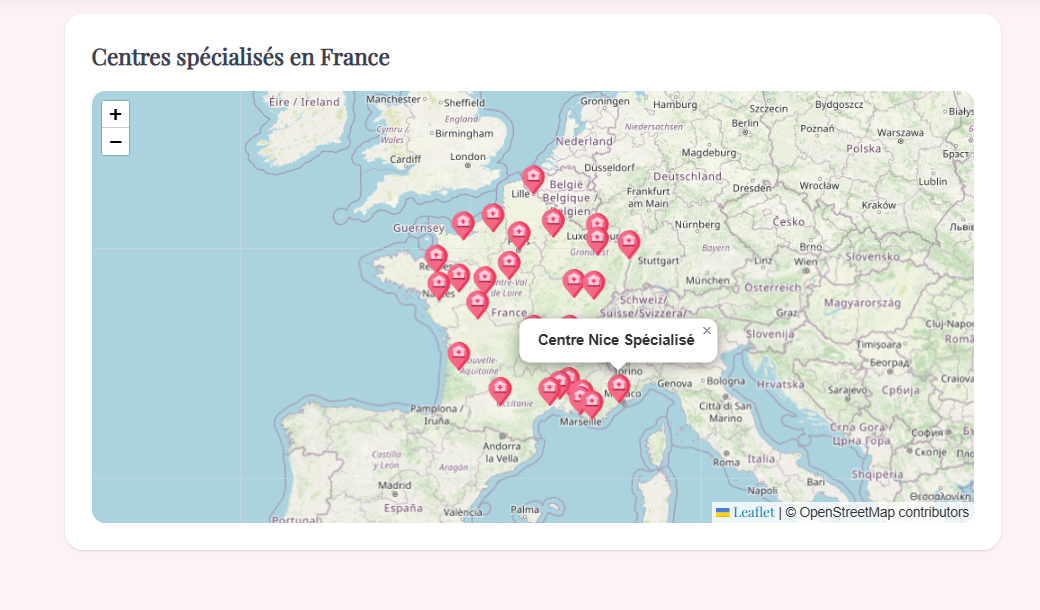
Ces graphiques servent à mieux comprendre l’endométriose sous differents angles :

|  |  |
| --- | --- |
| Graphique Barres/ Colonne | Graphique Donuts |
| * Nb de femmes atteintes * Retards de diagnostic * Nb Cas dans le MONDE | * Répartition par régions * Formes d’endométriose * Ages (diagnotiquées) |
| Meilleure visualisation de la répartitions des données entres les differentes catégorie. | Suivre l’évolution ou voir la comparaison entre plusieurs groupes. |



**Carte interactive avec Leaflet.js**

Cette section présente une carte interactive réalisée avec Leaflet.js, qui permet de visualiser facilement la localisation des centres liés à l’endométriose.

* La carte affiche les centres avec des pointeurs
* Elle facilite la localisation des centres géographiquement
  + - * *À venir : ajout des coordonnées précises*
      * *À venir : insertion de liens cliquables vers les sites des centres*

**Système d'authentification**

Ce projet intègre un système d’authentification qui permet aux utilisateurs de se connecter en toute sécurité pour accéder aux différentes fonctionnalités du site

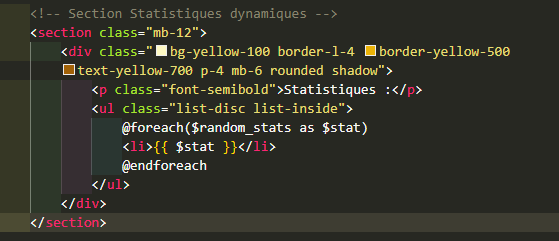
* L’utilisateur s’inscrit ou se connecte via un formulaire sécurisé
* Le système vérifie les informations fournies (identifiant et mot de passe)
* Une gestion de session assure la connexion pendant un certain temps
* L’utilisateur peut se déconnecter quand il le souhaite pour sécuriser son compte
* Des mesures sont prévues pour protéger les mots de passe (hachage, sécurisation)
* Le système pourra évoluer vers des options supplémentaires (comme liens cliquables ou authentification forte)

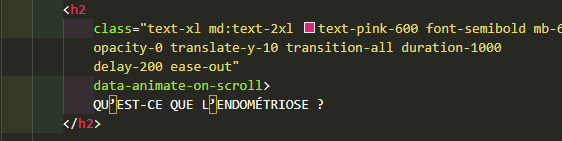




**3.2 Extrait de code statique**

Ce premier extrait de **index.blade.php** illustre l'architecture technique du projet Info-Endo. On y voit l'utilisation du moteur de **template Blade de Laravel** avec la directive @foreach qui parcourt un tableau $random\_stats venant du contrôleur.

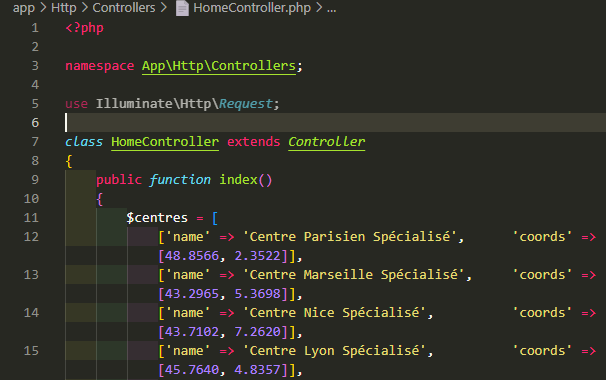
L'intégration de **Tailwind CSS** est évidente avec des classes utilitaires comme bg-yellow-100 et border-l-4



Cette implémentation démontre la maîtrise du responsive design via md:text-2xl et la préparation d'animations avec des classes  opacity-0 et translate-y-10 couplées à l'attribut data-animate-on-scroll.

Le code respecte la séparation des responsabilités avec des sections **HTML5** sémantiques et une logique métier claire

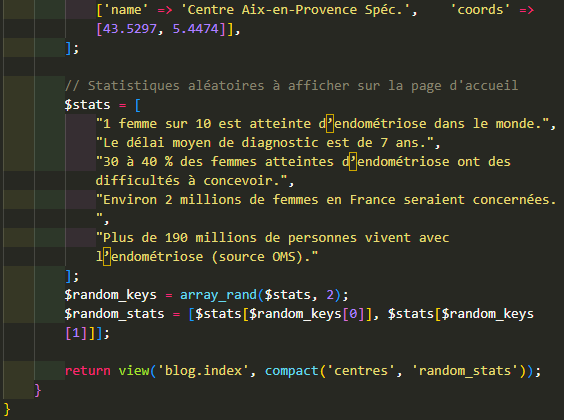
**3.3 Extrait de code dynamique**



[**Home.Controller.php**](http://Home.Controller.php)

Ce contrôleur Laravel illustre la génération de contenu dynamique côté serveur.

La méthode index() prépare deux jeux de données : un tableau $centres contenant les coordonnées GPS des centres médicaux spécialisés,



et un système de statistiques aléatoires avec  array\_rand() qui sélectionne 2 statistiques parmi 5 disponibles à chaque chargement de page.

Cette approche démontre la logique métier dynamique en **PHP** avant l'envoi des données vers la vue **Blade**

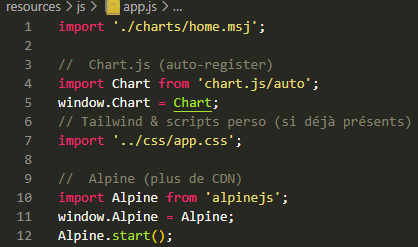
**routes / web.php**

****

Le fichier de routes démontre la gestion dynamique des **URLs avec Laravel**. On y voit une redirection automatique de la racine vers blog.index, l'utilisation de middleware auth et verified pour protéger certaines routes, et le groupement de routes avec Route::middleware('auth')-**>**group().

Cette structure permet un routage conditionnel et sécurisé selon l'état de connexion de l'utilisateur.

**js/app.js**

Ce fichier app.js utilise la syntaxe **ES6** [[1]](#endnote-2)moderne avec des imports modulaires pour structurer le code JavaScript.

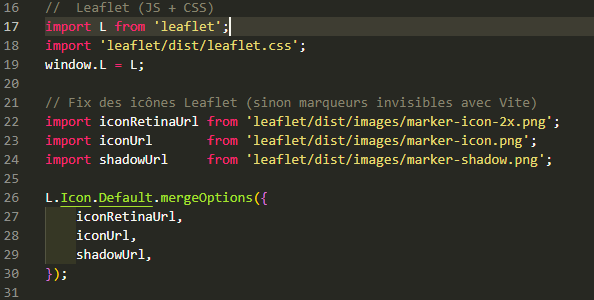
On y retrouve l'intégration de **Chart.js** pour les graphiques dynamiques,

**Alpine.js** pour la réactivité légère, et l'importation des styles **CSS de Leaflet.**

Cette approche modulaire permet une meilleure organisation du code et une optimisation des performances grâce au bundling automatique

La configuration **Leaflet** résout un problème technique courant :

l'affichage correct des marqueurs de carte.

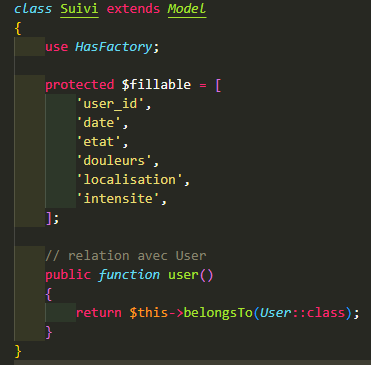
En important explicitement les icônes (iconRetinaUrl, iconUrl, shadowUrl) et en utilisant L.Icon.Default.mergeOptions(), le code garantit que les marqueurs s'affichent correctement sur tous les navigateurs

.

Cette configuration technique démontre la maîtrise des spécificités de la librairie**Leaflet** et l'attention portée à l'expérience utilisateur sur les cartes interactives

**Model/Suivi.php**

Le **modèle Eloquent** ***Suivi*** représente la structure de données pour le suivi de l'endométriose.



Il utilise le trait  HasFactory pour les tests et migrations,

définit les champs  $fillable pour la sécurité (mass assignment),

et établit une relation  belongsTo avec le modèle User.

Cette architecture permet de lier chaque suivi médical à un utilisateur spécifique via l' **ORM Eloquent de Laravel.**

**3.4 Sécurité (côté client)**

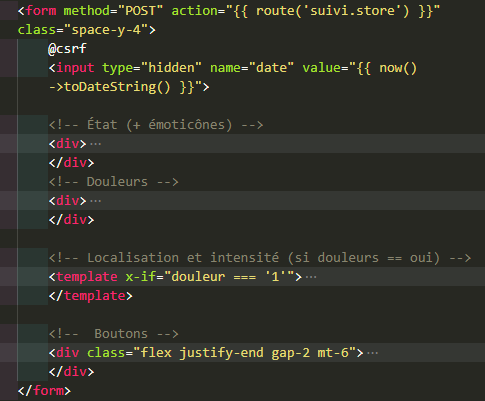
**Blog/suivis/index.blade.php**

Ce formulaire de suivi médical illustre l'implémentation de plusieurs mesures de sécurité côté client du projet Info-Endo

La directive @csrf  génère automatiquement un token de protection contre les attaques Cross-Site Request Forgery, tandis que le routage nommé route('suivi.store') évite les URLs en dur.

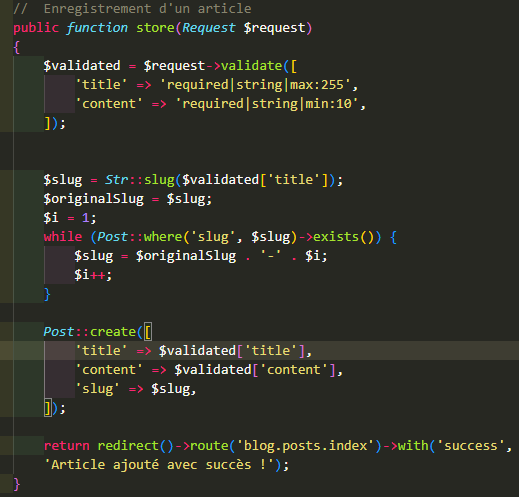
 L'utilisation de now()->toDateString() démontre l'échappement automatique des données par le moteur Blade,

et la structure conditionnelle x-if="douleur === '1'"  **(Alpine.js)** assure une logique d'affichage sécurisée sans exposition de données sensibles.



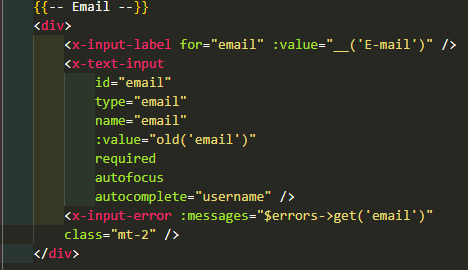
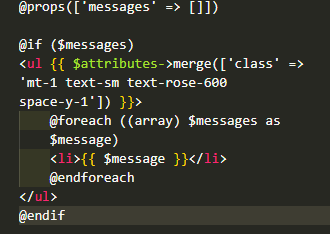
**Controller/PostController.php**

Le contrôleur **PostController** implémente une sécurisation complète du traitement des données utilisateur.

La méthode validate() applique des règles strictes (required, string, max:255, min:10) qui filtrent et valident les inputs côté serveur.

La génération automatique de slug avec Str::slug() prévient les injections dans les URLs, tandis que la boucle while() avec vérification exists() garantit l'unicité des slugs.

L'insertion via Post::create() utilise exclusivement les données $validated, empêchant toute injection de champs non autorisés (mass assignment protection)

**Components/input-error.blade.php & views/auth/login.blade.php**

Le composant input-error.blade.php centralise l'affichage sécurisé des messages d'erreur de validation dans le projet Info-Endo :

Sécurité XSS :

* Utilise l’échappement automatique de Blade {{ message}}}
* Prévient les attaques XSS sur les message d’erreur

Interface utlisateur :

* Applique une classe Tailwind text-rose-600
* Permet l’identification visuelle immédiate des erreurs

Accessibilité :

* Structure les messages en liste < url >
* Optimisé pour les lecteurs d’écran

Réutilisabilité :

* La directive @props permets une reutilisabilité maximale
* Fusion d’attributs avec $attributes**->**merge()
* Garantit une cohérence sécuritaire sur tout le projet

1. **Back- End**
   1. **Conception BDD avec la méthode MERISE**

Créer dans les années 197o, MERISE (***M****éthode d'****É****tude et de* ***R****éalisation* ***I****nformatique pour les* ***S****ystèmes d'****E****ntreprise*), est une méthode française de conception et de developpement de systèmes d’information.

Elle reste aujourd’hui une référence pour lla modélisation des bases de données.

La méthode MERISE repose sur un principe fondamental :

Elle sépare l’analyse des besoins et la conception physique en trois

niveaux hiérarchiques distincts mais liés, permettant ainsi une maîtrise

progressive et robuste du système à chaque étape.

Les trois niveaux sont :

1. Niveau Conceptuel (QUOI ?) :

Exprime les besoins fondamentaux du système sans entrer dans le détail technique, décrit ce que doit faire le système pour répondre aux attentes métier.

1. Niveau Logique (COMMENT ?) :

Précise l’organisation des données et leur structuration, sans tenir compte des contraintes techniques spécifiques. On y définit par exemple les tables, relations, cardinalités, etc., que l’on traduit en un MCD.

1. Niveau Physique (AVEC QUOI ?) :

Décrit la mise en œuvre concrète du système, c’est-à-dire les choix techniques, le système de gestion de base de données, les types de données précis, les index, etc.

**4.2 Base de données**

Le Modèle Physique de Données (MPD) a été conçu à l'aide de l'outil en ligne ***dbdiagram.io***, permettant une modélisation visuelle précise des tables, types de données et contraintes d'intégrité référentielle.

Cette approche garantit une structure de base de données optimisée et aux standards relationnels, facilitant l'implémentation technique sous **MySQL** et l'intégration avec **l'ORM Eloquent** **de Laravel**.

Cette méthodologie **MERISE** assure une conception robuste et évolutive de notre système de suivi médical personnalisé.

* 1. **Migration**

Cette migration Laravel définit la table de suivi médical du projet Info-Endo avec le Schema Builder. La méthode up() utilise une syntaxe déclarative moderne :

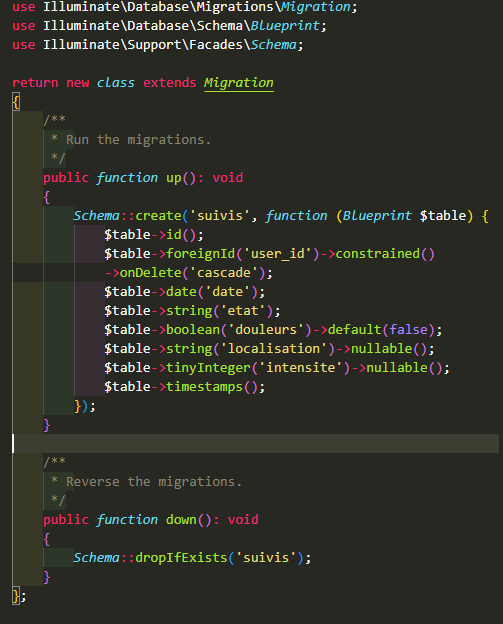
**Structure optimisée pour les données médicales :**

foreignId(‘user\_id')**->**constrained()**->**onDelete('cascade') : Relation utilisateur avec suppression en cascade

boolean('douleurs')**->**default(false) : Indicateur de douleur avec valeur par défaut

tinyInteger('intensite')**->**nullable() : Échelle d'intensité (0-10) optionnelle

timestamps() : Horodatage automatique pour audit médical

**database/migrations/xxxx\_create\_suivis\_table.php :**

**Avantages techniques :**

**SQL agnostiques :** Compatible MySQL, SQLite

**Rollback sécurisé :** La méthode down () permet Schema::dropIfExists('suivis')

**Versioning BDD :** Evolution collaborative et tracée de la base de données

Cette approche garantit une base de données robuste et évolutive pour les données sensibles de santé

* 1. **Composant d’acces aux données ( ORM Eloquant )**

L'ORM Eloquent abstrait l'accès aux données du projet Info-Endo via des modèles PHP élégants qui remplacent les requêtes SQL complexes

**Models/User.php :**

****

**Models/Suivi.php :**

Ces deux modèles Eloquent illustrent parfaitement l'architecture relationnelle du projet Info-Endo et démontrent la puissance de l'ORM Laravel

**Structure et sécurité des modèles :**

**Modèle User.php**  (Authentification Laravel) :

* Héritage spécialisé : extends Authenticatable pour gestion native des utilisateurs
* Traits intégrés : HasFactory (fabrication de données de test), Notifiable (système de notifications)
* Sécurité renforcée : $hidden masque automatiquement password et tokens dans les réponses JSON
* Casting automatique : 'password' **=>** 'hashed' et 'email\_verified\_at' **=>** 'datetime'

**Modèle Suivi.php**  (Données métier) :

* Protection mass assignment : $fillable autorise uniquement les champs sécurisés
* Structure médicale : Champs adaptés au suivi de l'endométriose (date, état, douleurs, localisation, intensité)
* Timestamps automatiques : Laravel gère created\_\_at et updated\_\_at sans intervention

Cette relation **One-to-Many** **(1:N)** reflète la logique métier d' Info-Endo :

Un utilisateur peut créer plusieurs entrées de suivi médical dans le temps, mais chaque suivi appartient à un seul utilisateur.

L'implémentation via belongsTo(User::class) et la clé étrangère user\_id garantit l'intégrité référentielle et la confidentialité des données de santé.

* 1. **Composant métier**

**Request/ProfileUpdateRequest.php :**

La classe ProfileUpdateRequest implémente le pattern FormRequest de Laravel pour encapsuler la logique de validation métier

Architecture technique :

* **Héritage** : extends FormRequest pour intégration automatique au cycle Laravel
* **Injection de dépendance** : Résolution automatique dans les méthodes de controller
* **Validation avant traitement** : Échec HTTP 422 si règles non respectées

Points techniques clés :

* Rule::unique()->ignore() : Unicité conditionnelle avec exclusion de l'utilisateur actuel
* **Validation multi-critères** : Format email + unicité en base + normalisation lowercase
* **Accès au contexte** : $this->user()->id pour récupérer l'utilisateur authentifié

Avantages developpeur :

* **Séparation responsabilités** : Validation séparée du controller
* **Réutilisabilité** : Règles centralisées pour profile update
* **Testabilité** : Validation isolée et mockable
* **Performance** : Validation avant appel base de données
  1. **Sécurité**

La sécurité serveur d'Info-Endo s'appuie sur trois piliers complémentaires à la protection CSRF (section 3.4)

**Middleware personnalisé**

**Middleware/EnsureUserIsAdmin.php**

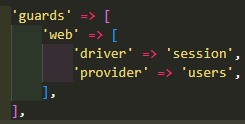
****

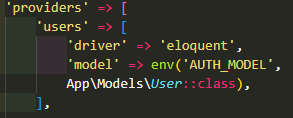
* Vérifie si l’utilisateur est authentifié ET administrateur
* Bloque avec un code 403 sinon
* Renforce la sécurité sur toutes tes routes sensibles

Grâce à ce middleware personnalisé, seuls les administrateurs peuvent accéder aux routes protégées, limitant ainsi les risques d’utilisation non autorisée de fonctionnalités sensiblesAuthentification

* Providers : définissent d’où viennent les utilisateurs (ici, modèle User via Eloquent)
* Guards : gèrent l’authentification à travers les sessions (web)
* Flexibilité : permet de facilement intégrer plusieurs méthodes de connexion ou providers
* Sécurité : toute la gestion de connexion passe par ce système standard Laravel

**config/auth.php**



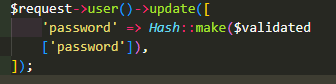


La configuration auth.php permet de gérer la sécurité des sessions et identités utilisateur via le système Eloquent. Elle assure une gestion centralisée de toutes les opérations sensibles liées à l’authentification.

Hachage des mots de passe

* Bonnes pratiques : hashage systématique avant sauvegarde
* Fiabilité : vérification du hachage en test automatisé
* Sécurité : mots de passe jamais stockés en clair

**Controllers/Auth/NewPasswordController.php :**

****

**test/Feature/Auth/PasswordUpdateTest.php :**



L’utilisation systématique de Hash::make() garantit une protection efficace des mots de passe en base.

Les tests automatisés valident le bon fonctionnement et la sécurité du processus de mise à jour.

1. **. Test et deploiement**

**5.1 Tests - Ma démarche pour Info-Endo**

Comment j'ai testé mon application

Pour tester Info-Endo, j'ai choisi de faire du test manuel plutôt que des tests automatisés. Étant seule sur le projet et avec le temps imparti, cette approche m'a semblé plus réaliste et plus efficace pour valider que tout fonctionne bien.

Ma méthode de test

**Je me suis mise dans la peau d'une utilisatrice**

J'ai testé mon app comme si j'étais une vraie patiente qui veut suivre son endométriose. J'ai reproduit tous les gestes qu'elle ferait : s'inscrire, se connecter, remplir ses suivis, consulter ses données.

Mon environnement de test

**Pour tester sereinement, j'ai mis en place :**

* Une base de données séparée pour ne pas polluer mes vraies données
* Quelques comptes utilisateurs fictifs pour tester différents scénarios
* Des données d'exemple cohérentes pour le suivi médical

Ce que j'ai testé concrètement

L'authentification avec **Laravel Breeze**

**Inscription d'un nouvel utilisateur :**

* J'ai testé tous les champs du formulaire (nom, email, mot de passe)
* Vérifié que les erreurs s'affichent bien quand on oublie un champ
* Contrôlé qu'on ne peut pas créer deux comptes avec le même email
* Testé la redirection vers le dashboard après inscription

**Connexion/déconnexion :**

* Testé avec de bons et mauvais identifiants
* Vérifié que les messages d'erreur sont clairs
* Contrôlé que la session se maintient bien
* Testé la déconnexion proprement
* Le système de suivi médical

**Saisie des données de suivi :**

* Testé tous les champs : date, état, douleurs, localisation, intensité
* Vérifié que les données se sauvegardent correctement en base
* Contrôlé l'affichage des formulaires sur mobile et desktop
* Testé les validations (dates cohérentes, intensité entre 0 et 10)

**Dashboard et affichage des données :**

* Vérifié que les statistiques s'affichent bien
* Testé les graphiques Chart.js avec différentes données
* Contrôlé que tout reste responsive sur mobile

La sécurité

**Protection des données personnelles :**

* Testé qu'un utilisateur ne peut voir que SES données
* Vérifié le middleware admin sur les pages sensibles
* Contrôlé que les formulaires ont bien leur protection **CSRF**
* Testé les tentatives d'accès direct aux URLs protégées

Les navigateurs testés

**J'ai testé Info-Endo sur :**

* Chrome : Tout fonctionne parfaitement
* Firefox : Bon fonctionnement des composants Alpine.js
* Safari : Responsive design validé
* Mobile : Interface adaptée aux petits écrans, boutons et fromulaires bien dimensionnés

Les problèmes identifiés et leurs résolutions

**Au cours de ma phase de test, j'ai identifié trois dysfonctionnements qui ont nécessité des corrections :**

*1. Dysfonctionnement de l'affichage des erreurs de validation*

* Problème constaté : Les messages d'erreur de validation ne s'affichaient pas de manière cohérente sur certains formulaires.
* Solution appliquée : J'ai analysé le composant input-error.blade.php et corrigé la logique d'affichage pour garantir que tous les messages d'erreur soient visibles par l'utilisateur.
* Résultat : Amélioration de l'expérience utilisateur avec des retours d'erreur clairs et systématiques.

2. Règles de validation trop restrictives

* Problème constaté : Les contraintes de validation dans ProfileUpdateRequest étaient excessivement strictes, créant des blocages pour les utilisateurs légitimes.
* Solution appliquée : J'ai ajusté les règles de validation pour trouver un équilibre entre sécurité des données et facilité d'utilisation.
* Résultat : Processus de mise à jour de profil plus fluide tout en maintenant l'intégrité des données.

*3. Faille de sécurité dans le contrôle d'accès*

* Problème constaté : Une route administrative n'était pas correctement protégée, permettant potentiellement l'accès à des fonctionnalités sensibles.
* Solution appliquée : Implémentation du middleware EnsureUserIsAdmin sur les routes concernées pour garantir un contrôle d'accès approprié.
* Résultat : Sécurisation complète de l'interface d'administration et protection des données sensibles.

*Mes résultats de test*

**Au final, j'ai validé :**

* Tous les scénarios d'inscription/connexion
* Toutes les fonctionnalités de suivi médical
* L'affichage sur tous les appareils
* La sécurité des données utilisateur
* La protection contre les accès non autorisés

*Pourquoi j'ai choisi cette méthode*

**Cette approche manuelle m'a permis de :**

* Vraiment comprendre comment fonctionne mon app
* Détecter les petits détails UX qui auraient pu gêner les utilisatrices
* Valider que tout est sécurisé pour des données médicales
* Être sure que l'app fonctionne dans de vraies conditions d'usage

*Perspectives d’évolutotion*

**Pour garantir la pérennité et la qualité d’Info-Endo, j’envisage les améliorations suivantes :**

* ***Automatisation des tests***

Déployer des tests **PHPUnit** ciblés (modèles User, Suivi) et **Tests Feature Laravel** pour les contrôleurs (inscription, suivi, dashboard) afin de détecter rapidement toute régression

* ***Intégration continue***

Mettre en place des scripts d’automatisation (deploiement via rsync ou GitHub Actions) pour appliquer migrations, installer les dépendances (composer install --no-dev) et vider les caches en une seule commande.

* ***Monitoring et supervision***

Installer **Laravel Telescope** en production pour tracer requêtes, exceptions et performances, et configurer des alertes en cas d’anomalies.

* ***Sécurisation avancée***

Chiffrer les données sensibles au repos **(AES-256, Advanced Encryption Standard),** renforcer l’authentification **(2FA)** et auditer régulièrement les logs pour rester conforme aux exigences RGPD.

* ***Application mobile***

Développer une app React Native avec Expo, en s’appuyant sur l’API/REST existante (endspoints JSON optimisés, authentification Sanctum/JWT) pour :

* Publier simultanément sur iOS et Android à partir dun seul code JavaScript
* Assurer des réponses légères et versionnées (v1, v2…)
* Préserver la sécurité des données médicales et simplifier la maintenance sur le long terme.

*En conclusion,*

Info-Endo est aujourd’hui une application stable et sécurisée, validée manuellement pour garantir la qualité des parcours clés (inscription, suivi médical, dashboard) et la protection des données sensibles.

Les perspectives d’évolution *-* ***automatisation des tests PHPUnit et Feature Laravel, monitoring via Telescope, alerting immédiat, chiffrement AES-256 et 2FA, audit RGPD et développement d’une application mobile ReactNatieve/Expo s’appuyant sur l’API REST*** - constitueront la prochaine étapes pour renforcer sa rebustesse et évolutivité et son accessibilité.

**5.2 Deploiement**

**Pour déployer Info-Endo, j’ai suivi ces étapes clés**

**Installation dépendances :**

**composer install --no-dev --optimize-autoloader**

**npm install && npm run build**

**Configuration environnement :**

**Copier .env.example → .env et ajuster les variables (DB, APP\_ENV=production, APP\_DEBUG=false)**

**Migrations et seeders :**

**php artisan migrate --force**

**php artisan db:seed --class=DatabaseSeeder**

**Optimisation Laravel :**

**php artisan config:cache**

**php artisan route:cache**

**php artisan view:cache**

**Accès au dépôt :**

***Consulter le script complet et les instructions détaillées ici***

**<** [**https://github.com/Tiph06/Projet\_final/tree/dev**](https://github.com/Tiph06/Projet_final/tree/dev) **>**

**6. Veille**

6.1 Erreur

1. ES6 : (ECMAScript 2015) revolution du JS en 2015 qui ajouté plein de nouvelle fonctionnalités moderne.

   Le Cross-Site Scripting (XSS) est une vulnérabilité de sécurité web qui permet à un attaquant d'injecter du code malveillant, typiquement des scripts (souvent en JavaScript), dans des pages web considérées comme fiables [↑](#endnote-ref-2)