

**Simplon.co**

**Youcode**

**- SAFI-**

**Projet Fil Rouge**

**1ère année Développement Web**

**Rapport du Projet Fil Rouge: RUBI**

Conception et Réalisation d'une application web dédié à la  
promotion du don de sang

**Réalisé par :** Mlle. Hajar Walfi

**Encadré par :** M. ABID Abdeladim

# Remerciements

À l'aube de la clôture de ce projet, qui marque une étape essentielle dans mon parcours à YouCode, je souhaite exprimer ma profonde reconnaissance pour la qualité de l'accompagnement et l'exigence bienveillante qui ont façonné mon chemin.

Chaque étape a été une opportunité de grandir, d'apprendre, et de croire en mes capacités, portée par un environnement où le savoir-faire se conjugue à l'esprit d'initiative.

Je tiens à remercier tout particulièrement Monsieur ABID Abdeladim, pour son regard juste, ses conseils précis, et sa présence attentive.

Son exigence, sa patience et sa capacité à toujours nous pousser à donner le meilleur de nous-mêmes ont profondément marqué mon évolution.

Son accompagnement n'a pas seulement enrichi mes compétences techniques ; il a forgé en moi une manière plus juste d'apprendre, de douter, et de grandir.

À travers ces lignes discrètes, je rends hommage à toutes celles et ceux qui, avec patience et conviction, ont nourri en moi l'envie d'aller toujours plus loin.

# Résumé

Dans le cadre pédagogique de notre formation à YouCode, nous étions amenés à élaborer un travail dans lequel nous présenterons nos connaissances cognitives et mettrons en pratique nos compétences acquises durant notre formation. C'est pour cela que chacun de nous est chargé de créer son propre projet.

Mon projet consiste à créer une application web RUBI avec les technologies: Laravel 10, PHP, PostgreSQL, en suivant le pattern Repository et Service.

J'ai commencé mon projet par la création des diagrammes de conception tels que diagramme de cas d'utilisation, diagramme de séquence, diagramme de classes, ensuite le design des interfaces des applications, et par la suite le codage des interfaces et les pages d'application. Après j'ai ajouté le codage de la partie backend, dont j'ai établi la connexion avec la base donnée PostgreSQL et enfin j'ai assuré que le fonctionnement est réalisé en utilisant Laravel 10 avec le pattern Repository et Service.

La réalisation de ce projet passe par trois étapes principales :

- Présentation du cahier des charges et des outils de développement
- Analyse et conception
- Réalisation et mise en œuvre de l'application

# Sommaire

**Remerciements**

**Résumé**

**Sommaire**

**Chapitre 1 : Contexte du projet et cahier des charges**

**1.1 Idée générale du projet:** 6

**1.2 Problématique** 7

**1.3 Solutions proposées** 8

**1.4 Étude des besoins** 10

**1.5 Acteurs** 12

**1.6 Conclusion**

**Chapitre 2 : Analyse et Conception** 14

**2.1 Architecture logicielle** 15

**2.2 Modélisation UML** 23

**2.3 Conception de la base de données** 23

**2.4 Conclusion**

**Chapitre 3 : Réalisation et mise en oeuvre du projet**

**3.1 Outils utilisés** 25

**3.2 Les interfaces** 29

**Webographie**

# Introduction

Mon projet est réalisé dans le cadre du projet de fil rouge afin de valider les compétences acquises lors de ma formation à YouCode et approfondir ces connaissances dans une mise en situation réelle sur un projet lié à notre formation.

Ce projet sert à la conception et au développement d'une application Web RUBI en utilisant Laravel 10 et en suivant le pattern Repository et Service. L'application utilise PostgreSQL comme système de gestion de base de données et Blade comme moteur de template pour les vues.

Ce rapport est subdivisé en trois chapitres : le premier décrit l'idée générale du projet et le cahier de charge, le deuxième présente théoriquement la spécification des besoins dont on élabore les diagrammes de cas d'utilisation, le diagramme de classe, le diagramme de séquence, le troisième est consacré à la représentation d'une vue globale sur l'application dans son état final tout en présentant les différentes interfaces de cette dernière, et finalement une conclusion.

# Chapitre 1 : Contexte du projet et Cahier des charges

Ce premier chapitre établit les fondations du projet RUBI, une application web dédiée à la promotion du don de sang. Il présente l'idée générale, la problématique adressée, les solutions proposées, ainsi que les besoins fonctionnels et non-fonctionnels identifiés. Ces éléments s'appuient sur le cahier des charges initial fourni pour ce projet de fin de première année.

## 1.1 Idée générale du projet:

### 1.1.1 Mission et objectif

RUBI est une application web conçue pour transformer l'expérience du don de sang à travers une plateforme numérique complète. Sa mission est triple : sensibiliser le public à l'importance vitale de ce geste, faciliter le processus pour les donneurs potentiels et réguliers, et créer une communauté engagée autour de cette cause essentielle.

Face aux besoins constants en produits sanguins et à la pénurie chronique de donneurs, RUBI se positionne comme un pont numérique entre les citoyens et les établissements de transfusion sanguine, contribuant ainsi directement à sauver des vies.

### 1.1.2 Public cible

L'application s'adresse à trois groupes d'utilisateurs distincts :

- **Visiteurs (non authentifiés)** : Toute personne cherchant des informations sur le don de sang ou souhaitant vérifier son éligibilité avant de s'engager.
- **Donneurs (authentifiés)** : Utilisateurs enregistrés qui suivent leur parcours de don, consultent leur historique, partagent leur expérience et restent informés des besoins en sang.
- **Administrateurs** : Représentants des centres de transfusion ou gestionnaires de la plateforme, responsables du contenu, de la modération et du suivi des activités.

### 1.1.3 Fonctionnalités principales

RUBI propose un écosystème complet de fonctionnalités pour répondre aux besoins de chaque utilisateur :

- **Espace informatif** : Articles, FAQ et ressources éducatives sur le don de sang
- **Outil d'auto-évaluation** : Questionnaire interactif pour vérifier l'éligibilité au don
- **Espace personnel donneur** : Profil, historique des dons, résultats sérologiques et statistiques personnalisées
- **Gestion des rendez-vous** : Planification et suivi des rendez-vous pour les dons
- **Dimension communautaire** : Publication d'expériences, témoignages et interactions entre donneurs
- **Interface d'administration** : Outils de gestion du contenu, des utilisateurs et des statistiques

## 1.2 Problématique

### 1.2.1 Enjeu central

Le projet RUBI répond à un enjeu de santé publique majeur : la pénurie chronique de dons de sang volontaires face à des besoins médicaux constants et croissants. Selon l'Organisation Mondiale de la Santé, seuls 62 pays atteignent un taux de dons suffisant pour répondre à leurs besoins nationaux, tandis que les besoins augmentent de 10% chaque année dans le monde.

### 1.2.2 Obstacles identifiés

Cette pénurie s'explique par plusieurs facteurs qui freinent l'engagement des donneurs potentiels :

- **Barrières psychologiques** : Peur de la douleur, anxiété face aux aiguilles ou au sang

- **Désinformation** : Idées reçues sur les impacts sur la santé ou les critères d'éligibilité
- **Manque de sensibilisation** : Méconnaissance de l'urgence et de l'impact réel du don
- **Contraintes logistiques** : Difficultés à identifier les lieux et horaires de collecte, perception d'un processus chronophage
- **Expériences négatives** : Mauvais souvenirs de dons précédents ou d'interactions avec le personnel
- **Absence d'engagement durable** : Manque de reconnaissance et de suivi personnalisé

### 1.2.3 Importance vitale

Cette problématique est cruciale car le sang et ses composants sont indispensables dans de nombreuses situations médicales :

- Interventions chirurgicales complexes
- Traitements des accidents et traumatismes graves
- Prise en charge des accouchements compliqués
- Traitement des maladies chroniques (anémie, leucémie, hémophilie)
- Soins aux patients cancéreux sous chimiothérapie
- Soins aux nouveau-nés prématurés

Un seul don peut sauver jusqu'à trois vies grâce à la séparation en trois produits sanguins labiles (globules rouges, plaquettes et plasma), soulignant l'impact considérable de chaque contribution.

## 1.3 Solutions proposées

### 1.3.1 Approche stratégique

RUBI aborde cette problématique complexe à travers une approche numérique intégrée qui agit sur plusieurs leviers :



- **Éducation et information** : Centralisation de ressources fiables et accessibles pour combattre la désinformation et démystifier le processus de don
- **Simplification du parcours** : Outils pratiques (éligibilité rapide, géolocalisation des centres) pour réduire les freins logistiques et informationnels
- **Engagement et fidélisation** : Création d'un lien durable avec les donateurs via un suivi personnalisé et des fonctionnalités communautaires
- **Valorisation de l'impact** : Visualisation concrète des résultats (vies sauvées, statistiques) et reconnaissance symbolique pour renforcer la motivation

### **1.3.2 Architecture fonctionnelle**

L'application est structurée autour de trois espaces principaux, chacun avec des fonctionnalités dédiées :

#### **Espace Visiteur (Public)**

- Consultation des informations générales sur le don de sang
- Accès à l'outil d'éligibilité préliminaire
- Localisation des centres de don les plus proches
- Inscription pour devenir donneur

#### **Espace Donneur (Authentifié)**

- Gestion du profil personnel
- Consultation de l'historique des dons et des résultats sérologiques
- Suivi des statistiques personnalisées (dons effectués, vies sauvées)
- Planification et gestion des rendez-vous
- Vérification détaillée de l'éligibilité
- Publication et gestion de témoignages et d'expériences
- Interaction avec la communauté des donateurs

## **Espace Administrateur**

- Tableau de bord avec statistiques en temps réel
- Gestion des utilisateurs et des profils donneurs
- Suivi et modification des informations relatives aux dons
- Modération des publications communautaires
- Création et gestion du contenu éditorial
- Supervision des rendez-vous programmés

## **1.4 Étude des besoins**

### **1.4.1 Besoins fonctionnels**

Les besoins fonctionnels définissent les actions que le système doit exécuter :

#### **1. Gestion des utilisateurs**

1. Inscription et authentification sécurisée
2. Gestion des profils et des rôles (visiteur, donneur, administrateur)
3. Récupération et modification des informations personnelles

#### **2. Gestion de l'information**

1. Consultation des ressources éducatives sur le don de sang
2. Évaluation de l'éligibilité via un questionnaire interactif
3. Localisation des centres de don les plus proches

#### **3. Gestion des dons**

1. Suivi de l'historique des dons pour chaque donneur
2. Accès aux résultats sérologiques personnels
3. Génération de statistiques personnalisées (dons effectués, impact)

#### **4. Gestion des rendez-vous**

1. Planification de nouveaux rendez-vous pour les dons
2. Consultation des rendez-vous passés, présents et futurs

#### **5. Interaction communautaire**

1. Publication et gestion de témoignages et d'expériences
2. Commentaires sur les publications d'autres donneurs
3. Partage de contenu sur les réseaux sociaux

#### **6. Administration de la plateforme**

1. Gestion complète des utilisateurs et de leurs données
2. Modération des publications communautaires
3. Création et édition du contenu éditorial
4. Analyse des statistiques d'utilisation et de dons

#### **1.4.2 Besoins non-fonctionnels**

Les besoins non-fonctionnels définissent les qualités et contraintes du système :

##### **1. Sécurité**

1. Protection des données personnelles et médicales (confidentialité, intégrité)
2. Authentification sécurisée et gestion des droits d'accès
3. Conformité aux réglementations sur la protection des données de santé
4. Prévention des failles web courantes (injections SQL, XSS, CSRF)

##### **2. Performance**

1. Temps de réponse rapide pour les requêtes utilisateurs (< 2 secondes)

2. Chargement efficace des pages et des données

### **3. Ergonomie et utilisabilité**

1. Interface intuitive, claire et accessible à tous les publics
2. Navigation fluide entre les différentes fonctionnalités
3. Design responsive adapté à tous les appareils
4. Accessibilité conforme aux standards WCAG

### **4. Maintenabilité**

1. Architecture logicielle modulaire suivant le pattern Repository et Service
2. Code source organisé, documenté et respectant les conventions Laravel
3. Facilité d'évolution et d'ajout de nouvelles fonctionnalités

### **5. Compatibilité**

1. Fonctionnement optimal sur les principaux navigateurs (Chrome, Firefox, Safari, Edge)
2. Adaptation aux différentes tailles d'écrans (ordinateur, tablette, mobile)

## **1.5 Acteurs**

Le système RUBI interagit avec trois types d'acteurs principaux, chacun ayant des rôles et des responsabilités spécifiques :

### **1.5.1 Visiteur**

Le visiteur représente tout utilisateur non authentifié naviguant sur les parties publiques de l'application.

#### **Caractéristiques :**

- Accès limité aux fonctionnalités informatives et préliminaires
- Aucune donnée personnelle stockée de façon permanente

- Potentiel futur donneur à convertir

#### **Responsabilités :**

- S'informer sur le don de sang et son importance
- Vérifier son éligibilité préliminaire
- S'inscrire pour devenir donneur (conversion)

#### **1.5.2 Donneur**

Le donneur est un utilisateur inscrit et authentifié, au cœur du système RUBI.

#### **Caractéristiques :**

- Profil personnel complet avec données médicales
- Historique de dons et résultats sérologiques
- Membre actif de la communauté

#### **Responsabilités :**

- Maintenir son profil à jour
- Planifier et honorer ses rendez-vous de don
- Consulter ses résultats et son historique
- Partager son expérience (optionnel)
- Interagir avec la communauté (optionnel)

#### **1.5.3 Administrateur**

L'administrateur dispose d'un accès privilégié à l'ensemble des données et fonctionnalités de la plateforme.

#### **Caractéristiques :**

- Droits étendus sur l'ensemble du système
- Responsabilité de la qualité et de l'intégrité des données

- Rôle de modération et de supervision

### **Responsabilités :**

- Gérer les utilisateurs et leurs données
- Modérer le contenu communautaire
- Créer et éditer le contenu éditorial
- Superviser les rendez-vous et les dons
- Analyser les statistiques d'utilisation
- Assurer la conformité aux règles et réglementations

## **1.6 Conclusion**

Ce chapitre a posé les fondations du projet RUBI en définissant clairement son objectif, la problématique qu'il adresse, les solutions proposées, ainsi que les besoins fonctionnels et non-fonctionnels identifiés. L'application RUBI se positionne comme une réponse technologique innovante à un enjeu de santé publique majeur : la pénurie de dons de sang face à des besoins médicaux constants.

En combinant information, facilitation du processus et engagement communautaire, RUBI ambitionne de transformer le don de sang d'un acte médical occasionnel en une habitude citoyenne régulière, contribuant ainsi directement à sauver des vies.

Le chapitre suivant détaillera l'analyse et la conception technique du projet, en présentant l'architecture logicielle et les diagrammes UML qui guideront le développement de l'application.

## Chapitre 2 : Analyse et Conception

Ce deuxième chapitre présente l'analyse et la conception détaillées du projet RUBI. Il expose l'architecture logicielle adoptée, les patterns de conception implémentés, ainsi que la modélisation UML qui structure le développement. Cette phase est cruciale car elle traduit les besoins fonctionnels et non-fonctionnels identifiés précédemment en une architecture technique robuste et évolutive.

### 2.1 Architecture logicielle

#### 2.1.1 Architecture MVC

Le projet RUBI s'appuie sur l'architecture Modèle-Vue-Contrôleur (MVC) fournie par le framework Laravel 10. Cette architecture permet une séparation claire des responsabilités, facilitant ainsi la maintenance et l'évolution de l'application.

##### Principes fondamentaux du MVC

L'architecture MVC décompose l'application en trois composants interconnectés :

- **Modèle:** Représente les données et la logique métier de l'application
- **Vue :** Gère l'interface utilisateur et la présentation des données
- **Contrôleur :** Traite les requêtes des utilisateurs et coordonne les interactions entre le modèle et la vue

##### Implémentation dans Laravel 10

Dans le contexte de Laravel 10, ces composants sont organisés comme suit :

- **Modèles:** Classes PHP étendant `\Illuminate\Database\Eloquent\Model`, représentant les tables de la base de données et encapsulant la logique d'accès aux données

- **Vues** : Templates Blade (`.blade.php`) contenant le code HTML et les directives de présentation, avec une séparation claire entre la logique de présentation et la logique métier
- **Contrôleurs** : Classes PHP étendant `App\Http\Controllers\Controller`, responsables de traiter les requêtes HTTP et de coordonner la réponse de l'application

## **Flux d'exécution**

Le flux d'exécution typique dans l'architecture MVC de RUBI suit ces étapes :

1. L'utilisateur interagit avec l'interface (Vue) en effectuant une action
2. Cette action génère une requête HTTP traitée par le routeur de Laravel
3. Le routeur redirige la requête vers le Contrôleur approprié
4. Le Contrôleur interagit avec les Services et Repositories pour traiter la demande
5. Les Repositories communiquent avec les Modèles pour accéder aux données
6. Le Contrôleur prépare les données et les transmet à la Vue
7. La Vue génère le HTML final présenté à l'utilisateur

### **2.1.2 Pattern Repository et Service**

Pour renforcer la séparation des responsabilités et améliorer la maintenabilité du code, le projet RUBI implémente les patterns Repository et Service au-dessus de l'architecture MVC de base.

#### **Pattern Repository**

Le pattern Repository agit comme une couche d'abstraction entre la logique métier et la couche d'accès aux données, offrant plusieurs avantages :

- **Isolation de la logique d'accès aux données** : Les détails d'implémentation de la persistance sont encapsulés



- **Facilitation des tests unitaires** : Les repositories peuvent être facilement mockés pour les tests
- **Réduction de la duplication de code** : Les opérations communes d'accès aux données sont centralisées
- **Flexibilité accrue** : Possibilité de changer la source de données sans impacter le reste de l'application

Dans RUBI, chaque entité principale dispose de son propre repository, défini par une interface et implémenté par une classe concrète :

L'organisation des dossiers du projet RUBI reflète l'architecture adoptée et facilite la navigation dans le code source :

```
CommentRepositoryInterface.php

<?php

namespace App\Repositories\Interfaces;

6 usages 1 implementation  👤 hajarwalfi
interface CommentRepositoryInterface
{
    1 usage 1 implementation  👤 hajarwalfi
    public function getCommentsByPostId($postId);

    1 usage 1 implementation  👤 hajarwalfi
    public function getCommentById($commentId);

    1 usage 1 implementation  👤 hajarwalfi
    public function createComment(array $commentData);

    1 usage 1 implementation  👤 hajarwalfi
    public function updateComment($commentId, array $newDetails);

    1 usage 1 implementation  👤 hajarwalfi
    public function deleteComment($commentId);
}
```

CommentRepository.php

```
<?php
```

```
namespace App\Repositories\Eloquent;
```

```
use ...
```

2 usages hajarwalfi

```
class CommentRepository implements CommentRepositoryInterface
```

```
{
```

1 usage hajarwalfi

```
public function getCommentsByPostId($postId)
```

```
{
```

```
    return Comment::with( relations: 'user')->where( column: 'post_id', $postId)->latest()->get();
```

```
}
```

1 usage hajarwalfi

```
public function getCommentById($commentId)
```

```
{
```

```
    return Comment::with( relations: 'user')->find($commentId);
```

```
}
```

1 usage hajarwalfi

```
public function createComment(array $commentData)
```

```
{
```

```
    return Comment::create($commentData);
```

```
}
```

1 usage hajarwalfi

```
public function updateComment($commentId, array $newDetails)
```

```
{
```

```
    return Comment::whereId($commentId)->update($newDetails);
```

```
}
```

1 usage hajarwalfi

```
public function deleteComment($commentId)
```

```
{
```

```
    return Comment::destroy($commentId);
```

```
}
```

```
}
```

## **Pattern Service**

Le pattern Service encapsule la logique métier complexe qui ne devrait pas être placée dans les contrôleurs ou les modèles :

- **Séparation des préoccupations** : Isole la logique métier de la logique de présentation
- **Réutilisabilité** : Permet d'utiliser la même logique métier dans différents contrôleurs
- **Testabilité améliorée** : Facilite les tests unitaires de la logique métier
- **Respect du principe de responsabilité unique** : Chaque service a une responsabilité bien définie

Dans RUBI, les services orchestrent les opérations complexes en utilisant un ou plusieurs repositories :

CommentService.php

```
<?php
```

```
namespace App\Services;
```

```
use ...
```

```
4 usages hajarwalfi
```

```
class CommentService
```

```
{
```

```
6 usages
```

```
protected CommentRepository;
```

```
no usages hajarwalfi
```

```
public function __construct(CommentRepositoryInterface $commentRepository)
```

```
{
```

```
    $this->commentRepository = $commentRepository;
```

```
}
```

```
no usages hajarwalfi
```

```
public function getCommentsByPostId($postId)
```

```
{
```

```
    return $this->commentRepository->getCommentsByPostId($postId);
```

```
}
```

```
3 usages hajarwalfi
```

```
public function getCommentById($commentId)
```

```
{
```

```
    return $this->commentRepository->getCommentById($commentId);
```

```
}
```

```
1 usage hajarwalfi
```

```
public function createComment($postId, $content)
```

```
{
```

```
    return $this->commentRepository->createComment([
```

```
        'post_id' => $postId,
```

```
        'user_id' => Auth::id(),
```

```
        'content' => $content,
```

```
    ]);
```

```
}
```

```
1 usage hajarwalfi
```

```
public function updateComment($commentId, $content)
```

```
{
```

```
    return $this->commentRepository->updateComment($commentId, [
```

```
        'content' => $content,
```

```
    ]);
```

```
}
```

```
2 usages hajarwalfi
```

```
public function deleteComment($commentId)
```

```
{
```

```
    return $this->commentRepository->deleteComment($commentId);
```

```
}
```

```
}
```

## Enregistrement des bindings

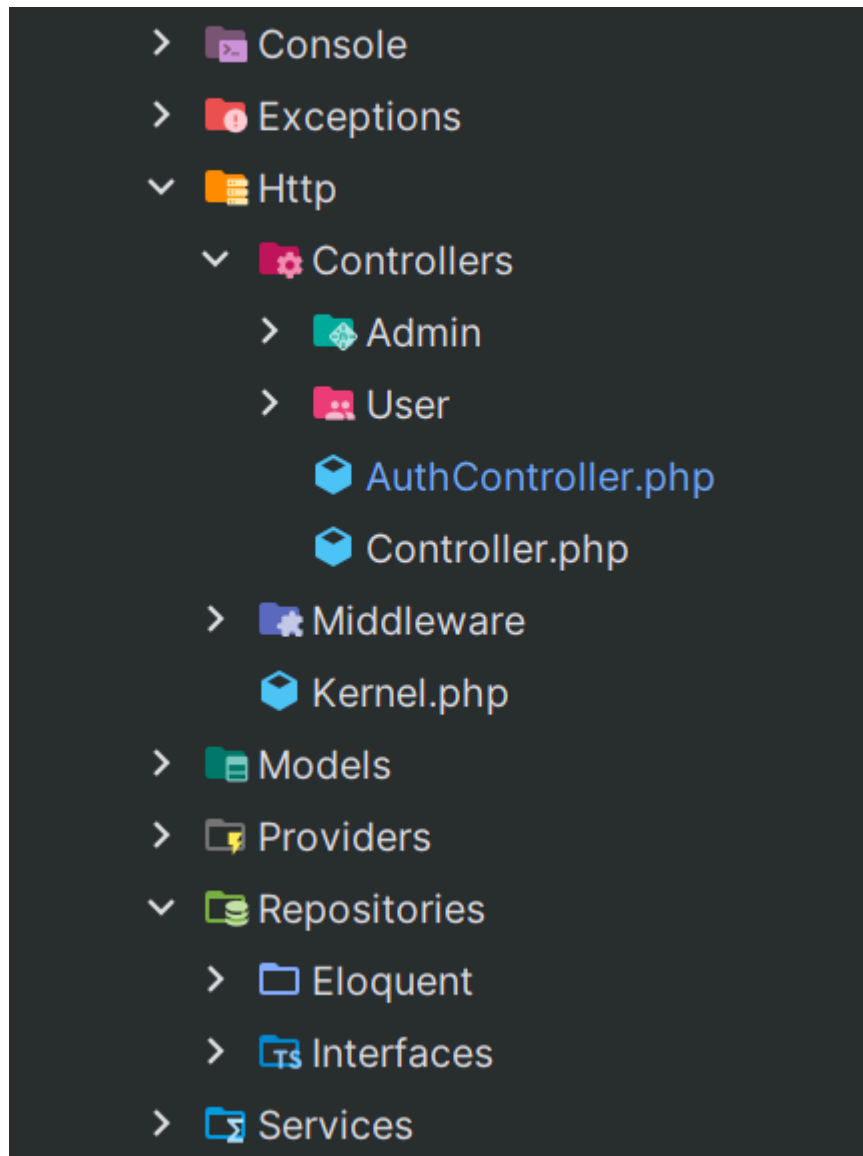
Pour que Laravel puisse injecter les implémentations concrètes lorsque les interfaces sont demandées, les bindings sont enregistrés dans un service provider dédié :

```
AppServiceProvider.php

class AppServiceProvider extends ServiceProvider
{
    /**
     * walfihajar +1 *
     *
     * @return void
     */
    public function register(): void
    {
        $this->app->bind(abstract: UserRepositoryInterface::class, concrete: UserRepository::class);
        $this->app->bind(abstract: DonationRepositoryInterface::class, concrete: DonationRepository::class);
        $this->app->bind(abstract: SerologyRepositoryInterface::class, concrete: SerologyRepository::class);
        $this->app->bind(abstract: ObservationRepositoryInterface::class, concrete: ObservationRepository::class);
        $this->app->bind(abstract: ObservationService::class, function ($app) {
            return new ObservationService($app->make(ObservationRepositoryInterface::class));
        });
        $this->app->bind(abstract: SerologyService::class, function ($app) {
            return new SerologyService($app->make(SerologyRepositoryInterface::class));
        });
        $this->app->bind(abstract: ArticleRepositoryInterface::class, concrete: ArticleRepository::class);
        $this->app->bind(abstract: PostRepositoryInterface::class, concrete: PostRepository::class);
        $this->app->bind(abstract: CommentRepositoryInterface::class, concrete: CommentRepository::class);
        $this->app->bind(abstract: EligibilityRepositoryInterface::class, concrete: EligibilityRepository::class);
        $this->app->bind(abstract: AppointmentRepositoryInterface::class, concrete: AppointmentRepository::class);
        $this->app->bind(abstract: AppointmentService::class, function ($app) {
            return new AppointmentService(
                $app->make(AppointmentRepositoryInterface::class)
            );
        });
    }
}
```

### 2.1.3 Structure des dossiers

L'organisation des dossiers du projet RUBI reflète l'architecture adoptée et facilite la navigation dans le code source :



Pour l'exemple du commentaire mentionné, cette structure permet une implémentation claire :

- Le modèle **Comment** se trouve dans le dossier **app/Models**
- L'interface **CommentRepositoryInterface** est définie dans **app/Repositories/Interfaces**

- L'implémentation concrète **CommentRepository** est placée dans **app/Repositories/Eloquent**
- Le service **CommentService** qui encapsule la logique métier est dans **app/Services**
- Les contrôleurs qui utilisent ce service sont dans les dossiers appropriés sous **app/Http/Controllers**

Cette organisation facilite la maintenance et l'évolution du code, tout en permettant une séparation claire entre l'accès aux données, la logique métier et la présentation.

## **2.2 Modélisation UML**

La modélisation UML (Unified Modeling Language) permet de représenter visuellement la structure et le comportement du système RUBI. Cette approche facilite la compréhension du système par toutes les parties prenantes et guide le développement.

### **2.2.1 Diagramme de cas d'utilisation**

Le diagramme de cas d'utilisation illustre les interactions entre les acteurs et le système, mettant en évidence les fonctionnalités principales de l'application RUBI.

### **2.2.2 Diagramme de classes**

Le diagramme de classes représente la structure statique du système, montrant les classes, leurs attributs, leurs méthodes et les relations entre elles.

## **2.3 Conception de la base de données**

La conception de la base de données est un élément crucial du projet RUBI, assurant une gestion efficace et cohérente des données.

### **2.3.1 Choix du SGBD**

Le projet RUBI utilise PostgreSQL comme système de gestion de base de données relationnelle pour plusieurs raisons :

- **Robustesse et fiabilité** : PostgreSQL est reconnu pour sa stabilité et son intégrité des données
- **Support avancé des types de données** : Notamment pour les données JSON, géospatiales et les tableaux
- **Performances optimales** : Gestion efficace des requêtes complexes et des grandes quantités de données
- **Conformité ACID** : Garantie de la cohérence des transactions
- **Extensibilité** : Possibilité d'ajouter des fonctionnalités via des extensions

## 2.4 Conclusion

Ce chapitre présente l'analyse et la conception détaillées du projet RUBI. L'architecture adoptée, combinant le pattern MVC de Laravel avec les patterns Repository et Service, offre une base solide pour le développement d'une application robuste, maintenable et évolutive.

La modélisation UML a permis de clarifier les interactions entre les différents acteurs et composants du système, tandis que la conception minutieuse de la base de données garantit une gestion efficace et cohérente des données.

Le chapitre suivant détaillera la mise en œuvre concrète de cette conception, présentant les principales fonctionnalités développées et les interfaces utilisateur de l'application RUBI.



## **Chapitre 3 : Réalisation et mise en oeuvre du projet**

Ce chapitre présente la phase de réalisation technique du projet RUBI. Il détaille les outils et technologies utilisés, le déroulement du développement, ainsi que les interfaces principales de l'application. Cette étape concrétise les analyses et conceptions élaborées dans les chapitres précédents, transformant les spécifications en une application fonctionnelle.

### **3.1 Outils utilisés**

La sélection des outils et technologies a été guidée par plusieurs critères : robustesse, maintenabilité, performance et adéquation avec les besoins spécifiques du projet RUBI.

#### **3.1.1 Langages de programmation**

##### **PHP 8.2:**

PHP constitue le langage principal du projet RUBI, dans sa version 8.2 qui apporte des améliorations significatives :

- Typage strict : Amélioration de la robustesse du code grâce au typage des propriétés, des paramètres et des valeurs de retour
- Attributs natifs : Simplification des annotations et métadonnées
- Fonctions de première classe : Manipulation plus flexible des fonctions
- Performance optimisée : Exécution plus rapide par rapport aux versions précédentes
- Gestion améliorée des erreurs : Détection précoce des problèmes potentiels

##### **JavaScript:**

JavaScript est utilisé pour les interactions côté client :

- Manipulation du DOM : Modifications dynamiques du contenu de la page

## **HTML5 et CSS3:**

Ces langages de balisage et de style forment la base de l'interface utilisateur :

- Structure sémantique : Organisation logique du contenu avec HTML5
- Responsive design : Adaptation à tous les formats d'écran avec CSS3
- Animations et transitions : Amélioration de l'expérience utilisateur
- Accessibilité : Respect des normes pour une application inclusive

### **3.1.2 Frameworks et bibliothèques**

#### **Laravel 10**

Laravel 10 constitue le framework principal du projet, offrant une base solide pour le développement :

- Architecture MVC : Organisation claire du code
- Eloquent ORM : Manipulation intuitive de la base de données
- Blade : Moteur de templates puissant et expressif
- Middleware : Filtrage et traitement des requêtes HTTP
- Artisan : Outils en ligne de commande pour automatiser les tâches répétitives
- Migration : Gestion versionnée du schéma de base de données
- Authentication : Système d'authentification robuste et personnalisable

#### **Tailwind CSS**

Tailwind CSS a été choisi pour le développement frontend :

- Approche utility-first : Classes utilitaires pour un développement rapide

- Personnalisation poussée : Adaptation précise au design de RUBI
- Responsive design : Classes adaptatives pour tous les formats d'écran
- Optimisation de production : Purge des classes non utilisées pour minimiser la taille
- Intégration avec Blade : Utilisation fluide dans les templates Laravel

### **3.1.3 Logiciels et environnements**

#### **PhpStorm**

IDE principal pour le développement du projet :

- Support natif de Laravel : Autocomplétion, navigation, refactoring
- Débogage intégré : Points d'arrêt, inspection des variables
- Intégration de base de données : Accès direct à PostgreSQL
- Contrôle de version : Intégration Git native
- Analyse de code : Détection des problèmes potentiels et suggestions d'amélioration

#### **XAMPP**

Environnement de développement local :

- Apache : Serveur web
- MariaDB/MySQL : Système de gestion de base de données (utilisé pour les tests)
- PHP : Interpréteur PHP configuré
- phpMyAdmin : Interface de gestion de base de données
- Configuration simplifiée : Démarrage rapide du développement

#### **PostgreSQL**

Système de gestion de base de données relationnelle :

- Robustesse : Fiabilité et intégrité des données
- Performance : Optimisation pour les requêtes complexes
- Types de données avancés : Support de types spécifiques (JSON, géographiques)
- Extensibilité : Fonctions et procédures stockées
- Conformité SQL : Respect des standards

## **Figma**

Outil de conception d'interface utilisé pour les maquettes :

- Design collaboratif : Travail simultané sur les maquettes
- Prototypage interactif : Test des parcours utilisateur
- Système de composants : Cohérence visuelle à travers l'application
- Inspection de code : Extraction des valeurs CSS pour Tailwind
- Exportation d'assets : Génération optimisée des ressources graphiques

## **Lucidchart**

Outil de modélisation pour les diagrammes UML :



- Diagrammes de cas d'utilisation : Visualisation des interactions utilisateur
- Diagrammes de classes : Modélisation de la structure du système
- Diagrammes de séquence : Représentation des flux d'interactions
- Diagrammes d'activité : Modélisation des processus métier
- Collaboration en temps réel : Travail d'équipe sur les modèles


## **Git et GitHub**

Outils de gestion de version et de collaboration :

- Versionnage : Suivi des modifications du code
- Branches : Développement parallèle de fonctionnalités
- Pull requests : Revue de code et intégration
- Issues : Suivi des bugs et améliorations
- Documentation : Wiki et README pour la documentation du projet

## 3.2 Les interfaces


[Home](#)
[Articles](#)
[Community](#)
[Eligibility](#)




**FEATURED ARTICLE**

### First Time Donating Blood? Here's Everything You Need to Know

Thinking about donating blood for the first time? That's amazing! It's a generous and lifesaving act—but we know it can...

[Read full article](#)

02 **Can I Donate Blood If...? Answering Common Eligibility Questions (Tattoos, Medication, Travel)**

You want to donate blood – that's fantastic! It's a generous act with a huge impact. But maybe you hesitate, thinking, "...

[Read more](#)

03 **The Science Behind Saving Lives: Where Does Your Donated Blood Go?**


You've done it – you generously donated a pint of blood. As you relax with your juice and cookies, you might wonder, "Wh...

[Read more](#)


04 **More Than Just a Pint: The Health Checks and Benefits for Blood Donors**


We all know the main reason to donate blood: it's a selfless act that saves lives. Giving just one pint can help multipl...

[Read more](#)



[Home](#) [Articles](#) [Community](#) [Eligibility](#)





**Hajar Walfi**

0

Posts

0

Comments

9

Donations

My Posts

Edit Profile

Donation History

Share

H

Title (optional)

Share your experience...

Media

Publish

RE

**Reda Elkissia**  
1 day ago

**Why Blood Donation Matters**

Blood donations don't just help save lives—they help heal communities, support medical treatments, and create a network of care. Let's come together and ensure there's always enough blood for those in need


14 Views

1 Comment


View Details

RE

**Reda Elkissia**  
1 day ago



[Home](#) [Articles](#) [Community](#) [Eligibility](#)



RE

**Reda Elkissia**  
1 day ago

**Why Blood Donation Matters**

Blood donations don't just help save lives—they help heal communities, support medical treatments, and create a network of care. Let's come together and ensure there's always enough blood for those in need

14 Views

1 Comment

Comments

IJ

**Ichrak Jaifra**  
1 day ago

Thank you for this inspirational post

HW

Write a comment...

Post Comment

[Home](#)[Articles](#)[Community](#)[Eligibility](#)

## Blood Donation Eligibility

Your generosity can save lives. Complete this quick assessment to check if you're eligible to donate blood today.

### ABOUT THIS TEST

#### Why This Test Matters

This preliminary test allows you to assess whether you meet the general conditions for donating blood. It does not replace a medical interview but can help you determine if you should visit a donation center.

Completing this assessment will give you immediate feedback on your eligibility status based on common donation criteria.

#### Why Your Donation Matters

Blood donation is a life-saving act of kindness. Before you donate, it's important to ensure you meet the basic requirements. This quick eligibility check will help you determine if you can donate blood today.

### Eligibility Questionnaire

Please answer all questions accurately



Age

☐ Under 18 years

☐ 18-65 years

☐ Over 65 years



Weight

☐ Under 50kg (110 lbs)

☐ 50kg (110 lbs) or more



Have you been ill in the past 14 days?

☐ Yes

☐ No



RUBI

#### ADMINISTRATION

[Admin Dashboard](#)

#### MEDICAL RECORD

[My Donations](#)

[My Appointments](#)

#### COMMUNITY

[My Posts](#)

[My Comments](#)

#### PERSONAL INFORMATION

[My Profile](#)

[Logout](#)

## My Posts

Manage your posts and track their status

0

TOTAL POSTS



0

APPROVED



0

PENDING



0

TOTAL VIEWS






























No posts found

You haven't created any posts with this status yet.


[Create a post](#)

# My Donations

Track your donation history and view your serological results

<div>9</div> <div>TOTAL DONATIONS</div> <div></div>	<div>27</div> <div>LIVES SAVED</div> <div></div>	<div>9000</div> <div>LOYALTY POINTS</div> <div></div>
<div><div> DON-2025-003</div><div>22/02/2220 at safi</div><div><div><div> TPHA</div><div>Pending</div></div><div><div> Hep B</div><div>Pending</div></div><div><div> Hep C</div><div>Pending</div></div><div><div> HIV</div><div>Pending</div></div></div></div>	<div><div> DON-2025-001</div><div>11/04/2025 at safi</div><div><div><div> TPHA</div><div>Negative</div></div><div><div> Hep B</div><div>Negative</div></div><div><div> Hep C</div><div>Negative</div></div><div><div> HIV</div><div>Negative</div></div></div></div>	<div><div> DON-2024-004</div><div>11/12/2024 at Casablanca</div><div><div><div> TPHA</div><div>Negative</div></div><div><div> Hep B</div><div>Negative</div></div><div><div> Hep C</div><div>Negative</div></div><div><div> HIV</div><div>Negative</div></div></div></div>
<div><div> DON-2024-005</div><div>11/08/2024 at Rabat</div><div><div><div> TPHA</div><div>Negative</div></div><div><div> Hep B</div><div>Negative</div></div></div></div>	<div><div> DON-2024-006</div><div>11/04/2024 at Marrakech</div><div><div><div> TPHA</div><div>Negative</div></div><div><div> Hep B</div><div>Negative</div></div></div></div>	<div><div> DON-2023-007</div><div>11/12/2023 at Tangier</div><div><div><div> TPHA</div><div>Negative</div></div><div><div> Hep B</div><div>Negative</div></div></div></div>


# My Profile




Hajar Walfi

admin@rubi.com

Member since April 2025

 O+

 Personal Information

First Name

Hajar

Last Name


Walfi

Phone Number

0691754845

Date of Birth

24/07/2002



Blood Type

O+

Gender

Female

CNI (National ID)

HH126910



RUBI Admin

MANAGEMENT

Users

Posts

Articles

Appointments

Donor Dashboard

Logout

## Donors

Manage blood donors registered in the RUBI application

Total Donors

33

All registered donors

Eligible Donors

61 %

Ready for blood donation

Ineligible

48 %

Temporarily excluded

Unconfirmed

0

Without ID number

Donor Management

View and manage blood donors registered in the RUBI application

Search for a donor...

EligibleIneligible

First & Last Name	Blood Group	Status	Phone	Email	Eligibility
Hamza Braik	B-	Regular	0614171819	hamza.braik@gmail.com	Eligible
Hamza Gbouri	O-	Regular	060481975	hamza.gbouri@gmail.com	Ineligible
Wissam Douskary	A+	Regular	0691478546	wissam.douskary@gmail.com	Eligible
Hajar Walfi	O+	Regular	0691754845	admin@rubi.com	Ineligible
Meriem Walfi	O+	Regular	0612345678	meriem.walfi@gmail.com	Eligible

RUBI Admin

MANAGEMENT

Users

Posts

Articles

Appointments

Donor Dashboard

Logout

Donors > Hamza Braik

Edit Profile

### Hamza Braik

DNR008 Regular

Personal Information

Contact details and donor information

B-

Eligible

Identity Card

UL123456

Birth Date

04/11/2002

Phone

0614171819

Email

hamza.braik@gmail.com

Address

Not provided

Donation History

Blood donation history

Add Donation

ID	DATE	TYPE	SEROLOGY	ACTIONS
DON-2025-074	02/05/2025	Compensated	Negative	...
DON-2024-075	10/05/2024	Voluntary	Negative	...
DON-2024-076	06/05/2024	Compensated	Negative	...
DON-2024-077	02/05/2024	Voluntary	Negative	...
DON-2023-078	10/05/2023	Compensated	Negative	...
DON-2023-079	06/05/2023	Voluntary	Negative	...
DON-2023-080	02/05/2023	Compensated	Negative	...
DONL-2022-DR1	10/05/2022	Voluntary	Negative	...

RUBI Admin

MANAGEMENT

Users

Posts

Articles

Appointments

Donor Dashboard

Logout

### Articles

Manage official publications in the RUBI application

Total Articles

13

All registered articles

Published

76.9 %

Active publications

Drafts

15.4 %

In progress

Archived

7.7 %

Inactive publications

#### Article Management

View and manage articles registered in the RUBI application

+ Add Article

Search for an article...

AllPublishedDraftsArchived

Published

First Time Donating Blood? Here's Everything You Need to Know

04/05/2025

Thinking about donating blood for the first time? That's amazing! It's a generous and lifesaving act—but we know it can also fe...

Archived

Why There's Always a Need for Blood: The Truth About Shortages

04/05/2025

You may have seen blood donation ads asking for help, especially after disasters or accidents. But did you know that the need...

Draft

how donating blood can actually benefit your own health

04/05/2025

How Donating Blood Can Improve Your Health TooMost people donate blood to help others—and that's amazing. But did you...

RUBI Admin

MANAGEMENT

Users

Posts

Articles

Appointments

Donor Dashboard

Logout

### Moderation

Manage and moderate user posts and comments

Pending

0

Approved

10

Rejected

0

Archived

0

User Posts

Pending

Approved

Rejected

Archived

Why Blood Donation Matters

By Reda Elkissia • 2025-05-04

Blood donations don't just help save lives —they he...

1 comments 15 views

Be a Blood Donor — It's the Ultimate Power Move

By Reda Elkissia • 2025-05-04

Forget flexing your muscles — real heroes flex the...

1 comments 11 views

Giving Blood is the New Cool

Why Blood Donation Matters

By Reda Elkissia • 2025-05-04

Approved

Blood donations don't just help save lives—they help heal communities, support medical treatments, and create a network of care. Let's come together and ensure there's always enough blood for those in need

1 15

Comments

Ichrak Jaifra

2025-05-04 11:31

Thank you for this inspirational post



# Webographie

Dans une démarche à la fois rigoureuse et créative, la recherche d'informations pour la conception et le développement de l'application RUBI s'est appuyée sur un ensemble de ressources web sélectionnées avec soin.

Les sources sont ici organisées par thématique pour offrir clarté, cohérence et pertinence.

## A. Contexte du Don de Sang et Stratégies de Sensibilisation

Ces ressources ont permis de comprendre l'enjeu vital du don de sang, d'identifier les besoins spécifiques (notamment au Maroc) et d'analyser des stratégies efficaces de mobilisation.

- Organisation Mondiale de la Santé (OMS)  
Accès aux statistiques mondiales, faits clés sur la sécurité transfusionnelle et recommandations internationales.
- Ministère de la Santé et de la Protection Sociale (Maroc)  
Informations spécifiques au contexte marocain : besoins nationaux, campagnes de dons, localisation des centres de collecte.
- Établissement Français du Sang (EFS)  
Source d'inspiration pour la promotion du don de sang et l'organisation de collectes efficaces.

## B. Design, UI/UX et Inspiration Visuelle

L'objectif ici était de forger une expérience utilisateur moderne, intuitive et émotionnellement engageante, en s'inspirant des meilleures pratiques du design numérique.

- Behance: Exploration de portfolios de designers, recherche de concepts UI/UX inspirants.
- Pinterest: Récolte d'idées visuelles, palettes de couleurs, éléments graphiques pertinents pour RUBI.

- Dribbble: Identification de tendances UI modernes, inspiration pour micro-interactions et animations.
- Figma Community: Recherche de templates UI, kits de design et exemples de prototypage collaboratif.

## **C. Ressources Techniques et Développement Web**

Ces plateformes ont constitué l'ossature technique du projet, soutenant le développement, la résolution de problèmes et la montée en compétences.

- Laravel: Consultation de la documentation officielle pour le développement backend de l'application.
- PHP.net: Accès à la documentation officielle du langage PHP.
- Laracasts: Suivi de tutoriels vidéo spécialisés dans Laravel et PHP pour approfondir les compétences techniques.
- Dev.to: Lecture de tutoriels et d'articles communautaires sur le développement web moderne.
- Stack Overflow: Recherche de solutions rapides aux problèmes de code rencontrés durant le développement.
- Medium: Exploration d'articles sur le développement web, l'UX/UI, et les méthodologies projet.
- Reddit: Participation à des discussions techniques (subreddits dédiés à PHP, Laravel et au développement général).

