Projet de Fin d'Année

Plateforme de partage d'expériences sur le soin de pathologies

Réalisé par :

Saber FADLI
Oussama ETTAOUIL

Encadré par :

Mme. Widad ETTAZI

Membre de Jury :

Mme. Widad ETTAZI
M. Abdellatif EL FAKER

Année académique 2023/2024

Plateforme de partage d'expériences sur le soin de pathologies

Table des matières

Table des figures

Remerciements :

Nous tenons tout d'abord à exprimer nos profonds remerciements à notre professeure, Mme. Widad ETTAZI, de nous avoir encadré tout au long de la réalisation de ce projet. Nous tenons aussi à remercier tous ceux qui nous ont aidé, de près ou de loin, à réaliser ce travail.

Nos remerciements vont également aux membres du jury, Mme. Widad ETTAZI et M. Abdellatif EL FAKER qui ont bien accepté d'évaluer notre travail, ainsi qu'a tout le corps professoral et administratif de l'école qui a toujours veillé à ce que notre formation soit dispensée dans de bonnes conditions.

Résumé :

Ce rapport présente notre projet de fin d'année, conçu pour approfondir et mettre en pratique nos compétences en développement informatique. Nous avons utilisé des techniques de web scraping et d'automatisation avec Python pour extraire des expériences de soin depuis divers sites web, et le modèle Llama pour classifier ces expériences afin de faciliter l'accès aux informations. L'objectif est de créer une plateforme permettant aux utilisateurs de partager et de rechercher des expériences liées aux pathologies.

Nous avons d'abord défini la problématique, les besoins et les objectifs, puis étudié l'architecture du système. La réalisation de l'application a commencé par la programmation des scripts de web scraping pour extraire des données pertinentes en ligne. Ensuite, nous avons conçu et programmé les interfaces utilisateur pour faciliter la navigation et la recherche d'informations.

Un système de gestion des données a été mis en place pour stocker les expériences collectées. Des tests rigoureux ont été effectués à chaque étape pour assurer le bon fonctionnement de l'application. Ce projet, développé en Python et Java avec divers frameworks et bibliothèques, nous a permis de renforcer nos compétences techniques et de créer une ressource précieuse pour ceux cherchant des informations sur les soins de pathologies.

Mots clés :
Web scraping, Automatisation, BeautifulSoup, Selenium, Expériences, LLM, Llama

---

Abstract :

This report presents our year-end project, designed to deepen and apply our skills in software development. We used web scraping and automation techniques with Python to extract care experiences from various websites, and the Llama model to classify these experiences to facilitate information access. The objective is to create a platform that allows users to share and search for experiences related to pathologies.

We first defined the problem, needs, and objectives, then studied the system architecture. The application development began with the programming of web scraping scripts to extract relevant data online. Next, we designed and programmed the user interfaces to facilitate navigation and information search.

A data management system was implemented to store the collected experiences. Rigorous tests were conducted at each stage to ensure the application functions correctly. This project, developed in Python and Java with various frameworks and libraries, allowed us to strengthen our technical skills and create a valuable resource for those seeking information on care for pathologies.

---

keywords :
Web scraping, Automation, BeautifulSoup, Selenium, Experiences, LLM, Llama

---

Liste des abréviations

NLP	Natural language processing
LLM	Large Language Model
HTTP	Hypertext Transfer Protocol
DOM	Document Object Model
CAPTCHA	Completely Automated Public Turing test to tell Computers and Humans Apart
API	Application Programming Interface
IDE	Integrated Development Environment
LLaMA	Large Language Model Architecture
GUI	graphical user interface
HTML	Hyper Text Markup Language
MVC	Modèle-Vue-Contrôleur
URL	Uniform Resource Locator
NoSQL	Not only SQL

Introduction Générale

De nos jours, nous sommes de plus en plus confrontés à la nécessité de trouver et de partager des informations fiables et pertinentes sur les soins de pathologies. Les expériences individuelles de patients peuvent offrir des perspectives précieuses, mais elles sont souvent dispersées à travers divers sites web. Pour résoudre ce problème, nous avons développé une plateforme de partage d'expériences sur le soin de pathologies, en utilisant des techniques de web scraping et d'automatisation avec Python.

Cette technologie permet d'extraire des témoignages et des expériences déjà vécues à partir de diverses sources en ligne, et de les centraliser dans une plateforme accessible. Les utilisateurs peuvent ainsi rechercher des informations par sujet, facilitant la découverte et l'échange de connaissances sur des traitements et des pathologies spécifiques. Le web scraping permet de collecter automatiquement des données à grande échelle, garantissant une base d'informations toujours à jour et pertinente.

La plateforme étudiée tout au long de ce rapport est un système de partage d'expériences sur le soin de pathologies, conçu pour offrir une interface conviviale et des recherches efficaces. Ce rapport est structuré en trois chapitres. Dans le premier chapitre, nous présentons l'importance du partage d'expériences dans le domaine des soins de santé et les technologies employées pour réaliser ce projet. Puis, nous définissons la problématique, les objectifs et les besoins de notre projet.

Le deuxième chapitre aborde les aspects techniques du web scraping et de l'automatisation en Python. Nous y présentons les différents algorithmes et outils utilisés, ainsi que l'analyse fonctionnelle du projet, qui comprend les exigences techniques, fonctionnelles et non fonctionnelles. Nous détaillons également le diagramme des cas d'utilisation, l'architecture générale du système et les maquettes de l'application.

Enfin, le troisième chapitre est consacré à la réalisation de l'application. Nous y exposons l'architecture technique de la plateforme, les outils utilisés, et les différentes fonctionnalités implémentées et leur fonctionnement. Nous concluons par une série de tests effectués pour assurer la fiabilité et l'efficacité du système, garantissant ainsi une expérience utilisateur optimale et une contribution significative à la communauté des soins de santé.

Chapitre 1

Contexte général du projet

1.1 Introduction

Dans ce premier chapitre du rapport, nous allons mettre en relief le contexte général du projet, en mettant en avant le lien entre le partage d'expériences sur le soin de pathologies et l'utilisation de techniques de web scraping et d'automatisation. Ce chapitre présente également le projet, dans son ensemble, qui vise à développer une plateforme permettant de centraliser et de faciliter l'accès aux témoignages de patients recueillis à travers divers sites web.

1.2 Contexte et objectifs du partage d'expériences

Le partage d'expériences sur les soins de pathologies est crucial pour offrir des perspectives diversifiées aux patients et aux professionnels de la santé. Cependant, ces informations sont souvent dispersées et difficiles à trouver. En utilisant des techniques de web scraping et d'automatisation, nous pouvons extraire systématiquement des témoignages et des expériences de patients à partir de plusieurs sources en ligne et les centraliser sur une plateforme accessible, en garantissant une expérience transparente aux utilisateurs, ces derniers peuvent simplement rechercher les informations dont ils ont besoin sans se soucier de la complexité de la collecte des données ou des processus sous-jacents.

1.3 Utilisation des techniques de Web Scraping et d'Automatisation

Le web scraping consiste à extraire des données de sites web de manière automatisée. En utilisant des bibliothèques Python telles que Beautiful Soup et Selenium, nous pouvons collecter des témoignages pertinents sur les soins de pathologies. Ces données sont ensuite nettoyées, structurées par le biais du modèle LLM : llama, qui se charge de l'analyse de l'expérience partagée et d'en extraire la pathologie en question avec des informations clés et stockées dans une base de données pour permettre une recherche facile et efficace par les utilisateurs, et pour donner une vue simplifiée de l'experience à consulter sans l'avoir consulter déjà.

1.4 Problématique et présentation du projet

1.4.1 Problématique

Le principal défi consiste à collecter, structurer et rendre accessibles des expériences de soins de pathologies provenant de diverses sources en ligne. Comment peut-on optimiser le processus de web scraping pour garantir l'exhaustivité et la pertinence des données extraites ? Comment structurer ces données pour faciliter leur consultation par les utilisateurs ? Enfin, comment rendre l'utilisateur capable de rechercher les expériences qu'il veut et de lui rendre des résultats adaptés à son choix à travers la technologie d'automatisation.

1.4.2 Présentation du projet

Ce projet consiste à réaliser une plateforme de partage d'expériences sur le soin de pathologies. Les utilisateurs peuvent rechercher des témoignages par sujet, cela leur offre la possibilité de découvrir rapidement des informations pertinentes sur des pathologies spécifiques grâce au modèle LLAMA, qui analyse les expériences recueillies et les organise de manière concise et accessible.. La plateforme utilise des techniques de web scraping pour extraire des données de manière automatisée et les centraliser. Les utilisateurs peuvent également contribuer en partageant leurs propres expériences.

1.4.3 Méthodologies de développement

1.4.3.1 Méthodologie Agile

La méthodologie Agile est une approche itérative et incrémentale de la gestion de projet et du développement logiciel. Elle favorise la flexibilité, la collaboration et l'adaptation aux changements tout au long du cycle de vie du projet.

- **Sprints :**

  Diviser le projet en courtes périodes appelées sprints (de 1 à 4 semaines). Chaque sprint a des objectifs spécifiques, comme le développement du module de web scraping, la mise en place de la base de données, ou la création de l'interface utilisateur.

- **Scrum meeting :**

  Organiser des réunions quotidiennes de type Scrum pour synchroniser l'équipe, discuter des progrès réalisés et identifier les obstacles.

- **Product Backlog :**

  Maintenir une liste priorisée des fonctionnalités et tâches à accomplir (product backlog). Réévaluer et ajuster les priorités à la fin de chaque sprint.

- **User stories :**

  Utiliser des user stories pour décrire les fonctionnalités du point de vue de l'utilisateur. Par exemple,

"En tant qu'utilisateur, je veux rechercher des témoignages par pathologie pour trouver des expériences similaires à la mienne".

Les avantages de cette méthodologie :

- Flexibilité et capacité à répondre aux changements rapidement.
- Collaboration étroite entre les membres de l'équipe et les parties prenantes.
- Livraison fréquente de fonctionnalités utilisables.

1.4.3.2 Méthodologie Devops

DevOps combine les pratiques de développement logiciel et d'opérations IT pour automatiser et intégrer les processus entre les équipes de développement et d'exploitation. Dans ce projet, nous avons principalement utilisé Docker et Docker Compose pour gérer l'environnement de développement et de déploiement.

- **Containerisation avec Docker :**
  - **Dockerfiles :** Créer des Dockerfiles pour chaque service de l'application (par exemple, l'api python, le serveur backend) afin de garantir un environnement cohérent et reproductible.
  - **Isolation :** Utiliser Docker pour isoler les différents services, assurant que chaque service fonctionne indépendamment avec toutes ses dépendances.
- **Orchestration avec Docker Compose :**
  - **docker-compose.yml :** Définir un fichier docker-compose.yml pour orchestrer les différents conteneurs. Ce fichier décrit la configuration des services, les réseaux et les volumes nécessaires au fonctionnement de l'application que les développeurs peuvent utiliser pour démarrer rapidement.
  - **Automatisation des Tâches :** Automatiser le démarrage, l'arrêt et la gestion des conteneurs à l'aide de Docker Compose, facilitant ainsi le développement et le déploiement.

Les avantages de cette méthodologie :

- **Cohérence des Environnements :** Utiliser Docker et Docker Compose pour garantir que l'application fonctionne de manière identique en environnement de développement et en production.
- **Configuration Simplifiée :** Simplifier la configuration de l'environnement de développement en fournissant un fichier docker-compose.yml que les développeurs peuvent utiliser pour démarrer rapidement.

1.4.4 Planning du projet

- **Etape 1 :Initialisation et Planification**
  - **Objectifs :**
    * Définir les exigences et les objectifs du projet.
    * Constituer l'équipe de projet.
    * Configurer l'environnement de développement.
  - **Tâches :**
    * Réunion de kick-off.
    * Création de la documentation des exigences et objectifs.
    * Configuration initiale de Docker et Docker Compose.
- **Etape 2 :Recherche et Conception**
  - **Objectifs :**
    * Rechercher et définir les sources de données pour le scraping.
    * Concevoir l'architecture de la plateforme.
  - **Tâches :**
    * Identification des sites web pertinents pour le scraping.
    * Conception de l'architecture de la base de données.
    * Définition de l'architecture globale de l'application (backend, frontend, services de scraping.
- **Etape 3 :Développement de l'Api python**
  - **Objectifs :**
    * Développer et tester les scripts de web scraping.
    * Mettre en place la collecte de données depuis les sources identifiées.
    * Automatiser le processus de scraping en fonction des entrées des utilisateurs.
  - **Tâches :**
    * Écriture des scripts de scraping avec Beautiful Soup et assurer l'automatisation du scraping grâce à Selenium.
    * Intégrer les scripts dans une api Django.
    * Création des endpoints pour l'api Python.
    * Test des scripts sur des sites web cibles, avec des choix de recherches différents.
- **Etape 4 :Développement du Backend**
  - **Objectifs :**

        * Mettre en place l'API backend pour gérer les données.
        * Configurer la base de données à travers docker.
    - **Tâches :**
        * Développement du Backend avec Spring Boot.
        * Configuration de la base de données NoSQL : MongoDB pour notre cas.
        * Création des endpoints pour accéder à l'api Python déjà créee.

- **Etape 5 :Développement du Frontend**
    - **Objectifs :**
        * Développer l'interface utilisateur pour la plateforme.
        * Intégrer le frontend avec le backend.
    - **Tâches :**
        * Conception des maquettes UI/UX.
        * Développement du frontend avec Angular.
        * Intégration des composants frontend avec l'API backend.
        * Test et ajustement de l'interface utilisateur.

- **Etape 6 :Tests et Validation**
    - **Objectifs :**
        * Effectuer des tests complets de l'application.
        * Valider les fonctionnalités et la performance.
    - **Tâches :**
        * Tests des différentes composantes de la plateforme.
        * Corrections des bugs identifiés.

1.5 Objectifs du projet

Ce projet a pour objectifs de :

- Développer des scripts de web scraping pour extraire des témoignages de patients sur divers sites web.
- Créer une base de données structurée pour stocker et organiser ces témoignages.
- Mettre en place une interface utilisateur conviviale permettant de rechercher et de consulter les expériences par sujet à travers l'automatisation du processus de scraping.
- Permettre aux utilisateurs de partager leurs propres expériences sur la plateforme.

1.6 Conclusion

Ce chapitre a permis de mettre en évidence le contexte général du projet en soulignant l'importance du partage d'expériences sur les soins de pathologies et les technologies employées pour ce faire. Nous avons vu que l'utilisation de techniques de web scraping et d'automatisation permet de centraliser et de faciliter l'accès à des témoignages précieux. Cependant, pour assurer le bon fonctionnement de la plateforme, il est essentiel de résoudre les problématiques liées à la collecte, à la structuration et à la mise à jour des données. Nous avons également exposé les objectifs de ce projet, qui visent à développer une ressource utile et accessible pour les patients et les professionnels de la santé.

Chapitre 2

État de l'art

2.1 Introduction

Dans ce chapitre, nous explorons les concepts fondamentaux et les techniques avancées de l'extraction de données, en particulier le web scraping, ainsi que les développements récents dans les modèles de langage génératif comme LLaMA. L'extraction de données est essentielle pour obtenir des informations précieuses à partir de vastes quantités de données disponibles sur le web. Avec l'avènement des technologies avancées et des modèles de langage de grande envergure, comme les modèles de langage large (LLM), il est devenu possible de traiter et d'analyser ces données de manière plus efficace et pertinente. Nous examinerons les différentes méthodes et outils de web scraping, leurs applications dans divers domaines, les défis associés, et les récentes innovations dans l'utilisation de l'intelligence artificielle pour améliorer le scraping. En outre, nous discuterons en détail de LLaMA, un exemple avancé de LLM, et de ses nombreuses applications.

2.2 L'extraction de données

2.2.1 Définition Web Scraping

Le web scraping est le processus d'extraction de données à partir de sites web. Cette technique permet de collecter des informations automatiquement en analysant le contenu des pages web, rendant possible l'acquisition de données en grande quantité et à grande vitesse. Les données collectées peuvent ensuite être utilisées pour diverses applications, telles que l'analyse de marché, la surveillance des prix, la collecte de données pour des projets de recherche, et bien d'autres.

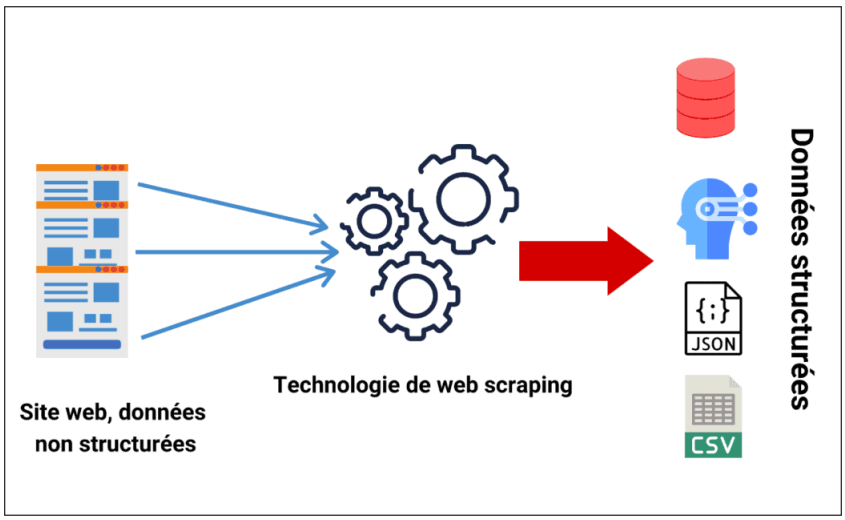


FIGURE 2.1 – Schéma du fonctionnement du web scraping

2.2.2 Fonctionnement du web scraping :

- **Demande-Réponse** : Le processus commence par l'envoi d'une requête à un site web pour récupérer le contenu de la page. Cette requête peut être une simple requête HTTP GET, qui demande au serveur de renvoyer le contenu de la page demandée.
- **Analyser et extraire** : Le contenu récupéré est ensuite analysé pour extraire les données pertinentes. Cette étape implique souvent l'utilisation de techniques d'analyse syntaxique pour décomposer le code HTML et identifier les éléments contenant les informations d'intérêt.
- **Télécharger les données** : Les données extraites sont téléchargées et stockées pour une utilisation ultérieure. Elles peuvent être sauvegardées dans divers formats, tels que des fichiers CSV, des bases de données, ou encore des fichiers JSON, facilitant ainsi leur manipulation et leur analyse.

2.2.3 Différentes techniques du web scraping

2.2.3.1 Accès au site Web :

Les scrapers accèdent aux sites web en envoyant des requêtes HTTP pour obtenir le contenu des pages. Cela peut se faire via des bibliothèques de requêtes HTTP telles que requests en Python, qui permettent de simuler un navigateur web et d'interagir avec les serveurs comme le ferait un utilisateur humain.

2.2.3.2 Analyse HTML et extraction du contenu :

L'analyse du code HTML des pages permet d'extraire les informations souhaitées en identifiant les balises et les structures pertinentes. Les outils tels que Beautiful Soup en Python sont largement utilisés pour cette tâche, car ils offrent des méthodes puissantes pour rechercher et manipuler des éléments HTML.

2.2.3.3 Analyse du modèle d'objet de document (DOM) :

Le DOM représente la structure hiérarchique des documents HTML. Les scrapers utilisent le DOM pour naviguer et extraire les données de manière plus efficace. Les navigateurs automatisés comme Selenium peuvent interagir avec le DOM de manière dynamique, permettant l'extraction de données même sur des sites web qui utilisent JavaScript pour générer du contenu de manière asynchrone.

2.2.3.4 Plateformes d'agrégation verticales :

Certaines plateformes se spécialisent dans l'agrégation de données spécifiques, offrant des solutions de scraping verticales pour des domaines particuliers comme l'immobilier, les offres d'emploi, etc. Ces plateformes fournissent souvent des interfaces prêtes à l'emploi et des API pour faciliter l'accès et l'extraction des données pertinentes.

2.2.4 Logiciels et outils du web scraping

Scrapy est un framework de scraping open-source permettant de construire et exécuter des spiders pour extraire des données de sites web. Scrapy est particulièrement puissant grâce à sa capacité à gérer des requêtes simultanées, ses fonctionnalités de pipeline de données et son extensibilité.

Beautiful Soup est une bibliothèque Python utilisée pour analyser des documents HTML et XML et extraire les données de manière simple et efficace. Elle permet de naviguer dans le parse tree et de rechercher les éléments en utilisant des sélecteurs CSS ou XPath.

Selenium est un outil qui permet d'automatiser les navigateurs web. Il est souvent utilisé pour scraper des sites web dynamiques qui nécessitent l'exécution de JavaScript. Selenium permet de simuler des interactions utilisateur, telles que les clics, les défilements et les soumissions de formulaires, rendant possible l'extraction de données sur des sites web complexes.

ScrapeSimple est une plateforme de scraping facile à utiliser qui offre des fonctionnalités prêtes à l'emploi pour extraire des données de sites web sans besoin de programmation. ScrapeSimple propose des interfaces conviviales et des outils visuels pour configurer et exécuter des tâches de scraping.

2.2.5 Les domaines d'application du Web Scraping

Le web scraping est utilisé dans divers domaines tels que le marketing, la finance, la recherche académique, le journalisme, et plus encore. Dans le marketing, par exemple, il permet de surveiller les prix des concurrents, d'analyser les avis des clients, et de suivre les tendances de l'industrie. En finance, il peut être utilisé pour collecter des données de marché en temps réel, analyser les mouvements des actions et des crypto-monnaies, et surveiller les indicateurs économiques. Les chercheurs académiques utilisent le web scraping pour collecter des données à grande échelle à partir de différentes sources en ligne pour des analyses quantitatives et qualitatives. Les journalistes peuvent utiliser cette technique pour extraire des informations pertinentes pour leurs enquêtes et reportages.

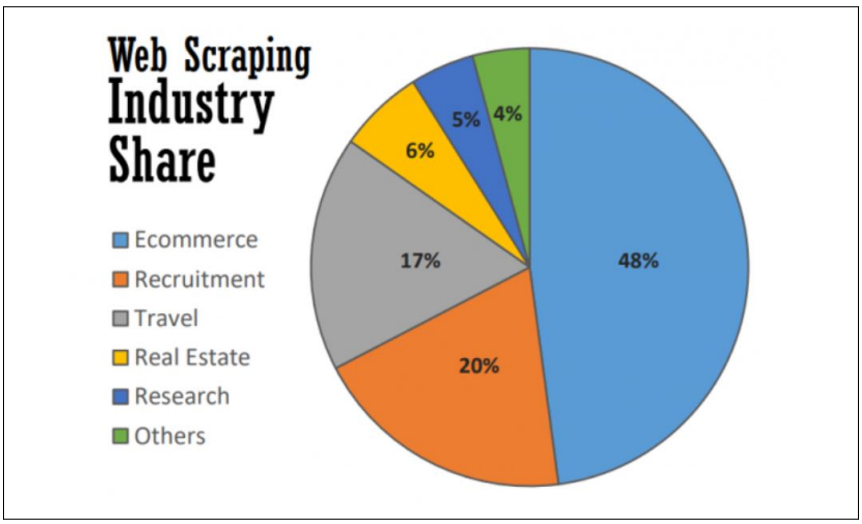


FIGURE 2.2 – Domaines d'application du web scraping

2.2.6 Défis du Web Scraping

2.2.6.1 Modifications fréquentes de la structure :

Les sites web changent souvent leur structure, ce qui peut briser les scrapers et nécessiter des mises à jour fréquentes. Les développeurs doivent donc continuellement surveiller les sites cibles et adapter leurs scripts en conséquence.

2.2.6.2 Pièges HoneyPot :

Certains sites mettent en place des pièges (honeypots) pour identifier et bloquer les scrapers. Ces pièges peuvent inclure des éléments invisibles aux utilisateurs normaux mais détectables par les scrapers, permettant aux administrateurs de sites de détecter et bannir les adresses IP suspectes.

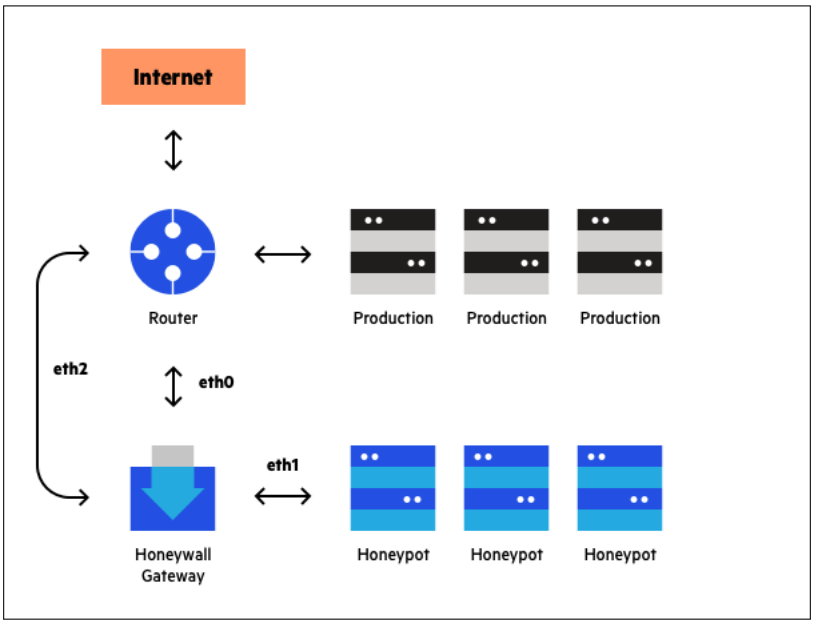


FIGURE 2.3 – Schéma du HoneyPot

2.2.6.3 Technologies anti-scraping :

Des technologies telles que les CAPTCHA, les limitations de taux et les blocs IP sont utilisées pour empêcher le scraping. Les CAPTCHA peuvent être particulièrement difficiles à contourner, nécessitant souvent l'utilisation de services de résolution de CAPTCHA ou de techniques de machine learning pour les déjouer.

2.2.6.4 Qualité des données :

Assurer la qualité des données extraites peut être un défi, surtout lorsque les sources sont diverses et hétérogènes. Les données doivent être nettoyées, normalisées et validées pour garantir leur précision et leur utilité. Cela peut impliquer des processus supplémentaires de traitement des données et l'utilisation de techniques de validation.

2.3 Techniques Avancées de Web Scraping

2.3.1 Scraping Éthique et Légal

2.3.1.1 Considérations éthiques :

Il est important de respecter l'éthique en matière de scraping, comme éviter de surcharger les serveurs web et respecter la vie privée des utilisateurs. Le scraping excessif peut perturber le fonctionnement des sites web et consommer des ressources significatives, ce qui peut être considéré comme une attaque par déni de service. Les scrapers doivent également veiller à ne pas collecter de données sensibles ou personnelles sans consentement.

2.3.1.2 Respect des conditions d'utilisation :

Les scrapers doivent respecter les conditions d'utilisation des sites web pour éviter les litiges juridiques. De nombreux sites web stipulent explicitement dans leurs termes d'utilisation que le scraping est interdit. Ignorer ces conditions peut entraîner des actions légales, y compris des interdictions d'accès, des amendes, et des poursuites judiciaires.

2.3.1.3 Législation autour du web scraping :

La législation varie selon les pays et peut affecter la légalité du web scraping. Par exemple, le Règlement Général sur la Protection des Données (RGPD) en Europe impose des restrictions strictes sur la collecte et l'utilisation des données personnelles. Aux États-Unis, le Computer Fraud and Abuse Act (CFAA) peut être invoqué contre des scrapers non autorisés. Il est crucial de se tenir informé des lois en vigueur et de consulter des experts juridiques pour s'assurer que les activités de scraping sont conformes aux réglementations locales.

2.4 Développements Récents et Tendances

2.4.1 Utilisation de l'IA et du Machine Learning dans le Web Scraping

L'IA et le machine learning sont utilisés pour améliorer la précision des scrapers en reconnaissant des motifs complexes dans les données. Ces technologies permettent aux scrapers de s'adapter automatiquement aux changements dans la structure des pages web et d'extraire les données de manière plus intelligente. Les modèles de machine learning peuvent être entraînés pour identifier les éléments pertinents sur une page web, même si leur apparence change.

2.4.2 Détection et contournement des défenses anti-scraping

Les techniques avancées de machine learning aident à détecter et contourner les mécanismes de défense anti-scraping mis en place par les sites web. Par exemple, des algorithmes peuvent être développés pour résoudre automatiquement les CAPTCHA, imiter les comportements humains pour éviter la détection par les systèmes de surveillance, et gérer les limitations de taux en distribuant les requêtes sur plusieurs adresses IP.

2.4.3 Intégration avec les API

Avantages de l'intégration API :

L'utilisation des API permet d'accéder aux données de manière plus fiable et structurée par rapport au scraping direct des pages web. Les API fournissent des points d'accès bien définis et documentés pour les données, réduisant ainsi la nécessité d'analyser le HTML et de traiter les changements de structure. Les données obtenues

2.5 Modèles de Langage Génératif et LLaMA

Les modèles de langage génératif, également appelés modèles de langage large (LLM), ont révolutionné le traitement automatique du langage naturel (NLP). Ces modèles, tels que LLaMA (Large Language Model Meta AI), sont capables de générer du texte cohérent et pertinent, de répondre à des questions, de traduire des langues, et bien plus encore. Ils sont formés sur de vastes quantités de données textuelles et utilisent des architectures avancées de réseaux de neurones pour comprendre et produire du langage humain. La capacité de ces modèles à comprendre le contexte et à générer du texte de manière fluide ouvre de nouvelles possibilités pour des applications variées dans de nombreux domaines.

2.5.1 Modèle de Langage Large (LLM)

Les modèles de langage large sont des modèles de machine learning conçus pour traiter et générer du texte. Ils sont formés sur des ensembles de données extrêmement volumineux et utilisent des architectures de réseaux neuronaux profondes, comme les Transformers. Ces modèles sont capables de comprendre le contexte et de

produire des réponses textuelles qui imitent le langage humain de manière impressionnante. Les LLMs peuvent être utilisés dans une variété d'applications, allant de la génération de texte à la traduction automatique, en passant par la rédaction assistée et le service à la clientèle automatisé. L'avènement des LLMs a marqué une avancée significative dans le domaine du NLP, rendant possible des interactions homme-machine plus naturelles et efficaces.

2.5.2 Introduction à LLaMA

LLaMA, ou Large Language Model Meta AI, est un exemple avancé de modèle de langage génératif développé par Meta AI. Il se distingue par sa capacité à générer du texte de haute qualité et à comprendre des contextes complexes. LLaMA utilise une architecture Transformer, qui permet une meilleure gestion des relations contextuelles dans les données textuelles. Ce modèle est capable d'effectuer une variété de tâches de NLP, y compris la complétion de texte, la réponse à des questions et la génération de texte créatif. Grâce à ses capacités avancées, LLaMA est un outil puissant pour automatiser et améliorer diverses applications nécessitant une compréhension et une génération de texte.

2.5.3 Architecture de LLaMA

LLaMA est basé sur l'architecture Transformer, une innovation clé dans le domaine des réseaux de neurones pour le NLP. Les Transformers utilisent des mécanismes d'attention pour gérer les relations contextuelles entre les mots de manière plus efficace que les architectures précédentes, comme les réseaux de neurones récurrents (RNN). LLaMA est entraîné sur de grandes quantités de données textuelles provenant de diverses sources, ce qui lui permet de comprendre et de générer du texte dans différents contextes avec une grande précision. Cette architecture avancée permet à LLaMA de surpasser les modèles traditionnels en termes de compréhension du contexte et de fluidité du texte généré.

2.5.4 Applications de LLaMA

Les applications de LLaMA sont vastes et variées. Elles incluent :

- **Génération de Contenu** : Création automatique d'articles, de billets de blog, et de contenu marketing.
- **Service à la Clientèle** : Réponse automatisée aux questions des clients avec un haut degré de pertinence.
- **Traduction Automatique** : Traduction de textes d'une langue à une autre tout en conservant le contexte et le ton.
- **Éducation et Formation** : Fourniture de matériel éducatif personnalisé et assistance à l'apprentissage.

2.6 Benchmark des outils de web scraping

Plusieurs outils sont disponibles pour réaliser web scraping, chacun ayant ses propres caractéristiques et avantages. Dans cette section, nous allons comparer quelques-uns des outils de web scraping les plus populaires : Selenium, Beautiful Soup, Scrapy, et Puppeteer

2.6.1 Axes d'évaluation

Pour effectuer cette comparaison, nous avons retenu les axes d'évaluation suivants :

1. **Facilité d'utilisation** : Simplicité et intuitivité de l'outil pour les développeurs.
2. **Performance** : Rapidité et efficacité de l'outil pour extraire des données.
3. **Flexibilité** : Capacité de l'outil à s'adapter à différents types de sites web et structures de données.
4. **Support des technologies web modernes** : Aptitude à gérer JavaScript, AJAX, et autres technologies web modernes.
5. **Communauté et support** : Taille et activité de la communauté, disponibilité de la documentation et des ressources de support.
6. **Écosystème et extensibilité** : Disponibilité de bibliothèques supplémentaires, plugins, et intégration avec d'autres outils.

2.6.2 Tableau comparatif

Critère	**Selenium**	**Beautiful Soup**	**Scrapy**	**Puppeteer**
Facilité d'utilisation	Modérée (nécessite configuration)	Facile (API simple)	Modérée (nécessite configuration)	Modérée (nécessite des connaissances en JS)
Performance	Moyenne (plus lent)	Élevée (très rapide)	Élevée (très rapide)	Moyenne (moins rapide)
Flexibilité	Très flexible	Moyenne (nécessite complément)	Très flexible	Très flexible
Support des technologies web modernes	Excellent (gère JavaScript)	Limité (ne gère pas JavaScript)	Limité (nécessite complément)	Excellent (gère JavaScript)

Critère	Selenium	Beautiful Soup	Scrapy	Puppeteer
Communauté et support	Grande communauté, bon support	Grande communauté, bon support	Grande communauté, bon support	Grande communauté, bon support
Écosystème et extensibilité	Large écosystème (nombreux plugins)	Limité	Large écosystème (nombreux plugins)	En développement

2.7 La méthodologie de scraping utilisée

La méthodologie de scraping que nous avons adoptée pour notre projet comprend plusieurs étapes clés, chacune étant essentielle pour garantir une extraction de données efficace et fiable.

2.7.1 Définir les objectifs

La première étape de notre méthodologie consiste à définir clairement les objectifs du scraping. Cela implique l'identification des données spécifiques que nous souhaitons extraire. Dans le cadre de notre projet, nous nous concentrons sur deux types de données principales : les listes de threads contenant les mots-clés mentionnés par l'utilisateur et les threads décrivant des expériences pathologiques.

2.7.2 Les outils choisis

En utilisant le benchmark des outils de web scraping, nous avons sélectionné les outils les plus appropriés pour notre projet :

Selenium : Utilisé pour naviguer d'une page web à l'autre, interagir avec des pages dynamiques et gérer JavaScript. **Beautiful Soup :** Utilisé pour ses performances élevées dans l'extraction des données HTML statiques.

2.7.3 Analyse du site web

Cette étape consiste à analyser en profondeur la structure du site web ciblé. Nous avons examiné le HTML, JavaScript et les requêtes AJAX pour comprendre comment les données sont chargées et présentées. De plus, nous avons vérifié la présence de mécanismes d'anti-scraping, tels que les captchas, les vérifications de bot, et les limitations de taux d'accès.

2.7.4 Développement du scraper

Le pipeline de scraping de notre application se déroule en deux étapes principales, permettant aux utilisateurs de rechercher et de sélectionner des threads de blogs contenant des expériences pathologiques. Voici une

description détaillée de ces deux étapes :

2.7.4.1 Étape 1 : Recherche de Threads

Lorsque l'utilisateur entre un mot-clé de recherche, une procédure est exécutée pour retourner une liste de threads potentiellement intéressants, sans importer réellement les données. Cette étape permet de filtrer les résultats et de présenter à l'utilisateur une liste de choix pertinents basés sur son mot-clé. Le script analyse les pages web correspondantes, extrait les titres et les descriptions des threads, et les affiche à l'utilisateur.

2.7.4.2 Étape 2 : Sélection et Extraction du Thread

Une fois que l'utilisateur a choisi le thread qui l'intéresse, une seconde procédure est exécutée pour retourner le contenu détaillé de ce thread. Cette étape implique le scraping des données du thread sélectionné, incluant le contenu complet du blog.

2.7.4.3 Étape 3 : Structuration des Données

Le texte brut des expériences pathologiques extraites est ensuite structuré sous forme d'objets en utilisant Ollama, un modèle de langage large (LLM). Ollama permet de convertir efficacement le texte brut en objets structurés, facilitant ainsi l'analyse et l'utilisation ultérieure des données extraites.

2.7.5 Gestion des limitations

Pour éviter d'être bloqués par le site web, on peut mettre en place plusieurs stratégies :

Implémentation de pauses : on a introduit des délais aléatoires entre les requêtes pour imiter un comportement humain.

Gestion des captchas : Si nécessaire, on intégre des solutions pour résoudre ou contourner les captchas.

Détection et évitement des honey pots : Certains sites insèrent des liens ou des champs invisibles destinés à piéger les scrapers. on a mis en place des techniques pour détecter et ignorer ces honey pots.

Rotation des adresses IP : Pour éviter d'être bloqués en raison d'un grand nombre de requêtes provenant de la même adresse IP, on utilise des proxies pour faire tourner les adresses IP.

Utilisation d'agents utilisateurs variés : on change régulièrement les agents utilisateurs (user agents) pour empêcher la détection basée sur les caractéristiques du navigateur.

2.7.6 Extraction et stockage des données

Les données extraites sont stockées dans une base de données NoSQL. Cette approche nous permet d'avoir des documents imbriqués, facilitant ainsi la représentation des relations entre les différentes entités définies dans notre projet. Le choix d'une base de données NoSQL est motivé par sa flexibilité et sa capacité à gérer des structures de données semi-structurées ou non structurées.

2.8 Conclusion

Ce chapitre a fourni une vue d'ensemble approfondie des techniques et outils utilisés dans le web scraping, ainsi que des défis et des solutions pour contourner les mécanismes anti-scraping. Nous avons également exploré l'importance et les applications des modèles de langage génératif, en mettant l'accent sur LLaMA, un modèle de langage de pointe développé par Meta AI. La combinaison de ces technologies permet d'extraire et de traiter des données web de manière plus sophistiquée et précise, ouvrant de nouvelles perspectives pour l'analyse de données, l'automatisation et l'intelligence artificielle. En comprenant les principes de base et les techniques avancées du web scraping, ainsi que l'impact des LLMs, les chercheurs et les professionnels peuvent mieux exploiter ces outils pour répondre à des besoins complexes en matière de données et de traitement du langage naturel.

Chapitre 3

Analyse et conception

3.1 Introduction

Le chapitre sur l'analyse et la conception de notre plateforme de partage d'expériences sur le soin des pathologies se concentre sur l'examen approfondi des besoins fonctionnels et non fonctionnels du projet. Cette phase essentielle permet de définir les fonctionnalités clés de la plateforme ainsi que les critères de qualité et les contraintes qui orienteront sa conception et son développement.

3.2 Analyse fonctionnelle

Dans cette section, nous allons analyser les aspects fonctionnels de notre projet de plateforme de partage d'expériences sur le soin de pathologies. Cette analyse est structurée en trois parties : les besoins techniques, les exigences fonctionnelles et les exigences non fonctionnelles.

3.2.1 Exigences fonctionnelles

- **Recherche et navigation :**
  Fonctionnalité de recherche par mot-clé, pathologie, et autres critères pertinents. Navigation claire et intuitive à travers les expériences et les catégories.

- **Affichage des expériences :**
  Présentation des expériences sous forme de blogs avec les détails pertinents (date, source, résumé), et option d'affichage par pertinence ou par nombre de upvotes.

- **Soumission des expériences :**
  Interface utilisateur pour permettre aux utilisateurs de soumettre leurs propres expériences, un formulaire de soumission dédié avec des champs requis pour garantir la cohérence et la qualité des données, et assurer que les blogs tous suivent le même schèma.

- **Recommendation des expériences :**

  L'utilisateur reçoit une liste d'expériences recommandées potentielles et en sélectionne une.

3.2.2 Exigences non fonctionnelles

Les exigences non fonctionnelles définissent les critères de qualité et les contraintes qui ne sont pas directement liés aux fonctionnalités spécifiques de la plateforme, mais qui sont tout aussi importants pour son succès. Voici quelques exemples d'exigences non fonctionnelles pour notre projet :

- **Performances :**

  Cette exigence définit les temps de réponse attendus de la plateforme, le nombre maximal d'utilisateurs simultanés qu'elle doit prendre en charge, et la capacité à gérer des charges de travail élevées sans compromettre les performances.

- **Scalabilité :**

  Architecture évolutive pour ajouter de nouvelles sources de données pour le scraping des expériences et gérer un nombre croissant d'utilisateurs. Facilité de mise à jour et d'ajout de nouvelles fonctionnalités sans interruption du fonctionnement.

- **Fiabilité :**

  Haute disponibilité du système avec une infrastructure robuste et des sauvegardes régulières. Système de récupération en cas de défaillance pour minimiser les temps d'arrêt, cette garantie repose sur la fiabilité de notre base de données, offrant la possibilité de générer des duplicatas pour assurer une résilience aux pannes.

- **Expérience utilisateur :**

  Interface utilisateur intuitive et facile à utiliser. Conception responsive pour assurer une utilisation optimale sur différents appareils (ordinateurs, tablettes, smartphones).

- **Maintenance :**

  Des exigences de documentation définissent les attentes en matière de documentation du système, y compris la documentation technique, les guides d'utilisation et les manuels d'administration, pour faciliter les modifications et les mises à jour futures.

3.3 Diagramme de classe

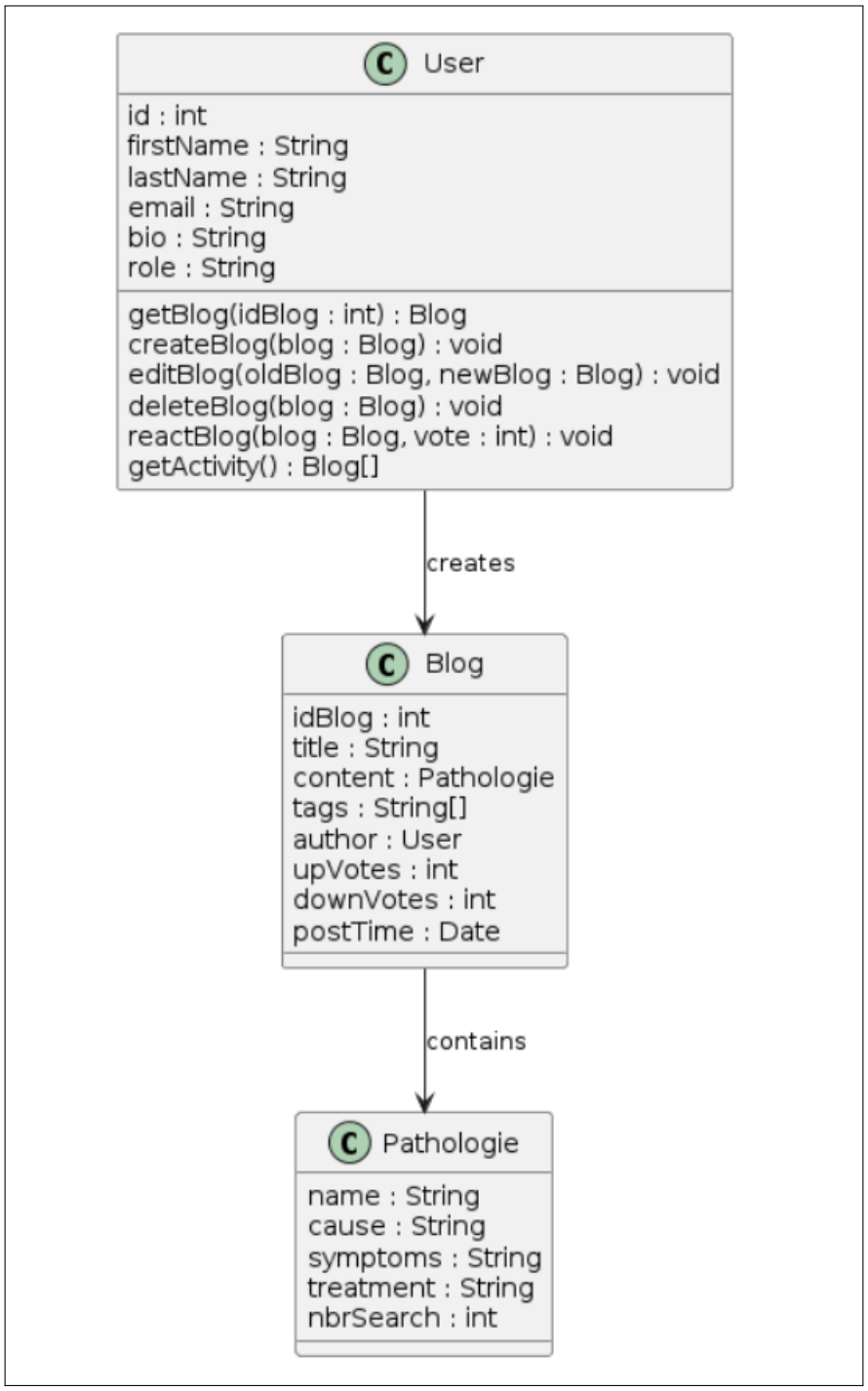


FIGURE 3.1 – Diagramme de classe

3.4 Diagramme de cas d'utilisation

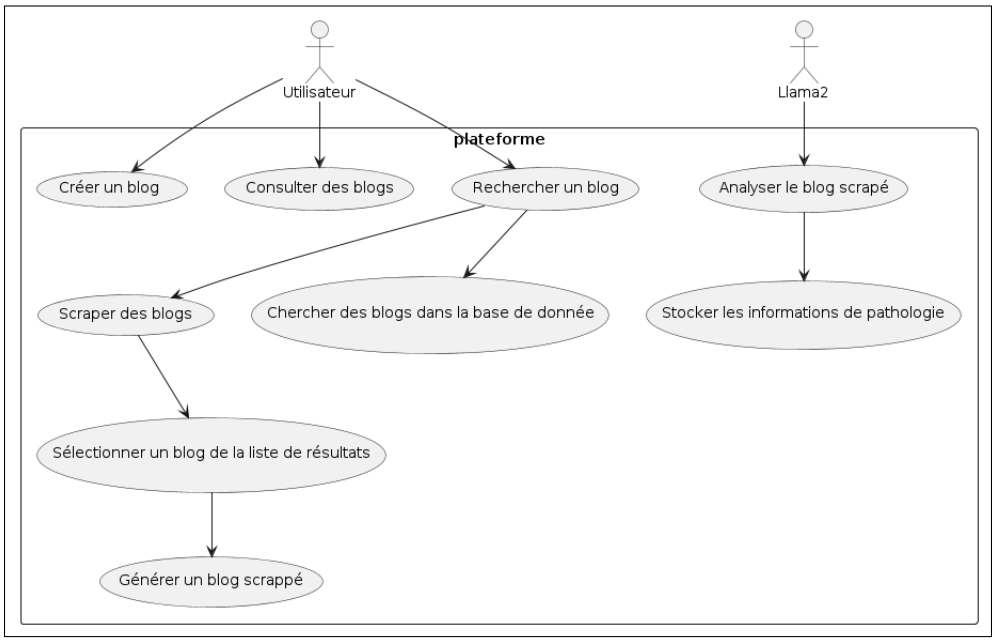


FIGURE 3.2 – Diagramme de cas d'utilisation

3.5 Diagramme de sequence

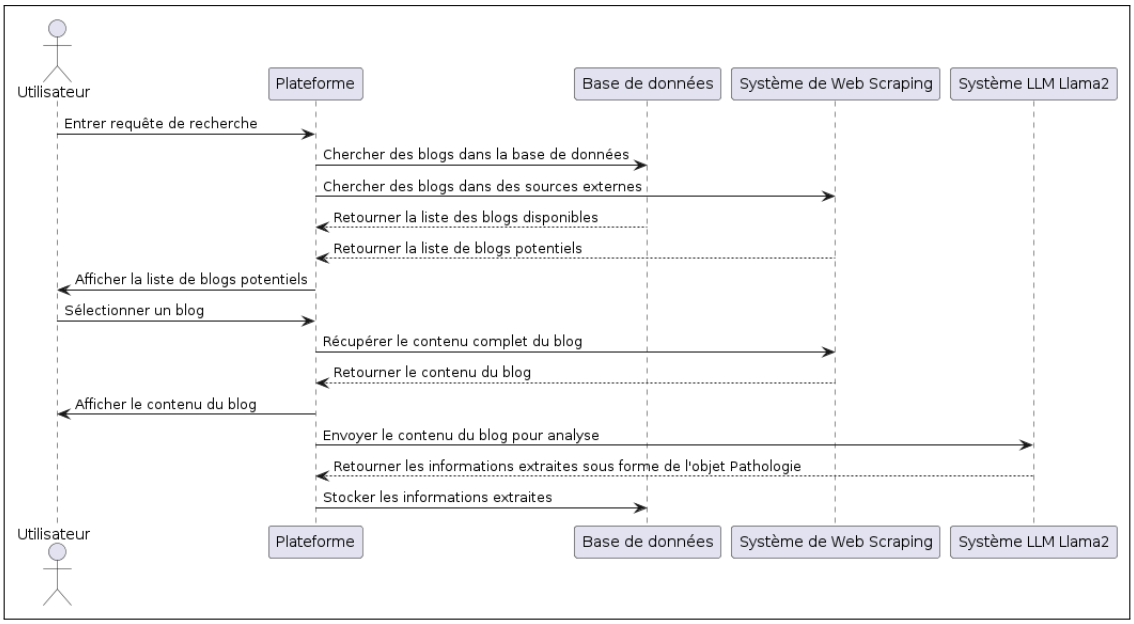


FIGURE 3.3 – Diagramme de sequence

3.6 Les maquettes de l'application

Les maquettes des interfaces graphiques de notre application web offrent un aperçu visuel de l'expérience utilisateur, présentant de manière intuitive la disposition, la navigation et les fonctionnalités clés de notre application.

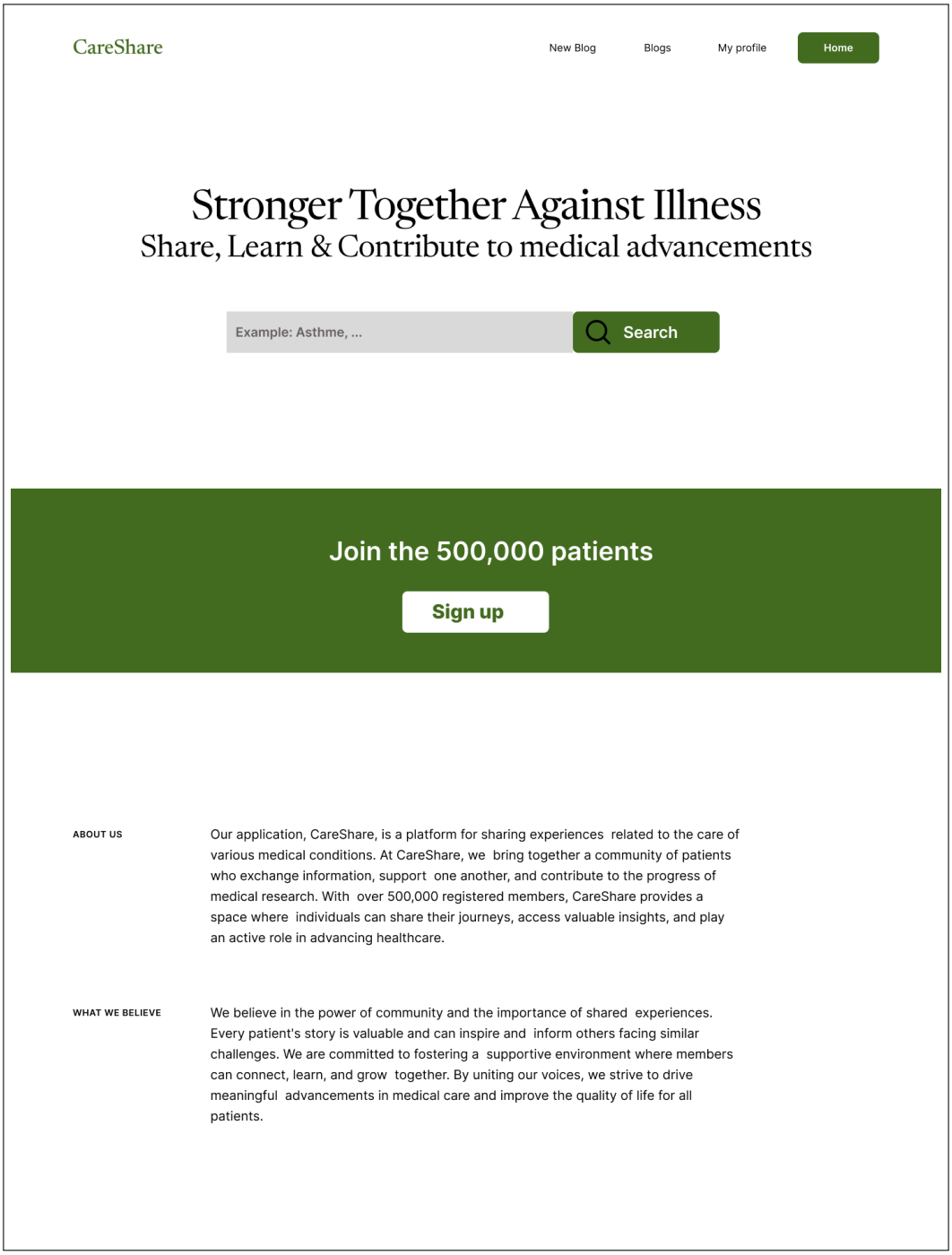


FIGURE 3.4 – GUI : Page d'acceuil

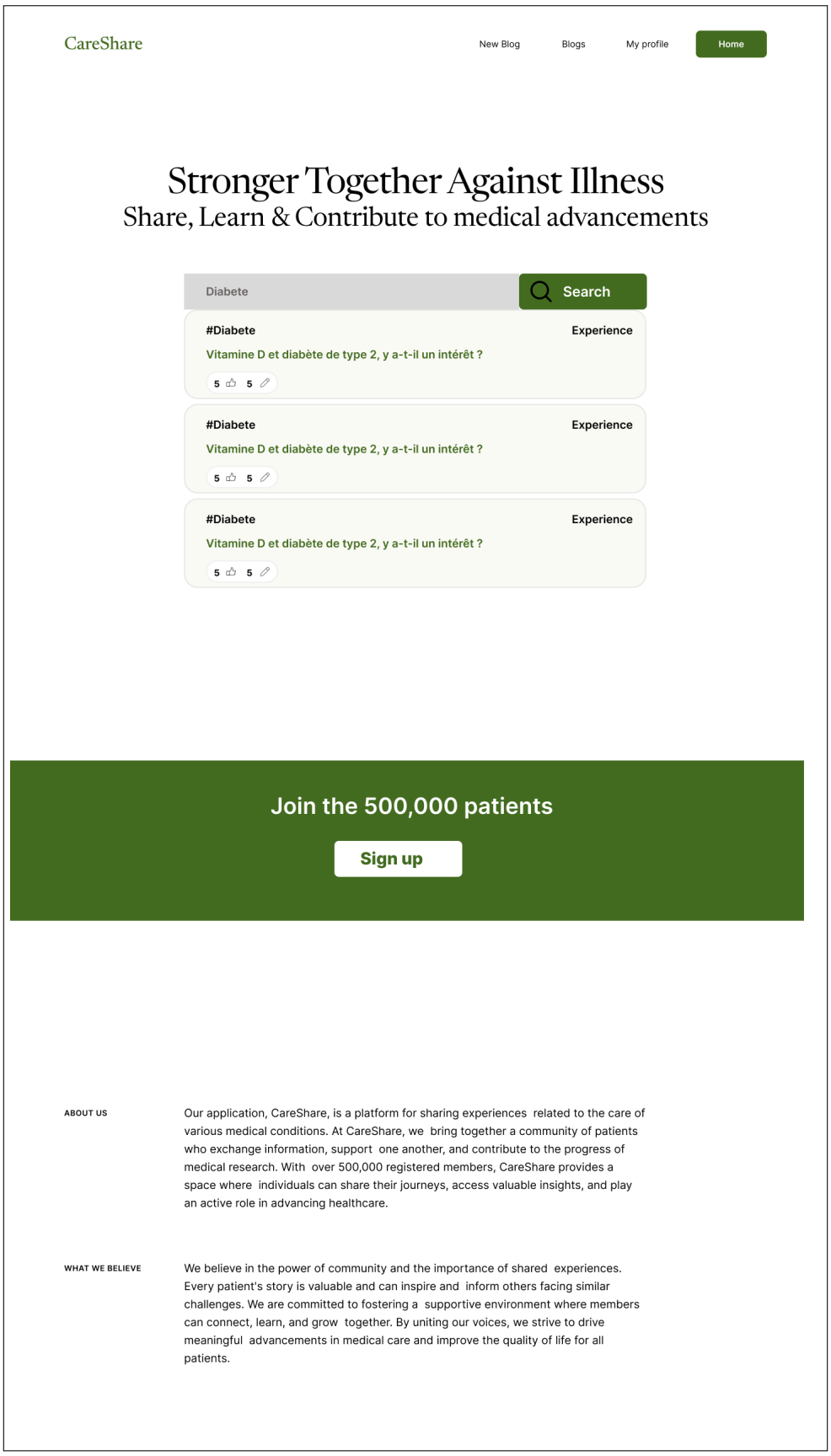


FIGURE 3.5 – GUI : Search bar

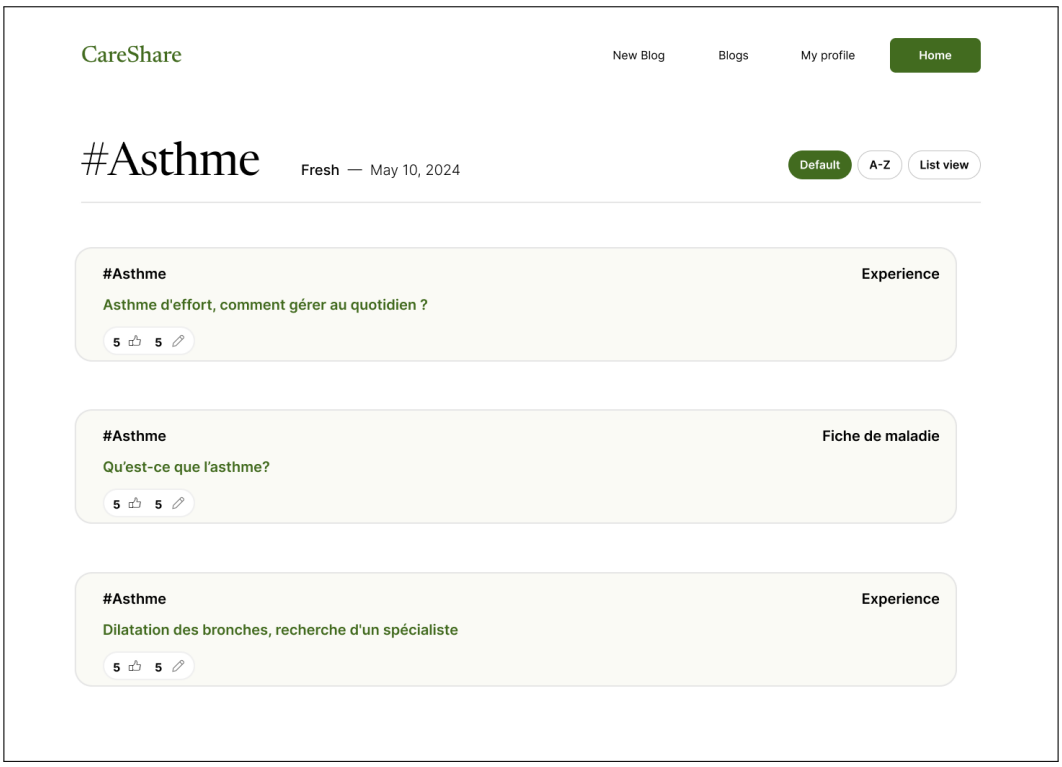


FIGURE 3.6 – GUI : Blogs

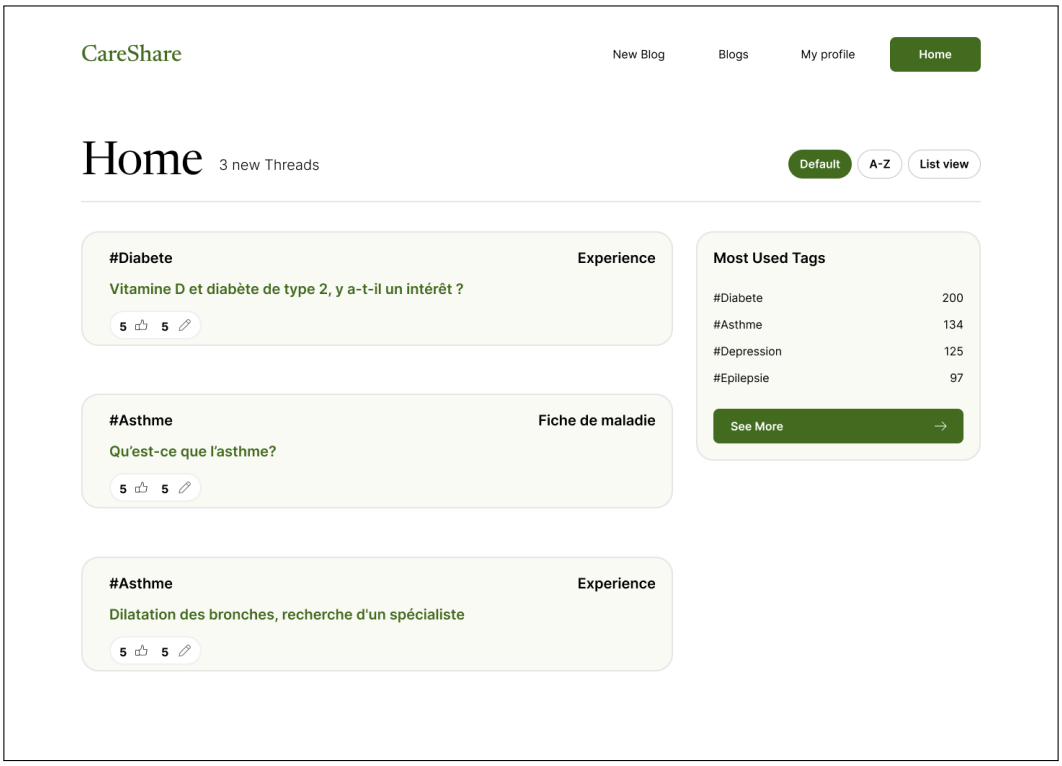


FIGURE 3.7 – GUI : Tags

3.7 Conclusion

En conclusion, cette phase d'analyse et de conception a permis de jeter les bases solides nécessaires à la réalisation de notre plateforme. En identifiant clairement les besoins fonctionnels tels que la recherche et la navigation intuitives, ainsi que les exigences non fonctionnelles telles que la performance, la fiabilité et l'expérience utilisateur, nous sommes en mesure de concevoir une solution qui répondra efficacement aux attentes des utilisateurs tout en assurant une expérience utilisateur optimale. Les diagrammes de classes, de cas d'utilisation et de séquence, ainsi que les maquettes d'interface graphique, offrent une vision claire et structurée du système à développer, permettant ainsi de passer à la prochaine phase de développement avec confiance et clarté.

Chapitre 4

Réalisation

4.1 Introduction

Ce chapitre présente les outils et technologies clés utilisés dans le développement de notre application, mettant en lumière les choix technologiques et les processus mis en œuvre pour atteindre nos objectifs. En plus de cela, nous avons également inclus une présentation des interfaces graphiques réalisées, offrant ainsi un aperçu visuel de l'expérience utilisateur que nous visons à offrir.

4.2 Architecture de l'application

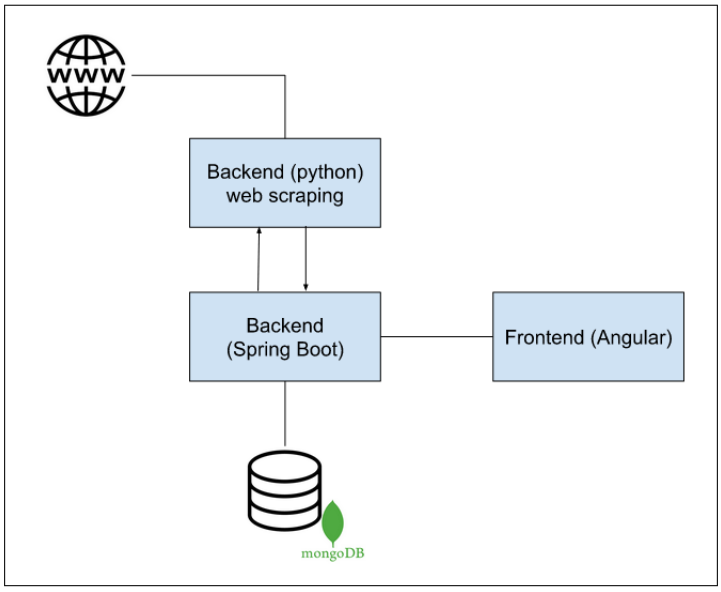


Figure 4.1 – Architecture de l'application

4.3 Outils utilisés

4.3.1 Langages de Programmation et Frameworks

4.3.1.1 Frontend

Angular est un framework de développement frontend open-source maintenu par Google. Il permet de construire des applications web dynamiques et réactives en utilisant TypeScript. Angular offre une structure solide pour les projets, favorisant une séparation claire des préoccupations grâce à son architecture basée sur des composants. Les développeurs peuvent tirer parti des nombreuses fonctionnalités intégrées, telles que la gestion des formulaires, les services HTTP, et les outils de test, pour créer des interfaces utilisateur performantes et maintenables. L'utilisation de Angular dans notre projet nous a permis de développer rapidement des fonctionnalités complexes tout en assurant une expérience utilisateur fluide.

4.3.1.2 Back-End

Spring Boot est un framework Java conçu pour simplifier le développement d'applications backend robustes et évolutives. En fournissant un ensemble de conventions et de configurations par défaut, Spring Boot permet de démarrer rapidement un projet sans avoir à se soucier des configurations complexes. Il intègre facilement divers modules de l'écosystème Spring, comme Spring Data, Spring Security et Spring MVC, facilitant ainsi la mise en place de fonctionnalités comme la gestion des bases de données, la sécurité et les API RESTful. Dans notre projet, Spring Boot a été utilisé pour gérer la logique métier, la persistance des données et les interactions avec le frontend.

python

Python est un langage de programmation polyvalent et très populaire, connu pour sa simplicité et sa lisibilité, ce qui en fait un excellent choix pour les débutants comme pour les développeurs expérimentés. Créé par Guido van Rossum et publié pour la première fois en 1991, Python prend en charge des paradigmes de programmation multiples, y compris la programmation orientée objet, procédurale et fonctionnelle. Il est largement utilisé dans divers domaines tels que le développement web, la science des données, l'intelligence artificielle, l'automatisation, et bien d'autres. Sa vaste bibliothèque standard et son écosystème riche en frameworks et modules facilitent le développement rapide et efficace d'applications complexes.

Outils de Web Scraping

Les outils de web scraping sont essentiels pour extraire automatiquement des données de sites web. Ces outils peuvent être des bibliothèques ou des frameworks spécialement conçus pour naviguer sur des pages web, analyser le contenu HTML et extraire les informations pertinentes. Dans notre projet, nous avons utilisé des bibliothèques comme BeautifulSoup en Python pour effectuer le scraping de manière efficace. Ces outils nous ont permis de collecter des données à partir de diverses sources en ligne, ce qui a été crucial pour alimenter notre base de données avec des informations à jour et pertinentes.

- **Selenium** est un framework de test utilisé pour l'automatisation des navigateurs web. Il permet aux développeurs de simuler des interactions utilisateur comme les clics, la saisie de texte et la navigation entre les pages web, ce qui est particulièrement utile pour tester des applications web complexes et dynamiques. Selenium est compatible avec plusieurs navigateurs et langages de programmation, offrant une grande flexibilité pour divers projets. Dans notre projet, Selenium a été utilisé pour automatiser les tâches de navigation sur les sites web et interagir avec des éléments dynamiques avant de procéder à l'extraction des données.

- **BeautifulSoup (bs4)** est une bibliothèque Python spécialisée dans l'analyse et l'extraction de données à partir de fichiers HTML et XML. Elle fournit des méthodes simples et efficaces pour parcourir, rechercher et modifier le contenu des documents, ce qui facilite le web scraping. BeautifulSoup est capable de gérer des documents mal formés et offre une interface intuitive pour accéder aux éléments de la page. Dans notre projet, BeautifulSoup a été utilisé pour extraire les données pertinentes après que Selenium ait navigué et interagi avec les pages web, permettant une collecte de données structurée et précise.

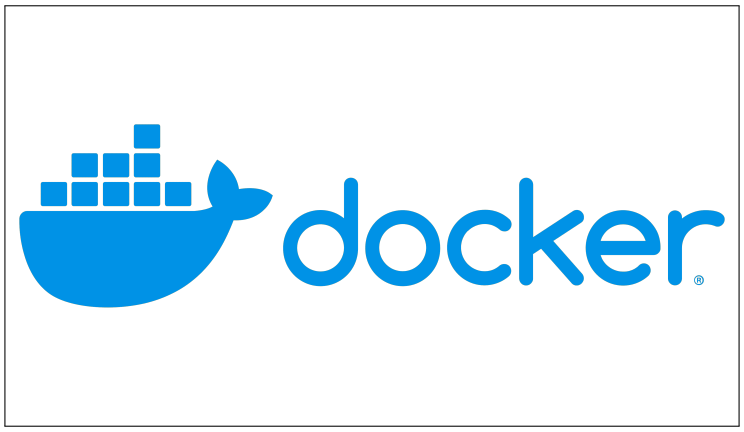
LLAMA2, acronyme de "Large Language Model Meta AI 2," est une version avancée d'un modèle de traitement du langage naturel développé par Meta (anciennement Facebook). Conçu pour comprendre et générer du texte de manière plus sophistiquée, LLAMA2 utilise des techniques de pointe en apprentissage profond et en intelligence artificielle. Avec une architecture optimisée pour traiter des quantités massives de données textuelles, il est capable d'effectuer diverses tâches telles que la traduction automatique, la génération de texte, et l'analyse de sentiments. LLAMA2 se distingue par sa capacité à produire des réponses contextuellement pertinentes et cohérentes, ce qui le rend particulièrement utile pour les applications nécessitant une compréhension fine du langage humain. De plus, il est conçu pour être plus efficace en termes de ressources computationnelles, facilitant ainsi son déploiement sur une large gamme de dispositifs et d'environnements.

4.3.2 Bases de Données

MongoDB est une base de données NoSQL orientée documents qui offre une grande flexibilité et évolutivité. Contrairement aux bases de données relationnelles traditionnelles, MongoDB stocke les données sous forme de documents JSON, permettant une structure de données plus dynamique et plus naturelle à manipuler pour les développeurs. Cette flexibilité facilite l'adaptation aux changements de schéma et aux nouvelles exigences sans nécessiter de migrations complexes. Dans notre projet, MongoDB a été choisi pour sa capacité à gérer de grandes quantités de données non structurées, offrant ainsi une performance optimale pour les opérations de lecture et d'écriture.

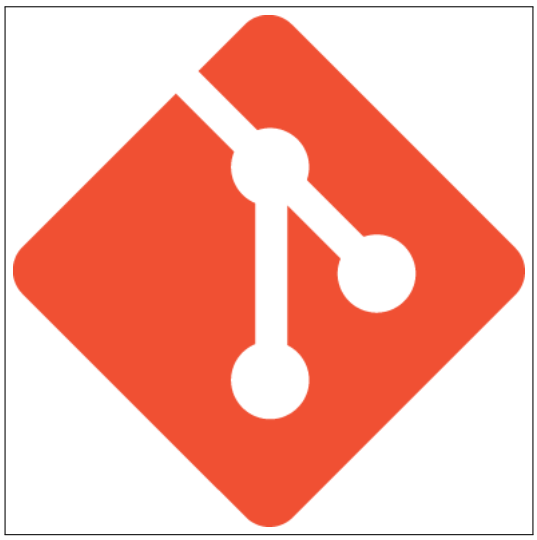
4.3.3 Outils DevOps

Docker est une plateforme de conteneurisation qui permet de créer, déployer et gérer des applications dans des environnements isolés appelés conteneurs. Chaque conteneur encapsule une application et toutes ses dépendances, assurant une portabilité et une cohérence indépendamment de l'environnement d'exécution. Docker simplifie le processus de déploiement en garantissant que les applications se comportent de la même manière en développement, en test et en production. Dans notre projet, Docker a été utilisé pour containeriser les différents services, facilitant ainsi leur déploiement et leur gestion sur diverses infrastructures cloud et on-premises.



4.3.4 Système de contrôle de version

Git est un système de contrôle de version distribué, largement utilisé pour la gestion du code source dans les projets de développement logiciel. Il permet aux développeurs de suivre les modifications apportées aux fichiers, de travailler simultanément sur différentes branches du projet et de fusionner les modifications de manière efficace. Git facilite également la collaboration entre les membres de l'équipe, en assurant que chacun puisse accéder à la version la plus récente du code et contribuer sans conflits majeurs. Les fonctionnalités de Git, telles que les commits, les branches et les merges, aident à maintenir un historique clair et détaillé des changements, ce qui est essentiel pour la gestion de projets complexes. Dans notre projet, Git a été utilisé pour gérer le code source, coordonner les contributions des développeurs et garantir un workflow de développement fluide et organisé.



4.3.5 Outils de Test

Postman est un outil populaire pour le développement, le test et la documentation des API. Il permet aux développeurs de créer et d'exécuter des requêtes HTTP de manière intuitive, de vérifier les réponses des serveurs, et de s'assurer que les API fonctionnent comme prévu. Postman offre également des fonctionnalités de création de scripts pour automatiser les tests, ainsi que des collections pour organiser les différentes requêtes. Dans notre projet, Postman a été utilisé pour tester les endpoints de l'API backend, garantissant ainsi que chaque fonctionnalité soit correctement implémentée et fonctionne sans erreur.

4.3.6 Environnement de Développement Intégré (IDE)

IntelliJ IDEA est un IDE robuste et intelligent pour le développement Java et d'autres langages. Il offre une multitude de fonctionnalités avancées comme la complétion de code intelligente, la refactorisation, et les outils de débogage intégrés, ce qui améliore considérablement la productivité des développeurs. IntelliJ prend en charge de nombreux frameworks et technologies, rendant la gestion de projets complexes plus simple et plus efficace. Dans notre projet, IntelliJ a été utilisé principalement pour le développement backend avec Spring Boot, offrant un environnement de développement riche et interactif.

Visual Studio Code (VSCode) est un éditeur de code source open-source développé par Microsoft. Il est léger, extensible et supporte une vaste gamme de langages de programmation et de frameworks grâce à ses nombreuses extensions. VSCode offre des fonctionnalités puissantes comme la complétion de code, le débogage intégré, et la gestion de contrôle de version, ce qui en fait un outil de choix pour le développement frontend. Dans notre projet, VSCode a été utilisé pour le développement Angular, bénéficiant de ses extensions spécifiques au framework et de son intégration avec des outils de gestion de projet et de collaboration.

4.3.7 Outil de conception et de prototypage

Figma est un outil de conception et de prototypage collaboratif basé sur le web, utilisé pour créer des interfaces utilisateur interactives. Il permet aux équipes de travailler ensemble en temps réel, facilitant ainsi la collaboration et la révision des designs. Figma offre une multitude de fonctionnalités comme les composants réutilisables, les styles partagés et les prototypes interactifs, ce qui accélère le processus de conception et garantit la cohérence visuelle. Dans notre projet, Figma a été utilisé pour concevoir et prototyper l'interface utilisateur, permettant de visualiser et de tester les interactions avant la phase de développement.

PlantUML est un outil open-source permettant de créer des diagrammes UML à partir d'un simple texte descriptif. Il supporte une variété de diagrammes, y compris les diagrammes de séquence, de classe, d'activité et de cas d'utilisation. PlantUML est particulièrement utile pour la documentation technique et la modélisation des systèmes, offrant une méthode rapide et efficace pour générer des diagrammes à partir de spécifications textuelles. Dans notre projet, PlantUML a été utilisé pour documenter l'architecture du système et les flux de données, assurant une compréhension claire et partagée de la structure et du fonctionnement de l'application.

4.4 Mise en oeuvre

4.4.1 Automatisation et Web Scraping

Pour collecter les expériences pathologiques partagées par des gens sur le web, nous avons mis en place un processus automatisé de web scraping. Ce processus utilise des outils comme Selenium et BeautifulSoup pour naviguer sur les sites web, interagir avec les pages dynamiques, et extraire les données pertinentes sous forme de blogs. Selenium nous permet d'automatiser les interactions utilisateur, telles que les clics et la saisie de texte, nécessaires pour accéder aux contenus dynamiques. Une fois les pages web chargées, BeautifulSoup est utilisé pour analyser le contenu HTML et extraire les informations pertinentes de manière structurée. Ce processus de scraping est exécuté régulièrement pour s'assurer que notre base de données reste à jour avec les nouvelles expériences partagées en ligne.

4.4.2 Les sites web ciblés

Pour notre projet de scraping, nous avons principalement ciblé le site web **Carenity** (`https://www.carenity.com/forum/index-forums`). Carenity est une plateforme de réseau social dédiée aux patients, offrant des forums de discussion où les utilisateurs peuvent partager leurs expériences, poser des questions et interagir sur divers sujets liés à la santé. Les données extraites de ce site sont précieuses pour notre analyse des expériences pathologiques des patients. Afin de permettre une extensibilité future et d'ajouter facilement d'autres sites web à notre projet, nous avons utilisé le design pattern Strategy.
Ce design pattern nous permet d'ajouter chaque nouveau site web en tant que classe concrète sans modifier le code principal. Ainsi, notre application sera flexible et évolutive, facilitant l'intégration de nouvelles sources de données.

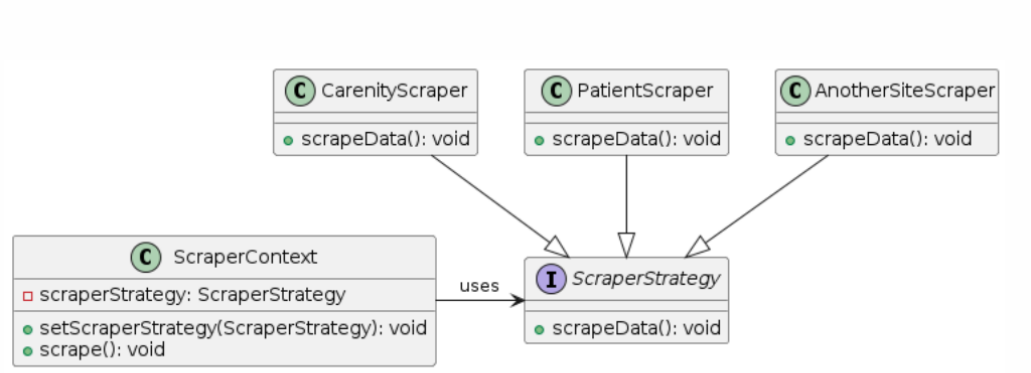


FIGURE 4.2 – Diagramme de classe du design pattern Strategy

4.4.3 Architecture du web scraping pipeline

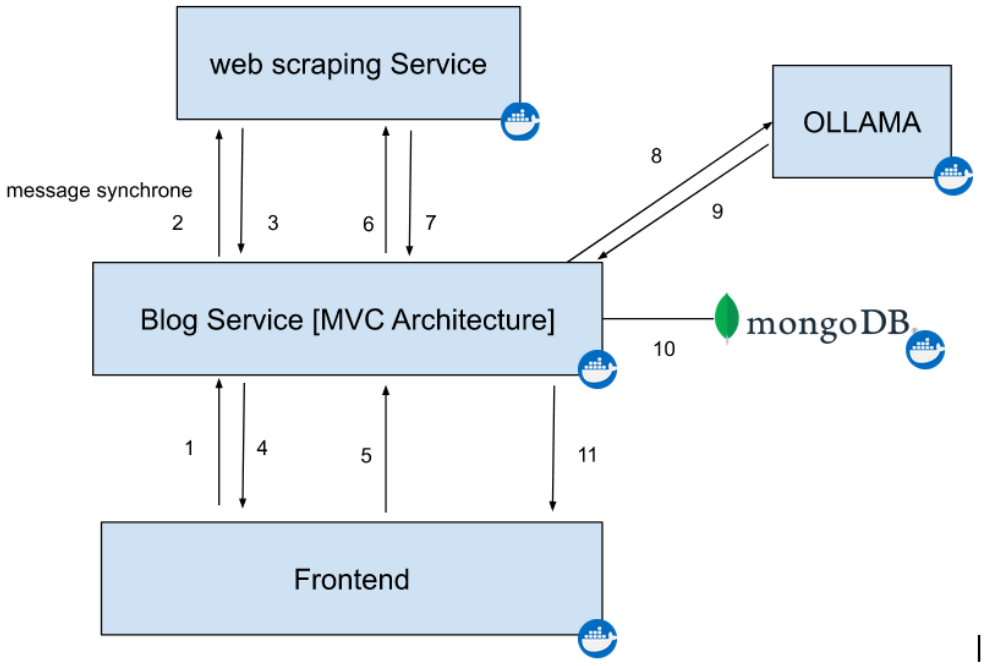


FIGURE 4.3 – Web scraping Pipeline

1. L'utilisateur entre le mot-clé de recherche.
2. Le service Blog envoie des requêtes synchrones contenant le mot-clé au service de Web Scraping via une API RESTful.
3. Le service de Web Scraping extrait la liste des threads contenant ce mot-clé.
4. Cette liste de threads est affichée à l'utilisateur.
5. L'utilisateur choisit le thread qui l'intéresse.
6. Le backend envoie l'identifiant de ce thread.
7. Le serveur de Web Scraping extrait ce thread sous forme de texte brut et le renvoie au service Blog.
8. Le service Blog, en utilisant une API RESTful, envoie ce texte au conteneur LLAMA pour utiliser le modèle LLAMA2 afin de structurer le texte sous forme d'entités liées à la pathologie définie dans le service.
9. Retourner la structure du thread.
10. Stocker le thread dans une base de données NoSQL.
11. Afficher ce thread sélectionné à l'utilisateur.

4.4.4 Pipeline de Scraping

Le pipeline de scraping de notre application se déroule en deux étapes principales, permettant aux utilisateurs de rechercher et de sélectionner des threads de blogs contenant des expériences pathologiques. Voici une description détaillée des deux étapes du processus :

4.4.4.1 Étape 1 : Recherche de Threads

Lorsque l'utilisateur entre un mot-clé de recherche, le fichier `scrapinglistthreads` est exécuté pour retourner une liste de threads potentiellement intéressants, sans importer réellement les données. Cette étape permet de filtrer les résultats et de présenter à l'utilisateur une liste de choix pertinents basés sur son mot-clé. Le script analyse les pages web correspondantes, extrait les titres et les descriptions des threads, et les affiche à l'utilisateur.

Pseudo-code pour `scrapinglistthreads` :

Algorithm 1 Web Scraping Algorithm for Getting List Pathologie

Initialize keyword, count = 1
Capitalize keyword
Initialize WebDriver with options
Determine first letter of keyword, lettre
for page_nb **from** 1 **to** count **do**
  Construct page_url using lettre and page_nb
  Navigate to page_url
  Handle cookies : Click "agree" button
  Wait for 12 seconds to avoid being blocked
  Search for links related to keyword
  if links are found **then**
    for each link in links **do**
      Print and store the link URL
      Click the link
      Break the loop
    end for
  end if
end for
Extract discussion threads on the new page
Initialize DataFrame result_interest with columns 'title' and 'link'
Parse the URL of the keyword-specific page to extract relevant information
Add the disease information link to the DataFrame
for each interest in p_interests **do**
  Extract thread_href and thread_title
  Add thread_title and thread_href to the DataFrame
end for
return DataFrame result_interest as a dictionary
=0

4.4.4.2 Étape 2 : Sélection et Extraction du Thread

Une fois que l'utilisateur a choisi le thread qui l'intéresse, le fichier `SelectedThread` est exécuté pour retourner le contenu détaillé de ce thread. Cette étape implique le scraping des données spécifiques du thread sélectionné, incluant le contenu complet du blog, les commentaires et autres informations pertinentes.

Pseudo-code pour `SelectedThread` :

Algorithm 2 Function `extract_text_without_tags`

Require: `html_content` : HTML content to be processed
1: Parse the HTML content using BeautifulSoup
2: Get the text content without HTML tags
3: **return** Stripped text =0

Algorithm 3 Function `get_selected_thread`

Require: `topic`, `th_url`
1: Configure the Chrome WebDriver
2: Navigate to the specified thread URL
3: Handle cookies by clicking the "agree" button
4: Wait for 12 seconds to avoid being blocked
5: Find HTML elements containing the pathology text
6: Extract the inner HTML of each element and append it to `html_text` list
7: Combine the HTML text into a single string
8: **return** Combined HTML text =0

Algorithm 4 Function `main`

1: Call `get_selected_thread()` to retrieve HTML text of the selected thread
2: Call `extract_text_without_tags()` to extract text from HTML content
3: **return** Extracted text as a dictionary =0

4.4.4.3 Étape 3 : Structuration du Texte Brut

Après avoir récupéré les expériences pathologiques extraites via l'API exposée par le backend Python, le texte brut est structuré sous forme d'objets en utilisant LLaMA2, un modèle de langage large (LLM). Cette transformation est réalisée au niveau du contrôleur de Blogs, permettant de convertir efficacement le texte brut en objets structurés. Cette structuration facilite l'analyse et l'utilisation ultérieure des données, assurant une manipulation et un traitement optimisés des informations extraites.

4.5 Présentation des interfaces

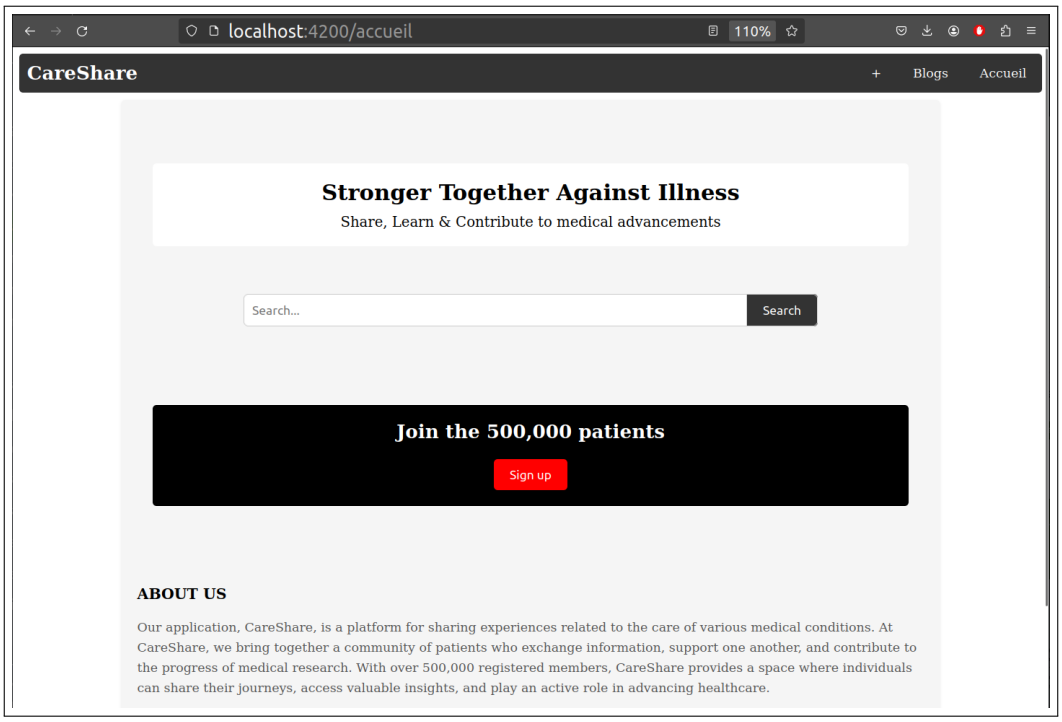


FIGURE 4.4 – Interface réalisée : page d'acceuil

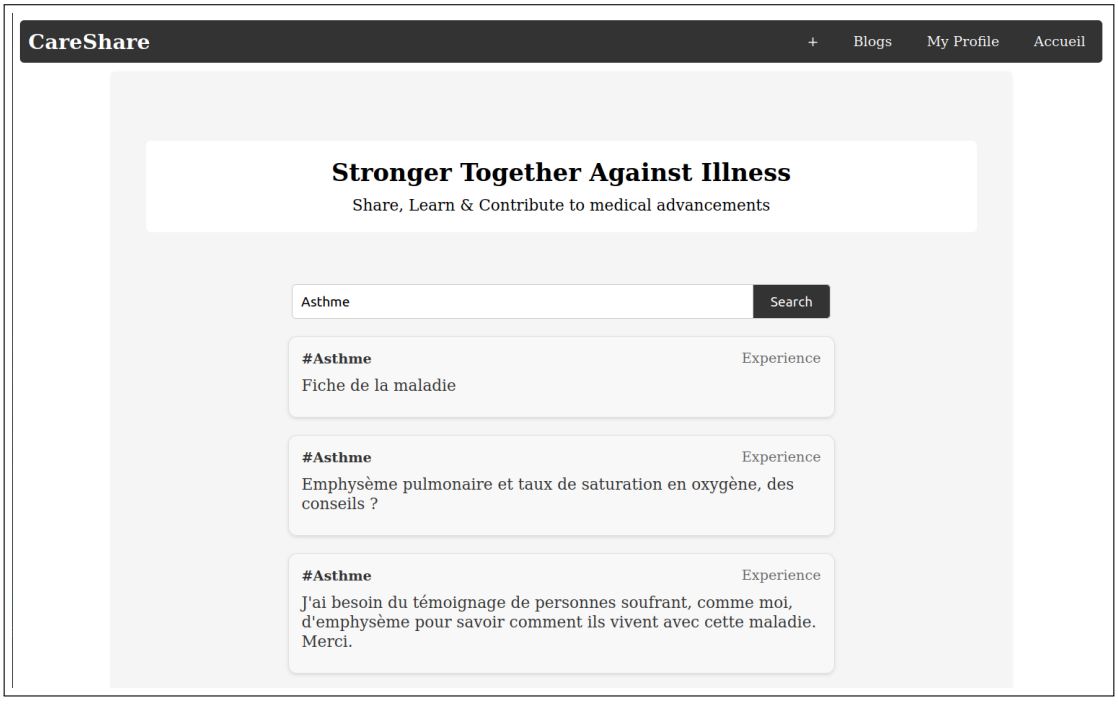


FIGURE 4.5 – Interface réalisée : Effectuer une recherche

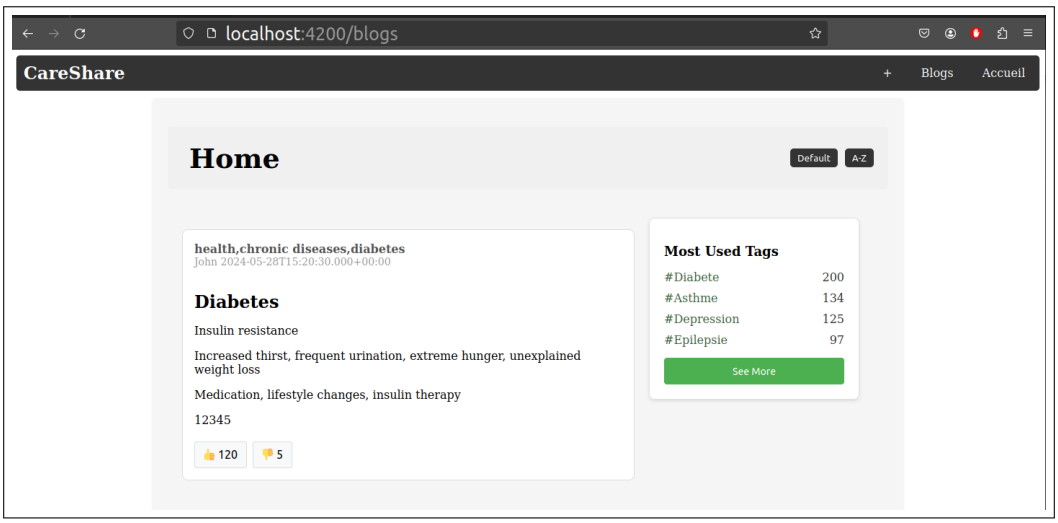


FIGURE 4.6 – Interface réalisée : Blogs

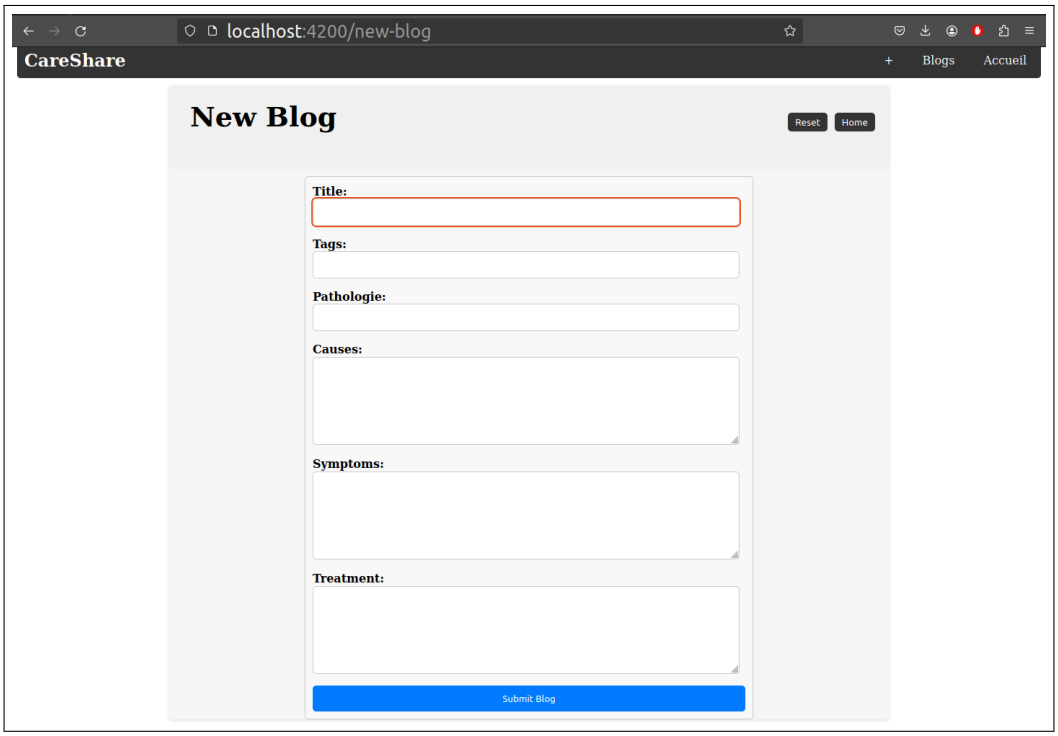


FIGURE 4.7 – Interface réalisée : Nouveau Blog

4.6 Conclusion

En résumé, la sélection minutieuse des outils et des technologies, combinée à des processus d'automatisation efficaces et à une méthodologie de développement agile, a été essentielle pour la réalisation réussie de notre application. Cette section fournit une base solide pour comprendre l'implémentation technique de notre projet, tout en offrant une visualisation concrète de l'interface utilisateur. Cela témoigne de notre engagement envers la qualité et l'efficacité dans tous les aspects de notre travail.

Conclusion générale et perspectives

La plateforme de partage d'expériences sur le soin de pathologies que nous avons développée représente une avancée significative dans l'utilisation des technologies web et de l'intelligence artificielle pour améliorer l'accès à l'information médicale et l'entraide entre patients. En permettant aux utilisateurs de créer et de consulter des blogs sur leurs expériences avec différentes pathologies, nous offrons un espace interactif et dynamique où l'information est non seulement partagée mais aussi enrichie par les contributions de chacun. Le système innovant de web scraping automatique constitue un pilier essentiel de notre plateforme. Cette fonctionnalité assure que même si le contenu n'est pas initialement présent sur notre site, les utilisateurs peuvent tout de même accéder à une riche base de données d'expériences provenant de diverses sources en ligne. En outre, l'intégration du modèle de langage Llama2 permet d'extraire des informations précises et structurées des blogs récupérés, optimisant ainsi la recherche et la classification des données.

Pour l'avenir, nous envisageons plusieurs axes d'amélioration et de développement. L'expérience utilisateur sera au cœur de nos préoccupations, avec le développement d'une interface plus intuitive et conviviale, ainsi que l'intégration de fonctionnalités de personnalisation pour offrir une expérience plus ciblée et pertinente. L'expansion des sources de données est également une priorité, avec l'augmentation du nombre de sites web surveillés pour enrichir davantage la base de données.

L'optimisation du web scraping est essentielle pour une extraction de données plus rapide et plus précise. Parallèlement, le renforcement de l'analyse et de la classification grâce à des modèles de machine learning plus avancés permettra d'améliorer la classification et l'analyse des blogs, et d'implémenter un système de recommandation basé sur les préférences et les historiques de recherche des utilisateurs.

En somme, notre projet se positionne comme une plateforme évolutive et prometteuse pour le partage d'expériences sur les pathologies, visant à créer une communauté solidaire et bien informée. Grâce à l'utilisation judicieuse des technologies modernes, nous envisageons de continuer à enrichir cette plateforme pour qu'elle devienne une ressource incontournable pour tous ceux qui cherchent à partager ou obtenir des informations sur les soins de santé.

Bibliographie

[1] 27-03-2024 :
Web Scraping Techniques and Applications : A Literature Review : https://www.researchgate.net/publication/367719780_Web_Scraping_Techniques_and_Applications_A_Literature_Review
Cet article propose une revue de littérature actualisée sur les techniques avancées de web scraping, en détaillant la conception de base d'un web scraper, ses applications dans divers secteurs, les différentes méthodes et technologies de web scraping, et en proposant une procédure pour développer un web scraping avec divers outils.

[2] 03-04-2024 :
A Practical Introduction to Web Scraping in Python : https://realpython.com/python-web-scraping-practical-introduction/
Le site présente une introduction pratique au techniques de web scraping en utilisant python et ses bibliothèques : Beautiful Soup, MechanicalSoup.

[3] 22-04-2024 :
Documentation de la bibliothèque Selenium https://robotframework.org/SeleniumLibrary/SeleniumLibrary.html
Le site présente la bibliothèque d'automatiser les tests de navigateurs web en Python, qui fournit des fonctions pour faciliter le web scraping en permettant d'interagir avec des pages web dynamiques.

[4] 23-04-2024 :
Data Transmission numerique : Scrapez automatiquement n'importe quel site web avec BeautifulSoup https://www.data-transitionnumerique.com/beautifulsoup-scraping/
Cet article présente la bibliothèque BeautifulSoup qui fournit des méthodes simples pour naviguer, rechercher et modifier un arbre d'analyse dans des fichiers HTML ou XML. Il transforme un document HTML complexe en un arbre d'objets Python.

[5] 02-05-2024 :
Documentation du framework Angular https://angular.dev/overview
Ce site présente le framework web Angular qui permet aux développeurs de créer des applications rapides et fiables.

[6] 04-05-2024 :
Guide de configuration de Llama https://llama.meta.com/docs/get-started/
Ce guide fournit des informations et des ressources pour vous aider à configurer Llama, y compris comment accéder au modèle, l'hébergement, des guides pratiques et d'intégration. De plus, vous trouverez des documents supplémentaires pour vous assister davantage lors de la construction avec Llama.

[7] 06-05-2024 :
Contenairisation des applications Spring Boot Docker https://spring.io/guides/topicals/spring-boot-docker
Ce site présente une façon pour intégrer les modèles AI dans des projets basés sur Spring Boot, pour notre cas on a utilisé le modèle Llama intégré à travers Ollama.

[8] 10-05-2024 :
Intégration de Ollama dans Spring Boot https://spring.io/
Ce site présente le framework java pour le developpement backend qui permet aux développeurs de créer des applications solides.

[9] 21-05-2024 :
Contenairisation des applications Spring Boot en utilisant Docker https://spring.io/guides/

`topicals/spring-boot-docker`
La conteneurisation des applications Spring Boot en utilisant Docker simplifie le déploiement et la gestion des applications en créant des conteneurs légers et portables.