

**Lume – Réseau d’articles intelligents**

**Réalisé par :**

-AZAMI Salim

- KHALAFTOU Youssef

-NADI Mohamed Ilyass

-OUAZRI Abdessalam

Encadre par :

-Dr. Hassna Bensag.

**Soutenu le 26/05/2025,**

Devant le jury composé de :

-Dr. Hassna Bensag .

.

***Remerciements***

***Nous adressons, en premier lieu, nos remerciements à Allah le Tout-Puissant, pour la force, la patience et la persévérance qu’Il nous a accordées tout au long de ce projet.***

***Nous exprimons notre profonde reconnaissance à Madame El Amrani, encadrante de ce projet, pour son encadrement de qualité, sa disponibilité, ainsi que ses conseils avisés qui ont grandement contribué à la bonne conduite de ce travail.***

***Nos remerciements vont également aux membres du jury, pour le temps qu’ils ont bien voulu consacrer à l’évaluation de notre travail et pour l’intérêt qu’ils y portent.***

***Enfin, nous remercions sincèrement l’ensemble du corps enseignant du département d’informatique de l’École Marocaine des Sciences de l’Ingénieur (EMSI) pour la richesse de leur enseignement et leur accompagnement tout au long de notre formation.***

***Dédicaces***

***Nous dédions ce mémoire à toutes les personnes qui, de près ou de loin, ont contribué à sa concrétisation.***

***À nos familles respectives, pour leur amour inestimable, leur patience, leur compréhension et leur soutien moral constant, qui ont été une source de force tout au long de ce parcours académique.***

***À nos enseignants et encadrants, pour leur encadrement rigoureux, leurs conseils avisés et leur disponibilité tout au long de cette aventure. Leur accompagnement a grandement enrichi notre travail et notre réflexion.***

***À nos collègues de promotion et amis, pour leurs encouragements, leurs échanges constructifs et leur présence bienveillante dans les moments de doute comme dans ceux de réussite.***

***Et enfin, à toutes celles et ceux qui nous ont soutenus, inspirés et motivés de quelque manière que ce soit, et qui ont cru en notre capacité à mener ce projet à terme. Recevez ici l’expression de notre profonde reconnaissance et de notre sincère gratitude.***

***Résumé***

**Lume** est une plateforme web sociale dédiée à la publication, la lecture et le partage d’articles, conçue pour encourager l’interaction intellectuelle entre utilisateurs. Développée avec Django, Tailwind CSS et HTMX, l’application offre une expérience fluide et responsive, adaptée à tous les types d’écrans. Elle permet aux utilisateurs de créer un compte, personnaliser leur profil, publier des articles, commenter les publications, liker les contenus, et suivre d’autres auteurs pour un fil d’actualité personnalisé.

Lume intègre un système de gestion d’articles avec des fonctionnalités avancées telles que la création, la modification, l’archivage (soft-delete) et la visualisation détaillée. Chaque article peut contenir du texte enrichi, des images, des tags, et propose une interface de lecture optimisée. Un système de commentaires nommé Echoes permet des réponses hiérarchisées avec likes et suppression logique, favorisant un débat structuré.

La plateforme propose également un module Blueprint dédié à la visualisation de diagrammes UML, où les utilisateurs peuvent consulter des schémas au format PlantUML avec aperçu visuel. L’interface inclut des animations modernes, des transitions douces, et respecte les bonnes pratiques d’accessibilité.

Sur le plan technique, Lume adopte une architecture modulaire, un découpage clair des responsabilités entre les applications internes, une base de données relationnelle PostgreSQL, et une gestion fine des accès. Des fonctionnalités futures sont envisagées : moteur de recherche, système de notification en temps réel, messagerie intégrée, et tableau de bord d’administration. Le projet est accompagné de tests unitaires, de documentation technique, et d’un script d’installation facilitant son déploiement et sa maintenance.

***Abstract***

## Lume

is a modern social web platform for publishing, reading, and sharing articles, built with Django, Tailwind CSS, and HTMX. It aims to offer an intuitive, interactive, and accessible experience that encourages intellectual exchange. Users can register, manage profiles, publish articles, comment, like, and follow authors to customize their content feed.

The platform includes rich article management, soft-delete functionality, and a responsive interface enhanced with smooth animations. The Echoes system enables nested comments with likes, while the Blueprint module allows users to view UML diagrams in **.puml** and image formats.

Technically, Lume features a modular Django architecture, PostgreSQL database, secure authentication, and adherence to accessibility standards. Planned features include real-time notifications, messaging, advanced search, and an admin dashboard. The project includes testing and documentation for deployment and maintenance.

***Table de matières***

Remerciements ……………………………………………………………………………… 1

Dédicace………………………………………………………………………….….………. 2

Résumé……………………………………………………………………………….……… 3

Abstract……………………………………………………………………………………….4

Table de matières……………………………………………………………………..………5

Liste des figures……………………………………………………………………................8

Introduction générale……………………………………………………………...………….9

Chapitre I : Généralités………………………………………………....................................10

1. Introduction………………………………………………………………………………..11

2. Définition d’un système d’information……………………………………………………11

3. Définition d’un système informatique…………………………………………………….11

3.1 Le matériel……………………………………………………………………………..11

3.2 Le logiciel……………………………………………………………………………...12

4. Types d’applications en informatique…………………………………………………….13

4.1 Applications Mobiles.…………………………………………………………...……..13

4.2 Applications desktop………………………………………………………………......13

4.3 Applications web………………………………………………………………………13

5. Cycle de vie d’un logiciel…………………………………………………………………14

5.1 Définition………………………………………………………………………………14

5.2 Modèles de cycle de vie………………………………………………………………..14

5.2.1 Modèle en cascade………………………………………………………………….14

5.2.2 Modèle en V………………………………………………………………………...14

5.2.3 Modèle par incrément….……………………………………………………………14

5.2.4 Modèle choisi………………………………………………………………………..15

6. Conclusion…………………………………………………………………………………15

Chapitre II : Analyse et Conception ………..………………………………………………..16

1. Introduction………………………………………………………………………………..17

2. Contexte du projet…………………………………………………………………………17

3. Problématique…………………………………………………………………………..17

4. Objectifs………………………………………………………………………………...17

5. Public cible……………………………………………………………………………..18

6. Étude de l’existant……………………………………………………………………...18

6.1 Analyse de sites similaires………………………………………………………….18

7. Cahier des charges……………………………………………………………………...19

8. Diagrammes UML……………………………………………………………………...19

8.1 Diagramme de cas d'utilisation……………………………………………………...20

8.2 Diagramme de classes ………….…………………………………………………..20

8.3 Diagramme de séquence ……………………………………………………………21

9. Modèle conceptuel de données (MCD)………………………………………………...23

10. Conclusion…………………………………………………………………………….24

Chapitre III : Réalisation du Projet………………………………………………………..25

1. Introduction……………………………………………………………………………..26

2. Présentation de l’environnement de développement……………………………………26

2.1 Technologies utilisées……………………………………………………………….26

2.1.1 Django……………………………………………………………………………..26

2.1.2 HTMX……………………………………………………………………….26

2.1.3 Mysql/sqlite……………………………………………………………………….27

2.1.4 Tailwind CSS……………………………………………………………………..27

2.1.5 Django Templates………………………………………………………………………….27

2.1.6 PlantUML………………………………………………………………………27

3. Architecture de l'application……………………………………………………………28

3.1 Structure du projet……………………………………………………………….….28

3.2 Organisation des composants………………………………………………………..29

4. Implémentation des fonctionnalités principales…………………………………..…….29

4.1 Page d’Accueil………………………………………………………………………30

4.2Authentification……………………………………………………………...………31

4.3 Navigation…………………………………………………………………….…….31

4.4 Publication & gestion des articles…………………………………………………...32

4.5 Commentaires et réactions (Echoes)……………………………………………….….33

4.6 Profil utilisateur………………………………………………………….…...33

4.7 Visualisation des diagrammes UML (Blueprint)……………………………..…34

5. Sécurité et gestion des accès………………………………………………..………34

6. Difficultés rencontrées et solutions…………………………………..…………….35

7. Conclusion…………………………………………………………………..….…..35

Conclusion générale……………………………………………………………..……36

Références bibliographiques…………………………………………………………..37

***Liste des figures***

Figure 1 : La machine virtuel………………………………………………..…………….. 12

Figure 2 : Diagramme de cas d’utilisation…………………………………........…………..20

Figure 3 : Diagramme de classes………...………………………………………………….21

Figure 4 : Diagramme de séquence pour la connexion….…………………………………. 22

Figure 5 : Diagramme de séquence pour passer des commandes..………………………… 22

Figure 7 : Structure du projet………………………………………………………………..28

Figure 8 : Page d’acceuil………...........................................................................................30

Figure 9 : Sign in/ Sign up………………...............................................................................31

Figure 10 : Page de navigation……………..……………………………………………….32

Figure 11 : Publication………………………………..……………………32

Figure 12 : commentaire…………………………………………………………...……………….33

Figure 13 : Visualisation des diagrammes UML……………………………………………33

Figure 14 : Profil utilisateur……………….………………………………………………..34

***Introduction Générale***

Avec l’essor des plateformes collaboratives et l’évolution rapide des technologies web, les modes de création, de partage et de consommation de contenu ont profondément changé. Dans ce contexte, les plateformes sociales dédiées à la lecture et à l’écriture s’imposent comme des outils essentiels pour encourager l’expression, la diffusion des idées et l’interaction intellectuelle en ligne.

C’est dans cette dynamique que s’inscrit notre projet, à travers la réalisation d’une application web baptisée **Lume**. Il s’agit d’une plateforme sociale moderne, intuitive et responsive, permettant aux utilisateurs de publier des articles, de les commenter, de les aimer, de suivre d’autres auteurs, et d’explorer un fil d’actualité personnalisé. L’objectif principal de Lume est de créer un espace d’échange structuré autour de contenus textuels, tout en favorisant la qualité des interactions et l’accessibilité de l’information.

La plateforme intègre également un module dédié à la visualisation de diagrammes UML, nommé Blueprint, qui permet de consulter des représentations visuelles de systèmes techniques à partir de fichiers PlantUML, renforçant ainsi la dimension éducative et documentaire du projet.

Afin de mener à bien ce projet, nous l’avons structuré comme suit :

* **Chapitre I : Généralités** – Ce chapitre introduit les notions de base relatives aux systèmes d’information, aux types d’applications web et aux modèles de cycle de vie logiciel.
* **Chapitre II : Analyse et Conception** – IIl traite de la problématique, des objectifs du projet, de l’analyse de l’existant, des besoins fonctionnels, et de la modélisation UML du système à travers des cas d’utilisation, diagrammes de classes, et autres représentations.
* **Chapitre III : Réalisation du projet** – Ce chapitre présente l’environnement technique, les technologies utilisées (telles que Django, HTMX, Tailwind CSS, PostgreSQL…), ainsi que les principales interfaces de l’application.

Nous conclurons ce mémoire par une **conclusion générale**, dans laquelle nous résumerons les résultats obtenus et proposerons des pistes d’évolution pour renforcer les fonctionnalités et la portée de la plateforme **Lume**.

Chapitre I

Généralités

**1. Introduction:**

Avec l’évolution rapide des technologies numériques, le développement d’applications web occupe aujourd’hui une place centrale dans notre quotidien. Que ce soit dans les domaines de la santé, de l’éducation, du commerce ou des services administratifs, les solutions informatiques sont devenues incontournables pour améliorer la productivité, l’accessibilité et la qualité des services.

Dans le cadre de notre projet, centré sur la conception et la réalisation d’une application web, il est essentiel de revenir sur certaines notions fondamentales en informatique. Ce chapitre a pour objectif de poser les bases conceptuelles nécessaires à la compréhension du travail réalisé. Nous y aborderons la définition d’un système d’information, celle d’un système informatique et de ses composants, les différents types d’applications informatiques, ainsi que les modèles de cycle de vie logiciel. Ces éléments permettront de situer notre démarche dans son contexte technique et méthodologique.

**2. Définition d’un système d’information:**

Le système d’information (SI) est défini comme un ensemble organisé de ressources (personnes, données, procédures, matériels et logiciels) permettant de collecter, stocker, traiter et diffuser l’information nécessaire au fonctionnement d’une organisation (entreprise, administration, établissement scolaire, etc.). Il peut s’agir de données financières, commerciales, comptables, managériales, ou fiscales [1].

**3. Définition d’un système informatique:**

Le système informatique constitue la composante technique du système d’information. Il regroupe les moyens matériels et logiciels permettant de traiter l’information de manière automatisée. Plus précisément, il s’agit d’un ensemble de ressources informatiques et de télécommunication ayant pour finalité d’élaborer, traiter, stocker, acheminer, présenter ou supprimer des données [2].

Le système informatique se compose de deux grandes catégories :

* + - Le matériel (hardware) : la partie physique et visible du système informatique.
    - **Le logiciel (software)** : la partie immatérielle, invisible à l’œil nu, comprenant les programmes et systèmes d’exploitation.

**3.1. Le matériel:**

Le matériel informatique, ou hardware, est composé de plusieurs éléments répartis en deux grandes catégories : les composants internes et les périphériques [3].

1. Composants internes

Ce sont les éléments logés dans l’unité centrale (UC) de l’ordinateur. Ils sont indispensables au bon fonctionnement du système, mais ne peuvent fonctionner seuls en dehors de l’UC :

* + Mémoires (vive, morte, disque dur)
  + Carte mere
  + Microprocesseur
  + Carte graphique
  + Carte son
  + Carte réseau
  + Bloc d’alimentation
  + Ventilateur

1. Les périphériques

Ce sont des dispositifs connectés généralement à l’extérieur de l’unité centrale, bien que certains puissent également s’y retrouver. Ils assurent l’entrée, la sortie ou le stockage de données :

Clavier

Souris

Écran

Imprimante

Scanner

Disques durs externes

Caméras

**3.2. Le logiciel: Du côté logiciel, un système informatique repose sur deux niveaux essentiels : le système d’exploitation et les applications. Le rôle principal du logiciel est d’offrir à l’utilisateur un ensemble de fonctionnalités adaptées à ses besoins, tout en masquant la complexité technique du matériel sous-jacent.**

**Pour cela, l’architecture logicielle est généralement organisée en couches, où chaque niveau fournit des services plus abstraits au niveau supérieur. Cette structuration progressive permet d’isoler la complexité technique et d’offrir une machine virtuelle à l’utilisateur final, c’est-à-dire une abstraction logicielle du matériel [4].**

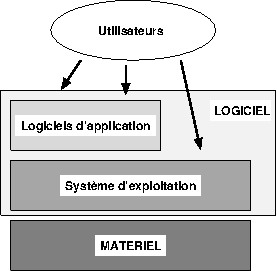


Figure 1:La Machine Virtuel.

**4. Types d’applications en informatique :**

Les applications informatiques se déclinent aujourd’hui en plusieurs catégories, selon leur usage, leur environnement d’exécution, et les technologies mobilisées. Les trois grandes familles dominantes sur le marché sont les **applications mobiles**, les **applications desktop** et les **applications web**.

## **4.1. Applications Mobiles :**

Les applications mobiles sont développées spécifiquement pour des systèmes d’exploitation mobiles comme **Android** ou **iOS**, et sont accessibles via des plateformes de distribution telles que **Google Play Store** ou **Apple App Store**. Elles sont conçues à l’aide de langages natifs (Java/Kotlin pour Android, Swift pour iOS), ou via des frameworks multiplateformes comme **React Native**, **Flutter** ou **Ionic**, permettant de cibler plusieurs systèmes avec une seule base de code.

Elles tirent parti des ressources matérielles du terminal (GPS, caméra, capteurs de mouvement, etc.) pour offrir des expériences interactives et personnalisées. La majorité de ces applications communiquent avec des serveurs distants pour récupérer des données, mettre à jour leur contenu, ou exécuter des traitements plus lourds [5].

**4.2. Applications desktop :**

Les applications desktop sont des logiciels installés et exécutés localement sur un ordinateur personnel. Elles ne nécessitent pas d’accès à Internet pour fonctionner. Parmi leurs avantages, on peut citer leur **rapidité**, leur **stabilité** et leur capacité à exploiter pleinement les ressources de la machine hôte.

Toutefois, ces applications présentent aussi des limites, notamment en termes de **portabilité** : elles doivent être installées manuellement sur chaque poste, et peuvent rencontrer des problèmes de compatibilité entre différents systèmes d’exploitation (Windows, macOS, Linux, etc.) [6].

## **4.3. Applications Web :**

Les applications web sont des applications accessibles via un navigateur Internet (comme Chrome, Firefox ou Edge), sans installation préalable sur le poste client. Elles utilisent les technologies web standards (HTML, CSS, JavaScript) et sont hébergées sur un serveur distant.

Leur principal atout réside dans leur accessibilité universelle : elles peuvent être consultées depuis n’importe quel appareil connecté (ordinateur, tablette, smartphone) et sont indépendantes du système d’exploitation. Elles offrent également une meilleure centralisation des données, une gestion simplifiée de la sécurité, et des mises à jour automatiques, réduisant les risques d’obsolescence.

Cependant, leur principal inconvénient est leur dépendance à une connexion Internet pour fonctionner correctement [6].

**5. Cycle de vie d'un logiciel :**

## **5.1. Définition :**

Le « **cycle de vie d'un logiciel** » (en anglais *software lifecycle*), désigne toutes les étapes du développement d'un logiciel, de sa conception à sa disparition. L'objectif d'un tel découpage est de permettre de définir des jalons intermédiaires permettant la validation du développement logiciel, c'est-à-dire il a conformité du logiciel avec les besoins exprimés, et la vérification du processus de développement, c'est-à-dire l'adéquation des méthodes mises en œuvre.

L'origine de ce découpage provient du constat que les erreurs ont un coût d'autant plus élevé qu'elles sont détectées tardivement dans le processus de réalisation. Le cycle de vie permet de détecter les erreurs au plus tôt et ainsi de maîtriser la [qualité](http://bomat.free.fr/site/qualite/qualite-intro.htm) du logiciel, les délais de sa réalisation et les coûts associés. [7]

## **5.2. Modèles de cycles de vie :**

Afin d'être en mesure d'avoir une méthodologie commune entre le client et la société de service réalisant le développement, des modèles de cycle de vie ont été mis au point définissant les étapes du développement ainsi que les documents à produire permettant de valider chacune des étapes avant de passer à la suivante. A la fin de chaque phase, des revues sont organisé.

### 5.2.1. Modèle en cascade :

## Le modèle de cycle de vie en cascade a été mis au point dès 1966, puis formalisé aux alentours de 1970. Il définit des phases séquentielles à l'issue de chacune desquelles des documents sont produits pour en vérifier la conformité avant de passer à la suivante. [8]

### 5.2.2. Modèle en V :

Le modèle de cycle de vie en V part du principe que les procédures de vérification de la conformité du logiciel aux spécifications doivent être élaborées dès les phases de conception. [8]

### 5.2.3. Modèle par incrément :

Dans un modèle incrémental, un noyau est développé au fur et à mesure des incréments bien sécrétés autour de ce noyau. Chaque incrément est développé selon un modèle de cycle de vie à savoir un modèle en V ou en cascade. C'est à dire que on part d'une pré étude que l'on adapte à un des modèles cités plus hauts (cascade ou V) afin de construire un système plus efficace et qui répond le mieux aux exigences du cahier de charges.[9]

### 5.2.4. Modèle choisi :

### Dans le cadre de notre projet, nous avons opté pour le modèle de cycle de vie en V. Ce choix s’appuie sur la rigueur qu’offre ce modèle, qui impose une validation systématique à chaque étape avant de poursuivre la suivante. Cette approche méthodique permet de détecter rapidement d’éventuelles erreurs et de les corriger à temps, limitant ainsi les problèmes en phase de développement avancé. En outre, le cycle en V assure une bonne traçabilité entre les besoins exprimés et les tests réalisés, garantissant ainsi la fiabilité et la cohérence du système que nous concevons. Ce cadre structurant nous semble particulièrement adapté aux contraintes et à la nature de notre projet.

**6. Conclusion :**

Dans ce chapitre, nous avons présenté les notions essentielles liées à notre projet, notamment les systèmes d’information, les types d’applications et le cycle de vie d’un logiciel. Ces bases théoriques nous aident à mieux comprendre le contexte et les choix techniques à venir.

Le prochain chapitre sera dédié à l’analyse des besoins et à la conception de notre application, étapes cruciales avant son développement.

Chapitre II

Analyse et Conception

**1. Introduction :**

Ce chapitre est consacré à l’étude approfondie du projet Lume, afin de définir avec précision ses besoins, ses objectifs et son périmètre fonctionnel. Nous y présentons la problématique à laquelle il répond, le contexte de sa mise en œuvre, l’analyse des solutions existantes ainsi que les exigences fonctionnelles et techniques. Cette phase d’analyse est essentielle pour concevoir une solution adaptée, fiable et évolutive. Enfin, nous exposons la conception du système à travers des modèles et diagrammes UML qui guideront son développement.

**2. Contexte du projet :**

Avec l’essor des réseaux sociaux et des plateformes collaboratives, la publication et la diffusion de contenu textuel prennent une place croissante dans les usages numériques. Cependant, au Maroc comme dans d’autres pays, peu de plateformes locales permettent de publier des articles, d’interagir autour de ceux-ci, et de créer un espace intellectuel dynamique et structuré. Le projet Lume vise à répondre à ce besoin, en proposant une application web sociale dédiée à la lecture, à l’écriture et au partage d’articles dans un environnement moderne, intuitif et accessible.

**3. Problématique :**

Dans un monde où l’information circule abondamment, les auteurs comme les lecteurs recherchent des plateformes à la fois simples, ergonomiques et interactives. Les solutions existantes manquent souvent de structuration dans la gestion des commentaires, des préférences de lecture ou du suivi d’auteurs. Il est donc essentiel de concevoir une plateforme permettant une publication fluide, une interaction intelligente (likes, commentaires imbriqués, suivi), et une organisation claire du contenu.

**4. Objectifs :**

Les objectifs de ce projet sont multiples et répondent aux besoins à la fois des créateurs de contenu et des lecteurs. Il s’agit de :

Concevoir une interface utilisateur claire, responsive et moderne ;

Permettre la publication, la gestion et l’archivage d’articles ;

Intégrer un système de commentaires imbriqués avec likes (Echoes) ;

Offrir un fil d’actualité personnalisé basé sur les auteurs suivis ;

Mettre en place un module de visualisation de diagrammes UML (Blueprint) ;

Garantir une authentification sécurisée et une gestion efficace des profils utilisateurs ;

Assurer une expérience fluide, accessible et évolutive.

### 5. Public cible :

La plateforme Lume s’adresse à un public large, composé :

* + - * + D’auteurs, rédacteurs et étudiants souhaitant publier et structurer leurs articles ;
        + De lecteurs passionnés de contenu technique, littéraire ou académique ;
        + De développeurs ou formateurs partageant des diagrammes UML pour illustrer des systèmes ;
        + D’utilisateurs en quête d’un espace d’interaction constructive autour du contenu.

Ce public varié partage un besoin commun : disposer d’une plateforme fiable et intuitive pour consulter, publier et interagir autour d’articles.

**6. Etude de l’existant :**

L’analyse des plateformes similaires est une étape essentielle pour mieux cerner les pratiques du marché et les attentes des utilisateurs. Elle permet d’identifier les bonnes pratiques, les fonctionnalités à inclure, ainsi que les lacunes à éviter.

**6.1 Analyse de sites similaires :**

Parmi les plateformes étudiées, Medium se distingue comme une référence internationale dans le domaine de la publication d’articles. Sa simplicité, son design épuré, la qualité de ses suggestions de contenu et son système de tags en font une expérience agréable pour les lecteurs et les auteurs.

Cependant, Medium ne permet pas de publier des diagrammes techniques intégrés de façon native, et ne répond pas toujours aux besoins spécifiques d’un public académique ou technique.

À partir de cette analyse, Lume se positionne comme une alternative locale et spécialisée, enrichie d’un module UML (Blueprint) et d’une interaction plus approfondie autour des commentaires (Echoes).

Site consulté : https://medium.com

**7. Cahier des charges :**

Le cahier des charges établit les exigences fonctionnelles et techniques du projet, garantissant ainsi un développement structuré et conforme aux attentes.

### Fonctionnalités principales

* Inscription, connexion et déconnexion sécurisées des utilisateurs.
* Gestion des profils avec avatar, biographie et personnalisation ;
* Publication d’articles avec éditeur de texte enrichi et images ;
* Archivage (soft-delete) et filtrage des articles par visibilité ;
* Fil d’actualité dynamique : articles récents, tendance, suivis ;
* Système de commentaires imbriqués avec likes (Echoes) ;
* Visualisation de fichiers .puml et diagrammes UML (Blueprint) ;
* Interface responsive et accessible (mobile, tablette, desktop) ;
* Notifications visuelles (toasts) pour les actions importantes ;
* Navigation fluide, chargement dynamique avec HTMX.

**8. Diagrammes UML :**

Qu'est-ce qu'un diagramme UML ? Un diagramme UML est un moyen de visualiser des systèmes et des logiciels à l'aide du langage de modélisation unifié (UML).[10]

A diagram of a diagram

AI-generated content may be incorrect.**8.1. Diagramme de cas d’utilisation :**

**The Lume platform involves three main actors: the Utilisateur (user), the Auteur (author), and the Administrateur (administrator). Users can register, log in, manage their profiles, browse articles, search by title or author, like and comment on articles using the Echoes system, reply to comments, follow authors, view UML diagrams via the Blueprint module, receive visual notifications, and access a personalized content feed. Authors, who are also users, have additional privileges allowing them to publish, edit, archive, and tag articles, as well as view reading statistics. The administrator has extended capabilities including user management, content moderation, and access to a dedicated admin dashboard. These use cases collectively define the interactive and collaborative nature of the Lume platform.**

Figure 2 Diagramme de cas d'utilisation

**8.2. Diagramme de classe :**

Ce diagramme de classes représente la structure des données principales de la plateforme .  
On y voit les entités clés comme **Utilisateur**, **Article**, **Commentaire**, **Like**, **Follow** et leurs relations.  
Par exemple, un utilisateur peut publier des articles, commenter ceux des autres, aimer des commentaires et suivre d’autres utilisateurs.  
Chaque article est associé à un auteur et peut contenir plusieurs commentaires, tout comme chaque commentaire peut recevoir plusieurs likes.  
Ce modèle permet une organisation claire et réutilisable des données dans le projet.

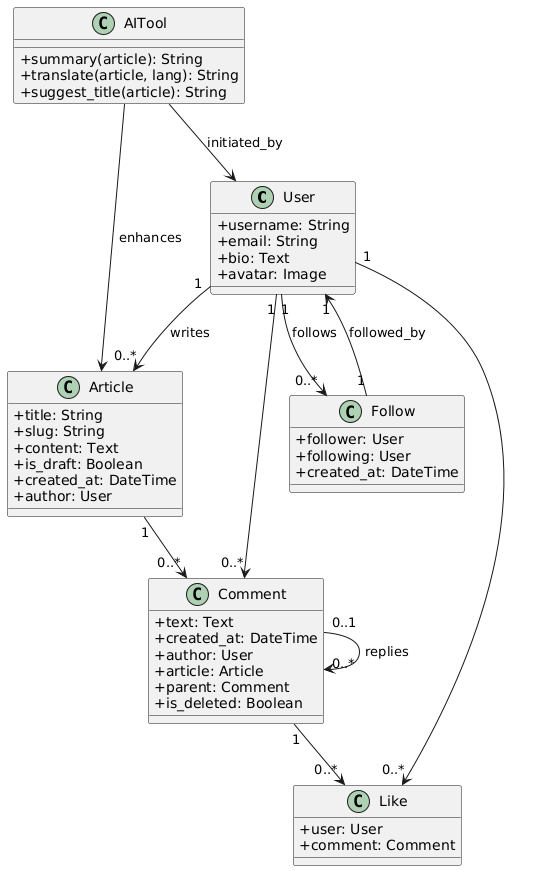


Figure 3 Diagramme de classes

**8.3. Diagramme de séquence :**

Ce diagramme illustre le processus de **connexion d’un utilisateur** sur la plateforme **Lume**.  
L'utilisateur saisit son adresse e-mail et son mot de passe dans le formulaire d’authentification. Ces informations sont envoyées au **système d’authentification Django**, qui vérifie leur validité à l’aide de la base de données.  
En cas de succès, une session sécurisée est créée, l’utilisateur est authentifié, et il est redirigé vers son **fil d’articles** ou son **profil**. En cas d’échec, un message d’erreur est affiché, lui indiquant que les identifiants sont incorrects.

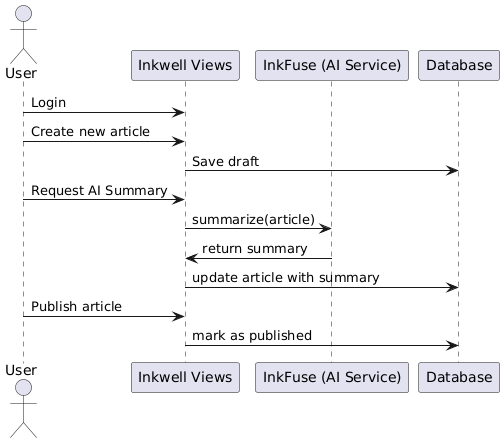
A diagram of a diagram

AI-generated content may be incorrect.

Figure 4 Diagramme de sequence pour la connexion

Figure 5 Diagramme de sequence pour passer des commandes

Ce diagramme de séquence illustre le processus de création et de publication d’un article sur la plateforme **Lume**, avec assistance d’un résumé généré par intelligence artificielle. L’utilisateur se connecte, crée un nouvel article, puis l’enregistre comme brouillon. Il peut ensuite demander un résumé automatique via le service IA **InkFuse**, qui traite le contenu et retourne une version résumée. Ce résumé est intégré à l’article, qui est ensuite mis à jour dans la base de données. Enfin, l’utilisateur publie l’article, marquant ainsi sa mise en ligne officielle. Ce scénario montre l’interaction fluide entre l’utilisateur, les vues Django (Inkwell), le service IA (InkFuse) et la base de données.



**9. Modèle conceptuel de données (MCD) :**

Le MCD montre les principales entités et leurs liens dans **Lume**.

* **Utilisateur** : stocke les informations des membres de la plateforme.
* **Article** : contient les détails des articles publiés.
* **Commentaire** : les réponses des utilisateurs aux articles ou à d’autres commentaires.
* **Like** : représente les réactions positives des utilisateurs sur les commentaires.
* **Suivi (Follow)** : enregistre les relations entre utilisateurs (qui suit qui).

Les relations indiquent qu’un utilisateur peut publier plusieurs articles, commenter, aimer des commentaires et suivre d’autres utilisateurs. Ce modèle organise les données pour gérer efficacement les utilisateurs, les publications, les interactions et les connexions sociales sur la plateforme.

**10. Conclusion :**

Dans ce chapitre, nous avons procédé à l’analyse fonctionnelle et conceptuelle de la plateforme Lume. À travers des diagrammes UML tels que le diagramme de classes, de séquence et de cas d’utilisation, nous avons modélisé les principales entités du système, leurs interactions ainsi que les scénarios d’utilisation les plus significatifs.

Ces représentations nous ont permis de structurer notre compréhension de la plateforme, de clarifier les relations entre les objets métiers (comme Utilisateur, Article, Commentaire et Suivi), et de poser une base solide pour la phase d’implémentation. Cette étape de conception joue un rôle clé pour assurer une architecture cohérente, évolutive et aisée à maintenir dans la suite du développement.

***Chapiter III***

Réalisation du projet

**1. Introduction :**

La mise en œuvre d’un projet informatique nécessite le recours à un ensemble d’outils, de technologies et de langages adaptés à ses objectifs. Dans ce chapitre, nous présentons tout d’abord l’environnement de développement utilisé pour construire notre application web Lume, en détaillant les choix technologiques effectués ainsi que leur justification.

Nous aborderons ensuite l’architecture de l’application et la modélisation physique des données, avant de passer à l’implémentation des principales fonctionnalités. Enfin, ce chapitre se conclura par une présentation visuelle de certaines interfaces clés de l’application, illustrant ainsi le résultat concret du travail réalisé.

**2. Présentation de l’environnement de développement :**

Dans cette section, nous allons détailler l’environnement de développement utilisé pour concevoir et développer notre application web Lume. Le choix des outils, langages et frameworks s’est basé sur leur robustesse, leur modernité, ainsi que leur pertinence dans une architecture modulaire et maintenable à long terme.

**2.1. Technologies utilisées :**

**2.1.1. Django:**

Django est un framework web Python open source, basé sur le modèle MVC, qui permet un développement rapide et sécurisé. Il intègre nativement un ORM, un moteur de templates, une interface d’administration, et une gestion fine des utilisateurs, ce qui en fait un choix idéal pour la création d’une application web structurée comme **Lume**. [11]

**2.1.2. HTMX:**

HTMX est une bibliothèque JavaScript légère qui permet d’enrichir les applications web avec des interactions dynamiques sans avoir à écrire du JavaScript complexe. Elle facilite le chargement partiel des contenus, les actions AJAX, et améliore considérablement l’expérience utilisateur tout en conservant une logique côté serveur. [12]

**2.1.3. Mysql/sqlite:**

est un système de gestion de base de données relationnelle puissant et open source. Il est particulièrement adapté pour gérer des données complexes et assurer l’intégrité des relations entre les différentes entités (articles, utilisateurs, commentaires, etc.). [14]

**2.1.4. Tailwind CSS :**

**Tailwind CSS** est un framework permettant aux développeurs de personnaliser totalement et simplement le design de leur application ou de leur site web. Avec ce **framework CSS**, il est possible de créer un design d’interface au sein même du fichier HTML. Cette façon de programmer n’interfère pas avec les pratiques recommandées par le W3C comme celle de séparer le HTML des feuilles de style CSS.[14]

**2.1.5. PlantUML  :**

PlantUML est un outil open source permettant de créer des diagrammes UML à partir d’un langage de description textuel simple. Il est utilisé dans le module Blueprint de Lume pour générer dynamiquement des schémas UML lisibles par les utilisateurs. [15]

**2.1.6. Python:**

Python est le langage de programmation principal utilisé pour développer **Lume**. Il est connu pour sa lisibilité, sa simplicité et sa grande richesse en bibliothèques. Il s’intègre parfaitement avec Django, tout en offrant des possibilités étendues pour l’automatisation, le traitement de données, et l’intelligence artificielle. [16]

**3. Architecture de l'application :**

L’architecture d’une application constitue l’ensemble des choix techniques et structurels qui permettent d’assurer la maintenabilité, la performance et l’évolutivité du projet.  
Dans cette section, nous allons décrire l’organisation générale de notre application **Lume**, en mettant l’accent sur la structure des applications Django (modulaires), les routes principales, la séparation front-end/back-end, ainsi que le rôle des différents composants fonctionnels tels que Echoes (commentaires), Blueprint (diagrammes UML), Inkwell (articles), et Persona (utilisateurs et authentification).

**3.1. Structure du projet :**

L’application BookHaven suit une architecture modulaire basée sur React, facilitant la réutilisation et la séparation des responsabilités. Voici une structure simplifiée du projet :

lume/

├── blueprint/ → Module de gestion des diagrammes UML (Blueprint)

│ ├── admin.py → Configuration d’administration Django

│ ├── apps.py → Déclaration de l’application

│ ├── migrations/ → Fichiers de migration de la base de données

│ ├── models.py → Modèles de données (DiagrammeUMA screenshot of a computer

AI-generated content may be incorrect.L, etc.)

│ ├── templatetags/ → Filtres personnalisés pour les templates

│ ├── urls.py → Routes propres au module

│ └── views.py → Vues de gestion des diagrammes UML

├── echoes/ → Module des commentaires imbriqués (Echoes)

│ ├── models.py → Modèles Comment, Like

│ ├── forms.py → Formulaires de soumission de commentaires

│ ├── urls.py → Routes liées aux commentaires

│ └── views.py → Vues de création et gestion des réponses

├── inkwell/ → Module central des articles

│ ├── models.py → Modèle Article

│ ├── urls.py → Routes (création, liste, détails)

│ └── views.py → Logique d’affichage des articles

├── persona/ → Module de gestion des utilisateurs

│ ├── models.py → Profil utilisateur

│ ├── views/ → Vues personnalisées (login, register, profil)

│ ├── templates/ → Templates liés à l’authentification

│ └── urls.py → Routes liées à l’authentification

Figure 6 Structure du projet

├── pulse/ → Module de gestion des feeds (fil d’actualité)

│ ├── views.py → Vues des feeds “suivis” et “tendances”

│ ├── models.py → Logique de suivi, tri, etc.

│ └── urls.py → Routes associées

├── lume/ → Répertoire principal du projet Django

│ ├── settings.py → Paramètres globaux du projet

│ ├── urls.py → Routes principales du projet

│ └── wsgi.py/asgi.py → Fichiers d’exécution serveur

├── media/ → Dossier pour les fichiers uploadés (avatars, UML, etc.)

├── static/ → Fichiers statiques (images, icônes, CSS compilé)

├── templates/ → Templates HTML globaux et partiels

├── manage.py → Point d’entrée pour les commandes Django

├── pyproject.toml → Configuration du projet (avec Poetry)

├── poetry.lock → Fichier de verrouillage des dépendances

├── README.md → Présentation du projet

├── start-djngo.sh → Script de lancement rapide du serveur

└── lume\_cahier\_des\_charges.md → Spécifications fonctionnelles du projet

**3.2. Organisation des composants :**

Les composants et modules de Lume ont été organisés de manière cohérente et modulaire afin de garantir une meilleure lisibilité, évolutivité et maintenabilité du code :

* + - * + Composants UI génériques : Boutons, cartes d’article, formulaires de commentaire, champs de recherche, etc.
        + Composants spécifiques : Publication d’articles, gestion des commentaires (Echoes), visualisation de diagrammes (Blueprint), barre de navigation, formulaires d’authentification, etc.
        + Pages : Chaque page de l’application correspond à une URL précise (ex. /articles, /profile, /login, /blueprint).
        + Vues Django : Les fonctionnalités principales sont réparties dans les vues des applications internes (persona, inkwell, echoes, blueprint), chacune traitant un domaine métier spécifique.
        + Templates : Les interfaces sont définies dans des templates partagés ou propres à chaque module, avec des partials réutilisables.

Ce découpage facilite la maintenance du projet et permet à chaque membre de l’équipe de se concentrer sur une fonctionnalité précise sans générer de conflits.

**4. Implémentation des fonctionnalités principales :**

Dans cette section, nous présentons les principales fonctionnalités que nous avons implémentées dans le cadre de notre application web **Lume**. Ces fonctionnalités ont été conçues pour offrir une expérience utilisateur fluide, interactive et cohérente, aussi bien pour les utilisateurs réguliers (lecteurs, auteurs) que pour l’administrateur de la plateforme.

**4.1. Page d’Accueil :**

A screenshot of a website

AI-generated content may be incorrect.

Figure 7 Page d’acceuil

**4.2. Authentification :**

L’authentification permet aux utilisateurs de se connecter de manière sécurisée à la plateforme **Lume**. Elle repose sur le système intégré de Django, utilisant des sessions protégées et des tokens CSRF pour sécuriser les échanges. Les utilisateurs saisissent leur adresse e-mail et leur mot de passe via un formulaire dédié. En cas d’identifiants incorrects, un message d’erreur est affiché, tandis qu’en cas de succès, l’utilisateur est redirigé vers son espace personnel ou le fil d’articles. Ce mécanisme assure un accès restreint aux fonctionnalités sensibles telles que la publication, la gestion du profil ou les interactions sociales.

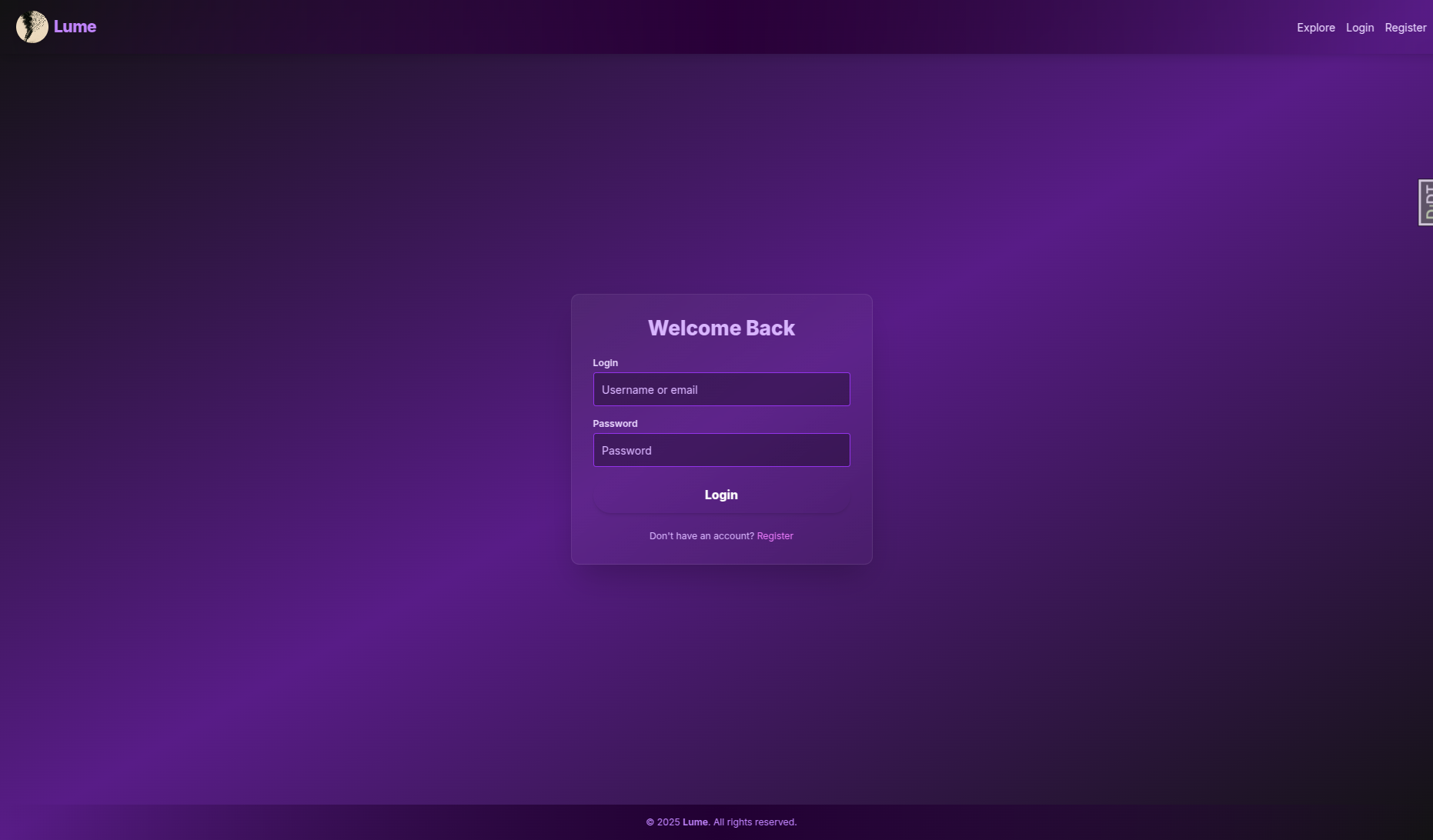


Figure 8 Sign in/ Sign up

**4.3. Navigation :**

Une barre de navigation est présente sur toutes les pages principales de l’application **Lume**. Elle permet d’accéder facilement aux différentes sections telles que l’accueil, les articles, la publication, le profil utilisateur, les notifications ou encore l’espace d’administration. Cette navigation a été conçue pour être claire, intuitive et responsive, garantissant une accessibilité optimale sur tous les types d’écrans.

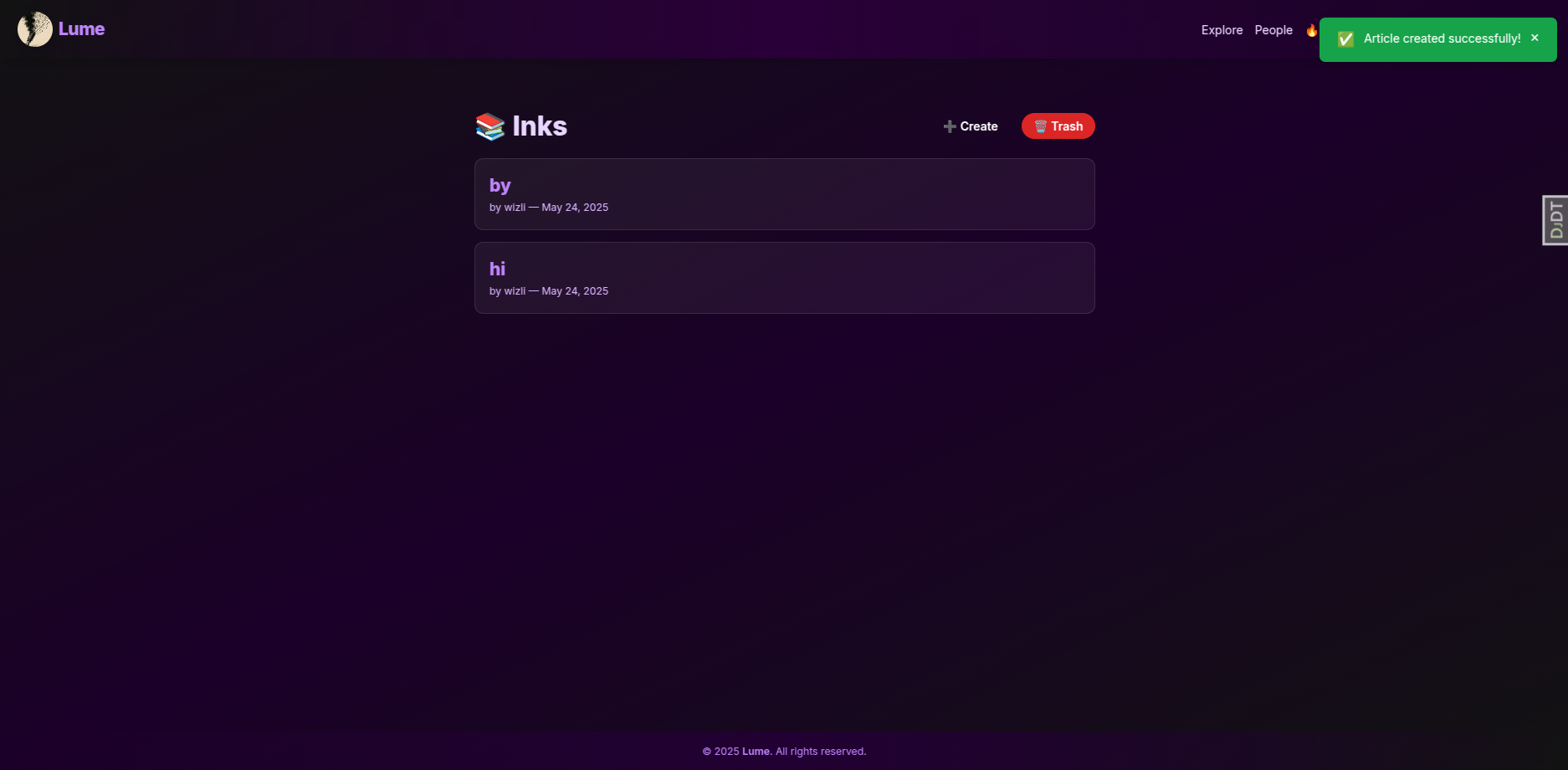


Figure 9 Page de navigation

**4.4. Publication & gestion des articles :**

L’utilisateur peut effectuer une recherche d’articles à partir d’un champ dédié, en saisissant un mot-clé lié au titre, à l’auteur ou au contenu. Des filtres sont également disponibles pour affiner les résultats en fonction de critères comme les tags, la date de publication ou l’auteur suivi. Ces fonctionnalités permettent d’accéder rapidement aux publications pertinentes, tout en améliorant l’expérience de navigation sur la plateforme.

A screenshot of a social media post

AI-generated content may be incorrect.

Figure 10 Barre de recherche et filtre

**4.5. Commentaires et réactions :**

L’utilisateur a la possibilité de commenter les articles publiés, de répondre à d’autres commentaires et d’exprimer son appréciation à travers un système de likes. Les commentaires sont organisés de manière hiérarchique grâce au module **Echoes**, permettant des discussions claires et structurées. Chaque utilisateur peut également supprimer ses propres commentaires, qui sont alors archivés via une suppression logique. Ce système favorise l’interaction, la participation active et l’échange d’idées au sein de la communauté.

A screenshot of a computer

AI-generated content may be incorrect.

Figure 11 Panier

**4.6. Profile utilisateur :**

Chaque utilisateur dispose d’un profil personnel qu’il peut consulter et personnaliser. Ce profil comprend des informations telles que le nom, la biographie, l’avatar, ainsi que la liste des articles publiés. L’utilisateur peut également accéder à ses commentaires, consulter les auteurs qu’il suit et gérer ses paramètres de compte. Cette section offre une vue centralisée de l’activité de l’utilisateur sur la plateforme, favorisant une expérience personnalisée et continue.

A screenshot of a computer

AI-generated content may be incorrect.

Figure 12 Favoris

**4.7. Visualisation des diagrammes UML :**

La plateforme **Lume** intègre un module nommé **Blueprint**, dédié à la visualisation de diagrammes UML. Les utilisateurs peuvent consulter des fichiers .puml ou des images de diagrammes directement depuis l’interface. Chaque diagramme est associé à un auteur, accompagné d’un titre et d’une description explicative. Ce module renforce l’aspect pédagogique de la plateforme, en permettant le partage de représentations techniques claires et structurées au sein des articles ou dans des pages dédiées.

A screenshot of a computer

AI-generated content may be incorrect.

Figure 13 Profil utilisateur

**5. Sécurité et gestion des accès :**

La sécurité est un aspect fondamental dans toute application web. Dans notre projet **Lume**, plusieurs mesures ont été mises en place afin d’assurer la protection des données utilisateurs et l’intégrité des contenus partagés :

**Authentification sécurisée** : Le système repose sur l’authentification fournie par Django, intégrant des sessions protégées et des tokens CSRF pour les formulaires sensibles.

**Chiffrement des mots de passe** : Les mots de passe sont hachés à l’aide d’algorithmes robustes (PBKDF2) fournis par Django, et ne sont jamais stockés en clair.

**Protection des vues** : Certaines pages (comme la publication d’articles, la gestion du profil ou les commentaires) sont restreintes aux utilisateurs authentifiés. Django utilise des décorateurs et des mixins (LoginRequiredMixin) pour sécuriser ces accès.

**Permissions et filtrage des données** : Les vues sont filtrées pour afficher uniquement les contenus associés à l’utilisateur connecté (ses articles, ses commentaires, ses suivis), évitant ainsi toute fuite de données.

**Validation côté client** : Des vérifications sont également effectuées dans les formulaires (via JavaScript ou HTMX) afin d’empêcher l’envoi de données incorrectes ou malveillantes.

**6. Difficultés rencontrées et solutions :**

Comme tout projet de développement, la réalisation de **Lume** a comporté plusieurs défis. Voici les principaux obstacles rencontrés ainsi que les solutions apportées :

* + - * + **Organisation modulaire avec Django** : Le découpage de l’application en modules indépendants (articles, commentaires, profils, etc.) a demandé une rigueur dans la gestion des routes, des dépendances et des templates. Cette complexité a été maîtrisée grâce à une architecture claire et à l’usage cohérent des namespaces.
        + **Commentaires imbriqués (Echoes)** : La gestion récursive des réponses aux commentaires a nécessité une logique spécifique dans les modèles et templates. L’utilisation d’une relation ForeignKey sur soi-même (self-relation) et le rendu conditionnel des sous-commentaires ont permis de résoudre ce point.

**Mise en page responsive** : Adapter l’interface à tous les écrans avec Tailwind CSS a exigé des ajustements fréquents. L’utilisation des classes utilitaires spécifiques aux breakpoints (sm:, md:, lg:) a permis de garantir une présentation fluide.

* + **Sécurité des données utilisateurs** : Protéger les actions sensibles comme la suppression d’un article ou la modification d’un profil a demandé l’ajout de contrôles d’autorisations dans les vues (request.user == object.author).
  + **Affichage des diagrammes UML (Blueprint)** : Intégrer PlantUML et permettre la prévisualisation d’images ou fichiers .puml a requis la gestion d’upload, de parsing et de rendu côté serveur. Une structuration claire du module Blueprint a permis d’y parvenir.

**7. Conclusion :**

Dans ce chapitre, nous avons présenté en détail la phase de réalisation de notre application web **Lume**. Nous avons tout d'abord décrit les technologies et outils utilisés, en expliquant le choix de chaque composant du stack technique. Ensuite, nous avons exposé l’architecture globale du projet ainsi que l'organisation du code et des modules internes.

La mise en œuvre des fonctionnalités principales — telles que l’authentification, la gestion des profils, la publication d’articles, les commentaires imbriqués (Echoes), et la visualisation des diagrammes UML (Blueprint) — a permis de concrétiser les besoins identifiés lors de la phase d’analyse, en un système interactif et opérationnel.

La gestion rigoureuse de la sécurité et les solutions apportées aux différentes difficultés rencontrées témoignent de l’importance d’une approche structurée, modulaire et adaptable dans le développement d’un projet web moderne.  
Cette étape marque l’aboutissement technique du projet, tout en ouvrant la voie à de futures évolutions fonctionnelles et ergonomiques.

**Conclusion générale :**

La réalisation de ce projet nous a permis de mettre en pratique les connaissances acquises au cours de notre formation, en combinant à la fois les aspects théoriques et techniques du développement d'une application web complète. À travers les différentes étapes — de l’analyse des besoins jusqu’à la mise en œuvre fonctionnelle — nous avons pu explorer les principes de conception logicielle, de modélisation UML, ainsi que l’intégration de technologies modernes telles que **Django**, **Tailwind CSS**, **HTMX**, ou encore My**SQL**.

Ce projet nous a également sensibilisés à l’importance de l’organisation, de la rigueur et de l’adaptabilité, notamment face aux difficultés techniques rencontrées au cours du développement. Chaque obstacle a représenté une opportunité d’apprentissage, en renforçant nos compétences en recherche de solutions, en autonomie et en prise d’initiative.

L’application **Lume** répond aux objectifs fixés en amont, en offrant une plateforme sociale de publication d’articles avec une interface moderne, un système de commentaires structuré (Echoes), et un module de visualisation de diagrammes UML (Blueprint). Elle pose les bases d’un outil évolutif, avec des perspectives d’amélioration futures telles que l’ajout de notifications en temps réel, une messagerie interne, ou encore une interface d’administration enrichie.

Enfin, cette expérience a été profondément enrichissante, tant sur le plan professionnel que personnel, et constitue une base solide pour relever avec confiance les futurs défis dans le domaine du développement web et des technologies de l’information.

**Références bibliographiques :**

[1] Payfit, Qu’est-ce qu’un système d’information ?, [en ligne] : https://payfit.com/fr/fiches-pratiques/systeme-information/ [Consulté le 11 mai 2025].

[2] Marché Public, Définition d’un système informatique, [en ligne] : http://www.marche-pubic.fr/Terminologie/Entrees/systeme-informatique.htm [Consulté le 11 mai 2025].

[3] UQAM, INF1026 – Cours Séance 1, [en ligne] : https://gdac.uqam.ca/INF1026/Cours/Seance1.pdf [Consulté le 11 mai 2025].

[4] Institut Mines-Télécom SudParis, Cours UNIX – Chapitre 9, [en ligne] : http://www-inf.it-sudparis.eu/cours/UNIX/Web/9.html [Consulté le 11 mai 2025].

[5] Aquilapp, Qu’est-ce qu’une application mobile ?, [en ligne] : https://www.aquilapp.fr/ressources/projet-mobile/quest-ce-quune-application-mobile [Consulté le 11 mai 2025].

[6] Uni2Grow Cameroun, Application mobile, desktop ou web : laquelle choisir ?, [en ligne] : https://www.uni2growcameroun.com/application-mobile-desktop-web-laquellechoisir [Consulté le 11 mai 2025].

[7] UNSW, Cycle de vie du logiciel – Partie I, [en ligne] : https://web.maths.unsw.edu.au/~lafaye/CCM/genie-logiciel/cycle-de-vie.htm [Consulté le 11 mai 2025].

[8] UNSW, Cycle de vie du logiciel – Partie II, [en ligne] : https://web.maths.unsw.edu.au/~lafaye/CCM/genie-logiciel/cycle-de-vie-duLogiciel.htm [Consulté le 11 mai 2025].

[9] Supinfo, Cycle de vie d’un logiciel, [en ligne] : https://www.supinfo.com/articles/3210-cycle-vie-logiciel [Consulté le 11 mai 2025].

[10] Miro, Qu’est-ce qu’un diagramme UML ?, [en ligne] : https://miro.com/fr/diagramme/qu-est-ce-qu-un-diagramme-uml/ [Consulté le 11 mai 2025].

[11] 50A, React : bibliothèque JavaScript, [en ligne] : https://www.50a.fr/0/react [Consulté le 11 mai 2025].

[12] Cellenza, An Introduction to TypeScript, [en ligne] : https://blog.cellenza.com/en/software-development/an-introduction-to-typescript/ [Consulté le 11 mai 2025].

[13] Alsacréations, Présentation de Vite.js, [en ligne] : https://www.alsacreations.com/article/lire/1903-Qu-est-ce-que-Vite.html [Consulté le 11 mai 2025].

[14] Numendo, Framework Tailwind CSS, [en ligne] : https://www.numendo.com/blog/framework/tailwind-css-framework-totalement-personnalisable/ [Consulté le 11 mai 2025].

[15] Hashnode, C’est quoi Supabase ?, [en ligne] : https://apprendre-supabase.hashnode.dev/cest-quoi-supabase [Consulté le 11 mai 2025].

[16] Oracle, Définition de PostgreSQL, [en ligne] : https://www.oracle.com/fr/database/definition-postgresql/ [Consulté le 11 mai 2025].