**Université Abdelmalek Essaâdi**

**FACULTE DES SCIENCES ET TECHNIQUES DE TANGER**

**Licence en Analyse de Données**

**Rapport de Projet : Développement de l'application Tech Horizons**

**Rédigé par : Nouhaila EL ATIFI**

**Kaoutar ZRAA**

**Fatima Zahra ELALLALI**

**Nisrine RAKTA**

**Encadré par : Mr. M'hamed AIT KBIR**

**Mr. Yasyn EL YUSUFI**

"**Tech Horizons**" est une application Web innovante destinée à gérer un magazine en ligne axé sur les dernières avancées technologiques. Elle permet à ses utilisateurs, qu'ils soient invités, abonnés, responsables de thème ou éditeurs, d'interagir de manière dynamique avec le contenu du magazine.

Les utilisateurs peuvent consulter les articles liés à des thèmes spécifiques tels que l’Intelligence Artificielle, la Cybersécurité, et la Réalité Virtuelle, gérer leurs abonnements et consulter leur historique de navigation. Les abonnés ont la possibilité de proposer des articles, de les noter, et de participer à des discussions via un chat intégré. Les responsables de thème et les éditeurs peuvent gérer le contenu, modérer les discussions et consulter des statistiques détaillées.

Ce projet, développé en PHP/MySQL, vise à offrir une plateforme interactive et sécurisée qui stimule l'engagement des utilisateurs autour des enjeux technologiques modernes. Grâce à son interface intuitive et ses fonctionnalités personnalisées, "Tech Horizons" s’impose comme une ressource incontournable pour les passionnés et professionnels de la technologie.

**Plan de Travail :**

1. Introduction
2. Analyse et Conception
3. Développement
4. Tests et Validation
5. Présentation des Résultats Visuels
6. Problèmes Techniques et Solutions
7. Conclusion

**Introduction :**

Le projet **"Tech Horizons"** consiste à développer une application Web innovante pour gérer un magazine en ligne axé sur les dernières avancées technologiques. L'objectif principal est d'offrir une interface intuitive et personnalisée permettant aux utilisateurs de consulter des articles, de gérer leurs abonnements aux thèmes (Intelligence artificielle, Cybersécurité, Internet des objets, etc.) et de proposer du contenu.

Les utilisateurs sont classés en quatre catégories :

* **Invité** : Accès limité aux numéros publics et informations générales.
* **Abonné** : Accès à un espace personnalisé pour consulter des articles, gérer des abonnements et l’historique.
* **Responsable de thème** : Gestion des abonnements et des articles liés à un thème précis.
* **Editeur** : Gestion globale du contenu, des utilisateurs et des statistiques.

Ce projet s’inscrit dans une démarche technologique moderne en répondant aux besoins des passionnés et professionnels de la technologie.

**Analyse et Conception :**

**1. Analyse des Besoins**

Pour garantir une expérience utilisateur optimale, nous avons identifié les besoins suivants :

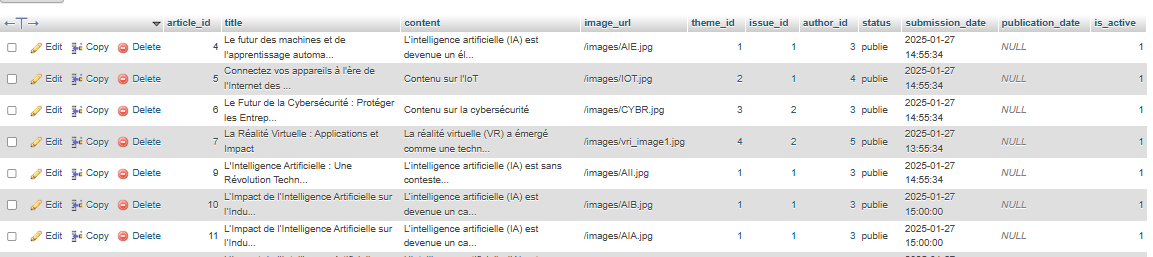
* Fonctionnalités adaptées à chaque type d’utilisateur.
* Gestion dynamique des abonnements, articles, statistiques et interactions (chat, commentaires).
* Interface responsive pour un accès fluide sur divers appareils.

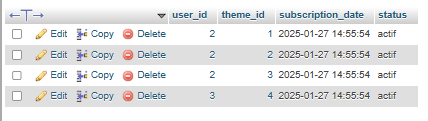
**2. Conception de la Base de Données**

La base de données **tech\_horizon** est conçue pour gérer un magazine en ligne, avec plusieurs fonctionnalités telles que la gestion des utilisateurs, des articles, des thèmes et des statistiques de vue. Elle comprend plusieurs tables interconnectées, permettant de stocker et d'organiser efficacement les informations relatives aux articles, aux utilisateurs, aux thèmes, et aux interactions des utilisateurs.

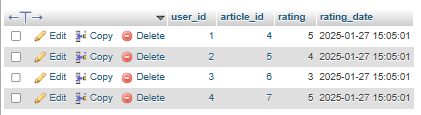
* **Table users**
* **Description** : Cette table contient les informations des utilisateurs, y compris leur identifiant, nom d'utilisateur, mot de passe, email, et rôle. Les utilisateurs peuvent avoir différents rôles tels que "invite", "abonne", "responsable" et "editeur".
* **Colonnes** :
  + user\_id: Identifiant unique de l'utilisateur (clé primaire).
  + username: Nom d'utilisateur unique.
  + email: Adresse email de l'utilisateur.
  + role: Rôle de l'utilisateur dans l'application (invite, abonne, responsable, editeur).
  + status: Statut de l'utilisateur (actif, bloque, en attente).
  + created\_at: Date de création du compte.
  + last\_login: Date de dernière connexion.



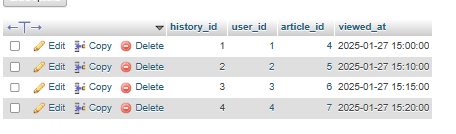
* **Table articles**
* **Description** : La table articles contient les informations relatives aux articles publiés sur le magazine en ligne. Chaque article peut avoir une image, un thème et une rubrique associée, et peut être dans différents états de publication.
* **Colonnes** :
  + article\_id: Identifiant unique de l'article (clé primaire).
  + title: Titre de l'article.
  + content: Contenu de l'article.
  + image\_url: URL de l'image associée à l'article.
  + theme\_id: Référence au thème de l'article (clé étrangère vers themes).
  + issue\_id: Référence au numéro du magazine auquel l'article appartient (clé étrangère vers issues).
  + author\_id: Référence à l'auteur de l'article (clé étrangère vers users).
  + status: Statut de l'article (propose, en\_cours, retenu, publie, refuse).
  + submission\_date: Date de soumission de l'article.
  + publication\_date: Date de publication de l'article.
* **Table themes**
* **Description** : La table themes contient les différents thèmes sous lesquels les articles sont classés. Chaque thème peut être géré par un responsable.
* **Colonnes** :
  + theme\_id: Identifiant unique du thème (clé primaire).
  + name: Nom du thème.
  + description: Description du thème.
  + responsable\_id: Référence au responsable du thème (clé étrangère vers users).
* **Table themesubscriptions**
* **Description** : Cette table gère les abonnements des utilisateurs aux différents thèmes. Elle permet de suivre l'abonnement, le statut et la date d'abonnement des utilisateurs pour chaque thème.
* **Colonnes** :
  + user\_id: Référence à l'utilisateur (clé étrangère vers users).
  + theme\_id: Référence au thème (clé étrangère vers themes).
  + subscription\_date: Date de l'abonnement.
  + status: Statut de l'abonnement (actif, expire, en\_attente).

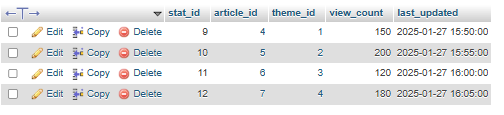


* **Table articleratings**
* **Description** : Cette table enregistre les évaluations des utilisateurs pour les articles. Les utilisateurs peuvent évaluer un article sur une échelle de 1 à 5.
* **Colonnes** :
  + user\_id: Référence à l'utilisateur ayant évalué l'article (clé étrangère vers users).
  + article\_id: Référence à l'article évalué (clé étrangère vers articles).
  + rating: Note de l'article.
  + rating\_date: Date de l'évaluation.



* **Table browsinghistory**
* **Description** : Cette table enregistre l'historique de navigation des utilisateurs, associant un utilisateur à un article qu'il a consulté.
* **Colonnes** :
  + history\_id: Identifiant unique de l'historique (clé primaire).
  + user\_id: Référence à l'utilisateur (clé étrangère vers users).
  + article\_id: Référence à l'article consulté (clé étrangère vers articles).
  + viewed\_at: Date et heure de la consultation de l'article.



* **Table viewstatistics**
* **Description** : Cette table stocke les statistiques de vues des articles et des thèmes. Elle permet de suivre le nombre de vues pour chaque article et thème.
* **Colonnes** :
  + stat\_id: Identifiant unique de la statistique (clé primaire).
  + article\_id: Référence à l'article (clé étrangère vers articles).
  + theme\_id: Référence au thème (clé étrangère vers themes).
  + view\_count: Nombre de vues de l'article ou du thème.
  + last\_updated: Date de la dernière mise à jour de la statistique.
* **Table articleimages**
* **Description** : La table articleimages est utilisée pour gérer les images associées aux articles. Un article peut avoir plusieurs images.
* **Colonnes** :
  + image\_id: Identifiant unique de l'image (clé primaire).
  + article\_id: Référence à l'article associé (clé étrangère vers articles).
  + image\_url: URL de l'image.
  + alt\_text: Texte alternatif pour l'image (utile pour l'accessibilité).
  + uploaded\_at: Date et heure de l'upload de l'image.
* **Table messages**
* **Description** : Cette table gère les messages échangés entre les utilisateurs pour les discussions autour des articles. Chaque message est lié à une conversation spécifique.
* **Colonnes** :
  + message\_id: Identifiant unique du message (clé primaire).
  + conversation\_id: Référence à la conversation (clé étrangère vers conversations).
  + sender\_id: Référence à l'expéditeur du message (clé étrangère vers users).
  + content: Contenu du message.
  + sent\_at: Date et heure d'envoi du message.

**Développement**

Le développement vise à concrétiser les fonctionnalités définies lors de l'analyse et la conception, en s'appuyant sur les technologies suivantes :

* **Backend** : PHP avec le framework Laravel pour une structure MVC (Model-View-Controller) organisée et efficace.
* **Base de données** : MySQL pour la gestion des données.
* **Frontend** : HTML et CSS pour la structure et le style, avec JavaScript pour l’interactivité.

**1. Développement Backend (Laravel)**

Laravel est utilisé pour organiser le backend et faciliter l’implémentation des fonctionnalités principales.

**Fonctionnalités Backend :**

* **Gestion des utilisateurs** :
  + Implémentation d'un système d'authentification avec Laravel (via php artisan make:auth ou Laravel Breeze/Fortify pour un système sécurisé).
  + Gestion des rôles :
    - **Invité** : Peut visualiser des numéros publics, mais sans accès personnalisé.
    - **Abonné** : Peut gérer ses abonnements, consulter l'historique et interagir avec les articles.
    - **Responsable de thème** : Peut approuver, refuser ou modifier les articles liés à son thème.
    - **Éditeur** : Supervise tous les articles, utilisateurs et statistiques globales.
* **Gestion des abonnements aux thèmes** :
  + Créer une relation entre les utilisateurs et les thèmes grâce à des tables pivot dans Laravel (belongsToMany).
  + Ajouter des fonctionnalités pour :
    - Souscrire à un thème.
    - Voir les thèmes auxquels un utilisateur est abonné.
    - Annuler un abonnement.
* **Gestion des articles** :
  + Ajout d’une fonctionnalité pour :
    - Proposer un article (avec une interface simplifiée pour les abonnés).
    - Mettre à jour le statut des propositions par les responsables ou les éditeurs (En cours, Accepté, Refusé).
    - Publier les articles validés avec une date de publication.
* **Historique de navigation** :
  + Implémenter un suivi automatique des articles consultés par chaque utilisateur.
  + Créer une API pour extraire ces données et générer des recommandations basées sur les consultations fréquentes.
* **Système de notation et commentaires** :
  + Permettre aux utilisateurs de noter les articles avec une note de 1 à 5 étoiles.
  + Ajout d’une section commentaires sous chaque article pour encourager l'interaction.
* **Statistiques et reporting** :
  + Générer des rapports pour les éditeurs :
    - Nombre de consultations par article ou par thème.
    - Nombre d’abonnements à chaque thème.

**Pourquoi Laravel ?**

Laravel simplifie la gestion des routes, des migrations pour la base de données et fournit des outils comme Eloquent ORM pour interagir avec MySQL de manière efficace.

**2. Développement Frontend (HTML, CSS, JavaScript)**

**Fonctionnalités Frontend :**

* **Page d'accueil** :
  + Présentation des thèmes disponibles sous forme de carte (exemple : "Intelligence artificielle", "Cybersécurité").
  + Affichage des numéros publics (avec des aperçus des articles récents).
  + Menu de navigation avec des liens vers les fonctionnalités principales (Connexion, Thèmes, Historique, etc.).
* **Espace abonné** :
  + Création d’un tableau de bord personnalisé :
    - Liste des thèmes auxquels l’utilisateur est abonné.
    - Historique des articles consultés.
    - Articles proposés par l'utilisateur et leur statut (suivi).
  + Bouton pour proposer un nouvel article avec un formulaire interactif.
* **Espace responsable de thème** :
  + Interface pour voir les articles proposés pour validation.
  + Tableaux dynamiques avec des options "Accepter" ou "Refuser".
  + Filtres pour trier les propositions par statut ou date.
* **Espace éditeur** :
  + Gestion des utilisateurs, des articles et des statistiques.
  + Tableaux interactifs pour surveiller les performances des thèmes et articles.
  + Widgets pour afficher des métriques clés (nombre total d’abonnés, articles en attente de validation, etc.).
* **Interactions et interactivité** :
  + **JavaScript** : Utilisation pour les actions dynamiques
  + **CSS** : Mise en place d’un design moderne et responsive, compatible avec les mobiles et tablettes.

**3. Développement Base de Données (MySQL)**

**Étapes principales :**

1. **Création des tables** :
   * Création de tables comme décrit dans la conception (ex. : users, themes, articles, etc.).
   * Utilisation des migrations Laravel pour versionner la structure de la base.
2. **Relations entre tables** :
   * Implémentation des relations One-to-Many et Many-to-Many pour les abonnements, propositions d’articles et historique.
3. **Optimisation des requêtes** :
   * Indexation des champs critiques (ex. : email, id\_theme) pour améliorer les performances.
   * Utilisation des transactions pour les actions sensibles (exemple : abonnement ou publication d'articles).

**Test et Validation**

**1. Tests Fonctionnels**

Chaque fonctionnalité a été testée séparément pour garantir son bon fonctionnement.

* **Proposition d’articles** : On s’assure que le statut d’un article change correctement (par exemple, de "En cours" à "Publié" ou "Refusé") après validation par un responsable ou un éditeur.
* **Abonnement aux thèmes** : Les tests vérifient que l’utilisateur peut s’abonner ou se désabonner d’un thème sans erreurs, avec des mises à jour instantanées dans la base de données.
* **Historique de navigation** : Chaque consultation d’article est correctement enregistrée et affichée dans l’interface utilisateur.

Ces tests permettent de corriger rapidement les bugs liés aux fonctionnalités principales.

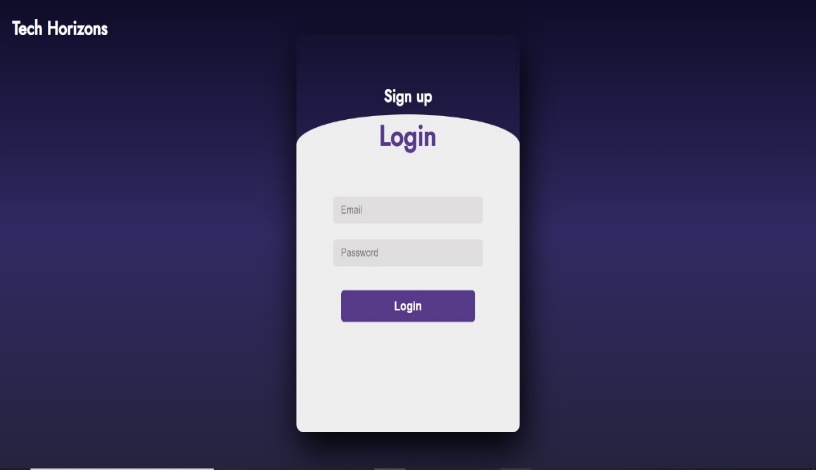
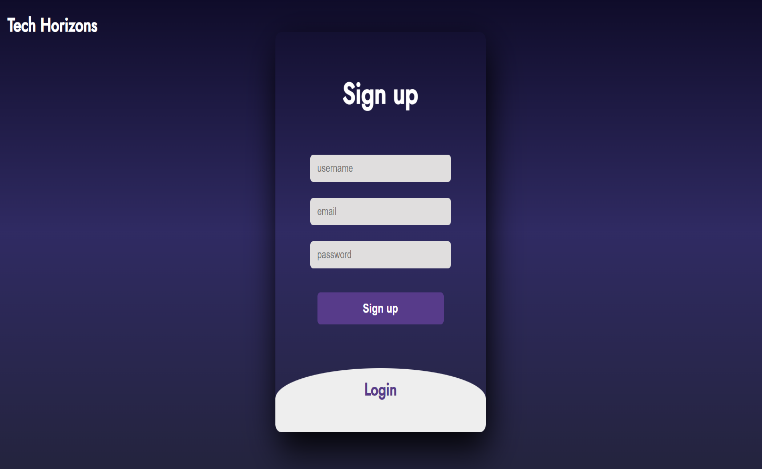
**2. Tests de Sécurité**

Pour protéger les données des utilisateurs et éviter les vulnérabilités :

* **Attaques SQL** : Les entrées utilisateur ont été testées contre des injections SQL grâce aux requêtes préparées, garantissant la sécurité des données dans la base.
* **Accès contrôlés** : Chaque utilisateur accède uniquement aux fonctionnalités autorisées selon son rôle. Les tentatives non autorisées sont bloquées.

Avec ces tests, l’application est prête à offrir une expérience fiable, sécurisée et conviviale.

**Présentation des Résultats Visuels**

1. Page de Connexion et Inscription

**Description :**  
 Cette page permet aux utilisateurs de se connecter ou de créer un compte. Elle inclut des validations sécurisées pour empêcher les erreurs et protéger les données utilisateur.

2. Tableau de Bord des Abonnés

**Description :**  
 Les abonnés peuvent accéder à un espace dédié où ils trouvent des contenus adaptés à leurs préférences, gèrent leurs abonnements, et suivent leurs lectures passées.

3. Proposition d’Articles par les Abonnés

**Description :**  
 Les abonnés peuvent proposer des articles directement via cette interface. Chaque proposition est enregistrée avec un statut visible (En attente, Refusé, ou Publié).

4. Gestion des Thèmes par le Responsable

**Description :**  
 Les responsables peuvent consulter toutes les propositions liées à leurs thèmes, valider celles qui sont conformes ou refuser celles qui ne respectent pas les normes.

5. Tableau de Bord pour l'Éditeur

**Description :**  
 L'éditeur dispose d'une interface avancée pour superviser l’ensemble des activités de l’application et gérer les utilisateurs et contenus.

6. Page Invité : Accès Public

**Description :**  
 Les invités peuvent explorer les thèmes et lire un aperçu des articles publics, mais doivent s’abonner pour accéder à du contenu premium.

**Difficultés Rencontrées**

**Problèmes techniques durant le développement**

1. Connexion à la base de données :

Lors de la configuration de la connexion à la base de données dans Laravel, plusieurs erreurs de connexion sont survenues à cause de paramètres mal configurés dans le fichier .env. Les erreurs les plus courantes étaient liées au nom de la base de données, au mot de passe, et au port. Ces erreurs ont retardé le processus de mise en place de l'application, car la connexion correcte à la base de données est cruciale pour récupérer et afficher les données dans les vues. Afin de résoudre ces problèmes, il a fallu ajuster les paramètres de la base de données dans le fichier .env et vérifier la configuration du serveur MySQL, en s'assurant que les informations de connexion étaient exactes et compatibles avec l'environnement de développement.

1. Problème d'affichage des vues :

Un autre défi majeur a été l'affichage correct des données dans les vues Laravel. Certaines informations ne s'affichaient pas comme prévu. Ce problème était principalement causé par des erreurs dans la syntaxe des vues Blade, où certaines variables n'étaient pas correctement passées ou affichées. Par exemple, certaines variables étaient mal nommées ou les données n'étaient pas correctement envoyées du contrôleur vers la vue. De plus, des erreurs liées aux routes et à la redirection entre les pages ont compliqué l'affichage des informations, notamment lors de la navigation entre les différentes pages de l'application.

1. Redirection basée sur le rôle de l'utilisateur :

Après que l'utilisateur se connecte avec son nom d'utilisateur et son mot de passe, il était nécessaire de rediriger l'utilisateur vers une page spécifique en fonction de son rôle. Par exemple, un utilisateur avec un rôle d'Admin devait être redirigé vers une page d'administration, tandis qu'un utilisateur normal devait être redirigé vers sa page de profil. Ce processus a été complexe à mettre en place, car Laravel utilise par défaut une redirection vers une seule page après la connexion (généralement le tableau de bord ou la page d'accueil). La gestion dynamique de la redirection selon le rôle nécessitait de modifier les contrôleurs et de définir des conditions spécifiques pour gérer correctement la redirection en fonction des rôles des utilisateurs.

**Solutions Apportées**

Connexion à la base de données : Correction des paramètres dans le fichier .env et redémarrage du serveur pour appliquer les changements.

Affichage des vues : Vérification des contrôleurs et correction des erreurs de syntaxe dans les fichiers Blade pour un bon affichage des données.

Redirection selon le rôle : Modification du contrôleur LoginController pour rediriger les utilisateurs vers la page appropriée en fonction de leur rôle, et utilisation de middleware pour restreindre l'accès aux pages.

**Conclusion :**

Projet Tech Horizons permet de créer une plateforme interactive et dynamique pour un magazine en ligne. Il offre aux utilisateurs la possibilité de consulter des articles, gérer leurs abonnements, proposer du contenu, et interagir avec les autres membres. Grâce aux outils d'analyse et de suivi, comme l'historique de navigation et les statistiques, le contenu peut être ajusté en fonction des préférences des abonnés, rendant l'expérience plus personnalisée.

En somme, ce projet répond bien aux attentes et offre de nombreuses opportunités pour évoluer et améliorer l'interaction avec les utilisateurs à l'avenir.