Tech Horizons

Rapport : Mini projet devloppement web

Licence IDAI 2024/2025

UNIVERSITE ABDELMALEK ESSAADI

FACULTE DES SCIENCES ET TECHNIQUES DE TANGER

➢ ENCADRÉ PAR :

Pr. M'hamed AIT KBIR

Pr. Yasyn EL YUSUFI

➢ PRÉPARÉ PAR :

Nouhaila Ayad

Hajar En-najib

Khaoula El bakkali

Lina El barrouk

TABLE DES MATIÈRES

INTRODUCTION

Dans le contexte actuel de transformation numérique, les applications web occupent une place essentielle dans la gestion de contenu, la diffusion d'informations et l'interaction avec les utilisateurs. **Tech Horizons**, un magazine en ligne focalisé sur les innovations technologiques et leurs impacts sur la société, nécessite une plateforme robuste et performante pour administrer ses contenus et offrir une expérience personnalisée à ses utilisateurs. Ce projet vise à répondre à ces enjeux en développant une application web moderne, intuitive et sécurisée. Grâce à l'utilisation du framework **Laravel**, couplé avec une base de données **MySQL**, cette plateforme permettra à divers types d'utilisateurs (**invités, abonnés, responsables de thèmes et éditeurs**) de consulter, gérer et interagir avec les articles du magazine. Les fonctionnalités clés du projet incluent la gestion des abonnements, la soumission d'articles, un système de recommandations basé sur les centres d'intérêt des utilisateurs, ainsi que des outils de modération et d'analyse des performances. Cette application aspire à offrir une expérience utilisateur fluide, tout en garantissant une gestion optimale des contenus et des abonnements.

Objectifs du projet :

Le projet a pour objectif de créer une plateforme interactive et sécurisée permettant aux utilisateurs de consulter et gérer leurs abonnements à divers thèmes technologiques et d'interagir avec les articles. Les fonctionnalités principales du projet incluent :

- ✓ ***Gestion des abonnements :*** Permet aux utilisateurs de s'abonner à différents thèmes technologiques et de gérer leurs préférences.
- ✓ ***Proposition d'articles :*** Permet aux abonnés de soumettre des articles à publier, avec un suivi de leur état (refusé, en cours, retenu, ou publié).
- ✓ ***Système de recommandations :*** Offre des recommandations d'articles basées sur l'historique de navigation et les intérêts des utilisateurs.
- ✓ ***Gestion des articles :*** Permet aux responsables de thèmes et aux éditeurs de gérer les articles, les abonnés, et les numéros du magazine.
- ✓ ***Modération et statistiques :*** Fournit des outils de modération des conversations et des statistiques sur les abonnés, les articles et les numéros du magazine.

Analyse et conception

Le projet a été conçu en tenant compte des besoins spécifiques des utilisateurs et de la nécessité d'offrir une expérience fluide et personnalisée. Voici les points clés de l'analyse et de la conception :

1. Identification des utilisateurs : Quatre types d'utilisateurs ont été définis, chacun ayant des rôles et des permissions distincts :

- ✓ ***Invité :*** Accède aux informations sur les thèmes et peut faire une demande d'inscription.
- ✓ ***Abonné :*** Accède à un espace personnalisé pour gérer ses abonnements, son historique de navigation, proposer des articles et interagir via des conversations.
- ✓ ***Responsable d'un thème :*** Gère les abonnements et les articles de son thème, consulte les propositions d'articles et peut modérer les conversations.
- ✓ ***Éditeur :*** A une gestion complète des numéros, des articles, des abonnés, et des responsables de thèmes, avec des outils d'analyse et de gestion des statistiques.

2. Architecture de l'application : Le choix de l'architecture MVC (Modèle-Vue-Contrôleur) a permis de séparer clairement la gestion des données, la logique métier et l'interface utilisateur, favorisant ainsi une gestion souple et efficace du projet. Ce découpage facilite la maintenance et l'extension de l'application, tout en assurant une scalabilité optimale.

- ➢ ***Les Modèles***

Les modèles représentent les entités de la base de données. Ils permettent de récupérer, insérer, mettre à jour et supprimer des données. Dans ce projet, les modèles ont été créés pour gérer les utilisateurs, les articles, les abonnements et les historiques. Laravel facilite cette gestion grâce à Eloquent, son ORM (Object-Relational Mapping), qui permet d'interagir avec la base de données de manière élégante et efficace.Prmi les on trouve :

Admin : Le modèle Admin gère les informations et l'authentification des administrateurs dans l'application Laravel. Il permet de stocker des données comme le prénom, le nom, l'email et le mot de passe des administrateurs tout en protégeant les informations sensibles. Ce modèle utilise des traits pour gérer les tokens API, les notifications et la création de données factices. Il établit également une relation many-to-many avec le modèle Theme, permettant aux administrateurs de s'abonner à plusieurs thèmes via une table pivot.

Article : Le modèle Article gère les articles publiés dans l'application. Il permet de stocker et d'associer des informations sur l'article telles que le titre, le contenu, l'image, l'ID de l'auteur, l'ID du thème, le lien de partage, le nombre de commentaires et de notes. Il établit des relations avec les modèles Theme, User (créateur de l'article), Comment et Rating. Le modèle offre également des méthodes pour calculer la note moyenne d'un article et vérifier si un utilisateur a déjà noté l'article.

ArticleRecommendation : Le modèle ArticleRecommendation gère les recommandations d'articles envoyées aux utilisateurs. Il enregistre l'utilisateur, l'article recommandé, ainsi que l'état de la notification (si l'utilisateur a été notifié ou non et à quel moment). Le modèle établit des relations avec les modèles User et Article pour lier les utilisateurs et les articles aux recommandations, tout en gérant les informations liées à la notification.

ArticleHistory :Le modèle ArticleHistory gère l'historique des interactions des utilisateurs avec les articles. Il enregistre les informations sur un utilisateur, l'article concerné et le statut de l'interaction (par exemple, "lu", "favori", etc.). Le modèle établit des relations avec les modèles User et Article pour associer un utilisateur et un article à un enregistrement d'historique spécifique.

Comment : Le modèle Comment gère les commentaires des utilisateurs dans l'application, permettant des relations polymorphes avec différentes entités (comme les articles, produits, etc.) via la méthode commentable. Il établit une relation avec le modèle User pour associer chaque commentaire à un utilisateur. Le modèle permet également de récupérer les commentaires associés à un autre commentaire grâce à une méthode de type "morphMany", créant ainsi une hiérarchie de réponses. En résumé, il permet de gérer les commentaires, leur association avec différentes entités et les utilisateurs qui les publient.

Post : Le modèle Post gère les publications dans l'application, permettant de stocker des informations comme le titre, le contenu et l'image associée à chaque publication. Il établit une relation avec le modèle User pour associer chaque publication à l'utilisateur qui l'a créée, via la méthode belongsTo. Le modèle utilise également le trait HasFactory pour permettre la génération de données factices. En résumé, il permet de gérer les publications et leur association avec l'utilisateur créateur, facilitant ainsi la gestion des posts dans l'application..

SavedArticle : Le modèle SavedArticle gère les articles sauvegardés par les utilisateurs. Il permet de stocker l'ID de l'utilisateur et l'ID de l'article sauvegardé. Le modèle établit des relations avec les modèles User et Article pour lier chaque enregistrement à un utilisateur et à un article spécifique. En résumé, ce modèle permet de gérer la fonctionnalité de sauvegarde d'articles, associant les utilisateurs aux articles qu'ils ont sauvegardés dans l'application.

Theme : Le modèle Theme gère les thèmes, enregistrant des informations comme le nom, l'icône, le responsable, et les statistiques liées aux abonnés et aux articles. Il établit des relations avec les articles, les utilisateurs abonnés, et les administrateurs abonnés via des tables pivot. Il inclut également une méthode pour obtenir le nombre d'articles associés à chaque thème.

User : Le modèle User gère les utilisateurs dans l'application, avec des informations comme le nom, l'email, le mot de passe, la photo de profil et la description. Il utilise des traits pour gérer l'authentification via tokens API, les notifications, et la création de données factices. Le modèle définit également une relation many-to-many avec le modèle Theme, permettant aux utilisateurs de s'abonner à plusieurs thèmes. En résumé, il permet de gérer les utilisateurs, leur profil et leurs abonnements aux thèmes dans l'application.

NewCommentPosted : définit une notification Laravel appelée NewCommentPosted qui est déclenchée lorsqu'un utilisateur poste un commentaire sur un article. Il stocke des informations comme le titre et l'identifiant de l'article, le nom de l'utilisateur ayant commenté, ainsi que l'identifiant du commentaire. La notification est enregistrée dans la base de données et distingue si le commentaire est une réponse à un autre ou un nouveau commentaire.

Rating : Le modèle Rating gère les évaluations des utilisateurs sur les articles. Il permet de stocker l'ID de l'utilisateur, l'ID de l'article et la note attribuée. En définissant les champs user_id, article_id et rating comme étant remplissables, ce modèle facilite l'ajout d'évaluations. Le modèle utilise également le trait HasFactory pour générer des données factices. En résumé, il permet de gérer les évaluations des utilisateurs, en enregistrant la note attribuée à chaque article par chaque utilisateur.

Follower.php : définit un modèle Follower dans une application Laravel. Ce modèle représente une relation de suivi entre utilisateurs, où un utilisateur (le "follower") suit un autre utilisateur (le "créateur"). Le modèle utilise les fonctionnalités d'Eloquent pour gérer les relations entre les tables de la base de données. Les champs creator_id et follower_id sont définis comme remplissables (fillable) pour permettre leur assignation de masse. Le modèle inclut deux relations belongsTo : une pour lier le "créateur" à l'utilisateur, et une autre pour lier le "follower" à l'utilisateur via la clé étrangère follower_id. Ce modèle facilite la gestion des relations de suivi dans l'application.

- ***Les Vues***

Les vues sont la partie de l'application qui s'occupe de l'affichage des données. Elles sont créées à l'aide de Laravel Blade, un moteur de templates. Les vues dans ce projet sont utilisées pour afficher les pages du site, telles que la liste des articles, le tableau de bord de l'utilisateur, et les formulaires d'inscription/connexion.

Blade home.blade.php : est le template de la page d'accueil de ton application Laravel.

son rôle :

- ✓ Il étend le layout welcome.blade.php.
- ✓ Il définit le titre de la page comme "**Home**".
- ✓ Il affiche une **section héro** avec un slogan, une image et une vidéo d'arrière-plan.
- ✓ Il propose un **bouton d'inscription** menant à `signUp.html`.
- ✓ Il affiche une **section "Explore Our Items"** avec un carrousel Swiper contenant différentes catégories d'articles technologiques (IA, cybersécurité, développement web, jeux vidéo).
- ✓ Il intègre **Swiper.js** pour un carrousel interactif avec pagination et navigation.

choisSingUp.blade.php: Cette vue présente une page d'accueil où l'utilisateur est invité à choisir son rôle. Elle affiche un titre "Welcome" et une question "What do you want to become ?". Ensuite, deux liens sont proposés : un pour s'inscrire en tant qu'utilisateur et un autre pour s'inscrire en tant qu'administrateur.

choixlogin.blade.php : Cette vue présente une page d'accueil où l'utilisateur est accueilli avec un titre "Welcome" et une question "How are you ?". Elle propose deux liens : un pour se connecter en tant qu'utilisateur et un autre pour se connecter en tant qu'administrateur.

Dashboard.blade.php pour l'abonnees : représente lun tableau de bord Laravel pour gérer les articles, commentaires et thèmes. Voici les points clés :

- ✓ **Navigation Admin :** Accès aux thèmes, abonnements, et articles.
- ✓ **Modal d'Article :** Formulaire pour ajouter des articles avec titre, contenu, image, et thème.
- ✓ **Articles et Commentaires :** Liste d'articles avec options pour commenter et répondre avec la suppression des commentaires possible.
- ✓ **Gestion des Thèmes :** Modal pour ajouter un thème avec nom et icônet et bouton flottant pour ajouter des thèmes.
- ✓ **JavaScript :** Gestion AJAX pour abonner ou désabonner aux thèmes avec retour visuel.

Login.blade.php : permet aux responsables de se connecter avec leur email et mot de passe. Il contient deux champs (email et mot de passe), chacun avec une icône, et un bouton de soumission. En bas du formulaire, il y a un lien pour s'inscrire si l'utilisateur n'a pas encore de compte.

mesthemes.blade.php : présente une page où un utilisateur peut voir ses thèmes créés, avec la possibilité d'ajouter un nouveau thème via un modal. Si l'utilisateur n'a pas de thèmes, un message l'indique. Chaque thème est affiché avec son nom, le nombre d'articles et d'abonnés, ainsi qu'une icône. Un bouton flottant permet d'ouvrir un formulaire pour ajouter un thème, où l'utilisateur doit

fournir un nom et une icône pour le thème. Le style de la page inclut une grille responsive pour afficher les thèmes et un effet de survol pour chaque carte de thème.

register.blade.php : Le formulaire d'inscription est divisé en cinq étapes : prénom/nom, email, description, photo de profil, et mot de passe. L'utilisateur navigue entre les étapes avec des boutons "Next" et "Previous". Une prévisualisation de la photo est affichée avant la soumission. Un message de succès apparaît une fois l'inscription terminée.

theme-articles.blade.php : une vue Laravel qui affiche des détails sur un thème, y compris une liste d'articles associés. Si aucun article n'est trouvé, un bouton flottant permet d'ajouter un nouvel article. Chaque article est affiché avec son titre, son contenu, des statistiques (commentaires, évaluations), et la possibilité de revenir à la liste des thèmes. Un modal est inclus pour ajouter un nouvel article avec un formulaire pour saisir le titre, le contenu, et l'image.

history.blade.php : Cette vue affiche l'historique des articles consultés par l'utilisateur. Elle présente un titre "**History**" avec un bouton pour effacer l'historique, ainsi qu'une barre de recherche. Chaque élément de l'historique montre le titre de l'article, le thème associé, la date relative de consultation, et le statut de l'article. Si aucun historique n'est disponible, un message indique "No search history".

about.blade.php : définit la page **"À propos"** de l'application **Tech Horizons** en utilisant Blade, le moteur de templates de Laravel.

Contenu de la page :

- ✓ **Présentation de l'équipe** : Une équipe de 4 passionnés encadrés par 2 professeurs.
- ✓ **Mission :** Offrir une plateforme interactive pour explorer les innovations technologiques.
- ✓ **Histoire :** Un projet académique combinant développement web et analyse technologique.

history.blade.php : cette page **"Historique"** affiche les articles consultés récemment, avec une barre de recherche, des filtres par thème (AI, PROGRAMMATION, etc.) et des informations comme le titre, le thème et la date

Save.blade.php : Cette page **"My Saved Posts"** affiche les articles sauvegardés par l'utilisateur, incluant l'auteur, la date de sauvegarde, le contenu et une image (si disponible). Un bouton permet de retirer un article des sauvegardes.

navbar.blade.php : une barre de navigation pour un site web sous Laravel, intégrant plusieurs fonctionnalités :

- ✓ **Affichage du logo :** Le logo du site est affiché via une image stockée localement.
- ✓ **Barre de recherche :** Une barre de recherche est disponible avec une icône pour permettre aux utilisateurs de rechercher du contenu.
- ✓ **Liens de navigation :** Plusieurs liens sont disponibles : Home, About Us et Community (uniquement pour les utilisateurs connectés).
- ✓ **Un menu déroulant "Themes"** (visible seulement pour les invités) permet d'accéder à différentes catégories comme AI, Cyber Security, Web Development, etc.
- ✓ **Authentification :** Les invités voient des boutons Log in et Sign up.Les utilisateurs connectés ont accès à un menu de notifications et un menu de profil.
- ✓ **Système de notifications :** Un icône de cloche affiche le nombre de notifications non lues.Chaque notification affiche un message selon son type (commentaire, réponse, note d'évaluation).Possibilité de consulter et gérer ses notifications.
- ✓ **Menu du profil :** L'utilisateur connecté voit son nom et une icône de profil.Un menu déroulant permet d'accéder à Mon profil, Articles enregistrés, Historique et Déconnexion.

edit.blade.php : affiche une page permettant aux utilisateurs de modifier leur profil. Il inclut un formulaire pour mettre à jour le nom, l'email et la photo de profil, avec une mise en page soignée grâce à Tailwind CSS. Le formulaire utilise Laravel Blade pour la gestion des données et de la sécurité avec @csrf. Deux boutons permettent soit d'annuler, soit d'enregistrer les modifications.

profile.blade.php : représente une page de profil utilisateur dynamique et stylisée, construite avec Laravel et Tailwind CSS. La page affiche des informations personnelles (**nom, email, date d'inscription**), une photo de profil, des statistiques (**followers, following, articles publiés**), ainsi que des sections pour les thèmes suivis et les articles rédigés par l'utilisateur. Elle inclut des fonctionnalités interactives comme la modification et la suppression d'articles via des modales, ainsi qu'un système de défilement horizontal pour les thèmes. Le design est moderne, avec des animations et des transitions fluides, et la page est responsive pour s'adapter à différents appareils. Les boutons "**Modifier le profil**" et "**Supprimer le compte**" permettent à l'utilisateur de gérer son compte facilement.

\resources\views\admin\authdashboard.blade.php : Page de tableau de bord pour les administrateurs.

Contenu : Affiche une liste de thèmes, permet de naviguer entre les thèmes, de s'abonner/désabonner, et de publier des articles. Inclut également une section pour les commentaires et les évaluations des articles.

\resources\views\admin\authlogin.blade.php : Page de connexion pour les responsables.

Contenu : Formulaire de connexion avec champs pour l'email et le mot de passe, et un lien vers la page d'inscription.

\resources\views\admin\auth\mesthemes.blade.php : Page affichant les thèmes créés parle responsable.
Contenu : Liste des thèmes avec des informations telles que le nombre d'articles et d'abonnés. Permet également d'ajouter de nouveaux thèmes via une modale.

\resources\views\admin\auth\register.blade.php : Page d'inscription pour les responsables.

Contenu : Formulaire d'inscription en plusieurs étapes, incluant la saisie des informations personnelles, de l'email, de la description, de la photo de profil, et du mot de passe.

\resources\views\admin\auth\theme-articles.blade.php : Page affichant les articles d'un thème spécifique.
Contenu : Affiche les articles d'un thème, avec des informations sur l'auteur, la date de publication, et les statistiques des articles (commentaires et évaluations). Permet également d'ajouter de nouveaux articles via une modale.

- ***Les Contrôleurs***

Les contrôleurs dans Laravel gèrent la logique métier de l'application. Ils reçoivent les requêtes des utilisateurs et déterminent quelles actions doivent être entreprises en fonction des règles de l'application. Par exemple, dans ce projet, les contrôleurs ont été utilisés sont :

Controller : Le Controller de base est une classe mère qui étend BaseController et inclut les traits AuthorizesRequests, DispatchesJobs et ValidatesRequests. Ces traits permettent de gérer la gestion des autorisations, le dispatch des tâches en arrière-plan (jobs) et la validation des requêtes dans les contrôleurs de l'application.

ArticleController : Le contrôleur ArticleController gère la logique liée aux articles.

- ✓ **index** : Affiche les articles filtrés par thème, nouveautés ou abonnements.
- ✓ **store :** Crée un nouvel article avec image et enregistre l'historique.
- ✓ **showFromNotification :** Affiche un article lié à une notification et la marque comme lue.
- ✓ **rate :** Gère les notations des articles et envoie des notifications pour les nouvelles notes.
- ✓ **show :** Affiche un article spécifique et l'ajoute à l'historique.
- ✓ **update :** Met à jour le titre et le contenu d'un article.
- ✓ **destroy :** Supprime un article.

ArticleHistoryController : Le contrôleur ArticleHistoryController gère l'historique des articles vus par un utilisateur. Il permet d'afficher, filtrer, mettre à jour le statut des articles dans l'historique, ainsi que d'ajouter des articles à l'historique automatiquement. Il inclut également une fonctionnalité de recherche d'articles et de suppression de l'historique complet.

CommentController : Le CommentController gère la création, la suppression et la gestion des réponses aux commentaires sur les articles. Il valide les données des commentaires, notifie les utilisateurs concernés lorsqu'un commentaire ou une réponse est posté, et permet de supprimer un commentaire si l'utilisateur est autorisé. Les notifications sont envoyées aux créateurs de l'article et des commentaires.

SavedArticleController: ce contrôleur permet aux utilisateurs de sauvegarder, consulter et retirer des articles sauvegardés. Il gère la validation des actions de sauvegarde, l'affichage des articles sauvegardés et leur suppression.

ThemeController : ce contrôleur gère l'abonnement des utilisateurs et des administrateurs à des thèmes, l'ajout de nouveaux thèmes, ainsi que l'affichage des articles d'un thème spécifique. Il permet aux utilisateurs et administrateurs de s'abonner ou se désabonner d'un thème et d'ajouter de nouveaux thèmes tout en veillant à ce que les administrateurs accèdent uniquement aux thèmes qu'ils gèrent.

HomeController : gère les pages principales de l'application et retourne des vues spécifiques en fonction des sections demandées.

Rôles des méthodes :

- ✓ **home() :** Affiche la page d'accueil (home/home).
- ✓ **history() :** Affiche la page d'historique (home/history).
- ✓ **about() :** Affiche la page "À propos" (home/about).
- ✓ **dashboard() :** Affiche le tableau de bord utilisateur (home/dashboard).

LoginController : Le contrôleur logincontroller gère les actions liées à la connexion et à l'inscription des utilisateurs. Voici un résumé de ses méthodes principales :

- ✓ **login() :** Affiche la vue de connexion.
- ✓ **singUp() :** Affiche la vue d'inscription.
- ✓ **logOut() :** Déconnecte l'utilisateur et le redirige vers la page de connexion.
- ✓ **store(Request $request) :** Valide les données d'inscription (nom, email, photo de profil, mot de passe), enregistre l'utilisateur et stocke la photo de profil. (Note : le champ password semble contenir une erreur de validation (exists:themes,id au lieu de required|min:6, par exemple).)

Concernant les collecteurs admin :

AdminArticleController :

- ✓ Gère la liste des articles dans le tableau de bord administrateur.
- ✓ Permet de filtrer les articles par thème et nouveautés (7 derniers jours).
- ✓ Permet de créer et de supprimer des articles, avec une gestion des images associées.

AdminCommentController :

- ✓ Permet aux administrateurs de poster des commentaires sur les articles.
- ✓ Gère aussi les réponses aux commentaires existants (commentaires imbriqués).
- ✓ Notifie l'auteur de l'article ou du commentaire lorsque celui-ci reçoit un nouveau commentaire.
- ✓ Permet de supprimer des commentaires.

LoginController :

- ✓ Gère la connexion et la déconnexion des administrateurs.
- ✓ Permet aux administrateurs de se connecter via un formulaire et redirige vers le tableau de bord.
- ✓ Permet de déconnecter un administrateur et redirige vers la page de connexion.

RegisteredUserController :

- ✓ Permet l'inscription d'un nouvel administrateur.
- ✓ Valide les informations et crée un compte administrateur avec une photo de profil optionnelle.
- ✓ Après inscription, l'administrateur est automatiquement connecté et redirigé vers le tableau de bord.

ProfileController :

Le contrôleur ProfileController gère les fonctionnalités liées au profil utilisateur dans notre application Laravel. Il contient plusieurs méthodes :

- ✓ ***edit() :*** Affiche le formulaire de modification du profil en récupérant les informations de l'utilisateur connecté.
- ✓ ***update(Request $request) :*** Met à jour les informations du profil (nom et email) après validation des données et redirige avec un message de succès.

- ✓ ***destroy() :*** Supprime définitivement le compte de l'utilisateur connecté et redirige vers la page d'accueil.
- ✓ ***show() :*** Affiche le profil de l'utilisateur avec des détails tels que ses articles, le nombre de followers, les utilisateurs suivis, et les thèmes disponibles.

\app\Http\Controllers\Admin\Auth\LoginController.php: son rôle est de gérer l'authentification des responsables

→ Il gère l'affichage de la page de connexion pour les responsables, leurs authentification et la redirection vers la dashboard, leurs déconnection, et enfin l'affichage de la dashboard de du responsable avec la liste des thèmes et les articles

\app\Http\Controllers\Admin\Auth\RegisteredUserController.php : son rôle est de gérer des nouveaux responsables

→ Il gère l'affichage de la page d'inscription pour les responsables, et enregistre un nouveau responsable dans la base de données après la validation

\app\Http\Controllers\Admin\Auth\AdminArticleController.php : son rôle set de gérer les articles pour le responsable

→ Il gère la création de nouveau articles (avec un titre, un contenu, une image et un thème associer qui est enregistré dans la base de donnée dont l'image est stocké dans le dossier **public/articles ,** et enfin il gérer la suppression d'article accompagnes par sa suppression dans leurs base de donnée

Les routes :

Dans web.php : contient des route pour les abonnes et les inviter

- **Routes Publiques** : (pour l'inviter)
  - Pages d'accueil, "À propos", "Nos services", et choix d'inscription/connexion.
- **Routes Protégées** : (pour l'abonnees)
  - Tableau de bord (/dashboard).
  - Gestion des articles (création, notation).
  - Gestion des commentaires (ajout, réponse, suppression).
  - Articles sauvegardés (sauvegarde, affichage, suppression).
  - Historique des articles (affichage, recherche, suppression).
  - Thèmes (abonnement, création).
- **Autres Routes** :
  - Déconnexion (/logOut).
  - Affichage d'articles via notifications.

Dans authadmin.php :

- **Routes Publiques** :
    - Inscription et connexion des administrateurs.
- **Routes Protégées** :
    - Tableau de bord (/admin/dashboard).
    - Gestion des articles (création, suppression).
    - Gestion des commentaires (ajout, réponse, suppression).
    - Gestion des thèmes (affichage, abonnement, suppression d'abonnés).
    - Pages statiques (accueil, "À propos").
    - Déconnexion (/admin/logout).

3. Système de recommandation : Un algorithme de recommandation a été intégré pour suggérer des articles en fonction des intérêts des utilisateurs. Ce système s'appuie sur l'historique de navigation, le type d'abonnement et les interactions avec les articles précédemment consultés.

4. Interface utilisateur : L'interface a été pensée pour être simple, intuitive et facile à naviguer. Les pages sont épurées et les utilisateurs peuvent accéder rapidement à leurs fonctionnalités respectives. Le design est entièrement personnalisé pour garantir une expérience cohérente et sans dépendance à des frameworks externes.

5. Gestion de la base de données : La base de données a été soigneusement conçue pour refléter les entités du domaine (utilisateurs, articles, abonnements, thèmes, etc.) avec des relations bien définies. **MySQL** a été choisi pour sa fiabilité et sa performance. L'utilisation de migrations a permis une gestion simplifiée des changements de schéma.

Choix techniques

1. Framework Laravel

Laravel est un **framework PHP** open-source, reconnu pour sa simplicité, sa sécurité et sa flexibilité dans le développement d'applications web modernes. Il repose sur une architecture **MVC (Modèle-Vue-Contrôleur)**, qui facilite la séparation des responsabilités dans l'application. **Laravel** offre de nombreux outils et fonctionnalités intégrées, tels que la gestion des migrations de base de données, le système d'authentification sécurisé, les contrôleurs de routes, et bien d'autres, qui rendent le développement plus rapide et sécurisé.

Le choix de **Laravel** a été motivé par sa robustesse, sa sécurité et sa capacité à simplifier le développement d'applications Web modernes. Ce **framework** offre des fonctionnalités prêtes à l'emploi telles que l'authentification, la gestion des sessions et des **middleware**, ce qui nous a permis de concentrer nos efforts sur la logique métier spécifique.

Dans ce projet, **Laravel** a été utilisé pour la gestion des utilisateurs, des abonnements, des articles et des statistiques. Sa structure modulaire et ses fonctionnalités prêtes à l'emploi ont permis de respecter le cahier des charges du projet.

2. Base de données MySQL

La base de données **MySQL** a été choisie en raison de sa performance éprouvée et de sa facilité d'intégration avec **Laravel**. Les données sont structurées de manière relationnelle pour faciliter les opérations de lecture et d'écriture tout en maintenant une bonne cohérence.

3. Front-End personnalisé

Le développement du **front-end** a été réalisé avec **HTML5**, **CSS3** et JavaScript personnalisé, afin de garantir un design unique, léger et performant. Ce choix a permis de contrôler intégralement l'aspect visuel et les interactions sans dépendre de solutions externes.

Fonctionnalités développées :

Définition de la bibliothèque Fortify

Fortify est une bibliothèque **Laravel** dédiée à la gestion de l'authentification des utilisateurs. Elle permet de gérer des fonctionnalités sécurisées comme l'enregistrement, la connexion, la réinitialisation de mot de passe, la vérification par e-mail et la gestion des sessions. Contrairement à d'autres solutions d'authentification comme **Jetstream** ou **Breeze**, **Fortify** se concentre uniquement sur la logique d'authentification sans inclure d'interface utilisateur prête à l'emploi, ce qui permet une personnalisation complète de l'apparence et des fonctionnalités.

Dans ce projet, **Fortify** a été utilisée pour gérer l'inscription et la connexion des utilisateurs, garantissant ainsi une sécurité optimale tout en restant flexible et personnalisable en fonction des besoins spécifiques du site.

Gestion des abonnements

Les utilisateurs peuvent s'abonner à plusieurs thèmes en fonction de leurs intérêts, et gérer ces abonnements via leur tableau de bord personnalisé. Ils peuvent également gérer leur historique de navigation et retrouver facilement des articles précédemment consultés grâce à des filtres de recherche.

La gestion des commentaire(chat)

La gestion des commentaires dans l'application permet aux utilisateurs d'interagir avec les articles en laissant des retours. Chaque commentaire est associé à un utilisateur et à un article via des relations de type "**appartient à**". Le **CommentController** prend en charge l'ajout des commentaires, en validant d'abord le contenu et l'article concerné. Il vérifie ensuite que l'utilisateur est authentifié avant d'enregistrer le commentaire avec les **IDs** correspondants. Enfin, une réponse **JSON** est renvoyée, contenant les détails du commentaire ou un message d'erreur en cas de problème.

La gestion des articles sauvegardés

La gestion des articles sauvegardés permet aux utilisateurs d'enregistrer et de retrouver facilement leurs articles. Le **SavedArticleController** gère l'ajout, l'affichage et la suppression des sauvegardes, en vérifiant qu'un article n'est pas déjà enregistré avant de l'ajouter. Le modèle **SavedArticle** établit les relations avec User et Article pour lier chaque sauvegarde à son propriétaire.

La gestion des notifications

Les notifications concernent trois événements distincts : les recommandations d'articles, les nouveaux commentaires postés, et les nouvelles évaluations d'articles.

- ✓ ***ArticleRecommendation :*** Cette notification est envoyée lorsqu'un article est recommandé à un utilisateur. Elle inclut des informations sur l'article recommandé, telles que son ID, son titre, et le type de notification ("**recommendation**").

- ✓ ***NewCommentPosted :*** Cette notification est envoyée lorsqu'un nouveau commentaire est posté sur un article. Elle contient des informations sur l'article, le commentaire, et l'utilisateur ayant commenté. Le type de notification indique s'il s'agit d'un "commentaire" ou d'une "réponse" (en fonction du type de commentaire).

- ✓ ***NewRatingPosted :*** Cette notification est envoyée lorsqu'un utilisateur évalue un article. Elle contient des informations sur l'article, l'utilisateur qui a attribué la note, la valeur de la note, et le type de notification ("**rating**").

Ces notifications sont envoyées via la base de données et peuvent être utilisées pour informer les utilisateurs de diverses interactions avec les articles sur la plateforme

Gestion des notes et des étoiles

Le système de notation permet aux utilisateurs d'évaluer un article en attribuant une note sous forme d'étoiles, généralement de 1 à 5. Lorsqu'un utilisateur sélectionne une étoile, la note

correspondante est enregistrée et peut être utilisée pour afficher une moyenne des avis ou personnaliser les recommandations d'articles. L'interface met en évidence la note sélectionnée, offrant ainsi une expérience utilisateur intuitive.

Affichage et mise à jour des évaluations

Une fois qu'une note est donnée, le système met à jour l'affichage des étoiles en temps réel, permettant aux autres utilisateurs de voir la moyenne des évaluations. Il est possible d'afficher un compteur indiquant le nombre total de votes reçus, ce qui renforce la crédibilité de l'évaluation. Certaines implémentations permettent aussi à l'utilisateur de modifier sa note après l'avoir attribuée.

Gestion du partage d'articles

Les utilisateurs ont la possibilité de partager un article via différents canaux tels que les réseaux sociaux, le courrier électronique ou un lien de partage direct. Un bouton dédié permet de générer un lien unique vers l'article concerné. Cette fonctionnalité améliore la visibilité du contenu et encourage l'interaction avec la plateforme.

Suivi et analyse des partages

Afin de mesurer l'engagement des utilisateurs, le système peut suivre le nombre de fois qu'un article a été partagé. Ces données peuvent être utilisées pour recommander des articles populaires ou proposer des statistiques aux administrateurs du site. Une intégration avec des services externes comme Google Analytics peut être mise en place pour un suivi plus détaillé.

Proposition d'articles

Les abonnés peuvent proposer des articles en fonction des thèmes choisis. Chaque proposition est soumise à un processus de validation qui permet de suivre son statut en temps réel : refusée, en cours, retenue ou publiée. Un système de notifications informe les utilisateurs de l'avancement de leur proposition.

Système de recommandations

Le système de recommandations analyse l'historique de navigation et les abonnements de l'utilisateur pour lui suggérer des articles en lien avec ses centres d'intérêt. Ce mécanisme offre une expérience personnalisée et facilite la découverte de nouveaux contenus.

Gestion des articles et des numéros

Les responsables de thèmes et les éditeurs peuvent gérer les articles et les numéros du magazine. Ils ont également accès à des statistiques détaillées permettant d'évaluer les performances des articles et des abonnés.

Modération et statistiques

Les responsables de thèmes jouent un rôle de modérateurs, en supervisant les conversations liées aux articles. Les éditeurs disposent de rapports détaillés sur les abonnés, les thèmes, les articles et les numéros, ce qui leur permet de prendre des décisions éclairées.

Sécurité et performance

La sécurité a été une priorité tout au long du développement de l'application. Des mécanismes tels que l'authentification via Laravel Sanctum, le hashage des mots de passe avec Bcrypt, et des contrôles d'accès via des middleware ont été mis en place pour assurer la confidentialité et la protection des données.

Le code a été optimisé pour garantir une performance optimale, notamment par le biais de la mise en cache des requêtes fréquentes et de l'optimisation des bases de données.

Fonctionnalités de l'Interface de Gestion de Profil et des Articles

Cette interface permet à un utilisateur de gérer son profil et ses articles. Voici les fonctionnalités principales :

Affichage du profil : Photo de profil, nom, email et date d'inscription.

Statistiques : nombre de followers, following et articles publiés.

Gestion du profil :

- ✓ ***Modifier le profil :*** Redirige vers un formulaire de modification.
- ✓ ***Supprimer le compte :*** Supprime définitivement le compte après confirmation.

Thèmes suivis :

- ✓ Affiche les thèmes auxquels l'utilisateur est abonné avec des détails (nom, description, nombre d'articles et d'abonnés).
- ✓ Navigation horizontale pour parcourir les thèmes.

Gestion des articles :

- ✓ ***Afficher les articles :*** Liste des articles publiés par l'utilisateur avec titre, date et extrait du contenu.

- ✓ ***Lire la suite :*** Affiche le contenu complet de l'article.
- ✓ ***Modifier un article :*** Ouvre une modal pour modifier le titre et le contenu de l'article.
- ✓ ***Supprimer un article :*** Supprime un article après confirmation.

Interactions :

- ✓ ***Notifications :*** Marque les notifications comme lues lors de l'affichage d'un article lié.
- ✓ ***Modal d'édition :*** Formulaire pour mettre à jour les articles directement depuis l'interface.

Cette interface est reliée au contrôleur **ProfileControlle**r pour la gestion du profil et au contrôleur **ArticleController** pour les opérations sur les articles (**création, modification, suppression, notation**).

Le responsible : le responsable est generer par breeze

CONCLUSION

Tech Horizons représente non seulement une prouesse technique mais aussi une vision innovante de ce que peut être un magazine technologique moderne. Notre plateforme combine expertise technique, expérience utilisateur intuitive et innovation continue pour créer un écosystème unique dans le paysage médiatique technologique.

L'architecture robuste, les fonctionnalités avancées et l'attention portée à l'expérience utilisateur font de **Tech Horizons** une référence dans son domaine. Les perspectives d'évolution et la roadmap ambitieuse garantissent un développement continu et une adaptation constante aux besoins des utilisateurs.