Royaume du Maroc

Université Abdelmalek Essaadi

Faculté des Sciences et Techniques de Tanger

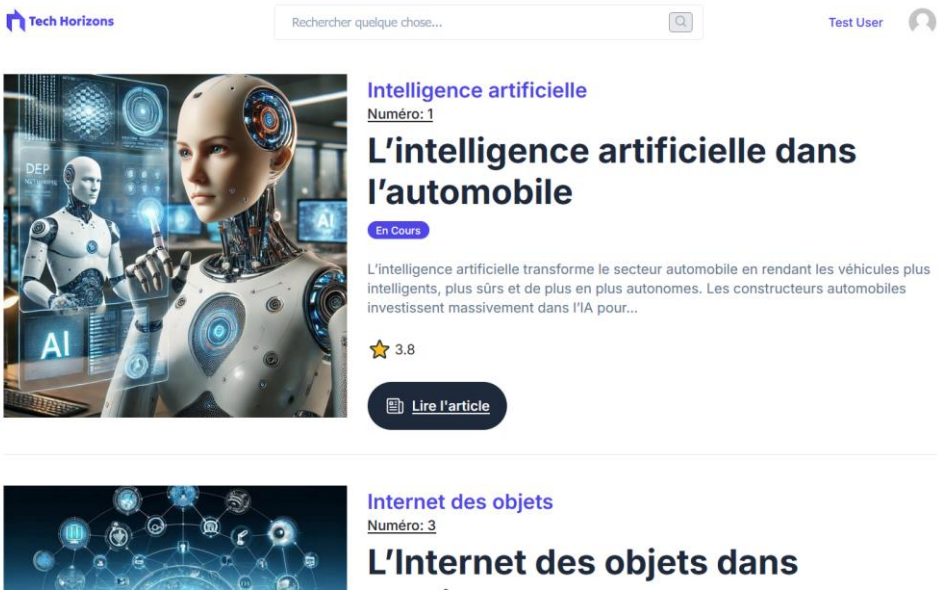
Département Informatique

Ingénierie de développement d'applications informatiques (IDAI)

Rapport de projet fin de module :

Tech Horizons

Laravel/MySQL



Encadré par :

Prof. M'hamed AIT KBIR

Prof. Yasyn EL YUSUFI

Réalisé par :

Taki eddine EL ATTARI

Fatima Ezzahra CHAYEB

Rabab ZITI

INTRODUCTION :

Tech Horizons est une plateforme numérique dédiée à un magazine en ligne explorant les innovations technologiques comme l'intelligence artificielle, l'Internet des objets, cybersécurité... Elle offre un espace interactif pour découvrir des articles, interagir avec des contenus personnalisés et contribuer à l'enrichissement du magazine, tout en mettant en lumière les enjeux éthiques et sociétaux de ces technologies.

L'application s'adresse à quatre profils d'utilisateurs : **invités**, **abonnés**, **responsables** de thème et **éditeurs**. Chacun dispose d'une interface intuitive pour des actions spécifiques, comme la consultation d'articles, la gestion d'abonnements, la proposition de publications ou la modération de conversations. Les abonnés bénéficient d'un espace personnalisé pour suivre leurs centres d'intérêt et participer à des discussions.

Tech Horizons se distingue par l'historique de navigation et les préférences des utilisateurs. Un suivi des propositions d'articles (avec statuts comme Refus ou Publié) et des outils de statistiques assurent une gestion efficace et transparente des contenus, répondant aux besoins d'un magazine dynamique.

Enfin, Tech Horizons combine innovation et simplicité. L'interface est facile à utiliser et les interactions sont fluides. La plateforme est conçue pour évoluer, permettant d'ajouter de nouvelles fonctionnalités tout en restant stable. Elle devient ainsi un outil pratique pour partager les connaissances technologiques avec une large communauté.

Tech Horizons

Le Sommaire :

INTRODUCTION :.....1

Conception : 4

1_Diagramme de classes :.....4

2_Diagrammes de séquences : 7

2.1_Invité :.....7

2.2_Abonné : 8

2.3_Responsable du thème :.....9

2.4_Editeur : 10

Configuration et Langage : 12

1_Configuration (.env) : 12

2_Langage : 12

Les modèles :.....13

1_Article :.....13

2_User :.....13

3_Theme : 13

4_Statut : 13

5_Numero :.....13

6_Historique : 14

7_Commentaire : 14

8_Abonnements :.....14

Les seeders et les migrations : 15

1_Les seeders : 15

1.1_ ArticleSeeder.php: 15

1.2_ DatabaseSeeder.php: 15

1.3_ NumeroSeeder.php: 15

1.4_StatutSeeder.php: 15

1.5_ThemeSeeder.php:.....15

2_Les migrations :.....16

Barre de navigation : 17

1_Vue :.....17

2_Contrôleur et route : 17

Page principale pour l'invité : 18

1_ Vue :.....18

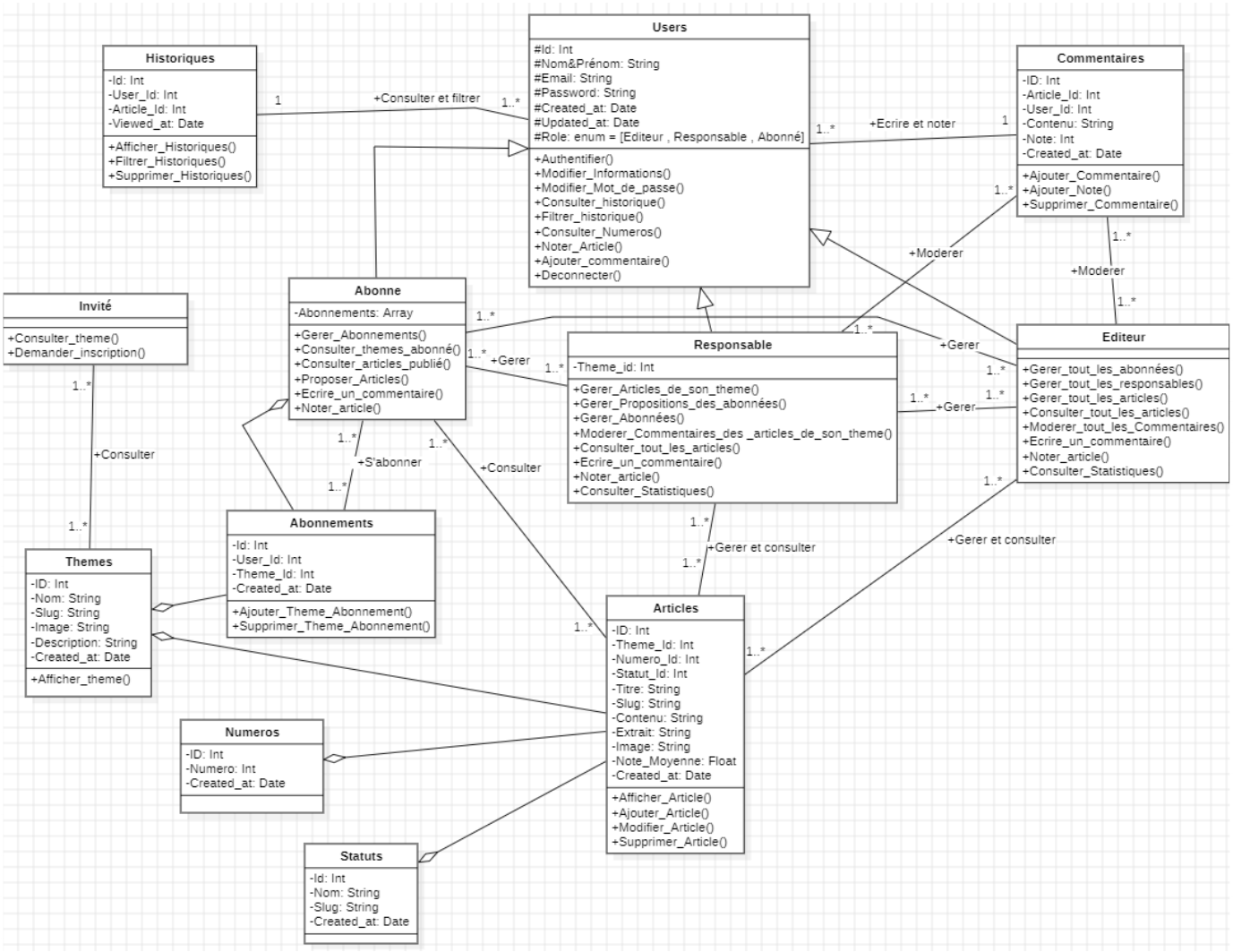
2_ Contrôleur et route : 18

Page de connexion : **19**
1_ Vue : 19
2_ Contrôleur et route : 19
Page d'inscription : **20**
1_ Vue : 20
2_ Contrôleur et route : 20
Page de modification des informations : **21**
1_ Vue: 21
2_ Contrôleur et route : 21
Page d'accueil pour authentifié : **22**
1_ Vue : 22
2_ Contrôleur et route : 22
Affichage de l'article : **23**
1_ Vue : 23
2_ Contrôleur et route : 23
Administration des articles : **24**
1_ Vue : 24
2_ Contrôleur et route : 24
Formulaire des articles : **25**
1_ Vue : 25
Administration des utilisateurs : **26**
1_ Vue: 26
2_ Contrôleur et route : 26
Page des statistiques : **27**
1_ Vue : 27
2_ Contrôleur et route : 27
Page des historiques : **28**
1_ Vue : 28
2_ Contrôleur et route : 28

Conception :

Dans le cadre de notre projet Tech_Horizons, nous avons conçu des diagrammes de classe et des diagrammes de séquence à l'aide de l'outil StarUML. Ces diagrammes visent à modéliser la structure statique et les interactions dynamiques du système, en mettant en évidence les classes, leurs relations, ainsi que les flux d'échanges entre les objets.

1_Diagramme de classes :



1. Users : Cette classe représente tous les utilisateurs de l'application, qu'ils soient abonnés, responsables de thèmes ou éditeurs. Elle stocke les informations essentielles telles que l'identifiant, le nom, l'email, le mot de passe et la date de création. Chaque utilisateur a un rôle spécifique qui lui permet d'accéder à certaines fonctionnalités, comme l'authentification, la

gestion de son historique, la notation et l'ajout de commentaires sur les articles.

2. Invité : Cette classe représente un utilisateur qui n'est pas encore inscrit sur la plateforme. Il peut consulter les informations sur les thèmes disponibles et soumettre une demande d'inscription afin d'obtenir un accès plus avancé aux fonctionnalités du site.

3. Abonné : Un abonné est un utilisateur ayant un compte actif sur la plateforme. Il peut gérer ses abonnements, consulter des articles, proposer des articles, noter et commenter.

4. Responsable : Cette classe représente un utilisateur ayant la responsabilité d'un thème spécifique du magazine. Il gère un thème spécifique, examine les articles proposés, modère les commentaires et consulte les statistiques de son thème.

5. Éditeur : L'éditeur est un utilisateur ayant un rôle administratif sur l'ensemble de la plateforme. Il gère tous les numéros, les abonnés, les responsables, les articles et les commentaires. Consulte les statistiques globales.

6. Articles : Représente les articles du magazine. Contient des attributs comme le titre, le contenu, l'extrait, une image et une note moyenne. Lié à un thème et un numéro.

7. Commentaires : Permet aux utilisateurs d'interagir avec les articles via des messages et des notes. Modéré par les responsables et les éditeurs.

8. Historiques : Enregistre les articles consultés par un utilisateur. Permet de retrouver et filtrer les articles déjà lus.

9. Thèmes : Catégorie d'articles (ex : IA, cybersécurité). Les utilisateurs peuvent s'abonner pour recevoir du contenu pertinent.

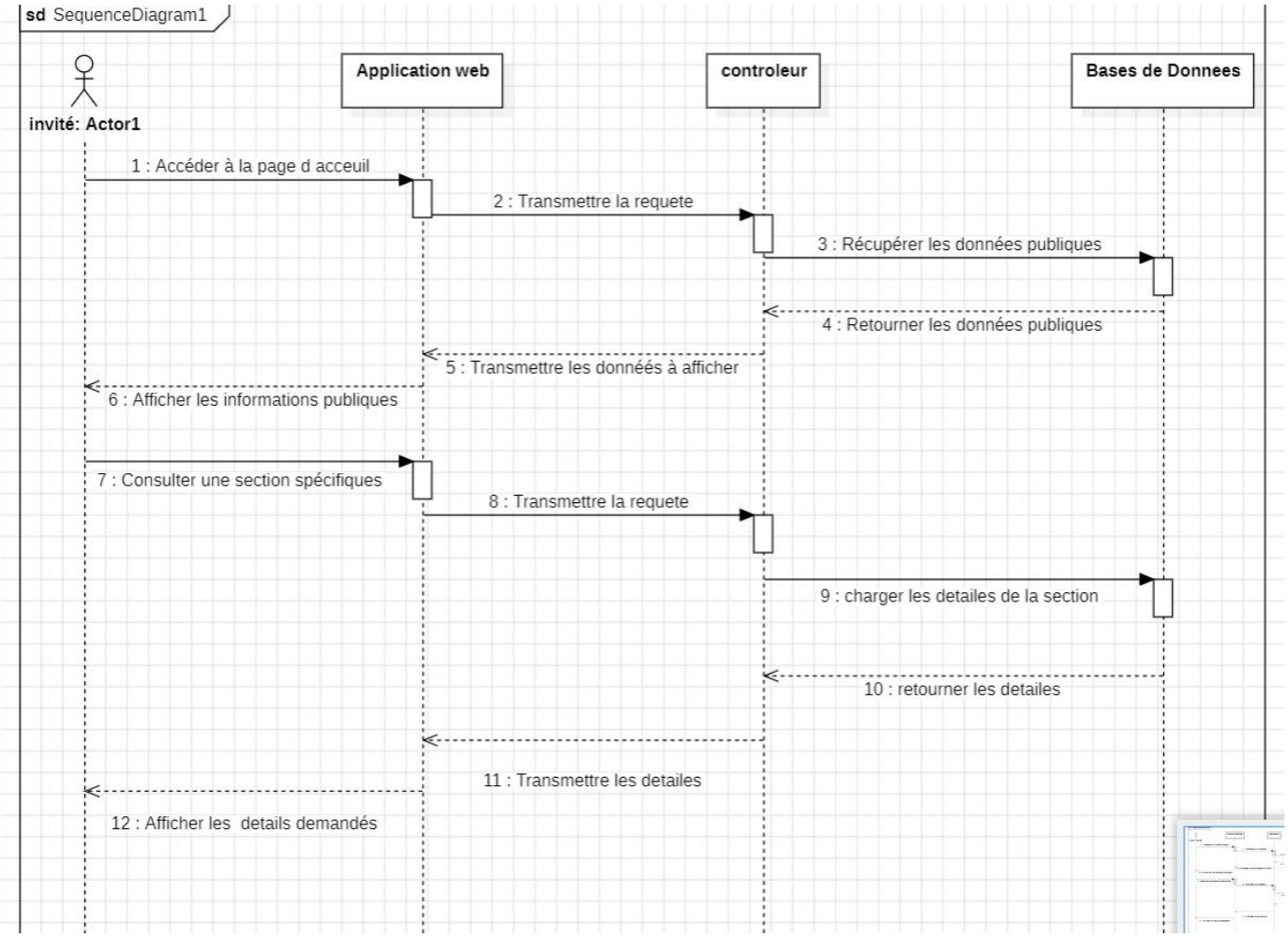
10. Abonnements : Gère l'association entre un utilisateur et un ou plusieurs thèmes. Permet de recevoir du contenu ciblé.

11. Numéros : Éditions périodiques du magazine. Regroupe des articles de différents thèmes. Possède un identifiant unique et une date de création.

12. Statuts : Définit l'état d'un article (en cours, publié, refusé). Suit le cycle de vie d'un article.

2_Diagrammes de séquences :

2.1_Invité :

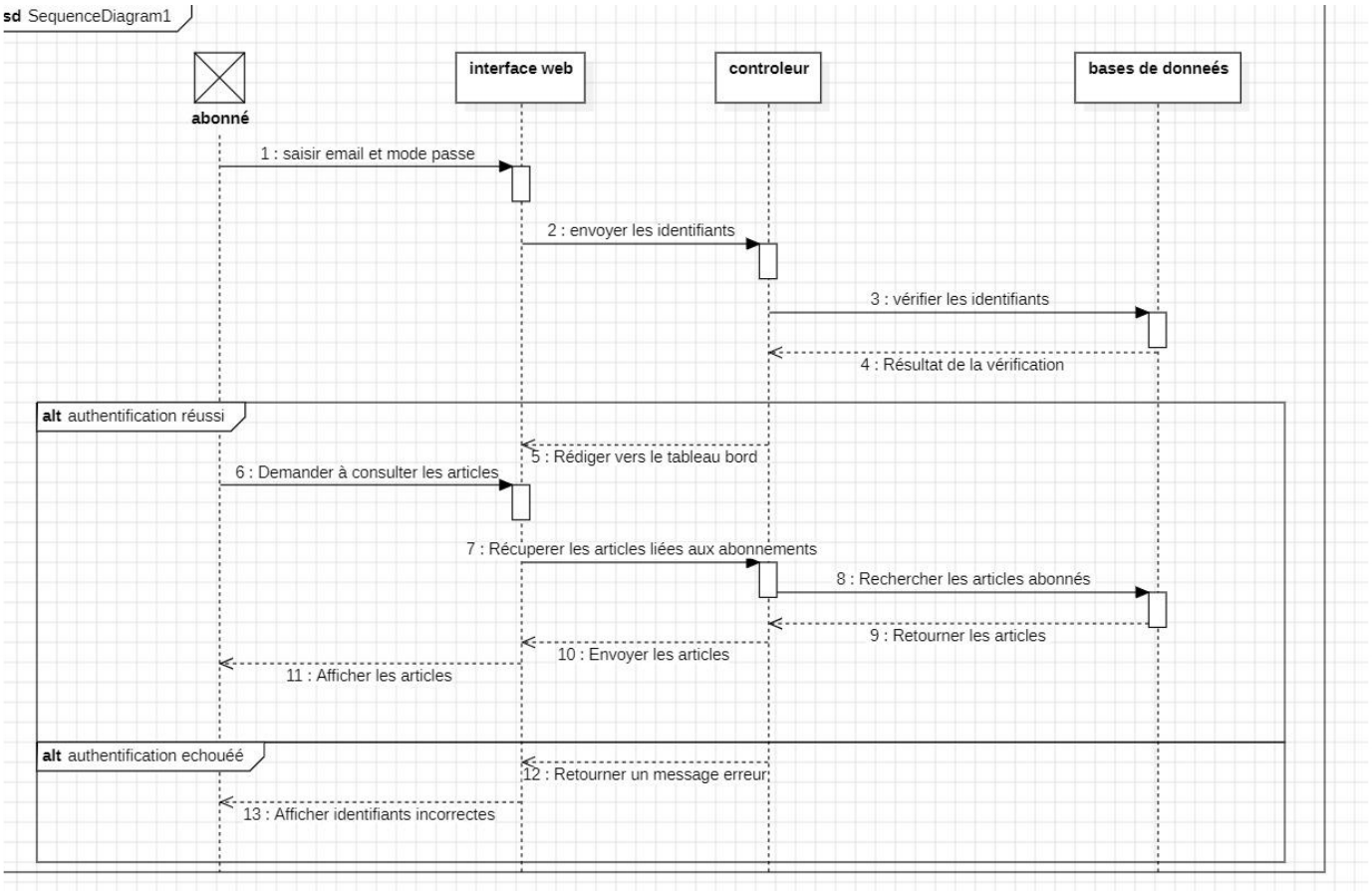


Ce diagramme de séquence représente l'interaction entre un utilisateur invité, une application web, un contrôleur et une base de données. Il illustre le processus permettant d'afficher des informations publiques et d'accéder aux détails d'une section spécifique.

Dans un premier temps, l'utilisateur accède à la page d'accueil de l'application web. Cette action déclenche l'envoi d'une requête au contrôleur, qui interroge ensuite la base de données pour récupérer les informations publiques. Une fois ces données obtenues, elles sont retournées au contrôleur, puis transmises à l'application web, qui les affiche à l'utilisateur.

Ensuite, si l'utilisateur souhaite consulter une section spécifique, il effectue une nouvelle requête via l'application web. Cette requête est transmise au contrôleur, qui sollicite la base de données pour charger les détails de la section demandée. Une fois ces informations récupérées, elles sont envoyées au contrôleur, puis à l'application web, qui les affiche à l'utilisateur.

2.2_Abonné :



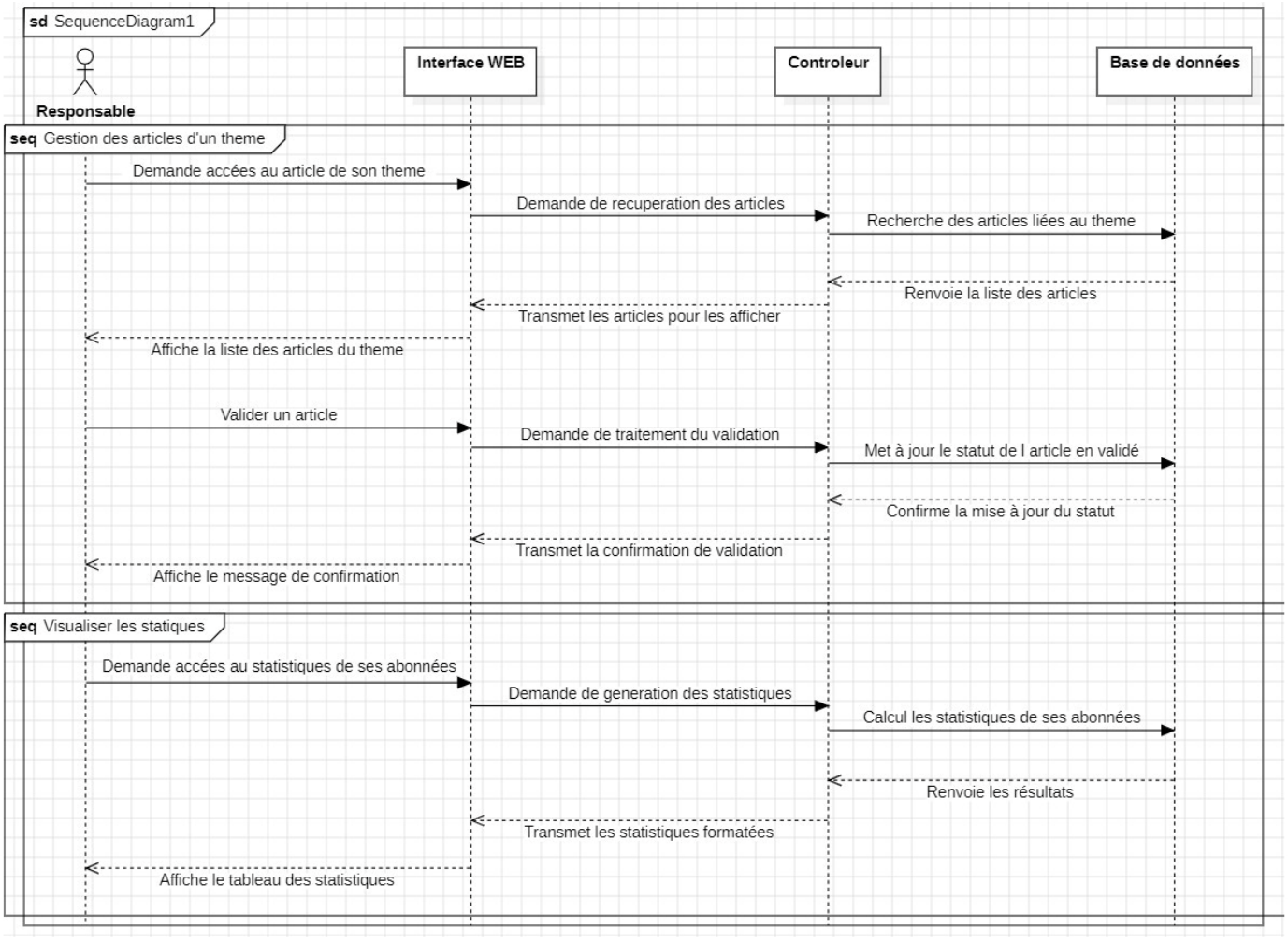
Ce diagramme de séquence représente le processus d'authentification d'un abonné et l'accès aux articles qui lui sont réservés.

Tout d'abord, l'abonné saisit son email et son mot de passe sur l'interface web. Ces informations sont ensuite transmises au contrôleur, qui les envoie à la base de données pour vérification. Une fois la vérification effectuée, la base de données retourne le résultat au contrôleur, qui le communique à l'interface web.

Si l'authentification réussit, l'utilisateur est redirigé vers son tableau de bord. Il peut alors demander à consulter les articles liés à son abonnement. L'interface web transmet cette demande au contrôleur, qui interroge la base de données afin de récupérer les articles abonnés. Les articles sont ensuite envoyés à l'interface web et affichés à l'utilisateur.

Si l'authentification échoue, un message d'erreur est retourné à l'interface web, qui informe l'utilisateur que ses identifiants sont incorrects.

2.3_Responsable du thème :

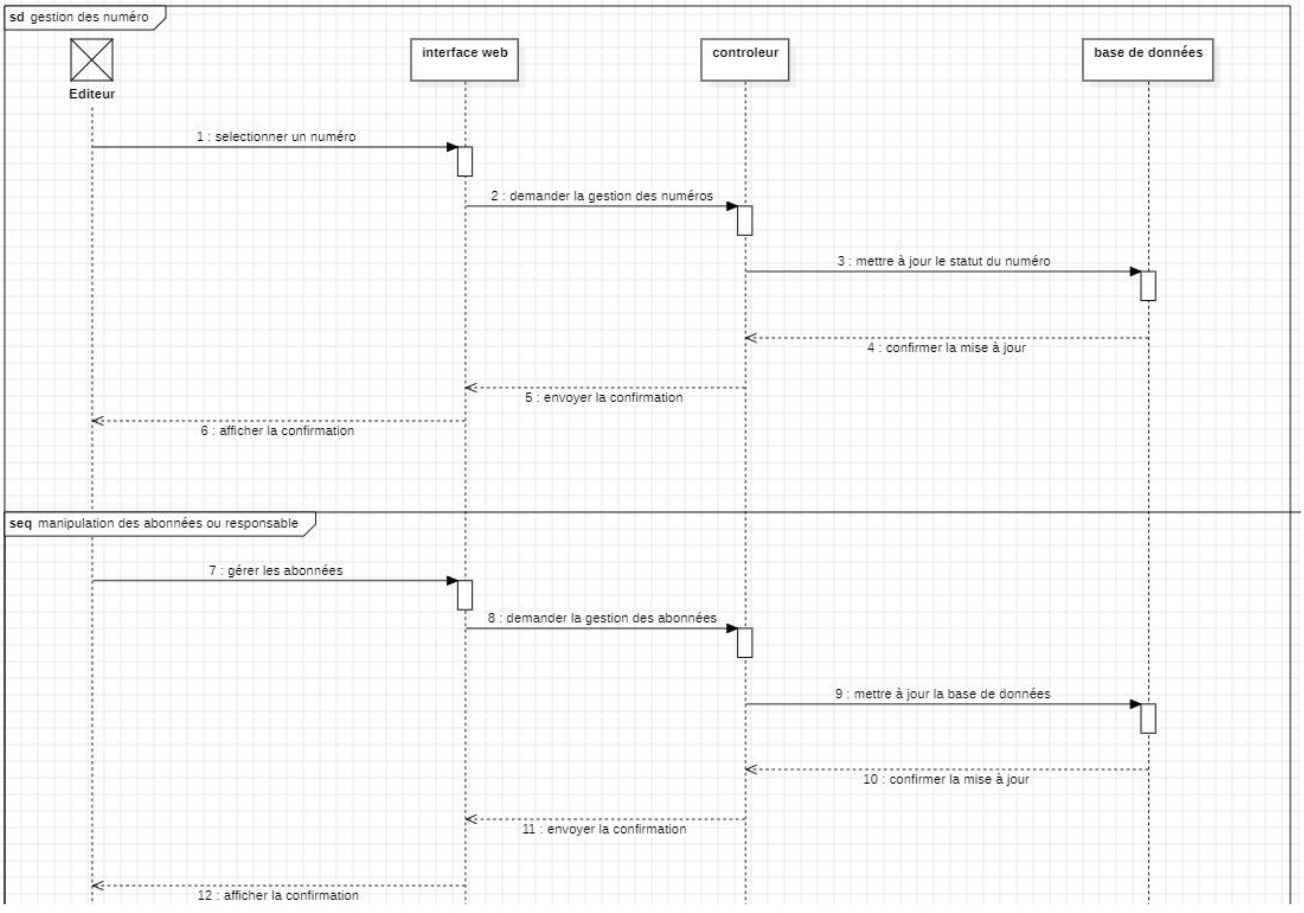


Ce diagramme de séquence présente les interactions entre un Responsable, une Interface Web, un Contrôleur, et une Base de Données pour la gestion des articles d'un thème ainsi que la visualisation des statistiques des abonnés.

Dans la première partie du diagramme, le Responsable demande l'accès aux articles liés à son thème. L'Interface Web transmet cette demande au Contrôleur, qui interroge la Base de Données pour rechercher les articles correspondants. Une fois la liste obtenue, les articles sont affichés à l'utilisateur. Le Responsable peut alors valider un article. Cette action déclenche une demande de traitement auprès du Contrôleur, qui met à jour le statut de l'article dans la Base de Données et confirme cette mise à jour. La confirmation est ensuite transmise à l'Interface Web, qui affiche un message de validation pour le Responsable.

La seconde partie du diagramme concerne la visualisation des statistiques des abonnés. Le Responsable demande à consulter ces statistiques. L'Interface Web transmet cette demande au Contrôleur, qui génère les statistiques en interrogeant la Base de Données. Une fois les calculs effectués, les résultats sont renvoyés au Contrôleur, qui transmet les statistiques formatées à l'Interface Web. Enfin, le Responsable peut visualiser les statistiques sous forme de tableau.

2.4_Editeur :



Ce diagramme de séquence illustre deux processus distincts : la gestion des numéros et la manipulation des abonnés ou responsables.

Dans la première partie, dédiée à la gestion des numéros, un éditeur sélectionne un numéro via l'interface web. Cette action déclenche une requête de gestion des numéros envoyée au contrôleur, qui met à jour le statut du numéro dans la base de données. Une fois la mise à jour effectuée, la base de données envoie une confirmation au contrôleur, qui la transmet ensuite à l'interface web. Cette dernière affiche alors une confirmation à l'éditeur, indiquant que l'opération a été réalisée avec succès.

Dans la seconde partie, consacrée à la manipulation des abonnés ou responsables, l'éditeur initie la gestion des abonnés via l'interface web. Cette dernière envoie une requête au contrôleur pour gérer ces abonnés. Le contrôleur met ensuite à jour les informations dans la base de données. Une fois la mise à jour effectuée, la base de données envoie une confirmation au contrôleur, qui la transmet à l'interface web. Enfin, l'interface web affiche une confirmation à l'éditeur pour signaler la réussite de l'opération.

Configuration et Langage :

1_Configuration (.env) :

Le fichier .env du site web a été personnalisé pour répondre aux besoins spécifiques du projet. Tech Horizons est configurée pour notre contexte marocain avec le fuseau horaire Africa/Casablanca et le français comme langue principale, l'anglais restant disponible comme langue alternative. La base de données, nommée "tech_horizons", est utilisée comme support centra pour le stockage des sessions, du cache et des files d'attente, optimisant ainsi la gestion des données.

2_Langage :

Le fichier lang/fr/validation.php a été entièrement traduit en français pour assurer une expérience utilisateur cohérente avec notre langue professionnelle. Cette adaptation linguistique couvre l'ensemble des messages de validation du système, depuis les règles de base jusqu'aux validations plus spécifiques.

Les modifications incluent la traduction des messages pour toutes les règles de validation courantes comme les champs requis, les validations d'email, les contraintes de longueur, et les vérifications de format. Une attention particulière a été portée à l'adaptation des attributs courants (noms des champs) en français, avec par exemple 'name' traduit en 'nom complet', 'email' en 'adresse e-mail', etc. Des règles personnalisées ont également été ajoutées, notamment pour la validation des images, avec des messages spécifiques en français concernant la sélection, le format et la taille des fichiers.

Les modèles :

Les modèles constituent la structure fondamentale du projet Tech Horizons, comprenant huit entités principales (User, Article, Theme, Commentaire, Historique, Numero, Statut et Abonnements) interconnectées par des relations Eloquent. Cette architecture permet une gestion fluide des fonctionnalités essentielles comme les abonnements, les interactions utilisateurs…

1_Article :

Le fichier Article.php est un modèle qui représente l'entité "Article" dans la base de données. Il étend la classe Model pour définir des relations avec d'autres modèles tels que Theme, Statut, Numero et Commentaire. Il inclut des méthodes pour filtrer les articles en fonction de critères de recherche, gérer les commentaires, et calculer la note moyenne d'un article basée sur les retours des utilisateurs.

2_User :

Le modèle User.php représente les utilisateurs du système avec différents rôles (Editeur, Responsable, Abonné) et gère leurs relations avec les thèmes et l'historique de navigation.

3_Theme :

Le modèle Theme.php catégorise les articles et gère les abonnements des utilisateurs, avec un système de slug pour le routage.

4_Statut :

Le modèle Statut.php définit les différents états possibles des articles, utilisant un slug pour le routage.

5_Numero :

Le modèle Numero.php représente simplement une numérotation d'articles avec une relation one-to-many vers les articles.

6_Historique :

Le modèle Historique.php trace l'historique de lecture des articles par les utilisateurs, incluant la date de consultation et des fonctionnalités de filtrage.

7_Commentaire :

Le modèle Commentaire.php permet aux utilisateurs de commenter et noter des articles, avec des relations vers l'utilisateur et l'article concerné.

8_Abonnements :

Le modèle Abonnements.php gère les souscriptions des utilisateurs aux différents thèmes, établissant une relation many-to-many entre les utilisateurs et les thèmes.

Les seeders et les migrations :

Le dossier 'database' dans notre projet joue un rôle central dans la gestion des données. Il regroupe les migrations pour structurer la base de données, les seeders pour peupler les tables avec des données initiales, et les factories pour générer des données de test. Ces éléments assurent une gestion cohérente et automatisée des données tout au long du développement.

1_Les seeders :

1.1_ ArticleSeeder.php:

Génère des articles de blog avec des données prédéfinies (titre, contenu, image, thème, statut, etc.), vérifie leur existence avant création, et ajoute des commentaires aléatoires associés à des utilisateurs. Les articles sont liés aux thèmes, statuts, et numéros.

1.2_ DatabaseSeeder.php:

Orchestre l'exécution des autres seeders dans un ordre spécifique : initialise les thèmes, numéros, statuts, crée des utilisateurs (dont un utilisateur de test), puis génère les articles via ArticleSeeder.

1.3_ NumeroSeeder.php:

Insère des numéros (1 à 5) dans la base de données s'ils n'existent pas déjà. Utile pour catégoriser ou versionner des éléments.

1.4_StatutSeeder.php:

Crée trois statuts (Refus, En Cours, Publié) avec des slugs associés. Permet de gérer le cycle de vie des articles (validation, publication).

1.5_ThemeSeeder.php:

Initialise des thèmes techniques (IA, IoT, Cybersécurité, etc.) avec des descriptions détaillées et des images. Les thèmes structurent la catégorisation des articles.

2_Les migrations :

Description générale des migrations utilisé dans le projet:

+ Authentification & Sécurité : Gestion des utilisateurs (users).

+ Contenu central : Articles (posts), commentaires (comments), et catégories (categories) structurés avec titres, contenus, et liaisons utilisateurs.

+ Relations : Table pivot post_category pour associer articles et catégories en many-to-many, et clés étrangères (user_id, post_id).

+ Optimisation : Champs slug indexés, suppression en cascade, et colonnes image optionnelles pour flexibilité.

Barre de navigation :

1_Vue :

La partie présentation de la barre de navigation est construite dans le fichier default.blade.php, qui utilise une structure HTML sémantique avec une balise <header> contenant le logo (<img class="logo">), la barre de recherche (<form class="search-form">) et un système de navigation adaptable. Les styles sont gérés dans index.css qui définit l'apparence via des classes comme .header, .logo, .search-form, et .nav-menu. Le comportement dynamique est implémenté dans index.js, notamment avec addEventListener pour gérer le menu déroulant mobile et la fonction toggleActive pour les interactions utilisateur.

2_Contrôleur et route :

Le routage est géré dans web.php à travers des routes définies avec le système de routage Laravel. La fonctionnalité de recherche est implémentée dans le modèle Article.php via la méthode scopeFilters, qui utilise des requêtes Eloquent avec des relations belongsTo et hasMany pour permettre une recherche sur plusieurs critères (titre, thème, numéro, statut, commentaires). Les résultats sont optimisés grâce au chargement anticipé défini dans la propriété $with du modèle.

Page principale pour l’invité :

1_ Vue :

Le fichier index.blade.php utilise une structure @forelse pour itérer sur les thèmes disponibles, avec une gestion élégante des cas où aucun résultat n'est trouvé via la directive @empty. Le composant theme.blade.php agit comme un template réutilisable qui structure chaque carte de thème. Il accepte les propriétés 'theme' et 'list' comme paramètres grâce à la directive @props, permettant une flexibilité dans l'affichage. Les styles sont gérés dans index.css qui définit une mise en page responsive.

2_ Contrôleur et route :

La logique côté serveur est gérée par InviteController.php qui implémente la méthode indexInvite(). Cette méthode utilise le modèle Theme avec une requête select() optimisée. Les données sont ensuite transmises à la vue via la fonction compact(), créant une liaison efficace entre le contrôleur et l'interface utilisateur.

Page de connexion :

1_ Vue :

L'interface de connexion est construite autour d'un layout réutilisable auth.blade.php qui sert de template principal pour l'authentification. Le composant connexion.blade.php étend ce layout en ajoutant les champs spécifiques à la connexion : email, mot de passe et une option "Rester connecté". Les styles définis dans auth.css créent une expérience visuelle moderne et cohérente.

2_ Contrôleur et route :

La logique d'authentification est gérée par ConnexionController qui implémente trois fonctionnalités principales : l'affichage du formulaire, le traitement de la connexion, et la déconnexion. Le contrôleur utilise le middleware 'guest' pour protéger les routes appropriées et empêcher les utilisateurs déjà connectés d'accéder au formulaire de connexion. La méthode connexion() valide les informations d'identification et gère la persistance de session via l'option "remember me".Les routes sont clairement définies dans web.php, avec des points d'entrée distincts pour l'affichage du formulaire (GET) et le traitement de la connexion (POST).

Page d’inscription :

1_ Vue :

L'interface d'inscription est construite autour d'un layout maître auth.blade.php servant de base aux pages d'authentification. La vue spécifique inscription.blade.php étend ce layout et définit un formulaire avec quatre champs : nom complet, adresse email, mot de passe et sa confirmation. Le design est géré par auth.css qui assure une présentation moderne avec une carte centrée sur fond clair, des champs de formulaire stylisés et des transitions fluides pour les interactions.

2_ Contrôleur et route :

La logique d'inscription est gérée par InscriptionController.php qui valide les données soumises, crée les nouveaux utilisateurs et gère leur première connexion. Les routes dans web.php définissent les points d'entrée /inscription pour afficher le formulaire (GET) et traiter l'inscription (POST). L'ensemble est protégé par le middleware 'guest' pour empêcher les utilisateurs déjà connectés d'accéder à l'inscription.

Page de modification des informations :

1_ Vue:

Cette page dédié pour les utilisateurs authentifiés est structurée dans Home/index.blade.php où il offre une interface utilisateur intuitive et moderne. La page présente trois sections principales : la gestion des abonnements (pour les abonnées), les informations personnelles, et la modification du mot de passe. Chaque section est encapsulée dans un formulaire distinct, avec des styles cohérents définis dans index.css. La section des abonnements permet aux utilisateurs de sélectionner plusieurs thèmes via un menu déroulant, tandis que les autres sections permettent de mettre à jour les informations personnelles et le mot de passe.

2_ Contrôleur et route :

La logique de la page d'accueil est gérée par HomeController, qui implémente plusieurs fonctionnalités clés pour les utilisateurs authentifiés. Le contrôleur utilise le middleware auth pour s'assurer que seuls les utilisateurs connectés peuvent accéder à cette page. Les méthodes updateProfile, updatePassword, et updatetheme gèrent respectivement la mise à jour des informations personnelles, du mot de passe, et des thèmes d'abonnement. Les routes correspondantes sont définies dans web.php, avec des points d'entrée distincts pour chaque fonctionnalité (GET pour l'affichage, PATCH pour les mises à jour).

Page d’accueil pour authentifié :

1_ Vue :

La page principale, implémentée dans Accueil/index.blade.php, offre une présentation moderne des articles sous forme de grille responsive. Chaque article est représenté par un composant article.blade.php qui affiche une image de couverture, le thème associé, le numéro, le titre, le statut, une description, et une note moyenne avec une icône d'étoile. La navigation est facilitée par une pagination élégante qui indique le nombre total d'articles et permet de naviguer entre les pages. Les styles définis dans index.css assurent une présentation cohérente et adaptative.

2_ Contrôleur et route :

La gestion des articles est orchestrée par ArticleController.php qui implémente plusieurs méthodes clés. La méthode index gère l'affichage principal avec un système de filtrage. Pour les utilisateurs abonnés, une logique spéciale dans articleView filtre les articles pour n'afficher que ceux correspondant aux thèmes abonnées et ayant le statut "Publié". Les routes dans web.php définissent les points d'accès, avec des routes spécifiques pour filtrer par thème (/themes/{theme:slug}), statut (/Statuts/{statut}), et numéro (/Numeros/{numero}).

Affichage de l’article :

1_ Vue :

La page de visualisation d'article show.blade.php utilise un layout par défaut qui encapsule élégamment le contenu de l'article via le composant <x-article>. Sous l'article, un système de notation avec 5 étoiles interactives et un formulaire de commentaire sont disponibles pour les utilisateurs authentifiés, suivis d'une section affichant tous les commentaires existants avec leurs notes, auteurs, horodatages et options de modération pour les utilisateurs autorisés (responsable du thème et éditeur). Une modale de confirmation stylisée s'affiche lors de la suppression d'un commentaire par index.js, et l'ensemble de l'interface s'adapte parfaitement aux différentes tailles d'écran grâce au fichier index.css.

2_ Contrôleur et route :

La logique de gestion est centralisée dans ArticleController.php, où la méthode show() gère l'affichage de l'article. Le contrôleur gère également les commentaires via la méthode commentaire() qui valide et enregistre les nouveaux commentaires avec leurs notes, et la méthode destroyComment() qui vérifie les autorisations (éditeur ou responsable du thème) avant de permettre la suppression. Les routes dans web.php sont configurées pour gérer l'affichage de l'article ('/{article}'), l'ajout de commentaires ('/articles/{article}/commentaire') et leur suppression ('/commentaires/{commentaire}'), avec une utilisation appropriée des middlewares d'authentification.

Administration des articles :

1_ Vue :

La page d'administration, implémentée dans index.blade.php, présente une interface épurée avec un en-tête affichant le titre "Posts" et un bouton "Créer un post". Le contenu principal est organisé sous forme de tableau responsive avec des colonnes pour le titre, le statut et les actions (voir, modifier, supprimer). Le style est géré par des classes dans index.css qui définissent une présentation professionnelle avec des couleurs distinctes. Un système modal de confirmation de suppression est implémenté avec une animation fluide et un fond semi-transparent dans index.js.

2_ Contrôleur et route :

Le CrudArticleController.php gère la logique d'administration avec une architecture CRUD complète. La méthode index() implémente une logique de filtrage basée sur les rôles : pour les responsables, seuls les articles de leur thème sont affichés. Les routes dans web.php sont protégées par le middleware 'auth' et suivent une convention de nommage cohérente ('/admin/articles') pour toutes les opérations CRUD.

Formulaire des articles :

1_ Vue :

Le formulaire de gestion d'articles form.blade.php propose une interface dynamique adaptée à la création ou modification d'articles. Il utilise des composants personnalisés comme <x-input>, <x-select>, et <x-textarea> pour structurer les champs (titre, slug, thème, numéro, statut, contenu, image). La sélection du thème est conditionnelle : les éditeurs/abonnés choisissent un thème via une liste, tandis que les responsables ont un champ caché lié à leur thème attribué. Les styles dans index.css assurent une mise en page cohérente, avec des transitions visuelles et une prévisualisation d'image.

2_ Contrôleur et route :

La logique est gérée par CrudArticleController.php. Les méthodes store() et update() utilisent ArticleRequest.php pour valider les données, puis sauvegardent ou modifient l'article via save(). Cette méthode gère l'upload d'images, définit automatiquement des valeurs comme l'extrait et attribue des statuts selon le rôle de l'utilisateur (publié pour les responsables/éditeurs et en cours pour les abonnées). Les routes dans web.php ('/admin/articles') activent les opérations CRUD, avec des middlewares d'authentification pour restreindre l'accès.

Administration des utilisateurs :

1_ Vue:

La page d'administration des utilisateurs index.blade.php affiche un tableau interactif listant les utilisateurs avec leurs informations clés (nom, rôle, thème associé, email). Un bouton "Créer un utilisateur" déclenche le formulaire d'ajout. Les lignes du tableau incluent des liens "Modifier" et "Supprimer", ce dernier activant une modale de confirmation via index.js. Les styles dans index.css assurent une UI cohérente.

2_ Contrôleur et route :

Le CrudUserController.php orchestre les opérations. La méthode index() filtre les utilisateurs selon le rôle de l'authentifié (le responsable peux gérer seulement les abonnées de son thème tandis que l'éditeur peux gérer tous les utilisateurs) . La méthode save() persiste les modifications. Les routes dans web.php sécurisées par 'auth' activent le CRUD via ('/admin/Utilisateurs').

Page des statistiques :

1_ Vue :

La page des statistiques admin/statistiques/index.blade.php organise les données en sections thématiques (articles, thèmes, numéros, utilisateurs, commentaires) avec une mise en page en grille adaptative. Chaque section utilise des cartes interactives affichant des indicateurs clés. Les styles dans statistiques.css assurent un design moderne et réactif, avec un affichage mobile optimisé.

2_ Contrôleur et route :

Le StatistiqueController.php centralise la collecte des données via Eloquent (Article::all(), User::all(), etc.). La méthode index() agrège les statistiques et les transmet à la vue via compact(). L'accès est restreint aux responsables et éditeurs grâce à un middleware vérifiant. La route /admin/statistiques dans web.php relie l'URL au contrôleur, garantissant une récupération sécurisée et efficace des données.

Page des historiques :

1_ Vue :

La page d'historique, implémentée dans index.blade.php et stylisée avec historiques.css, présente une interface moderne avec un système de filtrage complet en haut de la page. Le design inclut un formulaire de recherche flexible avec des entrées pour le texte, la sélection de thème et la date. Chaque entrée d'historique est affichée dans une carte interactive qui comprend le titre de l'article (avec un lien), le thème, la date de consultation et un bouton de suppression. Une modale de confirmation élégante apparaît lors de la suppression d'un élément, avec une animation fluide gérée par historiques.js.

2_ Contrôleur et route :

La gestion de l'historique fonctionne sur deux niveaux. Au niveau de HomeController.php, les méthodes history et destroyHistory gèrent l'affichage et la suppression des entrées d'historique. La création des entrées d'historique est gérée dans ArticleController.php à travers sa méthode show() qui enregistre automatiquement chaque consultation d'article par un utilisateur authentifié. Les routes dans web.php définissent un système complet avec trois points d'accès principaux : l'affichage, la suppression, et l'enregistrement de l'historique ('/historiques').