

DOSSIER PROFESSIONNEL^(DP)

<i>Nom de naissance</i>	▶ <i>USDI</i>
<i>Nom d'usage</i>	▶ <i>Aucun</i>
<i>Prénom</i>	▶ <i>Abdurahman</i>
<i>Adresse</i>	▶ <i>18 rue blavier, 51100 Reims</i>

Titre professionnel visé

Développeur Web et Web Mobile

MODALITE D'ACCES :

- ☐ Parcours de formation
- ☐ Validation des Acquis de l'Expérience (VAE)

DOSSIER PROFESSIONNEL (DP)

Présentation du dossier

Le dossier professionnel (DP) constitue un élément du système de validation du titre professionnel.

Ce titre est délivré par le Ministère chargé de l'emploi.

Le DP appartient au candidat. Il le conserve, l'actualise durant son parcours et le présente **obligatoirement à chaque session d'examen.**

Pour rédiger le DP, le candidat peut être aidé par un formateur ou par un accompagnateur VAE.

Il est consulté par le jury au moment de la session d'examen.

Pour prendre sa décision, le jury dispose :

1. des résultats de la mise en situation professionnelle complétés, éventuellement, du questionnaire professionnel ou de l'entretien professionnel ou de l'entretien technique ou du questionnement à partir de productions.
2. du **Dossier Professionnel** (DP) dans lequel le candidat a consigné les preuves de sa pratique professionnelle.
3. des résultats des évaluations passées en cours de formation lorsque le candidat évalué est issu d'un parcours de formation
4. de l'entretien final (dans le cadre de la session titre).

[Arrêté du 22 décembre 2015, relatif aux conditions de délivrance des titres professionnels du ministère chargé de l'Emploi]

Ce dossier comporte :

- ▶ pour chaque activité-type du titre visé, un à trois exemples de pratique professionnelle ;
- ▶ un tableau à renseigner si le candidat souhaite porter à la connaissance du jury la détention d'un titre, d'un diplôme, d'un certificat de qualification professionnelle (CQP) ou des attestations de formation ;
- ▶ une déclaration sur l'honneur à compléter et à signer ;
- ▶ des documents illustrant la pratique professionnelle du candidat (facultatif)
- ▶ des annexes, si nécessaire.

DOSSIER PROFESSIONNEL (DP)

Pour compléter ce dossier, le candidat dispose d'un site web en accès libre sur le site.

➤ <http://travail-emploi.gouv.fr/titres-professionnels>

Sommaire

Exemples de pratique professionnelle

Développer la partie front-end d'une application web ou web mobile sécurisée	p.	5
➤ <u>Tout pour un nouveau site application web responsive</u>	p.	5
1. <u>Installer et configurer son environnement de travail en fonction du projet web ou web mobile</u>	p.	5
2. <u>Maquetter des interfaces utilisateurs web ou web mobile</u>	p.	9
3. <u>Réaliser des interfaces utilisateurs web ou web mobile</u>	p.	13
4. <u>Développer la partie dynamique des interfaces utilisateurs web ou web mobile</u>	p.	16
➤ <u>E-learning Abdu application web responsive</u>	p.	26
1. <u>Installer et configurer son environnement de travail en fonction du projet web ou web mobile</u>	p.	26
2. <u>Maquetter des interfaces utilisateurs web ou web mobile</u>	p.	27
3. <u>Réaliser des interfaces utilisateurs web ou web mobile</u>	p.	28

DOSSIER PROFESSIONNEL ^(DP)

4. Développer la partie dynamique des interfaces utilisateurs web ou web mobile

p. 35

► Intitulé de l'exemple n° 3 : NIL

p. XXX

Développer la partie back-end d'une application web ou web mobile sécurisée

p. 41

► Tout pour une nouvelle application web back-end PHP

p. 41

1. Mettre en place une base de données relationnelle

p. 41

2. Développer des composants d'accès aux données SQL et NoSQL

p. 44

3. Développer des composants métier coté serveur

p. 48

4. Documenter le déploiement d'une application dynamique web ou web mobile

p. 53

► E-learning Abdu application web back-end PHP utilisation MVC

p. 59

1. Mettre en place une base de données relationnelle

p. 59

2. Développer des composants d'accès aux données SQL et NoSQL

p. 61

3. Développer des composants métier coté serveur

p. 66

4. Documenter le déploiement d'une application dynamique web ou web mobile

p. 70

► Intitulé de l'exemple n° 3 : NIL

p. XXX

Titres, diplômes, CQP, attestations de formation *(facultatif)*

p. 77

Déclaration sur l'honneur

p. 78

Documents illustrant la pratique professionnelle *(facultatif)*

p. 79

DOSSIER PROFESSIONNEL ^(DP)

Annexes *(Si le RC le prévoit)*

p.

80

EXEMPLES DE PRATIQUE PROFESSIONNELLE

Activité-type 1

Développer la partie front-end d'une application web ou web mobile sécurisée

Exemple n°1 ► Tout pour une nouvelle application web responsive pour les compétences du référentiel suivantes :

- Installer et configurer son environnement de travail en fonction du projet web ou web mobile
- Maquetter des interfaces utilisateur web ou web mobile
- Réaliser des interfaces utilisateur statiques web ou web mobile
- Développer la partie dynamique des interfaces utilisateur web ou web mobile

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

I. Installer et configurer son environnement de travail en fonction du projet web ou web mobile

Pour commencer le développement de mon projet, j'ai mis en place un environnement de travail local en utilisant plusieurs outils clés, notamment XAMPP, Git, Composer, MongoDB, et mon IDE préféré, Cursor, qui inclut une console intégrée facilitant l'utilisation de Git directement depuis l'éditeur de code.

1. Installation de XAMPP

Ma première étape a été d'installer XAMPP, un package qui inclut un serveur Apache, une base de données MySQL (ou MariaDB), PHP, et phpMyAdmin. XAMPP est particulièrement utile pour le développement en local car il regroupe tous les outils nécessaires au fonctionnement d'une application web complète.

- **Téléchargement** : J'ai téléchargé XAMPP depuis le site officiel d'Apache Friends. J'ai choisi la version correspondant à mon système d'exploitation (Windows, dans mon cas).
- **Installation** : Après avoir lancé le programme d'installation, j'ai suivi les instructions à l'écran. J'ai pris soin de sélectionner les composants essentiels, notamment Apache, MySQL, et PHP, car ils sont indispensables pour mon projet. L'installation s'est déroulée dans le répertoire par défaut C:\xampp.
- **Lancement de XAMPP** : Une fois installé, j'ai ouvert le panneau de contrôle XAMPP (xampp-control.exe). Ce panneau me permet de démarrer et d'arrêter les services Apache et MySQL en quelques clics.

A. Configuration de XAMPP

Pour m'assurer que mon serveur local fonctionnait correctement, j'ai démarré les modules Apache et MySQL à partir du panneau de contrôle XAMPP. Une fois ces services actifs, j'ai vérifié leur bon fonctionnement en accédant à <http://localhost> depuis mon navigateur. La page d'accueil de XAMPP s'est affichée, confirmant que le serveur était opérationnel.

B. Création du répertoire de projet

Mon projet étant bien défini, j'ai créé un dossier spécifique pour héberger tous les fichiers et ressources nécessaires. Ce dossier, nommé Toutpourunnouveaune, a été placé dans le répertoire htdocs de XAMPP (C:\xampp\htdocs). Le répertoire htdocs est l'emplacement où Apache recherche les fichiers pour les servir en tant que pages web.

Ce répertoire de projet contient les sous-dossiers suivants pour une organisation claire :

- **/src/public** : pour les fichiers accessibles publiquement comme les HTML, CSS, JS.
- **/src/** : pour les scripts PHP et la logique métier.
- **/config/** : pour les fichiers PHP réutilisables, comme les configurations de base de données.
- **/assets/** : pour les images, fichiers CSS et JavaScript personnalisés.

C. Utilisation de l'IDE Cursor et initialisation de Git

Pour écrire et gérer le code de mon projet, j'ai choisi d'utiliser l'IDE Cursor, qui propose une intégration fluide avec Git via sa console intégrée. Cette fonctionnalité me permet d'effectuer des opérations Git sans quitter mon environnement de développement, ce qui optimise mon flux de travail.

- **Initialisation de Git** : Une fois dans l'IDE Cursor, j'ai ouvert le terminal intégré et me suis positionné dans le répertoire de mon projet :

```
cd C:\xampp\htdocs\Toutpourunnouveaune ← La commande bash
```

Ensuite, j'ai initialisé un dépôt Git local avec la commande suivante :

```
git init ← La commande bash
```

Cela a créé un dépôt Git dans mon projet, ce qui me permet de suivre les modifications, de gérer les versions, et de collaborer avec d'autres développeurs, si nécessaire.

- **Connexion à GitHub** : J'ai ensuite configuré un dépôt distant sur GitHub pour sauvegarder mon projet en ligne et faciliter le travail collaboratif. Pour ce faire, j'ai ajouté le dépôt distant avec la commande suivante, toujours depuis la console de Cursor :

```
git remote add origin https://github.com/monnomutilisateur/Toutpourunnouveaune.git
```

Cela m'a permis de pousser (push) mon code vers GitHub et de le rendre accessible en ligne.

D. Installation et configuration de Composer

Composer est un gestionnaire de dépendances PHP qui simplifie grandement l'intégration de bibliothèques tierces dans mon projet. Son utilisation est essentielle pour gérer les packages nécessaires à mon projet.

- **Installation de Composer :** J'ai téléchargé Composer depuis le site getcomposer.org et l'ai installé en suivant les instructions spécifiques à mon système d'exploitation. Lors de l'installation, Composer a été ajouté à mon PATH système, ce qui me permet de l'utiliser depuis n'importe quel terminal ou console.
- **Initialisation du projet avec Composer :** Dans le répertoire de mon projet, j'ai initialisé Composer avec la commande suivante :

composer init

Cette commande m'a guidé à travers une série de questions concernant les détails de mon projet, comme son nom, sa description, et ses dépendances. Cela a généré un fichier `composer.json` contenant la configuration de mon projet.

- **Installation des dépendances :** J'ai ensuite installé les dépendances nécessaires à mon projet, telles que MongoDB, Dotenv pour gérer les variables d'environnement, et PHPMailer pour l'envoi d'emails. Voici les commandes que j'ai exécutées dans la console de Cursor :

```
composer require mongodb/mongodb
```

```
composer require vlucas/phpdotenv
```

```
composer require phpmailer/phpmailer
```

Ces packages sont désormais inclus dans mon projet et gérés automatiquement par Composer.

E. Configuration de MongoDB pour PHP

Pour utiliser MongoDB avec PHP, j'ai dû configurer mon environnement PHP afin de prendre en charge cette base de données NoSQL.

- **Téléchargement du fichier DLL MongoDB :** J'ai téléchargé le fichier `php_mongodb.dll` depuis le site PECL. J'ai veillé à choisir la version compatible avec ma version de PHP installée via XAMPP.
- **Installation du fichier DLL :** J'ai copié le fichier `php_mongodb.dll` dans le dossier `ext` de mon installation PHP, situé par défaut à `C:\xampp\php\ext`.
- **Modification du fichier `php.ini` :** Pour activer l'extension MongoDB dans PHP, j'ai ouvert le fichier `php.ini` depuis le panneau de contrôle XAMPP. Cela se fait en cliquant sur le bouton "Config" à côté de "Apache", puis en sélectionnant "php.ini". J'ai ajouté la ligne suivante à la fin du fichier :

extension=php_mongodb.dll

- **Redémarrage de Apache** : Enfin, j'ai redémarré Apache via le panneau de contrôle XAMPP pour que les modifications prennent effet. Une fois Apache redémarré, PHP était prêt à interagir avec MongoDB.

F. Conclusion

Grâce à ces étapes, j'ai mis en place un environnement de développement robuste et bien organisé. Avec XAMPP pour gérer le serveur et la base de données, Git pour le contrôle de version, Composer pour la gestion des dépendances, MongoDB pour le stockage des données, et Cursor pour le développement efficace avec une intégration Git fluide, je suis prêt à développer mon projet dans des conditions optimales.

Maquetter des interfaces utilisateur web et web mobile

Pour cette partie j'ai mis 3 wireframes desktop et mobile ainsi que 3 mockups desktop et mobile en annexe.

1. Définition des utilisateurs cibles

Pour commencer la conception de l'interface utilisateur de mon application, j'ai d'abord pris le temps de définir clairement les utilisateurs cibles. Il était crucial pour moi de comprendre qui utiliserait le site et quelles étaient leurs attentes pour pouvoir concevoir une interface qui répondrait à leurs besoins spécifiques.

- **Profil des utilisateurs** : J'ai identifié que les principaux utilisateurs de ce site seraient des parents, qu'ils soient en difficulté ou non, ayant au moins un enfant en bas âge. La profession ou les compétences techniques des utilisateurs n'étaient pas des critères déterminants, car le site devait être accessible et intuitif pour tous, peu importe leur niveau de familiarité avec la technologie.
- **Besoins et objectifs** : Ces utilisateurs recherchent un site complet qui regroupe tout ce dont un parent peut avoir besoin, avant et après la conception d'un bébé. Mon objectif était de créer une plateforme qui offrirait aux parents non seulement un lieu de ressourcement, mais aussi un espace d'échange entre eux, avec des docteurs et des professionnels de la santé. J'ai également voulu inclure des fonctionnalités spécifiques comme la création de cartes de suivi quotidien pour le bébé, des conseils médicaux et alimentaires, et bien d'autres ressources utiles.
- **Parcours utilisateur** : J'ai conceptualisé un parcours utilisateur simple et fluide. Dès la page d'accueil, les utilisateurs peuvent faire défiler la page vers le bas (scroll down) ou vers le haut (scroll up), ou bien cliquer sur des liens menant aux différentes sections du site affichées sur la page d'index. J'ai également prévu un accès rapide à l'onglet de connexion, permettant aux

utilisateurs de s'inscrire rapidement pour accéder à toutes les fonctionnalités du site. Une fois connectés, ils ont accès au forum, qui est réservé exclusivement aux utilisateurs inscrits.

2. Choix de l'outil de maquette

Pour matérialiser mes idées, j'ai choisi d'utiliser **uizard.io** comme outil de maquette. Ce choix s'est avéré judicieux pour plusieurs raisons :

- **Fonctionnalités avancées** : Uizard.io est un outil puissant qui me permet de créer facilement des wireframes, que je peux ensuite transformer en mockups détaillés. Cet outil est particulièrement efficace grâce à son système d'IA intégré, capable de générer des composants très utiles lors du maquettage, ce qui m'a permis de gagner du temps tout en assurant une grande qualité de design.
- **Facilité d'utilisation** : L'interface intuitive de Uizard.io m'a permis de rapidement prendre en main l'outil et de commencer à maquetter mes interfaces sans perte de temps. Cela m'a permis de me concentrer sur l'aspect créatif du projet.

3. Création des wireframes

Avec une compréhension claire des besoins des utilisateurs et un outil adapté, j'ai commencé par la création de wireframes. J'ai décidé de commencer par la version mobile, car une grande partie des utilisateurs accéderont probablement au site via leurs smartphones.

- **Wireframe mobile** : J'ai d'abord conçu un wireframe pour la page d'accueil, qui inclut les éléments essentiels comme la navigation vers la page de connexion/inscription, et les autres éléments présents dans la barre de navigation. Cette page d'accueil devait être claire et intuitive, permettant aux utilisateurs de trouver rapidement ce dont ils avaient besoin.
- **Autres pages** : J'ai également créé des wireframes pour des pages clés telles que la page de contact et la page de création d'un quiz. Chaque wireframe a été conçu en tenant compte de la simplicité et de l'efficacité de l'expérience utilisateur, en m'assurant que chaque élément était à sa place et facilement accessible.

4. Création des mockups

Après avoir validé les wireframes, j'ai entamé la création des mockups, en commençant cette fois par la version desktop. Cette étape m'a permis de donner vie à mes idées, en intégrant les éléments visuels et interactifs qui composeraient l'interface finale.

- **Mockups desktop** : J'ai débuté par la page d'accueil en version desktop, puis j'ai continué avec la page de connexion, la page de contact, et celle de la création d'un quiz. J'ai veillé à ce que chaque page soit cohérente avec les autres, en utilisant un design harmonieux et adapté aux besoins des

utilisateurs.

- **Utilisation des couleurs :** Pour le design des pages, j'ai opté pour dégradé linéaire (linear-gradient (to right, #98B46D, #DAE8C5)) que j'ai appliqué à la barre de navigation et à mon footer. Pour ce qui est des conteneurs j'ai opté pour un background en whitesmoke pour ne pas mettre un blanc qui flash ni une couleur trop vive. Ce choix de couleurs était stratégique : j'ai voulu créer une ambiance chaleureuse et apaisante, en choisissant des tons doux qui ne fatiguent pas les yeux et qui inspirent la tranquillité, ce qui est particulièrement important pour des parents potentiellement fatigués ou en difficulté.
- **Thème et visuels :** Pour renforcer le thème du site, j'ai utilisé des images de fond représentant une petite famille tenant leur bébé dans les bras. Ces images sont non seulement en adéquation avec le contenu du site, mais elles créent aussi une atmosphère rassurante et accueillante pour les utilisateurs.
- **Design responsive :** J'ai veillé à ce que chaque page soit non seulement belle sur un écran desktop, mais aussi adaptée pour les versions mobiles. J'ai créé des maquettes spécifiques pour les écrans plus petits, en intégrant des éléments comme un bouton burger pour la navigation sur mobile, garantissant ainsi une expérience utilisateur optimale, quel que soit l'appareil utilisé.

5. [Documentation des choix de design](#)

Enfin, j'ai soigneusement documenté tous les choix de design que j'ai faits au cours du processus de maquettage. Cette documentation est essentielle pour garantir la cohérence du design lors de la phase de développement.

Choix des couleurs : J'ai sélectionné des couleurs qui rappellent des tons chauds et apaisants, dans le but de créer une interface agréable pour les parents, en particulier ceux qui peuvent être stressés ou fatigués. L'objectif était de faire en sorte que les utilisateurs puissent naviguer sur le site sans effort, grâce à un design intuitif qui met en avant les informations clés sans surcharger l'écran.

- **Structure et organisation :** J'ai opté pour un style de présentation en cartes et tableaux, ce qui permet aux utilisateurs de visualiser facilement les informations et de naviguer intuitivement à travers le contenu du site. Ce choix est en ligne avec l'objectif d'offrir une expérience utilisateur fluide et sans frustration.

En résumé, cette partie du projet m'a permis de poser les bases visuelles et structurelles du site, en tenant compte des besoins des utilisateurs et en appliquant des principes de design adaptés à leur contexte. Grâce à une approche méthodique et à l'utilisation d'outils modernes, j'ai pu créer des maquettes qui reflètent fidèlement l'expérience que je souhaite offrir aux parents qui utiliseront ce site.

Réaliser des interfaces utilisateur statiques web ou web mobile

Pour la réalisation des interfaces utilisateur statiques de mon projet, j'ai utilisé un ensemble de technologies et de bonnes pratiques pour créer des pages web fonctionnelles et esthétiques. Voici les différentes étapes et techniques que j'ai mises en œuvre.

1. Structuration des pages HTML et PHP

- **Séparation des composants en fichiers réutilisables** : J'ai organisé le code en plusieurs fichiers PHP pour permettre une réutilisation des composants et faciliter la maintenance. Par exemple, les en-têtes, les barres de navigation, et les pieds de page sont inclus dans des fichiers distincts (header.php, navbar_admin.php, footer.php) qui sont ensuite inclus dans chaque page via `require_once`. Cette approche permet de centraliser les modifications pour tous les éléments partagés sur plusieurs pages.
- **Utilisation des templates** : L'interface utilisateur est divisée en plusieurs sections réutilisables, telles que des formulaires dynamiques, des listes d'éléments et des modals pour la mise à jour des informations. Ces sections sont générées dynamiquement avec PHP en fonction des données récupérées de la base de données.

1. **Intégration de Bootstrap pour un design réactif**

- **Mise en page fluide et responsive** : J'ai utilisé Bootstrap pour créer une interface responsive qui s'adapte automatiquement à différents types d'écrans (desktop, tablette, mobile). Grâce à Bootstrap, les éléments de la page, tels que les formulaires, les boutons, et les cartes, sont bien alignés et restent cohérents sur toutes les tailles d'écran.
- **Utilisation des composants Bootstrap** : Pour les fonctionnalités telles que les barres de navigation, les modals, et les boutons, j'ai tiré parti des composants Bootstrap prédéfinis. Par exemple, les barres de navigation sont fixées en haut de la page avec l'utilisation de classes telles que `navbar-fixed-top`, assurant ainsi une accessibilité facile aux utilisateurs.

2. CSS personnalisé pour un design cohérent

- **Stylisation des éléments** : Bien que Bootstrap offre une bonne base, j'ai ajouté des styles CSS personnalisés pour affiner l'apparence des pages et refléter les choix esthétiques du projet. Par exemple, j'ai appliqué un fond d'écran personnalisé et stylisé les titres et sections avec des couleurs douces pour correspondre au thème du site.

```
body {  
    background-image: url('../assets/image/background.jpg');  
    padding-top: 48px;  
}
```

```
h1, .mt-5 {  
  background: whitesmoke;  
  border-radius: 15px;  
}
```

- **Mise en forme des formulaires et des boutons** : J'ai veillé à ce que les formulaires soient non seulement fonctionnels mais aussi esthétiques, avec des bords arrondis, des couleurs contrastées pour les boutons, et des éléments interactifs bien définis pour améliorer l'expérience utilisateur.

3. [Ajout de l'interactivité avec JavaScript et jQuery](#)

- **Ajout dynamique des éléments** : J'ai utilisé JavaScript, notamment avec jQuery, pour permettre à l'administrateur d'ajouter dynamiquement des questions et des réponses dans les formulaires de création de quiz. Cette fonctionnalité permet une expérience utilisateur plus fluide, sans nécessiter de rechargement de la page.
- **Gestion des modals et de la navigation** : J'ai utilisé JavaScript pour gérer les interactions utilisateur telles que l'ouverture de modals pour la modification du profil ou l'ajout d'amis. Cela permet de maintenir une interface utilisateur propre et de minimiser le nombre de redirections ou de rechargements de page.

4. [Validation et sécurité](#)

- **Validation des formulaires** : Pour les formulaires, j'ai mis en place des validations côté client (avec HTML5 et JavaScript) et côté serveur (avec PHP) pour garantir que les données soumises sont correctes et sécurisées. J'ai utilisé des filtres PHP pour assainir les entrées utilisateur et protéger contre les attaques par injection.
- **Protection CSRF** : Pour chaque formulaire critique, j'ai intégré un jeton CSRF généré dynamiquement et validé à chaque soumission, renforçant ainsi la sécurité contre les attaques CSRF.

```
// Génération du token CSRF  
$_SESSION['csrf_token'] = bin2hex(random_bytes(32));  
// Validation du token CSRF  
if ($_SERVER['REQUEST_METHOD'] == 'POST') {  
  if (!isset($_POST['csrf_token']) || $_POST['csrf_token'] !== $_SESSION['csrf_token']) {  
    $_SESSION['error_message'] = "Erreur de sécurité : jeton CSRF invalide.";  
    header('Location: add_quiz.php');  
    exit;  
  }  
  // Suite du traitement...  
}
```

5. Conclusion

La réalisation des interfaces utilisateur statiques de ce projet a impliqué une combinaison de bonnes pratiques en matière de structuration du code, d'implémentation d'un design réactif, et de renforcement de la sécurité. En utilisant des outils modernes comme Bootstrap et jQuery, et en respectant les standards du développement web, j'ai pu créer des pages statiques qui sont non seulement fonctionnelles, mais aussi esthétiques et faciles à naviguer. Ces pages servent de fondation solide pour les futures fonctionnalités dynamiques de l'application.

Développer la partie dynamique des interfaces utilisateur web ou web mobile

Dans cette section, je vais expliquer comment j'ai développé la partie dynamique des interfaces utilisateur de mon projet. L'objectif était de permettre aux utilisateurs de créer des quiz de manière interactive et dynamique, en ajoutant des questions et des réponses directement depuis l'interface utilisateur.

1. Gestion dynamique des formulaires avec JavaScript

- **Ajout dynamique de questions** : J'ai implémenté une fonctionnalité qui permet aux utilisateurs d'ajouter dynamiquement des questions à un quiz en cours de création. Grâce à JavaScript, j'ai pu générer des formulaires supplémentaires au fur et à mesure que l'utilisateur en avait besoin, sans avoir à recharger la page.

Cette fonctionnalité repose sur l'écoute d'un événement click sur un bouton "Ajouter une question". Lorsqu'un utilisateur clique sur ce bouton, un nouveau bloc de formulaire, préconfiguré pour une nouvelle question, est inséré dans la page.

let questionIndex = 1;

- **Ajout dynamique de réponses** : De la même manière, les utilisateurs peuvent ajouter des réponses multiples à chaque question. Cette fonctionnalité utilise également JavaScript pour insérer de nouveaux champs de réponse dans le formulaire correspondant à une question spécifique.

Lorsqu'un utilisateur clique sur "Ajouter une réponse", un nouveau champ de réponse est inséré dans la section des réponses pour cette question. Cela permet de gérer un nombre variable de réponses par question, selon les besoins de l'utilisateur.

Voir l'extrait de code dans les annexes extraits de code : page 77-78

2. [Validation et sécurité côté serveur](#)

- **Protection CSRF** : Pour protéger le formulaire de création de quiz contre les attaques CSRF (Cross-Site Request Forgery), j'ai mis en place un jeton CSRF qui est généré à chaque chargement de la page et vérifié lors de la soumission du formulaire. Cela garantit que seules les requêtes légitimes provenant du formulaire actuel sont acceptées par le serveur.

```
// Génération du token CSRF
$_SESSION['csrf_token'] = bin2hex(random_bytes(32));

// Validation du token CSRF lors de la soumission du formulaire
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    if (!isset($_POST['csrf_token']) || $_POST['csrf_token'] !== $_SESSION['csrf_token']) {
        $_SESSION['error_message'] = "Erreur de sécurité : jeton CSRF invalide.";
        header('Location: add_quiz.php');
        exit;
    }
}
```

- **Validation des entrées utilisateur** : Lors de la soumission du formulaire, les données sont également validées côté serveur pour s'assurer que les informations sont correctes et sécurisées. Par exemple, le titre du quiz et le texte des questions/réponses sont filtrés pour empêcher les injections SQL ou XSS (Cross-Site Scripting).

```
$titre = filter_input(INPUT_POST, 'titre', FILTER_SANITIZE_STRING);
$questions = $_POST['questions']; // Validation personnalisée nécessaire pour les tableaux complexes

if ($titre && $questions) {
    $quiz->addQuiz($titre, $questions);
    header('Location: manage_quizzes.php');
    exit;
}
```

3. [Interaction dynamique et retours utilisateur](#)

- **Affichage des erreurs et confirmations** : Après la soumission du formulaire, si une erreur survient (comme un jeton CSRF invalide ou une validation échouée), un message d'erreur est stocké dans la session et affiché à l'utilisateur sur la même page. En cas de succès, l'utilisateur est redirigé vers la page de gestion des quiz avec un message de confirmation.


```
if ($titre && $questions) {  
    $quiz->addQuiz($titre, $questions);  
    header('Location: manage_quizzes.php');  
    exit;  
} else {  
    $_SESSION['error_message'] = "Veuillez remplir tous les champs requis."  
    header('Location: add_quiz.php');  
    exit;  
}
```

4. Conclusion

Le développement de la partie dynamique des interfaces utilisateur a permis de transformer une interface statique en un outil interactif et flexible, capable de gérer des données complexes de manière intuitive. Grâce à l'intégration de JavaScript pour la manipulation dynamique des formulaires et à la validation rigoureuse côté serveur, j'ai pu créer une expérience utilisateur riche et sécurisée, adaptée aux besoins des utilisateurs administrateurs du site. Cette approche assure une gestion fluide des contenus dynamiques tout en maintenant un haut niveau de sécurité et de réactivité.

2. Précisez les moyens utilisés :

Pour mener à bien l'activité de développement de la partie front-end d'une application web ou web mobile sécurisée, j'ai utilisé un ensemble d'outils et de technologies qui m'ont permis de structurer, coder, tester et déployer efficacement le projet **Toutpourunnouveaune**. Voici un aperçu des moyens utilisés :

I. Outils et technologies de développement

- **IDE Cursor** : Pour l'édition de code, j'ai choisi l'IDE Cursor, qui offre une intégration fluide avec Git grâce à sa console intégrée. Cela m'a permis de gérer efficacement le code source et les versions tout en bénéficiant d'un environnement de développement complet et personnalisé.
- **XAMPP** : Pour configurer mon environnement de développement local, j'ai utilisé XAMPP, un package tout-en-un qui comprend Apache, MySQL, et PHP. Cela m'a permis de simuler un serveur web complet sur ma machine locale, facilitant ainsi les tests et le développement.
- **Composer** : Pour la gestion des dépendances PHP, j'ai utilisé Composer, un outil essentiel qui m'a permis d'intégrer facilement des bibliothèques tierces comme MongoDB, Dotenv pour les variables d'environnement, et PHPMailer pour la gestion des emails.
- **MongoDB** : Pour le stockage des données, notamment pour les fonctionnalités de vues des

threads et le score des parents après un quiz, j'ai utilisé MongoDB, une base de données NoSQL. J'ai configuré MongoDB avec PHP en installant les extensions nécessaires et en les intégrant via Composer.

Configurations spécifiques

- **Environnement local** : Le serveur local Apache et la base de données MySQL ont été configurés via XAMPP. J'ai veillé à ce que les versions des logiciels soient compatibles entre elles et avec les extensions PHP que j'ai installées, comme `php_mongodb.dll`.
- **Gestion des versions avec Git** : Dès le début du projet, j'ai initialisé un dépôt Git pour suivre les modifications et gérer les versions du code. J'ai également configuré un dépôt distant sur GitHub pour le stockage en ligne et le partage éventuel du projet. Git m'a permis de revenir à des versions antérieures en cas de besoin.

Méthodologies et approches de développement

- **Approche modulaire** : J'ai structuré le code en modules réutilisables, séparant les différentes responsabilités dans des fichiers distincts (e.g., `header.php`, `footer.php`, `navbar_admin.php`). Cela a permis une meilleure organisation et facilité la maintenance du code.
- **Responsive Design** : Grâce à Bootstrap et à des styles CSS personnalisés, j'ai assuré que l'application soit entièrement responsive, adaptée aux écrans de toutes tailles. Cette approche a été renforcée par la création de maquettes spécifiques pour mobile et desktop, ce qui a guidé le développement.
- **Sécurité** : Pour garantir la sécurité de l'application, j'ai intégré des pratiques comme la génération et la validation de tokens CSRF pour prévenir les attaques Cross-Site Request Forgery, ainsi que la validation côté serveur des entrées utilisateur pour éviter les injections SQL et XSS.
- **Tests et validations** : Bien que je n'aie pas utilisé de cadre formel de test automatisé, j'ai effectué des tests manuels rigoureux à chaque étape du développement. Cela comprenait des vérifications de la compatibilité multi-navigateurs, des tests de performance, et des validations de sécurité pour m'assurer que l'application fonctionne de manière optimale et sécurisée.

Collaboration et gestion de projet

- **GitHub pour le contrôle de version** : GitHub a été utilisé pour héberger le dépôt du projet et faciliter le suivi des versions. Ce dépôt a permis de centraliser le code et d'assurer que toutes les modifications soient correctement documentées et récupérables en cas de problème.
- **Documentations et commentaires** : Tout au long du développement, j'ai pris soin de documenter le code avec des commentaires clairs et concis. J'ai également créé une documentation externe

DOSSIER PROFESSIONNEL (DP)

qui détaille les choix de design, les configurations spécifiques, et les instructions d'installation et d'utilisation, ce qui facilite la prise en main du projet par d'autres développeurs.

Conclusion

Les moyens utilisés pour développer l'application **Toutpourunnouveaune** m'ont permis de créer une solution robuste, sécurisée et évolutive. Grâce à un ensemble d'outils bien choisis, à des configurations adaptées, et à des méthodologies éprouvées, j'ai pu gérer efficacement chaque étape du projet, du prototypage initial jusqu'à la mise en place d'une application web fonctionnelle et sécurisée.

3. Avec qui avez-vous travaillé ?

Seul ou en équipe : avec un élève de la même formation que moi, pour s'expliquer ce que nous avons appris et nous nous sommes montré aussi nos projets afin d'apprendre l'un de l'autre. Souvent nous étions en session live Google Meet pour se parler.

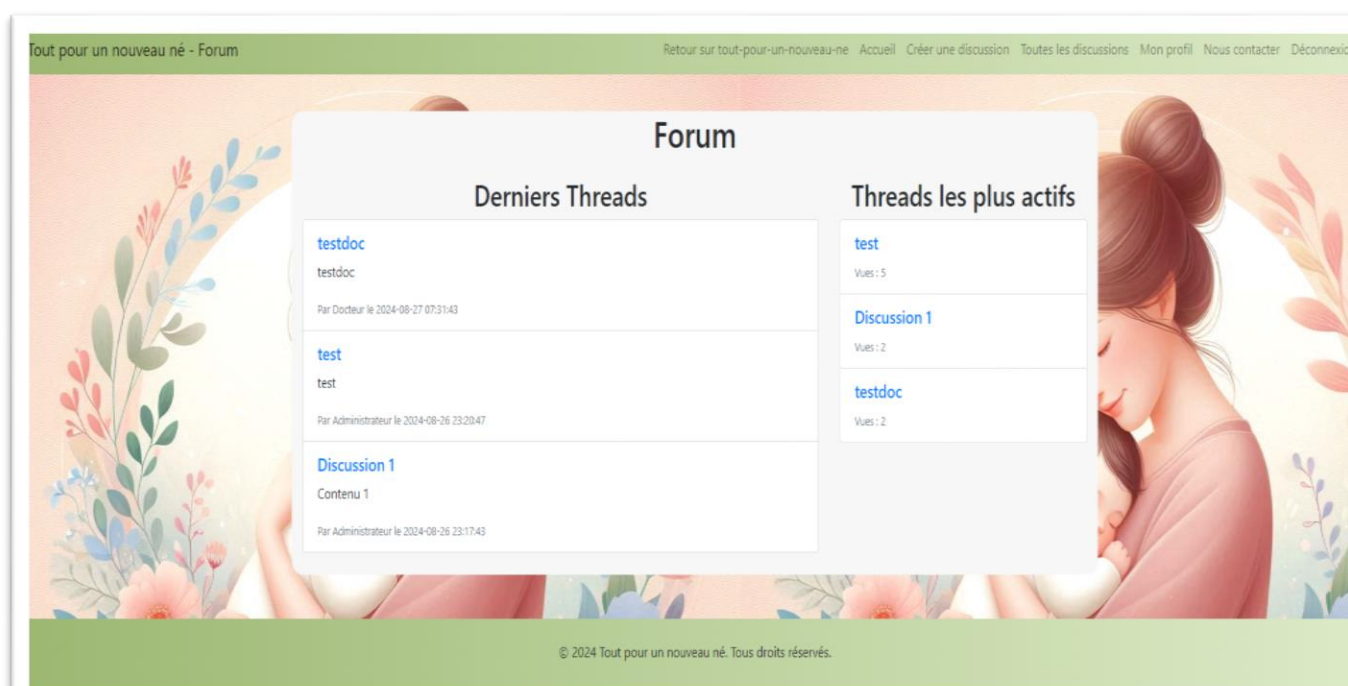
4. Contexte

Nom de l'entreprise, organisme ou association STUDI

Chantier, atelier, service Pendant la formation

Période d'exercice Du : 05/052024 au : 27/08/2024

5. Informations complémentaires (facultatif)



DOSSIER PROFESSIONNEL ^(DP)

Cette image montre la page d'accueil du forum, indiquant les dernières discussions et les discussions les plus actives (les vues sont récupérées de MongoDB)

Pendant la réalisation de ce projet, j'ai dû surmonter plusieurs défis personnels, notamment en jonglant entre ma vie de famille et ma formation. Ces obstacles m'ont parfois ralenti, mais ils ont aussi renforcé ma détermination à réussir et à poursuivre ma passion pour le développement.

Activité-type 1

Développer la partie front-end d'une application web ou web mobile sécurisée

Exemple n°2 ▶ E-learning Abdu application web responsive

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

I. Installation et configuration de l'environnement de développement

Pour le développement du projet **E-learning Abdu**, j'ai mis en place un environnement de travail local en utilisant plusieurs outils et technologies adaptés aux besoins du projet.

1. Installation de XAMPP

- **Téléchargement et installation** : J'ai téléchargé XAMPP depuis le site officiel d'Apache Friends, qui regroupe Apache, MySQL, PHP, et phpMyAdmin. L'installation a été réalisée sur un système Windows, avec les composants nécessaires tels qu'Apache et MySQL.
- **Lancement de XAMPP** : Une fois installé, le panneau de contrôle XAMPP a été utilisé pour démarrer les services Apache et MySQL, assurant ainsi un environnement serveur local fonctionnel pour le développement de l'application.

2. Configuration du projet

- **Création du répertoire de projet** : Le projet **E-learning Abdu** a été structuré dans le répertoire htdocs de XAMPP, avec une organisation claire des fichiers et dossiers pour faire un projet en MVC avec l'organisation suivante :
 - Model → Mon dossier models dans src ;
 - Vues → Mon dossier views dans src ;
 - Contrôleurs → Mon dossier controllers dans src ;
- **Utilisation de l'IDE Cursor** : Pour le développement, j'ai utilisé l'IDE Cursor, qui offre une intégration fluide avec Git, permettant de gérer les versions du code et de collaborer efficacement.

3. Gestion des dépendances avec Composer

- **Installation de Composer** : Composer a été utilisé pour gérer les dépendances PHP du projet, telles que MongoDB et PHPMailer. Composer a été configuré et utilisé pour installer les packages nécessaires via le terminal de Cursor.

- **Configuration de MongoDB :** MongoDB a été choisi comme base de données pour stocker les informations relatives aux utilisateurs, forums, cours, et autres contenus dynamiques. L'extension PHP `php_mongodb.dll` a été installée et activée via le fichier `php.ini` de XAMPP.

Maquetter des interfaces utilisateur web et web mobile

Les maquettes pour ce projet seront dans les annexes : Wireframe (3 mobiles et 3 desktops) Mockups (3 mobiles et 3 desktops)

Pour concevoir les interfaces utilisateur du projet **E-learning Abdu**, j'ai suivi une approche méthodique, en commençant par la définition des utilisateurs cibles et en terminant par la création de maquettes détaillées.

1. Définition des utilisateurs cibles

- **Profil des utilisateurs :** Les principaux utilisateurs identifiés pour ce projet sont des étudiants, des formateurs, et des administrateurs. Chacun de ces groupes d'utilisateurs a des besoins spécifiques que j'ai pris en compte lors de la conception des interfaces.
- **Besoins et objectifs :** Les étudiants doivent pouvoir accéder facilement aux cours, participer à des quiz, et soumettre des examens. Les formateurs ont besoin d'outils pour créer et gérer des cours, des quiz, et des examens. Les administrateurs doivent avoir un contrôle total sur le contenu et les utilisateurs du site.

2. Création des wireframes

- **Wireframe de la page d'accueil :** J'ai conçu une page d'accueil engageante avec une section "Hero", des carrousels d'images, et des sections mettant en avant les fonctionnalités clés du site, comme les cours de haute qualité, l'apprentissage flexible, et le contenu interactif.
- **Autres pages importantes :** Des wireframes ont également été créés pour les pages de gestion des formations, des catégories, des sous-catégories, et des pages de cours, en veillant à ce que chaque élément soit bien structuré et facile d'accès pour les utilisateurs.

3. Création des mockups

- **Mockups desktop :** Les maquettes desktop ont été réalisées en intégrant des éléments visuels cohérents et en assurant une navigation intuitive. Le choix des couleurs, comme le bleu pour les boutons et les éléments interactifs, a été fait pour garantir une interface visuellement agréable et professionnelle.
- **Design responsive :** J'ai également créé des maquettes pour les versions mobiles du site, assurant ainsi une compatibilité totale avec différents types d'appareils.

Réalisation des interfaces utilisateur statiques web ou web mobile

Pour la réalisation des interfaces statiques du projet **E-learning Abdu**, j'ai mis en œuvre une série de technologies et de bonnes pratiques pour créer une plateforme à la fois fonctionnelle, esthétique, et sécurisée.

4. Structuration des pages HTML et PHP

La structuration du projet a joué un rôle crucial dans la maintenabilité et la modularité du code. J'ai utilisé une architecture basée sur la séparation des responsabilités, où chaque composant du site est géré par des fichiers PHP spécifiques.

- **Séparation des composants** : J'ai structuré le projet en différents fichiers PHP réutilisables, tels que header.php, footer.php, et navbar.php. Cela a permis une gestion centralisée des éléments communs et facilité la maintenance du code. Par exemple, les éléments de navigation sont centralisés dans navbar.php, permettant ainsi une modification unique pour l'ensemble du site.

```
include_once '../../../public/templates/header.php';
```

```
include_once '../navbar_admin.php';
```

- **Templates réutilisables** : L'utilisation de templates a permis de générer dynamiquement des pages en fonction des données récupérées de la base de données. Cette approche garantit une expérience utilisateur cohérente à travers l'ensemble des pages du site, tout en facilitant les futures mises à jour.

Exemple de code :

```
<?php
session_start();
// Génération d'un token CSRF
if (empty($_SESSION['csrf_token'])) {
    $_SESSION['csrf_token'] = bin2hex(random_bytes(32));
}
$db = new Database();
$db = $db->getConnection();

$formationController = new FormationController($db);
$formations = $formationController->getAllFormations();

header('Content-Type: text/html; charset=utf-8');
include_once '../../../public/templates/header.php';
include_once '../navbar_admin.php';
?>
```

Ce code montre comment les composants réutilisables sont inclus dans les pages, assurant une gestion centralisée et dynamique des ressources partagées.

5. Intégration de Bootstrap pour un design réactif

Bootstrap a été utilisé pour assurer la réactivité et la compatibilité multi-plateformes de l'interface utilisateur.

- **Mise en page fluide** : En utilisant la grille de Bootstrap, j'ai créé des interfaces qui s'adaptent automatiquement aux différentes tailles d'écran, garantissant ainsi une expérience utilisateur optimale sur les ordinateurs de bureau, les tablettes, et les smartphones. Les composants Bootstrap, tels que les boutons, les cartes, et les barres de navigation, ont été personnalisés pour correspondre à l'identité visuelle de la plateforme.
- **Carousel et modals** : J'ai intégré des carrousels d'images pour rendre la page d'accueil plus dynamique et attirer l'attention des utilisateurs sur les fonctionnalités clés. Les modals ont été utilisés pour les formulaires de gestion, tels que l'ajout ou la modification de formations, permettant une interaction utilisateur fluide sans rechargement de page.

Exemple de code :

```
<!-- Modal Formation -->
<div id="formationModal" class="modal fade" tabindex="-1" role="dialog">
  <div class="modal-dialog" role="document">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title">Ajouter une Formation</h5>
        <button type="button" class="close" data-dismiss="modal">&times;</button>
      </div>
      <div class="modal-body">
        <form id="formationForm">
          <input type="hidden" name="csrf_token" value="<?php echo $_SESSION['csrf_token'];
?>">
          <div class="form-group">
            <label for="formationName">Nom</label>
            <input type="text" id="formationName" name="name" class="form-control" required>
          </div>
          <button type="submit" class="btn btn-primary">Enregistrer</button>
        </form>
      </div>
    </div>
  </div>
</div>
```


Ce code illustre l'utilisation des modals de Bootstrap pour une gestion intuitive des formations, améliorant ainsi l'expérience utilisateur.

6. CSS personnalisé pour un design cohérent

Bien que Bootstrap fournisse une base solide pour le design, j'ai également utilisé du CSS personnalisé pour affiner et adapter l'apparence des éléments de l'interface utilisateur afin de correspondre exactement aux besoins du projet.

- **Stylisation des éléments** : J'ai ajouté des styles CSS personnalisés pour des sections spécifiques comme la section "Hero" de la page d'accueil, qui utilise une image de fond avec des textes en surbrillance et un bouton d'appel à l'action bien visible. Cela permet de créer une première impression forte et engageante pour les visiteurs.
- **Mise en forme des formulaires** : Les formulaires, tels que ceux utilisés pour ajouter ou modifier des formations, catégories, et pages, ont été conçus pour être à la fois esthétiques et fonctionnels. Chaque formulaire a été stylisé pour s'intégrer parfaitement avec le reste de l'interface, tout en restant intuitif pour l'utilisateur.

A. Ajout de l'interactivité avec JavaScript et jQuery

Pour rendre l'application plus dynamique et interactive, j'ai utilisé JavaScript, principalement avec jQuery, pour gérer les interactions utilisateur.

- **Dynamisation des formulaires** : JavaScript a été utilisé pour permettre l'ajout et la modification dynamique d'éléments tels que des catégories, des sous-catégories, et des pages de cours sans rechargement de la page. Cette fonctionnalité améliore l'expérience utilisateur en rendant l'application plus réactive.
- **Modals et navigation** : JavaScript a également été utilisé pour gérer les modals et d'autres interactions utilisateur, comme le défilement fluide vers le haut et le bas de la page. Ces éléments interactifs rendent la navigation plus agréable et intuitive.

Exemple de code :

```
<script>
$(document).ready(function () {
  $('#formationForm').submit(function (e) {
    e.preventDefault();
    var formData = $(this).serialize();
    $.ajax({
      url: 'save_mEDIATEQUE.php',
      method: 'POST',
      data: formData + '&action=save_formation',
      success: function (response) {
```

```
        if (response.status === 'success') {
            $('#formationModal').modal('hide');
            location.reload();
        } else {
            alert('Erreur lors de l\'ajout de la formation.');
```

Ce code montre comment jQuery est utilisé pour gérer les soumissions de formulaires de manière asynchrone, permettant ainsi une interaction plus fluide et sans interruption pour l'utilisateur.

7. Validation et sécurité

La validation des données et la sécurité sont des aspects fondamentaux que j'ai rigoureusement intégrés à chaque étape du développement de la plateforme **E-learning Abdu**. Mon objectif était de créer un environnement sûr et fiable, où les données des utilisateurs sont protégées et les opérations sensibles sont exécutées de manière sécurisée.

- **Validation des formulaires** : Pour garantir l'intégrité des données soumises par les utilisateurs, j'ai mis en place une double validation des formulaires, à la fois côté client et côté serveur.
 - **Validation côté client** : Utilisant principalement JavaScript et HTML5, cette validation permet de capturer les erreurs immédiatement, avant même que les données ne soient envoyées au serveur. Par exemple, des vérifications de champ obligatoire, des formats d'email et des limites de longueur ont été appliquées pour assurer que les données de base respectent les règles avant soumission.
 - **Validation côté serveur** : Une fois les données soumises, une validation supplémentaire est effectuée côté serveur à l'aide de filtres PHP. J'ai utilisé des fonctions comme `filter_input()` et `filter_var()` pour nettoyer et valider les données. Cette étape est cruciale pour empêcher les attaques courantes telles que les injections SQL et les attaques XSS (Cross-Site Scripting). Toutes les entrées utilisateur sont scrutées pour s'assurer qu'elles ne contiennent pas de code malveillant, garantissant ainsi que seules des données sûres et valides sont traitées par le système.

Exemple de code :

```
// Validation côté serveur pour un champ de formulaire
$username = filter_input(INPUT_POST, 'username', FILTER_SANITIZE_STRING);
$email = filter_input(INPUT_POST, 'email', FILTER_VALIDATE_EMAIL);

if (!$username || !$email) {
```

DOSSIER PROFESSIONNEL (DP)

```
$error = "Nom d'utilisateur ou email invalide.";
}
```

- **Protection CSRF** (Cross-Site Request Forgery) : Pour protéger l'application contre les attaques CSRF, j'ai implémenté un mécanisme de génération et de vérification de tokens CSRF pour chaque soumission de formulaire critique. Chaque fois qu'un formulaire est rendu, un token unique est généré et intégré dans le formulaire en tant que champ caché. Ce token est ensuite vérifié lors de la soumission du formulaire pour s'assurer qu'il correspond au token généré pour la session utilisateur en cours.
 - **Génération du token CSRF** : Le token est généré en utilisant `bin2hex(random_bytes(32))`, garantissant ainsi un niveau élevé d'entropie et de sécurité.
 - **Vérification du token CSRF** : Lors de la soumission du formulaire, le token est comparé à celui stocké dans la session de l'utilisateur. Si les tokens ne correspondent pas, la soumission du formulaire est bloquée et une alerte de sécurité est déclenchée, empêchant ainsi toute action non autorisée.

Exemple de code :

```
// Génération du token CSRF
if (empty($_SESSION['csrf_token'])) {
    $_SESSION['csrf_token'] = bin2hex(random_bytes(32));
}
// Vérification du token CSRF lors de la soumission du formulaire
if ($_POST['csrf_token'] !== $_SESSION['csrf_token']) {
    die("Erreur de sécurité : token CSRF invalide.");
}
```

Cette approche renforce la sécurité de la plateforme en assurant que seules les requêtes légitimes, provenant de l'utilisateur autorisé, sont acceptées. Cela protège efficacement contre les attaques CSRF, où un attaquant pourrait tenter de soumettre un formulaire en se faisant passer pour l'utilisateur. En combinant ces techniques de validation et de protection, j'ai pu assurer que la plateforme **E-learning Abdu** offre une sécurité robuste, minimisant les risques d'exploitation des failles courantes et garantissant l'intégrité des données utilisateur.

Développer la partie dynamique des interfaces utilisateur web ou web mobile

Le développement de la partie dynamique des interfaces utilisateur du projet E-learning Abdu a été crucial pour permettre aux utilisateurs d'interagir efficacement avec la plateforme. J'ai mis en œuvre des

techniques avancées pour garantir une expérience utilisateur fluide et réactive, sans nécessiter de rechargement de page.

1. Gestion dynamique des formulaires avec JavaScript

A. Ajout dynamique de formations, catégories, et pages :

J'ai implémenté des fonctionnalités permettant d'ajouter, modifier, et supprimer des formations, catégories, sous-catégories, et pages de manière dynamique via AJAX. Par exemple, lors de l'ajout d'une formation, le code suivant montre comment cela est géré :

```
$('#formationForm').submit(function (e) {  
    e.preventDefault(); // Empêche le rechargement de la page  
    var formData = $(this).serialize();  
    formData += '&action=save_formation';  
    $.ajax({  
        url: 'save_mediateque.php',  
        method: 'POST',  
        data: formData,  
        success: function (response) {  
            if (response.status === 'success') {  
                loadFormations();  
                $('#formationModal').modal('hide');  
            } else {  
                alert('Erreur lors de l\'ajout de la formation.');            }  
        },  
        error: function(xhr, status, error) {  
            console.error('Erreur lors de l\'enregistrement de la formation:', error);  
            alert('Une erreur est survenue lors de l\'enregistrement de la formation.');        }  
    });  
});
```

Ce processus permet aux utilisateurs d'ajouter des formations sans quitter ou recharger la page, ce qui améliore l'efficacité et la convivialité de l'interface.

B. Utilisation de JavaScript, AJAX et preventDefault()

Pour rendre l'interface plus réactive et intuitive, j'ai utilisé JavaScript pour intercepter certaines actions utilisateur, comme la soumission de formulaires, afin de les traiter de manière asynchrone avec AJAX. Cela permet d'éviter les rechargements de pages, réduisant ainsi les interruptions pour l'utilisateur.

Exemple 1 : Formulaire de soumission dynamique

Lors de l'ajout d'une formation à un étudiant, j'ai utilisé `preventDefault ()` pour empêcher le comportement par défaut du formulaire qui aurait rechargé la page. Ensuite, une requête AJAX est envoyée pour traiter les données côté serveur et mettre à jour l'interface en conséquence.

```
const formationForm = document.getElementById('formationFormulaire');
formationForm.addEventListener('submit', function (e) {
  e.preventDefault(); // Empêche le rechargement de la page
  const formData = new FormData(formationForm);
  fetch('assign_course_to_student.php', {
    method: 'POST',
    body: formData
  })
  .then(response => response.json())
  .then(data => {
    alert(data.message);
    if (data.status === 'success') {
      formationForm.reset();
      $('#formationModal').modal('hide');
      location.reload();
    }
  })
  .catch(error => console.error('Erreur:', error));
});
```

Exemple 2 : Chargement dynamique des détails utilisateur dans une modal :

Pour afficher les détails d'un utilisateur sans recharger la page, j'ai utilisé une requête AJAX pour récupérer ces informations et les injecter directement dans une modal. Cela permet à l'utilisateur de consulter des informations complémentaires sans quitter la page actuelle.

```
$(document).ready(function() {
  $('.btn-profile').on('click', function() {
    var userId = $(this).data('id');

    $.ajax({
      url: 'get_profile.php', // Fichier PHP qui va traiter la requête AJAX
      type: 'GET',
      data: { user_id: userId },
      success: function(response) {
        // Insère la réponse (le contenu du profil) dans la modal
        $('#profileModal .modal-body').html(response);
      },
      error: function(xhr, status, error) {
        console.error('Erreur AJAX: ' + status + ' - ' + error);
      }
    });
  });
});
```

```
}  
});  
});  
});
```

Ces méthodes permettent de rendre l'application plus fluide et réactive, offrant ainsi une meilleure expérience utilisateur. En utilisant AJAX pour les interactions dynamiques et `preventDefault()` pour contrôler les événements, j'ai pu créer une interface utilisateur moderne et efficace, réduisant les temps de chargement et améliorant l'accessibilité.

C. Mise à jour dynamique des contenus

Les utilisateurs peuvent également modifier et supprimer des éléments via des interactions AJAX. Par exemple, lors de la modification d'une formation ou d'une page, les changements sont immédiatement reflétés dans l'interface, rendant l'application plus réactive et agréable à utiliser.

2. Validation et sécurité côté serveur

A. Protection CSRF :

Chaque requête soumise via un formulaire dynamique est protégée par un token CSRF (Cross-Site Request Forgery), qui est vérifié avant toute action côté serveur. Cela empêche les attaques malveillantes qui pourraient exploiter les sessions utilisateur pour envoyer des requêtes non autorisées.

B. Validation serveur :

Les données soumises par les utilisateurs sont systématiquement validées côté serveur pour garantir leur conformité. Les entrées sont filtrées afin d'éviter les injections SQL et les attaques XSS (Cross-Site Scripting), assurant ainsi la sécurité et l'intégrité des données traitées par l'application.

3. Interaction dynamique et retours utilisateur

A. Affichage des erreurs et confirmations :

Les utilisateurs reçoivent des retours immédiats après chaque action, qu'il s'agisse d'ajouter, de modifier ou de supprimer des éléments. Des messages de confirmation ou d'erreur sont affichés directement sur l'interface, ce qui améliore considérablement l'expérience utilisateur en offrant une communication claire et instantanée des résultats de leurs actions.

2. Précisez les moyens utilisés :

Pour mener à bien le développement du projet **E-learning Abdu**, j'ai utilisé un ensemble d'outils, de technologies, et de méthodologies qui m'ont permis de structurer, coder, tester, et déployer efficacement l'application.

I. Outils et technologies de développement

- **IDE Cursor** : Pour l'édition de code, j'ai utilisé l'IDE Cursor, qui offre une intégration fluide avec Git et permettant une gestion efficace du code source.
- **XAMPP** : J'ai utilisé pour configurer l'environnement de développement local, incluant Apache et MySQL, nécessaires au fonctionnement de l'application.
- **Composer** : Composer a été employé pour gérer les dépendances PHP, en intégrant des bibliothèques essentielles comme MongoDB et PHPMailer.

Configurations spécifiques

- **Environnement local** : Le serveur local Apache et la base de données MySQL ont été configurés via XAMPP, avec des ajustements spécifiques pour s'assurer de la compatibilité entre les différentes versions des logiciels utilisés.
- **Gestion des versions avec Git** : Un dépôt Git a été initialisé dès le début du projet pour suivre les modifications du code et gérer les versions. Le code a également été push sur GitHub pour un repository.

Méthodologies et approches de développement

- **Approche modulaire** : Le code a été structuré de manière modulaire, avec des fichiers réutilisables (navbar, footer, header) pour les composants de l'interface, ce qui a facilité la maintenance et l'évolution du projet.
- **Responsive Design** : Grâce à Bootstrap et des styles CSS personnalisés, l'application est entièrement responsive, s'adaptant aux différents types d'appareils.
- **Sécurité** : La sécurité de l'application a été renforcée par des pratiques comme la validation des données, la protection CSRF, gestion des sessions par un rôle défini et par une session auto-détruite au bout de 30 min d'inactivité.

Collaboration et gestion de projet

- **GitHub pour le contrôle de version** : GitHub a été utilisé pour héberger le code et assurer le suivi des versions, facilitant la gestion du projet.

DOSSIER PROFESSIONNEL (DP)

- **Documentations et commentaires** : Une documentation détaillée a été créée pour expliquer les choix de design, les configurations spécifiques, et les instructions d'utilisation, ce qui facilite la prise en main du projet par d'autres développeurs qui souhaiteraient fork le projet.

3. Avec qui avez-vous travaillé ?

J'ai travaillé seul sur ce projet, mais j'ai régulièrement échangé avec un camarade de formation pour partager des idées et des solutions techniques. Ces échanges ont été précieux pour affiner mes compétences et améliorer le projet.

4. Contexte

Nom de l'entreprise, organisme ou association STUDI



Chantier, atelier, service ▶ Pendant la formation

Période d'exercice ▶ Du : 05/05/2024 au : 27/08/2024

5. Informations complémentaires (facultatif)

Le développement du projet **E-learning Abdu** a été un gros défi, de loin mon projet le plus sophistiqué, m'obligeant à jongler entre mes responsabilités personnelles et professionnelles. Malgré les difficultés, j'ai persévéré et j'ai pu acquérir des compétences solides en développement web, notamment en PHP, JavaScript, et la gestion des bases de données. Ce projet a marqué une étape importante dans mon parcours, me permettant de mettre en pratique mes connaissances et de développer une application complète et sécurisée.

Activité-type 2

Développer la partie back-end d'une application web ou web mobile sécurisée

Exemple n° 1 ► Tout pour un nouveau application web back-end PHP

- Mettre en place une base de données relationnelle
- Développer des composants d'accès aux données SQL et NoSQL
- Développer des composants métier coté serveur
- Documenter le déploiement d'une application dynamique web ou web mobile

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

I. Mise en place d'une base de données relationnelle

Pour la réalisation de mon projet, j'ai pris en charge l'intégralité de la conception et de la mise en place de la base de données relationnelle en utilisant MariaDB comme système de gestion de base de données. Le choix de MariaDB s'est imposé par sa compatibilité avec MySQL, sa robustesse, et ses performances accrues, qui répondent parfaitement aux exigences d'une application web dynamique nécessitant une grande fiabilité et une gestion efficace des transactions.

1. Création et configuration de la base de données

La première étape que j'ai entreprise a consisté à créer la base de données elle-même. J'ai opté pour le jeu de caractères utf8mb4 et la collation utf8mb4_general_ci, car ils offrent une prise en charge étendue des caractères, incluant les emojis et autres caractères spéciaux, ce qui est essentiel pour une application moderne où l'expérience utilisateur est au centre des préoccupations.

```
CREATE DATABASE IF NOT EXISTS `toupourunnouveaune`  
DEFAULT CHARACTER SET utf8mb4  
COLLATE utf8mb4_general_ci;
```

2. Conception des tables

J'ai ensuite procédé à la conception des tables en tenant compte des besoins fonctionnels du système. Chaque table a été soigneusement structurée pour représenter fidèlement les entités du système, telles que les utilisateurs, les rôles, les amis, les commentaires, les recettes, et bien d'autres. Pour chaque table, j'ai défini des colonnes avec des types de données adaptés, ainsi que des contraintes pour garantir l'intégrité et la cohérence des données.

DOSSIER PROFESSIONNEL (DP)

Exemple : Table utilisateurs

La table utilisateurs est un élément central du système, stockant les informations de chaque utilisateur enregistré. J'ai veillé à inclure un identifiant unique (id), un nom d'utilisateur (nom_utilisateur), un email, un mot de passe, ainsi qu'une référence à leur rôle (role_id). Cette dernière colonne est une clé étrangère pointant vers la table roles, ce qui permet de structurer les permissions des utilisateurs en fonction de leur rôle dans le système.

```
CREATE TABLE `utilisateurs` (  
  `id` int(11) NOT NULL AUTO_INCREMENT,  
  `nom_utilisateur` varchar(50) NOT NULL,  
  `email` varchar(100) NOT NULL,  
  `mot_de_passe` varchar(255) NOT NULL,  
  `date_creation` timestamp NOT NULL DEFAULT current_timestamp(),  
  `date_mise_a_jour` timestamp NOT NULL DEFAULT current_timestamp() ON UPDATE  
current_timestamp(),  
  `role_id` int(11) DEFAULT NULL,  
  PRIMARY KEY (`id`),  
  UNIQUE KEY `fk_name` (`nom_utilisateur`),  
  KEY `fk_role` (`role_id`),  
  CONSTRAINT `fk_role` FOREIGN KEY (`role_id`) REFERENCES `roles` (`id`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

3. [Mise en place des relations entre les tables](#)

La puissance d'une base de données relationnelle réside dans la capacité à interconnecter les données de différentes tables de manière cohérente et performante. J'ai mis en place des relations robustes entre les tables, en utilisant des clés étrangères pour assurer l'intégrité référentielle. Ces relations permettent de structurer les données de manière à refléter les relations du monde réel entre les entités.

Exemple : Relation entre les tables questions et quizzes

Par exemple, la table questions est liée à la table quizzes par une clé étrangère quiz_id. Cela permet de regrouper les questions en fonction de leur quiz respectif. J'ai également ajouté la contrainte ON DELETE CASCADE pour que la suppression d'un quiz entraîne automatiquement la suppression de toutes les questions qui lui sont associées, ce qui simplifie la gestion des données et évite les incohérences.

```
CREATE TABLE `questions` (  
  `id` int(11) NOT NULL AUTO_INCREMENT,  
  `quiz_id` int(11) DEFAULT NULL,  
  `question_text` text DEFAULT NULL,
```

```
PRIMARY KEY (`id`),  
KEY `quiz_id` (`quiz_id`),  
CONSTRAINT `questions_ibfk_1` FOREIGN KEY (`quiz_id`)  
REFERENCES `quizzes` (`id`) ON DELETE CASCADE  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

En tant que développeur aspirant, j'ai veillé à ce que chaque aspect de la base de données soit optimisé pour la performance et la sécurité, tout en respectant les meilleures pratiques de conception de bases de données relationnelles. Cette approche me permet d'assurer que les opérations sur les données, qu'elles soient simples ou complexes, sont exécutées de manière efficace et fiable.

Développement des composants d'accès aux données SQL et NoSQL

Dans le cadre du projet, j'ai été chargé de développer des composants d'accès aux données en utilisant à la fois des bases de données SQL et NoSQL. Cette dualité de stockage de données m'a permis de tirer parti des avantages spécifiques de chaque type de base de données, tout en assurant une interaction fluide et cohérente entre elles.

1. Accès aux données SQL

Pour l'accès aux données SQL, j'ai utilisé PDO pour interagir avec une base de données MySQL (MariaDB). L'utilisation de PDO a été privilégiée pour sa capacité à supporter plusieurs bases de données, mais surtout pour les fonctionnalités de préparation de requêtes qui permettent de prévenir les attaques par injection SQL. J'ai structuré le code pour que chaque interaction avec la base de données soit encapsulée dans des classes modèles dédiées, ce qui facilite la maintenance et améliore la lisibilité du code.

Exemple : Classe Thread pour la gestion des threads

La classe Thread est un exemple typique de la manière dont j'ai structuré l'accès aux données SQL. Elle gère les opérations CRUD (Create, Read, Update, Delete) sur les threads du forum. Voici un extrait de cette classe :

Voir l'exemple de code dans les annexes – Page 71

Cette classe montre comment j'ai utilisé les requêtes préparées pour extraire les données des threads et les joindre aux informations des utilisateurs dans la base de données SQL. L'intégration de méthodes spécifiques pour récupérer les threads, soit par ID, soit par nombre limité, permet une flexibilité et une réutilisabilité accrues dans différentes parties de l'application.

2. Intégration avec une base de données NoSQL (MongoDB)

En parallèle de la base de données relationnelle, j'ai implémenté une solution NoSQL en utilisant MongoDB pour gérer certaines fonctionnalités spécifiques, comme le suivi des scores des quiz et le suivi des vues sur les threads du forum. L'utilisation de MongoDB s'est avérée particulièrement avantageuse pour ces cas d'utilisation où la structure des données est plus flexible et les opérations de lecture/écriture sont intensives.

Exemple : Classe MongoDB pour la gestion des scores

La classe MongoDB illustre comment j'ai structuré l'accès aux collections MongoDB pour gérer les scores des utilisateurs dans les quiz. Voici un extrait de cette classe :

Voir l'exemple de code dans les annexes – Page 72

Dans cet exemple, j'ai utilisé le pilote officiel MongoDB pour PHP, ce qui permet une intégration directe avec les services MongoDB hébergés sur le cloud. La méthode getScoresParents récupère et trie les scores des utilisateurs pour les afficher sur l'interface de l'application. J'ai veillé à ce que toutes les exceptions soient correctement gérées et consignées, garantissant ainsi une robustesse dans les opérations de lecture/écriture.

3. Utilisation combinée des bases de données SQL et NoSQL

Un aspect clé de ce projet a été l'intégration harmonieuse des données issues de deux types de bases de données distincts. Par exemple, lors de la gestion des threads du forum, j'ai utilisé MySQL pour stocker les informations principales du thread et MongoDB pour suivre les vues de chaque thread. Cela m'a permis de bénéficier des avantages des deux systèmes : la robustesse des transactions SQL et la flexibilité des opérations NoSQL.

Exemple d'utilisation combinée : Mise à jour des vues d'un thread

```
$viewsCollection = $mongoClient->getCollection('views');
$viewsCollection->updateOne(
    ['thread_id' => $threadId],
    ['$inc' => ['views' => 1]],
    ['upsert' => true]
);
```

Cette méthode permet d'incrémenter le compteur de vues pour un thread spécifique dans MongoDB chaque fois qu'un utilisateur accède à ce thread. L'approche upsert est utilisée ici pour créer un nouveau document si le thread n'existe pas encore dans la collection views, ou pour mettre à jour le document existant.

En conclusion, la combinaison des bases de données SQL et NoSQL m'a permis de concevoir un

système flexible et performant, capable de répondre à divers besoins fonctionnels tout en maintenant une intégrité et une sécurité des données de haut niveau. Cette approche hybride assure une meilleure performance et une capacité d'évolution pour les fonctionnalités futures de l'application.

Développer des composants métier côté serveur

Le développement des composants métier côté serveur est une étape cruciale qui lie la logique de l'application aux données stockées et aux interactions avec les utilisateurs. Dans le cadre de ce projet, j'ai mis en place divers composants métier pour gérer des fonctionnalités complexes, comme la gestion des utilisateurs, les forums, les quiz, et les avis médicaux. Ces composants sont au cœur du fonctionnement de l'application et permettent de transformer les données brutes en informations pertinentes et exploitables pour l'utilisateur final.

1. Gestion des utilisateurs et des rôles

La gestion des utilisateurs est une composante centrale de l'application, permettant de gérer l'inscription, l'authentification, et l'autorisation des utilisateurs. J'ai implémenté un système de rôles pour différencier les droits d'accès entre les administrateurs, les médecins, et les parents.

Exemple : Modèle User avec gestion des rôles

Voir l'extrait de code pour le modèle User – page 73

Dans cet exemple, la classe User gère la création et la récupération des utilisateurs, en tenant compte des rôles qui leur sont attribués. La méthode **getUsernames** permet de récupérer les noms d'utilisateurs à partir de leurs identifiants, facilitant ainsi l'affichage des informations dans différentes parties de l'application.

2. Gestion des threads de forum et des interactions

Les forums sont une partie essentielle de l'application, permettant aux utilisateurs de créer des discussions, de répondre à des questions, et d'interagir les uns avec les autres. J'ai développé un composant métier qui gère toutes les opérations liées aux threads du forum, incluant la création, la mise à jour, la suppression, ainsi que le suivi des interactions via MongoDB pour le comptage des vues.

Exemple : Gestion des threads avec mise à jour des vues

Voir l'extrait de code pour cette classe – page 71

La méthode **incrementViews** montre comment j'ai utilisé MongoDB pour suivre les vues d'un thread de manière efficace. Cette approche garantit que chaque visite sur un thread est correctement enregistrée, ce qui permet de mieux comprendre l'engagement des utilisateurs avec les discussions.

3. [Gestion des scores des quiz](#)

Un autre composant métier important est la gestion des scores des quiz. Ce composant prend en charge le calcul des scores, leur enregistrement dans MongoDB, et leur affichage sur le tableau des scores. J'ai conçu ce composant pour qu'il soit à la fois performant et extensible, permettant une personnalisation facile des quiz et du système de score.

[Exemple : Soumission des scores des quiz \(voir annexe car code trop long\)](#)

Le code en question se trouve dans les annexes annotées : Soumission score des quiz TPUNN.

Ce code montre comment les scores des utilisateurs sont calculés et mis à jour dans MongoDB. En utilisant les fonctionnalités avancées de MongoDB, telles que l'upsert et la mise à jour des sous-documents, j'ai pu concevoir un système de gestion des scores flexible et évolutif.

4. [Conclusion](#)

Le développement des composants métier côté serveur a exigé une approche rigoureuse pour garantir que toutes les opérations de l'application soient effectuées de manière sécurisée, fiable, et efficace. J'ai veillé à ce que chaque composant soit conçu pour être réutilisable et extensible, afin de faciliter la maintenance et l'évolution future du projet. Les solutions que j'ai mises en place assurent non seulement la robustesse de l'application, mais aussi une expérience utilisateur optimale, en combinant les forces des bases de données SQL et NoSQL.

Documenter le déploiement d'une application dynamique web ou web mobile

Le déploiement de l'application sur Heroku a impliqué l'intégration d'une base de données relationnelle MySQL via l'add-on StackHero for MySQL et l'utilisation de MongoDB Atlas pour la gestion des données non relationnelles. Voici un aperçu des étapes clés du déploiement et des configurations réalisées pour assurer un fonctionnement optimal de l'application.

5. Déploiement sur Heroku

J'ai choisi Heroku comme plateforme de déploiement pour son intégration facile avec les services de bases de données et sa capacité à gérer le scaling de manière fluide.

B. Étapes principales :

1. **Création de l'application sur Heroku :** Pour commencer, j'ai créé une nouvelle application sur Heroku afin de disposer d'une infrastructure de déploiement cloud scalable.

heroku create mon-application

2. **Ajout de l'add-on StackHero for MySQL :** J'ai intégré l'add-on StackHero for MySQL pour gérer les données structurées de l'application.

heroku addons:create stackhero-mysql

3. **Configuration de MongoDB Atlas :** MongoDB Atlas a été utilisé pour stocker des données non structurées telles que les scores des utilisateurs et les vues des threads du forum. J'ai configuré MongoDB Atlas avec une connexion sécurisée et j'ai intégré MongoDB à l'application via le SDK officiel de MongoDB en PHP.

Exemple de configuration MongoDB :

```
require __DIR__ . '/../vendor/autoload.php';
```

```
use MongoDB\Client;
```

```
class MongoDB {
```

```
    private $mongoClient;
```

```
    private $mongoCollection;
```

```
    public function __construct() {
```

```
        $uri = 'mongodb+srv://<username>:<password>@cluster.mongodb.net';
```

```
        $databaseName = 'tpunn_quizz_score';
```

```
        try {
```

```
            $this->mongoClient = new Client($uri);
```

```
            $this->mongoCollection = $this->mongoClient->selectDatabase($databaseName)->scores;
```

```
} catch (Exception $erreur) {  
    error_log("Erreur de connexion à MongoDB : " . $erreur->getMessage());  
    throw new Exception("Impossible de se connecter à la base de données MongoDB");  
}  
}  
  
public function getCollection($collectionName) {  
    return $this->mongoClient->selectDatabase('tpunn_quizz_score')->selectCollection($collectionName);  
}  
}
```

4. **Déploiement du code source** : Une fois les services de base de données configurés, j'ai déployé l'application sur Heroku via Git.

```
git push heroku main
```

6. Gestion des bases de données

Dans ce projet, j'ai utilisé deux types de bases de données pour répondre aux différents besoins de l'application :

- **MySQL** : Géré par l'add-on StackHero for MySQL, MySQL stocke les données structurées telles que les informations des utilisateurs, les rôles, et les forums de discussion.
- **MongoDB Atlas** : MongoDB est utilisé pour gérer des données non structurées, telles que les scores des utilisateurs et les statistiques de vues des threads. MongoDB Atlas, étant une solution cloud, assure une haute disponibilité et une sécurité robuste.

Utilisation de MongoDB dans le projet :

```
$mongoClient = new MongoDB();
```

```
$collection = $mongoClient->getCollection('scores');
```

7. Configuration SSL et SSH

Afin de sécuriser les communications entre le client et le serveur ainsi que les connexions SSH, j'ai mis en place les configurations suivantes :

SSL sur Heroku : Heroku permet d'activer le SSL pour sécuriser les communications entre le serveur et les clients. J'ai utilisé l'option "SSL automatique" de Heroku pour gérer les certificats SSL.

```
heroku certs:auto:enable
```

Accès SSH sécurisé : Pour les opérations nécessitant un accès direct au serveur ou aux bases de

DOSSIER PROFESSIONNEL (DP)

données, j'ai configuré un accès SSH sécurisé. Cela permet de se connecter aux services cloud de manière sécurisée.

- Génération de la clé SSH :

```
ssh-keygen -t rsa -b 4096 -C "your_email@example.com"
```

- Ajout de la clé SSH à Heroku :

```
heroku keys:add
```

En résumé, le déploiement sur Heroku, associé à l'intégration de StackHero for MySQL et MongoDB Atlas, assure un environnement de production robuste, sécurisé et évolutif pour l'application. Les configurations de SSL et SSH garantissent la sécurité des données en transit et l'accès sécurisé aux ressources de l'application.

2. Précisez les moyens utilisés :

Pour la réalisation de ce projet, j'ai utilisé une combinaison de technologies, de services cloud, d'outils de développement et de méthodologies afin d'assurer un déploiement sécurisé, évolutif et performant de l'application. Voici les moyens que j'ai employés :

I. Technologies et Langages

- **PHP** : Le langage principal utilisé pour développer l'ensemble de l'application côté serveur, y compris la logique métier, l'interaction avec la base de données, et la gestion des sessions utilisateur.
- **HTML/CSS/JavaScript** : Pour la création de l'interface utilisateur dynamique et réactive. J'ai utilisé Bootstrap pour garantir un design responsive, et jQuery pour la manipulation DOM et l'Ajax.
- **MySQL** : J'ai utilisé MySQL comme base de données relationnelle pour stocker les informations structurées telles que les utilisateurs, les rôles, les posts du forum, etc. StackHero for MySQL a été utilisé pour gérer cette base de données sur Heroku.
- **MongoDB Atlas** : MongoDB a été choisi pour stocker les données non structurées, telles que les scores des utilisateurs et les statistiques de vues des threads. MongoDB Atlas a permis d'héberger cette base de données NoSQL dans un environnement cloud sécurisé. J'ai utilisé des *cursors* pour parcourir et manipuler efficacement les ensembles de données stockés dans MongoDB.
- **Composer** : Outil de gestion des dépendances PHP, utilisé pour installer et gérer les bibliothèques nécessaires au projet, comme le SDK MongoDB et d'autres packages essentiels.

Services Cloud et Outils de Déploiement

- **Heroku** : La plateforme de déploiement choisie pour héberger l'application. Heroku offre une gestion simplifiée des environnements de production, des bases de données et des certificats SSL.

DOSSIER PROFESSIONNEL (DP)

L'application a été déployée directement depuis le dépôt Git via l'intégration continue avec Heroku.

- **StackHero for MySQL** : Add-on utilisé sur Heroku pour gérer la base de données MySQL. Ce service assure une haute disponibilité, des backups automatiques et une gestion simplifiée des performances de la base de données.
- **MongoDB Atlas** : Service cloud utilisé pour héberger la base de données MongoDB, avec des fonctionnalités avancées de sécurité, de redondance et de gestion des performances.

Outils de Développement et Collaboration

- **Git & GitHub** : Git a été utilisé pour la gestion de version du code source, permettant une collaboration efficace et un suivi précis des modifications apportées au projet. GitHub a servi de dépôt centralisé pour le code, facilitant le travail en équipe et les revues de code.
- **Cursor** : J'ai utilisé l'éditeur de code Cursor pour écrire, déboguer et gérer le code de l'application, profitant de ses fonctionnalités avancées pour accélérer le développement.

Sécurité et Gestion des Accès

- **SSL/TLS** : Pour sécuriser les communications entre les clients et le serveur, j'ai activé les certificats SSL sur Heroku. Cela assure que toutes les données transmises sont cryptées.
- **SSH** : L'accès sécurisé aux ressources du serveur et à la base de données a été géré via SSH, permettant une administration sécurisée de l'application en production.
- **Gestion des tokens CSRF** : Pour protéger contre les attaques CSRF, des tokens sont générés et validés pour chaque action sensible dans l'application, garantissant ainsi que seules les requêtes légitimes sont traitées.

Méthodologies de Développement

- **Développement Agile** : J'ai adopté une approche agile pour le développement de ce projet, avec des itérations régulières et des feedbacks continus pour affiner les fonctionnalités et corriger les problèmes au fur et à mesure.
- **Test-Driven Development (TDD)** : J'ai utilisé TDD pour certaines parties critiques du code, écrivant d'abord les tests avant de développer les fonctionnalités, ce qui a permis de réduire les bugs et d'assurer une meilleure qualité du code.

En résumé, j'ai utilisé un ensemble intégré de technologies, de services cloud, d'outils de développement et de méthodologies pour développer, déployer et gérer cette application web, assurant à la fois sa performance, sa sécurité, et sa facilité d'utilisation.

DOSSIER PROFESSIONNEL (DP)

3. Avec qui avez-vous travaillé ?

Seul ou en équipe (avec un élève de la même formation que moi, pour s'expliquer ce que nous avons appris et nous nous sommes montré aussi nos projets afin d'apprendre l'un de l'autre. Souvent nous étions en session live Google Meet pour se parler.

4. Contexte

Nom de l'entreprise, organisme ou association STUDI



Chantier, atelier, service Pendant la formation

Période d'exercice Du : 02/05/2024 au : 26/08/2024

5. Informations complémentaires (facultatif)

Activité-type 2

Développer la partie back-end d'une application web ou web mobile sécurisée

Exemple n° 2 ▶ E-learning Abdu application web back-end PHP utilisation MVC

- Mettre en place une base de données relationnelle
- Développer des composants d'accès aux données SQL et NoSQL
- Développer des composants métier coté serveur
- Documenter le déploiement d'une application dynamique web ou web mobile

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

I. Mise en place de la base de données relationnelle

Dans le cadre du projet E-learning Abdu, la conception et l'implémentation de la base de données relationnelle ont joué un rôle fondamental dans la structuration et la gestion des données. Le choix d'une base de données relationnelle, en l'occurrence MariaDB, a été motivé par la nécessité de gérer des données structurées avec des relations complexes et une forte intégrité des données. Voici une présentation technique des étapes suivies pour mettre en place cette base de données.

1. Conception du schéma de la base de données

Le processus de conception a débuté par la modélisation conceptuelle des données, suivie de la transformation en modèle logique, où chaque entité a été traduite en tables SQL. Le modèle logique a été normalisé pour éviter les redondances et garantir la cohérence des données. Le schéma relationnel final a été implémenté avec des tables respectant les meilleures pratiques en matière de gestion des types de données, d'indexation et de contraintes.

Exemple de création de la table users avec des contraintes et des index :

```
CREATE TABLE `users` (  
  `id` int(11) NOT NULL AUTO_INCREMENT,  
  `username` varchar(50) NOT NULL,  
  `email` varchar(100) NOT NULL,  
  `password` varchar(255) NOT NULL,  
  `role_id` int(11) NOT NULL,  
  `cursus_valide` tinyint(1) DEFAULT 0,  
  `certificate_issued` tinyint(1) DEFAULT 0,  
  `created_at` timestamp NOT NULL DEFAULT current_timestamp(),
```

```
PRIMARY KEY (`id`),  
UNIQUE KEY `email` (`email`),  
KEY `role_id` (`role_id`),  
CONSTRAINT `users_ibfk_1` FOREIGN KEY (`role_id`) REFERENCES `roles` (`id`) ON DELETE CASCADE  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

- **AUTO_INCREMENT** : Utilisé pour le champ id afin de générer automatiquement des identifiants uniques.
- **UNIQUE KEY** : Un index unique a été ajouté sur le champ email pour garantir que chaque adresse email dans la base de données est unique.
- **FOREIGN KEY** : La clé étrangère role_id est utilisée pour établir une relation entre les utilisateurs et leurs rôles, en garantissant l'intégrité référentielle via la clause ON DELETE CASCADE.

2. Implémentation des relations entre les tables

Les relations entre les tables ont été implémentées en utilisant des clés étrangères pour établir des liens solides entre les entités, conformément aux principes de la 3e forme normale (3NF). Cela assure non seulement la cohérence des données mais également leur intégrité lors des opérations de mise à jour et de suppression.

Exemple de création de relations :

```
ALTER TABLE `users`  
ADD CONSTRAINT `users_ibfk_1` FOREIGN KEY (`role_id`) REFERENCES `roles` (`id`) ON DELETE CASCADE;  
ALTER TABLE `categories`  
ADD CONSTRAINT `fk_categories_formation_id` FOREIGN KEY (`formation_id`) REFERENCES `formations` (`id`) ON DELETE CASCADE;
```

- **ON DELETE CASCADE** : Cette clause garantit que la suppression d'une entrée dans la table parent entraîne automatiquement la suppression des enregistrements associés dans la table enfant. Cela prévient les orphelins dans la base de données.

3. Gestion des performances et optimisation

L'indexation a été soigneusement planifiée pour améliorer les performances des requêtes, notamment sur des colonnes fréquemment utilisées dans les clauses WHERE et les jointures. Des index ont été ajoutés non seulement sur les clés primaires et étrangères, mais également sur des colonnes critiques comme email et role_id.

Exemple de création d'index :

```
ALTER TABLE `users`  
ADD UNIQUE KEY `email` (`email`),  
ADD KEY `role_id` (`role_id`);
```

Ces index permettent d'accélérer les recherches et les jointures, tout en garantissant que l'intégrité des données n'est pas compromise.

4. Scripts d'initialisation et migrations

Pour automatiser le déploiement de la base de données, des scripts SQL d'initialisation ont été créés. Ces scripts sont utilisés pour créer les tables, définir les contraintes, et peupler la base de données avec des données initiales. Ils sont également utilisés dans le cadre des migrations pour mettre à jour la structure de la base de données en fonction des évolutions du modèle de données.

Exemple de script d'initialisation pour la table roles :

```
INSERT INTO `roles` (`id`, `role_name`) VALUES  
(1, 'Administrateur'),  
(2, 'Formateur'),  
(3, 'Apprenant');
```

Ces scripts assurent que la base de données est dans un état cohérent et prêt à l'emploi lors du déploiement en production.

Développement des Composants d'Accès aux Données SQL et NoSQL

Dans le cadre du développement du projet E-learning Abdu, j'ai conçu et mis en œuvre des composants robustes pour accéder aux données en utilisant une architecture basée sur le modèle MVC (Modèle-Vue-Contrôleur). Ces composants facilitent la gestion des données stockées dans une base de données relationnelle en utilisant des requêtes SQL, tout en permettant l'intégration avec une base de données NoSQL pour des cas d'utilisation spécifiques. Cette approche garantit que les opérations sont réalisées de manière sécurisée, efficace, et évolutive.

1. Interaction entre le Modèle et le Contrôleur

Le modèle User et le contrôleur UserController illustrent parfaitement l'interaction entre ces deux couches. Le modèle User encapsule la logique d'accès aux données relationnelles, tandis que le contrôleur UserController orchestre les opérations de récupération, de modification et de suppression

des données en fonction des requêtes des utilisateurs.

Exemple de récupération de données :

Pour récupérer tous les utilisateurs, le contrôleur UserController appelle la méthode getAllUsers() du modèle User. Cette méthode exécute une requête SQL pour sélectionner toutes les lignes de la table users, et le résultat est renvoyé sous forme de tableau associatif.

```
public function getAllUsers()
{
    return $this->user->getAllUsers();
}
```

Dans le modèle User, la méthode getAllUsers() utilise la classe PDO pour préparer et exécuter la requête SQL :

```
public function getAllUsers() {
    $stmt = $this->conn->prepare("SELECT * FROM users");
    $stmt->execute();
    return $stmt->fetchAll(PDO::FETCH_ASSOC);
}
```

Exemple d'insertion de données :

Pour ajouter un nouvel utilisateur, le contrôleur UserController appelle la méthode addUser() du modèle. Le mot de passe de l'utilisateur est haché pour garantir la sécurité avant d'être stocké dans la base de données.

```
public function addUser($username, $email, $password, $role_id)
{
    $this->user->setName($username);
    $this->user->setEmail($email);
    $this->user->setPassword(password_hash($password, PASSWORD_DEFAULT));
    $this->user->setRoleId($role_id);

    return $this->user->addUser();
}
```

Dans le modèle User, la méthode addUser() construit une requête SQL INSERT pour insérer les données dans la table users :

```
public function addUser() {
    $query = "INSERT INTO users (username, email, password, role_id, created_at)
    VALUES (:username, :email, :password, :role_id, NOW())";
    $stmt = $this->conn->prepare($query);

    $stmt->bindParam(':username', $this->username);
```

```
$stmt->bindParam(':email', $this->email);
$stmt->bindParam(':password', $this->password);
$stmt->bindParam(':role_id', $this->role_id);
return $stmt->execute();
}
```

Exemple de mise à jour de données :

Pour mettre à jour les informations d'un utilisateur, le contrôleur utilise la méthode `updateUser()`, qui passe les nouvelles valeurs au modèle. Si un mot de passe est fourni, il est également mis à jour.

```
public function updateUser($id, $username, $email, $role_id, $password = null)
{
    $this->user->setId($id);
    $this->user->setName($username);
    $this->user->setEmail($email);
    $this->user->setRoleId($role_id);

    if (!empty($password)) {
        $hashedPassword = password_hash($password, PASSWORD_BCRYPT);
        $this->user->setPassword($hashedPassword);
    }

    return $this->user->updateUser();
}
```

Le modèle `User` gère ensuite l'exécution de la requête SQL `UPDATE`, en prenant en compte ou non le nouveau mot de passe en fonction de sa présence.

```
public function updateUser()
{
    $query = "UPDATE " . $this->table_name . " SET username = :username, email = :email, role_id = :role_id";
    if (!empty($this->password)) {
        $query .= ", password = :password"; }
    $query .= " WHERE id = :id";
    $stmt = $this->conn->prepare($query);
    $stmt->bindParam(':username', $this->username);
    $stmt->bindParam(':email', $this->email);
    $stmt->bindParam(':role_id', $this->role_id);
    $stmt->bindParam(':id', $this->id);

    if (!empty($this->password)) {
        $stmt->bindParam(':password', $this->password); }
    return $stmt->execute();
}
```


2. Sécurité et Validation

Toutes les interactions avec la base de données utilisent des requêtes préparées, ce qui protège contre les attaques par injection SQL. Les mots de passe sont systématiquement hachés avant d'être stockés, en utilisant l'algorithme bcrypt, pour garantir leur sécurité. Ces pratiques assurent que les données sensibles sont protégées et que le système reste résilient face aux menaces de sécurité courantes.

3. Intégration et Extensibilité pour NoSQL

En complément de la base de données relationnelle SQL, certaines fonctionnalités de l'application, telles que le suivi des vues de threads dans le forum de discussion, utilisent MongoDB, une base de données NoSQL. MongoDB est particulièrement adapté pour les opérations nécessitant une grande flexibilité de schéma et une scalabilité horizontale.

Exemple de mise à jour des vues de threads :

Pour chaque visite d'un thread, MongoDB est utilisé pour suivre le nombre de vues. Cela est réalisé en utilisant l'opération \$inc pour incrémenter le compteur de vues du thread :

```
$viewsCollection = $mongoClient->getCollection('views');  
$viewsCollection->updateOne(  
    ['thread_id' => $threadId],  
    ['$inc' => ['views' => 1]],  
    ['upsert' => true]  
);
```

Exemple de récupération des threads les plus actifs :

Les threads les plus actifs sont déterminés en fonction du nombre de vues stockées dans MongoDB :

```
$activeThreads = $viewsCollection->find([], ['sort' => ['views' => -1], 'limit' => 5])->toArray();
```

Cette approche hybride permet à l'application de tirer parti des avantages des deux types de bases de données. **MariaDB** est utilisé pour les opérations nécessitant des transactions complexes et une forte intégrité référentielle, tandis que **MongoDB** gère des opérations nécessitant une grande flexibilité de schéma et une rapidité d'exécution.

Développement des Composants Métier côté Serveur

Dans le cadre du projet E-learning Abdu, j'ai développé plusieurs composants métier côté serveur qui encapsulent les règles de gestion spécifiques au domaine de l'éducation en ligne. Ces composants sont cruciaux pour assurer la bonne gestion des processus comme l'attribution de certificats, la gestion des formations, la validation des parcours académiques, et l'évaluation des utilisateurs via des quiz. Voici un aperçu détaillé de ces composants :

1. Gestion de la Génération des Certificats

Le composant de génération de certificats est central pour valider et reconnaître les acquis des utilisateurs qui ont complété avec succès leur formation.

Exemple : Génération de Certificats

Le contrôleur CertificateController gère la génération des certificats pour les utilisateurs ayant validé leur cursus. Voici les étapes clés de ce processus :

- **Vérification de la Validation du Cursus** : Avant de générer un certificat, le contrôleur vérifie que l'utilisateur a validé son cursus. Cette validation est assurée par le modèle User, qui vérifie le statut de l'utilisateur dans la base de données.
- **Récupération des Détails de la Formation** : Une fois la validation confirmée, le contrôleur récupère les détails de la formation complétée par l'utilisateur en utilisant le modèle Formation.
- **Génération du Certificat en PDF** : Le certificat est généré sous forme de fichier PDF en utilisant la bibliothèque FPDF. Le document inclut les informations essentielles telles que le nom de l'utilisateur, le nom de la formation, et la date de validation.
- **Mise à Jour de la Base de Données** : Enfin, une fois le certificat généré, le système met à jour la base de données pour indiquer que le certificat a été délivré.

Exemple de code :

```
public function generateCertificate($userId)
{
    $userModel = new User($this->db);
    $user = $userModel->findById($userId);
    if (!$user || $user['cursus_valide'] != 1) {
        throw new \Exception("Cursus non validé ou utilisateur introuvable.");
    }
    $formationModel = new Formation($this->db);
    $formation = $formationModel->getFormationByUserId($userId);
    if (!$formation || !isset($formation['name'])) {
```

```
throw new \Exception("Formation introuvable ou nom de la formation non défini.");
}
$pdf = new FPDF();
$pdf->AddPage();
$this->drawBorder($pdf);
$pdf->SetFont('Arial', 'B', 24);
$pdf->SetTextColor(50, 50, 100); // Couleur bleu foncé
$pdf->Cell(0, 20, $this->convertToUtf8('CERTIFICAT D\'ACHÈVEMENT'), 0, 1, 'C');
$pdf->Ln(10);
// ... (suite de la génération du PDF)
$filename = $certificatesDir . 'certificat_' . $userId . '.pdf';
$pdf->Output('F', $filename);
$userModel->updateCertificateIssued($userId);
return $filename;
}
```

2. Gestion des Formations

La gestion des formations est un autre aspect clé du projet, permettant l'administration des cours proposés, l'attribution de ces cours aux utilisateurs, et le suivi de leur progression.

Exemple : Gestion des Formations

Le contrôleur FormationController permet d'effectuer diverses opérations sur les formations :

- **Création et Mise à Jour des Formations** : Les méthodes createFormation() et updateFormation() permettent respectivement de créer une nouvelle formation et de mettre à jour une formation existante. Ces méthodes s'assurent que les informations fournies sont correctement formatées et valides avant d'effectuer les opérations en base de données.
- **Assignation des Formations aux Utilisateurs** : Le système permet d'assigner des formations à des étudiants via la méthode assignFormationToStudent(). Cette assignation crée une relation entre l'utilisateur et la formation dans la base de données, permettant de suivre leur progression.
- **Suivi de la Progression** : La méthode getStudentProgress() calcule le pourcentage de progression d'un étudiant dans une formation, en comparant le nombre de catégories de la formation complétées avec le nombre total de catégories.

Exemple de code :

```
public function assignFormationToStudent($userId, $formationId)
{
    return $this->formationModel->assignFormationToStudent($userId, $formationId);
}

public function getStudentProgress($userId, $formationId) {
    return $this->formationModel->getStudentProgress($userId, $formationId);
}
```

3. Gestion des Quiz et des Résultats

L'évaluation des utilisateurs via des quiz est essentielle pour mesurer leur compréhension des formations suivies. Le composant QuizController permet de gérer cette évaluation.

Exemple : Gestion des Quiz

Le composant gère la présentation des questions aux utilisateurs, la collecte de leurs réponses, et le calcul de leur score :

- **Présentation des Questions** : Les questions d'un quiz sont récupérées et affichées à l'utilisateur. Les réponses possibles sont également chargées dynamiquement.
- **Calcul des Scores** : Après soumission des réponses, le système compare les réponses fournies par l'utilisateur avec les réponses correctes stockées dans la base de données. Le score est ensuite calculé en fonction du nombre de réponses correctes.
- **Enregistrement des Résultats** : Le score de l'utilisateur est enregistré dans la base de données pour permettre un suivi des performances.

Exemple de code :

```
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $userAnswers = $_POST['answers'] ?? [];
    $totalQuestions = count($questions);
    $score = 0;
    foreach ($questions as $question) {
        $correctAnswers = [];
        $selectedAnswers = $userAnswers[$question['id']] ?? [];
        // (suite du calcul des scores)
        if ($correctSelected && $incorrectSelected) {
            $score++;
        }
    }
}
```

```
}  
// Calcul du pourcentage de score  
$scorePercentage = ($score / $totalQuestions) * 100;  
$quizController->saveUserScore($quizId, $_SESSION['user']['id'], $scorePercentage);  
}
```

4. Conclusion

Ces composants métier sont conçus pour répondre aux exigences spécifiques du projet E-learning Abdu, en mettant l'accent sur la sécurité, la fiabilité, et la scalabilité. Le modèle MVC adopté permet une séparation claire des responsabilités, facilitant ainsi la maintenance et l'extensibilité du code. Que ce soit pour la gestion des certificats, des formations, ou des quiz, chaque composant assure que les opérations métier sont effectuées de manière cohérente et efficace, tout en garantissant une expérience utilisateur fluide et sécurisée.

Documentation de Déploiement de l'Application E-learning Abdu

Cette documentation décrit les étapes que j'ai suivies pour déployer l'application E-learning Abdu sur Heroku, en utilisant StackHero pour MySQL et MongoDB Atlas pour la gestion des bases de données NoSQL. Toutes les configurations sont gérées directement dans le code et via les variables d'environnement de Heroku.

1. Prérequis

- Compte Heroku
- Compte MongoDB Atlas
- Compte StackHero pour MySQL
- Application E-learning prête à être déployée
- Git installé sur ma machine

2. Préparation du Projet

A. Structure de l'Application :

- À la racine du projet, j'ai un fichier index.php qui redirige vers index.html dans le dossier auth situé dans app/.
- Tous les fichiers nécessaires, y compris le fichier composer.json, les fichiers PHP pour l'accès aux bases de données, et les dossiers public, app, vendor, etc., sont inclus dans le

dépôt Git.

B. Configuration des Bases de Données :

- Pour StackHero MySQL : Les informations de connexion sont stockées directement dans le code PHP (Database.php).
- Pour MongoDB Atlas : Les informations de connexion sont également codées en dur dans le fichier de configuration PHP (MongoDB.php).

3. Configuration de MongoDB Atlas

A. Création d'un Cluster MongoDB Atlas :

- Je me suis connecté à MongoDB Atlas et j'ai créé un nouveau cluster.
- J'ai créé un utilisateur avec les permissions adéquates pour accéder à la base de données.
- J'ai noté l'URI de connexion à utiliser dans l'application, par exemple :

```
mongodb+srv://AbduUSDI:heroku123456@abdurahmanusdi.lc9y4uk.mongodb.net
```

B. Configuration du Réseau :

- J'ai autorisé les adresses IP de mon application Heroku à accéder à MongoDB Atlas en ajoutant l'IP de Heroku dans les règles de sécurité du réseau.

4. Création de l'Application sur Heroku

A. Créer une Nouvelle Application :

- Je me suis connecté à Heroku et j'ai créé une nouvelle application en spécifiant un nom unique.

B. Ajout de l'Add-on StackHero pour MySQL :

- Dans le tableau de bord de l'application Heroku, je suis allé dans l'onglet "Resources".
- Dans la section "Add-ons", j'ai recherché "StackHero for MySQL" et je l'ai ajouté à mon application.
- Les informations de connexion à MySQL ont été automatiquement ajoutées en tant que variables d'environnement dans Heroku, mais dans ce cas, je les ai codées en dur dans

Database.php.

5. [Adaptation de la Configuration de l'Application](#)

A. [Configuration de la Base de Données MySQL :](#)

- J'ai modifié le fichier Database.php pour qu'il utilise les informations de connexion fournies par StackHero :

```
<?php
namespace App\Config;

use PDO;
use PDOException;
use Exception;
class Database {
    private $hote = 'cky0ko.stackhero-network.com';
    private $nom_base_de_donnees = 'e_learning';
    private $identifiant = 'root';
    private $mot_de_passe = 'a1QDfFRVtBpNgZvJB8cy0dtUllIKXyOL';
    private $port = '5040';
    private $connexion;
    public function getConnection() {
        $this->connexion = null;
        try {
            $this->connexion = new PDO(
                "mysql:host=" . $this->hote . ";port=" . $this->port . ";dbname=" . $this-
                >nom_base_de_donnees,
                $this->identifiant,
                $this->mot_de_passe
            );
            $this->connexion->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        } catch(PDOException $erreur) {
            // Logger l'erreur et lancer une exception personnalisée
            error_log("Erreur de connexion à la base de données : " . $erreur->getMessage());
            throw new Exception("Impossible de se connecter à la base de données");
        }
        return $this->connexion;
    }
}
```

B. Configuration de la Base de Données MongoDB :

- J'ai configuré le fichier MongoDB.php pour établir une connexion avec MongoDB Atlas :

```
<?php
namespace App\Config;
use MongoDB\Client;
use Exception;
class MongoDB {
    private $mongoClient;
    private $mongoCollection;
    public function __construct() {
        $uri = 'mongodb+srv://AbduUSDl:heroku123456@abdurahmanusdi.lc9y4uk.mongodb.net';
        $databaseName = 'e-learning';
        try {
            $this->mongoClient = new Client($uri);
            $this->mongoCollection = $this->mongoClient->selectDatabase($databaseName)->progression;
        } catch (Exception $erreur) {
            error_log("Erreur de connexion à MongoDB : " . $erreur->getMessage());
            throw new Exception("Impossible de se connecter à la base de données MongoDB");
        }
    }
    public function getCollection($collectionName) {
        return $this->mongoClient->selectDatabase('e-learning')->selectCollection($collectionName);
    }
}
```

6. Déploiement de l'Application

A. Initialisation du Dépôt Git :

- J'ai initialisé un dépôt Git dans le répertoire de mon projet :

```
git init
git add .
git commit -m "Initial commit"
```

B. Connexion à Heroku via Git :

- J'ai connecté mon projet à Heroku en utilisant la commande suivante :

```
heroku git:remote -a <nom-de-votre-application-heroku>
```


C. Déploiement de l'Application :

- J'ai déployé l'application sur Heroku en utilisant la commande :

```
git push heroku master
```

2. Précisez les moyens utilisés :

Pour mener à bien le déploiement de l'application, j'ai mis en œuvre une série de moyens techniques et organisationnels afin de garantir la sécurité, la performance, et la stabilité de l'ensemble du système.

I. Environnement de Développement et de Production :

- **Environnement de Développement** : J'ai commencé par développer et tester l'application dans un environnement local, en utilisant XAMPP sur mon poste de travail. Cette phase m'a permis de valider les fonctionnalités de base et d'identifier d'éventuels bugs avant de procéder au déploiement sur l'environnement de production.
- **Environnement de Production** : Une fois les tests locaux effectués, j'ai déployé l'application sur un serveur distant dédié, configuré pour répondre aux exigences de production. J'ai veillé à adapter les configurations en fonction des particularités de cet environnement, notamment en ce qui concerne la gestion des accès et la sécurité.

Gestion de la Base de Données :

- **Connexion Sécurisée au Serveur MySQL** : Pour la base de données, hébergée sur un serveur MySQL distant, j'ai mis en place une connexion sécurisée via SSL/TLS. J'ai configuré le Data Object (PDO) dans mon application pour utiliser ces connexions sécurisées, ce qui implique l'intégration de certificats SSL dans la configuration de PDO, assurant ainsi la confidentialité et l'intégrité des données échangées.
- **Paramétrage de la Connexion** : J'ai spécifiquement configuré les options PDO pour qu'elles utilisent le transport sécurisé requis par le serveur MySQL, en veillant à ce que chaque connexion soit authentifiée et cryptée.

Sécurisation des Transports :

- **SSL/TLS** : Étant donné que le serveur MySQL est configuré avec l'option `require_secure_transport=ON`, j'ai utilisé des certificats SSL pour établir des connexions sécurisées. Cette démarche a été essentielle pour respecter les normes de sécurité exigées par l'infrastructure du serveur.
- **Configuration du Pare-feu** : J'ai collaboré avec l'équipe de gestion des systèmes pour m'assurer que les règles de pare-feu étaient correctement configurées, limitant l'accès au serveur MySQL uniquement à partir d'adresses IP autorisées. Cela a renforcé la sécurité en empêchant les connexions non autorisées.

Collaboration avec l'Administration Systèmes :

- **Support Technique et Assistance** : Comprenant que je ne disposais pas d'un accès direct aux fichiers de configuration du serveur, j'ai travaillé en étroite collaboration avec l'équipe d'administration des systèmes. J'ai coordonné avec eux pour obtenir les certificats SSL nécessaires et pour s'assurer que toutes les configurations côté serveur étaient en place pour permettre des connexions sécurisées.
- **Communication Efficace** : J'ai maintenu une communication régulière avec l'équipe technique pour résoudre rapidement tout problème lié à la configuration ou à l'accès, ce qui a permis de fluidifier le processus de déploiement.

Surveillance et Tests :

- **Tests de Connexion** : Avant la mise en production complète, j'ai déployé un script minimal pour tester les connexions sécurisées à la base de données. Cela m'a permis de valider la configuration SSL et de m'assurer que les connexions étaient établies sans erreurs.
- **Surveillance Post-Déploiement** : J'ai mis en place des outils de surveillance pour détecter et alerter en cas de problème de connectivité ou de sécurité. Ces outils m'ont permis de garantir que l'application fonctionne correctement en production et que les utilisateurs finaux n'étaient pas affectés par des interruptions.

Documentation et Continuité :

- **Documentation des Processus** : Tout au long du processus, j'ai documenté les configurations, les paramètres utilisés, et les étapes clés du déploiement. Cette

DOSSIER PROFESSIONNEL (DP)

documentation est essentielle pour assurer la continuité du projet, faciliter les futures mises à jour, et permettre à d'autres membres de l'équipe de comprendre les choix techniques effectués.

- **Plan de Maintenance** : J'ai également prévu des procédures de maintenance régulières pour m'assurer que les certificats SSL sont mis à jour à temps et que les configurations restent conformes aux exigences de sécurité.

En utilisant ce cadre, j'ai pu non seulement déployer l'application de manière sécurisée, mais aussi m'assurer qu'elle resterait stable et performante, répondant ainsi aux attentes des utilisateurs finaux.

3. Avec qui avez-vous travaillé ?

Seul ou en équipe (avec un élève de la même formation que moi, pour s'expliquer ce que nous avons appris et nous nous sommes montré aussi nos projets afin d'apprendre l'un de l'autre.

Souvent nous étions en session live Google Meet pour se parler.

4. Contexte

Nom de l'entreprise, organisme ou association STUDI

▶

Chantier, atelier, service ▶ Pendant la formation

Période d'exercice ▶ Du : 04/06/2024 au : **27/08/2024**

DOSSIER PROFESSIONNEL ^(DP)

5. Informations complémentaires *(facultatif)*

DOSSIER PROFESSIONNEL ^(DP)

Titres, diplômes, CQP, attestations de formation

(facultatif)

Intitulé	Autorité ou organisme	Date

DOSSIER PROFESSIONNEL ^(DP)

Déclaration sur l'honneur

Je soussigné(e) Abdurahman USDI ,
déclare sur l'honneur que les renseignements fournis dans ce dossier sont exacts et que je suis
l'auteur(e) des réalisations jointes.

Fait à Reims le 27/08/2024

pour faire valoir ce que de droit.

Signature :

DOSSIER PROFESSIONNEL ^(DP)

Documents illustrant la pratique professionnelle

(facultatif)

Intitulé

ANNEXES

(Si le RC le prévoit)

Ajoute dynamique des question et réponses – TPUNN – Exemple 1

```
document.querySelector('.add-question').addEventListener('click', function() {
  const questionTemplate = `
    <div class="question">
      <div class="form-group">
        <label>Question</label>
        <input
          type="text"
          class="form-control"
          name="questions[${questionIndex}][question_text]" required>
        </div>
        <div class="answersContainer">
          <div class="form-group answer">
            <label>Réponse</label>
            <input
              type="text"
              class="form-control"
              name="questions[${questionIndex}][answers][0][answer_text]" required>
            <label>Bonne réponse</label>
            <input type="checkbox" name="questions[${questionIndex}][answers][0][is_correct]"
              value="1">
            </div>
          </div>
          <button type="button" class="btn btn-info add-answer">Ajouter une réponse</button>
        </div>`;
    document.getElementById('questionsContainer').insertAdjacentHTML('beforeend',
questionTemplate);
    questionIndex++;
  });
```

```
document.getElementById('questionsContainer').addEventListener('click', function(e) {
  if (e.target.classList.contains('add-answer')) {
    const questionDiv = e.target.closest('.question');
    const answersContainer = questionDiv.querySelector('.answersContainer');
```


DOSSIER PROFESSIONNEL (DP)

```
const answerIndex = answersContainer.querySelectorAll('.answer').length;
const questionIndex = Array.from(document.querySelectorAll('.question')).indexOf(questionDiv);
const answerTemplate = `
  <div class="form-group answer">
    <label>Réponse</label>
    <input                                type="text"                                class="form-control"
name="questions[${questionIndex}][answers][${answerIndex}][answer_text]" required>
    <label>Bonne réponse</label>
    <input                                type="checkbox"
name="questions[${questionIndex}][answers][${answerIndex}][is_correct]" value="1">
  </div>`;
  answersContainer.insertAdjacentHTML('beforeend', answerTemplate);
}
});
```

Soumission score des quizz – TPUNN – Exemple 1

```
session_start();
require_once '../config/Database.php';
require_once '../config/MongoDB.php';
require_once '../models/QuizModel.php';

$db = new Database();
$db = $db->connect();

$quiz = new Quiz($db);
$client = new MongoClient();
$collection = $client->getCollection('scores');

$quiz_id = filter_input(INPUT_POST, 'quiz_id', FILTER_SANITIZE_NUMBER_INT);
$user_answers = $_POST['answers'] ?? [];

$score = $quiz->calculateScore($quiz_id, $user_answers);
$user_id = $_SESSION['user']['id'];

try {
    $existingUser = $collection->findOne(['user_id' => $user_id]);

    if ($existingUser) {
        $newTotalScore = $score + $existingUser['total_score'];
```

```
$quizScores = $existingUser['quiz_scores'];
$quizFound = false;

foreach ($quizScores as &$quizScore) {
    if ($quizScore['quiz_id'] == $quiz_id) {
        $quizScore['score'] += $score;
        $quizFound = true;
        break;
    }
}

if (!$quizFound) {
    $quizScores[] = ['quiz_id' => $quiz_id, 'score' => $score];
}
$collection->updateOne(
    ['user_id' => $user_id],
    [
        '$set' => [
            'total_score' => $newTotalScore,
            'quiz_scores' => $quizScores,
            'updated_at' => new MongoDB\BSON\UTCDateTime()
        ]
    ]
);
} else {
    $collection->insertOne([
        'user_id' => $user_id,
        'total_score' => $score,
        'quiz_scores' => [
            ['quiz_id' => $quiz_id, 'score' => $score]
        ],
        'created_at' => new MongoDB\BSON\UTCDateTime(),
        'updated_at' => new MongoDB\BSON\UTCDateTime()
    ]);
}
} catch (Exception $e) {
    error_log('Erreur lors de la soumission du quiz : ' . $e->getMessage());
    die('Erreur lors de la soumission du quiz. Veuillez réessayer.');
```

```
header('Location: get_score.php?score=' . $score);
exit;
```

Extrait de code de la classe Thread – TPUNN – Exemple 1

```
class Thread {
    private $conn;
    private $table = 'threads';

    public function __construct($db) {
        $this->conn = $db;
    }
    public function getThreadById($id) {
        $query = "SELECT threads.*, utilisateurs.nom_utilisateur AS author
                FROM threads
                JOIN utilisateurs ON threads.user_id = utilisateurs.id
                WHERE threads.id = :id";
        $stmt = $this->conn->prepare($query);
        $stmt->bindParam(':id', $id);
        $stmt->execute();
        return $stmt->fetch(PDO::FETCH_ASSOC);
    }
    public function updateThread($id, $title, $body) {
        $query = "UPDATE " . $this->table . " SET title = :title, body = :body WHERE id = :id";
        $stmt = $this->conn->prepare($query);
        $stmt->bindParam(':id', $id, PDO::PARAM_INT);
        $stmt->bindParam(':title', $title, PDO::PARAM_STR);
        $stmt->bindParam(':body', $body, PDO::PARAM_STR);
        return $stmt->execute();
    }
    public function incrementViews($mongoClient, $threadId) {
        $viewsCollection = $mongoClient->getCollection('views');
        $viewsCollection->updateOne(
            ['thread_id' => $threadId],
            ['$inc' => ['views' => 1]],
            ['upsert' => true]
        );
    }
}
```

Extrait de code de la classe MongoDB – TPUNN – Exemple 1

```
class MongoDB {
    private $mongoClient;
    private $mongoCollection;
    public function __construct() {
        $uri = 'mongodb+srv://AbduUSDI:heroku123456@abdurahmanusdi.lc9y4uk.mongodb.net';
        $databaseName = 'tpunn_quizz_score';
        try { $this->mongoClient = new Client($uri);
            $this->mongoCollection = $this->mongoClient->selectDatabase($databaseName)->scores;
        } catch (Exception $erreur) {
            error_log("Erreur de connexion à MongoDB : " . $erreur->getMessage());
            throw new Exception("Impossible de se connecter à la base de données MongoDB");
        }
    }
    public function getScoresParents() {
        try {
            $collection = $this->mongoCollection;
            $cursor = $collection->find(
                [],
                [
                    'sort' => ['score' => -1],
                    'limit' => 10
                ] );
            $scores = [];
            foreach ($cursor as $document) {
                $scores[] = [
                    'user_id' => $document['user_id'] ?? null,
                    'score' => $document['score'] ?? 0,
                    'total_score' => $document['total_score'] ?? 0
                ];
            }
            return $scores;
        } catch (Exception $e) {
            error_log("Erreur lors de la récupération des scores : " . $e->getMessage());
            return [];
        }
    }
}
```

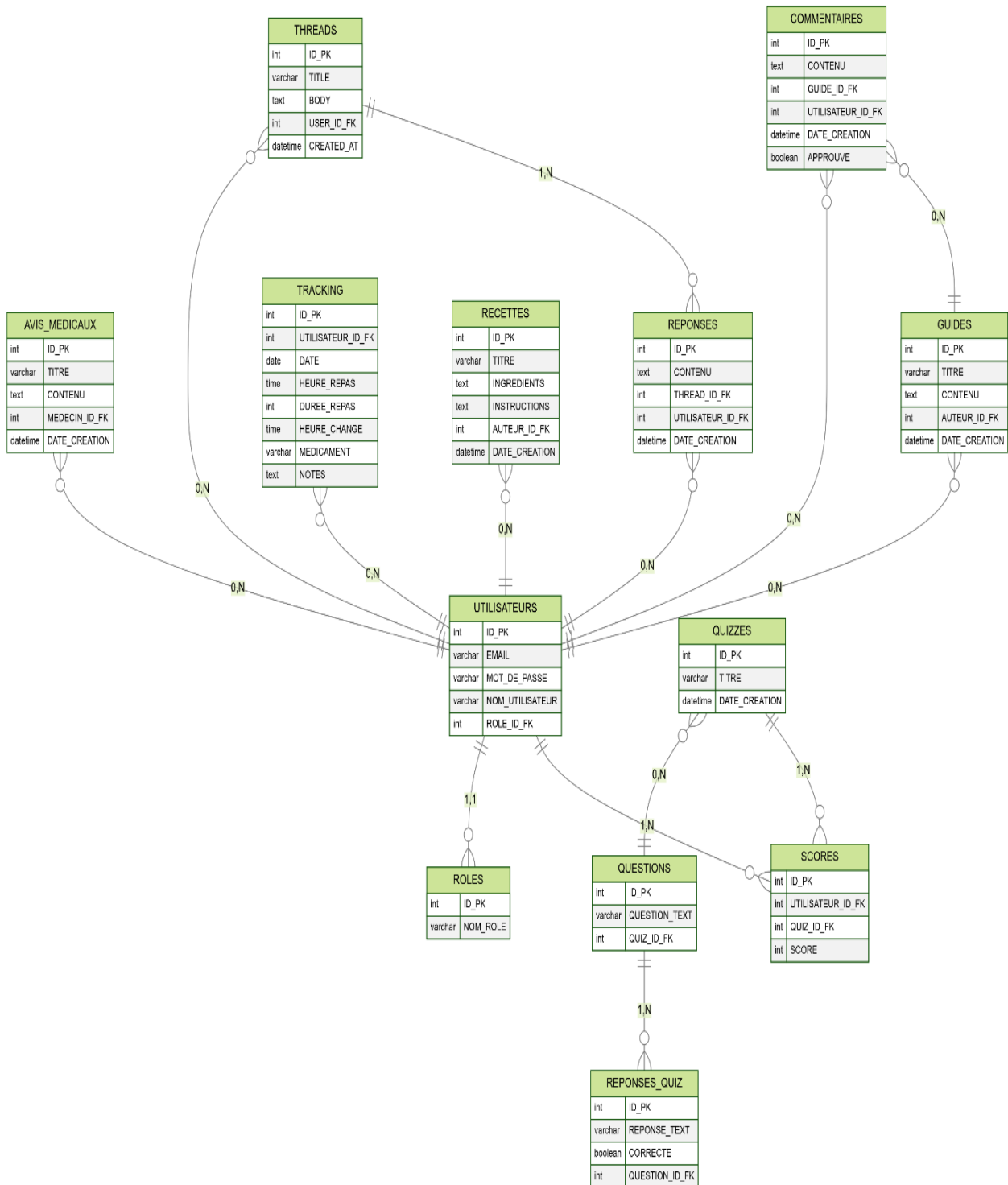
Extrait de code du modèle User – TPUNN – Exemple 1

```
class User {
    private $conn;
    private $table = 'utilisateurs';
    public function __construct($db) {
        $this->conn = $db;
    }
    public function getUserById($id) {
        $query = "SELECT * FROM " . $this->table . " WHERE id = :id";
        $stmt = $this->conn->prepare($query);
        $stmt->bindParam(':id', $id, PDO::PARAM_INT);
        $stmt->execute();
        return $stmt->fetch(PDO::FETCH_ASSOC);
    }
    public function getUsernames($db, $userIds) {
        $query = "SELECT id, nom_utilisateur FROM " . $this->table . " WHERE id IN (" . implode(',',
array_map('intval', $userIds)) . ")";
        $stmt = $db->prepare($query);
        $stmt->execute();
        $usernames = [];
        while ($row = $stmt->fetch(PDO::FETCH_ASSOC)) {
            $usernames[$row['id']] = $row['nom_utilisateur'];
        }
        return $usernames;
    }
    public function createUser($username, $email, $password, $role_id) {
        $query = "INSERT INTO " . $this->table . " (nom_utilisateur, email, mot_de_passe, role_id)
VALUES (:username, :email, :password, :role_id)";
        $stmt = $this->conn->prepare($query);
        $stmt->bindParam(':username', $username);
        $stmt->bindParam(':email', $email);
        $stmt->bindParam(':password', password_hash($password, PASSWORD_BCRYPT));
        $stmt->bindParam(':role_id', $role_id);
        return $stmt->execute();
    }
}
```

DOSSIER PROFESSIONNEL (DP)

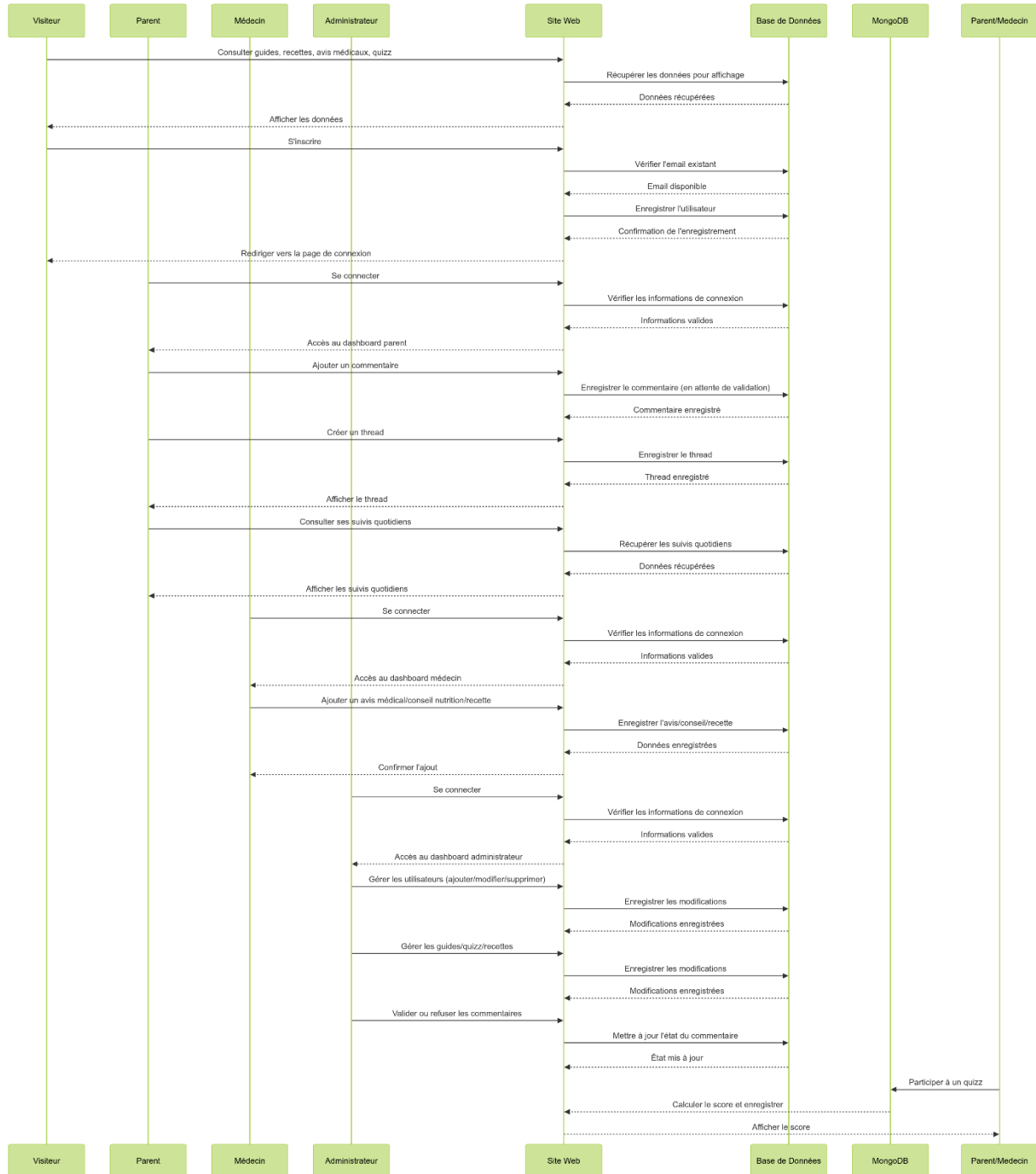
Les diagrammes pour Toutpourunnouveau :

A. MCD :



DOSSIER PROFESSIONNEL (DP)

B. Séquence :



DOSSIER PROFESSIONNEL (DP)

C. Diagramme pour E-learning :

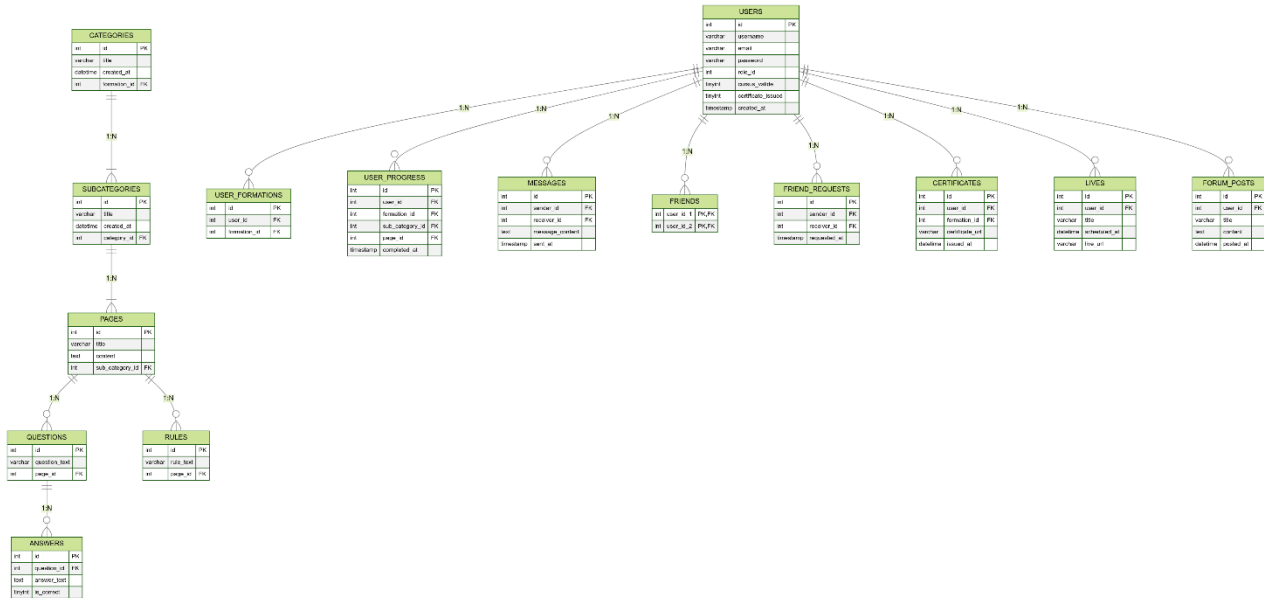
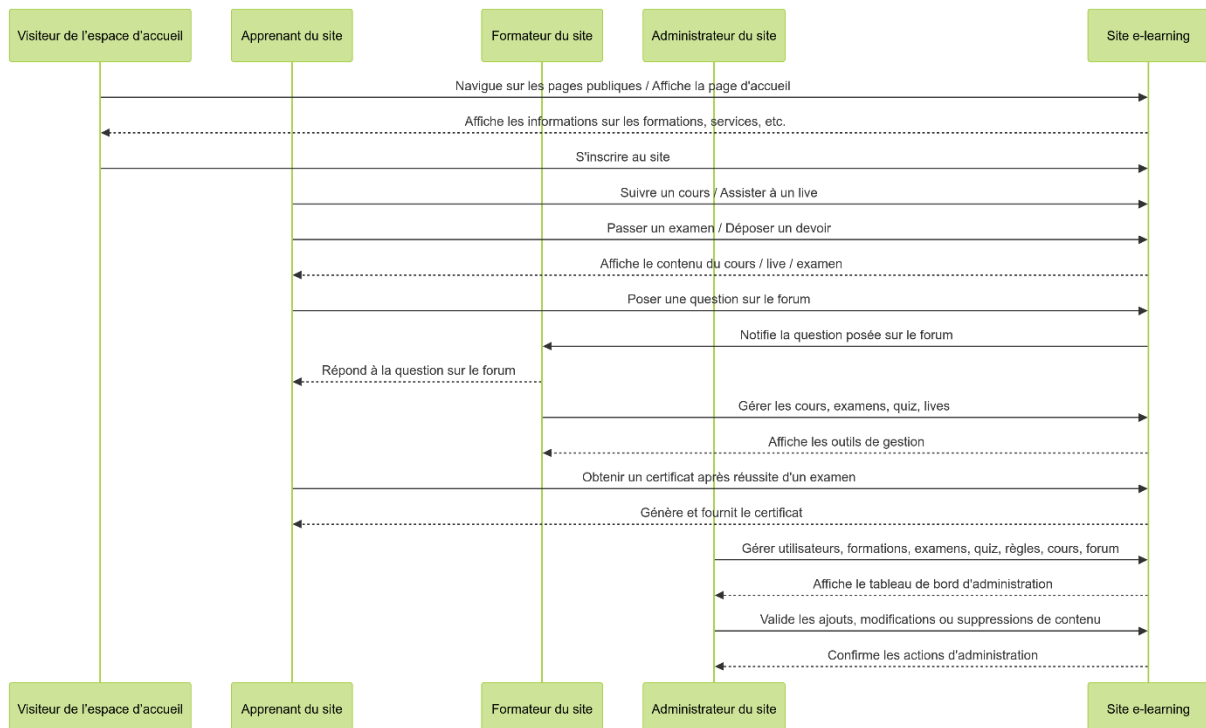


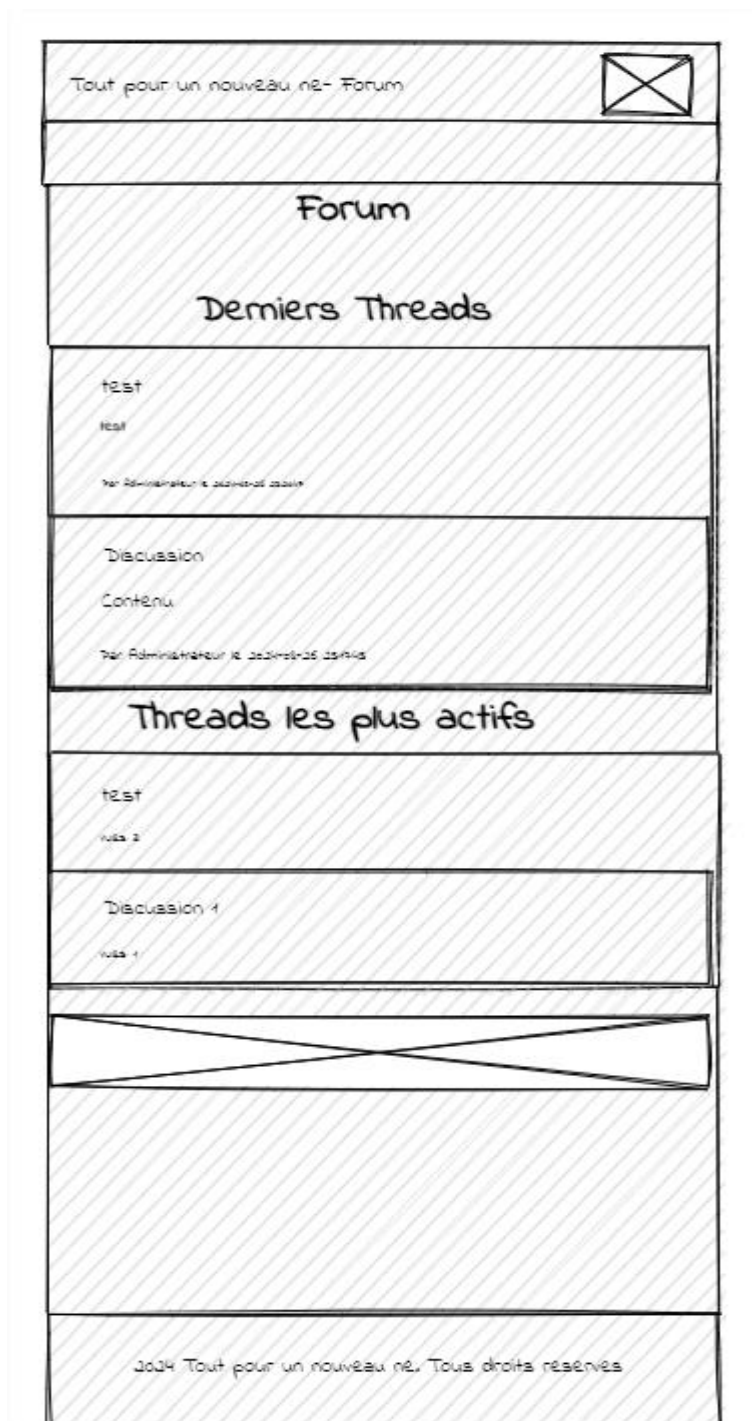
Diagramme de séquence :



Maquettes Tout pour un nouveau :

D. Wireframe :

1. Mobile :



DOSSIER PROFESSIONNEL (DP)

Desktop :

Tout pour un nouveau né Accueil Conseils de nutrition Nos quiz Nos guides pour les parents Diversification alimentaire/ Recettes Nous contacter Les avis médicaux Le forum Connexion

Connexion

Email

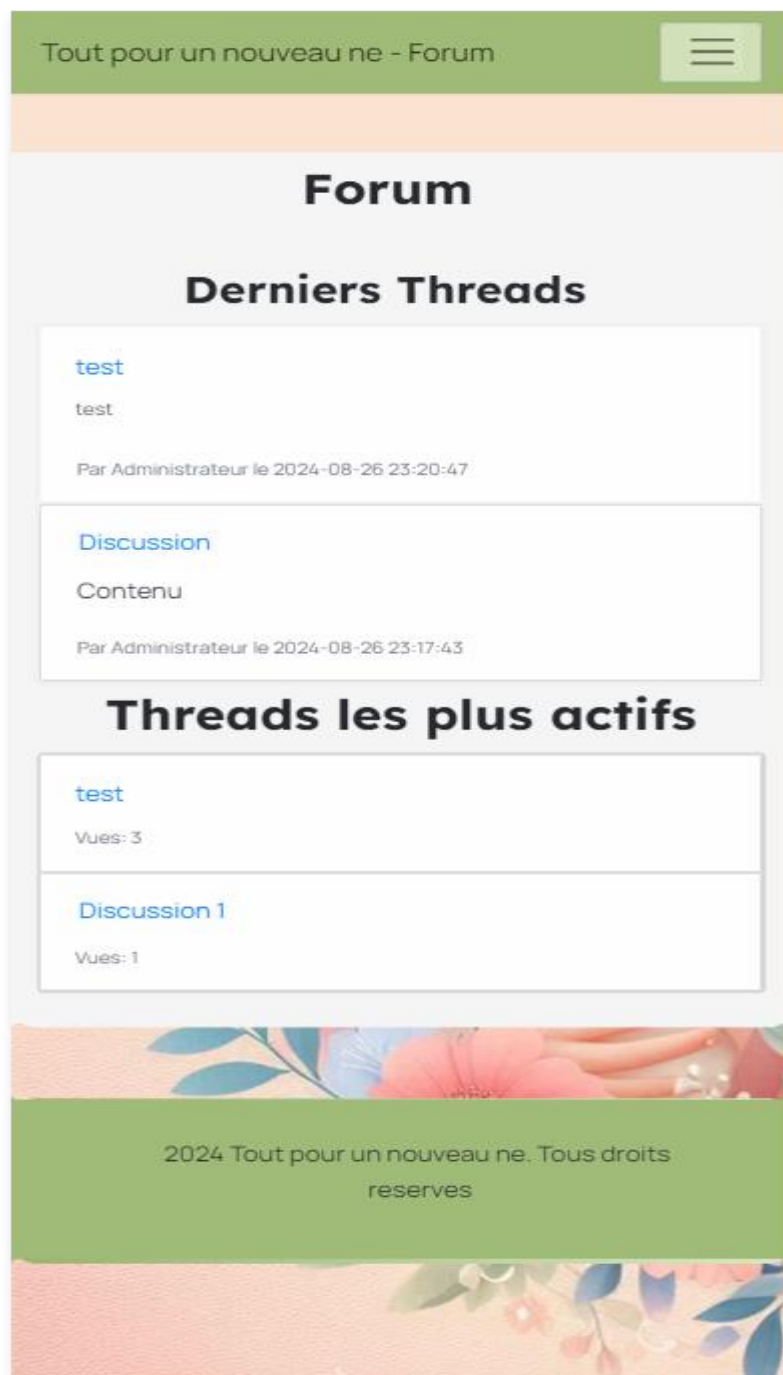
Mot de passe ☐

© 2024 Tout pour un nouveau né. Tous droits réservés.

DOSSIER PROFESSIONNEL ^(DP)

E. Mockups :

Mobile :



Made with 

DOSSIER PROFESSIONNEL (DP)

Desktop :

[Tout pour un nouveau né- Forum](#) [Retour sur tout- pour- un- nouveau- ne](#) [Accueil](#) [Créer une discussion](#) [Toutes les discussions](#) [Mon profil](#) [Nous contacter](#) [Déconnexion](#)

Forum

Derniers Threads

[Discussion 1](#)

Objet 1

Par Administrateur le 2024-07-08 22:10:59

[Discuss parent](#)

Parent

Par ParentAbdu le 2024-07-08 17:09:54

[Discussion docteur](#)

Docteur

Par Docteur le 2024-07-08 16:59:48

[Discussion administrateur](#)

Objet 2

Par Docteur le 2024-07-08 16:47:16

[Test 1](#)

Sous-test 1

Par Administrateur le 2024-07-08 16:32:44

[Discussion pour les femmes enceintes](#)

Ici vous pouvez poster tout les commentaires que vous voulez pour parlez entre femmes, pour vous entraidez .

Par Administrateur le 2024-07-07 16:53:17

Threads les plus actifs

[Discussion pour les femmes enceintes](#)

Vues: 46

[afaf](#)

Vues: 13

[Discuss parent](#)

Vues: 13

[GZZGZGrzf](#)

Vues: 8

[FEAQDAZ](#)

Vues: 5

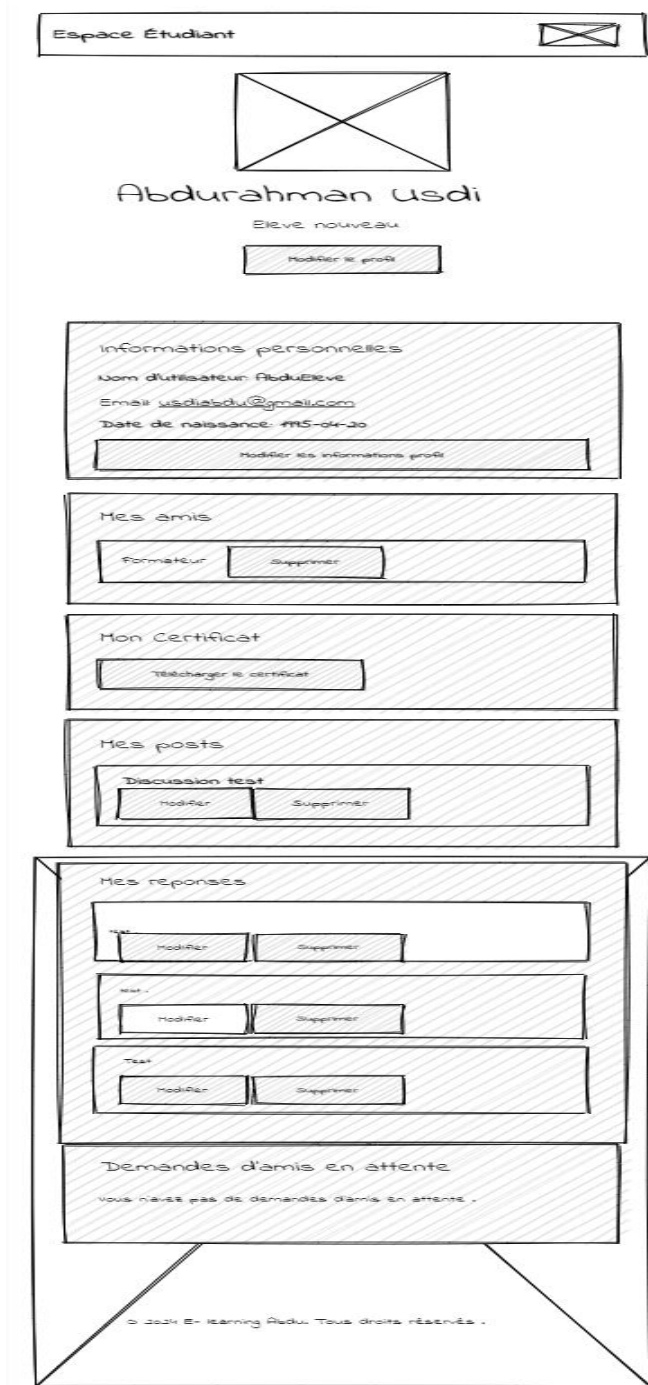
© 2024 Tout pour un nouveau né. Tous droits réservés .

DOSSIER PROFESSIONNEL (DP)

1. E-learning Abdu :

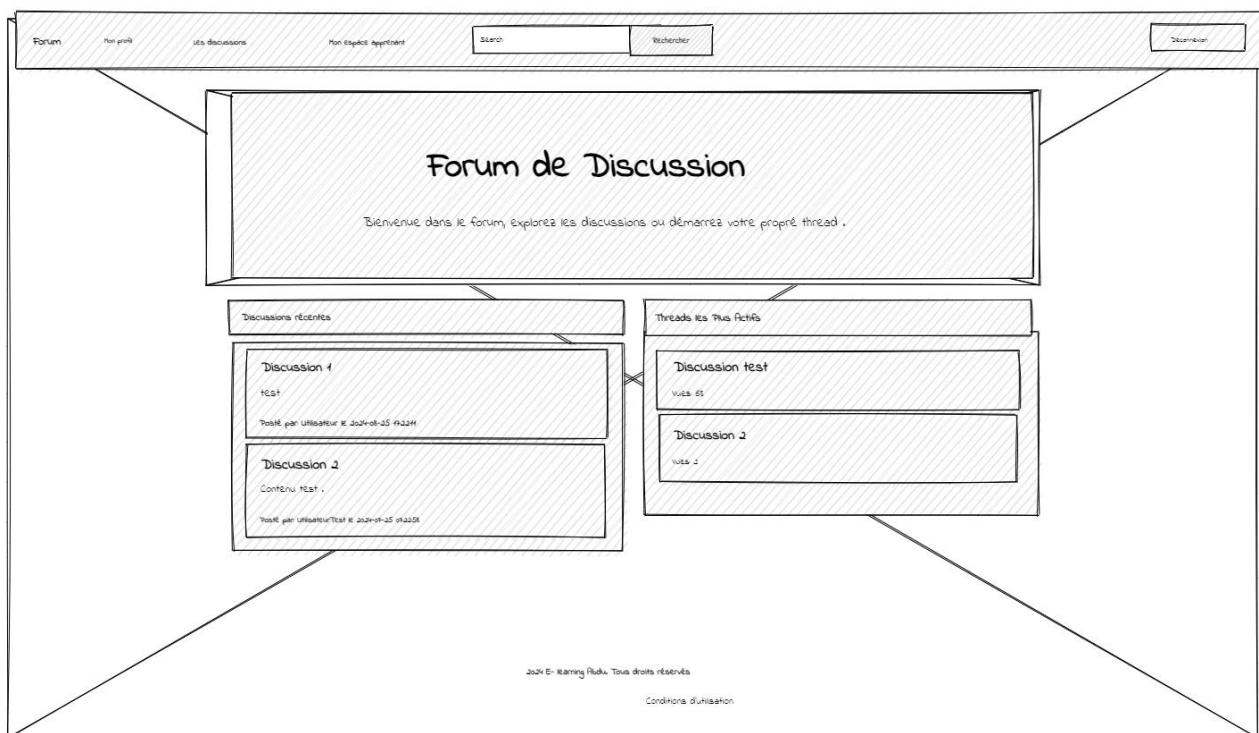
A. Wireframe :

Mobile :



DOSSIER PROFESSIONNEL (DP)

Desktop :



B. Mockups :

Mobile :




DOSSIER PROFESSIONNEL (DP)

Desktop :

The screenshot displays a web application interface for a student profile. At the top, a dark navigation bar contains links: Espace Etudiant, La mediateque, Les Quiz, Evaluations, Messagerie, Mon profil, Les lives, Forum, and a red 'Deconnexion' button. The main content area has a dark background. At the top center is a circular profile picture of a baby, followed by the name 'Abdurahman Usdi' and the status 'Eleve nouveau'. Below this is a 'Modifier le profil' button. The interface is divided into three main columns. The left column contains three white boxes: 'Informations personnelles' with fields for username (AbduEleve), email (usdiabdu@gmail.com), and birth date (1995-04-20), plus a 'Modifier les informations profil' button; 'Mes amis' with a 'Formateur' button and a 'Supprimer' button; and 'Mon Certificat' with a 'Telecharger le certificat' button. The middle column contains two white boxes: 'Mes posts' with a 'Discussion test' post having 'Modifier' and 'Supprimer' buttons; and 'Mes reponses' with three posts labeled 'test...', 'test...', and 'Test', each with 'Modifier' and 'Supprimer' buttons. The right column contains a white box titled 'Demandes d'amis en attente' with the text 'Vous n'avez pas de demandes d'amis en attente'. At the bottom, a dark footer bar contains the text '2024 E-learning Abdu. Tous droits reserves.', a link to 'Politique de confidentialite Conditions d'utilisation', and the 'istock' logo.

Espace Etudiant La mediateque Les Quiz Evaluations Messagerie Mon profil Les lives Forum [Deconnexion](#)



Abdurahman Usdi
Eleve nouveau
[Modifier le profil](#)

Informations personnelles
Nom d'utilisateur: AbduEleve
Email: usdiabdu@gmail.com
Date de naissance: 1995-04-20
[Modifier les informations profil](#)

Mes amis
[Formateur](#) [Supprimer](#)

Mon Certificat
[Telecharger le certificat](#)

Mes posts
Discussion test
[Modifier](#) [Supprimer](#)

Mes reponses
test...
[Modifier](#) [Supprimer](#)
test...
[Modifier](#) [Supprimer](#)
Test
[Modifier](#) [Supprimer](#)

Demandes d'amis en attente
Vous n'avez pas de demandes d'amis en attente

2024 E-learning Abdu. Tous droits reserves.
[Politique de confidentialite](#) [Conditions d'utilisation](#)

Made with 