

02/01/2025

Rapport sur le développement de l’application pour le module ISI-Web

Une image contenant texte, art, dessin, Graphique

Description générée automatiquement

MESSAOUDI Amin 12108951 NADIR Ayoub 12101440

**Organisation du binôme e** **t Architecture de l’application**

**Organisation du travail**

**Dans ce projet, le travail a été partagé entre deux membres du binôme selon leurs compétences respectives :**

* **MESSAOUDI Amin : Plus à l’aise avec le JavaScript, il s’est occupé de toute la logique et les interactions utilisateur via le fichier javascript.js.**
* **NADIR Ayoub : À l’aise en PHP, il a développé les contrôleurs, les modèles, et la gestion des bases de données.**
* **Ensemble : Les deux membres ont collaboré sur la partie visuelle et logique des templates avec Twig, ainsi que sur le style global de l'application.**

**Difficultés rencontrées et solutions apportées**

**Problème 1 : Affichage des pages selon l’accès**

**Lors d’un accès direct à une page, l’affichage était correct. Cependant, en passant par index.php, les pages perdaient leur mise en page.**

**Solutions explorées :**

1. **Créer des références absolues : peu efficace.**
2. **Définir dynamiquement une URL de base : sans effet.**
3. **Tout recommencer avec un système de routes : des problèmes de chargement de liens sont apparus.**
4. **Mieux organiser les fichiers pour faciliter leur accès : solution fonctionnelle.**

**Problème 2 : Récupération des données depuis la base**

**Un problème lié au fichier index.php a empêché la récupération correcte des données.**

**Solutions explorées :**

1. **Modifier le fichier index.php, réimplémenter routes.php, et adapter connect.php à partir d’anciens projets.**
2. **Remarques : ni route.page\_demandée, ni path('page\_demandée') ne sont des approches efficaces.**

**Problème 3 : Association des entreprises à leurs spécialités**

**Solution : Une fonction PHP utilise GROUP\_CONCAT pour récupérer les spécialités de chaque entreprise et les afficher correctement.**

**Remarque : Il est nécessaire de requérir les contrôleurs dans index.php avant de charger les pages.**

**Problème 4 : Pages affichées en double**

**Solution : Rédéfinir les données initiales comme un tableau vide qui se remplit dynamiquement selon le fichier .php demandé.**

**Problème 5 : Conflit entre suppression et ajout d’étudiants**

**Solution : Utiliser des conditions dans les contrôleurs et définir le type d’action (suppression ou ajout) via Twig pour la suppression et via JavaScript pour l’ajout.**

**Problème 6 : Récupération des spécialités lors de l’ajout d’une entreprise**

**Solution : Une fonction JavaScript récupère les spécialités et les affiche dans une liste déroulante.**

**Problème 7 : Erreur "Unexpected token '<'" lors de l’inscription**

**Solutions explorées :**

1. **Inclure la fonction hideSuccessMessage() dans Inscription.twig (peu esthétique).**
2. **Nettoyer le code de javascript.js pour éliminer les anciens morceaux de code conflictuels : solution retenue.**

**Problème 8 : Inclusion de la spécialité lors de l’ajout d’une entreprise**

**Solutions explorées :**

1. **Appeler une fonction pour récupérer l’identifiant de la nouvelle entreprise, une autre pour récupérer le numéro de spécialité, et une troisième pour ajouter les données à spec\_entreprise : cela a causé des bugs.**
2. **Faire appel à l’intelligence artificielle pour optimiser le processus.**

**Architecture de l'application**

**Structure des pages principales**

1. **Page Entreprise :**
   * **Recherche ou ajout d’une entreprise.**
   * **Affichage de toutes les entreprises avec des actions possibles pour chaque entreprise :**
     + **Modifier ses données.**
     + **Supprimer l’entreprise.**
     + **Afficher ses données et la liste des stagiaires qu’elle a accueillis.**
2. **Page Stagiaire :**
   * **Recherche ou ajout d’un étudiant.**
   * **Affichage de la liste de tous les étudiants avec des actions possibles pour chaque étudiant :**
     + **Modifier ses données.**
     + **Supprimer l’étudiant.**
     + **Afficher ses données.**
3. **Page Inscription :**
   * **Sélection d’un étudiant pour l’inscrire à un stage ou une alternance dans une entreprise.**
4. **Page Aide :**
   * **Fournit des informations générales sur la plateforme.**

**Structure des dossiers**

* **config/ : Contient les fichiers de configuration (par exemple, routes.php et geststages.sql).**
* **public/ : Regroupe les ressources accessibles publiquement (CSS, images, JavaScript, et index.php).**
  + **css/styles.scss**
  + **js/javascript.js**
* **src/ : Contient le code principal de l’application.**
  + **Controller/ : Gère la logique applicative (par exemple, Accueil.php, EditEnt.php).**
  + **Model/ : Regroupe les fonctions PHP pour la manipulation des données (par exemple, connect.php, model.php).**
  + **templates/ : Contient les fichiers Twig pour le rendu des pages.**

**Fonctionnement**

1. **Les requêtes utilisateurs passent par index.php, qui appelle le routeur défini dans routes.php.**
2. **Les routes définies redirigent vers les contrôleurs appropriés dans Controller/.**
3. **Les contrôleurs utilisent les fonctions PHP du dossier Model/ pour manipuler les données.**
4. **Les résultats sont affichés à l’utilisateur à l’aide des fichiers Twig dans templates/.**

**Cette architecture permet une séparation claire entre la logique, les données, et la présentation, tout en facilitant la maintenance et l’évolutivité du projet.**