Baimurat

ERGESHOV

SIO12

Compte Rendu

Projet 1 : PHP-MYSQL

Organisation du code

Contexte

Dans ce projet, j’ai repris le travail d’un ancien stagiaire afin d’organiser le code et de créer une interface d’administration permettant de gérer les salariés (ajout, suppression, modification).

Organisation du code

Au début, le fichier listeSalaries.php mélangeait un peu tout (HTML + requêtes SQL). J’ai donc essayé de mieux organiser le projet pour que ce soit plus clair et plus simple à maintenir.

Dossier partials

J’ai créé un dossier partials dans lequel j’ai mis :

- header.php
- footer.php

Ça évite de recopier le header et le footer dans chaque page.
On les inclut simplement avec :

```
include_once("./partials/header.php");
include_once("./partials/footer.php");
```

Si on veut modifier la barre de navigation ou le footer, on le fait une seule fois.

Fichier de connexion à la base de données

J’ai créé un fichier db.php pour gérer la connexion à la base de données.

Je me suis basé sur le fichier connexion.php donné dans le TP, puis j’ai modifié :

- le nom de la base de données
- les identifiants de connexion

J’utilise PDO avec gestion des erreurs.
Comme ça, si les paramètres changent, on n’a qu’un seul fichier à modifier.

Dossier utils et fonctions

Ensuite, j’ai créé un dossier utils avec un fichier function.php.

Dans ce fichier, j’ai mis toutes les fonctions qui font les requêtes SQL (SELECT, INSERT, UPDATE, DELETE).

J’inclus la connexion à la base avec :

include_once(__DIR__ . "/db.php");

Puis dans listeSalaries.php, j’inclus :

include_once("./utils/function.php");

Ça permet de séparer :

- la partie affichage (HTML)
- la partie traitement (requêtes SQL)

Affichage des salariés

Les salariés sont affichés dans un tableau HTML.

J’ai aussi ajouté :

- une pagination (pour limiter le nombre de résultats par page)
- une barre de recherche
- un tri des colonnes quand on clique sur l’en-tête du tableau

Le système se fait en passant par la bibliothèque Datatable, qui automatise ce système, il faut juste l’appelé avec la balise script.

Ajout d’un salarié

J’ai ajouté un bouton “Ajouter un salarié”.

Quand on clique dessus, on est redirigé vers ajouterSalarie.php, qui contient un formulaire vide avec :

- nom
- prénom
- date de naissance
- date d’embauche
- salaire
- service (liste déroulante)

Quand on valide, les données sont envoyées en POST vers /process/ajoutSalarie.php.

Dans ce fichier :

- je récupère les données
- j’exécute une requête SQL INSERT
- puis je redirige vers listeSalaries.php

Suppression d’un salarié

Chaque salarié a un bouton **Supprimer**.

Quand on clique dessus :

1. Une fonction JavaScript confirmerSuppression(id) s’exécute
2. Une confirmation s’affiche
3. Si on accepte, on est redirigé vers supprimerSalarie.php?id=...

Dans ce fichier :

- Je vérifie que l’id est bien présent dans l’URL
- Je vérifie qu’il existe en base
- J’exécute une requête DELETE

J’ai utilisé bindParam() pour sécuriser la requête et éviter les injections SQL.

Ensuite, redirection vers listeSalaries.php.

Modification d’un salarié

Il y a aussi un bouton Modifier.

Quand on clique dessus :

- Redirection vers modifierSalarie.php?id=...
- Vérification de l’id dans l’URL
- Vérification que le salarié existe

Si tout est bon, un formulaire s’affiche avec les champs déjà remplis grâce à l’attribut value.
Il y a aussi un champ caché contenant l’id.

Quand on valide :

- Les données sont envoyées en POST vers /process/modifSalarie.php
- Une requête UPDATE est exécutée
- Puis redirection vers listeSalaries.php

Sécurité

Dans le projet, j’ai essayé d’appliquer quelques bonnes pratiques :

- Utilisation de PDO
- Requêtes préparées avec bindParam
- Vérification des paramètres GET et POST
- Confirmation avant suppression
- Séparation entre affichage et traitement