

DOSSIER PROJET

Développeur Web Web Mobile

Année 2025

Sabrina Hadi

Introduction:

Le métier de Développeur Web et Web Mobile consiste à créer des sites internet et des applications accessibles depuis un ordinateur, une tablette ou un smartphone. Il demande à la fois de la logique, de la créativité et une bonne organisation.

Tout au long de ma formation, j'ai appris à concevoir des pages web, à les rendre interactives, à gérer les informations que les utilisateurs envoient (comme des messages ou des formulaires), et à assurer le bon fonctionnement du site. J'ai aussi appris à travailler en respectant des consignes précises, à collaborer avec d'autres personnes.

Formation en CDPI(Contrat Développement Professionnel Intérimaire).

Dans ce dossier, je présente les projets que j'ai réalisés.

SOMMAIRE

1. *Développer la partie front-end d'une application web ou web mobile sécurisée*
 - 1.1. *Installer et configurer son environnement de travail.....p.4-8*
 - 1.2. *Maquetter des interfaces utilisateurs.....p.9-11*
 - 1.3. *Réaliser des interfaces utilisateurs statiques.....p.11-13*
 - 1.4. *Développer la partie dynamique des interfaces utilisateurs.....p.14-16*

2. *Développer la partie back-end d'une application web ou web mobile sécurisée*
 - 2.1. *Mettre en place une base de données relationnelle.....p.16-21*
 - 2.2. *Développer des composants d'accès aux données SQL et NoSQL.....p.21-26*
 - 2.3. *Développer des composants métier côté serveur.....p.26-28*
 - 2.4. *Documenter le déploiement d'une application dynamique.....p.29-30*

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

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

J'ai choisi de réaliser un CV en ligne statique. Avant de commencer le développement, j'ai mis en place un environnement de travail local adapté au projet, en installant un éditeur de code : Visual Studio Code. Ce CV a été développé sans base de données ni serveur, en utilisant uniquement les langages HTML et CSS, car il s'agissait d'un projet purement front-end, centré sur la présentation visuelle et l'ergonomie de l'interface.

Les outils de base utilisés:

Windows

Google

Git

Vscode

Versionnig:

Github (pour pouvoir récupérer notre travail)

Qu'est ce que Git ?

Git est un outil de gestion de versions utilisé principalement en développement logiciel. Il permet de suivre l'évolution d'un projet, de sauvegarder chaque modification du code, et de collaborer efficacement à plusieurs sur le même projet, sans écraser le travail des autres.

Commandes principales:

git init (Initialise un dépôt git)

git add . (Prépare les fichiers pour le commit)

git commit -m "" (Enregistre les modifications)

git push (Envoie les changements sur Github)

git pull (Récupère les mises à jour)

git checkout -b (Crée une nouvelle branche)

J'ai également installé Git et configuré un dépôt sur GitHub afin de sauvegarder et suivre l'évolution de mon travail. Chaque commit représente une version du projet que je peux retrouver facilement, ce qui me permet de revenir à une étape précédente en cas d'erreur ou de tester différentes modifications.

Même si le projet ne comporte pas de logique dynamique ou de base de données, utiliser Git m'a permis de structurer mon travail, de tester différentes mises en page sans tout casser, et d'apprendre à versionner proprement un projet front-end.

Comment créer un repository sur Github :

Voici les étapes simples pour créer un nouveau dépôt GitHub et récupérer son lien :

1. Aller directement sur github.com/new pour accéder à la création d'un nouveau repository.
2. Donner un nom clair au dépôt.
3. Ne pas cocher la case "Initialize this repository with a README" pour éviter les conflits lors du premier push.
4. Cliquer sur Create repository.
5. Une fois le dépôt créé, cliquer sur le bouton "<> Code", puis :
 - Copier l'URL en HTTPS pour pouvoir cloner le dépôt localement.

Exemple de commande pour cloner le dépôt :

```
git clone https://github.com/mon-utilisateur/mon-projet.git
```

Envoyer les fichiers sur GitHub : dans bash

```
git push -u origin master
```

Qu'est ce qu'un commit dans git ?

Un commit est une étape clé dans l'utilisation de Git, un système de gestion de version. Il permet de sauvegarder les modifications apportées à des fichiers dans un projet

Comment faire un commit dans git ?

1. Initialiser un dépôt Git local

Créer un dépôt Git dans le dossier courant (si ce n'est pas déjà fait) :

```
git init
```

Cette commande crée un dossier caché .git qui va suivre l'historique de ton projet.

2. Vérifier l'état du dépôt

Afficher les fichiers modifiés, ajoutés ou non suivis :

```
git status
```

Très utile pour savoir où tu en es avant de faire un commit.

3. Ajouter des fichiers à l'index (zone de préparation)

Pour un fichier spécifique :

```
bash
```

```
git add monfichier.txt
```

Pour tous les fichiers modifiés :

```
bash
```

```
git add .
```

Cela prépare les fichiers pour le prochain commit.

4. Créer un commit

Enregistrer les changements avec un message clair :

```
bash
```

```
git commit -m "Ajout de la fonctionnalité de connexion utilisateur"
```

Ce message doit décrire ce que tu as modifié ou ajouté.

5. Pull — Récupérer les modifications distantes

Synchroniser avec le dépôt distant (ex. GitHub) avant d'envoyer tes commits :

```
bash
```

```
git pull origin main
```

Remplace main si ta branche a un autre nom (comme master, develop, etc.).

6. Push — Envoyer les modifications sur le dépôt distant

Envoyer les commits locaux vers GitHub (ou un autre dépôt distant) :

```
bash
```

```
git push origin main
```

Tu dois d'abord avoir lié ton dépôt local à un dépôt distant avec :

```
bash
```

```
git remote add origin https://github.com/ton-compte/ton-projet.git
```

J'ai aussi installé Laragon, qui est un environnement de développement local pour Windows, conçu pour faciliter la création de sites web et d'applications en PHP (et d'autres langages). Il inclut PhpMyAdmin, même si pour ce projet précis (un CV statique), aucune base de données n'a été utilisée. Cette installation me permet toutefois d'être prêt à travailler sur mes autres projets (cinetech etc...) .

J'ai mis en place une arborescence claire et structurée pour mes fichiers. Le fichier index.html est situé à la racine du projet, et un sous dossier Assets/Css.

J'ai également intégré Bootstrap 5 via un CDN (Content Delivery Network) afin de simplifier la création d'une interface responsive, compatible avec différents types d'écrans (ordinateur, tablette, smartphone).

Un CDN est un réseau de serveurs répartis géographiquement qui permet de charger plus rapidement des fichiers statiques tels que des feuilles de style CSS, des bibliothèques JavaScript ou des polices web.

En passant par un CDN, je n'ai pas eu besoin de télécharger Bootstrap dans mon projet : le fichier CSS est directement chargé depuis un serveur externe rapide et fiable. Cela permet non seulement de réduire les temps de chargement, mais aussi de garder le projet léger, et de profiter d'une version toujours à jour du framework.

J'ai adopté une approche mobile-first, en concevant d'abord l'affichage sur petit écran, puis en ajoutant des media queries pour les écrans plus larges, afin de garantir une bonne expérience utilisateur sur tous les supports.

Qu'est-ce que la méthode scrum ?

La méthode Scrum est un cadre Agile utilisé pour gérer des projets informatiques. Elle repose sur une organisation en Sprints (courtes périodes de développement), une collaboration continue, et des rôles bien définis :

- **Product Owner** : définit les besoins métier.
- **Scrum Master** : veille au respect de la méthode.
- **Équipe de développement** : conçoit et livre le produit.

Bien que ce projet ait été réalisé en autonomie, je m'inspire des principes Scrum en :

- planifiant mes tâches,
- testant progressivement mes interfaces,
- effectuant des ajustements réguliers selon les retours.

Il y a aussi react que je n'ai pas utilisé dans mes projets:

React (aussi appelé React.js ou ReactJS) est une bibliothèque JavaScript développée par Meta (Facebook) pour créer des interfaces utilisateurs dynamiques et réactives, principalement pour le web.

Installation de react:

1. Installer Node.js et npm (si ce n'est pas déjà fait)
2. Créer un projet React depuis le terminal (PowerShell, Bash, ou VS Code) :

`npx create-react-app mon-projet`

3. Aller dans le dossier du projet :

`cd mon-projet`

4. Lancer l'application :

`npm start`

Cela ouvre automatiquement l'application dans le navigateur à l'adresse :
`http://localhost:3000`

1.2 Maquetter des interfaces utilisateurs web ou web mobile.

Qu'est-ce qu'un wireframe ?

Un wireframe est une maquette fonctionnelle, simplifiée et schématique, utilisée lors des premières étapes de conception d'un site web ou d'une application. Il représente la structure générale d'une page sans éléments graphiques détaillés (pas de couleurs, typographies travaillées ou images finales). L'objectif principal est de visualiser rapidement l'organisation des contenus, la hiérarchie de l'information et les éléments de navigation.

Les wireframes sont généralement en noir et blanc et servent à :

- Définir la disposition des différents éléments d'une interface (menus, titres, boutons, formulaires, etc.) ;
- Faciliter la communication entre les équipes (développeurs, designers, chefs de projet) ;
- Valider la logique de navigation et l'ergonomie avant de passer à la conception graphique.

Outils utilisés pour les wireframes :

Des outils comme Figma, Balsamiq, Adobe XD ou même des croquis papier peuvent être utilisés pour réaliser rapidement des wireframes.

Qu'est-ce qu'une maquette ?

Une maquette est une représentation visuelle fidèle d'une interface utilisateur (site web, application mobile, etc.) réalisée après le wireframe, lors de la phase de conception graphique. Contrairement au wireframe, la maquette inclut les éléments visuels définitifs : couleurs, typographies, images, icônes, logos, et parfois les interactions simulées (liens, animations).

La maquette permet de :

- Se projeter dans le rendu final du site ou de l'application ;
- Tester différentes palettes de couleurs, typographies et styles graphiques ;
- Obtenir la validation du client ou de l'équipe avant le développement ;
- Prévoir l'accessibilité et la cohérence graphique.

Ux - L'expérience utilisateur

L'UX désigne la manière dont un utilisateur vit et ressent l'utilisation d'un site ou d'une application. Elle se concentre sur la facilité de navigation, la logique d'utilisation, la rapidité d'accès aux informations et la satisfaction globale.

Un bon UX design permet à l'utilisateur de trouver ce qu'il cherche sans se poser de questions, avec un chemin clair et intuitif.

UI – L'interface utilisateur

L'UI (interface utilisateur) concerne l'aspect visuel de l'interface : les couleurs, polices, boutons, icônes, et la mise en page. L'objectif est de rendre le site attrayant, lisible et cohérent, tout en facilitant les actions de l'utilisateur grâce à un design clair et esthétique.

Étapes de conception

1. Recherche d'inspiration
Avant de commencer, j'ai consulté différents modèles de CV en ligne pour m'inspirer des tendances actuelles en termes de design et d'ergonomie.
2. Réalisation du wireframe
J'ai d'abord esquissé un wireframe simple pour organiser les grandes zones du CV : informations personnelles, compétences, expériences, etc.
3. Passage à la maquette graphique
J'ai structuré la page en deux colonnes distinctes :
 - Colonne de gauche : informations personnelles, compétences techniques, qualités comportementales (savoir-être), langues parlées, centres d'intérêt.
 - Colonne de droite : formations et expériences professionnelles.
4. Choix graphiques
J'ai opté pour une couleur bleue dans la colonne de gauche, à la fois sobre et professionnelle, associée à un fond clair.
La typographie est simple et lisible, et j'ai veillé à espacer les différentes sections pour garantir une lecture fluide et agréable.
5. Accessibilité
J'ai vérifié les contrastes de couleurs et la taille des textes pour garantir une bonne lisibilité, y compris pour les personnes malvoyantes.

Côté design, j'ai opté pour une couleur bleue dans la colonne de gauche, à la fois sobre et professionnelle, associée à un fond clair. La typographie est simple et lisible, et j'ai veillé à espacer les différentes sections pour garantir une lecture fluide et agréable.

Une fois la maquette terminée et validée, je suis passée à l'intégration en HTML et CSS. J'ai utilisé Bootstrap 5 pour gagner du temps et respecter une structure cohérente. Le système de grille de Bootstrap m'a permis de reproduire facilement la disposition en deux colonnes tout en assurant une bonne responsivité du site : le CV s'adapte automatiquement à tous les formats d'écran (ordinateur, tablette, smartphone).

Chaque section du CV a été codée de manière claire et organisée, en respectant les bonnes pratiques front-end : titres hiérarchisés, mise en valeur des compétences par des barres de progression, séparation visuelle entre les différentes parties, etc.

J'ai aussi intégré une fonction d'impression, en utilisant @media print dans une feuille de style dédiée, afin de proposer un format A4 propre, sans éléments inutiles à l'écran.

1.3 Réaliser des interfaces utilisateurs statiques web ou web mobile.

Pour cette compétence, j'ai réalisé entièrement une page web statique sous forme de CV en ligne. Le projet a été codé en HTML et CSS, sans base de données ni langage serveur, car il s'agissait d'une interface purement front-end.

La page affiche les différentes sections de mon parcours (profil, compétences, expériences, formations...) mais ne réagit pas dynamiquement aux actions de l'utilisateur.

Pour cela, j'utilise principalement HTML pour structurer les contenus (titres, paragraphes, images, formulaires...) et CSS pour leur mise en forme (couleurs, polices, disposition, responsivité). Je suis capable de suivre une maquette fournie et de la transformer en une page fidèle et responsive, adaptée aussi bien à un affichage ordinateur qu'à un affichage mobile.

J'ai utilisé Git pour le suivi de version et publié le projet sur GitHub, afin de garder une trace claire des modifications et rendre mon travail accessible facilement.

Suivi de maquette

Je suis capable de suivre une maquette fournie (réalisée sur Figma) et de la transformer en une page fidèle et responsive, adaptée aussi bien à un affichage ordinateur qu'à un affichage mobile.

Pour ce projet, j'ai utilisé le système de grille de Bootstrap 5 afin de simplifier la gestion des colonnes et d'assurer la compatibilité multi-écrans.

Qu'est ce que les média queries ?

Les media queries sont une fonctionnalité du CSS3 qui permet d'ajuster le style d'une page web en fonction des caractéristiques du support utilisé (écran, fenêtre, imprimante...). Grâce à la règle `@media`, on peut appliquer des styles uniquement lorsque certaines conditions sont remplies, comme la largeur, la hauteur, la résolution ou l'orientation de l'écran.

Qu'est ce que le média print ?

Le média print est un type de média CSS utilisé spécifiquement pour l'affichage lors de l'impression d'une page web. Il permet de définir des styles adaptés à une impression papier, différents de ceux affichés à l'écran.

Dans mon projet, j'ai choisi d'adapter l'affichage de ma page lors de l'impression grâce à une règle CSS `@media print`. J'ai intégré ces styles directement dans mon fichier principal CSS, où je définis des règles spécifiques pour optimiser la présentation sur papier. Par exemple, je supprime les images inutiles, j'ajuste les couleurs pour un rendu en noir et blanc, je masque certains éléments comme les barres de progression, et je veille à éviter les coupures de contenu au mauvais endroit grâce à des propriétés comme `page-break-inside: avoid`. Cela permet d'obtenir une version imprimée claire, lisible et bien organisée, différente de la version affichée à l'écran.

```
28 @media print {
29     body {
30         background-color: #fff;
31         color: #000;
32         font-size: 12pt;
33     }
34
35     .left-column {
36         background-color: #fff;
37         color: #000;
38         padding: 0;
39         border-right: 1px solid #ccc;
40         page-break-inside: avoid;
41     }
42
43     .right-column {
44         background-color: #fff;
45         padding: 0;
46         margin-top: 20px;
47     }
48
49     .profile-img {
50         display: none; /* Masquer l'image lors de l'impression */
51     }
52
53     a[href]:after {
54         content: " (" attr(href) ")"; /* Ajouter les liens entre parenthèses après le texte */
55     }
56
57     .progress {
58         display: none; /* Masquer les barres de progression */
59     }
60
61     .container {
62         width: 100%;
63         padding: 0;
64         margin: 0;
65     }
66
67     .row {
68         margin: 0;
69         padding: 0;
70     }
}
```

Je n'ai pas utilisé de media queries pour les écrans (@media screen ou @media (max-width: ...)) dans ce projet, mais je connais leur usage : elles permettent d'adapter l'affichage d'un site web selon la taille ou l'orientation de l'écran, notamment pour les smartphones ou tablettes.

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

Pour cette compétence, j'ai réalisé un livre d'or dynamique, où seuls les utilisateurs connectés peuvent laisser des commentaires. Ce projet m'a permis de créer une interface web interactive, capable de s'adapter à l'état de l'utilisateur tout en affichant dynamiquement du contenu issu de la base de données.

Un encadré blanc met en avant le message d'accueil, avec des boutons pour se connecter, s'inscrire ou lire les commentaires.

J'ai utilisé des media queries en CSS pour assurer une compatibilité sur tous les supports : smartphone, tablette et ordinateur. Cela permet aux visiteurs d'avoir une expérience de lecture fluide quel que soit l'appareil utilisé.

L'interactivité repose en grande partie sur la connexion de l'utilisateur. J'ai intégré un système de session PHP qui permet de reconnaître l'utilisateur connecté tout au long de sa navigation.

Dès l'entrée sur la page principale, si l'utilisateur n'est pas authentifié, il est redirigé automatiquement vers la page de connexion. Cela garantit que seuls les utilisateurs enregistrés peuvent accéder à la fonction "ajouter un commentaire".

Pour sécuriser la fonctionnalité de connexion, j'ai mis en place plusieurs bonnes pratiques. Tout d'abord, les identifiants des utilisateurs sont vérifiés à l'aide d'une requête SQL préparée, ce qui empêche toute tentative d'injection SQL dans le champ de login. Le mot de passe, lui, est stocké de manière sécurisée dans la base de données grâce à la fonction `password_hash()`, et validé à la connexion avec `password`. Cela garantit que même si la base est compromise, les mots de passe restent chiffrés et inutilisables.

Le formulaire HTML impose que tous les champs soient remplis avant de pouvoir envoyer les données, grâce à l'attribut `required`, ce qui évite les requêtes incomplètes côté serveur. Enfin, le système de session permet de garder l'utilisateur connecté de manière sécurisée, tout en bloquant l'accès aux pages sensibles pour les utilisateurs non authentifiés. Ces mesures assurent une navigation fiable et protègent les données des utilisateurs contre les attaques courantes.

Une fois connecté, l'utilisateur voit s'afficher un message personnalisé avec son nom d'utilisateur, ainsi qu'un bouton de déconnexion.

Les commentaires sont affichés dynamiquement depuis une base de données MySQL. Chaque message est lié à son auteur (grâce à une clé étrangère `id_utilisateur`) et affiché avec la date et le nom de l'utilisateur.

J'ai aussi sécurisé le formulaire d'ajout de commentaire. Côté back-end, j'utilise des requêtes SQL préparées avec `bind_param()` pour éviter toute injection SQL. Pour garantir la sécurité

des données, j'utilise `htmlspecialchars()` afin d'éviter les injections de scripts ou de HTML malveillant dans les champs saisis. J'ai également ajouté des contrôles pour valider que le champ n'est pas vide, et qu'il ne dépasse pas une certaine longueur. Cela garantit une saisie propre et sécurisée par les utilisateurs.

Le bouton "Ajouter un commentaire" est affiché uniquement lorsque l'utilisateur est connecté, ce qui renforce l'aspect dynamique de l'interface. Le retour utilisateur est également pris en compte grâce à des messages de confirmation ou d'erreur après chaque action (connexion, ajout de commentaire...).

Côté client, le formulaire comporte trois champs obligatoires : le login, le prénom et le mot de passe. Grâce à l'attribut `required`, le navigateur bloque l'envoi du formulaire si un champ est vide.

Lors de l'inscription, le mot de passe est haché à l'aide de la fonction `password_hash()`

Après chaque action (comme la connexion, l'inscription ou l'ajout de commentaire), des messages de retour sont affichés dynamiquement pour informer l'utilisateur du succès ou de l'échec de l'opération. Cela améliore l'expérience utilisateur en apportant un retour immédiat.

Le site propose également une page de modification permettant aux utilisateurs connectés de modifier leurs informations personnelles, notamment leurs prénom, nom et mot de passe. Pour sécuriser cette fonctionnalité, j'ai prévu un double champ pour la saisie du nouveau mot de passe, avec une confirmation obligatoire. Le mot de passe n'est modifié que si les deux champs sont identiques, ce qui permet d'éviter les erreurs de saisie.

Une page dédiée (`livre-or.php`) permet à tous les utilisateurs authentifiés de consulter l'ensemble des commentaires. Ceux-ci sont classés du plus récent au plus ancien, et affichés sous la forme :

"Posté le jour/mois/année par utilisateur"

suivi du message.

J'ai également développé une fonctionnalité de déconnexion qui détruit la session de l'utilisateur, renforçant la sécurité de l'application et empêchant tout accès non autorisé après une session inactive.

Le code est organisé de manière logique avec des fichiers dédiés pour chaque fonctionnalité (connexion, inscription, ajout de commentaire), ce qui facilite la lisibilité, la maintenance et l'évolution future du projet.

Le projet est structuré en fichiers dédiés pour chaque fonctionnalité (connexion, inscription, commentaires...), facilitant la lisibilité et la maintenance. Il a été développé en local avec Laragon pour la gestion du serveur et de la base de données (via PhpMyAdmin). Le code a été versionné sur GitHub, ce qui m'a permis de suivre les évolutions et de revenir à des versions antérieures si nécessaire.

Qu'est ce que le DOM ?

Le DOM (Document Object Model) est une représentation en mémoire d'une page web, comme une structure en arbre où chaque élément (titre, paragraphe, image, bouton, etc.) est une branche. Grâce au DOM, JavaScript peut accéder à ces éléments, les modifier, les supprimer ou en créer de nouveaux dynamiquement, sans recharger la page.

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

2.1 Mettre en place une base de données relationnelle.

Qu'est ce qu'une base de données relationnelle ?

Une base de données relationnelle est un système qui permet de stocker et organiser des données dans des tables, un peu comme un tableur Excel. Chaque table contient des lignes (appelées enregistrements) et des colonnes (appelées champs).

Les tables peuvent être liées entre elles grâce à des identifiants. Par exemple, une table "Clients" peut être liée à une table "Commandes" pour savoir quel client a passé quelle commande.

Dans une base de données, chaque donnée a un type spécifique qui indique la nature de l'information stockée.

Un exemple avec ce tableau:

Type ou Élément	Définition	Exemples d'utilisation
INT	Nombre entier	Identifiants, âges
VARCHAR(n)	Texte court, limité à n caractères	Noms, emails, titres
TEXT	Texte long	Descriptions, commentaires
DATE	Date (YYYY-MM-DD)	Anniversaires, sorties
DATETIME	Date + heure	Horodatage complet
BOOLEAN	Vrai ou faux	Champs oui/non
FLOAT / DECIMAL	Nombre avec virgule	Prix, notes
BLOB	Données binaires (image, audio...)	Images, fichiers
PRIMARY KEY	Identifiant unique de chaque ligne	Clé primaire d'une table
AUTO_INCREMENT	Incrémenter automatiquement un champ	ID automatique
FOREIGN KEY	Lien vers une autre table	Liaison entre tables
NOT NULL	Interdiction de laisser vide	Champs obligatoires
DEFAULT	Valeur par défaut si aucune saisie	Valeur assignée si vide

Dans le cadre de mon projet (livre d'or) j'ai mis en place une base de données relationnelle en MySQL.

Qu'est-ce que MySQL ?

MySQL est un système de gestion de base de données relationnelle (SGBDR). Il permet de stocker, organiser et gérer des données dans des tables, en utilisant le langage SQL (Structured Query Language).

La première étape a été d'analyser et de modéliser la base de données :

- les besoins fonctionnels: il s'agit de permettre aux utilisateurs de pouvoir se connecter, s'inscrire, modifier leur profil et de laisser des commentaires.

A partir de cette analyse j'ai identifié deux entités nécessaires à la base :

Conception des tables:

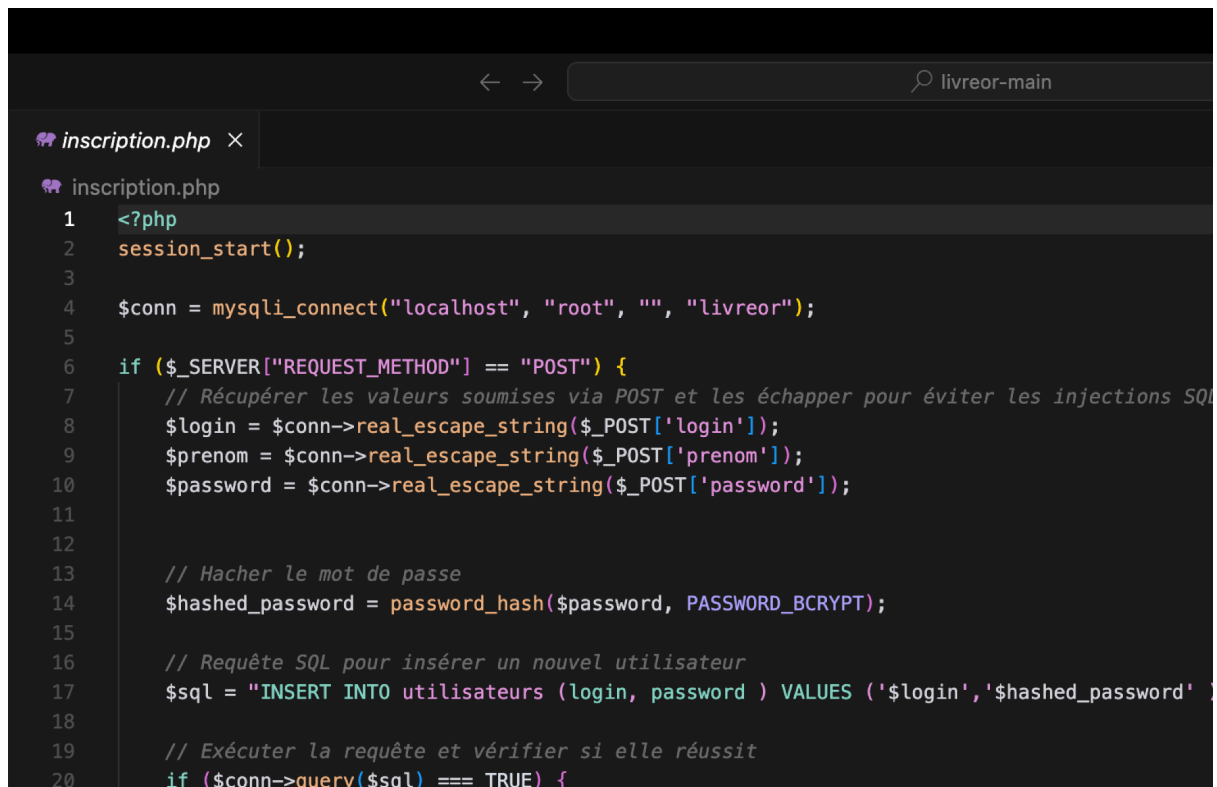
J'ai créé deux tables principales :

- utilisateurs : qui stocke les informations des utilisateurs inscrits
 - . id (clé primaire, auto-incrémentée)
 - . login (unique)
 - . password (haché avec password_hash)
 - . prénom , nom
- commentaires : qui contient les messages laissés par les utilisateurs
 - . id (clé primaire)
 - . commentaire (texte)
 - . date (date d'insertion automatique)
 - . id_utilisateur (clé étrangère vers [utilisateurs.id](#))

Chaque table possède une clé primaire (id) qui garantit l'unicité de chaque enregistrement. C'est pour cela que la clé étrangère (id_utilisateur) qui se trouve dans la table commentaires fait référence à la table utilisateurs .

Pour connecter ces tables au site j'ai utilisé ce code:

```
$conn = mysqli_connect("localhost", "root", "", "livreor");
```

A screenshot of a code editor with a dark theme. The editor shows a file named 'inscription.php'. The code is a PHP script for user registration. It starts with a session start, connects to a MySQL database named 'livreor' on localhost with root user. It checks if the request method is POST, then escapes the login, prenom, and password. The password is hashed using PASSWORD_BCRYPT. An SQL INSERT statement is prepared to add a new user to the 'utilisateurs' table. The query is executed, and a success message is shown if the query returns true.

```
1 <?php
2 session_start();
3
4 $conn = mysqli_connect("localhost", "root", "", "livreor");
5
6 if ($_SERVER["REQUEST_METHOD"] == "POST") {
7     // Récupérer les valeurs soumises via POST et les échapper pour éviter les injections SQL
8     $login = $conn->real_escape_string($_POST['login']);
9     $prenom = $conn->real_escape_string($_POST['prenom']);
10    $password = $conn->real_escape_string($_POST['password']);
11
12
13    // Hacher le mot de passe
14    $hashed_password = password_hash($password, PASSWORD_BCRYPT);
15
16    // Requête SQL pour insérer un nouvel utilisateur
17    $sql = "INSERT INTO utilisateurs (login, password ) VALUES ('$login','$hashed_password' )";
18
19    // Exécuter la requête et vérifier si elle réussit
20    if ($conn->query($sql) === TRUE) {
```

Relations et cardinalités:

Qu'est ce que la cardinalité ?

La cardinalité décrit le nombre de relations possibles entre les deux entités dans une base de données relationnelle.

Cette relation de type un-à-plusieurs (1-N) permet de relier plusieurs commentaires à un même utilisateur.

La relation entre les deux table suit une cardinalité 1 -> N:

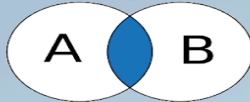
- Un utilisateur peut poster plusieurs commentaires.
- Un commentaire est associé à un seul utilisateur.

On peut dire qu'une ligne de la table utilisateurs peut être reliée à plusieurs lignes de la table commentaires , mais chaque commentaire est associé à un seul utilisateur.

Elle permet d'assurer l'intégrité référentielle, c'est-à-dire que chaque commentaire est toujours lié à un utilisateur existant. En cas de suppression d'un utilisateur, il est possible d'ajouter une règle on delete cascade (contrainte définie dans une relation entre deux tables en base de données. Elle est utilisée lors de la création d'une clé étrangère (foreign key) pour automatiser la suppression des enregistrements liés.)pour supprimer automatiquement ses commentaires.

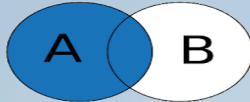
SQL JOINS

INNER JOIN



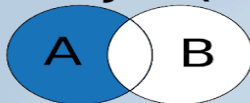
```
SELECT *  
FROM A  
INNER JOIN B ON A.key = B.key
```

LEFT JOIN



```
SELECT *  
FROM A  
LEFT JOIN B ON A.key = B.key
```

LEFT JOIN (sans l'intersection de B)



```
SELECT *  
FROM A  
LEFT JOIN B ON A.key = B.key  
WHERE B.key IS NULL
```

RIGHT JOIN



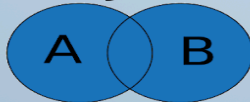
```
SELECT *  
FROM A  
RIGHT JOIN B ON A.key = B.key
```

RIGHT JOIN (sans l'intersection de A)



```
SELECT *  
FROM A  
RIGHT JOIN B ON A.key = B.key  
WHERE B.key IS NULL
```

FULL JOIN



```
SELECT *  
FROM A  
FULL JOIN B ON A.key = B.key
```

FULL JOIN (sans intersection)



```
SELECT *  
FROM A  
FULL JOIN B ON A.key = B.key  
WHERE A.key IS NULL  
OR B.key IS NULL
```

sql.sh

En SQL , une jointure permet de lier plusieurs tables en fonction d'un ou plusieurs champs communs, généralement une clé étrangère.

INNER JOIN : ne récupère que les lignes qui ont une correspondance dans les deux tables (A et B).

LEFT JOIN: récupère toutes les lignes de la table A, et les lignes correspondantes de B. Si aucune correspondance, les valeurs de B sont nulles.

LEFT JOIN (sans intersection) : ne récupère que les lignes de A sans correspondance dans B.

RIGHT JOIN: récupère toutes les lignes de la table B, et les lignes correspondantes de A.

RIGHT JOIN (sans intersection) : ne récupère que les lignes de B sans correspondance dans A.

FULL JOIN : récupère toutes les lignes de A et B, en affichant les correspondances quand elles existent, sinon avec des valeurs nulles.

FULL JOIN (sans intersection) : récupère uniquement les lignes de A et B qui n'ont pas de correspondance entre elles.

Qu'est ce que le MLD ? Modèle Logique de Données

Le Modèle Logique de Données (MLD) est une représentation structurée des données d'un système d'information, conçue à partir du Modèle Conceptuel de Données (MCD). Il décrit les tables, les colonnes, les types de données, ainsi que les relations entre les entités, en tenant compte des règles de gestion. Contrairement au MCD, le MLD est plus proche de la réalité d'une base de données relationnelle, mais reste indépendant d'un système de gestion de base de données (SGBD) particulier. Il constitue une étape essentielle avant la création du schéma physique de la base de données.

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

Pour illustrer cette compétence, j'ai choisi de présenter Cinetech, un site web dynamique que j'ai développé selon l'architecture MVC (Modèle-Vue-Contrôleur). Cinetech est une bibliothèque de films qui permet de consulter des œuvres classées par genres et catégories (films ou séries). Le site utilise également une API pour récupérer et gérer les données des films, ce qui facilite la séparation entre la logique métier, la gestion des données et l'affichage, tout en assurant une meilleure maintenabilité et évolutivité de l'application.

Qu'est ce qu'un composant métier côté serveur ?

Un composant d'accès aux données SQL est une partie du code d'une application qui sert d'interface entre la logique métier et la base de données relationnelle, en utilisant le langage SQL pour effectuer des opérations de consultation, modification, création ou suppression de données.

Le site propose plusieurs fonctionnalités :

- Création de compte et connexion utilisateur,
- Ajout de films en favoris,
- Publication de commentaires et d'avis,
- Recherche de films par nom, genre ou catégorie,
- Consultation des informations détaillées de chaque film : synopsis, casting, année de sortie, etc.

Le projet repose sur une base de données MySQL, conçue pour stocker les utilisateurs, les films favoris, les avis et les commentaires. J'ai utilisé PDO (PHP Data Objects) pour gérer toutes les interactions avec cette base. PDO m'a permis de :

- Établir une connexion sécurisée à la base de données,
- Utiliser des requêtes préparées avec `bindParam()` afin d'éviter les injections SQL,
- Gérer les erreurs grâce aux exceptions `PDOException`.

Qu'est ce que la POO ?

La POO, ou Programmation Orientée Objet, est une manière d'organiser et d'écrire du code en se basant sur des objets, c'est-à-dire des éléments qui regroupent des données (appelées *attributs*) et des actions (appelées *méthodes*).

Exemple avec le projet:

Chaque classe correspond à une entité métier (Livre, Catégorie, Sous-catégorie, Utilisateur). Chaque objet est une instance avec des propriétés (attributs) et des méthodes (fonctions) propres.

On utilise des méthodes pour encapsuler la logique métier (ex : récupérer tous les livres d'une catégorie).

La connexion à la base est souvent gérée dans une classe dédiée (ex : Database) pour éviter

la duplication.

Cela facilite la maintenance, la réutilisation et la compréhension du code.

Qu'est ce que le mvc ?

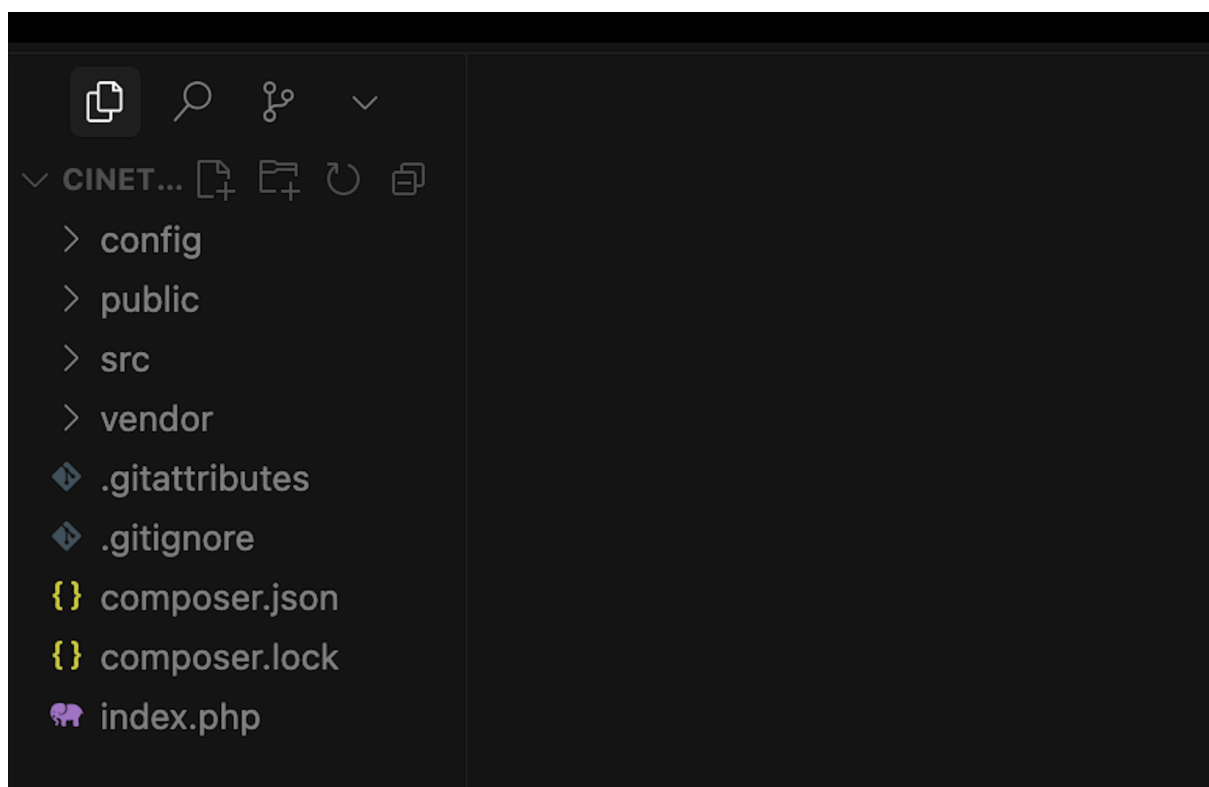
Le MVC est un modèle d'organisation du code utilisé pour structurer une application, en séparant clairement ses différentes parties pour faciliter sa compréhension et sa maintenance. Le sigle MVC signifie Modèle, Vue, Contrôleur.

Le Modèle est la partie qui gère les données et la logique métier. C'est lui qui interagit avec la base de données, récupère ou modifie les informations comme les livres, les catégories, etc. Il contient tout ce qui concerne la manipulation des données.

La Vue correspond à l'interface visible par l'utilisateur, c'est la page web affichée dans le navigateur. Elle sert à présenter les données fournies par le modèle de façon claire et agréable. La vue ne traite pas les données, elle se contente de les afficher.

Le Contrôleur fait le lien entre la vue et le modèle. Il reçoit les actions de l'utilisateur, comme cliquer sur un lien ou envoyer un formulaire, puis il demande au modèle d'exécuter les opérations nécessaires. Ensuite, il choisit la vue à afficher pour montrer le résultat à l'utilisateur.

Ainsi, quand un utilisateur demande à voir la liste des livres, le contrôleur interroge le modèle pour récupérer ces livres, puis transmet les données à la vue qui va les afficher. Cette organisation en trois parties distinctes permet de rendre le code plus clair, plus facile à maintenir et à faire évoluer.



Qu'est ce qu'une Api ?

Une API (Application Programming Interface) est un ensemble de règles et de protocoles qui permet à différentes applications ou systèmes de communiquer entre eux.

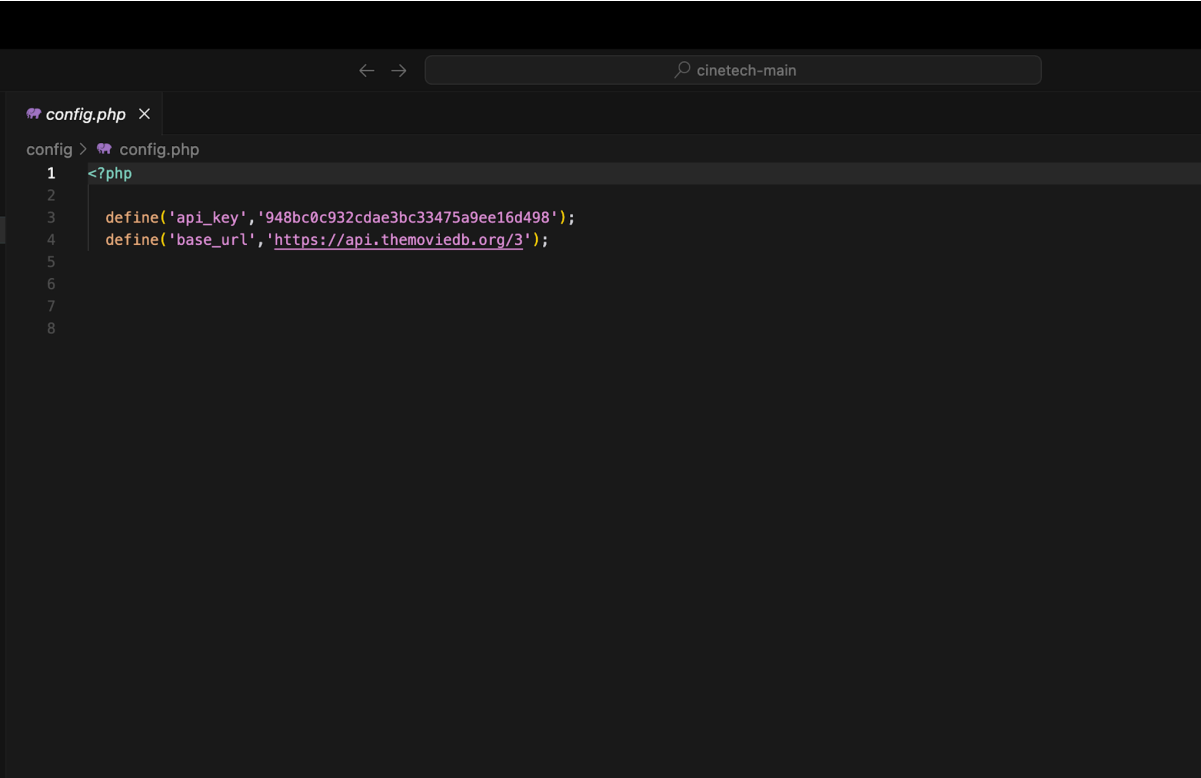
Exemple, c'est comme un distributeur automatique de boissons. Tu appuies sur un bouton (tu envoies une requête), le distributeur reçoit ta demande et prépare la boisson correspondante (le système traite la demande), puis il te délivre la boisson choisie (la réponse).

Dans Cinetech, j'utilise l'API de TMDB (The Movie Database) pour récupérer des informations sur les films, comme le titre, la description, l'affiche, la date de sortie, etc.

Pour récupérer les films populaire:

\$url =

["https://api.themoviedb.org/3/movie/popular?api_key={\\$this->apiKey}&language=fr-FR"](https://api.themoviedb.org/3/movie/popular?api_key={$this->apiKey}&language=fr-FR);



```
config.php x
config > config.php
1 <?php
2
3 define('api_key', '948bc0c932cdae3bc33475a9ee16d498');
4 define('base_url', 'https://api.themoviedb.org/3');
5
6
7
8
```

Grâce à cette API, je n'ai pas besoin de créer ni gérer ma propre base de données de films : j'utilise directement celles de TMDB de manière simple, propre et organisée.

L'API TMDB (The Movie Database API) est une interface fournie par le site The Movie Database (TMDB) qui permet aux développeurs d'accéder à une grande base de données de films, séries, acteurs, réalisateurs, affiches, résumés, bandes-annonces, etc.

Comment j'ai utilisé l'api TMDB dans mon projet :

Dans mon projet Cinetech, la configuration de l'API TMDB, comprenant la clé d'accès et l'URL de base, est stockée dans le fichier App/Config/config.php.

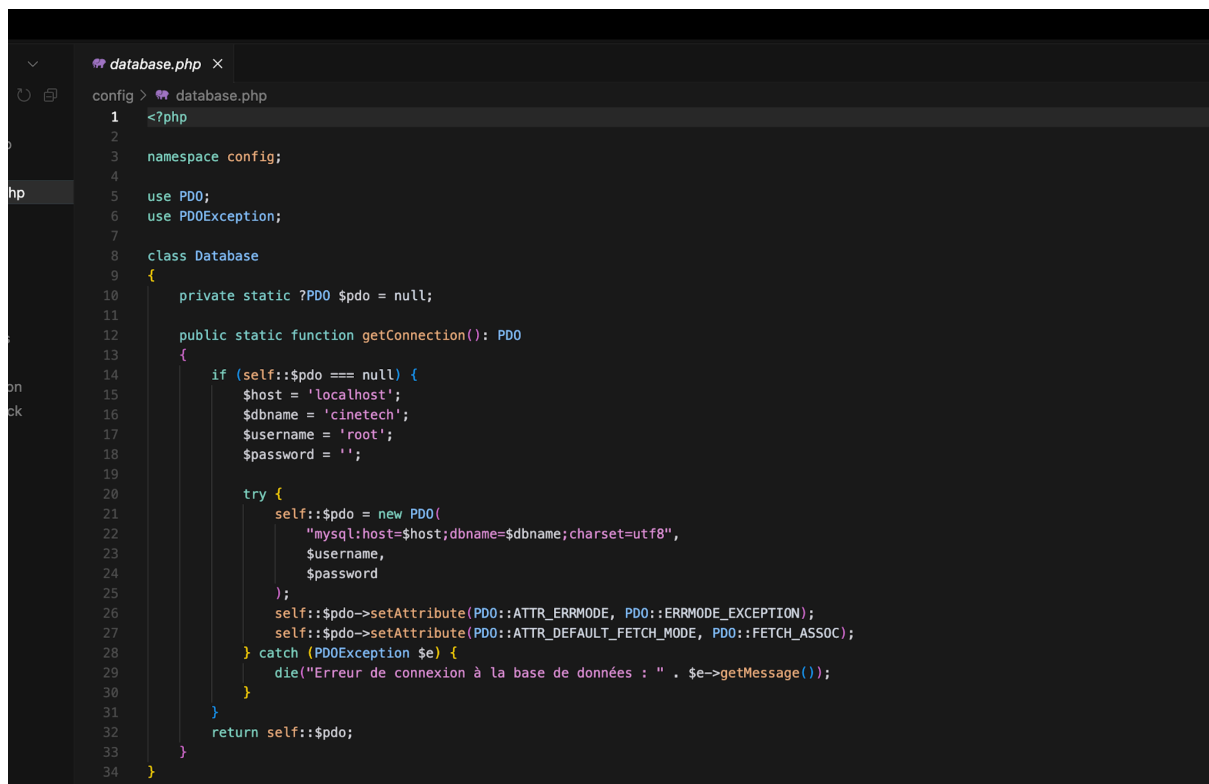
J'ai ensuite créé un service dans App/Models/TMDBservice.php qui contient différentes méthodes permettant de récupérer les films populaires, les détails d'un film, et d'autres données utiles.

Ces méthodes sont utilisées dans les contrôleurs, notamment dans App/Controllers/HomeController.php, pour afficher diverses catégories de films comme les plus populaires, les mieux notés, les films à venir ou ceux actuellement à l'affiche.

Qu'est ce que le PDO ?

PDO, pour PHP Data Objects, est une extension de PHP qui permet de se connecter à une base de données de manière sécurisée et uniforme, quel que soit le type de base (MySQL, PostgreSQL, SQLite, etc.).

PDO rend le code plus propre, protège contre les injections SQL grâce aux requêtes préparées, et permet de travailler avec plusieurs types de bases de données sans avoir à réécrire tout le code. C'est justement cette méthode que j'ai utilisée dans mon projet Cinetech, notamment pour gérer la connexion à la base de données via phpMyAdmin.



```
1 <?php
2
3 namespace config;
4
5 use PDO;
6 use PDOException;
7
8 class Database
9 {
10     private static ?PDO $pdo = null;
11
12     public static function getConnection(): PDO
13     {
14         if (self::$pdo === null) {
15             $host = 'localhost';
16             $dbname = 'cinetech';
17             $username = 'root';
18             $password = '';
19
20             try {
21                 self::$pdo = new PDO(
22                     "mysql:host=$host;dbname=$dbname;charset=utf8",
23                     $username,
24                     $password
25                 );
26                 self::$pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
27                 self::$pdo->setAttribute(PDO::ATTR_DEFAULT_FETCH_MODE, PDO::FETCH_ASSOC);
28             } catch (PDOException $e) {
29                 die("Erreur de connexion à la base de données : " . $e->getMessage());
30             }
31         }
32         return self::$pdo;
33     }
34 }
35
```

Le No SQL

NoSQL (qui signifie "Not Only SQL") est un type de base de données qui ne suit pas le modèle relationnel traditionnel. Contrairement à MySQL ou PostgreSQL (qui utilisent des tables, des relations et du SQL), les bases NoSQL sont conçues pour gérer des données non structurées ou semi-structurées, souvent de manière plus flexible et scalable (adaptée aux gros volumes).

Dans ce projet, je n'ai pas eu recours à une base de données NoSQL, car les besoins étaient majoritairement relationnels : la structure des données exigeait des relations fortes entre utilisateurs, commentaires et films (clés primaires/étrangères, jointures, etc.). Toutefois, je comprends que les bases NoSQL comme MongoDB sont utiles pour des projets nécessitant une grande flexibilité de structure, une forte scalabilité ou un traitement rapide de volumes non structurés.

2.3 Développer des composants métier côté serveur.

Dans le projet Cinetech, une plateforme de consultation de films et séries, j'ai mis en place une architecture MVC (Modèle-Vue-Contrôleur). Cette architecture permet une séparation claire entre la présentation (Vue), la logique de traitement (Contrôleur), et la gestion des données (Modèle).

Qu'est ce qu'un composant métier côté serveur ?

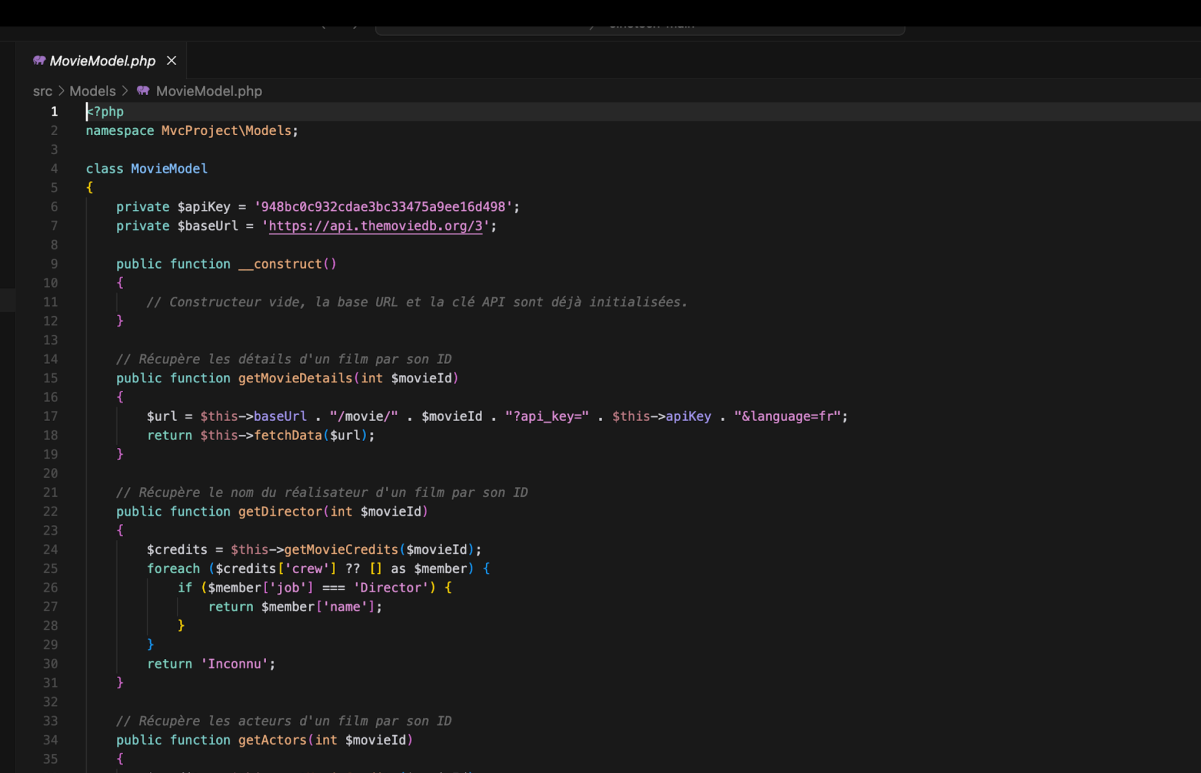
Un composant métier côté serveur est un élément logiciel du backend d'une application web ou mobile qui implémente une partie de la logique métier, c'est-à-dire les règles et traitements propres au domaine d'activité de l'application (gestion des utilisateurs, traitement de commandes, gestion des paiements, etc.).

Les composants métier sont localisés dans le dossier src/Models.

Leur rôle principal est de gérer toutes les opérations liées aux règles métier et à la manipulation des données. Cela inclut notamment la communication avec l'API externe TMDb (The Movie Database) : les modèles sont responsables des appels HTTP vers l'API, en utilisant la bibliothèque cURL pour un contrôle précis (gestion des timeouts, des erreurs, des réponses HTTP, etc.). Une fois les données récupérées, les composants métier se chargent

de les filtrer, de les transformer ou de les agréger selon les besoins de l'application. Enfin, les modèles exposent ces données de manière structurée aux contrôleurs, qui les transmettent ensuite à la vue pour affichage.

Par exemple, le fichier `MovieModel.php` est un composant métier central. Il encapsule toute la logique liée à la gestion des films, notamment la récupération des listes de films populaires, des détails d'un film ou des recherches par titre. Il gère aussi les paramètres d'appel à l'API (clé API, endpoints, pagination, etc.), ainsi que le traitement et la validation des réponses reçues avant de les transmettre au contrôleur.



```
1 1?php
2 namespace MvcProject\Models;
3
4 class MovieModel
5 {
6     private $apiKey = '948bc0c932cdae3bc33475a9ee16d498';
7     private $baseUrl = 'https://api.themoviedb.org/3';
8
9     public function __construct()
10    {
11        // Constructeur vide, la base URL et la clé API sont déjà initialisées.
12    }
13
14    // Récupère les détails d'un film par son ID
15    public function getMovieDetails(int $movieId)
16    {
17        $url = $this->baseUrl . "/movie/" . $movieId . "?api_key=" . $this->apiKey . "&language=fr";
18        return $this->fetchData($url);
19    }
20
21    // Récupère le nom du réalisateur d'un film par son ID
22    public function getDirector(int $movieId)
23    {
24        $credits = $this->getMovieCredits($movieId);
25        foreach ($credits['crew'] ?? [] as $member) {
26            if ($member['job'] === 'Director') {
27                return $member['name'];
28            }
29        }
30        return 'Inconnu';
31    }
32
33    // Récupère les acteurs d'un film par son ID
34    public function getActors(int $movieId)
35    {
36        $credits = $this->getMovieCredits($movieId);
```

Le fichier `TvModel.php` est structuré de manière similaire à `MovieModel.php`, mais il concerne spécifiquement la gestion des séries télévisées. Cette approche permet de conserver une cohérence dans la structure du code tout en répondant à des besoins métiers distincts. Parmi les principales fonctionnalités de `TvModel.php`, on retrouve :

- `getTvDetails($tvId)` : récupère les informations détaillées d'une série à partir de son identifiant.
- `getGenres()` : récupère la liste des genres disponibles pour les séries télévisées.
- `getTvShowsByGenre($genreId)` : permet de filtrer et d'obtenir les séries selon un genre spécifique.
- `getDirector($tvId)` : recherche le réalisateur ou le producteur exécutif associé à une série.
- `getActors($tvId)` : affiche les principaux membres du casting d'une série, en limitant la liste à cinq acteurs.

```

1  <?php
2  namespace MvcProject\Models;
3
4  class TvModel
5  {
6      private $apiKey = '948bc0c932cdae3bc33475a9ee16d498';
7      private $baseUrl = 'https://api.themoviedb.org/3';
8
9      public function getTvDetails(int $tvId)
10     {
11         if (!$this->validateId($tvId)) {
12             return null;
13         }
14         $url = $this->baseUrl . "/tv/" . $tvId . "?api_key=" . $this->apiKey . "&language=fr";
15         return $this->fetchData($url);
16     }
17
18     public function getGenres()
19     {
20         $url = $this->baseUrl . "/genre/tv/list?api_key=" . $this->apiKey . "&language=fr";
21         $response = $this->fetchData($url);
22         return $response['genres'] ?? []; // Retourne les genres ou un tableau vide
23     }
24
25     public function getTvShowsByGenre(int $genreId, int $page = 1)
26     {
27         if (!$this->validateId($genreId)) {
28             return [];
29         }
30         $url = $this->baseUrl . "/discover/tv?api_key=" . $this->apiKey . "&language=fr&with_genres=" . $genreId . "&page=" . $page;
31         $response = $this->fetchData($url);
32         return $response; // Retourne tous les résultats de la requête (y compris les pages, résultats, etc.)
33     }
34
35     public function getDirector(int $tvId)
36     {
37         $credits = $this->getTvCredits($tvId);
38         foreach ($credits['crew'] as $credit) {
39             if ($credit['job'] === 'Director') {
40                 return $credit['name'];
41             }
42         }
43         return null;
44     }
45 }

```

Comme pour le MovieModel, ce fichier contient également une méthode fetchData qui centralise les appels à l'API, ainsi qu'une méthode de validation des identifiants (validateId) afin de garantir la fiabilité et la sécurité des requêtes effectuées.

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

Cinotech est une application web dynamique qui permet de consulter les films et séries du moment, d'accéder aux détails d'une œuvre et de gérer une liste de favoris. L'application repose sur une architecture MVC (Modèle-Vue-Contrôleur) et utilise l'API externe TMDb pour récupérer les contenus.

Pré-requis techniques

Avant de déployer le projet Cinotech, il faut disposer des éléments suivants :

- Un serveur local (comme Laragon, XAMPP ou WAMP) ou un hébergeur web prenant en charge PHP et MySQL.
- Un éditeur de code, par exemple Visual Studio Code.
- Une connexion Internet pour interagir avec l'API TMDb.
- Une base de données MySQL disponible.
- Une clé API TMDb, à obtenir sur <https://www.themoviedb.org/>.

Structure du projet

- public/ : fichiers accessibles publiquement (CSS, JS, images)
- src/Controllers/ : contient les contrôleurs pour chaque fonctionnalité
- src/Models/ : gère les données et les requêtes SQL
- src/Views/ : vues affichées à l'utilisateur
- config/ : fichiers de configuration (connexion à la base de données, constantes)
- index.php : point d'entrée de l'application

Étapes de déploiement

1. Télécharger le projet
Cloner ou copier les fichiers du projet sur le serveur web (par exemple dans le dossier www ou htdocs).
2. Configurer la base de données
 - Créer une base de données MySQL dédiée à l'application.
 - Importer le fichier cinotech.sql (ou le script de création des tables fourni).
 - Vérifier et renseigner les identifiants de connexion dans le fichier de configuration (par exemple db.php).
3. Configurer l'accès à l'API
 - Créer un compte sur TMDb et obtenir une clé API.
 - Définir cette clé dans le fichier de configuration approprié (config.php ou .env si utilisé).

4. Vérifier l'arborescence

- S'assurer que les chemins des fichiers (CSS, JS, images) sont corrects.
- Vérifier que les routes (URL) fonctionnent selon l'environnement de déploiement.

5. Tester l'application

- Lancer l'application dans un navigateur web.
- Tester la navigation, la récupération des films et séries, l'ajout aux favoris, et les principales fonctionnalités.

Cette documentation synthétise les étapes essentielles pour installer, configurer et tester l'application Cinetech sur un serveur local ou distant.

Il est important de faire une documentation pour un projet afin de garder une trace claire de tout ce qui a été fait. La documentation explique comment le projet fonctionne, comment l'installer et l'utiliser, et quelles étapes ont été suivies.

Cela aide les membres de l'équipe à mieux comprendre le projet et facilite le travail en groupe. De plus, si quelqu'un doit reprendre le projet plus tard, la documentation lui permettra de s'y retrouver rapidement. Elle sert aussi à résoudre plus facilement les problèmes et à améliorer le projet dans le futur.

En résumé, faire une documentation, c'est écrire tout ce qu'il faut savoir sur le projet pour que tout soit clair et facile à utiliser pour tout le monde.