

Labo Développement Web

Plateforme de partage iLearn

Samou Patrick

Vatafu Jean

patrick.samou@student.hers.be

jean.vatafu@student.hers.be

Bachelier en Informatique - Bloc 2

Année académique 2025-2026

Table des matières

1	Introduction	1
2	Choix d'Implémentation	1
2.1	Architecture et Classe Database	1
2.2	Sécurité et Gestion des Sessions	1
3	Fonctionnement de l'application	1
3.1	Interface Utilisateur	1
3.2	Interface ressources	2
4	Difficultés rencontrées et Solutions	3
4.1	Gestion des fichiers (Upload)	3
4.2	Conception de la classe Database	3
4.3	Connexion au serveur de base de données	3
4.4	Syntaxe MySQL vs SQL Developer	3
5	Déploiement	4
6	Organisation travail	4
7	Identifiants de test	4
8	Conclusion	4

1 Introduction

Pour ce projet de labo, nous avons créé **iLearn**. C'est une application web qui sert à regrouper toutes les ressources de cours au même endroit. L'idée, c'est que les étudiants puissent envoyer des documents et que les administrateurs vérifient si le contenu est correct avant de le valider. On a fait attention à bien structurer notre code et à sécuriser le site, comme on l'a appris durant ce quadrimestre en Bloc 2.

2 Choix d'Implémentation

2.1 Architecture et Classe Database

Nous avons décidé d'utiliser la programmation orientée objet pour gérer nos données. On a créé une classe `Database.php` qui s'occupe de tout ce qui touche à la base de données.

- **Pourquoi ce choix :** En mettant toutes les requêtes SQL au même endroit, on évite de copier-coller le même code partout. C'est beaucoup plus simple pour trouver les erreurs et modifier le code plus tard.

2.2 Sécurité et Gestion des Sessions

Pour savoir si un utilisateur est connecté, nous utilisons les sessions PHP (`$_SESSION`).

- **Pourquoi ce choix :** Les sessions restent sur le serveur, donc l'utilisateur ne peut pas modifier ses droits tout seul. Pour éviter que des personnes malveillantes n'attaquent la base de données, on utilise des requêtes préparées avec PDO.

3 Fonctionnement de l'application

3.1 Interface Utilisateur

Quand on arrive sur le site, on voit les cours classés par blocs. On a voulu faire quelque chose de simple et de propre pour que les étudiants trouvent vite leurs cours et les fichiers associés.

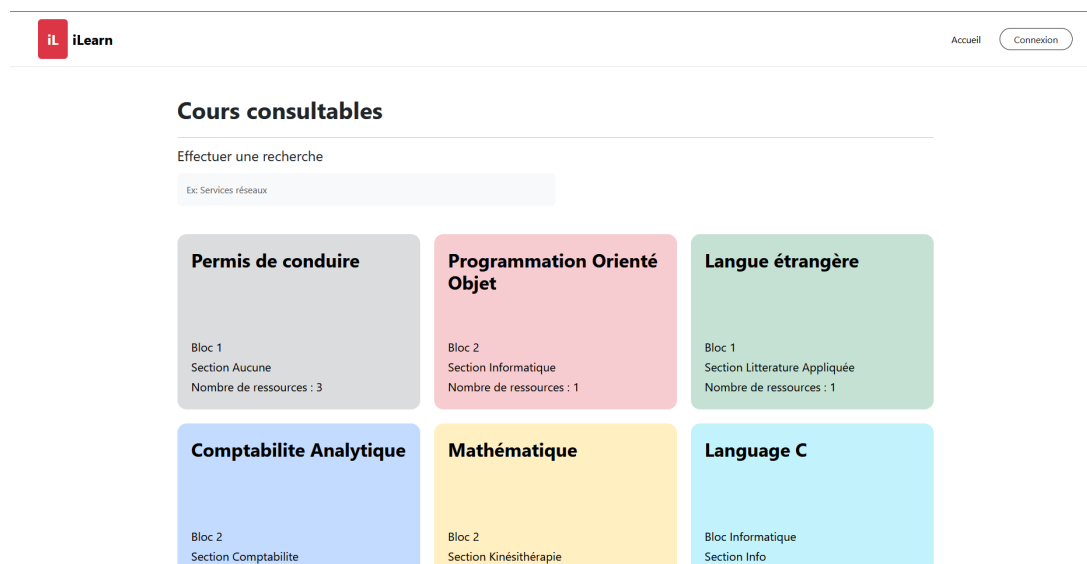


FIGURE 1 – Aperçu de la liste des cours : utilisateur non connecté

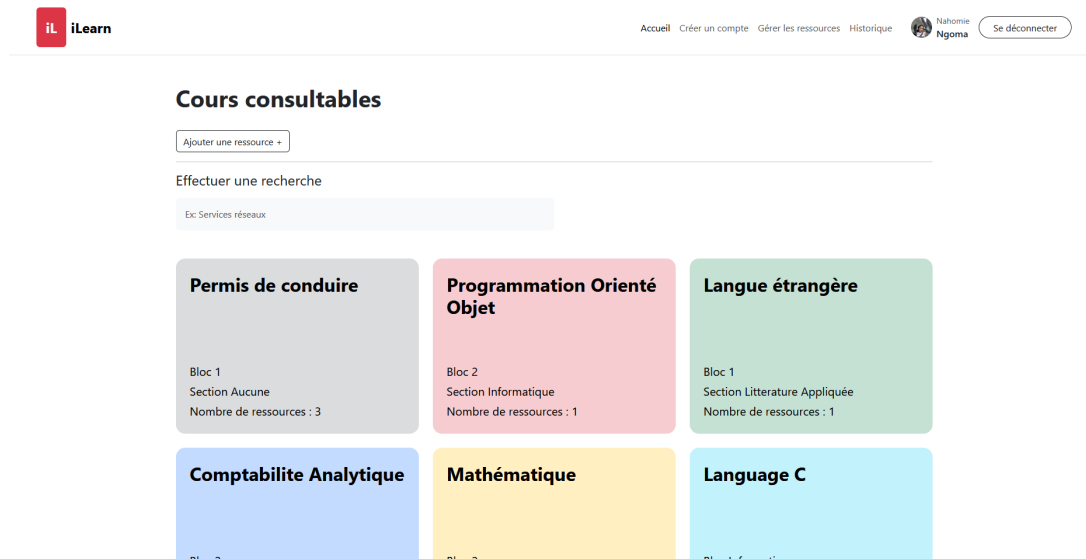


FIGURE 2 – Aperçu de la liste des cours : utilisateur connecté

3.2 Interface ressources

L'historique du projet a été dicté par le besoin de retrouver les fichiers rapidement. Sur le site, les ressources sont directement rattachées à des blocs de cours.

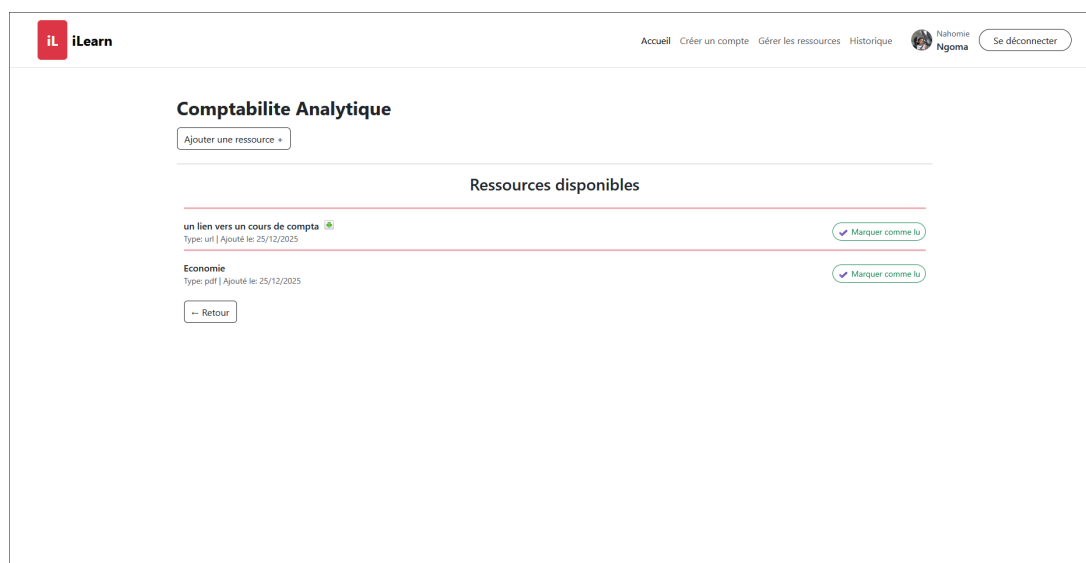


FIGURE 3 – Aperçu des ressources par cours

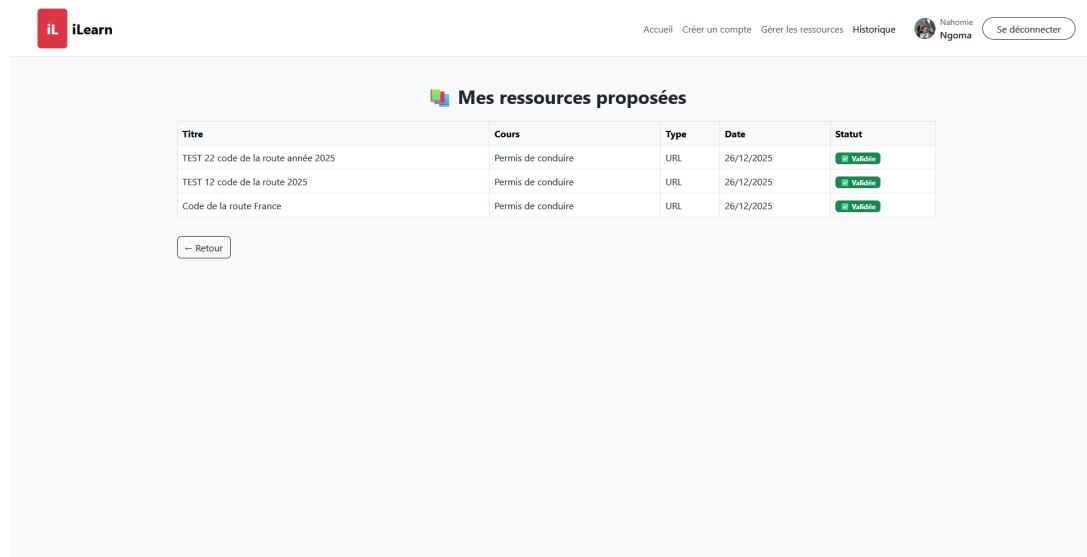


FIGURE 4 – Aperçu des ressources proposés

4 Difficultés rencontrées et Solutions

Pendant qu'on codait, on a eu quelques soucis techniques qu'on a dû régler :

4.1 Gestion des fichiers (Upload)

Le problème : Au début, si deux personnes envoyaient un fichier qui s'appelait `resume.pdf`, le deuxième fichier supprimait le premier sur le serveur.

La solution : On a utilisé la fonction `uniqid()`. Maintenant, chaque fichier reçoit un nom unique quand il est enregistré dans le dossier `uploads/ressources/`. Comme ça, plus rien ne s'écrase.

4.2 Conception de la classe Database

Le problème : Au départ, on ouvrait une nouvelle connexion à la base de données pour chaque petite requête, ce qui n'était pas très efficace et ralentissait le site.

La solution : On a modifié la classe `Database` pour que la connexion se fasse dans le constructeur. Comme ça, on utilise la même connexion tout le long, ce qui rend le site plus rapide.

4.3 Connexion au serveur de base de données

Le problème : Si le serveur SQL avait un problème, le site affichait des erreurs moches ou s'arrêtait d'un coup.

La solution : On a ajouté des blocs `try...catch`. Si la connexion ne marche pas, on capture l'erreur proprement pour afficher un message clair au lieu de laisser le système planter.

4.4 Syntaxe MySQL vs SQL Developer

Le problème : On a eu quelques difficultés avec la syntaxe propre à MySQL. Certains éléments sont écrits différemment par rapport à ce qu'on avait l'habitude de faire sur SQL Developer (Oracle).

La solution : Pour régler ça, on a fait pas mal de recherches sur Google et on a consulté la documentation officielle de MySQL pour trouver les bonnes équivalences et adapter nos scripts.

5 Déploiement

Pour faire fonctionner l'application chez vous, il faut :

1. Importer le fichier `init_project.sql` pour créer les tables et l'utilisateur `BDProjetPatrick&Jean`.
2. Vérifier que vous avez le droit d'écrire dans le dossier `uploads/`.
3. Lancer le tout sur un serveur local comme Apache.

6 Organisation travail

Pour la répartition du travail, nous avons fonctionné de manière simple et efficace :

- **Conception** : On a commencé par utiliser **Figma** pour créer les maquettes du site. Ça nous a permis de nous mettre d'accord sur le design avant de commencer à coder.
- **Développement** : Ensuite, on a décidé de travailler chacun de notre côté sur nos parties respectives.
- **Entraide** : Dès que l'un de nous avait un blocage ou une difficulté, on en discutait ensemble pour trouver une solution rapidement.
- **Finalisation** : Pour finir, on a tout mis en commun pour assembler le projet final.

7 Identifiants de test

Voici le compte pour tester la partie administration :

- **Admin 1** : Login `admin1` / MDP : `Samourai3`

8 Conclusion

Ce projet nous a bien aidé à comprendre comment utiliser PHP et les bases de données ensemble. On a réussi à régler nos problèmes de fichiers et à faire un code solide avec notre classe Database. On a maintenant une plateforme iLearn qui fonctionne bien et qui est sécurisée, ce qui nous prépare bien pour apprendre les frameworks plus tard.