📋 Compétences Référentiel BTS SIO SLAM

Projet BiblioTech - Séances 1 à 3

🎯 Vue d'Ensemble

Projet : BiblioTech - Application de gestion de bibliothèque

Durée : 9 heures (3 séances de 3h)

Technologies : Laravel 11, PHP 8.2, SQLite, Bootstrap 5, Docker

Période : Septembre - Octobre 2025

📚 BLOC B2 - CONCEPTION ET DÉVELOPPEMENT D'APPLICATIONS

B2.1 - Concevoir et développer une solution applicative

B2.1.1 - Analyser un besoin et proposer des solutions applicatives

Activités réalisées :

- ✅ Analyse des besoins d'une bibliothèque moderne
- ✅ Identification des **entités métier** : Livres, Utilisateurs, Emprunts

  > 💡 **Entité métier** : Objet du monde réel représenté en informatique (ex: un livre, un client)

- ✅ Choix de l'architecture **MVC** (Modèle-Vue-Contrôleur)

  > 💡 **MVC** : Façon d'organiser le code en 3 parties : Modèle (données), Vue (affichage), Contrôleur (logique)

B2.1.2 - Concevoir ou adapter l'interface utilisateur

Activités réalisées :

- ✅ Création de **templates Blade** responsive
  > 💡 **Template** : Modèle de page web réutilisable avec des zones variables
- ✅ Utilisation de **Bootstrap 5** pour le design
  > 💡 **Bootstrap** : Bibliothèque CSS qui fournit des styles prêts à l'emploi
- ✅ Navigation intuitive avec **fil d'Ariane**
  > 💡 **Fil d'Ariane** : Chemin de navigation (ex: Accueil > Livres > Harry Potter)
- ✅ Création de **composants** réutilisables
  > 💡 **Composant** : Morceau de code réutilisable (ex: une carte de livre)
- ✅ Interface **responsive** (mobile/tablette/desktop)
  > 💡 **Responsive** : Interface qui s'adapte automatiquement à la taille de l'écran

B2.1.3 - Concevoir ou adapter une base de données

Activités réalisées :

- ✅ Création de 3 **migrations** Laravel :
  > 💡 **Migration** : Fichier qui décrit la structure d'une table (comme un plan)

Migration 1 : Utilisateurs (utilisateurs)

- Fichier : `database/migrations/2024_01_01_000000_creer_table_utilisateurs.php`
- Champs français :
  - `id` : Identifiant unique
  - `nom` : Nom de famille
  - `prenom` : Prénom
  - `email` : Adresse email (unique)
  - `mot_de_passe` : Mot de passe chiffré

B2.1.4 - Définir les spécifications d'une solution applicative

Activités réalisées :

- ✅ Documentation technique complète par séance
- ✅ Schémas d'architecture (MVC, flux de données)
- ✅ Guide de structure des fichiers
- ✅ Spécifications fonctionnelles de chaque module

Preuves concrètes :

- README principal : `README.md` (lignes 1-150)
- Documentation séance 1 : `docs/SEANCE-1/README.md`
- Glossaire technique : `docs/GLOSSAIRE.md`
- Concepts expliqués : `docs/CONCEPTS.md`

Niveau de maîtrise : ★★★ Maîtrisé

B2.1.5 - Prototyper une solution applicative

Activités réalisées :

- ✅ **MVP** (Produit Minimum Viable) dès la séance 1
  > 💡 **MVP** : Version de base fonctionnelle, simple mais utilisable
- ✅ Approche **itérative** et **incrémentale**
  > 💡 **Itératif** : Développement par étapes successives qui s'améliorent
- ✅ Données de démonstration avec **Seeders**
  > 💡 **Seeder** : Fichier qui remplit la base avec des données de test
- ✅ Tests utilisateurs progressifs

Preuves concrètes :

- Version S1 : Affichage de livres statiques
  (`resources/views/livres/index.blade.php` , lignes 1-30)

B2.1.6 - Programmer ou adapter des composants logiciels

Activités réalisées :

Backend (Laravel 11, PHP 8.2) :

- ✅ **Contrôleurs** (Controleurs) :

  > 💡 **Contrôleur** : Fichier qui contient la logique métier (ce que fait l'application)

  - `LivreControleur` : Gestion des livres
    - Fichier : `app/Http/Controleurs/LivreControleur.php`
    - Méthode `index()` : Afficher la liste (lignes 20-25)
    - Méthode `stocker()` : Créer un livre (lignes 30-40)
    - Méthode `mettreAJour()` : Modifier un livre (lignes 45-55)
  - `AccueilControleur` : Page d'accueil

B2.1.7 - Manipuler des données

Activités réalisées :

- ✅ Utilisation d'**Eloquent ORM**

  > 💡 **ORM** : Outil qui permet de manipuler la base de données avec du code PHP simple au lieu de SQL

- ✅ **Requêtes** de base :

  > 💡 **Requête** : Demande d'informations à la base de données

  - `Livre::all()` : Récupérer tous les livres
    - Fichier : `app/Http/Controleurs/LivreControleur.php` (ligne 22)
  - `Livre::find($id)` : Trouver un livre par son ID
    - Fichier : `app/Http/Controleurs/LivreControleur.php` (ligne 35)

B2.2 - Assurer la maintenance corrective ou évolutive

B2.2.1 - Analyser et corriger un dysfonctionnement

Activités réalisées :

- ✅ **Debugging** avec `dd()` (dump and die) et logs Laravel

  > 💡 **Debugging** : Recherche et correction des erreurs dans le code
  > 💡 **dd()** : Fonction qui affiche le contenu d'une variable et arrête le programme

- ✅ Utilisation de **Docker logs** pour diagnostiquer les services

  > 💡 **Log** : Fichier qui enregistre tous les événements de l'application

- ✅ Commandes **Artisan** pour déboguer :

  > 💡 **Artisan** : Outil en ligne de commande de Laravel

  - `` : Console interactive pour tester du code

B2.2.2 - Adapter une solution applicative

Activités réalisées :

- ✅ Ajout progressif de fonctionnalités :
  - S1 : Affichage simple (statique)
  - S2 : Connexion base de données (dynamique)
  - S3 : CRUD complet + gamification
- ✅ Modification des vues pour améliorer l'UX

> 💡 **UX** : User Experience (expérience utilisateur) - facilité d'utilisation

- ✅ Évolution du modèle de données (ajout de champs)
- ✅ **Refactoring** du code pour plus de clarté

> 💡 **Refactoring** : Améliorer le code sans changer ce qu'il fait

B2.2.3 - Réaliser les tests nécessaires

Activités réalisées :

- ✅ Tests manuels fonctionnels après chaque développement
- ✅ **Validation** des formulaires (données correctes/incorrectes)
  > 💡 **Validation** : Vérifier que les données saisies sont correctes
- ✅ Tests de navigation entre les pages
- ✅ Vérification du bon fonctionnement des routes
- ✅ Tests de base avec **GitHub Actions** (CI/CD)
  > 💡 **CI/CD** : Intégration Continue / Déploiement Continu - automatisation des tests

Preuves concrètes :

- Checklist de tests : `docs/SEANCE-3/EVALUATION.md`

B2.3 - Gérer les données

B2.3.1 - Mettre en place la structure de données

Activités réalisées :

- ✅ Création de 3 migrations Laravel (voir détails en B2.1.3)

Preuves concrètes :

- Migration utilisateurs : `database/migrations/2024_01_01_000000_creer_table_utilisateurs.php` (lignes 15-25)
- Migration livres : `database/migrations/2024_01_01_000001_creer_table_livres.php` (lignes 15-30)
- Migration emprunts : `database/migrations/2024_01_01_000002_creer_table_emprunts.php` (lignes 15-35)

B2.3.2 - Développer des éléments d'accès aux données

Activités réalisées :

- ✅ Création de 3 modèles Eloquent avec relations (voir détails en B2.1.6)

Preuves concrètes :

- Modèle Utilisateur : `app/Modeles/Utilisateur.php` (lignes 25-28 pour relation)
- Modèle Livre : `app/Modeles/Livre.php` (lignes 15-18 pour relation)
- Modèle Emprunt : `app/Modeles/Emprunt.php` (lignes 12-20 pour relations)

Niveau de maîtrise : ⭐⭐⭐⭐ Avancé

Séance(s) : S2, S3

B2.3.3 - Manipuler les données

Activités réalisées :

- ✅ CRUD complet (voir détails en B2.1.7)

Preuves concrètes :

- LivreControleur CRUD complet : `app/Http/Controleurs/LivreControleur.php` (lignes 15-100)
- 5 formulaires validés dans `resources/views/livres/`
- Gestion des erreurs avec **flash messages** : `app/Http/Controleurs/LivreControleur.php` (lignes 40, 55, 65)

  > 💡 **Flash Message** : Message temporaire affiché une seule fois (ex: "Livre créé avec succès")

Niveau de maîtrise : ★★★★★ Expert

🔒 BLOC B3 - CYBERSÉCURITÉ

B3.1 - Protéger les données à caractère personnel

B3.1.2 - Sécuriser les données sensibles

Activités réalisées :

- ✅ Validation stricte de toutes les entrées utilisateur
- ✅ Protection **XSS** (Cross-Site Scripting) :

  > 💡 **XSS** : Attaque qui injecte du code malveillant dans une page web

  - Utilisation de `{{ $variable }}` dans Blade (échappement automatique)
    - Fichier : `resources/views/livres/index.blade.php` (ligne 25 : `{{ $livre->titre }}`)
  - Seul `{!! !!}` permet HTML (jamais utilisé sans contrôle)

B3.3 - Sécuriser les équipements et les usages

B3.3.1 - Protéger contre les attaques

Activités réalisées :

- ✅ Protection **SQL Injection** :

  > 💡 **SQL Injection** : Attaque qui injecte du code SQL malveillant dans une requête

  - Utilisation exclusive d'Eloquent ORM
  - Aucune requête SQL directe
  - Paramètres liés automatiquement
  - Fichier : `app/Http/Controleurs/LivreControleur.php` (ligne 22 : `Livre::all()` au lieu de requête SQL)

- ✅ Validation stricte des formulaires :

🛠️ COMPÉTENCES DEVOPS & INFRASTRUCTURE

Containerisation (Conteneurisation)

> 💡 **Containerisation** : Mettre l'application dans une "boîte" isolée avec tout ce dont elle a besoin

Activités réalisées :

- ✅ Configuration **Docker** multi-services :
  > 💡 **Docker** : Outil pour créer des conteneurs (applications isolées)
  - Service app (Laravel + PHP 8.2)
  - Service database (SQLite)
  - Service redis (cache)
  - Service mailhog (email de test)

CI/CD (Intégration Continue / Déploiement Continu)

> 💡 **CI/CD** : Automatisation des tests et du déploiement à chaque modification du code

Activités réalisées :

- ✅ Configuration **GitHub Actions** basique
  > 💡 **GitHub Actions** : Outil d'automatisation intégré à GitHub
  - Fichier : `.github/workflows/laravel-ci-cd.yml` (lignes 1-50)
- ✅ **Pipeline** automatisé de tests
  > 💡 **Pipeline** : Chaîne d'actions automatiques (tester → valider → déployer)
- ✅ Déploiement automatique en staging
  > 💡 **Staging** : Environnement de test avant la production

Cloud Development (Développement Cloud)

> 💡 **Cloud** : Travailler sur des serveurs distants au lieu de son ordinateur

Activités réalisées :

- ✅ Configuration **GitHub Codespaces**
  > 💡 **Codespaces** : Environnement de développement dans le navigateur (VS Code en ligne)
  - Fichier : `.devcontainer/devcontainer.json` (lignes 1-40)
- ✅ Setup automatisé à l'ouverture
  - Script : `.devcontainer/installation.sh` (lignes 1-60)
- ✅ Environnement cloud-native

Preuves concrètes :

📊 RÉCAPITULATIF PAR NIVEAU

★★★★★ Expert (Maîtrise complète)

- Structure de base de données (migrations)
- CRUD complet avec validation
- Manipulation de données Eloquent

★★★★ Avancé (Très bon niveau)

- Interface utilisateur responsive
- Conception base de données relationnelle
- Programmation Laravel (MVC)
- Relations entre tables
- Containerisation Docker

📈 PROGRESSION PAR SÉANCE

Compétence	S1	S2	S3	Niveau Final
Architecture MVC	★★	★★★	★★★★	Avancé
Base de données	-	★★★	★★★★	Avancé
Eloquent ORM	-	★★★	★★★★★	Expert
CRUD	-	★★	★★★★★	Expert
Sécurité web	-	-	★★★	Maîtrisé
Docker	★★★★	★★★★	★★★★	Avancé
Interface UI	★★★	★★★★	★★★★	Avancé

✅ VALIDATION DES COMPÉTENCES

Compétences B2.1 : 7/7 validées ✅

Compétences B2.2 : 3/3 validées ✅

Compétences B2.3 : 3/3 validées ✅

Compétences B3.1 : 1/1 validée ✅

Compétences B3.3 : 1/1 validée ✅

TOTAL : 15 compétences du référentiel BTS SIO SLAM validées

📝 NOTES POUR L'ÉVALUATION

Points forts du projet :

- Application complète et fonctionnelle
- Code respectant les conventions Laravel avec noms en français
- Sécurité prise en compte dès le départ
- Documentation technique complète
- Approche progressive et pédagogique
- Preuves concrètes avec fichiers et numéros de lignes

Axes d'amélioration possibles :

- Tests automatisés plus complets
- API REST (prévu séance 8)
- Authentification avancée (prévu séance 4)