💼 Compétences pour CV Professionnel

Projet BiblioTech - BTS SIO SLAM

📋 VERSION COMPLÈTE (pour Portfolio / GitHub)

BiblioTech - Application Web de Gestion de Bibliothèque

Contexte : Projet développé dans le cadre du BTS SIO option SLAM

Durée : 9 heures (3 séances de 3 heures)

Période : Septembre - Octobre 2025

Statut : Projet terminé et fonctionnel

🎯 Objectif du Projet

Développer une application web moderne de gestion de bibliothèque permettant de gérer les livres, les utilisateurs et les emprunts, avec une interface responsive et sécurisée.

💻 Stack Technique

Backend

- **Laravel 11** - Framework PHP MVC
  > 💡 **Framework** : Ensemble d'outils et de règles pour développer plus rapidement
- **PHP 8.2** - Langage de programmation
- **Eloquent ORM** - Gestion de la base de données
  > 💡 **ORM** : Outil qui traduit les objets PHP en requêtes de base de données
- **SQLite** - Base de données relationnelle
  > 💡 **Base de données** : Endroit où on stocke toutes les informations (livres, utilisateurs)

Frontend

✨ Fonctionnalités Développées

Gestion des Livres (CRUD complet)

> 💡 **CRUD** : Create (Créer), Read (Lire), Update (Modifier), Delete (Supprimer)

- ✅ Création de nouveaux livres avec validation
  - Fichier : `app/Http/Controleurs/LivreControleur.php` (lignes 30-40)
- ✅ Affichage de la liste complète avec pagination
  - Fichier : `resources/views/livres/index.blade.php` (lignes 10-30)
- ✅ Affichage des détails d'un livre
  - Fichier : `resources/views/livres/afficher.blade.php` (lignes 1-50)
- ✅ Modification des informations
  - Fichier : `resources/views/livres/modifier.blade.php` (lignes 1-45)
- ✅ Suppression avec confirmation
  - Fichier : `app/Http/Controleurs/LivreControleur.php` (lignes 60-65)

📊 Réalisations Chiffrées

Métrique	Valeur
Lignes de code	~2000 lignes (PHP/Blade)
Tables BDD	3 tables avec relations (noms français)
Contrôleurs	5 contrôleurs (noms français)
Vues Blade	10+ templates (dossiers français)
Routes	15+ routes configurées (`routes/web.php`)
Formulaires	5 formulaires validés
Services Docker	4 services orchestrés
Temps de setup	< 5 minutes avec Docker

🎓 Compétences Techniques Acquises

Développement Backend

- **Architecture MVC** (Modèle-Vue-Contrôleur)
  > 💡 **MVC** : Façon d'organiser le code en 3 parties séparées
  - Exemple : `app/Http/Controleurs/LivreControleur.php` (Contrôleur)
- **Programmation orientée objet** (POO)
- **Eloquent ORM** et requêtes optimisées
  - Exemple : `app/Modeles/Livre.php` (Modèle)
- **Gestion des relations** entre entités
  - Exemple : `app/Modeles/Utilisateur.php` (lignes 25-28)

Développement Frontend

- **Templates Blade** avec héritage

🔗 Liens

- **Code source :** github.com/username/bibliotech
- **Documentation :** Voir `README.md` (lignes 1-150)
- **Démo en ligne :** bibliotech-demo.netlify.app *(si applicable)*

📝 VERSION COURTE (pour CV)

Option 1 : Format Descriptif

BiblioTech - Application de Gestion de Bibliothèque
BTS SIO SLAM - Septembre-Octobre 2025 (9 heures)

Application web complète développée avec Laravel 11, permettant la gestion des livres, utilisateurs et emprunts. Architecture MVC avec base de données SQLite (3 tables avec noms de champs en français, relations), interface responsive Bootstrap 5, et infrastructure Docker. CRUD complet avec validation, sécurité (CSRF/XSS/SQL Injection), et système de gamification. CI/CD automatisé avec GitHub Actions.

Technologies : Laravel 11, PHP 8.2, SQLite, Bootstrap 5, Docker, GitHub Actions
Compétences : Architecture MVC, Eloquent ORM, sécurité web, Docker, CI/CD
Code : Contrôleurs et modèles avec noms en français (`LivreControleur.php`, `Livre.php`)

Option 2 : Format Bullet Points

BiblioTech - Application Laravel (9h, Sept-Oct 2025)

- Application web MVC complète avec Laravel 11 et PHP 8.2
- Base de données relationnelle SQLite (3 tables : utilisateurs, livres, emprunts - champs français)
- CRUD complet avec validation stricte et sécurité (CSRF, XSS, SQL Injection)
- Interface responsive Bootstrap 5 (mobile/tablette/desktop)
- Infrastructure Docker multi-services + CI/CD GitHub Actions
- Système de gamification (points, badges, classement)
- **Code :** `app/Http/Controleurs/LivreControleur.php` (100 lignes), `resources/views/livres/` (10+ vues)

Option 3 : Format Tableau

Projet	Technologies	Réalisations	Fichiers Clés (français)

BiblioTech
Gestion bibliothèque
Sept-Oct 2025 (9h)
	Laravel 11, PHP 8.2
	SQLite, Eloquent
	Bootstrap 5, Blade
	Docker, CI/CD
		• CRUD complet sécurisé
		• 3 tables relationnelles
		• Interface responsive
		• Gamification
		• Tests automatisés
			`LivreControleur.php`
			`Livre.php` (Modèle)
			`livres/*.blade.php`
			`docker-compose.yml`

🎤 VERSION POUR PRÉSENTATION ORALE

Pitch Elevator (30 secondes)

> "J'ai développé **BiblioTech** en septembre-octobre 2025, une application web de gestion de bibliothèque avec Laravel. Elle permet de gérer un catalogue de livres avec un système CRUD complet, une base de données relationnelle SQLite avec 3 tables (utilisateurs, livres, emprunts) et des champs en français, et une interface responsive. J'ai nommé tous mes contrôleurs et modèles en français : `LivreControleur`, `Livre`, `Utilisateur`, `Emprunt`. J'ai mis en place Docker pour que toute l'équipe ait le même environnement, et intégré un système de gamification. L'application est sécurisée contre XSS, CSRF et SQL Injection."

Présentation Détaillée (2-3 minutes)

Introduction :

> "BiblioTech est un projet que j'ai réalisé en septembre-octobre 2025 dans le cadre de ma formation BTS SIO SLAM. L'objectif était de créer une application moderne de gestion de bibliothèque en 9 heures réparties sur 3 séances."

Architecture et conventions :

> "J'ai appliqué l'architecture MVC avec une particularité : j'ai utilisé des noms en français pour tous les éléments du code. Par exemple, au lieu de `BookController`, j'ai `LivreControleur`, au lieu de `User`, j'ai `Utilisateur`. Les tables de la base de données ont aussi des noms français : `utilisateurs`, `livres`, `emprunts`, avec des champs comme `titre`, `auteur`, `date_emprunt`, `date_retour_prevue`."

Organisation du code :

> "Le contrôleur principal se trouve dans

💡 QUESTIONS FRÉQUENTES EN ENTRETIEN

Q1 : Montrez-moi un exemple de code que vous avez écrit

> "Bien sûr ! Dans `app/Http/Controleurs/LivreControleur.php` aux lignes 30-40, j'ai la méthode `stocker()` qui crée un nouveau livre :
>
> ```
> public function stocker(Request $requete) {
>     $donnees_validees = $requete->validate([
>         'titre' => 'required|max:255',
>         'auteur' => 'required|max:255',
>         'isbn' => 'required|unique:livres',
>     ]);
>
>     Livre::create($donnees_validees);
>     return redirect()->route('livres.index')
>         ->with('succes', 'Livre créé avec succès !');
> }
> ```
>
> J'ai utilisé des noms en français : `stocker` au lieu de `store`, `requete` au lieu de

Q2 : Pourquoi avoir utilisé des noms en français ?

> "C'est une décision pédagogique importante. Les noms en français rendent le code plus compréhensible pour les débutants et les francophones. Quand on lit `utilisateur_id` ou `date_emprunt`, c'est immédiatement clair, alors que `user_id` ou `loan_date` nécessitent de connaître l'anglais. Dans `app/Modeles/Utilisateur.php` ligne 27, la méthode `emprunts()` est plus parlante que `loans()`. Ça facilite la maintenance et la collaboration avec des équipes francophones."

Q3 : Comment avez-vous structuré votre base de données ?

> "J'ai créé 3 tables avec des noms et champs en français dans `database/migrations/` :
>
> 1. **Table utilisateurs** (`creer_table_utilisateurs.php` , lignes 15-25) : id, nom, prenom, email, mot_de_passe, date_creation, date_modification
> 2. **Table livres** (`creer_table_livres.php` , lignes 15-30) : id, titre, auteur, isbn, disponible, categorie, date_creation, date_modification
> 3. **Table emprunts** (`creer_table_emprunts.php` , lignes 15-35) : id, utilisateur_id, livre_id, date_emprunt, date_retour_prevue, date_retour_effective
>
> Les relations sont définies dans `app/Modeles/Utilisateur.php` lignes 25-28 avec la méthode `emprunts()` qui établit qu'un utilisateur peut avoir plusieurs emprunts."

Q4 : Comment gérez-vous la cohérence des noms français dans Laravel ?

> "Laravel accepte parfaitement les noms en français. Dans les migrations, je définis les colonnes avec `$table->string('titre')` au lieu de `$table->string('title')`. Dans les modèles, je spécifie le nom de la table avec `protected $table = 'livres';` et les champs modifiables avec `protected $fillable = ['titre', 'auteur', 'isbn', 'disponible', 'categorie'];`. Les relations fonctionnent aussi : `return $this->hasMany(Emprunt::class, 'utilisateur_id');`. C'est très cohérent et maintenable."

Q5 : Quelle est la partie dont vous êtes le plus fier ?

> "Je suis particulièrement fier d'avoir créé une application complète 100% en français, ce qui est rare. Le service de gamification dans `app/Services/ServiceGamification.php` (lignes 15-50) attribue automatiquement des points. Le classement dans `resources/views/classement.blade.php` affiche les meilleurs utilisateurs. Mais surtout, la cohérence des noms français dans toute l'application (contrôleurs, modèles, vues, migrations, variables) rend le code très lisible et accessible, ce qui facilite la collaboration et la maintenance à long terme."

📧 EXEMPLE DE MAIL DE CANDIDATURE

Objet : Candidature Stage Développeur Web - [Votre Nom] - BTS SIO SLAM

Bonjour,

Je suis actuellement en BTS SIO option SLAM à [Nom de l'école] et je recherche un stage de [durée] semaines à partir de [date].

En septembre-octobre 2025, j'ai développé **BiblioTech**, une application web de gestion de bibliothèque avec **Laravel 11** et **PHP 8.2**. Ce projet de 9 heures m'a permis de :

- Maîtriser l'**architecture MVC** avec code structuré en français (`LivreControleur.php`, `Livre.php`, vues Blade)
- Concevoir une **base de données relationnelle** SQLite (3 tables : utilisateurs, livres, emprunts avec champs français)
- Implémenter un **CRUD complet** avec validation stricte (lignes 32-37 de

🎯 MOTS-CLÉS POUR LINKEDIN / CV

Compétences Techniques

`Laravel 11` `PHP 8.2` `MVC` `Eloquent ORM` `SQLite` `Bootstrap 5` `Blade` `JavaScript ES6+` `Docker` `Docker Compose` `Git` `GitHub Actions` `CI/CD` `Responsive Design` `Migration` `Seeder` `Nommage Français`

Compétences Sécurité

`CSRF Protection` `XSS Prevention` `SQL Injection Protection` `Data Validation` `Secure Coding`

Compétences Fonctionnelles

`Développement Web` `Full-Stack` `Base de Données Relationnelle` `CRUD` `Architecture Logicielle` `UX/UI` `Gamification` `Code Français`

📌 EXEMPLES DE CODE À PRÉSENTER

Exemple 1 : Une Route Simple

Fichier : `routes/web.php` (ligne 15)

```php
Route::get('/livres', [LivreControleur::class, 'index'])->name('livres.index');
```

Explication : "Cette route associe l'URL `/livres` à la méthode `index()` du contrôleur `LivreControleur`. Le nom `livres.index` permet de réutiliser cette route facilement. J'ai gardé la cohérence en français partout."

Exemple 2 : Une Relation Eloquent

Fichier : `app/Modeles/Utilisateur.php` (lignes 25-28)

```php
public function emprunts()
{
    return $this->hasMany(Emprunt::class, 'utilisateur_id');
}
```

Explication : "Cette relation One-to-Many indique qu'un utilisateur peut avoir plusieurs emprunts. Le nom de la méthode `emprunts()` est en français, tout comme la clé étrangère `utilisateur_id` dans la table emprunts. Je peux récupérer tous les emprunts avec `$utilisateur->emprunts`."

Exemple 3 : Une Validation avec Variables Françaises

Fichier : `app/Http/Controleurs/LivreControleur.php` (lignes 32-40)

```
public function stocker(Request $requete)
{
    $donnees_validees = $requete->validate([
        'titre' => 'required|max:255',
        'auteur' => 'required|max:255',
        'isbn' => 'required|unique:livres',
    ], [
        'titre.required' => 'Le titre est obligatoire',
        'isbn.unique' => 'Cet ISBN existe déjà',
    ]);

    Livre::create($donnees_validees);

    return redirect()->route('livres.index')
        ->with('succes', 'Livre créé avec succès !');
}
```

Explication : "Méthode `stocker()` au lieu de `store()`, variable `$requete` au lieu de

Exemple 4 : Migration avec Champs Français

Fichier : `database/migrations/2024_01_01_000001_creer_table_livres.php` (lignes 15-30)

```
public function up()
{
    Schema::create('livres', function (Blueprint $table) {
        $table->id();
        $table->string('titre');
        $table->string('auteur');
        $table->string('isbn')->unique();
        $table->boolean('disponible')->default(true);
        $table->string('categorie')->nullable();
        $table->timestamps(); // date_creation, date_modification
    });
}
```

Explication : "Migration qui crée la table `livres` avec tous les champs en français : `titre`, `auteur`, `isbn`, `disponible`, `categorie`. L'ISBN est unique avec un index. Le

✅ CHECKLIST AVANT CANDIDATURE

Avant de mentionner BiblioTech dans une candidature, vérifiez :

- [ ] Le code est sur GitHub avec README complet (`README.md` à jour)
- [ ] Les commits sont propres et réguliers (historique Git clair)
- [ ] Le code est commenté (surtout les parties complexes)
- [ ] Tous les noms sont cohérents en français (contrôleurs, modèles, vues, champs BDD)
- [ ] L'application fonctionne (testée avec `docker-compose up`)
- [ ] Vous connaissez les fichiers principaux et leurs lignes importantes :
  - [ ] `app/Http/Controleurs/LivreControleur.php` (lignes 15-100)
  - [ ] `app/Modeles/Livre.php` (lignes 15-28)
  - [ ] `database/migrations/creer_table_*.php` (structure des tables)

📌 CONSEILS FINAUX

✅ À FAIRE

- Montrer le code source avec numéros de lignes précis
- Expliquer le choix des noms français (lisibilité, accessibilité, maintenance)
- Mentionner la cohérence : "Dans toute l'application, de la BDD aux vues"
- Donner des exemples concrets : "ligne 27 de Utilisateur.php, la méthode `emprunts()` ..."
- Valoriser cet aspect différenciant : peu d'applications Laravel sont en français

❌ À ÉVITER

- Dire "c'est juste un petit projet d'école"
- Oublier de mentionner la cohérence des noms français
- Ne pas savoir justifier ce choix de nommage

💡 **Astuce finale :** Le fait d'avoir une application Laravel entièrement en français est un **point différenciant fort**. En entretien, insistez sur la cohérence (contrôleurs, modèles, BDD, vues) et la lisibilité du code. Préparez ce document avec vos 5 fichiers clés :

```
MES 5 FICHIERS CLÉS (Tout en français)

1. app/Http/Controleurs/LivreControleur.php
   - Ligne 22 : Livre::all() - Récupération des livres
   - Lignes 32-37 : Validation avec champs français
   - Ligne 40 : Flash message 'succes'
   - Méthodes : index(), creer(), stocker(), modifier(), mettreAJour(), supprimer()

2. app/Modeles/Livre.php
   - Lignes 15-18 : Relation emprunts()
   - Ligne 10 : protected $table = 'livres'
   - Ligne 12 : $fillable avec champs français

3. database/migrations/creer_table_livres.php
   - Lignes 15-30 : Champs titre, auteur, isbn, disponible, categorie
   - Tous les noms en français

4. resources/views/livres/index.blade.php
   - Ligne 1 : @extends('layouts.application')
   - Ligne 25 : {{ $livre->titre }} (XSS protection)
   - Lignes 10-30 : Boucle d'affichage

5. resources/views/livres/creer.blade.php
```

📌 Avec cette approche cohérente tout en français, vous vous démarquez ! Bonne chance ! 🚀