ECOLE MAROCAINE DES SCIENCES DE L'INGENIEUR

Membre de HONORIS UNITED UNIVERSITIES

Rapport : Gestion de bibliothèque

Réaliser par :

Ilyass Lhajoui

Ziyad Mouzoun

Kari Abdul Bagui Yaya

2025/2026

Table des matières

1 – Introduction

Dans le cadre du cours de **Programmation Web – JavaScript** pour l'année universitaire 2025-2026, nous avons choisi de réaliser un projet sur la **gestion d'une bibliothèque**. L'objectif est de développer une **application web interactive** capable de gérer efficacement les livres, les utilisateurs et les emprunts, tout en offrant une interface intuitive et facile à utiliser.

Cette application, entièrement développée en **JavaScript natif (ES6+)**, inclut la gestion de plusieurs entités avec des opérations **CRUD** (Créer, Lire, Modifier, Supprimer), un **tableau de bord** avec statistiques et graphiques dynamiques, un **système d'authentification sécurisé**, ainsi qu'une interface **multilingue** (Français et Anglais). Le projet permet ainsi de mettre en pratique la **manipulation du DOM**, le stockage local via **localStorage.**

De plus, ce projet offre la possibilité d'explorer des concepts avancés tels que la **validation des formulaires en temps réel** et la visualisation des données à l'aide de graphiques interactifs. Il vise également à fournir une solution simple, fonctionnelle.

Ainsi, ce projet combine à la fois le **développement technique**, la **gestion des données** et la **conception d'une interface moderne**, permettant de renforcer nos compétences en programmation web tout en créant une application pratique et utilisable dans un contexte réel.

2 - Analyse

Dans cette phase, nous avons étudié les besoins fonctionnels et défini la structure de l'application pour assurer une gestion efficace de la bibliothèque.

1. Entités

Le projet est basé sur **5 entités principales**, chacune avec ses propres attributs et opérations CRUD :

1. **Client** : nom, prénom, email, mot de passe, rôle (Admin/User), 2. **Livre** : titre, auteur, catégorie, date de publication, disponibilité,prix.

- **Auteur** : nom, prénom, liste des livres publiés.
- **Catégorie** : nom, liste des livres associés.
- **Commande** : client, livres empruntés, date de commande.

Chaque entité est liée aux autres par des relations logiques : par exemple, un **Livre** appartient à une **Catégorie** et est écrit par un **Auteur**, tandis qu'un **Client** peut passer plusieurs **Commandes**.

2. Schéma de navigation

L'application comporte une **navigation simple et intuitive** :

- **Navbar** : logo, nom utilisateur, bouton de déconnexion, sélecteur de langue.
- **Menu latéral** : liens vers le tableau de bord et chaque entité CRUD.
- Tableau de bord avec accès rapide aux statistiques et graphiques.

3. Maquettes

Des **maquettes graphiques** ont été réalisées pour visualiser l'interface :

Page de connexion avec validation et messages d'erreur.

- Pages CRUD pour chaque entité avec formulaires, listes et modals de confirmation.
- Tableau de bord présentant les cartes d'indicateurs et au moins cinq types de graphiques pour suivre les statistiques.
- Interface multilingue (Français, Anglais).

----->Cette analyse a permis de clarifier la structure de l'application.

3 - Réalisation

3.1 Architecture du projet

L'application suit une architecture modulaire avec séparation claire entre logique métier, interface et ressources.

Structure des dossiers :

```
├── html/          # Pages (index, dashboard, CRUD pour chaque entité) ├── html/          # Pages (index, dashboard, CRUD pour chaque entité)
├── js/            # Modules JavaScript (auth, storage, CRUD, i18n, validation) ├── js/            # Modules JavaScript (auth, storage, CRUD, i18n, validation)
└── Outils/images/ # Ressources graphiques └── Outils/images/ # Ressources graphiques
```

Technologies utilisées : JavaScript ES6+, HTML5, CSS3, localStorage pour la persistance des données.

3.2 Gestion des données

Toutes les entités (Clients, Livres, Auteurs, Catégories, Commandes) sont stockées dans le localStorage via un module centralisé gérant les opérations CRUD :

3.3 Authentification

Système de connexion avec validation des identifiants, gestion des sessions via sessionStorage , et contrôle d'accès basé sur les rôles (Admin/User). Les administrateurs ont accès complet aux opérations CRUD, les utilisateurs standards peuvent consulter et emprunter.

3.4 Interface utilisateur

Page de connexion : Formulaire avec validation en temps réel, messages d'erreur dynamiques et changement de langue.

Pages CRUD : Chaque entité dispose d'un tableau paginé avec recherche/filtrage, formulaires modaux pour ajout/modification, et validation des données (email, dates, prix positifs).

Navigation : Navbar avec logo, utilisateur connecté et bouton de déconnexion. Menu latéral avec liens vers dashboard et toutes les entités, page active mise en évidence.

4 - Fonctionnalités avancées

4.1 Tableau de bord et statistiques

Le dashboard présente une vue d'ensemble via **6 cartes KPI** :

Total des livres / Livres disponibles / Livres empruntés

Total clients / Total commandes / Revenus générés

5 graphiques interactifs :

- **Barres** - Distribution des livres par catégorie
- **Circulaire** - Disponibilité des livres (disponibles vs empruntés)
- **Linéaire** - Évolution des commandes dans le temps
- **Aires** - Revenus mensuels
- **Donut** - Répartition clients par rôle (Admin/User)

Les statistiques sont calculées dynamiquement à partir du localStorage.

4.2 Internationalisation (i18n)

Support de deux langues (Français/Anglais) avec changement dynamique sans rechargement.
Les traductions

4.3 Design responsive

Interface adaptative à tous les écrans grâce à CSS Flexbox/Grid :

4.4 Sécurité

Protection des routes : vérification de l'authentification sur chaque page

Contrôle d'accès basé sur les rôles

Validation côté client pour tous les formulaires

Nettoyage des entrées utilisateur (sanitization)

Gestion des erreurs : Système de notifications toast pour informer l'utilisateur (succès/erreurs), confirmations avant suppressions, alertes pour champs invalides.

4.5 Fonctionnalités complémentaires

Recherche avancée : Recherche textuelle en temps réel et filtrage multi-critères

Export CSV : Possibilité d'exporter les données pour analyse externe

Notifications : Toast notifications pour toutes les actions utilisateur

Cette réalisation démontre une application web complète en JavaScript natif, intégrant gestion des données, interface moderne et expérience utilisateur optimale.

5.Optimisation et performances

5.1 Organisation et structuration du code JavaScript

Le code JavaScript a été organisé de manière claire et structurée afin d'améliorer sa lisibilité et de faciliter sa maintenance. Les scripts ont été séparés selon leurs fonctionnalités, ce qui permet une meilleure compréhension du code et une réduction des répétitions.

Exemple :

```
// ============================
// 8 - Gestion du formulaire de connexion
// ============================
loginForm.addEventListener("submit", function(e) {
    e.preventDefault();
    // Récupérer les valeurs saisies
    const email = emailEl.value.trim();
    const password = passwordEl.value.trim();
    // Vérifier si l'utilisateur existe
    const user = users.find(u => u.email === email && u.password === password);
    if (user) {
        // Sauvegarder l'utilisateur courant et rediriger
        localStorage.setItem("currentUser", JSON.stringify(user));
        window.location.href = "../html/dashboard.html";
    } else {
        // Afficher le message d'erreur dans la langue sélectionnée
        errorMsg.textContent = textes[langSelect.value].error;
    }
});
```

5.2 Optimisation des interactions utilisateur

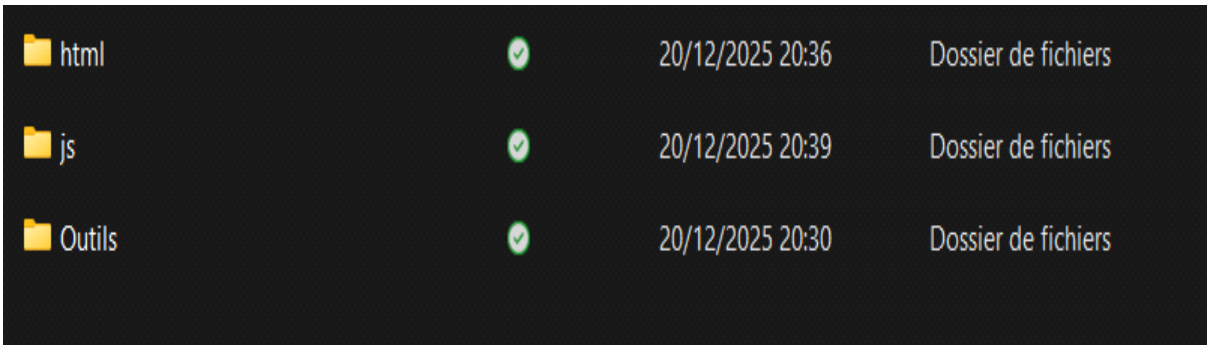
Les interactions utilisateur ont été optimisées grâce à JavaScript. Chaque événement de l'utilisateur, comme le clic sur le bouton ou la soumission du formulaire, est géré de manière efficace pour améliorer la réactivité de l'interface. La mise à jour des champs du formulaire et l'affichage des messages d'erreur se font sans rechargement de page, offrant ainsi une expérience fluide et dynamique à l'utilisateur.

```
    // Vérifier la correspondance des mots de passe
    if (password === confirmPassword) {
        const newUser = { email, password, role };

        // Ajouter le nouvel utilisateur et mettre à jour le localStorage
        users.push(newUser);
        localStorage.setItem("users", JSON.stringify(users));
        localStorage.setItem("currentUser", JSON.stringify(newUser));

        // Redirection vers le dashboard
        window.location.href = "../html/dashboard.html";
    } else {
        // Afficher le message d'erreur selon la langue
        errorMsg.textContent = textes[langSelect.value].error;
    }
});
```

5.3 Organisation des fichiers et structure du projet

Le projet front-end a été structuré de manière claire afin de faciliter la maintenance et l'évolution du code. Les fichiers JavaScript ont été séparés selon leurs fonctionnalités : un fichier pour la gestion du login, un autre pour l'inscription, et un autre pour la gestion de l'internationalisation. Cette organisation permet de

retrouver rapidement chaque fonctionnalité, d'éviter les répétitions et de rendre le code plus lisible et réutilisable. De plus, la structure des dossiers respecte les bonnes pratiques front-end, avec un dossier pour les fichiers HTML, un pour le CSS et un pour le JavaScript, ce qui améliore la clarté du projet.



5.4 Optimisations et améliorations futures

Bien que l'application fonctionne correctement et soit réactive, plusieurs améliorations pourraient être apportées dans le futur pour améliorer les performances et l'expérience utilisateur dans le cadre de la bibliothèque :

- **Gestion des quantités de livres** : Actuellement, le projet ne gère pas les quantités disponibles pour chaque livre. Il serait possible d'ajouter cette fonctionnalité afin de suivre le stock et d'empêcher l'emprunt de livres non disponibles.
- **Chargement asynchrone des données** : Les livres et les informations pourraient être chargés de manière dynamique avec JavaScript (AJAX ou fetch) pour améliorer la vitesse de la page et ne charger que ce qui est nécessaire.
- **Optimisation des interactions DOM** : Réduire encore plus les modifications répétitives des éléments HTML pour améliorer la fluidité, surtout si le catalogue devient très large.
- **Optimisation pour mobile et responsive** : Adapter encore mieux l'interface pour les tablettes et smartphones afin d'assurer une navigation rapide et agréable sur tous les appareils.
- **Amélioration de l'internationalisation** : Ajouter plus de langues et permettre de changer la langue sans recharger complètement la page, rendant l'application plus accessible aux utilisateurs internationaux.

7. Conclusion et perspectives

Ce projet de développement d'une application web pour la gestion d'une bibliothèque a permis de mettre en pratique les concepts étudiés en front-end et JavaScript.
Nous avons créé un système fonctionnel avec un formulaire de connexion, une gestion dynamique de la langue, un tableau de bord simple et des messages d'erreur interactifs.

Difficultés rencontrées :

- La gestion des interactions dynamiques avec JavaScript et la validation des formulaires a demandé de structurer le code soigneusement.
- La mise en place de l'internationalisation et la mémorisation de la langue de l'utilisateur dans le localStorage a nécessité une réflexion sur l'organisation du code et des données.

Solutions apportées :

- Utilisation du localStorage pour sauvegarder l'utilisateur et la langue sélectionnée.
- Séparation des scripts JavaScript selon les fonctionnalités pour faciliter la lisibilité et la maintenance.
- Mise en place de validations côté client et de messages dynamiques pour améliorer l'expérience utilisateur.

Perspectives d'amélioration :

- Ajouter la gestion des quantités de livres et le suivi du stock pour rendre l'application plus complète.
- Optimiser davantage l'interface pour les appareils mobiles et améliorer la réactivité avec des chargements asynchrones.
- Étendre l'internationalisation avec plus de langues et une traduction dynamique de tous les contenus de l'application.