EMSI

rapport: application javascript de gestion de biblio

module : Développement Web javascript

Année universitaire : 2025 - 2026

Filière : 3IIR

Rédiger par: SARA ZAMZAMI- NOUR DAHBI- SALMA ELKHAYAT

Encadrer par : Mme TLEMCANI KHADIJA

Micro-rapport – Application JavaScript de gestion de bibliothèque

Introduction générale du projet

Dans le cadre de l'apprentissage du développement web et de la programmation orientée client, ce projet intitulé Bibliotheca consiste à concevoir et implémenter une application web interactive de gestion de bibliothèque. L'application permet à l'utilisateur de gérer des livres et des auteurs, de consulter des statistiques et d'avoir une vue globale de l'état de la bibliothèque à travers un tableau de bord moderne.

Le projet repose essentiellement sur JavaScript, qui joue un rôle central dans la gestion de la logique métier, des interactions utilisateur et de la manipulation dynamique de l'interface graphique. L'objectif est de démontrer qu'il est possible de construire une application complète et fonctionnelle sans utiliser de framework backend, uniquement avec des technologies web standards.

Logique générale et structure du rapport

Ce micro-rapport suit une structure logique et progressive afin de faciliter la compréhension du projet :

1. Présentation du contexte et de la problématique
2. Description globale de la solution proposée
3. Présentation des logiciels et langages utilisés
4. Analyse détaillée de l'implémentation JavaScript
5. Description de la phase d'exécution et des interfaces
6. Description des figures
7. Conclusion générale

Cette organisation permet d'avoir une vision claire du projet, aussi bien sur le plan conceptuel que technique.

Problématique

Avec l'évolution rapide des technologies web, les applications modernes doivent être interactives, ergonomiques et capables de traiter des données en temps réel. Cependant, la mise en place d'un backend et d'une base de données peut représenter une complexité supplémentaire pour des projets pédagogiques ou de petite envergure.

La problématique principale de ce projet est donc la suivante :

Comment concevoir une application web interactive permettant la gestion, l'analyse et la visualisation de données, en utilisant uniquement JavaScript côté client, sans serveur ni base de données externe ?

Cette problématique englobe plusieurs enjeux :

- Gestion des données dans le navigateur
- Organisation du code JavaScript
- Interaction fluide avec l'utilisateur
- Visualisation claire des informations

Présentation générale de la solution

La solution adoptée consiste à développer une application web de type Single Page Application (SPA). Toutes les fonctionnalités sont intégrées dans une seule page HTML, et les différentes sections (Dashboard, Livres, Auteurs) sont affichées ou masquées dynamiquement grâce à JavaScript.

L'application propose :

- Un tableau de bord avec indicateurs clés
- Une gestion complète des livres (ajout, modification, suppression)
- Une gestion des auteurs
- Des statistiques graphiques
- Une zone de danger permettant la réinitialisation des données

Cahier de charge :

PROJET FRONT-END : SMART BACKOFFICE DASHBOARD

1. Présentation générale

Dans le cadre de ce module, vous êtes amenés à développer une application Web complète de type Backoffice Dashboard, similaire aux interfaces de gestion utilisées dans les entreprises.

L'application devra permettre la manipulation de données, la visualisation de statistiques, et l'affichage d'informations dynamiques dans une interface structurée, moderne et intuitive.

Le projet doit être réalisé exclusivement en HTML5, CSS3 et JavaScript Vanilla (sans frameworks).

Vous apprendrez à organiser un code professionnel, à structurer une Single Page Application (SPA), à manipuler le DOM de manière avancée, et à interagir avec une API externe grâce à l'asynchronisme.

Chaque groupe doit choisir un des cinq sujets proposés dans ce document.

2. Objectifs pédagogiques

Ce projet vise à vous faire maîtriser :

 Manipulation avancée du DOM :

Création, suppression et mise à jour d'éléments, gestion des classes CSS, interaction avec les formulaires, validation et retours visuels.

 Gestion des événements

Click, submit, input, keyup…

Vous apprendrez à relier vos vues et vos actions grâce à une architecture claire.

 Organisation d'une application en modules

Trois modules vous seront demandés, chacun correspondant à une partie métier différente.

 Implémentation de fonctionnalités CRUD (Create, Read, Update, Delete)

Créer, lire, mettre à jour et supprimer des données en JavaScript.

 Sauvegarde et persistance (LocalStorage)

Vos données doivent rester disponibles même après rechargement de la page.

 Création d'un Dashboard professionnel

Cartes KPI (Key Performance Indicator, ou en français Indicateur Clé de Performance), graphiques, statistiques dynamiques, interprétation de données.

 Utilisation d'une API externe (asynchrone)

Grâce à fetch() et aux promesses, vous intégrerez des données externes dans votre application.

Ce projet constitue une préparation essentielle aux frameworks modernes.

3. Structure générale attendue

Votre application doit respecter une structure de type SPA (Single Page Application).

Toutes les sections doivent se trouver dans un seul fichier HTML, et être affichées/masquées via JavaScript.

L'interface doit comporter :

* Une sidebar de navigation verticale
* Une navbar supérieure
* Une section Dashboard
* Une section Module 1 (CRUD complet)
* Une section Module 2 (CRUD light)
* Une section dédiée aux statistiques et API
* Une mise en page claire, moderne, lisible, inspirée des backoffices professionnels

Vous pouvez utiliser Bootstrap, TailwindCSS, FontAwesome et Chart.js.

4. Modules obligatoires

* Module 1 : Module principal (CRUD complet)

Ce module constitue le cœur de votre application. Il doit permettre :

* l’ajout d’un élément à partir d’un formulaire validé,

* l’affichage d’une liste (tableau ou cartes),

* la recherche par mot-clé,

* le tri (par ordre alphabétique, par valeur, selon le thème),

* l'affichage d'une fiche détaillée,

* la modification d’un élément,

* la suppression avec confirmation,

* la sauvegarde dans LocalStorage.

Ce module doit montrer une maîtrise totale de la manipulation du DOM et de la logique métier.

* Module 2 : Module secondaire (CRUD simplifié)

Ce module doit proposer au minimum :

* un formulaire d’ajout,

* un affichage de liste,

* une suppression d’élément.

La modification est facultative mais recommandée pour les plus avancés.

Ce module peut être lié au premier (ex : un employé appartient à un département).

* Module 3 : Dashboard & Partie Asynchrone (API)

Ce module constitue la vue analytique de votre application. Il doit présenter une vision synoptique des données sous forme de :

* minimum deux KPI,

* au moins un graphique Chart.js,

* statistiques calculées à partir des données des modules 1 et 2.

NB : Un KPI est une valeur numérique importante, calculée à partir des données de votre application, et qui permet de suivre rapidement un état, un volume, une tendance ou une performance.

Les KPI sont très utilisés dans les interfaces professionnelles car ils permettent de donner une vision immédiate et synthétique de la situation.

Un KPI est généralement :

* une valeur simple à lire,

* affichée dans une carte visuelle (ex : carré ou rectangle coloré),

* mise en avant en haut du Dashboard,

* relative à une information importante pour le module principal.

* Partie Asynchrone – API

Chaque groupe doit intégrer au moins un appel à une API publique, via fetch().

Cet appel doit :

* récupérer des données au format JSON,

* extraire des informations pertinentes,

* mettre à jour au moins un KPI ou un graphique,

* enrichir éventuellement la base locale,

* être géré proprement avec .then() et .catch().

5. Pour vous aider à visualiser l'objectif final de votre projet,

voici une sélection de

Dashboard officiels utilisés comme références dans le développement Web.

Ces exemples présentent des pages complètes : sidebar, tableaux, formulaires, KPIs et graphiques.

Ils servent uniquement d'inspiration visuelle, pas de modèle à copier.

* AdminLTE (Open Source)

Dashboard complet avec sidebar, tables, graphiques.

https://adminlte.io/themes/v3/

* Tabler (UI moderne, minimaliste)

Pages CRUD, Dashboard, formulaires professionnels.

https://preview.tabler.io/

* CoreUI (Interface Bootstrap professionnelle)

Excellent pour comprendre la structure d'un backoffice.

https://coreui.io/demos/bootstrap/4.0/free/

6. Organisation du travail

Pour réussir le projet sans accumuler de retard, vous devez avancer en parallèle sur le développement ET sur le rapport.

Le rapport doit être rédigé progressivement, au même rythme que le projet.

Semaine Travail de développement Rédaction du rapport

Semaine

1

* Comprendre le sujet

* Définir les modules & données

* Mettre en place la structure SPA

(sidebar + sections)

* Commencer Module 1 :

formulaire + ajout + affichage

* Écrire l'Introduction

* Décrire l’analyse du sujet

* Rédiger la conception

(modules, données)

* Ajouter les premières

captures de la structure

Semaine

2

* Terminer CRUD Module 1

(modification + suppression + tri)

* Faire Module 2 (CRUD simple)•

Stabiliser l’interface

* Rédiger la section

Développement – Module 1

* Ajouter captures d’écran

* Rédiger Module 2

* Documenter difficultés &

choix techniques

Semaine

3

* Dashboard : KPI + graphique

Chart.js

* Implémenter l’API (fetch)

* Finaliser design + tests

* Rédiger la section Dashboard

& API

* Écrire Tests & validation

* Rédiger la Conclusion

* Finaliser mise en forme du

rapport (12–15 pages)

* Préparation de la présentation

finale

Codez avec curiosité, organisez-vous intelligemment et faites preuve de créativité.

Votre progression est beaucoup plus importante que la perfection.

Chapitre 1 : Logiciels, outils et langages utilisés

1.1 Langages de programmation

- HTML5 : utilisé pour structurer le contenu de l'application (sections, formulaires, boutons, modales).
- CSS3 : responsable de la mise en forme, du design moderne, de la responsivité et de l'ergonomie.
- JavaScript (ES6) : langage principal du projet, chargé de toute la logique fonctionnelle et interactive.

1.2 Bibliothèques et API

- Chart.js : bibliothèque JavaScript permettant la création de graphiques dynamiques et interactifs.
- Font Awesome : utilisée pour l'ajout d'icônes améliorant l'expérience utilisateur.
- API externe (ex. OpenLibrary) : permet l'importation automatique de livres.

1.3 Environnement d'exécution

L'application s'exécute directement dans le navigateur web sans installation particulière, ce qui la rend accessible et portable.

Chapitre 2 : Implémentation et rôle général du JavaScript

Le JavaScript constitue le cœur de l'application. Il agit comme un moteur central qui coordonne les données, l'interface utilisateur et les interactions.

2.1 Gestion des sections et navigation

Le JavaScript permet d'afficher dynamiquement les différentes sections de l'application (Dashboard, Livres, Auteurs). Lorsqu'un bouton du menu est cliqué, la section correspondante est affichée tandis que les autres sont masquées. Cette approche améliore la fluidité et évite le rechargement de la page.

2.2 Gestion des données (livres et auteurs)

Le code JavaScript gère les données sous forme de structures internes (tableaux et objets). Chaque livre est caractérisé par des attributs tels que le titre, l'auteur, l'année et le genre.

Les principales opérations assurées sont :

- Ajout de nouveaux livres
- Modification des livres existants
- Suppression de livres
- Ajout et affichage des auteurs

Ces opérations correspondent aux actions classiques de type CRUD (Create, Read, Update, Delete).

2.3 Persistance des données

Afin d'éviter la perte des données après un rechargement de la page, le JavaScript utilise le LocalStorage du navigateur. Les données sont sauvegardées localement et automatiquement restaurées lors du redémarrage de l'application.

Cette solution remplace temporairement une base de données classique dans un contexte pédagogique.

2.4 Manipulation dynamique du DOM

Le JavaScript est responsable de la création, de la mise à jour et de la suppression des éléments HTML affichés à l'écran. Chaque action de l'utilisateur déclenche une mise à jour immédiate de l'interface, garantissant une expérience utilisateur fluide et intuitive.

2.5 Tableau de bord et indicateurs (Dashboard)

Le tableau de bord affiche des indicateurs clés tels que :

- Nombre total de livres
- Nombre total d'auteurs
- Nombre de livres issus de l'API

Le JavaScript calcule ces indicateurs en temps réel à partir des données disponibles et les affiche automatiquement.

2.6 Visualisation des données avec Chart.js

Les statistiques sont représentées graphiquement grâce à Chart.js. Le JavaScript prépare les données, les regroupe par genre et les transmet à la bibliothèque afin de générer des graphiques clairs et lisibles.

Cette visualisation facilite l'analyse et la prise de décision.

2.7 Intégration d'une API externe

Le JavaScript communique avec une API externe à l'aide de requêtes HTTP. Les données reçues sont analysées, filtrées et intégrées dans l'application sans duplication.

Cette partie démontre la capacité du JavaScript à interagir avec des services externes.

2.8 Gestion des modales et notifications

Le JavaScript contrôle l'ouverture et la fermeture des fenêtres modales affichant les détails d'un livre. Il gère également les notifications (toast) pour informer l'utilisateur des actions effectuées.

Chapitre 3 : Exécution du code et interface utilisateur

Lors de l'exécution, l'utilisateur interagit directement avec l'interface graphique. Les formulaires permettent l'ajout et la modification des données, tandis que le tableau de bord se met à jour automatiquement.

Les écrans affichés correspondent à la phase d'exécution du code JavaScript.

Description des figures

- Figure 1 : Tableau de bord principal avec indicateurs
- Figure 2 : Interface de gestion des livres
- Figure 3 : Interface de gestion des auteurs
- Figure 4 : Graphique de répartition des livres par genre

Chaque figure illustre une fonctionnalité clé de l'application.

Conclusion

Ce projet met en évidence l'importance du JavaScript dans le développement d'applications web modernes. Il démontre qu'une application complète, interactive et structurée peut être réalisée sans backend, en exploitant intelligemment les capacités du navigateur.

Les compétences acquises à travers ce projet constituent une base solide pour des projets plus avancés intégrant des technologies comme Symfony, Node.js ou React.

Conclusion générale

En conclusion, l'application Bibliotheca répond efficacement à la problématique posée. Elle illustre une approche professionnelle, pédagogique et évolutive du développement web, tout en mettant en valeur le rôle central du JavaScript.