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. Les cinq sujets proposés

Chaque groupe choisit un seul sujet.

Sujet	Module 1	Module 2	Dashboard & API
TechStore	Produits (CRUD complet)	Catégories	KPI stock + FakeStoreAPI
HR Portal	Employés	Départements	KPI RH + RandomUser
CineTech	Films	Réalisateurs	KPI cinéma + OMDB/TMDB
Bibliotheca	Livres	Auteurs	KPI bibliothèque + OpenLibrary
CryptoFolio	Actifs crypto	Portefeuilles	KPI crypto + CoinGecko

6. Exemples d'interfaces Web – Références officielles

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/

7. Livrables

- Dossier du projet (HTML, CSS, JS).
- Rapport contenant entre 12 à 15 pages
- Lien GitHub obligatoire.
- Présentation du projet

Ce projet vous permet d'acquérir une expérience complète dans la réalisation d'une application Web moderne, structurée et dynamique.
Il constitue un premier pas vers les frameworks JavaScript, les API et le développement de systèmes professionnels.

8. 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.
Très bon courage, et bon projet à tous !