Énoncé du Projet : Application de Gestion Backoffice

Titre du Projet : Backoffice de Gestion "MyManager"

Description Générale

Vous allez développer une application **backoffice** en **JavaScript (VanillaJS)**, **HTML5**, et **CSS3**, sans utiliser de frameworks tels que Angular ou React. L'objectif est de construire une interface utilisateur qui permet de gérer des entités avec des fonctionnalités CRUD (Créer, Lire, Mettre à jour, Supprimer) et d'afficher des tableaux de bord avec des statistiques pertinentes.

Objectifs Principaux

1. Implémenter au moins **5 entités CRUD** dans l'application (par exemple : Utilisateurs, Produits, Commandes, Clients, Factures).
2. Utiliser **des données simulées** (faux fichiers JSON ou librairies comme Faker.js) **ou consommer des APIs publiques disponibles sur Internet (recommandé).**
3. Créer un tableau de bord (*dashboard*) qui affiche des statistiques (ex : nombre total d'utilisateurs, commandes en attente, revenus générés, etc.). le dashbord dois contenir minimum 5 chartes (piechart, donutchart, barchart, linechart, scatter-plot chart, box chart, histogram, etc.).
4. Fournir une **interface responsive** avec un design professionnel.
5. Utiliser uniquement **JavaScript natif**, sans frameworks modernes.
6. L'usage de bibliothèques CSS3 est autorisé(Bootstrap, Tailwind, etc.).
7. L'usage de bibliothèques Javascript est autorisé(jQuery, RamdaJS, LodashJS, etc.).

Spécifications Techniques

Fonctionnalités CRUD

Pour chaque entité, vous devez :

- **Créer** : Ajouter un nouvel élément via un formulaire.
- **Lire** : Afficher les données dans un tableau avec pagination.
    - Le screen doit contenir des filtres.
    - Le screen doit contenir des options de trie (Sorting).
    - Le screen doit contenir des options d'export via des fichiers CSV.
- **Mettre à jour** : Modifier les informations d'un élément existant.

- **Consulter un élément** : Dans le tableau de consultation il faut avoir une option de visualiser les détails d'un élément (un bouton "see details") dans une page appart, cette page doit contenir plus de détails sur le l'objet en question.
    - Le screen doit contenir une option d'export PDF.
- **Supprimer** : Supprimer un élément (après confirmation via un pop-up).

Dashboard

Le tableau de bord doit inclure :

1. **Des indicateurs clés** sous forme de cartes (exemple : nombre total de produits, utilisateurs, etc.).
2. **Un graphique ou un diagramme** (utilisez une bibliothèque comme Chart.js ou créez votre propre graphique en SVG ou Canvas).
3. **Filtres dynamiques** pour visualiser des données spécifiques (par exemple, afficher les commandes d'un utilisateur donné).

Données

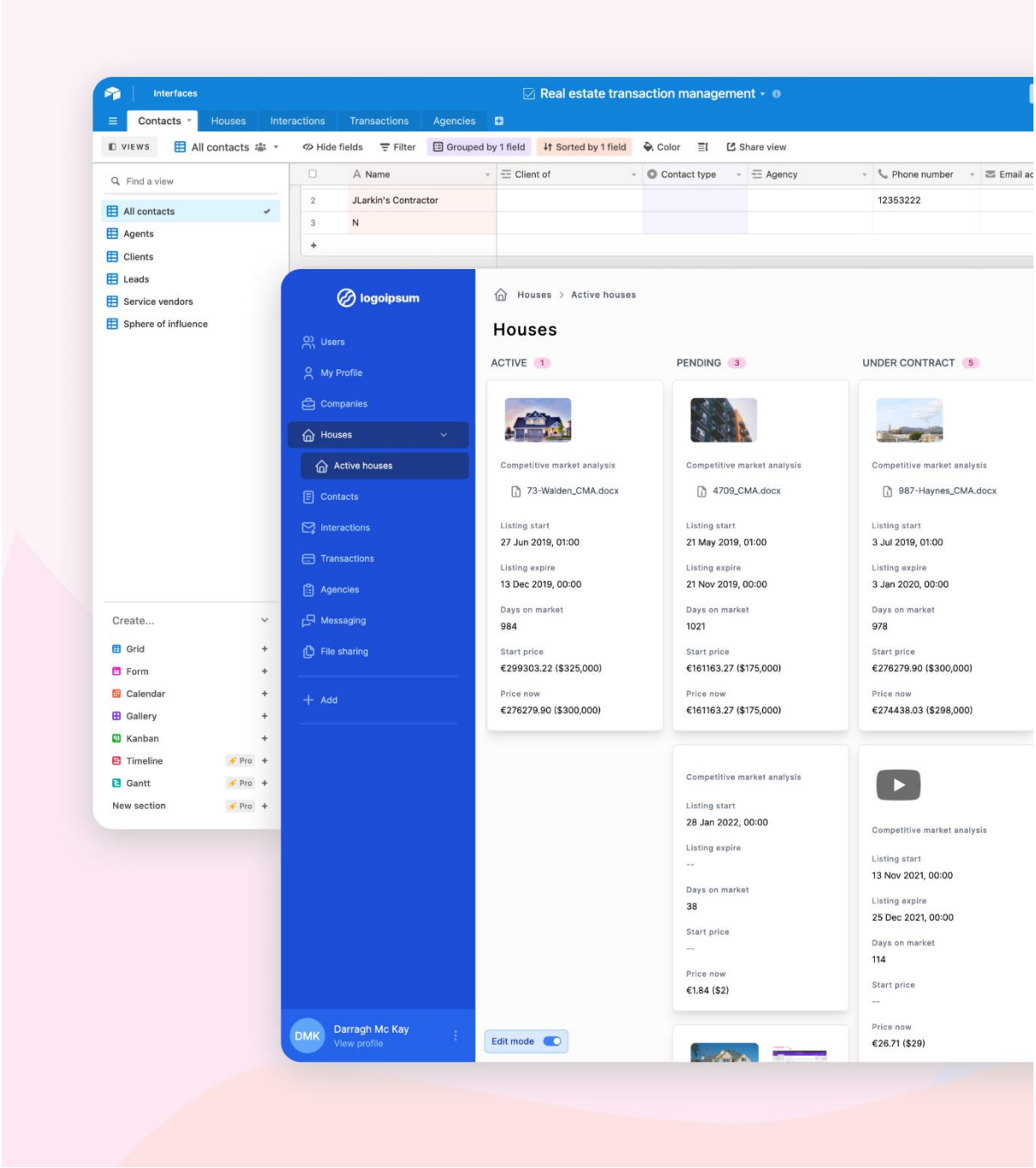
- **Option 1 : APIs publiques (Recommandé)**
  Sollicitez des APIs publiques telles que :
    - https://jsonplaceholder.typicode.com/ (Utilisateurs, Posts, Commentaires).
    - https://reqres.in/ (Utilisateurs).
    - https://pokeapi.co/ (Pokémon).
    - https://dummyapi.io/ (Utilisateurs, Posts).
- **Option 2 : Données simulées**
    - Utilisez un fichier JSON local pour stocker des données simulées.
    - Ou générez des données dynamiques en utilisant des librairies comme **JSONDB**, **Faker.js** ou **Mockaroo**.

Design (CSS3)

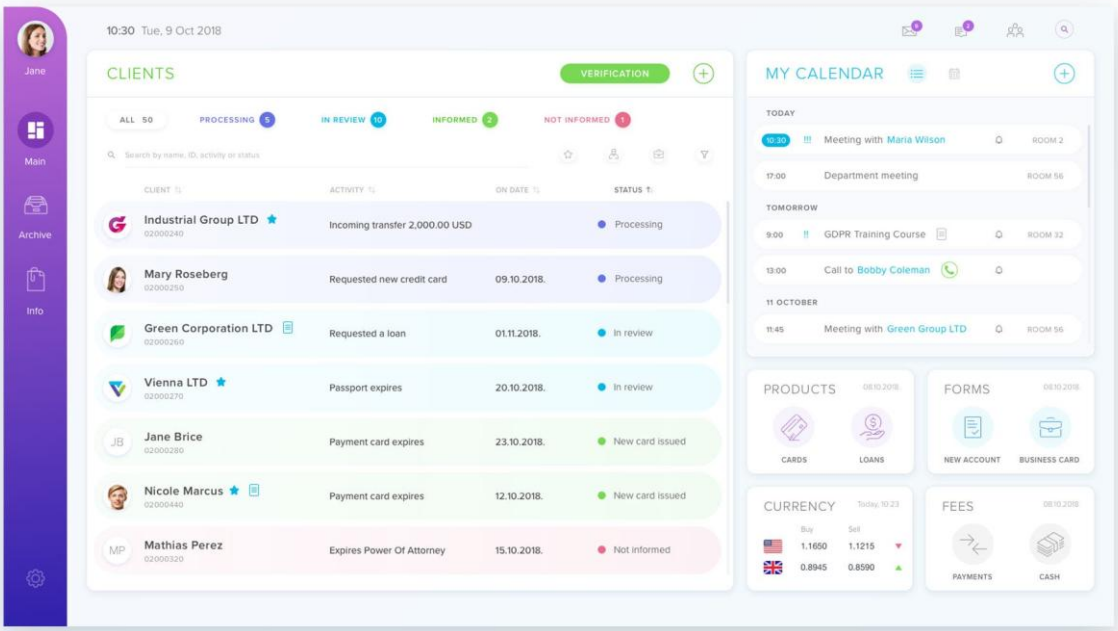
- Interface utilisateur avec une structure **responsive** (adaptée aux mobiles et aux tablettes).
- Utilisez **Grid** ou **Flexbox** pour organiser les éléments.
- Stylez les formulaires et les tableaux avec des couleurs, des bordures et des ombres modernes.
- Implémentez une page de Login avec un User statique (admin/admin).
- Implémentez un **menu latéral** pour naviguer entre les différentes entités CRUD et le tableau de bord.
- Implémentez une barre de navigation "navbar" contenant :

- le Logo de votre entreprise virtuelle.
- Un bouton de déconnexion.
- Un drop down permettant l'internationalisation des pages (Arabe, Français et Anglais).

Exemple 1 :

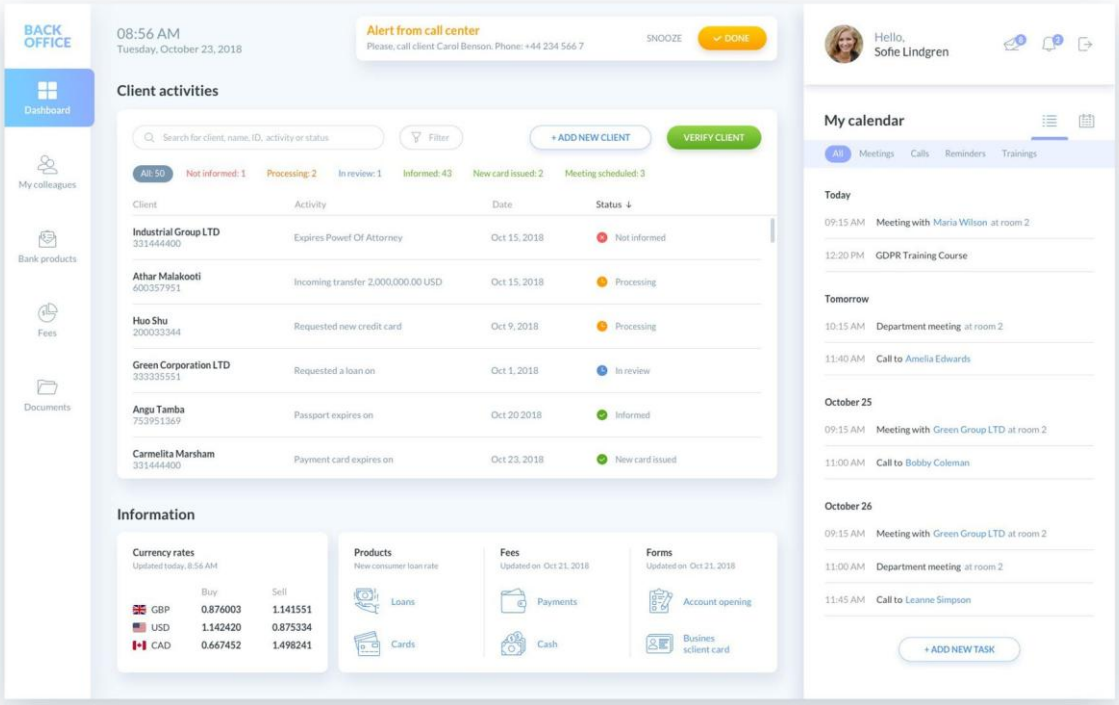


Exemple 2 :



Design by UXDA's
Oksana

Exemple 3 :



Design by UXDA's
Victoria

Contraintes

1. Aucun Framework (React, Angular, etc.) ne doit être utilisé.
2. Vous pouvez utiliser des bibliothèques tierces comme **Chart.js** ou **Lodash**.
3. L'utilisation d'outils/bibliothèques comme Tailwind ou Bootstrap est **autorisée** pour le CSS.

Livrables

1. Un dossier zip contenant :
    a. Les fichiers **HTML**, **CSS**, et **JavaScript**.
    b. Les fichiers JSON de données (si utilisés).
2. Un rapport PDF expliquant le "produit" en question.
3. Une présentation ppt (Power Point) à présenter le jour de la soutenance.
4. Un lien Github, Gitlab ou autre VCS contenant le code source, ainsi qu'un fichier **README.md** expliquant :
    a. Les fonctionnalités principales.
    b. Les APIs utilisées.
    c. Les étapes pour exécuter le projet.

Exemples d'APIs Publiques

1. **JSONPlaceholder** : https://jsonplaceholder.typicode.com/
   Entités disponibles : Utilisateurs, Posts, Commentaires.
2. **Reqres** : https://reqres.in/
   Entités disponibles : Utilisateurs.
3. **OpenWeatherMap** : https://openweathermap.org/
   Données météo pour afficher les conditions météo d'une ville.
4. **DummyAPI** : https://dummyapi.io/
   Données pour utilisateurs, posts, commentaires.
5. **PokeAPI** : https://pokeapi.co/
   Entité Pokémon pour tester une API amusante.

Fonctionnalités Avancées (Optionnelles)

1. **Recherche** : Implémentez une barre de recherche pour filtrer les données dans les tableaux.

2. **Filtres** : Ajoutez des filtres pour trier ou rechercher des données selon des critères spécifiques (par exemple, afficher uniquement les commandes en attente).
3. **Graphiques dynamiques** : Affichez des graphiques basés sur les données (exemple : évolution des ventes).
4. **Gestion des permissions** : Créez des utilisateurs admin, super_admin, user, agent, etc. avec des droits spécifiques.
5. **Déploiement :** Packagez puis Déployez votre application sur un serveur publique(vous pouvez utilisez un serveur Nginx local si aucun serveur publique gratuit n'est disponible).

Évaluation

- **Fonctionnalités CRUD** : 20%
- **Dashboard avec statistiques et graphiques** : 10%
- **Design et interface utilisateur** : 10%
- **Qualité du code (clean code), commentaires et organisation** : 10%
- **Bonne manipulation du DOM (via document ou via jQuery)** : 10%
- **Usage de fonctionnalités avancées (Classes, fonctions asynchrones et promesse, etc.)** : 10%
- **Bonne maitrise du i18n (Internationalisation)** : 10%
- **Mise en forme, mise en page, orthographe et bonne rédaction du rapport** : 10%
- **Présentation** : 10%

Ce projet permet aux étudiants de combiner des compétences en développement front-end (HTML, CSS, JS) avec des concepts pratiques comme les APIs et les CRUD, tout en restant dans un cadre sans frameworks