

TP3 - Application monopage interactive

Description

Produire une application monopage interactive (SPA) qui exploite et affiche une source de données ouvertes ainsi que plusieurs librairies JavaScript. L'application compilée doit être hébergée sur le service de déploiement automatisé Vercel.

Énoncé

Vous devez utiliser vos connaissances en programmation orientée objet et en requêtes asynchrones pour structurer le code d'une interface front-end. Vous devez écrire un code propre, bien structuré sans bug et sans avertissement et gérer les erreurs de manières appropriées. De plus, vous devrez obligatoirement utiliser des librairies pour accélérer le développement de votre application. Finalement, l'application doit être compilée et hébergée sur Vercel.

Vous ne disposez d'aucun dossier de travail. Vous devez démontrer votre compréhension du développement front-end en utilisant des outils modernes et en respectant les bonnes pratiques de développement. Vous devez montrer l'évolution à chaque cours. Vous aurez beaucoup de temps pour réaliser ce travail, ne le faites pas à la dernière minute. Votre application doit être fonctionnelle et bien présentée et en exploitant ce que l'on a vu en classe toute la session.

Au niveau du design, vous avez carte blanche pour votre application, mais elle doit être professionnelle et bien structurée. Vous devrez utiliser Font-Awesome pour les icônes et Tailwind CSS pour le CSS.

Vous n'avez pas d'obligation d'utiliser TypeScript, mais vous pouvez le faire si vous le souhaitez.

Objectifs

- Utiliser la programmation orientée objet pour structurer le code d'une interface front-end
- Récupérer et afficher des données dynamiquement à partir d'une source de données ouvertes à l'aide de l'API Fetch
- Exploiter efficacement des librairies via NPM.
- Gérer les erreurs de manière appropriée
- Afficher de la rétroaction à l'utilisateur lors des actions de l'utilisateur (ex: chargement, erreur, succès)
- Compiler, optimiser et héberger l'application
- Écrire un code propre, professionnel et bien structuré sans bug et sans avertissement dans la console

Consignes

- Vous devez réaliser une application monopage interactive (SPA) qui exploite une source de données ouvertes. L'application est un prototype et doit rester minimaliste.
- L'application comporter minimalement 2 pages dont une qui cite les sources de données et les librairies utilisées.

- Les données récupérées doivent être affichées et animées lors de l'affichage et lorsque l'utilisateur interagit avec celles-ci.
- Vous devez utiliser les librairies suivantes: Moment, PageJS, Vite, AnimeJS, TailwindCSS et Font-Awesome.
- Le CSS et le Javascript doit être séparé dans plusieurs fichiers distincts. À la fin, vous devrez compiler l'application avec Vite afin d'avoir 1 seul fichier CSS et 1 seul fichier Javascript.
- Vous devez avoir 2-3 images dans votre application, vous devez les compresser par programmation pour qu'elles soient légères et recadrées.
- Vous devez afficher une date formatée à l'aide de la librairie Moment.
- L'application doit être hébergée sur le service de déploiement Vercel.
- Vous devez utiliser Font-Awesome pour les icônes.

Sources de données

Vous avez le choix de la donnée que vous souhaitez exploiter. La donnée doit être accessible via une source de données ouvertes et doit retourner un objet au format JSON. Vous devez analyser la structure de la donnée et déterminer les informations que vous souhaitez afficher dans votre application.

Voici une liste de sources de données ouvertes que vous pouvez utiliser:

- **Données ouvertes Québec** : Le portail des données ouvertes du gouvernement du Québec.
- **Données Canada** : Le portail des données ouvertes du gouvernement canadien.
- **Ville de Montréal - Données ouvertes** : Le portail des données ouvertes de la ville de Montréal.
- **Bixi** : Données ouvertes sur les stations et les trajets de Bixi à Montréal en temps réel.
- **data.gouv.fr** : Le portail officiel des données ouvertes du gouvernement français.
- **European Data Portal** : Le portail des données ouvertes de l'Union européenne.
- **World Bank Open Data** : Données ouvertes de la Banque mondiale sur le développement mondial.
- **Google Dataset Search** : Un moteur de recherche pour trouver des ensembles de données disponibles en ligne.
- **Open Data Network** : Un réseau de données ouvertes provenant de diverses sources américaines.
- **Open Weather Map** : API fournissant des données météorologiques ouvertes.
- **US Census Bureau** : Données démographiques et économiques ouvertes des États-Unis.
- **Liste des pays**
- **Open Library**
- **Open Meteo**
- **Open Movie Database** nécessite une clé API
- **API de la NASA** nécessite une clé API
- **Cocktail DB** nécessite la clé API "1"

- **Open Brewery** nécessite une clé API

Librairies obligatoires

- **Moment** : Pour la gestion et le formatage des dates.
- **PageJS** : Pour la gestion du routage côté client.
- **Vite** : Pour la compilation et le bundling de l'application.
- **AnimeJS** : Pour les animations.
- **TailwindCSS** : Pour le design et le style de l'application.
- **Font-Awesome** : Pour les icônes.

Librairies optionnelles si vous voulez aller plus loin (optionnel)

La liste suivante contient des librairies supplémentaires que vous pouvez utiliser pour enrichir votre application. Leur utilisation n'est pas obligatoire, mais elles peuvent vous aider à ajouter des fonctionnalités intéressantes et à améliorer l'expérience utilisateur.

- **Three.js** : Pour les graphiques 3D et les animations avancées (optionnel).
- **Chart.js** : Pour les graphiques et les visualisations de données (optionnel).
- **Leaflet** : Pour les cartes interactives (optionnel).
- **D3.js** : Pour la manipulation avancée des données et les visualisations (optionnel).
- **SortableJS** : Pour le drag-and-drop (optionnel).
- **AOS (Animate On Scroll)** : Pour les animations au défilement (optionnel).
- **Swiper** : Pour les carrousels et les sliders (optionnel).
- **ToneJs** : Pour la synthèse et la manipulation audio (optionnel).
- **Toastr** : Pour les notifications (optionnel).

Modalités de remise

Date de remise

Le site doit être en ligne au plus tard le **23 novembre à 23h59**, aucun retard accepté.

Remise

Vous devez déposer votre projet compilé au format ZIP sur **Teams**. Vous devez remettre le lien déployé sur Vercel

Vous devez me présenter obligatoirement votre projet au cours 24 ou 25.

Pondération

Le travail compte pour 30 % de la note finale

Retard

Aucun retard accepté

Critères d'évaluation

- Juste compréhension des fonctionnalités demandées à l'intérieur du devis (10 %)

- Utilisation correcte d'un langage de programmation orienté objet côté client (30 %)
- Gestion adéquate de l'affichage dynamique (20 %)
- Traitement et exécution adéquates des réponses synchrones et asynchrones (25 %)
- Structure et qualité optimale de la programmation (15 %)

Plagiat

Vous devez citer les extraits de code qui dépassent une ligne ou deux et suivre un tutoriel (en tout ou en partie) sera considéré comme du plagiat. Vous avez tout dans les notes de cours pour réussir ce travail. L'utilisation d'un générateur de code par intelligence artificielle est interdite.

Il s'agit d'un travail strictement individuel. Vous pouvez poser des questions à vos collègues, mais ce doit être votre logique et vos scripts. L'objectif est de développer votre autonomie. En cas de plagiat, je devrai appliquer les sanctions de la PIEA (Politique institutionnelle d'évaluation des apprentissages).

Si vous êtes vraiment bloqué, n'hésitez pas à m'écrire sur Teams. N'attendez pas à la dernière minute pour demander de l'aide.