Projet Framework Web 2 - Consignes et Barème

Introduction

L'objectif de ce projet est de vous permettre de mettre en pratique tout ce que vous avez appris sur Vue.js. Vous allez devoir créer un **site web ou une application** autour d'un thème qui vous **intéresse personnellement** ou qui peut être pertinent pour votre parcours professionnel. Ce projet est une excellente opportunité de consolider vos compétences et de produire une réalisation que vous pourrez inclure dans votre portfolio.

Ce qui est attendu :

- **Travail individuel ou en binôme :**
  - Vous pouvez réaliser ce projet **seul** ou **en groupe de deux personnes maximum**.
  - Si vous travaillez à deux, vous devez détailler dans le fichier README :
    - **Qui a fait quoi.**
    - **Comment vous vous êtes organisés.**
- **Choix du thème :** Vous êtes libre de choisir un thème pour votre site ou application (exemple : site vitrine / CV interactif, cinéma, météo, sport, éducation, musique, etc.).
- **Contenu et pertinence :** Votre projet doit être **cohérent et pertinent** par rapport au thème choisi. Pensez à la valeur ajoutée pour l'utilisateur et à ce qui rendra votre projet unique.
- **Utilisation des fonctionnalités Vue.js :** Vous devrez utiliser les concepts vus en cours, tels que les composants, la réactivité, les routes, et les appels API.
- **Appel d'API :** Vous devrez intégrer une API (voir les suggestions ci-dessous ou utiliser votre propre API) pour ajouter du contenu dynamique à votre projet.
- **Finalisation et rendu :** Votre projet devra être complet, fonctionnel et respecter les bonnes pratiques (code propre, composants réutilisables, etc.).

Livrable :

- Le projet doit être compressé au format **ZIP**.
- Vous devez inclure :
  - Le dossier complet de votre projet Vue.js.
  - Un fichier **README.md** expliquant :
    - L'objectif de votre projet.
    - Les fonctionnalités principales.
    - Qui a fait quoi (si vous êtes deux).
    - Comment vous vous êtes organisés (si vous êtes deux).
    - Les difficultés que vous avez rencontrées et les solutions apportées.
    - Comment installer et lancer l'application.
- **Date limite de rendu : 19 mars 2025.**

⚠ Tout retard entraînera une pénalité dans la note finale.

Barème détaillé :

1. Maîtrise des bases de Vue.js – 5 points

- **Rendu déclaratif des variables**
  - Utilisation correcte des variables dans le template avec des données réactives (`ref`).
- **Liaison de données**
  - Bonne utilisation des directives comme `v-bind`, `v-model`, et `v-if`.
- **Utilisation intelligente des composants**
  - Découpage du projet en plusieurs composants bien définis et réutilisables.
- **Communication entre composant**
  - Passage de props pour les données descendantes.
  - Utilisation d'événements personnalisés ou d'émission (`emit`) pour les données montantes.
- **Utilisation pertinente des options d'un composant**
  - Déclaration et utilisation efficace de `props`, `computed`, `watch`, ou des hooks du cycle de vie.
- **Utilisation judicieuse des directives**
  - Application correcte et variée des directives (`v-for`, `v-on`, etc.) dans les composants.

---

2. Maîtrise du router – 5 points

- **Configuration correcte du router**
  - Mise en place d'un fichier de routes avec une structure claire et fonctionnelle.
- **Navigation entre composants**
  - Utilisation de `router-link` et de la méthode `router.push` pour naviguer dynamiquement.
- **Passage de paramètres ou de query intelligents**
  - Exploitation des paramètres dynamiques ou des queries pour transmettre des informations entre les routes.
- **Utilisation de sous-routes (si pertinent)**
  - Création et gestion d'une structure de sous-routes pour des sections imbriquées dans un composant.

---

3. Maîtrise des appels RESTful via Axios – 5 points

- **Appels API corrects**
  - Réalisation d'appels `GET`, `POST`, ou autres selon les besoins du projet.
- **Gestion des succès et des erreurs**
  - Traitement des erreurs API (affichage d'un message d'erreur, log).
- **Formatage des données reçues**
  - Transformation ou adaptation des données API pour les afficher correctement.
- **Affichage dynamique des données**
  - Mise à jour réactive de l'interface en fonction des données reçues.

---

4. Maîtrise du Store (Pinia) – 3 points

- **Utilisation de getters**
  - Définition et usage de getters pour centraliser des calculs ou des transformations.
- **Utilisation des mutations/actions**

- Modification des données avec des mutations claires ou des actions bien définies.

- **Réutilisation du store entre les composants**
  - Appels centralisés au store dans plusieurs composants.
- **Structure claire et logique du store**
  - Organisation des modules et des fonctions (si le store est modularisé).

---

5. Propreté du code – 2 points

- **Commentaires explicatifs**
  - Présence de commentaires clairs pour expliquer les parties importantes ou complexes.
- **Indentation et organisation du code**
  - Code bien structuré, facile à lire, avec des indentations cohérentes.
- **Présence d'un README explicatif**
  - Un fichier `README.md` qui présente le projet, ses fonctionnalités, et comment le lancer.

---

6. Bonus (optionnel) – jusqu'à 3 points supplémentaires

- **Personnalisation ou créativité**
  - Ajout de fonctionnalités ou d'effets visuels qui dépassent les exigences de base.
- **Tests unitaires ou E2E**
  - Mise en place de tests pour vérifier le bon fonctionnement des composants ou des fonctionnalités.
- **Performance ou accessibilité.**
  - Optimisation des performances (lazy loading, minimisation des redondances).
  - Amélioration de l'accessibilité (respect des normes ARIA, navigation au clavier).

Note totale : 20 points (+3 points bonus)

Suggestions d'APIs gratuites :

APIs populaires :

- **Studio Ghibli API**
  - Données sur les films et personnages des célèbres films du Studio Ghibli.
- **Rick and Morty API**
  - Informations sur les personnages, lieux et épisodes de Rick et Morty.
- **OpenWeather API**
  - Données météo pour une localisation précise.
- **The Dog API**
  - Informations sur différentes races de chiens, avec photos.
- **NASA API**
  - Accès à des données spatiales (images, missions, etc.).

Créer ou utiliser une API personnalisée :

Si vous avez une idée spécifique ou une API personnelle, vous êtes libre de l'utiliser à condition qu'elle apporte du contenu ou des fonctionnalités dynamiques à votre projet.