Travail écrit Web Mob. UI M51/1-2 - 25.03.2024

Nom/Prénom : _____

- Tout document autorisé.
- Utilisation de l'ordinateur restreinte à l'utilisation d'un lecteur de PDF, un éditeur de code, un browser limité au site Internet du cours (support de cours, liens de premier niveau, et exercices compris) et au localhost. Pas de logiciel d'aide ou de support AI, tel que Copilot, etc…
- Pas d'autre communication réseau (ou directe) autorisée pendant le travail.
- **Toute non-observation des règles ci-dessus entraîne la note 1 et supprime toute possibilité de remédiation.**

Vous rendrez **le dossier complet** de votre projet par **email**, via un **ZIP portant votre nom**, à l'adresse **loris.gavillet@heig-vd.ch**. Vous êtes **responsable** de son contenu, et devez vous **assurer** de la bonne réception du dossier avant de quitter la salle. **Le travail et la feuille de données** sont à rendre **au plus tard 3h (4 périodes) après le début de l'examen**. En plus des temps annoncés pour chaque exercice, 15 minutes sont réservées comme marge de relecture.

A. PARTIE PRATIQUE

MISE EN PLACE ET STRUCTURE (15 MINUTES)

Vous allez réaliser une petite application pour une plateforme de petites annonces en ligne, appelée "Anirdo". Il est possible d'y afficher une liste d'annonces complète, voir la vue détaillée de chaque annonce, une liste de catégories et afficher une liste d'annonces pour une catégorie précise. Une gestion des annonces interessantes ("starred") permet d'ajouter ou non une annonce dans la liste dédiée et un onglet affiche son contenu.

Une structure de base vous est donnée, contenant un projet Parcel, le markup HTML/CSS, des fichiers JS et autres assets nécessaires au développement. Vous pouvez la télécharger via ce lien https://github.com/lgavillet/webmobui-24-exa. Une fois téléchargé, effectuez un "npm install" et vous pourrez ensuite démarrer le projet avec "npm run start", comme vu en cours.

Avant de débuter, pour éviter tout problème, n'oubliez pas :
- De vider votre localStorage (soit par l'onglet dédié des outils développeurs, soit par le code "`localStorage.clear()`" dans la console javascript).
- De supprimer tout service worker installé via les outils développeurs

Toute trace de code (nom de variables, de fonctions, …) se référant au champ lexical d'autres projets entraine une diminution des points (artists, songs, …).

Le fichier src/index.js contient la logique de base pour naviguer entre les sections, selon le concept du hashchange vu ensemble. Pour la structure des hashs, vous devez utiliser les URLs suivantes:
- `#latest` - Liste de toutes les annonces
- `#annonces-[id]` - Vu détaillée d'une annonce (où [id] est à remplacer par l'id de l'annonce. Ex.: `#annonces-1234`)
- `#categories` - Liste de toutes les catégories
- `#categories-[id]` - Liste des annonces d'une catégorie (où [id] est à remplacer par l'id de la catégorie). Ex.: `#categories-1234`)
- `#interested` - Vue des annonces intéressantes marquées d'une étoile

Modèles de l'API

Catégorie
- *id* - ID de la catégorie
- *name* - Nom de la catégorie
- *count* - Nombre d'annonces
- *icon_url* - URL de l'icône de la catégorie

Annonce
- *id* - ID de l'annonce
- *title* - Titre de l'annonce
- *image_url* - Url de l'image de l'annonce
- *description* - Description détaillée de l'annonce (uniquement disponible sur l'endpoint du détail de l'annonce)
- *price* - Prix de l'objet

1. AFFICHER LA LISTE DE TOUTES LES ANNONCES (30 MINUTES)
La liste de toutes les annonces présentes sur le serveur est obtenue grâce à l'URL `/api/annonces` et devra se trouver sur le hash "#latest".

Vous pouvez créer une fonction d'aide qui fait le fetch et renvoie un tableau d'annonces dans le fichier src/api.js et placer la logique d'affichage dans un autre fichier, comme par exemple annonces.js (ou nommage similaire).

Servez-vous ensuite de l'élément d'exemple HTML présent dans la section "#section-annonces", puis dans la liste ".annonces" pour servir de base à chaque élément de la liste. Vous devez pour cela créer un custom élément, comme vu en cours. Référez-vous au code HTML pour connaitre le contenu des balises à remplacer. Vous pouvez utiliser soit .innerHTML ou les templates vus en cours.

Le titre de la section doit être "Dernières annonces ([nombre d'annonces])". Exemple : "Dernières annonces (20)".

2. AFFICHER LES CATÉGORIES DANS LA SECTION CATÉGORIES (20 MINUTES)
Dans la section "#categories-section", se trouve une liste de catégories à remplir que vous pouvez obtenir par l'API correspondante `/api/categories`. Le hash a utiliser est '#categories'.

Comme plus haut, créez une fonction d'aide dans api.js et placer la logique d'affichage dans un fichier correspondant, comme par exemple "categories.js".

Servez-vous ensuite de l'exemple d'élément HTML dans ".categories" présent dans la section "#section-categories" pour servir de base à chaque élément de la liste, comme vu en cours. Référez-vous au code HTML pour connaitre le contenu des balises à remplacer.

Le titre de la section doit être "Catégories".

3. AFFICHER LA LISTE DES ANNONCES D'UNE CATÉGORIE (20 MINUTES)
En réutilisant une bonne partie du code précédent, vous devez à présent gérer l'affichage des annonces d'une catégorie. Cela arrive lorsque l'on clique sur le nom d'une catégorie, avec le lien "#categories-[id]". L'endpoint à utiliser est le suivant: `/api/categories/[id]/annonces` où [id] est l'id de la catégorie souhaitée.

Le titre de la section doit être "Catégories > [nom de la catégorie] ([nombre d'annonces])". Référez-vous au code HTML pour connaitre le contenu des balises à remplacer.

4. AFFICHER LES DÉTAILS D'UNE ANNONCE (20 MINUTES)
L'affichage des détails d'une annonce se fait sur le hash '#annonces-[id]' et vous pouvez récupérer les détails d'une annonce via le endpoint `/api/annonces/[id]`.

Pour les détails d'une annonce, vous pouvez utiliser le markup de la section "#section-annonce-details" (annonce au singulier).

Le titre de la section doit être "Annonces > [titre de l'annonce]". Référez-vous au code HTML pour connaitre le contenu des autres balises à remplacer.

5. GESTION DE L'AJOUT AUX ANNONCES INTÉRESSANTES (20 MINUTES)
Pour permettre d'ajouter une annonce dans les annonces intéressantes, servez-vous des fonctions d'aides que nous avons vu ensemble, déjà présente dans "src/lib/local-storage.js".

Pour cela, créer un event listener au clic sur le bouton étoile de chaque élément de la liste qui va gérer son ajout/suppression des annonces "starred", ainsi que changer l'icône de la ligne correspondante (voir HTML pour les noms d'icône).

6. AFFICHAGE DES ANNONCES INTÉRESSANTES (20 MINUTES)
L'affichage des annonces importants se fait sur le hash '#starred' et doit simplement lister les annonces marquées comme intéressantes.

Lorsque l'on est dans la vue intéressantes et que l'on enlève un élément, on s'attend à ce que celui-ci soit directement enlevé de la liste (uniquement dans cette vue…).

Le titre de la section doit être "Intéressantes ([nombre])" où [nombre] représente le nombre d'éléments marqués.

BONUS. APPLICATION PWA (5 MINUTES)
A faire en dernier pour éviter des problèmes lors du développement.

Une fois votre application prête, transformez là en PWA en liant un fichier manifest et en enregistrant un service worker pour rendre l'application installable. Vous pouvez vous servir des fichiers manifest.webmanifest et worker.js présents dans src/, comme vu en cours. Vérifiez votre intégration grâce à l'inspecteur.

B. PARTIE THÉORIQUE

1. **Expliquez en quoi la structure de projet que nous avons vue en cours se rapproche d'un standard MVC (Model-Vue-Controller) et la répartition des différentes fonctionnalités.**

2. **Expliquez avec vos mots l'utilité d'un service worker et son fonctionnement approximatif**

3. **À quoi sert un fichier manifest ?**

4. **Quels sont les avantages d'une PWA par rapport à une application native ? Ses inconvénients/ limitations ?**