Projet #1

Pre-flight checklist

Bachelor année 2 - **L'École Multimédia**

l'école multimédia

Sommaire

Contexte

Présentation

Vous venez d'être recruté comme développeur par Karim - directeur d'une toute nouvelle startup dénommée **GoodStufForDev** - afin de participer au développement de leur plateforme.

GSFD a comme projet de développer **Pre-flight Checklist**, une application web permettant à des développeurs de partager des listes d'éléments à vérifier afin de valider des pré-requis à réaliser.

Par exemple, Pre-flight checklist peut être utilisé pour:

- Vérifier si toutes les actions pour un bon référencement naturel ont été réalisées
- Vérifier si toutes les tâches pour le déploiement d'un site web ont été réalisées
- Vérifier si toutes les mesures d'accessibilités ont été prises
- Etc …

Les checklists constituent ainsi des recettes pour vérifier si tout a été bien fait.

Vous êtes embauché en tant que **développeur front-end** et serez responsable de l'interface du produit.

GoodStufForDev

GSFD a été créé par Karim, ancien directeur technique Java.

L'idée du projet lui est venue en constatant le coût (en temps et en argent) des petits oublies que font les développeurs peu organisés.

Avec ces checklists, les entreprises peuvent mieux structurer leurs processus et gagner en productivité.

Charte graphique

GSFD est une toute nouvelle startup et la charte graphique est minimale :

Typographie : Roboto
Couleurs : https://coolors.co/palette/26547c-ef476f-ffd166

Contraintes

Vous travaillerez **seul** sur le projet.

L'application web que vous allez développer doit impérativement répondre à ces critères :

1. L’application doit utiliser le framework **React**
2. Elle doit répondre aux demandes décrites dans le dossier de conception
3. Elle doit utiliser l’API fournie
4. L’application est en **anglais**

Libertés

1. Vous êtes libre d’utiliser ou non **des librairies CSS**
2. Vous êtes libre d’utiliser **des librairies Javascript tiers**

Objectifs

A l’issue du module, vous devrez avoir réalisé les éléments suivant :

1. La conception de l’interface graphique de l’application
2. Un prototype fonctionnel et en ligne de l’application

Dossier de conception

Le produit est une **application web** utilisable avec un navigateur web et en **langue anglaise**

Elle doit **s’adapter au périphérique** comme toute application moderne (mobile, tablette, desktop, paysage et portrait)

Liste fonctionnelle

- Un **utilisateur**
  - peut créer, modifier et supprimer des **checklists**
- Une **checklist**
  - est composée
    - d’un titre
    - d’une description
    - d’une liste de **tâches** ordonnées
    - D’un état (**0**: vierge, **1**: en cours, **2**: terminée)
- Une **tâche** est composée
  - d’un titre
  - une description (optionnelle)
  - d’un statut (**0**: non fait, **1**: fait)

Note : Votre application ne permettra pas pour l’instant l’inscription de nouveaux utilisateurs.

Liste des écrans

L’application est composée de **3 écrans**

Dashboard

C’est le point d’entrée pour l’utilisateur.
Sur cet écran, il peut consulter l’ensemble des **checklists** qu’il a créé.
Si aucune liste n’a été créée, on lui indique par un message.

Affichage

Chaque **checklist** est affichée avec les informations suivantes :

1. Titre et description
2. Etat (vierge, en cours, terminée)
3. Nombre de **tâches** faites / nombre de **tâches** totales
4. Un bouton pour l’éditer
5. Un bouton pour la supprimer

Un bouton **NOUVEAU** de bonne taille est accessible.

Actions

1. Au click sur une **checklist**, l’utilisateur est dirigé vers l’écran **Checklist**
2. Au click sur le bouton editer d’une **checklist**, l’utilisateur est dirigé vers l’écran **Formulaire**
3. Au click sur le bouton supprimer d’une **checklist**, l’utilisateur est invité à confirmer la suppression. S’il confirme, la **checklist** est supprimée.
4. Au click sur le bouton **NOUVEAU**, l’utilisateur est dirigé vers l’écran **Formulaire**

Checklist

Cet écran affiche UNE **checklist**.
L’utilisateur peut interagir avec les **tâches** et revenir sur le **dashboard**

Affichage

Les informations suivantes doivent apparaître :

6. Titre et description
7. Liste des tâches
8. Etat (vierge, en cours, terminée)

Actions

1. Au click sur une **tâche**, son état est inversé (fait->non fait, non fait->fait)
2. Si toutes les **tâches** sont à l'état “fait”, l’état de la **checklist** passe à “terminée”

Formulaire

Cet écran affiche un formulaire permettant de créer/modifier une **checklist**.
L’utilisateur peut remplir les champs du formulaire, ajouter des **tâches** ou revenir sur le **dashboard**

Affichage

Le formulaire permet de renseigner l'ensemble des données d'une **checklist**.
Le formulaire permet d'ajouter, de modifier ou de supprimer une **tâche**.

Un bouton **SAUVEGARDER** de bonne taille est accessible.

Actions

3. Au click sur le bouton **Sauvegarder**, la **checklist** est mémorisée et l'utilisateur est redirigé vers le **dashboard**

L'API Serveur

Clément, le développeur back-end de **GSFD ,** vous a préparé une API pour sauvegarder les informations des **checklists**.

Il a également rédigé une documentation pour vous aider à interagir avec le serveur que vous retrouverez dans les **annexes** de ce document.

Compétences à valider

La notation de votre projet se fera en évaluant les compétences de la certification regroupées ci dessous :

Compétence 1.2 : Développer des interfaces utilisateur

L'interface est conforme au dossier de conception
L'interface s'adapte au type d'utilisation de l'application et notamment à la taille, au type et à la disposition du support
La charte graphique est respectée
La réglementation en vigueur est respectée
Le code est documenté
Les tests unitaires ont été réalisés pour les composants concernés
Le jeu d'essai fonctionnel est complet
Les tests de sécurité sont réalisés
La documentation technique des interfaces utilisateur est comprise, en langue française ou anglaise (niveau B1 CECRL pour l'anglais)
La démarche structurée de résolution de problème est adaptée en cas de dysfonctionnement
Le système de veille permet de suivre les évolutions technologiques et les problématiques de sécurité en lien avec les interfaces utilisateur

https://www.francecompetences.fr/recherche/rncp/37873/

Les éléments indispensables

- Documenter le code en anglais
- Respecter les normes d'accessibilité et de RGPD
- Valider systématiquement les entrées
- Planifier et suivre les tâches de développement

Rendu final

Votre rendu final prendra la forme d'une archive Zip et devra comporter les éléments suivants :

1. Les documents de conceptions de l'interface
    a. Zoning
    b. Wireframe
    c. Maquette Figma
2. Les documents réalisés durant le développement
    a. Journal de développement
    b. Recherches sur internet (résolution de problèmes, veille technique)
    c. Adresse Github
3. Le code de votre application

Vous devez livrer une archive de votre livrable avec l'ensemble des éléments, à l'exception des dossiers suivant :

- node_modules

Cette archive aura comme titre votre nom et prénom avec votre classe.
Exemple: `elon_musk_projet1_BACH2.zip`

Attention : Un rendu non livré ou en retard vous pénalise pour la certification (non validation des compétences RNCP)

Conseils

- Bien prendre le temps d'analyser le brief et comprendre le client
- Organisez-vous et planifiez votre travail : donnez vous des objectifs intermédiaires
- Planifiez des sessions de travail régulière
- Utilisez Git pour versionner votre code dès le départ
- Ne jamais être trop ambitieux
- Faites directement des documents de conception présentables
- Mettez en oeuvre les bonnes pratiques vues en cours
- Refactoriser pour éviter le code redondant
- Soignez la qualité de votre code (commentaires, indentation)
- N'attendez pas la fin pour commencer à travailler sur votre projet
- Pensez à la facilité d'utilisation et la qualité du résultat !

Annexe : Documentation de l'API

Pour accéder aux données, vous devrez réaliser des appels AJAX asynchrone avec les URL mentionnées et les données demandées.

Pour ce faire, vous pouvez utiliser la commande Javascript fetch() ou la librairie Axios.

URL racine de l'API

L'URL racine de l'API est accessible à cette adresse :

Identification

Certaines actions de l'API attendent une identification.

Elles sont noté avec la mention **Authentification par token**

Pour exécuter ces actions, vous devez transmettre **un token dans le header HTTP de la requête vers le serveur.**

Vous trouverez des informations pour réaliser cette action avec Axios ici :

- https://axios-http.com/fr/docs/req_config
- https://blog.logrocket.com/using-axios-set-request-headers/

L'API attend la clé "**token**" avec votre token dans le header HTTP.

Génerer un token

Vous avez donc besoin de générer un token unique qui va vous permettre de vous identifier.

Pour générer votre token, accédez à l'URL ci dessous :

https://greenvelvet.alwaysdata.net/pfc/subscribe

Copiez/coller ce token et gardez le précieusement, c'est lui qui vous permettra de vous identifier et d'accéder à vos données.

Attention : Ce token doit rester secret. Vous devrez n'utiliser qu'un unique token pour interagir avec l'API.

Réponse

L'API répondra toujours à votre requête au format JSON.

Commandes

L'API est composée de 8 commandes.

API infos

Slug	/
Http	get
Objectif	Renvoie des informations sur l'API
Données attendues	aucune
Données renvoyées	**name** : le nom de l'API **version** : la version actuelle de l'API **statut** : Le statut du serveur ("ready", "down")

Ping

Slug	/ping
Http	get
Objectif	Un point d'entrée pour tester l'API
Données attendues	aucune
Données renvoyées	**response** : "pong"

Exemple d'url: https://greenvelvet.alwaysdata.net/pfc/ping

Ajouter une checklist

Slug	/checklist/add
Http	post

Objectif	Ajoute une nouvelle checklist
Données attendues	**Authentification par token** **title** : le titre de la checklist **description**: sa description **todo**: Un tableau composé des tâches à réaliser. Chaque todo est un objet au format JSON comportant les clés ‘title’ et ‘description’
Données renvoyées	**Id**: l’identifiant unique de la nouvelle checklist

Récupérer toutes les checklists

Slug	/checklists
Http	get
Objectif	Renvoie les données de toutes les checklists d’un utilisateur
Données attendues	**Authentification par token**
Données renvoyées	**response** : Un tableau regroupant les données des checklists au format JSON

Récupérer une checklist

Slug	/checklist
Http	get
Objectif	Renvoie les données d’une checklist de l’utilisateur
Données attendues	**Authentification par token** **Id**: l’identifiant de la checklist
Données renvoyées	Un objet JSON regroupant les données d’une checklist (id, title, descriptions, status, todo, created_at)

Exemple d’url: `https://greenvelvet.alwaysdata.net/pfc/checklist?id=3`

Supprimer une checklist

Slug	/checklist/delete
Http	get
Objectif	Supprime une checklist de l’utilisateur

Données attendues	**Authentification par token** **Id**: l'identifiant de la checklist
Données renvoyées	**done**: true ou false suivant le résultat de l'opération

Exemple d'url:
`https://greenvelvet.alwaysdata.net/pfc/checklist/delete?id=3`

Modifier une checklist

Slug	/checklist/update
Http	post
Objectif	Modifie les données d'une checklist de l'utilisateur
Données attendues	**Authentification par token** **Id**: l'identifiant de la checklist **title** : le titre de la checklist **description**: sa description **todo**: Un tableau composé des tâches à réaliser. Chaque todo est un objet au format JSON comportant les clés 'title', 'description' et 'statut'
Données renvoyées	**done**: true ou false suivant le résultat de l'opération

Modifie le statut d'une checklist

Slug	/checklist/statut
Http	get
Objectif	Modifie le statut d'une checklist de l'utilisateur
Données attendues	**Authentification par token** **id**: l'identifiant de la checklist **statut**: le nouveau statut de la checklist (0, 1 ou 2)
Données renvoyées	**done**: true ou false suivant le résultat de l'opération

Exemple d'url:
`https://greenvelvet.alwaysdata.net/pfc/checklist/statut?id=3`