Modèle de copie :

Évaluation en cours de formation

Développeur Web et Web Mobile

Ceci est un modèle de copie. N'oubliez pas de renseigner vos prénom/nom, ainsi que le nom et le lien vers le projet.

Vous pouvez bien sûr agrandir les cadres pour répondre aux questions sur la description du projet si nécessaire.

Prénom : Christophe
Nom : LEMOINE

ATTENTION ! PENSEZ À RENSEIGNER VOS NOM ET PRÉNOM DANS LE TITRE DE VOS FICHIERS / PROJETS !

Nom du projet : Charles Cantin - Photographe.

Lien Github du projet : https://github.com/pantaflex44/studi-evals/tree/eval3
Lien Drive du projet (si nécessaire) : https://github.com/pantaflex44/studi-evals/tree/eval3/projet
Attention ! Merci de bien classer vos documents dans votre Github ou votre drive.
URL du site (si vous avez mis votre projet en ligne) : https://pantaflex44.me/eval3/

Description du projet

1. Liste des compétences du référentiel qui sont couvertes par le projet

Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité

2. Résumé du projet en français d'une longueur d'environ 20 lignes soit 200 à 250 mots, ou environ 1200 caractères espaces non compris

Cette évaluation à pour but d'exposer l'amplitude des connaissances acquises pour le développement la partie Front-End d'une application Web et Web mobile.

L'objectif demandé, est la réalisation de la partie Front d'un site vitrine (portfolio) pour un photographe particulier nommé Charles Cantin. Cette application devra se reposer sur un Backend disponible et fonctionnel auquel nous confierons la gestion du contenu. Un CMS Headless fera parfaitement l'affaire.

Pour cette réalisation, il m'est demandé d'effectuer une petite étude UI/UX pour laquelle je partage la charte graphique, et les maquettes / wireframes (version mobile et desktop) mettant en valeur l'interface de la future vitrine web. Par ailleurs, les polices d'écritures employées devront être indiquées dans un document annexe.

Ce projet devra être disponible sur Github, incluant les documents annexes et le fichier Markdown, README.md, expliquant les procédures à effectuer pour installer le projet localement ainsi que son déploiement sur un serveur distant.

Un cahier des charges est fourni ainsi qu'un logo que je devrais exploiter au mieux.

Les compétences acquises au cours de cette partie de la formation m'amèneront à l'utilisation des technologies HTML, SCSS (pré-processeur SASS), JavaScript ES2015 et ReactJS.

Le cahier des charges et le déroulement de la réalisation seront exposés dans les chapitres suivants.

(environ 1188 caractères)

3. Cahier des charges, expression des besoins, ou spécifications fonctionnelles du projet

Mise en place du Projet

Vous retrouverez les documents et annexes dans le dossier 'projet' du Github associé :
- https://github.com/pantaflex44/studi-evals/tree/eval3/projet
  - Projet et cahier des charges : ECF-Entrainement-Photo-Front.pdf
  - Annexes : annexes.pdf
  - Dossier rendu : ECF-FRONT.docx

Guide d'installation et d'utilisation en local du projet :
- https://github.com/pantaflex44/studi-evals/blob/eval3/README.md

Étude et réflexions

Je commence cette évaluation par la lecture approfondie du dossier. Le cahier des charges en main (ECF-Entrainement-Photo-Front.pdf), j'analyse les demandes et le logo donné.

Je commence la réflexion sur la charte graphique et les couleurs du futur site. Malgré ma difficulté à l'utiliser, le logo fourni me donne quelques idées :
- les coins arrondis du logo
- la couleur grisée du logo (couleur froide)
- icône 'caméra' du logo

J'y reviendrai plus bas.

En parallèle, je recherche un système pour le Backend. Je découvre et choisi Prismic.io pour son plan Gratuit et ses Fonctionnalités. C'est un CMS Headless que je trouve parfaitement adapté à la situation, sous peu d'être suffisamment créatif pour imaginer une gestion ergonomique des données.

Le Backend

Dans un tout premier temps, je crée un compte sur Prismic.io puis, je crée les différents modèles de pages (types) :

- Accueil
- Galerie
- Prestations
- Contact

, etc.

Puis je créé chaque page au modèle correspondant. Je peuple des données avec celles du cahier des charges.

Toutes les photos / images, proviennent des sites Pixabay (https://pixabay.com/fr/) et Unsplash (https://unsplash.com/), banque d'images en ligne. Je les ai toutes passée par le service TinyPng (https://tinypng.com/) pour optimiser leur taille sans trop perdre en qualité. Dans la rubrique 'Médias' de la partie administration de Prismic, j'en profite pour modifier quelques informations sur chaque image, comme, le texte alternatif qui sera utilisé par la partie Frontend du site.

Pour finir, je pars à la découverte de l'API associée au service.

Le Frontend

Choix des polices d'écritures

- J'ai remarqué l'originalité de la police d'écriture du logo ce qui m'a donné l'idée de mixer une police sans-serif et une police calligraphique. Je choisis sur Google Fonts, la police calligraphique 'Dancing Script' et la police sans-serif 'Poppins'.

Choix des couleurs

- Le logo fourni en annexe tire vers le gris. Une couleur froide. Par réflexe, je me tourne vers le bleu. Pour réaliser la palette de couleur, je recherche en premier l'image principale de la page d'Accueil, dans les tons bleus, puis en extraits les teintes principales via le logiciel photo 'The Gimp'.

Choix du style

- Le logo fourni possède des coins arrondis. Je décide d'appliquer cet effet de style à différents éléments du design, comme les illustrations, cartes, du site.
- Je décide aussi de créer un design facilement adaptable entre mobiles et desktop pour simplifier le développement.
- Je reprends l'icône 'Caméra' du logo proposé, pour en faire un Favicon.

Pour résumer, ne sachant placer le logo, donné en annexe, tel quel, j'ai décidé d'en éclater le style et le répartir au sein du design globale du futur Frontend.

Recherche des technologies à employer

- Pour mettre en pratique mon apprentissage j'emploie le trio HTML, CSS et Javascript. Comme j'ai une affinité particulière avec ReactJs et même si j'en suis qu'à mes débuts, je décide de me lancer et de suggérer une version en utilisant cette technologie.

Déroulement du développement

Pour commencer, j'initialise un nouveau projet. Pour ce faire j'utilise un outil que j'ai préalablement développé, 'QuickParcelProject'. Sans rentrer dans les détails, cet outil permet de créer une base fonctionnelle pour développer ses projets en ReactJs, utiliser Sass, et même Jest pour concevoir ses tests (que je n'utiliserai pas pour cette évaluation). Cet outil utilise ParcelJs (https://parceljs.org/) permettant de compiler, transpiler et de proposer un serveur pour produire différents projets web. Pour plus d'informations sur cet outil : https://github.com/pantaflex44/QuickParcelProject.

Donc pour ce faire, voici la méthode employée (commandes Linux, à adapter à votre système d'exploitation) :

Méthode 1 : Git clone

```
mkdir eval3
cd eval3
git clone https://github.com/pantaflex44/QuickParcelProject.git .
rm -rf .git
```

Méthode 2 : Téléchargement

```
wget https://github.com/pantaflex44/QuickParcelProject/archive/refs/heads/main.zip
unzip -o main.zip
mv QuickParcelProject-main eval3
rm -rf main.zip
cd eval3
```

La base du projet est disponible.

Intallation des dépendances, comme indiqué dans le fichier README.md :

```
npm install
```

Une fois fait, je versionne ce nouveau projet en utilisant Git. Pour une question de propreté et d'organisation je ne crée pas un nouveau dépôt sur mon Github mais j'utilise le dépôt que j'ai créé pour toutes les évaluations de Studi ou chaque branche correspond à une évaluation. En revanche pour l'ECF final, je créerais un dépôt dédié.

```
git init
git remote add origin https://github.com/pantaflex44/studi-evals.git
git add .
git commit -m "Initialisation du dépot"
git branch --move master eval3
git push --set-upstream origin eval3
git push origin eval3
```

Je poursuis cette initialisation en installant quelques paquets supplémentaires, comme 'Hemmet' (gestion des Metastags), 'Lazy', 'CSSTransitions', etc.

Avant d'entamer le développement de l'interface, j'écris les lignes des fonctions devant communiquer avec l'API de prismic.io (src/js/prismatic.js), récupérer le routage des pages, etc. ainsi qu'un tout petit parser permettant de convertir le code d'un texte enrichi, renvoyé par Prismic.io, en code HTML directement utilisable par le navigateur web du visiteur ('prismicContentToHtml').

Je reprends le cahier des charges et débute par la conception de la page d'accueil. Il est demandé une image de fond qui doit prendre toute la page, ainsi que le titre et slogan du site. Me basant sur les couleurs définies plus haut (voir annexe) je trouve une illustration libre de droits. Je crée le composant de la page d'accueil avec toutes les informations demandées, majoritairement en provenance de l'API.

Une fois cette première page d'accueil réalisée, je m'emploie à écrire le code nécessaire charger l'application dans le navigateur web. J'ajoute le routage via 'react-router-dom' et les données de l'API. Le routage devient semi-dynamique. Les pages et leurs contenus proviennent de Prismic.io, les composants associés à chaque page, définis 'en dur' dans le code :

```
<Loader
    prismicRoutes={[
      { name: "page_daccueil", component: <Accueil /> },
      { name: "cgv", component: <Cgv /> },
      { name: "contact", component: <Contact /> },
      { name: "gallerie_de_photos", component: <Galerie /> },
      { name: "mentions_legales", component: <MentionsLegales />, },
      { name: "tarifs_et_prestations", component: <Prestations /> },
    ]}
/>
```

Je continue le développement par la galerie de photos. Cette galerie me permet d'utiliser les Hooks de ReactJs (useState, useEffect, etc), notamment pour l'affichage des photos en fonction d'une catégorie choisie et sans provoquer le rechargement de la page web :

```
const [albums, setAlbums] = useState([]);
const [albumsSelection, setAlbumsSelection] = useState([]);
const minMappedPhotosCount = process.env.MIN_MAPPED_PHOTOS_COUNT;

useEffect(() => {
    page.data.albums_photos.map((item) => {
      const albumPromised = getPagesById(item.album.id);
      albumPromised.then((album) => {
        setAlbums((albums) => [...albums, album]);
        setAlbumsSelection((albumsSelection) => [
          ...albumsSelection,
          album,
        ]);
      });
    });
}, []);

const handleCategorieChange = (e) => {
    e.preventDefault();

    if (!e.target.value || e.target.value === "") {
      setAlbumsSelection([...albums]);
    } else {
      const album = JSON.parse(e.target.value);

      setAlbumsSelection([album]);
    }
};
```

Je précise que les styles appliqués proviennent d'un fichier SASS / SCSS nommé '_app.scss'. Le fichier '_reset.scss' est importé par le fichier '_app.scss' et permet de remettre à zéro les styles CSS modifiés par un navigateur web. Base saine :

@use "./reset";

Pour suivre, je prends du temps pour essayer d'optimiser mon code ReactJS et implémenter la gestion des Metatags avec Helmet. React n'étant pas le copain du SEO, j'essaie de palier à ce problème en ajoutant des Metas par défaut dans le fichier index.html. Metas, que je supprime avant le chargement du Virtual Dom pour ne pas créer de doublons avec Helmet. Je teste avec des outils en ligne, j'obtiens un résultat pertinent. Bien que fonctionnelle, cette méthode ne me convient pas. Pour cela, je compte, a l'avenir, me diriger vers NextJs qui serait une solution efficace. J'ai essayé la méthode du 'pre-renderer' mais malheureusement, avec mon routage semi-dynamique, je n'ai pas réussi à la faire fonctionner.

cf : src/components/Metas.jsx

Le reste du développement comporte la création des pages restantes. La page Contact sera la plus compliquée, car elle intègre un service d'envoi de mail couplé à React, la gestion du formulaire, la validation des champs, le contrôle de son apparence, etc. Le formulaire est fonctionnel. J'en ai profité pour ajouter un Google reCaptcha pour protéger l'envoie de messages.

cf : src/pages/Contact.jsx
cf : src/components/ContactForm.jsx

Le service employé est EmailJs (https://www.emailjs.com/). Très complet, ce service me permet d'envoyer des emails en Javascript via un formulaire maison, un 'Template' et une configuration simple. Il peut fonctionner en Javascript Vanilla (service que j'utilise comme tel sur mon Portfolio personnel, https://pantaflex44.me) ou via un package React. C'est ce package ReactJs qui sera utilisé ici. Comme j'utilise ce service à des fins personnelles je ne vous transmets pas les identifiants publiquement. Toutefois, je pourrai le faire en privé.

4. Spécifications techniques du projet, élaborées par le candidat, y compris pour la sécurité et le web mobile

Maquettes réalisées avec Whimsical : https://whimsical.com/

Le projet est scindé en 2 parties :

- Backend :

Utilisation d'un CMS Headless renvoyant une API Rest ou GraphQl : Prismic.io (https://prismic.io/) et répondant aux besoins de simplicité d'utilisation, aussi bien du côté du développeur que de l'utilisateur final, de gratuité pour cet exercice, et de fonctionnalités suffisantes au travers d'une API Rest.

- Frontend

Cette partie, utilise plusieurs technologies, telles l'HTML 5, le Scss, ainsi que le Javascript Vanilla es6/es2015. À cela est couplé la technologie ReactJs permettant de concevoir une interface Web de manière modulaire et dynamique.

Pour gérer l'ensemble, transpiler, compiler et fournir un environnement de développement local, j'utilise QuickParcelProject réalisé par mes soins (https://github.com/pantaflex44/QuickParcelProject).

Le déploiement de l'application 'Portfolio' se fait en 2 étapes. Première étape est indiquée dans le README.md du Github associé au projet (https://github.com/pantaflex44/studi-evals/blob/eval3/README.md), il s'agit de la compilation, en local, du projet. La deuxième étape consiste à verser dans votre serveur préféré, via FTP par exemple, le code compilé se trouvant dans le dossier 'dist' du projet. Un fichier '.htaccess' est également ajouté automatiquement à cette compilation pour pouvoir utiliser un serveur Apache, permettant ainsi d'utiliser un simple serveur mutualisé aux tarifs raisonnables.

L'outil ReactJs est étoffé de quelques paquets supplémentaires, pour simplifier le développement :

- l'utilisation du CMS Prismic.io (https://prismic.io - https://www.npmjs.com/package/prismic.io)
- le service externe d'email EmailJs (https://www.emailjs.com - https://www.npmjs.com/package/@emailjs/browser)
- mais aussi Helmet Async (https://www.npmjs.com/package/react-helmet-async) pour la modification des entêtes HTML
- React Router Dom (https://www.npmjs.com/package/react-router-dom) pour la gestion du routage entre les pages
- React Icons (https://www.npmjs.com/package/react-icons) pour utiliser des icones SVG directement dans du code JSX
- React Transition Group (https://www.npmjs.com/package/react-transition-group) utilisé avec 'CSSTransition' pour apporter un animation au chargement des pages
- Spinners React (https://www.npmjs.com/package/spinners-react), l'animation en question
- etc.

Du côté de la sécurité, j'utilise les accès via Token pour les services de Prismic.io et EmailJs (tokens disponibles dans le fichier .env), ainsi que les sécurités intégrées à leurs services.

Le routage étant semi-dynamique, il est limité aux pages disponibles dans Prismic.io, provoquant une erreur 404 ou un rendu vierge (c'est à dire, non traité par l'application) en cas d'accès imprévus.

cf : src/App.jsx

Le formulaire de contact est protégé par un Captcha V2 de Google (reCaptcha v2 Checkbox) du plus classique. Un compte a été créé pour l'occasion le rendant parfaitement fonctionnel.

Outils de développement :

- VScode et ses extensions
- Google Chrome et Mozilla Firefox

5. Description de la veille, effectuée par le candidat durant le projet, sur les vulnérabilités de sécurité

La veille technologique nécessaire à la réalisation du projet fut complète.

Chaque service, externe ou librairie utilisée étant versionné dans le temps, m'a obligé à rechercher leurs capacités à s'assembler correctement.

J'ai dû aussi vérifier leur obsolescence pour ne pas intégrer du code trop ancien, ajoutant des failles de sécurité supplémentaires. Par exemple le routeur DOM de React, récemment passé à la version 6 apportant par la même quelques changements importants et peu documentés sur Internet.
React ayant une très grande communauté, beaucoup de ressources sont disponibles sur le Web. Le danger est de trouver des ressources trop anciennement ou devenues obsolètes du jour au lendemain. React Router Dom en est le parfait exemple à l'heure ou j'écris ces lignes.

De la même manière, EmailJS propose un nombre d'exemples d'utilisations non négligeable, et pas forcément fonctionnels en toute circonstance, poussant le développeur à approfondir avec grand sérieux le sujet. Finalement chaque exemple devient très simple d'intégration une fois le bon trouvé.

Les technologies web évoluant à une vitesse extraordinaire, leur nombre donnant le tournis et rendant l'apprentissage de notre futur métier extrêmement compliqué (par ailleurs, de ma qualité de futur Junior, je trouve que les offres d'emplois deviennent parfois lunaires à ce propos...), la veille technologique est pour moi une part très importante et totalement nécessaire.

6. Description d'une situation de travail ayant nécessité une recherche, effectuée par le candidat durant le projet, à partir de site anglophone

Pour réaliser le projet, j'ai dû utiliser plusieurs services et paquets spécifiques, la plupart en anglais.

Prismic.io, EmailJs, Google reCaptcha sont des produits / services internationaux, anglophones.
Le gestionnaire de paquet NPM me fournissant les fonctionnalités indispensables à la réalisation des scripts React tels que je le souhaitais, est aussi intégralement en anglais.

Beaucoup de technologies, quelles soient d'origine Française ou non, utilisent la langue Anglaise pour se diffuser sur Internet. Langue internationale, comprise par le plus grand nombre. Donc nous faisons face à l'obligation de rechercher, comprendre et analyser des besoins, outils et services dans cette langue pour mener à bien nos projets.

7. Extrait du site anglophone, utilisé dans le cadre de la recherche décrite précédemment, accompagné de la traduction en français effectuée par le candidat sans traducteur automatique (environ 750 signes).

Pour sécurisér mon formulaire de contact, j'ai utilisé Google reCaptcha : https://www.google.com/recaptcha/about/

Pour ce faire, j'ai utilisé la documentation disponible ici : https://developers.google.com/recaptcha/docs/display

Extrait de la documentation en anglais:

reCAPTCHA v2
This page explains how to display and customize the reCAPTCHA v2 widget on your webpage.
To display the widget, you can either:

Automatically render the widget or
Explicitly render the widget

See Configurations to learn how to customize your widget. For example, you may want to specify the language or theme for the widget.

See Verifying the user's response to check if the user successfully solved the CAPTCHA.

Automatically render the reCAPTCHA widget
The easiest method for rendering the reCAPTCHA widget on your page is to include the necessary JavaScript resource and a g-recaptcha tag. The g-recaptcha tag is a DIV element with class name g-recaptcha and your site key in the data-sitekey attribute:

```
<html>
 <head>
  <title>reCAPTCHA demo: Simple page</title>
  <script src="https://www.google.com/recaptcha/api.js" async defer></script>
 </head>
 <body>
  <form action="?" method="POST">
   <div class="g-recaptcha" data-sitekey="your_site_key"></div>
   <br/>
   <input type="submit" value="Submit">
  </form>
 </body>
</html>
```

The script must be loaded using the HTTPS protocol and can be included from any point on the page without restriction.

Traduction :

reCAPTCHA v2
Cette page explique comment utiliser et personnaliser l'outil reCAPTCHA v2 sur vos pages web.
Pour afficher reCAPTCHA v2 sur votre page, vous pouvez utiliser les méthodes suivantes :

Rendu automatique ou choisi de l'élément HTML reCAPTCHA v2.
Se référer à la configuration pour personnaliser reCAPTCHA v2. Par exemple, vous pouvez modifier la langue et le thème utilisé par reCAPTCHA v2.

Surveillez l'événement de vérification pour connaitre l'état de validation du reCAPTCHA v2 présenté à l'utilisateur.

Rendu automatique du reCAPTCHA v2.

La méthode la plus simple pour rendre reCAPTCHA v2 sur votre page est d'inclure les fichiers JavaScript et le jeton g-recaptcha. Le jeton g-recaptcha est un élément HTML DIV ayant pour classe stylistique g-recaptcha et la clef du site indiquée dans la valeur de l'attribut data-sitekey :

```
<html>
 <head>
  <title>reCAPTCHA demo: Simple page</title>
  <script src="https://www.google.com/recaptcha/api.js" async defer></script>
 </head>
 <body>
  <form action="?" method="POST">
```

```
    <div class="g-recaptcha" data-sitekey="your_site_key"></div>
    <br/>
    <input type="submit" value="Submit">
   </form>
 </body>
</html>
```

Ce code doit être chargé en utilisant le protocole sécurisé HTTPS, et doit pouvoir être inclus dans n'importe quelle page web sans aucune restriction.

8. Autres ressources

Github (Frontend): https://github.com/pantaflex44/studi-evals/tree/eval3

Prismic.io (Backend): https://prismic.io:
- Nom: Cantin
- Prénom: Charles
- Email: pantaflex@hotmail.fr
- Mdp: 4H53qMXPRJtu#_q

Site démo: https://pantaflex44.me/eval3/

9. Informations complémentaires

J'ai trouvé cette évaluation très intéressante. Elle m'a permis de mettre en pratique mon apprentissage de ReactJs, faire face aux difficultés liées à son utilisation.

Respecter un cahier des charges, travailler sur un dossier Projet, mener à bien cet exercice a été un super entrainement.

J'ai passé un temps plutôt long pour réaliser ce projet. Je me suis pris au jeu de la recherche d'informations, lecture de documentations, recherche de technologies et approfondissements des méthodes.

J'en ai profité pour compléter ma ToDo List des Technos à découvrir dans le futur, à pousser mon apprentissage de ReactJs que j'aime beaucoup, en le couplant à NextJs par exemple.
J'ai découvert l'utilisation fort pratique d'un CMS Headless pour des projets de cette envergure.

Cette évaluation m'a aussi fait découvrir la charge de travail nécessaire pour réaliser ce genre de dossier. Certes, cette charge a été plus importante à cause de mon ignorance en certaines technologies, mais reflète très bien, je trouve, le parcours à suivre.

Correction

Bonjour ,

Merci pour votre réalisation.
Le résultat est très bon. Le site est en ligne, fonctionnel, et il est responsive : très bien.
Les liens fonctionnent.
Le site est très bien développé. Esthétiquement, c'est très réussi, c'est épuré, bien organisé.
Les nombreuses animations graphiques améliorent grandement l'expérience de navigation du visiteur.
La présentation est très qualitative. Bravo.

Par contre, attention à bien suivre les instructions de l'énoncé …

Mise en ligne : OK

Dans votre dépôt Github, devra aussi se trouver :
-Un fichier readme.md contenant la démarche à suivre pour le déploiement en local ou en ligne. *OK*
-Une charte graphique au format .pdf regroupant :
o La palette de couleurs *OK*
o La palette des polices d'écriture choisies *OK*
o L'export des wireframes et mockups des pages (en version mobile et desktop) *OK*

GitHub : bonne utilisation. commits réguliers, avec commentaires.
Readme.md (déploiement en local): OK

Site Responsive : fonctionnel, le redimensionnement reste ergonomique.

4 pages (accueil, Galerie, Tarifs, Contact): OK

Barre de navigation + icônes + logo : intégrés au menu déroulant.
"Depuis chacune des pages, une barre de navigation devra être disponible avec le menu ainsi que les icônes Facebook et Instagram (pour un lien vers son profil sur les réseaux sociaux à l'avenir)."

Description du projet : très bien. Pour votre dossier professionnel, il faudra condenser la description et se concentrer sur l'argumentation des choix techniques et technologiques.

"Tout contenu devra être modifiable facilement par Charles à l'aide d'un CMS."
Le site est-il modifiable ? à détailler.

Vous avez fait une très bonne réalisation, le site est bien pensé, visuellement agréable, et fluide. C'est très bien.

Vous vous êtes un peu arrangé avec l'énoncé, mais cela a permis de gagner en esthétique et en fluidité de navigation.
La page "galerie" permet bien de faire la sélection par famille, sans recharger la page. ..
Très bIen

1. Concevoir la maquette des interfaces ciblées (5 points) 5
2. Obtenir un rendu visuel optimisé et adapté à tout équipement (5 points) 5
3. Augmenter le dynamisme de l'expérience utilisateur grâce aux scripts événementiels. (3 points) 3
4. Exploiter un système de gestion de contenu existant (4 points) 4
5. Publier l'application sur un serveur web (3 points) 3