

## ▀ ▀ ▀ 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 - Photogtaphe.

Lien Github du projet : https://github.com/pantaflex44/studi-evals/tree/eval3

Lien Drive du projet (si nécessaire) : X

**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

1. 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 de 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 chartre 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 fournit ainsi qu’un logo que je devrai exploiter au mieux.

Les compétences acquises au cours de cette partie de la formation m’ameneront à l’utilisation des technologies HTML, CSS (pré-processeur SASS), Javascript ES2015 et ReactJS.

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

1. 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' sur le 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 : Modele-de-copie-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 fournit 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.

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.

1. Spécifications techniques du projet, élaborées par le candidat, y compris pour la sécurité et le web mobile
2. Description de la veille, effectuée par le candidat durant le projet, sur les vulnérabilités de sécurité
3. Description d’une situation de travail ayant nécessité une recherche, effectuée par le candidat durant le projet, à partir de site anglophone
4. 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).
5. Autres ressources
6. Informations complémentaires