PROJET DE FIN D'ANNÉE

WEBSTART - DEV 2 - 2022 - 2023

Objectifs

Le but du projet est de créer un système de type SaaS qui permet de proposer à des clients des services mensuels et/ou annuels sous forme d'abonnement.

Votre objectif est donc de créer et de mettre en production :

- une application mobile hybride de type SPA (Single Page Application) et PWA (Progressive Web Apps), basée sur Javascript avec Ionic et Vue/React (au choix), à destination de l'utilisateur final
- un site internet full stack basé sur PHP (Laravel) qui permettra d'administrer le projet
- une API HTTP relié à l'application mobile et basée sur Node (Express.js)

Le projet sera donc composé de 3 mini-projets qui communiquent ensemble via une base de données commune.

Vous êtes totalement libre du sujet ainsi que de l'interface de vos livrables tant qu'ils correspondent aux fonctionnalités et contraintes indiquées dans le présent cahier des charges.

Application mobile

L'application mobile permettra de présenter vos services. L'utilisateur pourra s'inscrire, s'abonner ou se connecter et pourra gérer son compte directement sur l'application mobile.

Site internet

Vous disposerez d'un dashboard qui permettra de gérer le site (actualités et gestion des utilisateurs et de leurs abonnements).

API

Vous devez créer une API permettant d'exposer les données qui permettront aux utilisateurs d'utiliser vos services sur l'application mobile.

Compétences à acquérir

Site internet full stack (Laravel)

- Authentification de type session (Laravel Breeze)
- MVC
- PHP moderne + Composer avec le framework Laravel
- CRUD
- Programmation objet
- Utilisation d'un routeur / middlewares / système de template (Blade)
- Base de données via un ORM (Eloquent)
- Validation de formulaires / Protection CSRF
- Responsive et CSS via un framework (Tailwind, Bootstrap…)

Application mobile hybride (Ionic + Vue/React)

- MVVM (composants)
- Single Page Application (SPA)
- Javascript moderne + NPM avec le framework Vue ou React
- Application mobile hybride avec le framework Ionic
- Manipulation du DOM
- Promesses / asynchronicité avec Fetch
- Utilisation du Localstorage
- Utilisation d'un store avec Pina
- Utilisation d'un routeur client-side via Vue Router
- Progressive Web App (PWA)

API (Node / Express.js)

- Authentification token (JWT)
- Node + NPM avec le framework Express.js
- SQL/NoSQL via un ORM (Sequelize / Prisma)
- API REST
- CRUD

Spécifications techniques

Généralités

- Le workflow Github (historique, branches, commits ...) doit être propre et détaillé afin de permettre la compréhension de la méthodologie et du process de développement
- Vous ne pouvez pas utiliser de système “tout fait” possédant déjà les fonctionnalités demandées.
- Vous pouvez utiliser des packages spécifiques si besoin comme http://image.intervention.io/ pour gérer la création de plusieurs formats d’images. Vous devrez justifier l'utilisation de tout package ou librairie utilisé et la faire valider par votre professeur.
- Vous pouvez ajouter toutes les fonctionnalités bonus que vous voulez. Par exemple, vous pouvez améliorer l’UX du site internet et permettre à l’administrateur de filter et/ou trier les abonnements selon les critères de votre choix.

Site internet

- Le développement du site internet devra être effectué intégralement à l'aide de Laravel (dernière version en date). Vous pouvez utiliser le framework Vue.js pour améliorer l'interface du site internet, voire même obtenir une Single Page Application.
- Vous devrez fournir une version de production du site internet : tous les fichiers js et css doivent être compilés et minifiés via l'outil Laravel Mix (utilisation de Vite possible).
- Vous pouvez utiliser Laravel Permissions pour gérer un rôle de “Super Admin” dans votre application web Laravel
- Vous utiliserez le service tiers Cloudinary pour le stockage de vos images/fichiers... vu que le déploiement sera sur Render.com qui est un système volatile. Vous utiliserez Cloudinary Laravel pour l’upload d’images dans Cloudinary via votre projet Laravel.

Application mobile PWA

- Le développement de l’application mobile devra être effectué intégralement à l’aide de Vue.js ou de React et du framework Ionic (dernières versions en date).
- Vous devrez fournir une version de production de l’application mobile via le système de Vite. Cette version de production devra être accessible via un navigateur.

API

- Le développement de l’API devra être effectué avec Node.js.
- Vous devrez utiliser un ORM pour accéder à la base de données (au choix, selon le type de base de données choisi)

Cahier des charges

Site internet (dashboard)

Le site doit être administrable via un back-office sécurisé par un administrateur du site authentifié (système d'authentification de Laravel, table "users") qui a un rôle de "super admin".

Accueil

La page d'accueil du site internet comportera le formulaire de connexion pour accéder au dashboard.

Note : l'inscription d'un utilisateur sera bloquée. On ne peut pas accéder à un formulaire d'inscription.

Utilisateurs

Le dashboard doit permettre de gérer les utilisateurs.
L'administrateur peut lister les utilisateurs avec une pagination, afficher et modifier les informations d'un utilisateur, supprimer un utilisateur, et lister ses paiements liés à son abonnement.

Abonnements

Vous devrez lister avec une pagination les abonnements souscrits par les utilisateurs.

Les abonnements doivent être rattachés à leur utilisateur respectif.

Actualités

Le dashboard permet aussi de gérer les actualités.

L'administrateur pourra lister les actualités avec une pagination, créer, modifier et supprimer une actualité.

- Titre (obligatoire)
- Image (obligatoire) (à uploader via un service tiers comme Cloudinary)
- Date de publication (obligatoire)
- Contenu avec un éditeur WYSIWYG (obligatoire)
- Actif ou non (activé par défaut) pour gérer des brouillons
- Auteur (administrateur connecté)

Une actualité non publiée n'apparaît pas sur le site, mais reste administrable via le back-office.

Application mobile PWA

Général

L'application mobile contiendra un menu de navigation qui permet d'accéder aux pages listées ci-dessous.

Page d'accueil

La page d'accueil doit présenter vos services.

Page d'actualités (+ page d'actualité)

Vous devrez créer une page listant les actualités avec une pagination. L'utilisateur pourra afficher une actualité sur une page dédiée avec son url de type (/actus/le-titre-de-l-actualite).

Page de contact

La page contact reprends les éléments du site internet et contient les éléments suivants :

- Coordonnées complètes (adresse de la boutique, téléphone, email…).
- Une intégration Google Map (centrée sur l'adresse de Webstart à Paris : 19 rue Yves Toudic 75010 Paris)
- Un formulaire de contact (tous les champs obligatoires)
    - Nom/prénom
    - Email (valide)
    - Objet
    - Message (minimum 50 caractères)

Le formulaire de contact utilisera votre API pour envoyer un email via Mailtrap.

Le formulaire de contact enverra un email sur votre compte Mailtrap en utilisant l'adresse email du formulaire comme email d'expéditeur, le nom et prénom du formulaire comme nom de l'expéditeur, l'objet du formulaire comme objet du mail, le message du formulaire comme message de l'email.

Vous devrez valider côté client (en plus de côté backend) les champs du formulaire avec Vuelidate.

Utilisez les coordonnées de Webstart pour votre entreprise.

Pages d’inscription/connexion

L’utilisateur pourra s’inscrire et se connecter à son compte. L'authentification se fera en utilisant un “token authentication” via les tokens JWT.

Vous devrez valider côté client (en plus de côté backend) les champs du formulaire avec Vuelidate.

Page d’abonnement

La page abonnement doit permettre à l’utilisateur de s’abonner à un de vos services une fois inscrit.

L’utilisateur pourra souscrire à un abonnement via le système de paiement Stripe.

L’utilisateur pourra entrer un code promo pour obtenir une réduction (WEBSTART10 par exemple).

Vous n’avez pas à gérer l’application ou non de la TVA sur le prix, ou les différents montants de TVA selon le produit. On considère que tous les prix sont TTC et que la TVA Intracommunautaire française de 20% est tout le temps incluse.

Une fois que l’utilisateur a souscrit à l’abonnement de son choix via Stripe, une page de confirmation reprenant les informations de l’abonnement doit être affichée :

- Montant de l’abonnement
- Date de l’abonnement
- 4 derniers chiffres de la carte bancaire utilisée
- Date du renouvellement

Vous devez envoyer un email à l’acheteur et à l’administrateur pour les notifier qu’un abonnement a été souscrit sur le site internet.

Page compte utilisateur

L’utilisateur pourra gérer son abonnement via Stripe Billing Portal et ses informations sur l’application mobile.

L’utilisateur peut modifier ses informations comme son adresse e-mail, son nom, son mot de passe...

L’utilisateur peut annuler / mettre en pause / reprendre / modifier son abonnement.

Page légales

Vous devrez créer les pages de :

- Mentions légales
- Conditions Générales de Vente et d'Utilisation
- Politique de Confidentialité

Vous afficherez un texte générique ("lorem ipsum") sur les pages CGV/CGU et Politique de Confidentialité.

Vous remplirez par contre correctement les mentions légales de votre site en vous aidant du site https://www.economie.gouv.fr/entreprises/site-internet-mentions-obligatoires

PWA

Vous devez utiliser une fonctionnalité d'une Progressive Web App (PWA) qui consiste à donner un aspect d'application mobile à votre site mobile, c'est à dire ajouter un "manifest" pour que l'utilisateur puisse ajouter l'application sur son terminal mobile

Vous devez donc créer toutes les icônes nécessaires pour les terminaux Windows, Android et Apple.

API

Vous devez créer une API permettant d'exposer les fonctionnalités de votre site internet à l'application mobile.

Vous devrez respecter la spécification qui se trouve à l'adresse :
https://cba85.stoplight.io/docs/laravel-saas-api/reference/API.yaml

Base de données

Vous devez utiliser une seule base de données unifiée entre l'application mobile, l'API et le site internet pour votre projet, de type MySQL, PostgreSQL, SQL Lite ou MongoDB.

Vous devez gérer le schéma de votre base de données via des migrations Laravel, ou via le système de migration de votre ORM si vous préférez utiliser Node pour cela..

Vous devez créer des seeders Laravel (ou Node) pour peupler votre base de données avec de données de test afin de permettre de tester votre projet.

Vous êtes libre sur le schéma de votre base de données.

Emails

Vous devrez générer des emails qui seront envoyés localement et en production via Mailtrap.

- Email de bienvenue à la création d'un compte
- Email de confirmation d'abonnement
- Email de confirmation d'annulation d'abonnement
- Email de réinitialisation de mot de passe

Interfaces

Aucune identité / charte graphique n'est exigée. Vous êtes libre de votre interface. Vous devrez néanmoins créer une interface correcte au niveau de l'UI/UX. Le site internet doit être responsive pour les trois formats desktop, tablette et mobile.

Accessibilité

Votre application web et mobile doit être accessible et respecter les bonnes pratiques d'accessibilité.

Un site Internet accessible, c'est garantir que tous vos publics pourront facilement visualiser et comprendre les informations diffusées.

Vous devrez tester l'accessibilité de vos applications web et mobile en utilisant Lighthouse (catégorie "Accessibilité").

Inclusivité

Le design inclusif désigne un processus de création qui vise à rendre accessible le produit ou le service à un maximum de personnes possible, quelles que soient leurs caractéristiques physiques, mentales ou sociales, que ce soit par le biais des images ou du texte.

Vous devez faire en sorte d'avoir un design inclusif. La notation du projet tiendra compte de cela.

SEO

Vous devrez respecter les bonnes pratiques SEO (Search Engine Optimisation) On Page.

Vous devrez tester le référencement de vos applications web et mobile en utilisant Lighthouse (catégorie "SEO").

Eco responsabilité

Vous devrez faire en sorte de créer une application web et une application mobile éco-responsable.

Vous testerez l'empreinte carbone de votre site internet via Website Carbon.

Bonus

Le cahier des charges est un document habituellement produit par le client (ou co-écrit avec lui) pour exprimer ses besoins de la manière la plus précise possible. Lors de la recette, le client vérifie si le produit livré remplit bien les attentes listées dans ce document.

Vous ne pouvez ajouter des fonctionnalités en plus de celles demandées dans le cahier des charges qu'une fois toutes celles-ci effectuées.

Une option est un bonus en plus de ce qui est demandé. Ne commencez pas le développement d'une option sans avoir terminé ce qui est demandé dans le cahier des charges.

Mise en production

Vous avez le choix de mettre en production sur le serveur de choix.
Vous pouvez utiliser des services gratuits comme :

- Firebase / Netlify / Vercel pour déployer votre application mobile
- Render.com pour déployer votre application mobile, votre site internet et/ou votre API

Stockage de fichiers médias (images, vidéos)

Vous pouvez utiliser le service que vous préférez.

- Cloudinary (gratuit, 10 Go) : vous pouvez utiliser le package Cloudinary Laravel pour l'upload d'images dans Cloudinary via votre projet Laravel.
- Cloudflare R2 (gratuit, 10 Go)

Dossier de présentation du projet

Votre dossier de présentation devra contenir :

- Présentation de votre projet
- Captures d'écran de l'interface de votre application mobile et de votre site internet
- Présentation de vos choix techniques

- Rapport de performance : notes de Lighthouse (SEO, accessibilité, performances, bonnes pratiques, progressive web apps pour l'application mobile hybride)
- Eco Responsabilité : note (pourcentage) de l'empreinte carbon de votre site sur Website Carbon

Présentation orale

1. Présentation de votre projet
2. Justification de vos choix techniques
3. Démo de votre application mobile et du dashboard sur le site internet
4. Rapport de performance / Eco-responsabilité

Planning

- ? : Jury blanc
    - Prototype de l'application mobile
    - Prototype du site internet
    - Début de fonctionnalité de l'API / Application mobile / Site internet
- ? : Jury final

Livrables

Vous devrez fournir les URLs suivantes :

- ☐ Lien du dépôt Github contenant le code source du site internet
- ☐ Lien du dépôt Github contenant le code source de l'API
- ☐ Lien du dépôt Github contenant le code source de l'application mobile hybride
- ☐ Lien de mise en production du site internet
- ☐ Lien de mise en production de l'API
- ☐ Lien de mise en production de l'application mobile hybride
- ☐ Dossier de présentation PDF

Application mobile

- ☐ Page d'accueil
- ☐ Page d'actualités
- ☐ Page d'une actualité
- ☐ Page d'inscription
- ☐ Page de connexion
- ☐ Page de compte utilisateur

- ☐ Page des offres
- ☐ Page de checkout (récapitulatif de commande) (accessible uniquement aux utilisateurs inscrits)
- ☐ Page de succès de paiement
- ☐ Page d'annulation de paiement
- ☐ Page d'accès au service (accessible uniquement aux abonnés)
- ☐ Page de formulaire de contact
- ☐ Création d'une clé API pour Google Maps
- ☐ Page de mentions légales
- ☐ Page de CGU/CGV
- ☐ Page de politique de confidentialité

Site internet

- ☐ Supprimer page d'inscription (de Laravel Breeze)
- ☐ Page d'accueil (page de connexion à remplacer)
- ☐ Page d'accueil de tableau de bord
- ☑ ~~Page de gestion des utilisateurs (CRUD)~~
- ☑ ~~Page de gestion des abonnements~~
- ☑ ~~Page de gestion des actualités (CRUD)~~

API

- ☐ JWT Token Auth
- ☐ CRUD User
- ☐ CRUD Actualités (Post)

Critères d'évaluation

Votre travail sera évalué selon les critères suivants :

- **Fonctionnel** : les fonctionnalités ont été implémentées conformément au cahier des charges fourni ci-dessus.
- **Lisibilité** : le ou les dépôts git contiennent un fichier README.md qui explique comment cloner, faire fonctionner et tester l'API, le site internet et l'application mobile depuis une autre machine que la sienne. Le code source doit respecter les conventions de codage standards. Vous devez respecter les bonnes pratiques en termes de développement Laravel, PHP, Vue, React, Node, Ionic…
- **Production** : l'API, le site internet et l'application mobile doivent être accessibles et fonctionnels en production, sur les URLs fournies.