NOM	CALVES
Prénom	Pierre
Date de naissance	13/07/1978

Copie à rendre

Graduate Développeur

(Android, Angular, Flutter, Front End, Full Stack, IOS, PHP/Symfony)

Lien GITHUB du projet : https://github.com/bobo13778/garageVP

Lien vers la version en ligne : https://garagevparrot.go.yj.fr/

Lien vers le logiciel de gestion du projet :
https://trello.com/b/KbsMktli/mod%C3%A8le-kanban

Lien FIGMA pour les wireframes :
https://www.figma.com/file/1QYxYF7a9FWcsU2qppCkwA/GARAGE_V_PARROT_Pierre_CALVES?type=design&node-id=0-1&mode=design&t=aVjgVlMjGIBvUr8k-0

Déroulement du projet :
Pour mener à bien ce projet j'ai commencé par créer un tableau Kanban pour lister les tâches à réaliser.

Ensuite j'ai pris soin d'analyser le besoin du client en détail, en listant les entités à créer en base de données et les fonctionnalités à développer.

J'ai ensuite réalisé une charte graphique et un logo pour le futur site, détaillée dans le fichier charte_graphique.pdf, disponible sur le dépôt GITHUB dans le dossier ''Livrables''.

J'ai réalisé le diagramme de cas d'utilisation du site à l'aide du logiciel diaw, ce diagramme est disponible dans le fichier documentation_technique.pdf disponible sur le dépôt GITHUB dans le dossier ''Livrables''..

J'ai ensuite réalisé le diagramme de séquence demandé toujours avec diaw, il est également disponible dans le fichier documentation_technique.pdf.

J'ai enfin réalisé le diagramme de classes correspondant aux entités à créer en base de données avec leurs caractéristiques, également disponibles sur le fichier documentation_technique.pdf.

Activité – Type 1 : Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité

Une fois mon travail préparatoire réalisé j'ai commencé à concevoir les pages du site sur mon compte figma.com.

J'ai réalisé en premier les modèles wireframe sur mobile afin de penser le site en ''mobile first''.
J'ai commencé par les pages destinées aux visiteurs (Accueil, page présentant les occasions, page présentant une occasion en particulier et page de contact).

J'ai ensuite réalisé les pages destinées aux employés (connexion, ajout/suppression de témoignages, ajout, modification ou suppression d'annonces).

J'ai enfin réalisé les pages pour l'administrateur, la page de connexion étant commune à celle des employés (ajout, modification de fiches d'employés, modification des horaires, modification des services).

J'ai enfin adapté ces pages en version desktop.

Ces wireframes sont disponibles sur le lien Figma indiqué en première page, ainsi que sur le document charte_graphique.pdf.

J'ai ensuite commencé par initialiser un nouveau projet php symfony, sans utiliser l'ORM doctrine afin de respecter la consigne d'interagir avec la base de donnée en SQL.

Le projet est pensé en MVC, j'ai donc créé créé les contrôleurs permettant l'affichage des vues.

J'ai commencé par créer un template de base qui sera affiché sur toutes les vues grâce au moteur de template TWIG.
Ce template comprends un header avec la navigation et un footer avec notamment les horaires.

J'ai ensuite créé les vues du site en m'aidant de Bootstrap afin de rendre le site responsive.

Une fois les accès en base de données créés et la gestion de la connexion/déconnection effectuée en back-end j'ai pu modifier les affichages de données et créer les formulaires nécessaires à la création, modification et suppression des éléments :

- Création d'un témoignage par un visiteur non connecté à valider par un employé.

- Création d'un message de contact par un visiteur avec une référence à l'annonce si la demande est faite à partir de cette dernière.

- Création, modification et suppression d'un employé par l'administrateur
- Création, modification et suppression d'un service par l'administrateur
- Modification des horaires par l'administrateur

- Création, modification d'une annonce par l'administrateur ou l'employé
- Création, validation ou suppression d'un témoignage par l'administrateur ou l'employé

Afin de limiter les pages d'administration j'ai utilisé javascript pour afficher les différents blocks de chaque catégorie dans la même page.

J'ai aussi modifié les menus de navigation en fonction du profil connecté (ou du simple visiteur)

Je n'ai malheureusement pas réussi par manque de temps à filtrer les annonces, les filtres sont présents mais pas fonctionnels.
Je n'ai pas réussi non plus pour la même raison à afficher les notes des témoignages sous forme d'étoiles.

Activité – Type 2 : Développer la partie back-end d'une application web ou web mobile en intégrant les recommandations de sécurité

J'ai initialisé la base de données en local avec phpMyadmin de Xampp au moyen des scripts disponibles sur le dépôt GIT dans le dossier ''Livrables''.

J'ai ensuite créé les modeles correspondants aux tables de la base de données dans mon projet symfony, je n'ai pas utilisé d'ORM pour respecter la consigne de rédiger les instructions de base de données en SQL.

Afin de créer les liens entre les modèles et la base de données, j'ai créé un modèle abstrait, étendu par les autres, permettant d'automatiser les appels en BDD, j'ai ensuite réalisé un CRUD général pour tous les modèles.

Afin de pouvoir afficher les pages j'ai ensuite utilisé un routeur en annotations.

J'ai ensuite entrepris de gérer les connexions administrateur et employés, j'ai pour cela alimenté la BDD avec les informations administrateurs en utilisant bcrypt.fr afin de stocker un mot de passe hashé et non en clair.

Pour utiliser le site en ligne les identifiants sont les suivants :

Email admin : vparrot@test.fr

Password : V1100P4rr0T

Email employé 1 : jean@michel.fr
Password : jean@michel.frpassword

Email employé 2 : jeanne@masse.fr
Password : jeanne@masse.frpassword

Email employé 3 : jacques@dupont.frpassword
Password : jacques@dupont.frpassword

J'ai ensuite mis en place le traitement des formulaires de login et de logout avec la variable $_SESSION et j'ai limité les accès à la page employé et à la page administrateur selon le profil connecté.

J'ai enfin modifié les contrôleurs pour traiter les données en provenance des formulaires des vues en GET ou en POST en faisant appel à mon modèle générique comportant le CRUD.

J'ai effectué les tests de fonctionnalités en local, après la résolution de quelques bugs finaux tout semble fonctionner.

Avant de mettre le site en ligne j'ai complété le fichier de remplissage de la base de données avec quelques valeurs.

J'ai créé un fichier readme.md pour indiquer la marche à suivre afin de tester l'exécution en local du site, il est disponible sur le dépôt github dans ''livrables''

J'ai enfin créé une branche GIT production avant de modifier les identifiants de connexion à la base de donnée et j'ai transféré les fichiers via le FTP Filezilla sur mon compte PlanetHoster.

Le site est en ligne et il me semble fonctionner, je n'ai malheureusement pas pu finaliser les tests et la mise en page faute de temps.