| | |
|---|---|
| **NOM** | **MAKAROV** |
| **Prénom** | **Dmitri** |
| **Date de naissance** | **21 décembre 1980** |

# Copie à rendre

# TP Développeur Web et Web Mobile

## Documents à compléter et à rendre

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

- Figma
- HTML 5
- CSS 3
- Bootstrap 5
- JavaScript

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

- PHP 8.2 sous PDO
- MySQL Community
- AWS S3 SDK
- DBML (Database Markup Language)
- Plant UML

Le lien du dépôt Github public: https://github.com/Dima-McArrow/ECF_LE_PROJET_Garage_automobile.git

Le lien de la version en ligne de l'application web:

- Le site web Garage V. Parrot: https://makarovdimitri.shop/
- Le site (l'application) web de la gestion du contenu (CMS) du site Garage V. Parrot: https://cryptic-island-77465-7670ca5f9230.herokuapp.com/

Pour accéder au CMS (login) il est possible d'utiliser le compte admin:

- Le e-mail: vincent.parrot@mail.com
- Le mot de passe: pa$$word

Le CMS offre, entre autres, la possibilité de créer un utilisateur admin ou employé, si la connexion est effectuée avec un compte admin (Vincent Parrot est admin).

Le lien vers le logiciel de gestion de projet (Trello):

- https://trello.com/b/saHxK2OJ/garage-v-parrot

Le lien Figma du projet:

- https://www.figma.com/file/C9Q6iNw5DtFr0hKYcW6ydy/Garage-V.-Parrot?type=design&mode=design&t=C2lavdUo7zEaJIJv-1

Les prototypes Figma:

- https://www.figma.com/proto/C9Q6iNw5DtFr0hKYcW6ydy/Garage-V.-Parrot?type=design&t=sZHfQb1x4mXgi12E-1&scaling=min-zoom&page-id=0%3A1&node-id=3-27&starting-point-node-id=3%3A27&show-proto-sidebar=1&mode=design

La liste des documents:

- Dossier “Docs” du repositoire GitHub :
  - Dossier racine :
    - README.md
  - Dossier “PDF” :
    - Charte graphique.pdf
    - Documentation_technique.pdf
  - Dossier “diagrammes” :
    - Diagramme des cas d’utilisation
    - Diagrammes de séquence (3 diagrammes dans un fichier)
    - Diagramme de classe
  - Dossier “maquettes” :
    - Dossier contenant les maquettes mobile
    - Dossier contenant les maquettes desktop

Mes réflexions initiales sur le projet m'ont conduit à des conclusions suivantes :

- La création d'un site web présentant le garage et ces services d'une manière visuellement agréable at claire.
- La création d'un autre site web, qui va faire office d'un CMS (Content Management System) - une application qui va permettre de gérer le contenu du site principal.

Comme il n'y avait pas d'indication précise comment le système doit être organisé, je me suis oriente sur ma propre vision. Ce qui était demandé, c'est que le propriétaire du garage ainsi que ces employés ont la possibilité de "se connecter" et gérer le contenu du site. L'admin doit avoir la main sur tout le contenu dynamique du site : horaire d'ouverture, les services, les voitures d'occasion, les messages et les commentaires des clients. Les employés doivent avoir accès à la gestion des messages et commentaire ainsi que la gestion des voitures. Ma vision personnelle me dit que "exposer" la possibilité de se connecter en tant que gestionnaire du site sur le site même n'est pas une bonne idée au niveau esthétique et au niveau de sécurité. Donc, j'ai décidé de créer deux sites distincts. Le site principal du garage, qui va respecter une charte graphique et en général être orienté client. D'autre part, un site, qui va représenter une application web qui va être maximalement simpliste visuellement et en même temps efficace et être en général être orienté gestion.

Aussi, d'une manière générale, il peut exister plusieurs approches en ce qui concerne le déploiement des deux applications. Le site web de toute manière doit être hébergé pour être accessible par le plus grand nombre de visiteurs (il est possible de configurer un serveur sur un PC directement dans les bureaux du garage, mais dans le cadre de ce projet, je propose une approche hosting).
C'est aussi ce cas de la base de données ainsi que le stockage des images. Cependant, l'app CMS peut être installé (mis dans un dossier local comme /var/www/html sur les systèmes Linux avec le serveur Apache 2 par exemple) localement. Un raccourci peut être créé pour accéder à l'app CMS depuis un

navigateur web. Je propose la solution ou les deux sites sont hébergé, pour permettre au chef d'entreprise ainsi qu'à ses employés de se connecter de n'importe où. Ma vision théorique sur "l'installation" de l'application dans le cadre de ce projet est la suivante : aucune installation est nécessaire, le CMS est "clef en main", juste un règlement mensuel (le paiement de la BD et stockage des images). Lors de la première connexion sur le Système de Gestion du Contenu il faut rentrer :

- le mail : admin@admin.com
- le mot de passe : admin

Puis dans le menu (sur la page) de gestion des utilisateurs il faut créer un utilisateur admin de votre choix et supprimer l'admin default. Pour le projet j'ai déjà créé l'admin Vincent Parrot (vous pouvez en créer d'autres).

Comme il était très claire qu'il faudra travailler avec des images des voitures, et comme j'ai déjà utilisé le service AWS (Amazon Web Services) pour leur service RDS (les bases de données relationnelles), j'ai décidé de m'orienter vers leur service de stockage cloud - S3. Après avoir configure le service et fait des tests, j'étais prêt pour le développement.

J'ai utilisé la plateforme Trello pour planifier le projet et définir le chemin du développement.

J'ai créé un projet Figma, j'ai élaboré une charte graphique en se basant sur celle, fournie dans l'énoncé.
J'ai créé les mockups mobile et desktop des pages que je pensais doivent être présentent sur le site.
J'ai décidé de ne pas perdre le temps à faire les mockups du site de gestion (CMS), car j'ai décidé d'utiliser le framework Bootstrap 5 - pour les deux sites pour l'adaptif - et pour le CMS en tant que "charte graphique", car Bootstrap offre des éléments stylisés et tous les visuels sont disponibles dans leur Documentation (https://getbootstrap.com/docs/5.3/getting-started/introduction/). Pour le CMS je pensais utiliser juste les éléments Bootstrap suivants : boutons, listes, tableaux, alertes, formulaires telles qu'ils existent.

Pour la logique générale du site principal j'ai choisi la suivante, qui ressemble au fonctionnement d'une API dans le sens ou un script JavaScript "fetch" des données en format JSON et insère l'information dans la page HTML :

- Un scripte PHP PDO génère un JSON depuis la base de données et S3 (images)
- Un scripte JS "fetch" les données (JSON) et insère les données dans la page

Cette logique me permet de travailler avec les deux langages de script.

Pour la logique de l'application CMS j'ai choisi une approche plus simple – des scripts PHP PDO et S3 qui génèrent des pages HTML correspondantes aux différentes actions demandés. Assablées dans un "dashboard" accessible grâce à une page login.

J'ai créé les différents diagrammes à l'aide de logiciels différents (dbdiagram.io, VisualParadigm, draw.io).

L'IDE que j'utilise pour tous mes projets et VSCode.

J'ai commencé le développement front-end du site principal en se basant sur les mockups. Après avoir réalisé les versions statiques mobiles et desktop je suis passé à la partie dynamique.

J'ai écrit le script MySQL en se basant sur mon diagramme de classe pour créer la base de données ainsi que les différentes tables.

J'ai écrit des scriptes pour insérer des données dans la BD (j'ai par la suite transformé ses scripts en pages du CMS)

J'ai écrit les scriptes back et front-end pour dynamiquement intégrer les données de la BD dans les pages. J'ai implémenté un timer pour "fetcher" les données et les mettre à jour sans rechargement de la page. Je comprends que c'est n'est peut-être pas la meilleure pratique, mais je voulais essayer ce

processus dans le projet. Le fonctionnement des filtres des voitures est réalisé dans la même approche (sans timer) – le script JS prend en compte les valeurs du(des) filtre(s) et “fetch” les voitures correspondantes sans recharger la page.

Puis, sur la base des scriptes PHP PDO que j’avais écrit pour insérer les données dans la BD, j’ai développé un “dashboard” pour la gestion du contenu dynamique du site (ce qui est égale à la gestion du contenu des tableaux de la DB et du “bucket” S3 avec les images) qui est accessible via une page de login.

J’ai testé les différentes fonctionnalités du site comme écrire des commentaires, des messages, appliquer les filtres voitures, visualiser les détails des voitures sur le site. En même temps je testais les fonctionnalités du “dashboard” - visualiser, approuver, supprimer les commentaires ; lire, supprimer les messages ; créer et supprimer les voitures ; gérer les heures d’ouverture ; créer, supprimer des utilisateurs ; ajouter, modifier, supprimer les services. J’ai mené ses tests en local grâce au serveur Apache 2.

Puis j’ai déployé les sites (les applis) sur la plateforme Heroku.com.

J’avais un nom de domaine que j’ai réadressé sur l’application Heroku du site principal (makarovdimitri.shop).

J’ai créé une version du projet sans les secrets (mot de passe, clefs, etc.) et j’ai push cette version sur un repositoire GitHub.