DOCUMENTATION TECHNIQUE

GAMESTORE

SOMMAIRE

REFLEXIONS INITIALES TECHNOLOGIQUES

HTML/CSS SASS AVEC BOOTSTRAP

J'ai décidé d'utiliser HTML pour la structure de base de mon site étant donné que c'est le seul langage qui m'est connu à ce jour pour réaliser un site web. A ceci j'ajoute le CSS pour le styliser mes pages et SASS pour une gestion plus efficace des variables, fonctions, mixins, et… SASS facilitera la maintenance des styles. J'ai également décidé d'utiliser en

parallèle Bootstrap que je modifie grâce à SASS pour ses composants réactifs ainsi que pour faciliter la mise en place de la partie responsive du site avec les différents breakpoints.

J'utilise le lazy loading de HTML pour les plus petites images qui ne sont pas des logos ou des icônes.

JAVASCRIPT

JavaScript sera utilisé pour des interactions dynamiques et ainsi l'amélioration de l'expérience utilisateur étant donné que JavaScript permet de modifier le contenu d'une page dynamiquement sans la recharger. Je l'utilise entre autres pour les filtres et tout ce qui est dynamique sur le site.

J'utilise également une technique de lazy loading pour toutes les grandes images qui devront être affichées à l'écran de l'utilisateur ainsi qu'un système de chargement des pages lorsque de nombreuses données sont récupérées.

PHP

PHP sera mon langage de Backend car pour le moment je n'ai réussi à approfondir mes connaissances et mon savoir-faire que dans ce langage et je ne me sens pas capable d'utiliser un autre langage pour le backend pour un projet d'une telle envergure actuellement. PHP traitera toute la logique serveur, interagira avec les bases de données et fera le traitement des données des formulaires.

PHP MAILER

J'ai décidé d'utiliser PHP Mailer pour l'envoi de mails de manière plus sécurisée et plus configurable qu'avec la fonction mail de PHP.

Google reCAPTCHA

J'intègrerai également Google reCAPTCHA pour sécuriser les formulaires contre les bots.

BASE DE DONNEES

En tant que base de données j'ai décidé d'utiliser une base de données MariaDB pour la base de données MySQL relationnelle hébergée sur IONOS et une base de données MongoDB hébergée sur Atlas pour la base de données NoSQL.

SEO ET MOTEURS DE RECHERCHE

Afin d'améliorer le SEO du site ainsi que pour les moteurs de recherche je décide d'effectuer un routage des pages en PHP car les pages seront directement visibles par les moteurs de recherche, alors qu'en JS les pages ne seront visibles qu'après le chargement par l'utilisateur étant donné qu'elles ne seront pas chargées sur le serveur mais directement sur le navigateur. Le CSR (Client-Side Rendering) est certes plus lent au chargement mais la navigation est plus fluide et il y a une meilleure persistance au niveau des données mais pour un site de commerce, ce n'est pas forcément ce que nous recherchons. Nous recherchons à être vus par les moteurs de recherche et un bon référencement. Un bon référencement est donc dans notre cas plus important qu'une meilleure expérience utilisateur et c'est pour cela que j'ai choisi le SSR (Server-Side Rendering). Les parties qui ne sont pas importantes pour les moteurs de recherche pourront être effectuée en JS.

REQUETES BASE DE DONNEES

Les requêtes à la base de données récolteront toutes les informations nécessaires en une fois afin de minimiser la connexion à la base de données et tout récupérer en une seule fois afin que les utilisateurs sur mobile avec une petite connexion ne doivent pas patienter à chaque rechargement dû à une requête à la base de données.

ENVIRIONNEMENT ET SECURITE DU SITE

J'ai décidé de placer toutes les données sensibles dans un fichier .env qui sera soit placé à la racine du site en déploiement et sécurisé par un fichier .htaccess ou placé dans un autre répertoire du serveur non accessible au public en cas d'utilisation de VPN. En plus de ceci j'ai modifié les droits d'accès à ce fichier et je l'exclu du public. DotEnv se chargera de les charger afin que ces variables d'environnement puissent être lues par le serveur.

En plus de ceci, je mets en place des mesures de sécurité lors de l'envoi et de la réception de données. Et je prends soin de faire attention à respecter les directives de l'ANSSI et de la CNIL concernant les mots de passe qui seront cryptés en BCRYPT. Les sessions d'utilisateur seront stockées dans des variables de SESSION avec des tokens pour plus de sécurité. Je fais également très attention aux failles Cross-Site Scripting (XSS) possibles.

Je mets en place une gestion des erreurs robuste aussi bien pour le développement que pour la production.

J'utilise également une CSP Content Security Policy afin de réduire le risque de chargement de contenu non désiré. Je la mets directement en place dans le fichier .htaccess (encore en cours).

J'ai également activé le protocole SSL pour mon nom de domaine afin de pouvoir utiliser le https (HyperText Transfer Protocol Secure) pour garantir la sécurité et la confidentialité des communications entre les utilisateurs et mon site web. Ce protocole chiffre mes données, me protège contre les attaques man-in-the-middle, authentifie le serveur, garantit l'intégrité des données, améliore le SEO, donne confiance aux utilisateurs avertis, respecte les normes et règlementations préconisées par le RGPD et me donne un support des fonctionnalités modernes du web.

En plus de ceci, je mets en place des paramètres de cookies avant de débuter une session et je créer un CSRF (Cross-Site Request Forgery) Token qui sera utilisé à chaque formulaire ainsi qu'à chaque fetch au sein de mon application.

Pour plus de sécurité lors de l'inscription et lors du login j'ai choisi de mettre en place un système 2FA par mail avec l'envoi d'un code à 6 chiffes qui est stocké dans la base de données temporairement. Je n'ai pas choisi l'authentification à 2 facteurs par code QR car le

site est principalement destiné aux utilisateurs avec mobile et il est difficile de scanner un code QR quand on se trouve déjà sur son mobile.

GIT

J'utilise GIT Hub et Trello pour la gestion de mon projet de développement pour la gestion des différentes versions lors de la phase de développement. Cette méthode de travail permet aussi de faciliter la collaboration par la suite et me permet d'avoir une historique des différentes versions et de toujours pouvoir récupérer une partie du code en cas de gros problème.

CONFIGURATION DE L'ENVIRONNEMENT DE TRAVAIL

MISE EN PLACE DE TRELLO

Création de l'espace sur Trello en prenant soin de le rendre public. Le lien vers le Trello du projet est : https://trello.com/b/thOCgTcx/gamestore

GITHUB

Création du repository sur GitHub en passant directement par l'interface web avec ajout directement du fichier readme.md et du fichier .gitignore :

Clic sur « Create a new Repository ».

Initialisation de mon dossier projet préalablement préparé avec différentes documentations ainsi que l'arborescence du site en passant par le terminal de mon IDE qui est VS Code. Tout en étant à la racine du projet j'entre les commandes suivantes :

```
git init
git commit -m "First commit"
git branch -M main
git remote add origin git@github.com:MyriamKuhn/Gamestore.git
git push -u origin main
```

J’effectue mon premier commit en prenant soin de vérifier les fichiers que je vais indexer puis je les pousse sur mon repository git afin de les rendre disponibles.

```
git status
git add .
git commit -m "Initialisation du projet"
git push -u origin main
```

Par la suite je crée différentes branches qui me permettront de mieux gérer mon travail : main restera toujours ma copie de secours qui sera celle publiée à l’état final et testée, dev sera la branche de travail et de développement, release sera la branche qui sera destinée à la publication et au déploiement de mon site.

CREATION DE MA BASE DE DONNEES

Je crée une base de données sur mon espace avec un utilisateur unique consacré uniquement à cette base de données et un mot de passe sécurisé. Je me connecte ensuite à cette base de données dans mon IDE et je lance le fichier de création de la base de données que j’ai créé préalablement afin de mettre en place toutes les tables.

CREATION DE MON ENVIRONNEMENT LOCAL DE TRAVAIL

J’utilise WAMP afin de disposer d’un serveur Apache local pour tester mon site en local.

Je crée un Virtual Host directement dans l’interface de WAMP qui sera nommé « gamestore.local » en suivant les instructions de WAMP.

J’ai à présent accès à mon site en local qui fonctionnera à l’aide d’un serveur Apache en entrant : http://gamestore.local/ dans mon navigateur lorsque WAMP sera actif.

Je me rends dans mon IDE pour commencer à travailler sur la première page index.php et sur l'arborescence de mon site en prenant soin de créer une sous branche de la branche dev sur mon git afin d'y publier régulièrement mon avancement.

Etant donné que j'ai décidé de travailler avec le framework Bootstrap, je prends le soin de télécharger les fichiers qui me sont nécessaires afin que je puisse adapter le framework à l'aide de SASS à l'aide de la commande « npm install bootstrap@5.3.3 »

J'installe DotEnv afin de pouvoir utiliser les variables d'environnement à l'aide de la commande « composer update vlucas/phpdotenv ». (Si ce n'est pas déjà fait je dois installer composer avant).

Je prends le soin d'installer l'extension mongodb sur mon WAMP si ce n'est pas déjà fait et j'ajoute mongodb à mon projet afin que je puisse me connecter à la base de données à partir de mon site grâce à la commande « composer require mongodb/mongodb ».

Je mets également en place PHP Mailer grâce à la commande « composer require phpmailer/phpmailer ».

Je me rends sur le site Google reCAPTCHA afin de créer une clé de site et une clé secrète pour pouvoir utiliser des CAPTCHA pour mes formulaires.

Je procède à la fin à la mise en place du rewriting de mes url par le biais du fichier .htaccess et je mets en place un système de report des erreurs directement sur mon index.php afin de pouvoir consulter des logs précises.

MCD, DIAGRAMME D'UTILISATION ET DE SEQUENCE

Ces diagrammes se trouvent dans le dossier « documentation » puis « technique » du git du projet.

DETAIL DU DEPLOIEMENT

Dans un premier temps je décide de passer par le SFTP/SSH de mon hébergeur à l'aide d'un client FTP comme FileZila afin de mettre les fichiers en ligne qui se trouvent dans la branche Release de mon git. Je configure naturellement toutes les données ainsi que le nom de domaine que je fais pointer vers un répertoire précis de mon espace web ainsi que je lui indique dans le fichier .htaccess de lancer directement l'index.php.

Par la suite j'envisage de procéder directement au déploiement en passant par Git à l'aide des commandes suivantes :

- Je me connecte à mon SSH par le terminal ou PuTTY.
- Je vais dans le répertoire de mon site (cd xyz
- Je clone le dépôt git : git clone https://github.com/username/repository.git .
- Pour mettre à jour mon site je vais à chaque fois qu'il y a des changements devoir procéder à un git pull origin release

Le site déployé est visible à l'adresse : https://gamestore.myriamkuhn.com

J'ai constaté que mon hébergeur ne me permettait pas d'installer des extension PHP sur mon pack d'hébergement et comme ma base de données NoSQL est MongoDB j'ai dû trouver rapidement une autre solution à mon problème étant donné qu'il faut installer l'extension pour PHP et MongoDB sur le serveur également.

Pour remédier à ceci j'ai pris un serveur VPS que j'ai configuré. J'utilise l'interface Plesk pour sécuriser et administrer mon serveur. J'ai également créé un sous-domaine sur lequel j'ai modifié les entrées DNS A et AAAA pour correspondre à l'adresse IP de mon serveur ainsi que les namespace NS (deux entrées). Une fois les enregistrements DNS (_acme-challenge.gamestore.mkcodecreations.dev en TXT) pris en compte je vais ajouter un certificat SSL gratuit de Lets Encrypt pour mon sous-domaine. J'en fais la demande dans Plesk et j'ajoute dans les DNS l'entrée correspondante avec la valeur demandée pour Lets Encrypt. Je vérifie que les DNS sont bien actualisés sur https://dnschecker.org/#TXT/_acme-challenge.mkcodecreations.dev et je valide ma demande sur Lets Encrypt. Je raccorde le nom de domaine à un dossier créé à l'avance sur mon serveur afin qu'il entre directement dans ce

dossier là pour chercher l'index.php. Et j'active le déploiement du site en passant par l'extension Git dans les Dev Tools de Plesk.

Elle récupèrera automatiquement les contenus de la branche production de mon repository Git et il ne me restera plus qu'à contrôler le bon fonctionnement du site déployé. Attention au reCAPTCHA de google, il faudra que j'y ajoute l'URL de mon site afin qu'il le reconnaisse.

Pour préparer ma branche production de Git je procède ainsi :

- git checkout production (pour me déplacer vers la branche production)
- git checkout dev -- dist/* (pour ne récupérer que le dossier dist de la branche dev)
- git checkout production (pour être sûre de bien me trouver sur la branche production)
- mv dist/* . (pour placer tous les fichiers du dossier dist à la racine de la branche dev)
- rmdir dist (pour supprimer le dossier vide à présent)
- git add . (pour indexer les fichiers)
- git commit -m "Message" (pour créer le commit)
- git push origin production (pour pousser le commit sur mon repository distant)

Le site déployé est visible à cette adresse : https://gamestore.mkcodecreations.dev/

Dans un avenir plus ou moins proche j'envisage d'utiliser Docker mais pour mon projet l'installation de PHP avec MySQL et MongoDB sont suffisants.