NOM	BUCHELET
Prénom	Loris
Date de naissance	20/06/1994

Copie à rendre

ECF – TP Graduate Développeur 23/29

(Android, Angular, Flutter, Front-End, Full Stack, IOS, PHP/Symfony ,No Code)

Documents à compléter et à rendre

Lien du git : https://github.com/Pandaru62/studi-arcadia.git
Lien de l'outil de gestion de projet : https://trello.com/b/wBQxZelx/arcadia
Lien du deploiement : https://studi-arcadia-24b07c2af798.herokuapp.com/

Partie 1 : Analyse des besoins

1. **Effectuez un résumé du projet en français d'une longueur d'environ 20 lignes soit 200 à 250 mots**

Le projet s'articule autour de la réalisation d'un site web responsive pour un zoo se situant dans la forêt de Brocéliande, en Bretagne. L'application web sert autant de site vitrine pour les visiteurs que d'outil pratique permettant de recenser les informations des animaux.

Tout d'abord, côté visiteur, l'application doit adopter un design qui respecte les attentes de José, pour qui l'inscription du zoo dans une dynamique écologique et respectueuse de l'environnement est indispensable. De même, l'accent est porté sur les animaux et leur bien-être. Le client insiste en effet sur l'importance de l'accès aux informations sur l'alimentation et l'avis médical des animaux tant pour les visiteurs que les employés. Ainsi, depuis la barre de navigation, les visiteurs accèdent aux animaux par le biais des trois habitats proposés par le zoo (le marais, la jungle et la forêt). Enfin, les visiteurs doivent être en mesure de laisser des commentaires et contacter le zoo.

D'autre part, pour les membres de l'équipe du zoo, un compte utilisateur leur permet de s'authentifier afin d'alimenter le site et ainsi la base de données sans laquelle l'application n'aurait pas d'intérêt. Trois rôles sont déterminés : l'administrateur (José), les employés et les vétérinaires qui ont, pour chacun, des droits très précis et limités. Les fonctions du CRUD doivent être appliquées aux animaux, habitats, services, commentaires et à la gestion des utilisateurs.

En somme, en plus de refléter les belles valeurs du zoo aux yeux des visiteurs, José souhaite une application intuitive qui favorisera l'échange des données et la communication au sein de l'équipe.

2. Exprimez le cahier des charges, l'expression du besoin ou les spécifications fonctionnelles du projet

Le client, José, a exprimé le besoin d'une application web pour son zoo. Cette application doit tout d'abord répondre à des critères esthétiques. Il est attendu du client un design reflétant l'aspect écologique qui est une valeur essentielle pour le lui. De plus, les besoins sont guidés par l'importance du bien-être des animaux ainsi que du travail quotidien effectué par les vétérinaires et les employés.

Le cahier des charges fait ainsi référence à :

1. La présence du page d'accueil faisant office de présentation du zoo, de ses habitats, ses animaux et des services proposés. Cette page est indispensable aux visiteurs puisqu'elle leur permet aussi d'accéder aux avis du zoo.
2. L'utilisation d'un menu accessible aux visiteurs afin de naviguer entre la page d'accueil, les services, les habitats, le formulaire de contact et le bouton connexion. Ce dernier bouton, bien qu'étant visible des visiteurs, ne leur donne pas la possibilité de se connecter car cette fonctionnalité n'est accessible qu'aux employés et vétérinaires.
3. Une section permettant de consulter les services proposés par le zoo. Par défaut, chaque service est défini par son nom et une description. Si actuellement trois services sont proposés par le zoo (restauration, visite des habitats gratuite accompagnée par un guide, et visite en petit train), un système de modification et d'ajout doit être mis à la disposition de l'administrateur.
4. Une page donnant accès aux informations des différents habitats. Les habitats doivent tout d'abord apparaître avec leur nom et leur image d'illustration, puis après un clic, permettre l'accès à leur description et à la liste des animaux le composant. Par ailleurs, un visiteur peut avoir accès aux dernières informations saisies par le vétérinaire (concernant son état, la nourriture recommandée et sa quantité, la date de passage et d'autres informations facultatives à propos de l'animal) ainsi qu'aux caractéristiques plus générales de l'animal (son prénom, sa race, une ou des images, et son habitat de rattachement).
5. Un accès aux avis des autres visiteurs et à un formulaire permettant d'envoyer son propre avis. L'avis envoyé ne sera toutefois en ligne qu'après validation d'un employé de celui-ci depuis son espace dédié.
6. La connexion de l'administrateur vers son espace. L'espace administrateur doit inclure :
    1. La possibilité de créer un nouvel employé ou vétérinaire (mais pas d'autre administrateur) grâce à une adresse e-mail et un mot de passe. La création de ce nouvel utilisateur sera par ailleurs confirmée grâce à l'envoi d'un e-mail à ce dernier.
    2. L'accès aux fonction CRUD (création, visionnage, modification, suppression) des services, horaires, habitats et animaux du zoo.
    3. Le visionnage des comptes rendus du vétérinaire pour chaque animal à l'aide de filtres et d'outils de tri pour mieux rechercher par animal ou par date.
    4. La consultation à un Dashboard indiquant l'animal le plus consulté.
7. La connexion de l'employé à son espace. L'espace employé propose :
    1. La validation ou invalidation d'un avis posté par un visiteur.
    2. La modification des services du zoo.
    3. L'accès à un formulaire d'ajout du nourrissage d'un animal (avec la date, l'heure , la nourriture proposée et sa quantité).
8. La connexion du vétérinaire à son espace. L'espace vétérinaire se compose de :
    1. L'accès à un formulaire d'ajout d'un compte rendu animal (concernant son état, la nourriture recommandée et sa quantité, la date de passage et d'autres informations facultatives à propos de l'animal)

2. La fonctionnalité de commentaire d'un habitat afin de signaler toute idée d'amélioration d'un enclos.
3. L'accès à l'historique de nourrissage d'un animal tel qu'il a été rempli par l'employé.

9. La connexion doit se faire par un des trois rôles (administrateur, employé, vétérinaire) et personne d'autre. La création de compte ne peut se faire que par un administrateur et la connexion s'effectue par le biais de l'e-mail et du mot de passe renseignés par l'administrateur au moment de la création du compte.
10. La possibilité pour un visiteur via un formulaire de contact d'envoyer un message à l'équipe du zoo. Ce message composé d'un titre, du corps du texte et de l'e-mail du visiteur, sera immédiatement retransmis sur l'adresse e-mail du zoo afin qu'un employé puisse directement répondre sur l'adresse e-mail fournie par le visiteur.
11. Des statistiques doivent être mises à la disposition de l'administrateur via son dashboard. Ces statistiques montreront l'animal le plus visité puisqu'un compteur de visite sera incrémenté à chaque passage sur la page d'un animal.

Pour des détails plus précis, je vous invite à consulter la documentation traitant de la gestion de projet de l'application.

Partie 2 : Spécifications techniques

1. **Spécifiez les technologies que vous avez utilisées en justifiant les conditions d'utilisation et pourquoi le choix de ses éléments.**

Afin de réaliser ce projet, diverses technologies ont été utilisées, du Front, au Back, comme pour le déploiement.

Sur le plan Front-End, les langages **HTML** et **CSS** sont incontournables afin de travailler l'aspect visuel traité directement par les navigateurs des utilisateurs de l'application Web. Afin de simplifier l'aspect responsive pour permettre une expérience utilisateur plus aboutie et appropriée à tous les appareils, le framework **Bootstrap** a été retenu pour sa facilité de prise en main, sa relative maniabilité et sa possibilité d'offrir un design dynamique et épuré à la fois. Afin d'étoffer l'utilisation de Bootstrap, **SASS** a été ajouté afin de personnaliser certains styles normalement imposés et customiser quelques outils comme les cartes ou le carrousel. Enfin **Javascript** a été ponctuellement retenu, notamment en collaboration avec Bootstrap pour faire fonctionner les menus déroulants et autres dispositifs en autoplay, mais aussi pour réagir en direct aux champs utilisateurs sur certains formulaires.

Sur le plan Back-End, **PHP** s'est imposé comme une évidence afin d'effectuer la liaison avec la base de données et proposer des contenus alimentés dynamiquement selon les besoins de l'utilisateur. PHP a également servi a créer les routes entre les diverses pages. **MAMP** a été retenu en tant que solution de travail en local afin d'utiliser le serveur **Apache** et permettre la gestion des bases de données relationnelles en **MySQL**. La base de données relationnelle permet dans le cadre de ce projet de mettre en lien diverses catégories et d'éviter la redondance grâce à l'usage des clés primaires et étrangères. Ainsi chacune des tables est directement connectée à une autre afin de mettre en relation les animaux à leur espèce puis leur habitat notamment. La base de données NoSql **Mongo DB** a enfin été retenue pour recenser le nombre de visites sur chacune des pages des espèces. Celle-ci n'a tout d'abord pas besoin d'être mise en relation directe avec les autres tables puisqu'il s'agit juste de répertorier un ID d'animal et d'y associer un nombre pour le compteur au sein d'une collection. En local, Mongo DB a été utilisé par le biais de **Docker Desktop.**

L'essentiel du projet s'est ainsi constitué autour de tous ces outils via **VS Code** pour des raisons pratiques et permettre grâce au terminal de dialoguer aisément avec **Git**. Pour le déploiement, un service PAAS comme **Heroku** s'est avéré être la solution la plus appropriée dans la mesure où il permet à la fois un déploiement à moindres coûts mais également une prise en charge des données SQL via l'add-on **JawsDB** Mysql et des données NoSQL en passant par **MongoDB Atlas**.

2. **Comment avez-vous mis en place votre environnement de travail ? Justifiez vos choix.**

- L'environnement de travail en local s'est d'abord effectué par le biais de l'utilisation d'un IDE : Visual Code. Après la création d'un dossier local '**studi_arcadia**', j'ai immédiatement créé un ficher **readme.md**. VSCode est également accompagné d'extensions utiles comme PHP, PHP getters and setters, ou encore composer, et s'accompagne du terminal permettant d'interagir notamment avec Git.

- Sur Github, un repository public 'studi_arcadia' a été créé, puis depuis le terminal les commandes suivantes ont permises d'initialiser le git, d'ajouter le fichier readme.md ainsi que d'effectuer un premier commit pour tester l'interaction entre le git local et le distant.

```
$ git init
$ git add .
$ git commit -m 'first commit'
$ git push
```

- Après cela, la création d'une branche dev découlant immédiatement de la branche main a été effectuée. Cette branche servira pour le développement de chaque fonctionnalité qui sera *push* après chaque test effectif en local.

- L'installation de MAMP est aussi essentielle afin de fournir un serveur local et tester aisément le langage PHP. L'installation de MAMP s'est faite par le biais du site officiel. Pour la configuration, les paramètres par défaut ont été gardés. Le dossier racine du serveur a été rattaché au dossier **studi_arcadia** précédemment créé en local. Enfin, la version PHP 8.3.1 a été retenue.

- Afin de simplifier l'interaction entre le terminal et PHP, la variable d'environnement a été ajoutée dans les configurations système de l'ordinateur. Path a été modifié pour y intégrer **C:\MAMP\bin\php\php8.3.1**.

- Le gestionnaire de paquets et de dépendances a aussi été intégré au dossier à l'aide des commandes suivantes. L'installation de *composer* est en effet indispensable à ce titre.

```
$ npm install
$ composer -v
$ composer install
```

- Afin de simplifier et rendre cohérente la charte graphique, le framework *Bootstrap* a été ajouté. Après la création d'un index.html, Bootstrap a été installé via CDN afin de pouvoir ensuite mieux personnaliser les styles pré-définis.

- `$ npm install bootstrap@5.3.3`

- Ainsi dans le dossier scss, il a été possible de personnaliser et ajouter mon propre CSS à partir de deux fichiers distincts _custom.scss et main.scss. L'extension VSCode ***Sass Compiler*** a ainsi permis de compiler le code sous un fichier .css présent dans le même dossier, ainsi ajouté dans le <head> des pages html.

3. Énumérez les mécanismes de sécurité que vous avez mis en place, aussi bien sur vos formulaires que sur les composants front-end ainsi que back-end.

Pour assurer la sécurité du site web, plusieurs mécanismes de sécurité ont été mis en place à la fois sur les formulaires, et sur les composants front-end ainsi que back-end. Voici un aperçu des mesures adoptées :

1. **Formulaires** :
    - **Utilisation de `htmlspecialchars`** : Pour prévenir les attaques de type XSS (Cross-Site Scripting), nous utilisons la fonction `htmlspecialchars` afin d'échapper les caractères spéciaux dans les entrées des utilisateurs.
    - **TOKENS CSRF** : Pour protéger contre les attaques CSRF (Cross-Site Request Forgery), des tokens CSRF sont intégrés dans les formulaires. Ces tokens sont vérifiés à chaque soumission de formulaire pour s'assurer que les demandes proviennent bien de l'utilisateur légitime.
2. **Composants Front-end** :
    - **Validation côté client** : Nous implémentons une validation robuste des entrées utilisateur à l'aide de JavaScript avant l'envoi des données au serveur, afin de prévenir les erreurs et les abus au niveau du client. Par exemple, les champs de saisie d'email sont vérifiés afin qu'ils contiennent une adresse email valide et que les champs de mot de passe respectent les critères de complexité (longueur minimale, inclusion de caractères spéciaux, etc.).
3. **Composants Back-end** :
    - **Hashage sur BDD** : Les mots de passe des utilisateurs et autres données sensibles sont stockés dans la base de données sous forme hachée, utilisant des algorithmes sécurisés comme bcrypt, afin de protéger les informations en cas de compromission de la base de données.
    - **Gestion des exceptions avec `try {}` et `catch {}`** : Les blocs `try {}` et `catch {}` sont intégrés pour gérer les exceptions et erreurs de manière sécurisée. Un gestionnaire d'exceptions est mis en place pour enregistrer les erreurs et fournir des messages d'erreur génériques à l'utilisateur, sans divulguer de détails sensibles sur le fonctionnement interne de l'application.
4. **Architecture** :
    - **Modèle MVC (Modèle-Vue-Contrôleur)** : L'application est structurée selon le modèle MVC. Cette architecture permet de séparer les préoccupations et de renforcer la sécurité en isolant la logique métier, les interfaces utilisateur, et les interactions avec la base de données.

4. **Décrivez une veille technologique que vous avez effectuée sur les vulnérabilités de sécurité.**

Pour assurer la sécurité de mon site web, j'ai mis en place une veille technologique continue sur les vulnérabilités de sécurité. Voici quelques-unes des pratiques et outils que j'utilise pour cette veille :

1. **Relecture du code à l'aide de l'extension SolarLint** :
    - J'utilise l'extension SolarLint pour analyser et vérifier mon code source. Cet outil permet de détecter automatiquement les vulnérabilités potentielles, les mauvaises pratiques de codage et les erreurs de sécurité. Grâce à SolarLint, je peux corriger les failles de sécurité dès les premières étapes du développement.
2. **Utilisation de ZAP 2.15.0** :
    - J'ai intégré OWASP ZAP (Zed Attack Proxy) version 2.15.0 dans mon processus de développement. Cet outil de test de sécurité des applications web m'aide à identifier et à corriger les vulnérabilités en simulant des attaques réelles. ZAP analyse mon application pour détecter les failles telles que les injections SQL, les failles XSS, et d'autres vulnérabilités courantes.
3. **Lecture partielle du guide OWASP Web Security Testing Guide** :
    - Je me réfère autant que possible à l'OWASP Web Security Testing Guide. Ce guide exhaustif fournit des méthodologies et des techniques pour tester la sécurité de mon application web.
    - Voici le lien : https://owasp.org/www-project-web-security-testing-guide/

En adoptant ces pratiques et en utilisant ces outils, je maintiens une veille technologique efficace sur les vulnérabilités de sécurité, ce qui me permet de rester à jour avec les dernières menaces et de renforcer continuellement la sécurité de mon site web.

Partie 3 : Recherche

1. **Décrivez une situation de travail ayant nécessité une recherche durant le projet à partir de site anglophone. N'oubliez pas de citer la source.**

Lors du développement de la partie « *Contact* » de l'application web, il m'a fallu plus de guidages concernant la fonctionnalité d'envoi d'e-mails. En effet, si je savais récupérer les informations saisies par le visiteur depuis le formulaire de contact, il me restait à retransmettre les champs remplis à une adresse e-mail (l'adresse du zoo) afin que les employés puissent répondre directement par e-mail au visiteur ayant soumis une question ou une réclamation.

Après quelques recherches en anglais, c'est l'article du site dont voici le lien https://mailtrap.io/blog/phpmailer/ que j'ai retenu pour sa clarté. Cet article a été réalisé par Dmitriy Shcherbakan et Ivan Djuric le 22/03/2024 pour le site *Mailtrap* (by railsware). J'avais besoin d'utiliser PHPMailer et de le relier à mon adresse e-mail afin de compléter cette fonctionnalité. Je me suis essentiellement appuyé sur cette article pour la configuration de PHPMailer, puis j'ai utilisé Gmail pour les configurations propres à mon adresse e-mail.

2. **Mentionnez l'extrait du site anglophone qui vous a aidé dans la question précédente en effectuant une traduction en français.**

Voici un extrait de l'article retenu en anglais :

> When it comes to adding email functionality to your PHP app, the PHPMailer class is the winning option.
>
> In this article, I'll describe the process of installing PHPMailer, configuring SMTP settings, and adding email-sending functionality to your app step-by-step.
>
> To jump straight into the action, click here.
>
> *The code snippets I provide in this article are for PHPMailer 6.9, which is compatible with PHP 5.5 and above.*
>
> **What is PHPMailer?**
>
> PHPMailer is the classic email-sending, open-source library for PHP. It's compatible with most PHP frameworks, such as Laravel and Symfony and is used by many open-source projects like WordPress. It also supports multiple ways of sending email messages, including the most reliable one, **direct dispatch to SMTP servers**, and **PHP mail() function**, **Sendmail**, and **qmail**.
>
> Some of the most important PHPMailer features include:
>
> [...]
>
> **How to install PHPMailer**
>
> To install PHPMailer, you'll need Composer, a dependency manager for PHP,

recommended by PHPMailer creators on GitHub.

Once installed, add this line to your **composer.json** file:

```
"phpmailer/phpmailer": "^6.9"
```

Note:

- The **vendor** folder and the **vendor/autoload.php** script are not part of PHPMailer. Instead, they are generated by Composer.

If you're not a fan of Composer, you can download PHPMailer files with source code manually and copy the contents of the folder into an **include_path** directory specified in your PHP configuration and load each class file manually, like so:

[…]

Adding an Exception class will help you handle errors and debug them. In PHP it works similarly to other programming languages. So, without it, if there is an error in your email sending code, you will just see a message saying that the Exception class is not found, but you won't be provided with any details on how to debug it.

[...]

PHPMailer SMTP settings

Sending emails with PHPMailer is most optimal when paired with an external SMTP server. This makes it reliable and secure as it can be encrypted (TLS/SSL), making it an ideal choice for your PHP application.

En voici une traduction de l'extrait :

Lorsque vous avez besoin d'ajouter des fonctionnalités e-mail à votre appli PHP, la classe PHPMailer est sans nul doute la meilleure option.

Dans cet article, je vous présente étape par étape le processus d'installation de PHPMailer, les paramètres de configuration SMTP, ainsi que l'ajout de la fonctionnalité d'envoi d'e-mail.

Les fragments de code que je donne dans cet article concernent PHPMailer 6.9, qui est compatible avec les versions PHP 5.5 et plus.

What is PHPMailer?

PHPMailer est la bibliothèque incontournable open-source pour envoyer des e-mails. Il est compatible avec la plupart des frameworks PHP comme Laravel et Symfony et est utilisé par de nombreux projets open-source comme Wordpress. Il est aussi compatible avec de nombreuses méthodes d'envois d'e-mail, notamment le plus fiable d'entre eux : **direct dispatch to SMTP servers**, et la **fonction PHP mail()**, **Sendmail**, et **qmail**.

[...]

Comment installer PHPMailer

Pour installer PHPMailer, vous aurez besoin de Composer, un gestionnaire de dépendances pour PHP, recommandé par les créateurs de PHPMailer sur GitHub.

Une fois installé, ajouter cette ligne à votre fichier composer.json :

```
"phpmailer/phpmailer": "^6.9"
```

Note:

- Le dossier **vendor** et le script **vendor/autoload.php** ne font pas partie de PHPMailer mais sont générés par Composer.

Si vous n'êtes pas un habitué de Composer, vous pouvez télécharger les fichiers de PHPMailer manuellement via le code source et copier le contenu du dossier dans un dossier **include_path** existant dans votre configuration PHP et charger le fichier de chaque classe manuellement comme ceci :

[...]

L'ajout d'une classe exception vous permettra de gérer les erreurs et les corriger. PHP fonctionne comme les autres langages de programmation. De ce fait, sans elle, si une erreur survient dans votre code d'envoi d'e-mail, vous verrez simplement un message indiquant que la classe Exception n'a pas pu être trouvée, mais vous n'aurez aucune information relative à la démarche permettant de la corriger.

[...]

Paramètres STMP PHPMailer

L'envoi d'e-mail via PHPMailer est le plus optimal lorsqu'il va de pair avec un serveur SMTP externe. Ceci en fait une méthode fiable et sécurisée puisqu'il peut être crypté (TLS/SSL) : un choix idéal pour votre application PHP.

Partie 4 : Informations complémentaires

1. Autres ressources

Vous trouverez l'ensemble des documents tels que le manuel d'utilisation, la charte graphique, les documentations technique et de gestion de projet sur le git dans le dossier nommé « Studi ».

2. Informations complémentaires

/