NOM	KOWALSKI
Prénom	Sullivan
Date de naissance	30/04/1989

Copie à rendre

TP – Développeur Web et Web Mobile

Documents à compléter et à rendre

Lien du git : https://github.com/Sullivankow/ArcadiaFront.git , https://github.com/Sullivankow/ArcadiaBack.git
Lien de l'outil de gestion de projet :
https://trello.com/invite/b/667c536534ba72013a6c2827/ATTIfbb234fb6a8b14764ac65fced5c7618305D75697/arcadia
Lien dudéploiement :https://parcarcadia.alwaysdata.net/,upbeat-happiness-production.up.railway.app
Login et mot de passe administrateur : admin@email.com **, mdp: Azerty123**

SANS CES ELEMENTS, VOTRE COPIE SERA REJETEE

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**

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

Partie 2 : Spécifications technique

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

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

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.**

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

Partie 3 : Recherches

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**

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

Partie 4 : Informations complémentaire

1. **Autres ressources**

2. **Informations complémentaires**

Partie 1 : ANALYSE DES BESOINS

1.Résumé du projet

Le projet consiste à développer une application web pour le Zoo Arcadia, un établissement situé près de la forêt de Brocéliande. Cette application a pour objectif d'améliorer l'image de marque du zoo, en mettant en avant ses valeurs écologiques et en offrant une expérience interactive à ses visiteurs et employés.
L'application devra comporter plusieurs fonctionnalités adaptées aux besoins de différents types d'utilisateurs. Les visiteurs pourront consulter une présentation du zoo, découvrir les habitats et les animaux, laisser des avis et accéder aux services proposés, tels que la restauration ou les visites guidées. Un formulaire de contact permettra également de joindre le zoo.
Les employés auront un espace dédié pour valider les avis des visiteurs, gérer les services et enregistrer l'alimentation donnée aux animaux. Les vétérinaires disposeront d'un outil pour rédiger des rapports sur l'état des animaux et commenter les habitats. Enfin, l'administrateur aura un contrôle total sur l'application, avec la possibilité de gérer les utilisateurs, les services, les habitats et les animaux. Il aura également accès à des statistiques détaillées pour analyser la popularité des animaux.
L'application devra être développée avec des technologies web modernes, incluant une base de données relationnelle et non relationnelle, et un design responsive. Avec une gestion de projet via un tableau Kanban, une documentation complète, des choix techniques, des maquettes et des procédures de déploiement.
En somme, ce projet vise à valoriser le zoo en offrant une interface interactive et écoresponsable, tout en facilitant la gestion interne.

2.Les besoins

Le Zoo Arcadia, soucieux de moderniser son image et de mettre en avant ses valeurs écologiques, souhaite une application web pour améliorer l'expérience des visiteurs et optimiser la gestion interne. Cette application doit permettre :

- **Aux visiteurs** : de consulter les habitats, animaux, services, et de laisser des avis.
- **Aux employés** : de gérer les avis, services, et l'alimentation des animaux.
- **Aux vétérinaires** : de suivre l'état des animaux et des habitats.
- **À l'administrateur** : de superviser les données et d'avoir un tableau de bord analytique.

Cette solution doit être ergonomique, responsive, sécurisée, et conforme aux standards actuels du développement web.

3.Spécifications Fonctionnelles

3.1. Pour les Visiteurs

- **Page d'accueil** : Présentation du zoo, de ses valeurs écologiques, des habitats, et des avis validés.

- **Menu de navigation** : Accès rapide aux sections principales (accueil, habitats, services, contact, connexion).
- **Consultation des habitats et animaux** : Vue des habitats avec description et animaux associés, consultation des rapports vétérinaires.
- **Avis** : Soumission d'un avis (pseudo, texte), avec validation par un employé.
- **Formulaire de contact** : Envoi de messages au zoo (titre, description, email).

3.2. Pour les Employés

- **Gestion des avis** : Validation ou rejet des avis soumis par les visiteurs.
- **Gestion des services** : Ajout, modification et suppression.
- **Suivi alimentaire** : Enregistrement des informations sur l'alimentation des animaux (type, quantité, date, heure).

3.3. Pour les Vétérinaires

- **Saisie de rapports** : État de santé des animaux, nourriture prescrite, observations facultatives.
- **Commentaires sur les habitats** : Évaluation de leur état et recommandations.
- **Consultation des informations alimentaires** : Accès à l'historique enregistré par les employés.

3.4. Pour l'Administrateur

- **Gestion des utilisateurs** : Création et gestion des comptes pour employés et vétérinaires.
- **Administration générale** : Modification des données sur les services, habitats, et animaux.
- **Statistiques** : Tableau de bord avec données sur la consultation des animaux, basée sur une base NoSQL.

3.5. Données et Analyses

- Enregistrement des interactions des visiteurs avec les animaux dans une base de données NoSQL.
- Génération de rapports pour suivre les tendances (ex. popularité des animaux).

Contraintes Techniques

- **Technologies** : Utilisation d'une base de données relationnelle (MySQL) et non relationnelle (MongoDB).
- **Sécurité** : Authentification robuste pour les utilisateurs (employés, vétérinaires, administrateurs).
- **Responsive** : Compatibilité avec le format bureau et mobile.
- **Déploiement** : Hébergement sur Always Data, PlatformSh.

L'objectif est de développer une application intuitive et complète, offrant des outils adaptés aux visiteurs, employés, vétérinaires, et administrateurs, tout en mettant en avant les valeurs écologiques du Zoo Arcadia.

Partie 2 Spécifications techniques

1.Technologies utilisées et justification des choix

Matériel requis

- **Ordinateur sous Windows** : Un environnement stable et largement compatible avec les outils de développement courants.
- **Connexion internet** : Nécessaire pour accéder aux ressources en ligne, bibliothèques, frameworks, et pour les tests d'intégration avec des services déployés.

Outils de développement

1. **Visual Studio Code** :
    a. Un éditeur de code léger et polyvalent.
    b. Offre une grande variété d'extensions (ex. Intellisense, intégration Git, et debug).
    c. Idéal pour écrire et gérer efficacement du code HTML, CSS, JavaScript, et PHP.
2. **XAMPP** :
    a. Simule un environnement de serveur local incluant Apache et MySQL.
    b. Utile pour tester rapidement le site et ses fonctionnalités backend avant déploiement.
3. **Postman** :
    a. Permet de tester les requêtes API (GET, POST, PUT, DELETE).
    b. Indispensable pour valider la communication entre le frontend et le backend.

Langages de programmation et frameworks

1. **HTML5, CSS3, JavaScript** :
    a. **HTML5** : Structure les pages web et garantit leur compatibilité moderne.
    b. **CSS3** : Stylise les interfaces avec une approche responsive pour une expérience utilisateur optimale.
    c. **JavaScript** : Ajoute de l'interactivité et contrôle les éléments dynamiques sur le frontend.
2. **PHP 8.2.12** :
    a. Langage backend robuste et bien supporté.
    b. Utilisé avec Symfony pour une organisation claire et des fonctionnalités avancées (ex. sécurité, routage).
3. **Symfony** :
    a. Framework PHP moderne et puissant.
    b. Facilite la gestion des API RESTful et l'interaction avec la base de données relationnelle grâce à Doctrine.
4. **PhpUnit** :
    a. Permet de réaliser des tests unitaires pour garantir la fiabilité du code.

Bases de données

1. **MySQL Workbench 8.0** :
    a. Interface graphique pour gérer la base de données relationnelle utilisée pour stocker les données principales (utilisateurs, habitats, services, etc.).
    b. Choisi pour sa performance et sa compatibilité avec Symfony.
2. **MongoDB** :
    a. Utilisé pour les statistiques non structurées, comme le suivi des consultations des animaux.
    b. Justifié par sa flexibilité et sa capacité à gérer des données volumineuses ou non relationnelles.

Design et gestion du projet

1. **Figma** :
    a. Création des maquettes, wireframes et mockup.
    b. Outil collaboratif, pratique pour visualiser et itérer sur le design avant développement.
2. **Trello** :
    a. Suivi du projet avec la méthode KANBAN.
    b. Permet de visualiser l'état d'avancement des tâches et de prioriser les fonctionnalités.

Déploiement

1. **Always Data / Platform.sh/Railway**:
    a. Plateformes performantes pour déployer les parties backend et frontend.
    b. Offrent des environnements flexibles avec support pour les technologies web modernes.
2. **FileZilla** :
    a. Simplifie les transferts de fichiers locaux vers le serveur distant pour le déploiement.

Outils de versionnage

1. **Git** :
    a. Suivi des modifications de code, collaboration efficace et gestion des branches.
2. **GitHub** :
    a. Hébergement du code source et documentation.
    b. Utilisé pour partager le projet avec les parties prenantes et faciliter le déploiement CI/CD.

Le choix de cette stack technique est basé sur sa compatibilité, ses performances, et son efficacité. Elle répond aux besoins du projet tout en garantissant un développement structuré, une gestion de données fiable, et une expérience utilisateur optimale.

2.Mise en place environnement de travail

2.1. Préparation du matériel

- **Ordinateur sous Windows** : J'ai choisi cet environnement pour sa compatibilité avec la plupart des outils de développement et pour bénéficier des performances nécessaires au développement web.
- **Connexion internet** : Essentielle pour télécharger les dépendances, accéder aux outils en ligne et gérer les dépôts GitHub.

2.2. Outils utilisés et installation

1. **Visual Studio Code** :
    a. Raison du choix : Léger, personnalisable, et adapté au développement frontend et backend. Les extensions comme Prettier, ESLint, et Symfony ont permis d'améliorer la qualité et la lisibilité du code.
    b. Installation : Téléchargé et installé depuis le site officiel.
2. **XAMPP** :
    a. Raison du choix : Permet de simuler un serveur local avec Apache et MySQL, indispensable pour tester les interactions entre le backend PHP et la base de données relationnelle.
    b. Installation : Version compatible téléchargée depuis le site officiel.
3. **PHP 8.2.12** :
    a. Raison du choix : Utilisé pour le backend, cette version récente est compatible avec Symfony et apporte des performances améliorées.
    b. Installation : Inclus dans XAMPP.
4. **MySQL Workbench 8.0** :
    a. Raison du choix : Fournit une interface graphique pour gérer efficacement la base de données relationnelle.
    b. Installation : Depuis le site officiel MySQL, pour interagir directement avec le serveur MySQL fourni par XAMPP.
5. **MongoDB** :
    a. Raison du choix : Choisi pour stocker des données non relationnelles (ex. consultations des animaux), MongoDB offre flexibilité et rapidité dans le traitement de données volumineuses.
    b. Installation : Serveur et Compass installés via les packages officiels.
6. **Postman** :
    a. Raison du choix : Outil intuitif pour tester et valider les requêtes API avant l'intégration au frontend.
    b. Installation : Version desktop téléchargée et configurée.
7. **Trello** :
    a. Raison du choix : Utilisé pour organiser le projet avec la méthode KANBAN, ce qui a permis un suivi clair des tâches.
    b. Mise en place : Création d'un tableau et organisation en colonnes (À faire, En cours, Terminé).

2.3. Langages et frameworks

- **HTML5, CSS3, JavaScript** : Pour le développement du frontend. Ces standards garantissent une compatibilité moderne.
- **PHP avec Symfony** : Utilisé pour structurer le backend, gérer les API et interagir avec MySQL. Symfony a été choisi pour ses outils puissants (Doctrine, Sécurité).
- **PhpUnit** : Utilisé pour écrire des tests unitaires afin d'assurer la robustesse du backend.

2.4. Déploiement

- **FileZilla** : Utilisé pour transférer les fichiers vers le serveur distant.
- **Always Data / Platform.sh** : Plateformes sélectionnées pour leur fiabilité et leur compatibilité avec les technologies utilisées (PHP, MySQL, MongoDB).

Justification des choix

L'environnement de travail a été conçu pour assurer :

- **Efficacité** : Outils faciles à configurer et utiliser (XAMPP, Visual Studio Code).
- **Fiabilité** : Utilisation de frameworks modernes et sécurisés (Symfony, MongoDB).
- **Productivité** : Intégration d'outils de gestion de projet (Trello) et de test (Postman).
- **Compatibilité** : Respect des standards de développement pour garantir un déploiement fluide et performant.

3.Mécanismes de Sécurité Implémentés

3.1. Sécurité des Formulaires (Frontend et Backend)

- **Validation côté client (Frontend)** :
  - Vérification des champs requis (ex. email, mots de passe) avec HTML5 (`required`, `type="email"`).
  - Utilisation de Regex pour contrôler les formats spécifiques (comme un email valide ou des mots de passe forts).
  - Messages d'erreur clairs pour guider l'utilisateur en cas d'erreur.
  - La session de l'utilisateur expire au bout de 7 jours.
- **Validation côté serveur (Backend)** :
  - Système d'authentification, entité User (création d'un apiTokenAuthentificator)
  - Système d'autorisation, sécuriser et bloquer les accès
  - Symfony security bundle
- **Protection contre les attaques XSS (Cross-Site Scripting)** :
  - Encodage des sorties dans le DOM avec des fonctions sécurisées. (Ajouter image)

3.2. Sécurité des Composants Frontend

- **HTTPS** :
  - Utilisation d'un certificat SSL pour crypter toutes les communications entre le frontend et le serveur.
- **Gestion des Tokens** :
  - Stockage des tokens d'authentification (JWT) dans un stockage sécurisé

3.3. Sécurité des Composants Backend

- **Authentification et Autorisation** :
  - Utilisation de rôles et permissions (`ROLE_ADMIN`, `ROLE_EMPLOYEE`, etc.) pour limiter les accès aux différentes parties de l'application.

- Implémentation de la méthode `getRoles()` pour chaque utilisateur afin d'éviter des accès non autorisés.
- **Hashage des mots de passe** :
  - Hashage sécurisé des mots de passe avec **bcrypt** ou **argon2**.
- **Protection contre les attaques SQL Injection** :
  - Utilisation de requêtes préparées ou d'un ORM comme Doctrine pour interagir avec la base de données.

3.4. Sécurité Générale

- **Configuration des CORS** :
  - Limitation des domaines autorisés à faire des requêtes sur l'API (ex. configuration de NelmioCorsBundle avec `%env(CORS_ALLOW_ORIGIN)%`).
- **Protection contre les en-têtes malveillants** :
  - Mise en place de politiques de sécurité strictes avec des en-têtes HTTP comme `Content-Security-Policy`, `X-Frame-Options`, et `Strict-Transport-Security`.
- **Gestion des erreurs** :
  - Personnalisation des pages d'erreur pour éviter de divulguer des informations techniques.

3.5. Sécurité de la Base de Données

- **Séparation des privilèges** :
  - Chaque utilisateur de la base de données a un niveau d'accès limité selon son rôle.

4.Veilles technologiques

J'ai pu à travers mon expérience STUDI faire des veilles technologiques sur la cybersécurité. Je me sers principalement de l'application Reddit, qui est possible d'avoir directement sur téléphone, où j'ai pu mettre en favoris divers sujet dans l'informatique, dont la cybersécurité.
J'ai pu en apprendre d'avantage sur :

- **Les failles XSS** : Injecter des scripts malveillants directement dans le navigateur
- **Les injections SQL** : Par manipulation de données en faisant des injections SQL à travers un champs de formulaire
- **Attaques DDOS** : Objectif, saturer le serveur avec des requêtes pour le rendre indisponible
- **Attaque par force brute** : Deviner un mot de passe ou un token en essayant toutes les combinaisons possible grâce à un logiciel automatisé

Reddit est une vraie mine d'or en termes d'informations ou il est même possible de pouvoir parler de différents sujets sur les forums.
D'autres sites ont pu attiser ma curiosité comme l'ANSSI qui est le site officiel de cybersécurité français qui nous propose diverses formations.
CERT FR est aussi une référence, sur ce site, il nous informe en temps réel, les diverses attaques en cours dans le pays grâce à ses alertes de sécurité afin de nous prévenir des menaces. Ces sites sont des sources sûr et me permettent de prendre en compte mes failles durant mon projet. Ils m'aideront à appréhender le danger sur mes futures sites et applications.

Partie 3 : RECHERCHE

1.Mise en situation

Durant le projet ARCADIA, j'ai pu faire face à un problème récurrent qui m'a obligé à faire des recherches, et creuser un peu plus le sujet pour essayer de comprendre pourquoi j'étais face à celui-ci. Il s'agit de l'erreur CORS (Cross-Origin Source Sharing). Cette erreur qui bloquait mes requêtes front vers l'api, m'a prise plusieurs jours de réflexion, elle a été mon plus gros défi mais aussi ma plus grosse perte de temps. J'ai dû dans un premier temps, comprendre ce qu'était le CORS.
Ayant compris qu'il s'agissait d'un dispositif de sécurité mis en place par les navigateurs pour sécuriser les requêtes utilisateur en vérifiant d'où provient leur origine.
Plusieurs options s'offraient à moi, reconfigurer les fichiers nelmio_cors ou bien désactivé cette sécurité directement depuis le navigateur. La première n'ayant pas eu un franc succès, j'opte pour la deuxième solution et décide de faire des recherches sur internet.
C'est sur le site Stack Overflow que ma solution fut trouvée.

2.Source

Lien source :
https://stackoverflow.com/questions/3102819/disable-same-origin-policy-in-chrome

Solution num 102 éditée le : 01 Août 2017 à 9h27
Par : Ognyan Dimitrov

Question demandée par : Landon Khun
Date de la demande : 23 juin 2010 à 15h00

Cette solution m'a permis de contourner l'erreur CORS et de tester mes requêtes. J'ai donc pu avancer sur mon projet.

Extrait de la solution :

For **windows** users with **Chrome Versions 60.0.3112.78 (the day the solution was tested and worked) and at least until today 24.11.2022 **(ver. 106.0.5249.119 (Official Build) (64-bit))**.
You **do not** need to close any chrome instance.

1. Create a shortcut on your desktop
2. Right-click on the shortcut and click Properties
3. Edit the Target property
4. Set it to **"C:\Program Files (x86)\Google\Chrome\Application\chrome.exe" --disable-web-security --user-data-dir="C:/ChromeDevSession"**
5. Start chrome and **ignore** the message that says --disable-web-security is not supported!

BEWARE NOT TO USE THIS PARTICULAR BROWSER INSTANCE FOR BROWSING BECAUSE YOU CAN BE HACKED WITH IT!

Traduction en français :

Pour les utilisateurs **Windows** avec les versions **Chrome 60.0.3112.78 (le jour où la solution a été testée et a fonctionné) et au moins jusqu'à aujourd'hui 24.11.2022 **(ver. 106.0.5249.119 (version officielle) (64 bits))** . Vous **n'avez pas** besoin de fermer une instance de Chrome.

1. Créez un raccourci sur votre bureau
2. Faites un clic droit sur le raccourci et cliquez sur Propriétés
3. Modifier la propriété cible
4. Réglez-le sur **« C:\Program Files (x86)\Google\Chrome\Application\chrome.exe » --disable-web-security --user-data-dir="C:/ChromeDevSession"**
5. Démarrez Chrome et **ignorez** le message indiquant que --disable-web-security n'est pas pris en charge !

ATTENTION À NE PAS UTILISER CETTE INSTANCE DE NAVIGATEUR PARTICULIÈRE POUR LA NAVIGATION CAR VOUS POUVEZ ÊTRE PIRATÉ AVEC ELLE !

Partie 4 : INFORMATIONS COMPLÉMENTAIRES

1.Autres ressources

J'ai pu également m'aider de tutoriels sur youtube. Beaucoup de développeurs y proposent des vidéos constructives et permettent de progresser.

J'ai pu être aidé sur l'erreur CORS grâce à la page "How to frontend et sa vidéo explicative :

- **How to frontend** : https://www.youtube.com/watch?v=MkiDyyBDuSE

D'autres ressources m'ont aussi beaucoup apporté, et ont ajouté de la plus-value complémentaire aux cours Studi.

- **Eloa Formation** : https://www.youtube.com/@ELOAFORMATION , propose des tutos sur les bases de données, mysql, mongo db
- **KipDev** : https://www.youtube.com/@KIPDEV33 , propose des tips et astuces dans le monde du développement
- **École du web** : https://www.youtube.com/@EcoleduWeb/videos , propose des tips et astuces sur des thèmes bien précis mais aussi des cours
- **MDN Web Docs** : https://developer.mozilla.org/fr/ , une ressource en ligne qui propose une documentation très complète sur les différentes technologies du web
- **PHP** : https://www.php.net/ , le site php propose une documentation très complète
- **Php Sandbox:** https://onlinephp.io/ , permet de s'exercer au langage php sans ide
- **Quick DBD** : https://app.quickdatabasediagrams.com/ , permet de créer une base de données en diagramme de classe de manière fluide et efficace et de l'exporter en PDF
- **Codepain** : https://codepen.io/ , permet de s'exercer sur les langages sans avoir d'ide

- **Responsinator** : http://www.responsinator.com/ , permet de vérifier le responsive de mon application

2. INFORMATIONS COMPLÉMENTAIRES

En complément, d'autres sources ont été importantes pour tester les performances de mon site. En effet il est important d'avoir un suivi permanent sur les performances de mon application. Pour cela, j'utilises divers outils en ligne gratuits qui offrent un visuel sur les résultats et permettent de savoir où il peut y avoir des disfonctionnements comme des pertes de vitesse par exemple, en voici une liste :

- **Lighthouse** : https://developer.chrome.com/docs/lighthouse/overview?hl=fr , est un outil automatisé directement intégré dans les outils de développement de google chrome, il vérifie la performance, l'accessibilité, le SEO et bien d'autres
- **GTmetrix** : https://gtmetrix.com/ , fournit une analyse détaillée des performances du site, avec des graphiques sur le temps de chargement

Afin de progresser et apprendre le Python et le Javascript par exemple, j'ai pu m'acheter certains livres :

- **Python pour les nuls,** complet et ludique
- **Javascript pour les nuls,** complet et ludique

Et une application su smartphone qui m'a permis de m'exercer au codage de manière ludique et de n'importe où :

- **Mimo : Apprendre le codage/code**

CONCLUSION

Ce projet m'a permis de progresser dans un lapse de temps assez court. J'ai pu faire face aux difficultés rencontrées. Chaque étape, était un vrai défi à réaliser, mais grâce à la prise de hauteur et à la patience, j'ai pu trouver des solutions adaptées à chacun de mes problèmes. Ma curiosité m'a permis d'aller au-delà de ce que j'espérais. L'aventure ne fait que commencer, je souhaite encore progresser et apprendre d'autres technologies et langages informatiques afin d'être le plus efficace possible et apporter des solutions digitales adaptées et performantes avec pour objectif, d'intégrer une entreprise et de faire carrière dans le développement informatique.