NOM	THERY-CAULIEZ
Prénom	Martin
Date de naissance	26/09/1998

Copie à rendre
TP – Développeur Web et Web Mobile

Documents à compléter et à rendre

Lien du git : https://github.com/THRPUB/ECF-ARCADIA-MARTIN-T
Lien de l'outil de gestion de projet : https://trello.com/b/uEcfJMdw/zoo-arcadia
Lien du déploiement : https://ecfarcadiazoo.alwaysdata.net/
Login et mot de passe administrateur : Voir "**Manuel d'utilisation**" dans le dossier "**DOCUMENT ECF**" sur Git

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**

Voir "**Manuel d'utilisation**" dans le dossier "**DOCUMENT ECF**" sur Git

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)**

Voir "**Documentation technique de l'application**" dans le dossier "**DOCUMENT ECF**" sur Git

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.**

FRONT

- Mise en place d'un système de routes pour naviguer entre les pages du site ce qui permet l'affichage de certaines pages en fonction du rôle de l'utilisateur connecté ou non.

```
/*
[] -> Tout le monde peut y accéder
["disconnected"] -> Réserver aux utilisateurs déconnecté
["ROLE_EMPLOYE"] -> Réserver aux utilisateurs avec le rôle client
["ROLE_VETERINAIRE"] -> Réserver aux utilisateurs avec le rôle vétérinaire
["ROLE_ADMIN"] -> Réserver aux utilisateurs avec le rôle admin
["ROLE_EMPLOYE", "ROLE_ADMIN"] -> Réserver aux utilisateurs avec le rôle employé OU admin
*/
```

```
//Définir ici vos routes
export const allRoutes = [
    new Route("/", "Accueil", "/pages/home.html", [], "/js/home.js"),
    new Route("/Habitats", "Habitats", "/pages/habitats.html", [], "/js/habitats.js", ),
    new Route("/Services", "Services", "/pages/services.html", [], "/js/services.js"),
    new Route("/Contact", "Contact", "/pages/contact.html", [], "/js/contact.js"),
    new Route("/signin", "Connexion", "/pages/signin.html", ["disconnected"], "/js/signin.js"),
    new Route("/signup", "Inscription", "/pages/signup.html", ["ROLE_ADMIN"], "/js/signup.js"),
    new Route("/Veterinaire", "Accès Vétérinaire", "/pages/veterinaire.html", ["ROLE_ADMIN", "ROLE_VETERINAIRE"], "/js/veterinaire.js"),
    new Route("/Messages", "Messages", "/pages/Messages.html", ["ROLE_ADMIN", "ROLE_VETERINAIRE"], "/js/Messages.js"),
    new Route("/Employe", "Accès Employé", "/pages/Employe.html", ["ROLE_ADMIN", "ROLE_EMPLOYE"], "/js/Employe.js"),
    new Route("/Admin", "Dashboard", "/pages/Administrateur.html", ["ROLE_ADMIN"], "/js/Administrateur.js"),
```

- **Sécurisation des communications**: J'utilise HTTPS pour chiffrer les communications entre le navigateur et le serveur afin de protéger les données transitant sur le réseau contre l'interception et la manipulation.

- **Validation des données côté client:** J'implémente une validation des données côté client à l'aide de JavaScript pour garantir que les données saisies par l'utilisateur sont conformes aux attentes avant même qu'elles ne soient envoyées au serveur.

FORMULAIRES :

- **Validation côté client et côté serveur**: Je m'assure que les données saisies dans les formulaires sont validées à la fois côté client (via JavaScript) et côté serveur (dans le code back-end) pour éviter les attaques par injection de code ou les données malveillantes.

- **Limitation des champs exposés**: Je m'assure que seuls les champs nécessaires sont exposés dans les formulaires, en évitant de transmettre des données sensibles ou privées qui pourraient être exploitées par des attaquants.

BACK

- Mise en place d’un système de contrôle de rôles pour contrôler L’API.

```
# Note: Only the *first* access control that matches will be used
access_control:
# - { path: ^/admin, roles: ROLE_ADMIN }
# - { path: ^/profile, roles: ROLE_USER }
            - { path: ^/api/registration, roles: ROLE_ADMIN, methods: [OPTIONS, POST] }
            - { path: ^/api/login, roles: IS_AUTHENTICATED_ANONYMOUSLY }
            - { path: ^/api/doc, roles: ROLE_ADMIN }
            - { path: ^/animal, roles: ROLE_ADMIN}
            - { path: ^/habitat, roles: ROLE_ADMIN}
            - { path: ^/services, roles: ROLE_ADMIN, ROLE_EMPLOYE, methods: [OPTIONS, POST, GET, PUT] }
            - { path: ^/api/visite-veterinaire, roles: ROLE_VETERINAIRE, methods: [OPTIONS, POST, GET, PUT] }
            - { path: ^/api/horraires, roles: ROLE_ADMIN, methods: [OPTIONS, POST, GET, PUT] }
            - { path: ^/api/avis/habitats, roles: ROLE_VETERINAIRE, methods: [OPTIONS, POST, GET, PUT] }
            - { path: ^/api/passage-employe, roles: ROLE_EMPLOYE, methods: [OPTIONS, POST, GET, PUT] }
            - { path: ^/api/pagecontacts, roles: ROLE_EMPLOYE, ROLE_ADMIN, methods: [OPTIONS, POST, GET, PUT] }
            - { path: ^/api/avis, roles: IS_AUTHENTICATED_ANONYMOUSLY }
```

- **Protection contre les injections SQL:** J'utilise des requêtes préparées ou des ORM (Object-Relational Mapping) pour interagir avec la base de données de manière sécurisée et éviter les injections SQL.

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

Lorsque je mène une veille technologique sur les vulnérabilités de sécurité pour les développeurs web, voici les étapes que je prends généralement :

- **Sources d'information:** Je parcours régulièrement les sites Web spécialisés dans la sécurité informatique tels que OWASP (Open Web Application Security Project).

- **Alertes de sécurité:** J'utilise des outils et des services qui fournissent des alertes en temps réel sur les vulnérabilités, telles que des flux RSS, des notifications par e-mail.

- **Analyse de code:** J'utilise des outils d'analyse statique et dynamique de code pour détecter les vulnérabilités potentielles dans le code source des applications web, en mettant l'accent sur les problèmes de sécurité courants tels que les injections SQL, les attaques XSS (Cross-Site Scripting), et les problèmes liés à l'authentification et à l'autorisation. **(J’utilise SONARLINT).**

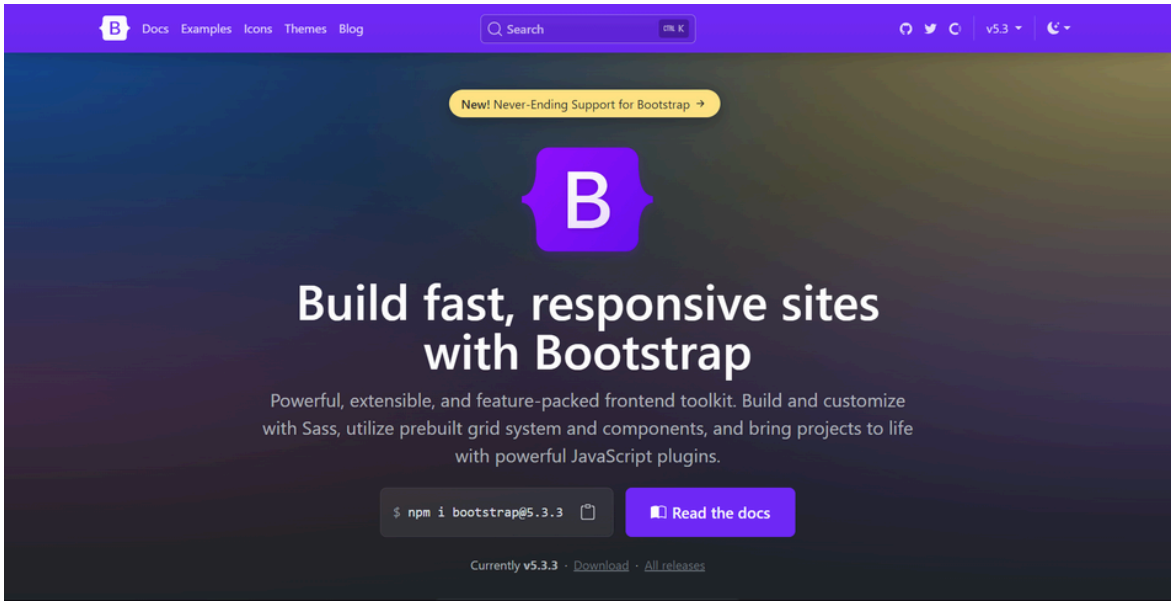
Partie 3 : Recherche

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

Pendant le projet, j'ai dû intégrer Bootstrap dans Visual Studio Code pour améliorer l'interface utilisateur. En suivant les étapes claires de cet article, j'ai pu installer Bootstrap via npm dans notre projet VSCode sans rencontrer de problèmes, économisant ainsi du temps et facilitant la progression dans le projet.

https://getbootstrap.com/

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



- Avoir déjà NodeJs d'installer sur son projet
- Faire la commande npm i bootstrap@5.3.3 pour installer la version 5.3.3 de Bootstrap
- Créez un nouveau index.html fichier à la racine de votre projet. Inclure <meta name="viewport">
- Importation des liens dans mon fichier Index.html pour le JS de Bootstrap

```
<script type="module" src="Router/Router.js"></script>
<script src="node_modules/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
```

- Ajout des lien css dans mon fichier main.SCSS

Partie 4 : Informations complémentaire

1. **Autres ressources**
2. **Informations complémentaires**

Vous pouvez retrouver toutes les autres documentations que j'ai faites pour ce projet dans mon Git dans le dossier **"DOCUMENTATION ECF"**