NOM	ALABAU
Prénom	Philippe
Date de naissance	23/08/1996

Copie à rendre

TP – Développeur Web et Web Mobile

Documents à compléter et à rendre

Lien du git : https://github.com/PhilAlb/ECF-2024-Arcadia
Lien de l'outil de gestion de projet : https://zoo-arcadia.atlassian.net/jira/core/projects/ZA/board
Lien du déploiement : NA (pas déployée)
Login et mot de passe administrateur : NA (pas d'espace admin)

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**

Le projet est le site vitrine pour un zoo nommé Arcadia.
Ce projet a pour but de tester nos compétences sur le front et le backend d'une application web.
Le zoo propose plusieurs services qui pourront être évolutifs à ses clients :

- Une restauration
- Une visite guidée
- Une visite du zoo en petit train

De plus, le zoo a une grande variété d'animaux et plusieurs habitats pour leur bien-être :

- La savane
- La jungle
- Le marais

Dans un premier temps j'ai dû créer les maquettes, le wireframe pour toutes mes pages en version desktop et mobile.
Puis j'ai dû créer le template de mon site, de mes composants header, footer et de mes différentes pages.
Mon objectif dans ce projet a été de factoriser au maximum mon code. Je dispose de plusieurs éléments de type cartes : il m'a alors été facile de créer une classe Card générale puis de l'étendre afin de l'adapter à mes besoins qu'elles concernent les services proposés par le zoo ou bien les différents habitats.
Ensuite concernant la partie backend j'ai créé l'API .NET et j'ai créé la connexion entre mon frontend et le back de mon application.

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

Le projet disposait de 11 User stories :

- Une page d'accueil qui présente les services et les habitats du zoo ainsi que les avis des visiteurs présents sur la page d'accueil
- Un menu de navigation pour l'application notamment avec l'accueil, les habitats, les services, un bouton contact pour communiquer avec le personnel du zoo, un bouton connexion pour accéder aux espaces admin, employés et vétérinaires.
- Une page services qui présente en détail les services dans le zoo avec un titre, une description et les horaires d'ouverture
- Une page habitats qui présente les différents habitats du zoo et qui, à l'aide d'un clic sur l'habitat ouvre une interface qui présente les animaux vivants dans ces habitats. L'état de l'animal renseigné par le vétérinaire doit apparaître sur cette page pour être accessible pour les visiteurs
- Les visiteurs peuvent ajouter des commentaires sur le zoo qui seront validés à posteriori par un employé afin d'être affiché sur la page d'accueil du zoo
- L'espace administrateur permettra de gérer les services, les horaires, les habitats et les animaux du zoo. Il permet de créer des comptes employés et vétérinaires grâce à l'adresse mail et le mot de passe généré. Il aura aussi accès à un dashboard visualisant le nombre de consultation de chaque animal sur le site
- L'espace employé permet de valider les avis envoyés par les visiteurs et de renseigner la nourriture donnée à chaque animal pour permettre le suivi par les vétérinaires
- L'espace vétérinaire permet de renseigner des comptes-rendus pour les animaux ainsi que déposer un commentaire sur l'état des habitats et l'historique de la nourriture donnée pour chaque animal
- Un système de connexion accessible depuis la page d'accueil devra être mis en place pour les employés, les vétérinaire et l'administrateur
- Les visiteurs pourront prendre contact avec un employé du zoo qui pourra y répondre
- Enfin, lorsqu'un visiteur clique sur un animal pour avoir des détails le concernant dans sur la page habitats une incrémentation cachée augmentera pour pouvoir être affiché du côté dashboard administrateur et lui permettre de savoir quel animal est le plus populaire

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**

Je suis actuellement employé dans une ESN en tant que testeur sur un projet Angular .NET. Depuis plusieurs mois je suis une formation en interne pour pouvoir me former sur ces technologies et évoluer sur un poste de développeur. J'y ai donc vu l'opportunité de mettre mon apprentissage à l'emploi lors du choix des technologies pour ce projet.

L'utilisation de l'angular m'a permis de factoriser mon code au maximum en créant des composants pour les pages, le header, le footer, les cartes et le carrousel.

Pour le .Net j'ai utilisé cette technologie car c'est la technologie back que je maîtrise le mieux à l'heure actuelle. N'ayant pas pu mettre en place de BDD j'ai décidé d'utiliser des requêtes http pour récupérer les différents éléments des modèles à afficher sur le front.

Pour la base de données SQL je pense utiliser SQL Server et réaliser mes diagrammes de classes, d'utilisation et de séquences plus tard.

Pour le déploiement je pense utiliser Azure mais je n'ai pas trouvé un moyen gratuit.

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

J'ai utilisé git hub et voici l'arborescence en branches mon projet que j'ai pris comme habitude pour travailler :

- Une branche main
- Une branche développement
    - Une branche feature / header
    - Une branche feature / footer
    - Une branche feature / home-page
    - Une branche feature / services-page
    - Une branche feature / habitats-page

Tout a été testé et mergé sur la branche main et les autres branches ont été supprimées.

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.**

Je n'ai pas encore pu intégrer des formulaires mais je cacherai les informations sensibles comme le mot de passe et je limiterai au maximum les informations demandées pour envoyer un commentaire ou une demande de contact et j'utiliserai une méthode POST pour ces mêmes informations.

Je compte passer le site sur un protocole HTTPS.

Pour les formulaires je compte aussi contrôler la nature des informations saisies par exemple pour une date seule une entrée en format « date » pourra être validée.

Les mots de passe créés devront être fort avec des majuscules, un certain nombre de caractères, des caractères spéciaux et des caractères numériques. Et au moment de la connexion le compte sera bloqué après 3 échecs.

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

Pour le moment, sans déploiement mon site n'est pas concerné par des vulnérabilités de sécurité mais j'ai d'ores et déjà réalisé des recherches à propos des risques encourus sur le frontend et le backend de l'application.

L'injection de code SQL, la bruteforce, les vols de données, les clés d'authentification,

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**

Pour le paramétrage du routing de l'application j'ai utilisé la documentation d'Angular :

https://angular.dev/guide/routing/common-router-tasks

J'ai pu apprendre la commande pour créer le routing de l'application ainsi que les formules à utiliser pour définir les routes et créer la communication entre l'API et l'application frontend.

J'ai aussi utilisé stackoverflow ainsi que la documentation bootstrap et MDN (pour le CSS).

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

   https://angular.dev/guide/routing/common-router-tasks

Définir une route de base

Il y a trois éléments fondamentaux pour créer une route.
Importez les routes dans app.config.ts et ajoutez les à la fonction provideRouter
Voici la configuration par défaut de l'application en utilisant le CLI.

```
export const appConfig: ApplicationConfig = {
  providers: [provideRouter(routes)]
};
```

Le CLI Angular effectue cette étape pour vous. Cependant, si vous créez une application manuellement ou travaillez avec une application existante sans CLI, assurez-vous que les imports et la configuration sont corrects.

1 Configurer un tableau Routes pour vos routes

Le CLI Angular effectue cette étape automatiquement.

```
import { Routes } from '@angular/router';
export const routes: Routes = [];
```

2 Définir vos routes dans votre tableau Routes

Chaque route dans ce tableau est un objet JavaScript qui contient deux propriétés. La première propriété, path, définit le chemin URL pour la route. La seconde propriété, component, définit le composant qu'Angular doit utiliser pour le chemin correspondant.

```
const routes: Routes = [
  { path: 'first-component', component: FirstComponent },
  { path: 'second-component', component: SecondComponent },
];
```

3 Ajouter vos routes à votre application

Maintenant que vous avez défini vos routes, ajoutez-les à votre application. Tout d'abord, ajoutez des liens vers les deux composants. Assignez la balise d'ancrage à laquelle vous souhaitez ajouter la route à l'attribut routerLink. Définissez la valeur de l'attribut sur le composant à afficher lorsqu'un utilisateur clique sur chaque lien. Ensuite, mettez à jour votre modèle de composant pour inclure <router-outlet>. Cet élément informe Angular de mettre à jour la vue de l'application avec le composant pour la route sélectionnée.

```
<h1>Angular Router App</h1>
<nav>
 <ul>
  <li><a routerLink="/first-component" routerLinkActive="active"
ariaCurrentWhenActive="page">First Component</a></li>
  <li><a routerLink="/second-component" routerLinkActive="active"
ariaCurrentWhenActive="page">Second Component</a></li>
 </ul>
</nav>
<!-- The routed views render in the <router-outlet>-->
<router-outlet></router-outlet>
```

Vous devez également ajouter RouterLink, RouterLinkActive et RouterOutlet au tableau des imports de AppComponent.

```
@Component({
 selector: 'app-root',
 standalone: true,
 imports: [CommonModule, RouterOutlet, RouterLink,
RouterLinkActive],
 templateUrl: './app.component.html',
 styleUrls: ['./app.component.css']
})
export class AppComponent {
 title = 'routing-app';
}
```

Partie 4 : Informations complémentaire

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