Projet ECF Zoo Arcadia

Ameur ouafi
Formation développeur fluter
2023/2024

Lien des dépôts

Dépôt GitHub : Toute la documentation technique, le code source et les instructions pour déployer l'application en local sont disponibles sur notre GitHub. Ce dépôt est régulièrement mis à jour pour refléter les dernières améliorations et corrections apportées à notre application.
https://github.com/Ameut/Arcadia_zoo

Site Web Déployé : Pour une immersion directe, visitez le site web déployé à l'adresse suivante : https://ameur24.pythonanywhere.com/. Ce site vous permettra de visualiser en temps réel
les habitats des animaux ainsi que d'autres informations pertinentes sur le zoo.

Gestion de Projet Trello : Pour suivre l'avancement du projet et les différentes tâches en cours, consultez notre tableau Ce tableau
offre une vue d'ensemble des activités planifiées et en cours.
visible sur le lien : https://trello.com/b/oWhMS9BU/zooarcadia-ameur

Résumé du projet Zoo Arcadia

Le projet consiste à développer une application web pour le zoo Arcadia en utilisant Django. Cette application permet aux visiteurs de visualiser les animaux, leurs états, les services et les horaires du zoo. Le développement suit une méthodologie agile pour cela j'ai utilisé Trello, en utilisant un kanban pour organiser les tâches.

Déroulement du projet :
Présentation de l'application : Inclure des identifiants pour les parcours possibles.
Charte graphique : Palette de couleurs et polices, wireframes et maquettes.
Documentation de gestion de projet : Explication de la gestion agile.
Documentation technique : Configuration de l'environnement, modèle conceptuel des données, diagrammes d'utilisation et de séquence, documentation du déploiement.

Concernant le codage : j'ai utilisé la documentation officielle de Django pour l'installation sur vs code. D'autre par sur les éléments d'intégration html css, je me suis appuyé sur les cours de Study et sur la documentation de W3C.
L'intégration back end, je me suis basé sur plusieurs support notamment les cours de code avec Jonathan sur YouTube et d'autre support notamment la documentation de python.
En résumé le projet donne un aperçu d'un site dynamique avec les intégrations js et d'une BD relationnel et no SQL, qui permet une interaction avec l'utlisateur.

Arcadia Zoo Read me

Description :
Arcadia Zoo est une application web conçue pour améliorer l'interaction des visiteurs en fournissant des informations détaillées sur les habitats des animaux et les services du zoo. L'application met l'accent sur les thèmes écologiques, reflétant l'engagement du zoo envers la durabilité.

Aperçu du projet :
L'application offre plusieurs espaces pour les visiteurs, les administrateurs, les vétérinaires et les employés, fournissant un accès personnalisé aux informations et fonctionnalités du zoo. Les espaces utilisateurs comprennent :

- Espace Visiteur :
  - Consultation des habitats et animaux du zoo
  - Affichage des services disponibles
  - Soumission d'avis sur les animaux et services
  - Page de contact pour les demandes et questions

- Espace Administrateur :
  - Gestion des comptes (employés, vétérinaires)
  - Gestion des services, habitats et animaux
  - Tableau de bord des statistiques de consultation

- Espace Employé :
  - Validation des avis des visiteurs
  - Gestion des services du zoo
  - Suivi de l'alimentation quotidienne des animaux

- Espace Vétérinaire :
  - Gestion des comptes rendus médicaux des animaux
  - Commentaires sur les habitats
  - Suivi alimentaire des animaux

Identifiants de Connexion

- Admin :
  - Username : `admin`
  - Mot de passe : `loulou25`

- Vétérinaire :
  - Username : `veterinaire`
  - Mot de passe : `zoovetoarcadia`

- Employé :
  - Username : `employe`
  - Mot de passe : `zooarcadia`
    La page rapport vétérinaire est protéger par un login, seul
    les personnes qui peuvent se connecter au paneau de configuration peuvent y accéder.

Pile Technologique:

- Frontend : HTML, CSS (Bootstrap), JavaScript
- Backend : Django (Python)
- Bases de données :
  -Relationnelle : SQLite3
  -NoSQL :MongoDB
- Contrôle de version : Git

Configuration Locale
Cloner le dépôt :

git clone https://github.com/Ameut/Arcadia_zoo.git
cd Arcadia_zoo
Installer les dépendances :

pip install -r requirements.txt
Configurer les bases de données :

SQLite est configurée par défaut avec Django.
Pour MongoDB, assurez-vous que MongoDB fonctionne localement ou connectez-vous à une instance MongoDB Atlas.

Étapes pour se Connecter avec MongoDB Compass

Téléchargez MongoDB Compass depuis le site officiel :
https://www.mongodb.com/try/download/compass
Installer MongoDB Compass :
Suivez le tutoriel d'installation pour Windows, Mac ou Linux :
https://www.mongodb.com/docs/compass/current/install/

Lancer les migrations :

python manage.py migrate

Créer un superutilisateur :

python manage.py createsuperuser

Démarrer le serveur de développement :
bash
Copy code

```
python manage.py runserver
Accédez a http://127.0.0.1:8000

Pour toute question, veuillez contacter ameur.ouafi@hotmail.fr ou
ouafiameur@gmail.com.
```

Gestion de projet

Liens trello : https://trello.com/b/oWhMS9BU/zooarcadia-ameur
Pour gérer efficacement le développement de l'application web pour le zoo Arcadia en utilisant Django avec des bases de données SQLite3 et MongoDB pour les statistiques des clics sur les animaux.

Analyse des Besoins et Planification :
Identification des besoins précis, suivant la documentation et des utilisateurs finaux.
Création d'un backlog de fonctionnalités basées sur les user stories fournies dans le document.
Priorisation des tâches et planification des scripts dans un outil de gestion de projet (ex : Jira, Trello).

Configuration de l'Environnement de Développement :
Installation et configuration de Django.
Mise en place des deux bases de données : SQLite3 pour la gestion des données relationnelles (utilisateurs, rôles, permissions) et MongoDB pour stocker les statistiques des clics sur les animaux.

Développement et Intégration :
Développement front-end : HTML5, CSS (Bootstrap), JavaScript pour créer des interfaces réactives.
Développement back-end : Django pour la logique métier, l'interaction avec la base de données, et la gestion des requêtes HTTP.
Intégration continue pour tester et déployer les fonctionnalités au fur et à mesure de leur développement.

Tests :
Tests unitaires et d'intégration pour s'assurer que chaque composant fonctionne correctement seul et en interaction avec les autres.
Tests d'interface utilisateur pour garantir que l'expérience utilisateur est conforme aux attentes.
Déploiement et Maintenance :
Déploiement de l'application sur une plateforme cloud (ex : Azure, python Anywere ect…).
Surveillance et maintenance de l'application pour s'assurer de sa stabilité et de sa performance.

Réflexion sur le Choix de la Technologie

Pour développer l'application web du zoo Arcadia, nous avons dû sélectionner les technologies qui correspondaient aux exigences du projet, notamment l'affichage des animaux, des services et des avis pour les visiteurs, et les fonctionnalités de gestion pour les vétérinaires, employés et administrateurs. Voici les principales raisons du choix des technologies utilisées.

Technologies Choisies :

Back-End: Django (Framework Web Python)

Pourquoi Django:

Rapidité de Développement: Offre une architecture solide (MTV) Model-View-Template et permet de créer rapidement des applications complexes.

Sécurité: Intègre par défaut des mécanismes de protection contre les vulnérabilités courantes.

ORM Intégré: Gère facilement les bases de données relationnelles.

Admin Généré Automatiquement: Facilite la gestion du contenu et des utilisateurs.

Configuration:

ASGI/WSGI: asgi.py et wsgi.py pour l'asynchronisme et le déploiement.

Routing: urls.py pour définir les routes principales.

Paramètres Globaux: settings.py pour la configuration des bases de données, fichiers statiques, etc.

Bases de Données: SQLite3 (Relationnel) et MongoDB (NoSQL)

Pourquoi SQLite3:

Légère et Simple: Convient bien au développement local et à des applications de petite échelle.

Intégrée: Fournie par défaut avec Django.

Pourquoi MongoDB:

Scalabilité: Supporte un grand volume de données non structurées.

Flexibilité: Permet de stocker les statistiques de consultation des habitats dans un format JSON.

Front-End: HTML5, CSS (Bootstrap), JavaScript

HTML5/CSS/JavaScript:

Compatibilité :

Normes standardisées pour créer des interfaces utilisateur dynamiques.

Bootstrap: Fournit un design réactif et adapté aux mobiles.

Déploiement et Gestion de Projet

Simple et efficace, nécessite uniquement un fichier Procfile.

Gestion de Projet avec GitHub et Trello:

GitHub: Branches main et dev pour la gestion des versions.

Trello: Suivi des tâches et organisation du projet en colonnes Kanban.

Concernant le déploiement, nous opterons pour pythonanywere pour sa compatibilité avec python.

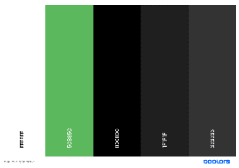
Le choix de Django, SQLite3, MongoDB et Bootstrap permet de créer une application web évolutive et rapide à développer, tout en répondant aux besoins spécifiques du projet du zoo Arcadia.

Charte graphique pour le projet zoo Arcadia

Logo :

Le logo doit avoir une connotation écologique et du bien-être animal.

Palette de couleurs :



Les teintes dominantes de vert évoquent la nature et la durabilité, pertinentes pour un zoo. L'utilisation de tons noir et blanc donne un coté sobre n'agresse pas l'utlisateur, qui visualisé et se focaliser sur les éléments essentiels.

Typographie :

Des polices sans empattement modernes pour les titres et le corps du texte contribuent à la clarté et à l'accessibilité. La hiérarchie typographique est bien définie, facilitant la navigation et la lecture.

```
font-family: 'Open Sans', sans-serif;
```

UI Design :

Les éléments interactifs comme les boutons et les icônes doivent être conçus pour être intuitifs et facilement identifiables. L'espacement et le placement des éléments suivent un agencement logique qui guide l'utilisateur naturellement à travers l'information.

Design émotionnel :

L'utilisation d'images de grande qualité des animaux, comme celles des lions, girafs ect, dans l'image fournie, aide à établir une connexion émotionnelle avec les visiteurs du site, le carrousel en javascript avec bootsarpt donne un coté dynamique.

Responsivit

La charte graphique doit garantir une expérience cohérente sur tous les appareils, mobiles comme desktop.

Accessibilité :

Une attention particulière sera accordée à l'accessibilité, garantissant que le site soit utilisable par tous, y compris les personnes en situation de handicap.

Description technique

Ce projet consiste en la création d'une application web pour le zoo Arcadia en utilisant Django. La base de données relationnelle utilisée est SQLite3 et la base NoSQL MongoDB. L'application permet aux visiteurs de consulter les animaux et les services du zoo, tandis que les vétérinaires, employés et administrateurs ont leurs propres espaces pour gérer les opérations du zoo.

Structure du Projet
Dossier zooArcadia/:
__init__.py: Indique que le dossier est un module Python.
asgi.py: Fichier ASGI pour l'asynchronisme et le déploiement.
settings.py: Contient les configurations globales du projet:
Bases de données (SQLite3 pour le relationnel, MongoDB pour le NoSQL).
Applications installées.
Fichiers statiques et médias.
urls.py: Définit les routes principales de l'application.
wsgi.py: Point d'entrée WSGI pour le déploiement.
Autres Fichiers:
manage.py: Utilitaire de ligne de commande pour gérer les tâches administratives du projet.
db.sqlite3: Base de données relationnelle pour le développement local.
README.md: Contient des informations générales sur le projet et sa configuration.
requirements.txt: Liste des dépendances Python nécessaires au projet.
Dossier liste/:
Contient les modèles, vues et templates spécifiques à chaque fonctionnalité.
Base de Données
SQLite3 (Relationnel):
Modèle de Données:
Utilisateurs: Administrateurs, Employés, Vétérinaires, Visiteurs.
Services: Nom, description.
Habitats: Nom, images, description, animaux associés.
Animaux: Prénom, race, images, habitat.

Avis: Pseudo, texte, validation.

Alimentation: Date, heure, nourriture donnée, quantité.

Configuration dans settings.py:

python

Copy code

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / "db.sqlite3",
    }
}
```

MongoDB (NoSQL):

Utilisé pour les statistiques de consultation des habitats.

Connexion via PyMongo:

python

Copy code

```
from pymongo import MongoClient

client = MongoClient('mongodb://localhost:27017/')
db = client['arcadia_stats']
collection = db['consultations']
```

Déploiement

Configuration Git:

Branches:

main: branche principale.

develop: branche de développement.

Fichier README.md:

Contient les étapes pour installer et exécuter l'application localement.

Veuille technologique des vulnérabilités

Lors de la réalisation de mon projet sur Django, j'ai effectué une veille technologique sur les vulnérabilités de sécurité afin de m'assurer que l'application soit sécurisée. Voici un résumé des points clés de cette veille :

1. Correctifs de Sécurité

Des correctifs de sécurité cruciaux ont été publiés pour Django 4.2.7, 4.1.13 et 3.2.23, visant à résoudre des vulnérabilités potentielles qui pouvaient être exploitées par des attaquants pour altérer le fonctionnement de l'application.

2. Vulnérabilités Courantes

Les vulnérabilités courantes dans les applications Django incluent l'injection SQL, le Cross-Site Scripting (XSS), et le Cross-Site Request Forgery (CSRF). Ces failles permettent à des attaquants d'exécuter des commandes malveillantes ou de voler des informations sensibles.

3. Pratiques de Sécurité Recommandées

Mises à jour régulières : Il est crucial de maintenir Django à jour avec les dernières versions pour bénéficier des correctifs de sécurité.

Utilisation des Middlewares de Sécurité :

Django fournit des middlewares comme SecurityMiddleware pour renforcer la sécurité en activant des protections supplémentaires.

Validation des Entrées Utilisateur : Utiliser les mécanismes intégrés de Django pour valider et nettoyer les données entrantes afin de prévenir les injections et autres attaques.

Configurations de Sécurité : Configurer correctement les paramètres de sécurité comme SECURE_SSL_REDIRECT, CSRF_COOKIE_SECURE, et X_FRAME_OPTIONS dans settings.py.

4. Surveillance des Vulnérabilités :
Effectuer une veille régulière des vulnérabilités publiées par des sources fiables comme le CERT-FR et OWASP. Cela permet de rester informé des nouvelles failles et de mettre en œuvre les mesures correctives nécessaires

Cette veille technologique m'a permis de sécuriser mon projet Django en adoptant des pratiques et outils appropriés pour prévenir et détecter les vulnérabilités.

Modèle Conceptuel de Données (MCD)

Entité Animal :
Attributs : ID, Prénom, Race, État, Nourriture, Grammage, Date de passage.
Relations : Appartient à un Habitat, Suivi par un Vétérinaire.

Entité Habitat :
Attributs : ID, Nom, Description.
Relations : Contient des Animaux.

Entité Vétérinaire :
Attributs : ID, Nom, Email.
Relations : Suit des Animaux, Remplit des Rapports.

Entité Service :
Attributs : ID, Nom, Description.
Relations : Proposé par le Zoo, Modifiable par Administrateur.

Entité Employé :
Attributs : ID, Nom, Email, Rôle (Admin, Vétérinaire, Employé standard).
Relations : Gère des Services, Valide des Avis.

Entité Avis :
Attributs : ID, Pseudo, Texte, Statut (Validé/Non validé).
Relations : Soumis par Visiteur, Validé par Employé.
Diagramme de Classe UML

Classe Animal :
Attributs et méthodes pour gérer les détails de l'animal et son suivi vétérinaire.

Classe Habitat :
Attributs et méthodes pour gérer les informations sur les habitats et lister les animaux qu'ils contiennent.

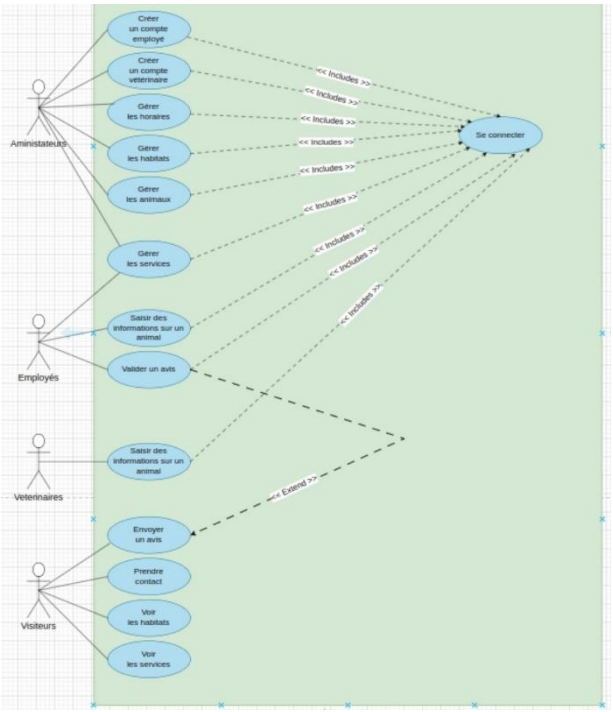
Classe Vétérinaire :
Attributs et méthodes pour gérer les informations personnelles du vétérinaire et ses interactions avec les animaux.
Classe Service :

Attributs et méthodes pour gérer les services offerts par le zoo, incluant les modifications par les administrateurs.
Classe Employé :
Attributs et méthodes pour gérer les informations de l'employé, ses privilèges et ses interactions avec les services et les avis.

Classe Avis :
Attributs et méthodes pour gérer les avis laissés par les visiteurs et leur validation par les employés.

Diagramme d'Utilisation :

Roles : Visiteur, Employé, Vétérinaire, Administrateur



Cas d'utilisation pour Visiteur :
Consulter la page d'accueil
Naviguer dans le menu
Voir les services et habitats
Laisser un avis
Consulter les détails d'un animal
Contacter le zoo

Cas d'utilisation pour Employé :
Valider ou invalider un avis
Modifier les services
Ajouter une consommation de nourriture pour un animal

Cas d'utilisation pour Vétérinaire :
Remplir les comptes rendus pour les animaux
Commenter sur les habitats
Visualiser l’historique alimentaire des animaux

Cas d'utilisation pour Administrateur :
Gérer les comptes utilisateurs
Modifier les services, habitats et animaux
Voir les comptes rendus des vétérinaires
Gérer les consultations et statistiques des animaux
Diagramme de Séquence
Scénario : Un visiteur consulte les détails d'un animal

Étapes :

Le visiteur sélectionne un habitat dans le menu.

Le système affiche la liste des animaux dans cet habitat.

Le visiteur clique sur un animal pour voir ses détails.

Le système récupère les informations de l'animal depuis la base de données.

Les détails de l'animal, y compris le dernier rapport vétérinaire, sont affichés.

Le visiteur peut ensuite naviguer vers d'autres sections ou laisser un avis.

Sécurité du framawork Django

Pour garantir la sécurité de l'application du zoo Arcadia, nous avons pris des dispositions spécifiques concernant l'authentification, la gestion des formulaires, la protection contre les attaques CSRF et le hachage des mots de passe. Voici une explication détaillée de nos choix techniques.

Sécurité dans Django :

Authentification (Django Auth)

Django offre un système d'authentification intégré pour gérer les utilisateurs et les groupes.

Système d'Authentification:

Modèle User: Fournit un ensemble de champs tels que username, email, password, etc.

Groupes :

Permet d'organiser les utilisateurs avec des permissions spécifiques.

Permissions: Contrôle l'accès aux différentes fonctionnalités.

Formulaires d'Authentification:

Formulaire de Connexion :

python

```
from django.contrib.auth.forms import AuthenticationForm
form = AuthenticationForm(request, data=request.POST)
```

Formulaire d'Inscription :

python

```
from django.contrib.auth.forms import UserCreationForm
form = UserCreationForm()
```

Gestion des Mots de Passe

Hachage des Mots de Passe:

Django utilise un système de hachage basé sur PBKDF2 avec SHA256.

Le mot de passe est stocké sous la forme suivante:
shell
hasher$salt$hash

Exemple :

python
from django.contrib.auth.hashers import make_password, check_password
password = make_password('motdepasse123')
check = check_password('motdepasse123', password)
Protection CSRF
Django protège automatiquement les formulaires contre les attaques Cross-Site Request Forgery (CSRF).

Protection par Décorateur:
@csrf_exempt: Exempte une vue spécifique de la protection CSRF.
@csrf_protect: Applique explicitement la protection CSRF.
python
from django.views.decorators.csrf import csrf_exempt

@csrf_exempt
def vue_sans_csrf(request):
    return HttpResponse("CSRF exemptée")
Gestion des Permissions

Décorateurs :

@user_passes_test: Restreint l'accès à une vue si le test conditionnel échoue.
@login_required: Redirige vers la page de connexion si l'utilisateur n'est pas authentifié.

Exemples :

@login_required:
python
from django.contrib.auth.decorators import login_required

@login_required

```
def vue_protegee(request):
    return HttpResponse("Vue accessible uniquement aux utilisateurs connectés")
@user_passes_test:
python
from django.contrib.auth.decorators import user_passes_test

def est_admin(user):
    return user.is_staff

@user_passes_test(est_admin)
def vue_admin(request):
    return HttpResponse("Vue accessible uniquement aux administrateurs")
```

En utilisant les fonctionnalités intégrées de Django pour l'authentification, la protection CSRF et le hachage des mots de passe, nous avons renforcé la sécurité de l'application du zoo Arcadia. L'utilisation judicieuse des décorateurs tels que @csrf_exempt, @user_passes_test et @login_required nous a permis d'assurer un contrôle d'accès sécurisé.

Déploiement sur PythonAnywhere

Dans le cadre de notre projet pour le zoo Arcadia, nous avons développé une application web en utilisant Django avec SQLite3 pour la gestion des données relationnelles et MongoDB pour les données non relationnelles via MongoDB Atlas. Notre objectif est de déployer cette application sur PythonAnywhere tout en intégrant la base de données MongoDB Atlas.

Étapes de déploiement sur PythonAnywhere

1. Préparation de l'environnement Django

Après avoir créé un compte sur PythonAnywhere, procédez comme suit pour configurer votre environnement Django:

Lancez une console Bash et créez un environnement virtuel :

bash

Copier le code

```
mkvirtualenv monenvdjango --python=python3.8
```

Installez Django dans cet environnement :

bash

Copier le code

```
pip install django
```

2. Configuration de la base de données SQLite3

python

Copier le code

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}
```

3. Intégration de MongoDB Atlas

Pour utiliser MongoDB avec Django, installez djongo :

pip install djongo

Configurez la base de données MongoDB dans settings.py :

python

Copier le code

```
DATABASES = {
    'default': {
        'ENGINE': 'djongo',
        'NAME': 'nom_de_votre_base',
        'ENFORCE_SCHEMA': False,
        'CLIENT': {
            'host': '_Chaine_De_Connection_Atlas'
        }
    }
}
```

Remplacez 'la_Chaine_De_Connection_compass' par la chaîne de connexion fournie par MongoDB Atlas.

4. Déploiement de l'application

Téléchargez le code source sur PythonAnywhere en utilisant FTP ou en clonant depuis votre dépôt Git. Configurez le fichier WSGI sur PythonAnywhere pour pointer vers votre application Django. Activez votre application via l'onglet Web de PythonAnywhere et rechargez votre application web.

5. Vérifications finales

Visitez l'URL fournie par PythonAnywhere pour voir l’application en action.

Testez toutes les fonctionnalités pour vous assurer que la connexion à MongoDB fonctionne correctement et que toutes les pages chargent comme prévu.