Gestion de Projet - Application Zoo Arcadia

1. **Lecture du cahier des charges**
    - **Étape initiale** : Lire le cahier des charges en entier pour comprendre les besoins et les attentes du projet.

2. **Création de briefs par utilisateurs avec Figjam (annexe 1)**
    - **Utilisation de Figjam** : Utiliser l'outil Figjam pour découper le cahier des charges en tâches plus petites et compréhensibles.
    - **Simplification** : Simplifier le cahier des charges pour créer des briefs spécifiques à chaque type d'utilisateur (visiteurs, administrateurs, vétérinaires, employés).

3. **Mise en place de l'architecture de l'application avec Figjam (annexe 2)**
    - **Architecture initiale** : Définir l'architecture de l'application basée sur les briefs.
    - **Planification technique** : Identifier les technologies et les outils nécessaires pour chaque composant de l'application.

4. **Mise en place d'un exemple de data Structure avec Figjam (annexe 3)**
    - **Strucure initiale** : Définir la hiérarchie initiale des tables et leurs propriétés en suivant les briefs (Sql et NoSql)
    - **Planification technique** : Identifier les technologies et les outils nécessaires pour chaque composant de l'application.

5. **Mise en place de Jira pour la méthode SCRUM (agile)**
    - **Création du projet dans Jira**
        - **Objectifs et livrables** : Définir les principaux objectifs et livrables du projet dans Jira.

- **Création des User Stories (US) et des Epics**
  - **Backlog** : Créer toutes les User Stories (US) dans le backlog en utilisant les briefs.
  - **Subtasks** : Créer des sous-tâches (subtasks) pour chaque US.
  - **Epics** : Définir les Epics pour regrouper les US par fonctionnalité majeure.

- **Vérification du backlog**
  - **Validation** : Vérifier que toutes les US et subtasks sont conformes aux briefs et à l'architecture de l'application.

- **Mise en place des sprints**
  - **Planification des sprints** : Utiliser le backlog pour organiser les sprints de manière à commencer par la page d'accueil pour les visiteurs, puis passer à la partie tableau de bord, et terminer par les statistiques avec MongoDB.
  - **Détails des sprints** : Chaque sprint dure une semaine, avec des dates de début et de fin, des descriptions détaillées pour chaque US et subtask, et des responsables désignés.
  - **Vérification du backlog** : Le backlog doit être vide suite à la mise en place des sprints.

6. **Lancement et suivi des sprints**
   - **Lancement du sprint 1**
     - **Organisation** : Lancer le premier sprint une fois le backlog et les sprints prêts.
     - **Suivi** : Suivre l'avancement du projet, commenter en cas de retard, de difficulté ou autre et maintenir un historique propre.

   - **Gestion des tâches et sous-tâches**
     - **Colonnes** : Utiliser les colonnes "à faire", "en cours", et "terminée" pour suivre l'état des tâches. Chaque tâche démarre dans « à faire », toute tâche commencée doit basculer dans « en cours », une fois accomplie elle passe à « terminée »
     - **Simplicité** : Ne pas ajouter de colonne "en développement" supplémentaire pour simplifier la gestion et éviter les erreurs. En

effet, une tâche en développement ou merge est soit « en cours » soit « terminée »

7. **Mise en place de l'UI/UX sur Figma**
    - **Conception UI/UX** : Après la fin des sprints de planification, j'ai utilisé Figma pour concevoir l'interface utilisateur (UI) et l'expérience utilisateur (UX) de l'application.
    - **Validation** : Vérifier que les maquettes correspondent au cahier des charges et au Figjam

8. **Développement**
    - **Début du développement** : Après la validation des maquettes UI/UX, j'ai commencé le développement de l'application en suivant les spécifications et l'architecture définies.
    - **Intégration continue** : Utilisation de pratiques d'intégration continue pour tester et déployer les fonctionnalités développées de manière itérative et incrémentale. Chaque nouvelle fonctionnalité est créée à partir d'une nouvelle branche puis merge après validation technique.

9. **ERREUR JIRA**
    - J'ai commis une erreur lors de la création du projet, j'ai créé un projet équipe plutôt qu'un projet entreprise. Il était impossible de rendre ce projet public alors j'ai contacté le support de jira qui m'a conseillé de créer un projet entreprise et migrer mes tickets dedans. Cependant tout l'historique a été perdu, il est possible de le retrouver en allant dans chaque tâche. Le projet a démarré le 12 Mai et a duré 4 semaines.

ANNEXES

Annexe 1 – Découpage Briefs

US 2 : Menu de l'application (Navigation)
Menu :
- Retour vers la page d'accueil
- Accès à tous les services
- Accès à tous les habitats
- Connexion
- Contact

US 9 : Connexion
Utilisateur concerné : Administrateur, vétérinaire, employé

Formulaire :
- Email (username)
- Password
  - Success state → Redirect Dashboard
  - Error state

US 12 : SignUp
- Admin crée le compte
- Employé/Vétérinaire reçoit un mail avec Username
- Employé/Vétérinaire doit voir Admin pour Password

US 1 : Page d'accueil (Homepage)
La page d'accueil doit comporter :
- Présentation du zoo en y incorporant quelques images
- Les différents habitats
- Les services
- Les animaux que possède le zoo
- Les avis du Zoo

En tant qu'utilisateur, sur la page d'accueil, je veux pouvoir consulter la présentation du Zoo avec des images, les différents habitats, ...

US 3 : vue globale de tous les services (Page "Services")
Une vue globale, interface simple et récapitulative de tous les services que propose le parc.

Contenu Services :
- Nom
- Description

Listing services :
- restauration
- visite des habitats avec un guide (gratuit)
- visite du zoo en petit train

US 4 : vue globale des habitats (Page "Habitats")
Listing habitats + Animaux associés + Détails animal

Contenu Habitat :
- Un nom
- Une ou des images
- Une description de l'habitat
- Une liste d'animaux

Contenu Animal :
- Un Prénom
- Une race
- Une ou des images
- Un habitat où il est affecté
- Avis vétérinaire

Listing Habitats → On Click → Détail Habitat
- Contenu Habitat

Listing Animal → On Click → Détail de l'animal :
- Contenu Animal
- Avis du vétérinaire

US 5 : Avis (Composant "Homepage")
Listing Avis

Formulaire :
- Pseudo
- Texte
  - *Success State*
  - *Error State*

US 10 : Contact (Page "Contact")
Formulaire :
- Titre
- Description
- Mail
  - *Success State*
  - *Error State*

Pour donner suite à cet envoi, la demande est envoyée par mail au zoo. L'employé peut répondre à la demande directement par mail.

US 11 : Statistique sur la consultation des habitats (Page "Animal")
On clic Animal "A" → Statistiques Animal "A" (Vues) → +1

DB → MongoDB

US 6 : Espace Administrateur (Page "Dashboard" + Page "Utilisateurs" + Page "Services" + Page "Comptes rendus")
Dashboard :
- Statistiques Animaux (MongoDB)❌

En tant qu'admin, sur mon dashboard, je veux pouvoir consulter les statistiques des animaux.❌

Utilisateurs :
- Listing comptes (Delete user) ✅
- Création comptes "Employé" + "Vétérinaire"✅
  - Formulaire :✅
    - Email (Username)✅
    - Password✅
      - Success State → Send Email❌
      - Error State❌

Services :
- Listing services✅
- CRUD services✅

Habitats :
- Listing Habitats✅
- CRUD Habitats✅

Zoo :
- Listing data✅
- CRUD data✅

CR :
- Listing CR ✅
  - Filtres + Tri (Animal + Date)

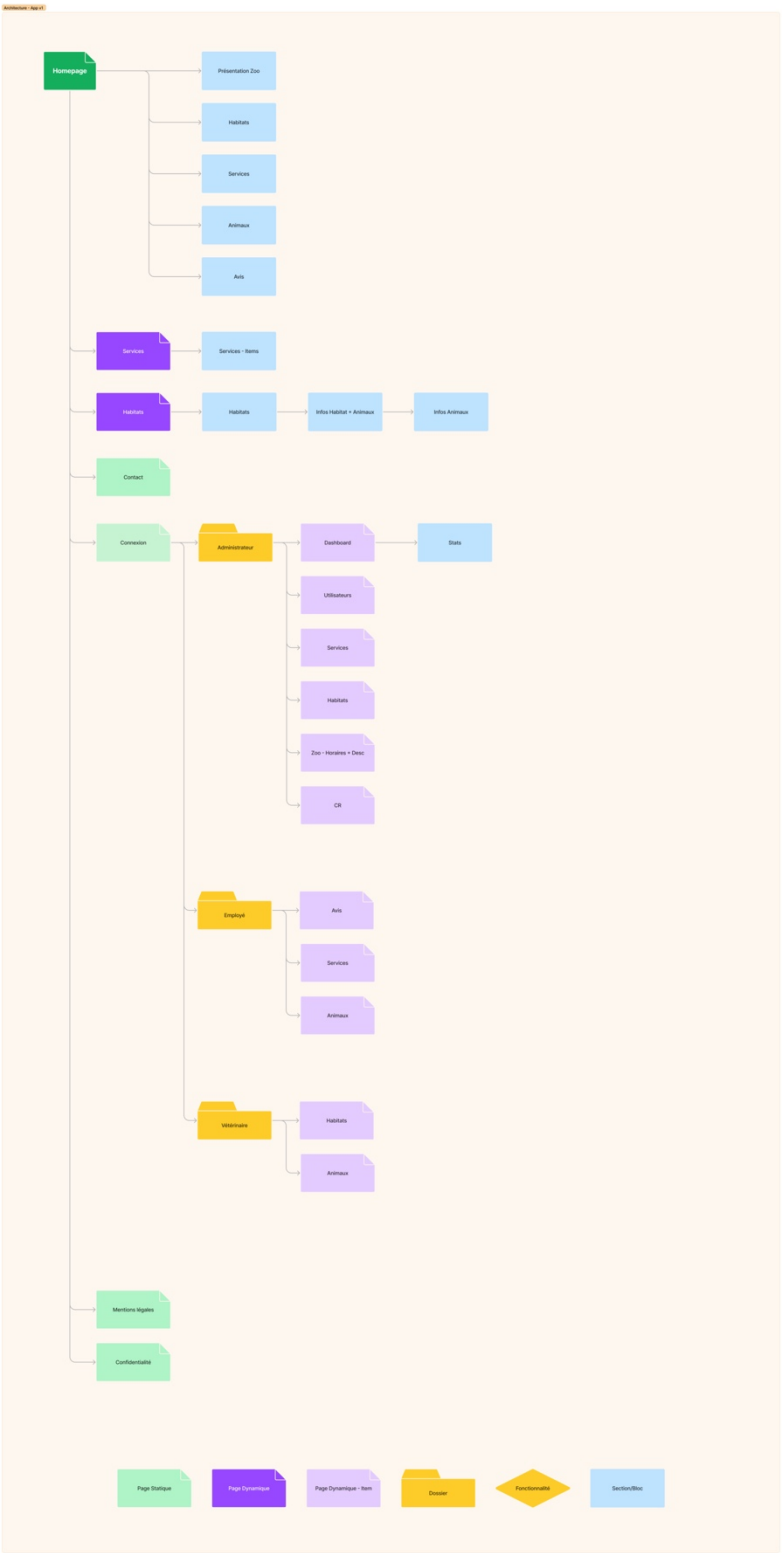
US 7 : Espace Employé (Page "Avis" + Page "Services" + Page "Animaux")
Avis :
- Listing avis users
- Valider / delete avis

Services :
- Listing services
- Modifier services

Animal :
- Listing Animal
  - Listing Nourriture
    - Formulaire nourriture :
      - Date
      - Heure
      - Type nourriture
      - Quantité
        - Success state
        - Error state

US 8 : Espace Vétérinaire (Page "Habitats" + Page "Animaux")
- Listing Animal
  - Listing Nourriture
  - Formulaire Compte rendu :
    - Etat de l'animal
    - Type de nourriture
    - Grammage nourriture
    - Date de passage
    - Détail état de l'animal (facultatif)
      - Success state
      - Error state
- Listing Habitat
  - Formulaire Habitat :
    - Avis
    - Etat
    - Amelioration (oui/non)
      - Success state
      - Error state

Annexe 2 – Architecture App



Annexe 3 – Data Structure