**Rapport de Projet : App Oiseaux**

**Guide Encyclopédique des Oiseaux du Monde**

**1. Introduction**

**Ce projet consiste en la création d'une application web full-stack (React, Flask, SQL) permettant de recenser et d'identifier les espèces d'oiseaux. L'objectif est d'offrir une plateforme intuitive aux ornithologues pour consulter des caractéristiques techniques, gérer une base de données collaborative et utiliser l'intelligence artificielle pour l'identification visuelle.**

**2. Maquettes UX et Inspirations UI**

**Inspirations UI**

* **L'identité visuelle repose sur un design moderne utilisant des dégradés de couleurs froides (Bleu nuit vers Violet) pour évoquer le ciel et la technologie (je me base sur l'imagerie de Canva). Et pour le contenu j'ai cherché des encyclopédies du web**
* **Couleurs clés : #1e3a8a (Bleu marine), #7c3aed (Violet), #00c2cb (Turquoise pour les actions).**
* **Composants : Utilisation de "Cards" blanches avec des ombres légères pour détacher l'information du fond.**

Une image contenant texte, logiciel, Icône d’ordinateur, Système d’exploitation

Le contenu généré par l’IA peut être incorrect. Une image contenant texte, capture d’écran, Logiciel multimédia, oiseau

Le contenu généré par l’IA peut être incorrect.

**Maquettes UX (Parcours utilisateur)**

**L'application est structurée autour d'un header persistant permettant d'accéder aux 5 sections clés :**

1. **Accueil : Galerie d'images.**
2. **Tableau : Vue d'ensemble technique avec filtres de tri.**
3. **Ajouter Oiseau : Formulaire de saisie des données.**
4. **Ajouter Image : Upload simple pour enrichir la base.**
5. **Détecter Oiseau : Interface d'analyse par IA.**

**Schéma de Navigation (User Flow)**

Le menu horizontal en dégradé bleu/violet est le "pivot" central. Chaque flèche représente une action de l'utilisateur :

**[ MENU PRINCIPAL : Accueil | Tableau | Ajouter Oiseau | Ajouter Image | Détecter Oiseau ]** | v

1. **ACCUEIL** (Galerie) ----------------> **DÉTAILS** (Tableau) *L'utilisateur parcourt les photos, puis clique sur "Voir Détails" pour voir les stats.* |
2. **AJOUTER OISEAU** (Formulaire) ------> **CONFIRMATION** (Tableau) *L'utilisateur saisit les données (taille, poids), puis est redirigé vers la liste complète.* |
3. **AJOUTER IMAGE** (Upload) -----------> **GALERIE** (Accueil) *L'utilisateur télécharge une nouvelle photo, celle-ci apparaît ensuite sur l'accueil.* |
4. **DÉTECTER OISEAU** (IA) -------------> **RÉSULTAT** (Tableau) *L'utilisateur upload une photo inconnue, l'IA l'identifie et ouvre sa fiche dans le tableau.*

**Explications du Parcours Utilisateur (UX)**

Voici comment un utilisateur interagit avec ton interface, étape par étape :

**A. Le Parcours de Consultation (Lecture seule)**

* **Étape 1 :** L'utilisateur arrive sur l'**Accueil** pour voir les photos des oiseaux.
* **Étape 2 :** Curieux d'en savoir plus sur l'Aigle Royal, il clique sur le bouton turquoise **"Voir Détails"**.
* **Étape 3 :** Il est instantanément envoyé sur la page **Tableau** où la ligne correspondante est mise en avant (avec sa longévité et sa population).

**B. Le Parcours de Contribution (Ajout de données)**

* **Étape 1 :** L'utilisateur a découvert une nouvelle espèce. Il clique sur **"Ajouter Oiseau"** dans le menu.
* **Étape 2 :** Il remplit le formulaire de saisie.
* **Étape 3 :** Une fois validé, il est redirigé vers le **Tableau** pour voir sa nouvelle entrée s'afficher parmi les autres.

**C. Le Parcours Interactif (Détection par IA)**

* **Étape 1 :** L'utilisateur possède une photo d'un oiseau dont il ignore le nom. Il va sur **"Détecter Oiseau"**.
* **Étape 2 :** Il clique sur la zone en pointillés turquoise **"Choisir une Image"** pour uploader son fichier.
* **Étape 3 :** Il appuie sur le bouton turquoise **"Détecter un Oiseau"**.
* **Étape 4 :** Le système analyse l'image et le redirige vers le **Tableau** filtré sur l'espèce identifiée pour lui donner toutes les informations techniques.

**Points clés de l'expérience utilisateur (UX)**

* **Cohérence visuelle :** Le bouton turquoise est l'élément d'action principal sur toutes les pages (Ajout, Détection, Voir détails).
* **Structure Fixe :** Le header en dégradé ne change jamais de place, ce qui évite à l'utilisateur de se perdre.
* **Lisibilité :** Les données complexes sont isolées dans des "cartes" blanches pour ne pas fatiguer l'œil contre le fond dégradé plus sombre.

**3. Base de données**

**1. Modèle Conceptuel et Relationnel**

**Le schéma de données a été normalisé pour garantir l'intégrité des informations ornithologiques. L'architecture repose sur une table centrale oiseau liée à plusieurs tables satellites par des clés étrangères ().**

* **Gestion de la Taxonomie : Les données d'ordre, de famille et de genre sont isolées pour éviter les redondances.**
* **Caractéristiques Physiques : Une table dédiée stocke les mesures (, , ).**
* **Relations N:N (Plusieurs-à-Plusieurs) : \* Un oiseau peut être observé dans plusieurs pays (oiseau\_pays).**
  + **Un oiseau peut avoir plusieurs auteurs (photographes/chercheurs) via auteur\_oiseau.**

Une image contenant texte, capture d’écran, Police, nombre

Le contenu généré par l’IA peut être incorrect.

**Modèle Physique de Données (MPD)**

**Le script de création de bdd se trouve ici**

**" bird-app\Database\Script de création des tables sql.sql"**

**4. Documentation de l'API REST (Flask)**

**L'interface de programmation (API) sert de pont entre la base de données SQL Server et l'application React. Elle est développée avec Flask et utilise CORS pour permettre les requêtes cross-origin.**

**1. Configuration et Connexion**

**L'API utilise une chaîne de connexion sécurisée (Trusted Connection) via le driver ODBC 17.**

* **Serveur : SQLEXPRESS**
* **Base de données : oiseau**
* **Port : 5000**

**2. Liste des Endpoints**

**A. Récupération des données**

| **Méthode** | **Endpoint** | **Description** |
| --- | --- | --- |
| **GET** | **/api/species** | **Récupère la liste complète des oiseaux via la vue SQL All\_info.** |
| **GET** | **/api/species/<id>** | **Récupère les informations détaillées d'un oiseau spécifique par son identifiant.** |

**Exporter vers Sheets**

* **Exemple de réponse (GET) :**

**JSON**

**{**

**"id\_oiseau": 1,**

**"nom\_commun": "Aigle Royal",**

**"famille": "Accipitridae",**

**"taille\_cm": 90,**

**"url\_image": "https://..."**

**}**

**B. Ajout d'une espèce**

| **Méthode** | **Endpoint** | **Description** |
| --- | --- | --- |
| **POST** | **/api/species** | **Ajoute une nouvelle espèce en base.** |

**Exporter vers Sheets**

* **Fonctionnement interne : Pour garantir l'intégrité des données, l'endpoint utilise des transactions SQL (commit / rollback). L'ajout s'effectue séquentiellement dans 5 tables : taxonomie, caracteristique\_physique, population\_repartition, oiseau et enfin oiseau\_pays.**
* **Corps de la requête (JSON) :**

**JSON**

**{**

**"nom\_commun": "Corbeau Commun",**

**"ordre": "Passeriformes",**

**"taille\_cm": 65,**

**"population\_estimee": 1000000,**

**"id\_pays": 5**

**}**

**C. Suppression**

| **Méthode** | **Endpoint** | **Description** |
| --- | --- | --- |
| **DELETE** | **/api/species/<id>** | **Supprime un oiseau et toutes ses dépendances.** |

**Exporter vers Sheets**

* **Sécurité des données : Le script gère la suppression en cascade manuelle. Avant de supprimer l'entrée dans la table oiseau, il nettoie les clés étrangères dans les tables image, oiseau\_pays et auteur\_oiseau pour éviter toute erreur de contrainte SQL.**

**3. Gestion des erreurs**

**L'API renvoie des codes d'état HTTP standardisés :**

* **201 Created : Succès de l'insertion.**
* **200 OK : Succès de la requête ou suppression.**
* **404 Not Found : L'oiseau demandé n'existe pas.**
* **500 Internal Server Error : Erreur SQL ou JSON invalide (avec retour du message d'erreur pour le débogage).**

**5. Frontend (Interface React)**

**L'application React utilise react-router-dom pour la navigation. L'UI est conçue pour être "Responsive" et épurée.**

**Une image contenant texte, capture d’écran, plein air, oiseau

Le contenu généré par l’IA peut être incorrect.Une image contenant texte, capture d’écran, logiciel, nombre

Le contenu généré par l’IA peut être incorrect.**

**Une image contenant texte, capture d’écran, logiciel, nombre

Le contenu généré par l’IA peut être incorrect.**

**Une image contenant texte, capture d’écran, logiciel, Système d’exploitation

Le contenu généré par l’IA peut être incorrect.Une image contenant texte, capture d’écran, logiciel, Page web

Le contenu généré par l’IA peut être incorrect.**

**6. Difficultés rencontrées**

* **Gestion des types de données : Le passage des fichiers images du Frontend React vers le Backend Flask a nécessité une configuration spécifique des headers (multipart/form-data).**
* **Synchronisation du Tableau : L'implémentation du tri dynamique en temps réel lors du clic sur les en-têtes de colonnes a demandé une gestion rigoureuse de l'état (useState) dans React.**
* **J'ai dû créer une vue dans ma DB SQL pour ressortir le tableau principal des oiseaux pour rendre plus simple l'API**
* **Je n’ai pas réussi a faire la partie IA**

**7. Conclusion et Perspectives**

**Bilan du projet**

**Ce projet a permis de mettre en place une architecture complète pour un guide encyclopédique ornithologique. L'objectif principal, qui était de connecter une application React à une base de données SQL Server via une API Flask, a été réussi.**

**Les fonctionnalités de visualisation (Galerie), de gestion technique (Tableau avec filtres de tri) et d'administration (Formulaires d'ajout et suppression) sont totalement opérationnelles. L'interface utilisateur respecte la charte graphique que j'ai définie en commençant le projet**

**Limites rencontrées**

**Bien que l'architecture globale soit robuste, la fonctionnalité de reconnaissance automatique par Intelligence Artificielle n'a pas pu être intégrée dans cette version finale du projet.**

**Perspectives d’améliorations**

**Pour les futures itérations du site App Oiseaux, plusieurs axes de développement sont envisagés :**

* **Finalisation du module IA : Intégrer le classifieur d'espèces pour automatiser l'identification à partir de la page "Détecter Oiseau".**
* **Optimisation des performances : Mise en place d'un système de mise en cache pour les requêtes SQL complexes afin d'accélérer l'affichage du tableau.**
* **Enrichissement communautaire : Ajouter un système d'authentification pour permettre aux utilisateurs de gérer leurs propres observations et favoris.**
* **Rajouter de la sécurité au site web**

**Lien du dépôt GitHub : https://github.com/anthonycoyaud-prog/bird-app**