Édition 2025

PRÉSENTATION DU PROJET

Nom de votre projet	Pokedex
Membre de l'équipe n°1 (prénom/nom)	**Naël ARRAHMANE**
Membre de l'équipe n°2 (prénom/nom)	**Lise BERNADI**
Membre de l'équipe N°3 (prénom/nom)	**Isis QUEMENER**
Membre de l'équipe n°4 (prénom/nom)	**Thomas QUINQUIS**
Membre de l'équipe n°5 (prénom/nom)	**Fayçal RASSIL**
Niveau d'étude (première ou terminale)	Terminal
Établissement scolaire	Lycée Jean Prevost
Responsable du dépôt (professeur de NSI)	Steeve PYTEL

Pokédex Interactif pour Terminale NSI

Présentation du Projet

Le Pokédex Interactif est une application développée en Python avec Streamlit qui permet aux utilisateurs de consulter, filtrer et capturer des Pokémon. Cette application a été **conçue pour le niveau Terminale NSI**, avec une architecture modulaire et un code structuré pour illustrer les concepts fondamentaux de la programmation.

Objectifs Pédagogiques

- Illustrer les principes de la programmation orientée objet
- Démontrer l'utilisation des bases de données relationnelles
- Présenter les concepts d'architecture logicielle modulaire
- Mettre en pratique les algorithmes de filtrage et de tri
- Montrer la création d'interfaces utilisateur interactives

Fonctionnalités Principales

Multi-utilisateurs : Chaque dresseur a son propre Pokédex et peut capturer des Pokémon indépendamment des autres utilisateurs.

Interface interactive : Affichage des Pokémon sous forme de grille avec leurs images et caractéristiques.

Filtres et tri : Possibilité de filtrer les Pokémon par type et de les trier selon différentes statistiques.

Capture de Pokémon : Les utilisateurs peuvent capturer et relâcher des Pokémon, qui sont mis en évidence dans l'interface.

Persistance des données : Toutes les captures sont automatiquement sauvegardées dans une base de données SQLite.

Technologies Utilisées

Python

Langage de programmation principal

Streamlit

Framework pour l'interface utilisateur

SQLite

Base de données légère

Pandas

Manipulation des données

Pillow

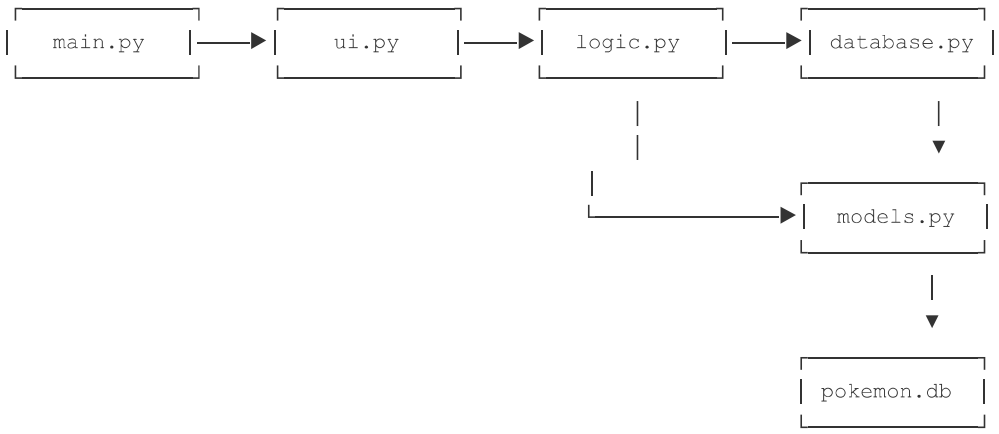
Traitement des images

Prérequis Techniques

- **Python 3.8 ou supérieur** : Nécessaire pour les fonctionnalités modernes utilisées
- **Pip** : Pour l'installation des dépendances
- **Bibliothèques** : Streamlit, Pandas, Pillow (spécifiées dans requirements.txt)

Architecture du Projet

Diagramme d'Architecture



Architecture modulaire avec séparation claire des responsabilités

Structure des Fichiers

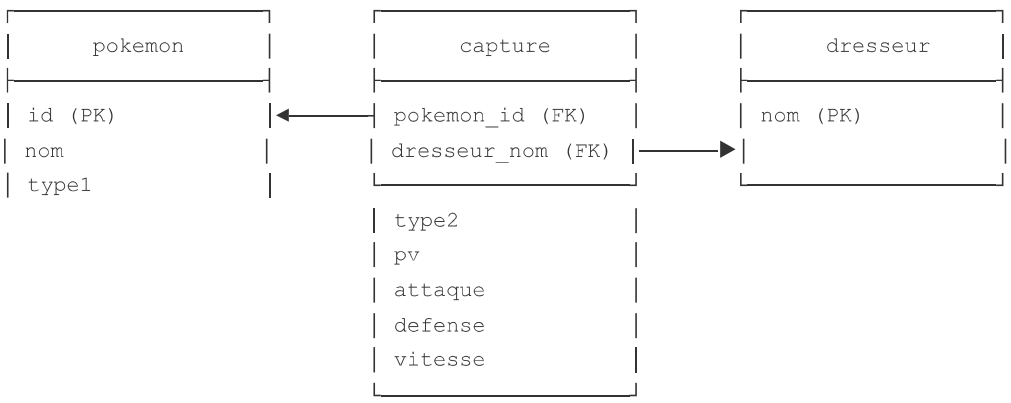
```
/pokedex_project
├── README.md                # Documentation principale du projet
├── LICENSE.txt              # Licence du projet (GPL v3+)
├── requirements.txt         # Dépendances du projet
├── presentation.html        # Présentation synthétique du projet
│
├── sources/                 # Code source du projet
│   ├── main.py              # Point d'entrée de l'application
│   ├── ui.py                # Interface utilisateur (Streamlit)
│   ├── models.py            # Classes de modèle (Pokemon, Dresseur)
│   ├── database.py          # Gestion de la base de données
│   ├── logic.py             # Logique métier (captures, interactions)
│   ├── convert_images.py    # Script pour convertir les images en base64 JSON
│   ├── init_database.py     # Script pour initialiser la base de données
│   └── clean_database.py    # Script pour nettoyer la base de données
│
├── docs/                    # Documentation du projet
│   ├── structure.md         # Explication des fichiers et du code
│   ├── installation.md      # Guide d'installation et d'utilisation
│   ├── architecture.md      # Diagramme et explication de l'architecture
│   └── glossaire.md         # Définitions des termes techniques
│
├── data/                    # Données du projet
│   ├── pokemon.db           # Base de données SQLite
│   ├── images.json          # JSON contenant les images en base64 (généré)
│   └── images/              # Dossier contenant les images PNG originales
│
├── test/                    # Tests unitaires
│   ├── test_database.py     # Tests de la base de données
│   └── test_logic.py        # Tests de la logique métier
│
└── exemples/                # Exemples d'utilisation
    └── screenshots/         # Captures d'écran de l'interface
```

Modules et Responsabilités

Module	Responsabilité	Classes Principales	Concepts NSI
`main.py`	Point d'entrée de l'application	-	Structure d'un programme
`models.py`	Définition des entités principales	Pokemon, Dresseur	POO, encapsulation, modélisation
`database.py`	Gestion des interactions avec la base de données	GestionnaireBD	POO, bases de données, SQL
`logic.py`	Logique métier (capture, relâchement, gestion des sessions)	GestionnairePokedex	POO, algorithmes de filtrage et de tri
`ui.py`	Interface utilisateur avec Streamlit	InterfacePokedex	Interfaces utilisateur, gestion d'événements
`convert_images.py`	Conversion des images PNG en base64	-	Manipulation de fichiers, encodage

Modèle de Données et Relations

Schéma de la Base de Données



Relation many-to-many entre Pokémon et Dresseurs via la table de jonction "capture"

Classes de Modèle

Classe Pokemon (models.py)

Représente un Pokémon avec ses attributs et comportements.

- **Attributs** : id, nom, types, pv, attaque, defense, vitesse, image_base64
- **Méthodes** : __str__, to_dict
- **Responsabilité** : Encapsuler les données et comportements d'un Pokémon

Classe Dresseur (models.py)

Représente un dresseur avec sa collection de Pokémon.

- **Attributs** : nom, pokemons_captures (liste d'IDs)
- **Méthodes** : capturer_pokemon, relacher_pokemon
- **Responsabilité** : Gérer la collection de Pokémon d'un dresseur

Classe GestionnaireBD (database.py)

Gère les interactions avec la base de données SQLite.

- **Attributs** : chemin_bd, chemin_images, connexion, curseur, images_base64
- **Méthodes principales** :
  - charger_tous_pokemons
  - charger_pokemon_par_id
  - charger_dresseur
  - sauvegarder_capture
  - sauvegarder_relachement
  - obtenir_tous_types

    - obtenir_top_dresseurs
- **Responsabilité** : Abstraire les opérations de base de données

Classe GestionnairePokedex (logic.py)

Gère la logique métier de l'application.

- **Attributs** : gestionnaire_bd
- **Méthodes principales** :
    - connecter_dresseur
    - capturer_pokemon
    - relacher_pokemon
    - filtrer_pokemons
    - trier_pokemons
    - est_pokemon_capture
- **Responsabilité** : Coordonner les opérations entre l'interface et la base de données

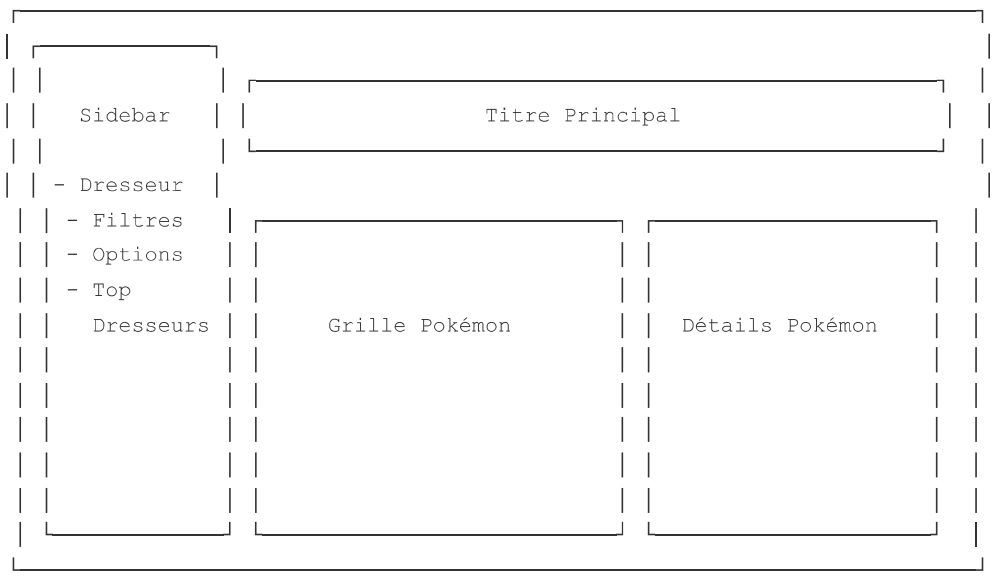
Classe InterfacePokedex (ui.py)

Gère l'interface utilisateur avec Streamlit.

- **Attributs** : gestionnaire
- **Méthodes principales** :
    - afficher_interface
    - afficher_sidebar
    - afficher_tableau_pokemons
    - afficher_details_pokemon
    - afficher_carte_pokemon
- **Responsabilité** : Gérer l'affichage et les interactions utilisateur

Interface Utilisateur et Expérience Utilisateur

Structure de l'Interface



Interface divisée en trois zones principales : barre latérale, grille de Pokémon et détails du Pokémon sélectionné

Composants de l'Interface

1. **Barre latérale (Sidebar)**
   - Informations du dresseur connecté
   - Filtres par type de Pokémon
   - Options de tri
   - Bouton pour réinitialiser les filtres
   - Classement des meilleurs dresseurs
2. **Grille de Pokémon**
   - Affichage des Pokémon sous forme de cartes
   - Mise en évidence des Pokémon capturés
   - Bouton "Détails" pour chaque Pokémon
   - Affichage de la statistique utilisée pour le tri
   - Badges colorés pour les types de Pokémon
3. **Détails du Pokémon**
   - Image du Pokémon en plus grand format
   - Nom et types du Pokémon
   - Statistiques avec barres de progression
   - Bouton "Capturer" ou "Relâcher" selon l'état

Flux d'Interaction Utilisateur

1. **Connexion** : L'utilisateur entre son nom de dresseur et se connecte
2. **Navigation** : L'utilisateur parcourt la grille de Pokémon
3. **Filtrage** : L'utilisateur peut filtrer les Pokémon par type et les trier selon différentes statistiques

4. ## Flux d'Interaction Utilisateur

   1. **Connexion : L'utilisateur entre son nom de dresseur et se connecte**
   2. **Navigation : L'utilisateur parcourt la grille de Pokémon**

3. **Filtrage : L'utilisateur peut filtrer les Pokémon par type et les trier selon différentes statistiques**
4. **Sélection : L'utilisateur clique sur un Pokémon pour voir ses détails**
5. **Capture : L'utilisateur peut capturer ou relâcher le Pokémon sélectionné**
6. **Persistance : Toutes les actions sont automatiquement sauvegardées dans la base de données**

Gestion de l'État avec Streamlit

```
# Dans logic.py
def initialiser_session(self):
    """Initialise les variables de session si elles n'existent pas déjà."""
    # Variable indiquant si un dresseur est connecté
    if 'dresseur_connecte' not in st.session_state:
        st.session_state.dresseur_connecte = False

    # Nom du dresseur connecté
    if 'nom_dresseur' not in st.session_state:
        st.session_state.nom_dresseur = ""

    # Objet Dresseur du dresseur connecté
    if 'dresseur' not in st.session_state:
        st.session_state.dresseur = None

    # Pokémon actuellement sélectionné pour afficher ses détails
    if 'pokemon_selectionne' not in st.session_state:
        st.session_state.pokemon_selectionne = None

    # Type de Pokémon sélectionné pour le filtrage
    if 'filtre_type' not in st.session_state:
        st.session_state.filtre_type = "Tous"

    # Critère de tri des Pokémon
    if 'tri_par' not in st.session_state:
        st.session_state.tri_par = "ID"
```

Explication

Streamlit recharge la page à chaque interaction, ce qui nécessite de stocker l'état de l'application entre les rechargements. La classe `GestionnairePokedex` initialise les variables de session pour :

- **Gérer la connexion du dresseur**
- **Stocker les préférences de filtrage et de tri**
- **Mémoriser le Pokémon sélectionné**

Concepts NSI : Gestion d'état, persistance des données

Aspects Pédagogiques

Concepts NSI Illustrés

Programmation Orientée Objet

- **Encapsulation : Les classes Pokemon et Dresseur encapsulent leurs données et comportements**
- **Abstraction : Les classes GestionnaireBD et GestionnairePokedex abstraient les opérations complexes**
- **Séparation des responsabilités : Chaque classe a une responsabilité unique et bien définie**

Bases de Données

- **Modèle relationnel : Tables pokemon, dresseur et capture avec relations**
- **Requêtes SQL : SELECT, INSERT, DELETE pour interagir avec la base de données**
- **Transactions : Utilisation de commit pour garantir l'intégrité des données**
- **Relation many-to-many : Implémentation avec une table de jonction**

Algorithmes

- **Filtrage : Implémentation d'un algorithme de filtrage avec une boucle for**
- **Tri : Utilisation de la fonction sorted avec des fonctions lambda**
- **Recherche : Recherche d'éléments dans des listes avec l'opérateur in**

Conclusion et Perspectives

Synthèse du Projet

Le Pokédex Interactif est une application complète qui illustre de nombreux concepts fondamentaux de la programmation et de l'architecture logicielle :

- **Architecture modulaire avec séparation des responsabilités**
- **Programmation orientée objet avec encapsulation et abstraction**
- **Gestion de base de données relationnelle**
- **Interface utilisateur interactive et réactive**
- **Persistance des données et gestion d'état**

Pistes d'Amélioration

- **Optimisation des performances : Utilisation de caching pour réduire les requêtes à la base de données**
- **Amélioration de l'interface : Ajout d'animations et de transitions pour une meilleure expérience utilisateur**
- **Fonctionnalités avancées : Système de combat, évolution des Pokémon, échange entre dresseurs**
- **Tests automatisés : Développement de tests unitaires et d'intégration plus complets**
- **Déploiement : Mise en ligne de l'application pour permettre un accès à distance**

Pokédex Interactif - Projet pédagogique pour Terminale NSI