A black and white logo

AI-generated content may be incorrect.A close-up of a logo

AI-generated content may be incorrect.

**Rapport de Projet**

**Système de Gestion d’une**

**Clinique Vétérinaire**

**Équipe : PySynergy**

* ADDI Safaa
* CHRAA Zakaria
* JAOUHARI Mounir
* LAMOURI Mohamed Amine

Sous l’encadrement du Mr ETTALALI Oussama

A logo with a snake and text

AI-generated content may be incorrect.Année 2025/2026

Résumé

Ce rapport présente la conception, l’implémentation et la validation d’un **système de gestion pour une clinique vétérinaire** développé en Python. Le projet vise à remplacer la gestion papier par un outil CLI robuste offrant : gestion des propriétaires, des animaux, archivage des consultations, persistance JSON, génération automatique d’identifiants, validations des saisies et fonctions d’analyse (Pandas/Matplotlib). Le document couvre : le cahier des charges, la modélisation objet, l’architecture logicielle, les choix techniques, la phase d’implémentation, les tests, ainsi que les perspectives d’évolution (interface graphique, base de données relationnelle, API).

**Mots-clés :** Python, POO, CLI, JSON, Pandas, Matplotlib, tests unitaires, UML.

# Table des matières

**Résumé exécutif 2**

[**Table des matières**](#_9dn4g0vhhb13) **3**

**1 : Introduction 5**

1.1 Contexte 5

1.2 Objectifs 5

1.3 Méthodologie 5

**2 : Problématique 5**

2.1 Situation 5

2.2 Problèmes identifiés 5

2.3 Bénéfices attendus 6

**3 : Cahier des charges 6**

3.1 Objectifs fonctionnels 6

3.2 Objectifs non-fonctionnels 6

3.3 Contraintes 6

**4 : Analyse et Conception 7**

4.1 Cas d’utilisation 7

[4.2 Diagramme de classes](#_covj1l9b1rda) 7

4.3 Choix d’architecture 7

**5 : Implémentation 8**

5.1 Organisation du code 8

5.2 Extraits de code source 8

5.2.1 ‘models.py’ 8

[5.2.2 ‘clinique.py ‘](#_faq39uvr32a2) 10

5.2.3 ‘persistence.py’ (extrait) 13

5.2.4 ‘analyse.py’ 15

[5.2.5 ‘main.py’](#_8dtz4icv6oov) 17

**6 : Interface utilisateur (CLI) 23**

6.1 Principes ergonomiques 23

6.2 Menu principal (extrait) 23

[6.3 Exemple d’interaction](#_80j9z9mrfgb0) 24

**7 : Persistance des données 24**

7.1 Format JSON 24

[7.2 Sauvegarde en JSON](#_2dnbxtowxgst) 25

[7.3 Chargement des données](#_gwm5s2vwqzjn) 26

7.4 Reconstruction des compteurs d’ID 27

**8 : Analyse de données et rapports 28**

8.1 Objectifs analytiques 28

8.2 Exemples de visualisations 28

8.3 Interprétation 29

**9 : Tests et validation 29**

9.1 Stratégie de tests 29

9.2 Cas de tests représentatifs 30

**10 : Gestion de projet 30**

10.1 Répartition des tâches 30

**11 : Résultats obtenus 30**

11.1 Livrables fournis 30

11.2 Exemples d’exécution 30

**12 : Difficultés rencontrées et solutions 33**

**13 :Conclusion et perspectives 33**

**Instructions d’installation 33**

**Bibliographie 34**

# 1 : Introduction

## 1.1 Contexte

Les petites et moyennes cliniques vétérinaires gèrent quotidiennement des dossiers patients souvent sur papier. Ce mode de fonctionnement est source d’erreurs, de pertes d’information et d’une faible efficacité administrative. Dans le cadre du formation JobInTech avec Ynov, nous avons développé un outil logiciel simple, maintenable et extensible pour traiter ces problèmes.

## 1.2 Objectifs

* Mettre en pratique les concepts de la programmation orientée objet (héritage, encapsulation, responsabilité des classes).
* Concevoir une architecture modulaire et testable.
* Manipuler la persistance via JSON et réaliser des analyses de données (Pandas).
* Produire un livrable professionnel et documenté.

## 1.3 Méthodologie

* Le projet a été développé en itérations successives :
* Spécification et modélisation (UML).
* Implémentation des modèles et de la persistance.
* Construction de la CLI et des validations.
* Ajout des fonctions d’analyse et génération de rapports.
* Tests unitaires et recettes fonctionnelles.

# 2 : Problématique

## 2.1 Situation

Description des pratiques manuelles courantes : fiches papier, classeurs, absence d’historique accessible rapidement, duplication d’informations.

## 2.2 Problèmes identifiés

* Risque de perte d’information et d’erreurs humaines.
* Temps perdu dans la recherche d’historique.
* Difficulté à établir des rapports (revenu, motifs fréquents).

## 2.3 Bénéfices attendus

Automatisation, traçabilité, génération de statistiques utiles à la direction, meilleur suivi médical des patients.

# 3 : Cahier des charges

## 3.1 Objectifs fonctionnels

Le système doit :

1. Enregistrer propriétaires et animaux.
2. Associer automatiquement animaux propriétaires.
3. Enregistrer des consultations (date, motif, diagnostic, coût).
4. Fournir un historique trié par date.
5. Rechercher dans les diagnostics (mot-clé).
6. Générer un rapport d’activité (revenu mensuel, top motifs, histogramme).
7. Sauvegarder/restaurer l’état via JSON.

## 3.2 Objectifs non-fonctionnels

* Interface CLI simple et robuste.
* Validation stricte des entrées.
* IDs générés automatiquement via compteurs de classe.
* Code lisible, modulaire et documenté.

## 3.3 Contraintes

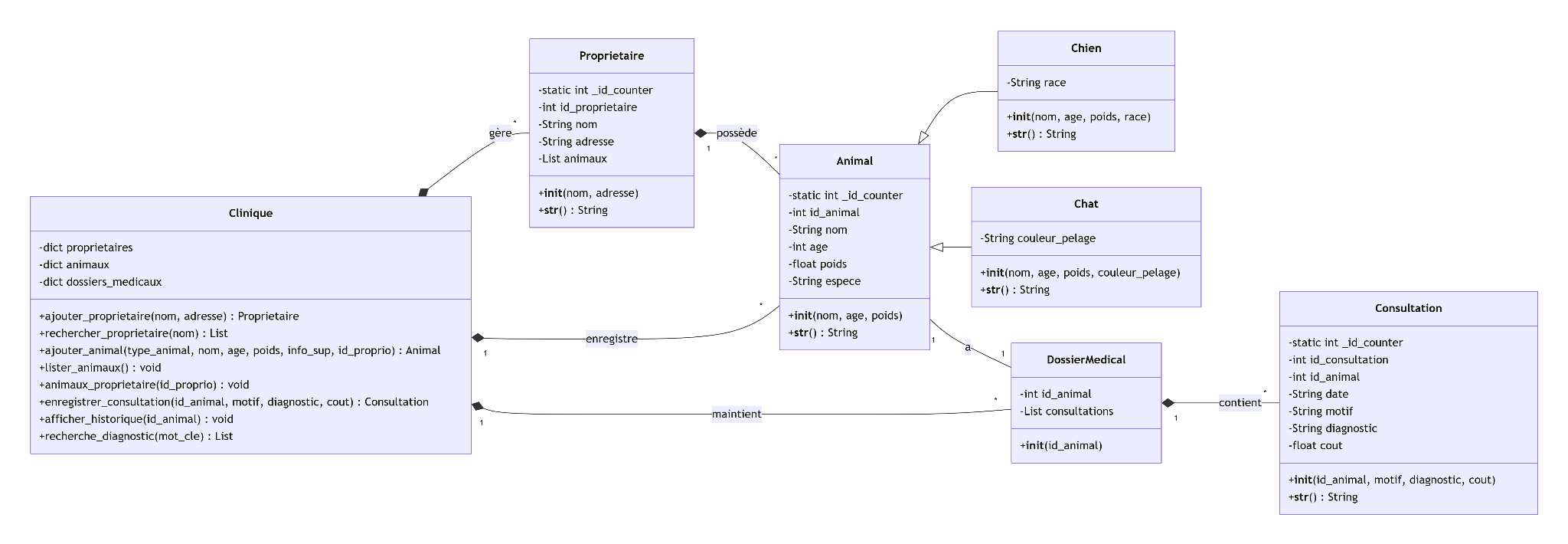
* Utiliser uniquement Python standard + Pandas et Matplotlib.
* Persistance simple (fichier JSON).
* Utilisation en mode offline sur un poste local.

# 4 : Analyse et Conception

## 4.1 Cas d’utilisation

1. **Gérer les propriétaires** - Ajouter/rechercher des propriétaires
2. **Gérer les animaux** - Ajouter chiens/chats et les associer aux propriétaires
3. **Enregistrer consultation** - Créer une consultation avec motif, diagnostic et coût
4. **Voir historique médical** - Consulter toutes les consultations d'un animal
5. **Recherche par diagnostic** - Trouver consultations avec un mot-clé
6. **Lister animaux** - Afficher tous les animaux enregistrés
7. **Voir animaux d'un propriétaire** - Lister les animaux par propriétaire
8. **Générer rapport** - Statistiques revenus, motifs fréquents, graphiques âges

## 4.2 Diagramme de classes



## 4.3 Choix d’architecture

L’application a été conçue selon une architecture modulaire afin de garantir une meilleure organisation, une maintenance facilitée et une évolutivité du projet. Chaque module a une responsabilité bien définie :

Architecture modulaire :

* models.py : classes métier (Animal, Chien, Chat, Proprietaire, Consultation, DossierMedical).
* clinique.py : logique métier et orchestration.
* persistence.py : lecture/écriture JSON.
* analyse.py : génération de rapports (Pandas/Matplotlib).
* main.py : interface CLI (point d’entrée).

# 5 : Implémentation

## 5.1 Organisation du code

|  |  |
| --- | --- |
| Fichier | Rôle |
| models.py | Définit toutes les classes métier et compteurs d’ID automatiques. |
| clinique.py | Contient la classe Clinique : API pour ajouter, rechercher, enregistrer consultations. |
| persistence.py | Sérialisation/désérialisation JSON, reconstruction d’objets. |
| analyse.py | Transforme les consultations en DataFrame, génère graphiques et rapports. |
| main.py | Menu CLI, validation des entrées et interactions utilisateur. |
| data.json | Fichier de sauvegarde généré à l’exécution. |

## 5.2 Extraits de code source

### 5.2.1 ‘models.py’

# Classe Animal et héritage

class Animal:

\_id\_counter = 1

def \_\_init\_\_(self, nom, age, poids):

self.id\_animal = Animal.\_id\_counter

Animal.\_id\_counter += 1

self.nom = nom

self.age = age

self.poids = poids

self.espece = "Animal"

def \_\_str\_\_(self):

return f"ID: {self.id\_animal} {self.nom}, {self.espece}, {self.age} ans, {self.poids} kg"

class Chien(Animal):

def \_\_init\_\_(self, nom, age, poids, race):

super().\_\_init\_\_(nom, age, poids)

self.race = race

self.espece = "Chien"

def \_\_str\_\_(self):

return super().\_\_str\_\_() + f", Race: {self.race}"

class Chat(Animal):

def \_\_init\_\_(self, nom, age, poids, couleur\_pelage):

super().\_\_init\_\_(nom, age, poids)

self.couleur\_pelage = couleur\_pelage

self.espece = "Chat"

def \_\_str\_\_(self):

return super().\_\_str\_\_() + f", Couleur pelage: {self.couleur\_pelage}"

#Classe Propriétaire

class Proprietaire:

\_id\_counter = 1

def \_\_init\_\_(self, nom, adresse):

self.id\_proprietaire = Proprietaire.\_id\_counter

Proprietaire.\_id\_counter += 1

self.nom = nom

self.adresse = adresse

self.animaux = []

def \_\_str\_\_(self):

return f"ID: {self.id\_proprietaire} {self.nom}, {self.adresse}, Animaux: {self.animaux}"

#Classe Consultations

from datetime import datetime

class Consultation:

\_id\_counter = 1

def \_\_init\_\_(self, id\_animal, motif, diagnostic, cout):

self.id\_consultation = Consultation.\_id\_counter

Consultation.\_id\_counter += 1

self.id\_animal = id\_animal

self.date = datetime.now().strftime("%Y-%m-%d %H:%M")

self.motif = motif

self.diagnostic = diagnostic

self.cout = cout

def \_\_str\_\_(self):

return f"ID: {self.id\_consultation} Animal {self.id\_animal}, Date: {self.date}, Motif: {self.motif}, Diagnostic: {self.diagnostic}, Coût: {self.cout} DH"

class DossierMedical:

def \_\_init\_\_(self, id\_animal):

self.id\_animal = id\_animal

self.consultations = []

### 5.2.2 ‘clinique.py ‘

# Classe Clinique (principale)

from models import Proprietaire, Chien, Chat, Consultation, DossierMedical

class Clinique:

def \_\_init\_\_(self):

self.proprietaires = {}

self.animaux = {}

self.dossiers\_medicaux = {}

def ajouter\_proprietaire(self, nom, adresse):

p = Proprietaire(nom, adresse)

self.proprietaires[p.id\_proprietaire] = p

return p

def rechercher\_proprietaire(self, nom):

return [p for p in self.proprietaires.values() if nom.lower() in p.nom.lower()]

def ajouter\_animal(self, type\_animal, nom, age, poids, info\_sup, id\_proprio):

if type\_animal.lower() == "chien":

a = Chien(nom, age, poids, info\_sup)

elif type\_animal.lower() == "chat":

a = Chat(nom, age, poids, info\_sup)

else:

return None

self.animaux[a.id\_animal] = a

self.proprietaires[id\_proprio].animaux.append(a.id\_animal)

return a

def lister\_animaux(self):

if not self.animaux:

print("Aucun animal enregistré.")

return

print("Liste des animaux :")

for a in self.animaux.values():

print(" ", a)

def animaux\_proprietaire(self, id\_proprio):

p = self.proprietaires.get(id\_proprio)

if not p:

print("Propriétaire non trouvé")

return

print(f"Animaux du propriétaire {p.nom} :")

for aid in p.animaux:

print(" ", self.animaux.get(aid))

def enregistrer\_consultation(self, id\_animal, motif, diagnostic, cout):

if id\_animal not in self.animaux:

print("Animal non trouvé")

return None

c = Consultation(id\_animal, motif, diagnostic, cout)

if id\_animal not in self.dossiers\_medicaux:

self.dossiers\_medicaux[id\_animal] = DossierMedical(id\_animal)

self.dossiers\_medicaux[id\_animal].consultations.append(c)

return c

def afficher\_historique(self, id\_animal):

dossier = self.dossiers\_medicaux.get(id\_animal)

if not dossier:

print("Aucun dossier médical pour cet animal")

return

print(f"Historique médical de l'animal {self.animaux[id\_animal].nom} :")

for c in sorted(dossier.consultations, key=lambda x: x.date):

print(" ", c)

def recherche\_diagnostic(self, mot\_cle):

resultats = []

for dossier in self.dossiers\_medicaux.values():

for c in dossier.consultations:

if mot\_cle.lower() in c.diagnostic.lower():

resultats.append(c)

return resultats

### 5.2.3 ‘persistence.py’ (extrait)

import json, os

from models import Proprietaire, Chien, Chat, Consultation, DossierMedical, Animal

# Sauvegarde les données actuel de clinique dans le fichier json

def sauvegarder(clinique, fichier="data.json"):

data = {

"proprietaires": {pid: vars(p) for pid, p in clinique.proprietaires.items()},

"animaux": {aid: vars(a) for aid, a in clinique.animaux.items()},

"dossiers\_medicaux": {

str(aid): [vars(c) for c in d.consultations]

for aid, d in clinique.dossiers\_medicaux.items()

}

}

with open(fichier, "w") as f:

json.dump(data, f, indent=4)

# Charge les données de clinique enregistrées dans le fichier json

def charger(clinique, fichier="data.json"):

if not os.path.exists(fichier) or os.path.getsize(fichier) == 0:

print("-> Première utilisation.")

return

with open(fichier, "r") as f:

data = json.load(f)

clinique.proprietaires = {}

for pid, p\_data in data.get("proprietaires", {}).items():

p = Proprietaire(p\_data['nom'], p\_data['adresse'])

p.id\_proprietaire = p\_data['id\_proprietaire']

p.animaux = p\_data['animaux']

clinique.proprietaires[p.id\_proprietaire] = p

clinique.animaux = {}

for aid, a\_data in data.get("animaux", {}).items():

if a\_data['espece'] == "Chien":

a = Chien(a\_data['nom'], a\_data['age'], a\_data['poids'], a\_data.get('race',''))

elif a\_data['espece'] == "Chat":

a = Chat(a\_data['nom'], a\_data['age'], a\_data['poids'], a\_data.get('couleur\_pelage',''))

else:

continue

a.id\_animal = a\_data['id\_animal']

clinique.animaux[a.id\_animal] = a

clinique.dossiers\_medicaux = {}

for aid, c\_list in data.get("dossiers\_medicaux", {}).items():

dossier = DossierMedical(int(aid))

for c\_data in c\_list:

c = Consultation(c\_data['id\_animal'], c\_data['motif'], c\_data['diagnostic'], c\_data['cout'])

c.id\_consultation = c\_data['id\_consultation']

c.date = c\_data['date']

dossier.consultations.append(c)

clinique.dossiers\_medicaux[int(aid)] = dossier

#Ajuste les compteurs d'ID après le chargement des données

def remettre\_a\_jour\_compteurs(clinique):

if clinique.animaux:

Animal.\_id\_counter = max(a.id\_animal for a in clinique.animaux.values()) + 1

else:

Animal.\_id\_counter = 1

if clinique.proprietaires:

Proprietaire.\_id\_counter = max(p.id\_proprietaire for p in clinique.proprietaires.values()) + 1

else:

Proprietaire.\_id\_counter = 1

toutes\_consultations = [c for d in clinique.dossiers\_medicaux.values() for c in d.consultations]

if toutes\_consultations:

Consultation.\_id\_counter = max(c.id\_consultation for c in toutes\_consultations) + 1

else:

Consultation.\_id\_counter = 1

### 5.2.4 ‘analyse.py’

import pandas as pd

import matplotlib.pyplot as plt

def generer\_rapport\_activite(clinique):

consultations = []

for dossier in clinique.dossiers\_medicaux.values():

for c in dossier.consultations:

consultations.append({

"id\_consultation": c.id\_consultation,

"id\_animal": c.id\_animal,

"date": c.date,

"motif": c.motif,

"diagnostic": c.diagnostic,

"cout": c.cout

})

if not consultations:

print("Aucune consultation à analyser.")

return

df = pd.DataFrame(consultations)

df['date'] = pd.to\_datetime(df['date'])

#Revenu mensuel

revenu\_mensuel = df.groupby(pd.Grouper(key='date', freq='ME'))['cout'].sum()

print("\n💰 Revenu par mois :")

for date, total in revenu\_mensuel.items():

print(f" {date.strftime('%Y-%m')}: {total:.2f} DH")

#Motifs les plus fréquents

motifs = df['motif'].value\_counts().head(5)

print("\n📌 Motifs les plus fréquents :")

for motif, count in motifs.items():

print(f" {motif}: {count}")

# --- Distribution des âges ---

ages = [a.age for a in clinique.animaux.values()]

plt.hist(ages, bins=10, color='skyblue', edgecolor='black')

plt.title("Distribution des âges des animaux")

plt.xlabel("Âge (ans)")

plt.ylabel("Nombre d'animaux")

plt.savefig("rapport\_patients.png")

print("\n📊 Graphique sauvegardé dans 'rapport\_patients.png'")

### 5.2.5 ‘main.py’

from clinique import Clinique

from persistence import sauvegarder, charger

from analyse import generer\_rapport\_activite

clinique = Clinique()

# Charger les données (premier lancement géré)

try:

charger(clinique)

print("Données chargées avec succès.")

except FileNotFoundError:

print("Première utilisation : aucune donnée trouvée, base initialisée vide.")

except Exception as e:

print("Erreur au chargement :", e)

remettre\_a\_jour\_compteurs(clinique)

# Fonctions de validation

def input\_nonvide(prompt):

while True:

val = input(prompt).strip()

if val == "0":

return None

if val:

return val

print("Champ obligatoire, veuillez réessayer.")

def input\_int(prompt):

while True:

val = input\_nonvide(prompt)

if val is None:

return None

if val.isdigit():

return int(val)

print("Veuillez entrer un entier valide.")

def input\_float(prompt):

while True:

val = input\_nonvide(prompt)

if val is None:

return None

try:

return float(val)

except ValueError:

print("Veuillez entrer un nombre valide.")

def input\_email(prompt):

while True:

val = input\_nonvide(prompt)

if val is None:

return None

if "@" in val:

return val

print("Email invalide, doit contenir '@'.")

# Menu principal

def menu():

print("\n===== CLINIQUE VÉTÉRINAIRE =====")

print("1. Ajouter un propriétaire")

print("2. Ajouter un animal")

print("3. Lister tous les animaux")

print("4. Lister animaux d'un propriétaire")

print("5. Enregistrer une consultation")

print("6. Historique d'un animal")

print("7. Rechercher diagnostic")

print("8. Générer rapport activité")

print("0. Quitter")

# Boucle de menu

while True:

menu()

choix = input\_nonvide("Choisir une option: ")

if choix is None or choix == "0":

print("Au revoir !")

break

if choix == "1":

nom = input\_nonvide("Nom du propriétaire (0 pour annuler): ")

if nom is None:

print("Opération annulée")

continue

email = input\_email("Adresse email (0 pour annuler): ")

if email is None:

print("Opération annulée")

continue

p = clinique.ajouter\_proprietaire(nom, email)

sauvegarder(clinique)

print("Propriétaire ajouté:", p)

elif choix == "2":

if not clinique.proprietaires:

print("Aucun propriétaire disponible.")

continue

print("Types disponibles: Chien, Chat")

type\_a = input\_nonvide("Type animal (0 pour annuler): ")

if type\_a is None:

print("Opération annulée")

continue

nom = input\_nonvide("Nom (0 pour annuler): ")

if nom is None:

print("Opération annulée")

continue

age = input\_int("Âge (ans) (0 pour annuler): ")

if age is None:

print("Opération annulée")

continue

poids = input\_float("Poids (kg) (0 pour annuler): ")

if poids is None:

print("Opération annulée")

continue

if type\_a.lower() == "chien":

info\_sup = input\_nonvide("Race (0 pour annuler): ")

elif type\_a.lower() == "chat":

info\_sup = input\_nonvide("Couleur pelage (0 pour annuler): ")

else:

print("Type inconnu")

continue

if info\_sup is None:

print("Opération annulée")

continue

print("Propriétaires disponibles:")

for p in clinique.proprietaires.values():

print(" - "+ p)

id\_proprio = input\_int("ID du propriétaire (0 pour annuler): ")

if id\_proprio is None:

print("Opération annulée")

continue

if id\_proprio not in clinique.proprietaires:

print("Propriétaire non trouvé")

continue

a = clinique.ajouter\_animal(type\_a, nom, age, poids, info\_sup, id\_proprio)

sauvegarder(clinique)

print("Animal ajouté:", a)

elif choix == "3":

clinique.lister\_animaux()

elif choix == "4":

id\_p = input\_int("ID du propriétaire (0 pour annuler): ")

if id\_p is None:

print("Opération annulée")

continue

clinique.animaux\_proprietaire(id\_p)

elif choix == "5":

id\_a = input\_int("ID de l'animal (0 pour annuler): ")

if id\_a is None:

print("Opération annulée")

continue

motif = input\_nonvide("Motif (0 pour annuler): ")

if motif is None:

print("Opération annulée")

continue

diagnostic = input\_nonvide("Diagnostic (0 pour annuler): ")

if diagnostic is None:

print("Opération annulée")

continue

cout = input\_float("Coût (€) (0 pour annuler): ")

if cout is None:

print("Opération annulée")

continue

c = clinique.enregistrer\_consultation(id\_a, motif, diagnostic, cout)

if c:

sauvegarder(clinique)

print("Consultation enregistrée:", c)

elif choix == "6":

id\_a = input\_int("ID de l'animal (0 pour annuler): ")

if id\_a is None:

print("Opération annulée")

continue

clinique.afficher\_historique(id\_a)

elif choix == "7":

mot = input\_nonvide("Mot-clé diagnostic (0 pour annuler): ")

if mot is None:

print("Opération annulée")

continue

res = clinique.recherche\_diagnostic(mot)

if res:

print("Résultats:")

for c in res:

print(" ", c)

else:

print("Aucun résultat trouvé.")

elif choix == "8":

generer\_rapport\_activite(clinique)

else:

print("Option invalide.")

# 6 : Interface utilisateur (CLI)

## 6.1 Principes ergonomiques

* Menu numéroté et lisible.
* Validation et rappel en cas d’erreur.
* Messages informatifs (succès / échec).

## 6.2 Menu principal (extrait)

===== CLINIQUE VÉTÉRINAIRE =====

1. Ajouter un propriétaire

2. Ajouter un animal

3. Lister tous les animaux

4. Lister animaux d'un propriétaire

5. Enregistrer une consultation

6. Historique d'un animal

7. Rechercher diagnostic

8. Générer rapport activité

0. Quitter

## 6.3 Exemple d’interaction

Lorsque l' utilisateur a choisi l' option 1

Choisir une option: 1

Nom du propriétaire (0 pour annuler): Assim

Adresse email (0 pour annuler): assim@gmail.com

✅ Propriétaire ajouté: [5] Assim, assim@gmail.com, Animaux: []

# 7 : Persistance des données

## 7.1 Format JSON

La structure du fichier ‘data.json‘ est organisée en trois objets : ‘proprietaires‘, ‘animaux‘, ‘dossiers‘. Chaque entité stocke les attributs nécessaires pour reconstruire les objets au chargement.

{

"proprietaires": {

"1": {

"id\_proprietaire": 1,

"nom": "Youssef",

"adresse": "youssef@gmail.com",

"animaux": [1]

}

},

"animaux": {

"1": {

"id\_animal": 1,

"nom": "Simo",

"age": 3,

"poids": 22.5,

"espece": "Chien",

"race": "Berger Allemand"

}

},

"dossiers\_medicaux": {

"1": [

{

"id\_consultation": 1,

"id\_animal": 1,

"date": "2025-10-01 11:30",

"motif": "Boiterie",

"diagnostic": "Entorse patte avant",

"cout": 200.0

}

]

}

}

## 7.2 Sauvegarde en JSON

# Sauvegarde les données actuel de clinique dans le fichier json

def sauvegarder(clinique, fichier="data.json"):

data = {

"proprietaires": {pid: vars(p) for pid, p in clinique.proprietaires.items()},

"animaux": {aid: vars(a) for aid, a in clinique.animaux.items()},

"dossiers\_medicaux": {

str(aid): [vars(c) for c in d.consultations]

for aid, d in clinique.dossiers\_medicaux.items()

}

}

with open(fichier, "w") as f:

json.dump(data, f, indent=4)

## 7.3 Chargement des données

# Charge les données de clinique enregistrées dans le fichier json

def charger(clinique, fichier="data.json"):

if not os.path.exists(fichier) or os.path.getsize(fichier) == 0:

print("-> Première utilisation.")

return

with open(fichier, "r") as f:

data = json.load(f)

clinique.proprietaires = {}

for pid, p\_data in data.get("proprietaires", {}).items():

p = Proprietaire(p\_data['nom'], p\_data['adresse'])

p.id\_proprietaire = p\_data['id\_proprietaire']

p.animaux = p\_data['animaux']

clinique.proprietaires[p.id\_proprietaire] = p

clinique.animaux = {}

for aid, a\_data in data.get("animaux", {}).items():

if a\_data['espece'] == "Chien":

a = Chien(a\_data['nom'], a\_data['age'], a\_data['poids'], a\_data.get('race',''))

elif a\_data['espece'] == "Chat":

a = Chat(a\_data['nom'], a\_data['age'], a\_data['poids'], a\_data.get('couleur\_pelage',''))

else:

continue

a.id\_animal = a\_data['id\_animal']

clinique.animaux[a.id\_animal] = a

clinique.dossiers\_medicaux = {}

for aid, c\_list in data.get("dossiers\_medicaux", {}).items():

dossier = DossierMedical(int(aid))

for c\_data in c\_list:

c = Consultation(c\_data['id\_animal'], c\_data['motif'], c\_data['diagnostic'], c\_data['cout'])

c.id\_consultation = c\_data['id\_consultation']

c.date = c\_data['date']

dossier.consultations.append(c)

clinique.dossiers\_medicaux[int(aid)] = dossier

## 7.4 Reconstruction des compteurs d’ID

Lors du chargement, on met à jour les compteurs de classes en prenant le maximum des IDs existants + 1 (pour éviter réutilisation d’ID).

#Ajuste les compteurs d'ID après le chargement des données

def remettre\_a\_jour\_compteurs(clinique):

if clinique.animaux:

Animal.\_id\_counter = max(a.id\_animal for a in clinique.animaux.values()) + 1

else:

Animal.\_id\_counter = 1

if clinique.proprietaires:

Proprietaire.\_id\_counter = max(p.id\_proprietaire for p in clinique.proprietaires.values()) + 1

else:

Proprietaire.\_id\_counter = 1

toutes\_consultations = [c for d in clinique.dossiers\_medicaux.values() for c in d.consultations]

if toutes\_consultations:

Consultation.\_id\_counter = max(c.id\_consultation for c in toutes\_consultations) + 1

else:

Consultation.\_id\_counter = 1

# 8 : Analyse de données et rapports

## 8.1 Objectifs analytiques

* Calculer le revenu par mois.
* Identifier les motifs les plus fréquents.
* Visualiser la répartition des consultations et l’âge des animaux.

## 8.2 Exemples de visualisations

Choisir une option: 8

Revenu par mois :

2025-09: 1330.00 DH

2025-10: 190.00 DH

Motifs les plus fréquents :

Boiterie: 2

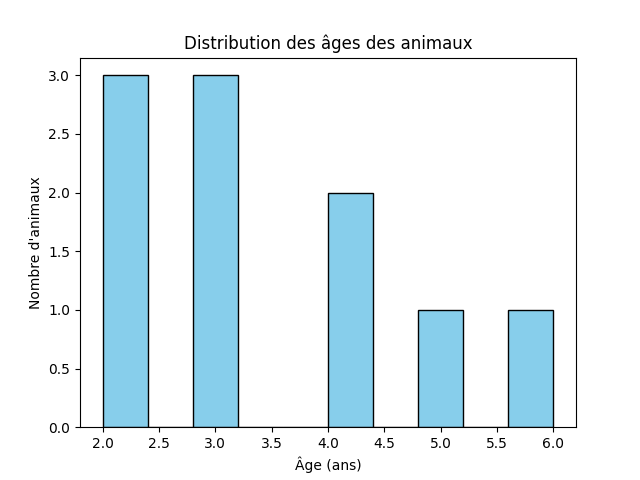
Perte dappetit: 1

Vaccination: 1

Eternuements frequents: 1

Perte de poils: 1

Graphique sauvegardé dans 'rapport\_patients.png'



|  |
| --- |
|  |

Exemple de graphique : distribution des consultations par mois (généré par Matplotlib)

## 8.3 Interprétation

Le rapport mensuel aide la clinique à prévoir les stocks et identifier les périodes de forte activité. Le top 5 des motifs guide la gestion des médicaments et fournit des indicateurs pour la prévention.

# 9 : Tests et validation

## 9.1 Stratégie de tests

* Tests unitaires pour les fonctions critiques (ajout, suppression, sérialisation).
* Tests d’intégration : scénario complet (ajout propriétaire → ajout animal → consultation → génération rapport).
* Tests manuels sur la CLI.

## 9.2 Cas de tests représentatifs

1. Ajouter propriétaire avec nom vide rejet.
2. Ajouter animal avec poids négatif rejet.
3. Enregistrer consultation pour animal inexistant message d’erreur.
4. Vérifier génération automatique des IDs après rechargement.

# 10 : Gestion de projet

## 10.1 Répartition des tâches

* Conception & UML .
* Implémentation POO .
* Persistance & JSON .
* Analyse & rapports .
* Tests & Documentation .

# 11 : Résultats obtenus

## 11.1 Livrables fournis

* Code source structuré (fichiers mentionnés).
* Fichier data.json exemple.
* Graphiques produits (rapport\_patients.png).
* Ce rapport détaillé.

## 11.2 Exemples d’exécution

===== CLINIQUE VÉTÉRINAIRE =====

1. Ajouter un propriétaire

2. Ajouter un animal

3. Lister tous les animaux

4. Lister animaux d'un propriétaire

5. Enregistrer une consultation

6. Historique d'un animal

7. Rechercher diagnostic

8. Générer rapport activité

0. Quitter

Choisir une option: 1

Nom du propriétaire (0 pour annuler): Amine

Adresse email (0 pour annuler): amine@gmail.com

✅ Propriétaire ajouté: ID: 6 Amine, amine@gmail.com, Animaux: []

Choisir une option: 2

Types disponibles: Chien, Chat

Type animal (0 pour annuler): chien

Nom (0 pour annuler): Appa

Âge (ans) (0 pour annuler): 3

Poids (kg) (0 pour annuler): 9

Race (0 pour annuler): Berger

Propriétaires disponibles:

ID: 1 Youssef, youssef@gmail.com, Animaux: [1, 2]

ID: 2 Khadija, khadija@gmail.com, Animaux: [3, 4]

ID: 3 Omar, omar@yahoo.com, Animaux: [5, 6]

ID: 4 Amina, amina@hotmail.com, Animaux: [7]

ID: 5 Rachid, rachid@gmail.com, Animaux: [8, 9, 10]

ID: 6 Amine, amine@gmail.com, Animaux: []

ID du propriétaire (0 pour annuler): 6

✅ Animal ajouté: ID: 11 Appa, Chien, 3 ans, 9.0 kg, Race: Berger

Choisir une option: 3

Liste des animaux :

ID: 1 Simo, Chien, 3 ans, 22.5 kg, Race: Berger Allemand

ID: 2 Bella, Chien, 3 ans, 8.0 kg, Race: Caniche

ID: 3 Mounia, Chat, 2 ans, 5.0 kg, Couleur pelage: Noir

ID: 4 Zaki, Chat, 2 ans, 6.5 kg, Couleur pelage: Blanc

ID: 5 Atlas, Chien, 5 ans, 30.0 kg, Race: Husky

ID: 6 Laila, Chat, 4 ans, 4.5 kg, Couleur pelage: Gris

ID: 7 Rif, Chien, 3 ans, 18.0 kg, Race: Pitbull

ID: 8 Sahara, Chat, 6 ans, 7.0 kg, Couleur pelage: Tigre

ID: 9 Noura, Chat, 4 ans, 5.5 kg, Couleur pelage: Blanc

ID: 10 Samir, Chien, 2 ans, 20.0 kg, Race: Labrador

ID: 11 Appa, Chien, 3 ans, 9.0 kg, Race: Berger

ID du propriétaire (0 pour annuler): 1

Animaux du propriétaire Youssef :

ID: 1 Simo, Chien, 3 ans, 22.5 kg, Race: Berger Allemand

ID: 2 Bella, Chien, 3 ans, 8.0 kg, Race: Caniche

Choisir une option: 5

ID de l'animal (0 pour annuler): 3

Motif (0 pour annuler): Faitigue

Diagnostic (0 pour annuler): 0

Opération annulée

Choisir une option: 5

ID de l'animal (0 pour annuler): 3

Motif (0 pour annuler): Fatigue

Diagnostic (0 pour annuler): Fievre

Coût (MAD) (0 pour annuler): 70

Consultation enregistrée: ID: 12 Animal 3, Date: 2025-10-03 12:04, Motif: Fatigue, Diagnostic: Fievre, Coût: 70.0 DH

Choisir une option: 6

ID de l'animal (0 pour annuler): 30

Aucun dossier médical pour cet animal

Choisir une option: 6

ID de l'animal (0 pour annuler): 5

Historique médical de l'animal Atlas :

ID: 6 Animal 5, Date: 2025-09-18 11:00, Motif: Toux persistante, Diagnostic: Bronchite canine, Coût: 180.0 DH

Choisir une option: 7

Mot-clé diagnostic (0 pour annuler): Fievre

Résultats:

ID: 12 Animal 3, Date: 2025-10-03 12:04, Motif: Fatigue, Diagnostic: Fievre, Coût: 70.0 DH

# 12 : Difficultés rencontrées et solutions

* **Sérialisation d’objets complexes** : utilisé conversion via .\_\_dict\_\_ et reconstruction explicite.
* **Maintien des compteurs d’ID** : solution = recalcul du maximum au chargement.
* **Validation centralisée** : création de fonctions utilitaires ‘demander\_entier‘, ‘demander\_float‘.

# 13 :Conclusion et perspectives

Le projet atteint ses objectifs initiaux : système CLI fonctionnel, persistance fiable et capacités d’analyse. Perspectives possibles :

* Migration vers une base relationnelle (SQLite).
* Développement d’une interface graphique (Tkinter / PyQt / web).
* API REST pour intégration multi-postes.
* Authentification & gestion des rôles.

# Instructions d’installation

1. Installer Python 3.8+.
2. Installer dépendances : pip install pandas matplotlib
3. Lancer : python main.py

# Bibliographie

* Documentation Python : <https://docs.python.org/3/>
* Pandas : <https://pandas.pydata.org/>
* Matplotlib : <https://matplotlib.org/>