Mini SI — Medical Project

Gestion d'un cabinet médical en Python

Rapport technique · Projet académique

Mahmoudi Sarah · 2026

Langage	Stockage	Visualisation	Statut
Python 3	Fichiers TXT	Matplotlib	Terminé

1. Présentation du projet

Mini SI — Medical Project est une application Python en mode console permettant de gérer un cabinet médical complet. Le projet illustre les fondamentaux de la programmation structurée en Python : gestion de fichiers texte, structures de données (pile, listes), validation des entrées et visualisation de données statistiques. Il s'inscrit dans le cadre d'un projet académique axé sur la maîtrise de la gestion de fichiers et de la programmation structurée.

Contexte académique : Projet Python sans base de données — toutes les données sont persistées dans des fichiers TXT structurés, simulant un système d'information léger.

2. Fonctionnalités principales

Gestion des patients

- Ajouter un nouveau patient avec ses informations personnelles
- Modifier les données d'un patient existant
- Supprimer un patient via son identifiant
- Afficher la liste complète des patients enregistrés

Gestion des rendez-vous

- Planifier un rendez-vous pour un patient
- Consulter les rendez-vous par date ou par patient
- Annuler ou modifier un rendez-vous existant

Gestion des ordonnances

- Générer une ordonnance pour un patient après consultation
- Associer des médicaments et posologies à une ordonnance
- Historiser les ordonnances par patient

Historique & traçabilité

- Suivi de l'historique médical complet par patient (structure Pile)
- Consultation des actes et visites précédentes
- Traçabilité des modifications (log des opérations)

Statistiques & visualisation

- Calcul de statistiques descriptives sur les consultations
- Génération de graphiques Matplotlib (répartition, fréquences)
- Export visuel pour analyse médicale

3. Architecture technique

L'application est organisée en modules fonctionnels distincts, chacun responsable d'un domaine métier. La persistance repose sur des fichiers TXT structurés (un fichier par entité), lus et écrits via les primitives File I/O de Python. La structure Pile est utilisée pour l'historique médical, garantissant un accès LIFO aux dernières interactions.

Module	Responsabilité	Stockage
patients.py	CRUD patients	patients.txt
appointments.py	Gestion RDV	appointments.txt
prescriptions.py	Ordonnances	prescriptions.txt
history.py	Historique (Pile)	history.txt
stats.py	Statistiques + graphiques	—
main.py	Menu interactif principal	—

---

4. Stack technique

Le projet utilise exclusivement des outils Python standard, sans dépendance à une base de données externe, conformément aux contraintes académiques.

Technologie	Version	Rôle dans le projet
Python	3.x	Langage principal — logique métier et CLI
File I/O (TXT)	Natif	Persistance des données (patients, RDV, ordonnar
Pile (Stack)	Natif	Structure de données pour l'historique médical
Matplotlib	3.x	Génération de graphiques statistiques
Jupyter Notebook	6.x+	Environnement d'exécution et présentation

---

5. Concepts informatiques illustrés

CRUD via File I/O

Toutes les opérations de création, lecture, mise à jour et suppression sont réalisées directement sur des fichiers TXT. Chaque ligne représente un enregistrement structuré, avec parsing manuel des champs.

Structure de données — Pile (Stack)

L'historique médical du patient est géré via une structure Pile (LIFO). Chaque nouvelle consultation est empilée ; la consultation la plus récente est accessible en premier, simulant un dossier médical chronologique inversé.

Validation des entrées

Chaque saisie utilisateur est validée avant traitement : vérification du format des dates, de l'existence d'un patient via son ID, et gestion des exceptions pour éviter les crashs applicatifs.

Statistiques descriptives

Le module stats.py calcule des indicateurs simples (nombre de consultations par période, répartition par type d'acte) et génère des visualisations Matplotlib (histogrammes, diagrammes en barres).

Menu interactif CLI

L'interface console propose un menu modulaire et hiérarchique permettant de naviguer entre les différents modules sans quitter l'application.

---

6. Flux de données — exemple patient

Le schéma ci-dessous illustre le cycle de vie d'un enregistrement patient dans le système, de la saisie à la persistance fichier :

1. Saisie	L'utilisateur entre les données patient via le menu CLI
2. Validation	Vérification du format et de l'unicité de l'ID patient
3. Structuration	Formatage en ligne structurée (séparateur |)
4. Écriture	Ajout dans patients.txt via File I/O (mode append)
5. Confirmation	Message de succès affiché à l'utilisateur

---

7. Résultats et apprentissages

Compétence	Acquis
Gestion File I/O Python	Lecture/écriture structurée sans ORM ni BDD

Structures de données	Implémentation d'une Pile pour l'historique
Validation & robustesse	Gestion des erreurs et cas limites (try/except)
Visualisation de données	Graphiques Matplotlib depuis données fichiers
Architecture modulaire	Séparation claire des responsabilités par module
Interface CLI	Menu interactif hiérarchique et navigable

8. Perspectives d'évolution

- Migration vers une base de données relationnelle (SQLite ou MySQL) pour remplacer les fichiers TXT
- Développement d'une interface graphique (Tkinter ou PyQt) pour améliorer l'ergonomie
- Ajout d'un moteur de recherche multi-critères (nom, date, pathologie)
- Export des données en CSV / Excel pour interopérabilité avec d'autres outils
- Système de rappel automatique de rendez-vous (notifications console ou email)
- Authentification médecin/secrétaire avec gestion des droits d'accès