**Rapport Détaillé d'Intégration de l'Intelligence Artificielle**

**SAE 302 : Développement d'Applications Communicantes**

Étudiant : Walid BADAOUI

Rôle : Lead Developer Java & Backend

Outils utilisés : ChatGPT (OpenAI), Gemini (Google) & Copilote (Github).

---

## 1. Introduction : Philosophie d'utilisation

Dans le cadre de la SAE 302, confronté à la nécessité de maîtriser simultanément plusieurs technologies nouvelles (JDBC, Architecture Réseau, Processus Système), j'ai fait le choix stratégique d'intégrer l'Intelligence Artificielle générative dans mon flux de travail.

Mon objectif n'était pas de déléguer la réalisation du projet, mais d'utiliser l'IA selon trois axes précis :

1. **Accélérateur de production** : Génération de code répétitif ou "boilerplate".
2. **Tuteur pédagogique** : Explication de concepts obscurs comme les Drivers JDBC ou les flux (Streams) Java.
3. **Analyste d'erreurs** : Débogage rapide des *stacktraces* (traces d'erreurs).

Ce rapport détaille, phase par phase, comment ce binôme Homme-Machine a permis la réalisation du module Java.

---

## 2. Phase d'Analyse et de Structuration (Prompt Engineering)

Avant d'écrire la moindre ligne de code, j'ai utilisé l'IA pour défricher le sujet qui me semblait complexe.

### 2.1. Clarification du Cahier des Charges

Le sujet mentionnait une interaction Java/Web/Android sans préciser l'infrastructure. J'étais confus sur la manière dont ces éléments allaient communiquer.

- **Mon Prompt :**

« Il faut faire quoi exactement et concrètement sur le projet, genre pour le TD 1, et comment ça marche s'il te plaît ? [...] Comment le fichier .db va communiquer avec le site web si on n'a pas de serveur ? »

- **Apport de l'IA :** L'IA a reformulé le sujet en m'expliquant le concept d'architecture centrée sur le fichier. Elle m'a fait comprendre que SQLite était "Serverless" (sans processus serveur), et que le fichier .db servirait de point

d'échange (pivot) entre mon programme Java (écrivain) et le site Web (lecteur).

- **Validation humaine :** J'ai confronté cette explication avec les consignes du professeur pour valider que nous n'avions pas besoin d'installer MAMP ou WAMP, ce qui a simplifié le démarrage du projet.

### 2.2. Organisation de l'équipe (Matrice RACI)

**Apport de l'IA :** L'IA a découpé les responsabilités techniques. créant  les rôles Java Backend en distinguant deux actions que je confondais :

  - **Scanner :** Lancer l'outil (ex: nmap).
  - **Parser :** Lire le texte brut renvoyé par l'outil pour le transformer en objet.

Cela m'a permis de structurer mes classes Java futures (ScannerReseau vs les méthodes de parsing).

---

## 3. L'IA comme Tuteur de Programmation (Pédagogie)

C'est l'aspect le plus important de mon utilisation. Je ne voulais pas copier du code que je ne comprenais pas. J'ai systématiquement demandé à l'IA d'agir comme un professeur.

### 3.1. Compréhension de la POO et des méthodes Object

Lors de la création de la classe Faille, l'IA a généré des méthodes @Override.

- **Mon Prompt :**

« Juste, écris le code comme un débutant et explique tout. Mets de longs commentaires s'il le faut car je ne comprends pas ça : @Override public String toString()... »

- **L'Explication reçue :** L'IA m'a expliqué que sans toString(), Java affiche l'adresse mémoire de l'objet (ex: Faille@45a877). Pour mon débogage, j'avais besoin de voir le contenu (IP, Nom).
- **Intégration :** J'ai conservé cette méthode car elle m'a été indispensable plus tard pour vérifier dans la console que mes requêtes SQL renvoyaient bien les bonnes données.

### 3.2. Apprentissage de la syntaxe JDBC (Try-with-resources)

La gestion des bases de données en Java est verbeuse. J'ai demandé à l'IA de m'expliquer la gestion des connexions.

- **Mon Prompt :**

« Je ne comprends pas les deux traits : if (this.connection == null || this.connection.isClosed()) »

et

« Explique chaque ligne [...] void open() / void close() »

- **Apprentissage :** L'IA m'a enseigné le cycle de vie d'une connexion. J'ai appris qu'il fallait vérifier si l'objet était null (jamais ouvert) OU closed (ouvert puis fermé) avant de tenter une requête. Elle m'a aussi introduit à la syntaxe try (Statement stmt = ...) qui ferme automatiquement les ressources, évitant des fuites de mémoire que j'aurais sûrement créées manuellement.

---

### 4. Génération et Refactoring de Code (Productivité)

Une fois les concepts compris, j'ai utilisé l'IA pour générer le code répétitif (Boilerplate) et les structures complexes.

#### 4.1. Le Module DatabaseManager (CRUD)

Écrire les requêtes SQL dans du Java est fastidieux. J'ai utilisé l'IA pour générer le squelette du CRUD.

- **La Demande :** J'ai fourni la structure de la table (id, nom, ip...) et demandé les méthodes Java correspondantes.
- **La Valeur Ajoutée de l'IA (Correction d'erreur logique) :** Initialement, je ne savais pas comment récupérer l'ID d'une faille que je venais d'insérer (pour l'afficher). L'IA a intégré dans le code la fonction Statement.RETURN_GENERATED_KEYS et la récupération via rs.getGeneratedKeys(). C'est une subtilité technique que je n'aurais pas trouvée seul rapidement.

#### 4.2. Intégration Système et Réseau (ScannerReseau)

C'était la partie la plus critique : faire sortir Java de sa machine virtuelle pour exécuter des commandes Linux (Nmap).

- **Mon Prompt :**

« Explique et apprends-moi comment le code pour lancer Nmap doit marcher [...] Et comment Java interagit avec Nmap ? »

- **Solution Technique (ProcessBuilder) :** L'IA m'a fourni le code utilisant ProcessBuilder.
    - *Difficulté surmontée :* J'ai appris grâce au code généré qu'il fallait capturer le flux de sortie (InputStream) du processus pour lire le résultat. Sans cela, la commande s'exécutait en "silence".

- *Adaptation :* L'IA proposait des scans complexes. J'ai simplifié le code pour qu'il soit modulaire, ajoutant une méthode runToolCommand générique qui peut lancer n'importe quelle commande Linux, pas juste Nmap.

### 4.3. Gestion des IPs et des Réseaux (CIDR)

Je voulais implémenter le scan d'une plage réseau (/24). L'algorithme de conversion IP String -> Entier binaire était trop complexe à écrire de zéro.

- **Apport IA :** L'IA a généré les méthodes de calcul de masque de sous-réseau (Bitwise operations) pour transformer 192.168.1.0/24 en une liste de 254 adresses IP cibles. J'ai intégré ce code dans ScannerReseau.

---

## 5. L'IA comme Assistant de Débogage (Troubleshooting)

C'est ici que le gain de temps a été le plus flagrant. Au lieu de bloquer des heures sur des erreurs cryptiques, je les soumettais à l'analyse de l'IA.

### 5.1. Le cas du Driver JDBC introuvable

- **L'Erreur :** No suitable driver found for jdbc:sqlite:failles.db
- **Analyse IA :** L'IA a détecté que le code était bon, mais que l'environnement d'exécution (le classpath) était incomplet.
- La Solution : Elle ne m'a pas dit de changer le code, mais de changer ma commande de lancement :

java -cp ".;sqlite-jdbc-3.51.0.0.jar" App

Elle m'a expliqué que Java ne "devine" pas où sont les librairies externes.

### 5.2. Le cas du NullPointerException

- **L'Erreur :** Cannot invoke "java.sql.Connection.createStatement()" because "this.connection" is null
- **Analyse IA :** J'appelais ma méthode createTable sans avoir appelé open avant.
- **Correction Architecturelle :** L'IA m'a suggéré d'appeler automatiquement open() au début de chaque méthode publique du DatabaseManager, rendant la classe plus robuste et moins dépendante de l'ordre d'appel dans le main.

---

## 6. Conclusion : Bilan de la collaboration

L'IA n'a pas "fait le projet à ma place". Elle a agi comme un accélérateur de compétence.

1. **Gain de temps :** Sans l'IA, j'aurais probablement réussi à faire le CRUD SQL, mais j'aurais mis 3 fois plus de temps à déboguer les erreurs de Driver et de connexion.
2. **Qualité du code :** Sans l'IA, je n'aurais pas implémenté des patterns propres comme l'interface ScanTool ou le try-with-resources, je serais resté sur du code procédural basique et moins fiable.
3. **Focus Métier :** L'IA m'a permis de me concentrer sur l'architecture et la logique métier (comment les pièces s'emboîtent) plutôt que de lutter contre la syntaxe verbeuse de Java.

J'ai appris qu'une bonne utilisation de l'IA demande de savoir poser les bonnes questions (**Prompt Engineering**) et surtout d'avoir assez de recul critique pour tester et intégrer le code produit.