📊 Rapport de Qualité du Code

Consultator - Application de Gestion

Analyse complète de la qualité du code  
Métriques, sécurité et recommandations d'amélioration  
Généré automatiquement

**Version 1.0.0**Généré le 15/09/2025 à 15:35:17  
Branche: Master

📊 RAPPORT DE QUALITÉ DU CODE - CONSULTATOR

**Date de génération:** 15/09/2025 15:34:59

**Version analysée:** Master Branch

📈 MÉTRIQUES GLOBALES

📁 Structure du projet

* \*\*Total fichiers:\*\* 225
* \*\*Fichiers Python:\*\* 162
* \*\*Fichiers de test:\*\* 47
* \*\*Ratio test/code:\*\* 0.41

📝 Métriques de code

* \*\*Lignes totales:\*\* 0
* \*\*Lignes de code:\*\* 41,972
* \*\*Lignes vides:\*\* 9,587
* \*\*Commentaires:\*\* 3,672
* \*\*Ratio commentaires/code:\*\* 8.75%

🏗️ Architecture

* \*\*Fonctions:\*\* 1247
* \*\*Classes:\*\* 113
* \*\*Imports:\*\* 1272
* \*\*Complexité moyenne:\*\* 0.14 warnings/fonction

🔒 SÉCURITÉ

⚠️ Problèmes de sécurité détectés: 20

**Par sévérité:**

* 🔴 Critique: 0
* 🟠 Haute: 5
* 🟡 Moyenne: 15
* 🟢 Faible: 0

HARDCODED\_SECRET

* \*\*Sévérité:\*\* MEDIUM
* \*\*Description:\*\* Secret potentiellement codé en dur
* \*\*Fichier:\*\* .\enhance\_tests\_coverage.py

SQL\_INJECTION

* \*\*Sévérité:\*\* HIGH
* \*\*Description:\*\* Possible injection SQL détectée
* \*\*Fichier:\*\* .\generate\_quality\_report.py

CODE\_INJECTION

* \*\*Sévérité:\*\* HIGH
* \*\*Description:\*\* Usage de eval() détecté
* \*\*Fichier:\*\* .\generate\_quality\_report.py

INSECURE\_DESERIALIZATION

* \*\*Sévérité:\*\* MEDIUM
* \*\*Description:\*\* Usage de pickle détecté (risque de désérialisation)
* \*\*Fichier:\*\* .\generate\_quality\_report.py

SQL\_INJECTION

* \*\*Sévérité:\*\* HIGH
* \*\*Description:\*\* Possible injection SQL détectée
* \*\*Fichier:\*\* .\generate\_quality\_report\_final.py

CODE\_INJECTION

* \*\*Sévérité:\*\* HIGH
* \*\*Description:\*\* Usage de eval() détecté
* \*\*Fichier:\*\* .\generate\_word\_report.py

HARDCODED\_SECRET

* \*\*Sévérité:\*\* MEDIUM
* \*\*Description:\*\* Secret potentiellement codé en dur
* \*\*Fichier:\*\* .\sonar\_integration.py

HARDCODED\_SECRET

* \*\*Sévérité:\*\* MEDIUM
* \*\*Description:\*\* Secret potentiellement codé en dur
* \*\*Fichier:\*\* .\app\pages\_modules\business\_managers.py

HARDCODED\_SECRET

* \*\*Sévérité:\*\* MEDIUM
* \*\*Description:\*\* Secret potentiellement codé en dur
* \*\*Fichier:\*\* .\app\pages\_modules\consultants.py

HARDCODED\_SECRET

* \*\*Sévérité:\*\* MEDIUM
* \*\*Description:\*\* Secret potentiellement codé en dur
* \*\*Fichier:\*\* .\app\pages\_modules\consultants\_clean.py

🧹 QUALITÉ DU CODE

⚠️ Problèmes de qualité détectés: {len(analysis\_results['code\_quality\_issues'])}

**Par sévérité:**

* 🔴 Critique: 0
* 🟠 Haute: 0
* 🟡 Moyenne: 0
* 🟢 Faible: 550

LINE\_TOO\_LONG

* \*\*Sévérité:\*\* LOW
* \*\*Description:\*\* Ligne trop longue (141 caractères)
* \*\*Fichier:\*\* .\analyze\_docstrings\_detailed.py
* \*\*Ligne:\*\* 57

UNUSED\_IMPORT

* \*\*Sévérité:\*\* LOW
* \*\*Description:\*\* Import potentiellement non utilisé: os
* \*\*Fichier:\*\* .\analyze\_docstrings\_detailed.py
* \*\*Ligne:\*\* 2

UNUSED\_IMPORT

* \*\*Sévérité:\*\* LOW
* \*\*Description:\*\* Import potentiellement non utilisé: Path
* \*\*Fichier:\*\* .\analyze\_docstrings\_detailed.py
* \*\*Ligne:\*\* 3

UNUSED\_IMPORT

* \*\*Sévérité:\*\* LOW
* \*\*Description:\*\* Import potentiellement non utilisé: os
* \*\*Fichier:\*\* .\analyze\_documentation.py
* \*\*Ligne:\*\* 7

UNUSED\_IMPORT

* \*\*Sévérité:\*\* LOW
* \*\*Description:\*\* Import potentiellement non utilisé: ast
* \*\*Fichier:\*\* .\analyze\_documentation.py
* \*\*Ligne:\*\* 9

UNUSED\_IMPORT

* \*\*Sévérité:\*\* LOW
* \*\*Description:\*\* Import potentiellement non utilisé: Path
* \*\*Fichier:\*\* .\analyze\_documentation.py
* \*\*Ligne:\*\* 10

UNUSED\_IMPORT

* \*\*Sévérité:\*\* LOW
* \*\*Description:\*\* Import potentiellement non utilisé: Dict
* \*\*Fichier:\*\* .\analyze\_documentation.py
* \*\*Ligne:\*\* 11

LINE\_TOO\_LONG

* \*\*Sévérité:\*\* LOW
* \*\*Description:\*\* Ligne trop longue (144 caractères)
* \*\*Fichier:\*\* .\analyze\_documentation\_complete.py
* \*\*Ligne:\*\* 88

LINE\_TOO\_LONG

* \*\*Sévérité:\*\* LOW
* \*\*Description:\*\* Ligne trop longue (140 caractères)
* \*\*Fichier:\*\* .\analyze\_documentation\_complete.py
* \*\*Ligne:\*\* 92

UNUSED\_IMPORT

* \*\*Sévérité:\*\* LOW
* \*\*Description:\*\* Import potentiellement non utilisé: ast
* \*\*Fichier:\*\* .\analyze\_documentation\_complete.py
* \*\*Ligne:\*\* 1

🎯 RECOMMANDATIONS D'AMÉLIORATION

🔴 PRIORITÉ HAUTE

1. **Sécurité**

* Supprimer les secrets codés en dur
* Remplacer eval() par des alternatives sûres
* Sécuriser la désérialisation pickle
* Implémenter une validation d'entrée robuste

2. **Performance**

* Optimiser les fonctions complexes (>10 décisions)
* Implémenter la mise en cache pour les requêtes fréquentes
* Réduire la taille des fonctions longues (>50 lignes)

🟡 PRIORITÉ MOYENNE

3. **Maintenabilité**

* Réduire la longueur des lignes (>120 caractères)
* Supprimer les imports non utilisés
* Améliorer la couverture de tests (actuellement ~{metrics['test\_files']/max(metrics['python\_files']-metrics['test\_files'], 1)\*100:.1f}%)

4. **Documentation**

* Augmenter le ratio commentaires/code (actuellement {metrics['comment\_lines']/max(metrics['code\_lines'], 1):.1%})
* Documenter les fonctions complexes
* Créer une documentation API

🟢 PRIORITÉ FAIBLE

5. **Standards**

* Uniformiser le style de code (PEP 8)
* Implémenter des types hints complets
* Configurer des hooks pre-commit

📊 SCORES DE QUALITÉ

Sécurité: {'🔴 CRITIQUE' if len(analysis\_results['security\_findings']) > 5 else '🟠 À AMÉLIORER' if len(analysis\_results['security\_findings']) > 0 else '🟢 BON'}

Maintenabilité: {'🔴 À RÉFACTORER' if metrics['complexity\_warnings'] > 20 else '🟠 À OPTIMISER' if metrics['complexity\_warnings'] > 10 else '🟢 BONNE'}

Testabilité: {'🔴 INSUFFISANTE' if metrics['test\_files']/max(metrics['python\_files']-metrics['test\_files'], 1) < 0.5 else '🟠 À AMÉLIORER' if metrics['test\_files']/max(metrics['python\_files']-metrics['test\_files'], 1) < 0.8 else '🟢 EXCELLENTE'}

Documentation: {'🔴 INSUFFISANTE' if metrics['comment\_lines']/max(metrics['code\_lines'], 1) < 0.1 else '🟠 À AMÉLIORER' if metrics['comment\_lines']/max(metrics['code\_lines'], 1) < 0.2 else '🟢 EXCELLENTE'}

---

*Rapport généré automatiquement par l'analyseur de qualité du code*