Parcours B — RAG Code & Documentation

Adservio | Dr Olivier Vitrac

2026-02-03

Contents

1 Parcours B — RAG Code & Documentation

"Rendre les contradictions visibles"

1.1 Contexte & Enjeu

La documentation ment. Pas volontairement — elle devient obsolète, incomplète, ou décalée par rapport au code réel.

Un LLM classique répond avec assurance, même quand documentation et code se contredisent. Il génère une réponse *plausible* mais potentiellement *fausse*.

Un système RAG bien conçu ne cherche pas à *répondre* — il cherche à **aligner** ce qu'il sait avec ce qui existe. Et quand l'alignement échoue, il le dit.

Enjeu pour Adservio : Construire des systèmes qui détectent les incohérences plutôt que de les masquer.

1.2 Objectif Pédagogique

À l'issue de ce parcours, vous serez capables de :

1. **Indexer** du code et de la documentation de manière structurée
2. **Interroger** un index avec des questions précises et falsifiables
3. **Détecter** les contradictions entre code, docs, et réalité
4. **Produire** des réponses avec citations vérifiables
5. **Accepter** "inconnu" ou "incohérent" comme réponses valides

---

1.3 Cahier des Charges Minimal

1.3.1 Périmètre

Choisissez un repo Adservio réel (ou `<your-repo>` en fallback) contenant : - Un `README.md` - Un dossier `/docs` ou documentation inline - Du code source (Python, Java, ou autre)

1.3.2 Corpus à Indexer

Source	Contenu
`README.md`	Description générale, installation, usage
`/docs/*.md`	Documentation technique
Code source	Signatures, docstrings, commentaires
(Optionnel)	Tests, CHANGELOG, issues

1.3.3 Fonctionnalités du Système

1. **Indexation**
   - Parser le code (AST ou regex)
   - Parser la documentation (Markdown)
   - Créer un index interrogeable
2. **Retrieval**
   - Questions en langage naturel
   - Récupération des passages pertinents
   - **Avec citations** (fichier:ligne ou section)
3. **Détection d'Incohérences**
   - Feature documentée mais non implémentée

- Feature implémentée mais non documentée
- Exemples obsolètes
- Signatures qui ne matchent pas

4. **Réponses Structurées**
   - Réponse à la question
   - Sources (code ET docs)
   - Flag de cohérence (aligné / incohérent / inconnu)

1.3.4 Scope Minimal Viable

- 1 repo indexé (code + docs)
- 5 questions testées
- Au moins 1 incohérence détectée et documentée

1.4 Contraintes Non-Négociables

Contrainte	Justification
Citations obligatoires	Toute affirmation doit pointer vers une source
“Inconnu” est valide	Mieux vaut avouer l’ignorance que halluciner
Pas de génération pure	Le système *retrouve*, il n’*invente* pas
Incohérences explicites	Si code ≠ docs, le dire clairement
Un livrable sans trace exploitable est considéré comme non livré	Règle d’or Adservio

1.4.1 Ce qui est Interdit

- Répondre sans citer de source
- Ignorer une contradiction détectée
- Prétendre que tout est cohérent sans vérification
- Utiliser uniquement des embeddings sans BM25/lexical (ou vice-versa)

1.5 Auditabilité & Traçabilité

Toute action de l’agent doit produire une trace exploitable par un humain : décisions, appels d’outils, hypothèses, erreurs, sorties intermédiaires.

1.5.1 Format de Réponse Attendu

```
{
  "query": "La fonction parse_config accepte-t-elle un fichier YAML ?",
  "answer": "Selon la documentation, oui. Selon le code, non.",
  "confidence": "contradiction_detected",
  "sources": {
    "documentation": [
      {"file": "docs/config.md", "section": "File Formats", "quote": "Supports JSON, YAML, ar
    ],
    "code": [
      {"file": "src/config.py", "line": 45, "quote": "def parse_config(path: str) -> dict: #
```

```
    ]
  },
  "inconsistency": {
    "type": "doc_code_mismatch",
    "description": "Documentation claims YAML support, code only handles JSON",
    "severity": "high"
  }
}
```

1.5.2 Types d'Incohérences à Tracer

Type	Description
`doc_code_mismatch`	Documentation dit X, code fait Y
`undocumented_feature`	Fonction existe mais pas documentée
`phantom_feature`	Documentée mais non implémentée
`stale_example`	Exemple ne compile/fonctionne plus
`signature_mismatch`	Paramètres documentés ≠ paramètres réels

1.6 Livrables Attendus

Livrable	Format	Description
Script d'indexation	Python	Code pour indexer le corpus
Script de requête	Python	Interface pour interroger l'index
Index	FAISS/BM25/autre	Index persisté
Rapport d'audit	Markdown	Synthèse des incohérences détectées
Traces de requêtes	JSON	Log des questions/réponses avec sources

1.6.1 Structure Suggérée

```
rag_code_docs/
├── indexer.py           # Script d'indexation
├── query.py             # Script de requête
├── index/
│   ├── vectors.faiss    # Index vectoriel (si utilisé)
│   └── bm25.pkl         # Index BM25 (si utilisé)
├── traces/
│   └── queries_*.json   # Traces des requêtes
└── audit_report.md      # Rapport final
```

1.7 Critères d'Évaluation

Critère	Poids	Indicateurs
Qualité de l'indexation	20%	Couverture du corpus, structure préservée

Critère	Poids	Indicateurs
Pertinence du retrieval	25%	Passages retournés répondent à la question
Détection d'incohérences	25%	Contradictions identifiées et classifiées
Citations	15%	Toute réponse pointe vers des sources vérifiables
Traçabilité	15%	Logs complets et exploitables

1.7.1 Barème

- **Insuffisant** : Index créé mais retrieval non fonctionnel
- **Passable** : Retrieval fonctionne mais sans citations ni détection
- **Satisfaisant** : Citations présentes, au moins 1 incohérence détectée
- **Excellent** : Système robuste, incohérences classifiées, propositions de fix

---

1.8 Pièges & Anti-Patterns

1.8.1 Piège 1 : L'embedding fetishism

Tout vectoriser sans réfléchir. Les embeddings capturent la sémantique, pas la précision. → **Solution** : Combiner dense (FAISS) + sparse (BM25) pour le meilleur des deux mondes.

1.8.2 Piège 2 : La réponse confiante mais fausse

Le LLM génère une réponse fluide qui ne correspond à aucune source. → **Solution** : Forcer la citation. Pas de source = pas de réponse.

1.8.3 Piège 3 : Le corpus mal découpé

Chunks trop grands → bruit. Chunks trop petits → perte de contexte. → **Solution** : Découper par unité sémantique (fonction, section, paragraphe).

1.8.4 Piège 4 : Ignorer les négatifs

Le système ne trouve rien → il invente. → **Solution** : "Je n'ai pas trouvé d'information sur ce sujet" est une réponse valide.

1.8.5 Piège 5 : La cohérence apparente

Code et docs disent la même chose… mais les deux sont faux. → **Solution** : Croiser avec l'exécution réelle quand possible.

1.9 Retour Critique & Limites de l'Agent

À compléter **obligatoirement** à la fin du parcours :

1. **Qu'est-ce que le système a *semblé* bien faire mais n'a pas réellement fait ?**
2. **Où lui avez-vous fait confiance à tort ?**

3. **Quelle hypothèse *vous* avez faite qui s'est révélée fausse ?**
4. **Qu'interdiriez-vous à ce système en production ?**
5. **Qu'est-ce qui vous a le plus surpris ?**

1.10 Questions Croisées (Préparation Demo Day)

1.10.1 Vers le parcours A (Testing)

- Comment votre détection d'incohérences pourrait-elle guider la génération de tests ?
- Un test qui échoue est-il une "incohérence" au sens de votre système ?

1.10.2 Vers le parcours C (MCP)

- Quels outils déterministes renforceraient la fiabilité de vos réponses ?
- Comment exposer votre index via MCP pour d'autres agents ?

1.10.3 Intégration Future

- Votre système pourrait-il générer automatiquement des PRs de correction doc ?
- Comment intégrer votre audit dans une CI/CD ?

1.11 Ressources

- FAISS documentation
- BM25 avec rank-bm25
- Sentence Transformers
- LangChain RAG patterns

Parcours B — RAG Code & Documentation — Adservio Workshop