Migration RAG OpenWebUI

Documentation technique

srv18 – Wiki PCStreet

BGE-M3 + Reranker + Qwen 2.5 14B

Serveur cible	srv18 (192.168.1.18)
Date de migration	25–26 mai 2026
Version OpenWebUI	0.8.12
Système	Debian 13
GPU	NVIDIA RTX 3060 (12 Go)

Document généré dans le cadre de la mise à niveau du RAG personnel.
À conserver pour reproduction ou diagnostic ultérieur.

Table des matières

1 Objectifs et résultats **2**
1.1 Contexte . . . 2
1.2 Évolution des composants . . . 2
1.3 Résultats observés . . . 2

2 Architecture finale **2**
2.1 Contraintes matérielles . . . 3

3 Composants installés **3**
3.1 LLM personnalisé : qwen-rag . . . 3
3.2 Modèle d'embeddings : bge-m3 . . . 3
3.3 Reranker : BAAI/bge-reranker-v2-m3 . . . 4

4 Configuration OpenWebUI **4**
4.1 Override systemd . . . 4
4.2 Paramètres dans l'interface web . . . 4
4.2.1 Section Général . . . 5
4.2.2 Section Embedding . . . 5
4.2.3 Section Récupération . . . 5

5 Migration de la base vectorielle **5**
5.1 Problématique . . . 5
5.2 Procédure de migration . . . 5
5.3 Monitoring de la réindexation . . . 6

6 Bugs rencontrés et solutions **6**
6.1 Bug : FETCH_URL_MAX_CONTENT_LENGTH not found . . . 6
6.2 Bug : Collection expecting embedding with dimension of 768 . . . 7
6.3 Piège systemctl edit avec service défini en /etc/systemd/system/ . . . 7

7 Annexe : durcissement Ollama (sécurité) **8**
7.1 Contexte . . . 8
7.2 Durcissement appliqué (les 3 serveurs) . . . 8
7.3 Nettoyage des modèles malveillants (srv18) . . . 8
7.4 Désinstallation d'OpenWebUI sur mail et backup . . . 8

8 Annexe : commandes utiles **8**
8.1 Inspection de la base ChromaDB . . . 8
8.2 Inspection de la base OpenWebUI . . . 9
8.3 Monitoring Ollama et VRAM . . . 9

9 Restauration en cas de problème **9**

1. Objectifs et résultats

1.1 Contexte

Le wiki personnel *Wiki PCStreet* (journal technique de la maison de Villennes : chauffage, pompes, plancher chauffant, etc.) était déjà indexé dans OpenWebUI avec **1953 fichiers** mais la qualité des réponses RAG dépendait largement du LLM utilisé et la pertinence des chunks récupérés laissait à désirer.

L'objectif de cette migration est d'améliorer trois axes simultanément :

— **Qualité du retrieval** : embeddings multilingues plus précis en français

— **Qualité de la sélection** : reranker cross-encoder pour affiner le top-K

— **Qualité de la génération** : LLM local Qwen 2.5 14B avec system prompt RAG strict, en parallèle des LLM distants OpenRouter conservés

1.2 Évolution des composants

Composant	Avant	Après
LLM par défaut	varié (OpenRouter)	`qwen-rag` (dérivé Qwen 2.5 14B)
Embeddings	`nomic-embed-text` (768 dim)	`bge-m3` (1024 dim)
Reranker	aucun	`BAAI/bge-reranker-v2-m3` (CPU)
Recherche	vectorielle simple	hybride (vectoriel + BM25)

1.3 Résultats observés

— **1953 fichiers** réindexés en **6396 chunks** (~3.3 chunks/fichier)

— Base vectorielle ChromaDB : **73 Mo** (vs 796 Mo avant, plus propre)

— Une seule collection ChromaDB en dimension 1024 (vs 2034 collections en 768 avant)

— Durée de réindexation : ~12 heures sur RTX 3060 partagée avec le bureau

2. Architecture finale

```
[Question utilisateur dans OpenWebUI]
            |
            v
[BGE-M3 via Ollama] -- embedding de la question (vecteur 1024d)
            |
            v
[ChromaDB] -- recherche hybride (vectoriel + BM25)
              retourne top-20 chunks
            |
            v
[BGE-reranker-v2-m3 sur CPU] -- reclassement
                                garde top-5
            |
            v
[qwen-rag via Ollama] -- generation avec system prompt RAG strict
            |
            v
[Reponse affichee dans l'UI]
```

2.1 Contraintes matérielles

Élément	VRAM (RTX 3060 12 Go)
Bureau MATE + Xorg + Firefox	~600 Mo
`qwen-rag` (`num_ctx` 12288)	~10.4 Go
`bge-m3`	~1.6 Go
Reranker (forcé CPU)	0 Go (en RAM)
Cohabitation Qwen + BGE-M3	**impossible**

Stratégie adoptée : swap automatique d'Ollama. Quand BGE-M3 est sollicité (réindexation, embedding de question), Qwen est déchargé. Quand Qwen est sollicité (génération de réponse), BGE-M3 est déchargé. Latence de ~5-10 sec lors du basculement, acceptable pour un usage personnel.

3. Composants installés

3.1 LLM personnalisé : qwen-rag

Modèle dérivé de `qwen2.5:14b-instruct-q4_K_M` avec un Modelfile personnalisé pour le RAG.
Fichier : `/root/Modelfile.qwen-rag`

```
FROM qwen2.5:14b-instruct-q4_K_M

PARAMETER num_ctx 12288
PARAMETER temperature 0.2
PARAMETER top_p 0.9
PARAMETER repeat_penalty 1.05

SYSTEM """Tu es un assistant qui repond aux questions en t'appuyant
exclusivement sur le contexte fourni. Regles strictes :
1. Reponds uniquement a partir des informations presentes dans le contexte.
2. Si l'information n'est pas dans le contexte, dis explicitement :
   "L'information ne figure pas dans la documentation fournie."
3. Cite les sources entre crochets quand c'est pertinent.
4. Reste concis, factuel et precis, en francais.
5. Si plusieurs sources se contredisent, signale-le."""
```

Création :

```
ollama create qwen-rag -f /root/Modelfile.qwen-rag
ollama list | grep qwen-rag
```

Choix de paramètres :

— `num_ctx 12288` : 12K tokens, compromis VRAM/utilité RAG

— `temperature 0.2` : favorise la factualité

— `top_p 0.9` : diversité modérée

— `repeat_penalty 1.05` : légère pénalité de répétition

Piège : `keep_alive` n'est PAS un paramètre Modelfile valide. C'est un paramètre d'*appel* (requête API ou ligne de commande). L'inclure dans le Modelfile provoque l'erreur `Error: unknown parameter 'keep_alive'` au moment du `ollama create`.

3.2 Modèle d'embeddings : bge-m3

```
ollama pull bge-m3
```

— Dimension : **1024**
— Multilingue avec excellent support du français
— Contexte natif jusqu'à 8192 tokens
— Empreinte VRAM : ~1.6 Go en inférence

Test :

```
curl -s http://127.0.0.1:11434/api/embeddings -d '{
  "model": "bge-m3",
  "prompt": "Test embedding en francais"
}' | python3 -c "import sys,json; d=json.load(sys.stdin); \
print(f'Dimension: {len(d[\"embedding\"])}')"
```

Résultat attendu : `Dimension: 1024`

3.3 Reranker : BAAI/bge-reranker-v2-m3

— Téléchargé automatiquement par OpenWebUI au premier usage
— Cache : `/root/.cache/huggingface/hub/`
— Taille : ~600 Mo
— Exécution : **CPU** (forcée pour préserver la VRAM)
— Latence : ~200 ms pour reranker 20 chunks

4. Configuration OpenWebUI

4.1 Override systemd

Fichier : `/etc/systemd/system/open-webui.service.d/override.conf`

```
[Service]
Environment="RAG_RERANKING_MODEL_AUTO_UPDATE=true"
Environment="SENTENCE_TRANSFORMERS_HOME=/root/.cache/huggingface"
Environment="CUDA_VISIBLE_DEVICES="
```

Application :

```
mkdir -p /etc/systemd/system/open-webui.service.d/
# (Ecrire le fichier override.conf ci-dessus)
systemctl daemon-reload
systemctl restart open-webui
```

Explication des variables :

— `RAG_RERANKING_MODEL_AUTO_UPDATE=true` : autorise OpenWebUI à télécharger le reranker depuis HuggingFace
— `SENTENCE_TRANSFORMERS_HOME` : emplacement du cache des modèles
— `CUDA_VISIBLE_DEVICES=""` (vide) : masque la GPU pour OpenWebUI, forçant le reranker sur CPU. **N'affecte PAS Ollama** qui tourne dans un autre service systemd

4.2 Paramètres dans l'interface web

Accès : *Panneau d'administration* → *Paramètres* → *Documents*

4.2.1 Section Général

Taille des chunks	`1000`
Chevauchement	`150`

4.2.2 Section Embedding

Moteur du modèle d'embedding	`Ollama`
Modèle d'embedding	`bge-m3:latest`
Taille du lot d'embedding	`8`

4.2.3 Section Récupération

Recherche hybride	**Activée**
Modèle de reclassement	`BAAI/bge-reranker-v2-m3`
Top K	`20`
Top K du reclassement	`5`
Seuil de pertinence	`0.3`

5. Migration de la base vectorielle

5.1 Problématique

Le changement d'embedder de `nomic-embed-text` (768 dim) vers `bge-m3` (1024 dim) est incompatible avec la base ChromaDB existante. ChromaDB refuse l'insertion avec :

```
chromadb.errors.InvalidArgumentError:
Collection expecting embedding with dimension of 768, got 1024
```

Limitation : le bouton *« Réindexer les vecteurs »* d'OpenWebUI ne recrée PAS la collection avec la nouvelle dimension. Il faut supprimer la base ChromaDB manuellement.

5.2 Procédure de migration

Étape 1 – Backup obligatoire

```
DATA_DIR="/root/IA/open-webui.debian12/venv/lib/python3.12/\
site-packages/open_webui/data"
BACKUP_DIR="/root/backups/open-webui-before-chroma-reset-\
$(date +%Y%m%d-%H%M%S)"

mkdir -p "$BACKUP_DIR"
cp -a "$DATA_DIR/vector_db" "$BACKUP_DIR/vector_db"
cp -a "$DATA_DIR/webui.db" "$BACKUP_DIR/webui.db"
du -sh "$BACKUP_DIR/"
```

Étape 2 – Arrêt de OpenWebUI

```
# Restart=always necessite disable, pas un simple stop
systemctl disable --now open-webui

# Tuer tout processus residuel
pgrep -f "open-webui serve" | xargs -r kill -9
sleep 2

# Verifier
pgrep -af "open-webui serve" || echo "OK, plus de process"
ss -tlnp | grep 8081 || echo "OK, port libre"
```

Étape 3 – Suppression de la base ChromaDB

```
# Suppression du dossier vector_db
# (les fichiers sources restent dans webui.db et uploads/)
rm -rf "$DATA_DIR/vector_db"
```

Étape 4 – Redémarrage

```
systemctl enable open-webui
systemctl start open-webui
sleep 20

systemctl status open-webui --no-pager | head -5
curl -s -o /dev/null -w "HTTP %{http_code}\n" \
  http://127.0.0.1:8081/
```

OpenWebUI recrée alors automatiquement le dossier `vector_db/` avec une base `chroma.sqlite3` vide.

Étape 5 – Lancement de la réindexation

Dans l'interface web :

1. *Espace de travail* → *Connaissance* → *Wiki PCStreet*
2. Cliquer sur « **Réindexer les vecteurs** » dans la zone de danger

OpenWebUI parcourt alors tous les fichiers en base, génère leurs embeddings avec `bge-m3` via Ollama, et les insère dans une nouvelle collection ChromaDB en dimension 1024.

5.3 Monitoring de la réindexation

Logs en temps réel :

```
journalctl -u open-webui -f --since "10 sec ago" | \
  grep -iE "embed|chunk|added|collection|error"
```

Compteur de progression :

```
DATA_DIR="/root/IA/open-webui.debian12/venv/lib/python3.12/\
site-packages/open_webui/data"

watch -n 30 "
echo 'Chunks indexes:'
sqlite3 \"$DATA_DIR/vector_db/chroma.sqlite3\" \
  'SELECT COUNT(*) FROM embeddings;'
echo ''
echo 'VRAM:'
nvidia-smi --query-gpu=memory.used,memory.free --format=csv
"
```

Durée observée : ~12 heures pour 1953 fichiers / 6396 chunks sur RTX 3060 partagée avec le bureau.

6. Bugs rencontrés et solutions

6.1 Bug : FETCH_URL_MAX_CONTENT_LENGTH not found

Symptôme : cliquer sur « Enregistrer » dans Paramètres → Documents provoque une exception :

```
AttributeError: Config key 'FETCH_URL_MAX_CONTENT_LENGTH' not found
  File ".../routers/retrieval.py", line 1177, in update_rag_config
```

Cause : la clé n'est pas déclarée dans `app.state.config` au démarrage d'OpenWebUI 0.8.12. C'est un bug de cette version (la clé existe dans le code de *lecture* mais pas dans la couche de *déclaration*).

Solution : patcher la base SQLite pour ajouter la clé manquante.

```
DATA_DIR="/root/IA/open-webui.debian12/venv/lib/python3.12/\
site-packages/open_webui/data"

# Backup avant patch
cp "$DATA_DIR/webui.db" \
   "$DATA_DIR/webui.db.backup-fix-fetch-$(date +%Y%m%d-%H%M%S)"

# Stop OpenWebUI
systemctl stop open-webui

# Patch SQL : ajouter la cle au bon niveau du JSON
sqlite3 "$DATA_DIR/webui.db" << 'EOF'
UPDATE config
SET data = json_set(data, '$.rag.fetch_url_max_content_length',
                    10485760)
WHERE id = 1;
EOF

# Verifier
sqlite3 "$DATA_DIR/webui.db" \
  "SELECT json_extract(data, '\$.rag.fetch_url_max_content_length')
   FROM config;"

# Restart
systemctl start open-webui
```

Note : ce patch ne corrige pas complètement le bug runtime. Les paramètres sont correctement sauvegardés dans la base malgré le crash sur la réponse HTTP. La fonction RAG fonctionne donc, seul le feedback UI est altéré.

6.2 Bug : Collection expecting embedding with dimension of 768

Voir section 5 (Migration de la base vectorielle).

6.3 Piège systemctl edit avec service défini en /etc/systemd/system/

Symptôme : `systemctl edit` affiche *« Found modifications outside of the staging area »* et refuse d'écrire.

Cause : `systemctl mask` tente de remplacer le service par un lien vers `/dev/null`, ce qui échoue si le service est défini dans `/etc/systemd/system/` (et pas `/lib/systemd/system/`).

Solution : écrire l'override directement plutôt que via `systemctl edit`.

```
mkdir -p /etc/systemd/system/open-webui.service.d/

cat > /etc/systemd/system/open-webui.service.d/override.conf << 'EOF'
[Service]
Environment="RAG_RERANKING_MODEL_AUTO_UPDATE=true"
Environment="SENTENCE_TRANSFORMERS_HOME=/root/.cache/huggingface"
Environment="CUDA_VISIBLE_DEVICES="
EOF

systemctl daemon-reload
systemctl restart open-webui
```

7. Annexe : durcissement Ollama (sécurité)

7.1 Contexte

Avant la migration RAG, deux instances Ollama étaient **exposées à Internet** sur le port 11434 sans authentification :

Serveur	Durée d'exposition	Modèles malveillants trouvés
`srv18`	13+ jours	6 (`attacker/leak_model_*`)
`mail` (ovh1)	50 jours	0
`backup` (ovh2)	1 jour	0

L'API Ollama étant non authentifiée par défaut, un acteur tiers pouvait `ollama pull` des modèles arbitraires depuis n'importe quel registry.

7.2 Durcissement appliqué (les 3 serveurs)

Override systemd : `/etc/systemd/system/ollama.service.d/override.conf`

```
[Service]
Environment="OLLAMA_HOST=127.0.0.1:11434"
Environment="OLLAMA_ORIGINS=http://localhost,http://127.0.0.1"
```

Application :

```
mkdir -p /etc/systemd/system/ollama.service.d/
# (Ecrire le fichier override.conf ci-dessus)
systemctl daemon-reload
systemctl restart ollama

# Verification
ss -tlnp | grep 11434
# Doit montrer : 127.0.0.1:11434 uniquement
```

7.3 Nettoyage des modèles malveillants (srv18)

```
ollama list | awk '/attacker|leak_model|205\.237\.106\.117/ \
  {print $1}' | while read m; do
    ollama rm "$m"
done
```

7.4 Désinstallation d'OpenWebUI sur mail et backup

OpenWebUI a été désinstallé sur `mail` (ovh1) et `backup` (ovh2) car non nécessaire. Ollama a été conservé en localhost-only sur les deux.

8. Annexe : commandes utiles

8.1 Inspection de la base ChromaDB

```
DATA_DIR="/root/IA/open-webui.debian12/venv/lib/python3.12/\
site-packages/open_webui/data"

# Collections et leurs dimensions
sqlite3 "$DATA_DIR/vector_db/chroma.sqlite3" \
  "SELECT id, name, dimension FROM collections;"
```

```
# Nombre total de chunks
sqlite3 "$DATA_DIR/vector_db/chroma.sqlite3" \
  "SELECT COUNT(*) FROM embeddings;"

# Chunks par collection
sqlite3 "$DATA_DIR/vector_db/chroma.sqlite3" "
SELECT c.name, c.dimension, COUNT(e.id) AS chunks
FROM collections c
LEFT JOIN segments s ON s.collection = c.id
LEFT JOIN embeddings e ON e.segment_id = s.id
GROUP BY c.id
ORDER BY chunks DESC;"
```

8.2 Inspection de la base OpenWebUI

```
# Fichiers indexes
sqlite3 "$DATA_DIR/webui.db" "SELECT COUNT(*) FROM file;"

# Collections de connaissance
sqlite3 "$DATA_DIR/webui.db" \
  "SELECT id, name, description FROM knowledge;"

# Configuration courante
sqlite3 "$DATA_DIR/webui.db" \
  "SELECT json_extract(data, '\$.rag') FROM config;" | \
  python3 -m json.tool
```

8.3 Monitoring Ollama et VRAM

```
# Modeles charges en VRAM
curl -s http://127.0.0.1:11434/api/ps | python3 -m json.tool

# Decharger explicitement un modele
curl -s http://127.0.0.1:11434/api/generate -d '{
  "model": "qwen-rag",
  "keep_alive": 0
}' > /dev/null

# VRAM en temps reel
watch -n 2 'nvidia-smi --query-gpu=memory.used,memory.free \
  --format=csv'
```

9. Restauration en cas de problème

Tous les backups sont dans `/root/backups/`. Pour restaurer une base avant la migration :

```
DATA_DIR="/root/IA/open-webui.debian12/venv/lib/python3.12/\
site-packages/open_webui/data"
BACKUP_DIR="/root/backups/open-webui-before-chroma-reset-\
20260525-153940"

# Arret
systemctl disable --now open-webui
pgrep -f "open-webui serve" | xargs -r kill -9

# Restauration
rm -rf "$DATA_DIR/vector_db"
cp -a "$BACKUP_DIR/vector_db" "$DATA_DIR/vector_db"
```

```
cp -a "$BACKUP_DIR/webui.db" "$DATA_DIR/webui.db"

# Redemarrage
systemctl enable open-webui
systemctl start open-webui
```

Fin du document — Migration RAG OpenWebUI srv18