ROYAUME DU MAROC المملكة المغربية

مكتب التكوين المهني وإنعاش الشغل

Office de la Formation Professionnelle et de la Promotion du Travail

SOCLE SOUVERAIN NIBRAS • EXPLOITATION DU SERVEUR ET DES APPLICATIONS PILOTES

GUIDE D'ADMINISTRATION DU VPS

Déploiement, mises à jour, sauvegardes, restauration, sécurité et dépannage des cinq applications hébergées : NIBRAS Admin, Correcto, Genero, Perform et Chat NIBRAS

Auteur & concepteur	Mohamed OUBEZZA (mohamed.oubezza@ofppt.ma)
Organisme de tutelle	Office de la Formation Professionnelle et de la Promotion du Travail (OFPPT)
Projet institutionnel	Projet NIBRAS - Intelligence artificielle souveraine
Serveur	`vmi2939194` - Ubuntu 22.04
Public	La personne qui déploie, met à jour, sauvegarde et dépanne les applications du VPS
Année de formation	2026/2027
Version du document	1.0 - 11 novembre 2026 (intègre le guide Vision / OCR de la Gateway en annexe)
Statut du document	Document technique interne - diffusion restreinte (noms d'hôtes, chemins et procédures du serveur)

SOMMAIRE

Avant de commencer 4

1. Règles à connaître avant tout 5

2. Carte du VPS 6

2.1 Applications couvertes 6
2.2 Autres services présents, non couverts par ce guide 6
2.3 Domaines 6

3. Vérification rapide de tout le serveur 8

4. Sauvegardes et restauration 9

4.1 Installer la sauvegarde quotidienne 9
4.2 Contenu d'une sauvegarde 9
4.3 Copier les sauvegardes hors du VPS 10
4.4 Restaurer 10

5. NIBRAS Admin 12

5.1 Mettre à jour 12
5.2 Modifier la configuration 12
5.3 Tester 12
5.4 Journaux 13
5.5 Dépannage 13

6. Correcto 14

6.1 Mettre à jour 14
6.2 Modifier la configuration 14
6.3 Tester 15
6.4 Journaux 15
6.5 Dépannage 15

7. Genero 16

7.1 Mettre à jour 16
7.2 Modifier la configuration 16
7.3 Tester 16
7.4 Dépannage 16

8. Perform 17

8.1 Mettre à jour 17
8.2 Modifier la configuration 17
8.3 Tester 17
8.4 Dépannage 18

9. Chat NIBRAS 19

9.1 Mettre à jour 19
9.2 Modifier la configuration 19
9.3 Tester 19
9.4 Dépannage 19

10. Nginx Proxy Manager et certificats 21

11. Entretien de Docker et du disque 22

12. Sécurité 23

13. Dépannage commun 24

Annexe A - Vision / OCR dans la Gateway NIBRAS 25

A.1 Format accepté 25
A.2 Paramètres de la requête 25
A.3 Configuration d'un modèle 26
A.4 Limites (fichier .env, redémarrer l'API) 27
A.5 Confidentialité 27
A.6 Modifications serveur 27
A.7 Test 28

Avant de commencer

Ce guide couvre cinq applications : **NIBRAS Admin**, **Correcto**, **Genero**, **Perform** et **Chat NIBRAS.** Chaque commande se copie telle quelle dans un terminal SSH ouvert sur le VPS ; les blocs marqués « PowerShell » se lancent depuis le PC Windows.

Besoin	Où regarder
Contrôle hebdomadaire ou après intervention	Chapitre 3 - Vérification rapide de tout le serveur
Sauvegarder, copier, restaurer	Chapitre 4 - Sauvegardes et restauration
Mettre à jour une application	Chapitres 5 à 9, rubrique « Mettre à jour » de chaque application
Un site ne répond plus	Chapitre 13 - Dépannage commun, puis la rubrique « Dépannage » de l'application
OCR et lecture d'images par l'IA	Annexe A - Vision / OCR dans la Gateway NIBRAS

Conventions

Les noms entre chevrons (`<service>`, `<conteneur>`, `<IP-du-VPS>`) sont à remplacer par la valeur réelle. `AAAAMMJJ-HHMM` désigne le dossier d'une sauvegarde datée.

Le manuel d'utilisation de NIBRAS Admin décrit l'interface ; ce guide décrit le serveur.

1. Règles à connaître avant tout

Trois règles qui protègent les données

Ne jamais utiliser `down -v`, `rm -v` ni `docker volume prune` ; sauvegarder avant chaque mise à jour ; ne jamais coller une clé ou un mot de passe dans un ticket, un chat ou une capture.

- **Commande Compose :** le VPS utilise `docker-compose` **v1.29** (avec un tiret). Toutes les commandes de ce guide l'utilisent. Elles se lancent **depuis le dossier de l'application**, car Compose y lit le fichier `docker-compose.yml` et le `.env`.
- **Ne jamais utiliser `down -v`, `rm -v` ni `docker volume prune`.** Ces commandes suppriment les volumes, donc les bases de données, les documents et les scans.
- **Sauvegarder avant chaque mise à jour** (chapitre 4).
- **Après une modification d'un `.env`**, recréer le conteneur (`up -d --force-recreate`) : `restart` ne relit pas les variables d'environnement.
- **Erreur `KeyError: 'ContainerConfig'`** pendant un `up` : c'est un défaut connu de `docker-compose` v1 avec Docker 29. Supprimer les conteneurs du service (les volumes sont conservés), puis relancer :

Terminal SSH (VPS)

```
sudo docker-compose rm -f -s <service>
sudo docker-compose up -d
```

- **Conteneur en boucle de redémarrage** (`Restarting`) : `exec` est refusé. Lire ses journaux avec `sudo docker logs --tail=80 <conteneur>` ; la cause est presque toujours dans les dernières lignes, souvent une variable de configuration invalide.
- **Commentaires dans les `.env`** : un commentaire doit être sur **sa propre ligne**. `VAR=valeur  # commentaire` peut être lu comme une seule valeur.
- **Secrets :** ne jamais coller une clé ou un mot de passe dans un ticket, un chat ou une capture. Pour une clé utilisée en test, la saisir sans l'afficher :

Terminal SSH (VPS)

```
read -r -s -p 'Clé NIBRAS : ' NIBRAS_KEY; echo
```

2. Carte du VPS

2.1 Applications couvertes

Application	Dossier	Conteneurs	Données persistantes	Accès local
NIBRAS Admin	`~/nibras-admin`	`nibras-admin_nibras-admin-frontend_1`, `nibras-admin_nibras-admin-api_1`, `nibras-admin_worker_1`, `nibras-admin_dispatcher_1`, `nibras-admin_embeddings_1`, `nibras-admin_redis_1`, `nibras_db` (+ `db-init` et `migrate`, ponctuels)	volumes `nibras-admin_postgres-data`, `nibras-admin_rag-data`, `nibras-admin_redis-data` ; `.env`	`nibras-admin-frontend:80` (réseau `web-network`)
Correcto	`~/correcto`	`correcto_correcto-web_1`, `correcto_correcto-api_1`	volume `correcto_correcto_scans` (SQLite + scans) ; `deploy/config/backend.env`, `deploy/vps.env`	`127.0.0.1:8088`
Genero	`~/genero`	`genero_web`, `genero_api`	base `genero_db` dans **`perform_db`** ; `backend/templates_docx`, `backend/generated_preparations`, `backend/.env`	web `127.0.0.1:8092`, API `127.0.0.1:8091`
Perform	`~/perform`	`perform_frontend`, `perform_backend`, `perform_db`	volume `perform_postgres_data` ; `backend/instance`, `.env`, `backend/.env`	`perform_frontend:80`
Chat NIBRAS	`~/chat`	`chat`	volume `open-chat-data` ; `.env`	`0.0.0.0:3000` (voir chapitre 12)

Dépendances entre applications

- **Genero dépend de Perform** : sa base est dans `perform_db`, joignable par le réseau `perform_perform_network`. Arrêter ou casser `perform_db` arrête aussi Genero.
- **Chat NIBRAS et Correcto appellent NIBRAS Admin** pour leurs modèles d'IA. Chat NIBRAS passe par `http://nibras-admin-frontend/v1` (réseau interne), Correcto par `https://admin.nibras.site/v1`. Si NIBRAS Admin s'arrête, les fonctions d'IA de ces applications s'arrêtent aussi.

2.2 Autres services présents, non couverts par ce guide

À laisser en place : `nginx-proxy-manager` (le proxy HTTPS de tous les sites, dossier `~/infrastructure/nginx`), site-home (~/www, site vitrine) et backup_api (~/backup_api). Les commandes de ce guide ne les modifient pas.

2.3 Domaines

Domaine	Destination dans Nginx Proxy Manager
`admin.nibras.site`	`nibras-admin-frontend:80`

Domaine	Destination dans Nginx Proxy Manager
correcto.nibras.site	`correcto-web:80` ou `127.0.0.1:8088`
chat.nibras.site	`chat:8080`
genero.nibras.site	`genero_web:80` ou `127.0.0.1:8092`
perform.nibras.site	`perform_frontend:80`

Pas de Cloudflare : les DNS pointent directement vers le VPS.

3. Vérification rapide de tout le serveur

À faire après chaque intervention, et une fois par semaine :

Terminal SSH (VPS)

```
# État de tous les conteneurs (attendu : Up, et healthy là où un healthcheck existe)
sudo docker ps -a --format 'table {{.Names}}\t{{.Status}}' | sort

# Mémoire et CPU par conteneur
sudo docker stats --no-stream --format 'table {{.Name}}\t{{.CPUPerc}}\t{{.MemUsage}}'

# Disque (alerter au-delà de 80 %)
df -h / ; sudo docker system df

# Sites publics (attendu : 200, ou 301/302 vers la page de connexion)
for u in https://admin.nibras.site/health https://correcto.nibras.site https://chat.nibras.site; do
  printf '%-40s ' "$u"; curl -s -o /dev/null -w '%{http_code} %{time_total}s\n' "$u"
done

# Accès locaux
curl -s -o /dev/null -w 'Correcto %{http_code}\n'   http://127.0.0.1:8088/
curl -s -o /dev/null -w 'Genero web %{http_code}\n' http://127.0.0.1:8092/
curl -s                                             http://127.0.0.1:8091/api/v1/health; echo
curl -s -o /dev/null -w 'Chat %{http_code}\n'       http://127.0.0.1:3000/health
```

Ajoutez à la boucle les domaines de Genero et de Perform une fois connus.

Comprendre un état « unhealthy » : l'application tourne, mais son contrôle de santé échoue.

Terminal SSH (VPS)

```
sudo docker inspect --format '{{json .State.Health}}' perform_backend | python3 -m json.tool | tail
-20
```

La sortie montre la commande de contrôle et ses derniers résultats.

4. Sauvegardes et restauration

4.1 Installer la sauvegarde quotidienne

Terminal SSH (VPS)

```
cd ~/nibras-admin && git pull --ff-only
chmod +x deploy/vps/sauvegarde-vps.sh
mkdir -p ~/sauvegardes && chmod 700 ~/sauvegardes

# Premier essai manuel (1 à 3 minutes)
sudo deploy/vps/sauvegarde-vps.sh
ls -lh ~/sauvegardes/*/
```

Le dernier message doit être `Terminé : ...`, sans ligne `ECHEC`. Planifiez ensuite le script tous les jours à 3 h 30, dans la crontab de **root** :

Terminal SSH (VPS)

```
sudo crontab -e
```

Ajoutez cette ligne, puis enregistrez :

```
30      3       *       *       *       /home/moubezza/nibras-admin/deploy/vps/sauvegarde-vps.sh       >>
/home/moubezza/sauvegardes/sauvegarde.log 2>&1
```

Le script conserve **14 jours** de sauvegardes. Correcto et Chat NIBRAS sont arrêtés quelques secondes pendant la copie de leur base.

Vérification les jours suivants :

Terminal SSH (VPS)

```
tail -20 ~/sauvegardes/sauvegarde.log
du -sh ~/sauvegardes/*
```

4.2 Contenu d'une sauvegarde

Fichier	Contenu
`nibras.dump`, `nibras-globals.sql`	Base PostgreSQL NIBRAS Admin et ses rôles
`nibras-rag.tar.gz`	Documents RAG importés
`perform-db-all.sql`	Toutes les bases de `perform_db` (Perform **et** Genero)
`perform_db-<base>.dump`	Une sauvegarde par base (`perform`, `genero_db`...)
`perform-files.tar.gz`, `genero-files.tar.gz`	Fichiers et `.env` de Perform ; modèles Word, préparations et `.env` de Genero
`correcto-data.tar.gz`	Base SQLite et scans de Correcto
`chatnibras-data.tar.gz`	Base, comptes, conversations et fichiers de Chat NIBRAS
`*.env`	Fichiers de configuration (**contiennent des secrets**)

4.3 Copier les sauvegardes hors du VPS

Une sauvegarde qui reste sur le VPS ne protège pas d'une panne du VPS. Depuis le PC Windows, dans PowerShell, une fois par semaine :

PowerShell (PC Windows)

```
scp -r moubezza@<IP-du-VPS>:/home/moubezza/sauvegardes/<dossier-du-jour> D:\Sauvegardes\VPS\
```

> **Copies confidentielles**
>
> Conservez ces copies dans un dossier privé : elles contiennent des données personnelles (copies d'étudiants, comptes) et des secrets.

4.4 Restaurer

> **Restaurer remplace les données actuelles**
>
> Faites d'abord une sauvegarde de l'état actuel, et testez si possible la restauration sur une copie.

Dans les commandes, `B=` désigne le dossier de sauvegarde choisi.

NIBRAS Admin

Terminal SSH (VPS)

```
B=~/sauvegardes/AAAAMMJJ-HHMM
cd ~/nibras-admin
sudo docker-compose stop nibras-admin-frontend nibras-admin-api worker dispatcher
sudo docker exec -i nibras_db pg_restore -U nibras_admin -d nibras --clean --if-exists --no-owner <
"$B/nibras.dump"
sudo docker run --rm -v nibras-admin_rag-data:/data -v "$B":/backup alpine \
  sh -c 'rm -rf /data/* && tar -xzf /backup/nibras-rag.tar.gz -C /data'
sudo docker-compose up -d
```

Perform ou Genero (une seule base)

Exemple pour Genero ; remplacez `genero_db` par `perform` pour Perform :

Terminal SSH (VPS)

```
B=~/sauvegardes/AAAAMMJJ-HHMM
sudo docker stop genero_api
sudo docker exec -i perform_db pg_restore -U postgres -d genero_db --clean --if-exists --no-owner <
"$B/perform_db-genero_db.dump"
sudo docker start genero_api
```

Fichiers de Genero : `tar -xzf "$B/genero-files.tar.gz" -C ~/genero/backend`.

Correcto

Terminal SSH (VPS)

```
B=~/sauvegardes/AAAAMMJJ-HHMM
sudo docker stop correcto_correcto-api_1
sudo docker run --rm -v correcto_correcto_scans:/data -v "$B":/backup alpine \
  sh -c 'rm -rf /data/* && tar -xzf /backup/correcto-data.tar.gz -C /data'
sudo docker start correcto_correcto-api_1
```

Chat NIBRAS

Terminal SSH (VPS)

```
B=~/sauvegardes/AAAAMMJJ-HHMM
sudo docker stop chatNibras
sudo docker run --rm -v chatNibras_chatNibras-data:/data -v "$B":/backup alpine \
  sh -c 'rm -rf /data/* && tar -xzf /backup/chatnibras-data.tar.gz -C /data'
sudo docker start chatNibras
```

5. NIBRAS Admin

Dossier : `~/nibras-admin`. **Configuration :** `.env`. **Docs :** guide de déploiement (`docs/deploiement-vps-docker.md`), vision / OCR (Annexe A), manuel d'utilisation de NIBRAS Admin.

5.1 Mettre à jour

À faire en dehors des heures d'utilisation (Chat NIBRAS et Correcto en dépendent) :

Terminal SSH (VPS)

```
cd ~/nibras-admin
sudo ~/nibras-admin/deploy/vps/sauvegarde-vps.sh      # sauvegarde d'abord
git pull --ff-only
sudo docker-compose build nibras-admin-api nibras-admin-frontend
sudo docker-compose up -d                             # lance aussi migrate (migrations)
sudo docker-compose ps -a
sudo docker-compose logs --tail=30 migrate
```

- `migrate` et `db-init` doivent finir en **Exited (0)** : c'est leur état normal.
- Tous les autres services doivent être **Up (healthy)**.
- La construction de l'image API télécharge le modèle d'embeddings et peut prendre plusieurs minutes.

5.2 Modifier la configuration

Terminal SSH (VPS)

```
cd ~/nibras-admin
nano .env
sudo docker-compose up -d --force-recreate nibras-admin-api worker dispatcher embeddings
```

Certains réglages se font dans l'interface et s'appliquent sans redémarrage : **Paramètres** (timeout LLM, modèle par défaut), **Fournisseurs**, **Modèles**, **Clés API**. Pour le timeout LLM, régler 180 à 300 s

5.3 Tester

Terminal SSH (VPS)

```
cd ~/nibras-admin
# Santé et base de données
curl -fsS https://admin.nibras.site/health; echo
sudo docker-compose exec nibras-admin-api python -m app.cli check-db --require-vector

# Gateway : modèles visibles par une clé
read -r -s -p 'Clé NIBRAS : ' NIBRAS_KEY; echo
curl -sS https://admin.nibras.site/v1/models -H "Authorization: Bearer $NIBRAS_KEY"; echo

# Génération texte (affiche la réponse ou l'erreur)
curl -sS https://admin.nibras.site/v1/chat/completions \
  -H "Authorization: Bearer $NIBRAS_KEY" -H "Content-Type: application/json" \
  -d '{"model":"nibras-tizimi","max_tokens":64,"messages":[{"role":"user","content":"Combien font 37
x 48 ? Réponds uniquement par le nombre."}]}' \
  |  python3  -c  "import  sys,json;  r=json.load(sys.stdin);  print(r.get('error')  or
(r['choices'][0]['message']['content'], r['usage']))"
```

Résultat attendu : `('1776', {... 'completion_tokens': 5 ...})`. Un `completion_tokens` de plus d'une centaine signifie que la réflexion (thinking) est active.

Test vision, avec une image `page.jpg` présente sur le VPS :

Terminal SSH (VPS)

```
IMG=$(base64 -w0 page.jpg)
curl -sS https://admin.nibras.site/v1/chat/completions \
  -H "Authorization: Bearer $NIBRAS_KEY" -H "Content-Type: application/json" -d @- <<EOF | head -c
600; echo
{"model":"nibras-tizimi-vision","max_tokens":2048,"response_format":{"type":"json_object"},
 "messages":[{"role":"user","content":[
   {"type":"text","text":"Transcris l'image. JSON seul : {\"blocks\":[{\"text\":\"...\"}]}"},
   {"type":"image_url","image_url":{"url":"$IMG"}}]}]}
EOF
unset NIBRAS_KEY IMG
```

5.4 Journaux

Terminal SSH (VPS)

```
cd ~/nibras-admin
sudo docker-compose logs --tail=100 nibras-admin-api
# Une ligne par génération : modèle, images, statut, durée, tokens (jamais le contenu)
sudo docker-compose logs nibras-admin-api | grep llm_generation | tail -20
# Erreurs du fournisseur
sudo docker-compose logs nibras-admin-api | grep llm_upstream_error | tail -20
# Indexation documentaire
sudo docker-compose logs --tail=50 worker dispatcher embeddings
```

L'écran **Requêtes** de l'interface donne le même détail, avec l'identifiant de requête.

5.5 Dépannage

Symptôme	Cause probable et action
`llm_timeout` (504) après le timeout configuré	Timeout LLM trop court, ou réflexion active : **Paramètres > Délai LLM** à 180–300 s ; vérifier le test « 37 × 48 ».
`llm_unavailable` (503)	Fournisseur indisponible : `grep llm_upstream_error` ;
`llm_auth_failed`	Clé du fournisseur refusée : corriger `LLM_PROVIDERS` dans `.env`, puis recréer l'API.
`model_vision_unsupported` (400)	Image envoyée à un modèle sans vision : cocher « Accepte les images » sur le modèle.
`request_too_large` (413)	Requête de plus de 32 Mo : envoyer moins d'images par requête.
`embeddings` redémarre en boucle	`sudo docker-compose logs --tail=50 embeddings` ; voir le guide de déploiement (`docs/deploiement-vps-docker.md`, « Si embeddings redémarre avec Permission denied »).
Indexation bloquée	`worker` et `dispatcher` doivent être `healthy` ; `sudo docker-compose restart worker dispatcher`.
`migrate` en `Exited (1)`	`sudo docker-compose logs migrate` ; ne pas relancer les autres services avant correction.

6. Correcto

Dossier : `~/correcto`. **Configuration :** `deploy/config/backend.env` (le fichier `backend/.env` n'est pas utilisé sur le VPS). **Docs :** `DEPLOIEMENT_VPS.md` et `OCR_DISTANT.md` dans le dépôt.

Correcto utilise un fichier compose spécifique. Définir ce raccourci dans chaque session :

Terminal SSH (VPS)

```
cd ~/correcto
DC="sudo docker-compose --env-file deploy/vps.env -f docker-compose.vps.yml"
```

Pour l'avoir dans toutes les sessions :

Terminal SSH (VPS)

```
echo 'alias dcc="sudo docker-compose --env-file deploy/vps.env -f docker-compose.vps.yml"' >> ~/.bashrc
```

Utilisez alors `dcc` à la place de `$DC`, depuis `~/correcto`.

> **Ne jamais lancer sudo docker-compose seul dans ~/correcto**
>
> Il lirait le `docker-compose.yml` de développement (MinIO, PostgreSQL…) et proposerait de recréer des services inutiles. Répondre **N** si cette question apparaît.

6.1 Mettre à jour

Terminal SSH (VPS)

```
cd ~/correcto && git pull --ff-only
DC="sudo docker-compose --env-file deploy/vps.env -f docker-compose.vps.yml"
$DC stop correcto-api
sudo docker run --rm -v correcto_correcto_scans:/data:ro -v "$PWD/backups":/backup alpine \
  tar -czf "/backup/correcto-data-$(date +%Y%m%d-%H%M).tar.gz" -C /data .
$DC up -d --build
$DC ps
$DC logs --tail=50 correcto-api
```

Puis **Ctrl+Maj+R** dans le navigateur pour charger la nouvelle interface.

En cas de `KeyError: 'ContainerConfig'` : `$DC rm -f correcto-api correcto-web`, puis `$DC up -d --build`.

6.2 Modifier la configuration

Terminal SSH (VPS)

```
cd ~/correcto
sudo nano deploy/config/backend.env
DC="sudo docker-compose --env-file deploy/vps.env -f docker-compose.vps.yml"
$DC up -d --force-recreate correcto-api
$DC ps        # doit rester « Up », pas « Restarting »
```

Configuration actuelle de l'IA, via NIBRAS Admin :

Fichier de configuration (.env)

```
OPENAI_API_BASE=https://admin.nibras.site/v1
OPENAI_MODEL=nibras-tizimi
AI_TEMPERATURE=none
AI_JSON_MODE=true
OCR_PROVIDER=remote
OCR_API_BASE=https://admin.nibras.site/v1
OCR_MODEL=nibras-tizimi-vision
OCR_REMOTE_TIMEOUT_SECONDS=300
OCR_REMOTE_MAX_TOKENS=4096
OCR_MAX_IMAGE_SIDE=2048
```

Limites à respecter :

- `OCR_REMOTE_TIMEOUT_SECONDS` vaut **300 au maximum**, sinon l'API refuse de démarrer.
- Pas de `AI_EXTRA_BODY` ni de `VISION_EXTRA_BODY` : la réflexion se règle dans NIBRAS Admin (paramètres du modèle ou réglage « Réflexion » de la clé).
- La ou les clés `nb_correcto_...` doivent autoriser `nibras-tizimi` et `nibras-tizimi-vision`.

6.3 Tester

Terminal SSH (VPS)

```
cd ~/correcto
DC="sudo docker-compose --env-file deploy/vps.env -f docker-compose.vps.yml"
$DC exec correcto-api python scripts/diagnose_ai.py          # notation
$DC exec correcto-api python scripts/diagnose_ai.py --ocr    # OCR distant
curl -sI http://127.0.0.1:8088/ | head -1
```

Toutes les lignes du diagnostic doivent afficher `[OK]`.

6.4 Journaux

Terminal SSH (VPS)

```
$DC logs --tail=100 correcto-api
$DC logs correcto-api | grep "OCR distant" | tail -15     # durée et tokens par page
```

Exemple normal : `page 1/2 (...) en 19.2 s (jetons entrée 3983, sortie 817)` et `assemblage (...) en 4.0 s`.

6.5 Dépannage

Symptôme	Cause probable et action
API en `Restarting`	`$DC logs --tail=60 correcto-api` : chercher `ValidationError` (valeur invalide dans `backend.env`).
« OCR distant indisponible... »	`grep "OCR distant en échec"` : **504 après ~120 s** = timeout NIBRAS (le monter à 300 s) ; 401/403 = clé ; 404 = modèle absent ou non autorisé ; 400 = modèle sans vision.
« Réponse OCR tronquée »	Augmenter `OCR_REMOTE_MAX_TOKENS`, ou vérifier la réflexion du modèle.
« Assemblage distant incomplet »	Le modèle a oublié trop de passages : relancer la lecture. Moins fréquent avec une température basse.
Ancienne interface après mise à jour	Ctrl+Maj+R.
Orphelins `xxx_correcto_correcto-api_1`	`$DC down --remove-orphans`, puis `$DC up -d`.

7. Genero

Dossier : `~/genero`. **Configuration :** `backend/.env` (`DATABASE_URL`, `LLM_API_BASE_URL`, `LLM_API_KEY`, `LLM_DEFAULT_MODEL`...). **Base :** `genero_db` dans `perform_db`.

7.1 Mettre à jour

Terminal SSH (VPS)

```
cd ~/genero
sudo ~/nibras-admin/deploy/vps/sauvegarde-vps.sh
git pull --ff-only
sudo docker-compose up -d --build
sudo docker-compose ps
sudo docker logs --tail=50 genero_api
```

Le réseau `perform_perform_network` doit exister : **Perform doit être démarré avant Genero.**

7.2 Modifier la configuration

Terminal SSH (VPS)

```
cd ~/genero
nano backend/.env
sudo docker-compose up -d --force-recreate api
```

7.3 Tester

Terminal SSH (VPS)

```
curl -s http://127.0.0.1:8091/api/v1/health; echo        # attendu : "status": "healthy"
curl -s -o /dev/null -w '%{http_code}\n' http://127.0.0.1:8092/
sudo docker exec perform_db psql -U postgres -d genero_db -c '\dt' | head -15
```

7.4 Dépannage

Symptôme	Cause probable et action
`genero_api` ne démarre pas : `network perform_perform_network not found`	Démarrer Perform d'abord : `cd ~/perform && sudo docker-compose up -d`.
Erreur de connexion à la base	`sudo docker ps | grep perform_db` ; vérifier `POSTGRES_*` et `DATABASE_URL`.
Génération IA en échec	Vérifier `LLM_API_BASE_URL`, la clé et le modèle ; s'ils pointent vers NIBRAS, voir **Requêtes** dans NIBRAS Admin.
Modèles Word perdus après mise à jour	Ils sont dans `backend/templates_docx` (dossier monté) : ne pas supprimer ce dossier ; restaurer depuis `genero-files.tar.gz`.

8. Perform

Dossier : `~/perform`. **Configuration :** `.env` (lu par Compose) et `backend/.env`. **Base :** `perform` dans `perform_db` (PostgreSQL 15), partagée avec Genero.

> **Point ouvert (inventaire du 11 novembre 2026)**
> `perform_backend` et `perform_frontend` sont marqués **unhealthy** depuis 3 jours. Lancer le diagnostic ci-dessous et transmettre le résultat au responsable technique.

Terminal SSH (VPS)
```
sudo docker inspect --format '{{json .State.Health}}' perform_backend  | python3 -m json.tool | tail -15
sudo docker inspect --format '{{json .State.Health}}' perform_frontend | python3 -m json.tool | tail -15
sudo docker logs --tail=60 perform_backend
```

Le backend expose `/api/health`. Si le site fonctionne normalement, le contrôle de santé est probablement mal configuré (par exemple `curl` absent de l'image, ou mauvaise URL) plutôt que l'application en panne.

8.1 Mettre à jour

Terminal SSH (VPS)
```
cd ~/perform
sudo ~/nibras-admin/deploy/vps/sauvegarde-vps.sh
git pull --ff-only
sudo docker-compose up -d --build backend frontend     # sans recréer perform_db
sudo docker exec perform_backend flask --app wsgi db upgrade   # migrations de schéma
sudo docker-compose ps
sudo docker logs --tail=50 perform_backend
```

- Ne pas recréer `perform_db` sans raison : Genero l'utilise aussi.
- Si `flask db upgrade` indique qu'il n'y a rien à faire, c'est normal.

Certaines évolutions ont demandé du SQL manuel, par exemple :

Terminal SSH (VPS)
```
sudo docker exec -it perform_db psql -U postgres -d perform -c "ALTER TYPE user_role ADD VALUE IF NOT EXISTS 'surveillant';"
```

8.2 Modifier la configuration

Terminal SSH (VPS)
```
cd ~/perform
nano .env
sudo docker-compose up -d --force-recreate backend
```

8.3 Tester

Terminal SSH (VPS)
```
sudo docker exec perform_backend python -c "import urllib.request;print(urllib.request.urlopen('http://127.0.0.1:5000/api/health',timeout=5).read())"
sudo docker exec perform_db psql -U postgres -d perform -c 'SELECT count(*) FROM pg_stat_activity;'
```

8.4 Dépannage

Symptôme	Cause probable et action
Erreur 502 sur le site	`sudo docker logs --tail=80 perform_backend` ; redémarrer : `sudo docker-compose restart backend`.
Requêtes lentes ou « timeout »	Gunicorn coupe à 120 s (4 workers) : regarder la requête dans les journaux.
Erreur d'enum ou de colonne manquante après mise à jour	Migration non appliquée : `flask --app wsgi db upgrade`, ou le script SQL fourni avec la mise à jour.
Base pleine ou lente	`sudo docker exec perform_db psql -U postgres -c "SELECT datname, pg_size_pretty(pg_database_size(datname)) FROM pg_database;"`

9. Chat NIBRAS

Dossier : `~/chatNibras` (dépôt `chatNibras`, branche `main`). **Configuration :** `.env` (`OPENAI_API_BASE_URL`, `OPENAI_API_KEY`, `CHAT_SECRET_KEY`, `NIBRAS_APP_NAME`...). **Données :** volume `chatNibras_chatNibras-data` (comptes, conversations, réglages).

Connexion à NIBRAS : `OPENAI_API_BASE_URL=http://nibras-admin-frontend/v1`, par le réseau interne `web-network`, avec une clé NIBRAS de scope `chat.completions`.

9.1 Mettre à jour

> **Construction longue et gourmande en mémoire**
> La compilation de l'interface demande plusieurs Go de RAM et 10 à 30 minutes. Faites-la quand le VPS est peu utilisé.

Terminal SSH (VPS)

```
cd ~/chatNibras
sudo ~/nibras-admin/deploy/vps/sauvegarde-vps.sh     # la base peut être migrée par la mise à jour
git pull --ff-only
sudo docker-compose build
sudo docker-compose up -d
sudo docker logs --tail=50 chatNibras
```

Si la construction échoue faute de mémoire (`Killed`, `exit code 137`), arrêter temporairement un service non essentiel, puis relancer `build` et redémarrer ce service.

9.2 Modifier la configuration

Terminal SSH (VPS)

```
cd ~/chatNibras
nano .env
sudo docker-compose up -d --force-recreate
```

Beaucoup de réglages (connexions, modèles, outils) sont aussi enregistrés dans l'interface **Administration** et priment alors sur le `.env`.

9.3 Tester

Terminal SSH (VPS)

```
curl -s http://127.0.0.1:3000/health; echo                          # attendu : {"status":true}
sudo docker exec chatNibras curl -s -o /dev/null -w '%{http_code}\n' http://nibras-admin-frontend/health
```

Le second test vérifie que Chat NIBRAS joint bien NIBRAS Admin (attendu : `200`).

9.4 Dépannage

Symptôme	Cause probable et action
Aucun modèle proposé	Clé NIBRAS invalide ou modèle non autorisé : tester la clé avec `curl /v1/models` (chapitre 5).

Symptôme	Cause probable et action
`body.tools: extra_forbidden`	Outils activés sur le modèle : les désactiver dans **Espace de travail > Modèles** (voir le guide de déploiement NIBRAS Admin).
Réponse coupée ou très lente	Voir **Requêtes** dans NIBRAS Admin (timeout, fournisseur).
Tout le monde est déconnecté après une mise à jour	`CHAT_SECRET_KEY` a changé : remettre l'ancienne valeur dans `.env`.

10. Nginx Proxy Manager et certificats

Interface : `http://<IP-du-VPS>:81`. **Dossier :** `~/infrastructure/nginx`.

Pour chaque **Proxy Host** :

- **Force SSL** et **HTTP/2** activés, avec un certificat Let's Encrypt (renouvelé automatiquement).
- **Websockets Support** activé pour Chat NIBRAS.
- Destination par **nom de conteneur** (réseau `web-network`), par exemple `nibras-admin-frontend:80`.

Le trafic interne entre conteneurs (Chat NIBRAS vers `http://nibras-admin-frontend/v1`) ne passe pas par Nginx Proxy Manager : Force SSL n'a aucun effet dessus.

Onglet **Advanced** de `admin.nibras.site` :

Nginx - onglet Advanced

```
client_max_body_size 110m;
client_body_buffer_size 1m;
proxy_request_buffering off;
proxy_read_timeout 615s;
proxy_send_timeout 615s;
proxy_buffering off;
proxy_cache off;
```

Pour `correcto.nibras.site`, prévoir `client_max_body_size 150m;` et des délais de 1000 s (imports de copies).

Terminal SSH (VPS)

```
sudo docker logs --tail=50 nginx-proxy-manager                # erreurs de certificat ou de proxy
sudo docker network inspect web-network --format '{{range .Containers}}{{.Name}} {{end}}'
```

La seconde commande liste les conteneurs joignables par le proxy. Un site en **502 Bad Gateway** signifie que le conteneur visé est arrêté ou n'est pas dans `web-network`.

11. Entretien de Docker et du disque

Terminal SSH (VPS)

```
sudo docker system df                          # place prise par images, conteneurs, volumes, cache
sudo docker image prune -f                     # images sans nom (anciennes versions) : sans risque
sudo docker builder prune -f --filter until=168h   # cache de construction de plus de 7 jours
```

- **Ne pas utiliser** `docker system prune -a --volumes` ni `docker volume prune` : ces commandes suppriment des données.
- Des conteneurs arrêtés depuis des mois (`hungry_shamir`, `magical_davinci`, `practical_proskuriakova`, `infallible_brahmagupta`, `blissful_mclean`) sont des restes de tentatives. Après vérification, ils peuvent être supprimés :

Terminal SSH (VPS)

```
sudo docker rm hungry_shamir magical_davinci practical_proskuriakova infallible_brahmagupta blissful_mclean
```

- Les volumes nommés seulement par un identifiant (`1c2396c7...`) sont anonymes. Ne les supprimez pas sans savoir à quel conteneur ils appartiennent :

Terminal SSH (VPS)

```
sudo docker ps -a --filter volume=<identifiant>
```

- Les journaux de NIBRAS Admin sont limités (3 × 10 Mo par service). Pour les autres conteneurs, surveiller `sudo du -sh /var/lib/docker/containers/*/*-json.log | sort -h | tail`.

Mises à jour du système, une fois par mois :

Terminal SSH (VPS)

```
sudo apt update && sudo apt upgrade -y
[ -f /var/run/reboot-required ] && echo "Redémarrage nécessaire"
```

Après un redémarrage du VPS, tous les conteneurs repartent seuls (`restart: always` ou `unless-stopped`). Vérifier avec le chapitre 3.

12. Sécurité

L'inventaire montre trois ports ouverts sur toutes les interfaces, qui contournent HTTPS :

Port	Service	Risque
`3000`	Chat NIBRAS	Accès direct en HTTP, sans chiffrement
`8080`	Service tiers (hors NIBRAS)	Idem
`81`	Interface de Nginx Proxy Manager	Page d'administration du proxy exposée

Vérifier si le pare-feu les bloque déjà :

Terminal SSH (VPS)
```
sudo ufw status verbose
```

S'il est inactif, l'activer en gardant SSH et HTTPS ouverts :

Terminal SSH (VPS)
```
sudo ufw allow OpenSSH
sudo ufw allow 80/tcp
sudo ufw allow 443/tcp
sudo ufw enable
```

> **Docker contourne ufw**
> Docker publie ses ports **en contournant ufw**. Pour Chat NIBRAS, la protection durable consiste à ne publier le port qu'en local.

Dans `~/chatNibras/.env` :

Fichier de configuration (.env)
```
CHAT_NIBRAS_PORT=127.0.0.1:3000
```

Puis `cd ~/chatNibras && sudo docker-compose up -d --force-recreate`. Le site reste accessible par `chat.nibras.site`, via Nginx Proxy Manager et le réseau `web-network`. Pour le port 81, y accéder par un tunnel SSH plutôt que publiquement :

PowerShell (PC Windows)
```
ssh -L 8181:127.0.0.1:81 moubezza@<IP-du-VPS>
```

Puis ouvrir `http://localhost:8181` sur le PC.

Autres règles

- Les fichiers `.env` doivent rester privés : `chmod 600` sur chacun.
- Renouveler toute clé collée par erreur dans un chat ou un ticket.

13. Dépannage commun

Symptôme	Commande ou action
Un site affiche 502	`sudo docker ps -a | grep <app>` : conteneur arrêté ou en boucle ? Puis `sudo docker logs --tail=80 <conteneur>`.
Un conteneur est en `Restarting`	`sudo docker logs --tail=80 <conteneur>` : la cause est dans les dernières lignes (variable invalide, base injoignable).
`KeyError: 'ContainerConfig'`	`sudo docker-compose rm -f -s <service>`, puis `up -d` (chapitre 1).
« Found orphan containers »	`sudo docker-compose down --remove-orphans` **avec le bon fichier compose**, puis `up -d`.
Compose demande de recréer un service inconnu (minio...)	Mauvais fichier compose : répondre **N** ; pour Correcto, utiliser `$DC`.
Disque plein	Chapitre 11 ; puis `du -sh ~/sauvegardes ~/*/backups`.
Mémoire saturée	`sudo docker stats --no-stream` ; redémarrer le conteneur le plus gourmand ; éviter les constructions d'images aux heures d'utilisation.
Fonctions IA en panne dans plusieurs applications	Commun à toutes : NIBRAS Admin ou le fournisseur. Voir `grep llm_upstream_error` (chapitre 5).
Certificat expiré	`sudo docker logs nginx-proxy-manager | grep -i certbot` ; renouveler depuis l'onglet **SSL Certificates**.

Avant de demander de l'aide, préparer :

- l'application et l'heure ;
- la commande lancée ;
- les 30 dernières lignes de journal ;
- l'identifiant de requête NIBRAS s'il existe.

> **À RETENIR**
> Ne jamais y joindre une clé ni un mot de passe.

Annexe A - Vision / OCR dans la Gateway NIBRAS

La Gateway `POST /v1/chat/completions` accepte le contenu multimodal au format OpenAI (texte + images) pour les modèles déclarés « vision ». Les clients texte existants ne changent pas.

A.1 Format accepté

Requête JSON

```
{
  "model": "nibras-tizimi-vision",
  "max_tokens": 8192,
  "response_format": {"type": "json_object"},
  "messages": [
    {"role": "system", "content": "Transcris fidèlement l'image... JSON seul"},
    {"role": "user", "content": [
      {"type": "text", "text": "Transcris."},
      {"type": "image_url", "image_url": {"url": "...", "detail": "auto"}}
    ]}
  ]
}
```

- `content` : chaîne, ou liste de parties `text` / `image_url` (tous rôles). Une liste texte seul est fusionnée en chaîne (comportement historique).
- `detail` facultatif : `auto`, `low`, `high`.
- Images : data URL base64 `image/jpeg`, `image/png` ou `image/webp`. Le base64 est vérifié, et le contenu doit correspondre au type annoncé.
- La liste `content` est transmise **telle quelle** au fournisseur : pas d'aplatissement, pas de réencodage.
- Les URL `http(s)` sont refusées par défaut (risque SSRF, fuite de données).

Refus (HTTP 422, `{"error": {"code": "validation_error", "message": ...}}`) :

Cas	Message
Image > limite	`Image trop volumineuse (6 Mo > 5 Mo).`
Type non supporté / contenu incohérent	`Format d'image non supporté : ...`
Base64 invalide	`Image invalide : base64 incorrect.`
URL http(s)	`URL d'image distante refusée : envoyez l'image en data URL base64 ...`
Trop d'images	`Trop d'images (11 > 10).`
Partie inconnue	`Type de contenu « input_audio » non supporté (admis : text, image_url).`

Une image envoyée à un modèle sans vision est refusée avec **400** `model_vision_unsupported` (« Le modèle X n'accepte pas les images. »), sans appel au fournisseur.

A.2 Paramètres de la requête

- Paramètres OpenAI acceptés : `temperature`, `top_p`, `max_tokens`, `max_completion_tokens`, `stop`, `n` (1), `seed`, `presence_penalty`, `frequency_penalty`, `response_format`, `user`, `stream`, `stream_options`, `reasoning_effort`.

- `response_format` : `{"type":"text"}` (équivaut à l'absence), `{"type":"json_object"}`, `{"type":"json_schema","json_schema":{"name":...,"schema":{...}}}`.
- `thinking` et `enable_thinking` envoyés par le client sont **ignorés** : c'est la configuration du modèle (et la politique « Réflexion » de la clé) qui décide.
- `max_tokens` au-delà du plafond du modèle est **borné** au plafond, sans refus. Il est aussi borné à la place restante dans la fenêtre de contexte.

A.3 Configuration d'un modèle

Deux options s'ajoutent dans **Catalogue IA > Modèles** (et dans l'API `/api/models`) :

Champ	Défaut	Rôle
`supports_vision`	`false`	Autorise les parties `image_url`.
`response_format_mode`	`forward`	`forward` : transmis tel quel. `strip` : retiré (fournisseur sans `response_format`). `force_json_object` : toute demande structurée (`json_object` ou `json_schema`) est envoyée en `{"type":"json_object"}`. Une requête sans `response_format` reste sans.

Créer nibras-tizimi-vision

1. Vérifier que le fournisseur local (connexion `.env`, type `openai`) existe dans **Fournisseurs**.
2. **Modèles > Ajouter** :
    1. Nom public : `nibras-tizimi-vision`
    2. Fournisseur : local
    3. Modèle servi
    4. Fenêtre de contexte : `262144` ; sortie maximale : `32768`
    5. Cocher **Accepte les images (vision / OCR)**
    6. Format de réponse : `Transmettre la demande du client`
    7. Paramètres par défaut : `{"thinking": {"type": "disabled"}}` (le thinking est désactivé quelle que soit la clé, puisque `thinking` venant du client est ignoré). Ajouter `"temperature": 0.2` seulement si le fournisseur l'accepte : certains déploiements refusent toute autre valeur que la leur.
3. Autoriser `nibras-tizimi-vision` dans les **modèles autorisés** de la clé API de Correcto.

Équivalent API :

Terminal SSH (VPS)

```
curl -sS https://admin.nibras.site/api/models -H "Authorization: Bearer $ADMIN_JWT" \
  -H "Content-Type: application/json" -d '{
  "name": "nibras-tizimi-vision", "display_name": "Tizimi vision",
  "provider_id": "<id du fournisseur>", "provider_model_name": "",
  "context_length": 262144, "max_output_tokens": 32768,
  "supports_vision": true, "response_format_mode": "forward",
  "request_defaults": {"thinking": {"type": "disabled"}}
}'
```

A.4 Limites (fichier .env, redémarrer l'API)

Variable	Défaut	Rôle
`LLM_MAX_REQUEST_MB`	`32`	Corps maximal de `/v1/chat/completions` et `/api/llm/test` (les autres routes restent à 1 Mo).
`LLM_IMAGE_MAX_MB`	`5`	Taille maximale d'une image décodée.
`LLM_MAX_IMAGES`	`10`	Nombre maximal d'images par requête.
`LLM_IMAGE_ALLOW_REMOTE_URLS`	`false`	Autorise les URL `http(s)` (déconseillé).
`LLM_IMAGE_TOKEN_ESTIMATE`	`16384`	Forfait de tokens si les dimensions d'une image sont illisibles.
`LLM_TIMEOUT_SECONDS`	`180`	Délai total d'une génération (au moins 180 s pour les démarrages à froid).

Le délai effectif est celui de **Paramètres > Délai LLM** (enregistré en base) s'il a été modifié dans l'interface : le régler à **180 s** au minimum.

Comptage avant envoi (quotas, fenêtre de contexte) : le base64 n'est jamais compté comme du texte. Chaque image compte pour une borne haute calculée à partir de ses dimensions (un token par carré de 28×28 px + 64, soit environ 3 900 tokens pour une page A4 de 2048 px). La facturation repose sur le champ `usage` renvoyé par le fournisseur, qui est retransmis au client.

A.5 Confidentialité

- Le base64, le texte des messages et la réponse du modèle ne sont jamais journalisés. Une ligne `llm_generation` donne le modèle, le nombre d'images, leur taille totale, le statut, la durée et les tokens.
- Les extraits d'erreurs 5xx du fournisseur sont nettoyés : suites base64 retirées, extrait supprimé s'il cite le texte de la requête.
- L'option « Conserver les prompts » (`store_prompts`) ne s'applique jamais aux requêtes avec images.
- Les images ne sont écrites ni sur disque ni en base. Voir aussi `proxy_request_buffering off` ci-dessous, qui évite les fichiers temporaires de Nginx.

A.6 Modifications serveur

Nginx du frontend (frontend/nginx.conf, déjà modifié dans le dépôt)

`location /v1/` passe de `client_max_body_size 1m` à `32m`, avec `proxy_send_timeout 615s` et `proxy_request_buffering off`. Reconstruire et redéployer l'image :

Terminal SSH (VPS)

```
cd ~/nibras-admin
git pull --ff-only
sudo docker-compose build nibras-admin-frontend nibras-admin-api
sudo docker-compose run --rm migrate          # migration 0013_models_vision
sudo docker-compose up -d nibras-admin-api nibras-admin-frontend
```

Nginx Proxy Manager, Proxy Host admin.nibras.site, onglet Advanced

La configuration documentée (`client_max_body_size 110m; proxy_read_timeout 615s; ...`) suffit déjà. Ajouter seulement la désactivation du tampon disque des requêtes :

Nginx - onglet Advanced

```
client_max_body_size 110m;
client_body_buffer_size 1m;
proxy_request_buffering off;
proxy_read_timeout 615s;
proxy_send_timeout 615s;
proxy_buffering off;
proxy_cache off;
```

Caddy (deploy/Caddyfile, installation Windows/Linux sans Docker)

Pas de limite de corps par défaut, et `response_header_timeout 750s` est déjà configuré : rien à changer.

Cloudflare

> **Sans objet aujourd'hui**
> Les DNS pointent directement vers le VPS, sans Cloudflare (§ 2.3). Ce paragraphe sert si le proxy Cloudflare est activé un jour.

Si `admin.nibras.site` est proxifié par Cloudflare (nuage orange), Cloudflare coupe toute réponse après **100 s** (erreur 524), et limite le corps à 100 Mo sur les offres Free et Pro. Un démarrage à froid peut dépasser 100 s.

Vérification :

Terminal SSH (VPS)

```
dig +short admin.nibras.site          # IP Cloudflare (104.x, 172.64-71.x) = proxifié
curl -sI https://admin.nibras.site/health | grep -i '^server: cloudflare'
```

Solutions, de la plus simple à la plus coûteuse :

1. **Sous-domaine API non proxifié** (recommandé) : enregistrement DNS `A api.nibras.site` vers l'IP du VPS en **DNS only** (nuage gris). Dans Nginx Proxy Manager, ajouter `api.nibras.site` aux Domain Names du Proxy Host (même destination, mêmes réglages Advanced) et demander le certificat Let's Encrypt. Côté Correcto : `OCR_API_BASE=https://api.nibras.site/v1`.
2. Désactiver le proxy Cloudflare sur `admin.nibras.site` (nuage gris).
3. Offre Enterprise : augmenter le délai de proxy (`proxy_read_timeout`, jusqu'à 6 000 s).

A.7 Test

Terminal SSH (VPS)

```
IMG=$(base64 -w0 page.jpg)
curl -sS https://admin.nibras.site/v1/chat/completions \
  -H "Authorization: Bearer $NIBRAS_KEY" -H "Content-Type: application/json" \
  -d @- <<EOF
{"model":"nibras-tizimi-vision","max_tokens":2048,"response_format":{"type":"json_object"},
 "messages":[{"role":"user","content":[
   {"type":"text","text":"Transcris l'image. JSON seul : {\"blocks\":[{\"text\":\"...\"}]}"},
   {"type":"image_url","image_url":{"url":"$IMG"}}]}]}
EOF
```

Réponse attendue : `choices[0].message.content` (chaîne JSON), `finish_reason: "stop"`, `usage`.

Contrôle des refus (modèle texte, doit renvoyer 400 sans appeler le fournisseur) :

Terminal SSH (VPS)

```
curl -sS -o /dev/null -w '%{http_code}\n' https://admin.nibras.site/v1/chat/completions \
  -H "Authorization: Bearer $NIBRAS_KEY" -H "Content-Type: application/json" \
  -d
tizimi\",\"messages\":[{\"role\":\"user\",\"content\":[{\"type\":\"image_url\",\"image_url\":{\"url\":\"data:image
```

Côté Correcto : `OCR_API_BASE=https://admin.nibras.site/v1` (ou `https://api.nibras.site/v1`), `OCR_MODEL=nibras-tizimi-vision`, puis `python scripts/diagnose_ai.py --ocr`.

Tests automatisés : `backend/tests/test_vision_gateway.py`.

NIBRAS - conçu pour l'excellence pédagogique et la valorisation du travail du personnel de l'OFPPT.