

Configuration, environnements et constantes

Vue d'ensemble

La configuration backend est principalement répartie entre les fichiers d'environnement et `uniksim-back-end/src/node.js/server/constantsVariable.js`. Ce fichier centralise de nombreuses valeurs utilisées par les serveurs, les intégrations et les règles métier.

Ne jamais recopier de secrets dans le wiki. Le fichier contient des clés, mots de passe et credentials qui doivent rester dans le code ou une configuration sécurisée et ne doivent pas être documentés en clair.

Fichiers d'environnement

- `dev.env` : environnement de développement.
- `dev2.env` : deuxième environnement de développement.
- `demo.env` : environnement demo.
- `live.env` : environnement live.

Les scripts de démarrage du backend sélectionnent explicitement l'un de ces fichiers via `node --env-file ... dist/main`.

constantsVariable.js

Le fichier `src/node.js/server/constantsVariable.js` sert de registre central pour plusieurs familles de constantes.

- `SERVER_INSTANCE` : live, dev, demo, dev2 et demoLive.
- `HOSTNAME`, `PORTS`, `USERNAME` : paramètres d'infrastructure.
- `SITE_URL_COMPLET` : URLs des instances Web.
- `ConstantsInUse` : valeurs effectivement utilisées par le processus courant.
- `ADMIN_SERVER_URL` / `AI_SERVER_URL` : services locaux auxiliaires.
- `ROLES_STRING` / `ROLES_NUMBER` : mapping des rôles.
- `SIMULATION` : identifiants des jeux.
- `AI_CONFIG` : provider, modèles, limites, timeouts et paramètres RAG.
- `SUPPORT_CONFIG` : uploads, limites de messages, événements et comportement IA du support.
- `NOTIFICATION_CONFIG` : rétention Redis et événements de notification.
- `FIREBASE_CONFIG` : paramètres techniques Firebase/FCM.

ConstantsInUse

`ConstantsInUse` agit comme configuration runtime commune. Elle contient notamment la base ciblée, le host SQL, le port SQL, le serveur courant, l'URL du site, la version WebGL locale et les ports des serveurs auxiliaires.

Lorsqu'un bug apparaît uniquement en dev/demo/live, ce bloc est un des premiers endroits à vérifier avec le fichier `.env` correspondant.

Frontend

Côté Angular, `uniksimFrontEnd/src/environments/environment.ts` définit notamment les URLs de serveur utilisables et le backend attendu.

Points d'attention

- Ne pas ajouter une nouvelle constante "temporaire" sans vérifier si elle existe déjà dans une famille logique.
- Une modification de port ou d'URL peut affecter plusieurs processus : NestJS, Admin Server, IA, Angular ou Redis.
- Les rôles numériques doivent rester cohérents avec les données en base.
- Les valeurs sensibles ne doivent jamais être copiées dans la documentation.

Configuration IA et temps réel actuelle

Au 30 septembre 2026 : `GOOGLE_CLOUD_PROJECT` est requis pour Vertex AI; `GOOGLE_CLOUD_LOCATION` vaut `global` par défaut et cette location est imposée par la politique tarifaire actuelle. L'authentification utilise les credentials Google par défaut de l'environnement. Chroma Cloud utilise les variables `CHROMA_API_KEY`, `CHROMA_TENANT` et `CHROMA_DATABASE`; la collection est `game`. Ne pas recopier leurs valeurs ici.

`AI_MODELS` configure `gemini-3.5-flash-lite` et `gemini-embedding-001`. `AI_PRICE_VERSIONS` référence `vertex-global-standard-2026-09-29`; `THINKING_LEVEL` vaut `LOW`. Le provider de génération et celui d'embedding restent indépendants. Le serveur IA écoute sur loopback; son token interne protège les opérations query/summary.

`NOTIFICATION_CONFIG` fixe le suffixe `_notifications`, la taille maximale de l'inbox (100), sa rétention (10 519 200 secondes), les types serveur réservés et la taille du payload client (2 048 caractères). Les constantes Angular sont séparées dans `realtime.constants.ts` (transport), `account.constants.ts` (activité du compte) et `support.constants.ts` (support).

Revision #2

Created 2026-09-18 17:42:16 UTC by Mathis

Updated 2026-09-30 14:27:59 UTC by Mathis