

Base de données et persistance

Vue d'ensemble

Le backend utilise principalement **MySQL via TypeORM** pour les données relationnelles, et **Redis** pour plusieurs données temporaires ou temps réel.

Architecture de persistance

flowchart TD

API[NestJS Controllers] --> SERVICES[Services métier]

SERVICES --> ORM[TypeORM]

ORM --> MYSQL[(MySQL)]

SERVICES --> REDIS[RedisService]

REDIS --> CACHE[(Redis)]

SERVICES --> BC[BrainCloud]

MYSQL --> E1[Users / rôles]

MYSQL --> E2[Produits / codes]

MYSQL --> E3[Support]

MYSQL --> E4[FCM installations]

CACHE --> R1[Présence]

CACHE --> R2[Sessions]

CACHE --> R3[Notifications]

CACHE --> R4[Rankings / données temps réel]

Configuration TypeORM

Le point central est `uniksim-back-end/src/database/providers.ts`.

La connexion est créée avec `TypeOrmModule.forRoot(...)` et utilise :

- `ConstantsInUse.HOSTNAME`
- `ConstantsInUse.PORTS`
- `ConstantsInUse.USERNAME`
- `ConstantsInUse.PASSWORD_DATABASE`
- `ConstantsInUse.PROVIDERS`

Le backend est configuré avec `synchronize: false`. Les changements de schéma ne doivent donc pas être supposés automatiques.

Entités principales

Les entités TypeORM sont dans `uniksim-back-end/src/entities/`. Parmi les principales :

- `User`, `Role`, `RoleUser`
- `Product`, `ProductsCode`, `ProductsResources`, `Resources`
- `Period`, `Constant`
- `SupportConversation`, `SupportTicket`, `SupportMessage`
- `UserFcmToken`

Redis

Redis n'est pas seulement un cache. Il est utilisé pour plusieurs fonctions runtime importantes, notamment :

- présence et heartbeats;
- sessions de jeu;
- notifications persistées temporairement;
- rankings et certaines données de simulation;
- communication de certains services historiques.

Choisir le bon stockage

flowchart LR

CHANGE[Besoin métier] --> ENTITY{Donnée relationnelle ?}

ENTITY -- Oui --> MODEL[Entity TypeORM]

MODEL --> ORM[Repository / Service]

ORM --> SCHEMA[Vérifier schéma SQL / migration]

ENTITY -- Non --> RUNTIME{Temporaire ou temps réel ?}

RUNTIME -- Oui --> REDIS[RedisService]

RUNTIME -- Non --> OTHER[Vérifier BrainCloud / stockage existant]

Points d'attention

- Ne pas activer `synchronize: true` sur un environnement partagé pour contourner un changement de schéma.
- Vérifier les relations d'entités avant de supprimer ou renommer une colonne.
- Redis contient des données applicatives importantes : un flush peut affecter sessions, présence ou notifications.
- Avant d'ajouter un nouveau stockage, vérifier si le système utilise déjà MySQL, Redis ou BrainCloud pour ce type de donnée.

Évolution de la persistance des notifications

Au 30 septembre 2026, les entités enregistrées dans `database.providers.ts` comprennent `UserFcmToken`, sans modèle de subscription par produit. Une installation appartient au compte; les push ciblent toutes ses installations. L'inbox Redis `user_<id>_notifications` est bornée à 100 entrées; son TTL de 10 519 200 secondes est renouvelé à chaque insertion. `NotificationService` possède maintenant la logique d'inbox; les anciennes fonctions de notification dans `server.js` ont été retirées. Voir [Notifications et Push](#).

Revision #2

Created 2026-09-18 17:42:33 UTC by Mathis

Updated 2026-09-30 14:23:46 UTC by Mathis