

Services externes et dépendances

Vue d'ensemble

Plusieurs fonctions du Web dépendent de services externes ou de couches historiques. Cette page sert de carte pour comprendre **quelle dépendance intervient où** et quels fichiers vérifier lorsqu'une intégration tombe en panne.

Carte des intégrations

flowchart LR

NEST[NestJS principal] --> MYSQL[(MySQL)]

NEST --> REDIS[(Redis)]

NEST --> BC[BrainCloud]

NEST --> FCM[Firebase / FCM]

NEST --> MAIL[Mail]

NEST --> AI[Serveur IA :8106]

ADMIN[Admin Server :8105] --> BC

ADMIN --> REDIS

AI --> CHROMA[(Chroma)]

AI --> PROVIDER[Provider IA]

UNITY[Unity WebGL] --> NEST

ANG[Angular] --> NEST

ANG <-->|Socket.IO| NEST

BrainCloud

BrainCloud est encore utilisé par plusieurs fonctions historiques liées aux simulations et au jeu.

Fichiers importants :

- `src/node.js/server/backend_braincloud.js`
- `src/node.js/server/server.js`
- `src/node.js/server/admin_server.js`
- `src/service/game.service.ts`

Lorsqu'un bug concerne une simulation, une copie de simulation, certains documents de jeu ou des opérations historiques admin, vérifier si le flux passe par BrainCloud avant de modifier MySQL.

Redis

Redis est utilisé pour les données runtime : présence, sessions, rankings, notifications et autres informations temporaires.

Fichier principal : `src/service/redis.service.ts`. Certaines couches historiques l'utilisent également depuis `src/node.js/server/`.

Firestore / FCM

Firestore Cloud Messaging assure les push notifications vers les appareils enregistrés.

Fichiers principaux :

- `src/service/firestore.service.ts`
- `src/service/notification.service.ts`
- `UserFcmToken` dans les entités TypeORM.

Voir **Notifications et Push** pour le flux complet.

IA / Chroma

Le serveur IA est séparé logiquement sous `src/node.js/server/ai/` et utilise un provider de génération, un provider d'embeddings et Chroma pour la recherche vectorielle.

Fichiers importants :

- `src/service/ai.service.ts`
- `src/node.js/server/ai/server.js`
- `src/node.js/server/ai/provider/`
- `src/node.js/server/ai/factory/`
- `src/node.js/server/ai/guard/`
- `src/embedding/`

Voir le chapitre **Support et IA** pour le pipeline RAG et l'ajout de providers.

Mail

Le backend possède une couche mail utilisée notamment pour le reset de mot de passe et les courriels applicatifs.

- `src/service/mail.service.ts`
- `src/mail/`
- `MailController` dans `app.controller.ts`

Comment diagnostiquer une intégration

flowchart TD

```
BUG[Erreur d'intégration] --> LOCAL{Le backend reçoit la requête ?}
LOCAL -- Non --> FE[Frontend / réseau / auth]
LOCAL -- Oui --> SERVICE[Identifier le service backend]
SERVICE --> EXT{Appel externe ?}
EXT -- Non --> INTERNAL[Logique interne / DB]
EXT -- Oui --> WHICH{Quelle dépendance ?}
WHICH --> BC[BrainCloud]
WHICH --> R[Redis]
WHICH --> F[Firebase]
WHICH --> AI[IA / Chroma]
WHICH --> M[Mail]
BC --> LOG[Logs + config + credentials]
R --> LOG
```

F --> LOG

AI --> LOG

M --> LOG

Points d'attention

- Ne jamais documenter de clé API, mot de passe ou credential en clair.
- Vérifier la configuration d'environnement avant de conclure que le service externe est en panne.
- Plusieurs intégrations possèdent une couche historique et une couche NestJS moderne; identifier le chemin réellement utilisé avant de modifier le code.
- Lorsqu'une dépendance externe échoue, distinguer l'échec réseau/provider d'un problème de mapping ou de logique interne.

Contrats d'intégration actuels — septembre 2026

Gemini utilise `@google/genai` en mode Vertex AI via `provider/gemini/vertex_client.js`. La configuration utilise `GOOGLE_CLOUD_PROJECT`, `GOOGLE_CLOUD_LOCATION=global` et les credentials Google par défaut de l'environnement. Chroma utilise `CloudClient` avec `CHROMA_API_KEY`, `CHROMA_TENANT` et `CHROMA_DATABASE`; la collection configurée est `game`.

Firebase envoie les notifications à toutes les installations du compte stockées dans `UserFcmToken`. La validation des membres d'une simulation pour les notifications du jeu relève de `GameNotificationService` et de BrainCloud; les installations ne sont plus sélectionnées par subscription produit.

Socket.IO sépare `/account` (présence et notifications) et `/support` (conversations et tickets). L'initialisation IA en arrière-plan permet au serveur principal de démarrer même si un provider ou Chroma est temporairement indisponible.

Revision #2

Created 2026-09-18 17:46:49 UTC by Mathis

Updated 2026-09-30 14:23:52 UTC by Mathis