

Unity WebGL et communication avec le Web

Vue d'ensemble

Le jeu Unity WebGL est intégré au site Angular et communique avec le backend NestJS. Cette page présente les responsabilités de chaque couche et les principaux points à vérifier lorsqu'une modification touche le jeu et le Web.

Architecture générale

flowchart LR

USER[Utilisateur] --> ANG[Angular / PageWebgl]

ANG --> BUILD[Build Unity WebGL]

BUILD --> UNITY[Runtime Unity]

UNITY -->|HTTP| GAME[GameController]

GAME --> GS[GameService]

GS --> BC[BrainCloud]

GS --> REDIS[(Redis)]

GS --> STORAGE[storage / images / logs]

ANG -->|Auth / session / navigation| NEST[NestJS]

NEST --> ANG

Côté Angular

`PageWebglComponent` est l'écran qui héberge le jeu pour plusieurs routes, notamment `/game`, `/demo`, `/alpha`, `/alpha2` et `/demoLive`.

Le build Unity est livré dans les assets frontend. Le frontend Angular gère le contexte Web autour du jeu : navigation, authentification, chargement et communication avec les autres fonctions du site.

Côté backend

`GameController` dans `uniksim-back-end/src/app.controller.ts` est protégé par `AuthGuardGame`. Il délègue la logique à `GameService`.

Les endpoints couvrent notamment :

- version du build;
- informations de code produit;
- textes et traductions;
- écoles;
- token Web;
- heartbeat et présence en jeu;
- classements;
- images utilisateur;
- journalisation de crash.

Flux heartbeat

```
sequenceDiagram
    participant U as Unity WebGL
    participant G as GameController
    participant S as GameService / server.js
    participant R as Redis

    U->>G: heartBeat(simUid, userId, token)
    G->>S: heartBeat(...)
    S->>R: Vérifie session / présence
    R-->>S: État session
    S->>R: Met à jour présence + expiration
    S-->>G: Version / heartbeats / sessionTime
    G-->>U: Réponse
```

Build et fichiers statiques

Les assets Unity font partie du frontend Angular. Le build Angular de production est ensuite généré dans `uniksim-back-end/src/front`, puis servi par NestJS avec `ServeStaticModule`.

Voir **Build et déploiement Web** pour l'ordre complet de build.

Quand modifier plusieurs projets

Une modification du contrat Unity/Web nécessite souvent de vérifier plusieurs couches :

flowchart LR

CHANGE[Changement Unity / Web] --> UNITY[Code Unity]

CHANGE --> ANG[Angular / PageWebgl]

CHANGE --> API[GameController]

API --> SERVICE[GameService]

SERVICE --> DATA[BrainCloud / Redis / stockage]

UNITY --> TEST[Test WebGL]

ANG --> TEST

API --> TEST

Fichiers à connaître

- `uniksimFrontEnd/src/app/Pages/PageWebgl/`
- `uniksimFrontEnd/src/assets/unity/`
- `uniksim-back-end/src/app.controller.ts` — `GameController`
- `uniksim-back-end/src/service/game.service.ts`
- `uniksim-back-end/src/guards/authGame.guard.ts`
- `uniksim-back-end/src/node.js/server/server.js`
- `uniksim-back-end/src/node.js/server/backend_braincloud.js`

Points d'attention

- Ne pas modifier seulement le frontend si le format attendu par Unity ou l'API change.

- Vérifier le guard et le contexte d'instance (live/dev/demo) pour les endpoints du jeu.
- Les heartbeats ont des effets sur Redis et la présence; les traiter comme un flux runtime important.
- Après un nouveau build Unity, vérifier que les nouveaux fichiers WebGL sont bien ceux inclus dans les assets/build déployés.

Présence du compte et notifications après refactoring

`PageWebglComponent` transmet le jeu actif à `AccountSocketService.setGamePresence(productCodeId, gameId)` et l'efface à la sortie. Cette présence sur `/account` complète le heartbeat/session HTTP du jeu : les deux chemins ont des responsabilités distinctes. La reconnexion du compte restaure le jeu actif.

Le jeu envoie les notifications via `POST /saveNotification`; `userId` désigne le profil jeu destinataire et `extraData.simUid` la simulation. Ce contrat est traité par `NotificationController` avec `AuthGuardUser`, puis `GameNotificationService`, et non par `GameController`. Le serveur vérifie expéditeur et destinataire dans le document BrainCloud puis utilise l'inbox et le push du compte Web. Voir [Notifications et Push](#).

Revision #3

Created 2026-09-18 17:31:05 UTC by Mathis

Updated 2026-09-30 14:28:09 UTC by Mathis