

Administration : API NestJS et Admin Server Express

Vue d'ensemble

L'administration backend d'Uniksim repose sur **deux couches distinctes** qu'il ne faut pas confondre :

1. **les routes d'administration de l'API NestJS principale**, protégées par des guards et reliées aux services backend;
2. **un serveur Express d'administration séparé** dans `uniksim-backend/src/node.js/server/admin_server.js`, qui écoute localement sur le port `8105` et gère surtout des opérations historiques/BrainCloud et certains flux SSE.

Architecture backend d'administration

flowchart LR

```
CLIENT[Clients Web / outils internes] --> NEST[NestJS principale]
```

```
CLIENT --> EXPRESS[Admin Server Express :8105]
```

```
NEST --> AC[AdminController]
```

```
NEST --> AUC[AdminUserController]
```

```
NEST --> APC[AdminPresenceController]
```

```
NEST --> SAC[SupportAdminController]
```

```
AC --> AS[AdminService]
```

```
AUC --> AUS[AdminUserService]
```

```
APC --> PS[PresenceService]
```

```
AS --> DB[(MySQL / TypeORM)]
AUS --> DB
PS --> REDIS[(Redis)]

EXPRESS --> BC[BrainCloud]
EXPRESS --> REDIS
EXPRESS --> SSE[SSE / copie de simulations]
```

1. Administration dans l'API NestJS

Les contrôleurs d'administration sont regroupés dans `uniksim-back-end/src/app.controller.ts` et enregistrés dans `AppModule`.

- `AdminController` sous `/admin` : récupération et gestion d'informations utilisateurs.
- `AdminPresenceController` sous `/admin/presence` : présence en ligne, état de jeu et télémétrie.
- `AdminController` : autres opérations administratives.
- `SupportAdminController` : fonctions administratives du support.

La logique métier correspondante se trouve notamment dans `src/service/admin.service.ts`, `src/service/adminUser.service.ts` et `src/service/presence.service.ts`.

2. Admin Server Express séparé

Le fichier `src/node.js/server/admin_server.js` démarre un serveur Express distinct sur `ConstantsInUse.ADMIN_SERVER_PORT`, actuellement `8105`. Son URL locale de référence est définie par `ADMIN_SERVER_URL` dans `constantsVariable.js`.

Ce serveur est utilisé notamment pour :

- `POST /admin/adminCommand` : commandes d'administration BrainCloud telles que lecture de documents, récupération de simulations, copie de simulation et suppression d'un utilisateur d'une simulation;
- `POST /redisBulkCommands` : commandes Redis groupées;
- `GET /getActiveSimulations/...` : récupération des simulations actives;
- `GET /copySimNotification` : progression d'une copie de simulation via SSE;
- surveillance de certains jobs BrainCloud et relance lorsqu'ils disparaissent.

Flux d'une commande Admin Server

```
sequenceDiagram
    participant C as Client / outil admin
    participant E as Admin Server Express
    participant B as BrainCloud
    participant R as Redis
    participant S as Flux SSE

    C->>E: Requête admin
    alt Commande BrainCloud
        E->>B: Exécute la commande
        B-->>E: Résultat
    else Commande Redis
        E->>R: redisBulkCommands(...)
        R-->>E: Résultat
    else Copie de simulation
        E->>B: Copie données / simulation
        E-->>S: Progression
        S-->>C: Événements SSE
    end
    end
    E-->>C: Réponse
```

Sécurité backend

Les routes NestJS utilisent notamment `AuthGuardAdmin` et `AuthGuardAdminUser`. Toute modification d'une route d'administration doit vérifier le guard attendu et le rôle autorisé.

Fichiers principaux

- `uniksim-back-end/src/app.controller.ts`
- `uniksim-back-end/src/app.module.ts`
- `uniksim-back-end/src/node.js/server/admin_server.js`
- `uniksim-back-end/src/node.js/server/constantsVariable.js`

- `uniksim-back-end/src/service/admin.service.ts`
 - `uniksim-back-end/src/service/adminUser.service.ts`
 - `uniksim-back-end/src/service/presence.service.ts`
-

Revision #3

Created 2026-09-18 17:30:07 UTC by Mathis

Updated 2026-09-18 17:40:43 UTC by Mathis