

Simulations, activités, équipes et utilisateurs

Hiérarchie métier commune

```
flowchart TD
  USER[UserDetails] --> SIM[Simulation]
  SIM --> ACT[Activities]
  SIM --> ST[Students / Team stubs]
  ACT --> TEAM[Teams]
  TEAM --> MEMBER[TeamMember + Roles]
  ACT --> GS[ActDataGameSpec]
  TEAM --> DEC[Decisions / Modules]
```

Simulation

La simulation contient notamment teacher, étudiants, équipes, ordre des activités, activité courante, mode async, permissions, ponderations et custom evaluations. Les commandes communes couvrent création, lecture, modification, activation et récupération des stubs.

Activity

Une activité appartient à une simulation et possède son propre document `activity`, plus un document `actDataGameSpec`. Le contenu game-specific est volontairement séparé du socle commun.

Team

Le document `teams` contient membres, rôles, activités/décisions, cash et métadonnées runtime. By Module ajoute des documents `teamsModules`; By Year garde davantage de décisions annuelles dans l'état équipe/activité.

Join / create / remove

Les commandes communes incluent `join_sim`, `join_team`, `create_team`, `remove_user_from_sim`, `remove_user_from_team` et `change_team_name`. Ces opérations mettent à jour plusieurs documents et émettent des messages RTT.

Rôles

`set_user_role`, `set_user_roles` et `unset_user_role` modifient les rôles d'équipe. Le runtime Unity propage ces changements via `TeamBaseController`.

Heartbeat / présence online

Les controllers simulation/activity gardent les derniers heartbeats et considèrent un utilisateur online selon l'âge du dernier heartbeat. Les listeners RTT mettent à jour les statuts visibles.

Écoles et access codes

Le backend commun gère aussi école, teacher lookup et access codes. `save_access_code` / `get_access_code` persistent le code dans `userDetails`.

Point de debug

Lorsqu'un utilisateur “disparaît” ou reste dans une mauvaise équipe, comparer au minimum `userDetails`, `simulation` et `teams`. Le problème peut être une incohérence multi-document et non un simple cache Unity.

Ajouter un champ de simulation/activity/team de A à Z

Un nouveau champ traversera souvent plusieurs couches. Le piège classique est de l'ajouter au document serveur mais d'oublier le modèle Unity, ou l'inverse.

1. Choisir le bon document

- `simulation` : configuration globale qui survit aux activités;
- `activity` : état commun d'une activité;
- `actDataGameSpec` : données spécifiques au jeu pour une activité;
- `teams` : état de l'équipe;
- `teamsModules` : données détaillées By Module;
- `userDetails` : préférences/état du compte utilisateur.

2. Ajouter le champ au modèle Unity

Ajouter la constante de clé et le champ dans la classe Data correspondante, puis le lire dans `Deserialize()`. Tolérer l'absence du champ pour les vieux documents lorsque nécessaire.

3. Ajouter l'écriture serveur

Utiliser une commande dédiée ou une commande générique `update_*_fields` si le champ s'y prête. Sauvegarder via `utils_documents`.

4. Ajouter la migration si le champ est structurel

Si le document peut exister longtemps, initialiser le nouveau champ dans `version_simulation.js`, `version_activity.js`, `version_teams.js`, etc.

5. Synchroniser les autres clients

Ajouter un RTT `*_updated_fields` ou spécialisé. Le controller doit filtrer sur l'UID du document avant d'appliquer la mutation.

6. Tester tous les rôles

Pour simulation/team, tester professeur + étudiant, join tardif et relog. Un client qui était offline doit récupérer la valeur correcte depuis le document, pas dépendre d'avoir reçu le RTT historique.

Opérations multi-documents

Join team, remove user, create team et suppression d'utilisateur touchent plusieurs documents. Pour toute nouvelle opération de ce type, écrire explicitement la liste des documents modifiés avant de coder. Un succès partiel peut laisser `userDetails`, `simulation` et `teams` incohérents.

Debug d'une incohérence utilisateur/équipe

1. Lire `userDetails.teamUid/simUid/actUid`.
2. Lire `simulation.students` / `teams`.
3. Lire le document `teams` et ses members.
4. Comparer les trois UUIDs.
5. Vérifier le dernier RTT join/remove.
6. Recharger complètement la simulation : si le bug disparaît, suspecter le controller/cache plutôt que la persistance.

Revision #2

Created 2026-09-18 19:23:39 UTC by Mathis

Updated 2026-09-18 20:10:18 UTC by Mathis