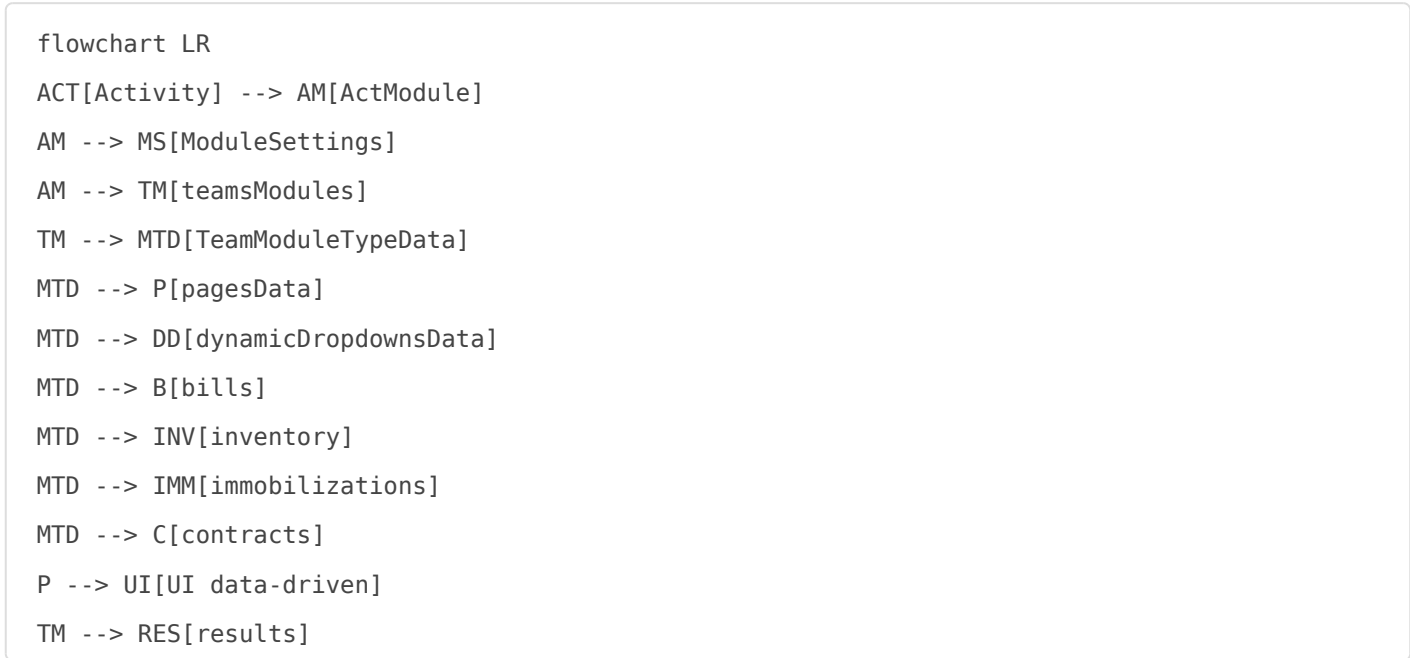


Architecture By Module / Accounting

Rôle

By Module est le framework où une activité est composée de modules configurables. Accounting (`acc1` et `acc2`) est l'implémentation principale de ce framework.



Frontières

Responsabilité	Emplacement
Modèles Unity	<code>Assets/Src/Game/Data/ByModule/</code>
Controlllers Unity	<code>Assets/Src/Game/Controlllers/*ByModule*</code> et <code>Assets/Src/UI/SimByModule/Controlllers/</code>
UI By Module	<code>Assets/Src/UI/SimByModule/</code>
Static data partagée	<code>Assets/GameSpecific/byModule/Resources/data_byModule/</code>

Responsabilité	Emplacement
Données propres Acc1/Acc2	Assets/GameSpecific/acc1 acc2/Resources/data_acc*/
Moteur serveur	server_scripts/_utils/by_module/
Base Accounting	server_scripts/acc_base/
Wrappers game-specific	server_scripts/acc1 acc2/command/

Chargement Unity

StaticDataControllerByModule initialise StaticDataByModule, puis les classes Acc1/Acc2 spécialisent au besoin. Le runtime charge à la fois la static data byModule et la static data du GameUid concret.

Flux d'une modification étudiante

```
sequenceDiagram
    participant UI as Page By Module
    participant C as TeamByModuleController
    participant B as BackEndBrainCloud
    participant GS as acc1/acc2 custom_command
    participant U as utils_custom_command_by_module
    participant D as teamsModules

    UI->>C: modification
    C->>B: CustomCommand(commandUid, data)
    B->>GS: game-specific Cloud Script
    GS->>U: executeCommandByModule
    U->>D: lire / modifier / sauvegarder
    U-->>C: réponse + message RTT
    C-->>UI: mise à jour
```

Pages spécialisées déjà existantes

Les pages **Système de formules**, **Configuration Accounting — guide des JSON**, **Pipeline des Bills** et **Procédure annuelle pour updater les taxes** restent les références spécialisées et ne sont pas dupliquées ici.

Ajouter un nouveau module

Accounting — vue end-to-end

Cette checklist est volontairement transversale. Les pages spécialisées donnent le détail de chaque moteur.

1. **Static data** : ajouter le module dans `modules.json` / settings associés et vérifier `regions_settings`.
2. **Pages** : déclarer les `module_pages` et `pages_settings`.
3. **Bills** : configurer les bills; si nouveau type, suivre la page protégée Pipeline des Bills.
4. **Module Type Data** : vérifier que les données initiales générées contiennent tout ce dont le module a besoin.
5. **UI** : vérifier que chaque page est rendue par un type/prefab existant ou ajouter le support manquant.
6. **Answer Key** : ajouter le calcul de chaque nouvelle page évaluée.
7. **Grading** : ajouter le correcteur symétrique et les pondérations.
8. **Results** : vérifier que le module produit les scores et evaluationTeam attendus.
9. **Textes** : ajouter titres, labels, erreurs et tutoriels éventuels.
10. **Tutoriel** : si le module introduit un nouveau flow, ajouter les TutorialUIObject/ScenarioStep nécessaires.
11. **Tests** : create module data → answer key → grade à 100 % → mauvaises réponses.
12. **Région** : tester au moins une région où le module est visible et une où il ne l'est pas.

Test de cohérence global

recommandé

Le meilleur smoke test serveur est de générer le Module Type Data avec un seed fixe, construire l'Answer Key, utiliser cette Answer Key comme décisions étudiant, puis lancer grading + reassemble. Le résultat doit être cohérent à 100 % sur les pages évaluées.

ACC1 vs ACC2

Pour les différences réelles entre les deux jeux — contexte métier, datasets, petite caisse, tutoriels, wrappers et assets — voir [Accounting 1 vs Accounting 2 — différences réelles](#).

