

Answer Key — génération des bonnes réponses

Rôle

`byModule_utils_answer_key_module.js` construit les décisions correctes complètes d'un module. Ce n'est pas une simple liste de réponses : le moteur reconstruit les livres, journaux, états, formulaires et rapports nécessaires pour obtenir l'état comptable attendu.

Pipeline

```
flowchart TD
    MTD[Module Type Data] --> INIT[init answer key]
    INIT --> PAGE[compute each module page]
    PAGE --> AJ[Auxiliary journals]
    PAGE --> GJ[General Journal]
    AJ --> STATE[Account state]
    GJ --> STATE
    STATE --> LED[General Ledger]
    LED --> TB[Trial Balance]
    TB --> FS[Financial Statements]
    PAGE --> FORMS[Tax / payroll / bank forms]
    PAGE --> REG[Regularizations]
    FS --> OUT[pagesData + dropdowns + finished bills]
    FORMS --> OUT
    REG --> OUT
```

Calculs couverts

Le moteur contient des calculateurs spécialisés pour achats, ventes, encaissements, décaissements, payroll, petty cash, formulaires de taxes employé/employeur, subledgers fournisseur/client/stock, general journal, ledger, bank statements, trial balance, income statement, retained earnings/equity, balance sheet, accounting journal et bank reconciliation.

Ordre des dépendances

L'ordre est critique : les journaux alimentent l'état des comptes, qui alimente ledger/trial balance, qui alimente les états financiers. Une modification d'un calcul "local" peut donc modifier plusieurs pages suivantes.

Dropdowns dynamiques

L'answer key produit aussi des `dynamicDropdownsData` lorsque les options dépendent des données générées (partenaires, documents, relations ou lignes existantes).

Tests

- `answer_key_module.test.js`;
- `answer_key_module_functions.test.js`;
- `petty_cash_answer_key_module.test.js`.

Debug

Dans `server_scripts_debugging.js`, `testGeneralFunctionalities()` enchaîne création du Module Type Data → answer key → grading → reassemble. C'est le test manuel le plus direct lorsqu'un module complet ne grade plus à 100 % avec ses bonnes réponses.

Index détaillé des calculateurs

Answer Key

`byModule_utils_answer_key_module.js` est un moteur de plusieurs milliers de lignes. Il faut le lire par **type de page**, pas du haut vers le bas.

Orchestration

- `byModuleAnswerKeyModule()` — entrée publique;
- `answerKeyModule()` — orchestration;
- `initAnswerKeyModuleVariables()` — contexte/settings/pages;
- `computeAnswerKeyModule()` — boucle globale;
- `computeAnswerKeyModulePage(modulePageUid)` — routeur par page.

Journaux

- `computeCorrectPurchasesJournal()`;
- `computeCorrectSalesJournal()`;
- `computeCorrectCashingsJournal()`;
- `computeCorrectWithdrawalsJournal()`;
- `computeCorrectPayrollJournal()`;
- `computeCorrectPettyCashJournal()`;
- `computeCorrectGeneralJournal()`.

Subledgers / ledger / banque

- `computeCorrectProvidersSubledger()`;
- `computeCorrectClientsSubledger()`;
- `computeCorrectStocksSubledger()`;
- `computeCorrectLedger()`;
- `computeCorrectLedgerAutoReport()`;
- `computeCorrectLedgerForBankReconciliation()`;
- `computeCorrectBankStatement*`;
- `computeCorrectBankReconciliation()`.

États financiers

- `computeCorrectTrialBalance()`;
- `computeCorrectStatementIncome()` / Services;
- `computeCorrectStatementRetainedEarnings()`;
- `computeCorrectStatementEquityPartnership()`;
- `computeCorrectStatementEquity()`;
- `computeCorrectBalanceSheet()`;
- `computeCorrectAccountingJournal()`.

Ajouter une nouvelle page à l'Answer Key

1. Ajouter/configurer la page dans les JSON et l'UI — voir Configuration Accounting et UI data-driven.
2. Ajouter le `modulePageUid` au routeur `computeAnswerKeyModulePage()`.
3. Créer une fonction `computeCorrect...` qui lit uniquement les données dont elle dépend.
4. Construire la réponse via les helpers `getOrCreateTableData*`, section/row/input plutôt qu'en réinventant le format.
5. Pour une page dérivée des comptes, utiliser l'account state partagé.
6. Pour une page basée sur des bills, réutiliser `addBillAnswers()`.
7. Ajouter les options/dynamic dropdowns si la correction en dépend.
8. Ajouter le grading symétrique dans `byModule_utils_grade_module.js`.
9. Ajouter un test Answer Key + un test grading.

Account State — cœur transversal

Trial balance et états financiers dépendent de l'état des comptes, lui-même construit en appliquant les relations dans un ordre précis :

- `getFormsAccountsRelations()`;
- `getRegularizationsAccountsOrdersAndRelations()`;
- `getClosingAccountsOrdersAndRelations()`;
- `getAccountStatesOrdersAndRelations()`;
- `updateAccount()` / `updateAccountsState()`.

Si plusieurs pages financières deviennent fausses en même temps, debugger l'account state avant chaque page individuelle.

Guide symptôme → zone

Symptôme	Commencer ici
Achats/ventes faux	Aux journal + bill answers.
Journal correct, ledger faux	<code>computeCorrectLedger*</code> / reconstruction ledger.
Trial balance fausse	Account state + <code>computeCorrectTrialBalance</code> .

Symptôme	Commencer ici
Income statement / balance sheet faux ensemble	Account state / closing / regularizations.
Bank reconciliation fausse	Bank statement + ledger BR + <code>computeCorrectBankReconciliation</code> .
Row/dropdown manquant	Builders structure + dynamic dropdowns + bill answers.

Validation forte

Une Answer Key correcte doit, lorsqu'elle est utilisée comme décisions étudiant, grader à 100 % sur toutes les pages évaluées. `testGeneralFunctionalities()` exécute exactement ce chemin et doit être utilisé comme test de cohérence globale après une modification importante.