

Construire le squelette

Référence d'article

Article complet

<https://b-ziq.com/fr/blog/filemaker-construire-le-squelette/>

Doctrine fondamentale

- L'application des contraintes architecturales devient possible uniquement après la séparation par fichiers et l'établissement de règles d'interaction explicites.
- La structure doit exister avant que l'exécution commence.
- Un système correct sépare le stockage, la mutation et l'initiation dans des fichiers distincts possédant des responsabilités distinctes.
- La séparation par fichiers sans règles d'interaction ne produit pas une architecture en couches.
- La frontière de fichier rend les violations délibérées ; les règles d'interaction déterminent quelles interactions inter-frontières sont valides.
- Les concessions de plateforme sont des impuretés structurelles explicites, localisées et bornées exigées par FileMaker.
- La structure physique résout le problème spatial mais ne résout pas encore le problème d'exécution.

Définitions

- **Concession de plateforme** : impureté structurelle requise par FileMaker, explicitement contenue, localisée et sans propagation.
- **Règles d'interaction** : règles obligatoires définissant quelles interactions entre fichiers sont autorisées, interdites ou contraintes.
- **Passerelle** : frontière obligatoire par laquelle toute communication autorisée entre l'interface et la logique doit passer.

Responsabilités

Data

- Contient les tables de base.
- Stocke ce qui lui est écrit.
- N'exécute aucun comportement.

Logic

- Possède toutes les règles métier.
- Possède toutes les méthodes définissant du comportement métier.
- Possède tous les chemins d'exécution.

- Possède toute la validation avant persistance.
- Possède toutes les écritures dans la couche données.

Menu

- Sert de point principal de coordination de la couche interface.
- Constitue le point d'entrée de la couche interface.

Directory

- Gère les parties dans la couche interface.

Project

- Gère les projets dans la couche interface.

Couche interface

- Collecte l'intention de l'utilisateur.
- Construit les requêtes.
- Appelle les méthodes.
- Reçoit les résultats.
- Détermine leur présentation.

Contraintes

Data

- Doit contenir uniquement les tables de base.
- Ne doit contenir aucune méthode.
- Ne doit contenir aucune règle de saisie automatique calculant des valeurs.
- Ne doit contenir aucun modèle écrivant dans les données.
- Ne doit contenir aucune suppression en cascade via le graphe de relations.
- Ne doit exécuter aucun comportement.

Logic

- Doit contenir toutes les règles métier.
- Doit contenir toutes les méthodes définissant du comportement métier.
- Doit contenir tous les chemins d'exécution.
- Doit contenir toute la validation avant persistance.
- Doit contenir toutes les écritures dans la couche données.
- Ne doit pas dépendre de la structure de l'interface.
- Ne doit pas dépendre des fichiers d'interface pour le comportement métier.

Interface

- Ne doit appliquer aucune règle de validation.
- Ne doit écrire dans aucune table de base.
- Ne doit contenir aucun comportement métier.
- Les méthodes d'interface ne doivent contenir aucune logique métier.

- Les méthodes d'interface ne doivent modifier aucune donnée.

Règles d'interaction

- UI → Logic : autorisé via la Passerelle.
- UI → Data : autorisé en lecture seule, pour la présentation uniquement.
- Logic → Data : autorisé en lecture et en écriture.
- Logic → UI : interdit.
- Les interactions inter-fichiers doivent passer par une frontière.
- Les appels directs UI → Logic sont invalides.
- Tout appel inter-fichiers qui ne passe pas par la frontière est invalide.
- Le fichier Logic est le seul fichier autorisé à écrire dans les tables de base.
- Les lectures UI → Data ne doivent servir ni à la validation, ni aux décisions, ni aux vérifications d'éligibilité, ni aux règles métier.

Concessions de plateforme

- Une concession doit être explicite.
- Une concession doit être localisée.
- Une concession doit être bornée.
- Une concession ne doit pas se propager au-delà de sa portée déclarée.

Invariants

- Un système correct contient trois couches, chacune possédant une responsabilité unique.
- Aucune responsabilité n'est partagée entre deux couches.
- Le stockage est séparé de la mutation.
- La mutation est séparée de l'initiation.
- L'initiation est séparée de la décision.
- La structure détermine les chemins possibles.
- L'exécution détermine quels chemins deviennent réels.
- Le fichier Data est inerte.
- La propriété du comportement métier est exclusive.
- Les fichiers d'interface peuvent évoluer indépendamment du fichier Logic lorsqu'ils ne contiennent aucun comportement propre.
- La direction des dépendances découle des responsabilités des couches.
- Une violation demeure possible après la séparation par fichiers.
- La séparation par fichiers rend les violations vérifiables.

Validation

- Vérifier que la structure existe avant le début de l'exécution.

- Vérifier que chaque fichier contient uniquement les responsabilités qui lui sont assignées.
- Vérifier qu'aucun comportement n'existe dans le fichier Data.
- Vérifier que tout le comportement métier réside dans le fichier Logic.
- Vérifier que les fichiers d'interface ne contiennent aucun comportement métier.
- Vérifier que toutes les écritures dans les tables de base proviennent du fichier Logic.
- Vérifier que toutes les interactions inter-fichiers passent par des frontières déclarées.
- Vérifier que toutes les interactions UI → Logic passent par la Passerelle.
- Vérifier que la direction des dépendances respecte les règles d'interaction déclarées.
- Vérifier que les concessions de plateforme demeurent explicites, localisées, bornées et sans propagation.
- Vérifier que les lectures UI → Data demeurent limitées à la présentation, à la navigation et au contexte des modèles.

Conséquences des violations

- Le comportement émerge à l'extérieur de son propriétaire déclaré.
- La propriété du comportement métier devient distribuée.
- La mutation distribuée réapparaît.
- Le comportement devient conditionnel au contexte de présentation.
- L'inertie de la couche Data est rompue.
- La mutation non contrôlée est réintroduite.
- Le couplage implicite est réintroduit par inversion des dépendances.
- L'architecture se transforme en interactions inter-fichiers sans restriction.
- Les frontières cessent de contraindre le comportement.
- Les violations deviennent des composants structurels du système plutôt que des défauts isolés.
- Le système perd les conditions nécessaires à l'application des contraintes architecturales.