

Ce que des décennies de génie logiciel peuvent apprendre à un développeur FileMaker

Référence d'article

Article complet

<https://b-ziq.com/fr/blog/principes-fondamentaux/>

Doctrine fondamentale

- Les défaillances observées dans les systèmes FileMaker maintenus sur le long terme sont des problèmes de génie logiciel, non des problèmes propres à FileMaker.
- Les contraintes existent pour éliminer des classes de comportements implicites et rendre le système prévisible.
- Une méthode est la seule unité de comportement autorisée : liée par contrat, indépendante du contexte, avec des entrées et sorties déclarées.
- La séparation des responsabilités, les contrats explicites, le déterminisme, le couplage et la cohésion, ainsi que la gestion explicite de l'état sont des exigences structurelles.
- Le contexte implicite, l'état ambiant, les chemins d'exécution non contrôlés et les dépendances non déclarées sont incompatibles avec un système maintenable.
- L'application cohérente de ces contraintes converge vers une structure où une couche logique possède les règles métier, où l'état est explicite et où l'exécution passe uniquement par des chemins contrôlés.

Définitions

Méthode

Unité de comportement liée par contrat, avec une entrée déclarée et une sortie déclarée, indépendante du contexte d'exécution.

Script

Détail d'implémentation constitué d'une séquence d'étapes exécutées par FileMaker.

Séparation des responsabilités

Séparation de l'interaction et de la présentation, des règles métier et du stockage en responsabilités distinctes.

Contrat explicite

Frontière comportementale déclarée définissant les entrées requises et les sorties produites.

Déterminisme

Propriété selon laquelle des entrées identiques produisent toujours des sorties identiques, indépendamment de l'origine ou du chemin d'exécution.

Couplage

Mesure du degré de dépendance d'un comportement envers d'autres composants, états, contextes ou historiques d'exécution.

Cohésion

Mesure du degré auquel une unité de comportement accomplit une seule responsabilité.

Gestion explicite de l'état

Exigence selon laquelle tout état nécessaire à un comportement doit être déclaré et transmis intentionnellement plutôt qu'hérité du contexte.

Responsabilités

Couche Interface

- Gère l'interaction.
- Gère la présentation.

Couche Logique

- Gère les règles métier.
- Possède l'exécution des comportements.

Couche Données

- Gère le stockage.

Contraintes

- La logique doit s'exécuter uniquement par l'intermédiaire de méthodes.
- Une méthode doit définir explicitement ses entrées et ses sorties.
- Une méthode ne doit pas dépendre du contexte d'exécution.
- La couche Interface ne doit pas posséder de règles métier.
- La couche Logique ne doit pas dépendre de la structure de présentation.
- Les responsabilités de stockage doivent rester séparées des règles métier.
- Les entrées nécessaires à un comportement doivent être déclarées explicitement.
- Les sorties produites par un comportement doivent être déclarées explicitement.
- Le comportement ne doit pas dépendre du modèle actif.

- Le comportement ne doit pas dépendre de l'ensemble trouvé courant.
- Le comportement ne doit pas dépendre de l'enregistrement actif.
- Le comportement ne doit pas dépendre de variables globales ou d'un état ambiant.
- Les chemins d'exécution doivent être contrôlés.
- Les dépendances doivent être visibles, déclarées et intentionnelles.
- Tout état important doit être transmis explicitement et ne doit pas circuler de façon invisible à travers l'exécution.

Invariants

- Les contraintes éliminent des classes de comportements implicites.
- Un script est un détail d'implémentation et non un contrat comportemental.
- Une méthode est un contrat comportemental.
- Un comportement déterministe dépend uniquement d'entrées déclarées.
- Une forte cohésion et un faible couplage caractérisent un système maintenable.
- L'état implicite est hérité du contexte.
- L'état explicite est déclaré et transmis intentionnellement.
- Le contexte est une forme d'état implicite.
- Un système compréhensible demeure maintenable dans le temps.

Validation

Un système respecte cette doctrine lorsque :

- Les comportements sont exprimés sous forme de méthodes avec entrées et sorties déclarées.
- Chaque méthode peut être exécutée indépendamment du contexte de son appelant.
- Des entrées identiques produisent des sorties identiques.
- Les responsabilités de l'interface, de la logique et des données demeurent séparées.
- Les dépendances sont explicites et observables.
- L'état nécessaire à un comportement est transmis intentionnellement.
- Les comportements demeurent indépendants du modèle actif, de l'ensemble trouvé, de l'enregistrement actif, de l'historique d'exécution et de l'état ambiant.
- Les règles métier s'exécutent dans la couche Logique.
- Les écritures s'effectuent par l'intermédiaire de la couche Logique et non par mutation pilotée par l'interface.

Conséquences des violations

- Le comportement devient dépendant du contexte.
- Les bogues deviennent conditionnels et difficiles à reproduire.

- Des modifications dans une partie du système provoquent des défaillances dans des parties apparemment sans rapport.
- Les dépendances deviennent invisibles.
- Les tests deviennent non fiables.
- La compréhension du système se dégrade avec le temps.
- Le coût de maintenance augmente à chaque modification.
- Les corrections introduisent de nouvelles surfaces de défaillance.
- Le système devient progressivement plus difficile à modifier de manière sûre.
- Le système tend vers un remplacement ou une réécriture complète.