Architecture multi-tiers

AMT

1 - Multi-Tier Architecture And Patterns

Résumé du document

Ce document présente l'architecture multi-tier, également appelée n-layer ou architecture en couches, un modèle couramment utilisé pour le design d'applications modernes. Il explique comment cette approche divise l'application en plusieurs couches avec des responsabilités spécifiques, permettant une meilleure organisation du code et une séparation des responsabilités.

Table des matières

1. Définition

L’architecture multi-tier aussi connue sous le nom de **n-layer** ou **layered** architecture est un sytle d’architecure commune pour le design d’application modernes.

Le principe est de décomposé l’application en couche, ou chaqu’une est responsable de tâches spécifiques.

L’architecture la plus commune des architectures multi-tiers est celle en **3 couches** contenant les couches **presentation**, **business** et **data**.

2. Layers et tiers

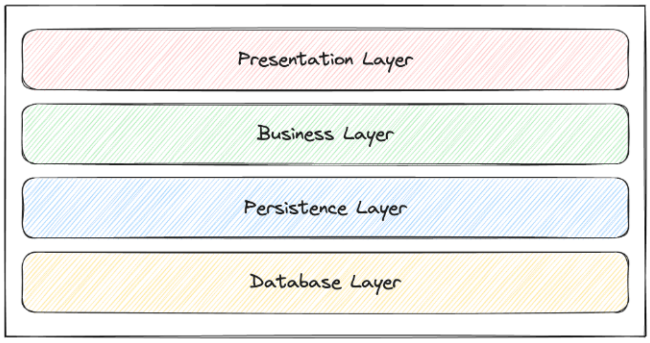
Les **layers** sont des divisions logiques du code pour organiser et séparer les fonctionnalités, tandis que les **tiers** sont des divisions physiques qui répartissent l'application sur plusieurs machines pour renforcer la sécurité, les performances, l'évolutivité et la tolérance aux pannes.

2.1. La loie de conways

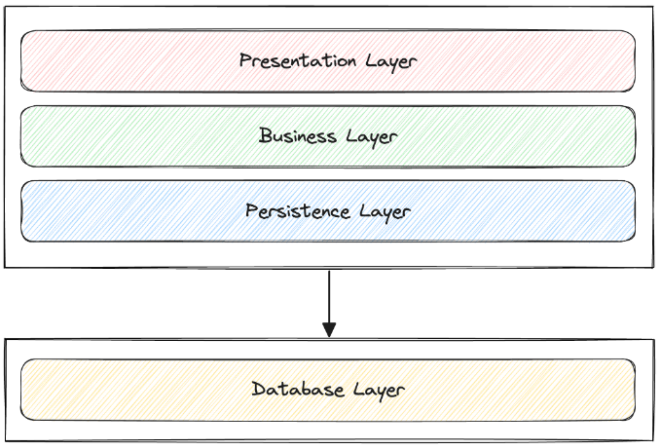
Toute organisation qui conçoit un système (défini au sens large) produira une conception dont la structure est une copie de la structure de communication de l'organisation.

2.2. Structure des architectures

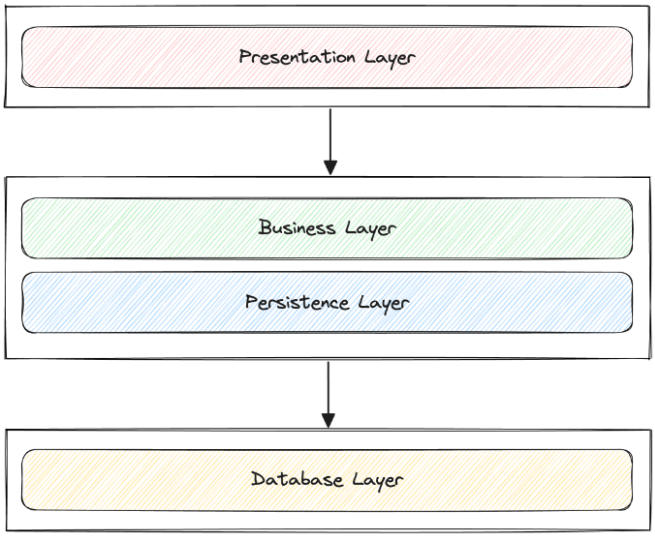
2.2.1. Architecture en couches



2.2.2. Architecture en 2 couches



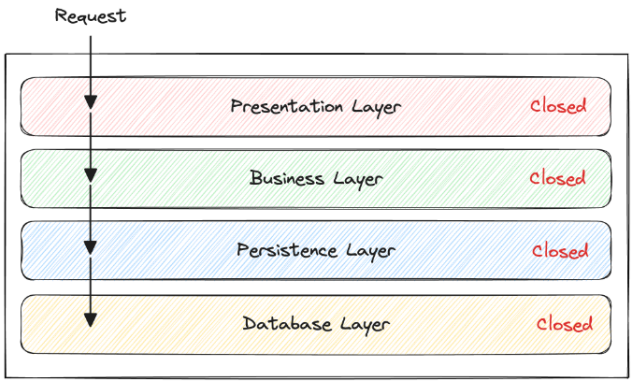
2.2.3. Architecture en 3 couches



- La couche de présentation est responsable de l'interface utilisateur (par exemple, web, mobile).
- La couche métier est responsable de la logique métier (par exemple, l'authentification, l'autorisation, la validation, les transactions, etc.)
- La couche de persistance est responsable de l'accès aux données (entités, référentiels, objets de domaine, etc.).
- La base de données est responsable du stockage des données (tables, vues, index, etc.).

Il est courant de distinguer la couche de persistance de la couche de base de données, car la base de données est rarement intégrée ou gérée par la couche de persistance.

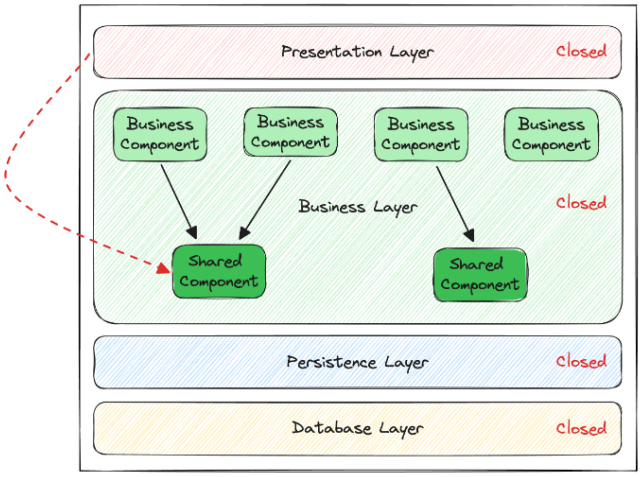
2.3. Isolation des couches

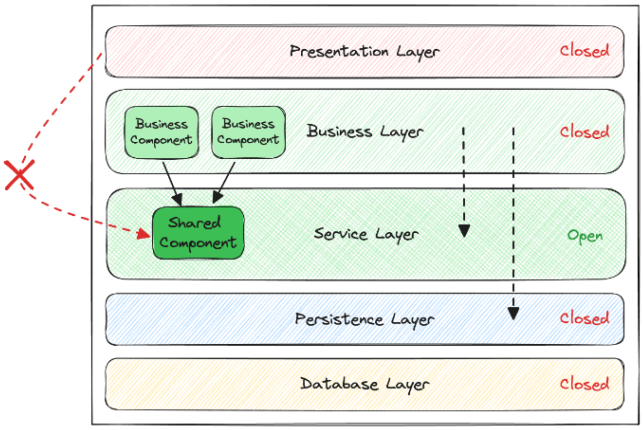


- **Types de couches :**
  - **Couche fermée** : accessible uniquement par la couche au-dessus.
  - **Couche ouverte** : accessible par n'importe quelle autre couche.

- **Inconvénients des couches ouvertes :**
  - Tentation de rendre toutes les couches ouvertes pour simplifier l'accès.
  - Peut entraîner des pratiques inadéquates, comme l'accès direct de la couche de présentation à la base de données.
  - Provoque un couplage fort, rendant l'application plus difficile à maintenir et à faire évoluer.
  - Risque de créer un code spaghetti
- **Avantages des couches fermées :**
  - Meilleure séparation des préoccupations, réduisant le couplage entre les couches.
  - Facilite la maintenance, car les changements dans une couche impactent moins les autres.
  - Permet des changements prévisibles et faciles à tester.
- **Difficultés :**
  - Maintenir une isolation entre les couches nécessite du code supplémentaire (**boilerplate**).
  - Limiter le nombre de couches peut réduire la complexité accidentelle.

2.4. Réduire la complexité en ajoutant une couche?





Dans cette situation une nouvelle couche de service a du être ajoutée permettant de gérer les composants partagés. Cette couche est ouverte et accessible par l'ensemble des autre couches.

- **Accès des couches :**
  - La couche de présentation ne peut accéder qu'à la couche métier, qui est fermée, renforçant la séparation des responsabilités.
- **Complexité de l'architecture :**
  - Ajouter une nouvelle couche augmente la complexité de l'application.
  - Cette décision est souvent prise par l'architecte lorsque l'application grandit et que la couche métier devient trop complexe.
- **Bonne pratiques :**
  - Commencer avec un nombre réduit de couches et en ajouter au fur et à mesure que l'application se développe.
  - Le respect de ces principes architecturaux repose sur la discipline de l'équipe.