Comparatif des architectures logicielles

Architecture MVC

- **Description:**

L'architecture MVC divise une application en trois parties : le modèle (représentation des données), la vue (représentation graphique des données) et le contrôleur (gère les entrées de l'utilisateur et met à jour le modèle).

- **Avantages:**
  1. Séparation claire des responsabilités entre les différentes parties de l'application.
  2. Facilite la maintenance et l'évolutivité de l'application.
  3. Favorise la réutilisation du code.

- **Inconvénients:**
  1. Peut devenir complexe à mesure que l'application se développe.
  2. Peut être difficile à mettre en place pour les débutants.

Architecture SOA

- **Description:**

L'architecture SOA découpe une application en services autonomes, qui peuvent être combinés pour fournir une fonctionnalité globale.

- **Avantages:**
  1. Permet une grande flexibilité et un découplage entre les différents services.
  2. Facilite l'intégration de nouvelles fonctionnalités dans une application existante.
  3. Favorise la réutilisation des services.

- **Inconvénients**
  1. Peut être coûteux à mettre en place et à maintenir.
  2. Peut être difficile à déboguer en raison de la complexité des interactions entre les différents services.

Architecture Microservices

- **Description:**

L'architecture de microservices consiste à découper une application en services indépendants, qui peuvent être développés, déployés et mis à l'échelle de manière autonome.

- **Avantages:**
  1. Permet une grande flexibilité et un découplage entre les différents services.
  2. Facilite l'évolutivité et la maintenance de l'application.
  3. Favorise la réutilisation des services.
  4. Facilite la gestion des services individuels.

- **Inconvénients:**
  1. Peut être complexe à mettre en place et à maintenir.
  2. Peut entraîner des problèmes de cohérence des données entre les différents services.
  3. Nécessite une gestion rigoureuse des communications entre les différents services.

Comparatif des architectures logicielles

Client/Serveur

- **Description:**

L'architecture client/serveur divise une application en deux parties : le client (interface utilisateur) et le serveur (traitement des requêtes et gestion des données).

- **Avantages:**
  1. Séparation claire des responsabilités entre le client et le serveur.
  2. Permet une meilleure gestion des ressources et une répartition des tâches entre le client et le serveur.
  3. Favorise la scalabilité et la disponibilité du système..

- **Inconvénients:**
  1. Peut entraîner une dépendance étroite entre le client et le serveur.
  2. Peut engendrer une charge réseau importante.
  3. Nécessite une gestion appropriée des mises à jour côté client et serveur.

N-tiers

- **Description:**

Cette architecture respecte le principe de séparation des responsabiltés et facilite la compréhension des échanges au sein de l'application.
Lorsque chaque couche correspond à un processus distinct sur une machine, on parle d'architecture n- tiers, n désignant le nombre de couches.

- **Avantages:**
  1. Favorise la modularité et la réutilisation du code.
  2. Permet une gestion simplifiée des modifications et des mises à jour.
  3. Facilite la maintenance et l'évolutivité du système.

- **Inconvénients:**
  1. Peut entraîner une dépendance étroite entre le client et le serveur.
  2. Peut engendrer une charge réseau importante.
  3. Nécessite une gestion appropriée des mises à jour côté client et serveur.