ASI - Atelier 2 : Architectures Logicielles

MVC :

L'architecture MVC (Modèle-Vue-Contrôleur) est l'une des architectures logicielles les plus utilisées pour les applications Web. Elle se compose de 3 modules :

- Modèle : noyau de l'application qui gère les données, permet de récupérer les informations dans la base de données, de les organiser pour qu'elles puissent ensuite être traitées par le contrôleur.
- Vue : composant graphique de l'interface qui permet de présenter les données du modèle à l'utilisateur.
- Contrôleur : composant responsable des prises de décision, gère la logique du code qui prend des décisions, il est l'intermédiaire entre le modèle et la vue.

Avantages de l'approche MVC :

- Meilleure organisation du code
- Diminution de la complexité lors de la conception
- Conception claire et efficace grâce à la séparation des données de la vue et du contrôleur
- Possibilité de réutilisation de code dans d'autres applications
- Un gain de temps de maintenance et d'évolution du site
- Une plus grande souplesse pour organiser le développement du site entre différents développeurs
- Plus de facilité pour les tests unitaires

Inconvénient de cette approche :

- Augmentation de la complexité lors de l'implémentation
- Eventuel cloisonnement des développeurs
- Architecture complexe pour des petits projets
- Le nombre important de fichiers représente une charge non négligeable dans un projet

SOA :

L'architecture orientée services (ou SOA, Service-Oriented Architecture) est un modèle de conception qui rend des composants logiciels réutilisables, grâce à des interfaces de services qui utilisent un langage commun pour communiquer via un réseau.

En d'autres termes, l'architecture SOA permet à des composants logiciels déployés et gérés séparément de communiquer et de fonctionner ensemble sous la forme d'applications logicielles communes à différents systèmes.

Avantages par rapport à une approche monolithique :

- **Mise sur le marché accélérée et plus grande flexibilité :** le caractère réutilisable des services facilite et accélère le regroupement des applications. Les développeurs ne doivent plus repartir de zéro à chaque fois, comme c'est le cas pour les applications monolithiques.
- **Utilisation de l'infrastructure existante sur les nouveaux marchés :** grâce à l'architecture SOA, les développeurs peuvent plus facilement étendre et mettre à l'échelle les fonctionnalités d'une plateforme ou d'un environnement.

- **Maintenance facilitée :** les services étant autonomes et indépendants, ils peuvent être modifiés et mis à jour autant que nécessaire, sans affecter les autres services.
- **Évolutivité :** comme l'architecture SOA s'adapte à plusieurs services, plateformes et langages de programmation, l'évolutivité est considérablement accrue. En outre, le protocole de communication standardisé limite les interactions entre les clients et les services pour les entreprises, En diminuant le degré d'interaction, les applications peuvent être mises à l'échelle plus facilement et avec moins de pression.
- **Fiabilité améliorée :** il est plus facile de déboguer des petits services plutôt qu'un long code, ce qui permet de créer des applications plus fiables.
- **Grande disponibilité :** les fonctions de l'architecture SOA sont à la portée de tous.

Inconvénients du SOA :

- **Coût de la maintenance** : La maintenance d'un système SOA est assez coûteuse en termes de main-d'œuvre. Plusieurs équipes d'unités informatiques et de développeurs sont nécessaires pour configurer et gérer chaque code de service. Bien que ce coût soit principalement supporté par les prestataires de services tiers, il peut en fin de compte se traduire par des coûts financiers pour l'entreprise et ses clients.
- **Surcharge du trafic de données** : Dans le cadre des protocoles de sécurité, les commandes et les entrées doivent être vérifiées avant d'être transférées au tiers. À mesure que le nombre de services sur une architecture SOA augmente, le nombre d'entrées et de codes de vérification pour ces entrées augmente également. La quantité supplémentaire de trafic de données chargera votre système.
- **Besoin en ressources plus élevé** : L'élément d'une charge de données plus élevée nécessite une bande passante plus importante pour garantir que la vitesse du réseau est suffisante pour assurer la satisfaction du client.

MicroService :

Une architecture de MicroServices est un type d'architecture d'application dans laquelle l'application est développée sous la forme d'un ensemble de services. Elle fournit le framework permettant de développer, déployer et gérer de manière indépendante des diagrammes et des services d'architecture de MicroServices.

Les avantages :

- Equipe de travail minimale
- Evolutivité
- Fonctionnalité modulaire, modules indépendants
- Liberté des développeurs pour développer et déployer des services de manières indépendante
- Utilisation de conteneurs, permettant un déploiement et un développement rapides de l’application

Les inconvénients :

- Utilisation élevée de la mémoire
- Temps nécessaire pour fragmenter différents MicroServices
- Complexité de la gestion d’un grand nombre de services
- Les développeurs doivent résoudre des problèmes tels que la latence du réseau ou l’équilibrage de charge
- Tests complexes sur le déploiement distribué