Web Statique & Dynamique

2023-2024

Théo CLERE, Maxime BROSSARD, Sandro SPINA, Ceif-Edine MAROUANI, Julien BUC
CPE LYON

Chaque architecture a ses propres contextes d'utilisation où elle excelle et ses défis spécifiques. Le choix de l'architecture dépend des exigences spécifiques du projet, telles que la scalabilité, la complexité, la maintenance et les performances. Vous trouverez le tableau comparatif de différentes architectures ci-dessous.

Tableau comparatif

Architecture	Description	Avantages	Inconvénients
MVC (Model-View-Controler)	L'application sera séparée en trois composants : - Modèle (données) - Vue (interface utilsateur) - Contrôleur (logique)	Bonne séparation des tâches. Facilite la maintenance et l'évolution. Réutilisabilité des composants.	Complexité accrue pour les petites applications. Communication intensive entre les composants.
SOA (Service-Oriented-Architecture)	Basée sur des services indépendants qui communiquent via des protocoles de réseau.	Réutilisabilité des services. Scalabilité. Facilite l'intégration avec des systèmes hétérogènes.	Complexité de gestion des services. Performance potentiellement impactée par la communication réseau.
Microservices	Division de l'application en services indépendants, chaque service gère une fonctionnalité spécifique.	Scalabilité et flexibilité élevées. Déploiement indépendant des services. Résilience et tolérance aux pannes.	Complexité de gestion et de déploiement. Problèmes de latence et de gestion des données distribuées.
Monolithique	Application unique et indivisible où tous les composants sont interconnectés.	Simplicité de développement et de déploiement. Performance interne élevée.	Difficulté de maintenance et d'évolution. Scalabilité limitée.
Serverless	Les fonctions sont déployées sur une infrastructure cloud, exécutées en réponse à des événements.	Pas de gestion de serveur. Scalabilité automatique. Coût réduit (dépend des cas).	Dépendance vis-à-vis du fournisseur cloud. Limitation en termes de temps d'exécution et de ressources.
Client-Serveur	Architecture traditionnelle où le client demande des services au serveur, qui les fournit.	Séparation claire entre client et serveur. Centralisation des ressources.	Dépendance du client vis-à-vis du serveur. Problèmes de scalabilité du serveur.