Architecture logicielle et modèles de conception

Architecture logicielle et modèles de conception

Catégories : Expertise DSI Par Christine Moronval
Tags: Architecture Logicielle Commentaires fermés

Une architecture logicielle rassemble, à un haut niveau, une vue d'ensemble des différents composants, langages, méthodologies

permettant, in fine, de répondre au besoin exprimé par le demandeur.

Comme le précise Luc Fournier, Consultant Architecte Cloud chez SoftFluent

> *L'objectif est de décrire les solutions ainsi que les modèles qui seront mis en œuvre sans les détails d'implémentation qui seront traités lors des phases de conception logicielle. Il s'agit de rendre les choix technologiques secondaires et interchangeables jusqu'au dernier moment.*

Une architecture logicielle 'saine' est une architecture qui est caractérisée par différentes couches indépendantes les unes par rapport aux autres : de l'interface utilisateur jusqu'aux règles métier et à la persistance qui sont au cœur des applicatifs présents dans les systèmes d'informations.

Le but ultime de l'architecture logicielle c'est de :

- Faciliter le développement, l'évolution, le déploiement et la maintenance d'un système
- Minimiser le temps et le coût d'intervention
- Maximiser et maintenir la productivité des développeurs face aux changements

- Rendre tout ajout ou modification simple et rapide
- Optimiser les capacités d'interconnections et de compatibilité avec les autres briques du système d'information

Avant de faire des choix d'architecture, il est donc utile de commencer par réfléchir aux objectifs stratégiques avec lesquels l'application devra être en adéquation.

Vous pourrez alors concevoir une architecture susceptible de répondre à vos besoins, plutôt que d'opter pour une architecture et essayer ensuite de l'adapter à votre application.

L'architecture fournit une feuille de route ainsi que les meilleures pratiques à suivre pour créer une application bien structurée.

Quel type d'architecture logicielle ?

Architectures monolithiques

L'architecture monolithique est composée d'une unique pile qui rassemble toutes les fonctionnalités au sein d'une seule application.

Chaque changement apporté au code implique de republier l'application tout entière ce qui est forcément un frein aux mises à jour, sans même parler d'ajout de nouvelles fonctionnalités.

Les architectures modernes, au contraire, essaient de décomposer les services en fonctionnalités faiblement couplées pour fournir

davantage d'agilité et de flexibilité.

Architectures de microservices

Une architecture microservices a pour objet de diviser une application en fonctionnalités encapsulées au sein de services autonomes. Chacun de ces services est géré et évolue indépendamment des autres services.

Les Microservices peuvent être mis à jour, étendus et déployés indépendamment les uns des autres et, par conséquent, beaucoup plus rapidement tout en limitant le risque d'une mise en péril l'ensemble de l'applicatif.

Les Microservices peuvent communiquer entre eux sans état, à l'aide d'interfaces de programmation d'application (API) indépendantes de tout langage.

Le choix technologique dans une architecture Microservices est donc totalement ouvert. Il dépend majoritairement des besoins, de la taille des équipes et de leurs compétences.

Ces architectures d'applications faiblement couplées qui reposent sur des Microservices avec des APIs et des pratiques DevOps constituent la base des applications cloud-native.

Le développement cloud-native est une manière d'accélérer la création des nouvelles

applications, d'optimiser les applications existantes, d'harmoniser le développement et d'automatiser la gestion pour l'ensemble des clouds privés, publics et hybrides.

Architectures orientées événements

Une architecture orientée événements est un modèle d'architecture logicielle asynchrone et faiblement couplée. Elle est donc adaptée aux architectures d'applications modernes distribuées

Dans un système orienté événements, la structure centrale repose sur le traitement et la persistance des événements, à savoir tout phénomène ou changement d'état significatif.

Une architecture Microservices peut être orientée événement.

Architectures orientées services

L'architecture orientée services (SOA) est un modèle de conception largement utilisé, similaire au modèle d'architecture de Microservices.

Alors que les Microservices structurent une application comme une série de services distincts à usage unique, la SOA regroupe des services modulaires qui « dialoguent » entre eux pour prendre en charge les applications et leur déploiement par l'intermédiaire d'un service Bus.

Les files d'attente de messages assurent donc la

coordination entre ces applications distribuées. Elles peuvent considérablement simplifier le codage des applications découplées, fluidifier les pics de charge, tout en améliorant les performances, la fiabilité et l'évolutivité.

Pour construire une application, vous pouvez vous aider de modèles de conception logicielle.

Les modèles de conception dans l'architecture logicielle

Les modèles de conception sont des solutions éprouvées et réutilisables par rapport à un problème fréquent, considérées comme bonnes pratiques et pas spécifiques à un langage de programmation en particulier. Imaginons une application ayant besoin d'utiliser un ensemble complexe d'API, avec des appels à de nombreuses procédures différentes, l'utilisation d'un modèle 'façade' fournit une interface plus simple et plus uniforme. C'est une couche de code intermédiaire qui appelle les API appropriées.

appropriées.

Au lieu de créer vous-même l'architecture logicielle de bout en bout, il est possible de lier plusieurs modèles de conception existants ou de vous en inspirer.

Ces modèles, pourtant ni savants ni compliqués, mais issus des bonnes pratiques au fil du temps, élargissent les possibilités de votre application et améliorent la qualité des développements.

Généralement associés à une programmation orientée objet, il existe trois modèles de conception distincts :

1. Modèles de création

Les modèles de création permettent une représentation simplifiée du processus pour certaines instances, en d'autres termes, ils séparent le développement des objets complexes de leur représentation. Par exemple, le modèle « Singleton » est utilisé pour créer une classe de base qui n'aura qu'une seule instance.

2. Modèles de structure

Les modèles de structure sont prêts à l'emploi pour les relations entre les classes. Par exemple, le modèle « Données de classe privées » est utilisé pour limiter l'accès à une classe spécifique et ainsi prévenir la modification non souhaitée d'un objet. Le modèle « Décorateur », permet quant à lui, d'ajouter des fonctionnalités aux classes existantes.

Le modèle 'Façade' évoqué plus haut entre dans cette catégorie.

3. Modèles de comportements

Les modèles de comportement permettent de modéliser les comportements d'un logiciel,

notamment la manière dont ils communiquent entre eux. Les comportements sont définis via des abstractions avec des responsabilités bien spécifiques.

Les modèles de conception sont des outils utiles pour la programmation, il est aussi possible de s'en inspirer pour créer une application au plus près du besoin. Il ne faut pas pour autant s'interdire de chercher de nouvelles solutions qui peuvent être plus efficaces.