Génie Logiciel – Architectures Applicatives

Sommaire

1. Définition des architectures applicatives

Les architectures applicatives peuvent être vues comme les fondations d'un projet. Elle répartie la distribution des ressources (ou des rôles) au sein d'un réseau.

Chaque élément d'un réseau peut être vu comme une couche qui représente un rôle à jouer (exemple : une base de données et un serveur applicatif).

Il existe plusieurs architectures applicatives comme celle du Client/serveur, des architecture tiers ou encore l'architecture MVC.

2. Architecture Client/serveur

2.1. Présentation

Dans l'architecture client-serveur, le client demande une ressource ou une fonctionnalité au serveur. Lorsque le serveur a renvoyé l'information au client, la communication est terminée.

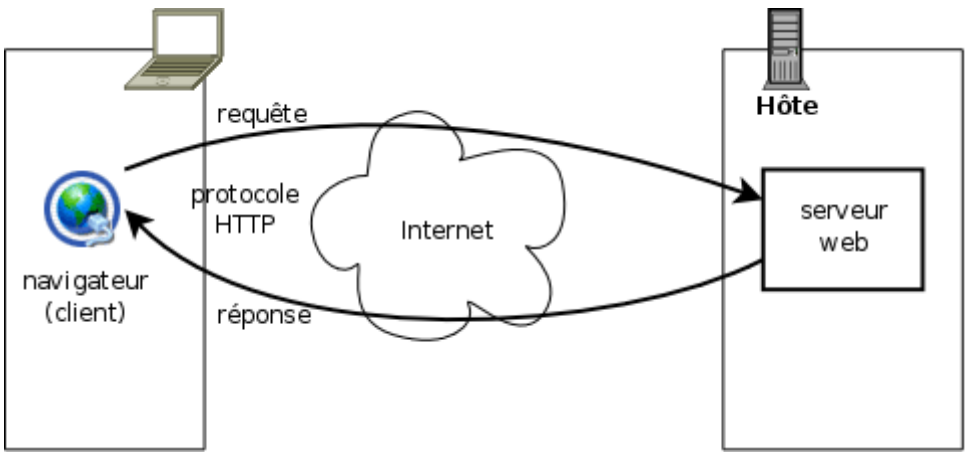
2.2. Rôles

L'architecture client-serveur permet de fournir un service à un ou plusieurs clients à partir d'un serveur. Ce type d'application fait souvent partie d'une application à plus grande échelle qui utilisera une architecture type 3 tiers ou N-tiers.

2.3. Interaction avec les composants

L'architecture client-serveur la plus connue est celle du site internet qui est consulté via un navigateur web.

Le navigateur représente le client tandis que le l'application web représente le serveur. Dans le cas de l'architecture client-serveur, le site internet est une application statique qui n'a pas de base de données associée.



2.4. Avantages et inconvénients

L'avantage principal de l'architecture client-serveur est l'administration simplifiée puisqu'un seul élément centralisé supporte l'ensemble des ressources coté serveur.

Cet élément est aussi un inconvénient puisqu'un seul élément centralisé créé un maillon faible étant donné que tous les clients gravitent autour de lui.

3. Architecture trois-tiers

3.1. Présentation

Dans cette architecture, chaque niveau est indépendant et peuvent être implantés sur des machines différentes.

De plus, le poste client est appelé client léger par opposition au client lourd de l'architecture client/ serveur. Il ne prend en charge que la présentation des applications.

3.2. Rôles

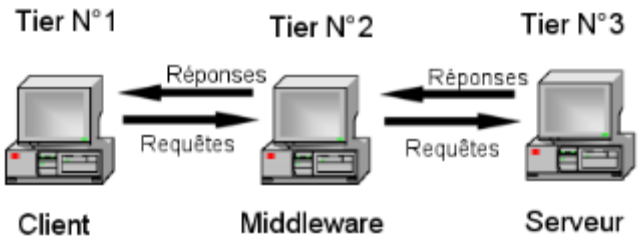
L'architecture 3-tiers sépare l'application en trois niveaux

Niveau 1 : correspond à la couche présentation et est la partie de l'application visible et interactive avec les utilisateurs. On parle d'interface homme-machine.

Niveau 2 : correspond à la partie fonctionnelle de l'application, c'est celle qui décrit les opérations que l'application exécute sur les données en fonction des requêtes des utilisateurs effectués à partir de la couche présentation. De plus, c'est dans cette couche que les différentes règles de gestion et de contrôle sont mise en œuvre.

Niveau 3 : autrement dit la couche données, elle est la partie qui gère la base de données du système. Les données peuvent être propres au système ou gérées par un autre système.

3.3. Interaction avec les composants



1- La couche présentation relaie les requêtes de l'utilisateur à destination de la couche métier et en retour lui présente les informations renvoyées par le traitement.
2- La couche applicative suggère des services applicatifs et métier à la couche présentation. Elle s'appuie sur les données du système accessible depuis la couche données
3- La couche applicative renvoie à la couche présentation les résultats qu'elle a calculés.

3.4. Avantages et inconvénients

Avantages	Limites
Déploiement immédiat Evolution transparentes pour l'utilisateur Caractéristiques du poste client libres	Le serveur d'application réalise la majorité des traitements

4. Architecture N-tiers

4.1. Présentation

L'architecture n-tiers est aussi appelée architecture distribuée ou architecture multi-tiers.

L'architecture n-tiers reprend les 3 niveaux de celle du 3 tiers. De plus, elle permet de concevoir des applications puissantes et simples à maintenir. Ce type d'architecture permet de distribuer plus librement la logique applicative, ce qui facilite la répartition de la charge entre tous les niveaux.

Cette distribution est facilitée par l'utilisation de composants « métier », spécialisés et indépendants, introduits par les concepts orientés objets (langages de programmation et middleware). Elle permet de tirer pleinement partie de la notion de composants métiers réutilisables.

Ces composants rendent un service si possible générique et clairement identifié. Ils sont capables de communiquer entre eux et peuvent donc coopérer en étant implantés sur des machines distinctes.

La distribution des services applicatifs facilite aussi l'intégration de traitements existants dans les nouvelles applications. On peut ainsi envisager de connecter un programme de prise de commande existant sur le site central de l'entreprise à une application distribuée en utilisant un middleware adapté.

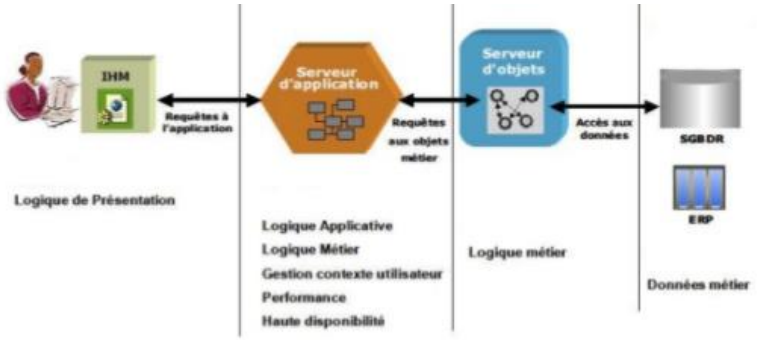
Les technologies :

Tiers clients	Tiers applicatifs	Tiers données
Un navigateur WEB Un PDA ou Smartphone	CGI ASP JSP Java Php Javascript	Base de données J2EE ERP EAI

4.2. Interaction entre les composants

L'architecture n-tiers est composée généralement :

- Une couche présentation : contient le client (léger ou lourd)
- Une couche applicative : contient les traitements représentant les règles métier (créer un compte, rechercher un client, etc.)
- Une couche objets métier est composée des objets du domaine, autrement dit des entités persistantes de l'application (Client, Facture, etc.)
- Une couche base de données



4.3. Avantages / Inconvénients

Avantages	Inconvénients
Séparation de tous les niveaux de l'application Offre de grandes capacités d'extension Facilite la gestion des sessions	Les coûts de développement sont plus élevés

5. Architecture modèle vue contrôleur (MVC)

5.1. Présentation

Le modèle MVC décrit une manière d'architecturer une application informatique en la décomposant en trois sous-parties :

- la partie Modèle ;
- la partie Vue ;
- la partie Contrôleur.

Ce modèle de conception (« *design pattern* ») a été imaginé à la fin des années 1970 pour le langage Smalltalk afin de bien séparer le code de l'interface graphique de la logique applicative. Il est utilisé dans de très nombreux langages : bibliothèques Swing et Model 2 (JSP) de Java, *frameworks* PHP, ASP.NET MVC, etc.

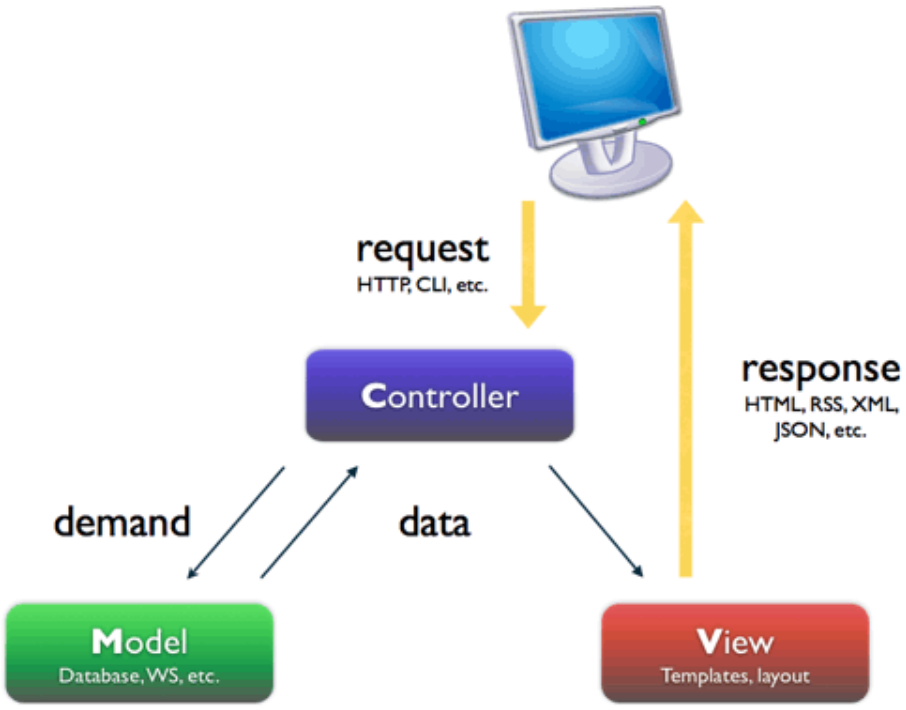
5.2. Rôles

La partie **Modèle** d'une architecture MVC encapsule la logique métier (« *business logic* ») ainsi que l'accès aux données. Il peut s'agir d'un ensemble de fonctions (Modèle procédural) ou de classes (Modèle orienté objet).

La partie **Vue** s'occupe des interactions avec l'utilisateur : présentation, saisie et validation des données.

La partie **Contrôleur** gère la dynamique de l'application. Elle fait le lien entre l'utilisateur et le reste de l'application.

5.3. Interaction entre les composants



1. La demande de l'utilisateur (exemple : requête HTTP) est reçue et interprétée par le **Contrôleur**.
2. Celui-ci utilise les services du **Modèle** afin de préparer les données à afficher.
3. Ensuite, le **Contrôleur** fournit ces données à la **Vue**, qui les présente à l'utilisateur (par exemple sous la forme d'une page HTML).

Une application construite sur le principe du MVC se compose toujours de trois parties distinctes. Cependant, il est fréquent que chaque partie soit elle-même décomposée en plusieurs éléments. On peut ainsi trouver plusieurs modèles, plusieurs vues ou plusieurs contrôleurs à l'intérieur d'une application MVC.

5.4. Avantages et inconvénients

Le modèle MVC offre une séparation claire des responsabilités au sein d'une application, en conformité avec les principes de conception déjà étudiés : responsabilité unique, couplage **faible** et cohésion **forte**. Le prix à payer est une augmentation de la complexité de l'architecture.

Dans le cas d'une application Web, l'application du modèle MVC permet aux pages HTML (qui constituent la partie Vue) de contenir le moins possible de code serveur, étant donné que le scripting est regroupé dans les deux autres parties de l'application.

6. Architecture orientée service (SOA)

6.1. Présentation

SOA (ou Architecture Orienté Service) a pour but de standardiser les échanges inter-applications. Il repose sur le standard XML pour la description des données et sur SOAP pour le transport des données.

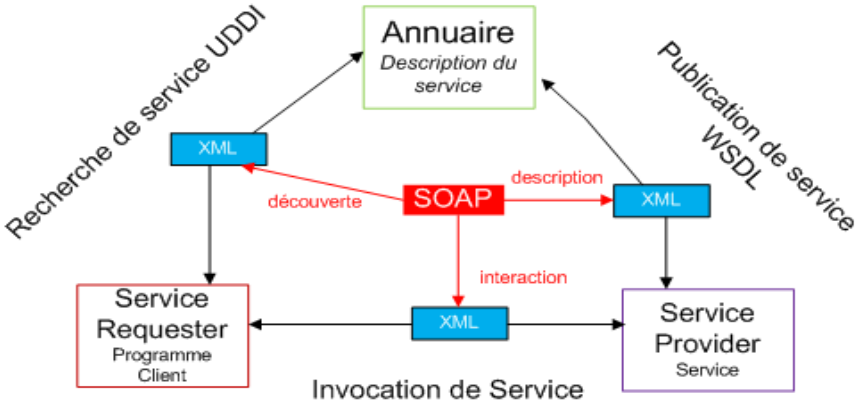
6.2. Rôles

Les différents rôles sont:

- Annuaire: Il référence l'ensemble des services disponibles. Il forme une sorte de cartographie dynamique des services de l'application.
- Service Provider: Le service est une fonction ou un ensemble de fonctionnalités disponibles qui fonctionne de manière autonome sans dépendre d'un contexte particulier.
- Service Requester : C'est le client qui va interagir avec l'Annuaire et le Service Provider.

6.3. Interaction entre les composants

L'annuaire contient la liste de l'ensemble des fonctionnalités disponibles qui sont mises à disposition par le(s) service(s) provider. Le service requester, c'est-à-dire le client, contact l'annuaire pour récupérer la liste des fonctionnalités et s'adresse ensuite au service provider pour accéder à la fonctionnalité elle-même. L'ensemble de la communication est assurée via le protocole SOAP et l'échange est réalisé par WSDL (proche de XML).



6.4. Avantages et inconvénients

Les avantages sont diverses :

- Une obligation d'avoir une modélisation poussée ;
- Forte compatibilité avec les applications orientées objets.

Ainsi que des inconvénients :

- Les applications existantes du SI non compatible SOA ;
- Des performances parfois réduites.