Architecture N-Tier

Architecture Multi-Tier

Traditionnellement une application informatique est un programme exécutable sur une machine qui représente la logique de traitement des données manipulées par l'application.

Ces données peuvent être saisies interactivement via l'interface ou lues depuis un disque.

L'application émet un résultat sous forme de données qui sont, soit affichées, soit enregistrées sur un disque.

Dans ce schéma, les traitements, les données d'entrées, les données de sortie sont sur une seule machine.

Architecture N-Tier

Architecture Multi-Tier

On peut alors séparer l'application en différentes parties :

- La couche interface homme machine

- La couche de traitement

-La couche de gestion des données.

Et toute application possède ces trois parties.

On parle de couches, de niveaux ou de tier (de l'anglais tier : étage).

Architecture N-Tier

Architecture Multi-Tier

Jusqu'au années 90, ces trois couches étaient la plupart du temps sur la même machine.

L'application utilisait les disques de la machine pour lire les données a traiter et stocker les données résultantes du traitement (sauf sur les gros systèmes).

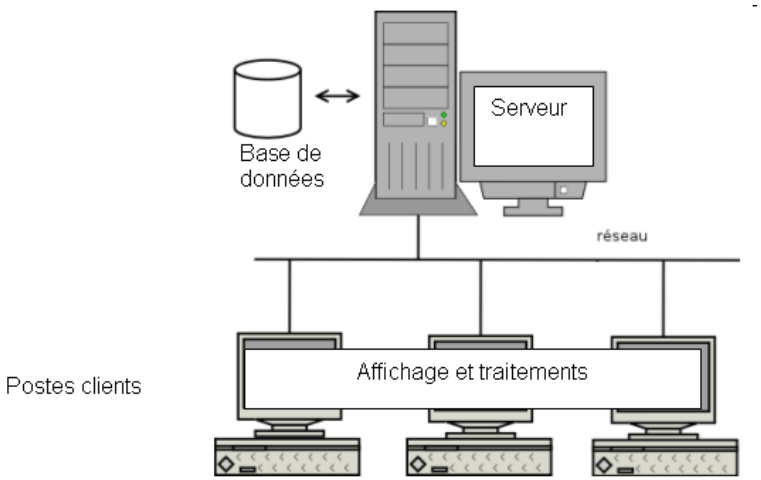
A partir des années 90, vu l'explosion de la volumétrie des données, on a séparé les applications des données. Ces dernières étaient alors stockées dans des bases de données sur d'autres machines.

Les applications accédaient aux données à travers les réseaux sur des machines serveurs de données disposant d'un SGBD.

C'est une architecture 2-Tier, ou encore client serveur.

Architecture N-Tier

Architecture 2-Tier



Ce type d'application permet d'utiliser toute la puissance des ordinateurs présents sur le réseau et permet de fournir à l'utilisateur une interface riche, tout en garantissant la cohérence des données qui restent gérées de façon centralisée.

La gestion des données est prise en charge par un SGBD centralisé sur un serveur dédié. On interroge ce serveur à travers un langage de requête, le plus courant étant SQL.

Le dialogue entre le client et le serveur se résume donc à l'envoi de requêtes et aux données en réponse.

Architecture N-Tier

Architecture 2-Tier

On distingue donc deux parties :

1. Le client
2. Le serveur qui se contente de répondre aux requêtes du client.

Le client provoque l'établissement d'une conversation afin d'obtenir des données ou un résultat de la part du serveur.

Cet échange de messages transite à travers le réseau reliant les deux machines. Il met en œuvre des mécanismes complexes qui sont en général pris en charge par un intergiciel appelé middleware.

Le *middleware* est l'ensemble des couches réseau et services logiciels qui permettent le dialogue entre différents composants d'une application répartie. Ce dialogue s'établie sur des protocoles applicatifs communs, défini par l'API du *middleware*.

Le *middleware* se contente d'unifier, pour les applications mises en jeu, l'accès et la manipulation de l'ensemble des services disponibles sur le réseau, afin de rendre l'utilisation de ces derniers presque transparente.

Architecture N-Tier

Limite de l'architecture 2-Tier

L'architecture client-serveur de première génération présente les inconvénients suivants :

- on ne peut pas soulager la charge du client qui supporte déjà tous les traitements applicatifs ;
- le poste client est fortement sollicité, il devient de plus en plus complexe et nécessite des mises à jour régulière, ce qui est contraignant ;
- le client et le serveur sont assez bruyants, ce qui s'adapte mal à des bandes passantes étroites ; ce qui cantonne ce type d'application à des réseaux locaux ;
-il est difficile de modifier l'architecture initiale, les applications supportent donc mal les fortes montées en charge ;
- la relation forte et étroite entre le programme du client et l'organisation de la partie serveur complique les évolutions de cette dernière ;
- ce type d'architecture est grandement rigidifié par les coûts et la complexité de maintenance.

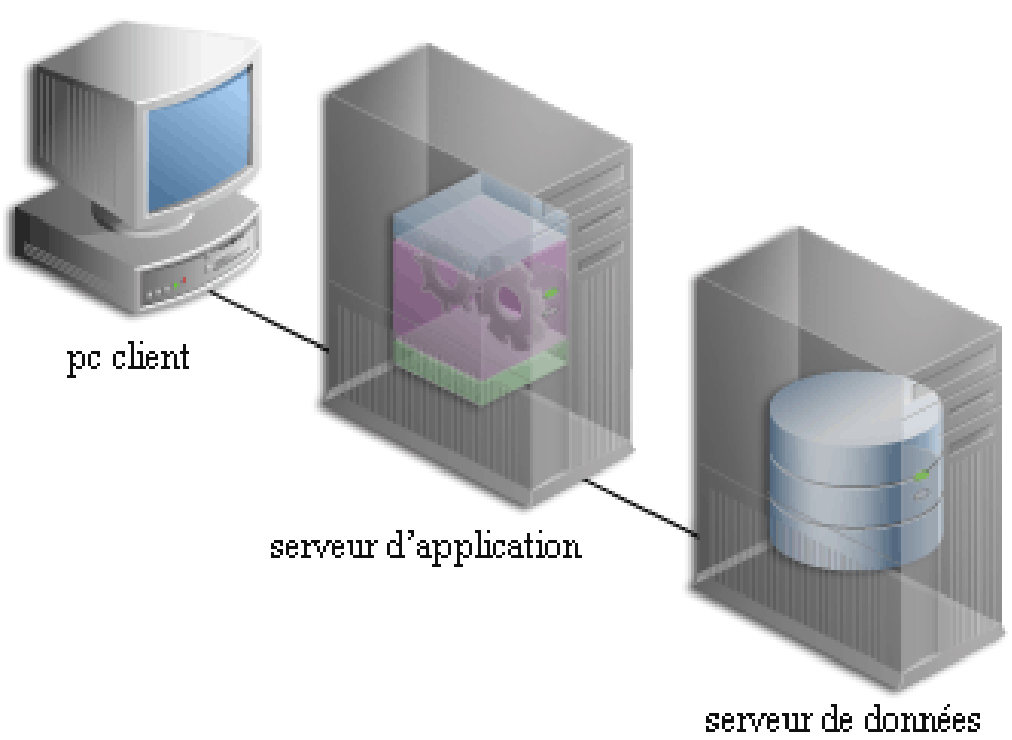
Néanmoins, on peut noter quelques avantages non négligeables de ce type d'architecture :

- elle permet l'utilisation d'une interface utilisateur riche ;
- elle a permis l'appropriation des applications par l'utilisateur ;
- elle a introduit la notion d'interopérabilité.

Architecture N-Tier

Architecture 3-Tiers

Avec l'apparition des technologies Web, il est possible de séparer la couche présentation de la couche applicative (aussi appelée couche métier) :



Architecture N-Tier

Architecture 3-Tier : Couche Présentation

Elle correspond à la partie de l'application visible et interactive avec les utilisateurs.

On parle d'IHM. Elle peut être réalisée par une application graphique ou textuelle, en HTML, en WML (on parle alors de clients légers).

Cette interface peut prendre de multiples facettes sans changer la finalité de l'application.

Dans le cas d'un système de distributeurs de billets, l'automate peut être différent d'une banque à l'autre, mais les fonctionnalités offertes sont similaires et les services identiques (fournir des billets, donner un extrait de compte, etc.).

Toujours dans le secteur bancaire, une même fonctionnalité métier (par exemple, la commande d'un nouveau chéquier) pourra prendre différentes formes de présentation selon qu'elle se déroule sur Internet, sur un distributeur automatique de billets ou sur l'écran d'un chargé de clientèle en agence.

La couche présentation relaie les requêtes de l'utilisateur à destination de la couche métier, et en retour lui présente les informations renvoyées par les traitements de cette couche. Il s'agit donc ici d'un assemblage de services métiers et applicatifs offerts par la couche inférieure.

Architecture N-Tier

Architecture 3-Tier : Couche applicative ou couche métier ou encore serveur d'applications

Elle correspond à la partie fonctionnelle de l'application, celle qui implémente la « logique », et qui décrit les opérations que l'application opère sur les données en fonction des requêtes des utilisateurs, effectuées au travers de la couche présentation.

Les différentes règles de gestion et de contrôle du système sont mises en œuvre dans cette couche.

La couche métier offre des services applicatifs et métier à la couche présentation. Pour fournir ces services, elle s'appuie, le cas échéant, sur les données du système, accessibles au travers des services de la couche inférieure.

En retour, elle renvoie à la couche présentation les résultats qu'elle a calculés.

Architecture N-Tier

Architecture 3-Tier : Couche données

Elle consiste en la partie gérant l'accès aux gisements de données du système.

Ces données peuvent être propres au système, ou gérées par un autre système.

La couche métier n'a pas à s'adapter à ces deux cas, ils sont transparents pour elle, et elle accède aux données de manière uniforme, on dit qu'il y a un faible couplage.

Architecture N-Tier

Architecture 3-Tier : Couche données

Données propres au système

Ces données sont pérennes, car destinées à durer dans le temps, de manière plus ou moins longue, voire définitive.

Les données peuvent être stockées indifféremment dans de simples fichiers texte, fichiers XML, ou encore dans un SGBD

Quel que soit le support de stockage choisi, l'accès aux données doit être le même. Cette abstraction améliore la maintenance du système.

Le DATA ACCESS OBJET (DAO ou design pattern) consiste à représenter les données du système sous la forme d'un modèle objet. Par exemple un objet pourrait représenter un contact ou un rendez-vous.

La représentation du modèle de données objet en base de données (appelée persistance) peut être effectuée à l'aide d'outils tel que HIBERNATE.

Architecture N-Tier

Architecture 3-Tier : Couche données

Données gérées par un autre système

Les données peuvent aussi être gérées de manière externe.

Elles ne sont pas stockées par le système considéré, il s'appuie sur la capacité d'un autre système à fournir ces informations.

Par exemple, une application de pilotage de l'entreprise peut ne pas sauvegarder des données comptables de haut niveau dont elle a besoin, mais les demander à une application de comptabilité.

Celle-ci est indépendante et pré-existante, et on ne se préoccupe pas de savoir comment elle les obtient ou si elle les sauvegarde, on utilise simplement sa capacité à fournir des données à jour.

Architecture N-Tier

Architecture n-Tier

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

L'architecture *n*-tiers qualifie la distribution d'applications entre de multiples services et non la multiplication des niveaux de service : les trois niveaux d'abstraction d'une application sont toujours pris en compte.

Cette distribution est facilitée par l'utilisation de composants métier, spécialisés et indépendants, introduits par les concepts orientés objets.

Elle permet de tirer pleinement partie de la notion de composants métier réutilisables et modulables.

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 et hétérogènes.

Architecture N-Tier

Architecture n-Tier : Technologies

Les technologies :

Le tier client :

- Un navigateur Web

- Un PDA ou SmartPhone

Le tier Applicatif :

- CGI
- ASP
- Java Servlets
- JSP
- PHP, Python
- JavaScripts

Architecture N-Tier

Architecture n-Tier : Technologies

Les technologies :

Le tier données :

- Base de données via SQL, JDBC, .NET
- J2EE
- ERP (Enterprise Resource Planning)
- EAI (Enterprise Application Integration)

Architecture N-Tier

Architecture N-tier : Point de vue matériel

L'architecture présentée en purement logicielle.

En conséquence, chaque étage peut être implémenté sur n'importe quelle machine.

Ainsi, Les trois étages peuvent être réunies sur une seule ou sur plusieurs machines.

L'intérêt de répartir les différents étages sur plusieurs machines permet un accroissement de la sécurité comme nous allons le voir au travers d'un exemple.

Architecture N-Tier

Architecture N-Tier : Matériel exemple Solution de micro-Payement