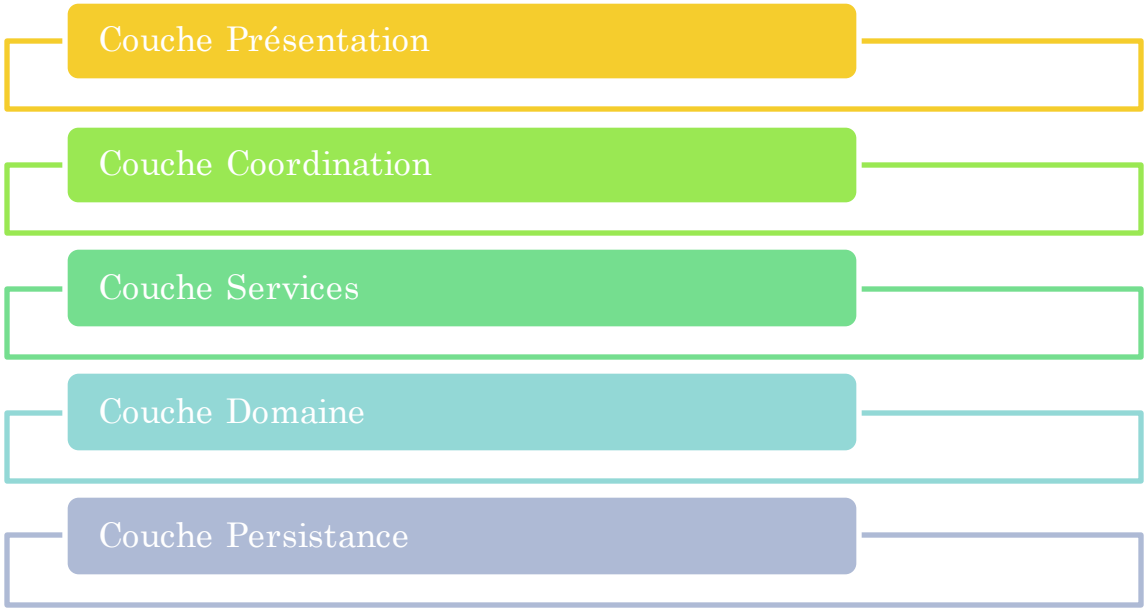
Vue en couches (Layer View)
Vue en niveaux (Tier View)

Structuration des applications

- La structuration du système peut être vue sous différents angles, selon que l'on considère :
  - le découpage « logique » hors de tout contexte d'exécution (machines, OS et réseaux)
  - le découpage « physique » qui prend en compte le contexte d'exécution
- L'architecte structure le système selon plusieurs « vues » :
  - **Vue en couches (Layer View)** : vue « logique » montrant le découpage des fonctions de l'application
    - elle est indépendante des considérations physiques
    - la littérature propose des modèles standards de structuration qui couvrent les types classiques d'applications
    - le modèle de référence est le modèle à 5 couches qui s'applique aux applications munies d'une interface graphique manipulant des données persistantes
  - **Vue en niveaux (Tier View)** : vue « physique » de la structuration de l'application

Structuration des applications EN couche (Suite)



- Chaque couche a ses propres responsabilités et utilise la couche située en dessous d'elle
- En fonction du projet, les architectes enrichissent et élaguent le modèle. La structuration est alors guidée par les contraintes exprimées et existantes

Couche Présentation

- La couche Présentation gère et assure l'affichage de l'interface graphique utilisateur ou les Interfaces Homme-Machine (IHM : fenêtres, pages, composants graphiques…)
- Cette couche intègre principalement :
  - la gestion du domaine visuel
  - l'interaction avec les utilisateurs
  - l'interception des événements utilisateurs et l'appel à la couche Contrôleur
  - la gestion du multi canal (web, voix, mobile, fax)
  - les services de portail (agrégation d'IHM, bouquets de services)
  - les services d'impression (impressions PDF, gestion de templates…)

Couche Présentation

- On distingue trois catégories d'IHM pour les applications interactives :
  - **Client léger** : Dans ce modèle, aucun déploiement n'est réalisé sur le poste client à l'exception d'un navigateur Web. Les différents écrans de l'application sont générés en temps réel côté serveur et téléchargés par le poste client
  - **Client lourd** :Dans ce modèle, l'ensemble des écrans de l'application sont stockés ou générés sur le poste client et doivent avoir été déployés sur celui-ci préalablement à l'exécution. Ce type de client n'impose à priori pas de restriction sur le contenu et l'ergonomie des écrans. En règle générale, une complexité croissante va de pair avec une taille croissante de l'application à télécharger

Couche Présentation

- **Client riche** (Smart Client) :Ce modèle constitue un compromis entre le client léger et le client lourd. Il présente une ergonomie comparable à celle d'un client lourd tout en limitant les problématiques de déploiement inhérentes à ce dernier. Outils: Adobe Flex, Microsoft Silverlight, Google Web Toolkit qui permettent d'exécuter directement le code dans le navigateur
  - **Adobe Flex/Flash** permet la création de graphiques vectoriels et de bitmap animés par un langage script appelé ActionScript, et la diffusion de flux (*stream*) bi-directionnels audio et vidéo.
  - ***Silverlight*** est un plugin pour navigateur Web multiplate-forme (Windows et Mac OS, Linux *via* le projet Moonlight), qui permet de développer des applications Web riches dans un moteur de rendu vectoriel. Il fonctionne de façon similaire à Adobe Flash dont il se veut une alternative. Techniquement, Silverlight est l'équivalent de la CLR de Microsoft mais pour les navigateurs Web. Elle permet aux développeurs d'utiliser des outils de développement et les langages Microsoft .NET en place et lieu de JavaScript .

Couche Coordination

- La couche Coordination/Contrôleur gère :
  - le contrôle de la cinématique des écrans
  - l'invocation des appels de services
  - les erreurs et les exceptions qui peuvent être levées
  - les sessions / espace de travail utilisateur
  - les habilitations et les droits d'accès
- L'architecture applicative de gestion des interactions utilisateur est généralement mise en œuvre autour du modèle de conception MVC (Modèle-Vue-Contrôleur)

Couche Services

- La couche Services correspond aux traitements qu'effectue l'application.
- Cette couche doit :
  - implémenter la logique métier
  - gérer la sécurité applicative
  - gérer les transactions étendues (processus, compensation)
  - gérer l'intégrité transactionnelle (transactions locales et distribuées)
  - gérer les appels aux objets métiers de la couche Domaine
- Elle gère les services métiers qui enchaînent des règles métiers (processus métier) et des appels à la couche Domaine
  - Exemple : virement de compte à compte

NOTION SUR TRANSACTION

- Une transaction est une suite d'opérations effectuées comme une seule unité logique de travail. Une unité logique de travail doit posséder quatre propriétés appelées propriétés ACID (Atomicité, Cohérence, Isolation et Durabilité), pour être considérée comme une transaction :
  - **Atomicité** Une transaction doit être une unité de travail indivisible ; soit toutes les modifications de données sont effectuées, soit aucune ne l'est.
  - **Cohérence** Lorsqu'elle est terminée, une transaction doit laisser les données dans un état cohérent. Dans une base de données relationnelle, toutes les règles doivent être appliquées aux modifications apportées par la transaction, afin de conserver l'intégrité de toutes les données.
  - **Isolement** Les modifications effectuées par des transactions concurrentes doivent être isolées transaction par transaction. Une transaction reconnaît les données dans l'état où elles se trouvaient avant d'être modifiées par une transaction simultanée, ou les reconnaît une fois que la deuxième transaction est terminée, mais ne reconnaît jamais un état intermédiaire. Cette propriété est nommée mise en série, car elle permet de recharger les données de départ et de répéter une suite de transactions dont le résultat sur les données sera identique à celui des transactions d'origine.
  - **Durabilité** Lorsqu'une transaction est terminée, ses effets sur le système sont permanents. Les modifications sont conservées même en cas de défaillance du système.
- **Traitement transactionnel:** La technologie garantissant un échange équilibré et prévisible s'appelle le traitement transactionnel. Les transactions garantissent que les ressources orientées données ne font pas l'objet d'une mise à jour définitive tant que toutes les opérations de l'unité transactionnelle n'ont pas abouti. Grâce à la combinaison d'un jeu d'opérations connexes dans une unité qui a entièrement réussi ou entièrement échoué, vous pouvez simplifier la récupération des erreurs et accroître la fiabilité de votre application.
- Les systèmes de traitement transactionnel sont constitués de matériel et de logiciel informatiques hébergeant une application orientée transaction qui procède aux transactions habituelles nécessaires au traitement des affaires. Les systèmes qui gèrent la saisie de bons de commande, les réservations aériennes, les salaires, les dossiers du personnel, la fabrication et l'expédition en sont des exemples.

- Les transactions distribuées sont réparties sur plusieurs serveurs nommés gestionnaires de ressources. La gestion de la transaction doit être coordonnée entre les gestionnaires de ressources par un composant du serveur nommé gestionnaire de transactions.

Couche Domaine

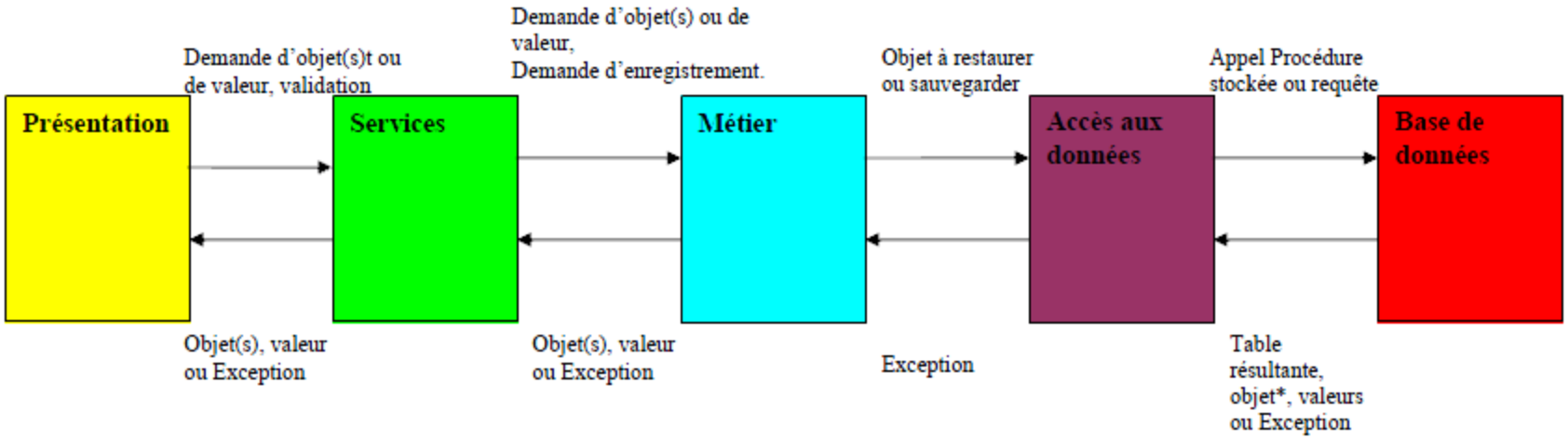
- La couche Domaine\Métier gère l'intégrité du modèle « métiers ». Cette couche intègre principalement:
  - la gestion des règles métiers « élémentaires»
  - la fourniture des moyens d'accès à l'information (SGBDR, Mainframe...)
  - le respect des propriétés transactionnelles de la couche persistance
- La couche Domaine recense les objets métiers manipulées par l'application
- La couche Domaine est concentrée sur le métier de l'entreprise, commun à toutes les applications
  - Elle contient les Objets Métier qui implémentent le modèle métier. Ils offrent à la couche Services une abstraction pour la manipulation unitaire ou multiple des occurrences de données, ainsi que la mise en œuvre des règles de gestion associées.
  - Exemple bancaire : l'opération de virement de compte à compte
    - l'opération de virement de compte à compte est un élément de la couche Services
    - le compte bancaire et le client et leurs règles de gestion respectives, se situent dans la couche Domaine.

Couche Persistance

- La couche Persistance intègre principalement :
  - la persistance complète du Système d'Informations (données structurées ou non structurées, gérées entre autres via un SGBDR, annuaire LDAP, transaction CICS, ...)
  - la fourniture des services de stockage des données, moteurs relationnels, bases objets, bases XML…
  - la création, la modification, la suppression d'occurrences des objets métiers
- Elle contient un niveau d'abstraction de données les DAO (Data Access Object) qui prennent en charge l'accès à la source de données (SGBDR, fichiers XML, …).
- La couche Persistance offre les fonctionnalités de base qui permettent :
  - de créer, rechercher, modifier et supprimer des composants objets métiers dans le respect des propriétés transactionnelles classiques
  - d'utiliser le mécanisme de projection objet vers relationnel (mapping Objet / Relationnel) qui consiste en la transformation de la représentation des données en une représentation objet

Services entre couche

Services échangés entre couches.



* dans le cas d'une base de données objets.

COUCHE TRANSVERSE

Couche Sécurité

La sécurité n'est pas une couche isolée, mais transverse aux autres couches:

- authentification des utilisateurs et contrôle des habilitations au niveau des services IHM, sécurisation des traitements (authentification, habilitations grosse maille et habilitations fines…)
- sécurisation des échanges, sécurisation des données…

Services Techniques (Core Services)

Indépendamment des fonctionnalités des applications et de leur découpage en couches logicielles, on retrouve des composants et services de base communs (Core Services) et transverses à l'ensemble des couches :

- gestion des traces
- statistiques et logs
- gestion des erreurs
- gestion des propriétés de configuration
- gestion des fichiers de messages (internationalisation, messages d'erreurs)
- monitoring…

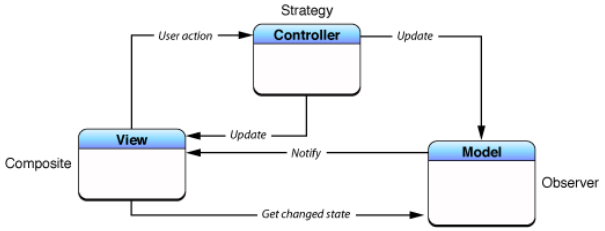
Couche supplémentaire

- Les architectes peuvent être amenés à effectuer des découpages plus fins lorsque les contraintes deviennent plus industrielles
- Un tel découpage s'explique par :
  - La séparation des traitements dans une couche Service a pour objectif de permettre leur réutilisation entre des processus « automatiques » (arrivée de messages en provenance de systèmes externes) et des opérations manuelles effectuées via les IHMs
  - Une couche Domaine est pertinente dans le cas où les traitements à effectuer sont nombreux, portent sur des entités métiers identifiées, récurrentes et ont une importante durée de vie
  - Le recours à une couche **Echanges (comprenant les couches Connectivité, Transformation et Routage) permet d'intégrer des sources d'informations multiples et hétérogènes, en les transformant en** un ensemble plus réduit de formats pivots pour les router vers les traitements adéquats. Elle propose des services d'échanges entre traitements (échanges synchrones, asynchrones), entre système de persistance (synchronisation de référentiels, ETL, ...), services de garantie de livraison de message, Message Broker (Transformation, Routage, DataFlow), services de gestion de transactions étendues (processus, compensation)

MVC

C'est un ensemble de modèles:

- Modèle utilise l' Observateur afin de garder les vues à jour par rapport aux derniers changements d'état.
- La vue et le contrôleur mettre en œuvre le pattern Stratégie. Le contrôleur a le comportement de la vue et peut être facilement échangé avec un autre contrôleur si l'on veut un comportement différent.
- La Vue utilise aussi un modèle interne pour gérer les boutons des fenêtres et des d'autres composants de l'écran: le Pattern Composite.
- Exemple implémentation: http://baptiste-wicht.developpez.com/tutoriels/conception/mvc/#LIII



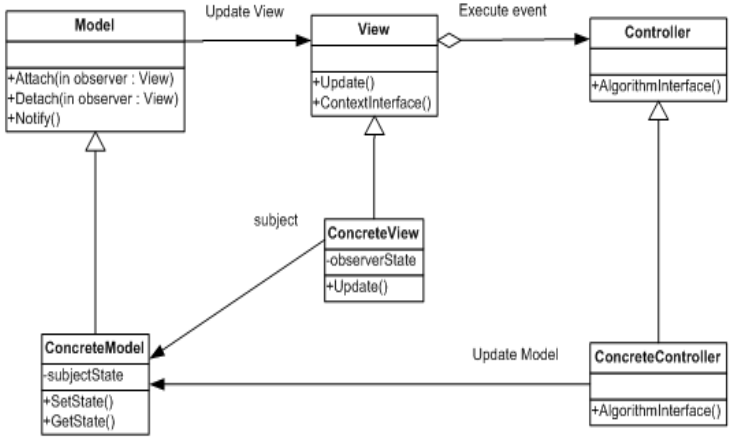


Figure 3: MVC

Vue en niveaux (Tier View)

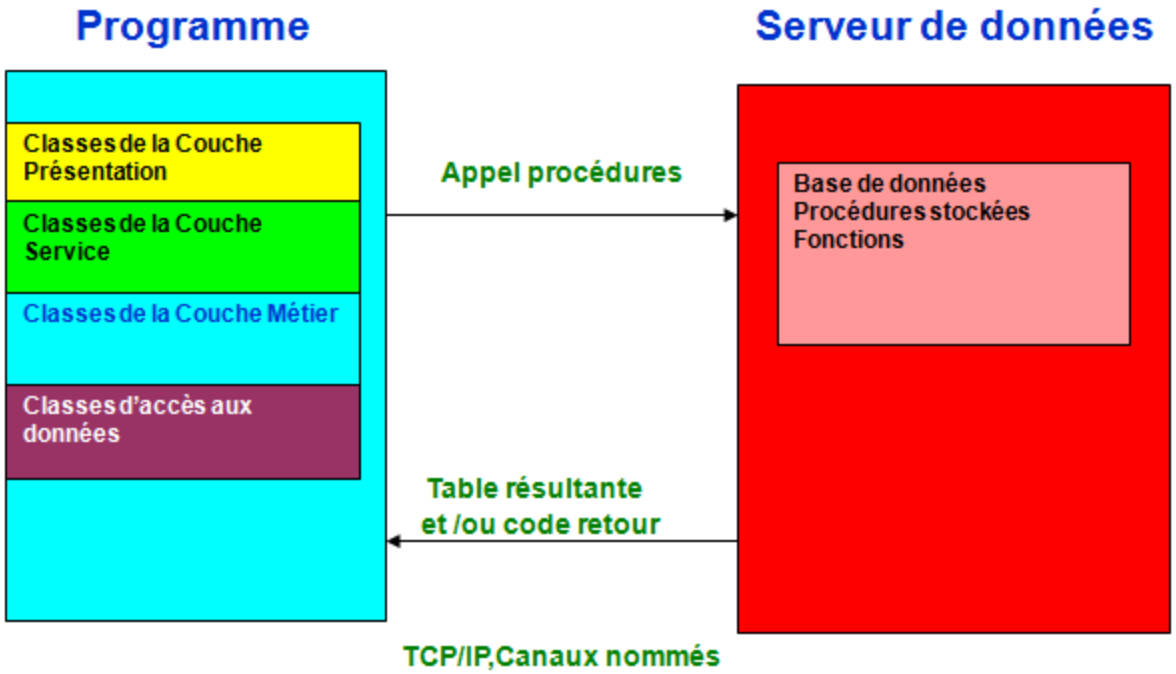
- La vue en niveaux (la tier view) donne une vision plus « physique » de la structuration de l'application. Les niveaux (ou tiers) peuvent être répartis physiquement sur différents composants matériels.
- On identifie un changement de « niveau » dès qu'un module logiciel doit passer par un intermédiaire de communication (middleware) pour en invoquer un autre. Si l'utilisation du middleware est en général transparente pour les développeurs, elle n'est pas sans impact sur l'architecture. L'architecte doit donc maîtriser les caractéristiques (client/serveur, publication/abonnement, sécurité, support du transactionnel, …) et en justifier l'usage.
- Des modèles standards de répartition de niveaux ont été définis dans les projets par l'industrie au fur et à mesure de l'évolution des capacités matérielles et des besoins

1 TIERS

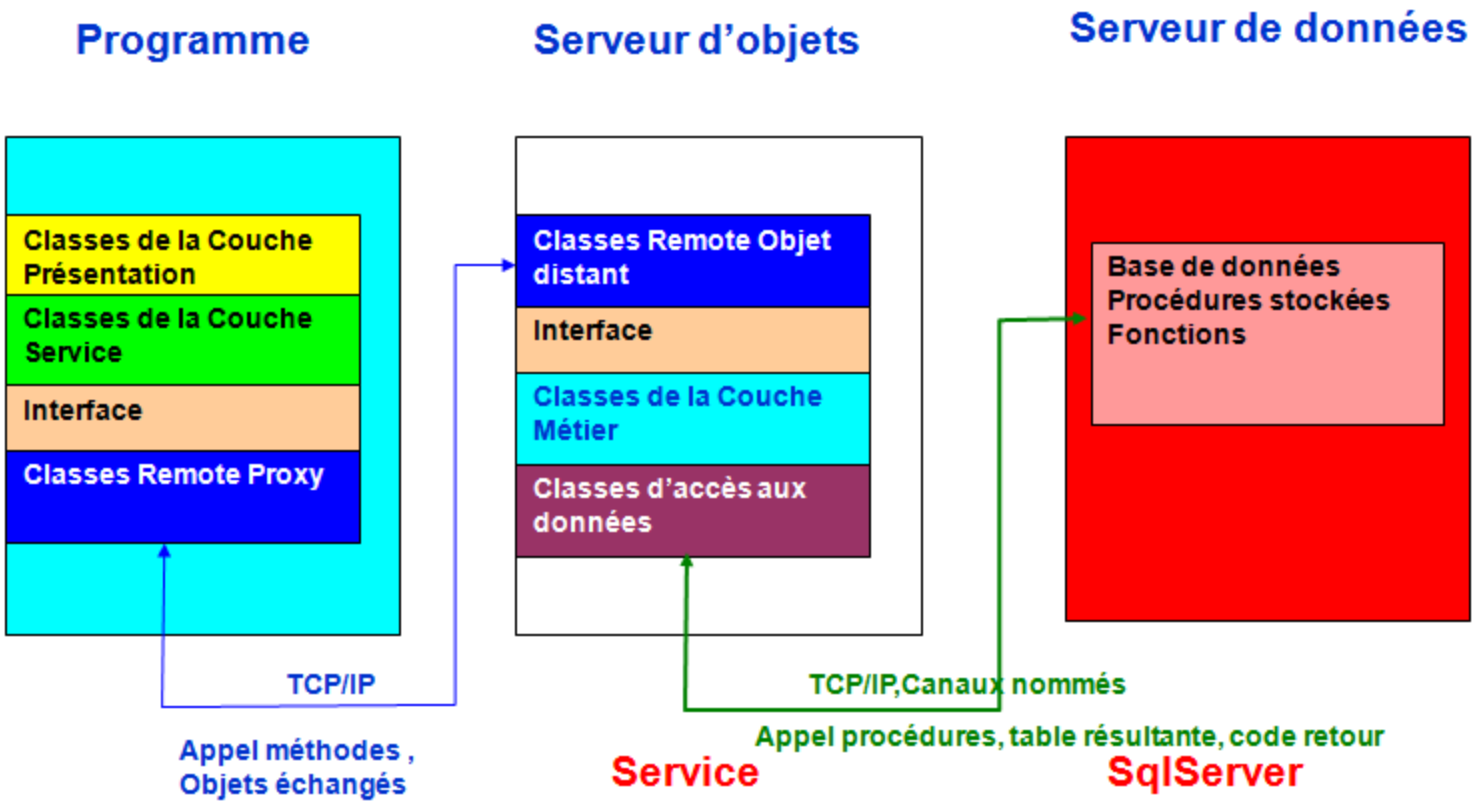
- Le modèle à 1 niveau (ou tiers) correspond à un executable dans lequel s'exécutent toutes les couches, de la présentation à la persistance.
- C'est l'exemple de l'application utilisée en monoposte ou sur un réseau de serveurs de fichiers, ainsi que de l'application sur système central.
- Les données sont stockées sur un fichier local ou partagées sur un serveur de fichier

2 TIERS

- Le modèle à 2 niveaux (ou tiers), encore appelé « client/serveur première génération », repose sur l'utilisation de moteurs de bases de données relationnelles.
- Ces moteurs permettent de distribuer la gestion de la persistance sur un serveur ce qui permet de mieux répondre au besoin d'accès concurrents et de supporter d'importants volumes, de gagner en flexibilité et de se passer des onéreux systèmes centraux
- L'application d'entreprise peut ainsi être accédée depuis un ordinateur personnel avec des standards de présentation moderne

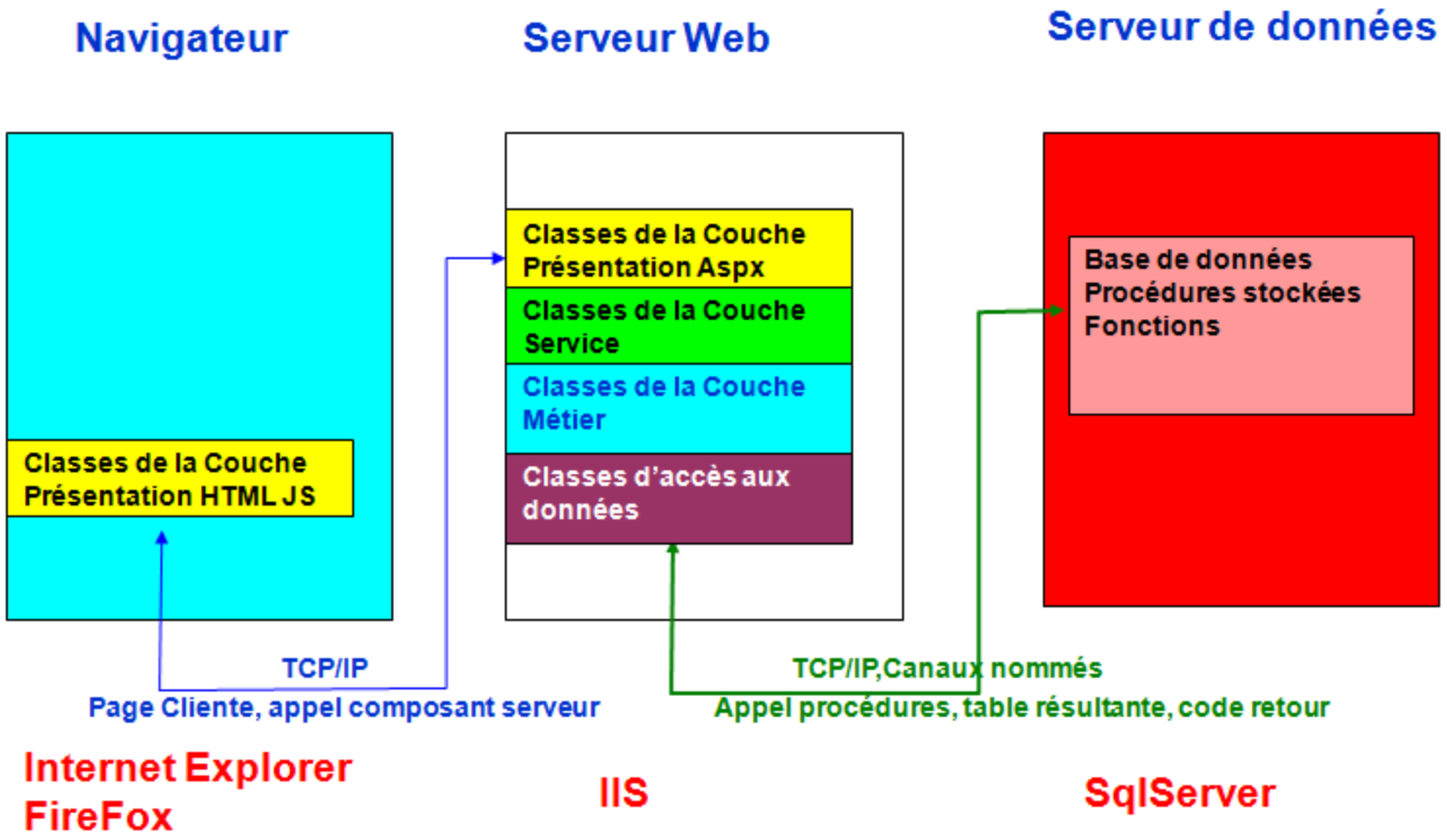


3 TIERS CLIENT SERVEUR D'OBJETS

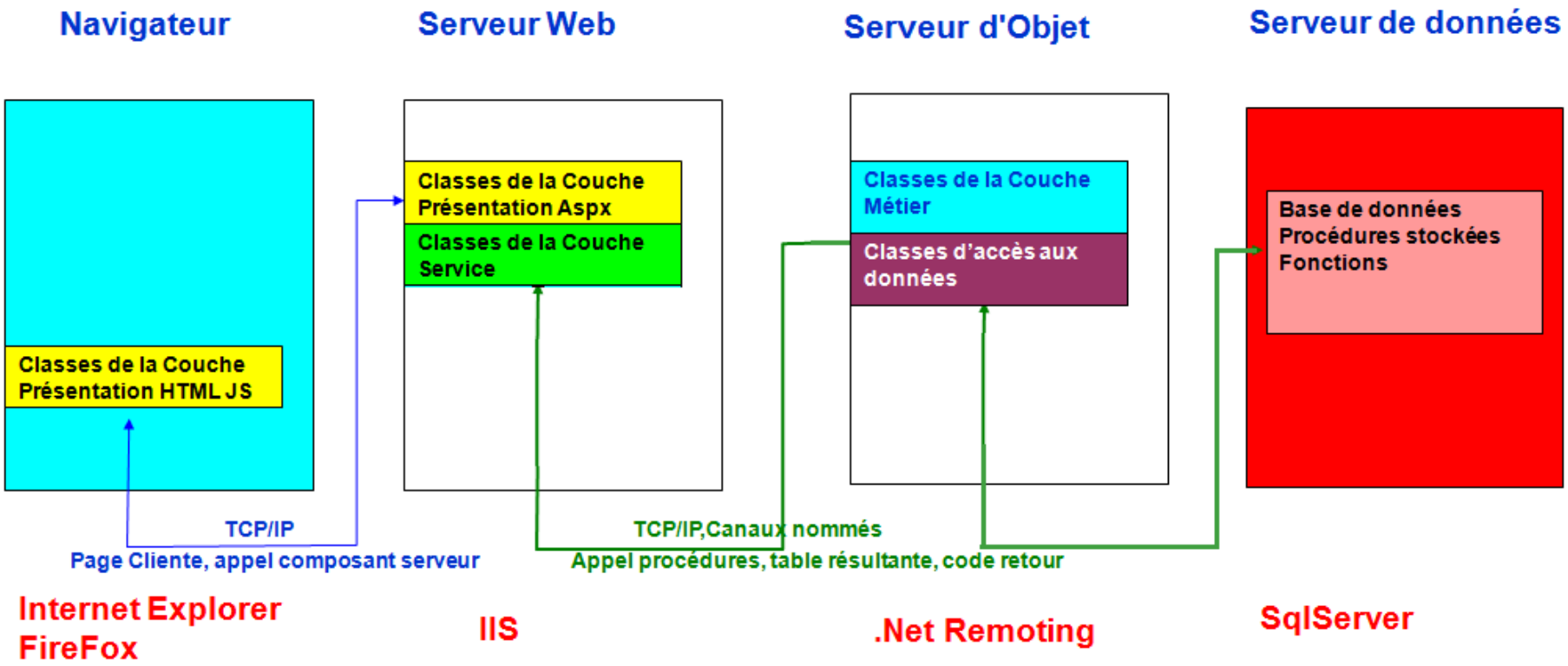


1 TIERS WEB STATIQUE

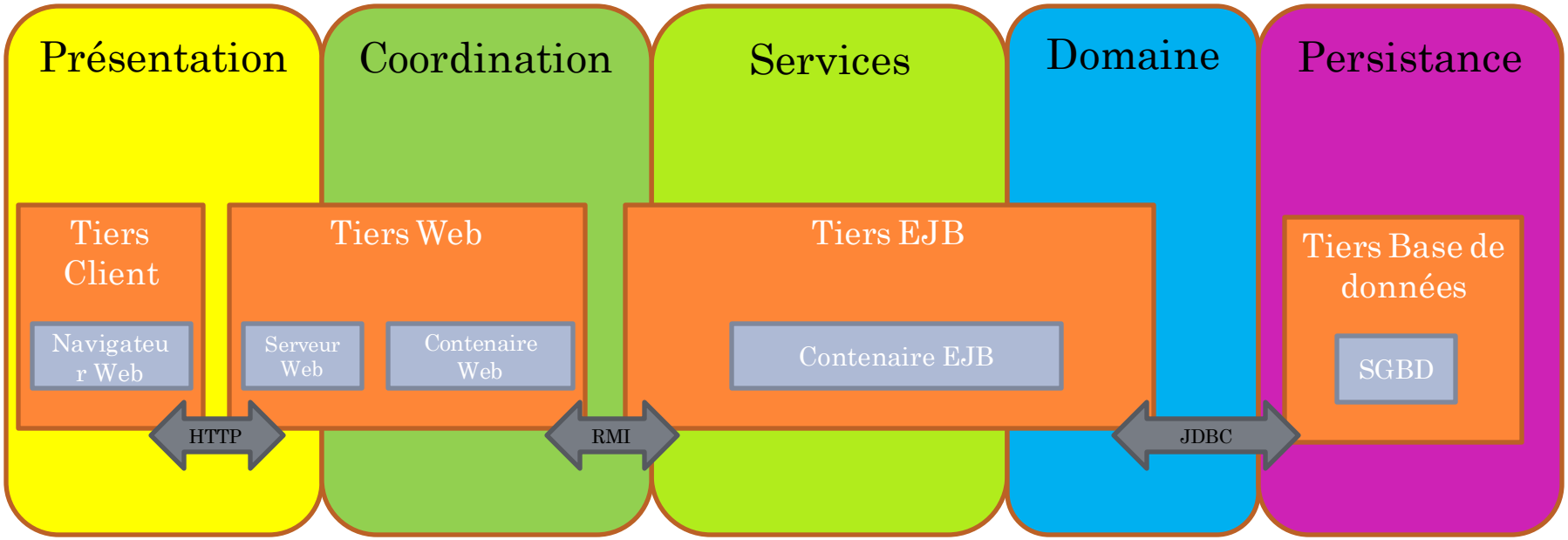
3 TIERS WEB DYNAMIQUE



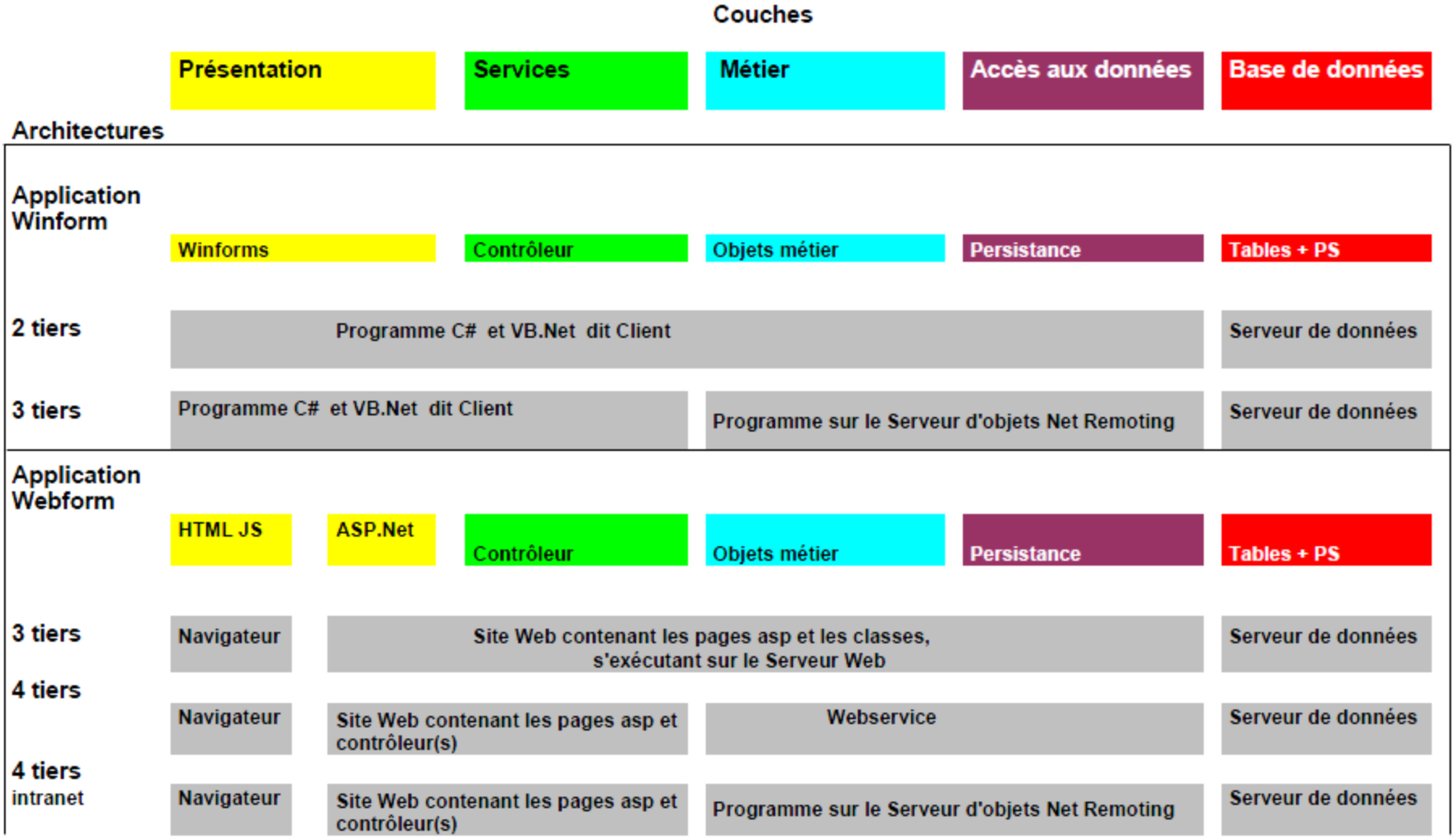
4 TIERS WEB DYNAMIQUE



N TIERS WEB DYNAMIQUE DANS JEE

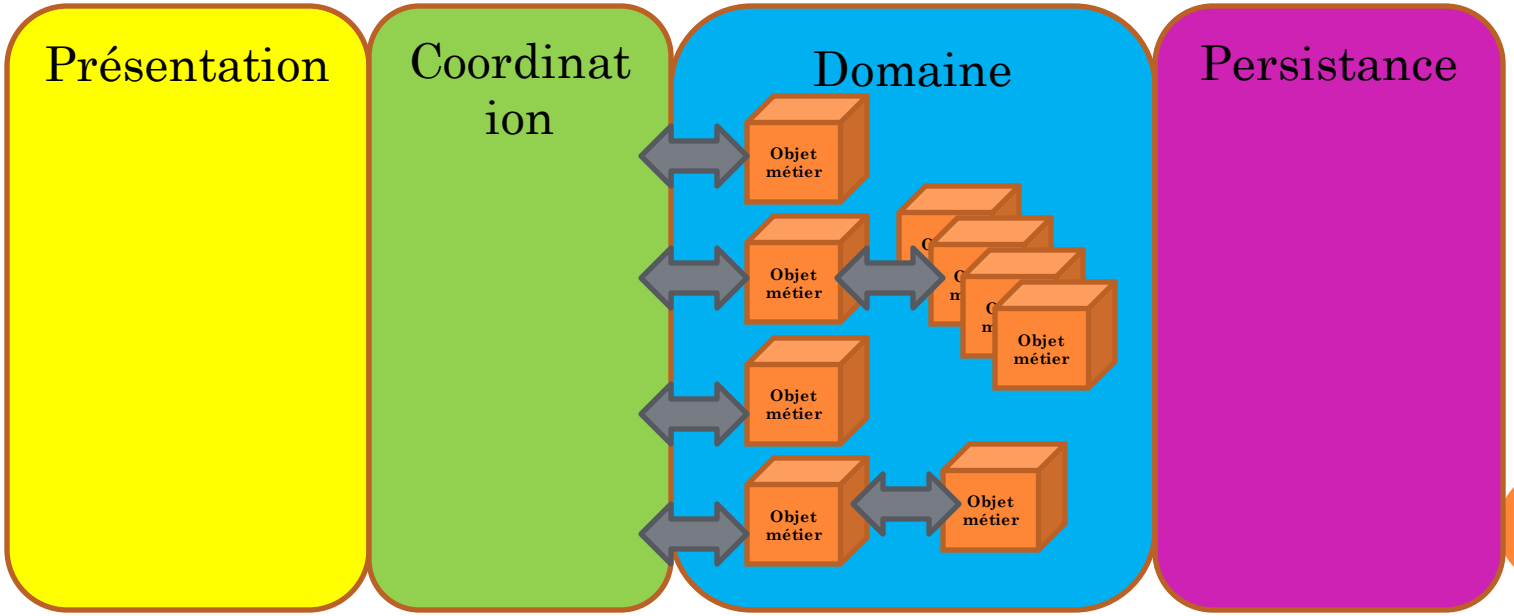


N TIER DANS DOT.NET



L'ARCHITECTURE ORIENTÉE OBJETS (OOA)

- Dans une architecture orientée manipulation d'objets, on remarque tout de suite le nombre de liens entre la couche Coordination et les objets métiers de la couche Domaine.
- Le code client doit traiter directement avec le modèle objet de la couche Domaine, ce qui a pour conséquence de lier celle-ci très fortement à un modèle spécifique et requiert un nombre d'appels important entre les deux couches.
- La multiplication des appels entre couches pose problème lors de la mise à disposition à distance des objets métiers. De plus le nombre d'objets à manipuler réduit l'indépendance entre couches et complexifie la prise en main de la couche métier

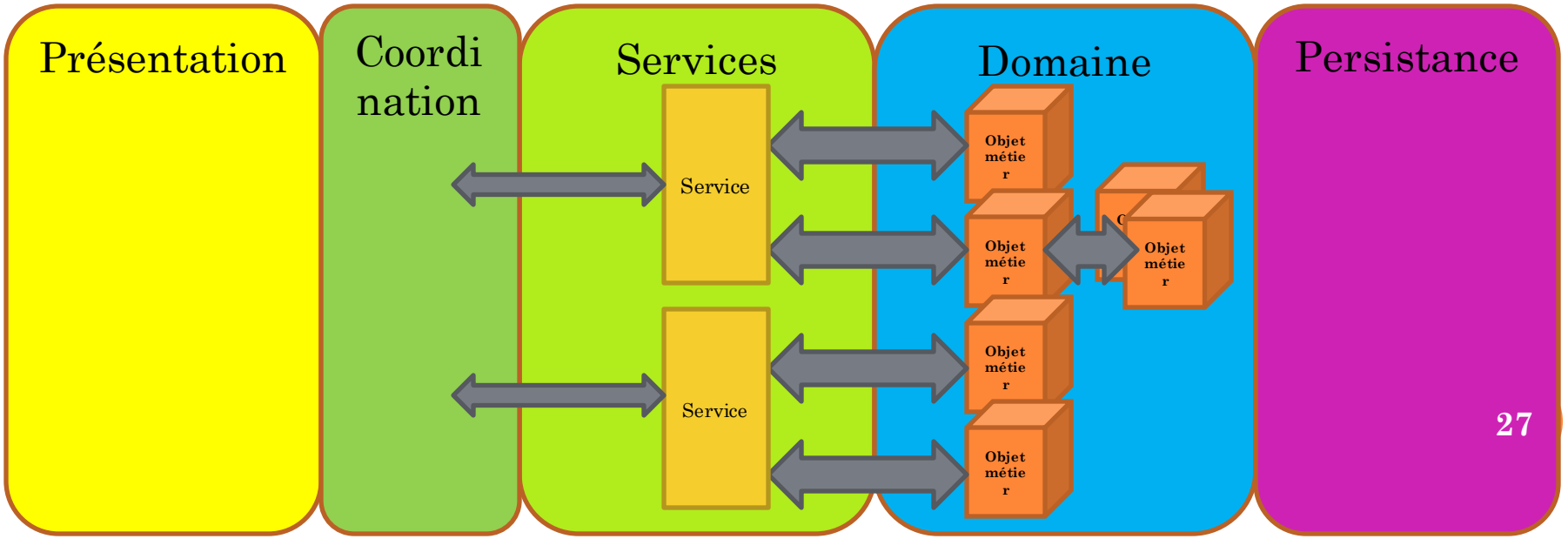


L'architecture Orientée Services (SOA)

- L'architecture SOA consiste à traiter toute application du système d'information comme un fournisseur de services. Et ces services doivent être réutilisables.
- Le service est l'unité atomique d'une SOA. Une application est un ensemble de services qui dialoguent entre eux par des messages.
- Le couplage entre services est un couplage faible et les communications peuvent être synchrones ou asynchrones.
- Le service peut :
  - être codé dans n'importe quel langage
  - s'exécuter sur n'importe quelle plate-forme (matérielle et logicielle).
- Le service doit :
  - offrir un ensemble d'opérations dont les interfaces sont publiées ;
  - être autonome (disposer de toutes les informations nécessaires à son exécution : pas de notion d'état) ;
  - respecter un ensemble de contrats (règles de fonctionnement),
  - correspondre aux processus métier et fonctions mutualisables au niveau de l'entreprise afin d'aligner l'informatique aux changements des décisions stratégiques et tactiques.

L'architecture Orientée Services

- Pour une architecture SOA, un niveau supplémentaire est introduit sous la forme de la couche Services.
- La couche Coordination ne manipule plus directement les objets métiers, mais passe par des appels de services.
- Les services agissent comme des « boites noires » faisant abstraction de la complexité du modèle objet, présentant un ensemble de fonctionnalités restreints et permettant de réduire les échanges entre les couches.



L'architecture Orientée Services

- Un service doit pouvoir être utilisé par exemple pour un traitement batch ou encore pour un traitement TP
  - **TP** (Transaction Processing): il s'agit d' un traitement qui s'effectue **en transactionnel**, c'est-à-dire en temps réel (synchrone).
  - **Batch**: il s'agit d' un traitement qui s'effectue par lots, en réponse différée (asynchrone).
- Quel implémentation?
  - Lorsque les couches se trouvent sur des machines physiquement distinctes, des mécanismes tels que le remoting ou les Services Web peuvent être mis en œuvre.
  - Lorsque les couches d'une application se trouvent toutes sur la même machine, il convient d'optimiser la performance en privilégiant des appels directs entre les couches

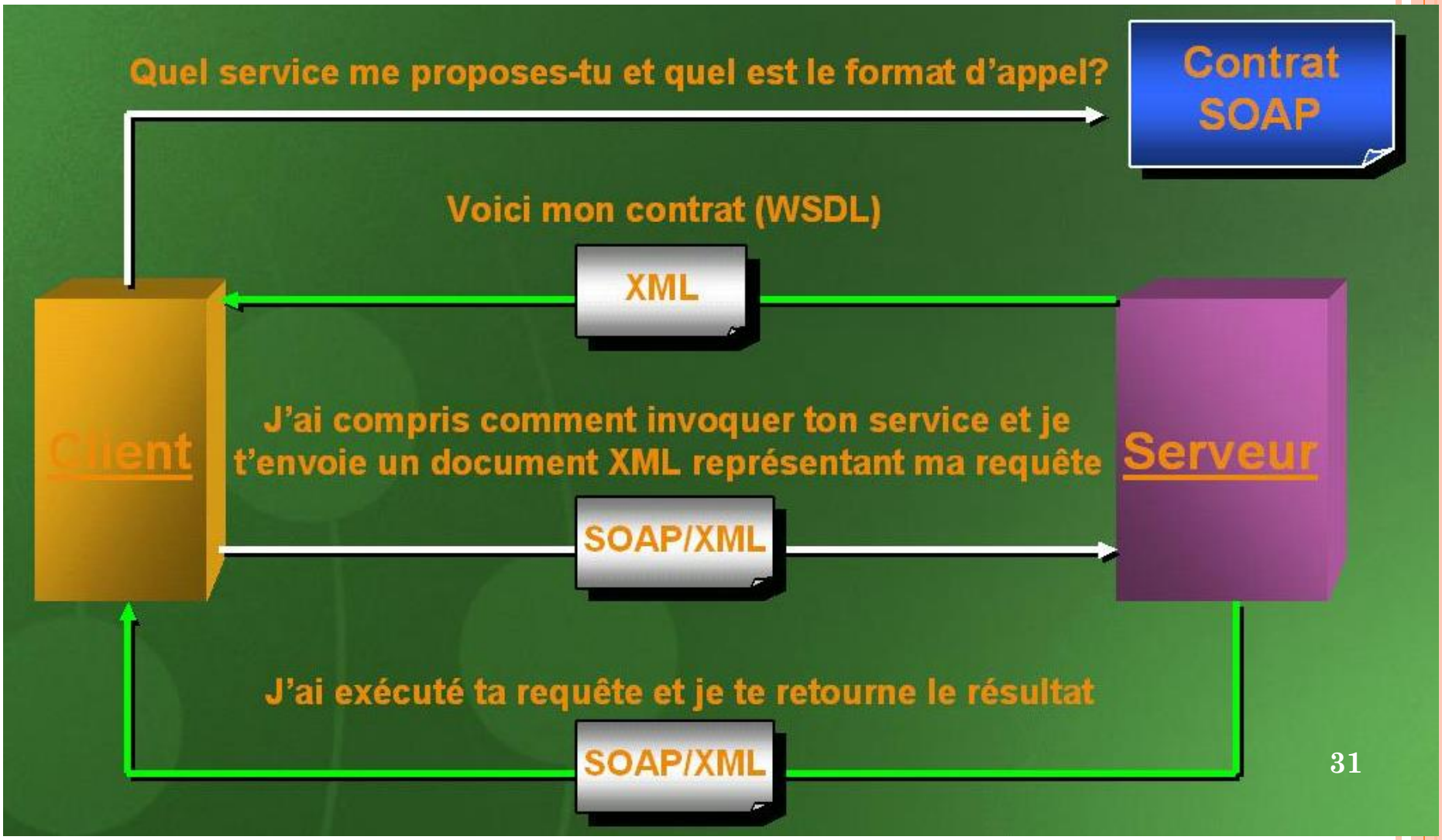
L'architecture Orientée Services

- Généralement, une architecture SOA peut être construire sans utiliser XML ni les services Web, mais avec des formats de type CVS, ou des technologies comme Corba ou COM/DCOM, mais XML offre certainement une plus grande ouverture.

Web service

- Un **service web** est un programme informatique permettant la communication et l'échange de données entre applications et systèmes hétérogènes dans des environnements distribués: il permet aux applications de dialoguer à distance via Internet indépendamment des plates-formes et des langages sur lesquelles elles reposent.

WEB SERVICE : WS



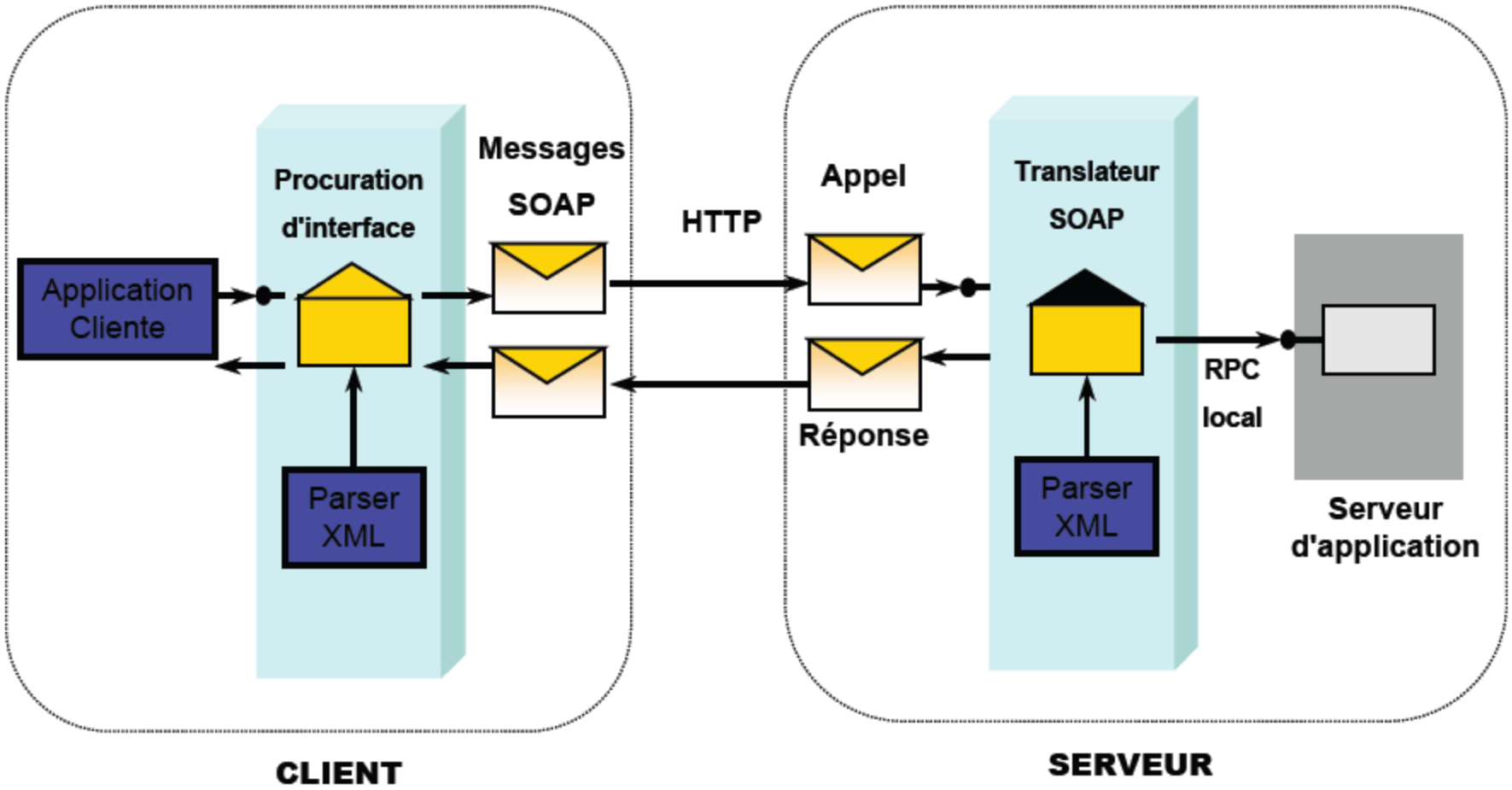
Web service WS: les acteurs

- Sur un serveur, le Service Registry est l'annuaire des services publiés par les providers (UDDI)
- Sur le serveur, le Service Provider: application s'exécutant sur un serveur et comportant un module logiciel accessible en XML
- Sur le client, le Service Requester: application cliente se liant à un service et invoquant ses fonctions par des messages XML (REST, XML-RPC,SOAP)

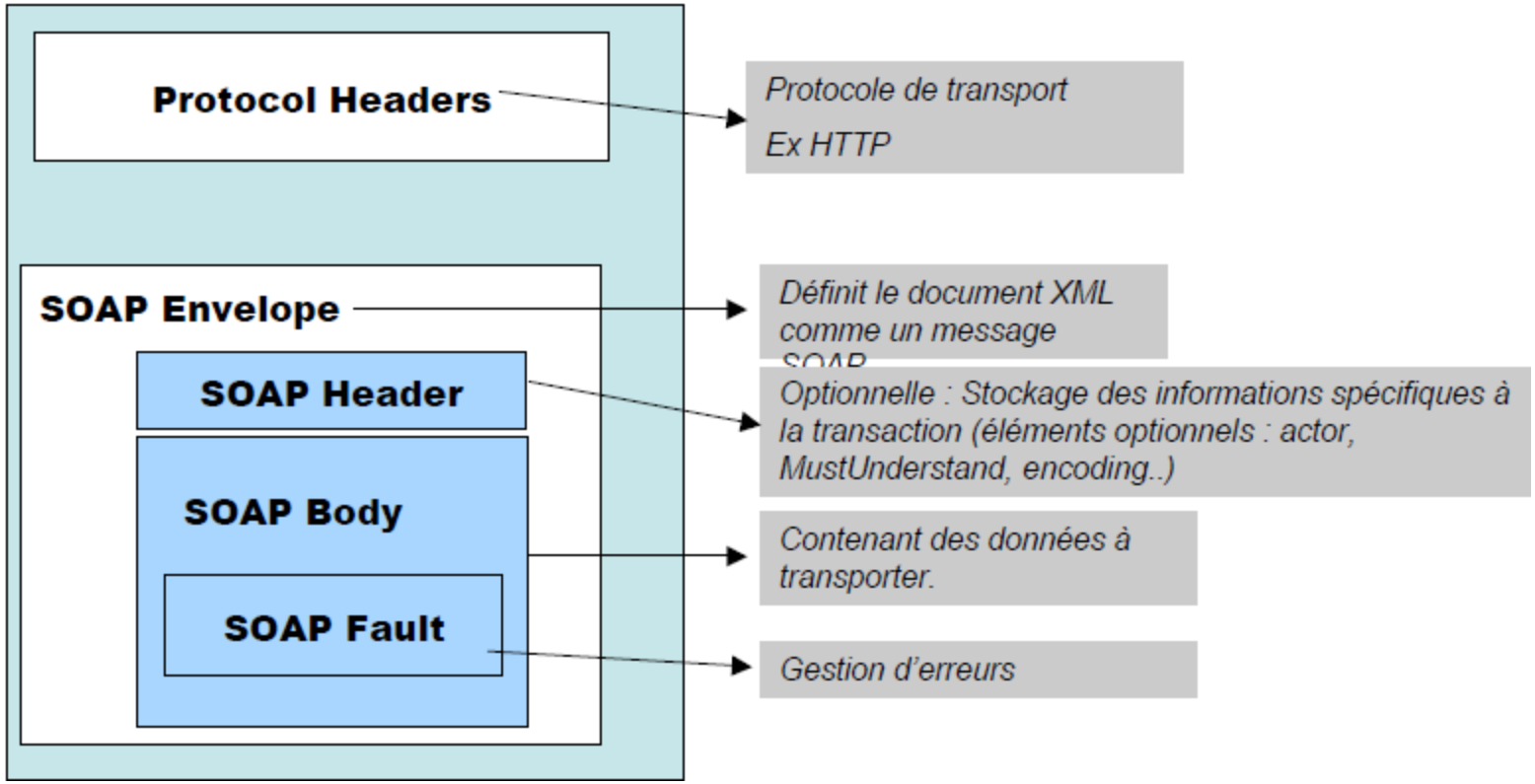
Web service WS

- WSDL (Web Services Description Language) donne la description au format XML des Web Services en précisant les méthodes pouvant être invoquées, leur signature et le point d'accès (URL, port, etc..).
- UDDI (Universal Description, Discovery and Integration) normalise une solution d'annuaire distribué de Web Services, permettant à la fois la publication et l'exploration. UDDI se comporte lui-même comme un Web service dont les méthodes sont appelées via le protocole SOAP. Il s'agit d'un annuaire permettant d'enregistrer de rechercher des service web.
- SOAP (Simple Object Access Protocol) : Protocole de communication en service Web par échange de message XML.

Web service WS: SOAP



Web service WS: Structure d'un message SOAP



Signature de la Méthode

Int doubleAnInteger (int numberToDouble);

Requête

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<SOAP-ENV:Envelope SOAP-
    ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/1999/XMLSchema">
<SOAP-ENV:Body>
<ns1:doubleAnInteger xmlns:ns1="urn:MySoapServices">
<param1 xsi:type="xsd:int">123</param1>
</ns1:doubleAnInteger>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Réponse

<?xml version="1.0" encoding="UTF-8" ?> <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance" xmlns:xsd="http://www.w3.org/1999/XMLSchema">

<SOAP-ENV:Body>

<ns1:doubleAnIntegerResponse xmlns:ns1="urn:MySoapServices" SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">

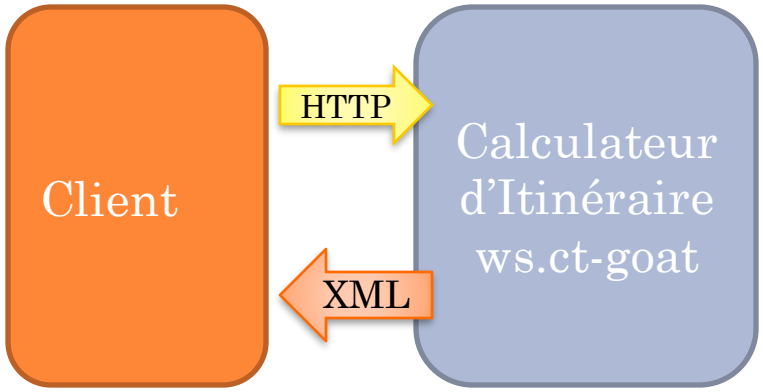
<return xsi:type="xsd:int">246</return>

</ns1:doubleAnIntegerResponse>

</SOAP-ENV:Body>

</SOAP-ENV:Envelope>

Web service REST (Representation State Transfer)



- La consommation d'un WebService REST revient à appeler une simple URL en HTTP.
- Chaque 'méthode' ou 'service' est attaché à une URL
- Le serveur renvoie sa réponse, la plupart du temps en XML

Web service REST (Representation State Transfer)

- Exemple : Récupération des Informations d'une commune par une requête http en mode GET

Requête

```
http://ws.ct-goat.com/getCityInfos.asp ?uID=xxxxxxxxxxxxx&comID=562
```

Réponse

```
<RETURN>
  <ERROR>
    <NUMBER>[numéro d'erreur]</NUMBER>
    <DESCRIPTION>[description de l'erreur]</DESCRIPTION>
  </ERROR>
  <RESULT>
    <ROWS>
      <ROWCOUNT>1</ROWCOUNT>
      <ROW>
        <COM_ID>[ID de la commune]</COM_ID>
        <COM_NAME>[nom de la commune]</COM_NAME>
        <COM_COMINSEE>[code INSEE de la commune]</COM_COMINSEE>
        <COM_PAE>[Id des points d'arrêts principaux de la commune]</COM_COMINSEE>
      </ROW>
    </ROWS>
  </RESULT>
</RETURN>
```