Technologies **W**eb et **A**rchitectures de **S**ervices pour les **O**rganisations

Architecture Applicative $\Rightarrow$ Architecture Logicielle

—

Module IF-4-S1-EC-WASO – Cours

Yann Gripay

28 Septembre 2023

Pour les Organisations ?

Organisations

- Entreprises
- Associations
- Administrations

Système d'Information pour une Organisation

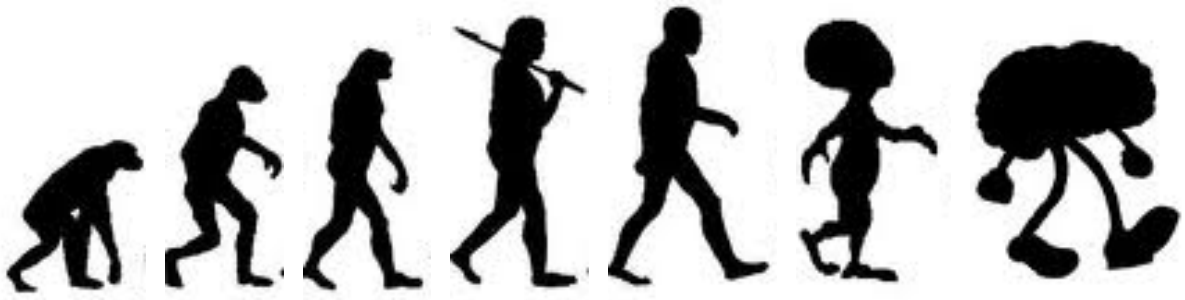
Besoins d'une Application

- Multi-Utilisateur
- Multi-Site
  - Ville / Pays / Monde / ...
- Orienté Métier
  - Et non orienté BD...
  - Gestion Métier
  - Collaboratif entre Utilisateurs

Besoins du Système d'Information

- Multi-Application
- Orienté Métier
  - Gestion Métier unifiée
  - Collaboratif entre Applications

Évolution des Architectures de Systèmes d'Information



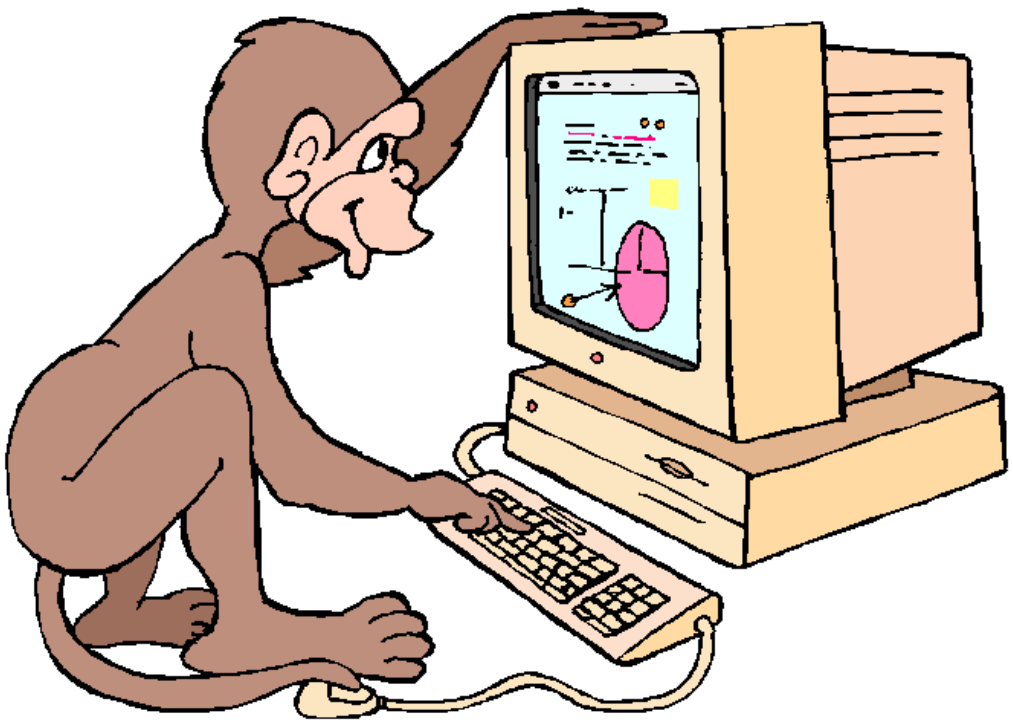
Évolution

- Monobloc : application "lourde" / données locales
- Client/Serveur : client lourd / données distantes
- Web : client léger (navigateur) / serveur Web + données
- N-Tier
  - client lourd / serveur d'application / serveur de données
  - client léger / serveur Web / serveur d'application / ...
- Cloud : client léger / SaaS (*aaS) / Data in the Cloud
- ...

Évolution des Architectures de Systèmes d'Information

Mais...

- Une évolution n'efface pas les autres
- Données de Gestion dans un Tableur ?
  - Excel / LibreOffice Calc ⇒ Application Monobloc



Évolution des Architectures de Systèmes d'Information

Plus de répartition...

- Diffusion simple des applications utilisateurs
  - Clients légers : navigateurs Web standards
  - Clients lourds : seulement l'IHM
- Performance (serveurs puissants, répartition de charge)
- Collaboration / Ouverture vers l'extérieur

...et plus de centralisation !

- Maintenabilité
  - IHM (si client léger)
  - Logique métier
  - Données (stockage, mise à jour, *etc.*)
- Sécurité renforcée
  - Authentification centralisée
  - Données : accessibilité, intégrité, confidentialité

Focus du Cours

Architecture Applicative ⇒ Architecture Logicielle

- Architecture Logicielle de Services
- Technologie de Services Web
- Architecture Technique (Serveurs)

Notions sous-jacentes

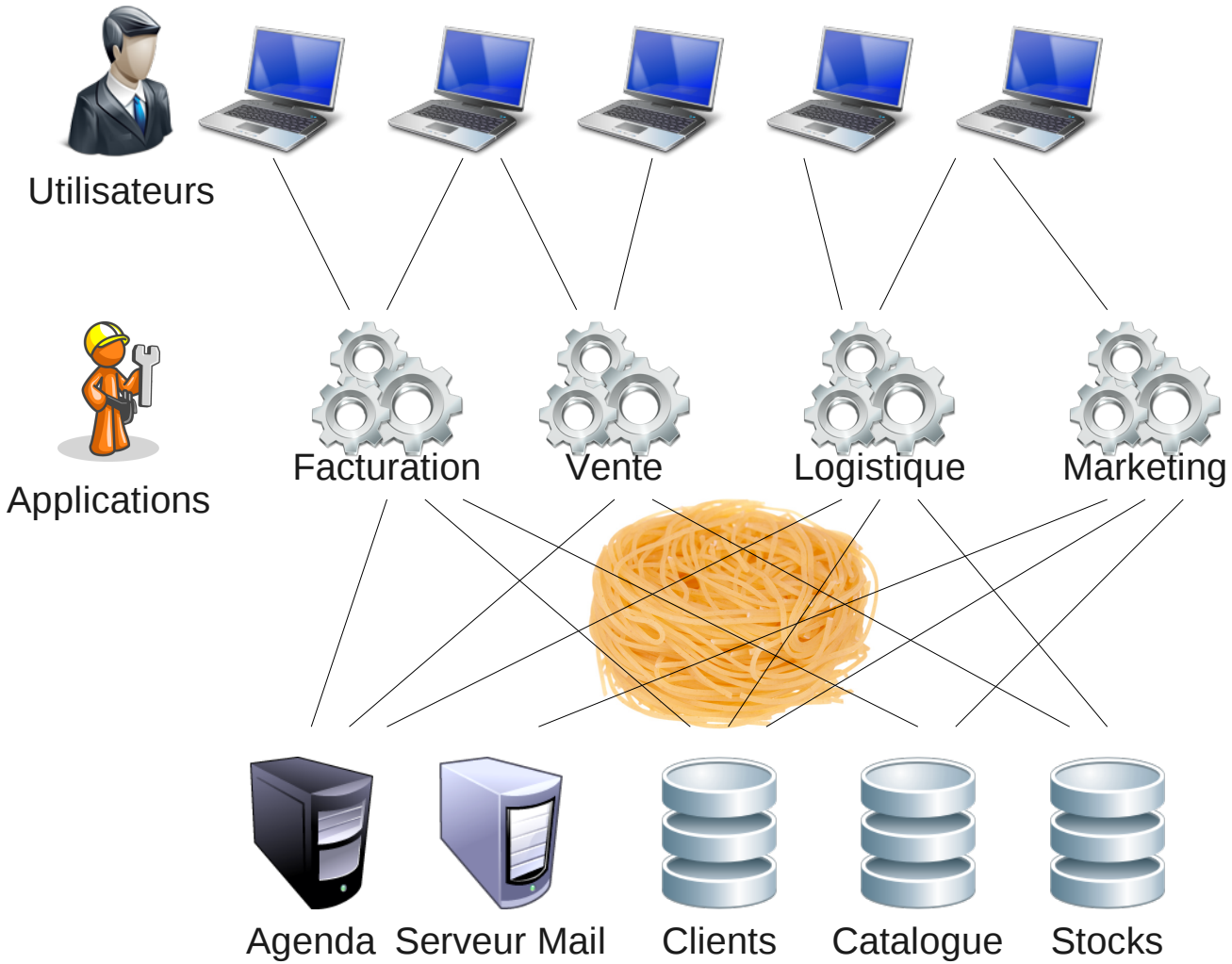
- Interopérabilité des applications
- Ouverture des SI

Plan

Plan

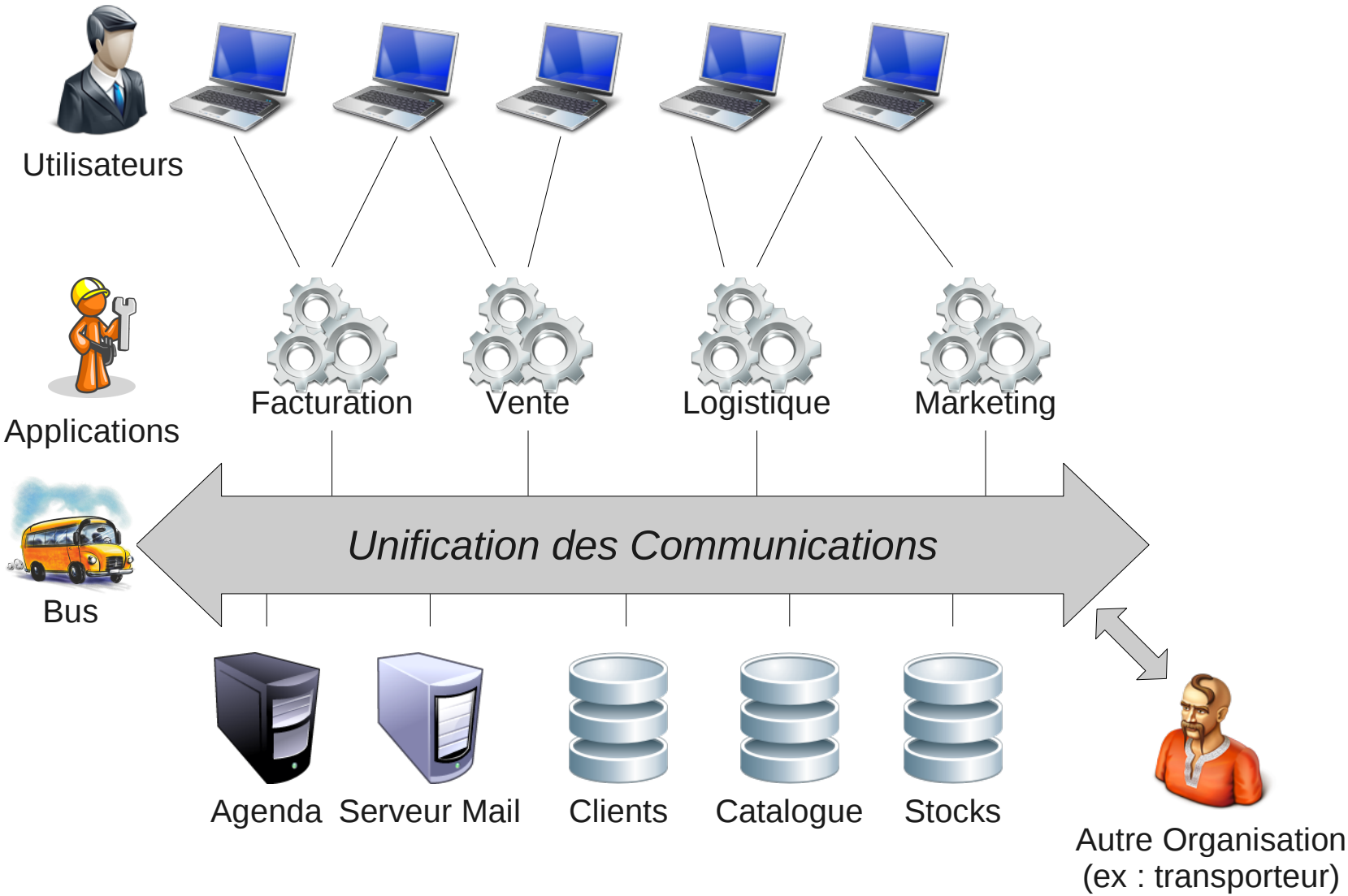
Architecture du SI

Avec Spaghetti



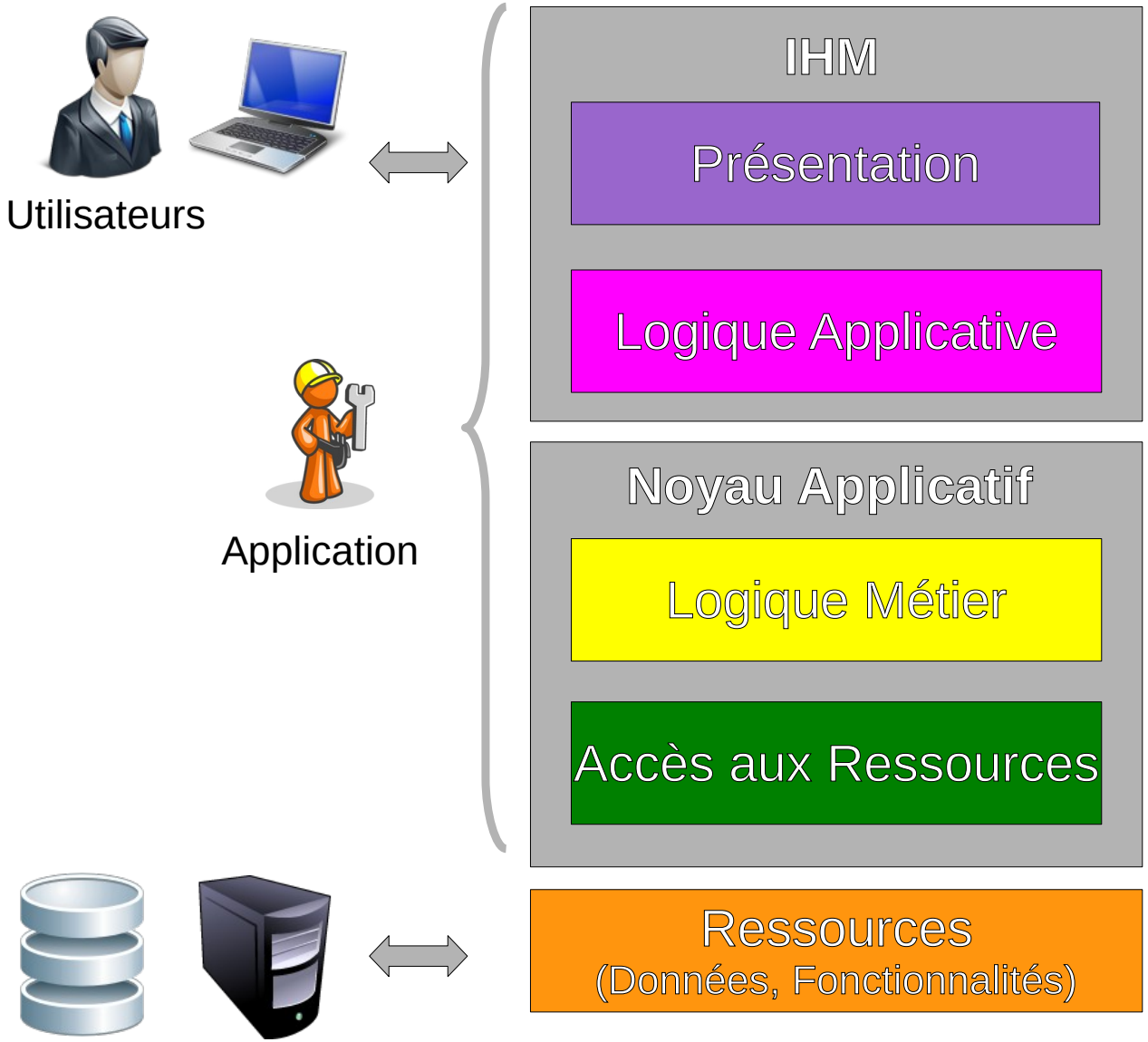
Architecture du SI

Sans Spaghetti



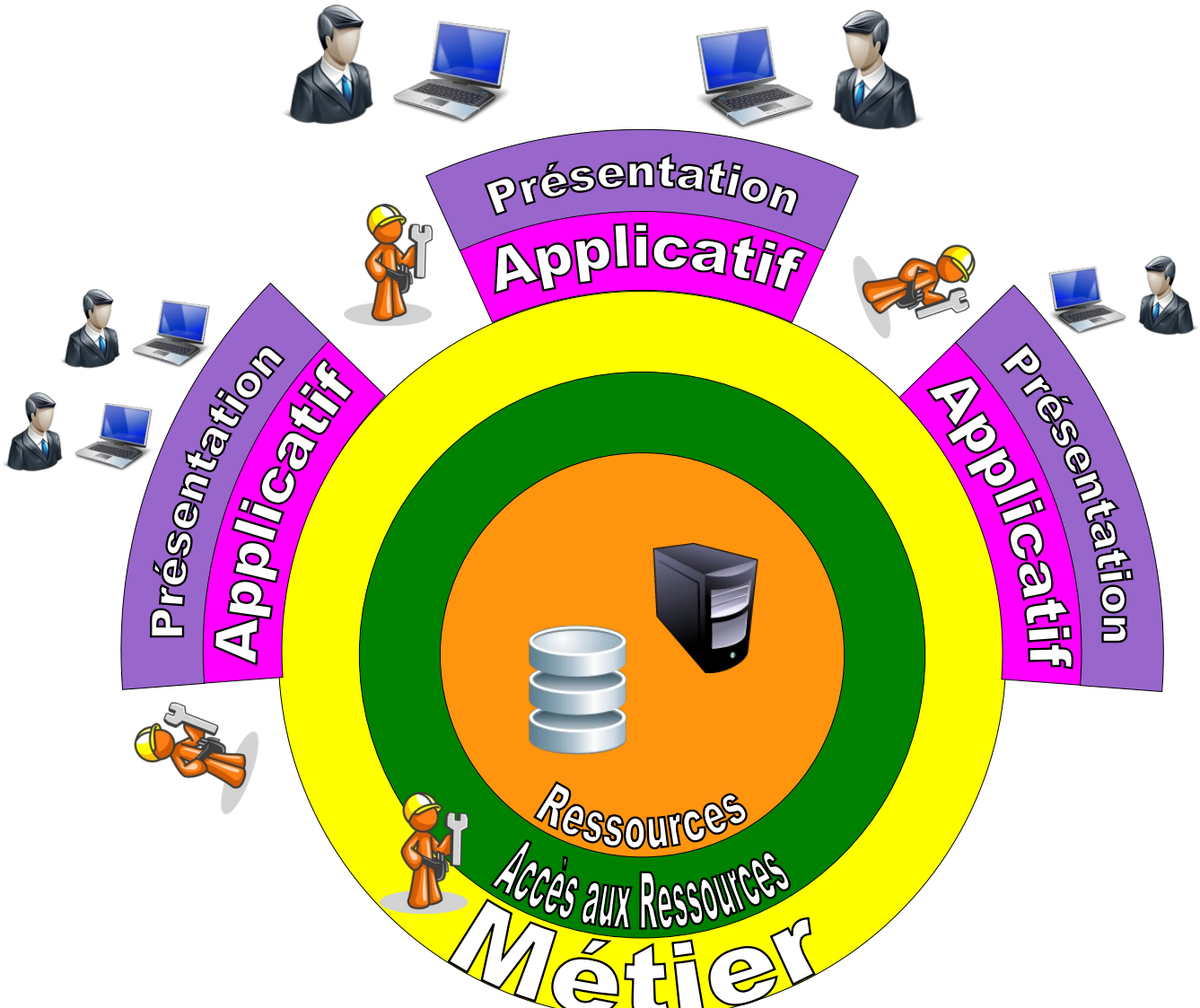
Architecture d'une Application

Architecture en couches



Architecture d'une Application

Architecture en couches



Architecture d'une Application

Architecture en couches & Composants logiciels

Intérêt

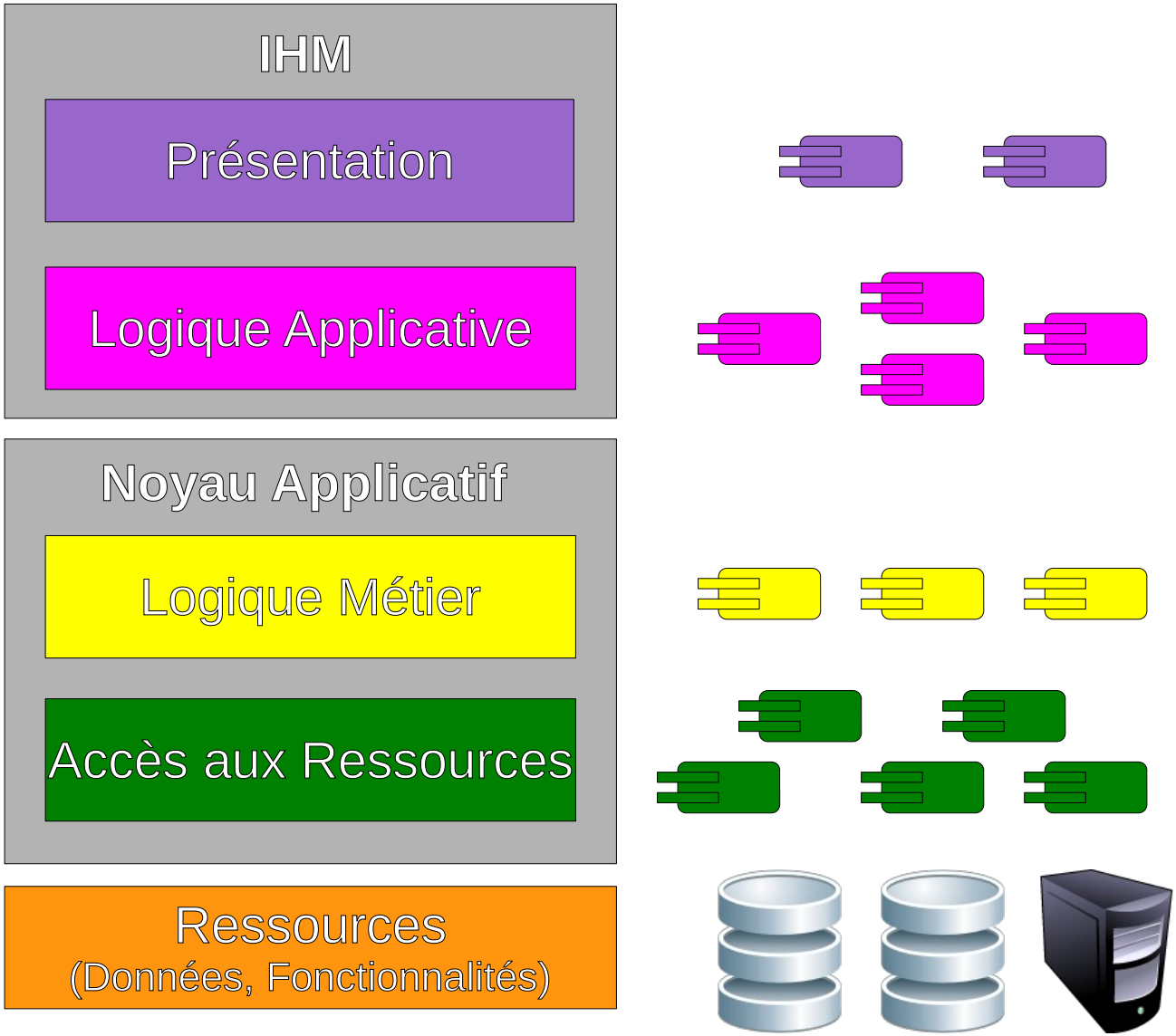
- Séparation / abstraction des problèmes
- Intégration et maintenance facilitées

Décomposition en composants logiciels

- "Morceau" de logiciel cohérent (avec dépendances identifiées)
- Organisation interne des couches
- Mutualisation entre applications
- Base pour la communication unifiée
- Répartition possible sur l'architecture technique

Architecture d'une Application

Architecture en couches & Composants logiciels



Architecture d'une Application

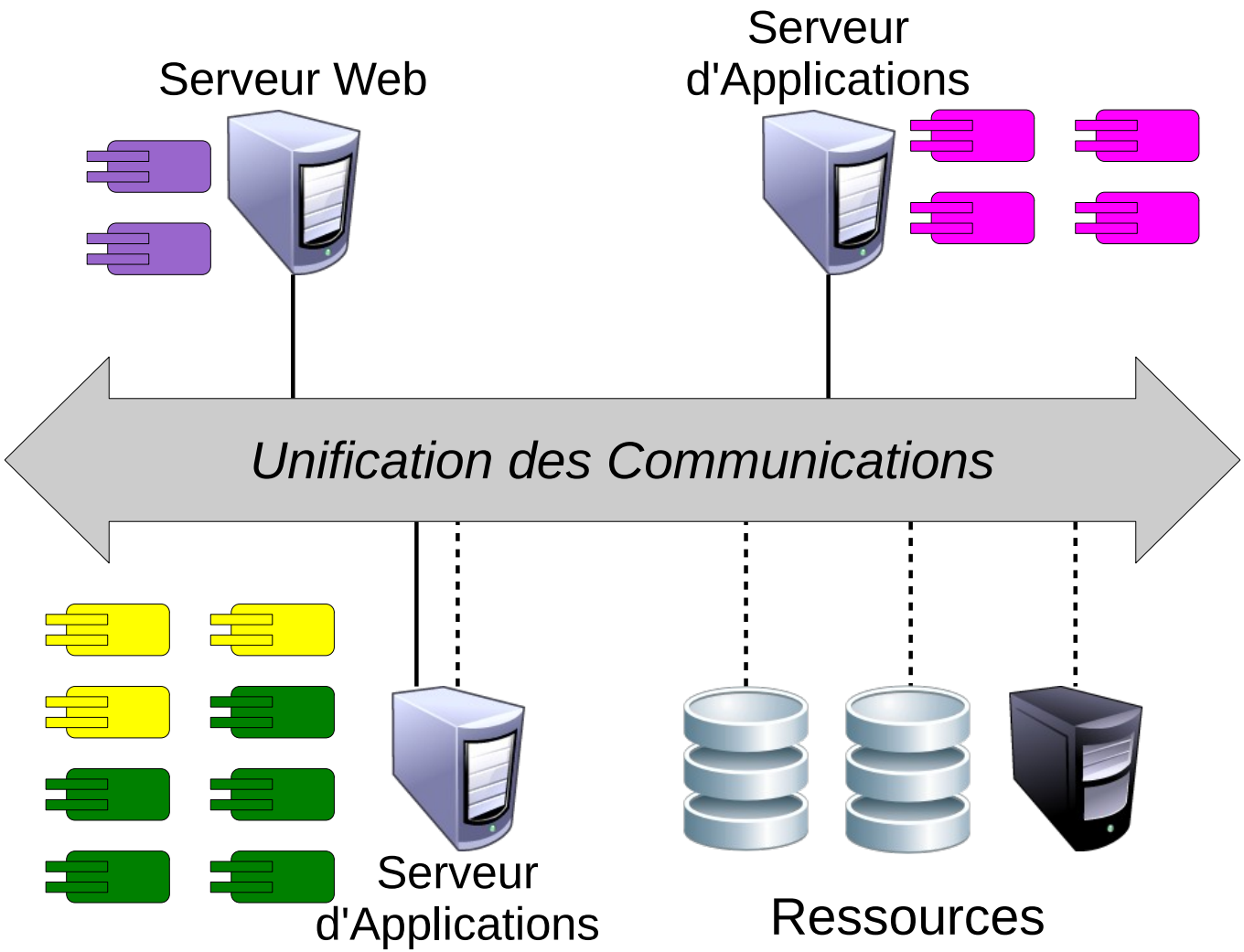
Architecture N-Tier

Vocabulaire ambigü

- Architecture N-Tier $\Rightarrow$ *N* niveaux logiques
  - **Architecture en couche** (*layer*)
  - Conception de l'application
- **Architecture N-Tier $\Rightarrow$ *N* serveurs**
  - **Architecture distribuée** (*tier* = étage)
  - Déploiement de l'application sur l'architecture technique

Architecture d'une Application

Architecture **technique** N-Tier



Architecture de Services

Principes

SOA : Service Oriented Architecture

- Style architectural pour des applications faiblement couplées
  - Interopérabilité entre composants répartis des applications
  - Communication unifiée : Composants ⇒ Services
    - Description des interfaces
    - Appels à distance des méthodes
  - Ouverture possible : Composants ⇒ Services Web
    - Exposition de composants internes
    - Intégration de composants externes

SOA = "Style Architectural"

- Conception du SI ⇒ Urbanisation
  - cf. cours de Sylvie Servigne
- Abstraction
  - Interfaces entre couches
  - Fonctionnalités et données manipulées

Construction d'une Architecture SOA

Démarche SOA

Conception d'Applications pour un Système d'Information

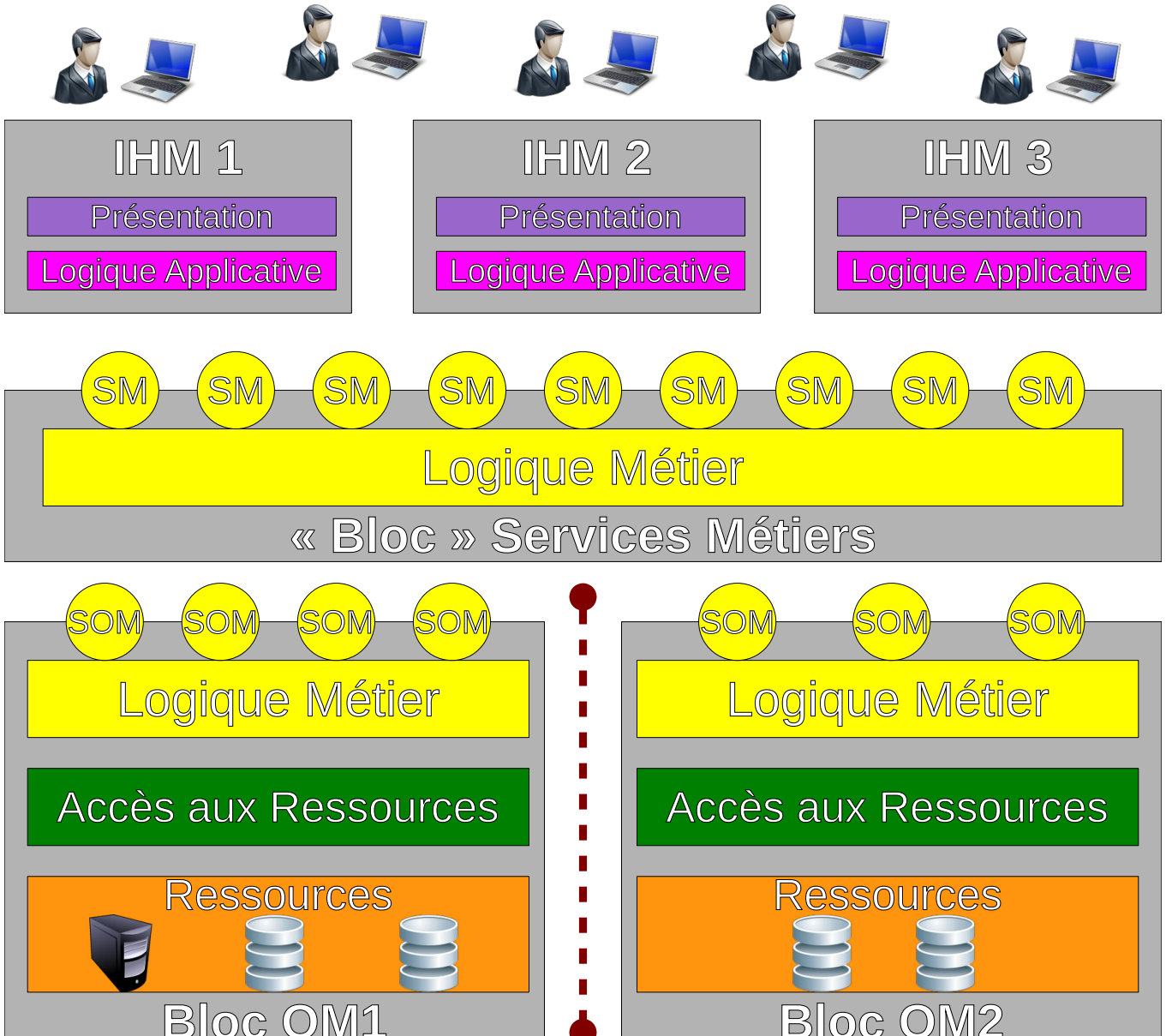
- IHMs
  - Présentation (Web / Client lourd)
  - Logique Applicative
- Services Métiers
  - Logique Métier du SI
- Services Objets Métiers
  - Découpage en Bloc (par Objet Métier : Client, Produit...)
  - Logique Métier par Bloc
  - les Blocs sont isolés !

Attention : Objet Métier $\neq$ Orienté-Objet

- Services $\simeq$ Fonctions $\neq$ Méthodes d'une Classe
- Entrées : tout dans les Paramètres (pas d'État)
- Sorties : Entités (Données) $\neq$ Objets (Références, Encapsulation)

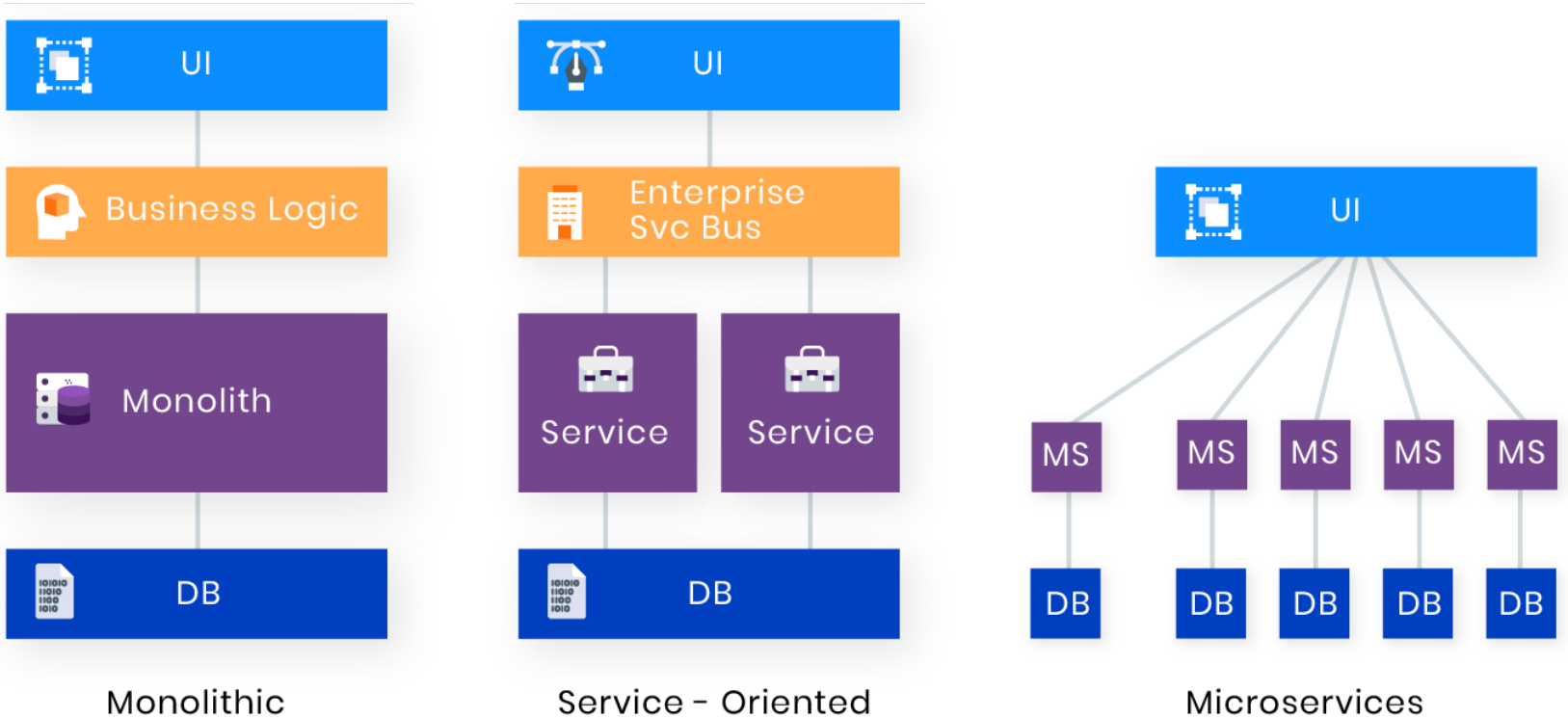
Construction d'une Architecture SOA

Démarche SOA & Architecture de Services Web



Construction d'une Architecture SOA

Démarche SOA & Architecture de Services Web – Comparaison de 3 Architectures



[**Source** : `https://rubygarage.org/blog/monolith-soa-microservices-serverless`]

Construction d'une Architecture SOA

Définition des Services

Services Métiers (Applicatifs) – SM (ou SMA)

- Appelés par les IHMs
  - En fonction des CUs
  - "Factoriser" si possible !
- Spécification de l'interface
  - Nom du Service
  - Paramètres d'entrée : id / entité / valeur, liste de ...
  - Valeur**s** de retour : id / entité / valeur, liste de ...
- Spécification du comportement
  - Appel de Services Objets Métiers des Blocs
  - Algorithme

Exemple

- SMA getIdentiteClient (idClient) : Client, Liste<idPersonne,Personne>
- SMA creerRDV (idClient, idAgent, date, heure) : idRDV

Construction d'une Architecture SOA

Définition des Services

Services Objets Métiers – SOM

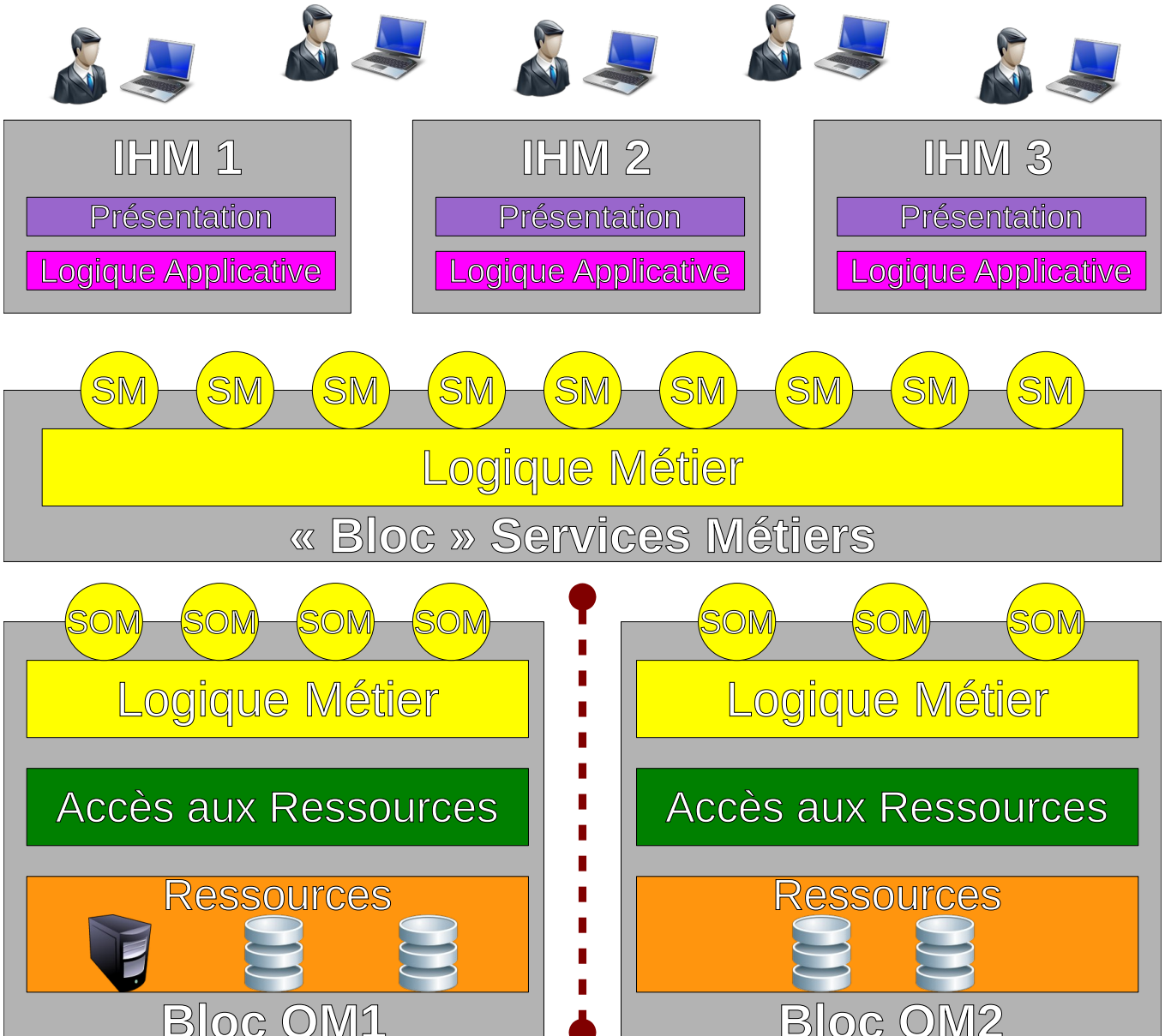
- Interface d'un Bloc Objets Métiers
  - Manipulation d'Objets Métiers
  - Abstraction des Ressources
- Spécification de l'interface
  - Nom du Service
  - Paramètres d'entrée : id / entité / valeur, liste de ...
  - Valeurs de retour : id / entité / valeur, liste de ...
- Spécification du comportement
  - Opérations CRUD abstraites (Services de Base)
  - Algorithme
  - Mais pas de "jointure" avec un autre bloc !

Exemple – Bloc Client, Bloc Personne

- SOM Client getIdentiteClient (idClient) : Client, Liste<idPersonne>
- SOM Personne getIdentitePersonne(idPersonne) : Personne, idAdresse, Adresse

Construction d'une Architecture SOA

Architecture de Services Web



Construction d'une Architecture SOA

Architecture Technique

Les Serveurs...

- Types de Serveur
  - Serveur(s) Web
  - Serveur(s) d'Application
  - Serveur(s) de Données
- Implantation
  - Emplacement physique
  - Composants déployés (Services Métiers, Blocs)
  - Redondance ?

...et le reste !

- Postes Clients
  - Client Web
  - Client lourd
- Infrastructure Réseau

Plan

Technologie de Services Web

Généralités

Protocole de communication à distance

- Publication de fonctionnalités
  - Description de l'interface exposée
- Basé sur les standards du Web
  - XML, HTTP, JSON, *etc.*
- Différents styles
  - RPC : communication synchrone
  - Messages : communication asynchrone
  - REST : manipulation de ressources

Technologie de Services Web

Protocoles “historiques” SOAP/WSDL

WSDL

- Document XML
- Description de l’interface
  - Types de données (schéma)
  - Interface abstraite (opérations)
  - Interfaces concrètes (binding)
  - Service (endpoint $\Rightarrow$ URL)

Binding SOAP

- Sérialisation en XML
- Directives WS-* (extensions) dans l’entête

Binding REST

- Limitations aux méthodes HTTP : GET, POST, PUT, DELETE
- Encodage URL possible des paramètres

Technologie de Services Web

Style REST *vs* SOAP/WSDL

REST : un style architectural ? ⇒ Web API, Open API

- Gestion de "ressources"
- Protocole plus léger pour le "Web 2.0"
  - Requêtes HTTP simples
  - Réponses en XML ou JSON

Spécifications d'API style REST & Outils

- OpenAPI / Swagger [https://swagger.io/]
- RAML [https://raml.org/]
- API Blueprint [https://apiblueprint.org/]

Exemples d'API style REST

- API OVH [https://api.ovh.com/console/]
- Google Maps Plateform [https://developers.google.com/maps/documentation/]

Serveur Web & Java

Description (*cf.* Cours DASI + Cours Réseau)

Serveur HTTP

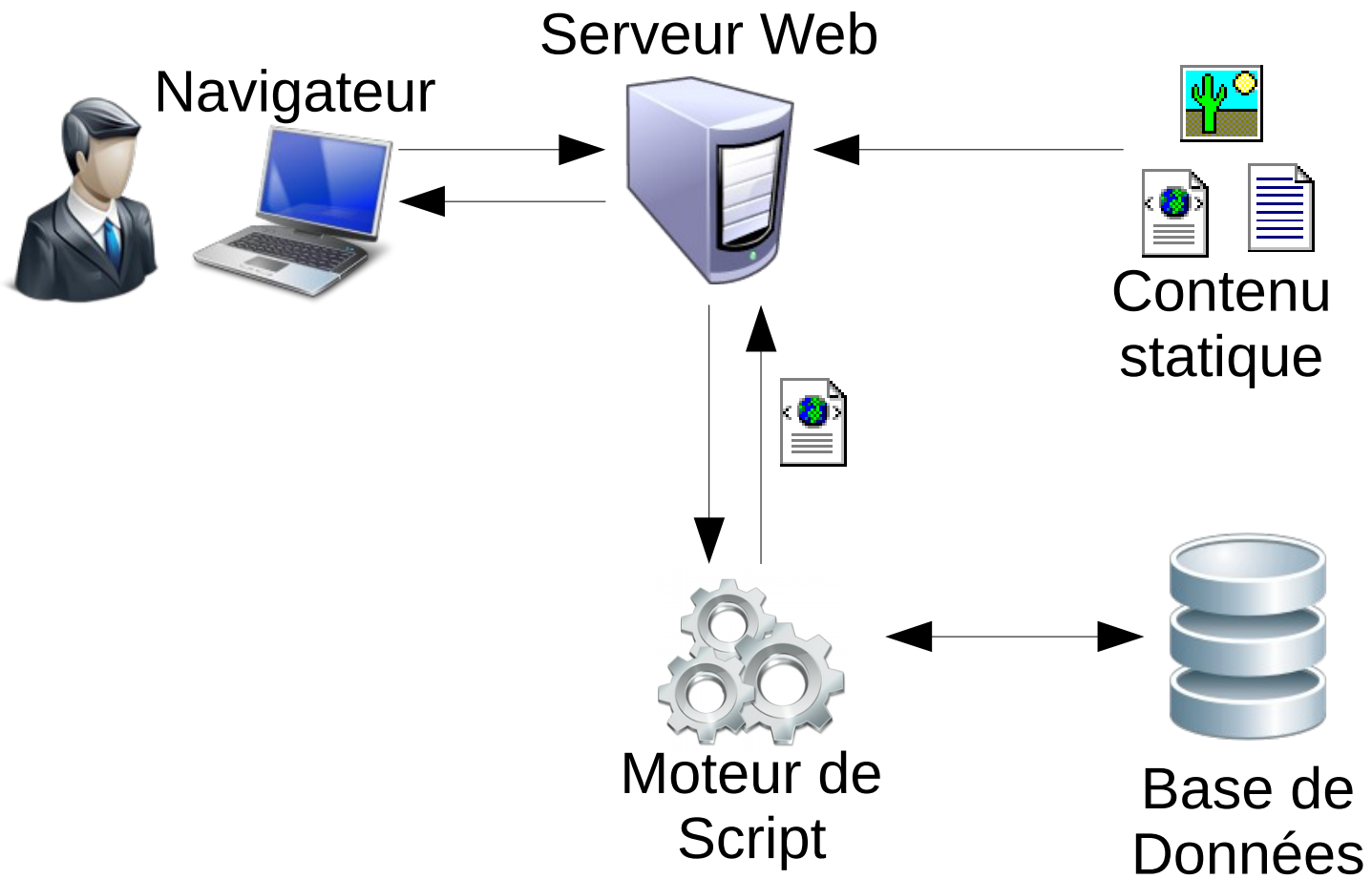
- Protocole HTTP
- Transmission de documents selon l'URL
  - Contenu Web (HTML, CSS, images)
  - Documents divers (PDF, *etc.*)
- Génération dynamique de contenu Web
  - Scripts (PHP, Perl, Python, *etc.*)
  - Gestion du contexte et de la session (cookie, *etc.*)

Serveur Web Java

- Serveur HTTP
- Conteneur de composants Java Web
  - Servlets
  - Java Server Pages (JSP)

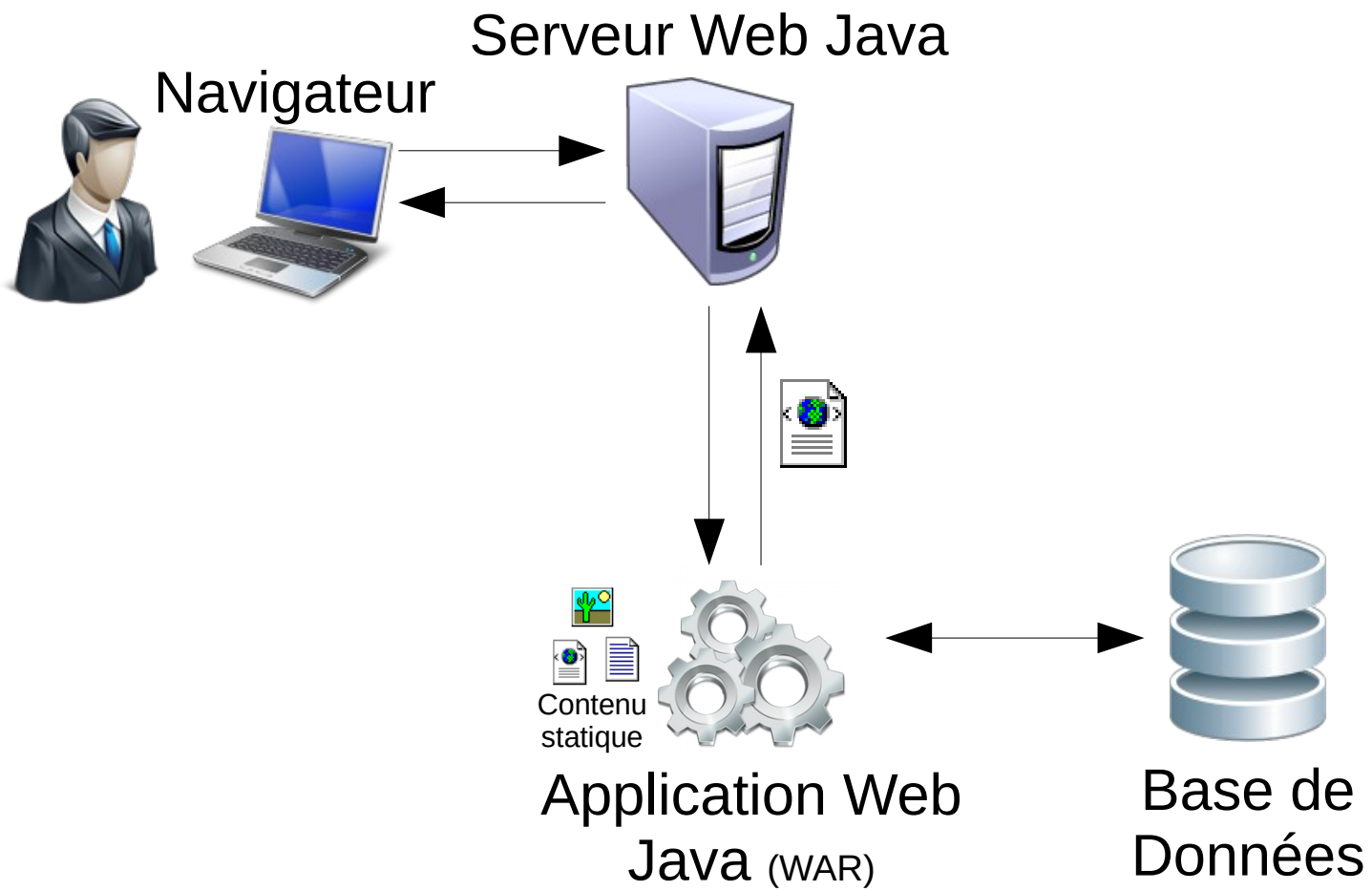
Serveur Web

Schéma



Serveur Web Java

Schéma



Requête Web

Exemple

Dans le Navigateur

```
http://localhost:8080/DASI-Web/index.html
```

Requête (simplifiée) au serveur localhost sur le port 8080

```
GET /DASI-Web/index.html HTTP/1.1
```

Réponse (simplifiée)

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=UTF-8
Content-Length: 1254

<!DOCTYPE html>
<html>
<head><title>DASI-Web - Page d'accueil</title></head>
<body>
  <h1>Bienvenue</h1>
...
```

JSON

Format de Données

JSON (JavaScript Object Notation)

- Format d'échange de données issu de JavaScript
- Simple & Léger (*vs* XML)
- Hiérarchie de Clés / Valeurs
  - Valeur simple : nombre, chaîne de caractère, booléen, `null`
  - Tableau de valeurs `[ 1, 2, "trois", 4.765, "cinq"]`
  - Objet `{ a: "val", b: 123, c: null, d: [ 1, 2, 3 ] }`

```
{
    "numero": 11223344,
    "nom": "Mentor",
    "prenom": "Gerard",
    "annee": "3IF",
    "UEs": [ "IF-3-DL", "IF-3-SI", "IF-3-SR" ],
    "adresse": {
        "rue": "17 rue des Nombres Premiers",
        "code": 17335,
        "ville": "Villeurbanne-sur-Saone"
      }
}
```

Sérialisation JSON

Sérialisation par une Servlet

Génération par une Servlet

- Sérialisation des Données
- Type Mime `application/json` au lieu de `text/html`
- Écriture sur le flux de sortie de la Servlet, et en UTF-8 ! ;-)

Librairie Java Gson

- Construction d'une structure JSON en Java
  - JsonObject – `addProperty(name,value)`
  - JsonArray – `add(element)`
- Sérialisation sur un flux de sortie
  - GsonBuilder – `toJson(element)`, `toJson(element,writer)`

Sérialisation JSON

Librairie Gson – Exemple (Java)

```java
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import com.google.gson.JsonArray;
import com.google.gson.JsonObject;

public static void printListeRestaurants(
        PrintWriter out, List<Restaurant> restaurants) {

    JsonArray jsonListe = new JsonArray();

    for (Restaurant r : restaurants) {
        JsonObject jsonRestaurant = new JsonObject();

        jsonRestaurant.addProperty("id", r.getId());
        jsonRestaurant.addProperty("denomination", r.getDenomination());
        jsonRestaurant.addProperty("description", r.getDescription());
        jsonRestaurant.addProperty("adresse", r.getAdresse());

        jsonListe.add(jsonRestaurant);
    }

    // Objet JSON "conteneur"
    JsonObject container = new JsonObject();
    container.add("restaurants", jsonListe);

    // Serialisation & Ecriture sur le flux de sortie
    Gson gson = new GsonBuilder().setPrettyPrinting().create();
    String json = gson.toJson(container);
    out.println(json);
}
```

Sérialisation JSON

Librairie Gson – Exemple (JSON)

```
 1 {
 2   "restaurants": [
 3     {
 4       "id": 10,
 5       "denomination": "Chez Maman",
 6       "description": "Plats originaux a la presentation raffinee pour ce bistrot
             intimiste au decor chaleureux, avec terrasse.",
 7       "adresse": "145 r Marcel Merieux, 69007 Lyon "
 8     },
 9     {
10       "id": 9,
11       "denomination": "L\u0027entre-potes",
12       "description": "Cadre feutre et elegant pour ce restaurant gastronomique
             aux produits de saison cuisines artisanalement.",
13       "adresse": "6 pl Gabriel Peri, 69007 Lyon "
14     },
15     {
16       "id": 7,
17       "denomination": "La Ciboulette",
18       "description": "Une carte bio et vegetarienne volontairement restreinte
             servie dans une salle au mobilier simple et colore.",
19       "adresse": "3 pl Jules Ferry, 69006 Lyon "
20     }
21   ]
22 }
```

Services Web style REST en Java

Implémentation “à la main”

1 Servlet = 1 Service Web

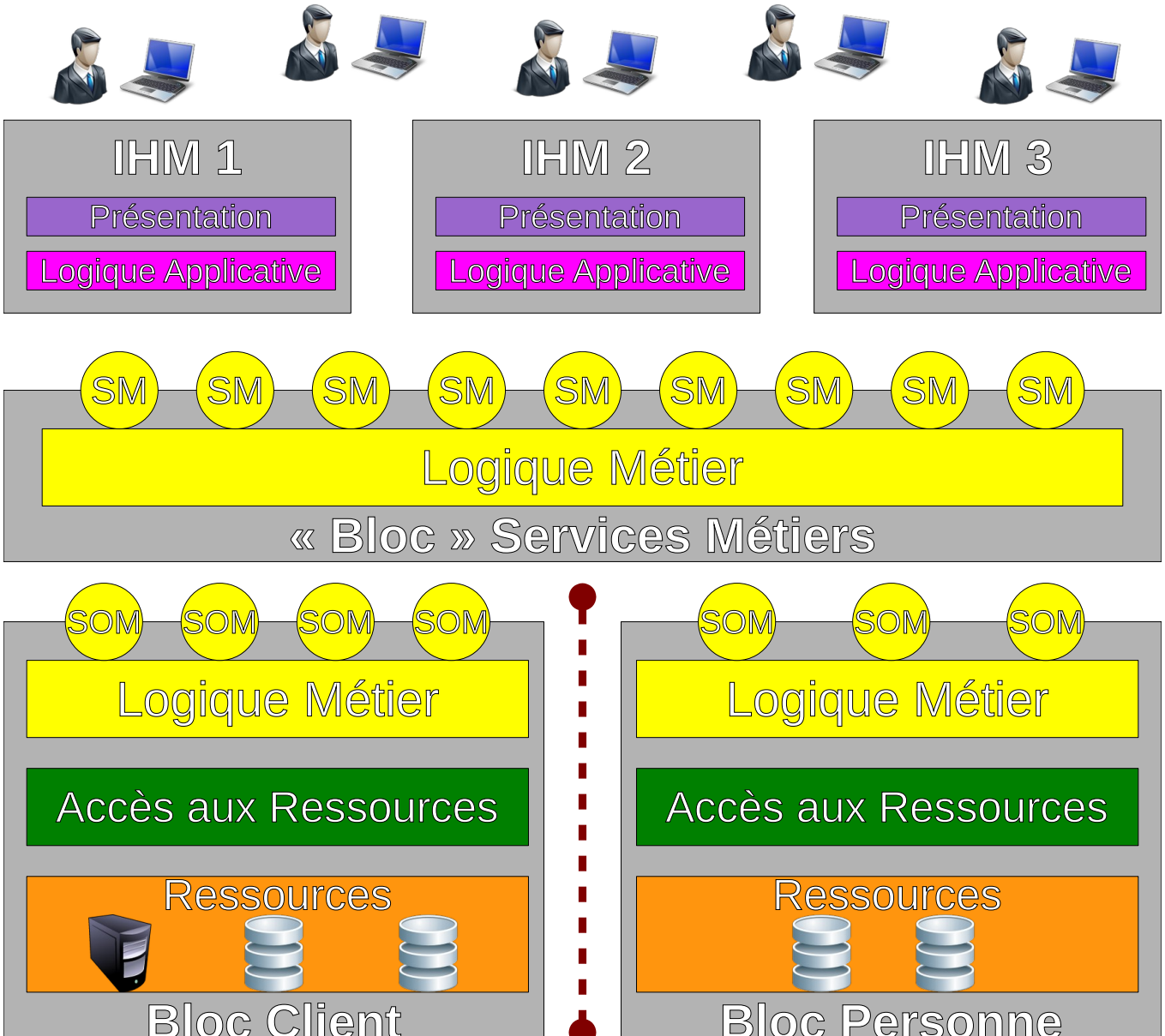
- URL de la Servlet
- Nom de la méthode du Service Web
- Paramètres comme les formulaires Web (GET ou POST)
- Logique du Service (algorithme)
- Retour au format JSON

Appel du Service Web

- Créer une requête Web : URL + méthode
- Passer les paramètres (GET ou POST)
- Lire le retour au format JSON

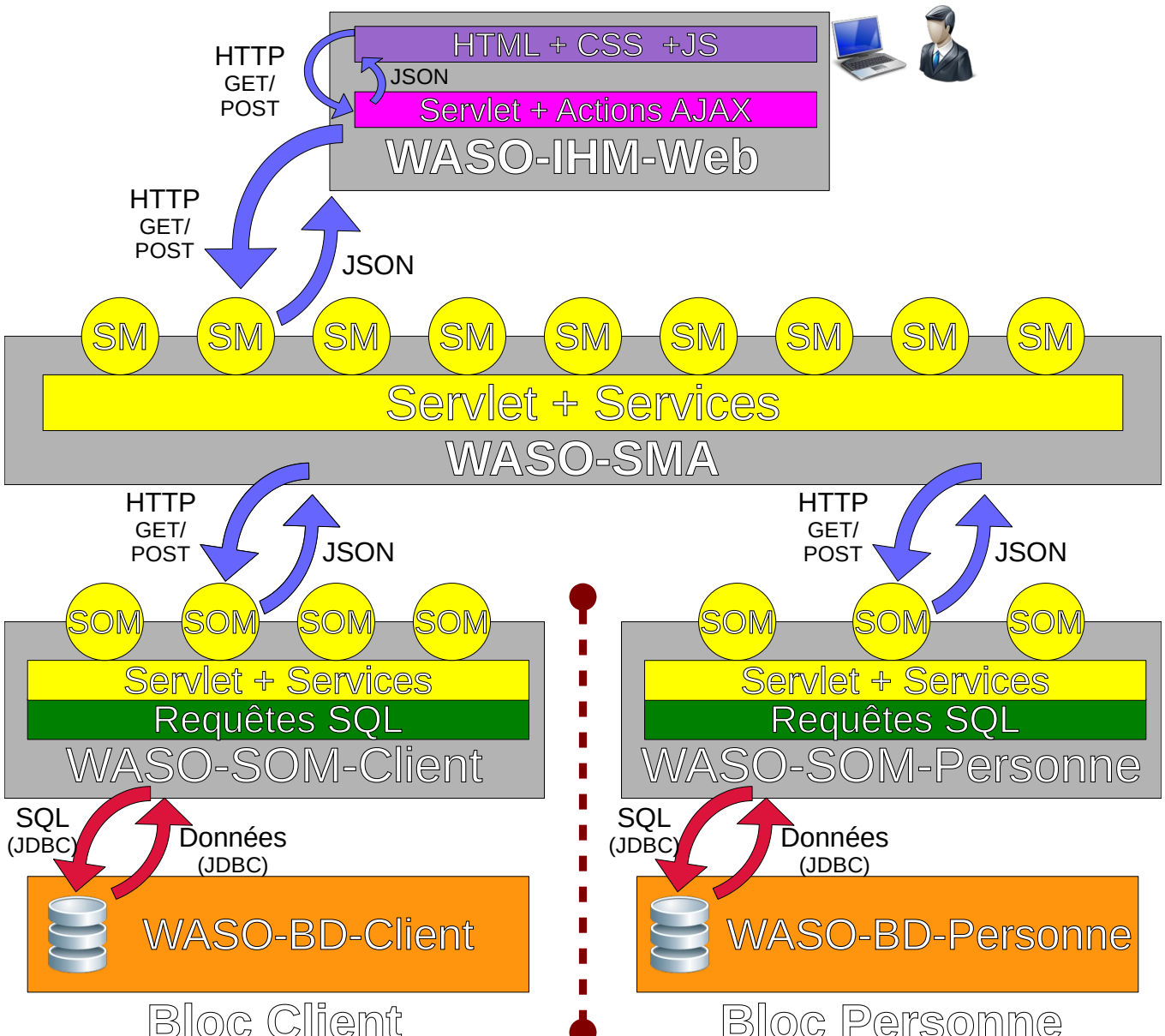
Services Web style REST en Java

Retour sur l'Architecture – Conception



Services Web style REST en Java

Retour sur l'Architecture – Développement



Services Web style REST en Java

Retour sur l'Architecture – Développement

1 Application Web (WAR) = 1 Composant

- Unité de déploiement indépendante
- À déployer sur un Serveur d'Application Java
  - Tomcat, Glassfish...

Configuration

- URL de l'Application Web (ex : `/ServiceMetier`)
- URL relative de la Servlet (ex : `/WS`)
- *URL vers les autres Services à appeler*
- *URL JDBC vers la BD*

Architecture distribuée...

- Ne pas oublier l'URL complète du Service Web !
- Ex : `http://IF213-03.insa-lyon.fr:8080/ServiceMetier/WS`

Requête au Service Web

Exemple

Pour les Tests : directement dans le Navigateur !

```
http://localhost:8080/WASO-SMA/WS?method=getClientIdentite&id=17
```

Requête (simplifiée) au serveur `localhost` sur le port 8080

```
GET /WASO-SMA/WS?method=getClientIdentite&id=17 HTTP/1.1
```

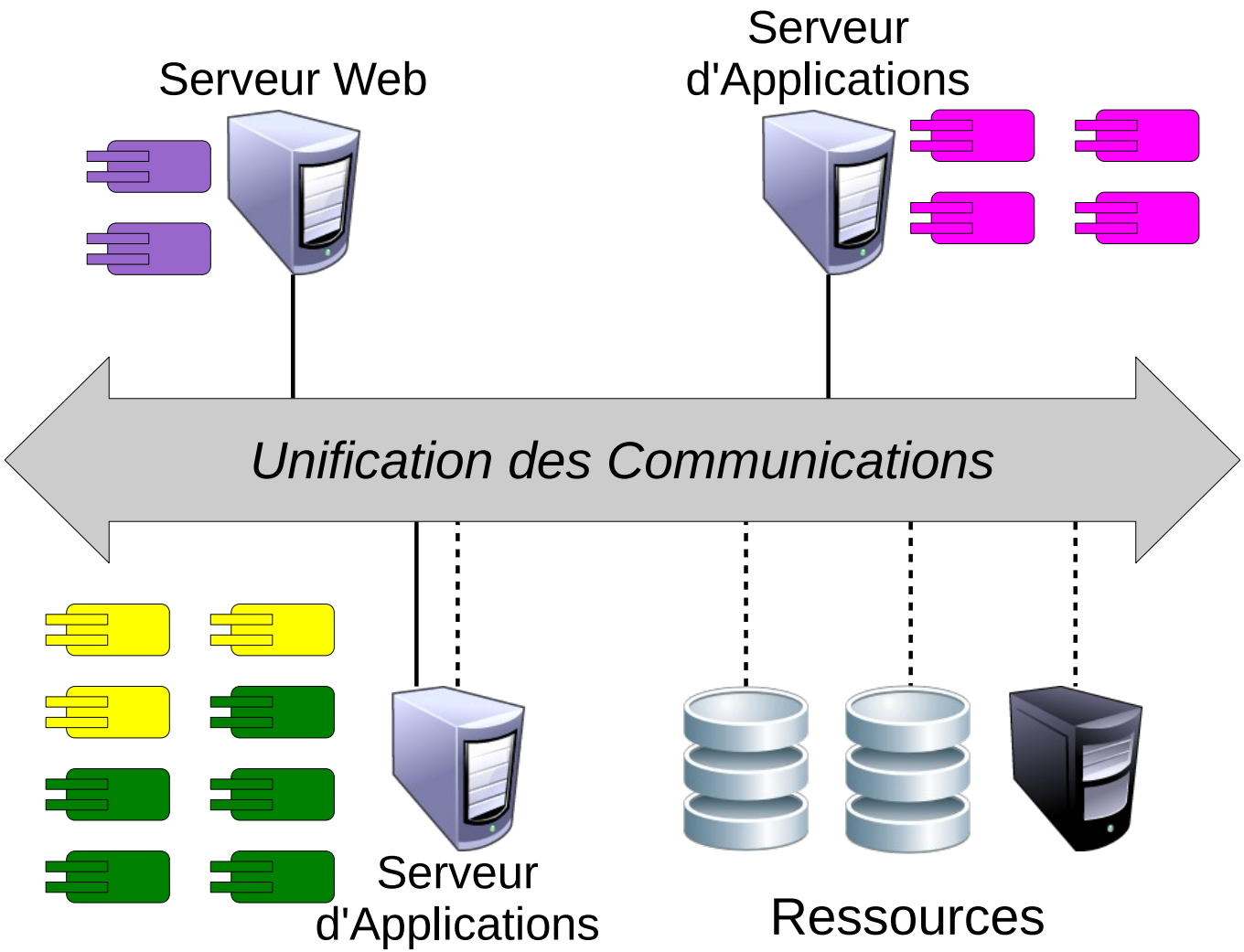
Réponse (simplifiée)

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Content-Length: 895

{ "client": {
    "id": 17,
    "nom": "Mentor",
    "prenom": "Gerard",
    "adresse": {
        "rue": "17 rue des Nombres Premiers", ...
```

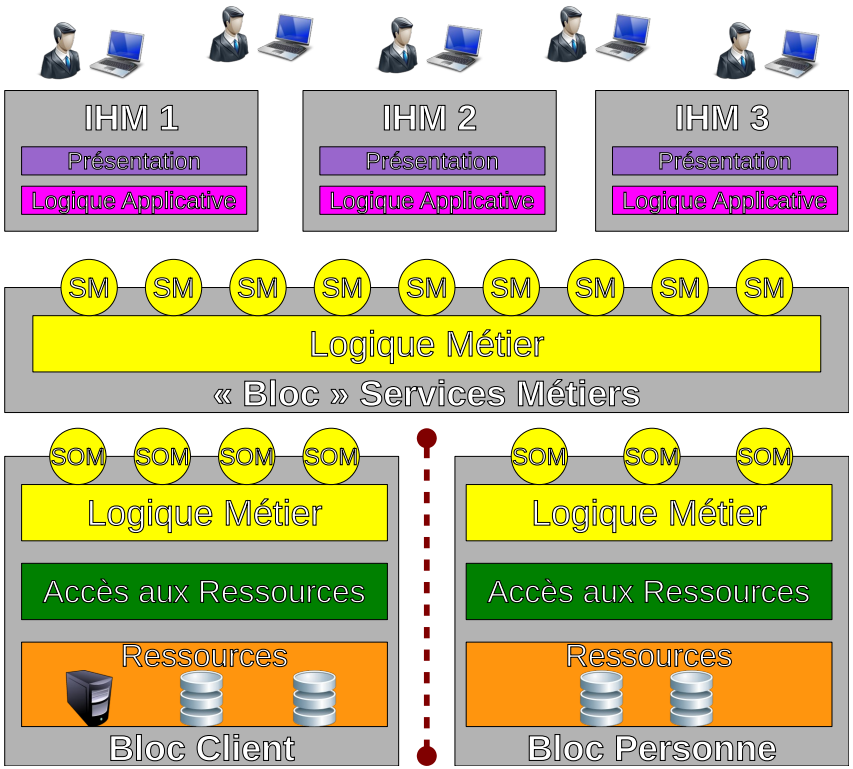
Architecture d'une Application

⇒ Plus qu'à déployer...



Construction d'une Architecture SOA

⇒ C'est fait !



Plan

Ouverture des SI

Standards et Interopérabilité

Utilisation des standards

- Interopérabilité des systèmes
  - Indépendance vis-à-vis des plateformes techniques
  - Réutilisabilité : librairies, applications (clients, serveurs)
- Choix des formats et protocoles
  - Ouverts *vs* fermés
  - Libres *vs* propriétaires
  - Standards ?
- $\Rightarrow$ ouverture, accessibilité, internationalisation, pérennité

Un véritable enjeu (dont vous êtes le héros)

- Choix techniques, mais impacts sociétaux
- Problématique actuelle (Web, format audio/vidéo, *etc.*)
- Liberté, Égalité, Fraternité

Problème d'encodage de caractères ?

Vive Unicode !

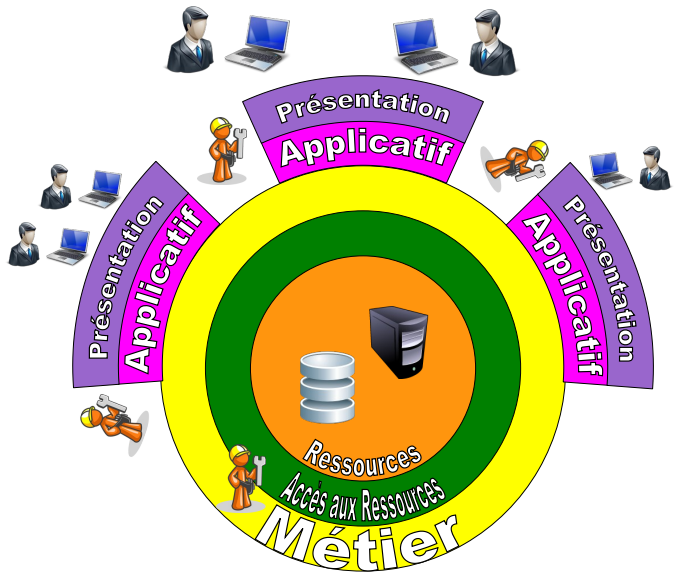
Unicode

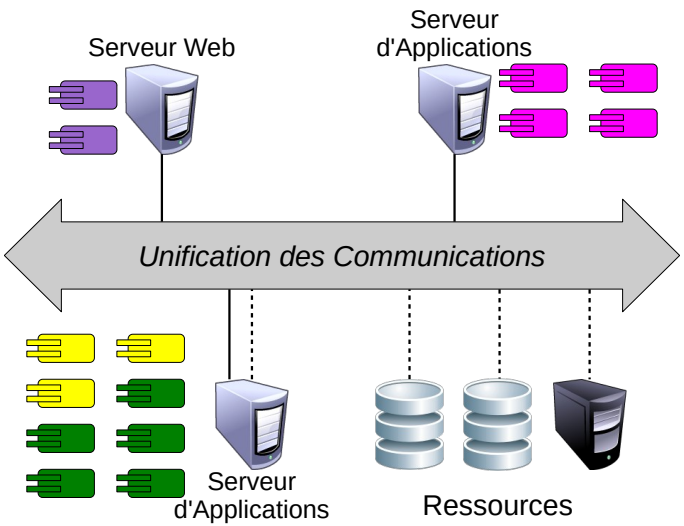
- À savoir : le "texte brut" n'existe pas...
  - [http://french.joelonsoftware.com/Articles/Unicode.html]
- Standard Unicode
  - Standard international de représentation des caractères
  - Tous les caractères (ou presque) de différents langages
    - Plus de 100000 caractères
  - Chaque caractère = 1 code numérique ('A' = U+0041)
- Encodages Unicode
  - Représentation binaire des caractères
  - Code ⇒ séquence d'octets
  - **UTF-8**, UTF-16, UTF-32...et les autres (ISO-8859-1, *etc.*)

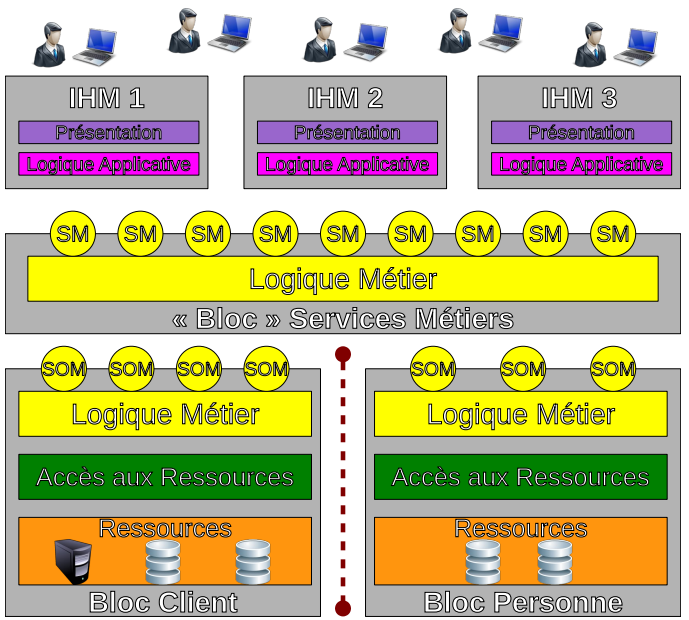
Programmer en Japonais ?

\ (^ _ ^) /

FIN


Présentation
Applicatif
Présentation
Applicatif
Présentation
Applicatif
Ressources
Accès aux Ressources
Métier


Serveur Web
Serveur
d'Applications
Unification des Communications
Serveur
d'Applications
Ressources


IHM 1
Présentation
Logique Applicative
IHM 2
Présentation
Logique Applicative
IHM 3
Présentation
Logique Applicative
SM SM SM SM SM SM SM SM SM
Logique Métier
« Bloc » Services Métiers
SOM SOM SOM SOM
Logique Métier
Accès aux Ressources
Ressources
Bloc Client
SOM SOM SOM
Logique Métier
Accès aux Ressources
Ressources
Bloc Personne