Service Web .Net

Cours n°2 : API

Pierre-Loïc CHEVILLOT – Sébastien BEREIZIAT
Pierre-loic.chevillot@capgemini.com – sebastien.bereiziat@capgemini.com

Plan

- WebServices
  - Architectures
  - Standards
- API
  - NetCore
  - MicroService vs Function (AzFunction)

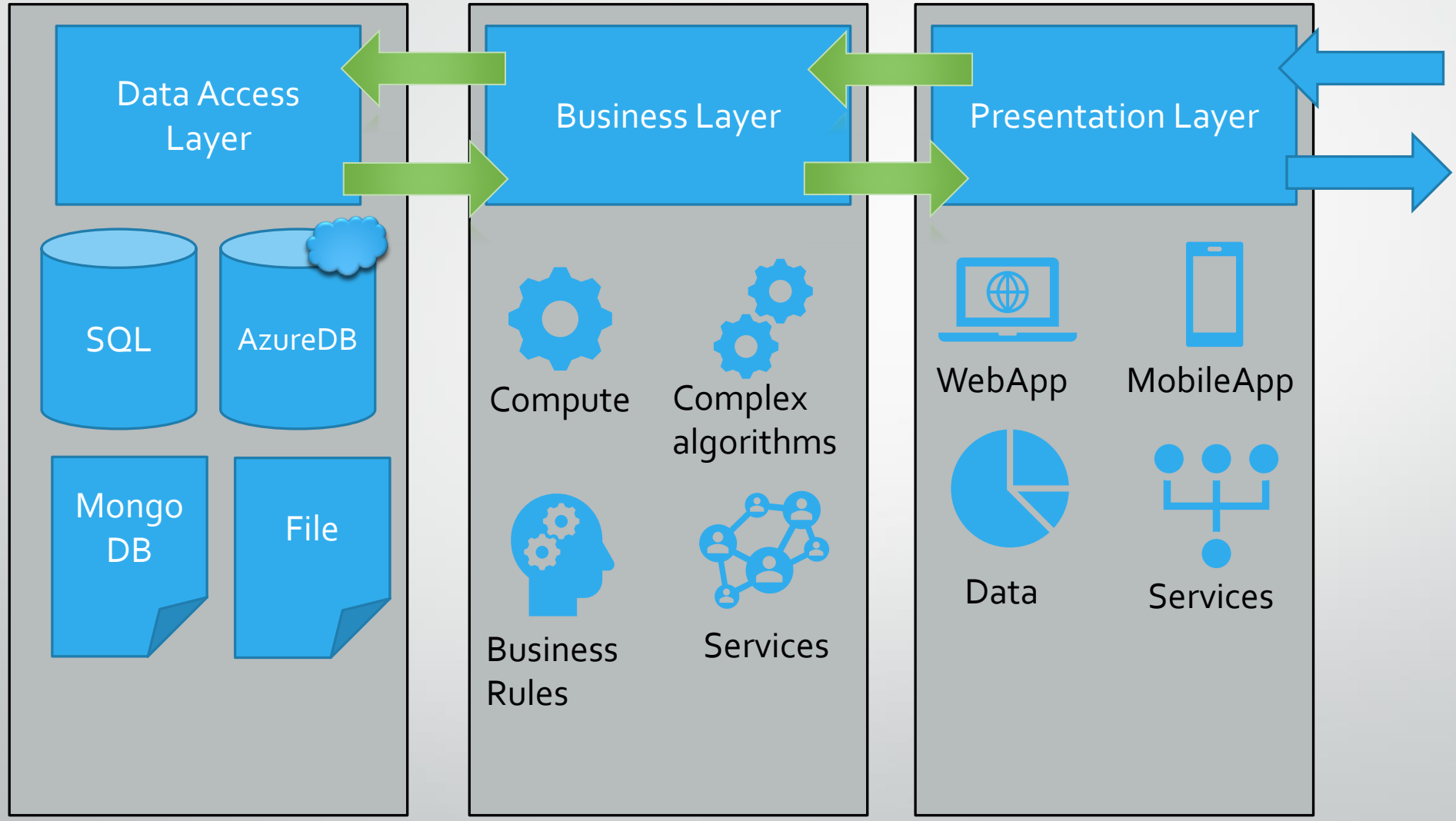
Architecture logicielle

- L'**architecture logicielle** décrit d'une manière symbolique et schématique les différents éléments d'un ou de plusieurs systèmes informatiques, leurs interrelations et leurs interactions.
- Un modèle d'architecture logicielle ne décrit pas ce que doit réaliser un système informatique, mais comment il est conçu afin de répondre aux spécifications (dimensionnement, sécurité, hébergement, …)

Architecture en couches

- But : Modéliser une application comme un empilement de X couches logicielles
- 3-Tiers est la modélisation la plus couramment utilisée
  - Couche de présentation : Correspond a l'affichage de vos données a l'utilisateur final, et le dialogue avec celui-ci
  - Couche d'application (ou métier) : Correspond a la mise en œuvre des règles de gestion et de logique applicative
  - Couche de données : Correspond a la persistances des données.
- Avantage :
  - SoC (Separation of Concern) : chaque couche gère son domaine

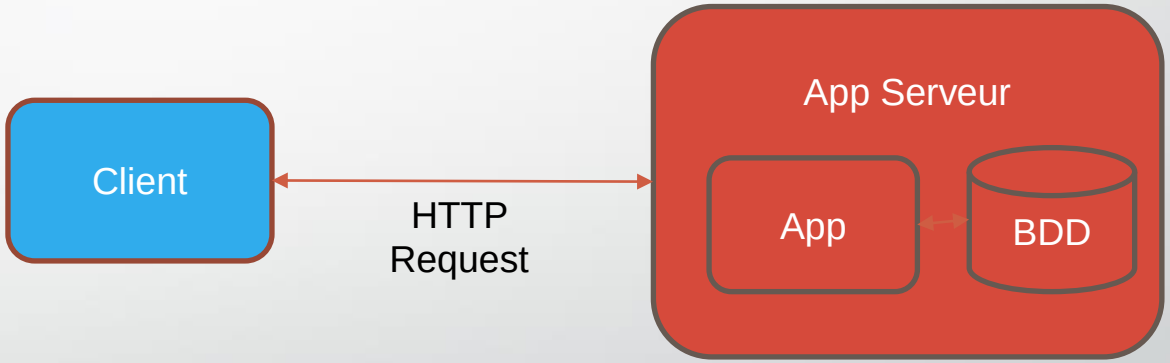
Architecture en couches



Architecture Client / Serveur

- Client : Demandeur d'une information / action
- Serveur : Effectue l'action, restitue l'information

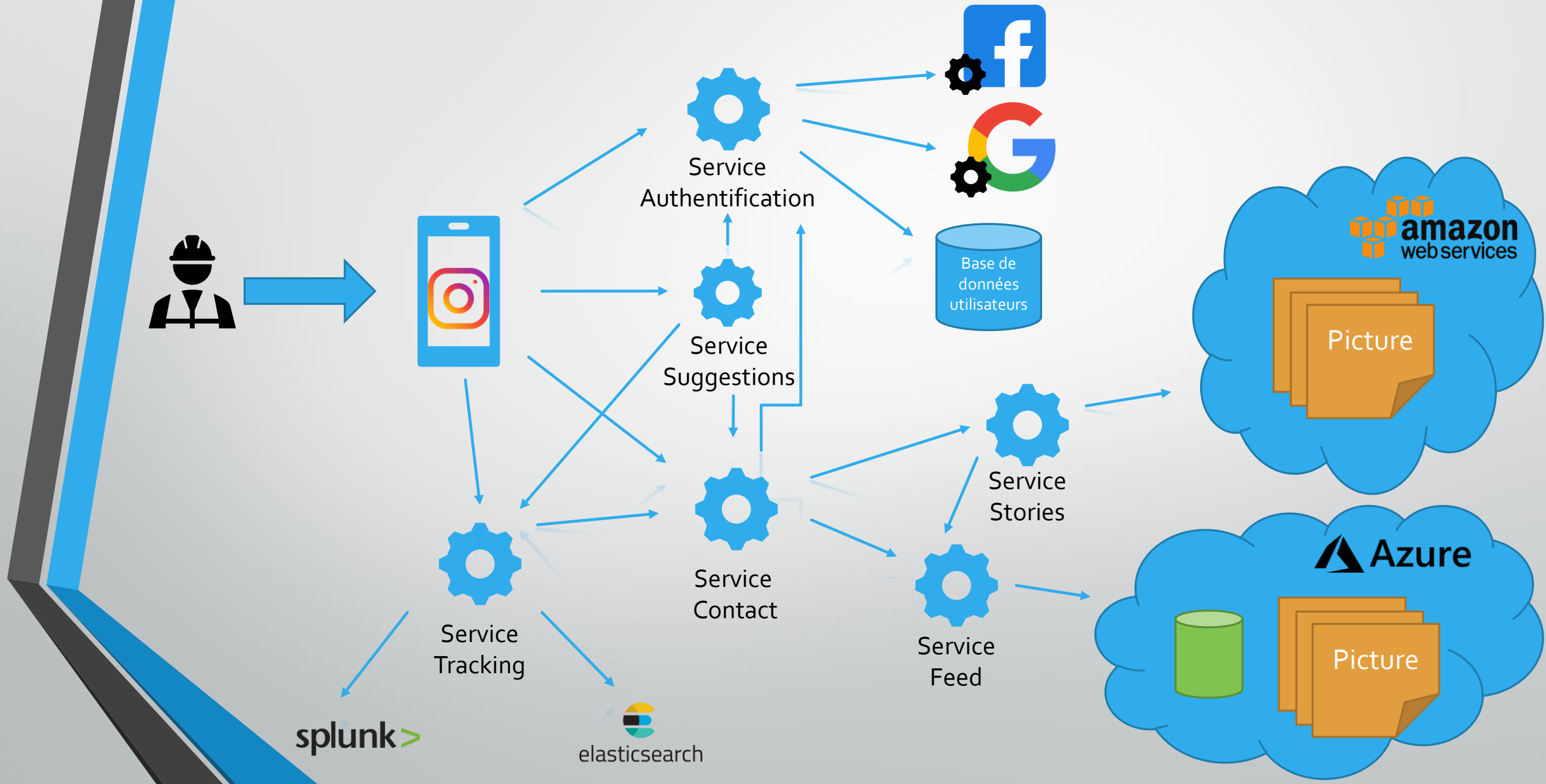
- SoC (Separation of Concern) : Séparation des responsabilités
  - Interface cliente dissociée du stockage des données
  - Composants indépendants les uns des autres



Architecture Micro/Nano Service

- But : Réduire la complexité du changement, et accroitre la compétitivité d'une « entreprise »
- Conception drivée par les données a exposer.
- Avantage :
  - Déploiement indépendant
  - Conception indépendante
  - Adhérence faible entre services

Architecture Micro/Nano Service

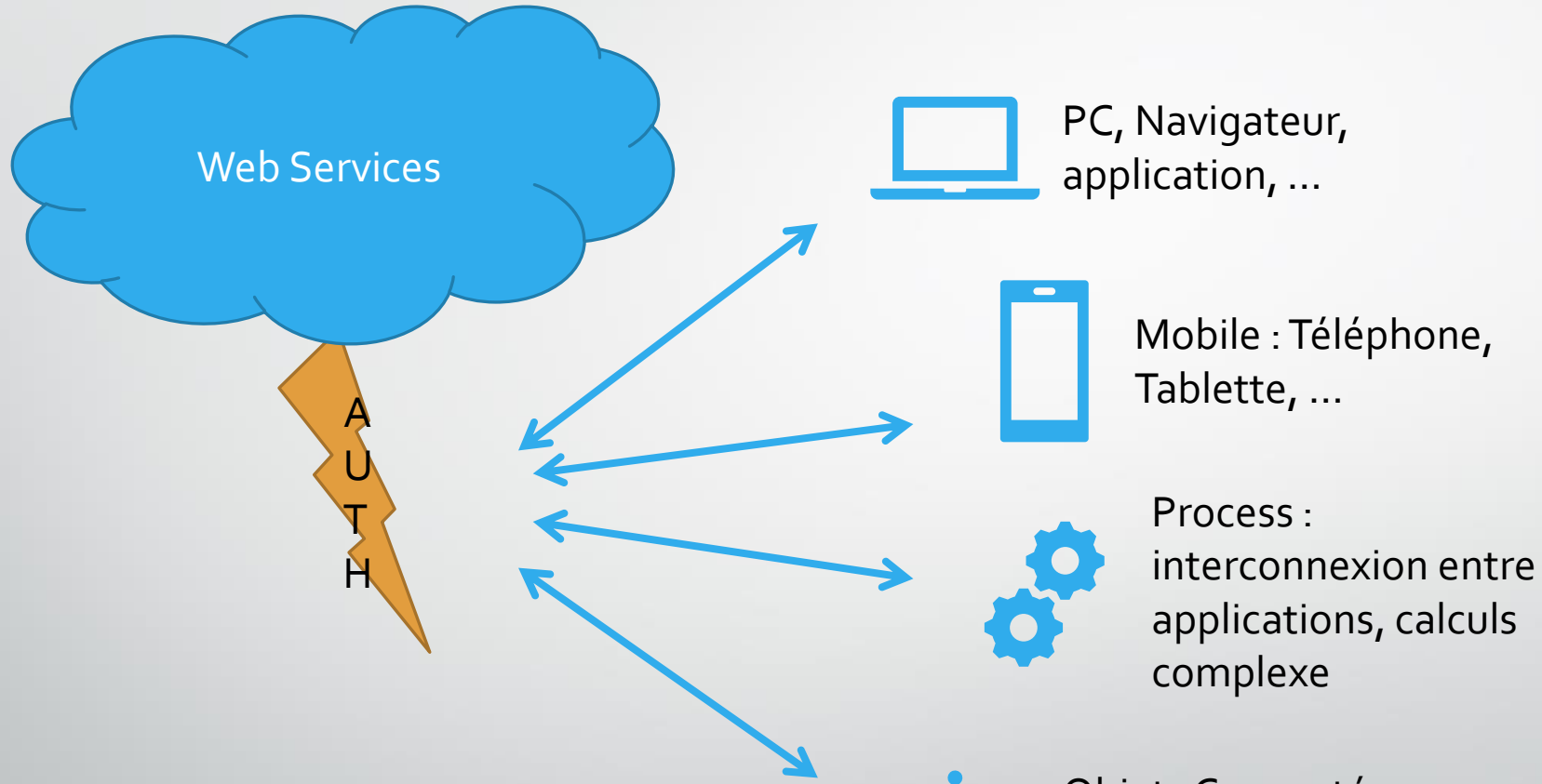


Web Services

- Web service : Application avec laquelle on communique par l'échange de message HTTP
- Service Oriented Architecture
  - Frontières explicites
- Coût des transactions
  - Entité autonome
- Déploiement et évolution d'un service indépendant des clients
  - Structure d'utilisation définie sans ambiguïté
  - Sémantique d'utilisation définie sans ambiguïté (avant WSDL, aujourd'hui, documentations (Swagger …)

API

- API : Application Programming Interface
  - Interface logicielle
- Application a laquelle on demande d'effectuer des actions via HTTP
  - Récupérer des données
  - Ajouter / Supprimer / Modifier des données
  - Lier des données entre elles
  - Transmettre des données quelques part


Web Services
AUTH
PC, Navigateur, application, ...
Mobile : Téléphone, Tablette, ...
Process : interconnexion entre applications, calculs complexe
Objets Connectés : Transmissions de data, IoT

SOAP/WS

- **SOAP** (Simple Object Access Protocol) : protocole permettant l'échange de données quelque soit la plateforme (flux de données XML/Json sur HTTP).
- **WSDL** (Web Service Description Language) : description au format XML des fonctionnalités du service web

SOAP/WS

Request

```
<?xml version="1.0" encoding="UTF-8" ?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/
"
                  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <ns0:additionner
      xmlns:ns0="http://axis.test.com"
      soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <valeur1 xsi:type="xsd:int">40</valeur1>
      <valeur2 xsi:type="xsd:int">2</valeur2>
    </ns0:additionner>
  </soapenv:Body>
</soapenv:Envelope>
```

Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
                  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <ns1:additionnerResponse
      xmlns:ns1="http://axis.test.com"
      soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <additionnerReturn href="#id0" />
    </ns1:additionnerResponse>
    <multiRef xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
              id="id0" soapenc:root="0"
              soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
              xsi:type="xsd:long">42</multiRef>
  </soapenv:Body>
</soapenv:Envelope>
```

WSDL Documentation

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="http://axis.test.com"
                  xmlns:apachesoap="http://xml.apache.org/xml-soap"
                  xmlns:impl="http://axis.test.com"
                  xmlns:intf="http://axis.test.com"
                  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
                  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
                  xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
                  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<!--WSDL created by Apache Axis version: 1.3
Built on Oct 05, 2005 (05:23:37 EDT)-->
   <wsdl:message name="additionnerRequest">
      <wsdl:part name="valeur1" type="xsd:int"/>
      <wsdl:part name="valeur2" type="xsd:int"/>
   </wsdl:message>
   <wsdl:message name="additionnerResponse">
      <wsdl:part name="additionnerReturn" type="xsd:long"/>
   </wsdl:message>
   <wsdl:portType name="Calculer">
      <wsdl:operation name="additionner" parameterOrder="valeur1 valeur2">
         <wsdl:input message="impl:additionnerRequest" name="additionnerRequest"/>
         <wsdl:output message="impl:additionnerResponse" name="additionnerResponse"/>
      </wsdl:operation>
   </wsdl:portType>
   <wsdl:binding name="CalculerSoapBinding" type="impl:Calculer">
      <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
      <wsdl:operation name="additionner">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="additionnerRequest">
            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
              namespace="http://axis.test.com" use="encoded"/>
         </wsdl:input>
         <wsdl:output name="additionnerResponse">
            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
              namespace="http://axis.test.com" use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
   </wsdl:binding>
   <wsdl:service name="CalculerService">
      <wsdl:port binding="impl:CalculerSoapBinding" name="Calculer">
         <wsdlsoap:address location="http://localhost:8080/TestWS/services/Calculer"/>
      </wsdl:port>
   </wsdl:service>
</wsdl:definitions>
```

"If you can't explain it to a six year old, you don't understand it yourself"

Albert Einstein

ReST

- ReST (Representational State Transfer) ou RESTFul
  - Les services web REST permettent aux systèmes effectuant des requêtes de manipuler des ressources web via leurs représentations textuelles à travers un ensemble d'opérations uniformes et prédéfinies
  - HTML/JSON/XML
- Six contraintes architecturales définissent un système REST
  - Performances
  - Extensibilité
  - Simplicité
  - Visibilité
  - Portabilité
  - Fiabilité
- Stateless : votre API ne doit conserver aucun contexte entre deux appels successifs. Chaque appel est autoportant et son effet ne dépend que de l'état des ressources au moment de l'appel.
- Cacheable : Des serveurs intermédiaires peuvent mettre en cache les réponses.
- Layered system : Un client ne peut pas dire s'il est connecté directement au serveur final ou à un serveur intermédiaire. Les serveurs intermédiaires peuvent améliorer l'extensibilité du système en mettant en place une répartition de charge et un cache partagé. Ils peuvent aussi renforcer les politiques de sécurité
- Vision Ressource : Les informations retournées sont des ressources facilement identifiable, et qui comportent toutes les informations nécessaires pour le resquester.

« Keep It Stupid Simple »

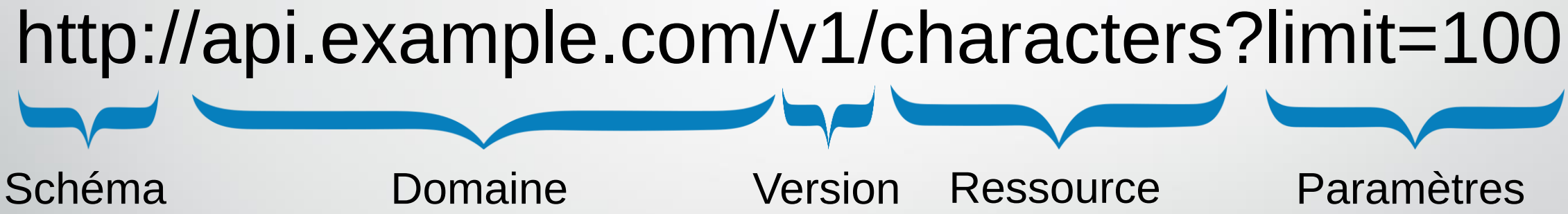
Rest

Méthode HTTP	Rôle
GET	Récupérer une ressource ou une collection
POST	*Créer une ressource*
PUT	*Mettre à jour une ressource*
DELETE	*Supprimer une ressource*
Plus exotique mais existe tout de même	
HEAD	*Récupérer seulement le Header de la réponse*
PATCH	*Mise à jour partielle de la ressource*
OPTION	*Demande d'information sur les options de communications*

Rest

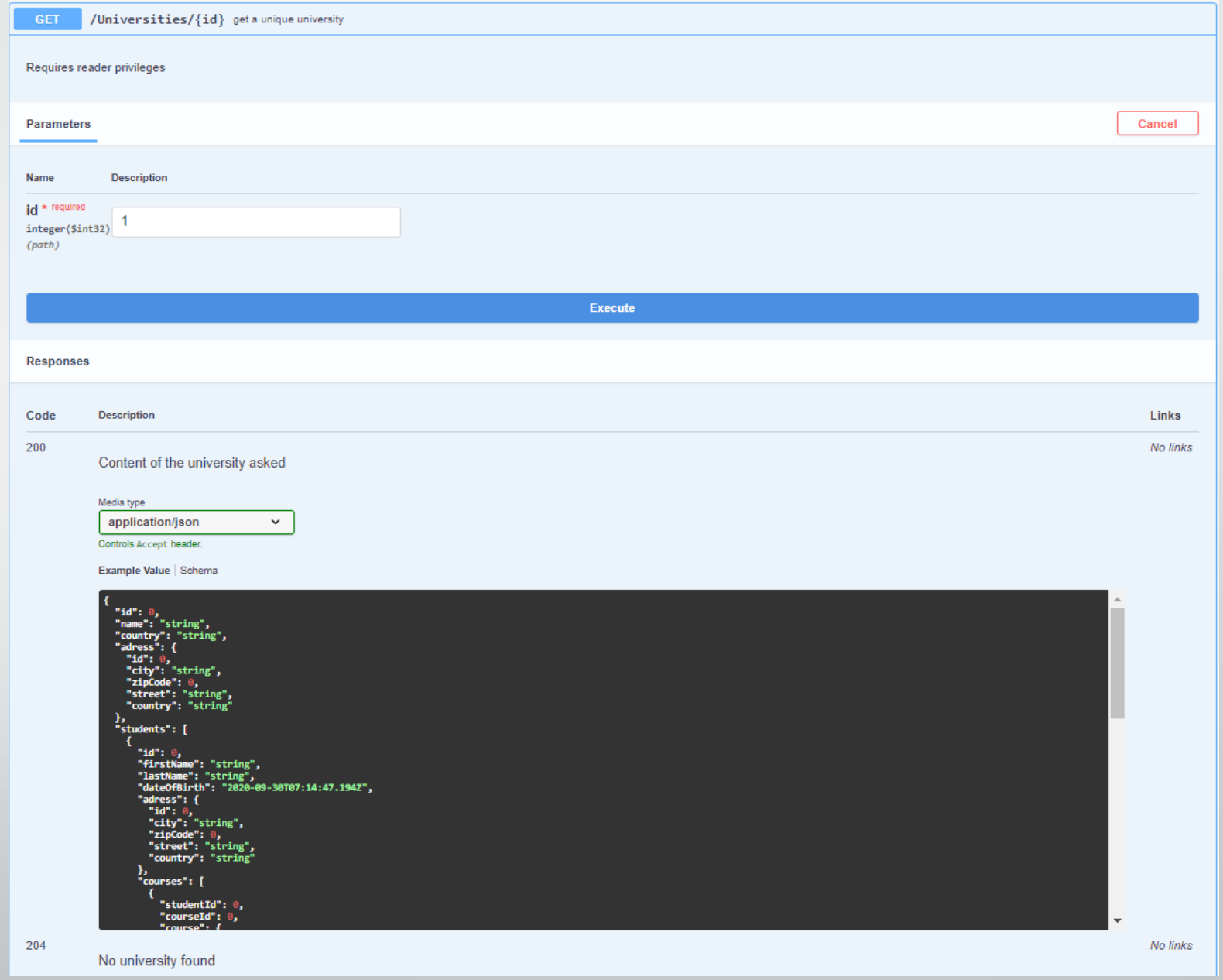
☺	Bad request	Error of the server
200 ok	400 bad request	500 internal server error
201 created	403 forbidden	502 Bad Gateway
202 accepted	404 not found	
303 see other	406 not acceptable	
304 not modified	415 unsupported media type	
	429 too many request	

Rest


http://api.example.com/v1/characters?limit=100
Schéma
Domaine
Version
Ressource
Paramètres

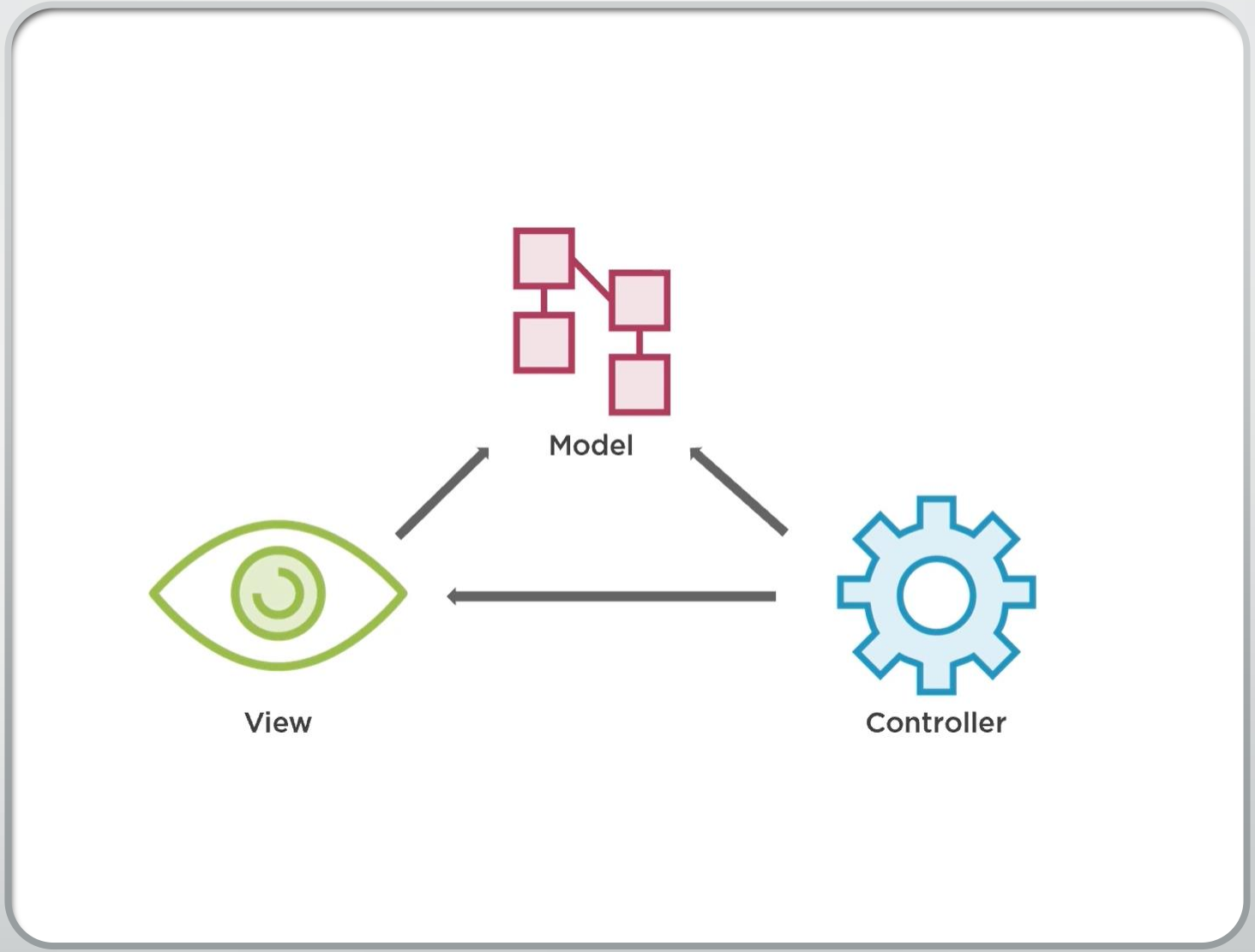
Rest - Documentation

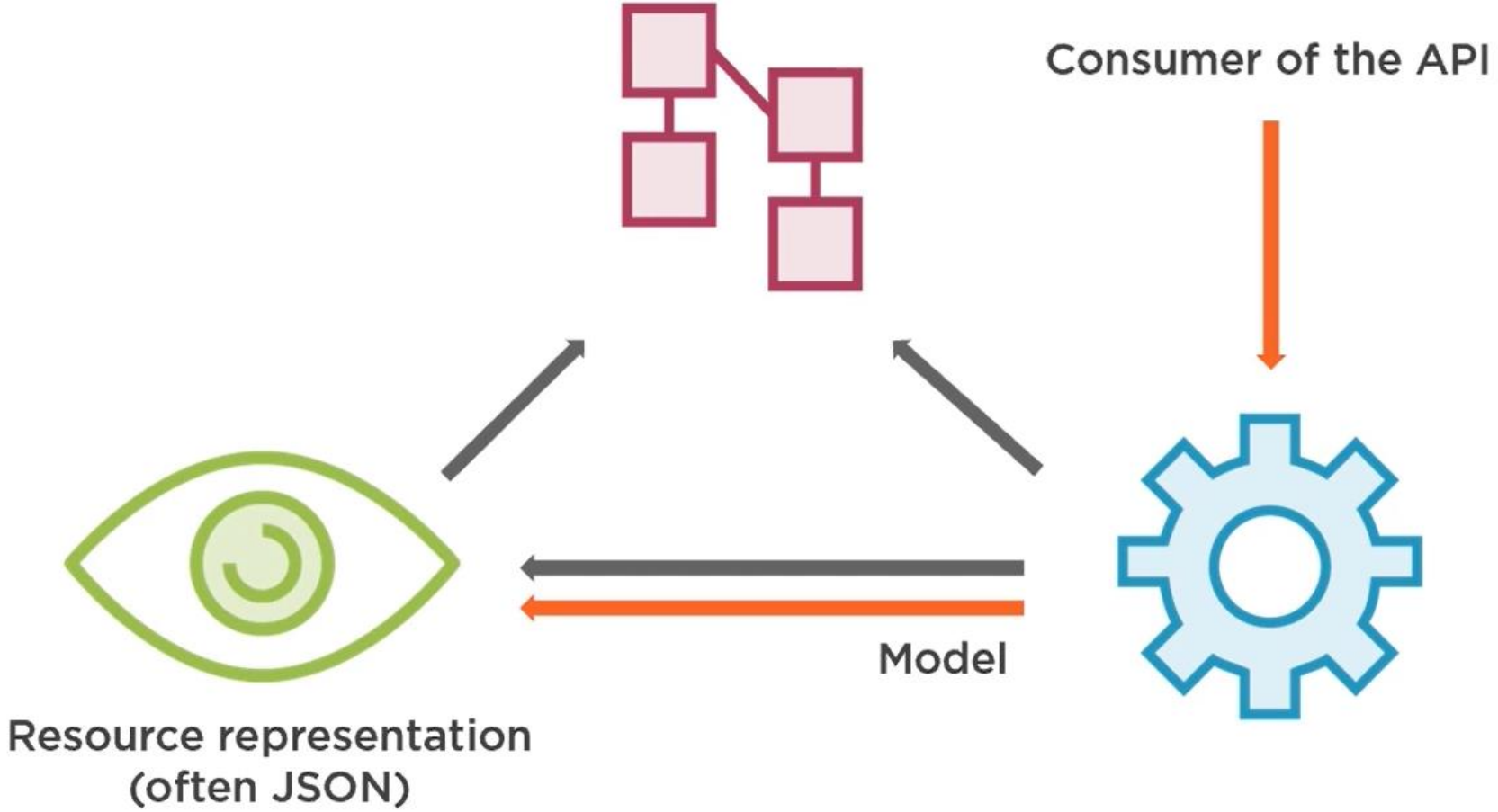
- Via Swagger (TP)



NetCore API - MVC Pattern

- Model-View-Controller est un pattern d'architecture pour implémenter les interfaces utilisateurs
- Encourage le loose coupling et SoC (Separation of concern)
- Ce n'est pas un pattern d'architecture d'application complète




Consumer of the API
Model
Resource representation
(often JSON)

- example tp visual studio