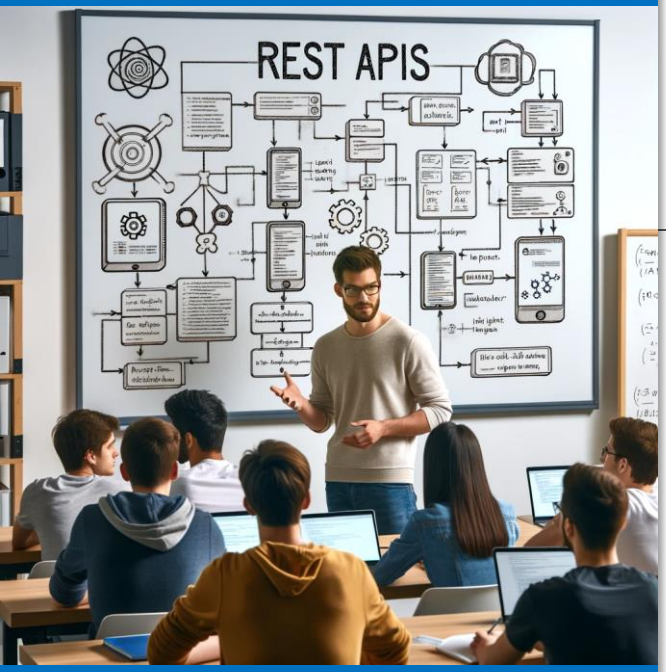


API

Présentation Concepts

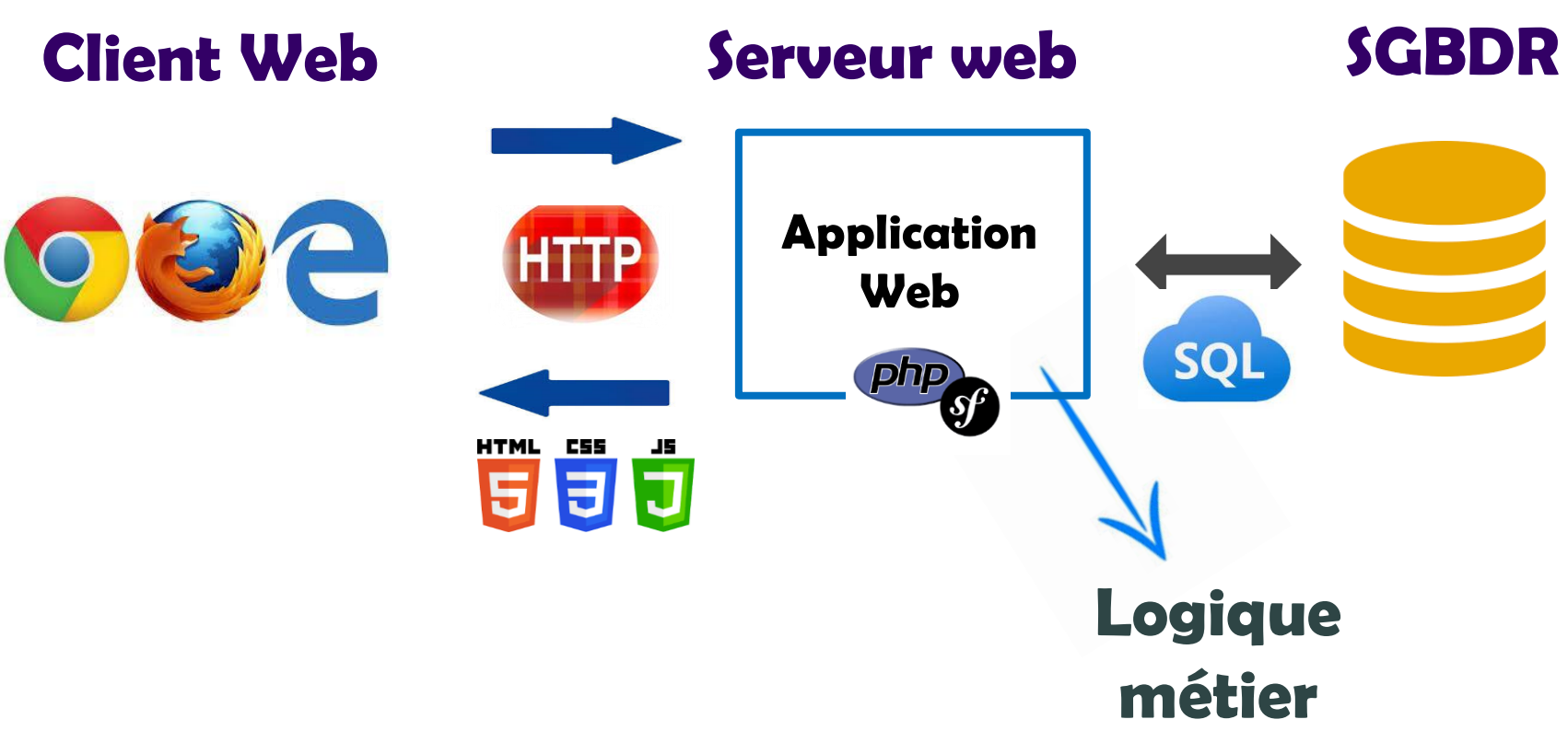
Franck LAMY – 2023/2024 – BTS SIO2

Don't Worry

Be API !

Introduction

Développement "standard"



Exemple

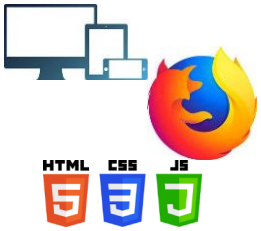
Le site web d'une bibliothèque municipale

Marie (user)

SGBDR

Serveur web

Site Web biblio



Pierre (admin)

Stockage de fichiers (images, pdf, ...)

Exemple

Le site web d'une bibliothèque municipale

Ajouter un livre

Réserver un livre

Commenter un livre

Lister les livres

Consulter un livre

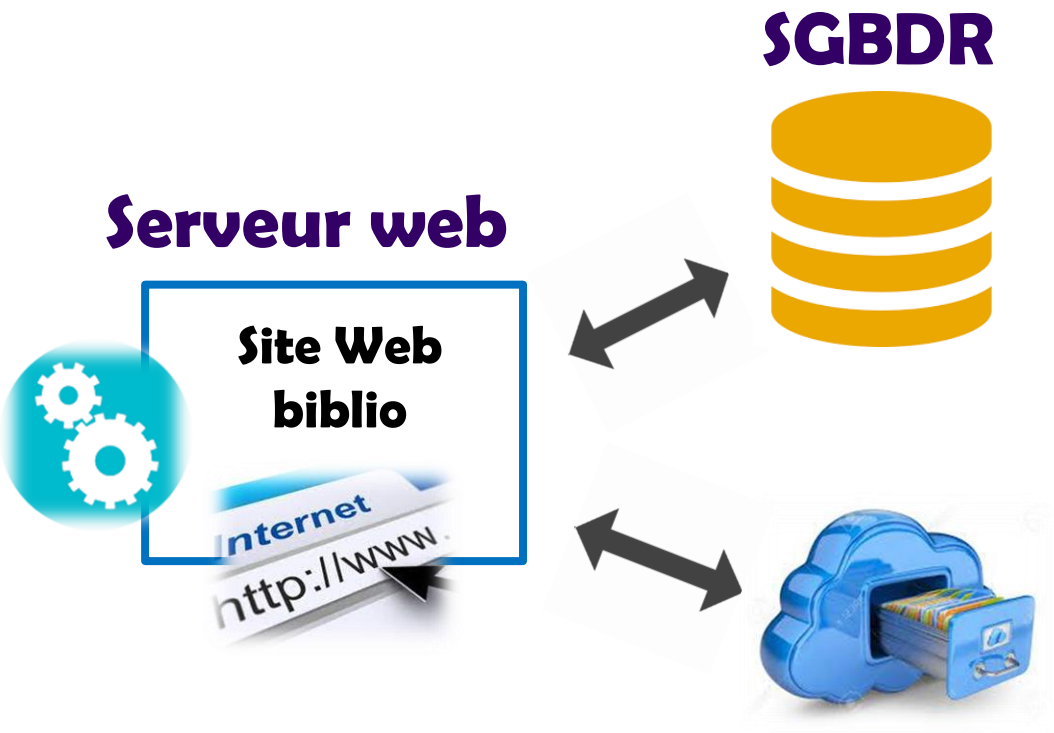
Lister les auteurs

Modifier une réservation

Modifier un livre

Supprimer un livre

Supprimer un commentaire



Exemple

Le site web d'une bibliothèque municipale

Ajouter un livre

Réserver un livre

Commenter un livre

Lister les livres

Consulter un livre

Lister les nouveautés

Modifier une réservation

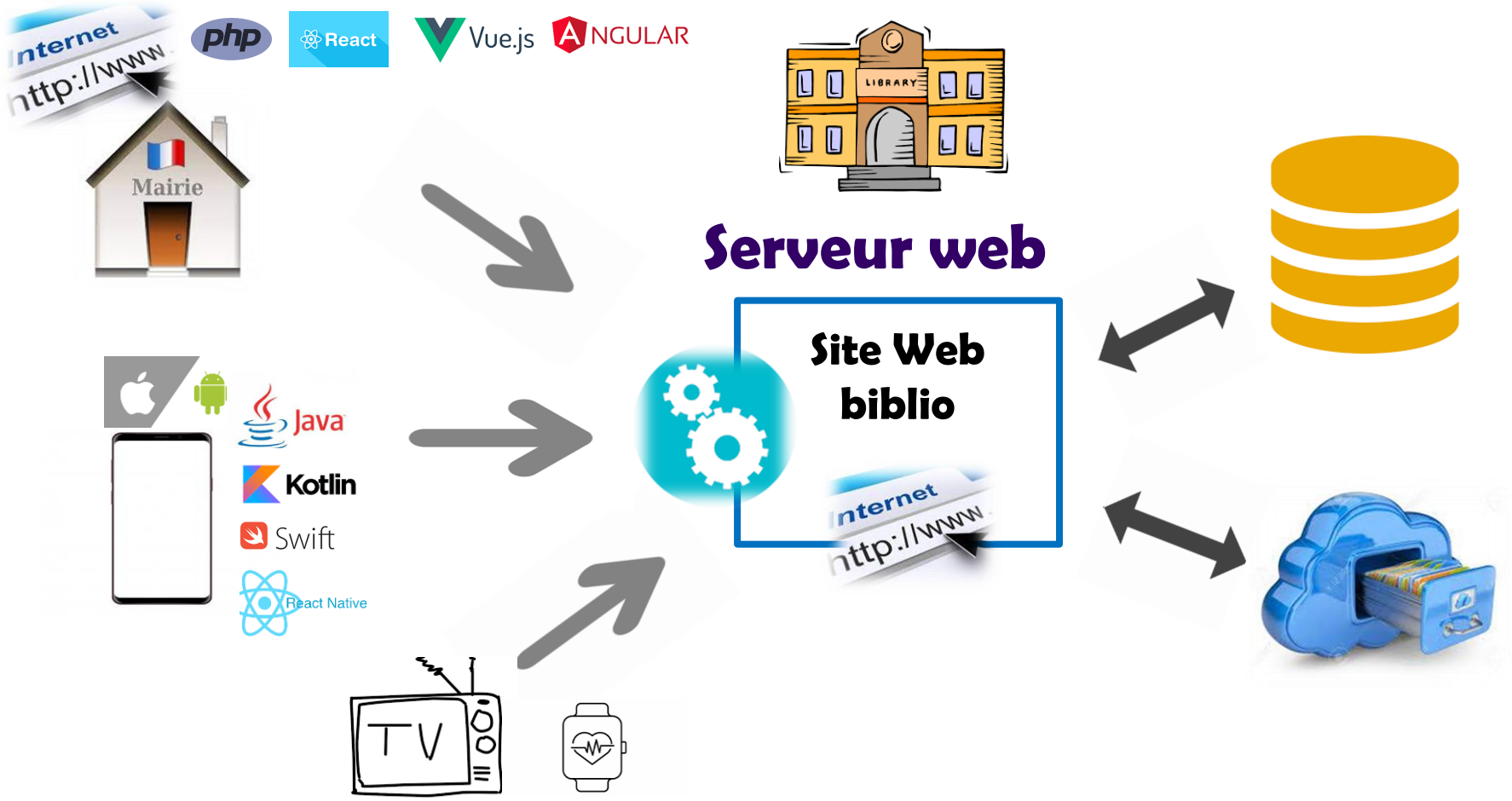
Modifier un livre

Supprimer un livre

Supprimer un commentaire

Evolution !

Utilisation "externe" des fonctionnalités du site



Problème !

- Le site génère du **HTML** : uniquement **interprétable** par un **navigateur**
- **Impossible d'exploiter du HTML** dans une application développée dans un langage de programmation quelconque

Problème !

RAPPEL

HTML : contient les données mais pas que ... !

Mise en situation !

La mairie souhaite intégrer dans son site une fonctionnalité permettant de lister les nouveautés de la bibliothèque municipale

Lister les nouveautés

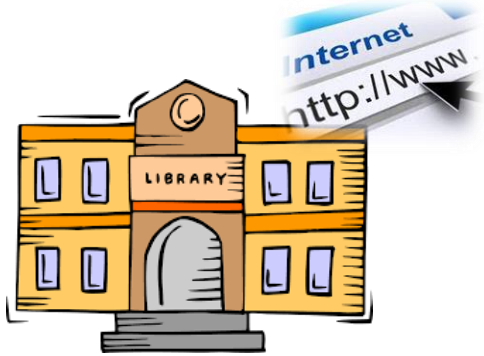
Serveur web

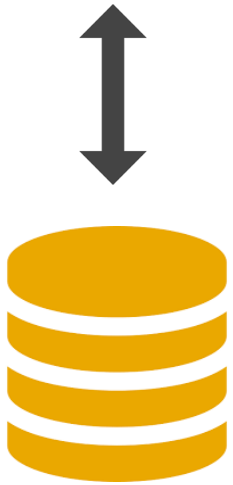
Problème !

Lister les nouveautés

Paul (dev mairie)

Jules (dev biblio)





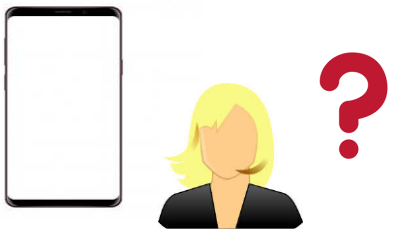
Mise en situation !

La bibliothèque souhaite mettre à disposition de ses utilisateurs une application mobile pour Android et IOS reprenant en partie les fonctionnalités métier proposées par le site WEB

android Java Kotlin

Jules (dev biblio)

iOS Swift



Sophie (dev biblio)

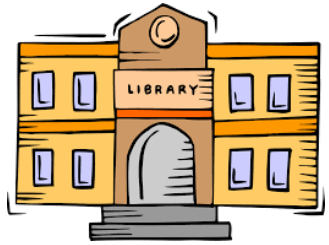
HTTP

HTML 5

Serveur web

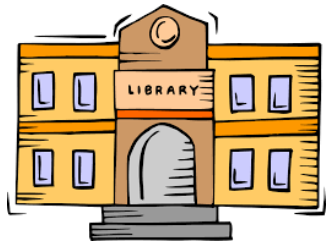
Problème

Il va falloir que je développe l'application **ANDROID** en **Java** de A à Z.
Je vais utiliser la base de données existante !
Par contre cela va prendre beaucoup de temps !



Jules (dev biblio)

MOI AUSSI Jules il va falloir que je développe l'application **IOS** en **Swift** de A à Z.
Je vais également utiliser la base de données existante !
Par contre cela va aussi prendre beaucoup de temps !



Sophie (dev biblio)

Réflexion...

Au final, les 2 applications sont identiques en termes de fonctionnalités métier

Seule l'IHM est différente !

Comment faire pour ne pas développer plusieurs fois les mêmes fonctionnalités ?

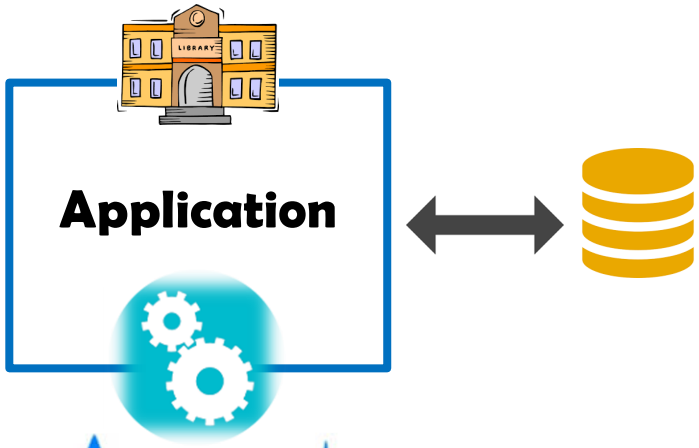
Tu as raison Jules ! Comment faire pour partager des fonctionnalités métier ?

As-tu une solution ?

Solution...

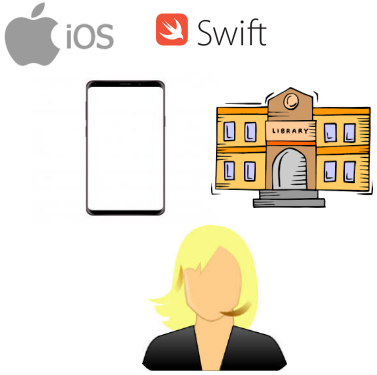
Fonctionnalités centralisées

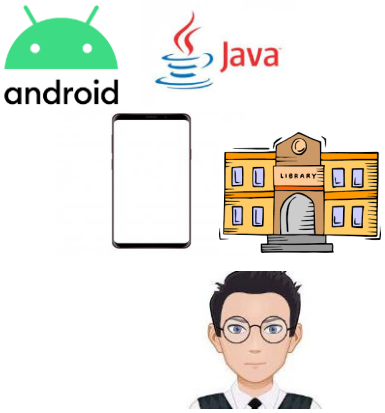
ACCESSIBLE



Requête : fonctionnalité

Réponse : données





Paul (dev mairie)

Application WEB

Sophie (dev biblio)

Application IOS

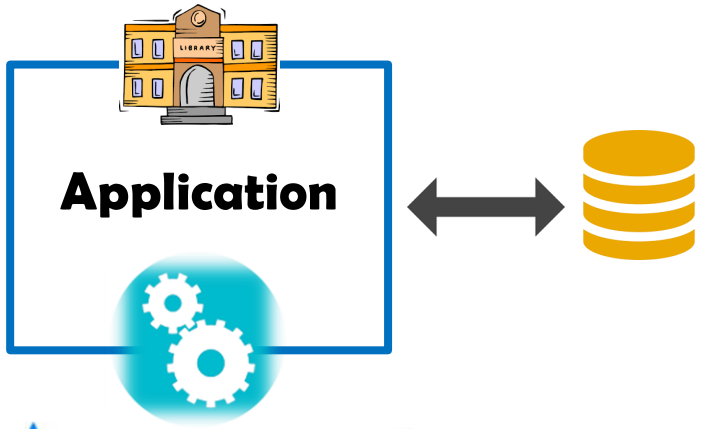
Jules (dev biblio)

Application Android

Solution

Fonctionnalités centralisées

ACCESSIBLE

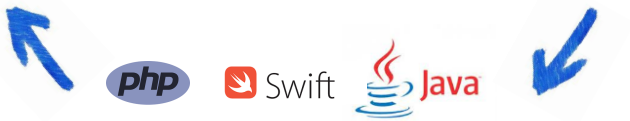


Requête : fonctionnalité

Réponse : données

Appeler la fonctionnalité

Récupérer et interpréter les données



Application

Solution, oui mais ...

Fonctionnalités centralisées

ACCESSIBLE

Application

Protocole de communication

Requête : fonctionnalité

Réponse : données

Identification
Localisation

Appeler la fonctionnalité

Récupérer et interpréter les données

Format

Application

Standard

Besoin d'un standard

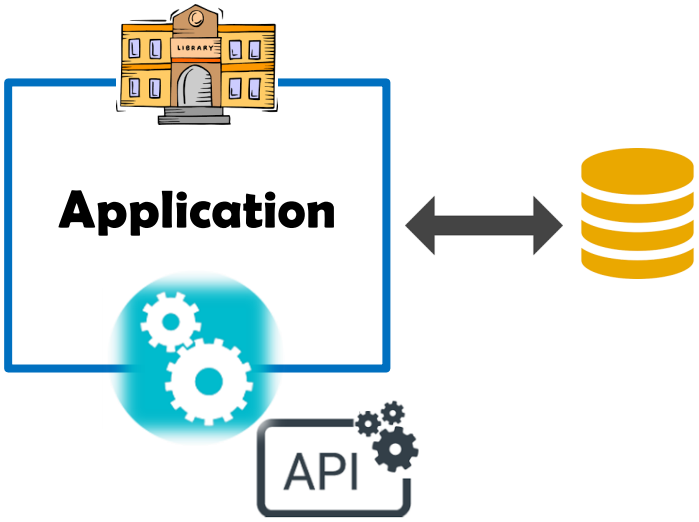
Solution API

Créer une API !

API

Fonctionnalités centralisées

ACCESSIBLE

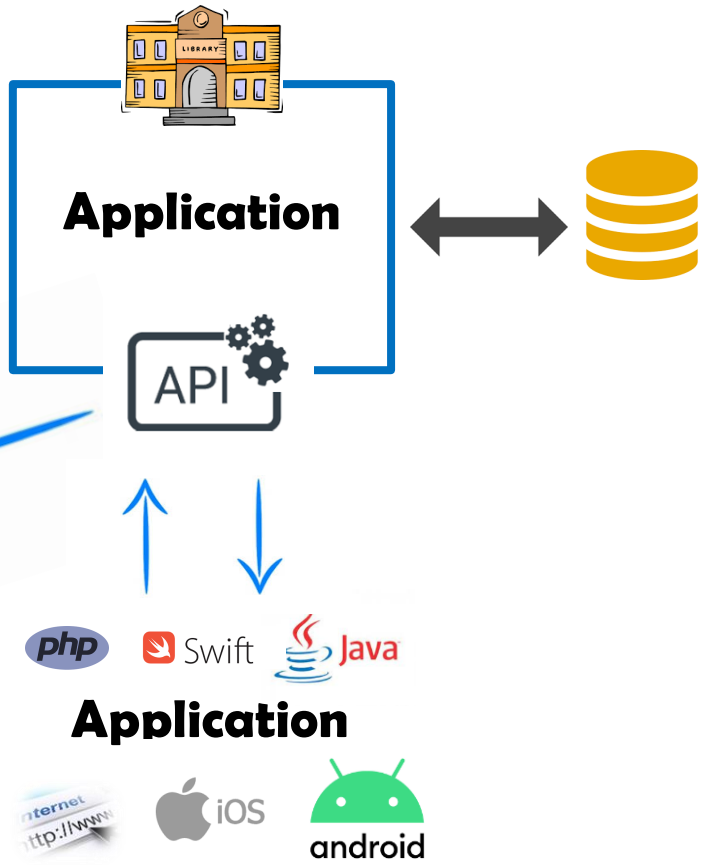


Les **fonctionnalités** sont exposées (accessibles) via une **API**

API : Application Programming Interface

API - Définition

C'est un moyen de communication standardisé entre 2 applications (locales ou distantes)



API - Architecture

Plusieurs **types d'API**, chacun proposant une **architecture** particulière

Simple **O**bject
Access **P**rotocol

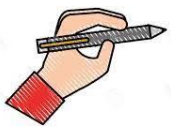
REpresentational
State **T**ransfer

API – REST - Définition

REpresentational **S**tate **T**ransfer

Propose une **architecture** permettant de définir :

- le **protocole de communication** entre les 2 applications
- la **localisation** des fonctionnalités sur l'application qui les expose
- le **format d'échange** des données



API REST : API RestFul

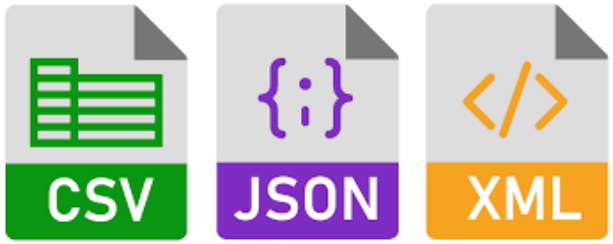
API – REST - Architecture

{ REST }

Protocole de communication

Localisation des fonctionnalités

Format des données (texte)



API – REST - Ressources

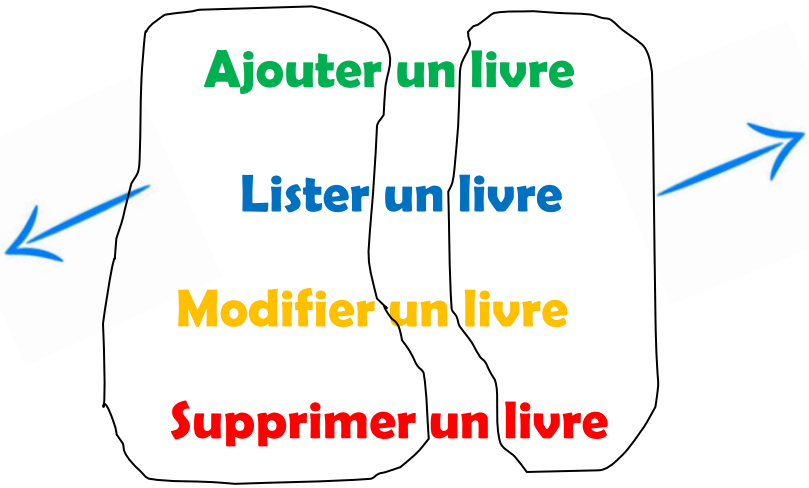
REST repose sur la notion de ressources

API – REST - Ressource

{ REST }

Une fonctionnalité

Actions que l'on peut réaliser sur le livre



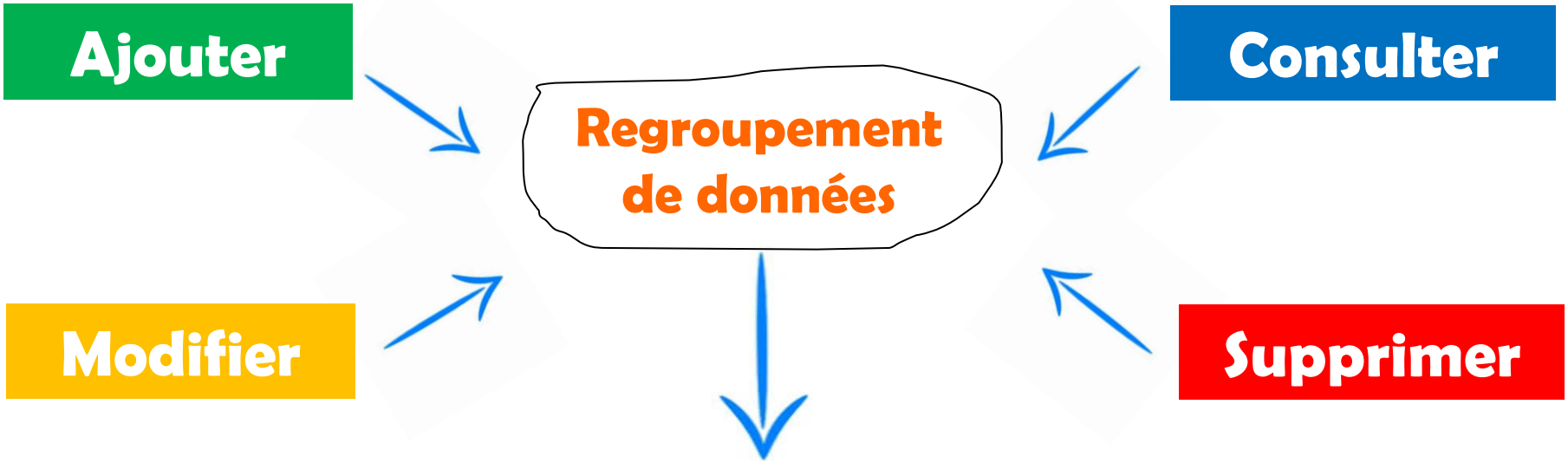
Livre
regroupement de données

API – REST - Ressource

Une **fonctionnalité** : **action** que l'on peut réaliser sur un **regroupement de données**

API – REST - Ressource

Une **fonctionnalité** : ***action*** que l'on peut réaliser sur un **regroupement de données**



RESSOURCE

API – REST - Ressource

Une fonctionnalité est une action que l'on peut réaliser sur une ressource

API – REST - Ressource

{ REST }

Une ressource regroupe des données et possède un nom

ENTITE

API – REST - Ressource

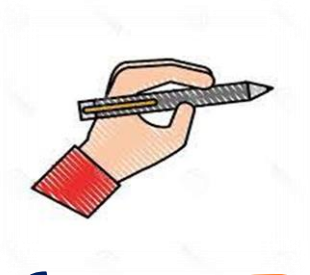
API REST : application qui permet d'exposer à d'autres applications des ressources et le moyen d'agir sur ces ressources (réaliser des actions)

Le protocole HTTP

API – REST - URI

{ REST }

- Une ressource doit être identifiable et unique
- On associe à chaque ressource un identifiant appelé URI



URI

Uniform Resource Identifier

API – REST - URI

{ REST }

URI : Uniform Resource Identifier

EXAMPLES

/livres/123

Ressource

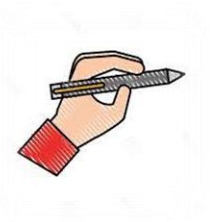
le livre d'id 123

/livres/{id}

API – REST - URL

Pour accéder complètement à une ressource, l'URI ne suffit pas

On a besoin de connaître son emplacement (sa localisation)

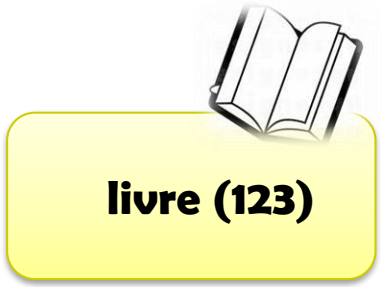


URL

Uniform **R**esource **L**ocator

API – REST - URL

URL : Uniform Resource Locator



EXAMPLES http(s)://biblio.fr/livres/123

- Le protocole de communication
- Le nom de domaine
- URI de la ressource

API – REST - URL

URL : Uniform Resource Locator

livre
(id)

ressource

http(s)://biblio.fr/livres/{id}

API – REST - ENDPOINT

{ REST }

URL : Uniform Resource Locator

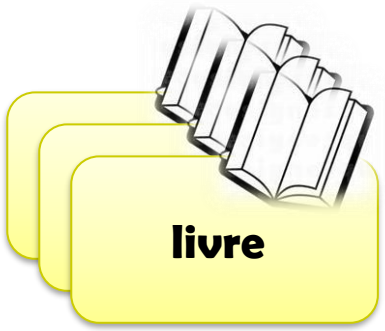
EXAMPLES

http(s)://biblio.fr/livres/{id}

endPoint

API – REST - Collection

Les **ressources** sont **regroupées** dans des **collections**



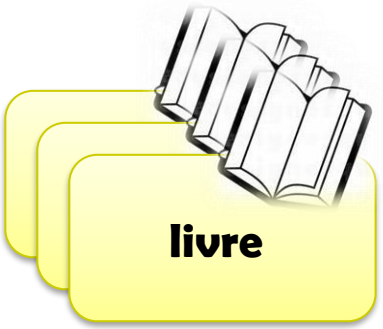
Collection des livres

Collection des clients

API – REST - Collection

Identification d'une collection de ressources

URI : Uniform Resource Identifier



/livres

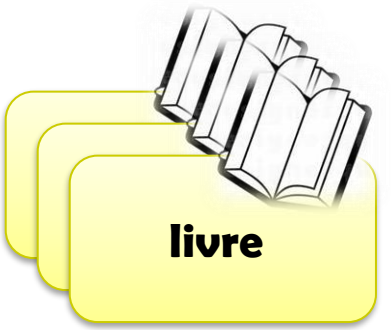
URI de la collection

API – REST - Collection

{ REST }

Localisation d'une collection de ressources

URL : Uniform Resource Locator



EXAMPLES

http(s)://biblio.fr/livres

endPoint

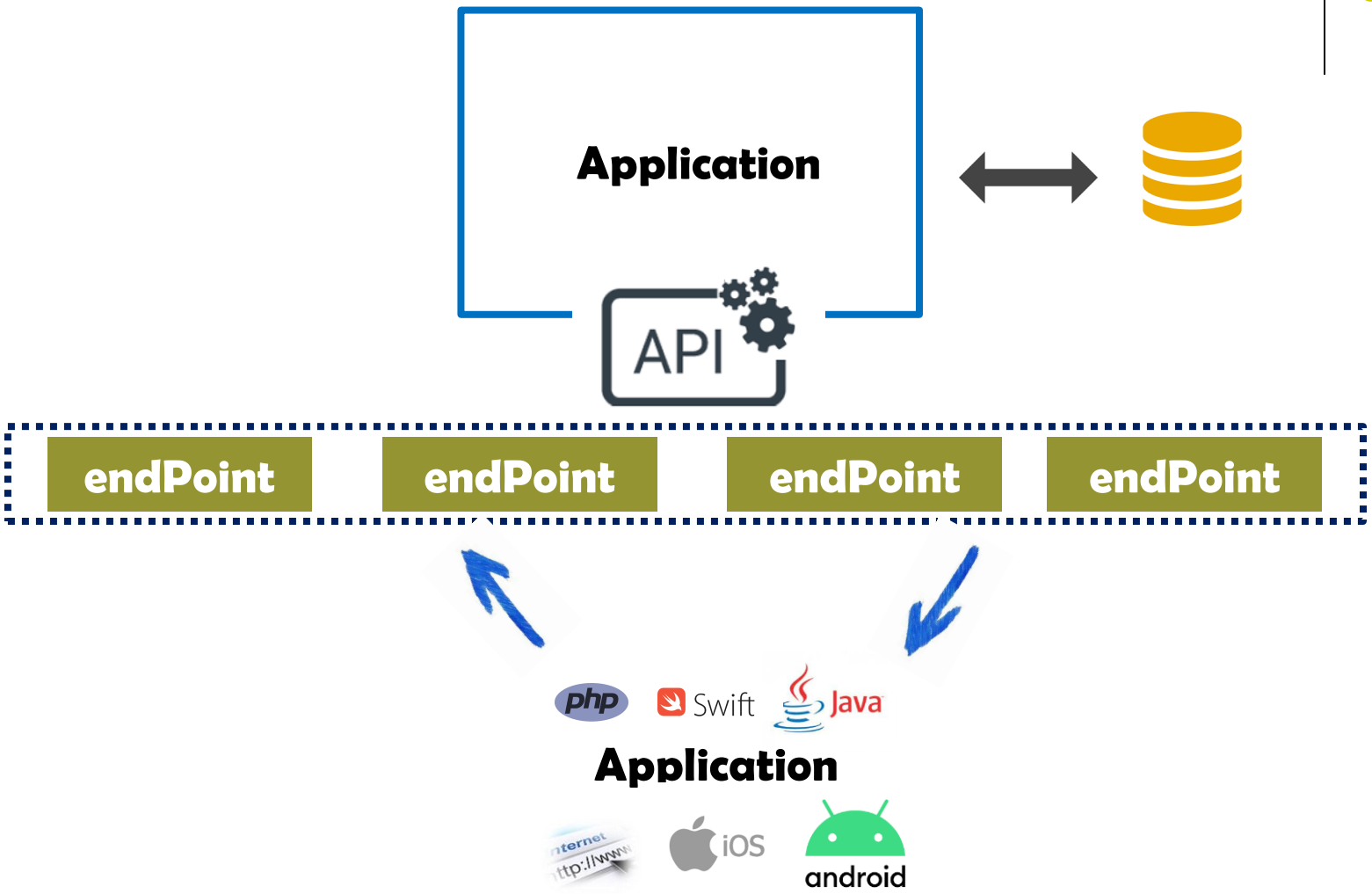
API – REST - ENDPOINT

Un **Endpoint** est donc une **URL** permettant d'accéder à une **ressource** (ou une **collection** de **ressources**)

API – REST - ENDPOINT

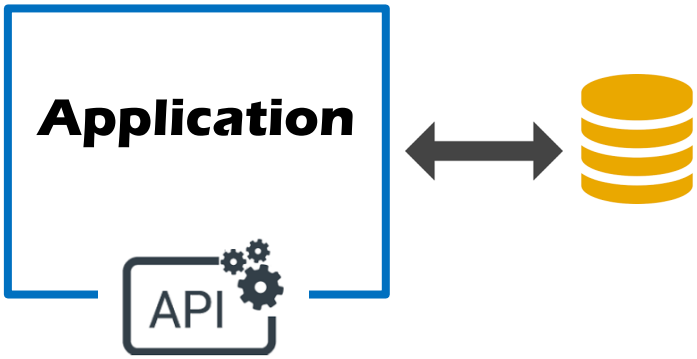
Une API REST expose des endPoints qui représentent des ressources

API – REST - ENDPOINT



API – REST - ENDPOINT

{ REST }



endPoint endPoint endPoint endPoint

http(s)://biblio.fr/livres/123

REQ HTTP

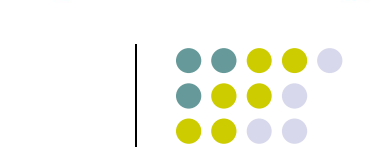
php Swift Java

Application

internet http://www iOS android

Une application va donc pouvoir utiliser l'API à l'aide des endPoints afin d'accéder aux ressources exposées

API – REST - Action



API REST : permet d'exposer à d'autres applications des ressources (via des EndPoints) et le moyen d'agir sur ces ressources (réaliser des actions)

Comment agir sur une ressource ?

API – REST - Action

{ REST }

Ajouter

Consulter

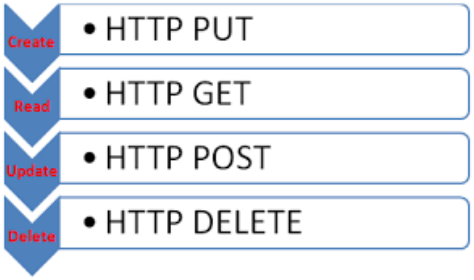
RESSOURCE

Modifier

Supprimer

API – REST – Méthodes HTTP

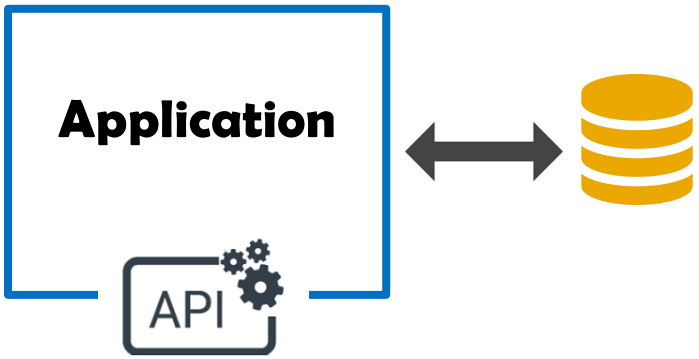
HTTP définit un ensemble de méthodes qui précisent l'action que l'on souhaite réaliser sur la ressource demandée



Action	Méthode HTTP (verbes)
Consulter	GET
Ajouter	POST
Modifier	PUT
Supprimer	DELETE

API – REST – Méthodes HTTP

{ REST }



endPoint | endPoint | endPoint | endPoint

http(s)://biblio.fr/livres/123

REQ

HTTP

Application

Préciser dans la requête la méthode que l'on souhaite exécuter sur la ressource demandée

REQ

DELETE | POST | GET | PUT

API – REST – Requête HTTP

{ REST }

Pour utiliser une API REST, on a besoin d'émettre une requête HTTP en précisant:

- Le endPoint (localisation de la ressource)
- La méthode HTTP (action à réaliser sur la ressource)

Consulter le livre 123

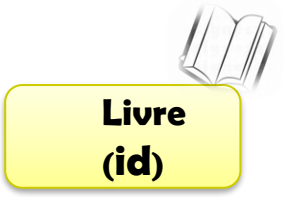
REQ HTTP

GET http(s)://biblio.fr/livres/123

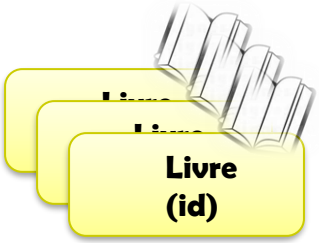
Méthode endpoint

API – REST - Consultation

Pour **consulter** une **ressource** ou une **collection** de **ressources**



GET /livres/{id} → Consulter le livre {id}



GET /livres → Consulter les livres

API – REST - Ajout

Pour **ajouter** une **ressource**

POST /livres → **Ajouter** un nouveau **livre**

Préciser les données de la nouvelle ressource dans la requête HTTP

API – REST - Modification

Pour **modifier** une **ressource**

PUT /livres/{id} → **Modification** totale du **livre** {id}

Préciser les données à modifier de la ressource dans la requête HTTP

API – REST - Suppression

Pour **supprimer** une **ressource**

DELETE /livres/{id} → **Suppression du livre {id}**

API – REST – En général

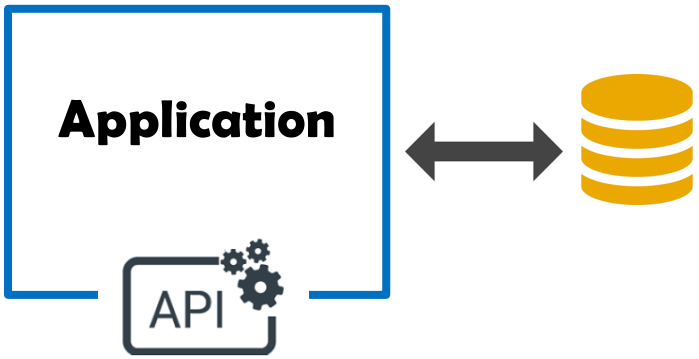
{ REST }

Soit une ressource nommée ressource

GET	/ressources/{id}
GET	/ ressources
POST	/ ressources
PUT	/ressources/{id}
DELETE	/ressources/{id}

Le "S" à la fin du nom de la ressource !

API – REST - Format des données



endPoint | endPoint | endPoint | endPoint

GET http(s)://biblio.fr/livres/123

REQ — HTTP — REP ?

Les données renvoyées par l'API doivent être **interprétables** par l'application qui a émis la requête HTTP

REP

Application

API – REST - Format des données

{ REST }

- La **réponse HTTP** va donc contenir les **données** demandées
- Les **données** doivent être **interprétées** "**facilement**" par l'application qui a émis la **requête**
- Les **données ne doivent pas dépendre** de l'application qui a émis la **requête**

API – REST - Format des données

{ REST }

Les données doivent être exprimées dans un format simple facilement interprétable par

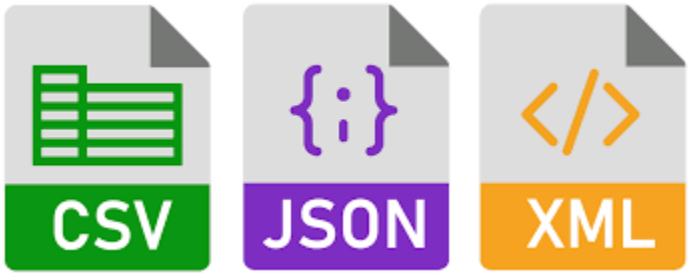
n'importe quelle application

php React Vue.js ANGULAR Swift Java

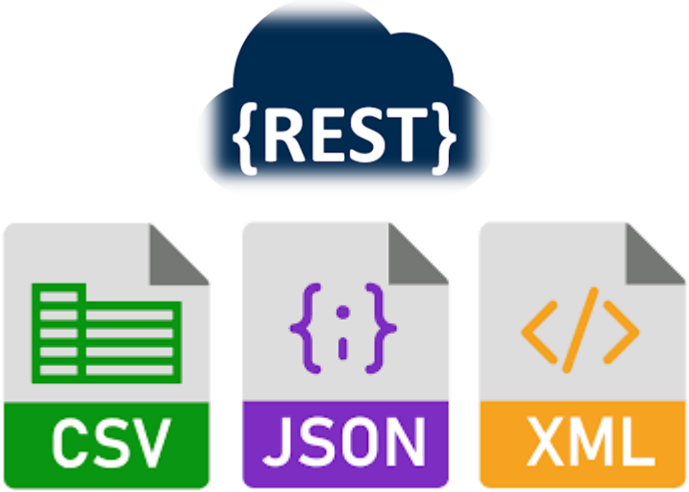
API – REST - Format des données

{ REST }

- Format simple : du texte
- Format facilement interprétable : structuré

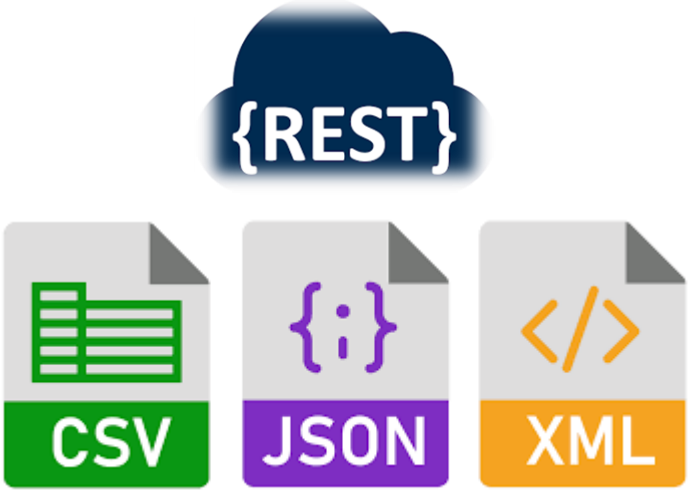


API – REST - Format des données



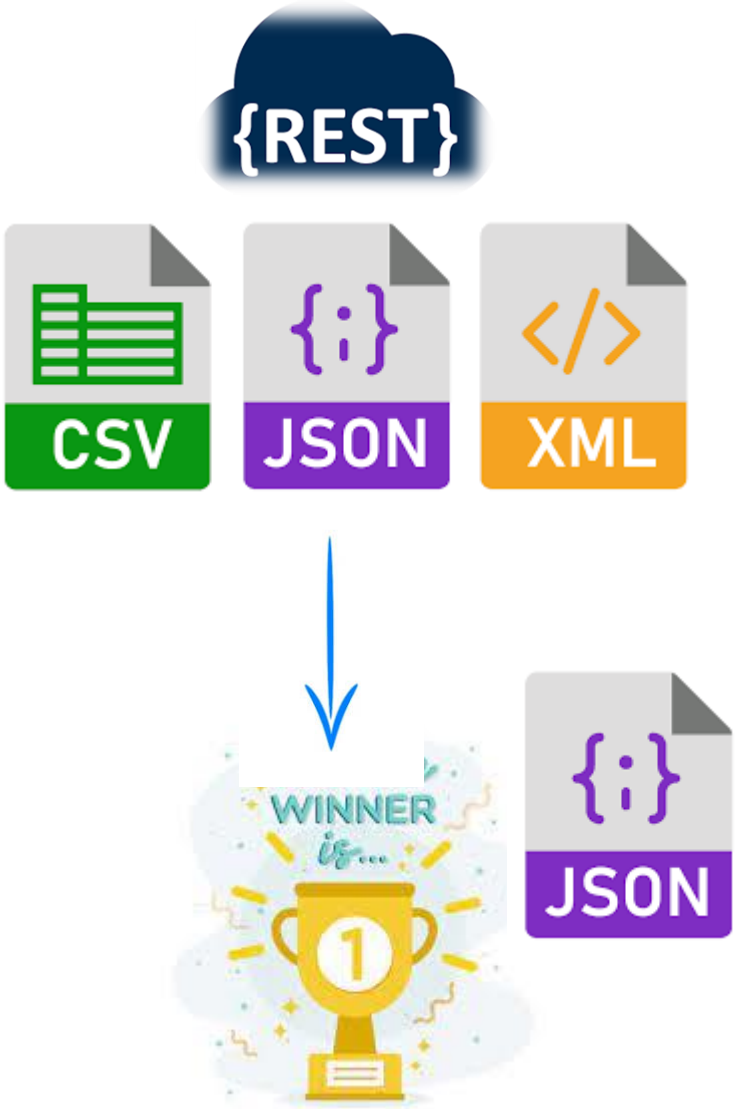
Formats permettant de structurer des données textuelles

API – REST - Format des données

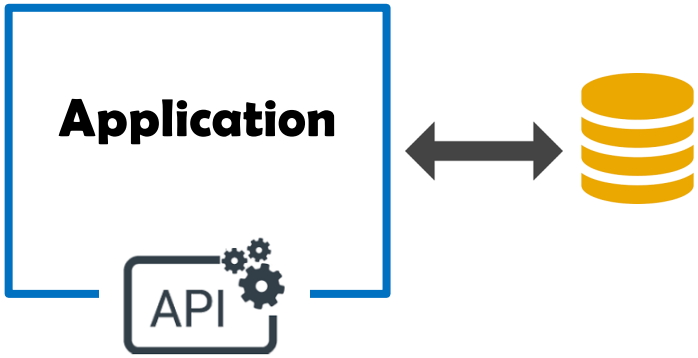


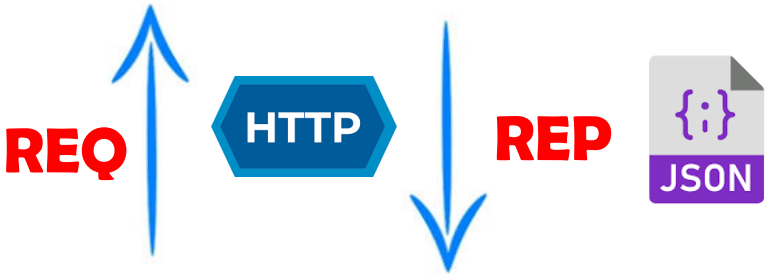
Formats permettant de faciliter l'échange de données entre applications

API – REST - Format des données



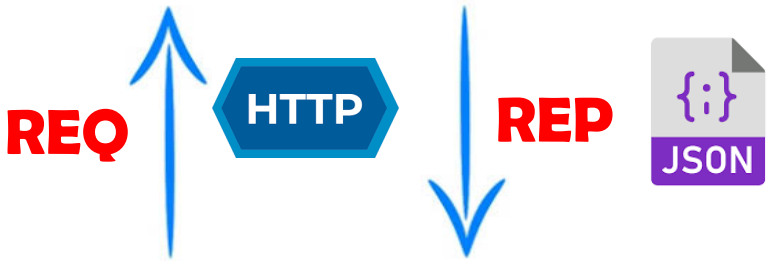
API – REST - Résumé





Application

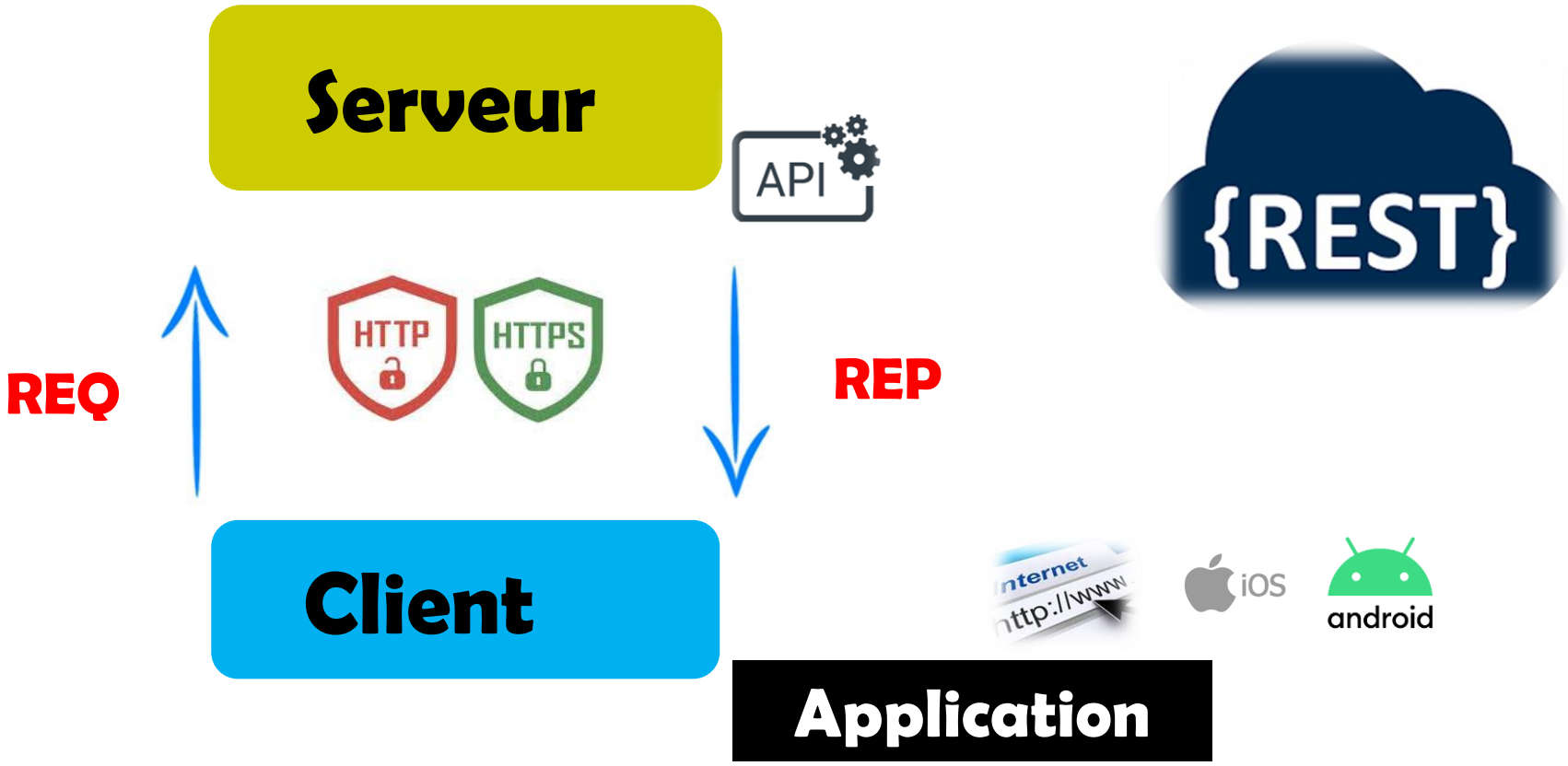
API – REST – Quelques exemples



Application

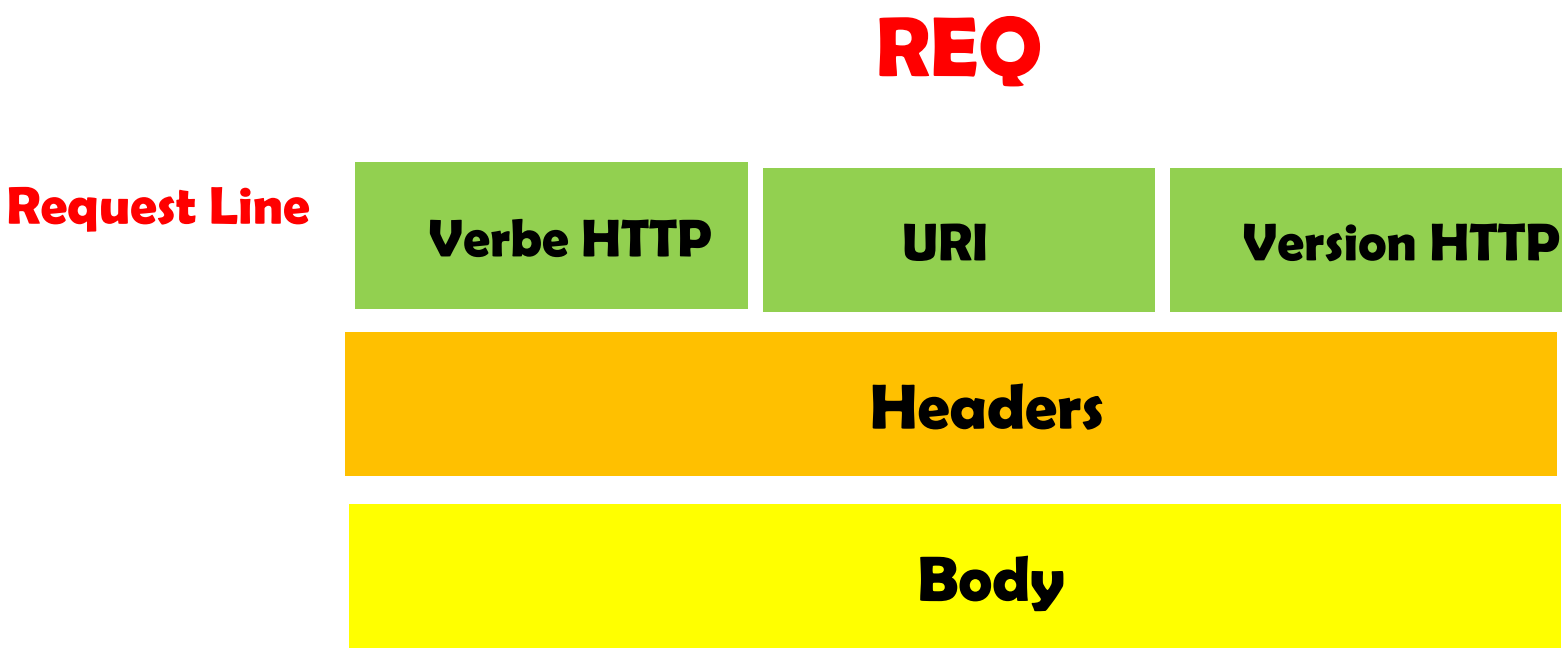
API – REST – Protocole HTTP

Protocole de communication entre un client et un serveur



API – REST – Requête HTTP

Format d'une requête HTTP



API – REST – Requête HTTP

La **Request Line** indique la **méthode HTTP**, l'**URI** de la **ressource** demandée et la **version** du **protocole HTTP**

Verbe HTTP	URI	Version HTTP	CRLF
GET	/livres/123	HTTP/2.0 HTTP/1.1	CRLF

API – REST – Requête HTTP

Les **Headers** précisent les **informations** permettant au **serveur** d'interpréter la **requête** du **client**

Une ligne par entête

Headers

Nom : **valeur** CRLF

Nom de domaine

Host : **biblio.fr** CRLF

Accept : **application/json** CRLF

API – REST – Requête HTTP

Le **Body** contient les éventuelles **données formatées** envoyées par le **client** au **serveur**

Content-type : **application/json** CRLF

Body

Actions concernées POST PUT

API – REST – Requête HTTP

Consulter le livre 123

GET http(s)://biblio.fr/livres/123

REQ **HTTP**

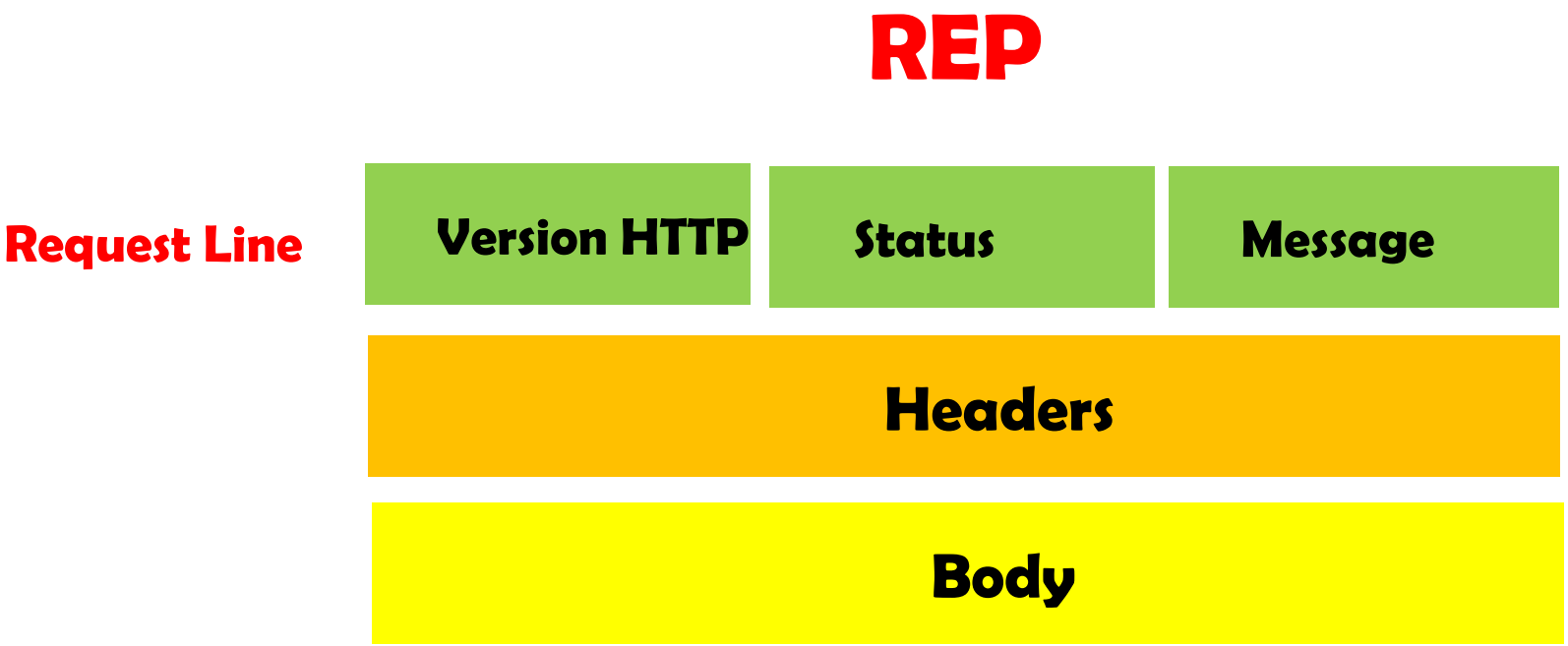
GET /livres/123 HTTP/2.0 **CRLF**

Host : biblio.fr **CRLF**

Accept : application/json **CRLF**

API – REST – Réponse HTTP

Format d'une réponse HTTP



API – REST – Réponse HTTP

La **Request Line** indique la **version** du **protocole HTTP**, le **status** (code) de la **réponse** et un **message** texte correspondant au **status**

Version HTTP	Status	Message	CRLF
HTTP/2.0 HTTP/1.1	200	OK	CRLF

https://fr.wikipedia.org/wiki/Liste_des_codes_HTTP

API – REST – Réponse HTTP

Les **Headers** précisent les **informations** permettant au **client** d'interpréter la **réponse** du **serveur**

Une ligne par entête

Headers

Nom : **valeur** CRLF

Content-type : **application/json** CRLF

API – REST – Réponse HTTP

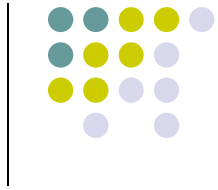
Le **Body** contient les éventuelles **données formatées** renvoyées par le **serveur** au **client**

Content-type	:	application/json	CRLF

Body

API – REST – Réponse HTTP

{ REST }



EXAMPLES

Le livre 123

REQ **GET** http(s)://biblio.fr/livres/123

REP **HTTP**

HTTP/2.0 **200** OK CRLF

Content-type : application/json CRLF

```
{
    id : 123,
    titre : "API-REST pour les nuls",
    nbPages : 310
}
```

JSON

API – REST – REQ/REP

{ REST }

REQ

GET /livres/123 HTTP/2.0 CRLF

Host : biblio.fr CRLF

Accept : application/json CRLF

REP

HTTP/2.0 200 OK CRLF

Content-type : application/json CRLF

```
{
    id : 123,
    titre : "API-REST pour les nuls",
    nbPages : 310
}
```

API – REST – Langages

La plupart des langages de programmation permettent d'utiliser des API REST

API – REST – Langages

Utiliser une API REST

Consommer une API REST