

A thick black L-shaped frame is positioned around the text. It starts at the top left, goes right, then down, then right again, and finally down to the bottom right corner.

# ARCHITECTURES REST ET RESTFUL

Élaborée par : Ben Hdia Souhir  
Chabeen Hajer

# Plan

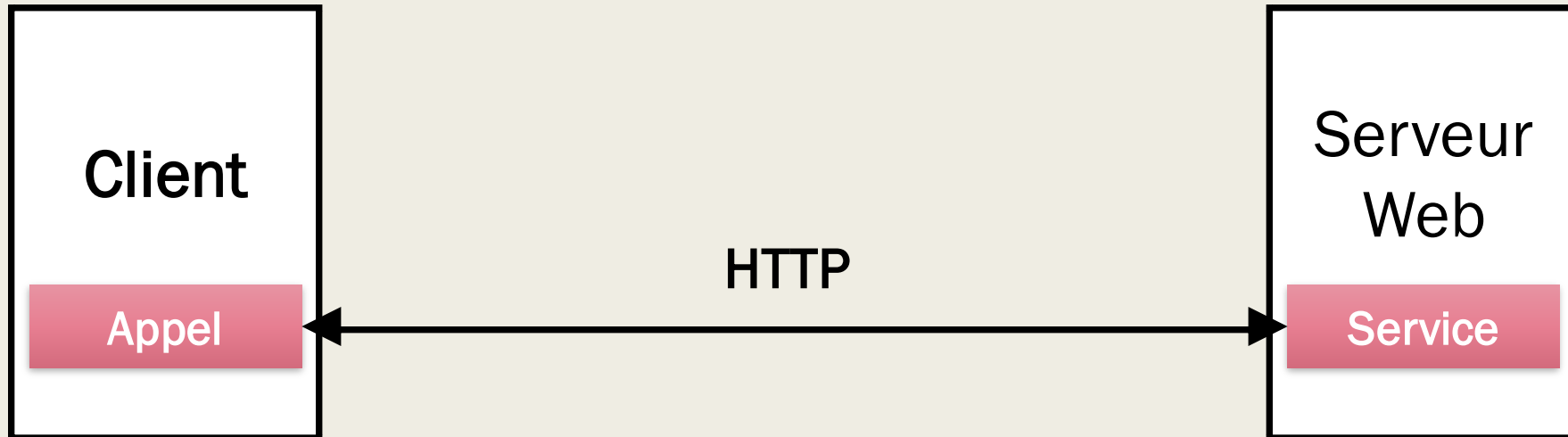
- *Introduction*
- *Web Service REST*
- *Avantages et inconvénients*
- *Développer des Web Services REST Avec Spring Boot*
- *Conclusion*

# Introduction

# Web Service

- Service web est un système logiciel
- Identifier par une URL
- Dont les interfaces publiques et les fixations sont définit et décrit en utilisant XML,JSON

- Un service web est utilise dans une architecture Client/serveur.



# C'est quoi REST ?

# REST

- **REST**(**RE**présentation State Transfert) est un style architecture inspire de l'architecture de web fortement base sur HTTP.
- Ce qu'il est :
  - *Système d'architecture*
  - *Approche pour construire une application*
- Ce qu'il n'est pas :
  - *Un protocole :*
  - *Un format*
  - *Un standard*

# Utilisation

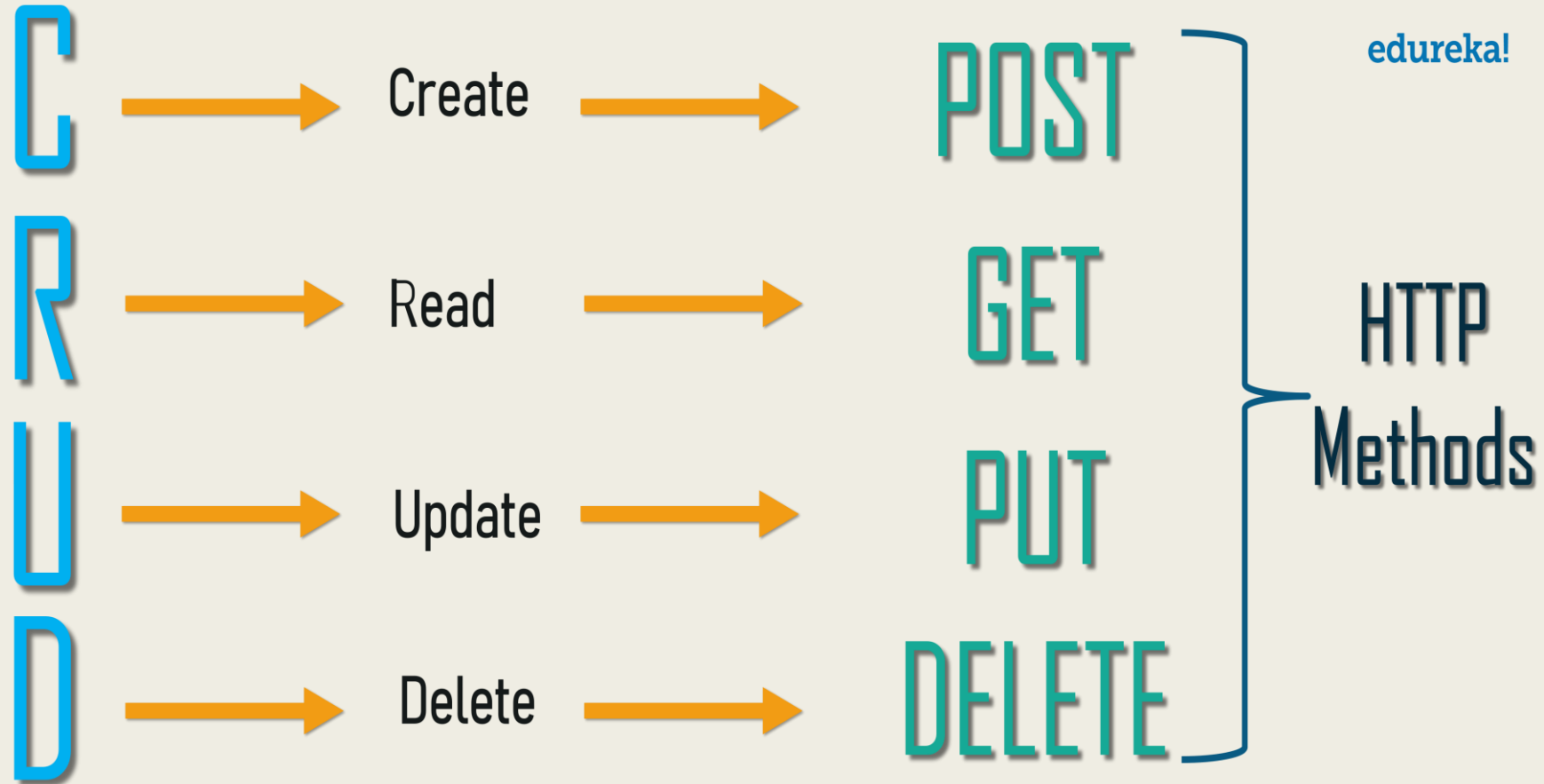
- Utilise dans les développements des application oriente ressource (ROA) ou oriente données (DAO)
- Les applications respectent l'architecture REST sont dits RESTful



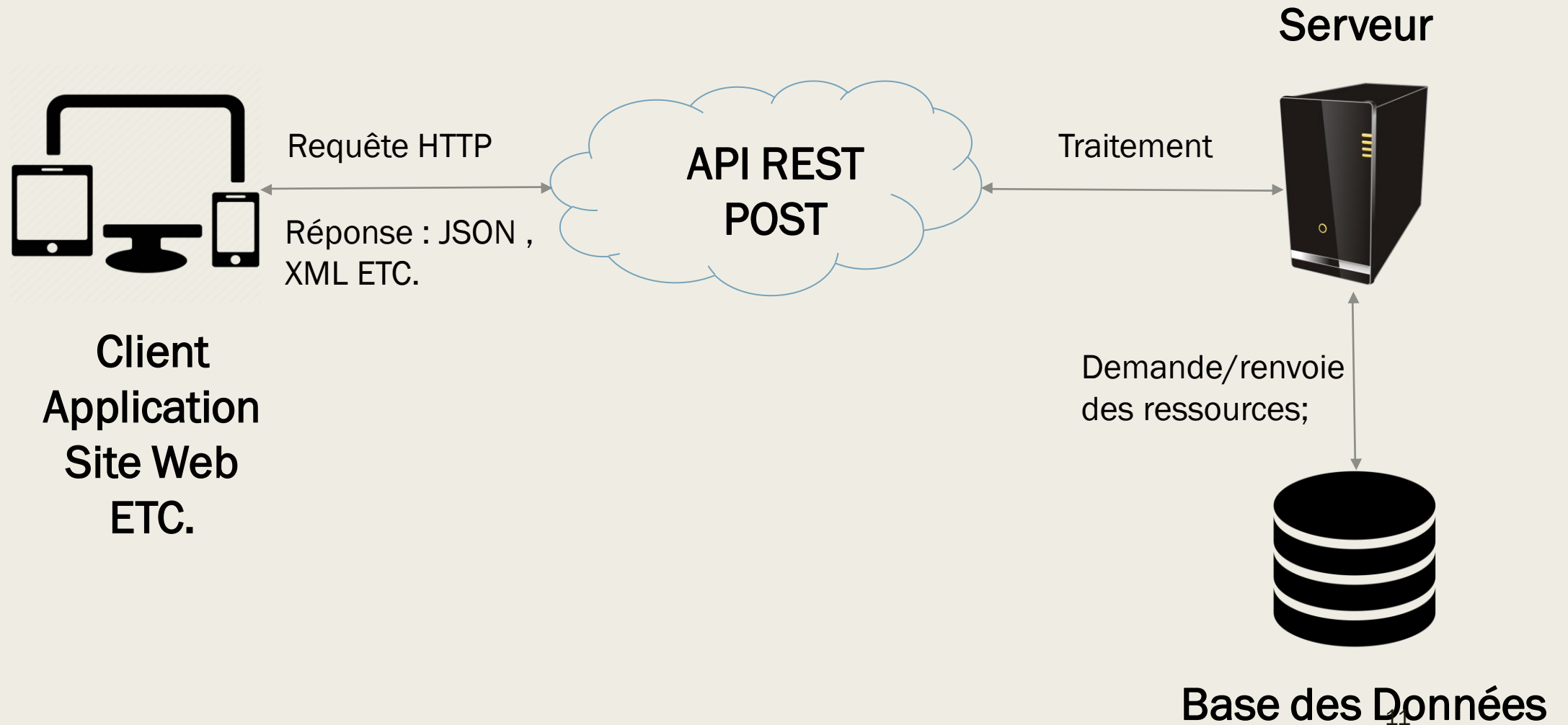
# Les caractéristiques

- Les services REST sont sans états (*stateless*)
- Les architectures RESTful sont construire a partir des ressources uniquement identifiées par URI
- Transférer XML, JavaScript Object Notation (JSON), ou les deux.
- Interface uniforme base sur le méthode HTTP

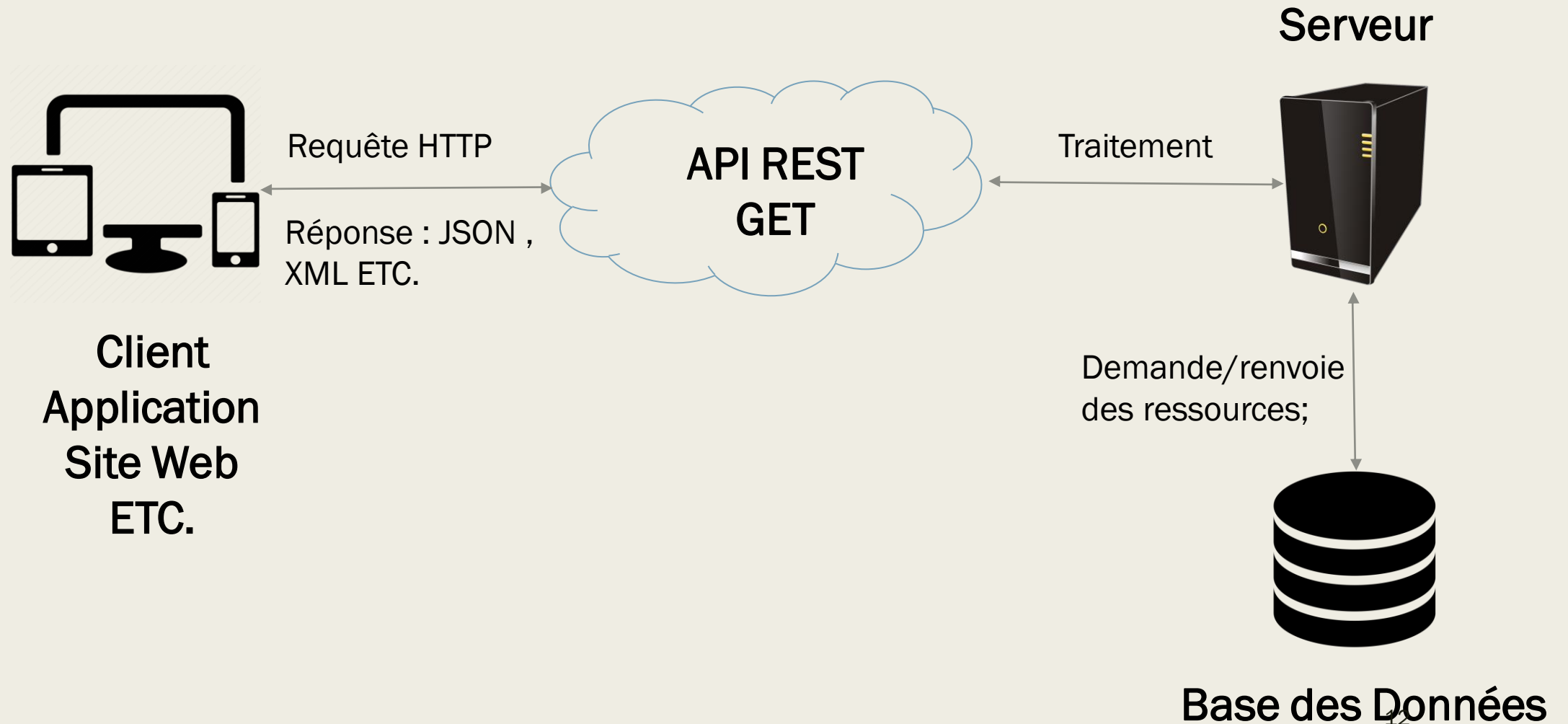
# CRUD



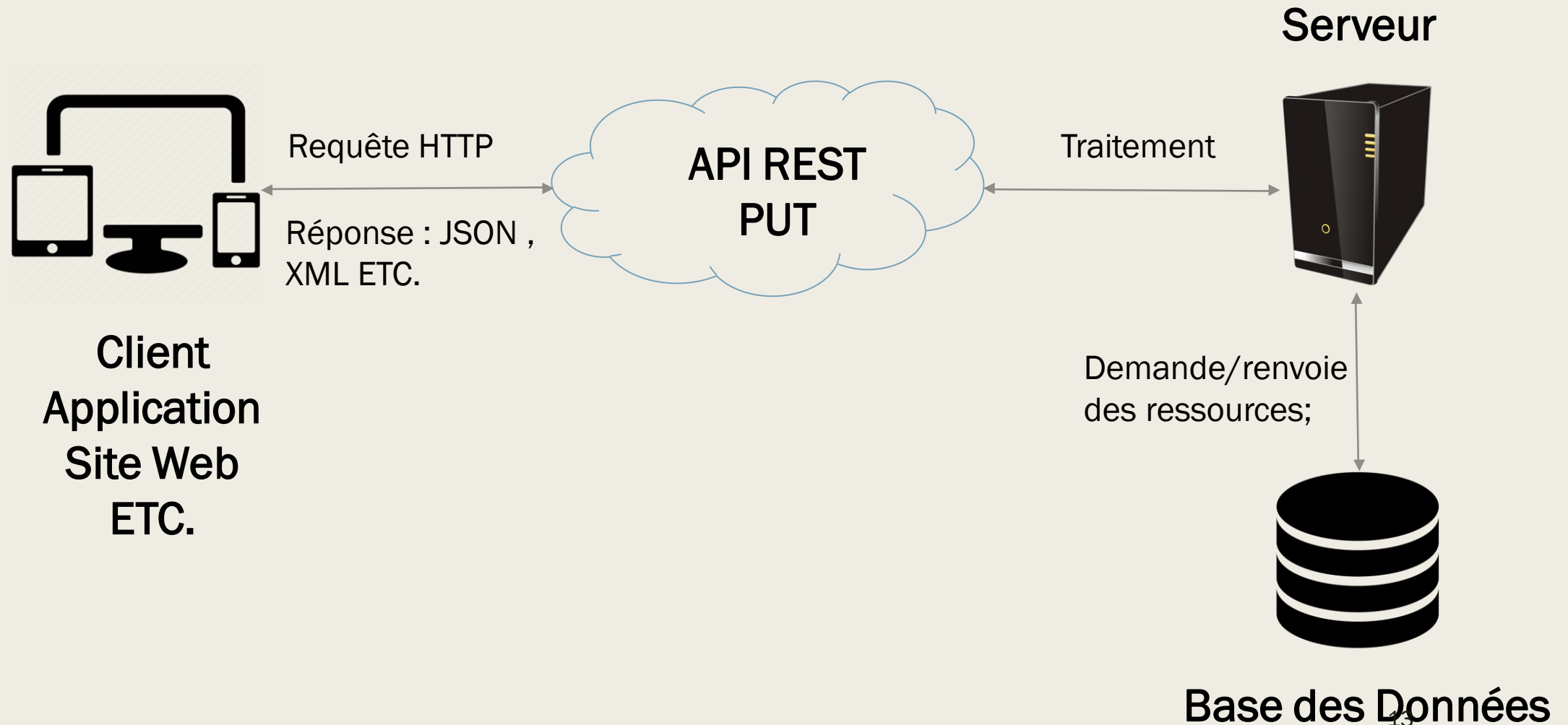
# POST (Create)



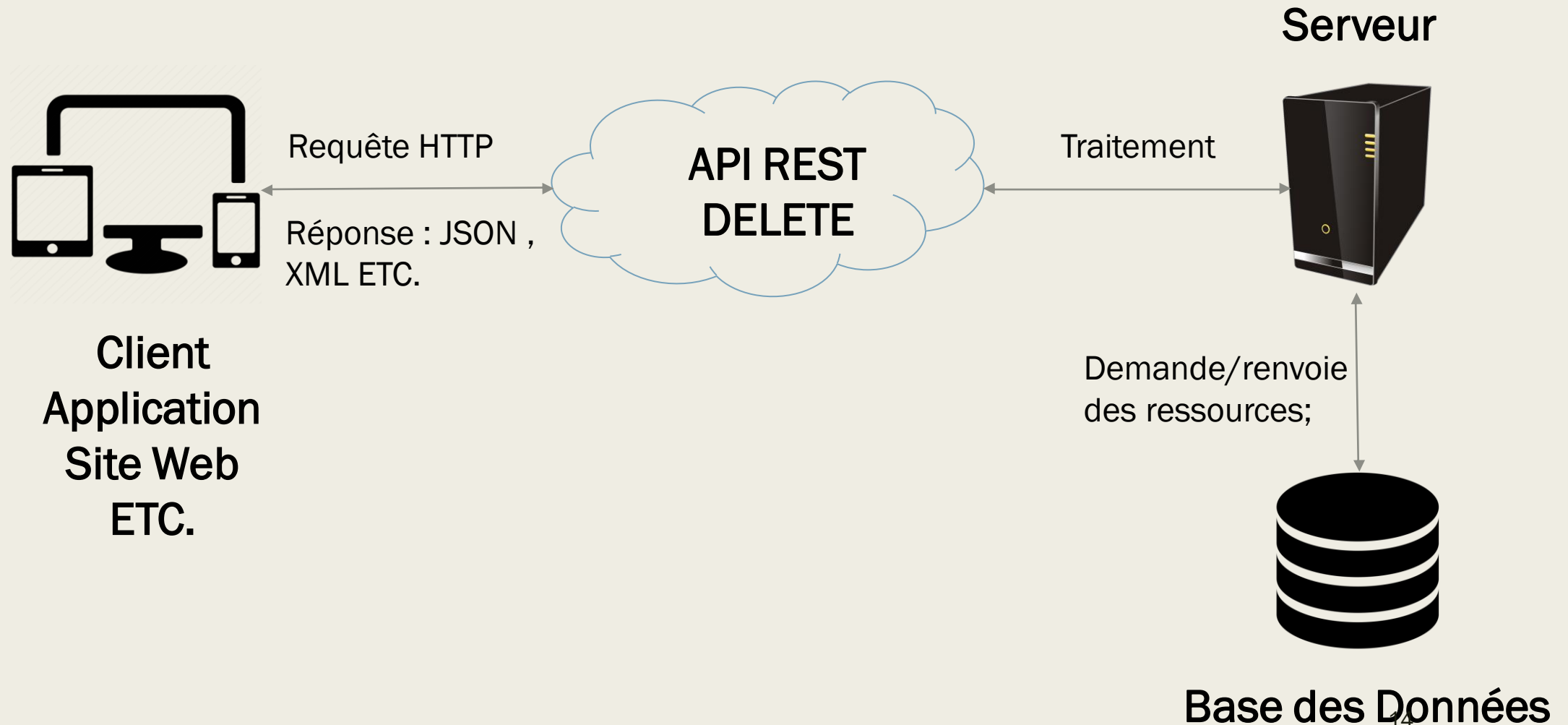
# GET (Read)



# PUT (Update)



# DELETE (Delete)



# Fournisseur



# Avantages et inconvénients



# Avantages

- + *Simplicité de mise à jour*
- + *Lisibilité par un humain*
- + *la séparation du client et du serveur, qui aide à scaler plus facilement les applications*
- + *le fait d'être stateless, ce qui rend les requêtes API très spécifiques et orientées vers le détail*
- + *Représentation multiple (XML,JSON,.....)*

# Inconvénients

- *Sécurité restreinte par l'emploi des méthode HTTP*
- *Cible l'Apple de ressources*

# REST et SOAP

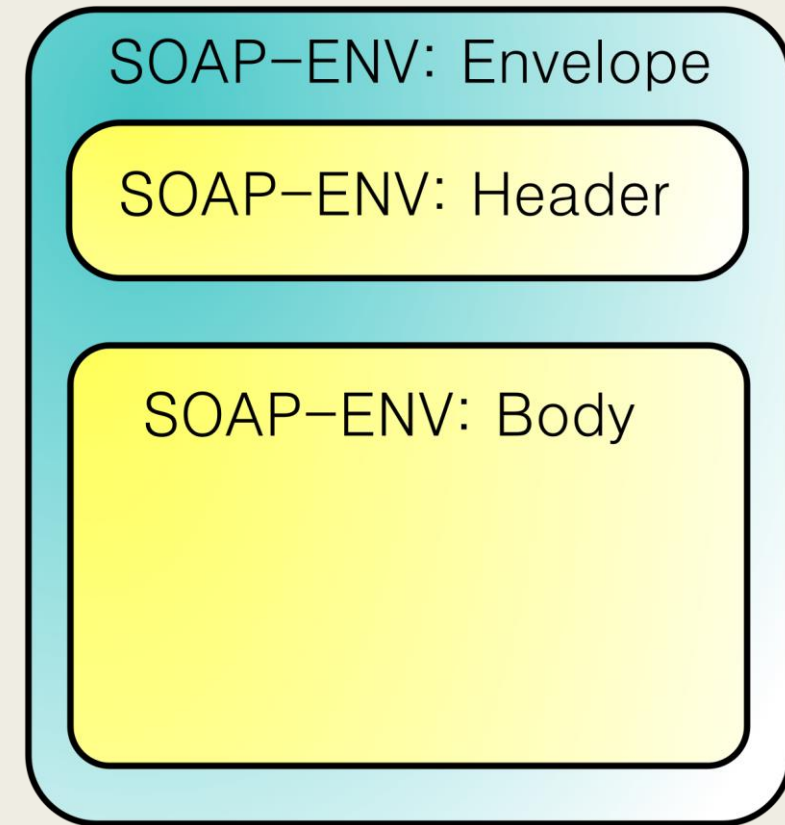
- REST et SOAP sont des approches différentes de la transmission des données en ligne. Plus précisément, toutes deux définissent la manière de développer des interfaces de programmation d'application (API) qui permettent les échanges de données entre plusieurs applications web.

# SOAP : Simple Object Access Protocol

- **SOAP** est un protocole standard initialement conçu pour que des applications développées avec différents langages sur différentes plateformes puissent communiquer.
- Comme il s'agit d'un protocole, il impose des règles intégrées qui augmentent la complexité et les coûts, ce qui peut ralentir le chargement des pages.

- Cependant, ces standards assurent la conformité et sont ainsi privilégiés pour certains scénarios d'entreprise.
- Les standards de conformité intégrés incluent:
  1. la sécurité,
  2. l'atomicité,
  3. la cohérence,
  4. l'isolement et
  5. la durabilité (ACID)

- Autre mot SOAP est un protocole d'échange d'information structurée dans l'implémentation de services web bâti sur XML.
- Il permet la transmission de messages entre objets distants, ce qui veut dire qu'il autorise un objet à invoquer des méthodes d'objets physiquement situés sur un autre serveur.



- Lorsqu'une requête de données est envoyée à une API SOAP, elle peut être gérée par n'importe quel protocole de couches de l'application :
  1. *HTTP (pour les navigateurs web),*
  2. *SMTP (pour les e-mails),*
  3. *TCP*
  4. *et autres.*
- En revanche, suite à la réception de la requête, les messages SOAP doivent être renvoyés sous la forme d'un document XML, un langage balisé lisible aussi bien par les humains que par les machines.
- Une fois finalisée, une requête destinée à une API SOAP ne peut pas être mise en cache par un navigateur. Il n'est donc pas possible d'y accéder plus tard sans la renvoyer vers l'API.

# SOAP ou REST : comment choisir ?

- De nombreux systèmes d'anciennes générations reposent encore sur le protocole SOAP.
- REST est arrivé plus tardivement et est souvent considéré comme une solution plus rapide pour des scénarios basés sur le web.
- REST est un ensemble de recommandations qui permet une mise en œuvre flexible, tandis que SOAP est un protocole avec des exigences spécifiques comme l'envoi de messages au format XML.



- Les API REST sont plus légères et donc plus adaptées aux concepts récents tels que l'Internet des objets (IoT), le développement d'applications mobiles et le serverless.
- Les services web SOAP intègrent des spécifications de sécurité et de conformité des transactions qui répondent aux besoins de nombreuses entreprises, mais qui les rendent également plus lourds.
- De plus, de nombreuses API publiques, telles que l'API Google Maps, suivent les recommandations REST.

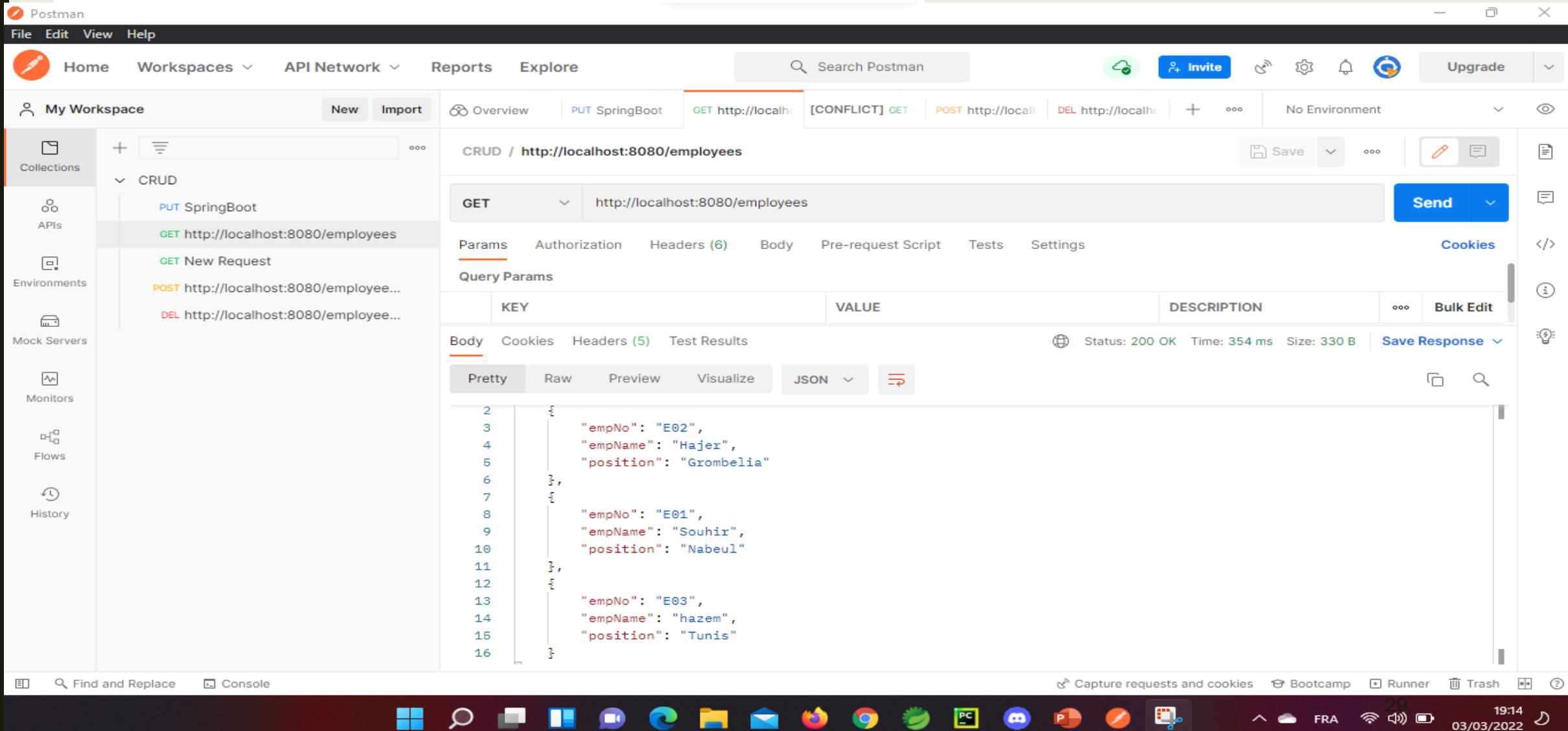
# Développement

# Développement

- Pour créer un contrôleur, il suffit de créer une classe et de l'annoter `@RestController` et de lui affecté un point d'accès.
- Chacune des méthodes aura l'annotation `@RequestMapping` qui indique quel chemin de l'API la méthode couvre et quelle méthode HTTP lui correspond.
- Ces annotations permettent à simplifier le code et à le rendre plus lisible.
- Le Framework s'occupe de démarrer le serveur web et de rediriger les requêtes aux méthodes concernées.

# Partie Pratique

# Get Tous les Employées



The screenshot shows the Postman interface with a GET request to `http://localhost:8080/employees` executed. The response is a JSON array of three employee objects. The status is 200 OK, and the response body is displayed in a pretty-printed JSON format.

**Request Details:**

- Method: GET
- URL: `http://localhost:8080/employees`

**Response Details:**

- Status: 200 OK
- Time: 354 ms
- Size: 330 B

**Response Body (JSON):**

```
[
  {
    "empNo": "E02",
    "empName": "Hajer",
    "position": "Grombelia"
  },
  {
    "empNo": "E01",
    "empName": "Souhir",
    "position": "Nabeul"
  },
  {
    "empNo": "E03",
    "empName": "hazem",
    "position": "Tunis"
  }
]
```

# Get Un Seul Employée avec son ID

The screenshot shows the Postman application interface. The top navigation bar includes 'File', 'Edit', 'View', and 'Help'. Below it, the main navigation bar has 'Home', 'Workspaces', 'API Network', 'Reports', and 'Explore'. The 'My Workspace' section is active, showing a list of collections on the left and a 'New Request' button. The 'New Request' button is selected, and the 'GET' method is chosen for the URL 'http://localhost:8080/employee/E02'. The 'Send' button is visible. The 'Params' tab is selected, showing a table with 'KEY' and 'VALUE' columns. The 'Body' tab is also visible, showing the response body in JSON format. The status bar at the bottom indicates 'Status: 200 OK', 'Time: 60 ms', and 'Size: 220 B'.

Postman

File Edit View Help

Home Workspaces API Network Reports Explore

Search Postman

Invite Upgrade

My Workspace

New Import

Overview PUT SpringBoot GET http://localhost:8080/employees [CONFLICT] POST http://localhost:8080/employee... DEL http://localhost:8080/employee... No Environment

CRUD / New Request

GET http://localhost:8080/employee/E02

Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (5) Test Results

Status: 200 OK Time: 60 ms Size: 220 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "empNo": "E02",
3   "empName": "Hajer",
4   "position": "Grombelia"
5 }
```

Find and Replace Console

Capture requests and cookies Bootcamp Runner Trash

19:17 03/03/2022

# Put Un Employée

The screenshot shows the Postman application interface. The main workspace displays a REST client request to `http://localhost:8080/employees` using the `GET` method. The request is saved under the collection `CRUD` in the `PUT SpringBoot` environment. The response is a JSON array of three employee objects, returned with a `200 OK` status.

**Request Details:**

- Method: `GET`
- URL: `http://localhost:8080/employees`
- Environment: `PUT SpringBoot`
- Collection: `CRUD`

**Response Details:**

- Status: `200 OK`
- Time: `8 ms`
- Size: `335 B`
- Response Type: `JSON`

**Response Body (JSON):**

```
1 [
2   {
3     "empNo": "E02",
4     "empName": "abdelkaderdjo",
5     "position": "Sousse"
6   },
7   {
8     "empNo": "E01",
9     "empName": "Souhir",
10    "position": "Nabeul"
11  },
12  {
13    "empNo": "E03",
14    "empName": "hazem",
15    "position": "Tunis"
16  }
17 ]
```

# Delete Un Employée

The screenshot shows the Postman application interface. The top navigation bar includes 'File', 'Edit', 'View', and 'Help'. Below it, there's a 'Home' button and tabs for 'Workspaces', 'API Network', 'Reports', and 'Explore'. A search bar labeled 'Search Postman' is present. The main workspace is titled 'My Workspace' and contains a list of collections. The 'Collections' sidebar on the left shows a 'CRUD' collection with several requests. The selected request is 'GET http://localhost:8080/employees'. The main panel displays the details of this request, including the method 'GET', the URL 'http://localhost:8080/employees', and the response body. The response is a JSON array of two employee objects. The status bar at the bottom indicates 'Status: 200 OK', 'Time: 10 ms', and 'Size: 273 B'. The system tray at the very bottom shows the Windows taskbar with various application icons and the system clock displaying '19:39 03/03/2022'.

Postman

File Edit View Help

Home Workspaces API Network Reports Explore

Search Postman

My Workspace

New Import

Overview PUT SpringBoc GET http://localh [CONFLICT] POST http://local DEL http://loca + ... No Environment

CRUD / http://localhost:8080/employees

GET http://localhost:8080/employees

Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
-----	-------	-------------	-----	-----------

Body Cookies Headers (5) Test Results

Status: 200 OK Time: 10 ms Size: 273 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   {
3     "empNo": "E01",
4     "empName": "Souhir",
5     "position": "Nabeul"
6   },
7   {
8     "empNo": "E03",
9     "empName": "hazem",
10    "position": "Tunis"
11  }
12 }
```

Find and Replace Console

Capture requests and cookies Bootcamp Runner Trash

19:39 03/03/2022



# Post Un Employée

The screenshot shows the Postman application interface. The left sidebar displays the 'My Workspace' with a collection named 'CRUD'. The main panel shows a GET request to 'http://localhost:8080/employees'. The response is a JSON array of three employee objects, each with 'empNo', 'empName', and 'position' fields.

**Request Details:**

- Method: GET
- URL: http://localhost:8080/employees

**Response Details:**

- Status: 200 OK
- Time: 12 ms
- Size: 329 B

**Response Body (JSON):**

```
[{"empNo": "E02", "empName": "mo3iz", "position": "tataouin"}, {"empNo": "E01", "empName": "Souhir", "position": "Nabeul"}, {"empNo": "E03", "empName": "hazem", "position": "Tunis"}]
```

# Conclusion

# Conclusion

Le Framework Spring Boot permet de créer rapidement des API REST solides selon une architecture de code respectant le modèle MVC.

# Références

- <https://slideplayer.fr/slide/15548242/>
- [https://devstory.net/11645/exemple-crud-restful-webservice-avec-spring-boot?fbclid=IwAR2XQNSXSzqrXD-IkxjCLmegmUG6Q-74LPucrHUTtE\\_PpTY3wXqGFyysWuo](https://devstory.net/11645/exemple-crud-restful-webservice-avec-spring-boot?fbclid=IwAR2XQNSXSzqrXD-IkxjCLmegmUG6Q-74LPucrHUTtE_PpTY3wXqGFyysWuo)
- <https://devstory.net/10773/qu-est-ce-que-restful-web-service?fbclid=IwAR2v1GOPAPFtKIBhliJYg6bQbIY5Ja9zZajN7JoqDBwKIHmHz8Q80cDqRs0>
- [https://slideplayer.fr/slide/5523684/?fbclid=IwAR3n2VKmbp\\_qrPZ9nWXekF8obX\\_AQfpqo\\_tKvcjqJd5PgzxLR63I2LNswVc](https://slideplayer.fr/slide/5523684/?fbclid=IwAR3n2VKmbp_qrPZ9nWXekF8obX_AQfpqo_tKvcjqJd5PgzxLR63I2LNswVc)
- <https://slideplayer.fr/slide/4792791/?fbclid=IwAR01asgJAKB-195ryCRkba48K63iU8aX57i-P81026XN6FpYxiAk-gjZYrQ>
- <https://www.redhat.com/fr/topics/integration/whats-the-difference-between-soap-rest>

Fin de la présentation

**MERCI POUR VOTRE  
ATTENTION**