TheCodingMachine

Web Architecture basics & API

[NOOBS 2023 #2]

Thibault Balmette

Web architecture basics

From client request to server response

SUMMARY

01
DNS Resolution

02
TCP Protocol

03
HTTP Protocol

04
Server treatment

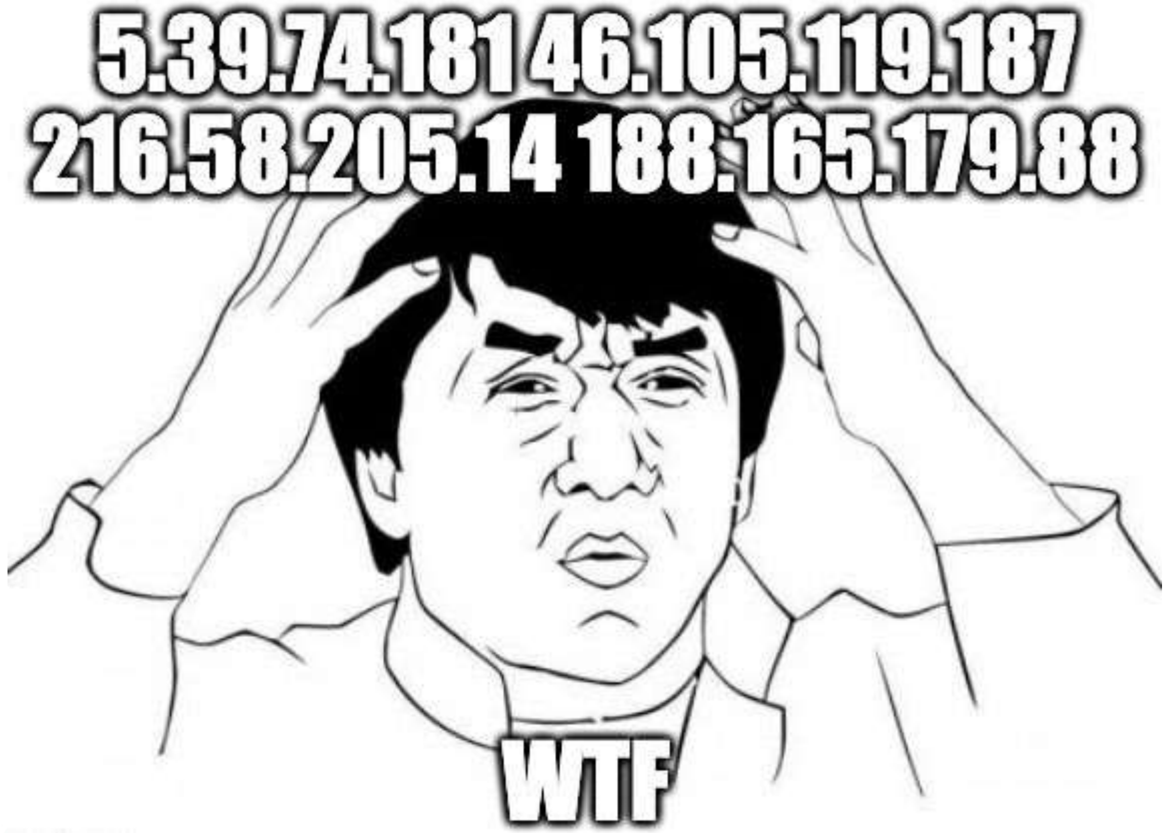
05
Technologies

01

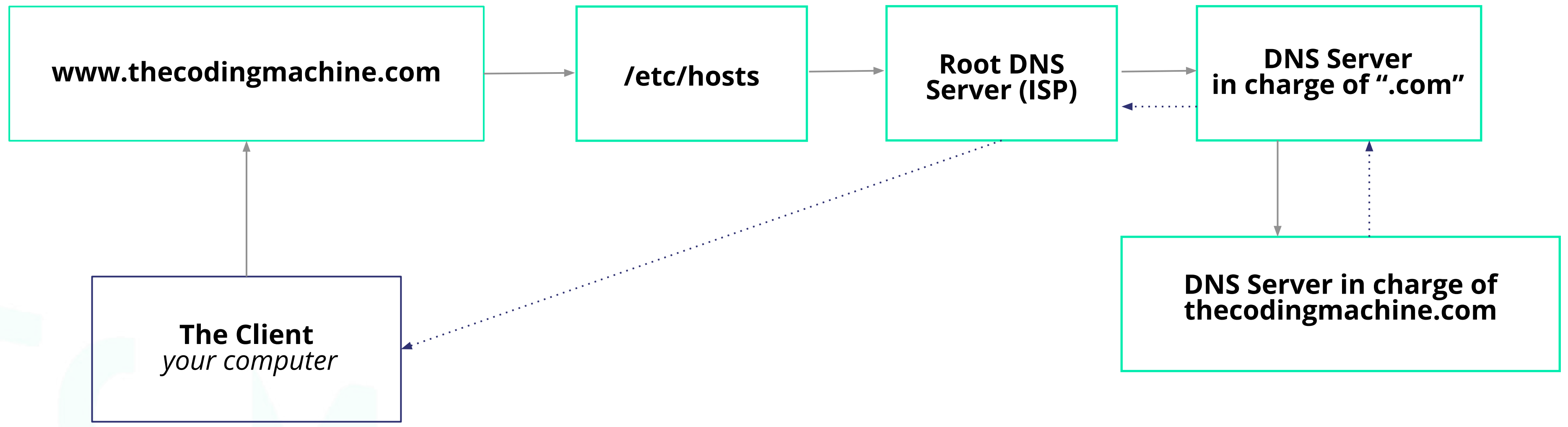
DNS RESOLUTION

DOMAIN NAME SYSTEM

- Translates domain names to the numerical IP addresses
- Designed for humans
- *www.thecodingmachine.com* is a domain name



FROM CLIENT REQUEST TO SERVER RESPONSE



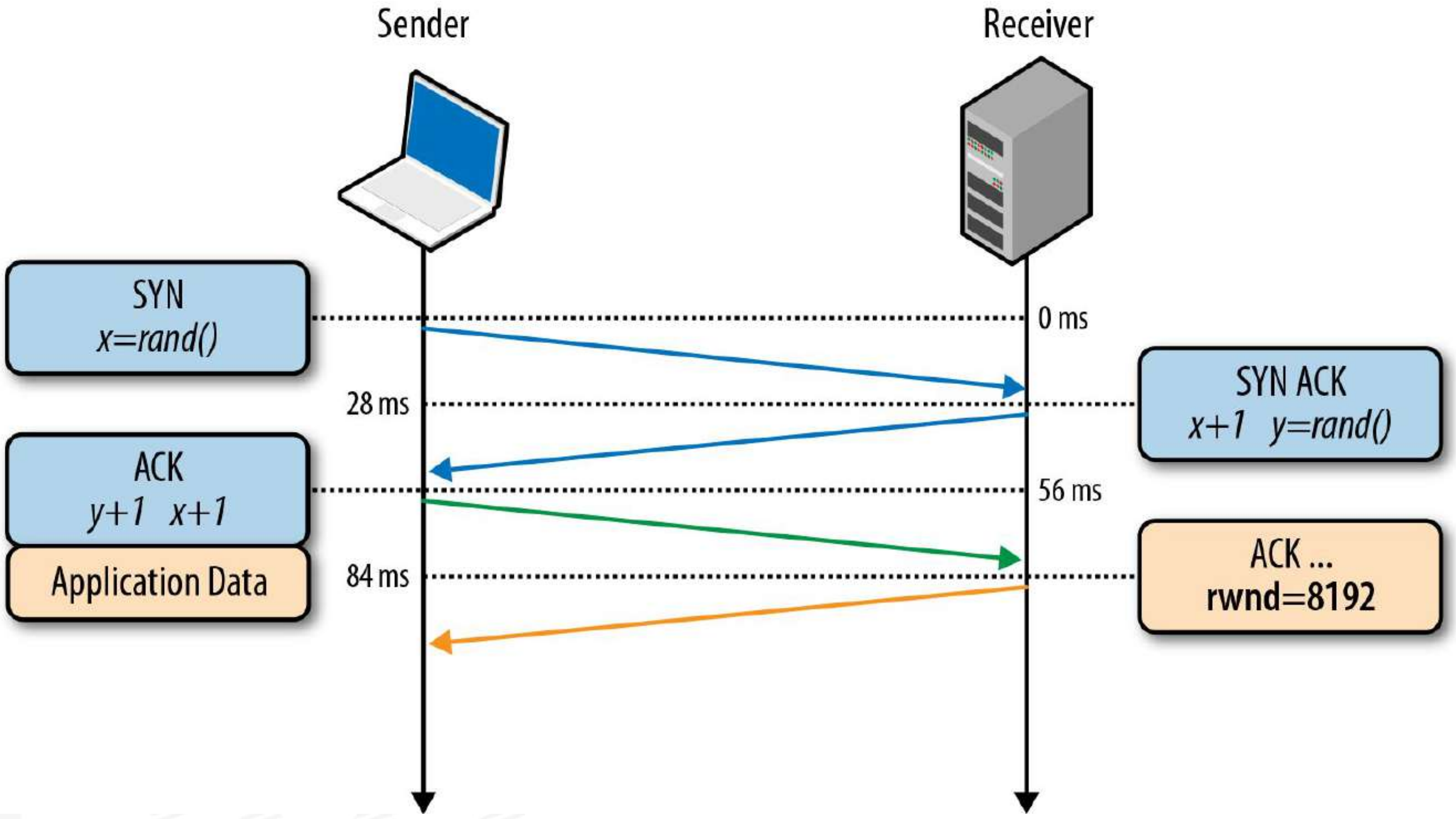
02

TCP PROTOCOL

TCM://

THE BASICS OF HTTP AND THE WEB

3 ways handshake



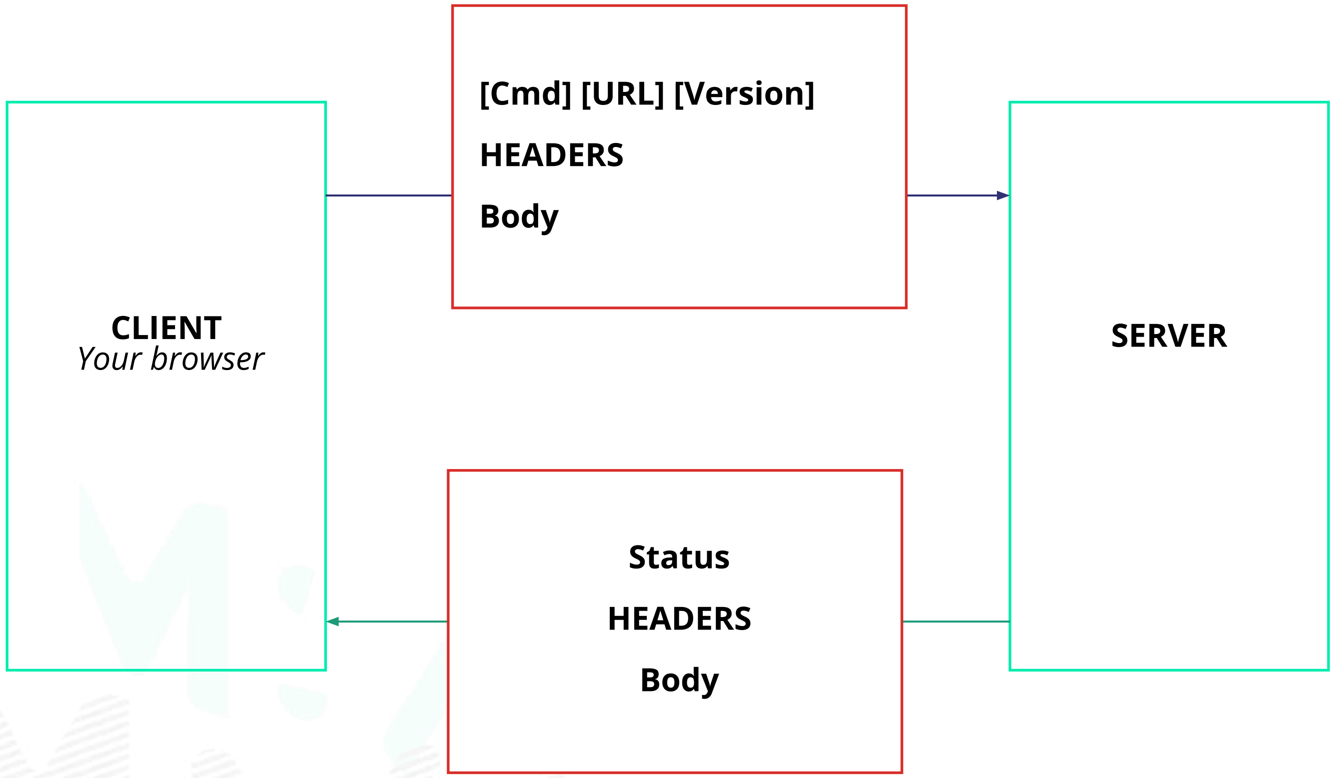
03

HTTP PROTOCOL

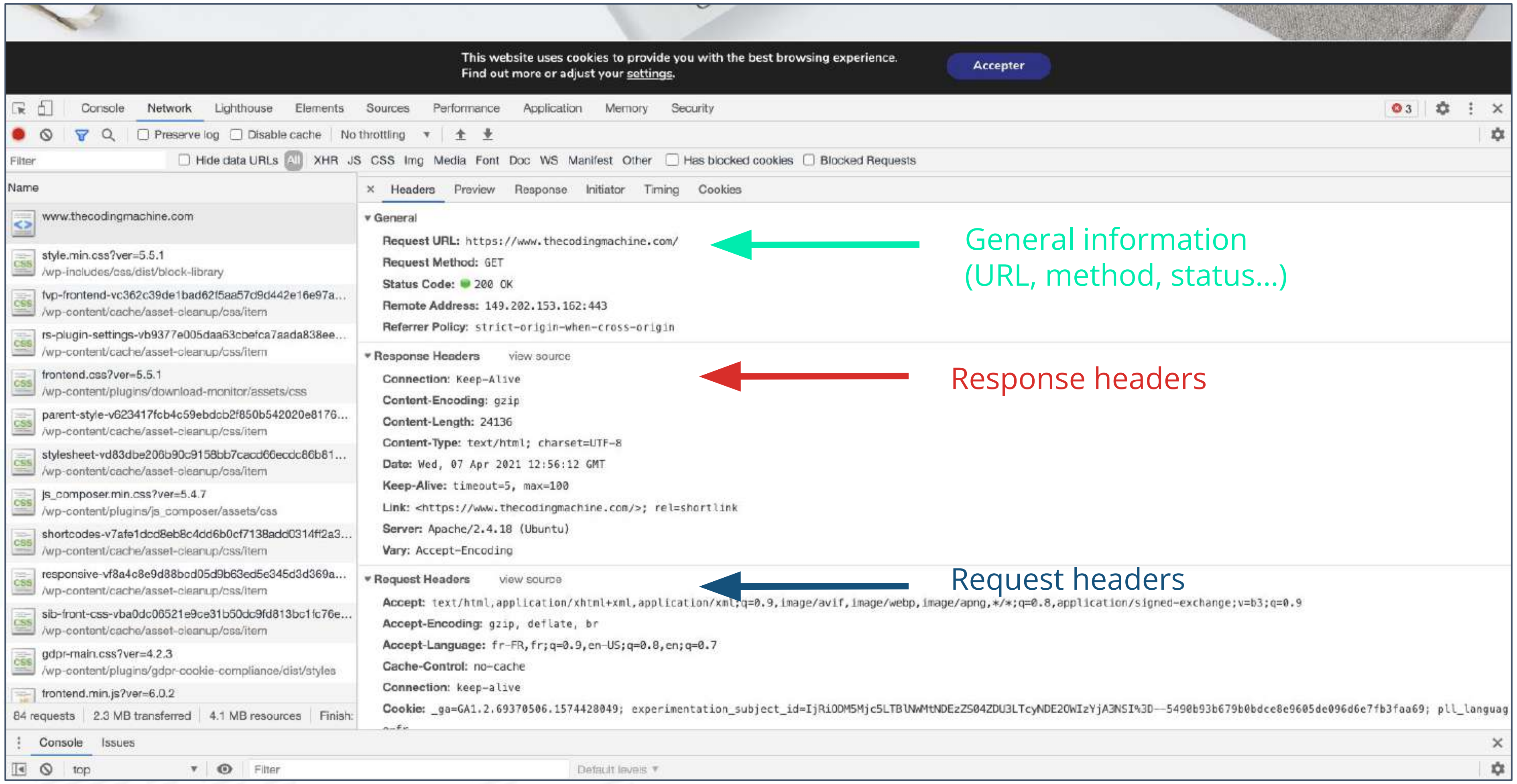
TCM://

WEB COMMUNICATION

Extension of the TCP protocol



REQUEST / RESPONSE EXAMPLE



```
Name
www.thecodingmachine.com
style.min.css?ver=5.5.1
/wp-includes/css/dist/block-library
fvp-frontend-vc362c39de1bad62f5aa57d9d442e16e97a...
/wp-content/cache/asset-cleanup/css/item
rs-plugin-settings-vb9377e005daa63cbefca7aada838ee...
/wp-content/cache/asset-cleanup/css/item
frontend.css?ver=5.5.1
/wp-content/plugins/download-monitor/assets/css
parent-style-v623417fcb4c59ebdcb2f850b542020e8176...
/wp-content/cache/asset-cleanup/css/item
stylesheet-vd83dbe206b90c9158bb7cacd66ecdc66b61...
/wp-content/cache/asset-cleanup/css/item
js_composer.min.css?ver=5.4.7
/wp-content/plugins/js_composer/assets/css
shortcodes-v7afe1dcd8eb8c4dd6b0cf7138add0314ff2a3...
/wp-content/cache/asset-cleanup/css/item
responsive-vf8a4c8e9d88bcd05d9b63ed5e345d3d369a...
/wp-content/cache/asset-cleanup/css/item
sib-front-css-vba0dc06521e9ce31b50dc9fd813bc1fc76e...
/wp-content/cache/asset-cleanup/css/item
gdpr-main.css?ver=4.2.3
/wp-content/plugins/gdpr-cookie-compliance/dist/styles
frontend.min.js?ver=6.0.2
84 requests | 2.3 MB transferred | 4.1 MB resources | Finish:
```

```
▾ General
  Request URL: https://www.thecodingmachine.com/
  Request Method: GET
  Status Code: ● 200 OK
  Remote Address: 149.202.153.162:443
  Referrer Policy: strict-origin-when-cross-origin
▾ Response Headers      view source
  Connection: Keep-Alive
  Content-Encoding: gzip
  Content-Length: 24136
  Content-Type: text/html; charset=UTF-8
  Date: Wed, 07 Apr 2021 12:56:12 GMT
  Keep-Alive: timeout=5, max=100
  Link: <https://www.thecodingmachine.com/>; rel=shortlink
  Server: Apache/2.4.18 (Ubuntu)
  Vary: Accept-Encoding
▾ Request Headers      view source
  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
  Accept-Encoding: gzip, deflate, br
  Accept-Language: fr-FR,fr;q=0.9,en-US;q=0.8,en;q=0.7
  Cache-Control: no-cache
  Connection: keep-alive
  Cookie: _ga=GA1.2.69370506.1574428049; experimentation_subject_id=IjRiODM5Mjc5LTBlNWMtNDEzZS04ZDU3LTcyNDE2OWIzYjA3NSI%3D--5490b93b679b0bdce8e9605de096d6e7fb3faa69; pll_languag
```

Always keep your “Network” tab open when you are working!
(and don’t forget to filter your XHR requests for example)

THE METHODS

GET / PUT / DELETE / POST

•GET :

Used to get data
The parameters are directly encoded in the URL

•POST / PUT / DELETE

Used to submit data
The parameters are in the body of the request

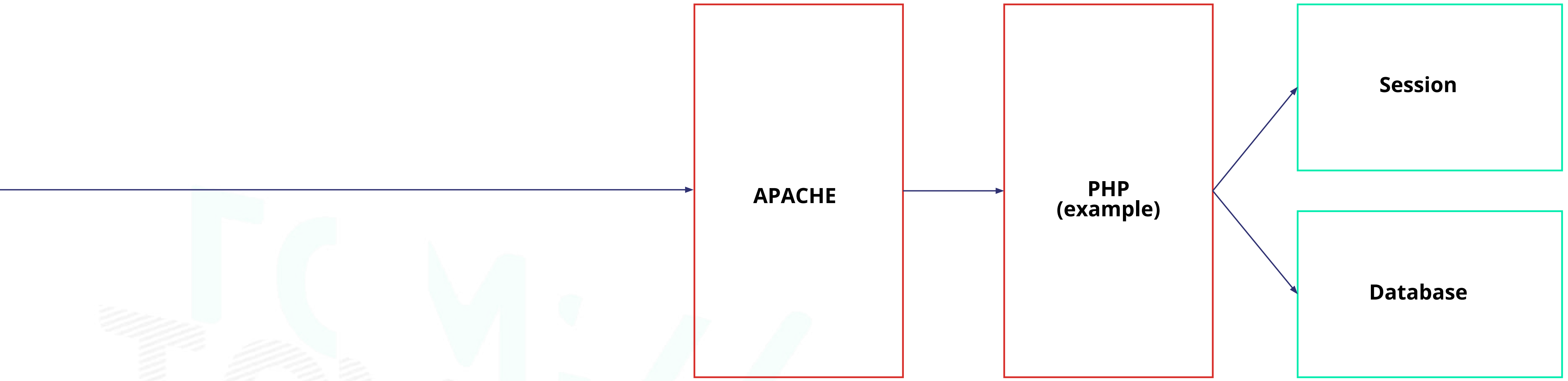
If the HTTPS protocol is used, the body and the header are encrypted

04

SERVER TREATMENT

TCM://

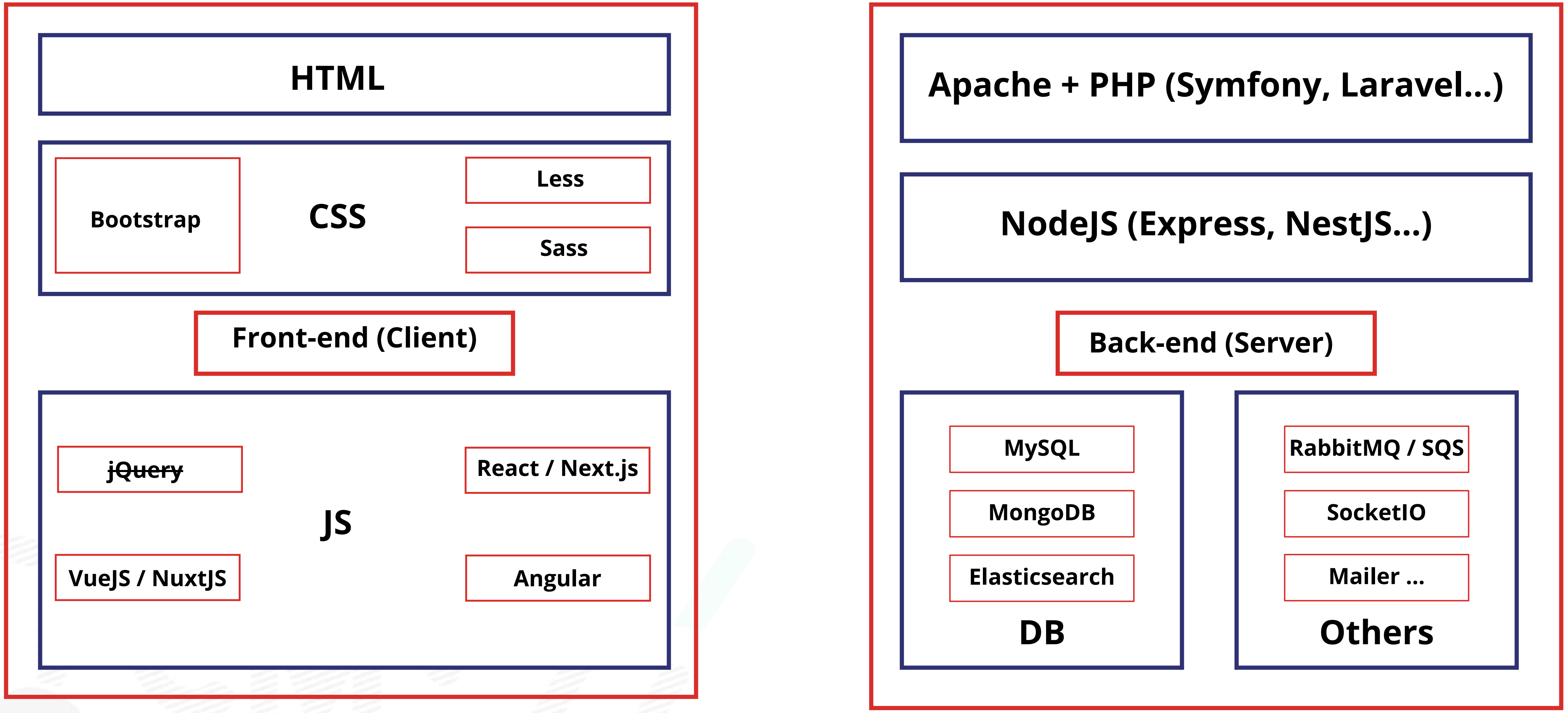
CONNECTION TO THE DATABASE & SESSION MANAGEMENT



05

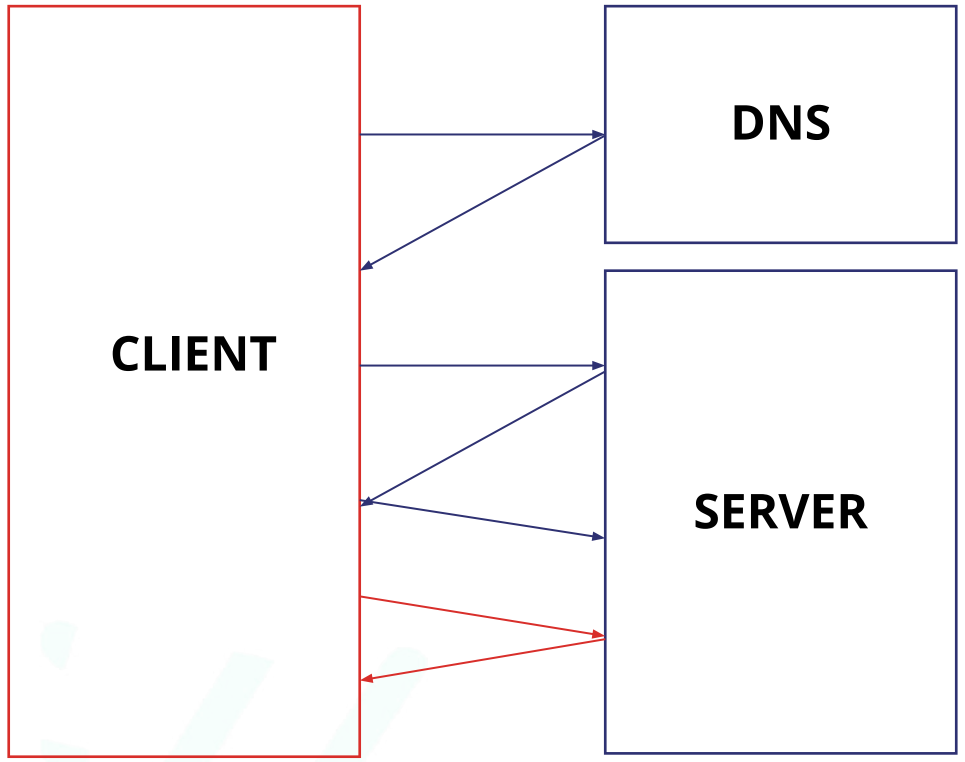
TECHNOLOGIES

COMMON WEB TECHNOLOGIES USED AT TCM



Exercice: describe the architecture of your project

SUM UP



THE CODING MACHINE

API

Integration architecture

SUMMARY

01 API

02 REST - Web service

03 GraphQL

04 OAUTH2

05 POSTMAN

06 PHP - VCR

07 Asynchronous tasks

TCM://

01

API

TCM://

APPLICATION PROGRAMMING INTERFACE

Standardized set of classes, methods or functions that serves a front through which software offers services to other software.

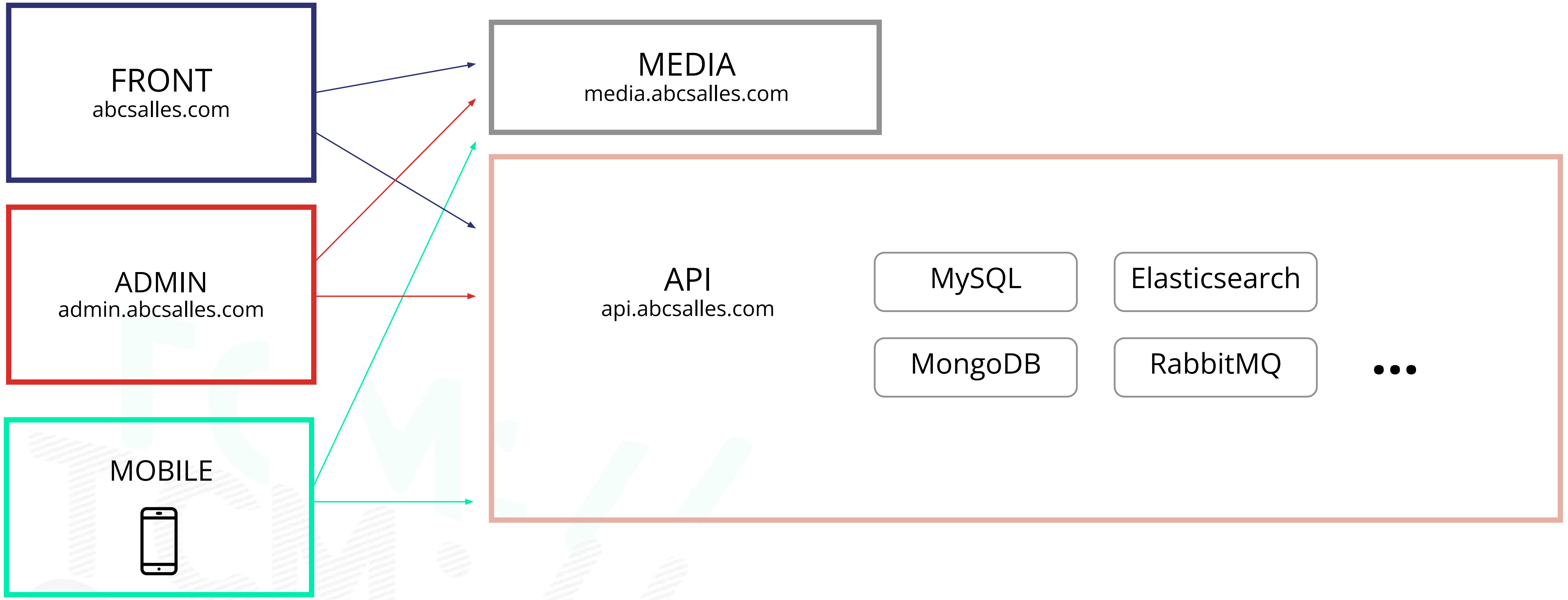
It is offered by:

- A software library (for example the *Geolocation API* in JavaScript exposed by your browser)
- A web service (what interests us here)

It is (usually) bundled with a description (*documentation*) which specifies how consumer programs can make use of the functionality of the supplier program.

APPLICATION PROGRAMMING INTERFACE

ABC Salles example



02

REST
Web service

DEFINITION

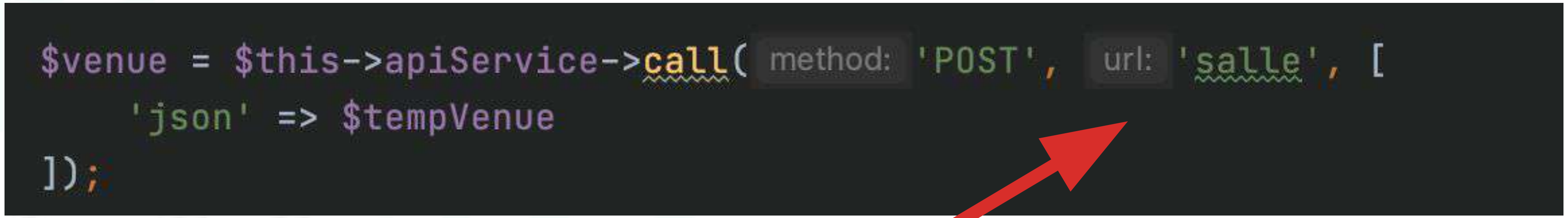
It's a type of architecture/protocol that uses HTTP and mainly the JSON (or XML) format.

It respects the following constraints:

- **Client-Server :** The two are separate and can evolve independently.
- **Stateless :** Each request must contain all the information necessary to allow the server to understand the request
- **Caching possible**
- **Hierarchical layer system :** the application states are identified by individual resources
- **Code on demand** (optional)
- **Uniform interface :** resource identification (URI), resource manipulation, ...

URL	GET	PUT	POST	DELETE
Collections: http://api.website.com/x.x/object	List objects	Replace an entire collection by another	Add an object to the collection	Delete the entire collection
Object: http://api.website.com/x.x/object/12	Return the representation of an object	Update the object or create it if it does not exist	Add an element in a sub collection of an object (rare)	Delete the object

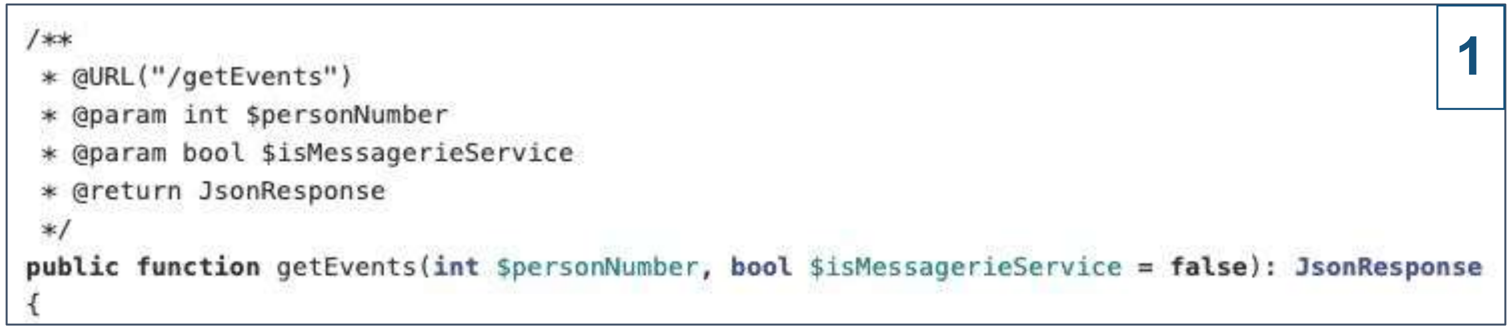
The goal of a "RESTful API" is to contain <u>the maximum of meaning</u> without needing a specific (and external) documentation



```
$venue = $this->apiService->call( method: 'POST',  url: 'salle', [
    'json' => $tempVenue
]);
```

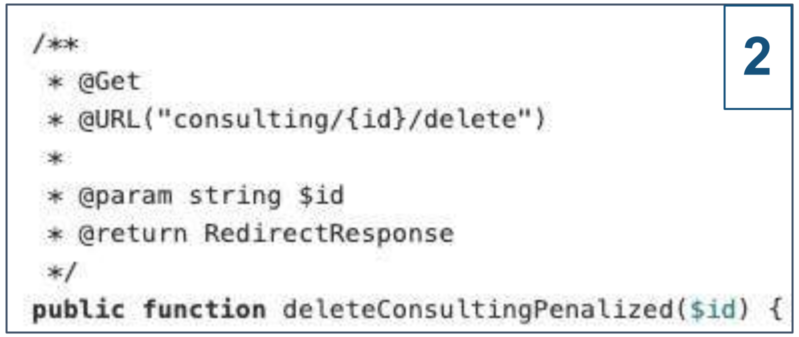
Exercice: what is the purpose of this call?

EXERCICE : WHAT IS WRONG WITH THESE METHODS?



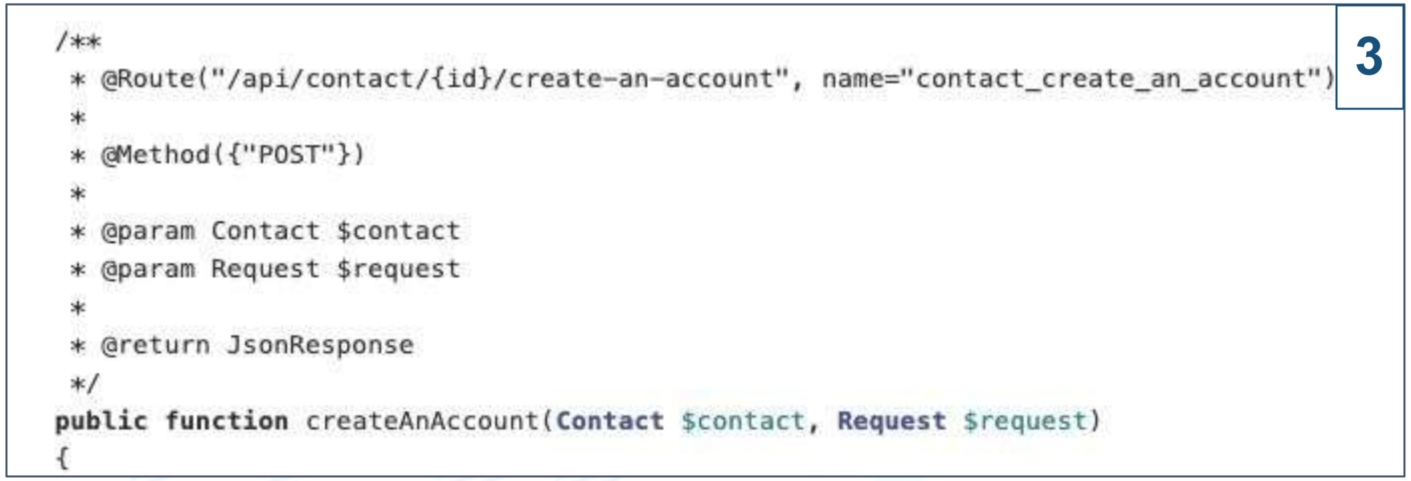
1

```
/**
 * @URL("/getEvents")
 * @param int $personNumber
 * @param bool $isMessagerieService
 * @return JsonResponse
 */
public function getEvents(int $personNumber, bool $isMessagerieService = false): JsonResponse
{
```



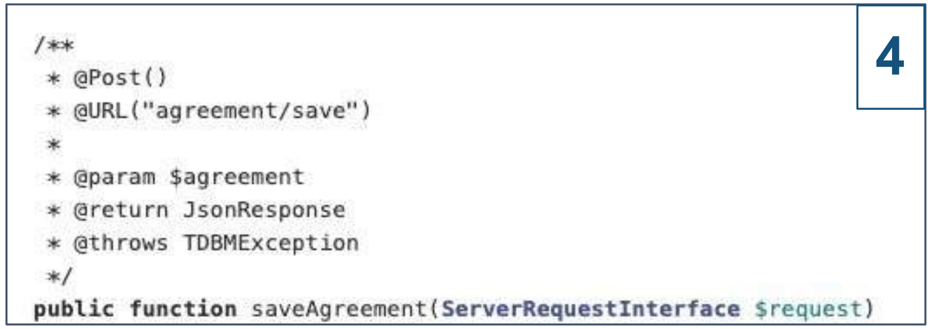
2

```
/**
 * @Get
 * @URL("consulting/{id}/delete")
 *
 * @param string $id
 * @return RedirectResponse
 */
public function deleteConsultingPenalized($id) {
```



3

```
 /**
  * @Route("/api/contact/{id}/create-an-account", name="contact_create_an_account")
  *
  * @Method({"POST"})
  *
  * @param Contact $contact
  * @param Request $request
  *
  * @return JsonResponse
  */
 public function createAnAccount(Contact $contact, Request $request)
 {
```



4

```
/**
 * @Post()
 * @URL("agreement/save")
 *
 * @param $agreement
 * @return JsonResponse
 * @throws TDBMException
 */
public function saveAgreement(ServerRequestInterface $request)
```

RETURNED CODES ARE IMPORTANT

- **2XX : Success**
  - 200 : OK
  - 201 : Created
- **5XX : Server Error**
  - 500 : Internal Error
  - 501 : Not Implemented
  - 503 : Service Unavailable
- **3XX : Redirection**
- **4XX : Client Error**
  - 400 : Bad Request
  - 401 : Unauthorized
  - 403 : Forbidden
  - 404 : Not found
  - 409 : Conflict

AS WELL AS THE HEADERS SENT

- **Content-Type :** application/json
- **Authorization :** Bearer 0b79bab50daca910b000d4f1a2b675d604257e42

SOAP Web Service

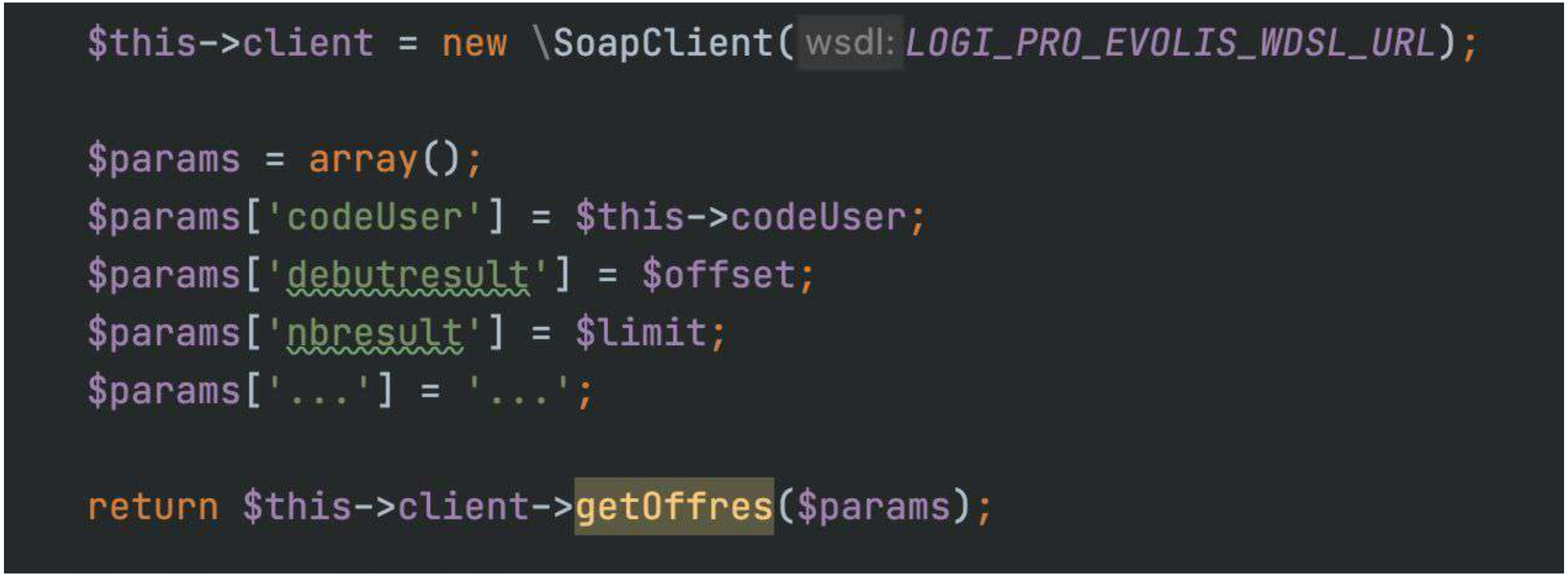
- These web services expose the same functionalities in the form of remotely executable services.
- Their specifications are based on SOAP and WSDL standards.

SOAP

- **Object oriented RCP (Remote Procecedure Call) protocol**, build on XML
- **Transmission of messages between remote objects,** allows an object to invoke methods of objects physically located on another server

WSDL

- **Web Services Description Language** is an XML grammar used to describe a web services. It contains the definition of objects (classes) and methods.



```php
$this->client = new \SoapClient( wsdl: LOGI_PRO_EVOLIS_WDSL_URL);

$params = array();
$params['codeUser'] = $this->codeUser;
$params['debutresult'] = $offset;
$params['nbresult'] = $limit;
$params['...'] = '...';

return $this->client->getOffres($params);
```

Example of a SOAP Web Service using PHP

ADVANTAGES AND DISADVANTAGES OF THE REST PROTOCOL

✓

- The application is easier to maintain because the client and the server are independent
- Lack of client state management on the server
  - No permanent connection
  - Distribution of requests on several servers
- Allows caching
- Use of HTTP (header, descriptive return code)
- Universal Element Identification System (URI)

✗

- The client must locally store all the data necessary for the smooth running of the application
- Higher bandwidth consumption

03

GRAPHQL

GRAPHQL IS A PROTOCOL

It is not :

- A new trendy database
- A database query language such as SQL

GraphQL is a challenger for these other protocols :

- REST
- SOAP/WSDL based web services

It is developed by Facebook and was used for the first time in the Facebook API

GraphQL is strongly typed

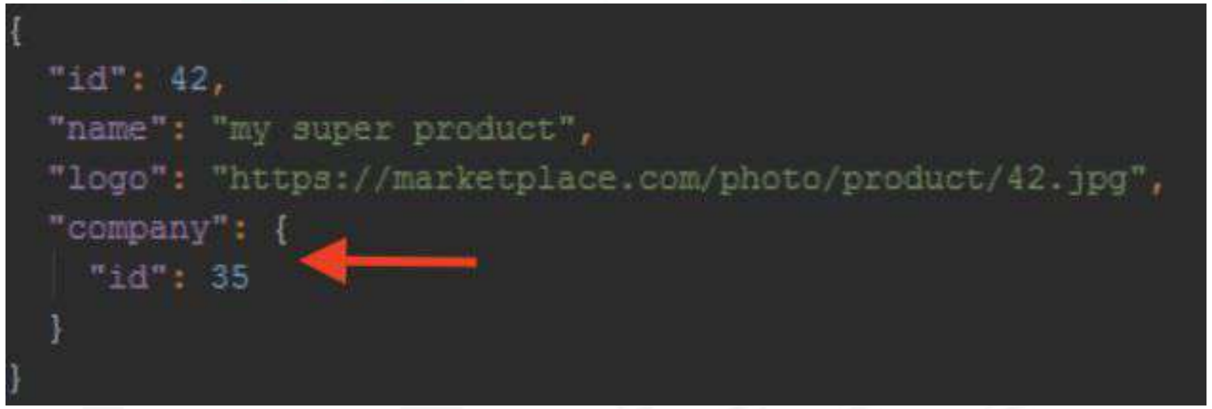
WHAT PROBLEM DOES GRAPHQL SOLVE ?

Your API changes often
You are developing a new feature but your API does not exactly meet your needs.

For example: you are developing a marketplace. You need a page to display a product, as well as company information.

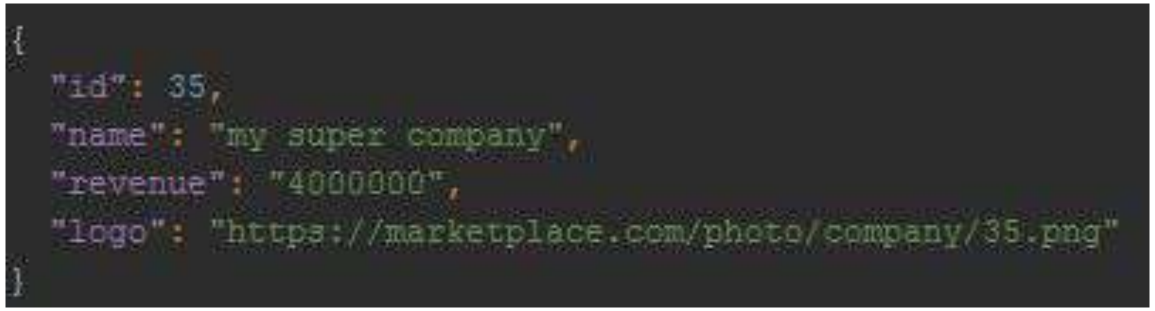
REST (under fetching)

/api/product/42



```
{
  "id": 42,
  "name": "my super product",
  "logo": "https://marketplace.com/photo/product/42.jpg",
  "company": {
    "id": 35
  }
}
```

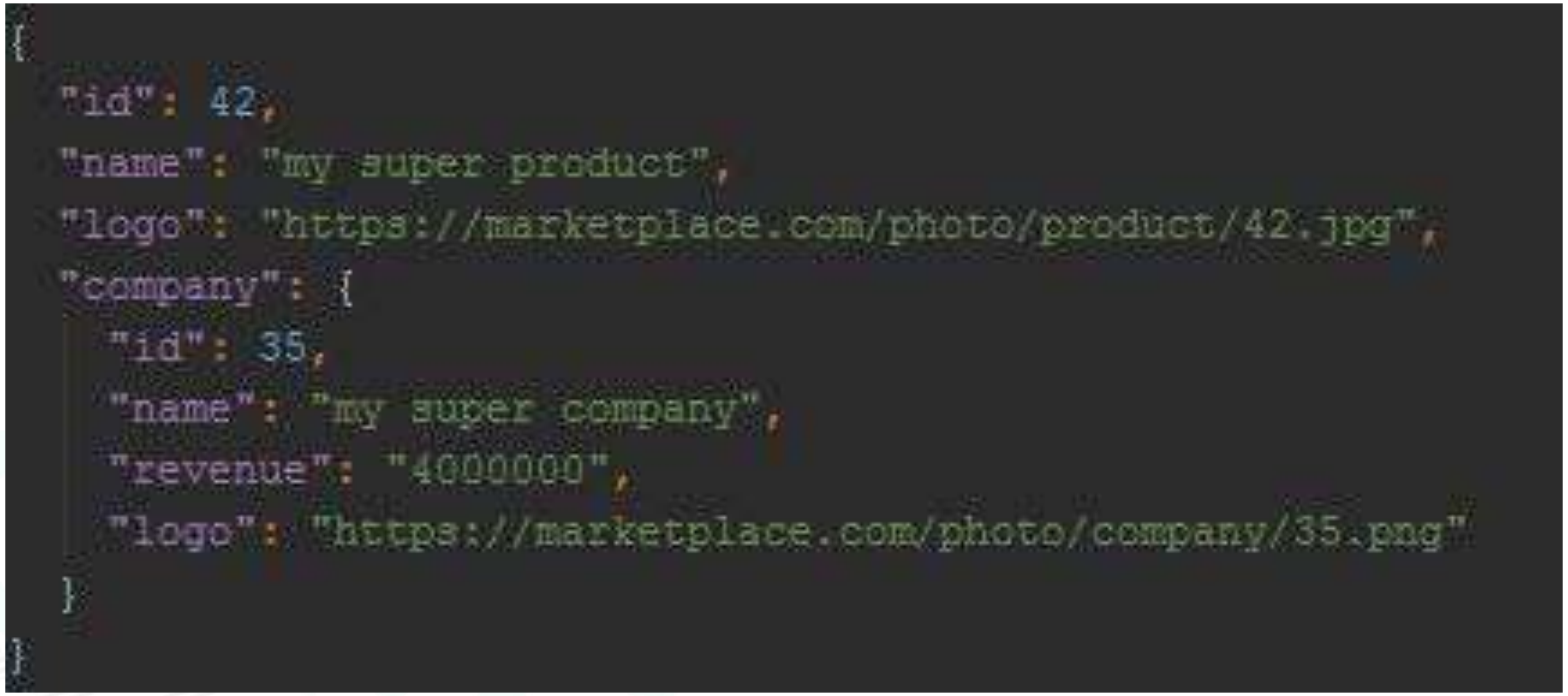
/api/company/35



```
{
  "id": 35,
  "name": "my super company",
  "revenue": "4000000",
  "logo": "https://marketplace.com/photo/company/35.png"
}
```

AN ALTERNATIVE (STILL REST)

/api/product/42 (over fetching)

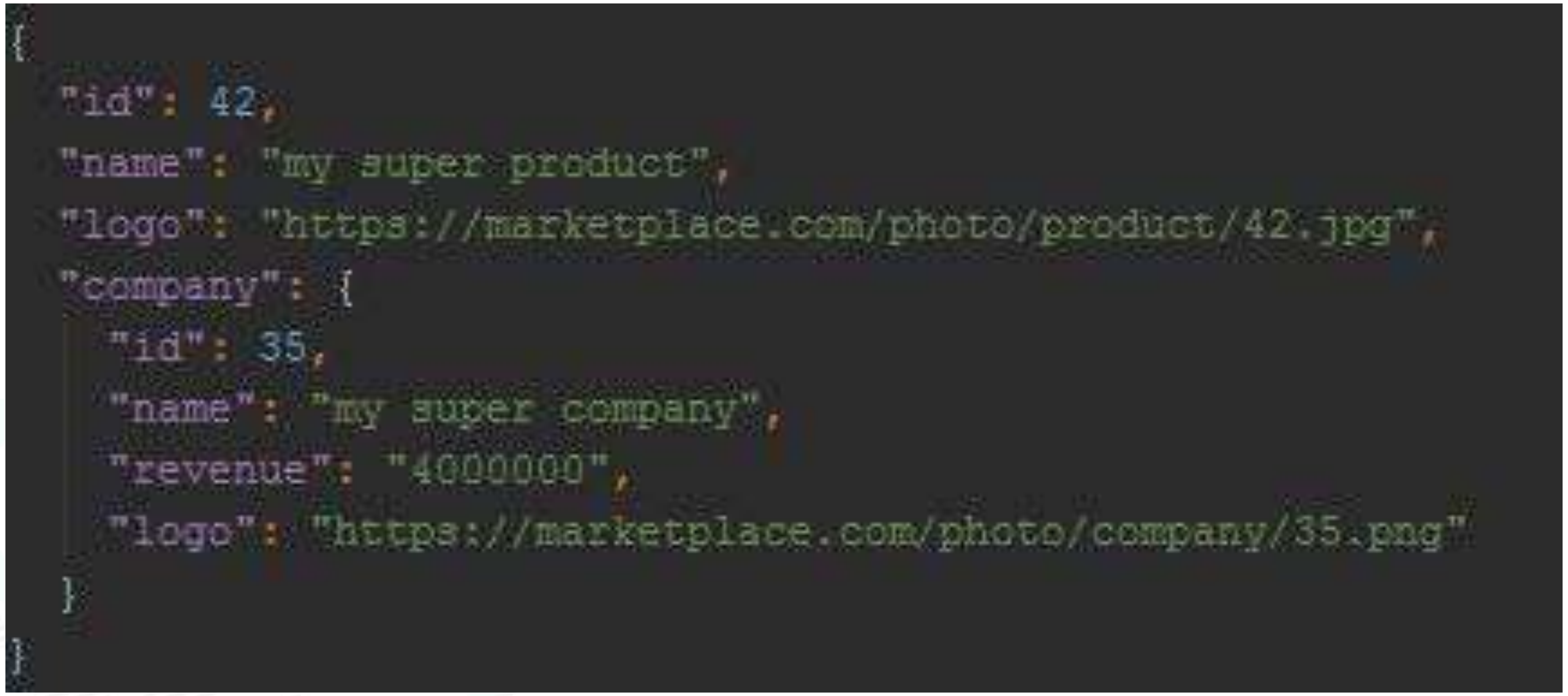


```
{
  "id": 42,
  "name": "my super product",
  "logo": "https://marketplace.com/photo/product/42.jpg",
  "company": {
    "id": 35,
    "name": "my super company",
    "revenue": "4000000",
    "logo": "https://marketplace.com/photo/company/35.png"
  }
}
```

ANOTHER ALTERNATIVE (STILL REST)

/api/product/42**?with_company=true**

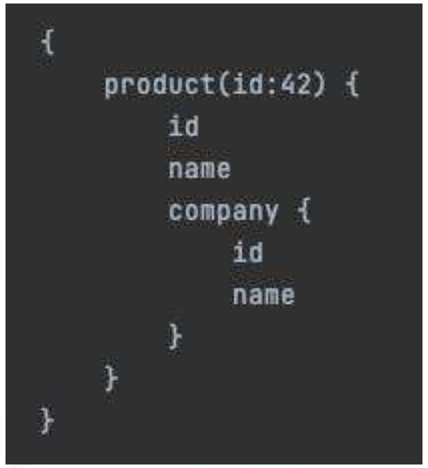
Flags hell 😨!
Probably one flag by API consumer



```
{
  "id": 42,
  "name": "my super product",
  "logo": "https://marketplace.com/photo/product/42.jpg",
  "company": {
    "id": 35,
    "name": "my super company",
    "revenue": "4000000",
    "logo": "https://marketplace.com/photo/company/35.png"
  }
}
```

The client requests the list of fields they want

GET /graphql?query= ← single endpoint



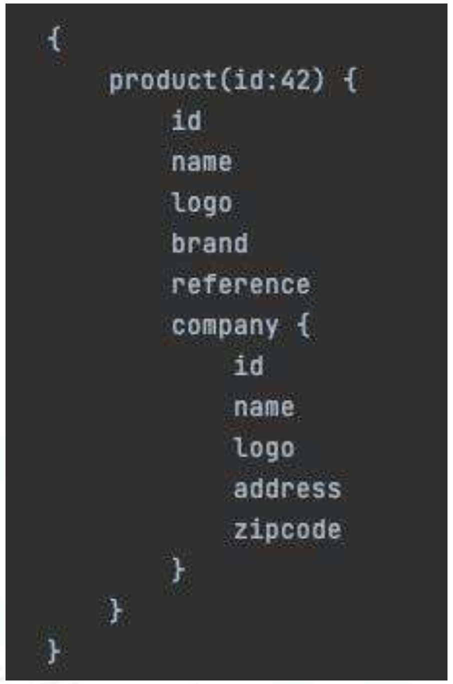
```
{
    product(id:42) {
        id
        name
        company {
            id
            name
        }
    }
}
```

← the name of the query is “product”

← lists of fields requested

- Another possible query on the same "query" with a different set of fields

GET /graphql?query=



```
{
    product(id:42) {
        id
        name
        logo
        brand
        reference
        company {
            id
            name
            logo
            address
            zipcode
        }
    }
}
```

No need to change the server side code!
All data is in one API call!

- GraphQL can also make mutations (to change the state of the DB)

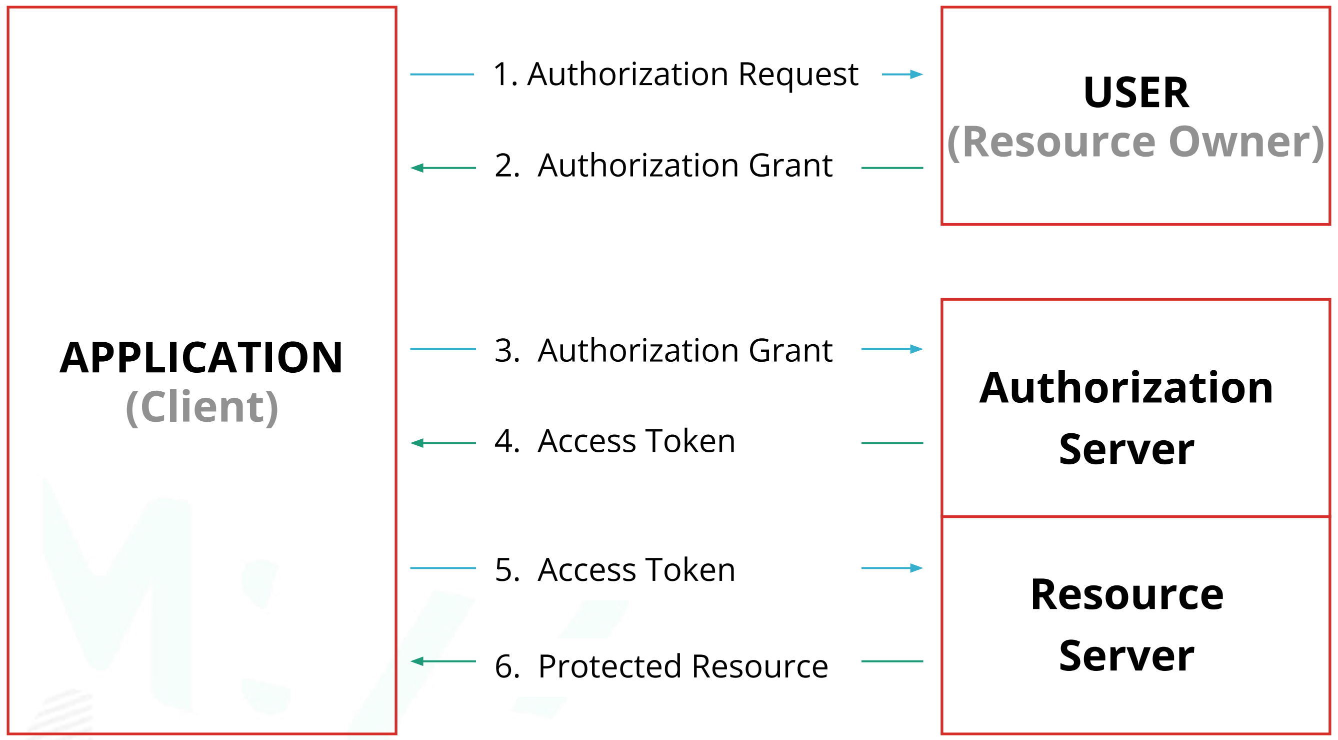
Cf. GraphQLite presentation!

https://drive.google.com/open?id=0B33pp5vqFdJhN3hJQmxZZDZwX00

04

OAUTH2

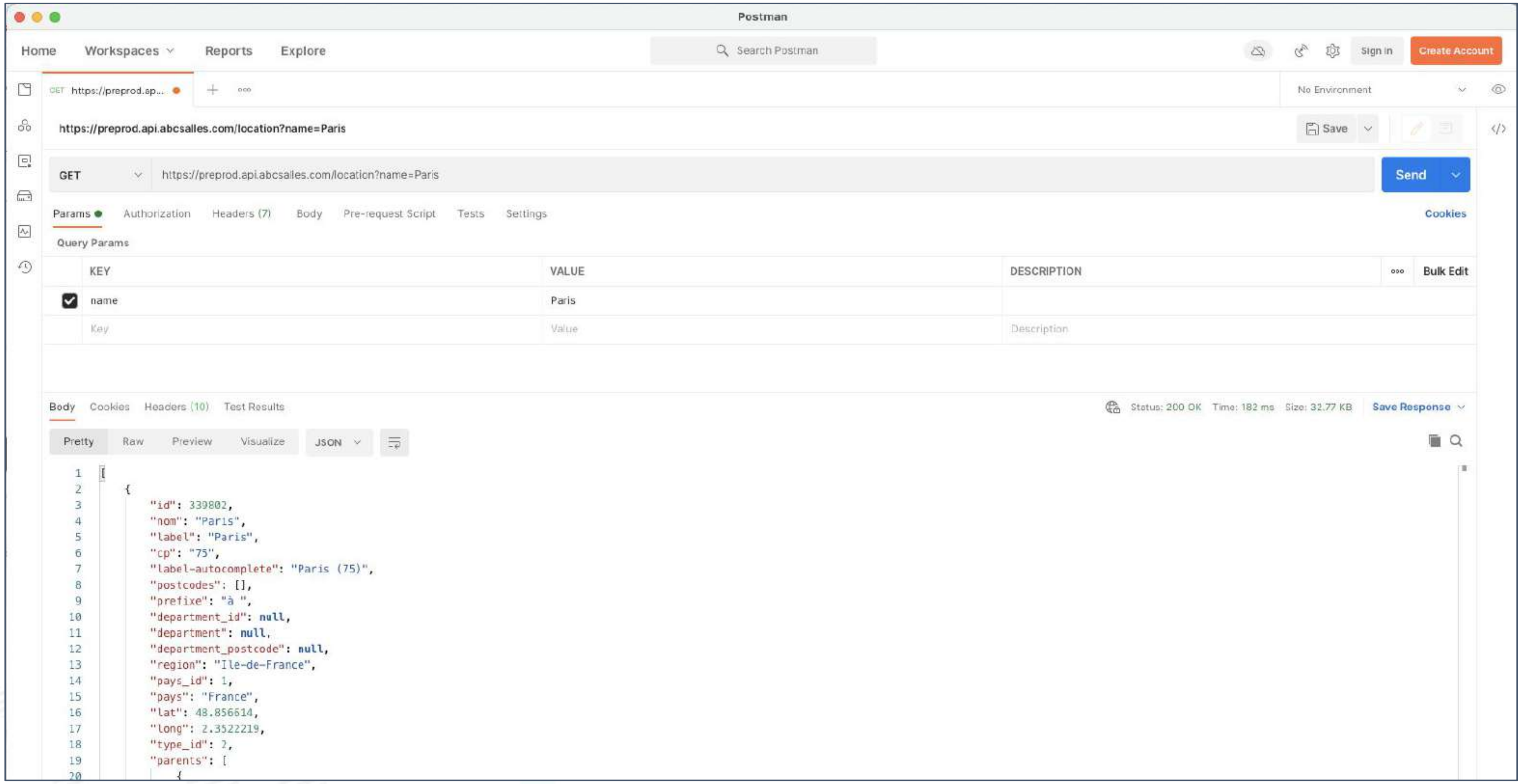
ABSTRACT PROTOCOL FLOW



05

POSTMAN

TCM://



https://preprod.api.abcsalles.com/location?name=Paris

GET https://preprod.api.abcsalles.com/location?name=Paris

	KEY	VALUE	DESCRIPTION
✓	name	Paris	
	Key	Value	Description

```
[
    {
        "id": 339802,
        "nom": "Paris",
        "label": "Paris",
        "cp": "75",
        "label-autocomplete": "Paris (75)",
        "postcodes": [],
        "prefixe": "à ",
        "department_id": null,
        "department": null,
        "department_postcode": null,
        "region": "Ile-de-France",
        "pays_id": 1,
        "pays": "France",
        "lat": 48.856614,
        "long": 2.3522219,
        "type_id": 2,
        "parents": [
            {
```

Postman is a software that can be used to test API (especially REST but also GraphQL). It is very easy to use.

06

PHP - VCR

THE NEED

- My project interfaces with third-party systems

API calls

WS calls

...

- I need a stable environment to test

And

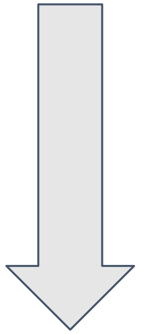
- I always need to get the same results when I request for my integration tests

But ...

I don't have control over the API

POSSIBLE SOLUTION :

Create an API “mock”



TOO LONG!

THE SOLUTION

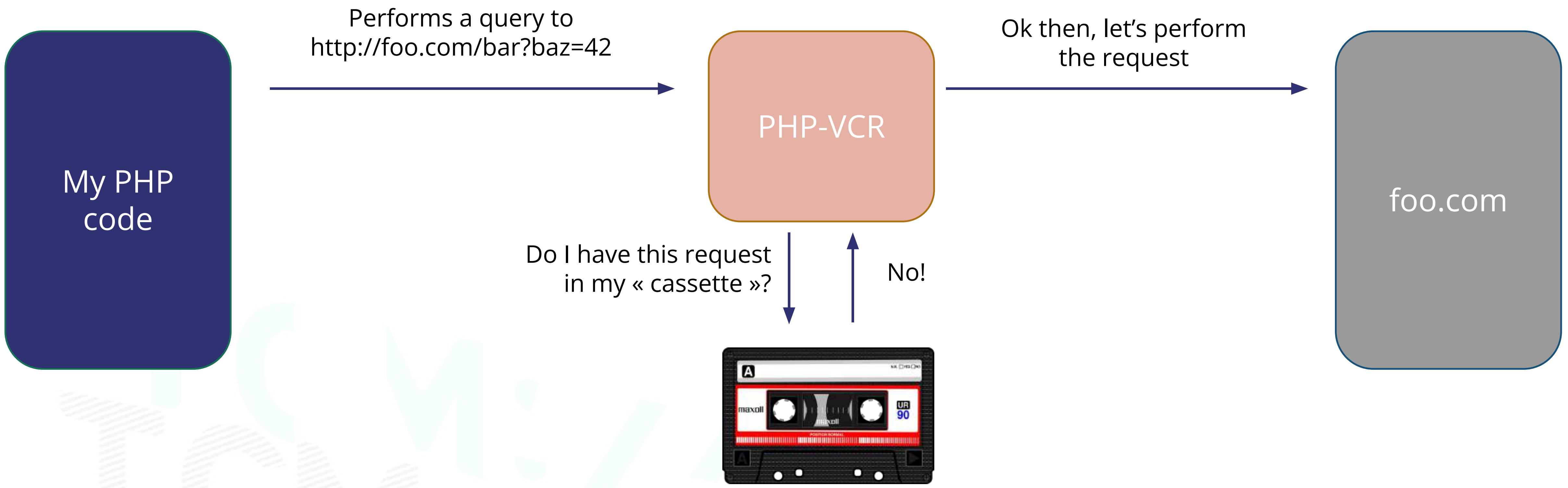
PHP-VCR!

A PHP package that we install in the project.

```
composer require --dev phpvcr/phpvcr
```

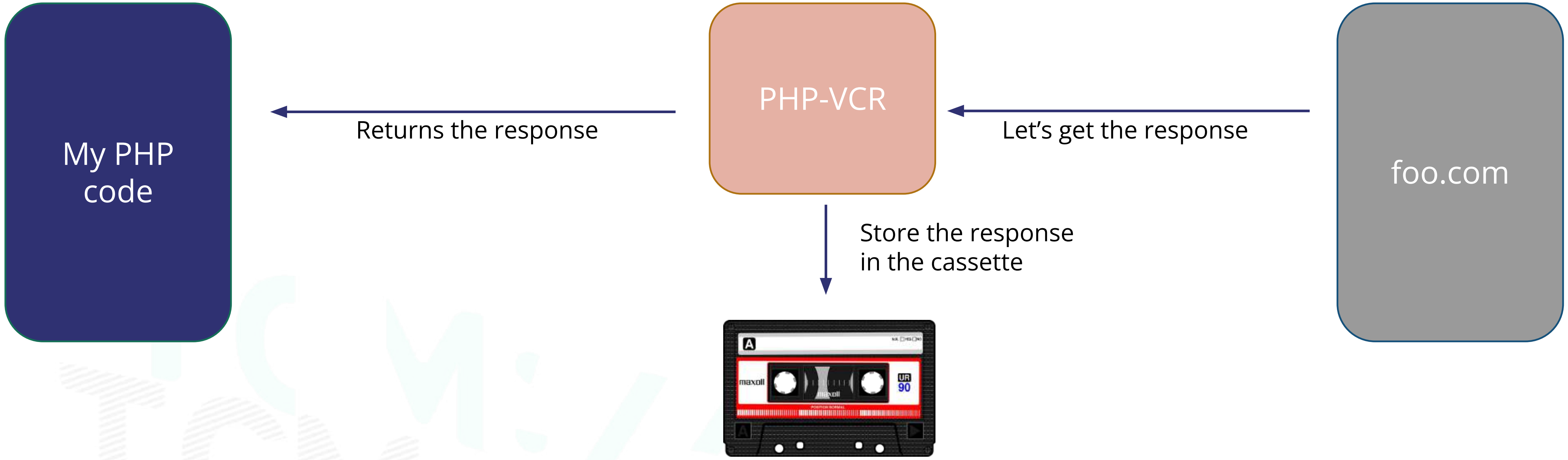
PHP-VCR records the requests made, and is able to replay them.
GitHub : https://github.com/php-vcr/php-vcr

IN PRACTICE : 1ST RUN

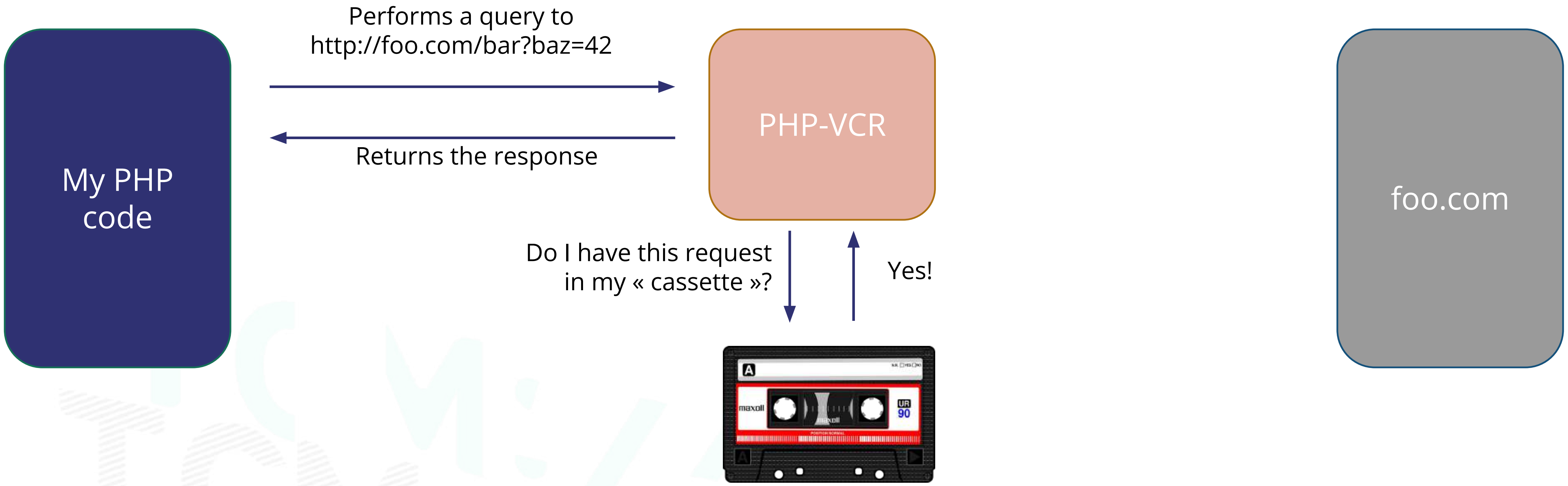

My PHP code
Performs a query to http://foo.com/bar?baz=42
PHP-VCR
Do I have this request in my « cassette »?
No!
Ok then, let's perform the request
foo.com

Note: PHP-VCR can « hijack » any call that uses curl, http sockets or SoapClient!

IN PRACTICE : 1ST RUN

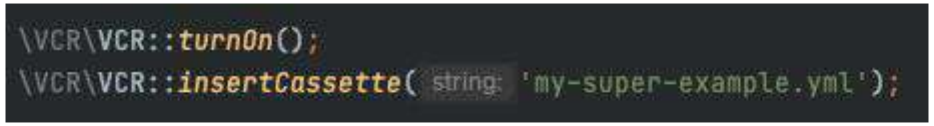


IN PRACTICE : 2ST RUN



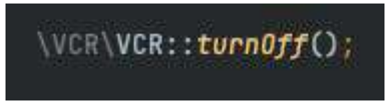
USAGE

Start PHP-VCR:



```
\VCR\VCR::turnOn();
\VCR\VCR::insertCassette( string: 'my-super-example.yml');
```

Stop PHP-VCR (write the cassette):



```
\VCR\VCR::turnOff();
```

07

ASYNCHRONOUS TASKS

BATCH

- Script allowing to carry out important / expensive treatments
  - Importing data
  - Updating data across the whole database
- Planning with CRON
- PHP : Symfony Console / Mouf Console and different configuration (php.ini)

WARNING

- VOLUMETRY
- MEMORY CONSUMPTION / RUN TIME
- RESUME ON ERROR
- REPORTING OF EACH COMMAND

RABBITMQ

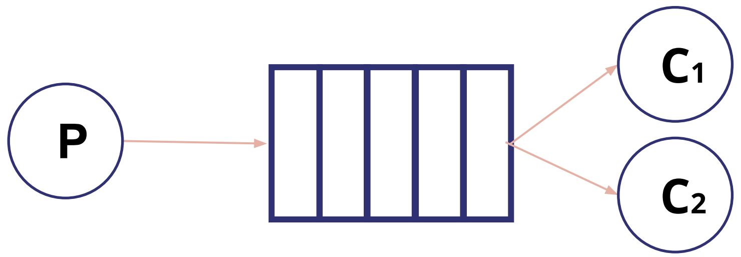
RabbitMQ is a message broker based on the AMQP standard in order to communicate with different customers.

It allows for example to:

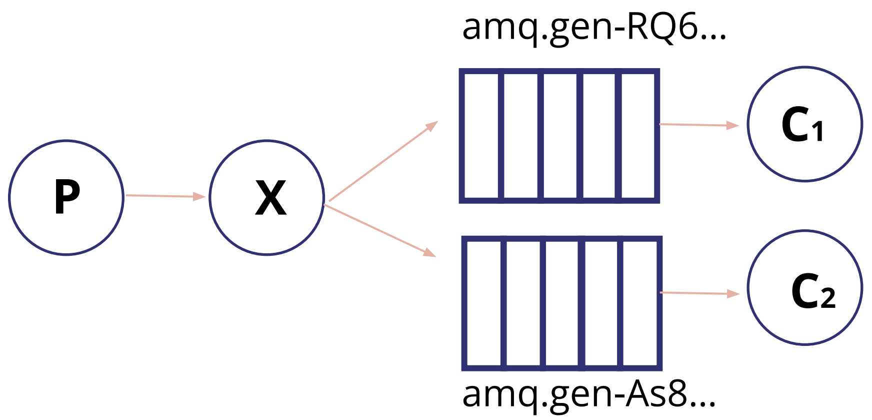
- Deport the execution of a task asynchronously (ex: send mail, upload file, delete cache...)
- Perform a task in several specific services (Publish/Subscribe)
- Manage errors and downtime

When working with AWS stack: ***SQS*** *(Simple Queue Service)*

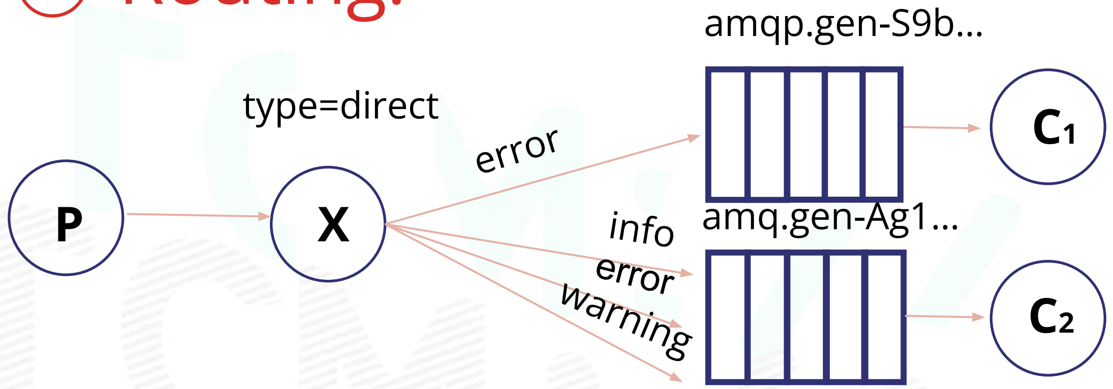
Work queue:



Publish/Subscribe:

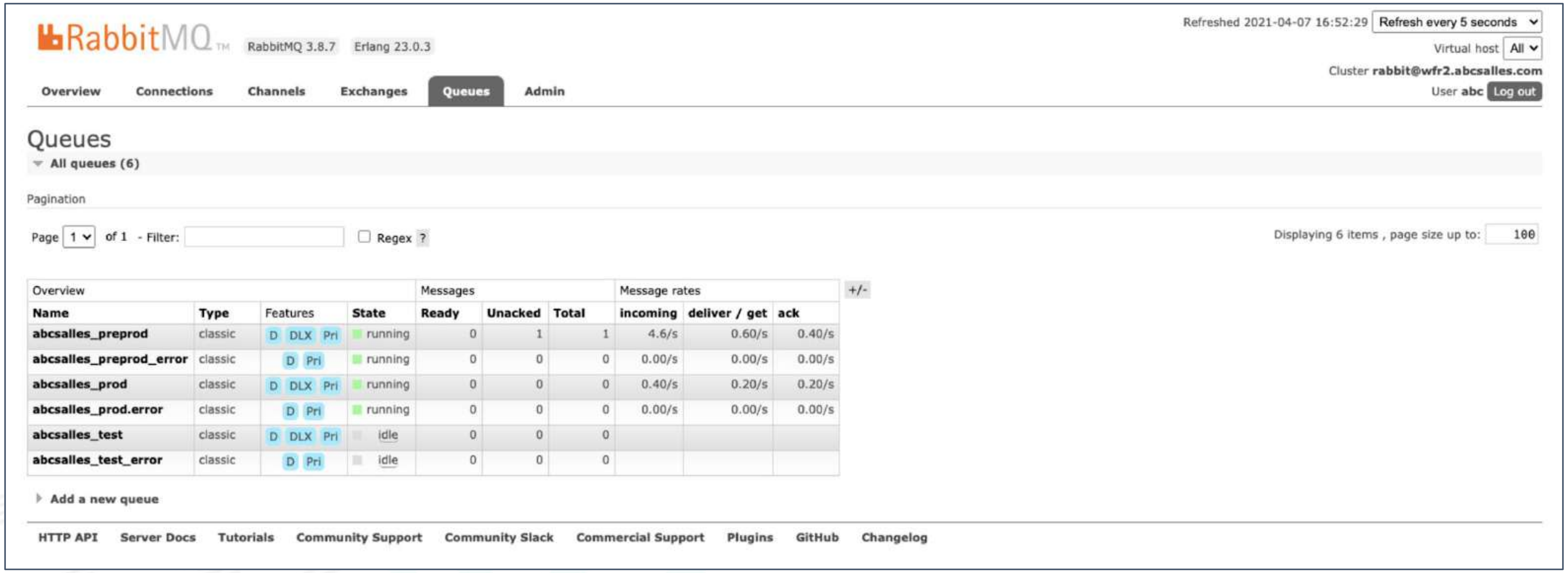


Routing:



P : Producer
C : Consumer
X : Exchange

RABBITMQ: THE MANAGEMENT INTERFACE



Refreshed 2021-04-07 16:52:29 Refresh every 5 seconds

Virtual host All

Cluster rabbit@wfr2.abcsalles.com

User abc

Queues

All queues (6)

Pagination

Page 1 of 1 - Filter: Regex ?

Displaying 6 items , page size up to: 100

Overview				Messages			Message rates		
Name	Type	Features	State	Ready	Unacked	Total	incoming	deliver / get	ack
abcsalles_preprod	classic	D DLX Pri	running	0	1	1	4.6/s	0.60/s	0.40/s
abcsalles_preprod_error	classic	D Pri	running	0	0	0	0.00/s	0.00/s	0.00/s
abcsalles_prod	classic	D DLX Pri	running	0	0	0	0.40/s	0.20/s	0.20/s
abcsalles_prod.error	classic	D Pri	running	0	0	0	0.00/s	0.00/s	0.00/s
abcsalles_test	classic	D DLX Pri	idle	0	0	0			
abcsalles_test_error	classic	D Pri	idle	0	0	0			

SUM UP

- There isn’t a typical web architecture, each application has its own requirements and its own specificities
- APIs are the core of any application (especially the growing ones) : REST is the most common one. More modern protocols exist : GraphQL, gRPC…
- Many tools exist to ease the development of web services
- In order to handle heavy treatments asynchronously, a queue service (RabbitMQ/SQS) can be set up

At TCM, we work with every technology described in this presentation (and many others !)

Thank you!

Any questions?

Thibault Balmette
t.balmette@thecodingmachine.com

contact@thecodingmachine.com
www.thecodingmachine.com

TheCodingMachine
56 rue de Londres - 75008 - Paris

TCM:// | PARIS

56 rue de Londres
75008 Paris

TCM:// | LYON

35 Rue de Marseille
69007 LYON

TCM:// | HONG-KONG

20/F, Tower 535
Causeway Bay, Hong Kong

TCM:// | LISBOA

Rua da Palma, 219, 3ºEsq
1100-391 Lisboa

TCM://

THE CODING MACHINE

Spécialisée depuis 2005 dans le développement Open Source, nous assurons l’ensemble des projets qui sont au cœur de votre stratégie digitale.

Nous intervenons depuis la mise en place jusqu’à la livraison (et même au-delà) en nous adaptant à vos besoins que ce soit en mode Agile ou au Forfait.

TCM://