



Aprèn a programar una API en PHP i MySQL

amb Codeigniter

09/2024

Ismael Kale



La nostra missió

Impulsar la política econòmica i el desenvolupament local per promoure la millora de la qualitat de vida de les ciutadanes i ciutadans de Barcelona a través del foment de l'ocupació, l'impuls de l'emprenedoria i el suport a les empreses, responent a les diferents necessitats de les persones en el seu territori i des de la perspectiva de l'economia plural, que inclou, entre altres, l'economia social i solidària.

Potenciar l'empoderament de la ciutadania i el reequilibri entre territoris per assolir un model just de desenvolupament econòmic, així com de creació, manteniment i repartiment de l'ocupació.





Què fem?

Acompanyem la ciutadania durant tot el procés de recerca de feina

barcelonactiva.cat/treball

Donem suport a les persones emprenedores per fer realitat la seva idea de negoci, sigui de forma col·lectiva, comunitària o individual

barcelonactiva.cat/emprenedoria

Ajudem les empreses i organitzacions a créixer, connectar-se amb l'ecosistema i consolidar-se amb models socialment responsables

barcelonactiva.cat/empreses

I tot això ho fem en clau de territori, incloent la perspectiva de gènere i la diversitat des d'una visió d'economia plural

Oferim formació tecnològica a les persones que cerquen feina, emprenedores i professionals

barcelonactiva.cat/cibernarium



Equipaments al servei de la ciutat



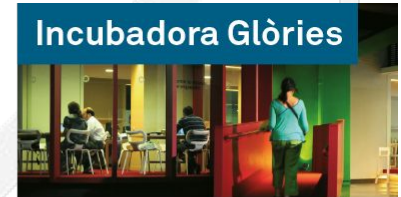
Porta22



Cibernàrium



OAE



Incubadora Glòries



Xarxa de proximitat



Incubadora Almogàvers



Parc Tecnològic



Centre Iniciativa Emprenedora



Ca n'Andalet



innoBA
Innovació Socioeconòmica



Convent de St. Agustí



Incubadora MediaTIC



Nou Barris Activa



ESABIC Barcelona



Barcelona Activa, molt present als barris





01

Què és una API i quins tipus existeixen?



Què és una API?

Es defineix com una interfície que afavoreix la comunicació entre dos sistemes o plataformes diferents.

Existeixen multitud de plataformes per desenvolupar una API: (.NET, Java, Node.js, PHP...)

En el nostre cas crearem una API en PHP.

Exemples de API:

- APIs de Google Maps
- API de Pokemon (<https://pokeapi.co/api/v2/pokemon/>)
- API d'Acudits (<https://api.chucknorris.io/jokes/random>)
- API de BarcelonActiva
-



Què és POSTMAN?

- Ens permet realitzar peticions d'una manera simple per testear API de tipus REST, siguin pròpies o de tercers. <https://www.postman.com/downloads/>

The screenshot shows the Postman interface with a GET request to `https://pokeapi.co/api/v2/pokemon/`. The response is a JSON object with the following structure:

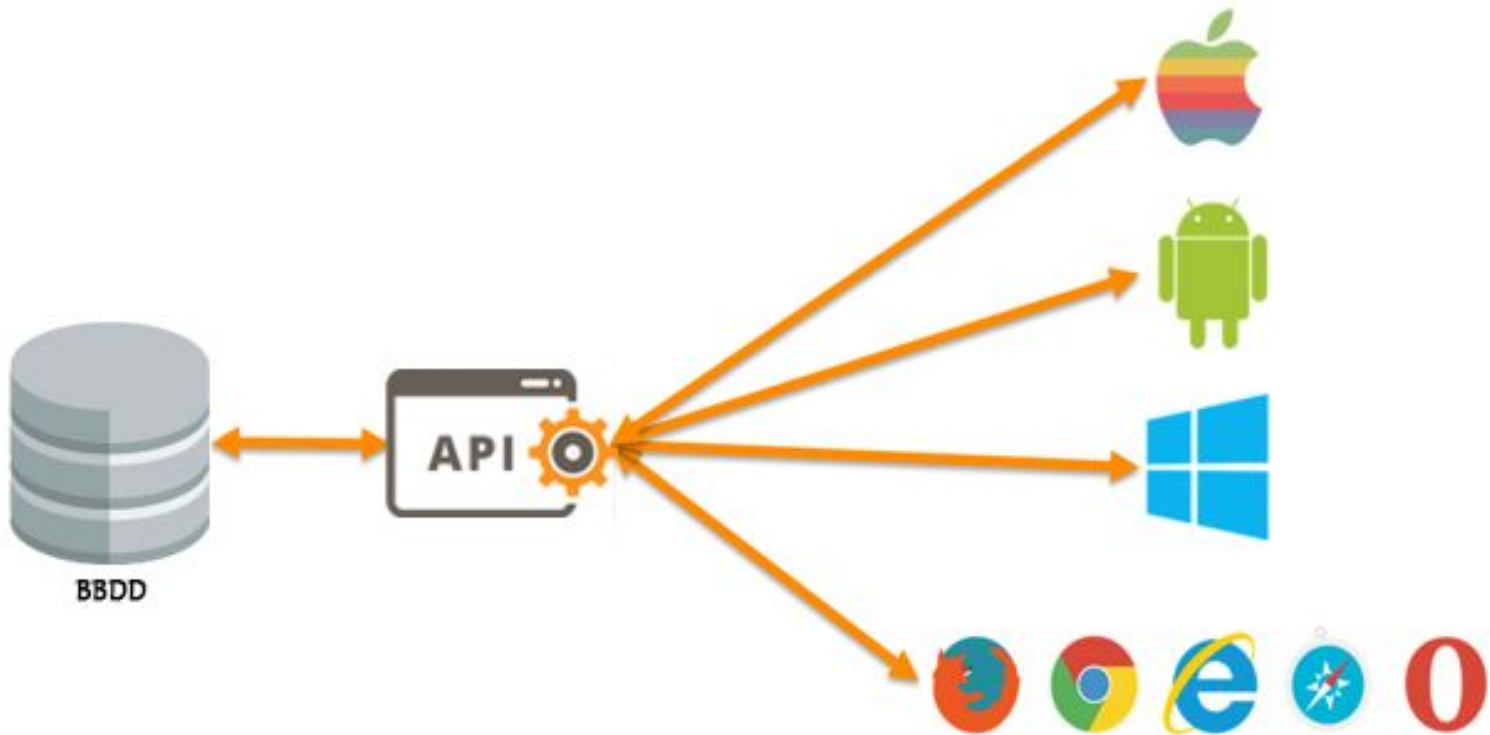
```
1 {
2   "count": 1154,
3   "next": "https://pokeapi.co/api/v2/pokemon/?offset=20&limit=20",
4   "previous": null,
5   "results": [
6     {
7       "name": "bulbasaur",
8       "url": "https://pokeapi.co/api/v2/pokemon/1/"
9     },
10    {
11      "name": "ivysaur",
12      "url": "https://pokeapi.co/api/v2/pokemon/2/"
13    },
14    {
15      "name": "venusaur",
```

The interface includes tabs for Params, Authorization, Headers (6), Body, Pre-request Script, Tests, and Settings. The Params tab is active, showing a table with columns KEY, VALUE, and DESCRIPTION. The Body tab is also active, showing the JSON response in Pretty format. The status bar at the bottom indicates a 200 OK response with a response time of 84 ms and a size of 2.63 KB.



Com funciona una API?

- Els diferents dispositius es connecten a una API centralizada que conté metodes com:
 - Crear Usuari, Editar Usuari i Eliminar Usuari.



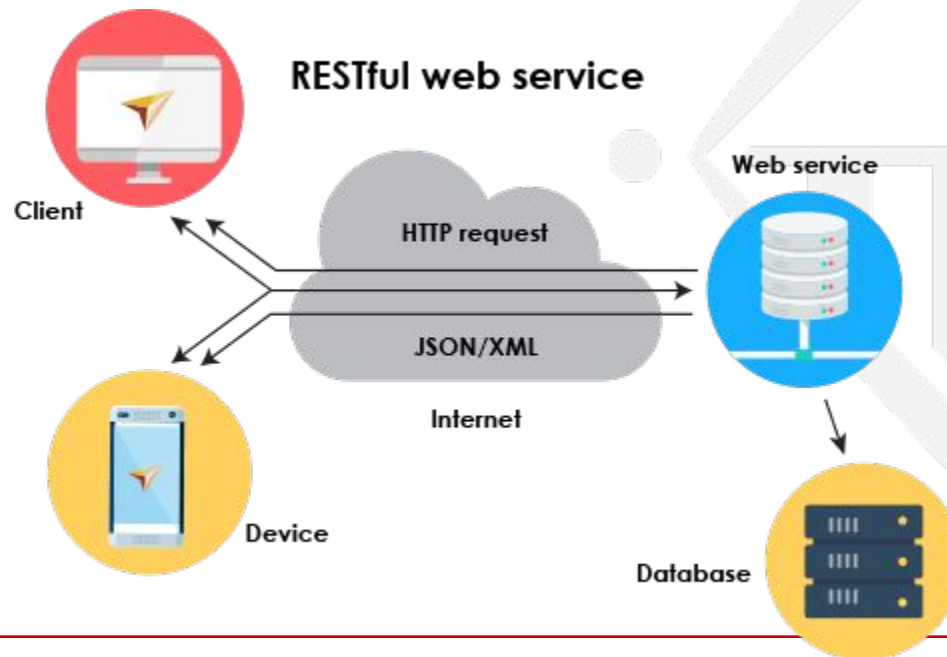


Tipus de API

Existeixen diferents tipus de API: REST, SOAP, WebSocket i RPC

- <https://aws.amazon.com/es/what-is/api/>

En el nostre utilitzarem API REST, el flexible i utilitzat.





Què es millor XML o JSON?

- **XML** és un format marcado similar a HTML.
- **JSON** és un format estàndard per a dades que destaca per ser lleuger i ràpid. Les dades en format JSON poden ser utilitzades per pràcticament tots els llenguatges de programació (com Java, C#, C, C++, PHP, JavaScript, Python, etc.).

XML

```
<empinfo>
  <employees>
    <employee>
      <name>James Kirk</name>
      <age>40</age>
    </employee>
    <employee>
      <name>Jean-Luc Picard</name>
      <age>45</age>
    </employee>
    <employee>
      <name>Wesley Crusher</name>
      <age>27</age>
    </employee>
  </employees>
</empinfo>
```

JSON

```
{ "empinfo" :
  {
    "employees" : [
      {
        "name" : "James Kirk",
        "age" : 40,
      },
      {
        "name" : "Jean-Luc Picard",
        "age" : 45,
      },
      {
        "name" : "Wesley Crusher",
        "age" : 27,
      }
    ]
  }
}
```

Notació literal d'objectes de JavaScript



Per crear la nostra API REST...

Utilitzarem el Framework CodeIgniter...





02

Introducció als Frameworks



Què és un framework?

Un framework és un programa per desenvolupar altres programes/API

Com ens ajuda CodeIgniter?

CodeIgniter, per tant, és un framework(programa) o aplicació web desenvolupat en PHP.

És de codi lliure y té una curva d'aprenentatge molt senzilla.





Quins son els seus avantatges?

Desenvolupament ràpid: ens permet l'obtenció del seu producte mínim viable el més ràpid possible. PHP ajuda bastant en aquesta tasca.

Seguretat: Totes les validacions de dades ja estan preparades per ser segures.

Helpers i altres ajudes: Les funcions més utilitzades ja estan disponibles de manera nativa. **Ex: Una validació de email**

Limita la teva llibertat: evita que cada programador escrigui el seu codi de forma lliure y desorganitzada.



¿Com editarem el codi?

- Notepad++
- SublimeText
- VisualStudioCode (millor)



 Sublime Text





03

Instal·lació del XAMPP (Apache+Mysql+Php)



Com treballlem?

Ens caldrà instal·lar una base de dades juntament amb algun gestor de base de dades.

El programa WAMP o el **XAMPP** ens instal·larà tot el necessari per començar a treballar amb un servidor en local.

Accedirem per defecte desde <http://localhost>



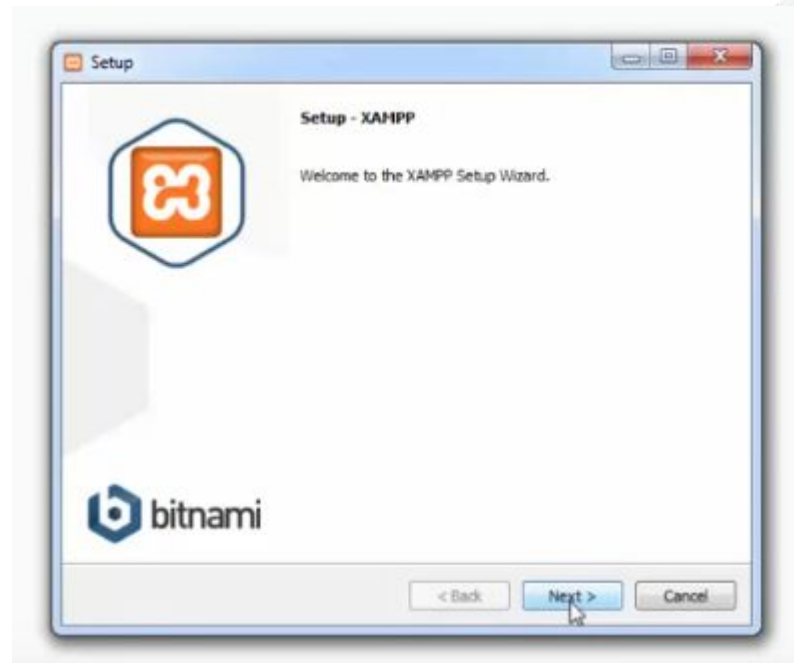
Aprèn a programar una API en PHP i MySQL

Instal·lació del Servidor Xampp

Pas 1: Descarregar e instal·lar el Xampp

Es pot descarregar gratuïtament a la web:

<https://www.apachefriends.org/es/index.html>

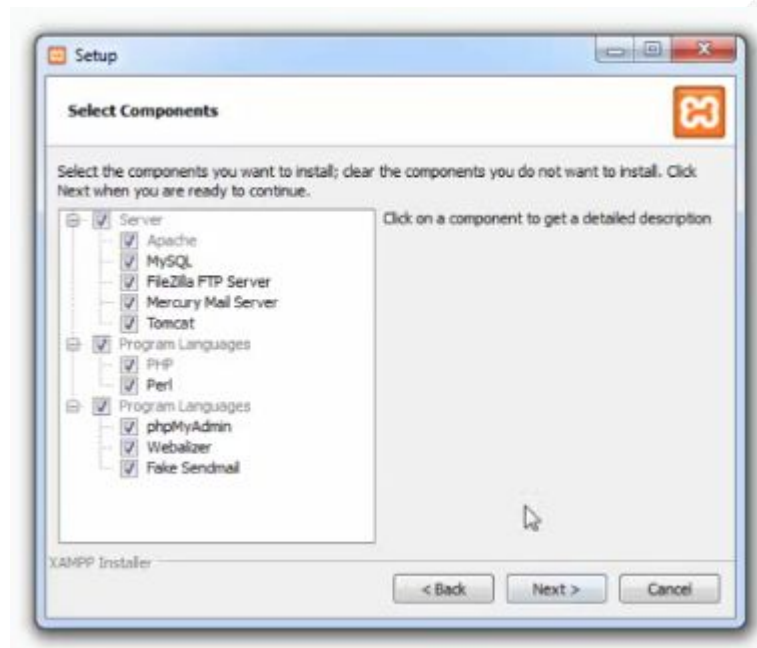




Instal·lació del Servidor Xampp

Pas 2: Configurar el servidor

Apareixen diferents components per instal·lar, ens interessa sobretot instal·lar el Apache, PHP, el MySQL i el phpMyAdmin.

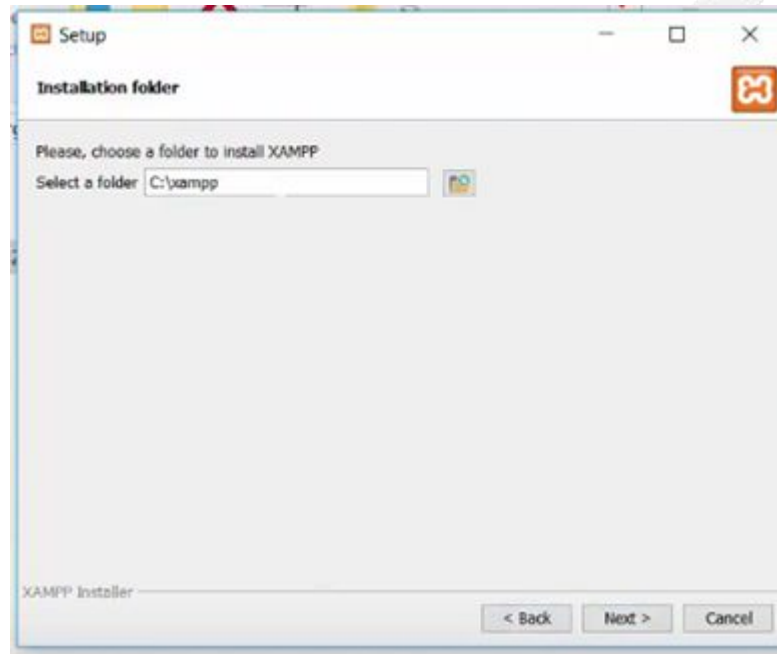




Instal·lació del Servidor Xampp

Pas 3: Seleccionar la carpeta d'instal·lació

Totes les pàgines web, prestashop, wordpress, phpMyAdmin... S'instal·laran a la carpeta seleccionada, per defecte, a la carpeta C:/xampp:





Instal·lació del Servidor Xampp

Pas 4: Activem el servidor

Premem el botó "Start", en els serveis Apache i MySQL.

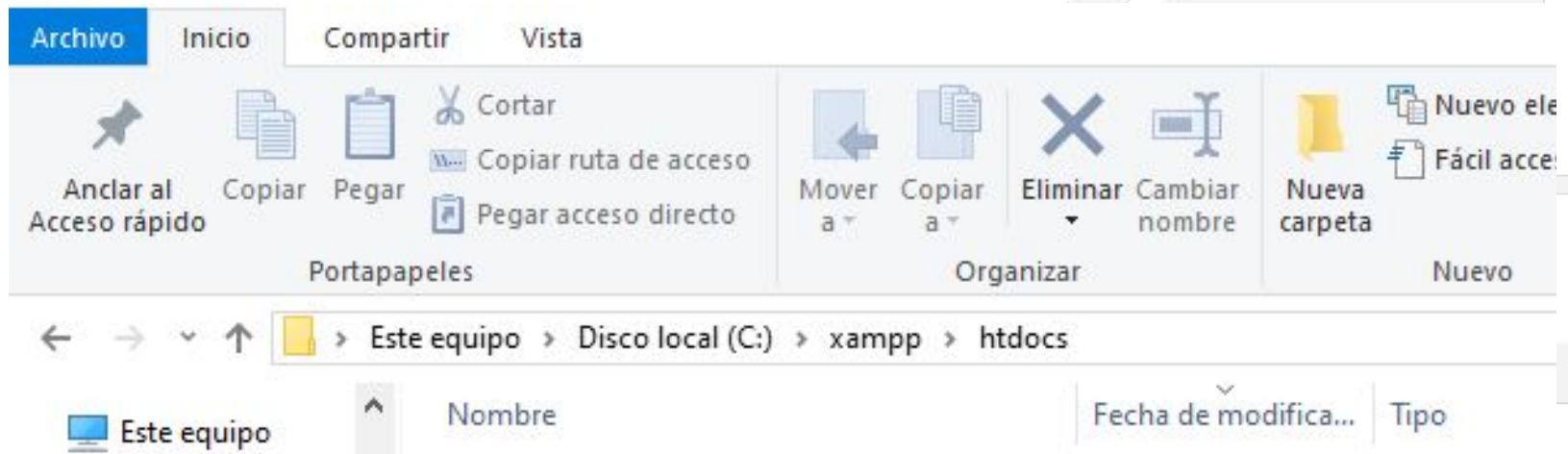




On s'ha instal·lat?

Els nostres fitxers del Framework aniran a:

Carpeta `C:\xampp\htdocs`





04

Bases de dades amb MySQL



Creació de la Base de dades

Servidor: 127.0.0.1

Bases de datos SQL Estado actual Cuentas de usuarios E

Bases de datos

Crear base de datos ⓘ


my_api utf8mb4_general_ci ▼

Crear



Creació d'una taula

id_book	isbn	title	description	author	created_at	updated_at
1	4353245345	Los 3 cerditos	En el corazón del bosque vivían los tres cerditos,...	Joseph Garcia	2022-07-13 11:48:02	2022-07-13 09:22:14

 Crear tabla

Nombre:

Número de columnas:

Continuar



Tipus de dades

Dades de tipo numèriques:

- **INT**: Ocupació de 4 bytes amb valors entre -2147483648 i 2147483647 o entre 0 i 4294967295.
- **SMALLINT**: Ocupació de 2 bytes amb valors entre -32768 i 32767 o entre 0 i 65535.
- **TINYINT**: Ocupació de 1 bytes amb valors entre -128 i 127 o entre 0 i 255.
- **DECIMAL**: Emmagatzema números amb decimals.

Dades de tipo dates:

- **DATE**: Emmagatzema una data amb any, mes i dia. El rang oscil·la entre '1000-01-01' i el '9999-12-31'.
- **DATETIME**: Emmagatzema una data (any, mes i dia) i una hora (hora-minut-segon), el seu rang oscil·la entre '1000-01-01 00:00:00' i '9999-12-31 23:59:59'.
- **TIMESTAMP**: Emmagatzema una data i hora UTC. El rang oscil·la entre '1970-01-01 00:00:01' i '2038-01-19 03:14:07'.

Dades de tipo text:

- **VARCHAR**: El rang oscil·la entre 1 a 65.535 caràcters.
- **TEXT**: Longitud màxima de 65.535 caràcters. Emmagatzema text, sense format. Distingeix entre minúscules i majúscules.



Creació d'una taula

Servidor: 127.0.0.1 » Base de datos: my_api » Tabla: books

Examinar Estructura SQL Buscar Insertar Exportar Importar Más

Estructura de tabla Vista de relaciones

#	Nombre	Tipo	Cotejamiento	Nulo	Predeterminado	Extra
<input type="checkbox"/> 1	id_book	int(6)		No	Ninguna	AUTO_INCREMENT
<input type="checkbox"/> 2	isbn	varchar(10)	utf8mb4_general_ci	No	Ninguna	
<input type="checkbox"/> 3	title	varchar(100)	utf8mb4_general_ci	No	Ninguna	
<input type="checkbox"/> 4	description	text	utf8mb4_general_ci	No	Ninguna	
<input type="checkbox"/> 5	author	varchar(100)	utf8mb4_general_ci	No	Ninguna	
<input type="checkbox"/> 6	created_at	datetime		No	current_timestamp()	
<input type="checkbox"/> 7	updated_at	datetime		No	current_timestamp()	ON UPDATE CURRENT_TIMESTAMP()



Importar una base de dades (Descarregar)



Importando en la base de datos "my_api"

Archivo a importar:

El archivo puede ser comprimido (gzip, bzip2, zip) o descomprimido.

Un archivo comprimido tiene que terminar en **[formato].[compresión]**. Por ejemplo: **.sql.zip**

Buscar en su ordenador: Sin archivos...leccionados (Máximo: 40MB)

También puede arrastrar un archivo en cualquier página.

Conjunto de caracteres del archivo:

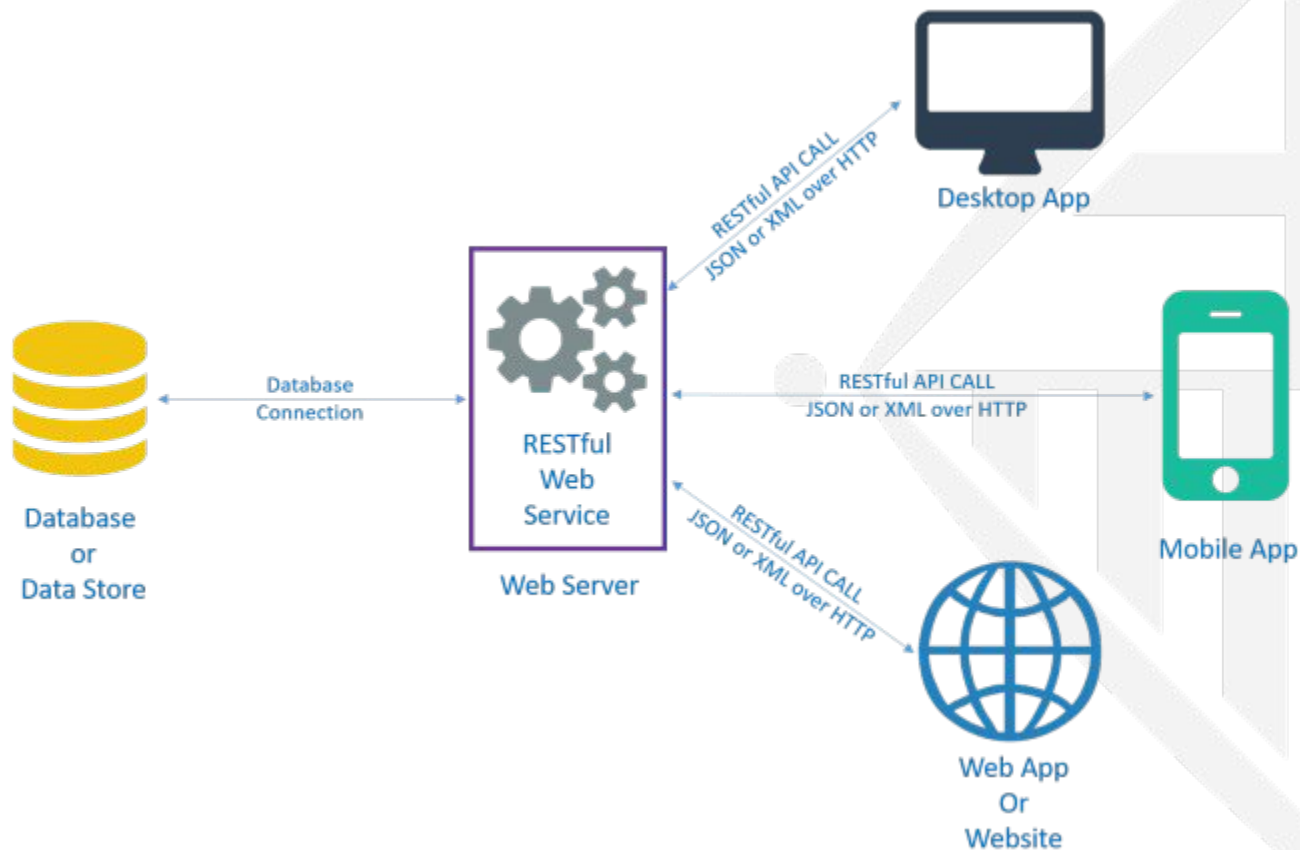


Passos per crear una taula

1. Definir nom de la taula i els seus camps
2. Decidir la clau Primària i si serà autoincrement
3. Decidir si hi ha clau Única
4. Definir dades per defecte i marcar les possibles campos NULL



Ara ja tenim una base de dades





05

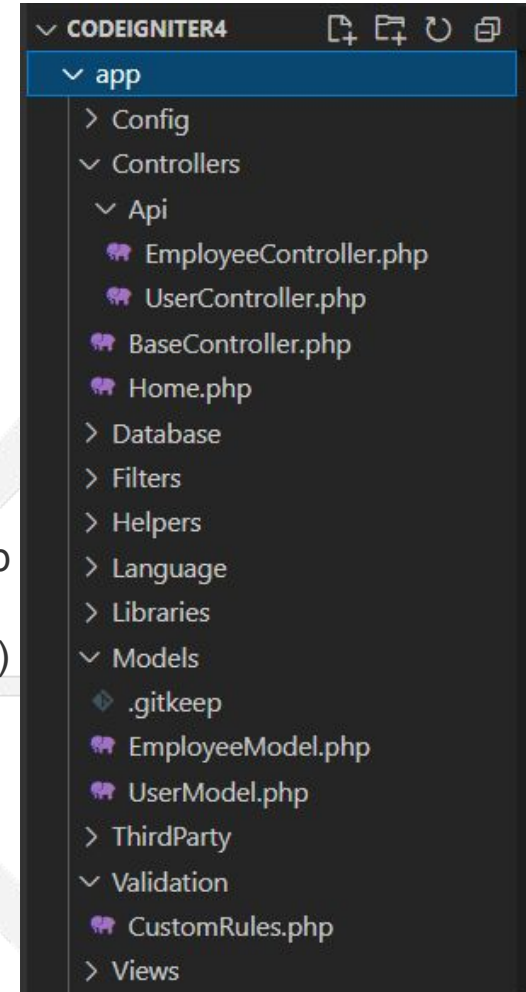
Instal·lació de Codeigniter



Com instal·lar Codeigniter

La instal·lació de Codeigniter és molt senzilla, només has de seguir els següents passos:

- **Descarrega** la última versió de Codeigniter de la [pàgina oficial](#) o la [versió adaptada per API](#).
- **Descomprimeix** el .rar que has descarregat i còpia el seu contingut dins del projecte situat a C:/xampp/htdocs/api_php
- **composer install** (si no tenim composer, ho [descarreguem](#))
- Ja has instal·lat Codeigniter i està llest per començar a treballar amb ell. L'estructura del teu projecte ha d'haver quedat així:



Obrim navegador amb: http://localhost/api_php/public/

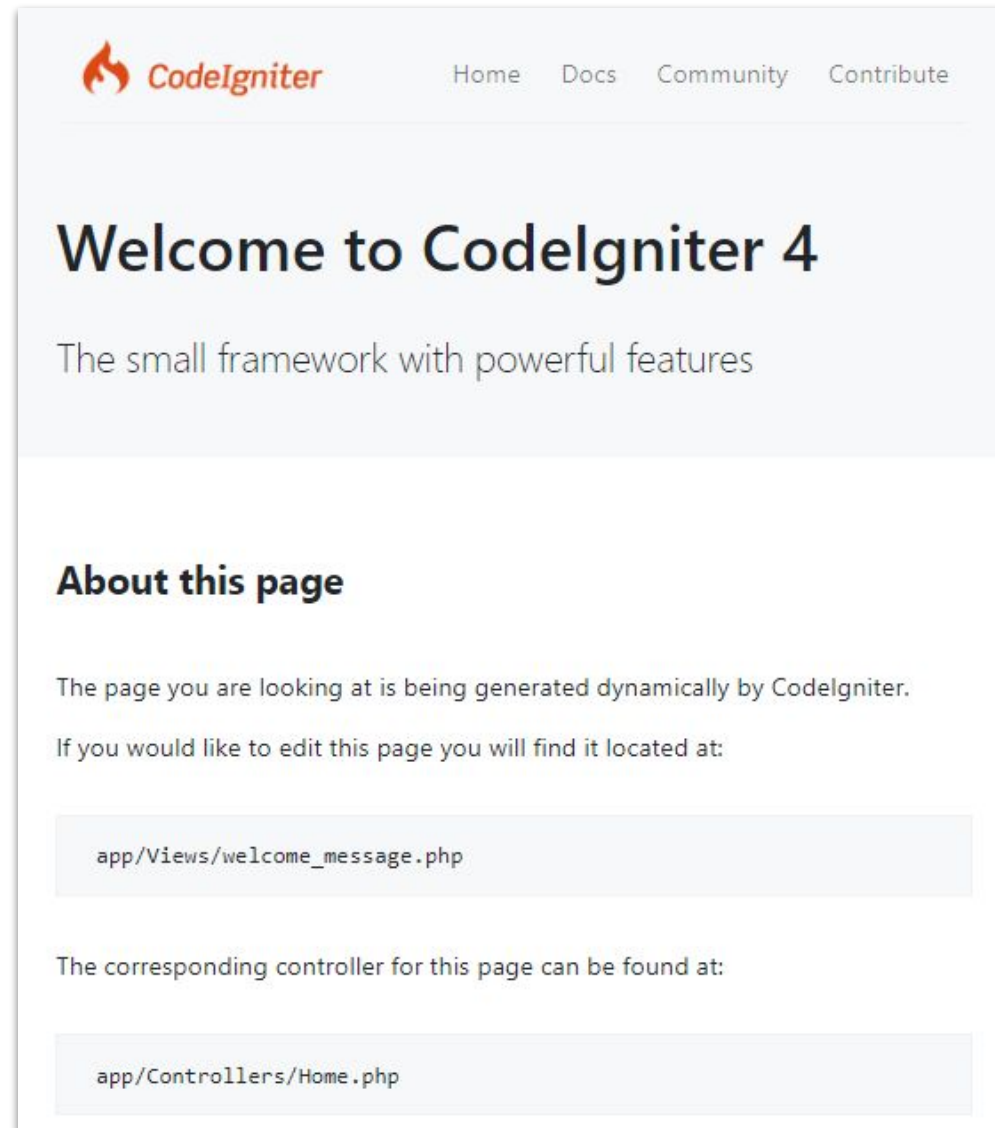


Aprèn a programar una API en PHP i MySQL

Ja està instal·lat!!

Per accedir a la nostra web:

http://localhost/api_php/public/

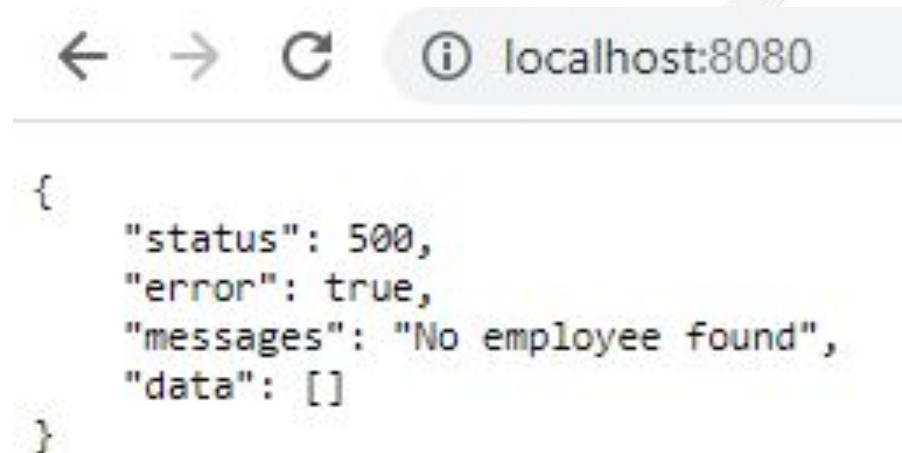




Activem el nostre servidor

php spark serve

<http://localhost:8080>





Problemes amb el CORS (opcional)

Solucionar problema con el CORS:

<https://onlinewebtutorblog.com/how-to-enable-cors-in-codeigniter-4-for-rest-apis/>



Problemes amb el Intl y zip (opcional)

XAMPP Control Panel v3.3.0 [Compiled: Apr 6th 2021]

XAMPP Control Panel v3.3.0

Service	Module	PID(s)	Port(s)	Actions
<input type="checkbox"/>	Apache	4016 924	80, 443	Stop Admin Config
<input type="checkbox"/>	MySQL	7128	3306	Stop Admin Cor
<input type="checkbox"/>	FileZilla			Start Admin Cor
<input type="checkbox"/>	Mercury			Start Admin Cor
<input type="checkbox"/>	Tomcat			Start Admin Cor

9:20:39 [main] The Mercury module is disabled
9:20:39 [main] The Tomcat module is disabled
9:20:39 [main] Starting Check-Timer

Config menu options:

- Apache (httpd.conf)
- Apache (httpd-ssl.conf)
- Apache (httpd-xampp.conf)
- PHP (php.ini)**
- phpMyAdmin (config.inc.php)
- <Browse> [Apache]
- <Browse> [PHP]
- <Browse> [phpMyAdmin]

```
extension=gettext  
;extension=gmp  
extension=intl  
;extension=imap  
extension=mbstring  
extension=exif ; Mus  
extension=mysqli  
;extension=oci8_12c ; Us  
;extension=oci8_19 ; Use  
;extension=odbc  
;extension=openssl  
;extension=pdo_firebird  
extension=pdo_mysql  
;extension=pdo_oci  
;extension=pdo_odbc  
;extension=pdo_pgsql  
extension=pdo_sqlite  
;extension=pgsql  
;extension=shmop  
  
; The MIBS data available  
; See https://www.php.net  
;extension=snmp  
  
;extension=soap  
;extension=sockets  
;extension=sodium  
;extension=sqlite3  
;extension=tidy  
;extension=xsl  
extension=zip
```

RESET APACHE



Estructura de CodeIgniter

Ens centrarem amb les carpetes

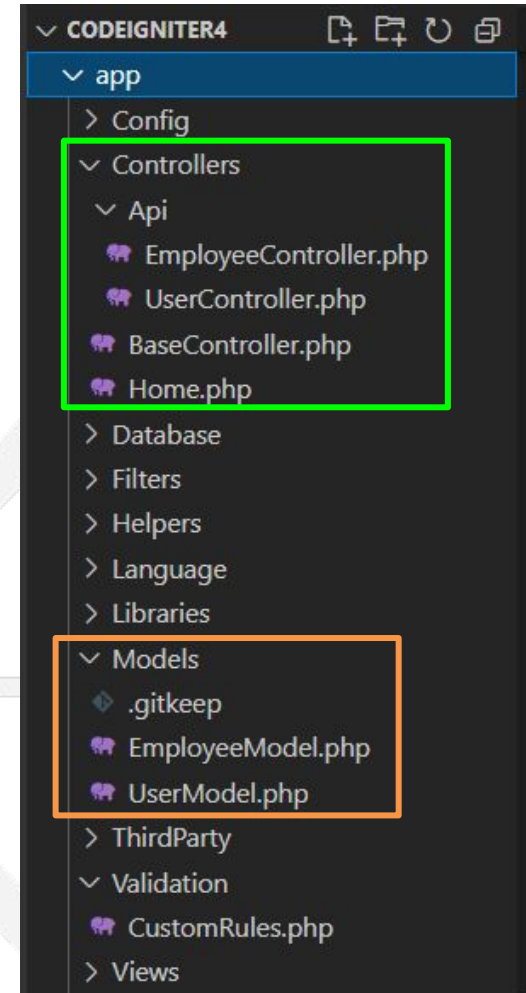
controllers o models

controllers

És el responsable de la lògica de negoci:
entrada d'informació, validacions de dades.
Connecta directament amb el Model.

models

És el responsable de la capa de dades, que
connecta amb la base de dades.





06

amb POSTMAN



Anàlisi amb POSTMAN

```
$routes->get('/', 'HomeController::index');
```

```
namespace App\Controllers;
use CodeIgniter\RESTful\ResourceController;

class HomeController extends ResourceController
{
    public function index()
    {
        $response = [
            'status' => 500,
            'error' => true,
            'messages' => 'No employee found',
            'data' => []
        ];
        return $this->respond($response);
    }
}
```

```
{
  "status": 500,
  "error": true,
  "messages": "No employee found",
  "data": []
}
```

http://localhost:8080/

GET http://localhost:8080/

Params Auth Headers (10) Body ● Pre-req. Tests Settings

Query Params

KEY	VALUE
Key	Value

Body

Pretty Raw Preview Visualize JSON

```
1 {
2   "status": 500,
3   "error": true,
4   "messages": "No employee found",
5   "data": []
6 }
```




Routing (config/routes.php)

- `http://localhost:8080/api/employee/list`
- `http://localhost:8080/api/employee/add`
- `http://localhost:8080/api/employee/show/2`
- `http://localhost:8080/api/employee/update/2`
- `http://localhost:8080/api/employee/delete/2`

```
public function showEmployee($emp_id)
{
    $emp = new EmployeeModel();

    $data = $emp->find($emp_id);
    //$data = $model->where(['id' => $

    if (!empty($data)) {
```

```
$routes->get('/', 'Home::index');

$routes->group("api", ["namespace" => "App\Controllers\Api"] , function($routes){

    $routes->group("employee", function($routes){
        $routes->get("list", "EmployeeController::listEmployee");
        $routes->post("add", "EmployeeController::addEmployee");
        $routes->get("show/(:num)", "EmployeeController::showEmployee/$1");
        $routes->put("update/(:num)", "EmployeeController::updateEmployee/$1");
        $routes->delete("delete/(:num)", "EmployeeController::deleteEmployee/$1");
    });
});
```



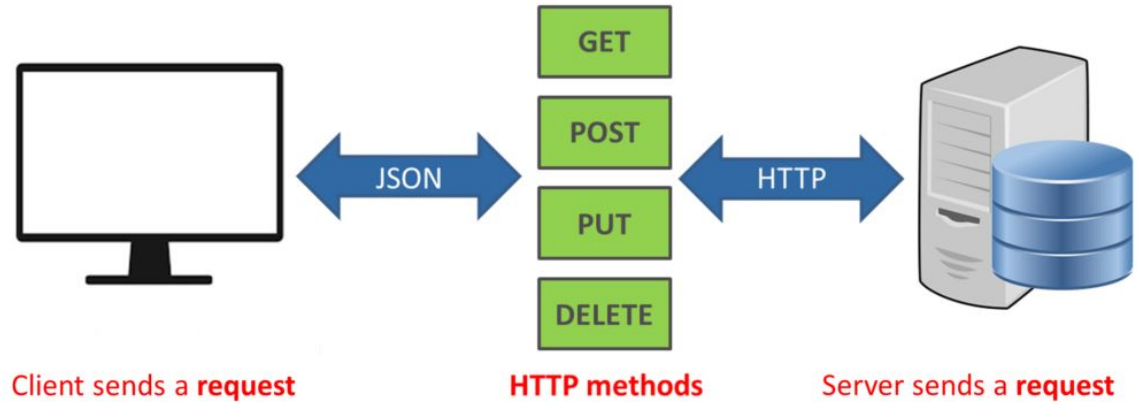
Métodos de petición HTTP

GET (list, show)

POST (add)

PUT (update)

DELETE (delete)



+INFO

<https://developer.mozilla.org/es/docs/Web/HTTP/Methods>

<https://www.ionos.es/digitalguide/hosting/cuestiones-tecnicas/http-request/>



HTTP status Codes

GET

- + 200 OK
- + 400 Bad Request
- + 401 Unauthorized
- + 403 Forbidden
- + 404 Not Found
- + 500 Internal Server Error

POST

- + 201 Created
- + 400 Bad Request
- + 401 Unauthorized
- + 403 Forbidden
- + 409 Conflict
- + 500 Internal Server Error

PUT

- + 200 OK
- + 201 Created*
- + 204 No Content
- + 400 Bad Request
- + 401 Unauthorized
- + 403 Forbidden
- + 500 Internal Server Error

DELETE

- + 204 No Content
- + 400 Bad Request
- + 401 Unauthorized
- + 403 Forbidden
- + 500 Internal Server Error



ADD

- <http://localhost:8080/api/employee/add>

The screenshot shows a REST client interface with the following details:

- URL:** `http://localhost:8080/api/employee/add`
- Method:** `POST`
- Body Type:** `x-www-form-urlencoded`
- Body Fields:**

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> name	Ismael	
<input checked="" type="checkbox"/> email	ismael@ismael.cat	
<input checked="" type="checkbox"/> phone_no	685458742	
Key	Value	Description
- Response:**

```
1 {
2   "status": 200,
3   "error": false,
4   "message": "Employee added successfully",
5   "data": []
6 }
```



UPDATE

- <http://localhost:8080/api/employee/update/1>

The screenshot shows a REST client interface with the following details:

- URL:** `http://localhost:8080/api/employee/update/1`
- Method:** PUT
- Body Type:** x-www-form-urlencoded
- Body Data:**

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> name	Ismael	
<input checked="" type="checkbox"/> email	ismael@ismael.cat	
<input checked="" type="checkbox"/> phone_no	685458742	
Key	Value	Description
- Status:** 200 OK, 121 ms, 672 B
- Response Body (JSON):**

```
1 {
2   "status": 200,
3   "error": false,
4   "message": "Employee updated successfully",
5   "data": []
6 }
```



DELETE

- <http://localhost:8080/api/employee/delete/4>

The screenshot shows a REST client interface with the following components:

- URL Bar:** `http://localhost:8080/api/employee/delete/4`
- Method:** `DELETE`
- Send Button:** A blue button labeled "Send".
- Params Tab:** The "Params" tab is selected, showing "Query Params".
- Query Params Table:**

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		
- Body Tab:** The "Body" tab is selected, showing the response.
- Response Status:** `200 OK`, `121 ms`, `673 B`.
- Response Content:**

```
1 {
2   "status": 200,
3   "error": false,
4   "messages": "Employee deleted successfully",
5   "data": []
6 }
```



07

Controllers, Models i Routes



Creació d'un Model Book

- php spark make:model Book --suffix

```
namespace App\Models;
use CodeIgniter\Model;

class BookModel extends Model
{
    protected $DBGroup          = 'default';
    protected $table            = 'books';
    protected $primaryKey       = 'id_book';
    protected $useAutoIncrement = true;
    protected $insertID         = 0;
    protected $returnType       = 'array';
    protected $useSoftDeletes   = false;
    protected $protectFields    = true;
    protected $allowedFields    = [
        "isbn",
        "title",
        "description",
        "author"
    ];
}
```




Creació del Controller Book

- `php spark make:controller Api/Book --suffix --restful`

Per defecte es creen varies funcions, algunes amb paràmetre (\$id) i altres no.

```
namespace App\Controllers\Api;
use CodeIgniter\RESTful\ResourceController;

class BookController extends ResourceController
{
    public function index(){
        //per mostrar el llistat de llibres
    }
    public function show($id = null) {
        //per mostrar un llibre
    }
    public function create(){
        //per crear un llibre
    }
    public function update($id = null){
        //per actualitzar un llibre
    }
    public function delete($id = null){
        //per eliminar un llibre
    }
}
```



Routing (config/routes.php)

- <http://localhost:8080/api/book/list>
- <http://localhost:8080/api/book/add>
- <http://localhost:8080/api/book/show/2>
- <http://localhost:8080/api/book/update/2>
- <http://localhost:8080/api/book/delete/2>

Cal adaptar-ho al routing de Book

```
$routes->get('/', 'Home::index');

$routes->group("api", ["namespace" => "App\Controllers\Api"] , function($routes){

    $routes->group("employee", function($routes){
        $routes->get("list", "EmployeeController::listEmployee");
        $routes->post("add", "EmployeeController::addEmployee");
        $routes->get("show/(:num)", "EmployeeController::showEmployee/$1");
        $routes->put("update/(:num)", "EmployeeController::updateEmployee/$1");
        $routes->delete("delete/(:num)", "EmployeeController::deleteEmployee/$1");
    });
});
```



Llistat de routing de total la API

- php spark routes

Method	Route	Handler
GET	/	\App\Controllers\Home::index
GET	api/employee/list	\App\Controllers\Api\EmployeeController::listEmployee
GET	api/employee/show/([0-9]+)	\App\Controllers\Api\EmployeeController::showEmployee/\$1
GET	api/profile	\App\Controllers\Api\UserController::details
POST	api/employee/add	\App\Controllers\Api\EmployeeController::addEmployee
POST	api/register	\App\Controllers\Api\UserController::register
POST	api/login	\App\Controllers\Api\UserController::login
PUT	api/employee/update/([0-9]+)	\App\Controllers\Api\EmployeeController::updateEmployee/\$1



08

Introducció a PHP



Què és PHP?

Llenguatge de programació interpretat. Per tant el tindrem que executar-ho desde localhost.

Obrim http://localhost/mi_web/index.html

Mostrar text per pantalla:

```
echo "Hola soy un texto";
```



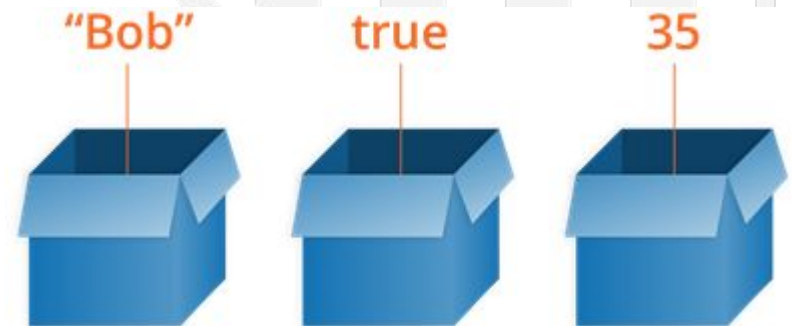
Què és una variable?

Una variable ens permet guardar informació per tal de poder-la utilitzar després.

```
$nom = "PEPE";  
echo "El meu nom és ".$nom;
```

Per concatenar un text i una variable s'utilitzarà el (punt)

La assignació de variables es farà de dreta a esquerra





Activitat

Mostra per pantalla la següent frase:

L'alumne **PEPE** ha tret un **9.8** a l'examen de **Naturals**.

Caldrà generar la frase per amb 3 variables.



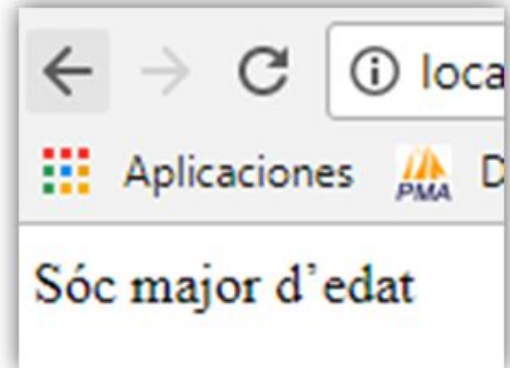
Què és un IF i un ELSE?

Exemple:

```
$edat = 19;
```

```
if( $edat >= 18 ) echo "Soc major d'edat";
```

```
else echo "Soc menor";
```





Què és un array?

```
$response = [  
    "status" => 500,  
    "error" => true,  
    "messages" => 'No employee found',  
    "data" => []  
];  
  
echo "El resultat de la API es ".$response["status"];
```

El resultat serà:

- El resultat de la API es 500



Què és una funció?

```
public function deleteEmployee($emp_id)
{
    $emp = new EmployeeModel();
    $data = $emp->find($emp_id);

    if (!empty($data)) {
        $emp->delete($emp_id);
        $response = [
            'status' => 200,
            "error" => false,
            'messages' => 'Employee deleted successfully',
            'data' => []
        ];
    } else {
        $response = [
            'status' => 500,
            "error" => true,
            'messages' => 'No employee found',
            'data' => []
        ];
    }

    return $this->respond($response);
}
```



Access a les Dades del model

Funcions integrades amb CI4 ([docu](#)):

- `insert($data)`
- `update($id, $data)`
- `save($data)` [inserta, o actualitza]
- `find($id)`
- `findAll()`
- `delete($id)`

```
public function listEmployee()
{
    $emp = new EmployeeModel();

    $response = [
        'status' => 200,
        "error" => false,
        'messages' => 'Employee list',
        'data' => $emp->findAll()
    ];

    return $this->respond($response);
}
```

```
public function showEmployee($emp_id)
{
    $emp = new EmployeeModel();

    $data = $emp->find($emp_id);
    //$data = $model->where(['id' => $emp_id])->first();

    if (!empty($data)) {
        $response = [
            'status' => 200,
            "error" => false,
            'messages' => 'Single employee data',
            'data' => $data
        ];
    } else {
        $response = [
            'status' => 500,
            "error" => true,
            'messages' => 'No employee found',
            'data' => []
        ];
    }

    return $this->respond($response);
}
```



Capturar paràmetres

```
$userModel = new UserModel();

$data = [
    "name" => $this->request->getVar("name"),
    "email" => $this->request->getVar("email"),
    "phone_no" => $this->request->getVar("phone_no"),
    "password" => password_hash($this->request->getVar("password"), PASSWORD_DEFAULT);
];

$userModel->insert($data);
```



09

Validacions amb Codeigniter4



Validacions al Controller

```
public function addEmployee()
{
    $rules = [
        "name" => "required",
        "email" => "required|valid_email|is_unique[employees.email]|min_length[6]",
        "phone_no" => "required",
    ];

    $messages = [
        "name" => [
            "required" => "Name is required"
        ],
        "email" => [
            "required" => "Email required",
            "valid_email" => "Email address is not in format",
            "is_unique" => "Email address already exists"
        ],
        "phone_no" => [
            "required" => "Phone Number is required"
        ],
    ];

    if (!$this->validate($rules, $messages)) {
```



Validacions al Controller

Llista de validacions Natives:

- <https://codeigniter4.github.io/userguide/libraries/validation.html#available-rules>

Com per exemple:

- required
- valid_email
- is_unique[users.email]
- min_length[6]

```
$rules = [  
    "email" => "required|valid_email|min_length[6]",  
    "password" => "required",  
];  
  
$messages = [  
    "email" => [  
        "required" => "Email required",  
        "valid_email" => "Email address is not in format"  
    ],  
    "password" => [  
        "required" => "password is required"  
    ],  
];  
  
if (!$this->validate($rules, $messages)) {
```




```
use \App\Validation\CustomRules;
```

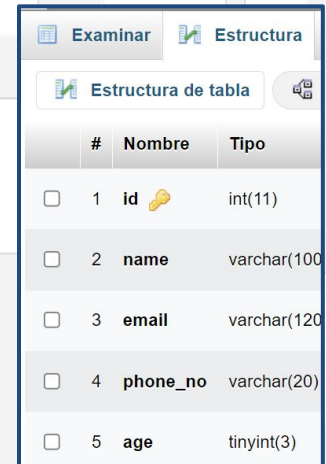
Validacions personalitzades

Carreguem el nostre fitxer de validacions personalitzades

```
$rules = [
    "age" => "required|ageValidation"
];

$messages = [
    "age" => [
        "required" => "Age is required",
        "ageValidation" => "You must be over 18 years old"
    ],
];

if (!$this->validate($rules, $messages)) {
    //Error
} else {
    //Todo OK
}
```



#	Nombre	Tipo
1	id	int(11)
2	name	varchar(100)
3	email	varchar(120)
4	phone_no	varchar(20)
5	age	tinyint(3)

Validation

CustomRules.php

```
public function ageValidation($age){
    if($age>=18) return true;
    else return false;
}
```




10

Api de Usuari (amb JWT)



Pases per crear API User

composer require firebase/php-jwt

php spark make:controller Api/User --suffix --restful

php spark make:model User --suffix

```
namespace App\Controllers\Api;

use CodeIgniter\RESTful\ResourceController;
use App\Models\UserModel;
use Firebase\JWT\JWT;
use Firebase\JWT\Key;
use Exception;
use \App\Validation\CustomRules;

class UserController extends ResourceController
{
```



Codificació del Password

```
$userModel = new UserModel();

$data = [
    "name" => $this->request->getVar("name"),
    "email" => $this->request->getVar("email"),
    "phone_no" => $this->request->getVar("phone_no"),
    "password" => password_hash($this->request->getVar("password"), PASSWORD_DEFAULT)
];
$userModel->insert($data);
```

La funció **password_hash**, crea un nou hash de contrasenya utilitzant un algoritme robust de Hash de sentit únic. Requereix dos paràmetres:

1. La contrasenya sense encriptar
2. Constant que està dissenyada per canviar sempre que aparegui un nou algoritme nou i més fort a PHP.

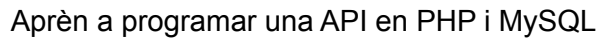


Verificació del Password

```
$userModel = new UserModel();  
$email = $this->request->getVar("email");  
$userdata = $userModel->where(["email" => $email])->first();  
if (password_verify($this->request->getVar("password"), $userdata['password'])){  
    //Login OK  
}else{  
    //Login Incorrecto  
}
```

La funció **password_verify**, comprova que la contrasenya coincideix amb el Hash. Requereix dos paràmetres:

1. La contrasenya de l'usuari que ens arriba desde la api de login
2. El Hash guardat a la taula users, que va ser generat per `password_hash()`.



Un token JWT està codificat en Base64. Té 3 parts:

- ```
CI_ENVIRONMENT = production
CI_ENVIRONMENT = development
JWT_SECRET = '56fdGYTR_e52ew1'
```

eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpYXQiOiJlbnRlc2MTY2MjAsIm5iZil6MTY1NzYxNjYzMjw  
iZlXhwIjoxNjU3NjIwMjIwLCJkYXRhIjpw7ImlkljoiMyIsImVtYWlsIjoibWFydGFAZ21haWwuY29tliwicm9sZSI  
6NH19.XZ\_loqe2cuF9\_Gq-eBeOYyv8Mcj9dfsCnXTZTriLL8g



# Cicle del JWT



+INFO: <https://openwebinars.net/blog/que-es-json-web-token-y-como-funciona/>



# Crear un Token JWT

```
$key = getenv('JWT_SECRET');

$iat = time(); // current timestamp value
$nbfs = $iat + 10;
$exp = $iat + 3600;

$payload = array(
 "iat" => $iat, // issued at
 "nbf" => $nbfs, //not before in seconds
 "exp" => $exp, // expire time in seconds
 "data" => array(
 'id' => $userdata['id'],
 'email' => $userdata['email'],
 'role' => 4,
),
);

$token = JWT::encode($payload, $key, 'HS256');
```

**iat:** Identifica la marca temporal en què el JWT va ser creat.

**nbf:** Identifica la marca temporal en què el JWT comença a ser vàlid.

**exp:** Identifica la marca temporal després de la qual el JWT no ha de ser validat. (1hora)

**data:** informació general que volem guardar.





# Decodificar un Token JWT

```
$token = $this->request->getServer("HTTP_AUTHORIZATION");
$token = str_replace('Bearer ', '', $token);
$decoded = JWT::decode($token, new Key($key, 'HS256'));
if ($decoded) {
 $response = [
 'status' => 200,
 'error' => false,
 'messages' => 'User details',
 'data' => [
 'profile' => $decoded->data
]
];
}
```

Quan ens arriba un token, conté la Paraula Bearer i això no ens interessa.





# API Register

http://localhost:8080/api/register?email=marta@gmail.com

Save

POST http://localhost:8080/api/register?email=marta@gmail.com Send

Params Auth Headers (7) Body Pre-req. Tests Settings Cookies

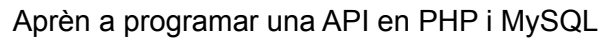
|                                     | KEY   | VALUE           | DESCRIPTION | ... | Bulk Edit |
|-------------------------------------|-------|-----------------|-------------|-----|-----------|
| <input checked="" type="checkbox"/> | email | marta@gmail.com |             |     |           |
|                                     | Key   | Value           | Description |     |           |

Body

200 OK 110 ms 839 B Save Response

Pretty Raw Preview Visualize JSON

```
1
2 "status": 500,
3 "error": true,
4 "message": {
5 "name": "Name is required",
6 "email": "The email field must contain a unique value.",
7 "phone_no": "Phone Number is required",
8 "password": "password is required"
9 },
10 "data": []
11
```



 **Barcelona**  
**Activa**

[barcelona.cat/barcelonactiva](http://barcelona.cat/barcelonactiva)



# API Profile

The screenshot shows a REST client interface with the following components:

- Method and URL:** GET `http://localhost:8080/api/profile`
- Buttons:** Send, Params, Auth (selected), Headers (7), Body, Pre-req., Tests, Settings, Cookies.
- Auth Section:** Type: Bearer Token. Token: `eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1Ni...`
- Response Section:** Body (selected). Status: 200 OK, 84 ms, 777 B. Save Response button.
- Response Body:** Pretty view of JSON data.

```
1 {
2 "status": 200,
3 "error": false,
4 "messages": "User details",
5 "data": {
6 "profile": {
7 "id": "3",
8 "email": "marta@gmail.com",
9 "role": 4
10 }
11 }
12 }
```



# 11

## Anàlisi d'un Login (amb JWT)



# Activitat Final Login

Material:

<https://drive.google.com/drive/folders/1ykPZIEeP4ox4Pq5pBIKBoVigsKoQURWD?usp=sharing>

## Formulario de Login

Email

marta@gmail.com

Contraseña

.....

Login

The screenshot shows the Network tab of a web browser's developer tools. The 'login' request is selected, and the response preview is displayed. The response is a JSON object with the following structure:

```
{status: 200, error: false, messages: "User logged In successfully", data: {token: "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpYXQiOiJlMjNTc2Mj", error: false, messages: "User logged In successfully", status: 200}}
```



# Activitat 1: Canvi de contrasenya amb el token

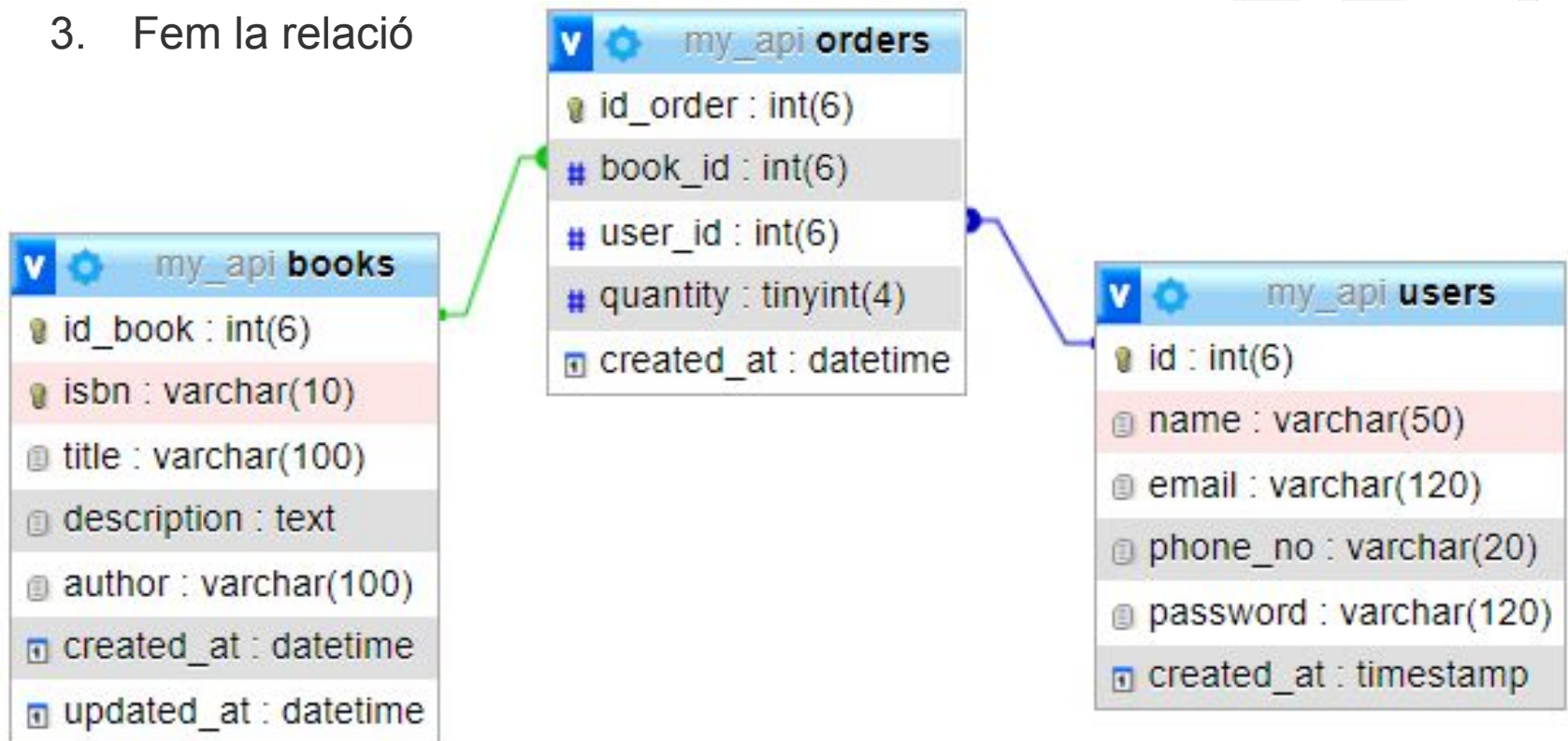
Dades d'entrada:

- Email
- Nova Contrasenya
- El token JWT que identifica l'usuari



## Activitat 2: Joins amb taules

1. Creació de la taula **orders**
2. Afegim les claus **Index** (book\_id i user\_id)
3. Fem la relació





## Activitat 2: Joins amb taules

1. Crea un Endpoint de **Order** amb el mètode **listOrders**  
**`http://localhost:8080/api/order/list`**
2. Crea un Endpoint de **Order** amb el mètode **listOrdersByUser**  
**`http://localhost:8080/api/order/list/user/2`**
3. Crea un Endpoint de **Order** amb el mètode **showOrder**  
**`http://localhost:8080/api/order/show/2`**

### En realitat executem això:

```
SELECT *
FROM orders
LEFT JOIN books ON books.id_book = orders.book_id
LEFT JOIN users ON users.id = orders.user_id
WHERE orders.user_id = XXXX
```





## Activitat 3: Plugin dataTables

- Creació d'una taula dinàmica (amb <https://datatables.net>) que llegeixi dades de la nostre taula de orders.

Material Base:

<https://drive.google.com/file/d/1sOnXRbUqaqiWMB87zdIUpwYMS2OpR86F/view?usp=sharing>



[https://drive.google.com/drive/folders/1f\\_As8a17pLlvjouZnDr2fmTq-PCh5VNy?usp=sharing](https://drive.google.com/drive/folders/1f_As8a17pLlvjouZnDr2fmTq-PCh5VNy?usp=sharing)