API REST

Altas, modificaciones y bajas

Lista de tareas (nuestros pendientes)

Completamos los servicios que faltan

- **Borrar Tarea**
- **Agregar Tarea**
- **Actualizar Tarea**

Agregamos nuevos endpoints

Lista de Tareas - ENDPOINTS

¿Qué servicios nuevos vamos a tener?

Borrar Tarea (recurso **tareas**)

`(DELETE) /api/tareas/:ID`

Agregar Tarea (recurso **tareas**)

`(POST) /api/tareas`

Actualizar Tarea (recurso **tareas**)

`(UPDATE) /api/tareas/:ID`

Borrado de un recurso (DELETE)

TareasApiController - Borrado de Tarea

Borrar Tarea (recurso **tareas**)

`(DELETE) /api/tareas/:ID`

```php
// TareasApiController.php
public function deleteTask($params = []) {
        $task_id = $params[':ID'];
        $task = $this->model->getTask($task_id);

        if ($task) {
            $this->model->deleteTask($task_id);
            $this->view->response("Tarea id=$task_id eliminada con éxito", 200);
        }
        else
            $this->view->response("Task id=$task_id not found", 404);
}
```

Resultado

https://gitlab.com/unicen/Web2/livecoding2019/bolivar/todo-list/commit/71ab6015ed99da7015b31065539d8a515681f4ca

Creación de un recurso (POST)

¿Cómo enviamos los datos?

Agregar Tarea (recurso **tareas**)

`(POST) /api/tareas`

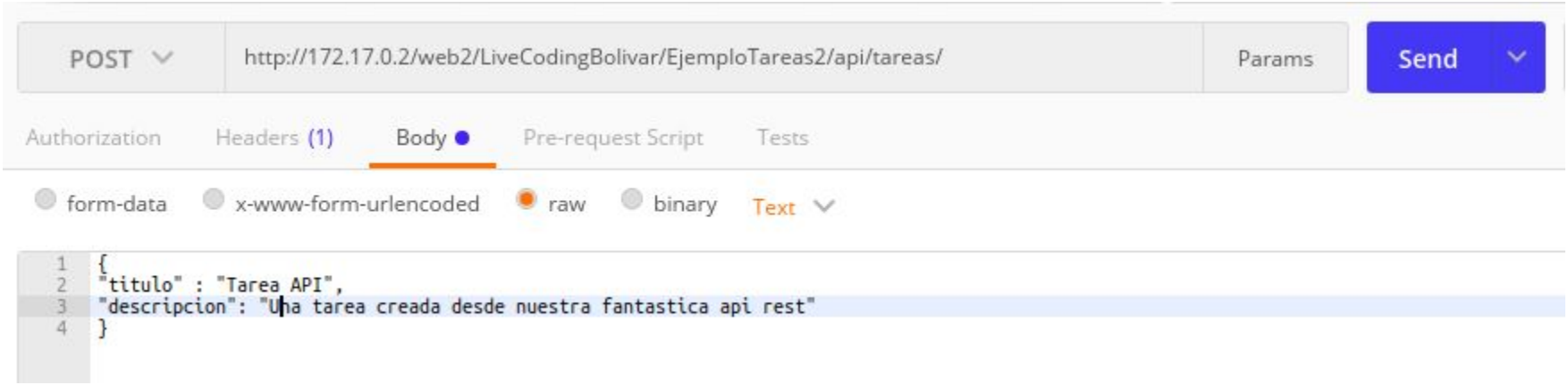
Para trabajar con APIs REST, esperamos los datos en formato JSON

Para la estructura del dato, **usamos la misma de salida**:

```
{
   "titulo": "Tarea API Rest",
    "descripcion": "Una tarea creada desde la API",
    "prioridad": 5
 }
```

Enviando desde POSTMAN

Hacemos un POST y escribimos nuestro objeto JSON en el **body** de la solicitud.



¿Cómo recibimos los datos?

1. Leer el cuerpo del mensaje

```php
file_get_contents("php://input");
```

Permite leer la entrada enviada en formato RAW

Similar a $_POST, excepto que:

- No es un arreglo, es un string de los datos crudos
- No importa que verbo se uso (POST, GET, PUT, ...)

2. Convertir el string recibido a JSON

```php
json_decode("string de un json");
```

Devuelve un objeto JSON

APIController - Parsear la entrada

APIController

Definimos una **clase abstracta**, común para todos los servicios para encapsular los métodos comunes:

```
abstract class ApiController {
    protected $model; // lo instancia el hijo
    protected $view;

    private $data;

    public function __construct() {
        $this->view = new APIView();
        $this->data = file_get_contents("php://input");
    }

    function getData(){
        return json_decode($this->data);
    }
}
```

Lo agregamos al constructor de la clase base. De esta manera esta disponible siempre que queremos usarlo

Usamos **json_decode** sobre la entrada

TareasAPIController - Clase Concreta

TareasAPIController

Definimos la **clase concreta** que implementa el controlador para tareas:

```
class TaskApiController extends ApiController {

   public function __construct() {
       parent::__construct();
       $this->model = new TaskModel();
    }


      //TBC
}
```

Usando $data

Podemos acceder a los campos

```
$body->título
$body->descripcion
```

```
// TaskApiController.php
    public function addTask($params = []) {
        // devuelve el objeto JSON enviado por POST
        $body = $this->getData();

        // inserta la tarea
        $titulo = $body->titulo;
        $descripcion = $body->descripcion;
        $tarea = $this->model->saveTask($titulo, $descripcion);
    }
```

Resultado

https://gitlab.com/unicen/Web2/livecoding2019/bolivar/todo-list/commit/648427ecdaed4ddf65fd67c53107fb6f2976a237

Modificación del recurso (PUT)

Lógica

- Similar al **POST**
- En lugar de **crear** una tarea, vamos a **modificar** una que ya existe.
- Vamos a necesitar al :**ID** de la tarea a modificar.

¿Cual es la URL que vamos a crear?

Combina **parametros** y el acceso al **body** del request

Usando $data y $params

```php
// TaskApiController.php
public function updateTask($params = []) {
        $task_id = $params[':ID'];
        $task = $this->model->getTask($task_id);

        if ($task) {
            $body = $this->getData();
            $titulo = $body->titulo;
            $descripcion = $body->descripcion;
            $finalizada = $body->finalizada;
            $tarea = $this->model->updateTask($task_id, $titulo, $descripcion, $finalizada);
            $this->view->response("Tarea id=$task_id actualizada con éxito", 200);
        }
        else
            $this->view->response("Task id=$task_id not found", 404);
    }
?>
```

Resultado

https://gitlab.com/unicen/Web2/livecoding2019/bolivar/todo-list/commit/959333fe6fc4844a944fafbe550bba7fdaa84ff8

Otras acciones del dominio

Parametros GET

Parámetros GET

/api/tareas?sort=prioridad&order=asc

Por parámetro GET recibe el valor de “*sort*” y “order”

- Devuelve el arreglo de tareas ordenado por prioridad ascendente

/api/tareas/?pending=true

- Por parámetro GET recibe el valor de “*pending*”
- Devuelve el arreglo de tareas que **NO** están finalizadas

Subrecurso

/api/tarea/1/descripcion

- Devuelve solo la descripción de la tarea