



4 pasos para crear una API de REST simple en MySQL PHP

Compartir

INTRODUCCIÓN DESCANSO FÁCIL

Bienvenido a un tutorial paso a paso sobre cómo crear una API REST simple en PHP y MYSQL . Las personas acceden a Internet hoy con una variedad de dispositivos diferentes: computadoras de escritorio, computadoras portátiles, tabletas y teléfonos inteligentes; La tecnología web moderna ya no se trata solo de sitios web, también tiene aplicaciones de Android, aplicaciones de iOS e incluso la necesidad de comunicaciones cruzadas entre servidores.

Probablemente esta sea la razón por la que está buscando formas de crear una API, crear servicios web para aplicaciones móviles y / o abrir un canal de comunicación entre servidores. Esto es exactamente lo que le guiará en esta guía: los conceptos, un ejemplo sencillo de cómo crear una API de PHP para la administración de usuarios y algunos datos adicionales. ¡Sigue leyendo para descubrirlo!

He incluido un archivo zip con todo el código fuente de ejemplo al comienzo de este tutorial, para que no tenga que copiar y pegar todo ... O si solo desea bucear directamente.

NAVEGACIÓN ÍNDICE

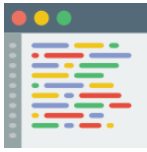


Utilizamos cookies para asegurarnos de brindarle la mejor experiencia en nuestro sitio web. Si continúa utilizando este sitio, asumiremos que usted lo autoriza.

Ok

[política de privacidad](#)

Descarga de código fuente
adicional



Paso 2
La Biblioteca



Extras
Seguridad y Tidbits

Paso 0
Los conceptos básicos



Paso 3
El punto final



Cierre
¿Qué sigue?

Paso 1
La Base de Datos



Paso 4
Implementar y probar

monday.com

Herramienta **Simple** de Gestión de Proyectos

Comenzar Prueba



Primero, aquí está el enlace de descarga al código fuente de ejemplo como se prometió.

CÓDIGO FUENTE DESCARGAR 




Utilizamos cookies para asegurarnos de brindarle la mejor experiencia en nuestro sitio web. Si continúa utilizando este sitio, asumiremos que usted lo autoriza.

Ok

[política de privacidad](#)

Las carpetas

Aquí hay una introducción rápida de cómo se organizan las carpetas en el archivo zip.

-  API Donde ponemos todos los scripts de punto final de API.
-  lib Donde almacenamos todos los archivos de la librería PHP.
-  SQL Todos los archivos SQL necesarios para construir la base de datos. Se puede eliminar de forma segura después de que haya importado todos ellos.

INICIO RÁPIDO

- Descargar y descomprimir en una carpeta.
- Crea una base de datos e importa todos los archivos en la sqlcarpeta.
- Cambia la configuración de la base de datos lib/2a-config.php a la tuya.
- Inicia 4-test.html en tu navegador.
- ¿Falta el punto final de la API? Trate de usar la URL absoluta en 4-test.html- `<form action="HTTP://SITE.COM/api/3-users.php">`

PASO 0

LOS CONCEPTOS BÁSICOS



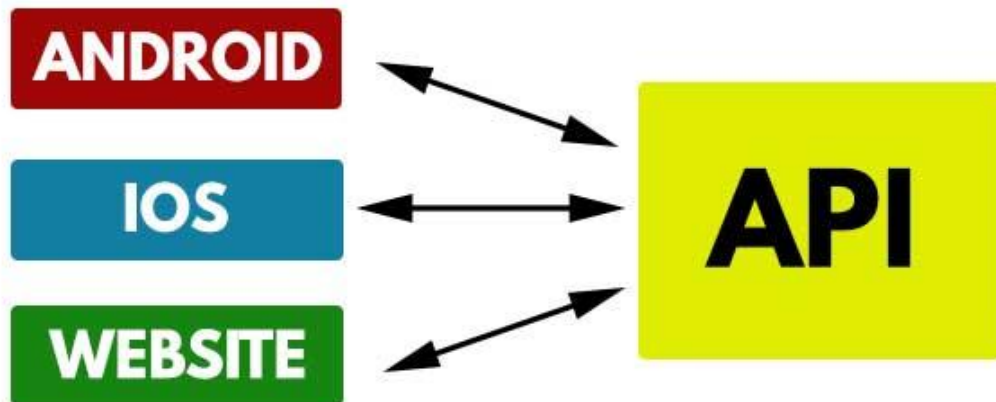
Antes de profundizar en el ejemplo, esta sección explicará los conceptos básicos: Qué es una API, REST y CRUD. Por favor, siéntase libre de saltarse esta sección si ya sabe cuáles son.

¿QUÉ ES UNA API?

Utilizamos cookies para asegurarnos de brindarle la mejor experiencia en nuestro sitio web. Si continúa utilizando este sitio, asumiremos que usted lo autoriza.

Ok

[política de privacidad](#)



API significa una plicación P rogramación I nterface, y no es sin duda el nombre de marca de una cerveza. Buena vieja [Wikipedia](#) lo define como:

Un conjunto de métodos de comunicación claramente definidos entre varios componentes.

Es algo críptico, pero eso explica bastante bien lo que es una API. En la Internet moderna, un sistema basado en la web probablemente tendrá que comunicarse con el sitio web, las aplicaciones de Android, las aplicaciones de iOS y tal vez incluso otros sistemas de terceros. La interfaz que maneja toda la comunicación es la API.

Por ejemplo, ejecuta una tienda en línea y desea desarrollar una aplicación de Android / iOS para ella. Será un poco tonto (y duro) desarrollar 3 conjuntos diferentes de código para tratar con cada plataforma, por lo que tendremos una única API "estándar" para procesar todo el registro de usuarios, el catálogo de productos, el proceso de pago y la gestión de pedidos. .

PS A veces llamamos API "servicio web" o "punto final". No hay que confundirse, literalmente significan lo mismo.

¿QUÉ PASA CON REST?

REST es la abreviatura de R e presentational S tate T ransfer. De buena vieja [Wikipedia](#) de nuevo:

Utilizamos cookies para asegurarnos de brindarle la mejor experiencia en nuestro sitio web. Si continúa utilizando este sitio, asumiremos que usted lo autoriza.

Ok

[política de privacidad](#)

arquitectónico de REST, o los servicios web de REST, proporcionan interoperabilidad entre los sistemas informáticos en Internet. Los servicios web compatibles con REST permiten que los sistemas solicitantes accedan y manipulen representaciones textuales de recursos web mediante el uso de un conjunto uniforme y predefinido de operaciones sin estado.

Eso parece un puñado de cosas técnicas confusas, pero vamos a separarlas poco a poco. REST es básicamente la idea detrás de lo que ya está en Internet: los sitios web. En cualquier arquitectura REST, debe haber:

- Una relación cliente / servidor.
- Protocolo sin estado (por ejemplo, HTTP).
- Protocolo en capas (por ejemplo, TCP / IP).

API RESTFUL



Entonces, cuando combinamos los poderes de REST y API juntos, obtenemos una API compatible con REST (API REST para abreviar):

- La API debe " **proporcionar interoperabilidad entre sistemas informáticos en Internet** ". Lo que significa que la API debe ser capaz de lidiar con los procesos en todo el sitio web, la aplicación de Android, la aplicación de iOS y cualquier otro sistema.
- " **Permitir que los sistemas solicitantes accedan y manipulen representaciones textuales de recursos web** ". Cuando el cliente realiza una solicitud al servidor, el servidor debe devolver una respuesta de texto. Doh
- " **Conjunto predefinido de operaciones sin estado** ". La API solo tiene un conjunto fijo de funciones, y no es como una línea de comandos donde puede hacer lo que quiera. Operaciones sin estado, lo que significa que, una vez que el servidor responde, la conexión se cierra.

CRUD

Hablando de un conjunto fijo de funciones predefinido, también debe saber CRUD:

Utilizamos cookies para asegurarnos de brindarle la mejor experiencia en nuestro sitio web. Si continúa utilizando este sitio, asumiremos que usted lo autoriza.

Ok

[política de privacidad](#)

- Actualizar
- Borrar

Sí, el concepto de CRUD es simple, pero se mantiene fiel a casi cualquier proceso y API que construya. Por ejemplo:

- Creando un nuevo usuario.
- Leyendo los datos del usuario.
- Actualización del usuario.
- Eliminando el usuario.

Entonces, para cualquier sistema en el que esté trabajando, recuerde que CRUD son los primeros 4 procesos básicos que su sistema debe cubrir.



PASO 1

LA BASE DE DATOS



Comencemos por sentar las bases primero, he creado una tabla de usuario ficticia con el propósito de demostración en este tutorial. Sé que es posible que ya tenga un proyecto existente y que quiera trabajar en él. Pero recomendaré experimentar primero con esta tabla ficticia, comprender los mecanismos detrás antes de implementar realmente las cosas en su propio proyecto.

TABLA DE USUARIO DUMMY

sql / 1-users.sql

```
CREATE TABLE `users` (  
  `id` int(11) NOT NULL,  
  `name` varchar(255) NOT NULL,
```

Utilizamos cookies para asegurarnos de brindarle la mejor experiencia en nuestro sitio web. Si continúa utilizando este sitio, asumiremos que usted lo autoriza.

Ok

[política de privacidad](#)

```
ADD PRIMARY KEY (`id`),
ADD UNIQUE KEY `email` (`email`),
ADD KEY `name` (`name`);
```

Campo	Descripción
carné de identidad	Clave primaria, el ID de usuario.
nombre	El nombre de eull del usuario.
correo electrónico	Dirección de correo electrónico del usuario, configurada como única para evitar registros dobles.
contraseña	La contraseña del usuario.

DATOS FICTICIOS

Ahora, para algunos datos ficticios, la contraseña de todos los usuarios aquí es "test123", cifrada con PHP openssl_encrypt y la clave secreta "CodeB0xx".

sql / 1-users.sql

```
INSERT INTO `users` (`id`, `name`, `email`, `password`) VALUES
(1, 'John Doe', 'john@doe.com', 'Xc1HAPhnvi07v6mTfwxsHg=='),
(2, 'Jane Doe', 'jane@doe.com', 'Xc1HAPhnvi07v6mTfwxsHg=='),
(3, 'Apple Doe', 'apple@doe.com', 'Xc1HAPhnvi07v6mTfwxsHg=='),
(4, 'Beck Doe', 'beck@doe.com', 'Xc1HAPhnvi07v6mTfwxsHg=='),
(5, 'Charlie Doe', 'charlie@doe.com', 'Xc1HAPhnvi07v6mTfwxsHg=='),
(6, 'Charles Doe', 'charles@doe.com', 'Xc1HAPhnvi07v6mTfwxsHg=='),
(7, 'Dion Doe', 'dion@doe.com', 'Xc1HAPhnvi07v6mTfwxsHg=='),
(8, 'Dee Doe', 'dee@doe.com', 'Xc1HAPhnvi07v6mTfwxsHg=='),
(9, 'Emily Doe', 'emily@doe.com', 'Xc1HAPhnvi07v6mTfwxsHg=='),
(10, 'Ethan Doe', 'ethan@doe.com', 'Xc1HAPhnvi07v6mTfwxsHg=='),
(11, 'Frank Doe', 'frank@doe.com', 'Xc1HAPhnvi07v6mTfwxsHg=='),
(12, 'Gina Doe', 'gina@doe.com', 'Xc1HAPhnvi07v6mTfwxsHg=='),
(13, 'Hela Doe', 'hela@doe.com', 'Xc1HAPhnvi07v6mTfwxsHg=='),
(14, 'Hubert Doe', 'hubert@doe.com', 'Xc1HAPhnvi07v6mTfwxsHg=='),
(15, 'Ivy Doe', 'ivy@doe.com', 'Xc1HAPhnvi07v6mTfwxsHg=='),
(16, 'Ingrid Doe', 'ingrid@doe.com', 'Xc1HAPhnvi07v6mTfwxsHg=='),
(17, 'James Doe', 'james@doe.com', 'Xc1HAPhnvi07v6mTfwxsHg=='),
(18, 'Jace Doe', 'jace@doe.com', 'Xc1HAPhnvi07v6mTfwxsHg=='),
(19, 'Kate Doe', 'kate@doe.com', 'Xc1HAPhnvi07v6mTfwxsHg=='),
(20, 'Luke Doe', 'luke@doe.com', 'Xc1HAPhnvi07v6mTfwxsHg==');
```

PASO 2

LA BIBLIOTECA



Ahora que tenemos la base de datos preparada, es hora de construir la clase / biblioteca de usuario. Sí, podemos "codificar directamente" todo en un solo script, pero recordamos las buenas prácticas de programación: facilidad de uso y fácil mantenimiento.

Por ejemplo, la base de datos de usuarios se comparte en un sitio web de comercio electrónico. Hay un escaparate, un administrador y una API para soportar las aplicaciones móviles. Si escribimos "un script para cada uno", terminaremos teniendo 3 conjuntos de scripts separados haciendo lo mismo. Pero una vez que ponemos todo en una biblioteca, puede compartirlo en las diferentes plataformas, solo tenemos una biblioteca para mantener.

El archivo de configuración

lib / 2a-config.php

```
<?php
// MUTE NOTICES
error_reporting(E_ALL & ~E_NOTICE);

// DATABASE SETTINGS
define("DB_HOST", "localhost");
define("DB_DB", "test");
define("DB_CHAR", "utf8");
define("DB_USER", "root");
define("DB_PASS", "");

// SECRET KEY FOR PASSWORDS
define("SECRET_KEY", "CodeB0xx");

// FILE PATH
// Manually define the absolute path if you get path problems
define('PATH_LIB', __DIR__ . DIRECTORY_SEPARATOR);
?>
```

Utilizamos cookies para asegurarnos de brindarle la mejor experiencia en nuestro sitio web. Si continúa utilizando este sitio, asumiremos que usted lo autoriza.

Ok

política de privacidad

LA BIBLIOTECA DEL USUARIO

lib / 2b-lib-user.php

```
<?php
class User{
    /* [DATABASE HELPER FUNCTIONS] */
    private $pdo = null;
    private $stmt = null;
    public $error = "";

    function __construct(){
        try {
            $this->pdo = new PDO(
                "mysql:host=".DB_HOST.";dbname=".DB_DB.";charset=".DB_CHAR,
                DB_USER, DB_PASS, [
                    PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,
                    PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
                    PDO::ATTR_EMULATE_PREPARES => false,
                ]
            );
        } catch (Exception $ex) { die($ex->getMessage()); }
    }

    function __destruct(){
        if ($this->stmt!==null) { $this->stmt = null; }
        if ($this->pdo!==null) { $this->pdo = null; }
    }

    function query($sql, $cond=[]){
        try {
            $this->stmt = $this->pdo->prepare($sql);
            $this->stmt->execute($cond);
        } catch (Exception $ex) {
            $this->error = $ex->getMessage();
            return false;
        }
        $this->stmt = null;
        return true;
    }

    /* [USER FUNCTIONS] */
    function getAll(){
        $this->stmt = $this->pdo->prepare("SELECT * FROM `users`");
        $this->stmt->execute();
        $users = $this->stmt->fetchAll();
        return count($users)==0 ? false : $users;
    }

    function getEmail($email){
        $this->stmt = $this->pdo->prepare("SELECT * FROM `users` WHERE `email`=?");
        $cond = [$email];
        $this->stmt->execute($cond);
        $user = $this->stmt->fetchAll();
        return count($user)==0 ? false : $user[0];
    }
}
```

Utilizamos cookies para asegurarnos de brindarle la mejor experiencia en nuestro sitio web. Si continúa utilizando este sitio, asumiremos que usted lo autoriza.

Ok

[política de privacidad](#)

```

    $this->stmt->execute($cond);
    $user = $this->stmt->fetchAll();
    return count($user)==0 ? false : $user[0];
}

function create($name, $email, $password){
    return $this->query(
        "INSERT INTO `users` (`name`, `email`, `password`) VALUES (?, ?, ?)",
        [$name, $email, openssl_encrypt($password, "AES-128-ECB", SECRET_KEY)]
    );
}

function update($name, $email, $password="", $id){
    $q = "UPDATE `users` SET `name`=?, `email`=?";
    $cond = [$name, $email];
    if ($password!="") {
        $q .= ", `password`=?";
        $cond[] = openssl_encrypt($password, "AES-128-ECB", SECRET_KEY);
    }
    $q .= " WHERE `id`=?";
    $cond[] = $id;
    return $this->query($q, $cond);
}

function delete($id){
    return $this->query(
        "DELETE FROM `users` WHERE `id`=?",
        [$id]
    );
}

/* [LOGIN] */
function login($email, $password){
    $user = $this->getEmail($email);
    if ($user==false) { return false; }
    return openssl_decrypt($user['password'], "AES-128-ECB", SECRET_KEY) == $password ? $user : false;
}
}
?>

```

¿Recuerdas a CRUD? Cuando construimos una biblioteca, debería tener funciones para cubrir todos esos.

Funciones de ayuda de base de datos

Función	Descripción
__constructor	El constructor. Se conectará automáticamente a la base de datos cuando se cree el objeto de usuario.
__destruct	El destructor. Cerrará automáticamente la conexión de la base de datos cuando el objeto de usuario sea destruido.
consulta	Ejecutar una consulta SQL dada en la base de datos.

Utilizamos cookies para asegurarnos de brindarle la mejor experiencia en nuestro sitio web. Si continúa utilizando este sitio, asumiremos que usted lo autoriza.

Ok

política de privacidad

Función	Descripción
obtener toda	Consigue todos los usuarios.
getEmail	Obtener usuario por la dirección de correo electrónico dada.
getID	Obtener usuario por el ID de usuario dado.

Establecer y eliminar usuarios (CREAR, ACTUALIZAR, BORRAR)

Función	Descripción
crear	Agregar un nuevo usuario a la base de datos.
actualizar	Actualizar datos del usuario especificado.
borrar	Eliminar la ID de usuario dada.

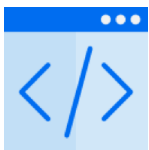
Iniciar sesión

Función	Descripción
iniciar sesión	Comprueba si el correo electrónico y la contraseña dados son correctos, devuelve falso si no, devuelve los datos del usuario en caso afirmativo. Tome nota: esto solo comprueba y no inicia la sesión de cookies.



PASO 3

LA PUNTUACIÓN



Ahora hemos establecido todas las bases necesarias, y el paso final es construir la API real o el propio punto final. Esto debería ser una brisa siempre y cuando tenga las funciones de la biblioteca correctamente en primer lugar.

Utilizamos cookies para asegurarnos de brindarle la mejor experiencia en nuestro sitio web. Si continúa utilizando este sitio, asumiremos que usted lo autoriza.

Ok

[política de privacidad](#)

api / 3-users.php

```
<?php
if (isset($_POST['req'])) {
    // INIT
    require dirname(__DIR__) . DIRECTORY_SEPARATOR . "lib" . DIRECTORY_SEPARATOR . "2a-config";
    require PATH_LIB . "2b-lib-user.php";
    $users = new User();

    // PROCESS REQUEST
    switch ($_POST['req']) {
        default:
            echo json_encode([
                "status" => false,
                "message" => "Invalid Request"
            ]);
            break;

        case "get-all":
            $all = $users->getAll();
            echo json_encode([
                "status" => $all==false?false:true,
                "data" => $all
            ]);
            break;

        case "get-email":
            $usr = $users->getEmail($_POST['email']);
            echo json_encode([
                "status" => $usr==false?false:true,
                "data" => $usr
            ]);
            break;

        case "get-id":
            $usr = $users->getID($_POST['id']);
            echo json_encode([
                "status" => $usr==false?false:true,
                "data" => $usr
            ]);
            break;

        case "create":
            $pass = $users->create($_POST['name'], $_POST['email'], $_POST['password']);
            echo json_encode([
                "status" => $pass,
                "message" => $pass ? "User Created" : "Error creating user"
            ]);
            break;

        case "update":
            $pass = $users->update($_POST['name'], $_POST['email'], $_POST['password'], $_POST['id']);
            echo json_encode([
                "status" => $pass,
                "message" => $pass ? "User Updated" : "Error updating user"
            ]);
            break;

        case "delete":
            $pass = $users->delete($_POST['id']);
            echo json_encode([
                "status" => $pass,
                "message" => $pass ? "User Deleted" : "Error deleting user"
            ]);
            break;
    }
}
```

Utilizamos cookies para asegurarnos de brindarle la mejor experiencia en nuestro sitio web. Si continúa utilizando este sitio, asumiremos que usted lo autoriza.

Ok

[política de privacidad](#)

```

        break;

    case "login":
        if (is_array($_SESSION['user'])) {
            die(json_encode([
                "status" => true,
                "message" => "Already signed in"
            ]));
        }
        $pass = $users->login($_POST['name'], $_POST['password']);
        if ($pass!==false) { $_SESSION['user'] = $pass; }
        echo json_encode([
            "status" => is_array($pass),
            "message" => is_array($pass) ? "OK" : "Error"
        ]);
        break;

    case "logout":
        unset($_SESSION['user']);
        echo json_encode([
            "status" => true,
            "message" => "OK"
        ]);
        break;
    }
}
?>

```

Creo que el guión se explica por sí mismo por qué dije que el punto final es muy sencillo cuando la biblioteca está bien hecha. 😊 Pero, ¿cómo funciona todo el punto final de la API aquí?

- Las solicitudes de API se publicarán en este `api/3-users.php` archivo.
- Todas las solicitudes contendrán `$_POST['req']` para indicar la función requerida, seguida de los parámetros requeridos. Por ejemplo, si queremos obtener un usuario por correo electrónico, deberíamos publicar `$_POST['req']="get-email"` y `$_POST['email']="john@doe.com"` para este usuario endpoint.
- El sistema dará una respuesta en el formato JSON. Por lo general, me gusta mantener un formato estándar para que la respuesta no confunda a ningún tercero utilizando la API. Por ejemplo:

Clave de respuesta	Descripción
estado	Verdadero o falso, si el proceso es un éxito o un fracaso.
mensaje	Mensaje del sistema, en su caso.
datos	Contiene los datos solicitados, por ejemplo, esto contendrá una lista de los usuarios

Utilizamos cookies para asegurarnos de brindarle la mejor experiencia en nuestro sitio web. Si continúa utilizando este sitio, asumiremos que usted lo autoriza.

Ok

[política de privacidad](#)



PASO 4

DESPLIEGUE Y PRUEBA



Finalmente, tenemos un punto final totalmente funcional que cubre las funciones básicas de CRUD. Todo lo que queda es publicar algunas variables en la API para probarlo, luego ponerlo en línea.

Probando la API

4-test.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>USER API TEST</title>
  </head>
  <body>
    <h1>Get All Users</h1>
    <form action="api/3-users.php" method="post" target="_blank">
      Request <input type="text" name="req" value="get-all" readonly/><br>
      <input type="submit" value="Get"/>
    </form>

    <h1>Get User By Email</h1>
    <form action="api/3-users.php" method="post" target="_blank">
      Request <input type="text" name="req" value="get-email" readonly/><br>
      Email <input type="email" name="email" required/><br>
      <input type="submit" value="Get"/>
    </form>

    <h1>Get User By ID</h1>
    <form action="api/3-users.php" method="post" target="_blank">
      Request <input type="text" name="req" value="get-id" readonly/><br>
      Email <input type="text" name="id" required/><br>
      <input type="submit" value="Get"/>
    </form>

    <h1>Create New User</h1>
    <form action="api/3-users.php" method="post" target="_blank">
      Request <input type="text" name="req" value="create" readonly/><br>
      Name <input type="text" name="name" required/><br>
      Email <input type="email" name="email" required/><br>
      Password <input type="text" name="password" required/><br>
```

Utilizamos cookies para asegurarnos de brindarle la mejor experiencia en nuestro sitio web. Si continúa utilizando este sitio, asumiremos que usted lo autoriza.

Ok

[política de privacidad](#)

```

<form action="api/3-users.php" method="post" target="_blank">
Request <input type="text" name="req" value="update" readonly/><br>
ID <input type="text" name="id" required/><br>
Name <input type="text" name="name" required/><br>
Email <input type="email" name="email" required/><br>
Password <input type="text" name="password"/><br>
<input type="submit" value="Update"/>
</form>

<h1>Delete</h1>
<form action="api/3-users.php" method="post" target="_blank">
Request <input type="text" name="req" value="delete" readonly/><br>
ID <input type="text" name="id" required/>
<input type="submit" value="Delete"/>
</form>
</body>
</html>

```

Hay muchas maneras de probar la API, pero he creado algunos formularios HTML simples para que la pruebe. Para los principiantes que todavía están confundidos sobre lo que hacen los 3 pasos anteriores, creo que estas simples pruebas le mostrarán exactamente cómo funciona la API completa. Por ejemplo, si desea crear un nuevo usuario, todo lo que tiene que hacer es publicar los datos necesarios en `api/3-users.php`:

req	crear
nombre	uvuwewewe
correo electrónico	uvuwewewe@onyetenyevwe.com
contraseña	prueba12345

Si el proceso es exitoso, debería obtener una respuesta del servidor:

```
{ "status": true, "message": "User Created" }
```

Por supuesto, en el mundo de las aplicaciones web y móviles profesionales, debería haber sido una llamada AJAX y tendrá que JSON decodificar los resultados para mostrarlos correctamente en la interfaz de usuario.

SEGURIDAD EN TENDAS Y TENDRITS

Utilizamos cookies para asegurarnos de brindarle la mejor experiencia en nuestro sitio web. Si continúa utilizando este sitio, asumiremos que usted lo autoriza.



Ahora hemos completado con éxito un punto final de API completo y funcional. Pero tenga en cuenta que este punto final se mantiene extremadamente simplista para fines de demostración, no como lo harán los profesionales. He omitido todos los controles de seguridad y todo lo que debe implementar en su propio proyecto. ¿Cuáles son algunos de estos elementos "de seguridad y demás" que faltan en el ejemplo anterior? Éstos son algunos de ellos.

URL amistoso con HTACCESS REWRITE

Si el nombre del archivo en bruto lo `api/3-users.php` hace temblar, use la `htaccess` reescritura de URL para ayudarlo a crear URLs amigables. Por ejemplo, podemos apuntar `https://site.com/api/users/a` `users/3-users.php`:

```
<IfModule mod_rewrite.c>
RewriteEngine On
RewriteRule ^api/users/?$ /users/3-users.php [L]
</IfModule>
```

Recomiendo encarecidamente hacerlo, porque de esta manera, también puede usarlo para administrar varias versiones de su API en el futuro:

```
<IfModule mod_rewrite.c>
RewriteEngine On
RewriteRule ^api/users/?$ /api/3-users-new.php [L]
RewriteRule ^api/users/1.3/?$ /api/3-users-old.php [L]
RewriteRule ^api/users/1.2/?$ /api/3-users-older.php [L]
</IfModule>
```

ANTI-INYECCIÓN Y CONTROL DE FORMAS

La demostración anterior incluirá cualquier cosa que publiquemos. Probablemente no sea una buena idea en un sistema de producción, y necesita tener algunas verificaciones de entrada en sus proyectos. Por ejemplo, un nombre válido debe tener al menos 2 caracteres, una dirección de correo electrónico válida y verificaciones contra inyecciones de SQL.

Utilizamos cookies para asegurarnos de brindarle la mejor experiencia en nuestro sitio web. Si continúa utilizando este sitio, asumiremos que usted lo autoriza.

Ok

[política de privacidad](#)

Con el ejemplo anterior, cualquier persona puede crear y eliminar usuarios. Esa debería haber sido una función de "solo administrador", y necesita implementar algunas verificaciones de permisos de usuario en su propio proyecto.

HTTPS

¿De qué sirve una contraseña cuando la transmite a través de Internet en texto sin formato? Exigir el uso de HTTPS, y algunas compañías de alojamiento en estos días incluso proporcionan un certificado SSL de forma gratuita.

¿QUÉ PASA CON LA SALIDA DE XML?

Por supuesto, JSON no es la "salida estándar fija" de las API. Lo utilicé solo porque es uno de los estándares más utilizados. Si necesita compatibilidad con XML, siga adelante y ayúdese.

CSRF TOKEN

Al igual que los formularios HTML tradicionales, las API web también son propensas a los ataques CSRF. ¿Qué es CSRF y cómo lo prevenimos? Lee mi otra guía:

3 Steps to Implement Simple CSRF Token in PHP



Share INTRODUCTION SECURE YOUR APPS! Welcome to a step-by-step tutorial on how to implement simple CSRF token in PHP. In this tutorial, we will walk through an example of what cross-site request forgery is, and how we can prevent that with a simple trick – In just 3 steps. Read on to find out! I ... Continue reading



Code Boxx

6

Utilizamos cookies para asegurarnos de brindarle la mejor experiencia en nuestro sitio web. Si continúa utilizando este sitio, asumiremos que usted lo autoriza.

Ok

política de privacidad

- Haga sus bibliotecas - Recuerde que una buena biblioteca tiene que cubrir CRUD.
- Trabaja en la API: esto es fácil una vez que tienes una buena biblioteca.
- Implementar y probar - POST a su propia API.

CERRANDO LO SIGUIENTE?



Gracias por leer, y hemos llegado al final de esta guía. Espero que le haya dado una idea de cómo crear su propia API, y si tiene algo que agregar a la guía, no dude en comentar a continuación. Buena suerte y feliz codificación!

[← Publicación anterior](#)

[Siguiente post →](#)

Deja un comentario

Su dirección de correo electrónico no será publicada. Los campos requeridos están marcados *

Type here..

Name*

Email*

Utilizamos cookies para asegurarnos de brindarle la mejor experiencia en nuestro sitio web. Si continúa utilizando este sitio, asumiremos que usted lo autoriza.

Ok

[política de privacidad](#)