# Restful API WisioLMS

WisioLMS es una plataforma basada en Web que permite a colaboradores de una organización poder consumir contenido multimedia que apoye el proceso de aprendizaje de los estudiantes. Está desarrollado con el Framework Laravel en su versión 5.5.40.

Los requerimientos que se tienen para poder hospedar el sistema y tenerlo en funcionamiento son los siguientes:

* PHP >= 7.0.0
* OpenSSL PHP Extension
* PDO PHP Extension
* Mbstring PHP Extension
* Tokenizer PHP Extension
* XML PHP Extension

Se puede encontrar más información al respecto en la siguiente dirección: https://laravel.com/docs/5.5#server-requirements

Existe un API en wisiolms con funciones para configurar el sistema, insertar contenido, usuarios y ver los reportes del avance en cursos. Para poder utilizarla es necesario ser un usuario con permisos administrativos en la plataforma; esto es, un correo electrónico y su respectiva contraseña.

En este caso la API ocupa REST. REST es cualquier interfaz entre sistemas que use HTTP para obtener datos o generar operaciones sobre esos datos en todos los formatos posibles, como XML y JSON.

Las características más destacadas de REST son las siguientes:

● Protocolo cliente/servidor sin estado: cada petición HTTP contiene toda la información necesaria para ejecutarla, lo que permite que ni cliente ni servidor necesiten recordar ningún estado previo para satisfacerla. Aunque esto es así, algunas aplicaciones HTTP incorporan memoria caché. Se configura lo que se conoce como protocolo cliente-caché-servidor sin estado: existe la posibilidad de definir algunas respuestas a peticiones HTTP concretas como ‘cacheables’, con el objetivo de que el cliente pueda ejecutar en un futuro la misma respuesta para peticiones idénticas. De todas formas, que exista la posibilidad no significa que sea lo más recomendable.

● Las operaciones más importantes relacionadas con los datos en cualquier sistema REST y la especificación HTTP son cuatro: POST (crear), GET (leer y consultar), PUT (editar) y DELETE (eliminar).

● Los objetos en REST siempre se manipulan a partir de la URI. Es la URI y ningún otro elemento el identificador único de cada recurso de ese sistema REST. La URI nos facilita acceder a la información para su modificación o borrado, o, por ejemplo, para compartir su ubicación exacta con terceros.

● Interfaz uniforme: para la transferencia de datos en un sistema REST, este aplica acciones concretas (POST, GET, PUT y DELETE) sobre los recursos, siempre y cuando estén identificados con una URI. Esto facilita la existencia de una interfaz uniforme que sistematiza el proceso con la información.

● Sistema de capas: arquitectura jerárquica entre los componentes. Cada una de estas capas lleva a cabo una funcionalidad dentro del sistema REST.

● Uso de hipermedios: hipermedia es un término acuñado por Ted Nelson en 1965 y que es una extensión del concepto de hipertexto. Ese concepto llevado al desarrollo de páginas web es lo que permite que el usuario puede navegar por el conjunto de objetos a través de enlaces HTML. En el caso de una API REST, el concepto de hipermedia explica la capacidad de una interfaz de desarrollo de aplicaciones de proporcionar al cliente y al usuario los enlaces adecuados para ejecutar acciones concretas sobre los datos.

Al tratarse de una API, se utiliza Laravel Passport, que proporciona una implementación completa del servidor OAuth2. Este token se construye sobre el servidor de League OAuth2 que mantiene Alex Bilbie.

0ath2-server está documentado en el siguiente enlace: <https://github.com/thephpleague/oauth2-server>

Para poder realizar cualquier acción es necesario tener ese token, o de lo contrario no se podrá realizar la petición y se mostrará un mensaje de no autorizado.

Al momento de realizar alguna petición, es necesario enviar en el header el token, esto está especificado en el siguiente enlace: <https://tools.ietf.org/html/rfc6750>

Un ejemplo es:

Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImp0aSI6IjIyMjQxNjRlNjIyOGI5YTJmODhhYTkzZGYxY2NkMjU3MzFjZTQ2NDQ0YTRjYTdmMWIxYWY1NDA3OTI3ZjkwZDZiZjlhZjhlMTI4YWQxZTVlIn0.eyJhdWQiOiI1IiwianRpIjoiMjIyNDE2NGU2MjI4YjlhMmY4OGFhOTNkZjFjY2QyNTczMWNlNDY0NDRhNGNhN2YxYjFhZjU0MDc5MjdmOTBkNmJmOWFmOGUxMjhhZDFlNWUiLCJpYXQiOjE1NDQxNDI3NzQsIm5iZiI6MTU0NDE0Mjc3NCwiZXhwIjoxNTc1Njc4Nzc0LCJzdWIiOiI2ODA4Iiwic2NvcGVzIjpbXX0.liZNk24V1KCHTL\_2Mm8dLzaulXKLjfaPFPNVqOmQFo-RPhx\_FLixIYJ6MYtNd\_9MGYFjheB0gVZM1WCgF1sCCV2eex6Xm\_2ItYu\_IaHU9dR5BMyMjykSHzKb-lpaKfG47J4edolND-G7nu6F341QF-JZsqFGvJmhPxYHPqBiJcVwP8fZpgD62OCWR91QSH-vG7ifsqCcqVPR0mfEbiMnCR5McEbVfczUgoKm523t\_YMtgtU9v9GwpZ07zxHYSz85wVw\_9eSKHiCQEuVkkg0KHPjzt8BYJ3py5THspuJmJAHwxJr7euWKQMFayCmZydD03RDHvknM8HMTqgtGV7YkL8NrT7vgZjkJEhEBV5OWHCfSzM9yDXP0i6fQ\_p0kHUNljmXeLj3gAFRCqmkxlCbmZ9PxPERa6fL6e-388sQ7W\_C0De\_wXnESkeTSV2sB44LwoViSKgU7peTmzp-8JUKk8e9qv9N9hnqqCorMzFdQ1MFo4AkCQHcrvkNwOSjWlEUaMokjztG1z3D83OrklaBlM4t\_SlPn1DBZZfbxHi-g2V9B6sCPCjmmx2pqJuPynVQ1JlBDRvKPYSdSuy4nIarfo73SydHkLBETj\_CoJUxVYpni7ytfnWuxF9AgYicGH0Kk8eVpm1ZWct-LJOrX0Q1PsKxDg-E15RFWwVNMKFgtmUg

Otra cabecera que debe ser especificada es la siguiente:

Accept: application/json

# Endpoints

Esta API se encuentra actualmente en la versión 1, y todas comparten la siguiente estructura:

url/api/v1/nombre-del-recurso

| **Verbo** | **URI** | **Action** |
| --- | --- | --- |
| GET | url/api/v1/nombre-del-recurso | Consulta de recursos |
| POST | url/api/v1/nombre-del-recurso | Inserción de recurso/recursos |
| GET | url/api/v1/nombre-del-recurso/{id o identificador} | Se muestra un recurso |
| PUT/PATCH | url/api/v1/nombre-del-recurso/{id o identificador} | Actualización de un recurso |
| DELETE | url/api/v1/nombre-del-recurso/{id o identificador} | Eliminación de un recurso |

# Parámetros permitidos

Se pueden establecer distintos parámetros para ajustar las consultas, a continuación se muestran los disponibles en cada una de las acciones.

## Consulta de recursos

Select. Se utiliza para elegir los campos que se obtendrán en la consulta, en caso de escoger uno inexistente, simplemente se omitirá. Ejemplos:

/users?select={firstname,lastname,inexistent\_field}

/users?select=firstname,lastname,inexistent\_field

/users?select=[firstname,lastname,inexistent\_field]

Where. Da el resultado de realizar una consulta con where en SQL; las respuestas que se obtienen de los datos se toman en cuenta los campos disponibles y en el caso de que se envíe un parámetro que no está en la lista de los atributos requeridos, este será ignorado.

La estructura de cada condición deberá ser la siguiente:  
[nombre del campo][operador][valor]

Operadores disponibles:

| **Operador** | **Uso** |
| --- | --- |
| > | Mayor que |
| < | Menor que |
| == | Igual que |
| = | Igual que |
| >= | Mayor o igual que |
| <= | Menor o igual que |
| like | Like |

Ordenamiento de los elementos

OrderBy: también se permite el ordenar los resultados dependiendo de los atributos que el recurso tenga disponible, esto se hace enviando el parámetro {orderby} por medio de GET junto con el nombre del atributo por el cual se ordenarán los resultados.

/users?orderby=lastname

OrderByDesc: funciona de igual forma que orderby, solo que en esta ocasión se ordenan en sentido contrario a orderby.

/users?orderbydesc=lastname

## Inserción de un recurso

Se enviará una petición a través del método POST.

En esta versión, la API de wisiolms está diseñada para poder agregar un recurso o hacerlo de forma masiva. Si se quiere ingresar uno solo, la petición debe contener los campos que conformarán el recurso a crear. Si alguno de los campos que se ingresan no existen en la tabla del recurso, simplemente se omiten para no generar errores. También se agregan validaciones en cada uno de los recursos para revisar si se está duplicando información que debería ser única o si es requerida y no se está enviando. En esos casos no se creará el recurso y se regresarán los errores correspondientes.

Listado de recursos:   
url/api/v1/users

url/api/v1/users

url/api/v1/users

url/api/v1/users

url/api/v1/users

url/api/v1/users

url/api/v1/users

url/api/v1/users

url/api/v1/users

Para hacer válido el envío de elementos se deben encapsular en un arreglo para ser enviado en la petición, este arreglo está en la

Existen recursos en los cuales se recomienda subir una imagen, por ello se da la posibilidad de subir la imagen en la misma petición POST. Los recursos que permiten agregar imágenes son los siguientes: course, category, module, ascription, resource. Para enviar la imagen se deben cumplir los siguientes puntos:

* Enviar el archivo con el siguiente nombre: file
* Agregar la siguiente cabecera a la petición: Content-Type: multipart/form-data

En ellos se debe agregar solamente un elemento

## Consulta de un solo recurso

## Actualización de un recurso

## Eliminación de un recurso

# Información de los atributos de los recursos:

## Recurso: ascriptions

Atributos que pueden ser modificados/consultados: id, name, slug, description, is\_pharmacy, enabled, code.

Atributos requeridos: slug, name.

Atributos únicos: slug.

## Recurso: categories

Atributos que pueden ser modificados/consultados: name, description.

Atributos requeridos: name.

## Recurso: courses

Atributos que pueden ser modificados/consultados: id, name, slug, credits, description, minimum\_score, has\_constancy, enabled, start\_date, end\_date, category\_id, support\_email, minimum\_diploma\_score, certificate\_template\_id.

Atributos requeridos: name, slug.

Atributos únicos: slug.

## Recurso: diplomas

Atributos que pueden ser modificados/consultados: name, slug, description, minimum\_score, minimum\_previous\_score, ascription\_id, attachment\_id, diploma\_id.

Atributos requeridos: name, ascription\_id.

Atributos únicos: slug.

## Recurso: evaluations

Atributos que pueden ser modificados/consultados: id, module\_id, diploma\_id, name, type, description, maximum\_attempts.

Atributos requeridos: name, type.

## Recurso: experts

Atributos que pueden ser modificados/consultados: name, slug, summary.

Atributos requeridos: name, slug, summary.

## Recurso: modules

Atributos que pueden ser modificados/consultados: id, name, description, previous, sort, is\_for\_diploma, course\_id.

Atributos requeridos: name, course\_id.

## Recurso: options

Atributos que pueden ser modificados/consultados: id, question\_id, content, feedback, score.

Atributos requeridos: question\_id, content, score.

## Recurso: questions

Atributos que pueden ser modificados/consultados: id, evaluation\_id, name, content, type, is\_true.

Atributos requeridos: evaluation\_id, content.

## Recurso: references

Atributos que pueden ser modificados/consultados: module\_id, content.

Atributos requeridos: module\_id, content.

## Recurso: resources

Atributos que pueden ser modificados/consultados: attachment\_id, type, previous, weight, module\_id, name.

Atributos requeridos: type, weight, attachment\_id.

## Recurso: specialties

Atributos que pueden ser modificados/consultados: id, name, code.

Atributos requeridos: name, code.

## Recurso: states

Atributos que pueden ser modificados/consultados: name, code.

Atributos requeridos: name, code.

## Recurso: tags

Atributos que pueden ser modificados/consultados: tag.

Atributos requeridos: tag.

## Recurso: users

Atributos que pueden ser modificados/consultados: firstname, lastname, email, password, birthday, gender, mobile\_phone, city, state\_id, address, professional\_license, consultation\_type, last\_access, enabled, role\_id, is\_validated, refered\_code, institution, ascription\_id, id.

Atributos requeridos: password, email.

Atributos únicos: email.

Agregar en la plantilla el periodo en el cual se tienen ingresos, el texto lo estaban agregando en la matriz.

Botón atrás en las partes donde se ingresen cantidades: misingresos.html, misgastos\_hogar.html, mis-gastos-fijos-ahorro.html, mis-gastos-fijos-deudas.html, misgastos\_salud.html, mis-gastos-variables-transporte.html, misgastos\_personales, misgastos\_entretenimiento