

**Facultad de Ingeniería   
Escuela de computación**

**Lenguajes Interpretados en el Servidor**

**Investigación aplicada II:   
REST y GraphQL en PHP**

**Docente:** Guillermo Calderón

**Alumnos: Carnet:**

Ludwig Stanley Pérez Campos PC191777

Eduardo Josué Deras Mancia DM201736

**API REST**  
  
"Representational State Transfer" es un estilo de arquitectura para sistemas distribuidos en la web.

Una API RESTful es una interfaz de programación de aplicaciones (API) basada en el protocolo HTTP. Esta API permite a los desarrolladores acceder y manipular recursos en un servidor a través de URLs y métodos HTTP estándar, como GET, POST, PUT y DELETE.

Con una API RESTful, los recursos son representados por URLs únicas y los métodos HTTP correspondientes se utilizan para crear, leer, actualizar o eliminar (CRUD) estos recursos.

Las API RESTful son populares en la construcción de servicios web y aplicaciones, ya que permiten a los desarrolladores crear servicios web escalables y altamente disponibles que se integran fácilmente con otras aplicaciones y servicios.

**Generalidades sobre las API’s tipo REST**

1. **Utilizan el protocolo HTTP:** Las APIs REST se basan en el protocolo HTTP, que es el mismo protocolo que se utiliza para navegar por la web. Los verbos HTTP, como GET, POST, PUT y DELETE, se utilizan para definir las operaciones CRUD sobre los recursos.
2. **Utilizan el formato JSON:** La mayoría de las APIs REST utilizan JSON (JavaScript Object Notation) como formato de intercambio de datos. JSON es un formato de texto sencillo y ligero que es fácil de leer y escribir para los humanos, y fácil de parsear y generar por las aplicaciones.
3. **Son Stateless:** Las APIs REST no mantienen información sobre el estado de la sesión del usuario, por lo que cada solicitud realizada por el cliente debe incluir toda la información necesaria para que el servidor responda adecuadamente.
4. **Utilizan URIs para identificar los recursos:** Cada recurso en una API REST se identifica mediante una URI única. Los clientes utilizan estas URIs para acceder y manipular los recursos.
5. **Son independientes del lenguaje de programación:** Las APIs REST son independientes del lenguaje de programación utilizado para crear el servidor y el cliente. Esto significa que se pueden crear clientes y servidores REST en cualquier lenguaje de programación que tenga soporte para HTTP y JSON.
6. **Son fáciles de probar:** Las APIs REST son fáciles de probar con herramientas como Postman y cURL. Con estas herramientas, los desarrolladores pueden enviar solicitudes HTTP y ver las respuestas en tiempo real.

**Códigos de estado de respuesta devueltos por las API REST**

Los códigos de estado de respuesta son un componente clave de cualquier API RESTful. Estos códigos se devuelven en respuesta a una solicitud realizada por un cliente y proporcionan información sobre el éxito o el fracaso de la solicitud. Los códigos de estado se envían en la línea de estado de la respuesta HTTP y se clasifican en cinco categorías:

1. 1xx (Respuestas informativas): Este tipo de respuesta es una respuesta provisional, indicando que el servidor ha recibido la solicitud del cliente y está procesando la solicitud.
2. 2xx (Respuestas satisfactorias): Este tipo de respuesta indica que la solicitud del cliente se ha procesado correctamente y se ha devuelto una respuesta satisfactoria. El código de estado más común en esta categoría es el 200 OK, que indica que la solicitud se procesó correctamente.
3. 3xx (Redirecciones): Este tipo de respuesta indica que el cliente debe tomar medidas adicionales para completar la solicitud. Los códigos de estado más comunes en esta categoría son el 301 Moved Permanently y el 302 Found, que se utilizan para redirigir el tráfico a una nueva URL.
4. 4xx (Errores del cliente): Este tipo de respuesta indica que la solicitud del cliente no se pudo procesar correctamente debido a un error del cliente. Los códigos de estado más comunes en esta categoría son el 400 Bad Request, el 401 Unauthorized y el 404 Not Found.
5. 5xx (Errores del servidor): Este tipo de respuesta indica que la solicitud del cliente no se pudo procesar correctamente debido a un error del servidor. Los códigos de estado más comunes en esta categoría son el 500 Internal Server Error y el 503 Service Unavailable.

Es importante que los desarrolladores que trabajan con API RESTful comprendan los códigos de estado de respuesta y cómo interpretarlos, ya que estos códigos proporcionan información valiosa sobre el estado y el éxito de las solicitudes que se envían a la API.

**Métodos y tecnologías para la autenticación de APIS REST**

La autenticación es un aspecto crítico en la seguridad de cualquier API REST. Algunos de los métodos y tecnologías más utilizados para la autenticación de API REST:

1. **Autenticación basada en token:** Este método implica la creación de tokens de acceso que se utilizan para autenticar las solicitudes del cliente. Cuando un cliente inicia sesión, el servidor crea un token de acceso y lo devuelve al cliente. El cliente incluye este token en todas las solicitudes subsiguientes para autenticar su identidad.
2. **Autenticación basada en OAuth:** OAuth es un protocolo de autorización que permite a los usuarios autorizar a las aplicaciones para acceder a sus recursos protegidos sin compartir sus credenciales de inicio de sesión. En este método, el usuario autoriza a la aplicación a acceder a sus recursos protegidos, y la aplicación obtiene un token de acceso que utiliza para acceder a estos recursos.
3. **Autenticación basada en API keys:** Este método implica la asignación de una clave de API única a cada cliente que se registra. La clave se utiliza para autenticar la identidad del cliente en cada solicitud.
4. **Autenticación basada en SSL/TLS:** SSL/TLS es un protocolo de seguridad que se utiliza para cifrar las comunicaciones entre un cliente y un servidor. Al utilizar SSL/TLS para autenticar una API REST, se puede garantizar que la comunicación entre el cliente y el servidor sea segura y que la autenticación sea válida.

Es importante destacar que, aunque estos métodos y tecnologías son útiles para garantizar la seguridad y la autenticación de una API REST, es importante evaluar cuidadosamente las necesidades de seguridad de su aplicación y seleccionar la solución de autenticación adecuada para su caso de uso específico.

**GraphQL**

**a. Generalidades sobre GraphQL:**

GraphQL es un lenguaje de consulta y manipulación de datos de código abierto que se utiliza para crear APIs. Fue desarrollado por Facebook en 2012 como una alternativa a las API RESTful. Desde su lanzamiento, GraphQL ha ganado popularidad entre los desarrolladores de todo el mundo debido a su eficiencia y flexibilidad.

La eficiencia de GraphQL se debe a que los clientes pueden solicitar solo los datos que necesitan, evitando solicitudes innecesarias. Además, al utilizar una única solicitud, se reduce la sobrecarga de la red y se mejora el rendimiento de la aplicación.

La flexibilidad de GraphQL se debe a su sistema de tipos, que permite a los clientes definir la estructura de los datos que desean. Esto significa que los clientes pueden solicitar solo los campos que necesitan y omitir los que no necesitan, lo que reduce la sobrecarga de datos y mejora el rendimiento de la aplicación.

**Playground**

GraphQL Playground es una herramienta gráfica interactiva para probar y depurar consultas, mutaciones y suscripciones GraphQL. Es una aplicación web que se ejecuta en el navegador y se puede usar para interactuar con cualquier servidor GraphQL.

Una de las características principales de GraphQL Playground es su interfaz de usuario intuitiva y fácil de usar, que permite a los desarrolladores crear y probar consultas y mutaciones con facilidad. La herramienta ofrece resaltado de sintaxis, autocompletado y documentación integrada, lo que ayuda a los usuarios a escribir consultas y mutaciones de manera más eficiente.

Además, GraphQL Playground permite la edición de variables y ofrece una vista previa en tiempo real de los resultados de la consulta o mutación. Los desarrolladores también pueden ver los errores de sintaxis y los errores de tiempo de ejecución en la herramienta, lo que ayuda a depurar los problemas más rápidamente.

Otra de las ventajas de GraphQL Playground es que ofrece una funcionalidad de introspección, que permite a los desarrolladores explorar la estructura de los esquemas GraphQL y comprender mejor la forma en que los datos están organizados y cómo se pueden consultar.

GraphQL Playground es compatible con cualquier servidor GraphQL que admita la especificación de GraphQL, y se puede utilizar como una herramienta independiente o integrada en un proyecto. En resumen, es una herramienta muy útil para cualquier desarrollador que trabaje con GraphQL y necesite probar y depurar consultas y mutaciones

**b. Comparación de REST y GraphQL:**

REST y GraphQL son dos enfoques diferentes para la creación de APIs. REST utiliza endpoints para definir la estructura de los datos y las operaciones permitidas. Los clientes realizan solicitudes a estos endpoints para obtener los datos que necesitan. En comparación, GraphQL utiliza un único endpoint y una sintaxis de consulta para definir la estructura y los datos requeridos.

En GraphQL, los clientes pueden especificar exactamente qué datos necesitan y cómo desean que se estructuren, lo que reduce la necesidad de múltiples solicitudes y aumenta la eficiencia. En REST, los clientes deben realizar múltiples solicitudes para obtener los datos que necesitan, lo que puede ser ineficiente y aumentar la sobrecarga de la red.

**c. Tipos y consultas en GraphQL:**

En GraphQL, los tipos son un conjunto de reglas que definen la estructura de los datos disponibles en la API y las relaciones entre ellos. Los tipos pueden ser objetos, interfaces, uniones y enumeraciones. Cada tipo tiene campos que definen los datos que se pueden recuperar y las relaciones entre los datos.

Las consultas en GraphQL se realizan utilizando una sintaxis de consulta que define la estructura y los datos que se solicitan. Las consultas pueden incluir argumentos para filtrar y ordenar los datos solicitados. También es posible anidar consultas y especificar relaciones entre los datos. La sintaxis de consulta de GraphQL es sencilla y fácil de entender para los desarrolladores.

**d. Mutaciones y suscripciones en GraphQL:**

Además de las consultas, GraphQL también permite realizar mutaciones, que son operaciones que modifican los datos en el servidor, y suscripciones, que permiten a los clientes recibir actualizaciones en tiempo real de los datos en el servidor.

Las mutaciones en GraphQL se definen utilizando una sintaxis similar a las consultas, pero con un conjunto de palabras clave diferente. Las mutaciones pueden incluir argumentos y variables para especificar los datos que se van a modificar. Las mutaciones también permiten a los desarrolladores especificar la estructura de los datos que se devolverán después de la modificación.

Las suscripciones en GraphQL permiten a los clientes recibir actualizaciones en tiempo real de los datos en el servidor. Las suscripciones se definen utilizando una sintaxis especial y se utilizan para recibir actualizaciones de los datos en tiempo real. Las suscripciones se pueden utilizar para enviar mensajes push a los clientes cuando se actualizan los datos, lo que es útil para aplicaciones que requieren actualizarse.

**La instalación básica de GraphQL y Laravel**

1. Crear un nuevo proyecto Laravel utilizando el comando:

laravel new NOMBRE\_DEL\_PROYECTO

1. Instalar el paquete nuwave/lighthouse de GraphQL utilizando el comando:

composer require nuwave/lighthouse

1. Instalar el paquete mll-lab/laravel-graphql-playground para tener acceso a la herramienta GraphQL Playground, utilizando el comando:

composer require mll-lab/laravel-graphql-playground

1. Publicar el archivo de esquema GraphQL utilizando el comando:

php artisan vendor:publish --tag=lighthouse-schema.

Este comando copiará el archivo graphql/schema.graphql en el directorio app/ de tu proyecto Laravel.