CREAR UN SERVICIO WEB (API REST) CON LARAVEL 5.5 Y CONSUMIRLO DESDE ANDROID

Cómo crear un servicio Web (Api Rest) con Laravel 5.5 y consumirlo desde Android, lo que vamos hacer es básicamente consumir información de una aplicación hecha en Laravel (PHP) y mostrarla en una aplicación en Android, aunque para entender el tutorial es de mucha importancia tener los conceptos claros (términos como Web Services, ApiRest).

Como te mencionaba anteriormente en el artículo Cómo crear un Crud con con Laravel 5.5 habíamos creado un crud básico sobre una tabla, lo que vamos hacer que estos datos se puedan ver desde una aplicación Android lo que quiere decir que también debes tener instalado Android Studio 3.0 para este ejemplo. Si todavía no tienes instalado Android Studio en este artículo te explico Cómo instalar Android Studio 3.0 para desarrollar Aplicaciones Móviles.

CREAR EL SERVICIO DESDE LA APLICACIÓN EN LARAVEL

La creación de servicios de acuerdo a la filosofía REST parte de la creación de métodos que puedan ser consumidos con peticiones HTTP (en el caso de consultas se puede usar el verbo GET), lo que quiere decir que cada método identificará un único recurso a ser consumido en la aplicación.

Ahora lo que vamos hacer es crear un servicio que permita leer la información (registros de libros) del crud anterior en la aplicación Android.

CREAR EL SERVICIO

El diseño de una Api Rest en Laravel es muy sencilla, para esto en nuestra aplicación crud en el archivo LibroController vamos a crear el siguiente método:

/\*\*

\* Ejemplo de método REST

\*

\* @return \Illuminate\Http\Response

\*/

public function getLibros(){

$libros=Libro::all();

return response()->json($libros);

}

Lo que estamos haciendo es obtener los registros de la tabla libros y devolverlos en formato JSON, para que puedan ser consumido por cualquier aplicación en nuestro caso por Android.

Lo siguiente que nos queda es mapear el método anterior en la ruta para poder hacer la petición y comprobar que funciona, para esto vamos adicionar lo siguiente en el archivo web.php (dentro de la carpeta route):

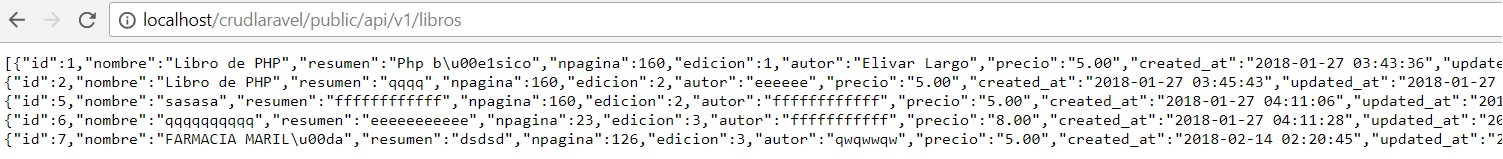
Route::get('api/v1/libros','LibroController@getLibros');

Route::get('api/v1/libros','LibroController@getLibros');

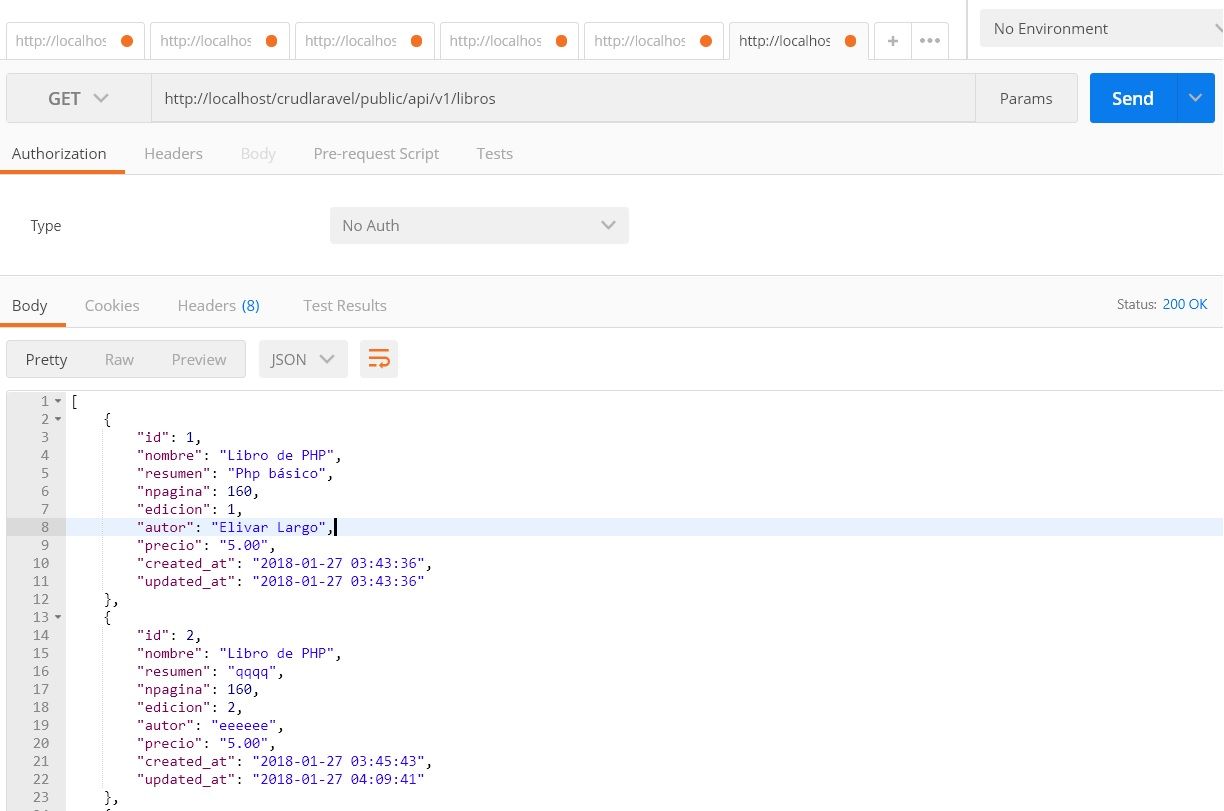
Aquí lo que le estamos diciendo es que cuando haya una petición HTTP y que sea de tipo GET, con la terminación ‘api/v1/libros’, se mapee al método getLibros del controlador.

PROBAR QUE EL SERVICIO PUEDA SER CONSUMIDO POR UN CLIENTE

Para esto debe estar iniciado XAMP puesto que la aplicación y el método que creamos está en el servidor y para consultar que todo va bien vamos al navegador y en la URL escribimos localhost/nombre-del-proyecto/nombre-del-servicio (nombre definido en la ruta) como se ve a continuación (los registros devueltos dependerán de lo que tengas guardado en tu base de datos).



Lo que hicimos en el paso anterior es una buena forma de probar nuestro servicio pero también podemos usar un aplicación que se llama POSTMAN y que la puedes descargar del siguiente enlace descarga de postman, una vez instalada pones la url del proyecto junto con el servicio (el método con el que haces la petición), luego le das SEND y deberías tener algo como esto.



Si te das cuenta lo que nos devuelve nuestro servicio es un arreglo de libros y con esto nos aseguramos que estamos devolviendo los datos en formato JSON y que se pueden consumir con nuestro cliente en Android.

CREAR LA APLICACIÓN MÓVIL EN ANDROID

Ya con nuestra aplicación creada y que en mi caso se llama Cliente Libros vamos a crear los archivos necesarios para crear el cliente y poder consumir nuestro servicio REST creado en Laravel.

Aunque hay varias librerías para crear clientes que consuman servicios REST en Android en este tutorial voy ha usar Retrofit y para esto debemos agregar las dependencias que vamos a usar en nuestros archivo build.gradle (Module:app).

//librerías para crear un cliente REST en Android

compile 'com.squareup.retrofit2:retrofit:2.2.0'

compile 'com.squareup.retrofit2:converter-gson:2.2.0'

1

2

3

//librerías para crear un cliente REST en Android

compile 'com.squareup.retrofit2:retrofit:2.2.0'

compile 'com.squareup.retrofit2:converter-gson:2.2.0'

También debemos agregar los permisos de acceso a internet para esto añadimos lo siguiente dentro del archivo AndroidManifiest.xml (Justo antes de <aplication):

<uses-permission android:name="android.permission.INTERNET" />

1

<uses-permission android:name="android.permission.INTERNET" />

ARCHIVOS PARA CONSUMIR EL SERVICIO DESDE LA APLICACIÓN ANDROID

Lo primero que vamos a crear es nuestro modelo que es donde vamos a recibir cada objeto Libro, los atributos debe llamarse igual que los recibidos en el servicio creado, creamos la clase Libro como se ve a continuación.

Clase LibroJava

package com.ecodeup.clientelibros;

/\*\*

\* Created by Elivar on 22/2/2018.

\*/

public class Libro

{

private int id;

private String nombre;

private String resumen;

private String npagina;

private String edicion;

private String autor;

private String precio;

private String created\_at;

private String updated\_at;

public Libro() {

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getNombre() {

return nombre;

}

public void setNombre(String nombre) {

this.nombre = nombre;

}

public String getResumen() {

return resumen;

}

public void setResumen(String resumen) {

this.resumen = resumen;

}

public String getNpagina() {

return npagina;

}

public void setNpagina(String npagina) {

this.npagina = npagina;

}

public String getEdicion() {

return edicion;

}

public void setEdicion(String edicion) {

this.edicion = edicion;

}

public String getAutor() {

return autor;

}

public void setAutor(String autor) {

this.autor = autor;

}

public String getPrecio() {

return precio;

}

public void setPrecio(String precio) {

this.precio = precio;

}

public String getCreated\_at() {

return created\_at;

}

public void setCreated\_at(String created\_at) {

this.created\_at = created\_at;

}

public String getUpdated\_at() {

return updated\_at;

}

public void setUpdated\_at(String updated\_at) {

this.updated\_at = updated\_at;

}

@Override

public String toString() {

return this.id+" "+this.nombre+" "+this.resumen+" "+this.npagina+" "+edicion+" "+this.autor+" "+this.precio+" "+this.created\_at+" "+this.updated\_at;

}

}

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

package com.ecodeup.clientelibros;

/\*\*

\* Created by Elivar on 22/2/2018.

\*/

public class Libro

{

private int id;

private String nombre;

private String resumen;

private String npagina;

private String edicion;

private String autor;

private String precio;

private String created\_at;

private String updated\_at;

public Libro() {

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getNombre() {

return nombre;

}

public void setNombre(String nombre) {

this.nombre = nombre;

}

public String getResumen() {

return resumen;

}

public void setResumen(String resumen) {

this.resumen = resumen;

}

public String getNpagina() {

return npagina;

}

public void setNpagina(String npagina) {

this.npagina = npagina;

}

public String getEdicion() {

return edicion;

}

public void setEdicion(String edicion) {

this.edicion = edicion;

}

public String getAutor() {

return autor;

}

public void setAutor(String autor) {

this.autor = autor;

}

public String getPrecio() {

return precio;

}

public void setPrecio(String precio) {

this.precio = precio;

}

public String getCreated\_at() {

return created\_at;

}

public void setCreated\_at(String created\_at) {

this.created\_at = created\_at;

}

public String getUpdated\_at() {

return updated\_at;

}

public void setUpdated\_at(String updated\_at) {

this.updated\_at = updated\_at;

}

@Override

public String toString() {

return this.id+" "+this.nombre+" "+this.resumen+" "+this.npagina+" "+edicion+" "+this.autor+" "+this.precio+" "+this.created\_at+" "+this.updated\_at;

}

}

Ahora vamos a crear la interface ApiService donde declaramos el servicio que vamos a consumir y le decimos que traemos una lista de libros:

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

package com.ecodeup.clientelibros;

import java.util.List;

import retrofit2.Call;

import retrofit2.http.GET;

/\*\*

\* Created by Elivar on 22/2/2018.

\*/

public interface ApiService {

public static final String URL="http://192.168.1.7:80/crudlaravel/public/";

@GET("api/v1/libros")

Call<List<Libro>>listaLibros();

}

Es importante mencionar que la URL es la direccion IP del equipo donde vas a consumir el servicio, no funciona con localhost así que debes poner la IP y debes tener cuidado ya que cuando apagas y luego la enciendes tu computadora tu router te va asignar otra IP y si quieres de nuevo probar el cliente deberás poner la IP que actualmente te asigne tu router.

CONSUMO DEL SERVICIO DESDE LA APLICACIÓN EN ANDROID

A continuación creamos el cliente para consumir el servicio y enviamos a imprimir la lista de libros que nos devuelve el servicio creado en Laravel.

package com.ecodeup.clientelibros;

import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

import android.util.Log;

import java.util.List;

import retrofit2.Call;

import retrofit2.Callback;

import retrofit2.Response;

import retrofit2.Retrofit;

import retrofit2.converter.gson.GsonConverterFactory;

public class MainActivity extends AppCompatActivity {

List<Libro> listaLibros;

Retrofit cliente;

ApiService apiService;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

cliente= new Retrofit.Builder().baseUrl(ApiService.URL).addConverterFactory(GsonConverterFactory.create()).build();

apiService=cliente.create(ApiService.class);

apiService.listaLibros().enqueue(new Callback<List<Libro>>() {

@Override

public void onResponse(Call<List<Libro>> call, Response<List<Libro>> response) {

Log.i("Cliente","Cliente Android");

if (response.isSuccessful()){

listaLibros=response.body();

for (Libro libro:listaLibros){

Log.i("Libro",libro.toString());

}

}

}

@Override

public void onFailure(Call<List<Libro>> call, Throwable t) {

Log.i("Error",t.getMessage());

}

});

}

}

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

package com.ecodeup.clientelibros;

import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

import android.util.Log;

import java.util.List;

import retrofit2.Call;

import retrofit2.Callback;

import retrofit2.Response;

import retrofit2.Retrofit;

import retrofit2.converter.gson.GsonConverterFactory;

public class MainActivity extends AppCompatActivity {

List<Libro> listaLibros;

Retrofit cliente;

ApiService apiService;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

cliente= new Retrofit.Builder().baseUrl(ApiService.URL).addConverterFactory(GsonConverterFactory.create()).build();

apiService=cliente.create(ApiService.class);

apiService.listaLibros().enqueue(new Callback<List<Libro>>() {

@Override

public void onResponse(Call<List<Libro>> call, Response<List<Libro>> response) {

Log.i("Cliente","Cliente Android");

if (response.isSuccessful()){

listaLibros=response.body();

for (Libro libro:listaLibros){

Log.i("Libro",libro.toString());

}

}

}

@Override

public void onFailure(Call<List<Libro>> call, Throwable t) {

Log.i("Error",t.getMessage());

}

});

}

}

Y bien como puedes ver que en cuestión de minutos puedes crear un sencillo servicio REST en Laravel y consumirlo desde Android.

Guía:

https://www.ecodeup.com/como-crear-un-servicio-web-api-rest-con-laravel-5-5-y-consumirlo-desde-android/