

LCKM INNOVATY
JUAN CAMILO RODRIGUEZ RAMIREZ

PROYECTO BINGO FOPRE

**PRUEBAS DE FUNCIONALIDAD PARA LA BASE DE DATOS Y APlicativo CONSUMIDAS EN
POSTMAN**
JUAN CAMILO RODRIGUEZ RAMIREZ
UNIVERSIDAD DE LOS ANDES COLOMBIA – DIRECCIÓN DE RELACIONAMIENTO
2023

CONTENIDO

INTRODUCCIÓN	5
OBJETIVO	5
REQUISITOS PREVIOS	5
ESTRUCTURA DEL MANUAL.....	5
SECCIONES	6
CONFIGURACIÓN DE ESTADOS.....	6
Función para listar todos los estados.....	6
Función para crear estados	8
Función para visualizar estados por ID.....	9
Función para editar estado por ID	10
Función para eliminar estados	11
CONFIGURACIÓN DEL USUARIOS.....	12
Función para listar usuarios	12
Función para crear usuarios	14
Función para visualizar usuarios por ID	15
Función parar editar usuarios por id	16
Función para eliminar de usuarios por id.....	17
CONFIGURACIÓN DEL SITIO WEB.....	18
Función para listar componentes de sitio	18
Función para crear componente del sitio	19
Función para visualizar id de componentes.....	20
Función para editar componentes	21
Función para eliminación de componentes	22
NOTICIAS GENERALES.....	23
Función para listar noticias	23
Función para crear noticias	24
Función para Visualizar noticias	25
Función para editar noticias general.....	26
Función para eliminar noticias general	27
PATROCINADORES.....	28
Función para listar patrocinadores	28

Función para crear patrocinadores	29
Función para visualizar patrocinadores por ID.....	30
Función para edicitar patrocinador por id	31
INSTRUCCIONES.....	33
Función para listar instrucciones	33
Función para crear instrucciones	34
Función para visualizar instrucciones por id	35
Función para editar instrucciónes por ID	36
Función para eliminar instrucción por ID.....	37
DINÁMICAS DEL JUEGO	38
Función para listar las dinámicas del juego.....	38
Función para crear dinámicas del juego.....	39
Función para Visualizar dinámicas del juego por ID.....	40
Función para editar dinámica del juego por ID	41
Función para eliminar dinámica de juego por id.....	42
PREMIOS	43
Función para listar de premios.....	43
Función para crear premios	44
Función para visualizar premios por ID	45
Función para editar premios	46
Función para eliminar de premio por id.....	47
CARTONES.....	48
Función para listar todos los cartones	48
Función para crear cartones masivamente.....	49
Función para visualizar cartones por ID	51
Función para añadir un carton al carrito.....	52
Función para finalizer compra del carton	53
Función para eliminar un carton en el carrito.....	55
Función para editar carton por ID	56
Función para eliminar carton por id.....	57
GRUPO DE CARTONES	58
Función para listar grupo de cartones	58
Función para crear un grupo carton.....	59

Función para visualizar un grupo de carton por id.....	60
Función para editar grupo de cartones.....	61
Función para eliminar grupo de cartones	62

INTRODUCCIÓN

El siguiente manual de Pruebas del Bingo Fopre desarrollado en el framework Laravel v10, donde se desarrollaron las pruebas utilizando Postman. Este manual está diseñado para proporcionar una guía completa sobre cómo evaluar y verificar la funcionalidad de la aplicación, así como para asegurar su calidad y fiabilidad.

OBJETIVO

El objetivo principal de este manual es ofrecer a los miembros del equipo de desarrollo, QA y cualquier otra persona interesada en el proyecto, una referencia detallada de las pruebas que se han llevado a cabo. Además, proporciona información sobre cómo ejecutar estas pruebas utilizando Postman, una herramienta popular para probar API's.

REQUISITOS PREVIOS

Antes de comenzar a utilizar este manual, es necesario tener instalado y configurado Postman en su entorno de desarrollo. Asegúrese de contar con una copia funcional de la aplicación Laravel y de disponer de acceso a la API para realizar pruebas.

ESTRUCTURA DEL MANUAL

Este manual se organiza en secciones que corresponden a las áreas mencionadas anteriormente. Cada sección contiene instrucciones detalladas sobre cómo realizar las pruebas, así como ejemplos de solicitudes y respuestas.

SECCIONES

En cada sección se podrá evidenciar el código necesario y la petición para realizar cada una de las pruebas con Postman.

CONFIGURACIÓN DE ESTADOS

Ruta: esto lo podrá encontrar App/Http/Controller/Api/States/StatesController.php

Función para listar todos los estados.

```
class StatesController extends Controller
{
    /**
     * Display a listing of the resource.
     */
    public function index()
    {
        try {
            $states = State:: all();
            return response()->json($states, 200);
        } catch (\Throwable $th) {
            return response()->json([
                'errors' => $th
            ], 400);
        }
    }
}
```

Petición Postman: aquí se recibe la petición de todos los estados.

LCKM INNOVATY

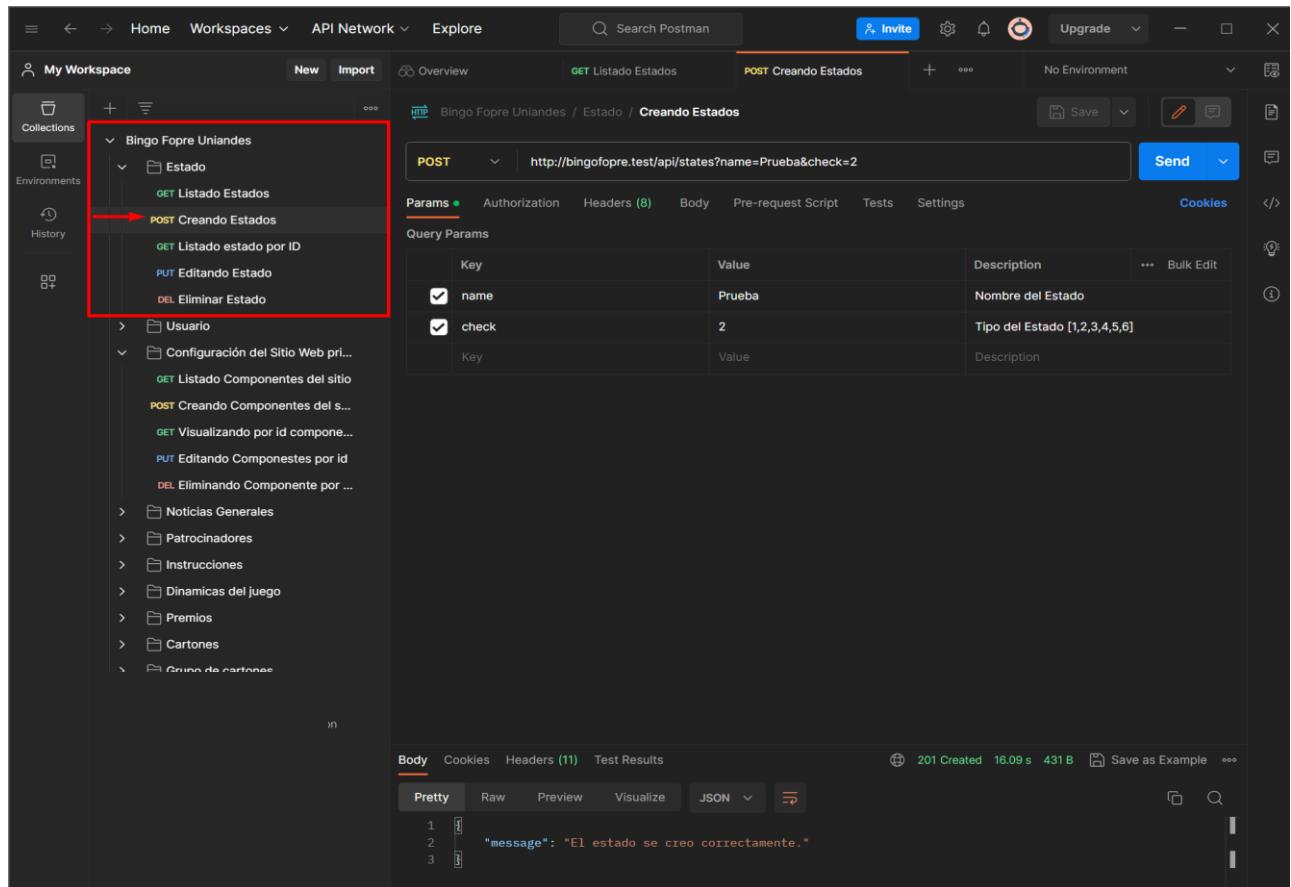
JUAN CAMILO RODRIGUEZ RAMIREZ

The screenshot shows the Postman application interface. On the left, the 'My Workspace' sidebar displays a collection named 'Bingo Fopre Unidos'. A red box highlights the 'Estado' folder, which contains several API endpoints: 'GET Listado Estados', 'POST Creando Estados', 'GET Listado estado por ID', 'PUT Editando Estado', and 'DEL Eliminar Estado'. An arrow points to the 'GET Listado Estados' endpoint. The main workspace shows a 'Listado Estados' request with the URL 'http://bingofopre.test/api/states'. The 'Headers' tab is selected, showing a table with one row and two columns: 'Key' and 'Value'. The response section shows a 200 OK status with a response time of 21.55 s and a size of 1.12 KB. The response body is displayed in JSON format:

```
[{"id": 1, "name": "Disponible", "check": "1", "created_at": "2023-09-16T17:06:57.000000Z", "updated_at": "2023-09-16T17:06:57.000000Z"}, {"id": 2, "name": "No disponible", "check": "2", "created_at": "2023-09-16T17:06:57.000000Z", "updated_at": "2023-09-16T17:06:57.000000Z"}, {"id": 3, "name": "Circulación", "check": "3", "created_at": "2023-09-16T17:06:57.000000Z", "updated_at": "2023-09-16T17:06:57.000000Z"}, {"id": 4, "name": "Anulado", "check": "4", "created_at": "2023-09-16T17:06:57.000000Z", "updated_at": "2023-09-16T17:06:57.000000Z"}]
```

Función para crear estados

```
/**  
 * Show the form for creating a new resource.  
 */  
  
/**  
 * Store a newly created resource in storage.  
 */  
public  
function store(Request $request)  
{  
    try {  
        $state = State:: create($request->all());  
        return response()->json(['message' => 'El estado se creo correctamente.'], 201);  
    } catch (\Throwable $th) {  
        return response()->json([  
            'errors' => $th  
        ], 400);  
    }  
}
```



The screenshot shows the Postman application interface. On the left, there's a sidebar with 'My Workspace' containing collections like 'Bingo Fopre Uniandes' and 'Usuario'. A red box highlights the 'Bingo Fopre Uniandes' collection, specifically the 'Estado' folder which contains a 'POST Creando Estados' item. The main workspace shows a POST request to 'http://bingofopre.test/api/states?name=Prueba&check=2'. The 'Params' tab is selected, showing 'name' with value 'Prueba' and 'check' with value '2'. The 'Body' tab shows a JSON response: { "message": "El estado se creo correctamente." }. The status bar at the bottom indicates a 201 Created response.

Función para visualizar estados por ID

```
/**  
 * Display the specified resource.  
 */  
public  
function show(string $id)  
{  
    try {  
        $state = State:: find($id);  
        return response()->json($state, 200);  
    } catch (\Throwable $th) {  
        return response()->json([  
            'errors' => $th  
        ], 400);  
    }  
}
```

The screenshot shows the Postman application interface. On the left, the 'Bingo Fopre Unlandes' collection is expanded, revealing the 'Estado' folder. Inside 'Estado', the 'GET Listado estado por ID' endpoint is highlighted with a red box and an arrow. The main workspace displays the 'Listado estado por ID' request details. The method is set to 'GET', the URL is 'http://bingofopre.test/api/states/7', and the 'Params' tab is selected. The response body is shown in JSON format:

```
1  [ {  
2      "id": 7,  
3      "name": "Prueba",  
4      "check": "2",  
5      "created_at": "2023-09-16T19:49:05.000000Z",  
6      "updated_at": "2023-09-16T19:49:05.000000Z"  
7  } ]
```

Función para editar estado por ID

```

/**
 * Show the form for editing the specified resource.
 */

/**
 * Update the specified resource in storage.
 */
public
function update(Request $request, string $id)
{
    try {
        $state = State::find($id)->update($request->all());
        return response()->json(['message' => 'El estado se actualizó correctamente'], 202);
    } catch (\Throwable $th) {
        return response()->json([
            'errors' => $th
        ], 400);
    }
}

```

The screenshot shows the Postman interface with the following details:

- Collection:** Bingo Fopre Unionandes
- Request Type:** PUT
- URL:** http://bingofopre.test/api/states/7?name=Prueba 10
- Params:** A table with two rows:

Key	Value	Description
name	Prueba 10	Nombre del estado
check		Tipo del Estado [1,2,3,4,5,6]
- Body:** A JSON object with a single key "message": "El estado se actualizó correctamente".
- Status:** 202 Accepted

Función para eliminar estados

```
/**  
 * Remove the specified resource from storage.  
 */  
public  
function destroy(string $id)  
{  
    try {  
        $state = State:: find($id)->delete();  
        return response()->json([ 'message' => 'El estado se eliminó correctamente'],202);  
    } catch (\Throwable $th) {  
        return response()->json([  
            'errors' => $th  
        ], 400);  
    }  
}
```

The screenshot shows the Postman interface for testing an API endpoint. On the left, the 'Collections' sidebar is visible with the 'Bingo Fopre Uniandes' collection expanded, showing the 'Estado' folder which contains several requests: 'Listado Estados', 'Creando Estados', 'Listado estado por ID', 'Editando Estado', and 'Eliminar Estado'. A red arrow points to the 'Eliminar Estado' item. The main workspace shows a 'DELETE' request for the URL `http://bingofopre.test/api/states/7`. The 'Params' tab is selected, showing a single query parameter 'Key' with the value 'Value'. The 'Body' tab shows a JSON response with the key 'message' and the value 'El estado se eliminó correctamente'. The status bar at the bottom indicates a 202 Accepted response with 427 ms and 439 B.

CONFIGURACIÓN DEL USUARIOS

Ruta: /App/Http/Controllers/Api/Users/UsersController.php

Función para listar usuarios

```
class UsersController extends Controller
{
    /**
     * Display a listing of the resource.
     */
    public function index()
    {
        try {
            $users = User::all();
            return response()->json($users, 200);
        } catch (\Throwable $th) {
            return response()->json([
                'errors' => $th
            ], 400);
        }
    }
}
```

LCKM INNOVATY

JUAN CAMILO RODRIGUEZ RAMIREZ

The screenshot shows the Postman interface with the following details:

- Collection:** Bingo Fopre Unlandes
- Request Type:** GET
- URL:** http://bingofopre.test/api/users
- Headers:** Authorization, Headers (7), Body, Pre-request Script, Tests, Settings
- Query Params:** Key, Value, Description
- Body:** Status: 200 OK, Time: 1230 ms, Size: 19.07 KB, Save as Example
- JSON Response (Pretty Print):**

```
1 [ { 2   "id": 1, 3     "name": "Juan", 4     "lastname": "Developer", 5     "email": "camilo@gmail.com", 6     "two_factor_confirmed_at": null, 7     "state_id": 1, 8     "created_at": "2023-09-17T03:20:16.000000Z", 9     "updated_at": "2023-09-17T03:20:16.000000Z", 10    "avatar": "", 11    "external_id": "", 12    "external_auth": "Azure", 13    "profile_photo_url": "https://ui-avatars.com/api/?name=J&color=7F9CF5&background=EBF4FF" 14  }, 15  { 16    "id": 2, 17    "name": "Estudiante", 18    "lastname": "Unianandes", 19    "email": "estudiante@gmail.com", 20    "two_factor_confirmed_at": null, 21    "state_id": 1, 22    "created_at": "2023-09-17T03:20:16.000000Z", 23    "updated_at": "2023-09-17T03:20:16.000000Z", 24    "avatar": "", 25    "external_id": "", 26    "external_auth": "Azure", 27    "profile_photo_url": "https://ui-avatars.com/api/?name=E&color=7F9CF5&background=EBF4FF" 28  } ]
```

LCKM INNOVATY
JUAN CAMILO RODRIGUEZ RAMIREZ

Función para crear usuarios

```
/*
 * Show the form for creating a new resource.
 */

/**
 * Store a newly created resource in storage.
 */
public function store(Request $request)
{
    try {
        $user = User::create($request->all());
        return response()->json(['message' => 'El usuario se creó correctamente'], 201);
    } catch (\Throwable $th) {
        return response()->json([
            'errors' => $th
        ], 400);
    }
}
```

The screenshot shows the Postman interface with the following details:

- Request URL:** http://bingofopre.test/api/users?name=Usuario&lastname=apellido&email=usuario@gmail.com&password=\$2y\$10\$tVgd147Pjx82oggFQzc1kem...
- Method:** POST
- Params Tab:** Shows the parameters used for the creation:

Key	Value	Description
name	Usuario	Nombre del Usuario
lastname	apellido	Apellido del Usuario
email	usuario@gmail.com	Correo del Usuario
password	\$2y\$10\$tVgd147Pjx82oggFQzc1kem7G00zf.mfSe4xp...	Contraseña 1234
state_id	1	Estado del usuario

- Body Tab:** Shows the JSON response received from the server:

```
1 {"message": "El usuario se creó correctamente"}
```

Función para visualizar usuarios por ID

```
/*
 * Display the specified resource.
 */
public
function show(string $id)
{
    try {
        $user = User:: find($id);
        return response()->json($user, 200);
    } catch (\Throwable $th) {
        return response()->json([
            'errors' => $th
        ], 400);
    }
}
```

The screenshot shows a Postman collection named "Bingo Fope Unidades". The "User" section contains several requests:

- GET Listando Usuarios**
- POST Creando Usuario**
- GET Visualizando Usuario por id** (highlighted with a red arrow)
- PUT Editando Usuario por ID**
- DELETE Eliminado Usuario**
- Others: Estado, Noticias Generales, Patrocinadores, Instrucciones, Dinamicas del juego, Premios, Cartones, Grupo de cartones.

The "GET Visualizando Usuario por id" request details:

- Method: GET
- URL: http://bingofopre.test/api/users/3
- Params tab (selected): No parameters defined.
- Headers tab: Authorization, Headers (7), Body, Pre-request Script, Tests, Settings.
- Body tab: Shows a JSON response with 14 items (id, name, lastname, email, two_factor_confirmed_at, state_id, created_at, updated_at, avatar, external_id, external_auth, profile_photo_url).
- Tests tab: Status: 200 OK, Time: 504 ms, Size: 742 B, Save as Example.

Función para editar usuarios por id

```

/**
 * Show the form for editing the specified resource.
 */

/**
 * Update the specified resource in storage.
 */
public
function update(Request $request, string $id)
{
    try {
        $user = User:: find($id)->update($request->all());
        return response()->json(['message' => 'El usuario se editó correctamente'], 202);
    } catch (\Throwable $th) {
        return response()->json([
            'errors' => $th
        ], 400);
    }
}

```

The screenshot shows a Postman interface with the following details:

- Request Method:** PUT
- URL:** http://bingofopre.test/api/users/3?name=Usuario Pruebas
- Params:** A table showing query parameters:

Key	Value	Description
name	Usuario Pruebas	Nombre del Usuario
lastname	dsasd	Apellido del Usuario
email	Usuario@gmail.com	Correo del Usuario
state_id	2	Estado del Usuario
password	1234	Auth Azure
avatar		Auth Azure
external_id		Auth Azure
external_auth		Auth Azure
- Body:** A table showing body parameters:

Key	Value	Description
- Headers:** (8 items listed)
- Tests:** (None)
- Settings:** (None)
- Cookies:** (None)

At the bottom, the response is shown as:

Body	Cookies	Headers	Test Results
<pre> 1 2 "message": "El usuario se editó correctamente" 3 </pre>			Status: 202 Accepted Time: 435 ms Size: 438 B Save as Example ...

Función para eliminar de usuarios por id

```
/**  
 * Remove the specified resource from storage.  
 */  
public  
function destroy(string $id)  
{  
    try {  
        $user = User:: find($id)->delete();  
        return response()->json(['message' => 'El usuario se eliminó correctamente'], 202)  
    } catch (\Throwable $th) {  
        return response()->json([  
            'errors' => $th  
        ], 400);  
    }  
}
```

The screenshot shows the Postman interface. On the left, there's a sidebar with a tree view of API endpoints under 'Bingo Fopre Uriándes'. One endpoint, 'Eliminado Usuario', is highlighted. The main area shows a 'DELETE' request to 'http://bingofopre.test/api/users/3'. The 'Params' tab is selected. In the 'Body' section, the response is displayed as a JSON object:

```
1 "message": "El usuario se eliminó correctamente"
```

CONFIGURACIÓN DEL SITIO WEB

Ruta: /App/Http/Controllers/Api/TemplateConfigs/TemplateconfigController.php

Función para listar componentes de sitio

```
class TemplateConfigsController extends Controller
{
    /**
     * Display a listing of the resource.
     */
    public function index()
    {
        try {
            $templateconfigs = TemplateConfig:: all();
            return response()->json($templateconfigs, 200);
        } catch (\Throwable $th) {
            return response()->json([
                'errors' => $th
            ], 400);
        }
    }
}
```

The screenshot shows the Postman interface with the following details:

- Collection:** Bingo Fopre Uniandes
- Request Type:** GET
- URL:** http://bingofopre.test/api/templateconfigs
- Params:** Authorization, Headers, Body, Pre-request Script, Tests, Settings, Cookies
- Query Params:** Key, Value, Description
- Operations:**
 - GET Listado Componentes del sitio (highlighted with a red arrow)
 - POST Creando Componentes del s...
 - GET Visualizando por id compone...
 - PUT Editando Configuración del s...
 - DELETE Eliminando Componente por ...
- Sidebar:** Notion API Essential Collection

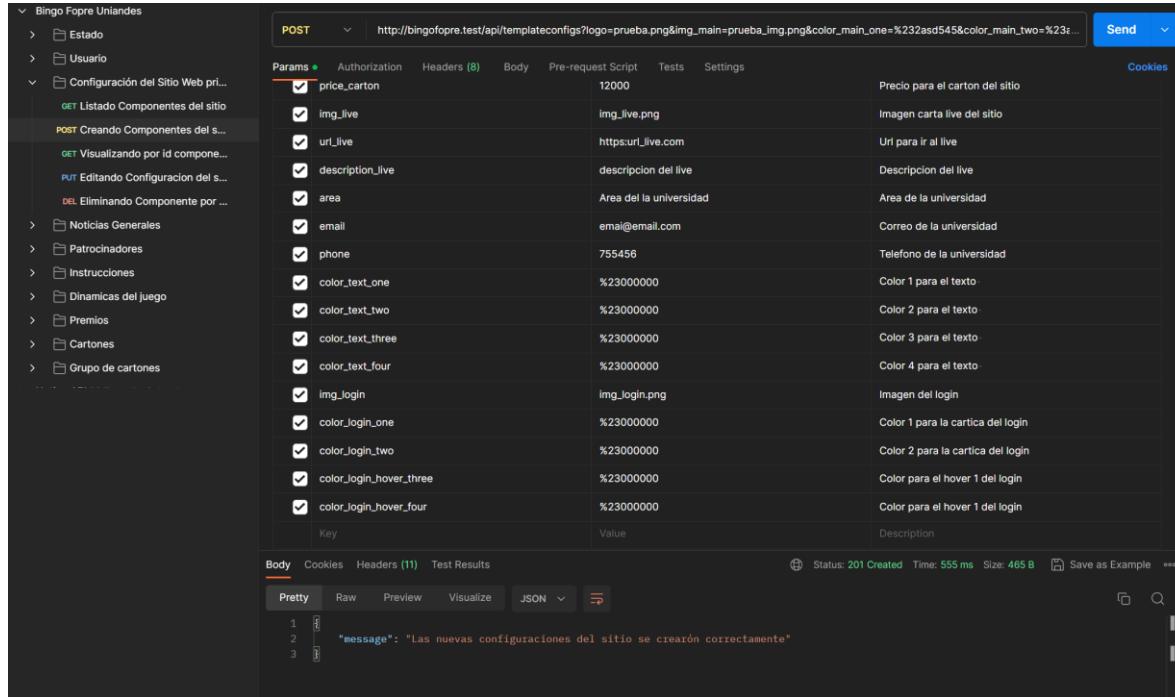
The screenshot shows the Postman interface with the following details:

- Body:** Status: 200 OK, Time: 25.89 s, Size: 1.34 KB, Save as Example
- JSON Response:**

```
1: [
2:     {
3:         "id": 1,
4:         "logo": "logo.png",
5:         "img_main": "img_main.png",
6:         "color_main_one": "#408dec1",
7:         "color_main_two": "#3c81da1",
8:         "img_main_two": "img_main_two.png",
9:         "img_login": "img_login.png",
10:        "url_carton": "https://evento.unianandes.edu.co/es/bingo-fopre-2022/Compr-de-cartones",
11:        "description_carton": "Adquiere uno en el campus con nuestros voluntarios",
12:        "price_carton": "12000",
13:        "img_live": "img_live.png",
14:        "url_live": "https://www.youtube.com/watch?v=6ff0WmMESLY&ab_channel=UniversidaddeLosAndes",
15:        "description_live": "El evento iniciara el 11 de noviembre de 2022 a las 2:00 p.m.",
16:        "area": "Dirección de Relacionamiento",
17:        "email": "bingofopre@unianandes.edu.co",
18:        "phone": "332 4090",
19:        "color_text_one": "#000000",
20:        "color_text_two": "#ffff00",
21:        "color_text_three": "#00ffff",
22:        "color_text_four": "#ffff00",
23:        "img_login": "img_login.png",
24:        "color_login_one": "#408dec1",
25:        "color_login_two": "#3c81da1",
26:        "color_login_three": "#1e4ae966",
27:        "color_login_hover_four": "#0043ff96",
28:        "created_at": "2023-09-17T03:20:22.000000Z",
29:        "updated_at": "2023-09-17T03:20:22.000000Z"
30:    }
31: ]
```

Función para crear componente del sitio

```
/**  
 * Show the form for creating a new resource.  
 */  
  
/**  
 * Store a newly created resource in storage.  
 */  
public  
function store(Request $request)  
{  
    try {  
        $templateconfig = TemplateConfig:: create($request->all());  
        return response()->json(['message'=> 'Las nuevas configuraciones del sitio se crearón correctamente'],201);  
    } catch (\Throwable $th) {  
        return response()->json([  
            'errors' => $th  
        ], 400);  
    }  
}
```



The screenshot shows a Postman interface with the following details:

- Request URL:** http://bingofopre.test/api/templateconfigs?logo=prueba.png&img_main=prueba_img.png&color_main_one=%232asd545&color_main_two=%23e...
- Method:** POST
- Params:**
 - price_carton: 12000 (Precio para el carton del sitio)
 - img_live: img_live.png (Imagen carta live del sitio)
 - url_live: https:url_live.com (Url para ir al live)
 - description_live: descripcion del live (Descripcion del live)
 - area: Area de la universidad (Area de la universidad)
 - email: email@email.com (Correo de la universidad)
 - phone: 755456 (Telefono de la universidad)
 - color_text_one: %23000000 (Color 1 para el texto)
 - color_text_two: %23000000 (Color 2 para el texto)
 - color_text_three: %23000000 (Color 3 para el texto)
 - color_text_four: %23000000 (Color 4 para el texto)
 - img_login: img_login.png (Imagen del login)
 - color_login_one: %23000000 (Color 1 para la cartica del login)
 - color_login_two: %23000000 (Color 2 para la cartica del login)
 - color_login_hover_three: %23000000 (Color para el hover 1 del login)
 - color_login_hover_four: %23000000 (Color para el hover 1 del login)
- Body:**

```
1 [ ]  
2 [ ] "message": "Las nuevas configuraciones del sitio se crearón correctamente"  
3 [ ]
```
- Headers:** Authorization, Headers (8), Body, Pre-request Script, Tests, Settings, Cookies
- Status:** Status: 201 Created Time: 555 ms Size: 465 B Save as Example

Función para visualizar id de componentes

```
/**
 * Display the specified resource.
 */
public
function show(string $id)
{
    try {
        $templateconfig = TemplateConfig:: find($id);
        return response()->json($templateconfig, 200);
    } catch (\Throwable $th) {
        return response()->json([
            'errors' => $th
        ], 400);
    }
}
```

The screenshot shows a Postman interface with the following details:

- Request URL:** GET http://bingofopre.test/api/templateconfigs/1
- Body:** JSON (Pretty)
- Response Body (JSON):**

```

1  "id": 1,
2   "logo": "logo.png",
3   "img_main": "img_main.png",
4   "color_main_one": "#498dcd01",
5   "color_main_two": "#3c61dddf",
6   "img_carton": "img_carton.png",
7   "url_carton": "https://evento.unianandes.edu.co/es/bingo-fopre-2022/Compra-de-cartones",
8   "description_carton": "o adquiere lo en el campus con nuestros voluntarios",
9   "price_carton": "12000",
10  "img_live": "img_live.png",
11  "url_live": "https://www.youtube.com/watch?v=6ff0WmMESLY&ab_channel=UniversidaddelosAndes",
12  "description_live": "El evento iniciará el 11 de noviembre de 2022 a las 2:00 p.m.",
13  "area": "Dirección de Relacionamiento",
14  "email": "bingofopre@unianandes.edu.co",
15  "phone": "332 4099",
16  "color_text_one": "#000000",
17  "color_text_two": "#ffff",
18  "color_text_three": "#110077",
19  "color_text_four": "#f0e011",
20  "img_login": "img_login.png",
21  "color_login_one": "#498dcd01",
22  "color_login_two": "#3c61dddf",
23  "color_login_three": "#3c61ddaf",
24  "color_login_hover_three": "#1e4ae966",
25  "color_login_hover_four": "#0043f496",
26  "created_at": "2023-09-17T05:07:16.000000Z",
27  "updated_at": "2023-09-17T05:07:16.000000Z"
28

```

Función para editar componentes

```

    /**
     * Show the form for editing the specified resource.
     */

    /**
     * Update the specified resource in storage.
     */
public
function update(Request $request, string $id)
{
    try {
        $templateconfig = TemplateConfig:: find($id)->update($request->all());
        return response()->json([
            'message' => 'La configuraciones del sitio se editarón correctamente'],202);
    } catch (\Throwable $th) {
        return response()->json([
            'errors' => $th
        ], 400);
    }
}

```

The screenshot shows the Postman interface with a request to `http://bingofopre.test/api/templateconfigs/2?logo=prueba.png`. The request method is `PUT`. The body contains the following JSON:

```

{
    "price_carton": 12000,
    "img_live": "img_live.png",
    "url_live": "https://url.live.com",
    "description_live": "descripcion del live",
    "area": "Area de la universidad",
    "email": "email@email.com",
    "phone": 755456,
    "color_text_one": "#23000000",
    "color_text_two": "#23000000",
    "color_text_three": "#23000000",
    "color_text_four": "#23000000",
    "img_login": "img_login.png",
    "color_login_one": "#23000000",
    "color_login_two": "#23000000",
    "color_login_hover_three": "#23000000",
    "color_login_hover_four": "#23000000"
}

```

The response status is `202 Accepted` with a message: `"message": "La configuraciones del sitio se editarón correctamente"`.

Función para eliminación de componentes

```
/**  
 * Remove the specified resource from storage.  
 */  
public  
function destroy(string $id)  
{  
    try {  
        $templateconfig = TemplateConfig:: find($id)->delete();  
        return response()->json([ 'message' => 'La configuración del sitio se eliminó correctamente'],202);  
    } catch (\Throwable $th) {  
        return response()->json([  
            'errors' => $th  
        ], 400);  
    }  
}  
}
```

The screenshot shows the Postman interface. On the left, there's a sidebar with a tree view of API endpoints for 'Bingo Föpre Unilandes'. The selected endpoint is 'DELETE http://bingofopre.test/api/templateconfigs/{id}'. The main area shows the request configuration: method 'DELETE', URL 'http://bingofopre.test/api/templateconfigs/2', and various tabs like Params, Authorization, Headers, Body, etc. Under 'Body', the 'Pretty' tab is selected, showing the JSON response:

```
1 "message": "La configuración del sitio se eliminó correctamente"
```

The status bar at the bottom indicates: Status: 202 Accepted, Time: 564 ms, Size: 461 B, Save as Example.

NOTICIAS GENERALES**Ruta:** /App/Http/Controllers/Api/CardMains/CardMainsController.php

Función para listar noticias

```
class CardMainsController extends Controller
{
    /**
     * Display a listing of the resource.
     */
    public function index()
    {
        try {
            $cardmains = CardMain::all();
            return response()->json($cardmains, 200);
        } catch (\Throwable $th) {
            return response()->json([
                'errors' => $th
            ], 400);
        }
    }
}
```

The screenshot shows the Postman interface with a successful API call. The URL is `http://bingofpre.test/api/cardmains`. The response body is a JSON array containing a single element:

```

{
    "id": 1,
    "image": "a",
    "category": "new",
    "description": "add",
    "max_info": "add",
    "state_id": 1,
    "deleted_at": null,
    "created_at": "2023-09-17T03:26:22.000000Z",
    "updated_at": "2023-09-17T03:26:22.000000Z"
}

```

Función para crear noticias

```
/*
 * Show the form for creating a new resource.
 */

/*
 * Store a newly created resource in storage.
 */
public function store(Request $request)
{
    try {
        $cardmain = CardMain::create($request->all());
        return response()->json(['message' => 'La noticia ' . $cardmain->title . ' Se creó correctamente'], 201);
    } catch (\Throwable $th) {
        return response()->json([
            'errors' => $th
        ], 400);
    }
}
```

The screenshot shows the Postman application interface. On the left, there is a sidebar with a tree view of API endpoints under 'Bingo Fopre Uniandes'. A red arrow points to the 'POST Creando Noticia General' endpoint. The main panel shows a POST request configuration. The URL is set to `http://bingofopre.test/api/cardmains?imagen&title=Prueba&description=Descripcion&mas_info=https://prueba.com&state_id=2`. The 'Params' tab is selected, displaying query parameters: 'imagen' (Imagen Principal de la noticia), 'title' (Título de la noticia), 'description' (Descripción de la noticia), 'mas_info' (Url para mas informacion), and 'state_id' (Estado para la noticia). Below this, the 'Body' tab shows a JSON response with the key 'message' and the value 'La noticia Prueba Se creó correctamente'. The status bar at the bottom indicates a 201 Created response.

Función para Visualizar noticias

```
/**
 * Display the specified resource.
 */
public
function show(string $id)
{
    try {
        $cardmain = CardMain:: find($id);
        return response()->json($cardmain, 200);
    } catch (\Throwable $th) {
        return response()->json([
            'errors' => $th
        ], 400);
    }
}
```

The screenshot shows the Postman interface with the following details:

- Request URL:** http://bingofopre.test/api/cardmains/1
- Method:** GET
- Response Status:** 200 OK
- Response Size:** 571 B
- Body (Pretty):**

```

1
2     "id": 1,
3     "imagen": "a",
4     "title": "sasad",
5     "description": "sdd",
6     "mas_info": "ddd",
7     "state_id": 1,
8     "deleted_at": null,
9     "created_at": "2023-09-17T03:20:22.000002",
10    "updated_at": "2023-09-17T03:20:22.000002"
11

```

Función para editar noticias general

```

    /**
     * Show the form for editing the specified resource.
     */

    /**
     * Update the specified resource in storage.
     */
    public
    function update(Request $request, string $id)
    {
        try {
            $cardmain = CardMain:: find($id)->update($request->all());
            return response()->json(['message' => 'La noticia se editó correctamente'], 202);
        } catch (\Throwable $th) {
            return response()->json([
                'errors' => $th
            ], 400);
        }
    }
}

```

The screenshot shows the Postman interface with a PUT request to `http://bingofopre.test/api/cardmains/1?imagen=prueba.png&title=asdas&description=asasd&mas_info=&state_id=1`. The request parameters are:

Key	Value	Description
Imagen	prueba.png	Imagen Principal de la noticia
title	asdas	Título de la noticia
description	asasd	Descripción de la noticia
mas_info		Url para mas informacion
state_id	1	Estado para la noticia

The Body tab shows the JSON response:

```

1   [
2     "message": "La noticia se editó correctamente"
3   ]

```

At the top left, the navigation tree shows the API structure under `Bingo Fopre Unlandes`.

Función para eliminar noticias general

```

/**
 * Remove the specified resource from storage.
 */
public
function destroy(string $id)
{
    try {
        $cardmain = CardMain:: find($id)->delete();
        return response()->json(['message' => 'La noticia se eliminó correctamente'], 202);
    } catch (\Throwable $th) {
        return response()->json([
            'errors' => $th
        ], 400);
    }
}

```

The screenshot shows the Postman interface with the following details:

- Request Method:** DELETE
- URL:** http://bingofopre.test/api/cardmains/1
- Params:** (None)
- Headers:** (7)
- Body:** (Pretty)
- Response:**
 - Status: 202 Accepted
 - Time: 504 ms
 - Size: 440 B
 - Content: {"message": "La noticia se eliminó correctamente"}

PATROCINADORES

Ruta: /App/Http/Controllers/Api/Sponsor/SponsorController.php

Función para listar patrocinadores

```
class SponsorsController extends Controller
{
    /**
     * Display a listing of the resource.
     */
    public function index()
    {
        try {
            $sponsors = Sponsor::all();
            return response()->json($sponsors, 200);
        } catch (\Throwable $th) {
            return response()->json([
                'errors' => $th
            ], 400);
        }
    }
}
```

The screenshot shows the Postman application interface. On the left, there is a sidebar with a tree view of API collections and endpoints. A red arrow points to the 'GET Listando patrocinadores' endpoint under the 'Patrocinadores' collection. The main panel shows the request details for this endpoint:

- Method: GET
- URL: http://bingofopre.test/api/sponsors
- Params tab (selected):
 - Key: Key
 - Value: Value
 - Description: Description
- Headers tab: Authorization, Headers (7)
- Body tab: Pre-request Script, Tests, Settings
- Tests tab: Status: 200 OK, Timer: 634 ms, Size: 542 B
- Settings tab: Cookies

Below the request details, the response body is displayed in JSON format:

```
1 [ { 2   "id": 1, 3   "logo": "prueba.png", 4   "name": "Nombre del patrocinador", 5   "state_id": 1, 6   "created_at": "2023-09-17T04:07:07.000000Z", 7   "updated_at": "2023-09-17T04:07:07.000000Z" }, 8 ] 9 }
```

Función para crear patrocinadores

```
public function store(Request $request)
{
    try {
        $sponsor = Sponsor::create($request->all());
        return response()->json(['message' => 'El patrocinador ' . $sponsor->name . ' Se creo correctamente'], 201);
    } catch (\Throwable $th) {
        return response()->json([
            'errors' => $th
        ], 400);
    }
}
```

The screenshot shows the Postman interface with the following details:

- Request Method:** POST
- URL:** http://bingofopre.test/api/sponsors?logo=prueba.png&name=Nombre del patrocinador&state_id=1
- Params:** logo: prueba.png, name: Nombre del patrocinador, state_id: 1
- Body:** (Empty)
- Headers:** (8 items)
- Tests:** (Empty)
- Settings:** (Empty)
- Query Params:** (Empty)
- Pre-request Script:** (Empty)
- Tests:** (Empty)
- Settings:** (Empty)
- Cookies:** (Empty)

On the left sidebar, under the "Bingo Fopre Uniandes" category, the "Patrocinadores" section is expanded, showing the "POST Creación del patrocinador" endpoint highlighted with a red arrow.

At the bottom, the response is shown in JSON format:

```

1: {
2:     "message": "El patrocinador Nombre del patrocinador Se creo correctamente"
3: }

```

Status: 201 Created Time: 372 ms Size: 460 B Save as Example

Función para visualizar patrocinadores por ID

```
/**
 * Display the specified resource.
 */
public
function show(string $id)
{
    try {
        $sponsor = Sponsor:: find($id);
        return response()->json($sponsor, 200);
    } catch (\Throwable $th) {
        return response()->json([
            'errors' => $th
        ], 400);
    }
}
```

The screenshot shows the Postman interface with the following details:

- Request URL:** http://bingofopre.test/api/sponsors/1
- Method:** GET
- Headers:** Authorization, Headers (7), Body, Pre-request Script, Tests, Settings, Cookies
- Params:** Query Params
- Body:** Key Value Description
- Test Results:** Status: 200 OK, Time: 463 ms, Size: 540 B, Save as Example
- Response Body (Pretty):**

```

1
2     "id": 1,
3     "logo": "prueba.png",
4     "name": "Nombre del patrocinador",
5     "state_id": 1,
6     "created_at": "2023-09-17T04:07:07.000000Z",
7     "updated_at": "2023-09-17T04:07:07.000000Z"
8

```

Función para edicitar patrocinador por id

```
/*
 * Show the form for editing the specified resource.
 */

/**
 * Update the specified resource in storage.
 */
public
function update(Request $request, string $id)
{
    try {
        $sponsor = Sponsor:: find($id)->update($request->all());
        return response()->json(['message' => 'El patrocinador se editó correctamente'], 202);
    } catch (\Throwable $th) {
        return response()->json([
            'errors' => $th
        ], 400);
    }
}
```

The screenshot shows the Postman interface with the following details:

- Request Method:** PUT
- URL:** http://bingofopre.test/api/sponsors/1?logo=img.png
- Params:** logo (selected)
- Query Params:**

Key	Value	Description
logo	img.png	Logo del patrocinador
name		Nombre del patrocinador
state_id		Estado del patrocinador [1,2,3,4,5,6]
- Body:** (Pretty) Response body:


```

1   "message": "El patrocinador se editó correctamente"
2
3
      
```
- Headers:** (11) (partially visible)
- Status:** Status: 202 Accepted Time: 493 ms Size: 443 B Save as Example

Función para eliminar patrocinador por id

```
/**  
 * Remove the specified resource from storage.  
 */  
public  
function destroy(string $id)  
{  
    try {  
        $sponsor = Sponsor:: find($id)->delete();  
        return response()->json(['message' => 'El patrocinador se eliminó correctamente'], 202);  
    } catch (\Throwable $th) {  
        return response()->json([  
            'errors' => $th  
        ], 400);  
    }  
}
```

The screenshot shows the Postman interface. On the left, there's a sidebar with a tree view of API endpoints for 'Bingo Fope Unlandes'. The 'Patrocinadores' section is expanded, showing various methods: GET Listando patrocinadores, POST Creación del patrocinador, GET Visualización por Id del patro... (partially visible), PUT Edición del patrocinador por Id, and DEL Eliminar Patrocinador. The 'DEL Eliminar Patrocinador' endpoint is selected. The main right panel shows a 'DELETE' request to the URL `http://bingofopre.test/api/sponsors/1`. The 'Params' tab is active, showing a single parameter 'Key' with value 'Value'. Below the request, the 'Body' tab is selected, showing a JSON response with the key 'message' and the value 'El patrocinador se eliminó correctamente'. The status bar at the bottom indicates a 202 Accepted status, 284 ms time, and 445 B size.

INSTRUCCIONES

Ruta: /App/Http/Controllers/Api/Instructions/InstructionController.php

Función para listar instrucciones

```
/*
 * Display a listing of the resource.
 */
public function index()
{
    try {
        $instructions = Instruction::all();
        return response()->json($instructions, 200);
    } catch (\Throwable $th) {
        return response()->json([
            'errors' => $th
        ], 400);
    }
}
```

The screenshot shows the Postman interface for testing a REST API endpoint. On the left, there's a sidebar with a tree view of API endpoints under 'Bingo Fopre Uniandes'. The 'Instrucciones' node is expanded, and its 'GET Listar Instrucciones' endpoint is selected, indicated by a red arrow. The main panel shows a 'Send' button at the top right. Below it, the URL is set to `http://bingofopre.test/api/instructions`. The 'Params' tab is active, showing a table with one row: 'Key' (empty) and 'Value' (empty). At the bottom, the 'Body' tab is selected, showing a JSON response with two objects. The JSON output is:

```
1 [ { 2   "id": 1, 3     "description_one": "adddddddd", 4     "description_two": "sasadasdasffff", 5     "created_at": "2023-09-17T03:20:22.000000Z", 6     "updated_at": "2023-09-17T03:20:22.000000Z" 7 }, 8   { 9     "id": 2, 10    "description_one": "Prueba 1", 11    "description_two": "prueba 2", 12    "created_at": "2023-09-17T04:19:19.000000Z", 13    "updated_at": "2023-09-17T04:19:19.000000Z" 14 } 15 ] 16 }
```

Función para crear instrucciones

```
/** 
 * Show the form for creating a new resource.
 */

/** 
 * Store a newly created resource in storage.
 */
public function store(Request $request)
{
    try {
        $instruction = Instruction::create($request->all());
        return response()->json(['message' => 'La instrucción se creo correctamente'], 201);
    } catch (\Throwable $th) {
        return response()->json([
            'errors' => $th
        ], 400);
    }
}
```

The screenshot shows the Postman interface with the following details:

- Request URL:** http://bingofopre.test/api/instructions?description_one=Prueba_1&description_two=prueba_2
- Method:** POST
- Params:**

Key	Value	Description
description_one	Prueba 1	Texto para la instrucción
description_two	prueba 2	Texto para cantar Bingo
- Body:**

```
Pretty Raw Preview Visualize JSON
1   "message": "La instrucción se creo correctamente"
2
3
```
- Response Headers:**
 - Status: 201 Created
 - Time: 388 ms
 - Size: 440 B
 - Save as Example

Función para visualizar instrucciones por id

```
/**
 * Display the specified resource.
 */
public
function show(string $id)
{
    try {
        $instruction = Instruction:: find($id);
        return response()->json($instruction, 200);
    } catch (\Throwable $th) {
        return response()->json([
            'errors' => $th
        ], 400);
    }
}
```

The screenshot shows the Postman interface with the following details:

- Request URL:** http://bingofopre.test/api/instructions/2
- Method:** GET
- Headers:** Authorization, Headers (7), Body, Pre-request Script, Tests, Settings
- Query Params:** Key, Value, Description
- Body:** Status: 200 OK, Time: 394 ms, Size: 532 B, Save as Example
- Response Body (Pretty):**

```

1  {
2      "id": 2,
3      "description_one": "Prueba 1",
4      "description_two": "prueba 2",
5      "created_at": "2023-09-17T04:19:19.000000Z",
6      "updated_at": "2023-09-17T04:19:19.000000Z"
7  }
```

A red arrow points to the "GET Visualización por ID de la ins..." item in the left sidebar under the "Instrucciones" section.

Función para editar instrucciones por ID

```
/*
 * Show the form for editing the specified resource.
 */

/**
 * Update the specified resource in storage.
 */
public
function update(Request $request, string $id)
{
    try {
        $instruction = Instruction:: find($id)->update($request->all());
        return response()->json(['message' => 'La instrucción se edito correctamente'], 202);
    } catch (\Throwable $th) {
        return response()->json([
            'errors' => $th
        ], 400);
    }
}
```

The screenshot shows the Postman application interface. On the left, there is a sidebar navigation tree for 'Bingo Fopre Unlandes' with categories like Estado, Usuario, Configuración del Sitio Web, Noticias Generales, Patrocinadores, Instrucciones (with sub-options: Listar Instrucciones, Crear Instrucción, Visualización por ID), and Edición de la instrucción por id. The 'Instrucciones' section is currently selected.

The main right panel displays a 'PUT' request configuration. The URL is set to `http://bingofopre.test/api/instructions/2?description_one=prueba 36`. The 'Params' tab is active, showing a table with one row:

Key	Value	Description	... Bulk Edit
<input checked="" type="checkbox"/> description_one	prueba 36	Texto para la instrucción	

Below the table, there are sections for 'Cookies', 'Headers (11)', 'Body', 'Pre-request Script', 'Tests', and 'Settings'. At the bottom, the status bar shows 'Status: 202 Accepted Time: 521 ms Size: 442 B Save as Example'.

The 'Body' tab is expanded, showing a JSON response with three lines of code:

```
1 "message": "La instrucción se edito correctamente"
```

Función para eliminar instrucción por ID

```

public
function destroy(string $id)
{
    try {
        $instruction = Instruction:: find($id)->delete();
        return response()->json(['message' => 'La instrucción se eliminó correctamente'], 202);
    } catch (\Throwable $th) {
        return response()->json([
            'errors' => $th
        ], 400);
    }
}

```

The screenshot shows the Postman interface with the following details:

- Request Method:** DELETE
- URL:** http://bingofopre.test/api/instructions/2
- Params:** Authorization, Headers (7), Body, Pre-request Script, Tests, Settings
- Query Params:** Key, Value, Description, Bulk Edit
- Body:** Status: 202 Accepted, Time: 481 ms, Size: 449 B, Save as Example
- JSON Response:**

```

1 {"message": "La instrucción se eliminó correctamente"}
2
3

```

A red arrow points to the "DELETE Eliminar la Instrucción" option in the API documentation sidebar.

DINÁMICAS DEL JUEGO

Ruta: /App/Http/Controllers/Api/DynamicGames/DynamicGameController.php

Función para listar las dinámicas del juego

```
/*
 * Display a listing of the resource.
 */
public function index()
{
    try {
        $dynamicgames = DynamicGame::all();
        return response()->json($dynamicgames, 200);
    } catch (\Throwable $th) {
        return response()->json([
            'errors' => $th
        ], 400);
    }
}
```

The screenshot shows the Postman interface. On the left, there's a sidebar with a tree view of API endpoints under 'Bingo Fopre Unlandes'. A red arrow points to the 'GET Listando las dinamicas del j...'. The main panel shows a 'Send' button at the top right. Below it, the URL is set to 'http://bingofopre.test/api/dynamicgames/1'. The 'Params' tab is selected, showing a table with one row: 'Key' (empty) and 'Value' (empty). The 'Body' tab is selected, showing a JSON response with 12 items. The first item is:

```
1  "id": 1,
2  "logo": "logo.png",
3  "title": "Prueba",
4  "letra": "a",
5  "fila": "1",
6  "column": "2",
7  "state_id": 1,
8  "deleted_at": null,
9  "created_at": "2023-09-17T04:27:24.000000Z",
10  "updated_at": "2023-09-17T04:27:24.000000Z"
```

Función para crear dinámicas del juego

```

/**
 * Show the form for creating a new resource.
 */

/**
 * Store a newly created resource in storage.
 */
public function store(Request $request)
{
    try {
        $dynamicgame = DynamicGame:: create($request->all());
        return response()->json(['message' => 'La dinámica ' . $dynamicgame->title . ' Se creó correctamente'], 201);
    } catch (\Throwable $th) {
        return response()->json([
            'errors' => $th
        ], 400);
    }
}

```

The screenshot shows the Postman interface with the following details:

- Request Method:** POST
- URL:** http://bingofopre.test/api/dynamicgames?logo=logo.png&title=Prueba&letra=a&fila=1&colum=2&state_id=1
- Params:**

Key	Value	Description
logo	logo.png	Logo para las dinámicas
title	Prueba	Título para la dinámica
letra	a	Letra para la dinámica
fila	1	Fila para la dinámica
colum	2	Columnas para la dinámica
state_id	1	Estado para la dinámica
- Body:** (Pretty) JSON response:


```

1   "message": "La dinámica Prueba Se creó correctamente"
2
3
      
```
- Headers (11):** (List not shown)
- Test Results:** Status: 201 Created, Time: 404 ms, Size: 449 B

Función para Visualizar dinámicas del juego por ID

```
/** 
 * Display the specified resource.
 */
public
function show(string $id)
{
    try {
        $dynamicgame = DynamicGame:: find($id);
        return response()->json($dynamicgame, 200);
    } catch (\Throwable $th) {
        return response()->json([
            'errors' => $th
        ]);
    }
}
```

The screenshot shows the Postman interface with the 'Bingo Fope Unlandes' collection selected. Under the 'Dinamicas del juego' folder, a red arrow points to the 'GET Visualizando dinamicas por id' request. The request details are as follows:

- Method: GET
- URL: http://bingofope.test/api/dynamicgames
- Params tab is selected.
- Query Params table:

Key	Value	Description	...	Bulk Edit
Key	Value	Description	...	
- Authorization, Headers, Body, Pre-request Script, Tests, and Settings tabs are visible at the top.

The screenshot shows the Postman interface displaying the JSON response for two dynamic games. The response is as follows:

```

1 [
2     {
3         "id": 1,
4         "logo": "logo.png",
5         "title": "trueba",
6         "letra": "a",
7         "fila": "1",
8         "column": "2",
9         "state_id": 1,
10        "deleted_at": null,
11        "created_at": "2023-09-17T04:27:24.000000Z",
12        "updated_at": "2023-09-17T04:27:24.000000Z"
13    },
14    {
15        "id": 2,
16        "logo": "logo.png",
17        "title": "trueba",
18        "letra": "a",
19        "fila": "1",
20        "column": "2",
21        "state_id": 1,
22        "deleted_at": null,
23        "created_at": "2023-09-17T04:28:39.000000Z",
24        "updated_at": "2023-09-17T04:28:39.000000Z"
25    }
]

```

The 'Body' tab is selected, showing the raw JSON. The 'Pretty' tab is also visible. The status bar at the top right indicates: Status: 200 OK Time: 490 ms Size: 773 B Save as Example.

Función para editar dinámica del juego por ID

```
/*
 * Show the form for editing the specified resource.
 */

/**
 * Update the specified resource in storage.
 */
public
function update(Request $request, string $id)
{
    try {
        $dynamicgame = DynamicGame:: find($id)->update($request->all());
        return response()->json(['message' => 'La dinámica se editó correctamente'], 202);
    } catch (\Throwable $th) {
        return response()->json([
            'errors' => $th
        ], 400);
    }
}
```

The screenshot shows the Postman interface with the following details:

- Request Method:** PUT
- URL:** http://bingofopre.test/api/dynamicgames/1?logo=123.png
- Params:** logo (checked, value: 123.png)
- Body:** (Pretty) JSON response:


```
1:   "message": "La dinámica se editó correctamente"
```
- Headers (11):** Status: 202 Accepted, Time: 482 ms, Size: 444 B, Save as Example, etc.

Función para eliminar dinámica de juego por id.

```
/**
 * Remove the specified resource from storage.
 */
public
function destroy(string $id)
{
    try {
        $dynamicgame = DynamicGame::: find($id)->delete();
        return response()->json(['message' => 'La dinámica se eliminó correctamente'], 202);
    } catch (\Throwable $th) {
        return response()->json([
            'errors' => $th
        ], 400);
    }
}
```

The screenshot shows the Postman interface with the following details:

- Request Method:** DELETE
- URL:** http://bingofopre.test/api/dynamicgames/2
- Params:** Key: Key, Value: Value, Description: Description
- Body:** Status: 202 Accepted, Time: 527 ms, Size: 446 B, Save as Example
- JSON Response:**

```

1: [
2:     "message": "La dinámica se eliminó correctamente"
3: ]

```

A red arrow points to the "DEL Eliminación de la dinámica" entry in the left sidebar under the "Dinamicas del juego" category.

PREMIOS

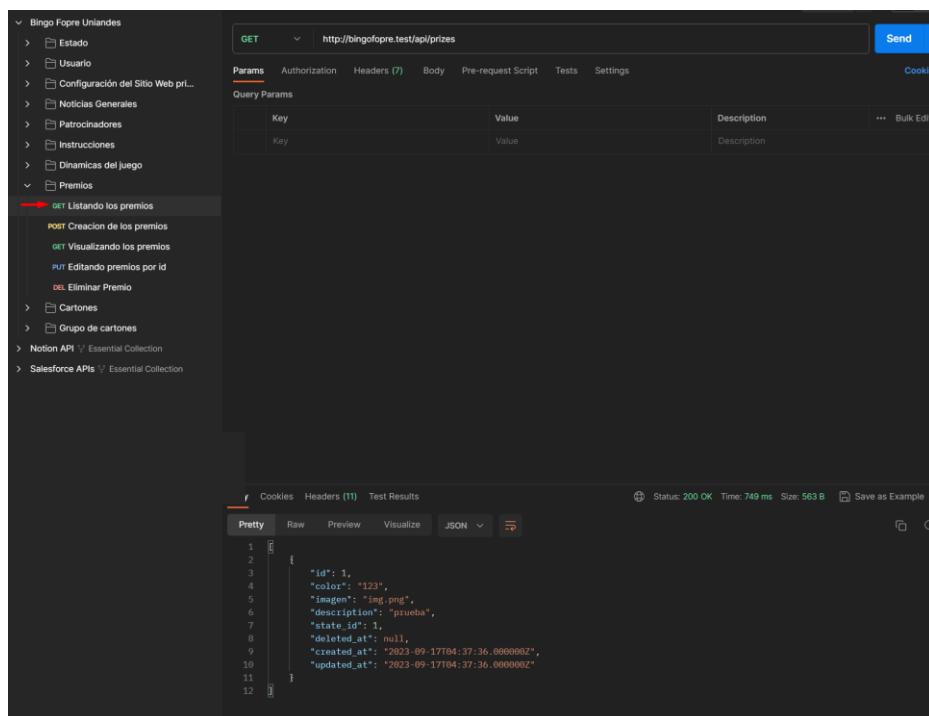
Ruta: /App/Http/Controllers/Api/Prizes/PrizeController.php

Función para listar de premios

```

    /**
     * Display a listing of the resource.
     */
    public function index()
    {
        try {
            $prizes = Prize::all();
            return response()->json($prizes, 200);
        } catch (\Throwable $th) {
            return response()->json([
                'errors' => $th
            ], 400);
        }
    }
}

```



The screenshot shows the Postman interface with the following details:

- Collection:** Bingo Fope Unlandes
- Request Type:** GET
- URL:** http://bingofope.test/api/prizes
- Headers:** Authorization, Headers, Body, Pre-request Script, Tests, Settings
- Query Params:** Key, Value, Description, Bulk Edit
- Body:** None
- Tests:** Status: 200 OK, Time: 749 ms, Size: 563 B, Save as Example
- Preview:** Shows a JSON response with the following data:

```

1  [
2   {
3     "id": 1,
4     "color": "123",
5     "imagen": "img.png",
6     "description": "prueba",
7     "state_id": 1,
8     "deleted_at": null,
9     "created_at": "2023-09-17T04:37:36.000000Z",
10    "updated_at": "2023-09-17T04:37:36.000000Z"
11  ]
12

```

Función para crear premios

```

/**
 * Show the form for creating a new resource.
 */

/**
 * Store a newly created resource in storage.
 */
public function store(Request $request)
{
    try {
        $prize = Prize::create($request->all());
        return response()->json(['message' => 'El premio se creó correctamente'], 201);
    } catch (\Throwable $th) {
        return response()->json([
            'errors' => $th
        ], 400);
    }
}

```

The screenshot shows the Postman interface with the following details:

- Request Method:** POST
- URL:** http://bingofopre.test/api/prizes?color=123&imagen=img.png&description=prueba&state_id=1
- Params Tab:** color: 123, imagen: img.png, description: prueba, state_id: 1
- Body Tab:** (Empty)
- Headers Tab:** (Empty)
- Tests Tab:** (Empty)
- Settings Tab:** (Empty)
- Body Response:**

```

1   "message": "El premio se creó correctamente"
2
3

```
- Status:** 201 Created
- Time:** 561 ms
- Size:** 435 B
- Save as Example:** icon

Función para visualizar premios por ID

```
/**
 * Display the specified resource.
 */
public
function show(string $id)
{
    try {
        $prize = Prize:: find($id);
        return response()->json($prize, 200);
    } catch (\Throwable $th) {
        return response()->json([
            'errors' => $th
        ], 400);
    }
}
```

The screenshot shows the Postman interface with the following details:

- API Structure:** On the left, under "Bingo Fope Unlandes", there is a tree view of endpoints:
 - Estado
 - Usuario
 - Configuración del Sitio Web
 - Noticias Generales
 - Patrocinadores
 - Instrucciones
 - Dinámicas del juego
 - Premios
 - GET** Listando los premios
 - POST** Creacion de los premios
 - GET** Visualizando los premios (highlighted with a red arrow)
 - PUT** Editando premios por id
 - DEL** Eliminar Premio
 - Cartones
 - Grupo de cartones
- Request Details:** In the center, a GET request is defined for the URL `http://bingofope.test/api/prizes/1`. The "Params" tab is selected, showing a single parameter "Key" with value "Value".
- Response Preview:** At the bottom, the response is shown in JSON format:


```
1  "id": 1,
2   "color": "123",
3   "imagen": "img.png",
4   "description": "prueba",
5   "state_id": 1,
6   "deleted_at": null,
7   "created_at": "2023-09-17T04:37:36.000000Z",
8   "updated_at": "2023-09-17T04:37:36.000000Z"
```

Función para editar premios

```

    /**
     * Show the form for editing the specified resource.
     */

    /**
     * Update the specified resource in storage.
     */
    public
    function update(Request $request, string $id)
    {
        try {
            $prize = Prize:: find($id)->update($request->all());
            return response()->json(['message' => 'El premio se Edito correctamente'], 202);
        } catch (\Throwable $th) {
            return response()->json([
                'errors' => $th
            ], 400);
        }
    }
}

```

The screenshot shows the Postman interface with the following details:

- Request Method:** PUT
- URL:** http://bingofopre.test/api/prizes/1?color=456
- Params:** color: 456
- Query Params:** None
- Body:** JSON (Pretty) - Response body:

```

1   "message": "El premio se Edito correctamente"
2
3

```
- Status:** 202 Accepted
- Time:** 576 ms
- Size:** 432 B
- Headers:** (11)

The left sidebar shows the API structure under the 'Bingo Fope Uniandes' collection, with a red arrow pointing to the 'PUT Editando premios por id' endpoint.

Función para eliminar de premio por id

```
/**
 * Remove the specified resource from storage.
 */
public
function destroy(string $id)
{
    try {
        $prize = Prize:: find($id)->delete();
        return response()->json(['message' => 'El premio se elimino Correctamente'], 202);
    } catch (\Throwable $th) {
        return response()->json([
            'errors' => $th
        ], 400);
    }
}
```

The screenshot shows the Postman interface with a successful API call. The left sidebar shows a tree structure of API endpoints under 'Bingo Fopre Uriandres'. The 'Premios' endpoint is expanded, and the 'DELETE' method under it is highlighted with a red arrow. The main panel shows a DELETE request to 'http://bingofopre.test/api/prizes/1'. The 'Params' tab is selected, showing a table with one row: 'Key' (empty), 'Value' (empty), 'Description' (empty), and 'Bulk Edit' (checkbox). Below the table, the 'Body' tab shows a JSON response:


```
1 "message": "El premio se elimino Correctamente"
```

 The status bar at the bottom indicates: Status: 202 Accepted Time: 567 ms Size: 434 B Save as Example ...

CARTONES**Ruta:** /App/Http/Controllers/Api/Cartons/CartonController.php

Función para listar todos los cartones

```
/**
 * Display a listing of the resource.
 */
public function index()
{
    try {
        $cardboards = Cardboard::all();
        return response()->json($cardboards, 200);
    } catch (\Throwable $th) {
        return response()->json([
            'errors' => $th
        ], 400);
    }
}
```

id	name	date_finish	price	document_number	state_id	group_id	created_at	updated_at
1	"001"	"2023-09-16"	"12000"	null	3	1	"2023-09-17T05:07:32.000000Z"	"2023-09-17T05:07:32.000000Z"
2	"002"	"2023-09-16"	"12000"	null	3	1	"2023-09-17T05:07:32.000000Z"	"2023-09-17T05:07:32.000000Z"
3	"003"	"2023-09-16"	"12000"	null	3	1	"2023-09-17T05:07:32.000000Z"	"2023-09-17T05:07:32.000000Z"
4	"004"	"2023-09-16"	"12000"	null	3	1	"2023-09-17T05:07:32.000000Z"	"2023-09-17T05:07:32.000000Z"
5	"005"	"2023-09-16"	"12000"	null	3	2	"2023-09-17T05:07:32.000000Z"	"2023-09-17T05:07:32.000000Z"

Función para crear cartones masivamente

```
public function create(Request $request)
{
    try {
        $startNumber = strval($request->input('start_number')); // Convierte a cadena
        $endNumber = strval($request->input('end_number')); // Convierte a cadena
        $groupSize = $request->input('group_size');
        $date = $request->input('date');
        $price = $request->input('price');

        // $user_id = auth()->user()->id; // Obtener el ID del usuario autenticado

        $group = null;
        $groupCount = 0;

        for ($i = $startNumber; $i <= $endNumber; $i++) {
            if ($groupCount % $groupSize === 0) {
                $group = CartonGroup::create(['user_id' => null]);
            }

            // Formatea el nombre del cartón con ceros a la izquierda
            $formattedName = str_pad($i, strlen($endNumber), '0', STR_PAD_LEFT);

            $cardboard = Cardboard::create([
                'name' => $formattedName,
                'date_finish' => $date,
                'price' => $price,
                'state_id' => 3, // Reemplaza con el estado correcto
                'group_id' => $group ? $group->id : null,
            ]);

            $groupCount++;
            // Asigna el group_id al cartón
            $cardboard->group_id = $group ? $group->id : null;
            $cardboard->save();
        }
        return response()->json(['message' => 'Los cartones se crearon correctamente'], 202);
    } catch (ValidationException $e) {
        // Manejo de excepciones de validación
        return response()->json(['error' => 'Error de validación'], 400);
    } catch (QueryException $e) {
        // Manejo de excepciones de consulta de base de datos
        return response()->json(['error' => 'Error de base de datos'], 500);
    } catch (Exception $e) {
        // Manejo de otras excepciones no previstas
        return response()->json(['error' => 'Error interno del servidor'], 500);
    }
}
```

LCKM INNOVATY

JUAN CAMILO RODRIGUEZ RAMIREZ

The screenshot shows the Postman interface with a successful API call to create multiple bingo cards.

Request Details:

- Method:** POST
- URL:** http://bingofopre.test/api/admin/cartones/create?start_number=001&end_number=010&group_size=4&date=2023-09-16&price=12000
- Params:** start_number: 001, end_number: 010, group_size: 4, date: 2023-09-16, price: 12000

Body (Pretty JSON):

```
1:   "message": "Los cartones se crearon correctamente"
```

Status: 202 Accepted

Función para visualizar cartones por ID

```
/**
 * Display the specified resource.
 */
public
function show(string $id)
{
    try {
        $cardboard = Cardboard:: find($id);
        return response()->json($cardboard, 200);
    } catch (\Throwable $th) {
        return response()->json([
            'errors' => $th
        ], 400);
    }
}
```

Key	Value	Description
Key	Value	Description

```

1
2     "id": 1,
3     "name": "001",
4     "date_finish": "2023-09-16",
5     "price": "12000",
6     "document_number": null,
7     "state_id": 3,
8     "group_id": 1,
9     "created_at": "2023-09-17T05:07:32.000000Z",
10    "updated_at": "2023-09-17T05:07:32.000000Z"
11

```

Función para añadir un cartón al carrito

```

public function addToCart($name)
{
    try {
        // Buscar el cartón por el nombre en lugar del ID
        $carton = Cardboard::where('name', $name)->first();

        // Verificar si se encontró el cartón
        if (!$carton) {
            return response()->json(['error' => 'Ocurrió un error inesperado'], 500);
        }

        $cart = session()->get('cart');

        $cart[$carton->id] = [
            'name' => $carton->name,
            'quantity' => 1,
            'date_finish' => $carton->date_finish,
            'price' => $carton->price,
            'state_id' => $carton->state_id,
            'user_id' => $carton->user_id,
            'document_number' => $carton->document_number,
        ];

        session()->put('cart', $cart);
        return response()->json(['message' => 'El cartón se ha añadido al carrito'], 202);

    } catch (Exception $e) {
        // Manejo de excepciones
        return response()->json(['error' => 'Ocurrió un error inesperado'], 500);
    }
}

```

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'Collections' and 'Environments'. Under 'Collections', there's a tree view for 'Bingo Fopre Unlandes' with several sub-endpoints listed under 'Cartones'. A red arrow points to the 'POST Añadiendo cartones al carrito de compra...' endpoint. The main workspace shows a POST request to 'http://bingofopre.test/api/admin/cartones/addToCart/001'. The 'Params' tab has one entry: 'Key' (empty) and 'Value' (empty). The 'Body' tab shows a JSON response with a single key 'message' and its value is 'El cartón se ha añadido al carrito'. The status bar at the bottom indicates a 202 Accepted status.

Función para finalizar compra del cartón

```
public function finishPurchase(Request $request)
{
    try {
        // Obtener el carrito desde la sesión
        $cart = Session::get('cart', []);

        // Obtener datos de estado y usuario de cada cartón desde el formulario
        $cartonData = $request->input('cartons');
        $documentoComprador = $request->input('document_number');

        // Iterar a través de los elementos del carrito y actualizar la base de datos
        foreach ($cart as $cartonId => $carton) {
            // Verificar si el cartón existe en la base de datos
            $cartonDB = Cardboard::find($cartonId);

            if ($cartonDB) {
                // Actualiza el estado y el campo user_id del cartón en la base de datos
                $cartonDB->state_id = $cartonData[$cartonId]['state_id'];
                $cartonDB->document_number = $documentoComprador;
                $cartonDB->save();
            }
        }

        // Limpia el carrito en la sesión
        Session::forget('cart');

        return response()->json(['message' => 'El cartón se ha vendido correctamente'], 202);
    } catch (ValidationException $e) {
        // Manejo de excepciones de validación
        return response()->json(['error' => 'Error de validación'], 400);
    } catch (QueryException $e) {
        // Manejo de excepciones de consulta de base de datos
        return response()->json(['error' => 'Error de base de datos'], 500);
    } catch (Exception $e) {
        // Manejo de otras excepciones no previstas
        return response()->json(['error' => 'Ocurrió un error inesperado'], 500);
    }
}
```

LCKM INNOVATY

JUAN CAMILO RODRIGUEZ RAMIREZ

The screenshot shows the Postman interface with the following details:

- Request Method:** POST
- URL:** http://bingofopre.test/api/admin/cartones/finishPurchase
- Params:** Authorization, Headers (8), Body, Pre-request Script, Tests, Settings
- Query Params:** Key, Value, Description, Bulk Edit
- Body:** Headers (11), Test Results, Pretty, Raw, Preview, Visualize, JSON
- Response Status:** Status: 202 Accepted, Time: 482 ms, Size: 442 B, Save as Example
- Response Body:** "message": "El cartón se ha vendido correctamente"

Función para eliminar un cartón en el carrito

```
public function removeFromCart($cartonId)
{
    try {
        // Obtener el carrito desde la sesión
        $cart = Session::get('cart', []);

        // Verificar si el cartón existe en el carrito
        if (isset($cart[$cartonId])) {
            // Elimina el cartón del carrito
            unset($cart[$cartonId]);

            // Actualiza el carrito en la sesión
            Session::put('cart', $cart);

            return redirect()->route('user.cart.index')->with('success', 'Cartón eliminado del carrito con éxito.');
        }

        return response()->json(['message' => 'Cartón eliminado del carrito con éxito'], 202);
    } catch (Exception $e) {
        // Manejo de excepciones
        return response()->json(['error' => 'Ocurrió un error inesperado.'], 400);
    }
}
```

Bingo Fopre Unlandes / Cartones / Eliminar cartón del carrito

POST http://bingofopre.test/api/admin/cartones/removeFromCart/1

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

Key	Value	Description	...	Bulk Edit
Key	Value	Description		

Body Cookies Headers (11) Test Results

Pretty Raw Preview Visualize JSON

```

1   {
2     "message": "Cartón eliminado del carrito con éxito"
3   }

```

Status: 202 Accepted Time: 496 ms Size: 448 B Save as Example

Función para editar cartón por ID

```

    /**
     * Show the form for editing the specified resource.
     */

    /**
     * Update the specified resource in storage.
     */
public
function update(Request $request, string $id)
{
    try {
        $cardboard = Cardboard::find($id)->update($request->all());
        return response()->json(['message' => 'El cartón se ha editado correctamente'], 202);
    } catch (\Throwable $th) {
        return response()->json([
            'errors' => $th
        ], 400);
    }
}

```

The screenshot shows the Postman interface with a dark theme. On the left, there's a sidebar with a tree view of API endpoints under 'Bingo Fopre Unlandes'. The selected endpoint is 'PUT Editando cartón po ID'. The main area shows a 'PUT' request to the URL `http://bingofopre.test/api/cardboards/1?document_number=1222&state_id=5`. The 'Params' tab is active, displaying the following query parameters:

Key	Value	Description
<input type="checkbox"/> name		Nombre Carton.
<input type="checkbox"/> date_finish		
<input type="checkbox"/> price		
<input checked="" type="checkbox"/> document_number	1222	
<input checked="" type="checkbox"/> state_id	5	
<input type="checkbox"/> group_id		
Key	Value	Description

At the bottom, the response section shows the status: 202 Accepted, Time: 571 ms, Size: 442 B. The body of the response is displayed in JSON format:

```

1   "message": "El cartón se ha editado correctamente"
2
3

```

Función para eliminar carton por id

```
/**
 * Remove the specified resource from storage.
 */
public
function destroy(string $id)
{
    try {
        $cardboard = Cardboard:: find($id)->delete();
        return response()->json(['message' => 'El cartón se a eliminado con éxito'], 202);
    } catch (\Throwable $th) {
        return response()->json([
            'errors' => $th
        ], 400);
    }
}
```

The screenshot shows the Postman interface. On the left, there's a sidebar with a tree view of API endpoints under 'Bingo Fopre Uniandes'. The 'Cartones' section is expanded, showing various methods including 'DELETE Eliminando carton por Id'. The main panel shows a 'DELETE' request to 'http://bingofopre.test/api/cardboards/1'. The 'Params' tab is selected. In the 'Body' tab, the response is displayed as JSON:

```
1   "message": "El cartón se a eliminado con éxito"
```

At the bottom, the status bar indicates: Status: 202 Accepted, Time: 400 ms, Size: 444 B.

GRUPO DE CARTONES**Ruta:** /App/Http/Controllers/Api/CartonGroups/CartonGroupsController.php

Función para listar grupo de cartones

```

/**
 * Display a listing of the resource.
 */
public function index()
{
    try {
        $cartonGroups = CartonGroup::all();
        return response()->json($cartonGroups, 200);
    } catch (\Throwable $th) {
        return response()->json([
            'errors' => $th
        ], 400);
    }
}

```

The screenshot shows the Postman interface with the following details:

- Request URL:** GET http://bingofopre.test/api/cartongroups
- Body:** Pretty (JSON response shown below)
- Response Headers:**
 - Status: 200 OK
 - Time: 17.39 s
 - Size: 750 B
 - Save as Example
- Response Body (Pretty Print):**

```

1 [
2   {
3     "id": 1,
4     "user_id": null,
5     "state_id": 3,
6     "created_at": "2023-09-17T05:07:32.000000Z",
7     "updated_at": "2023-09-17T05:07:32.000000Z"
8   },
9   {
10    "id": 2,
11    "user_id": null,
12    "state_id": 3,
13    "created_at": "2023-09-17T05:07:32.000000Z",
14    "updated_at": "2023-09-17T05:07:32.000000Z"
15  },
16  {
17    "id": 3,
18    "user_id": null,
19    "state_id": 3,
20    "created_at": "2023-09-17T05:07:32.000000Z",
21    "updated_at": "2023-09-17T05:07:32.000000Z"
22  }
]

```

Función para crear un grupo carton

```
/**
 * Show the form for creating a new resource.
 */

/**
 * Store a newly created resource in storage.
 */
public function store(Request $request)
{
    try {
        $cartonGroup = CartonGroup::create($request->all());
        return response()->json(['message' => 'El grupo se creó correctamente'],
    } catch (\Throwable $th) {
        return response()->json([
            'errors' => $th
        ], 400);
    }
}
```

The screenshot shows the Postman interface with the following details:

- Request Method:** POST
- URL:** http://bingofopre.test/api/cartongroups?user_id=1&state_id=2
- Params:**

Key	Value	Description
user_id	1	
state_id	2	
- Body:** (Empty)
- Test Results:** Status: 201 Created, Time: 803 ms, Size: 434 B
- Response Body:**

```
1
2   "message": "El grupo se creó correctamente"
3
```

Función para visualizar un grupo de carton por id

```
/**
 * Display the specified resource.
 */
public
function show(string $id)
{
    try {
        $cartonGroup = CartonGroup:: find($id);
        return response()->json($cartonGroup, 200);
    } catch (\Throwable $th) {
        return response()->json([
            'errors' => $th
        ], 400);
    }
}
```

The screenshot shows the Postman interface with the following details:

- Collection:** Bingo Fope Unlandes
- Request Type:** GET
- URL:** http://bingofopre.test/api/cartongroups/2
- Params:** Key: Key, Value: Value
- Body:** (Pretty) JSON response:

```

1: {
2:     "id": 2,
3:     "user_id": null,
4:     "state_id": 3,
5:     "created_at": "2023-09-17T05:07:32.000000Z",
6:     "updated_at": "2023-09-17T05:07:32.000000Z"
7: }
```
- Status:** 200 OK
- Time:** 569 ms
- Size:** 502 B

Función para editar grupo de cartones

```
/** 
 * Show the form for editing the specified resource.
 */

/** 
 * Update the specified resource in storage.
 */
public
function update(Request $request, string $id)
{
    try {
        $cartonGroup = CartonGroup:: find($id)->update($request->all());
        return response()->json(['message' => 'El grupo se editó correctamente']);
    } catch (\Throwable $th) {
        return response()->json([
            'errors' => $th
        ], 400);
    }
}
```

The screenshot shows the Postman interface with the following details:

- Request Method:** PUT
- URL:** http://bingofopre.test/api/cartongroups/5?user_id=2&state_id=1
- Params:** user_id (2), state_id (1)
- Body:** JSON response:


```
1 "message": "El grupo se editó correctamente"
```
- Status:** 202 Accepted
- Time:** 481 ms
- Size:** 436 B

Función para eliminar grupo de cartones

```

    /**
     * Remove the specified resource from storage.
     */
    public
    function destroy(string $id)
    {
        try {
            $cartonGroup = CartonGroup:: find($id)->delete();
            return response()->json(['message' => 'El grupo se eliminó correctamente'], 202);
        } catch (\Throwable $th) {
            return response()->json([
                'errors' => $th
            ], 400);
        }
    }
}

```

The screenshot shows the Postman interface with the following details:

- Request URL:** http://bingofopre.test/api/cartongroups/6
- Method:** DELETE
- Response Body (Pretty JSON):**

```

1   "message": "El grupo se eliminó correctamente"
2
3

```
- Response Headers:**
 - Status: 202 Accepted
 - Time: 481 ms
 - Size: 438 B