Las distintas APIS fueron creadas con el objetivo de **crear , recuperar, modificar y eliminar** dependiendo el recurso que el cliente necesite.

Esto funciona de la siguiente manera:

Al tener un backend modularizado con el patrón de MVC y la arquitectura API REST , el **Cliente / User** va enviar un request dependiendo el tipo de recurso que necesite. Ese request es enviado en formato JSON, y el encargado de agarrar ese JSON es el **controlador del modelo o clase.**

Como se mencionó anteriormente , dependiendo del tipo de recurso que el cliente necesite (CRUD) , el controlador a través del ORM se encarga de realizar la interacción con la base de datos que en nuestro caso vendría hacer la carpeta repositorio.

Cuya carpeta repositorio se encuentran los métodos (CRUD) para cada clase o modelo necesario. Derivamos la funcionalidad a la base de datos que nos va devolver algún resultado (error, ok) para enviarlo al controlador y el mismo a la vista.

**A Continuación dejamos la documentación de una API:**

**API SOCIO / URL BASE: localhost:8080/api/v1/socio**

Esta API tiene como objetivo 5 funcionalidades:

1. **Crear un socio.**

API URL: localhost:8080/api/v1/socio

Método: POST.

Parámetros: Datos de la persona.

Response:

* 200: Ok..
* 400 - 500: Mensaje de error.

1. **Recuperar un socio.**

API URL: localhost:8080/api/v1/socio/**idSocio**

Método: GET

Parámetros: ID Socio.

Response:111

* 200: Ok.
* 400 - 500: Mensaje de error.

1. **Recuperar los socios.**

API URL: localhost:8080/api/v1/socio

Método: Get

Parámetros: Datos de la persona.

Response:

* 200: Ok.
* 400 - 500: Mensaje de error.

1. **Modificar un socio.**

API URL: localhost:8080/api/v1/socio/**idSocio**

Método: Put

Parámetros: ID Socio y Datos de la persona.

Response:

1. 200: Ok.
2. 400 - 500: Mensaje de error.
3. **Eliminar un socio.**

API URL: localhost:8080/api/v1/socio/**idSocio**

Método: Delete

Parámetros: ID Socio

Response:

1. 200: Ok
2. 400 - 500: Mensaje de error.

**Validaciones:**

* Se agregó validación de que un socio no tenga número de doc y tipo iguales.
* Se agregó validación de tipo y número de doc duplicados.
* Se agregaron validaciones para el alta de película que haya un género creado y que no se pueda agregar una película con el mismo nombre.