1. **INSTALACIÓN DE DEPENDENCIAS**

Primero que todo hay que crear la base de datos Postgres local que se va a utilizar. Para este fin es necesario descargar e instalar la versión más reciente de PostgreSQL, la se puede descargar desde el siguiente enlace:

[PostgreSQL: File Browser](https://www.enterprisedb.com/downloads/postgres-postgresql-downloads)

Es importante que al momento de instalar te asegures de que el puerto asignado sea el 5432 y la contraseña “TexMUAKM1995” sin las comillas, para evitar conflictos con la configuración que se tiene establecida en el archivo backend/app/config.py y que la conexión se pueda realizar sin problemas.

Una vez hecho, revisa si también se te instaló el pgAdmin 4, si no es el caso entonces descárgalo e instálalo manualmente desde el siguiente enlace:

[pgAdmin: Download](https://www.pgadmin.org/download/pgadmin-4-windows/)

Abre el pdAdmin, en la parte superior izquierda verás un icono de servidor acompañado del texto “Servers”, al intentar desplegar las opciones que este tiene te pedirá una contraseña, pon la misma contraseña que definiste en la primera instalación (TexMUAKM1995) y se te desplegarán varias opciones. Sobre la parte que dice “Databases” haz click derecho, pon el mouse sobre la opción “Create” y luego haz click sobre la opción de “Database…”. Se te desplegará un formulario para crear la base de datos, sólo debes ponerle como nombre “test” y darle en “Save”, con esto ya deberías tener tu base de datos creada, aunque todavía no tiene tablas.

Clona el repositorio del proyecto usando GitHub Desktop o desde la terminal. Ya con todos los archivos del proyecto entra en la carpeta backend e instala poetry desde la terminal.

-> pip install poetry

Instala los paquetes necesarios desde el archivo requirements.txt:

-> pip install -r requirements.txt

Instala las dependencias usando poetry:

-> poetry install

En Visual Studio se puede presentar un problema al ingresar este comando, y es que no te reconozca “poetry” como término válido. Para solucionar este problema debes identificar la ruta donde está ubicado Python, esta debe tener una carpeta llamada scripts dentro de la cual debes encontrar entre otras cosas un archivo llamado poetry. Ahora abre el panel de control y busca “editar las variables del entorno” metete ahí, agrega una nueva ruta poniéndole la dirección de la carpeta donde ubicaste el poetry, cierra y vuelve a abrir el Visual Studio y ejecuta el comando otra vez, esto debe haber solucionado el problema.

Genera las tablas para tu base de datos usando alembic:

-> alembic upgrade head

Si en este paso te da problemas, contáctate conmigo, ya que el alembic puede ser un poco quisquilloso. Si todo sale bien, al ingresar a pgAdmin y dirigirte a la base de datos test > Schemas (1) > public > Tables (10), te deberían aparecer todas las tablas de nuestra base de datos. Para garantizar que todo esté en orden, selecciona una de las tablas > click derecho > ERD For Table, y te debería aparecer el diagrama entidad-relación de la base de datos. Con esto la base de datos ya está configurada y cualquier cambio que hagamos en ella por medio de los servicios del backend se verán reflejados.

Corre FastAPI usando:

-> poetry run start

Sabes que te funcionó si te aparecen palabras en verde y un enlace que dice “localhost:8888”. Ya con esto deberías ser capaz de acceder a los servicios del backend, bien sea ingresando directamente a la url que ofrece FastAPI para las pruebas (localhost:8888/docs) o por medio del frontend.

1. **ESTRUCTURA DE CARPETAS**

Backend

|

|

- - - > app

|

- - - > controller

|

- - - > model

|

- - - > repository

|

- - - > service

|

- - - > config.py

|

- - - > Main.py

|

- - - > schema.py

Los nombres de los archivos contenidos en cada carpeta coinciden con la entidad o funcionalidad a la que están asociados.

**Controller:** Maneja la interfaz REST para la lógica de negocios. Básicamente contienen los routers que van a ser llamados desde el front, estos van a recibir un schema (esquema) que contiene los datos de entrada y devolverán un resultado o una excepción HTTP.

**Model:** Contiene las entidades de nuestra base de datos. Una entidad/modelo es una estructura de datos como una tabla del ERD (Diagrama Entidad Relación).

**Repository:** Contiene las consultas a la base de datos para cada una de las entidades. El archivo base\_repo.py contiene consultas básicas aplicables a cualquier entidad, las cuales son:

* crear(\*\*kwargs)
* buscar\_todo()
* buscar\_por\_id(model\_id: str, name\_id: str)
* actualizar\_por\_id(model\_id: str, name\_id: str, \*\*kwargs)
* eliminar\_por\_id(model\_id: str)

Las clases correspondientes al uso de tokens se encuentran definidas en el archivo auth\_repo.py, y sus métodos son utilizados en toda la parte del back.

**Service:** Implementa la lógica del negocio. Aquí se encuentran los servicios que serán usados por el front mediante llamadas al router correspondiente. Los servicios incluyen consultas a la base de datos que requieren del uso de varias entidades, consultas que requieren campos específicos y funciones específicas complejas, que pueden involucrar la validación de datos. El archivo auth\_service.py contiene los servicios que tengan que ver con verificar información, mientras que el resto están reservados para consultas más concretas a la base de datos.

**Config.py:** Establece la conexión con la base de datos, las operaciones commit y rollback, además de contener los parámetros que se usan para la tokenización.

**Main.py:** Como su nombre indica, es el archivo principal que se encarga de iniciar la API, establecer los router que se van a usar al importarlos del directorio controller, controlar el acceso a los servicios y realizar los llamados al config.py para que inicie y cierre la conexión con la base de datos.

**schema.py:** Contiene los esquemas, que son algo así como las plantillas que se usan para recibir o enviar información, estas definen los campos que cada mensaje tendrá, ya sea de entrada o de salida. De ser necesario, se pueden incluir validaciones para los campos que se ejecutarán automáticamente apenas se use el esquema.

1. **DISEÑO DE MODELOS**

**Actividad:**

ID\_actividad: int

ID\_entrenador: str

nombre: str

descripcion: str

precio: int

fecha: date (formato: %d-%m-%Y)

hora: time

duracion\_minutos: int

lugar: str

creado\_en: datetime.now

modificado\_en: datetime.now

**Administrador:**

ID\_admin: str

password: str

sexo: str [MASCULINO/FEMENINO]

nombre: str

segundo\_nombre: Optional[str] = None

apellido: str

segundo\_apellido: Optional[str] = None

fecha\_nacimiento: date (formato: %d-%m-%Y)

correo: str

direccion: str

telefono: str

rango: str [Entrenador]

creado\_en: datetime.now

modificado\_en: datetime.now

**Cita:**

ID\_cita: int

ID\_cliente: str

ID\_actividad: int

descripcion: str

fecha: date (formato: %d-%m-%Y)

hora: time

duracion\_minutos: int

lugar: str

creado\_en: datetime.now

modificado\_en: datetime.now

**Cliente:**

ID\_cliente: str

ID\_titular: Optional[str]

password: str

sexo: str [MASCULINO/FEMENINO]

nombre: str

segundo\_nombre: Optional[str] = None

apellido: str

segundo\_apellido: Optional[str] = None

fecha\_nacimiento: date (formato: %d-%m-%Y)

correo: str

telefono: str

direccion: str

activo: bool = True

peso: float

altura: float

rango: str [Cliente]

creado\_en: datetime.now

modificado\_en: datetime.now

**CompraMembresia:**

ID\_comprador: str

ID\_membresia: str

metodo\_pago: str

tipo\_compra: bool

creado\_en: datetime.now

modificado\_en: datetime.now

**Entrenador:**

ID\_entrenador: str

password: str

sexo: str [MASCULINO/FEMENINO]

nombre: str

segundo\_nombre: Optional[str] = None

apellido: str

segundo\_apellido: Optional[str] = None

fecha\_nacimiento: date (formato: %d-%m-%Y)

correo: str

telefono: str

direccion: str

especialidad: str

activo: bool = True

rango: str [Entrenador]

creado\_en: datetime.now

modificado\_en: datetime.now

**Entrenamiento:**

ID\_entrenamiento: str

ID\_entrenador: str

ID\_cliente: str

tipo: str

descripcion: str

**Membresia:**

ID\_membresia: str

ID\_admin\_creador: str

nombre: str

descripcion: str

descuento: float

max\_miembros: int

precio: int

duracion\_meses: int

creado\_en: datetime.now

modificado\_en: datetime.now

**Proveedor:**

ID\_proveedor: str

nombre: str

telefono: str

direccion: str

correo: str

producto: str

creado\_en: datetime.now

modificado\_en: datetime.now

1. **USO DE RUTAS**

Para hacer uso de los servicios que ofrece el back, se deben enviar mensajes en formato JSON a unas rutas específicas de cada servicio. Los routers se utilizan para agrupar las operaciones por rutas, reciben los mensajes y llaman a los servicios correspondientes, en caso de ocurrir algún error se hará saber mediante una excepción HTTP, si todo transcurre correctamente el router devolverá un mensaje en formato JSON que contiene dos campos: detalles y resultado. El mensaje de respuesta puede contener sólo el campo “detalles” si es un servicio que realiza alguna operación sobre la base de datos que no necesite el retorno de información contenida en la misma (véase el caso del login).

Todas las rutas empiezan de la misma forma: “<http://localhost:8888>”, pero de aquí en adelante va a depender del servicio que se desee consultar. Dentro del directorio controller hay varios archivos, cada uno está identificado a nivel de ruta por un prefijo que se especifica al inicio de cada archivo, cada uno de los subdirectorios contenidos en el archivo corresponden a un servicio y su ruta se puede identificar como el primer parámetro del router asociado.

Hay servicios que requieren el envío de parámetros por medio de la ruta, estos se pueden identificar fácilmente, ya que el nombre del parámetro se encuentra encerrado dentro de corchetes. Para este tipo de servicios no hay que poner los corchetes, en su lugar se pone directamente la variable que contiene el valor que se desea enviar.

Entonces, para llamar a cualquiera de los servicios de la API se utiliza la siguiente estructura de ruta:

http://localhost:8888(prefijo de ruta)(ruta del servicio)(variable de ruta)

Ejemplos:

*Ruta sin variable:* <http://localhost:8888/admin/proveedores>

*Ruta con variable:* <http://localhost:8888/admin/info_cliente/1091272102>

Lógicamente los servicios están distribuidos entre los archivos de manera coherente, agrupados de acuerdo a su función.

1. **LLAMADO A SERVICIOS**

Los servicios que ofrece la API se describen en la siguiente tabla:

| **ID** | **Archivo** | **Prefijo** | **Ruta y variable de ruta** | **Método** | **Descripción** |
| --- | --- | --- | --- | --- | --- |
| 1 | authentication.py | /auth | /registro\_cliente | post | Permite registrar un nuevo cliente en la base de datos |
| 2 | authentication.py | /auth | /login | post | Permite consultar si un usuario se encuentra en la base de datos |
| 3 | Usuario.py | /usuario | / | get | Permite consultar la información de un usuario sin importar su rango, exceptuando su contraseña |
| 4 | Administradores.py | /admin | /registro\_entrenador | post | Permite registrar un nuevo entrenador en la base de datos |
| 5 | Administradores.py | /admin | /registro\_administrador | post | Permite registrar un nuevo entrenador en la base de datos |
| 6 | Administradores.py | /admin | /actualizar\_entrenador | post | Permite actualizar la información de un entrenador registrado en la base de datos, exceptuando su identificación y fecha de nacimiento |
| 7 | Administradores.py | /admin | /actualizar\_administrador | post | Permite actualizar la información de un administrador registrado en la base de datos, exceptuando su identificación y fecha de nacimiento |
| 8 | Administradores.py | /admin | /eliminar\_cliente | delete | Permite eliminar un cliente de la base de datos |
| 9 | Administradores.py | /admin | /eliminar\_entrenador | delete | Permite eliminar un entrenador de la base de datos |
| 10 | Administradores.py | /admin | /eliminar\_administrador | delete | Permite eliminar un administrador de la base de datos |
| 11 | Administradores.py | /admin | /clientes | get | Permite consultar la lista completa de los clientes registrados en la base de datos, sin importar si su estado es activo o no. La lista está ordenada alfabéticamente de acuerdo al nombre de los clientes. |
| 12 | Administradores.py | /admin | /agregar\_proveedor | post | Permite agregar un nuevo proveedor a la base de datos |
| 13 | Administradores.py | /admin | /modificar\_proveedor | post | Permite actualizar la información de un proveedor registrado en la base de datos, exceptuando su identificación y la del administrador creador |
| 14 | Administradores.py | /admin | /eliminar\_proveedor | delete | Permite eliminar un proveedor de la base de datos |
| 15 | Administradores.py | /admin | /proveedores | get | Permite consultar la lista completa de los proveedores registrados en la base de datos. La lista está ordenada alfabéticamente de acuerdo al nombre de los proveedores |
| 16 | Cliente.py | /cliente | /actualizar\_cliente | post | Permite actualizar la información de un cliente registrado en la base de datos |
| 17 | Entrenadores.py | /entrenador | /clientes\_activos | get | Permite consultar la lista de los clientes activos registrados en la base de datos. La lista está ordenada alfabéticamente de acuerdo al nombre de los clientes |
| 18 | Administradores.py | /admin | /modificar\_estado\_cliente | post | Permite modificar el estado activo de un cliente, el cual está dado por un booleano |
| 19 | Administradores.py | /admin | /modificar\_estado\_entrenador | post | Permite modificar el estado activo de un entrenador, el cual está dado por un booleano |
| 20 | Administradores.py | /admin | /info\_cliente/{id\_cliente} | patch | Permite consultar la información de un cliente pasando su id como variable de ruta |
| 21 | Administradores.py | /admin | /info\_entrenador/{id\_entrenador} | patch | Permite consultar la información de un entrenador pasando su id como variable de ruta |
| 22 | Administradores.py | /admin | /info\_proveedor/{id\_proveedor} | patch | Permite consultar la información de un proveedor pasando su id como variable de ruta |
| 23 | Administradores.py | /admin | /agregar\_membresia | post | Permite agregar una membresía a la base de datos |
| 24 | Administradores.py | /admin | /entrenadores | get | Permite consultar la lista completa de los entrenadores registrados en la base de datos. La lista está ordenada alfabéticamente de acuerdo al nombre de los entrenadores |
| 25 | Administradores.py | /admin | /administradores | get | Permite consultar la lista completa de los administradores registrados en la base de datos. La lista está ordenada alfabéticamente de acuerdo al nombre de los administradores |
| 26 | Admin\_Cliente.py | /info | /membresias | get | Permite consultar la lista completa de las membresías disponibles. En el caso de los administradores, retorna la lista completa de las membresías registradas en la base de datos, pero en el caso de los clientes retorno sólo aquellas membresías que no haya comprado y que por lo tanto están disponibles para la venta |
| 27 | Cliente.py | /cliente | /membresias\_compradas | get | Permite consultar la lista de membresías adquiridas por el cliente, los registros de compra se encuentran en una tabla aparte en la base de datos. La lista está ordenada alfabéticamente de acuerdo al nombre de las membresías |
| 28 | Cliente.py | /cliente | /comprar\_membresia/{id\_membresia} | post | Permite agregar un registro de compra de membresía a la base de datos |
| 29 | Admin\_Cliente.py | /info | /membresia/{id\_membresia} | get | Permite a administradores y clientes consultar la información completa de una membresía y saber si existe un registro de compra a su nombre en la base de datos sin importar su rango. En el caso de los administradores nunca existirá un registro de compra, pero en los clientes sí existe esa posibilidad. |

1. **ENVÍO DE DATOS PARA SERVICIOS**

Los campos que se dejan vacíos al momento de enviar un formulario se almacenan como un valor nulo en la base de datos, y se retornan en forma de cadena vacía.

Los únicos campos que tienen restricciones en cuanto a los valores que pueden tener son **sexo** y **rango**, además de ser case-sensitive. Los únicos valores que pueden tomar cada uno son:

**Sexo** = MASCULINO / FEMENINO

**Rango** = Cliente / Entrenador / Administrador

Lógicamente otros campos tienen restricciones intrínsecas, como la cantidad de dígitos del número de identificación (cédula), pero estos tienen un mayor rango de posibles valores que pueden adoptar, mientras que los dos mencionados anteriormente se ven limitados a las opciones mostradas.

Cuando se menciona el “token\_de\_usuario” en realidad no se refiere únicamente al token de autenticación, sino que es una concatenación entre el tipo de token y el token en sí, separados por un espacio en blanco.

token\_de\_usuario = (Tipo de token) + “ “ + (Token de autenticación)

Obviamente, el tipo de token y el token en sí deben haber sido recuperados del almacenamiento interno previamente.

La columna de “Excepciones” hace referencia a aquellas excepciones específicas de HTTP que se pueden llegar a producir tanto por el schema utilizado como por las validaciones internas que se hacen. En todo caso se lista el conjunto completo de excepciones, pero puede darse el caso de que algunas excepciones no lleguen a presentarse por la naturaleza del servicio, o se integran únicamente como forma de garantizar la integridad de los datos.

| **ID de servicio** | **Datos de entrada** | **Retorno** | **Excepciones** |
| --- | --- | --- | --- |
| 1 | *{*  *"ID": "string",*  *"password": "string",*  *"sexo": "string",*  *"nombre": "string",*  *"segundo\_nombre": "string",*  *"apellido": "string",*  *"segundo\_apellido": "string",*  *"fecha\_nacimiento": "string",*  *"correo": "string",*  *"telefono": "string",*  *"direccion": "string",*  *"peso": integer,*  *"altura": float*  *}*  No requerido en formulario:  - segundo\_nombre | *{*  *detalles:*  *“Cuenta creada exitosamente”*  *resultado:*  *null*  *}* | - Tipo de cédula inválida  - Número de teléfono inválido  - Género inválido  - Ya hay un cliente registrado con esa cédula  - Ya hay un entrenador registrado con esa cédula  - Ya hay un administrador registrado con esa cédula  - Ya hay un cliente registrado con ese correo  - Ya hay un entrenador registrado con ese correo  - Ya hay un administrador registrado con ese correo |
| 2 | *{*  *"correo": "string",*  *"password": "string",*  *"rango": "string"*  *}* | *{*  *detalles:*  *“Sesión iniciada”,*  *resultado:*  *{*  *“tipo\_de\_token”:*  *“Bearer”,*  *“token\_de\_acceso”:*  (Token del usuario)  *}*  *}* | - Contraseña inválida  - El cliente no existe  - El administrador no existe  - El entrenador no existe  - Rango inválido |
|
| 3 | Se entrega directamente el token\_de\_usuario | *{*  *detalles:*  *“Recuperación de datos exitosa”,*  *resultado:*  *{*  *“rango\_usuario”:*  (String con el rango del usuario)*,*  *“info\_usuario”:*  (Diccionario con información completa del usuario)  *}*  *}* | - Esquema de autenticación inválido  - Token inválido o expirado  - Código de autorización inválido |
| 4 | *{*  *"ID": "string",*  *"password": "string",*  *"sexo": "string",*  *"nombre": "string",*  *"segundo\_nombre": "string",*  *"apellido": "string",*  *"segundo\_apellido": "string",*  *"fecha\_nacimiento": "string",*  *"correo": "string",*  *"telefono": "string",*  *"direccion": "string",*  *"especialidad": "string"*  *}*  No requerido en formulario:  - segundo\_nombre | *{*  *detalles:*  *“Entrenador creado exitosamente”*  *resultado:*  *null*  *}* | - Tipo de cédula inválida  - Número de teléfono inválido  - Género inválido  - Ya hay un cliente registrado con esa cédula  - Ya hay un entrenador registrado con esa cédula  - Ya hay un administrador registrado con esa cédula  - Ya hay un cliente registrado con ese correo  - Ya hay un entrenador registrado con ese correo  - Ya hay un administrador registrado con ese correo |
| 5 | *{*  *"ID": "string",*  *"password": "string",*  *"sexo": "string",*  *"nombre": "string",*  *"segundo\_nombre": "string",*  *"apellido": "string",*  *"segundo\_apellido": "string",*  *"fecha\_nacimiento": "string",*  *"correo": "string",*  *"telefono": "string",*  *"direccion": "string"*  *}*  No requerido en formulario:  - segundo\_nombre | *{*  *detalles:*  *“Administrador creado exitosamente”*  *resultado:*  *null*  *}* | - Tipo de cédula inválida  - Número de teléfono inválido  - Género inválido  - Ya hay un cliente registrado con esa cédula  - Ya hay un entrenador registrado con esa cédula  - Ya hay un administrador registrado con esa cédula  - Ya hay un cliente registrado con ese correo  - Ya hay un entrenador registrado con ese correo  - Ya hay un administrador registrado con ese correo |
| 6 | *{*  *"ID": "string",*  *"password": "string",*  *"sexo": "string",*  *"nombre": "string",*  *"segundo\_nombre": "string",*  *"apellido": "string",*  *"segundo\_apellido": "string",*  *"fecha\_nacimiento": "string",*  *"correo": "string",*  *"telefono": "string",*  *"direccion": "string",*  *"especialidad": "string",*  *"activo": bool*  *}*  No se pueden modificar:  - fecha\_nacimiento  - ID | *{*  *detalles:*  *“Perfil de entrenador actualizado exitosamente”*  *resultado:*  *null*  *}* | - Tipo de cédula inválida  - Número de teléfono inválido  - Género inválido  - El entrenador no existe |
| 7 | *{*  *"ID": "string",*  *"password": "string",*  *"sexo": "string",*  *"nombre": "string",*  *"segundo\_nombre": "string",*  *"apellido": "string",*  *"segundo\_apellido": "string",*  *"fecha\_nacimiento": "string",*  *"correo": "string",*  *"telefono": "string",*  *"direccion": "string"*  *}*  No se pueden modificar:  - fecha\_nacimiento  - ID | *{*  *detalles:*  *“Perfil de administrador actualizado exitosamente”*  *resultado:*  *null*  *}* | - Tipo de cédula inválida  - Número de teléfono inválido  - Género inválido  - El administrador no existe |
| 8 | *{*  *"ID": "string"*  *}* | *{*  *detalles:*  *“Perfil de cliente eliminado exitosamente”*  *resultado:*  *null*  *}* | - Número de cédula inválido  - El cliente no existe |
| 9 | *{*  *"ID": "string"*  *}* | *{*  *detalles:*  *“Perfil de entrenador eliminado exitosamente”*  *resultado:*  *null*  *}* | - Número de cédula inválido  - El entrenador no existe |
| 10 | *{*  *"ID": "string"*  *}* | *{*  *detalles:*  *“Perfil de administrador eliminado exitosamente”*  *resultado:*  *null*  *}* | - Número de cédula inválido  - El administrador no existe |
| 11 | No aplica | *{*  *detalles:*  *“Clientes accedidos correctamente”,*  *resultado:*  (Lista que contiene diccionarios, donde cada uno tiene únicamente el id, el nombre, el sexo y el teléfono de un cliente)  *}* | - No existen clientes |
| 12 | *{*  *"ID\_proveedor": "string",*  *"ID\_admin\_creador": "string",*  *"nombre": "string",*  *"telefono": "string",*  *"direccion": "string",*  *"correo": "string",*  *"producto": "string"*  *}* | *{*  *detalles:*  *“Proveedor agregado exitosamente”*  *resultado:*  *null*  *}* | - Cédula de proveedor inválida  - Cédula de administrador inválida  - Teléfono de proveedor inválido  - El administrador no existe  - El proveedor ya existe  - Ya hay un proveedor registrado con ese nombre |
| 13 | *{*  *"ID\_proveedor": "string",*  *"ID\_admin\_creador": "string",*  *"nombre": "string",*  *"telefono": "string",*  *"direccion": "string",*  *"correo": "string",*  *"producto": "string"*  *}*  No se pueden modificar:  - ID\_proveedor  - ID\_admin\_creador | *{*  *detalles:*  *“Proveedor modificado exitosamente”*  *resultado:*  *null*  *}* | - Cédula de proveedor inválida  - Cédula de administrador inválida  - Teléfono de proveedor inválido  - El administrador no existe  - El proveedor no existe |
| 14 | *{*  *"ID": "string"*  *}* | *{*  *detalles:*  *“Proveedor eliminado exitosamente”*  *resultado:*  *null*  *}* | - Número de cédula inválido  - El proveedor no existe |
| 15 | No aplica | *{*  *detalles:*  *“Proveedores accedidos correctamente”,*  *resultado:*  (Lista que contiene diccionarios, donde cada uno tiene únicamente el id, el nombre, la dirección y el teléfono de un proveedor)  *}* | - No existen proveedores |
| 16 | *{*  *"ID": "string",*  *"password": "string",*  *"sexo": "string",*  *"nombre": "string",*  *"segundo\_nombre": "string",*  *"apellido": "string",*  *"segundo\_apellido": "string",*  *"fecha\_nacimiento": "string",*  *"correo": "string",*  *"telefono": "string",*  *"direccion": "string",*  *"peso": 0,*  *"altura": 0,*  *"activo": true*  *}*  No se pueden modificar:  - fecha\_nacimiento  - ID | *{*  *detalles:*  *“Perfil de cliente actualizado exitosamente”*  *resultado:*  *null*  *}* | - Tipo de cédula inválida  - Número de teléfono inválido  - Género inválido  - El cliente no existe |
| 17 | No aplica | *{*  *detalles:*  *“Clientes activos accedidos correctamente”,*  *resultado:*  (Lista que contiene diccionarios, donde cada uno tiene únicamente el id, nombre, sexo y teléfono de un cliente activo)  *}* | - No existen clientes activos |
| 18 | *{*  *"ID": "string",*  *"activo": true*  *}* | *{*  *detalles:*  *“Estado de cliente actualizado”*  *resultado:*  *null*  *}* | - Cédula inválida  - El cliente no existe |
| 19 | *{*  *"ID": "string",*  *"activo": true*  *}* | *{*  *detalles:*  *“Estado de entrenador actualizado”*  *resultado:*  *null*  *}* | - Cédula inválida  - El entrenador no existe |
| 20 | Variable: Id\_cliente | *{*  *detalles:*  *“Información recuperada con éxito”,*  *resultado:*  (Lista que contiene todos los datos del cliente, exceptuando su password)  *}* | - El cliente no existe |
| 21 | Variable: Id\_entrenador | *{*  *detalles:*  *“Información recuperada con éxito”,*  *resultado:*  (Lista que contiene todos los datos del entrenador, exceptuando su password)  *}* | - El entrenador no existe |
| 22 | Variable: Id\_proveedor | *{*  *detalles:*  *“Información recuperada con éxito”,*  *resultado:*  (Lista que contiene todos los datos del proveedor)  *}* | - El proveedor no existe |
| 23 | *{*  *"ID\_admin\_creador": "string",*  *"nombre": "string",*  *"descripcion": "string",*  *"descuento": 0,*  *"max\_miembros": 0,*  *"precio": 0,*  *"duracion\_meses": 0*  *}*  El ID\_membresia es autogenerado | *{*  *detalles:*  *“Membresía agregada exitosamente”*  *resultado:*  *null*  *}* | - Cédula de administrador inválida  - Descuento inválido  - El administrador no existe  - Ya existe una membresía con ese nombre |
| 24 | No aplica | *{*  *detalles:*  *“Entrenadores accedidos correctamente”,*  *resultado:*  (Lista que contiene diccionarios, donde cada uno tiene únicamente el id, el sexo, el nombre y el teléfono un entrenador)  *}* | - No existen entrenadores |
| 25 | No aplica | *{*  *detalles:*  *“Administradores accedidos correctamente”,*  *resultado:*  (Lista que contiene diccionarios, donde cada uno tiene únicamente el id, el sexo, el nombre y el teléfono un administrador)  *}* | - No existen administradores |
| 26 | No aplica | *{*  *detalles:*  *“Membresías accedidas correctamente”,*  *resultado:*  (Lista que contiene diccionarios, donde cada uno tiene únicamente el nombre, id, precio y duración en meses de una membresía)  *}* | - No existen membresías en la base de datos  - No hay membresías disponibles para la compra |
| 27 | No aplica | *{*  *detalles:*  *“Lista de membresías compradas accedida correctamente”,*  *resultado:*  (Lista que contiene diccionarios, donde cada uno tiene únicamente el nombre, descripción, número máximo de miembros, duración en meses y descuento de una de las membresías compradas por el cliente)  *}* | - No hay membresías compradas  - El cliente no existe |
| 28 | Variable: Id\_membresia | *{*  *detalles:*  *“Compra realizada exitosamente”,*  *resultado:*  *null*  *}* | - La membresía no existe  - El cliente no existe  - El cliente ya compró esta membresía |
| 29 | Variable: Id\_membresia | *{*  *detalles:*  *“Información recuperada con éxito”,*  *resultado:*  *“Membresia\_comprada”:*  *bool,*  *“Info\_membresia”:*  (Diccionario que contiene toda la información de la membresía indicada)  *}* | - La membresía especificada no existe  - El administrador no existe  - El entrenador no existe  - El cliente no existe |