



TAREA INTEGRADORA

GYM BULL'S

UDN: ATENCIÓN AL CLIENTE

OSIEL PAREDES CASTILLO
MAXIMILIANO AMADOR PEÑA
ADÁN SALAS GALVAN

PLAN DE TRABAJO



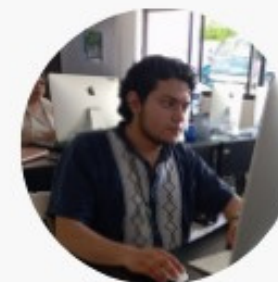
ORGANIGRAMA



Oscar Osiel Paredes Castillo
Líder de Equipo
Desarrollador Backend



Maximiliano Amador Peña
Desarrollador Frontend
Desarrollador de Bases de datos



Adán Salas Galván
Documentador

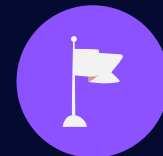
PLAN DE TRABAJO



Cargos



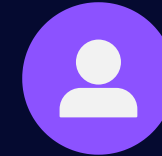
Objetivo



Meta



Actividades



Responsable

Isabel Mercado

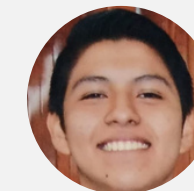
Cargo

Desarrollador Frontend
Desarrollador de BD

Tener desarrolladas las BD de nuestro módulo y darnos correcta comunicación las otras BD del proyecto

- Desarrollar Todas las Bases de Datos que almacenarán la información del Módulo.

Crear BD, crear comunicación entre ellas y las BD que sean requeridas en el proyecto
Desarrollar los componentes con los que los usuarios finales interactuarán



Maximiliano Amador Peña

Isabel Mercado

Cargo

Líder de Equipo
Desarrollador Backend

- Haber liderado un equipo competentemente en el correcto desarrollo de un proyecto para conseguir experiencia práctica.
- crear y mantener la lógica y la funcionalidad que respalda el funcionamiento del sitio web.

- Llevar al equipo al éxito en el desarrollo del proyecto.
- Encargarse de que cada componente responda y cumpla su función.

- Guiar al equipo para el correcto y optimo desarrollo de nuestro módulo.
- Adaptar y crear funciones que peritan la comunicación entre BD y el sitio web, y la responda ante la interacción con el frontend.



Oscar Osiel Paredes Castillo

Isabel Mercado

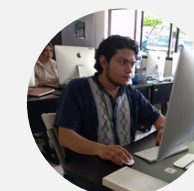
Cargo

Documentador

Haber creado todos los documentos requeridos que sirvan a para consultar el proceso de desarrollo del proyecto.

- Contar con todo el proyecto correctamente documentado para su posterior consulta y archivamiento.

Crear documentos que almacenen el proceso del desarrollo del proyecto.



Adán Salas Galván

Ricardo Salinas

Commits

Apr 7, 2024 – Apr 19, 2024

Contributions: Commits ▾

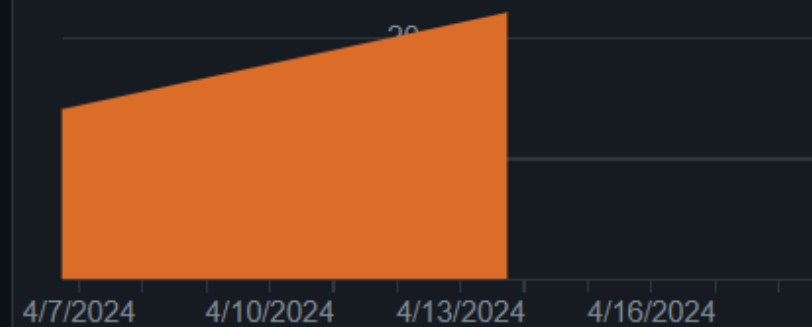
Contributions to main, excluding merge commits



Osiel-Paredes

36 commits 1,131,010 ++ 673,284 --

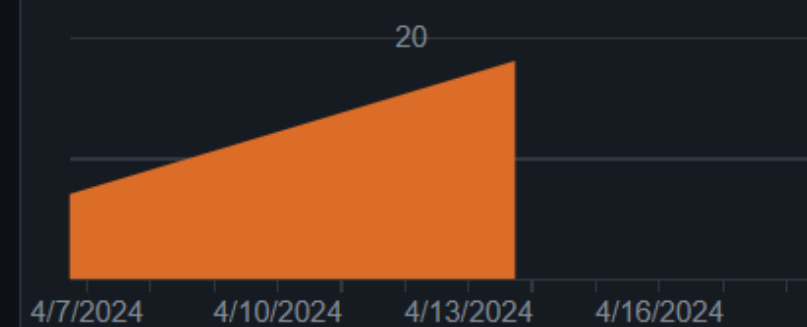
#1



maxhithub12

25 commits 56,721 ++ 205 --

#2



AztecEagleWarrior

10 commits 32 ++ 3 --

#3



DOCUMENTACION

Requerimientos Funcionales

1. RF02: Contar con validaciones para CRUD o formularios.
2. RF05: El sistema debe mostrar un calendario con los horarios de clases, sesiones de entrenamiento y disponibilidad de equipos.
3. RF07: El sistema debe permitir a los usuarios realizar un seguimiento de su progreso físico y de salud, registrando datos como peso, medidas corporales, objetivos de entrenamiento y logros alcanzados.
4. RF09: Los usuarios deben poder realizar pagos en línea por membresías, clases adicionales, entrenamientos personales y productos del gimnasio, con opciones seguras de procesamiento de pagos.
5. RF11: El usuario debe poder reservar clases de fitness, sesiones de entrenamiento personal y equipos específicos a través del sistema en línea, con la opción de seleccionar la fecha, hora y tipo de actividad deseada.
6. RF19: El sistema debe permitir la programación y promoción de eventos especiales, como competencias, talleres y conferencias relacionadas con el fitness y la salud.
7. RF25: El sistema debe permitir la realización de encuestas periódicas de satisfacción del cliente para recopilar comentarios y sugerencias sobre los servicios y la experiencia del usuario.
8. RF17: El sistema debe proporcionar programas de entrenamiento personalizados y planes de ejercicio adaptados a los objetivos y necesidades específicas de los usuarios.
9. RF04: Los usuarios deben poder registrarse con su información personal de manera correcta e intuitiva.
10. RF15: El sistema debe proporcionar un sistema de soporte al cliente en línea donde los usuarios puedan enviar consultas, reportar problemas o solicitar asistencia del personal del gimnasio.
11. RF28: Los usuarios deben poder registrar y hacer seguimiento de su ingesta de alimentos, calorías consumidas y cumplimiento de planes dietéticos a través del sistema.
12. RF29: El sistema debe ofrecer múltiples opciones de pago, incluyendo tarjetas de crédito, débito, transferencias bancarias, y métodos de pago en línea como PayPal.
13. RF30: El sistema debe cumplir con los estándares de seguridad de datos y las normativas de pago, como el cumplimiento del estándar PCI DSS, para garantizar la protección de la información financiera de los usuarios.
14. RF14: El sistema debe generar informes periódicos sobre la asistencia a clases, el uso de instalaciones, la satisfacción del cliente y otros datos relevantes para la gestión del gimnasio.
15. RF24: El sistema debe permitir el seguimiento del consumo de productos y servicios del gimnasio por parte de los usuarios, incluyendo alimentos, suplementos y tratamientos adicionales.

Requerimientos no Funcionales

1. RNF01 . El sistema debe cumplir con estándares de seguridad de datos como GDPR y PCI DSS para proteger la información financiera y personal de los usuarios.
2. RNF02. El sistema debe tener un tiempo de respuesta rápido, con una carga de página y procesamiento de transacciones que no exceda los 2 segundos en condiciones normales de carga.
3. RNF03 . El sistema debe estar disponible las 24 horas del día, los 7 días de la semana, con una disponibilidad del 99.9% para garantizar el acceso continuo de los usuarios
4. RNF04. El sistema debe ser capaz de escalar verticalmente y horizontalmente para manejar un aumento en el número de usuarios y transacciones sin degradación del rendimiento.
5. RNF05. El sistema debe ser compatible con una amplia variedad de dispositivos y navegadores web, incluyendo computadoras de escritorio, dispositivos móviles y tabletas.
6. RNF06. El sistema debe ser fácil de usar y accesible para usuarios de todas las habilidades, incluyendo aquellos con discapacidades visuales o motoras.
7. RF11 . El sistema debe permitir la personalización de la interfaz y funcionalidades según las necesidades y preferencias específicas del gimnasio y sus usuarios.
8. RNF08. El sistema debe contar con una base de datos optimizada que garantice tiempos de consulta rápidos y eficientes, incluso bajo cargas pesadas de trabajo.
9. RNF10. La interfaz de usuario del sistema debe ser atractiva visualmente y proporcionar una experiencia de usuario intuitiva que fomente la participación y retención de los usuarios.
10. RNF07. El sistema debe ser fácil de mantener y actualizar, con código limpio y bien documentado que permita a los desarrolladores realizar cambios sin interrupciones en el servicio.

Reglas de negocio

1. RN01: Todos los mensajes de quejas y sugerencias deben ser respondidos dentro de un plazo máximo de 24 horas laborables.
2. RN02: Los usuarios deben recibir una confirmación de recepción de su queja o sugerencia inmediatamente después de enviarla.
3. RN03: Las sesiones de entrenamiento personalizado deben programarse según la disponibilidad del cliente y confirmarse con al menos 48 horas de anticipación.
4. RN04: Cualquier cancelación de sesión programada debe notificarse al cliente con al menos 24 horas de anticipación, ofreciendo la posibilidad de reprogramación sin penalización.
5. RN05: Los usuarios que realicen quejas deben recibir un seguimiento regular para garantizar su satisfacción y resolver cualquier problema pendiente.
6. RN06: Los clientes deben poder acceder fácilmente a los horarios de clases, sesiones de entrenamiento y disponibilidad de equipos a través del sitio web o la aplicación móvil.
7. RN07: Se debe proporcionar asistencia técnica y de navegación para aquellos usuarios que tengan dificultades para utilizar las funciones del sitio web o la aplicación móvil.
8. RN08: Las notificaciones automáticas, como recordatorios de citas y actualizaciones de horarios, deben ser claras y enviarse con suficiente antelación para permitir la planificación del cliente.
9. RN09: Los usuarios deben recibir una confirmación inmediata por correo electrónico o mensaje de texto después de realizar un pago en línea por membresías, clases adicionales, etc.
10. RN10: Cualquier cambio en los términos y condiciones de membresía debe ser comunicado a los clientes con al menos 30 días de anticipación.
11. RN11: Se debe proporcionar asistencia personalizada a los usuarios que necesiten ayuda para establecer objetivos de entrenamiento o planificar su programa de ejercicio.
12. RN12: Los informes de satisfacción del cliente deben ser generados y revisados regularmente por el equipo de gestión para identificar áreas de mejora y tomar medidas correctivas.
13. RN13: Los usuarios deben tener la opción de proporcionar comentarios y calificaciones sobre su experiencia general con el gimnasio a través del sitio web o la aplicación móvil.
14. RN14: Se debe ofrecer un servicio de chat en vivo o una línea telefónica de asistencia al cliente para resolver consultas urgentes o problemas técnicos.
15. RN15: Las promociones y descuentos especiales deben ser comunicados claramente a los clientes a través de correos electrónicos, mensajes de texto o notificaciones en la aplicación.
16. RN16: Los usuarios deben recibir un seguimiento periódico de su progreso físico y de salud por parte del equipo de entrenadores personales del gimnasio.
17. RN17: Las consultas de los usuarios deben ser canalizadas de manera eficiente a los departamentos correspondientes para una resolución oportuna.
18. RN18: Se debe ofrecer capacitación periódica al personal de atención al cliente para garantizar un servicio de alta calidad y satisfacción del cliente.
19. RN19: Los usuarios deben tener acceso a una sección de preguntas frecuentes (FAQ) donde puedan encontrar respuestas a las consultas comunes sobre el gimnasio y sus servicios.
20. RN20: El personal de atención al cliente debe estar capacitado para manejar situaciones difíciles y resolver quejas de manera profesional y respetuosa.

BASE DE DATOS (SQL)

Se construyo una base de datos de acuerdo a las necesidades del sitio web, en el caso de nuestro modulo Atencion al Cliente se requirio crear tablas que cumplieran con los requisitos de estos.

SERVICIO_AL_CLIENTE

servicio_al_cliente

🔑 ID: int
nombre_servicio: varchar(50)
disponibilidad: enum
descripcion: text
♦ instructor_id: int UNSIGNED
♦ sucursal_id: int UNSIGNED

QUEJAS_SUGERENCIAS

quejas_sugerencias

🔑 ID: int
fecha: timestamp
descripcion: text
estatus: enum
tipo: enum
♦ persona_id: int UNSIGNED

PREGUNTA

pregunta



ID: int

Tipo: enum



persona_id: int UNSIGNED

SERVICIO_USUARIOS

servicios_usuarios

- ◆ miembros_id: int UNSIGNED
- ◆ servicio_id: int
- ◆ empleado_id: int UNSIGNED
- fecha_de_uso: date
- Tiempo_uso: time

SUCURSALES_SERVICIOS

sucursales_servicios

- ◆ sucursal_id: int UNSIGNED
- ◆ servicio_id: int
- estatus: enum

EVALUACION_SERVICIO

evaluacion_servicio

- servicio_id: int
- comentarios: text
- fecha_registro: date
- puntuacion: enum
- ◆ persona_id: int UNSIGNED

BASE DE DATOS (NOSQL)

Tambien se construyo una base de datos NoSQL ya que vimos necesario implementarla mediante una coleccion.

SEGUIMIENTO_QUEJA

```
[{
  "_id": {
    "$oid": "6621f6b4a13bf393eb562845"
  },
  "queja_id": "QJ1234",
  "fecha": {
    "$date": "2024-04-16T00:00:00.000Z"
  },
  "usuario": {
    "nombre": "Juan Pérez",
    "email": "juan@example.com"
  },
  "empleado": {
    "nombre": "María García",
    "email": "maria@example.com"
  },
  "descripcion": "Se ha recibido la queja y se está investigando el caso.",
  "estatus": "En proceso",
  "tipo": "Seguimiento estándar",
  "createdAt": {
    "$date": "2024-04-19T04:44:27.169Z"
  },
  "updatedAt": {
    "$date": "2024-04-19T04:44:27.169Z"
  }
}]
```


ESQUEMA VALIDACION

```
{
  "$jsonSchema": {
    "bsonType": "object",
    "required": ["queja_id", "fecha", "usuario", "empleado", "descripcion", "estatus", "tipo", "createdAt", "updatedAt"],
    "properties": {
      "queja_id": {
        "bsonType": "string",
        "description": "ID de la queja",
        "minLength": 1
      },
      "fecha": {
        "bsonType": "date",
        "description": "Fecha del seguimiento"
      },
      "usuario": {
        "bsonType": "object",
        "required": ["nombre", "email"],
        "properties": {
          "nombre": {
            "bsonType": "string",
            "description": "Nombre del usuario"
          },
          "email": {
            "bsonType": "string",
            "description": "Correo electrónico del usuario",
            "pattern": "^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,}$"
          }
        }
      }
    }
  }
}
```

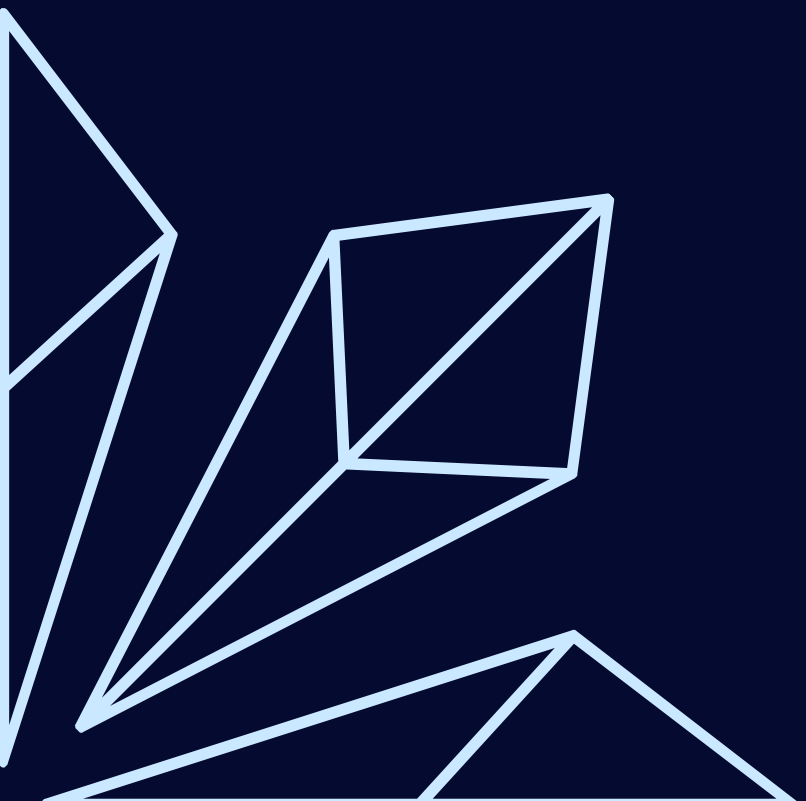
DESARROLLO COLABORATIVO

Visual Studio Code: es un editor de código fuente desarrollado por Microsoft que se ha vuelto muy popular entre los desarrolladores de software. Es gratuito, de código abierto y multiplataforma, lo que significa que está disponible para Windows, macOS y Linux.

MYSQL Workbench: Es una aplicación oficial de MySQL, desarrollada y mantenida por Oracle Corporation. MySQL Workbench proporciona una serie de características que hacen que trabajar con bases de datos MySQL sea más eficiente y fácil, especialmente para aquellos que prefieren una interfaz gráfica.



Git Hub: Es una plataforma de desarrollo de software basada en la nube que ofrece alojamiento de repositorios de control de versiones Git, gestión de proyectos, seguimiento de problemas (issue tracking), colaboración en el desarrollo de código y otras herramientas para equipos de desarrollo de software.



SITIO WEB (CRUD)

<http://localhost:5173/auth/login>

SITIO WEB (TABLA DINAMICA)

<http://localhost:5173/dashboardmao/analytical>

SITIO WEB (DASHBOARD)

<http://localhost:5173/dashboardmao/modern>

PLAN DE SEGURIDAD

ROLES

```
sql> create role cliente;  
Query OK, 0 rows affected (0.05 sec)
```

```
mysql> create role administrador;  
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> create role desarrollador;  
Query OK, 0 rows affected (0.02 sec)
```

```
mysql> create role empleado;  
Query OK, 0 rows affected (0.01 sec)
```


USUARIOS

Se crean los usuarios que tendran manejo a la base de datos con su contraseña de iidentificacion.

```
mysql> create user 'Adan'@'%' identified by 'Adan123';  
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> create user 'Maximiliano'@'%' identified by 'Max123';  
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> create user 'Osiel'@'%' identified by 'Osiel123';  
Query OK, 0 rows affected (0.03 sec)
```

PRIVILEGIOS

Se le asignan los privilegios que tendran a los roles de acuerdo a las necesidades de estos asi limitando el manejo de la base de datos.

```
mysql> GRANT select on bd_gimnasio_210857.* to cliente;  
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> GRANT SELECT, INSERT, UPDATE ON bd_gimnasio_210857.* TO empleado;  
Query OK, 0 rows affected (0.01 sec)
```

```
sql> GRANT SELECT, INSERT, UPDATE, ALTER, DELETE ON bd_gimnasio_210857.* TO administrador;  
ery OK, 0 rows affected (0.02 sec)
```

RESPALDOS

DB_HOST=3306

DB_USER=root

DB_PASS=12345

DB_NAME=bd_gimnasio_210857

BACKUP_DIR=/max/documnets/maxx

BACKUP_FILE="\$BACKUP_DIR/\$DB_NAME_\$(date +%Y%m%d_%H%M%S').sql"

Se ejecuta mysqldump para respaldar las tablas seleccionadas

```
mysqldump -h $DB_HOST -u $DB_USER -p$DB_PASS $DB_NAME quejas_sugerencias servicio_al_cliente >  
$BACKUP_FILE
```

Se programa de manera que se haga periodicamente

```
0 3 * * * /max/documents/script/respaldo_mysql_gimnasio.sh
```

ROLES EN NOSQL

Al crear roles en nosql permite asignar los privilegios desde ese momento

```
db.createRole({
  role: "cliente",
  privileges: [
    { resource: { db: "bd_gimnasio", collection: "" }, actions: ["find", "listCollections"] }
  ],
  roles: []
})
< { ok: 1 }
```

```
db.createRole({
  role: "empleado",
  privileges: [
    { resource: { db: "bd_gimnasio", collection: "" }, actions: ["find", "insert", "update"] }
  ],
  roles: []
})
< { ok: 1 }
```

```
db.createRole({
  role: "administrador",
  privileges: [
    { resource: { db: "bd_gimnasio", collection: "" }, actions: ["find", "insert", "update", "remove", "createIndex", "dropIndex", "dropCol
  ],
  roles: []
})
{ ok: 1 }
```

```
db.createRole({
  role: "desarrollador",
  privileges: [],
  roles: [
    { role: "dbOwner", db: "bd_gimnasio" }
  ]
})
< { ok: 1 }
```


CREACION DE USUARIOS EN NOSQL

Cuando se crean usuarios en nosql se pueden asignar los roles que ya tenemos establecidos.

```
> db.createUser({
  user: "Maximiliano",
  pwd: "Max123",
  roles: ["administrador"]
})
< { ok: 1 }
```

```
> db.createUser({
  user: "Adan",
  pwd: "Adan123",
  roles: ["empleado","cliente"]
})
< { ok: 1 }
```

```
> db.createUser({
  user: "Osiel",
  pwd: "Osiel123",
  roles: ["desarrollador"]
})
< { ok: 1 }
```

RESPALDOS EN NOSQL

```
DB_HOST=3306
```

```
DB_USER=root
```

```
DB_PASS=12345
```

```
DB_NAME=bd_gimnasio_210857
```

```
BACKUP_DIR=/max/documnets/maxx
```

```
BACKUP_FILE="$BACKUP_DIR/$DB_NAME_$(date +%Y%m%d_%H%M%S').sql"
```

Se ejecuta mysqldump para respaldar las tablas seleccionadas `mysqldump -h $DB_HOST -u $DB_USER -p$DB_PASS $DB_NAME quejas_sugerencias servicio_al_cliente > $BACKUP_FILE`

Se programa de manera que se haga periodicamente `0 3 * * * /max/documents/script/respaldo_mysql_gimnasio.sh`

API - REST

DOCUMENTACION CON CORE API

The screenshot shows a web browser at the address `127.0.0.1:8000/docs/`. The browser's address bar and tabs are visible at the top. The page content is divided into three main sections:

- Left Sidebar (Api Documentation):** Contains a list of API endpoints with expandable dropdowns: `EvaluacionServicio`, `Pregunta`, `QuejaSugerencia`, `ServicioCliente`, `ServicioUsuario`, and `SucursalServicio`. At the bottom, it shows `Authentication` (none) and `Source Code` (shell).
- Main Content Area:**
 - Api Documentation:** The main heading.
 - EvaluacionServicio:** The selected endpoint.
 - list:** A GET endpoint with the URL `/api/v1/EvaluacionServicio/`. It includes an "Interact" button and a terminal snippet:

```
# Install the command line client
$ pip install coreapi-cli
```
 - create:** A POST endpoint with the URL `/api/v1/EvaluacionServicio/`. It includes an "Interact" button and a terminal snippet:

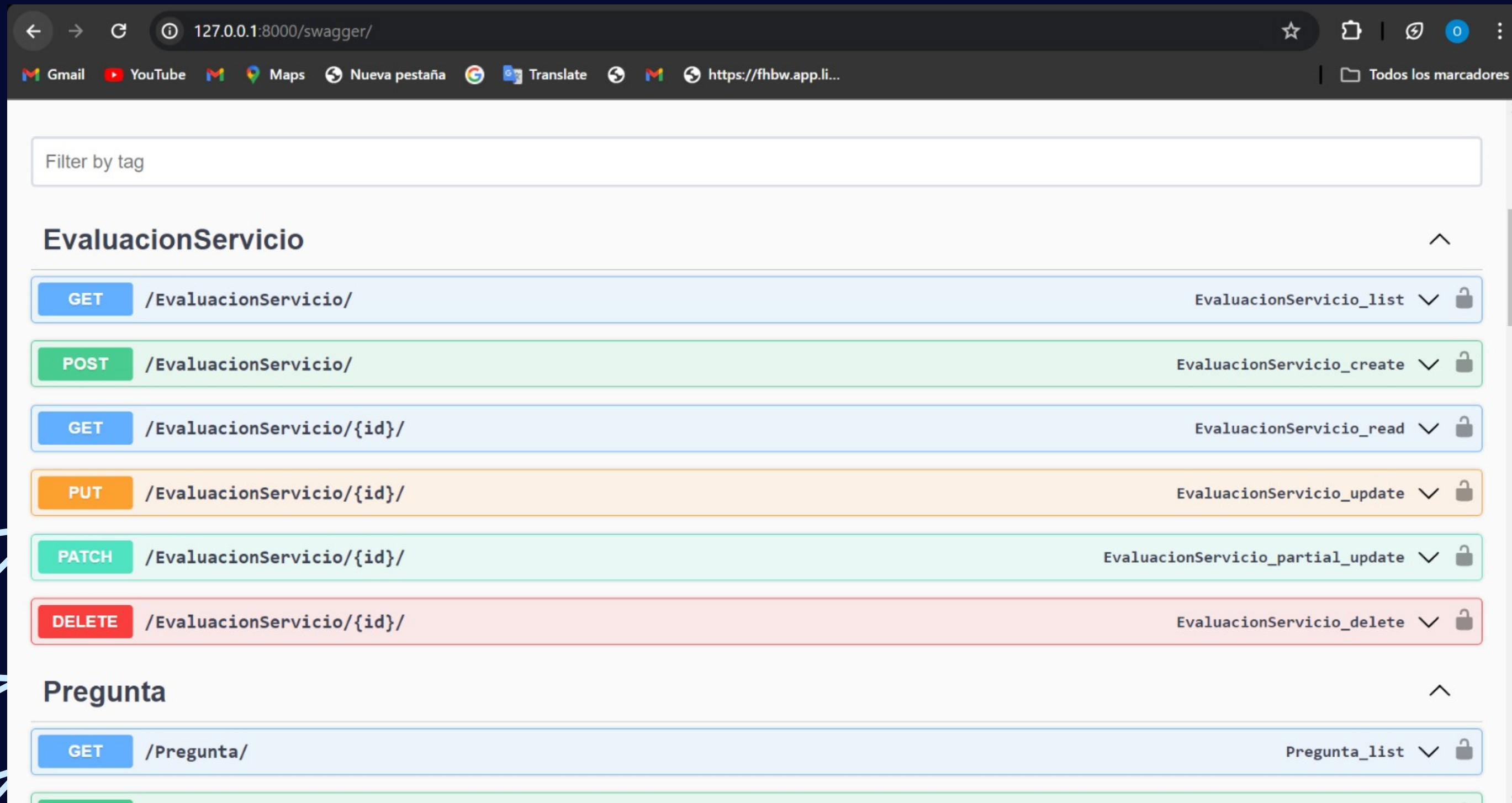
```
# Load the schema document
$ coreapi get http://127.0.0.1:8000/docs/

# Interact with the API endpoint
$ coreapi action EvaluacionServicio list
```
 - Request Body:** A section explaining that the request body should be a "application/json" encoded object, containing the following items:
- Right Sidebar:** Contains a table with two columns: "Parameter" and "Description".

Parameter	Description
<code>servicio_id</code>	
<code>comentarios</code>	

API - REST

DOCUMENTACION CON SWAGGER





GRACIAS