

INTEGRADORA

EQUIPO: DEWEB

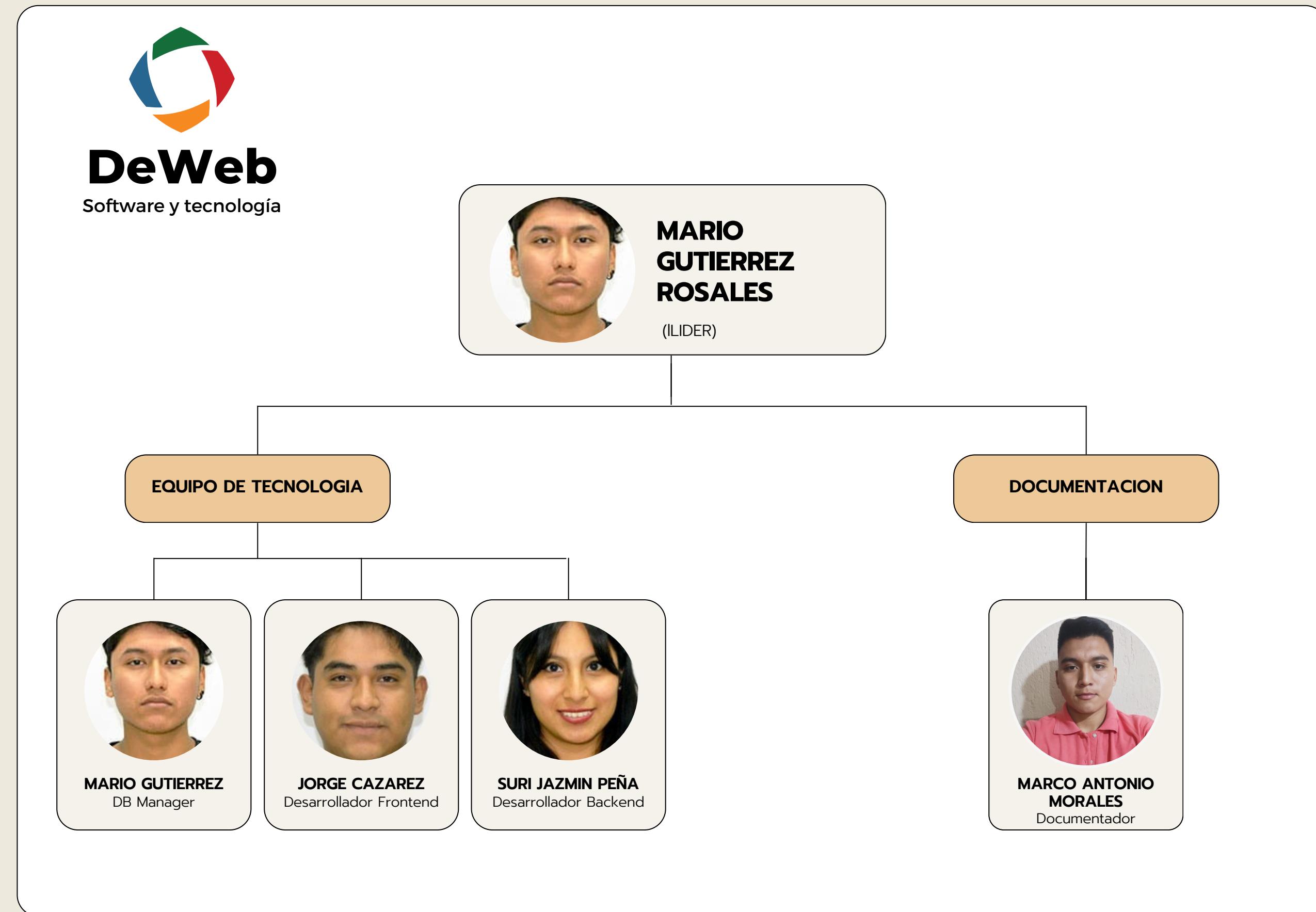
UDN: TRAINING

- SURI JAZMÍN PENA LIRA 210644
- JORGE CRUZ CAZARES 210458
- MARIO GUTIÉRREZ ROSALES 210115
- MARCO ANTONIO MORALES RIVERA
210615



DeWeb
Software y tecnología

ORGANIGRAMA



PROPUESTA DE PLAN DE TRABAJO



UDN: TRAINING

OBJETIVO GENERAL

Desarrollar un módulo de entrenamiento dentro del sistema web de un gimnasio. Este módulo estará destinado a optimizar la gestión integral de la institución, brindando una plataforma digital que mejore la experiencia tanto de los empleados como de los miembros.

Además, se buscará proporcionar herramientas que faciliten la creación, seguimiento y personalización de programas de entrenamiento, rutinas de ejercicios y sesiones específicas para los clientes del gimnasio todo esto con el apoyo de instructores y el uso de una dieta adecuada, promoviendo así la adopción de hábitos saludables y una mejor interacción entre el personal y los usuarios.



OBJETIVOS ESPECIFICOS

- **Inicio de Sesión:** Diseñar una interfaz de inicio de sesión intuitiva y atractiva, segura que permita a los usuarios acceder al sistema web del gimnasio siguiendo el diseño presentado en el mockup.
- **Módulo de Creación de Rutinas:** Crear un formulario de creación de rutinas que permita a los instructores agregar nuevos ejercicios, establecer repeticiones, series y descansos, validando los datos ingresados para evitar errores y garantizar la coherencia de las rutinas creadas.
- **Módulo de Lectura de Rutinas:** Diseñar una interfaz limpia y fácil de entender que muestre las rutinas existentes de forma clara y ordenada, permitiendo a los usuarios ver detalles específicos de cada rutina, como ejercicios incluidos, series y repeticiones.
- **Módulo de Actualización de Rutinas:** Habilitar la edición de rutinas existentes, permitiendo a los instructores realizar cambios en los ejercicios, repeticiones y otros detalles según sea necesario.
- **Módulo de Eliminación de Rutinas:** Integrar una función de eliminación que permita a los instructores eliminar rutinas obsoletas o incorrectas de manera segura y eficiente.
- **Módulo de Programas Saludables:** Diseñar una página que muestre información detallada sobre los programas saludables ofrecidos por el gimnasio, siguiendo el diseño presentado en el mockup. Mostrar tablas con datos relevantes, como nombres de programas, descripciones y beneficios.
- **Dashboard de Seguimiento:** Desarrollar un dashboard interactivo que proporcione a los usuarios una visión general de su progreso en el gimnasio. Incluir gráficos y estadísticas que muestren datos clave, como el rendimiento en las rutinas, la frecuencia de entrenamiento y el cumplimiento de los programas saludables.

REQUERIMIENTOS FUNCIONALES

1. El sistema debe de asignar permisos y accesos específicos a cada rol de usuario.
2. El sistema debe de realizar las siguientes operaciones en el apartado de rutinas: creación, edición, eliminación y actualización.
3. El sistema debe de contener un apartado que muestre la información de los programas saludables disponibles en el gimnasio.
4. El sistema debe de mostrar un monitoreo de los programas de los miembros.
5. El sistema debe realizar operaciones de lectura a tablas externas del departamento de training (Dietas).
6. El sistema debe permitir operaciones CRUD a miembros en el apartado de rutinas.
7. El sistema debe permitir visualización y filtro de programas saludables disponibles para los miembros.
8. El sistema debe reflejar el seguimiento del programa saludable por medio de una gráfica.
9. El sistema debe reflejar el seguimiento del programa saludable al que el usuario este inscrito.
10. El sistema debe mostrar únicamente las vistas que le correspondan al departamento.
11. El sistema tiene que incluir 4 dashboard de monitoreo.
12. El sistema tiene que incluir 2 dashboard con una base de datos SQL.
13. El sistema tiene que incluir 2 dashboard con una base de datos NoSQL.
14. El sistema debe de contener datos dinámicos.
15. El sistema debe de contener un archivo técnico de documentación.

REQUERIMIENTOS NO FUNCIONALES

1. El sistema web debe ser capaz de soportar una gran cantidad de usuarios simultáneos sin afectar el rendimiento.
2. El tiempo de respuesta del sistema debe ser rápido y fluido, además las páginas web deben cargarse rápidamente.
3. El sistema debe ser escalable para poder adaptarse al crecimiento del gimnasio.
4. Se deben implementar medidas de seguridad para evitar el acceso no autorizado a la información, como cifrado de datos.
5. El sistema debe cumplir con las leyes de protección de datos.
6. El sistema web debe ser fácil de usar para todos los usuarios, la interfaz de usuario tiene que ser intuitiva y sencilla.
7. El sistema web debe proteger la información confidencial de los clientes y del personal.
8. Se debe proporcionar una documentación completa del sistema web.
9. El sistema tiene que incluir un archivo documentación con información sobre uso del sistema web.
10. La documentación debe incluir un organigrama de trabajo.

REGLAS DE NEGOCIO

1. Los entrenadores deben comunicarse claramente con los clientes y estar disponibles para discutir su progreso y cualquier inquietud que puedan tener.
2. Cada cliente debe recibir un plan de entrenamiento personalizado según sus objetivos y necesidades específicas.
3. Los entrenadores deben mantenerse actualizados en las últimas tendencias de fitness y técnicas de entrenamiento a través de la capacitación continua.
4. Los entrenadores deben ser conscientes de la diversidad de los clientes y adaptar sus enfoques de entrenamiento según las necesidades individuales.
5. Los entrenadores deben colaborar con otros departamentos del gimnasio, como nutrición, para ofrecer un enfoque integral para el bienestar de los clientes.
6. Se espera que los entrenadores mantengan los estándares de calidad del gimnasio y representen a la marca de manera positiva en todo momento.
7. Los entrenadores deben proporcionar retroalimentación constructiva a los clientes para ayudarles a mejorar su técnica y alcanzar sus objetivos.
8. Los entrenadores deben seguir un código ético que prohíba cualquier tipo de comportamiento inapropiado o abusivo hacia los clientes.
9. Se espera que los entrenadores motiven y alienten a los clientes para que alcancen sus objetivos.

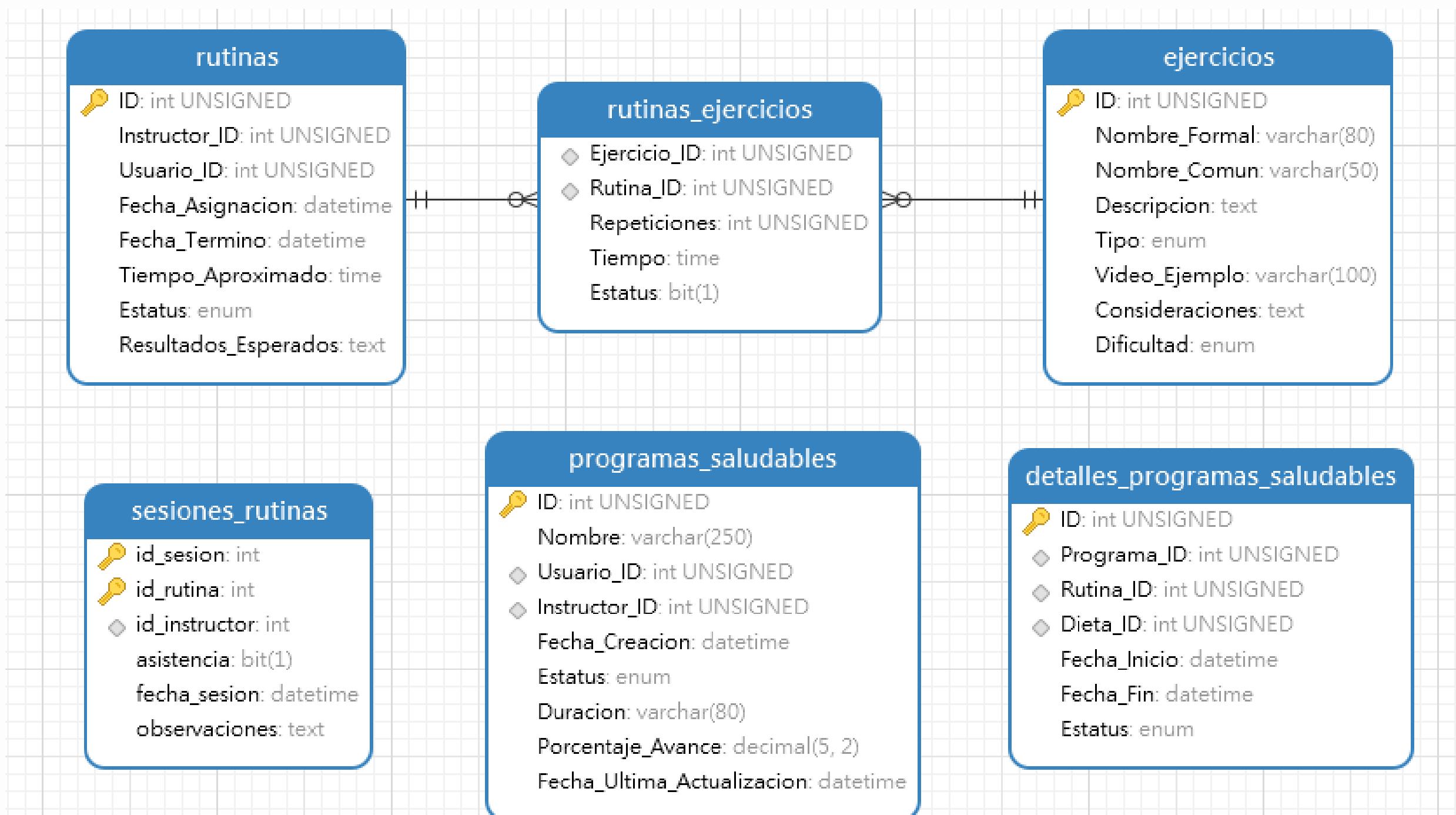
REGLAS DE NEGOCIO

1. Los entrenadores deben ser flexibles para adaptarse a las necesidades cambiantes de los clientes y reprogramar sesiones según sea necesario
2. Los entrenadores deben mantener registros precisos del progreso de cada cliente y ajustar sus programas según sea necesario.
3. Los entrenadores deben respetar la privacidad y confidencialidad de la información personal de los clientes.
4. Se espera que los entrenadores traten a los clientes con respeto y consideración en todo momento, independientemente de su nivel de condición física.
5. Se prohíbe a los miembros del gimnasio participar en discusiones agresivas, peleas o cualquier actividad que perturbe el ambiente de entrenamiento.
6. Se prohíbe a los miembros llevar comida o bebida no autorizada en las áreas de entrenamiento, excepto aquellas permitidas por el gimnasio para su consumo durante el ejercicio.
7. Se prohíbe a los miembros participar en cualquier conducta inapropiada, incluyendo acoso, discriminación, o cualquier comportamiento que haga sentir incómodos a otros usuarios del gimnasio.
8. Los instructores pueden crear y gestionar las clases.
9. Se puede registrar y gestionar las rutinas del gimnasio en el sistema web.
10. Los miembros no pueden tener un dashboard de seguimiento hasta que estén inscritos en un programa saludable.
11. No se pueden realizar operaciones CRUD si nos son instructores verificados.

BD (SQL Y NOSQL)

BD SQL

Se desarrolló una base de datos compuesta por seis tablas, las cuales cuentan con procedimientos y triggers que automatizan el proceso de inserción e historial de actualizaciones y borrado.



BD NOSQL

Se desarrolló una base de datos nosql con una única tabla la cual es **seguimiento_programa**.

```
  [
    {
      _id: "MTS-1",
      fecha: new Date("2024-04-16"),
      usuario: { nombre: "Mario Gutierrez", email: "mario@example.com" },
      instructor: { nombre: "Gabriela Rosales", email: "gabriela@example.com" },
      asistencia: { asistencia: true, estatus: "Iniciada" }
      porcentaje_avance: 90.0,
      tipo: "Seguimiento estándar",
      createdAt: new Date(),
      updatedAt: new Date()
    },
    {
      _id: "MTS-2",
      fecha: new Date("2024-03-16"),
      usuario: { nombre: "Marco Morales", email: "marco@example.com" },
      instructor: { nombre: "Antonio Rivera", email: "antonio@example.com" },
      asistencia: { asistencia: true, estatus: "Iniciada" }
      porcentaje_avance: 85.0,
      tipo: "Seguimiento estándar",
      createdAt: new Date(),
      updatedAt: new Date()
    },
    {
      _id: "MTS-3",
      fecha: new Date("2024-02-16"),
      usuario: { nombre: "Jude Bellingham", email: "jude@example.com" },
      instructor: { nombre: "Bruno Valdez", email: "bruno@example.com" },
      asistencia: { asistencia: true, estatus: "Iniciada" }
      porcentaje_avance: 90.0,
      tipo: "Seguimiento avanzado",
      createdAt: new Date(),
      updatedAt: new Date()
    },
    {
      _id: "MTS-4",
      fecha: new Date("2024-04-16"),
      usuario: { nombre: "Cristiano Ronaldo", email: "cristiano@example.com" },
      instructor: { nombre: "Eduardo Camavinga", email: "eduardo@example.com" },
      asistencia: { asistencia: true, estatus: "Iniciada" }
      porcentaje_avance: 95.0,
      tipo: "Seguimiento avanzado",
      createdAt: new Date(),
      updatedAt: new Date()
    }
  ]
```

DESARROLLO COLABORATIVO

Herramientas de Trabajo

- **Git Hub:** Es una plataforma de desarrollo colaborativo de software que utiliza el sistema de control de versiones Git. Permite a los desarrolladores trabajar juntos en proyectos, realizar seguimiento de cambios en el código, gestionar problemas y realizar solicitudes de extracción.
- **Visual Studio Code:** Es un editor de código fuente, desarrollado por Microsoft.
- **Mysql Worbench:** Herramienta de diseño y administración visual de bases de datos MySQL. Permite modelar bases de datos, crear y gestionar esquemas, ejecutar consultas SQL, y administrar usuarios y permisos.
- **Navicat:** Es una herramienta de administración de bases de datos que proporciona una interfaz gráfica para múltiples sistemas de gestión de bases de datos.
- **Vue.js:** Framework progresivo de JavaScript utilizado para construir interfaces de usuario interactivas y de una sola página.
- **MongoDB:** Es un sistema de base de datos NoSQL orientado a documentos.
- **Paquetería de Office:** En un proyecto de desarrollo de software, estas herramientas pueden ser utilizadas para la documentación del proyecto, la elaboración de informes y la comunicación con los interesados.

SITIO WEB (CRUD)

SITIO WEB (TABLA DINAMICA)

SITIO WEB (DASHBOARD)

PLAN DE SEGURIDAD

Nuestra BD cuenta con los siguientes **roles**:

- **Usuario:** Persona que no es miembro, pero funciona como visitante dentro del gimnasio.
- **Instructor:** Es el responsable de supervisar y dirigir los entrenamientos dentro del gimnasio, así mismo crear, editar y eliminar rutinas.
- **Miembro:** Persona que cuenta con una membresía, con acceso al gimnasio y funciones del sistema.
- **Desarrollador:** Encargado de gestionar el sitio del gimnasio enfocado a la base de datos.

```
CREATE ROLE desarrollador;  
CREATE ROLE instructor;  
CREATE ROLE miembros;  
CREATE ROLE usuario;
```

89	17:32:06	CREATE ROLE desarrollador	0 row(s) affected	0.031 sec
90	17:32:08	CREATE ROLE instructor	0 row(s) affected	0.000 sec
91	17:32:09	CREATE ROLE miembros	0 row(s) affected	0.000 sec
92	17:32:09	CREATE ROLE usuario	0 row(s) affected	0.000 sec

PLAN DE SEGURIDAD

Los **usuarios** de una base de datos desempeñan una variedad de funciones esenciales, desde el acceso y la manipulación hasta la administración de los datos almacenados. Por esta razón, resulta fundamental establecer con precisión el rol específico que se asignará a cada usuario, para mantener la organización y la seguridad de la base de datos.

- Creación de usuarios y asignación de roles:

```
CREATE USER 'mario'@'%' IDENTIFIED BY 'mario123';
CREATE USER 'marco'@'%' IDENTIFIED BY 'toni123';
CREATE USER 'suri'@'%' IDENTIFIED BY 'jaz123';
CREATE USER 'jorge'@'%' IDENTIFIED BY 'jorge123';
```

93	17:32:11	CREATE USER mario@'%'; IDENTIFIED BY mario123;	0 row(s) affected	0.016 sec
94	17:32:12	CREATE USER marco@'%'; IDENTIFIED BY toni123;	0 row(s) affected	0.015 sec
95	17:32:13	CREATE USER suri@'%'; IDENTIFIED BY jaz123;	0 row(s) affected	0.000 sec
96	17:32:13	CREATE USER jorge@'%'; IDENTIFIED BY jorge123;	0 row(s) affected	0.000 sec

PLAN DE SEGURIDAD

En una base de datos, los **privilegios** juegan un papel fundamental para asegurar la gobernanza de las operaciones efectuadas en las distintas tablas. Además, reflejan la integridad de cada usuario, lo que conlleva a la prevención de fallos y al establecimiento de la responsabilidad sobre las actividades ejecutadas por los usuarios.

- Asignación de privilegios:

```
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, ALTER, DROP ON bd_gimnasio_210115.* TO 'mario'@'%';
GRANT SELECT, INSERT, UPDATE, CREATE, DROP, DELETE ON bd_gimnasio_210115.* TO 'marco'@'%';
GRANT SELECT, CREATE, UPDATE, DROP ON bd_gimnasio_210115.* TO 'suri'@'%';
GRANT SELECT ON bd_gimnasio_210115.* TO 'jorge'@'%';
```

97	17:32:16	GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, ALTER, DROP ON bd_gim... 0 row(s) affected	0.000 sec
98	17:32:17	GRANT SELECT, INSERT, UPDATE, CREATE, DROP, DELETE ON bd_gimnasi... 0 row(s) affected	0.000 sec
99	17:32:18	GRANT SELECT, CREATE, UPDATE, DROP ON bd_gimnasio_210115.* TO 'sur... 0 row(s) affected	0.000 sec
100	17:32:19	GRANT SELECT ON bd_gimnasio_210115.* TO 'jorge'@'%' 0 row(s) affected	0.016 sec

PLAN DE SEGURIDAD

Creación de roles con privilegios en **NoSQL**:

```
> db.createRole({
  role: "admin_seguimiento",
  privileges: [
    { resource: { db: "seguimiento_programa", collection: "" }, actions: ["anyAction"] }
  ],
  roles: []
})
< { ok: 1 }

> db.createRole({
  role: "escritura",
  privileges: [
    { resource: { db: "seguimiento_programa", collection: "" }, actions: ["insert", "update", "remove"] }
  ],
  roles: []
})
< { ok: 1 }

> db.createRole({
  role: "lectura",
  privileges: [
    { resource: { db: "seguimiento_programa", collection: "" }, actions: ["find"] }
  ],
  roles: []
})
< { ok: 1 }
```

PLAN DE SEGURIDAD

Creación de usuarios y asignación de roles en **NoSQL**:

```
> db.createUser({  
    user: "desarrollador",  
    pwd: "1234",  
    roles: ["admin_seguimiento"]  
})  
< { ok: 1 }  
> db.createUser({  
    user: "instructor",  
    pwd: "1234",  
    roles: ["escritura"]  
})  
< { ok: 1 }  
> db.createUser({  
    user: "usuario",  
    pwd: "1234",  
    roles: ["lectura"]  
})  
< { ok: 1 }  
> db.createUser({  
    user: "miembro",  
    pwd: "1234",  
    roles: ["escritura"]  
})  
< { ok: 1 }
```

PLAN DE SEGURIDAD

CALENDARIO DE RESPALDOS: Los datos de nuestra base tanto sql y nosql se respaldan de la siguiente manera:

- 1.Copia de Seguridad Completa (Manual):** Esta es la forma más sencilla y completa de respaldo. Incluye todos los datos en la base de datos, así como parte del registro de transacciones para asegurar la consistencia de la base de datos.
- 2.Copia de Seguridad Diferencial:** Solo respalda los cambios realizados desde la última copia de seguridad completa. Esto reduce el tiempo y el espacio de almacenamiento necesario para los respaldos.
- 3.Copia de Seguridad de Registro de Transacciones:** Específica para bases de datos con un modelo de recuperación completa o con registro de operaciones masivas. Permite restaurar la base de datos a un punto específico en el tiempo.

Los respaldos mencionados anteriormente deben ser aplicados o considerados para nuestras tablas de mayor relevancia dentro del esquema de trabajo, las tablas a considerar son:

- Programas Saludables
- Rutinas
- Ejercicios
- Sesiones Rutinas

Respecto a las demás tablas sobrantes el respaldo es importante, pero, se puede realizar periódicamente.

PLAN DE SEGURIDAD

La localización de los **backups** puede variar según la configuración y las preferencias del administrador de la base de datos. Algunas de las opciones comunes para almacenar los backups son:

- 1. Disco Local:** Se puede almacenar los backups en el mismo servidor donde reside la BD.
- 2. Disco Remoto:** Utilizar una ubicación de red para almacenar los backups.
- 3. Almacenamiento en la Nube:** Servicios como Azure Blob Storage ofrecen una opción segura y escalable para almacenar backups fuera del sitio.

Evaluación de respaldos

Es esencial realizar evaluaciones regulares de cada método de respaldo para confirmar la legibilidad de los datos. En ocasiones, se producen respaldos que resultan ser ilegibles por distintos motivos. El problema radica en que, a menudo, la ilegibilidad de estos respaldos pasa inadvertida hasta que ocurre una pérdida de datos y se requiere su restauración.

Diversos factores pueden causar este problema, incluyendo desajustes en los cabezales de las unidades de cinta, configuraciones erróneas en el software de respaldo o errores humanos. Independientemente de la causa, la ausencia de revisiones regulares impide garantizar la creación de respaldos fiables de los cuales se puedan recuperar los datos en el futuro.

DESARROLLO DE API



DJANGO

127.0.0.1:8000/admin/

Administración de Django

Administración del sitio

API

- Detalles_programas_saludables [Agregar](#) [Modificar](#)
- Ejercicios [Agregar](#) [Modificar](#)
- Programas_saludables [Agregar](#) [Modificar](#)
- Rutinas_ejercicios [Agregar](#) [Modificar](#)
- Rutinas [Agregar](#) [Modificar](#)
- Sesiones_rutinas [Agregar](#) [Modificar](#)

AUTENTICACIÓN Y AUTORIZACIÓN

- Grupos [Agregar](#) [Modificar](#)
- Usuarios [Agregar](#) [Modificar](#)

CUENTAS

- Correos electrónicos [Agregar](#) [Modificar](#)

CUENTAS DE REDES SOCIALES

- Aplicaciones de redes sociales [Agregar](#) [Modificar](#)
- Cuentas de redes sociales [Agregar](#) [Modificar](#)
- Tokens de aplicación de redes sociales [Agregar](#) [Modificar](#)

Acciones recientes

Mis acciones

Ninguna disponible

127.0.0.1:8000/docs/

gimnasio API'S

v1detalle_ps

list [Interact](#)

GET /gimnasio/api/v1detalle_ps/

create [Interact](#)

POST /gimnasio/api/v1detalle_ps/

Request Body

The request body should be a "application/json" encoded object, containing the following items.

Parameter	Description
programa_id <small>required</small>	
rutina_id <small>required</small>	
dieta_id <small>required</small>	
fecha_inicio <small>required</small>	
fecha_fin <small>required</small>	
estatus <small>required</small>	

read [Interact](#)

GET /gimnasio/api/v1detalle_ps/{id}/

Path Parameters

The following parameters should be included in the URL path.

Parameter	Description
id <small>Required</small>	A unique integer value identifying this detalles_programas_saludables.

Install the command line client
\$ pip install coreapi-cli

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

Interact with the API endpoint
\$ coreapi action v1detalle_ps list

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

Interact with the API endpoint
\$ coreapi action v1detalle_ps create -p programa_id=... -p rutina_id=... -p dieta_id=... -p fecha_inicio=... -p fecha_fin=...

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

Interact with the API endpoint
\$ coreapi action v1detalle_ps read -p id=...

¡MUCHAS GRACIAS!