FitTrack – Documentación Frontend Oficial

Autor: Sergio Javier Lorente Guerra

Índice

Documentación front Fittrack

Introduccion

Esta documentación describe el funcionamiento del frontend de la aplicacion FitTrack. La aplicación esta desarrollada usando Angular junto con Ionic, y se conecta a una API RESTful desarrollada en Spring Boot. El objetivo de la app es permitir a entrenadores y usuarios gestionar rutinas de entrenamiento divididas por bloques, semanas, subbloques, ejercicios y series. También incluye una sección de graficas para visualizar la evolución del peso en los ejercicios.

Estructura general del proyecto

El proyecto está organizado por páginas y servicios. Cada página corresponde a una vista concreta y tiene su propio archivo HTML, SCSS y TS. Los servicios se usan para conectarse con el backend a través de endpoints protegidos con JWT.

Las páginas principales son:

Login y registro

Vista de bloques (entrenador / usuario)

Vista de semanas

Vista de subbloques

Vista de ejercicios

Vista de series

Vista de estadisticas (grafica)

Además hay componentes compartidos como el menu inferior.

Login y registro

login.page.ts

Esta página permite al usuario o entrenador autenticarse. El formulario tiene campos para email y contraseña. Cuando se pulsa el botón de login, se llama al método login() que usa el servicio authService.login() para enviar las credenciales al backend.

Si el rol que devuelve el backend es "admin" (entrenador), se redirige a la vista de bloques del admin. Si es "usuario", se redirige a su vista personalizada.

FitTrack Frontend – Documentación Oficial

También se guarda el token en localStorage.

login.page.html

Es un formulario simple con campos de entrada y un botón. También tiene un enlace para ir al registro si el usuario no tiene cuenta.

register.page.ts

Permite registrar a un nuevo usuario o entrenador. El formulario incluye nombre, email, contraseña y rol. Se llama a authService.register().

register.page.html

Es similar al login pero con un campo adicional para elegir el rol.

Vista de bloques (admin)

admin-bloques.page.ts

Esta página carga todos los bloques disponibles para un entrenador. Se muestran en tarjetas, cada una con su nombre. Hay botones para editar o eliminar cada bloque y otro para crear uno nuevo.

Los bloques se obtienen usando adminBloquesService.obtenerBloquesDelEntrenador().

Cuando se elimina un bloque, aparece un modal de confirmación.

admin-bloques.page.html

Se renderiza una lista de tarjetas con el nombre del bloque y botones para editar y eliminar. También hay un botón + para crear uno nuevo.

crear-bloque.page.ts y editar-bloque.page.ts

Ambas paginas son formularios simples. En el caso de crear, se llama a adminBloquesService.crearBloque(), y en el de editar, a adminBloquesService.editarBloque().

Vista de semanas (admin)

admin-semanas.page.ts

Esta página muestra todas las semanas dentro de un bloque. Cada tarjeta tiene el número de semana y botones para editar o eliminar. También hay un botón para crear una nueva semana.
Al iniciar la vista, se recupera el ID del bloque desde la URL y se llama al método cargarSemanas(), que obtiene los datos del backend usando adminBloquesService.obtenerSemanasDeBloque().

Al pulsar en una semana se navega a la vista de subbloques.
La eliminación de semanas también muestra un modal de confirmación.

admin-semanas.page.html

Se muestra una lista de tarjetas con el número de semana.
Cada una tiene los botones "Editar" y "Eliminar".
También hay un botón flotante con el icono + que lleva a la pantalla de creación.

crear-semana.page.ts y editar-semana.page.ts

Ambas páginas contienen un formulario para escribir el número de la semana.
En crear se usa adminBloquesService.crearSemana(), y en editar, adminBloquesService.editarSemana().

crear-semana.page.html y editar-semana.page.html

Formulario simple con un campo numérico para ingresar el número de semana.
Hay botones para confirmar o cancelar la acción.

Vista de subbloques (admin)

admin-subbloques.page.ts

Aquí se cargan todos los subbloques de una semana. Se muestra una tarjeta por subbloque, con su nombre y botones para editar o eliminar.
Se usa el servicio adminBloquesService.obtenerSubbloquesDeSemana() para obtener los datos.
Cuando se pulsa un subbloque se accede a los ejercicios sugeridos que tiene.

El botón "+" abre el formulario para crear un nuevo subbloque, pasándole por URL el semanaId.

admin-subbloques.page.html

Se renderiza la lista de subbloques en tarjetas, con botones para editar o eliminar.
Se usa una lógica similar a la de semanas.

crear-subbloque.page.ts y editar-subbloque.page.ts

El archivo de creación recoge el semanaId desde la URL y crea un subbloque con adminBloquesService.crearSubbloque().
En el de edición, se obtiene también el nombre y id del subbloque y se llama a adminBloquesService.editarSubbloque() con el nuevo nombre.

crear-subbloque.page.html y editar-subbloque.page.html

Cada vista tiene un campo de texto para ingresar o modificar el nombre, y botones para guardar o cancelar.

Vista de ejercicios (admin)

admin-ejercicios.page.ts

Esta vista muestra todos los ejercicios sugeridos que hay dentro de un subbloque. Se obtienen llamando a adminBloquesService.obtenerEjerciciosSugeridos() pasando el subbloqueId desde la URL.
Cada ejercicio se muestra en una tarjeta con su nombre y grupo muscular, además de dos botones para editar o eliminar.

Cuando se pulsa sobre una tarjeta, se abre la vista de series sugeridas para ese ejercicio.
El botón "+" en la cabecera permite crear un nuevo ejercicio.

También hay lógica para mostrar un modal de confirmación antes de eliminar un ejercicio. Si se confirma, se llama a adminBloquesService.eliminarEjercicioSugerido().

admin-ejercicios.page.html

Contiene un ngFor que recorre la lista de ejercicios y crea una tarjeta con:

El nombre del ejercicio.

El grupo muscular con un icono.

Botones "Editar" y "Eliminar".

El click en la tarjeta llama a verSeries().

El botón "+" abre el formulario de creación.
También hay un bloque de estado vacío si no hay ejercicios, con un mensaje animando a crear uno.

crear-ejercicio.page.ts

Permite añadir un nuevo ejercicio al subbloque.
Usa el ID del subbloque desde los parámetros y envía los datos del nuevo ejercicio (nombre, grupoMuscular) al backend usando adminBloquesService.crearEjercicioSugerido().

Después de crearlo, redirige automáticamente a la vista de ejercicios del mismo subbloque.

crear-ejercicio.page.html

Formulario muy simple con dos campos:

Nombre del ejercicio.

Grupo muscular.

Botón "Crear" para guardar y botón "Cancelar" para volver.

editar-ejercicio.page.ts

Muy similar al de crear, pero en este caso se cargan los datos existentes del ejercicio para modificarlos.
Se llama al método adminBloquesService.editarEjercicioSugerido() pasando el ejercicioId y los nuevos datos.
También se redirige al final de nuevo a la vista de ejercicios.

editar-ejercicio.page.html

Mismo formulario que el de creación, pero adaptado al contexto de edición.

Vista de series (admin)

series-sugeridas.page.ts

Esta página muestra todas las series sugeridas asociadas a un ejercicio específico.
Recibe el ejercicioId como parámetro desde la URL. En el ngOnInit se llama a adminBloquesService.obtenerSeriesSugeridas() para cargar todas las series.
Cada serie contiene los campos numeroSerie, repeticiones y peso, que pueden editarse directamente desde la lista.

Además:

Se pueden editar las series existentes usando editarSerieSugerida().

Se pueden eliminar usando un botón que abre un modal de confirmación.

Se pueden crear nuevas series con un formulario en la parte inferior que usa crearSerieSugerida().

También hay manejo del modal de confirmación con mostrarModalConfirmacion, serieIdAEliminar, y los métodos cancelarEliminacion() y confirmarEliminacion().

Al eliminar o editar una serie, se recarga toda la lista usando cargarSeries() para tener los datos actualizados.

series-sugeridas.page.html

Se estructura en varias partes:

Una lista ion-list con un ngFor para cada serie ya creada.
Cada item tiene:

Inputs para modificar el número de serie, repeticiones y peso.

Botón para guardar cambios.

Botón para eliminar (abre el popup de confirmación).

Un bloque visual al final con el formulario de creación de nueva serie.
Tiene tres campos para número de serie, repeticiones y peso, y un botón “AÑADIR”.

Un popup de confirmación que se muestra cuando se va a eliminar una serie.
Este se maneja mediante clases CSS (.popup-overlay.visible) y botones de cancelar y confirmar.

El botón de cerrar redirige de nuevo a la vista de ejercicios del admin, manteniendo el subbloqueId en los parámetros.

Vista de bloques (usuario)

usuario-bloques.page.ts

Esta página es la versión del listado de bloques para un usuario logueado.
Se carga desde el controlador al iniciar sesión si el rol es "usuario".
En el ngOnInit se extrae el usuarioId desde los parámetros (queryParamMap) y se hace una llamada a usuarioService.obtenerBloquesDelUsuario(usuarioId).

La respuesta es una lista de bloques que han sido asignados por el entrenador. Cada bloque contiene su id y nombre.

Cuando el usuario hace clic en un bloque, se navega a la vista de semanas correspondientes, pasando también el usuarioId y el bloqueId como queryParams.

usuario-bloques.page.html

El HTML es muy parecido al del admin:

Se usa un *ngFor para mostrar las tarjetas de cada bloque.

En cada tarjeta se muestra el nombre del bloque.

No hay botones de editar ni eliminar.

Todo el div actúa como botón, llamando al método verSemanas(bloque.id) al hacer clic.

Además, en la parte inferior está el componente <app-menu-inferior>, que es común a varias vistas del usuario.

Vista de semanas (usuario)

usuario-semanas.page.ts

Esta página muestra las semanas asignadas al usuario para un bloque concreto.
Al inicializar (ngOnInit), se recogen dos parámetros de la URL: usuarioId y bloqueId.

Con esos datos se llama a usuarioService.obtenerSemanasDelBloque(usuarioId, bloqueId), que devuelve un array de semanas con id y numero.

Cada semana se muestra como una tarjeta. Al hacer clic en una de ellas, se navega a la vista de subbloques, pasando también los parámetros usuarioId, bloqueId y semanaId.

El código también incluye un botón para volver atrás, que redirige a la vista de bloques del usuario (/usuario-bloques) con el usuarioId.

usuario-semanas.page.html

Se usa un *ngFor para recorrer las semanas.

Cada tarjeta muestra "Semana X", donde X es el número.

El contenedor entero actúa como botón para acceder a los subbloques.

Hay un botón con un icono de flecha para volver a la vista anterior.

También se incluye <app-menu-inferior> para mantener la navegación inferior visible.

Vista de subbloques (usuario)

usuario-subbloques.page.ts

Esta página muestra todos los subbloques asociados a una semana concreta asignada al usuario.

Se recuperan los parámetros usuarioId, bloqueId y semanaId desde la URL (ActivatedRoute).
Con esos datos se llama a usuarioService.obtenerSubBloquesDeSemana(usuarioId, semanaId).

El resultado es una lista de subbloques, cada uno con su id y nombre.

Cuando el usuario hace clic en un subbloque, se navega a la vista de ejercicios sugeridos, pasando usuarioId, bloqueId, semanaId y subbloqueId como parámetros de navegación.

También hay un botón para volver a la vista de semanas del usuario (/usuario-semanas), manteniendo los parámetros usuarioId y bloqueId.

usuario-subbloques.page.html

Usa *ngFor para mostrar los subbloques en tarjetas.

Cada tarjeta muestra el nombre del subbloque.

Al hacer clic en una tarjeta, se invoca verEjercicios() con el subbloqueId.

Hay un botón con una flecha para volver a la vista anterior (semanas).

Se incluye el componente <app-menu-inferior> como en las demás vistas para mantener la navegación.

Vista de ejercicios sugeridos (usuario)

usuario-ejercicios.page.ts

Esta vista muestra los ejercicios que el entrenador ha sugerido para un subbloque concreto. Se reciben por URL los parámetros usuarioId, bloqueId, semanaId y subbloqueId.

El método ngOnInit() los extrae y luego se llama a usuarioService.obtenerEjerciciosSugeridos(subbloqueId) para cargar los ejercicios desde el backend.

Cada ejercicio muestra su nombre y grupoMuscular, y tiene dos botones:

Uno para ver las series sugeridas (verSeries()), que redirige a /usuario-series pasando el ejercicioId y ejercicioUsuarioId como parámetros.

Otro para añadir el ejercicio sugerido como propio, invocando añadirEjercicio(), que llama al endpoint POST /api/usuarios/{usuarioId}/subbloques/{subbloqueId}/ejercicios.

Si se añade correctamente, se muestra un mensaje de éxito. Si falla, se muestra un error.

Hay un botón de volver que redirige a la vista de subbloques del usuario.

usuario-ejercicios.page.html

Cada tarjeta muestra:

El nombre del ejercicio.

El grupo muscular con un emoji .

Botones: "Ver series" y "Añadir ejercicio".

Hay un mensaje de estado (éxito o error) que aparece según la acción realizada.

Botón de regreso a la vista de subbloques con volverASubbloques().

También se incluye <app-menu-inferior> como en las demás vistas.

Vista de series del usuario

usuario-series.page.ts

Esta vista permite al usuario gestionar las series reales (entrenamiento propio) asociadas a un ejercicio que ha añadido desde los sugeridos.

Se reciben por URL los siguientes parámetros:

usuarioId

subbloqueId

ejercicioId

ejercicioUsuarioId ← este es el identificador del ejercicio personalizado creado por el usuario.

Lógica general:

En ngOnInit() se extraen los parámetros y se llama a cargarSeries() para obtener las series del ejercicio del usuario.

La llamada se hace a usuarioService.obtenerSeriesUsuario(ejercicioUsuarioId), que accede al endpoint del backend correspondiente.

Cada serie se puede editar directamente cambiando sus campos (numeroSerie, repeticiones, peso) y pulsando el botón .

También se pueden eliminar con , lo cual abre un modal de confirmación antes de eliminarla.

La creación de una nueva serie se hace desde los campos de entrada en la parte inferior, pulsando "AÑADIR".

Modal de confirmación:

Igual que en las vistas del admin, se muestra un popup si el usuario pulsa eliminar. Desde ahí puede cancelar o confirmar la acción. Si la confirma, se llama al endpoint de eliminación del backend (usuarioService.eliminarSerieUsuario()).

usuario-series.page.html

Lista de ion-item donde cada uno muestra:

Inputs para número de serie, repeticiones y peso.

Botón para guardar los cambios.

Botón para abrir el modal de confirmación.

Abajo hay un bloque separado con inputs para crear una nueva serie.

También se incluye el popup de confirmación centrado como en otras vistas.

El botón "CERRAR" en la cabecera llama a cerrarModal() y redirige de nuevo a la vista de ejercicios.

Vista de estadísticas (usuario)

estadisticas.page.ts

Esta vista permite al usuario visualizar gráficamente la evolución del peso registrado en sus ejercicios personalizados a lo largo del tiempo.

Lógica general:

Al entrar, se cargan dinámicamente los elementos de filtro:

Bloques asignados al usuario (usuarioService.obtenerBloquesUsuario()).

Al seleccionar un bloque, se cargan sus semanas (usuarioService.obtenerSemanasUsuario(bloqueId)).

Luego los subbloques (usuarioService.obtenerSubbloquesUsuario(semanaId)).

Finalmente, los ejercicios (usuarioService.obtenerEjerciciosUsuario(subbloqueId)).

El usuario debe elegir uno en cada nivel para poder generar la gráfica.

Una vez seleccionados todos los niveles, se llama a:

usuarioService.obtenerDatosGrafica(usuarioId, ejercicioUsuarioId)

Este endpoint devuelve un array de puntos con:

Fecha

Peso correspondiente en esa fecha

Estos datos se transforman y se envían al componente Chart.js para renderizar una línea de evolución del peso.

Validaciones:

Si el usuario no ha seleccionado todos los campos, el botón de "Generar gráfica" no hace nada. También se muestra un mensaje si no hay datos para ese ejercicio.

estadisticas.page.html

Cuatro ion-select que muestran los niveles jerárquicos: bloque, semana, subbloque, ejercicio.

Un botón "Generar gráfica".

Si hay datos, se muestra una gráfica de líneas (usando el componente de Chart.js embebido).

Si no hay datos o no se ha seleccionado nada, aparece un mensaje informativo.

También se incluye el menú inferior como en otras vistas.

Vista de gráfica (admin)

Esta vista permite al entrenador visualizar la evolución del peso que han manejado sus usuarios en ejercicios concretos. Es una sección importante para hacer seguimiento del progreso y evaluar si los entrenamientos están funcionando como se esperaba.

Estructura y navegación

La página se accede desde el menú inferior (cuando se es entrenador) mediante un botón con el icono de gráfico . Una vez dentro, se muestra una interfaz con varios selectores desplegables: primero se elige el bloque, luego la semana, el subbloque y por último el ejercicio. Esta selección jerárquica permite filtrar progresivamente la información hasta llegar al ejercicio concreto.

Comportamiento

Cuando se selecciona un ejercicio, se lanza una petición al backend mediante el servicio adminBloquesService.obtenerDatosGrafica(), que devuelve los datos necesarios para construir la gráfica. Estos datos son las fechas y los pesos registrados en las series de ese ejercicio por parte de los usuarios.

La gráfica se renderiza con Chart.js, una librería muy popular para representar datos. En este caso, se utiliza una línea temporal que representa la evolución del peso a lo largo del tiempo. Cada punto representa una serie registrada por un usuario en un día determinado.

Diseño

La vista es sencilla pero clara. Los selectores están bien espaciados y la gráfica ocupa la parte central. Si no hay datos disponibles (porque aún no se han registrado series del ejercicio seleccionado), se muestra un mensaje indicándolo. Esto evita que la gráfica se quede vacía sin explicación.

Además, los colores y líneas de la gráfica están pensados para ser legibles tanto en modo claro como en modo oscuro (según el tema que tenga el sistema operativo del usuario).

Limitaciones y detalles

Solo se muestran los datos de ejercicios que hayan sido registrados al menos una vez por los usuarios.

La gráfica es puramente visual: no permite editar ni interactuar con los datos.

Cada vez que se cambia un selector, se actualiza todo el gráfico.

Componente compartido: Menú inferior

El menú inferior es un componente reutilizable que aparece en casi todas las vistas principales de la app, tanto para usuarios como para entrenadores. Su función es permitir la navegación rápida entre las secciones clave.

El componente detecta el rol del usuario actual usando la información almacenada en localStorage. En función de ese rol, muestra los accesos correspondientes. Por ejemplo, si el rol es "usuario", aparecen botones para acceder a los bloques personales, a la sección de estadísticas (gráfica) y al perfil. Si el rol es "admin", se muestran los botones adecuados para la gestión de bloques del entrenador.

La navegación entre secciones se realiza mediante el router de Angular. Cada botón del menú inferior lleva a una ruta específica al pulsarlo. Aunque el diseño recuerda al sistema de tabs de Ionic, este componente no usa el sistema de pestañas como tal, sino que es un menú personalizado que se ha situado en la parte inferior de cada vista manualmente.

Visualmente, los botones utilizan emojis como íconos (por ejemplo, una caja para bloques, un gráfico para estadísticas y una cara para el perfil), lo cual le da un toque más amigable y directo a la interfaz. El menú está estilizado para permanecer fijo en la parte inferior de la pantalla y tener buena separación entre los elementos.

Para incluirlo, simplemente se agrega al final del HTML de cualquier vista con la etiqueta correspondiente.

Servicios del frontend

Los servicios en Angular se utilizan para abstraer las llamadas HTTP hacia el backend. En FitTrack, cada servicio tiene una responsabilidad concreta y permite acceder a los endpoints protegidos mediante el token JWT guardado en localStorage tras el login. Todos los servicios se encuentran en la carpeta src/app/shared/services.

login.service.ts

Este servicio gestiona el inicio de sesión. Expone un único método principal:

login(email: string, password: string):
Hace una llamada POST al endpoint /api/auth/login enviando el email y la contraseña. Si las credenciales son correctas, el backend devuelve un token JWT y los datos del usuario (incluido el rol).
Este token se guarda en localStorage y será usado por los demás servicios para autenticar las peticiones.
En caso de error (por ejemplo, contraseña incorrecta), devuelve un error manejable desde la vista.

Este servicio se utiliza exclusivamente en la página login.page.ts.

register.service.ts

Encargado del registro de nuevos usuarios o entrenadores. Tiene también un método principal:

register(nombre: string, email: string, password: string, rol: string):
Hace un POST a /api/auth/register con los datos del formulario.
El backend crea el nuevo usuario y devuelve un token JWT al igual que en el login.
El token también se guarda en localStorage automáticamente.

Este servicio se usa únicamente en la página register.page.ts.

bloques.service.ts (servicio de usuario)

Este servicio se usa para obtener la información de entrenamiento del usuario normal (rol "usuario"). Se conecta a los endpoints de usuario y permite leer la información asociada a su rutina personalizada.

Principales métodos:

obtenerBloquesDelUsuario()
Llama a /api/usuarios/bloques para traer los bloques asignados al usuario.

obtenerSemanasPorBloque(bloqueId: number)
Hace una petición a /api/usuarios/bloques/{bloqueId}/semanas y devuelve las semanas del bloque seleccionado.

obtenerSubbloquesPorSemana(semanaId: number)
Llama al endpoint /api/usuarios/semanas/{semanaId}/subbloques para obtener los subbloques disponibles.

obtenerEjerciciosPorSubbloque(subbloqueId: number)
Consulta los ejercicios sugeridos del subbloque (GET
/api/usuarios/subbloques/{subbloqueId}/ejercicios).

agregarEjercicioAPersonales(usuarioId, subbloqueId, ejercicioData)
Permite al usuario copiar un ejercicio sugerido a su lista personal.
Se hace con un POST a /api/usuarios/{usuarioId}/subbloques/{subbloqueId}/ejercicios.

obtenerEjerciciosPersonales(subbloqueId: number)
Devuelve todos los ejercicios personalizados del usuario para ese subbloque.
Llama a /api/usuarios/subbloques/{subbloqueId}/ejercicios-personales.

Este servicio se usa en todas las vistas del usuario: bloques, semanas, subbloques, ejercicios personales y lectura general.

admin-bloques.service.ts (servicio del entrenador)

Este servicio es el más grande, porque cubre todas las acciones que un entrenador puede hacer sobre bloques, semanas, subbloques, ejercicios sugeridos y series sugeridas. Básicamente, cualquier cosa que sea gestión (crear, editar, borrar) pasa por aquí.

Métodos principales:

 Bloques

obtenerBloquesDelEntrenador()
Hace un GET a /api/admin/bloques y devuelve todos los bloques creados por ese entrenador.

crearBloque(nombre: string)
Llama a POST /api/admin/bloques para crear un nuevo bloque.

editarBloque(bloqueId, nuevoNombre)
Hace un PUT /api/admin/bloques/{bloqueId} para actualizar el nombre.

eliminarBloque(bloqueId)
Petición DELETE a /api/admin/bloques/{bloqueId}.

 Semanas

obtenerSemanasPorBloque(bloqueId)
Llama a /api/admin/bloques/{bloqueId}/semanas y devuelve todas las semanas de ese bloque.

crearSemana(bloqueId, numeroSemana)
POST /api/admin/semanas con el bloque y el número.

editarSemana(semanaId, numeroSemana)
PUT /api/admin/semanas/{semanaId} para cambiar el número.

eliminarSemana(semanaId)
DELETE /api/admin/semanas/{semanaId}.

 Subbloques

obtenerSubbloquesPorSemana(semanaId)
Devuelve los subbloques de una semana concreta (GET /api/admin/semanas/{semanaId}/subbloques).

crearSubbloque(semanaId, nombre)
POST /api/admin/subbloques con la semana y el nombre.

editarSubbloque(subbloqueId, nuevoNombre)
PUT /api/admin/subbloques/{subbloqueId}.

eliminarSubbloque(subbloqueId)
DELETE /api/admin/subbloques/{subbloqueId}.

 Ejercicios sugeridos

obtenerEjerciciosSugeridos(subbloqueId)
GET /api/admin/subbloques/{subbloqueId}/ejercicios.

crearEjercicioSugerido(subbloqueId, nombre, grupoMuscular)
POST /api/admin/ejercicios-sugeridos.

editarEjercicioSugerido(ejercicioId, nombre, grupoMuscular)
PUT /api/admin/ejercicios-sugeridos/{ejercicioId}.

eliminarEjercicioSugerido(ejercicioId)
DELETE /api/admin/ejercicios-sugeridos/{ejercicioId}.

 Series sugeridas

obtenerSeriesSugeridas(ejercicioId)
GET /api/admin/ejercicios-sugeridos/{ejercicioId}/series.

crearSerieSugerida(ejercicioId, datosSerie)
POST /api/admin/series-sugeridas.

editarSerieSugerida(serieId, datosActualizados)
PUT /api/admin/series-sugeridas/{serieId}.

eliminarSerieSugerida(serieId)
DELETE /api/admin/series-sugeridas/{serieId}.

Este servicio se usa en absolutamente todas las vistas del entrenador: bloques, semanas, subbloques, ejercicios y series.

grafica.service.ts

Este servicio es el que se encarga de consultar los datos necesarios para la gráfica de evolución de peso. Está conectado tanto en la vista del usuario como del admin.

Métodos disponibles:

obtenerEjerciciosPersonalesPorSubbloque(subbloqueId: number)
Llama a /api/grafica/ejercicios-personales?subbloqueId=X
Sirve para mostrar el selector de ejercicios disponibles con series reales para ese subbloque.

obtenerDatosGrafica(ejercicioUsuarioId: number)
Llama a /api/grafica/datos?ejercicioUsuarioId=X
Este endpoint devuelve los puntos que se dibujan en el gráfico, con fecha y peso.

Se usa únicamente en la vista de gráficas (grafica.page.ts y .html).