

PROYECTO TFG

CICLO: Desarrollo de aplicaciones Web

Autor: Leandro Manuel Ribeiro Mouro

Tutor: Markel Sancho

Curso académico 2024-2025

TABLA DE CONTENIDO

JUSTIFICACIÓN	3
INTRODUCCIÓN.....	4
OBJETIVO.....	5
DESARROLLO	5
Tecnologías por usar	5
Orden de desarrollo	6
Justificación de Estructura de Frontend	8
Justificación de Estructura de Backend.....	9
Desarrollo de proyecto	10
Creación Base de datos.....	10
Desarrollo de Backend	11
Desarrollo de Frontend	14
Scene: HOME	15
Scene: REGISTRO.....	16
Scene: LOGIN.....	17
Scene: Dashboard Cliente	17
Scene: Dashboard Entrenador	18
CONCLUSION.....	20

JUSTIFICACIÓN

La sociedad se vio involucrada en las últimas décadas a llevar a la tecnología a los siguientes niveles, a hacerlas partes de la vida diaria. Cada sector de la sociedad se ha ido actualizando y usando las distintas herramientas tecnológicas que se han creado para poder mejorar el uso y rendimiento de lo que se requiere hacer. El sector de los entrenadores personales, en general siguen usando herramientas similares a llevar el rendimiento de sus clientes en papel o en los más “moderno”, llevar un archivo de Excel para poder organizar, analizar y revisar progresos de sus clientes.

Por eso se plantea este proyecto, que brindara la herramienta que llevara a los entrenadores y clientes a poder llevar el progreso de una manera más moderna, que brinda más interacción y que ayudara a poder mejorar el rendimiento y progreso de las personas.

Ayudará también a no manejar decenas de archivos Excel que, a la hora de ver la trayectoria del cliente, evitando las pérdidas de los mismos y también ayudando a que se pueda mejorar la toma de datos de cada cliente y así de esta manera analizar mejor y tomar mejores decisiones en cada caso.

INTRODUCCIÓN

El bienestar físico es un objetivo buscado por gran parte de la sociedad. El objetivo siempre se plantea, pero muy pocas veces se logra debido a distintos factores de la vida cotidiana, entre esos la falta de motivación así también de disciplina de la persona, muchos inconvenientes con el plan nutricional que se decide llevar o la planificación de actividad física que se realiza.

El abandonar el bienestar físico aumenta la probabilidad de riesgo de que el cuerpo pueda desarrollar ciertas enfermedades ya sean cardiovasculares, patológicas y/o trastornos metabólicos, por lo que el poder desarrollar un bienestar físico, creando masa muscular, reduciendo grasa y mantenerlo en el tiempo, ayuda a poder mejorar no solo el bienestar físico, sino la calidad de vida de vida que se ve mejorada gracias a el desarrollo de actividades físicas.

La motivación por poder realizar actividades físicas se ha visto muy reducidas a lo largo del tiempo debido a la inmediatez en la que se logran los objetivos hoy en día, así también con la rapidez y facilidad que podemos obtener ciertos objetos en nuestra vida. No hace falta ir muy lejos, las compras del supermercado cada vez se ven más común que te la envíen a la puerta de tu casa, o el hecho de trasladarse para ir a ciertos lugares, por ejemplo, ir al trabajo, que actualmente es utilizado el método de trabajo remoto, lo cual hace que se reduzca la actividad física de trasladarse hasta el centro de trabajo. Así podríamos relatar muchos casos que hacen que hoy día la sociedad se vea en un estado sedentario de actividades físicas.

Para esto se ha propuesto una aplicación Web, que permita a las personas lograr sus objetivos físicos, sin importar su punto de partida ni el objetivo a lograr. Se propone crear un sitio donde las personas puedan sentirse cómodas, “como en casa”. El objetivo principal, es brindar un sitio donde las personas puedan tener un entrenador personal, que este dedicado a ellos, que esté involucrado, que brinde la interacción necesaria para poder mantener a lo largo del tiempo los objetivos físicos que se plantean cada uno de ellos.

OBJETIVO

El desarrollo de una aplicación web que ayude a los entrenadores a crear u organizar planes de ayudas físicas a sus clientes de manera personalizada usando tecnologías modernas que brinden las herramientas para brindar una interfaz intuitiva, con un diseño moderno y un backend eficiente y escalable, usando React y Node-JS. A continuación, enumero los siguientes objetivos principales que se buscan alcanzar:

- Desarrollar una plataforma web que permita a los entrenadores crear rutinas de entrenamiento personalizadas.
- Implementar una funcionalidad para asignar rutinas específicas a cada usuario.
- Diseñar una estructura de datos escalable que organice rutinas por días y ejercicios.
- Optimizar la experiencia del usuario mediante una interfaz intuitiva y responsive.
- Garantizar la comunicación eficiente entre el Frontend en React y el backend en Node.js mediante una API RESTful

DESARROLLO

Tecnologías por usar

El desarrollo de esta aplicación web se utilizaron distintas herramientas de desarrollo. Se escogieron las tecnologías de React y Node JS + Express debido a que son tecnologías modernas y muy utilizadas en el mercado laboral.

La ventaja de usar la librería de React es la posibilidad de reutilizar componentes, la actualización que se realiza sobre el DOM, su documentación y la gran cantidad de herramientas con las que podemos brindar distintas funcionalidades y nos permite escalar con la aplicación. Por otro lado, Node JS + Express es el uso de JavaScript para el desarrollo de este, el gran rendimiento que este brinda y la facilidad de aprendizaje que posee.

Se decidió usar un Git como sistema de control de versiones lo que brindo al desarrollo la posibilidad de avanzar de manera protegida y hacerlo de la manera más eficiente. Al tener la posibilidad de crear ramas y que estas presten la posibilidad del desarrollo de forma aislada, se vio reducido el riesgo de errores y continuidad de desarrollo en otros sectores sin depender de la finalización de otra funcionalidad.

Orden de desarrollo

Se estableció un orden en el que se van a realizar las tareas para poder cumplir con los objetivos del proyecto. Lo primero que se hizo fue identificar que puntos son los más importantes que tienen que desarrollarse primero para poder tener una base firme el cual permita escalar con el proyecto de una forma correcta.

Se realizo un análisis de requisitos Identificando las siguientes entidades que era primordiales, las cuales íbamos a crear, estas son:

- Rutina (id, entrenador_id, fecha_creacion, descripción, objetivo, tipo)
- Rutina_Dia(id, orden, rutina_id, nombre)
- Rutina_Ejercicio (id, series, repeticiones, descanso, notas, día_dia, nombre, peso)
- Usuario (id, nombre, email, password, role, entrenador_id)

Después de tener las entidades básicas con las que íbamos a partir (teniendo como margen que se puedan modificar más adelante), se optó por marcar una estructura de código para poder seguir buenas prácticas y que el mismo pueda seguir ciertos estándares que son usados hoy día así también que sean legibles y principalmente escalable. A continuación, se mostrará la estructura a seguir tanto en el Frontend (Figura 1) como en el Backend (Figura 2).

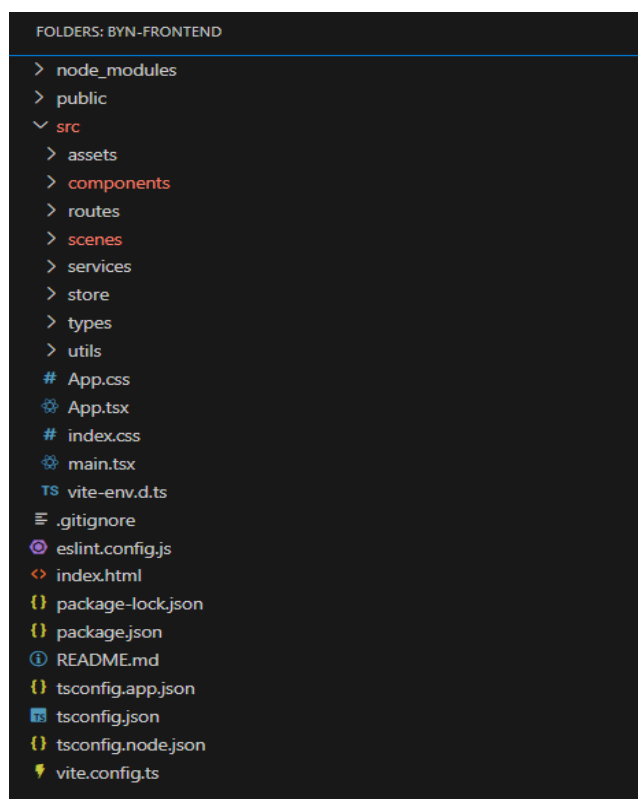


Figura 1. Estructura básica de proyecto Frontend. Elaboración Propia.

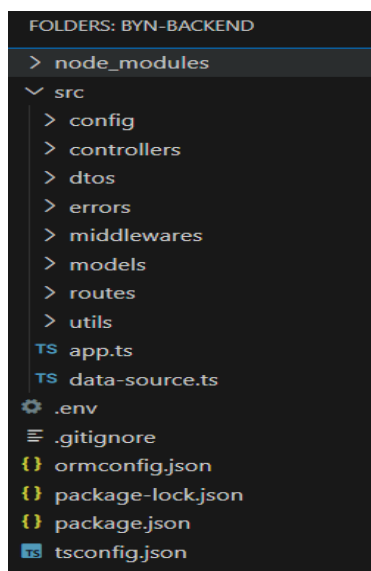


Figura 2. Estructura básica de proyecto Backend. Elaboración propia

Justificación de Estructura de Frontend

Como se puede ver en la Figura 1, se decidió tener dentro de la carpeta src, un directorio con todos los componentes, que en estos se van a encontrar la separación de todos cada uno de los componentes customizados. Se tomo como una buena práctica en este proyecto evitar el código duplicado, llevando a este código a un componente para que se pueda hacer uso de el en distintos sectores del proyecto sin la necesidad de repetir el código.

Tendremos el directorio “routes”, donde se definirán todas las rutas y la asignación del componente correspondiente.

Así también el directorio “scenes”, donde definiremos las escenas o pantallas principales que se van a utilizar, que serán : DashboardLayout, Home, Login, Registro. Cada una de las escenas contendrá su archivo .tsx con la lógica del y el renderizado del componente, sus estilos, su archivo con tipados usados en esa escena y opcional si así lo requiere, un directorio de componentes customizados que sirvan para esa escena en especial. Se pondrá un ejemplo de cómo funciona en el árbol de directorios del proyecto (Ver Figura 3), donde se apreciará, que tiene un “DashBoard.tsx”, acompañado de su archivo “Dashboard.module.css” así también de un “types.ts” y además se incluirá el directorio componentes, que son los componentes propios de la propia escena, que van a ser los el dashboard de los clientes y del entrenador.

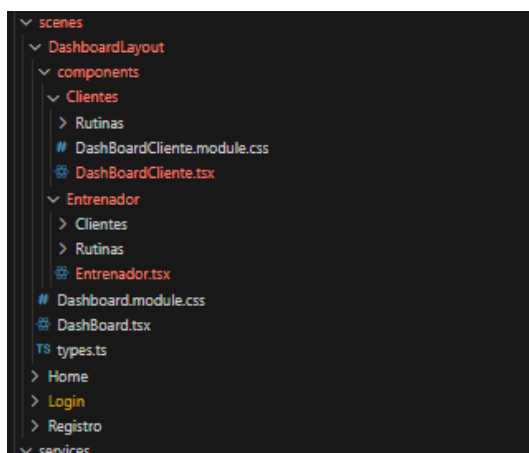


Figura 3. Árbol de directorio de las escenas del proyecto. Elaboración propia.

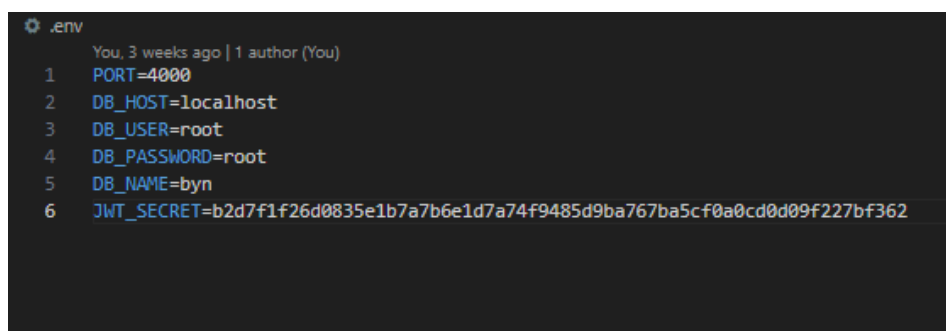
Siguiendo con la explicación de la Figura 1, se tendrá un directorio llamado “services” este gestionará las llamadas al backend.

Para gestionar los estados y el localStorage de mi proyecto, decidí por usar Zustand, debido a que investigando es una herramienta que se está usando que va en subida con respecto al uso en el mercado, sumando que este brinda simplicidad y mínima configuración, es ligero y rápido. En la experiencia personal al haberlo usado, se notó de forma inmediata la asincronía que hace de los datos, me vi beneficiado por la básica configuración que este necesita para poder gestionarlo. Se utilizó este debido a que el localStorage presenta problemas a la hora de anejar el estado en tiempo real.

Tendremos además el directorio de “types” y “utils” que serán para almacenar los tipos generalizados en el proyecto y para las funciones utilizadas en distintas partes del proyecto (por ejemplo, aquí guardamos el manejo de los errores del Api mediante Toast proporcionado por React-toastify).

Justificación de Estructura de Backend

Como se aprecia en la Figura 2, podremos ver que se dividen los archivos según las funcionalidades y por grupos. Usaremos el directorio “config” que contiene la configuración básica para realizar la conexión con la base de datos, en el que se incluyen el host, el puerto, user, password y el nombre de usuario. Tomando como archivo base “.env” que contendrá los parámetros de conexión (Figura 4).



```
.env
You, 3 weeks ago | 1 author (You)
1 PORT=4000
2 DB_HOST=localhost
3 DB_USER=root
4 DB_PASSWORD=root
5 DB_NAME=byn
6 JWT_SECRET=b2d7f1f26d0835e1b7a7b6e1d7a74f9485d9ba767ba5cf0a0cd0d09f227bf362
```

Figura 4. Archivo “.env” ubicado en la raíz del proyecto con los parámetros de conexión a base de datos. Elaboración propia.

Tendremos por otra parte, los controladores que almacenara de cada entidad los distintos métodos para controlar lo que se quiere realizar y devolver hacia el Frontend. Veremos también el directorio “dtos”, que contiene todos los Dtos de cada una de las entidades. Un DTO define qué datos se reciben o se envían entre el Frontend y el Backend, sin exponer directamente la estructura interna de la base de datos o los modelos del servidor.

Tendremos el directorio “errors”, donde definiremos una clase para tipar los errores que se generan desde la Api para que de esta manera sea más eficiente y fácil la comunicación y muestra de errores tanto en el Backend como en el Frontend.

Tendremos el directorio “middlewares”, que fue creado debido a que evalué como buena práctica tener un middleware que interfiera entre la consulta al api y su respuesta, capturando de esta manera los errores que sean controlados o no.

Tendremos el directorio “models”, donde almacenaremos los archivos de las entidades de cada una que hemos planteado anteriormente.

Tendremos el directorio “routes”, donde definiremos las rutas de los distintos controllers cada una en su propio archivo, permitiendo de esta manera gestionar de una manera más eficaz cada una de las rutas, y luego exportándolas en el archivo “index.ts”.

Desarrollo de proyecto

Teniendo una base de información y una estructura de proyecto a seguir, comenzamos con el desarrollo oficial del proyecto. A continuación, se detallará en 3 secciones en la que se detallaran por qué y el cómo fue el desarrollo de estas. Estas 3 secciones serán: Creación de base de datos, Desarrollo de Backend y Desarrollo de Frontend.

Creación Base de datos

Se van a crear las siguientes tablas: usuario, rutina, rutina_dia, rutina_ejercicio, rutina_asignada. Con los atributos nombrados anteriormente.

Desarrollo de Backend

En esta sección veremos cómo fue el desarrollo del Backend. Como se mencionó anteriormente, la estructura de proyecto en el backend es de tener distintos directorios los cuales mantendrán ordenados las diferentes funcionalidades.

Uno de los componentes principales es el directorio de modelos, ya que en él se define el mapeo de las tablas de la base de datos mediante entidades.

Para el desarrollo de las entidades, se comienza creando el archivo con el nombre específico de “nombre-entidad + .entity.ts”, poniendo esto como modelo para la definición de los archivos.

Para poder realizar el mapeo de las entidades se usó la librería TypeORM que es una librería que sirve especialmente para poder hacer el mapeo de las clases a tablas en una base de datos relacional. Se crearon cada una de las entidades siguiendo la siguiente estructura de archivo.

Usamos el decorador `@Entity(nombreTabla)` antes de definir la clase de la entidad que queremos mapear. Luego TypeOrm nos brinda distintos decoradores para poder crear la tabla como una clase en nuestro Backend, como se puede apreciar en la Figura 5, usamos los siguientes:

- “PrimaryGeneratedColumn”: Generamos una columna primaria
- “Column”: Definimos una columna en la base de datos, donde además le podemos especificar el tipo, la longitud, si es única, su valor por defecto entre muchas otras posibilidades
- “OneToMany”: Definimos la relación uno a muchos
- “ManyToOne”: Definimos una relación de muchos a uno
- “JoinColumn”: Especifica la columna que hace la relaciones `@OneToOne` y `@ManyToOne`

También se puede notar que se usa el decorador “@Exclude” que se usa principalmente en la librería class-transformer y se usa para cuando no quieres que ciertos campos aparezcan en las respuestas de la API, esto se hizo principalmente porque posteriormente se anticipó que se crearía un endpoint que haga un GET del usuario, ya sea que traiga todos los usuarios o traer alguno en específico, no era de buena práctica enviar consigo su contraseña, lo cual abriría una vulnerabilidad en la seguridad del sistema.

Luego en el directorio de controllers, tendremos los controladores de servicio que se van a brindar, los creados para este proyecto fueron, de rutina, usuario y auth. La definición de los archivos será de la siguiente manera: “nombreTabla”.controller.ts .

Primeramente, tendremos el controlador de la autenticación, que se creó para poder autenticar el usuario o sea brindar la funcionalidad de poder hacer login, este se encuentra en el archivo /src/controllers/auth.controller.ts.

Recibirá las credenciales del usuario en el req.body, validará que las credenciales sean validas y luego devolveremos una respuesta dependiendo de si el usuario existe con el estatus 404, o el estatus 401 en el caso de la contraseña sea incorrecta. En el caso de que sea correcta devolveremos un token de autenticación y datos básicos del usuario. Este token expirara en 1 hora.

Por otro lado, tendremos el controlador de la rutina, entre ellos se crearon los distintos métodos: getAllRoutines, createRoutine, getRoutinesByEntrenador, getRoutineById, updateRoutine, deleteRoutine, asignarRutinaAUsuario, getRutinasAsignadasPorUsuario.

En cada método se recibirán por parámetros los datos necesarios (req de tipo Request) y el parámetro res de Response que será lo que devolveremos como respuesta. Todos siguen una misma estructura de método, recibiendo por el parámetro req (req.params) si es necesario para poder realizar luego la consulta interna y posteriormente la devolución de los datos solicitados.

Por ejemplo, en el caso de obtener las rutinas por Id, ósea obtener una rutina por su id, en el parámetro de req, esperaremos que se nos envíe el id, utilizando la siguiente línea “const rutinaId = parseInt(req.params.id);”, luego usaremos las funciones del repository, que se obtienen usando el dataSource (datasource es que hace la conexión con la base de datos, trayendo las entidades y colocando los parámetros de conexión) y a partir de ahí podremos usar las herramientas que necesitemos, en este caso de obtener la rutina por su id, usaremos la función findOne en el cual le podremos pasar como parámetro de filtro el id o también la relación que hay entre estos.

Internamente este realiza la query para la base de datos y obtener los datos, solo que nos brinda la facilidad de no tener que escribir la consulta letra a letra, sino que lo hace internamente.

Se usarán las distintas funcionalidades según la necesidad por la que fue creado el método, los cuales fueron: find, find({where:...}), findOne que son funcionalidades básicas para obtener datos de una tabla, lo que sería equivalente al “SELECT” en las query.

Luego según las necesidades, se usó las funcionalidades: save, remove que fueron usadas para realizar un INSERT o hacer un UPDATE en los datos, realizar un DELETE respectivamente.

Por último, se han creado el controlador de los usuarios, la cual se siguió la misma estructura y se usó la misma lógica que en el controlador de las rutinas. Los métodos definidos fueron los siguiente: getAllUsers, createUser, getCientesByEntrenador, getRutinasAsignadasPorUsuario, deleteUser.

En otro directorio, podremos ver la definición de las rutas que se establecieron, lo cual es necesario para poder dar por finalizado el “endpoint” (endpoint, es el punto por el cual se van a comunicar el Backend con el Frontend).

Se mostrará como se hizo separar las rutas de cada una de las entidades. Se separó en las de autorización, las rutinas y los usuarios.

Estas se crean generando la variable `rutinaRouter = Router` el cual se importa de `express` y luego se le asigna todos los métodos HTTP que serán utilizados, acompañado del método correspondiente ya mostrado anteriormente. Luego se importan todas las rutas de cada una de las entidades en el archivo `src/routes/index.ts` que será el que usa nuestro archivo `app.ts` para utilizar las rutas. En conclusión, en el archivo `index.ts` centralizaremos todas las rutas creadas.

Estas se crean generando la variable `rutinaRouter = Router` el cual se importa de `express` y luego se le asigna todos los métodos.

Finalizando parte del desarrollo del Backend, tendremos nuestro archivo `app.ts` donde se realizarán todas las configuraciones y asignaciones de funcionalidades para poder iniciar nuestra aplicación. Aquí utilizaremos la configuración y la declaración de `express`, así también definiremos la conexión con el frontend, configuramos como será el inicio de la aplicación, definiendo logs por consola dependiendo de si el levantamiento de la aplicación fue exitoso o no. También declaramos el uso de las rutas, con base en la url de `"/api"` y luego realizamos la configuración de a que puerto estará corriendo la aplicación.

Desarrollo de Frontend

En esta sección veremos cómo fue el desarrollo del Frontend. El proyecto está estructurado en diferentes carpetas que agrupan las funcionalidades de manera modular, que lo que facilita la mantenibilidad y escalabilidad del código.

Para la documentación del desarrollo del Frontend, se realizará por cada una de las escenas las cuales se nombrarán como `"Scene"` que fue la palabra usada en el código como referencia a las escenas, o sea cada una de las pantallas.

Se decidió por seguir un diseño limpio, simple y moderno. La intención de la aplicación es lograr el diseño `"Black & White minimal"` el cual hoy en día está muy utilizado para el diseño el cual busca transmitir sofisticación, profesionalismo y minimalismo.

Scene: HOME

Se ilustrará en la Figura 5, el resultado actual de la scene.

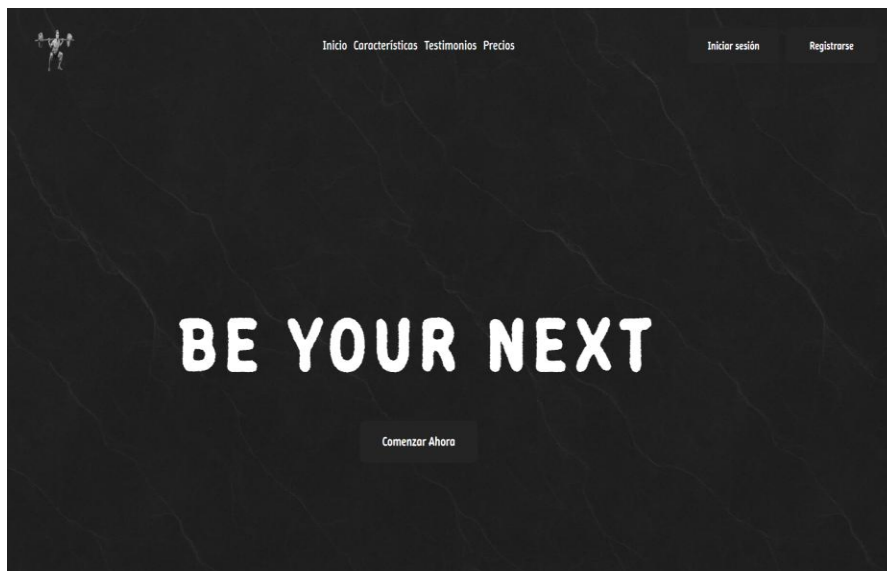


Figura 5. Pantalla Home. Elaboracion propia.

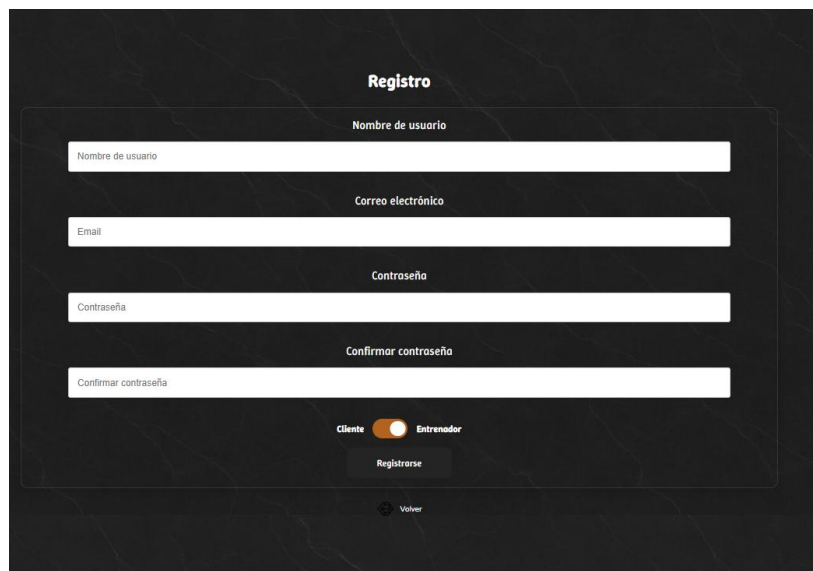
En esta primera pantalla, que será la primera que vera el usuario al ingresar a la aplicación, se ven internamente distintos componentes. Tendremos primeramente el componente “NavBar” que es el navegador, el cual incluye el icono de la aplicación, un navegador con sus 4 opciones (2 de los cuales están para próximas versiones, “Testimonios” y “Precios”) y dos botones, los cuales nos conducirán a la scene de “Login” o “Registro” según hayamos seleccionado.

Luego tendremos un componente personalizado llamado “Content” que lo que pretende este componente es centralizar en un componente contenido como si fuera una etiqueta div, a este le podremos pasar el contenido (Childrens), así también pasarle distintas propiedades incluyendo estilos personalizados funciones como el onClick.

Dentro de este componente, contiene el título de la aplicación y un botón, que este navegara hacia la scene “Registro”.

Scene: REGISTRO

En esta scene (Figura 6), tendremos un formulario donde se pedirá que se ingrese los datos básicos para crear un usuario. La contraseña deberá tener un largo mínimo de 6 caracteres. Se verificará que el email tenga un formato correcto. Todos los campos son obligatorios.



The image shows a registration form titled "Registro". It has a dark background with a light gray border around the form area. The form contains the following elements:

- Label: "Nombre de usuario" above a text input field.
- Label: "Correo electrónico" above a text input field.
- Label: "Contraseña" above a text input field.
- Label: "Confirmar contraseña" above a text input field.
- A toggle switch below the password fields, with "Cliente" on the left and "Entrenador" on the right. The "Entrenador" option is selected, indicated by a white circle.
- A "Registrarse" button below the toggle switch.
- A "Volver" button at the bottom center.

Figura 6. Scene Registro.tsx. Elaboracion propia

El formulario se desarrolló en un componente a parte, donde cada input y botón fue diseñado y customizado para mantener el mismo uso de inputs, labels y botones en toda la aplicación.

Actualmente, si bien se muestra la opción para que el usuario que se va a registrar pueda escoger si es un cliente o un entrenador, la funcionalidad deshabilitada, lo cual solo deja crear un entrenador, en próximas versiones se podrán crear clientes.

Al hacer click en el botón Registrarse, hace la validación del formulario y llama al método "handleSubmit" el cual hará la petición a la API. Si se realiza de forma correcta la creación del usuario navegara al LOGIN.

La contraseña se guardará codificada para brindar la mínima seguridad necesaria.

Scene: LOGIN

En esta scene (Figura 7) tendrá la funcionalidad de iniciar sesión.

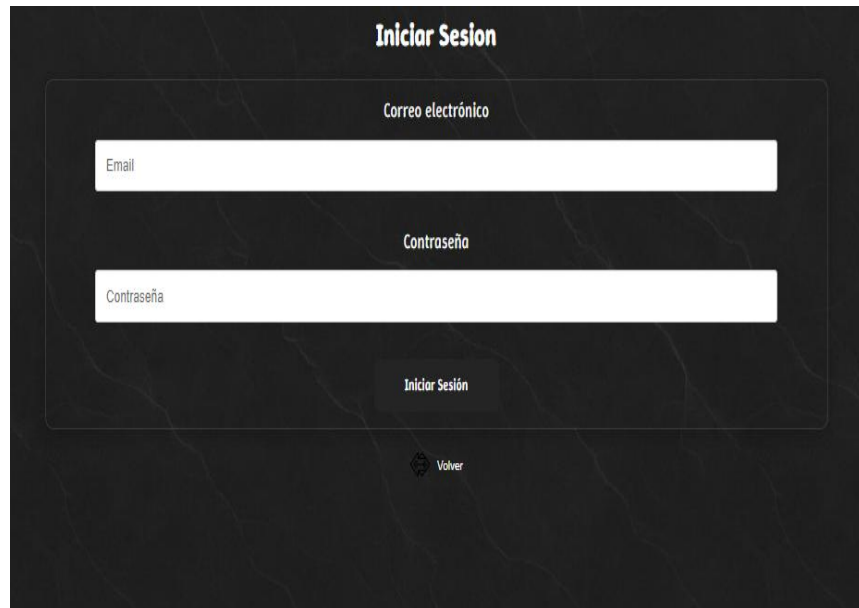
The image shows a login form on a dark background. At the top, the title 'Iniciar Sesión' is displayed in white. Below it, there is a light gray rounded rectangle containing the form elements. Inside this rectangle, the label 'Correo electrónico' is centered above a white input field with the placeholder text 'Email'. Below this, the label 'Contraseña' is centered above another white input field with the placeholder text 'Contraseña'. At the bottom of the rounded rectangle is a dark gray button with the text 'Iniciar Sesión' in white. Below the rounded rectangle, centered, is a small circular icon with a right-pointing arrow followed by the text 'Volver'.

Figura 7. Scene Login.tsx. Elaboracion propia.

Podremos realizar un inicio de sesión, el cual hará la petición a la API, haciendo una decodificación de la contraseña y comparándola con la ingresada desde el frontend.

Scene: Dashboard Cliente

En esta scene (Figura 8), podemos apreciar la pantalla que ve el cliente al iniciar sesión. Le aparece un dashboard, el cual el componente del SideBar, se usa en el cliente como en el de entrenador, cambiando las opciones según el rol del usuario que haya iniciado sesión.

Al iniciar sesión va a poder ver las rutinas que tiene asignadas y al hacer click sobre la rutina, se desglosara la rutina con los ejercicios puestos.

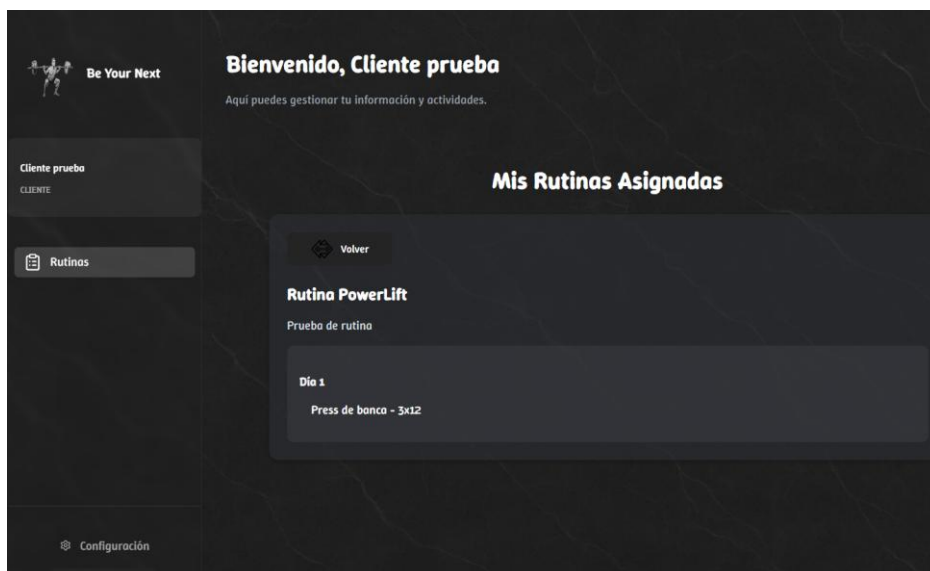


Figura 8. DashBoard del cliente. Elaboracion propia.

Scene: Dashboard Entrenador

En esta scene (Figura 9), se le mostrara al entrenador al hacer sesión un usuario con el role de entrenador. Podra filtrar entre los clientes que tenga disponible, para cada cliente puede ver las rutinas que tenga asignadas, asignarle más rutinas y borrar el cliente.

Tendremos la posibilidad de crear un cliente también, haciendo click en el botón “Nuevo Cliente”, se abrirá un popup (es un componente customizado para poder agregarle todas las funcionalidades posibles (Figura 10).

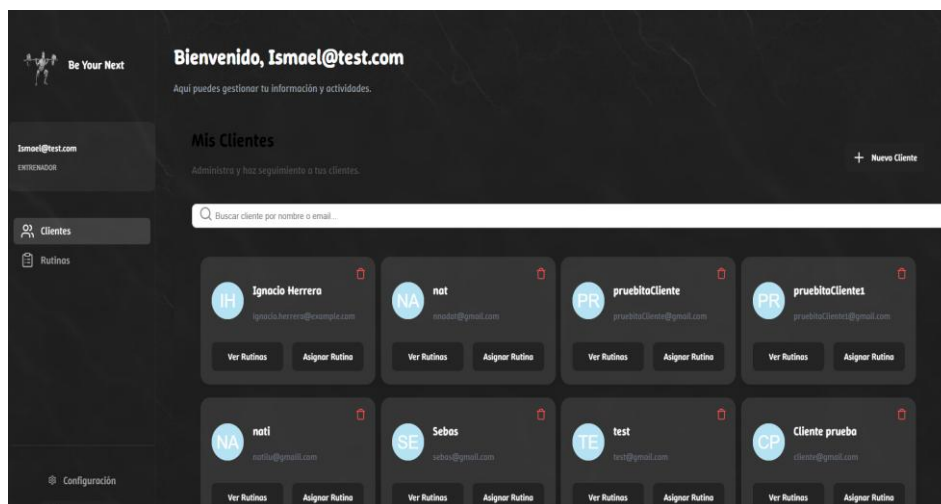


Figura 9. Dashboard Entrenador. Elaboracion propia.

Figura 10. Creación de cliente desde el Dashboard de entrenadores. Elaboracion Propia.

En la Figura 11, veremos la sección de rutinas desde el dashboard del entrenador, donde podremos crear nuevas, editarlas, borrarlas y verlas cumpliendo de esta manera con todos los métodos de GET, POST, PUT y DELETE.

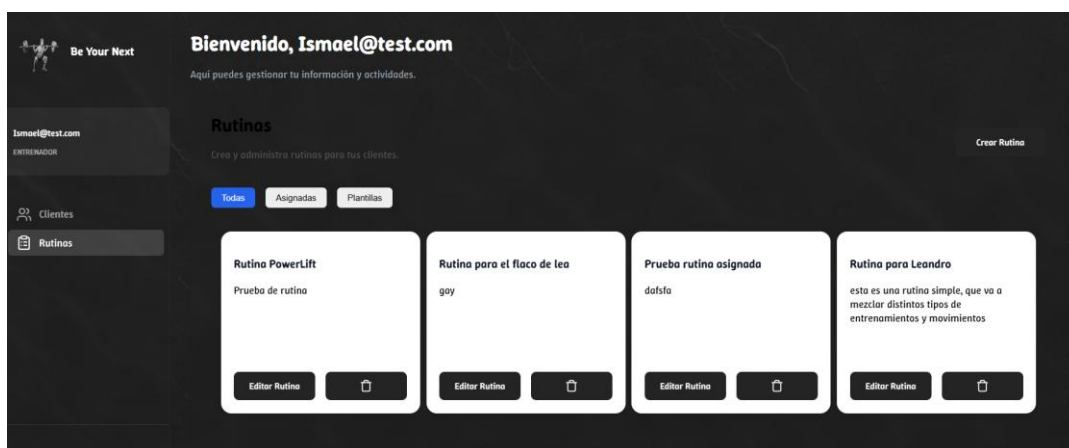


Figura 11. Dashboard cliente, rutinas. Elaboracion propia.

Para crear una rutina se abrirá un popup como en el registrar un cliente y daremos la posibilidad de crear un rutina, agregando los días y en cada día registrar los ejercicios que se van a usar. (Figura 12)

Figura 12. Crear Rutinas personalizadas. Elaboración propia.

CONCLUSIÓN

Al comenzar a desarrollar el proyecto, se vio una necesidad en un sector, el cual siempre estuvo presente en la sociedad, como lo es el bienestar físico y vi la incidencia de que hoy en día no existían maneras de poder incentivar a las personas a poder mantener sobre el tiempo sus cambios físicos, lo que llevaba abandonar el objetivo de esa persona.

El objetivo fue crear una aplicación que le brinde la posibilidad de que los entrenadores personales pudieran gestionar de una manera más eficiente y brindándole al cliente poder ver su progreso en el tiempo.

La aplicación desarrollada en React para el Frontend y Node JS con Express para el Backend, me brindaron la oportunidad de formarme mejor como desarrollador, debido a que son librerías tienen una curva de nivel de aprendizaje baja (dando por hecho que yo había programado antes en JavaScript) y gracias a la gran documentación que estas brindan, fue más sencillo desarrollar según las ideas que yo tenía acerca del proyecto.

Se logró con respecto a los objetivos iniciales, el poder crear una aplicación moderna, que brinde la posibilidad al entrenador de crear rutinas personalizadas y asignarlas a sus clientes. Brinda la herramienta de gestionar sus clientes de una forma intuitiva y fácil. Por otro lado, los clientes pueden acceder a sus rutinas y ver cada ejercicio asignado.

Si bien hay aspectos que pueden mejorar, por ejemplo, en darle más funcionalidades al cliente, sin embargo, se logró cumplir con los objetivos principales, brindando una aplicación moderna y que haga sentir al usuario ya sea que sea Entrenador o Cliente.

En conclusión, fue una experiencia que me ayudo a aprender muchas herramientas y metodologías de trabajo. Pude aprender a como realizar una gestión de usuarios organizada, tener una estructura de proyecto organizada y que pueda ser escalable. Sin dudas, este proyecto me ayudo a incrementar mis habilidades como desarrollador de aplicaciones Web.