**Integrantes:**

* Nayeli Suarez Portillo - A00382425
* Felipe Rojas Prado - A00393918

**Informe de proyecto:**

El objetivo principal del frontend fue conectar de manera efectiva con el backend preexistente, implementando una interfaz intuitiva y protegida para gestionar las funcionalidades del gimnasio. Este módulo se desarrolló utilizando **React** y **TypeScript**, cumpliendo con los requisitos establecidos

**Objetivos cumplidos**

* Navegación protegida según su autoridad:
* Uso de React Router junto con el componente RequireAuth para restringir el acceso a rutas según su autoridad.
* Integración de un contexto de autenticación (AuthContext) para manejar la sesión del usuario y almacenar su rol y nombre.

**Pantallas desarrolladas:**

* Inicio (Index): Página principal accesible por todos los usuarios, con navegación básica e intuitiva para un redireccionamiento a login y signup.
* Login: Formulario para que los usuarios inicien sesión, con conexión al endpoint de autenticación del backend.
* Registro: Permite registrar nuevos usuarios en el sistema, interactuando con el backend para enviar los datos.
* Dashboard de user y admin: Según el rol que decodifica el token, se lleva a una menú de inicio que observará dependiendo de sí es administrador o usuario, además administrador puede navegar entre ambos url, pero usuario no.
* Gestión centralizada de recursos como máquinas, rutinas, roles y horarios.
* Gestión de Máquinas: Página que permite observar las máquinas existentes, una descripción, crearlas y eliminarlas.
* Gestión de Rutinas: Página intuitiva de rutinas donde se pueden agregar o editar rutinas existentes donde se observa el nombre y descripción de la misma.
* Gestión de Horarios: Página que puede ver el usuario la asociación entre las rutinas y usuarios, puede asociar según id de usuario a un id de rutina, editar o eliminarlas.
* Roles: Página para que los administradores gestionen roles (creación, edición y eliminación, asignación de roles a usuarios).
* Asignar rol: Página para los administradores donde pueden asignar los diferentes tipos de roles a los usuarios.
* Lista de usuario: Página que observan los administradores que lista los usuarios existentes en la aplicación con su respectivo nombre, email y rol actual

**Conexión con el backend:**

* Configuración de Axios con interceptores para manejar automáticamente los tokens JWT y redirigir en caso de errores de autenticación.
* Servicios dedicados para cada funcionalidad (AuthService, RoleService, MachineService, etc.), asegurando una separación clara de responsabilidades.
* Listado: Listado de usuarios con su información de cada uno

**Estilización:**

* Uso exclusivo de TailwindCSS para crear una interfaz responsiva, moderna y fácil de navegar.

**Control de calidad del código**:

* Configuración de ESLint y Prettier para mantener un código consistente y libre de errores.
* Uso de tipado fuerte en TypeScript para reducir errores y mejorar la mantenibilidad del código.
* Husky para los commits en github

**Descripción de problemas encontrados:**

* **Problemas de configuración inicial del proyecto:** La integración del frontend con el backend requirió ajustes en la configuración de CORS y permisos en las rutas de Spring Security.
* **Manejo de roles y permisos:** Se presentó dificultad para garantizar que ciertos endpoints estuvieran restringidos a usuarios con roles específicos, debido a la lógica del proyecto (esta la podemos borrar porque quedamos como un zapato diciendo que modelamos la peor base de datos)
* **Autenticación y sesiones:** Durante el desarrollo de la funcionalidad de login y registro, se encontraron problemas para obtener correctamente el nombre del usuario desde el token JWT.
* **Gestión de dependencias en frontend:** Hubo problemas al configurar ESLint y Prettier para garantizar un estilo consistente en TypeScript. Esto enfocado con el precommit y el push, pues a pesar de que todo estuviese funcional, la configuración llegó a generar conflictos.

**Seguimiento:**

**11 al 25 de Noviembre 2024**

| * Nayeli Suarez |
| --- |
| * *¿Qué hice desde la última reunión?* * Implementé servicios en TypeScript para consumir las APIs de Machine, Routine y Schedule, y configuré Axios para la autenticación con JWT y manejo centralizado de errores. Usé React Router con autenticación protegida y ajusté las páginas para mostrar funcionalidades según el rol del usuario. Implementé AuthContext para obtener el rol desde el token y desarrollé una página de Roles, permitiendo la creación, edición y eliminación de roles para usuarios ADMIN. * *¿Qué haré hasta la siguiente?* * nada * *¿Qué impedimentos encontré?* * Configuración de ESLint causó errores en algunos commits debido a reglas. * Problemas para extraer información del token JWT |

| * Felipe RojasPrado |
| --- |
| * *¿Qué hice desde la última reunión?* * Corregir ciertos endPoints para que funcionaran de tal modo que fuera más fácil de manejar en el front, gestionar roles, asignar roles * Hice la configuracion del admin y sus funcionalidades desde el back y el front * *¿Qué haré hasta la siguiente?* * Encontrar el cómo hacer el deploy de front ya que hay dificultades * *¿Qué impedimentos encontré?* * El listar usuarios tuve problemas al obtener el rol que tienen actualmente cada usuario |

**Link al repositorio:**[**Repo**](https://github.com/Computacion-2/proyecto-del-curso-nafe-y-sale.git)