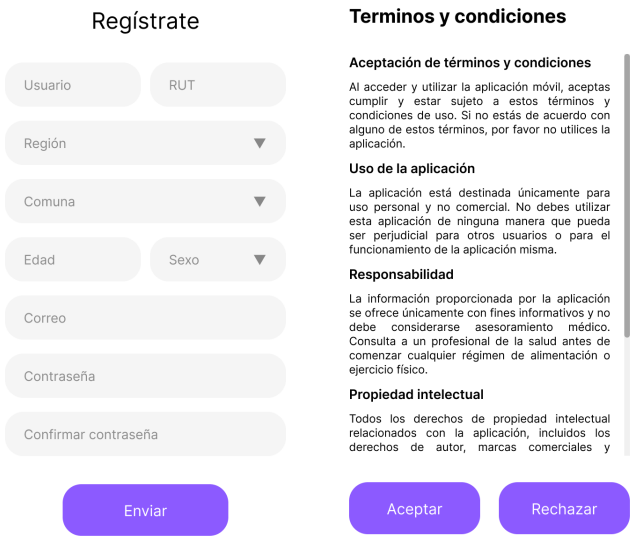
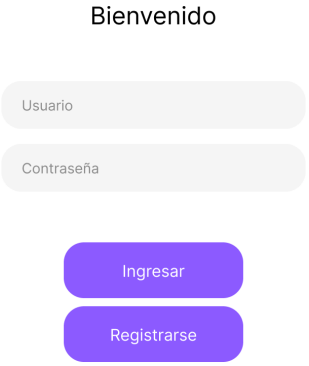
Informe Entrega Final

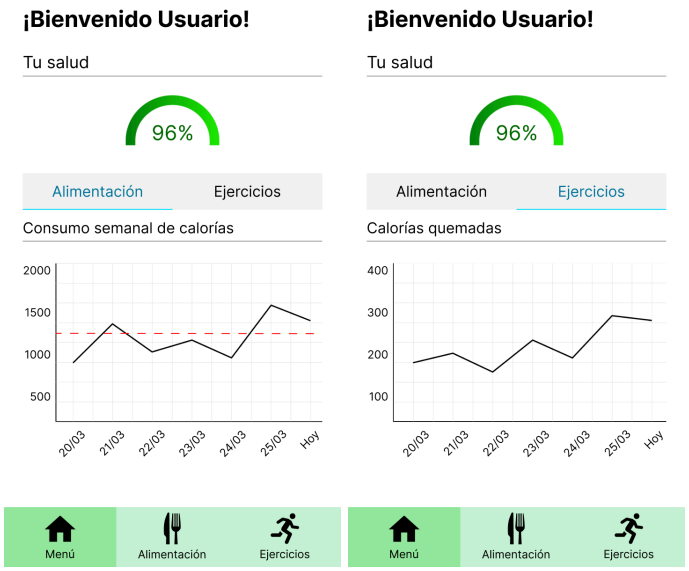
Asignatura: Ingeniería Web y Móvil
Integrantes:

- Sebastián Maximiliano Jeria López
- Benjamín Alonso Robles Arancibia
- Oscar Francisco Rojas Gaete

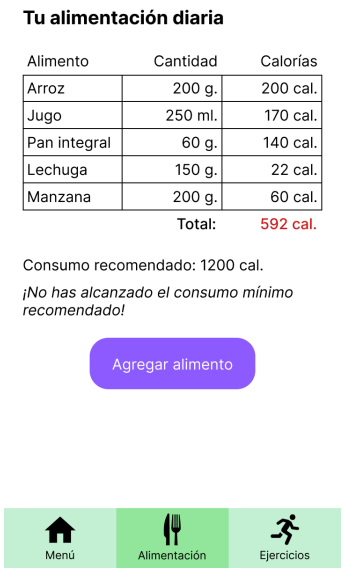
Tema Proyecto: Alimentación y actividad física saludable

1.1. Funcionalidades de la aplicación

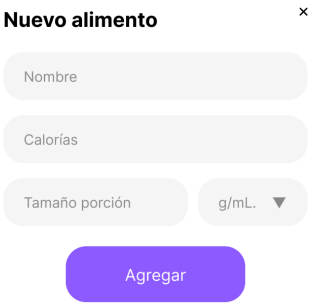
	**Registro de usuario**: El usuario podrá crear una cuenta en la aplicación a partir de los siguientes datos: nombre de usuario, RUT, región, comuna, edad, sexo, correo y contraseña. Además, deberá aceptar los términos y condiciones de la aplicación.
	**Inicio de sesión:** Una vez registrado, el usuario podrá ingresar a la aplicación mediante su nombre de usuario y su contraseña.



Generación de estadísticas: La aplicación contará con un indicador que permitirá al usuario medir qué tan saludable es su vida, en base a un porcentaje entre 0 y 100. Además, se incorporarán gráficos donde el usuario podrá revisar su consumo de calorías y una aproximación de las calorías quemadas durante la semana.



Registro diario de alimentos: El usuario podrá añadir los alimentos que haya ingerido durante el día en base a una lista predefinida de alimentos, especificando también el tamaño de la porción consumida. La aplicación se encargará de calcular y mostrar las calorías equivalentes a tal porción.

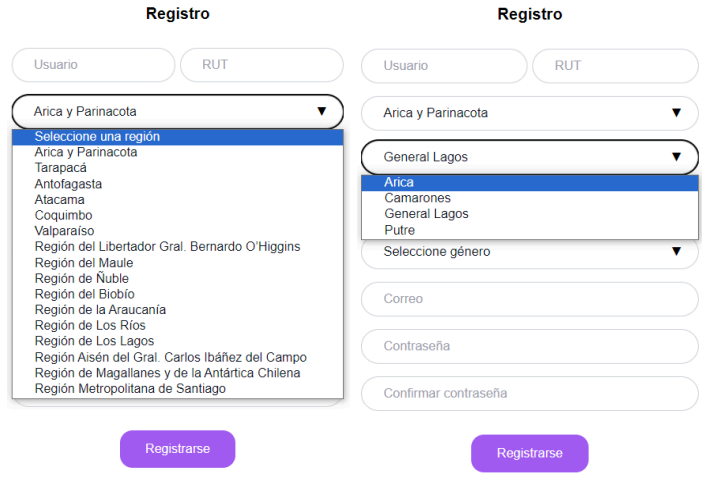


Crear nuevo alimento: En caso de que el alimento no se encuentre en la lista, el usuario podrá crear un nuevo alimento añadiendo su nombre, un tamaño de porción de referencia y las calorías correspondientes a esta porción.

	**Registro diario de ejercicios**: El usuario podrá seleccionar un ejercicio base, además de agregar el tiempo en minutos que realizó dicho ejercicio. A partir de esto, se realizará un cálculo de las calorías quemadas por medio de una función preestablecida.

[Link a Figma]

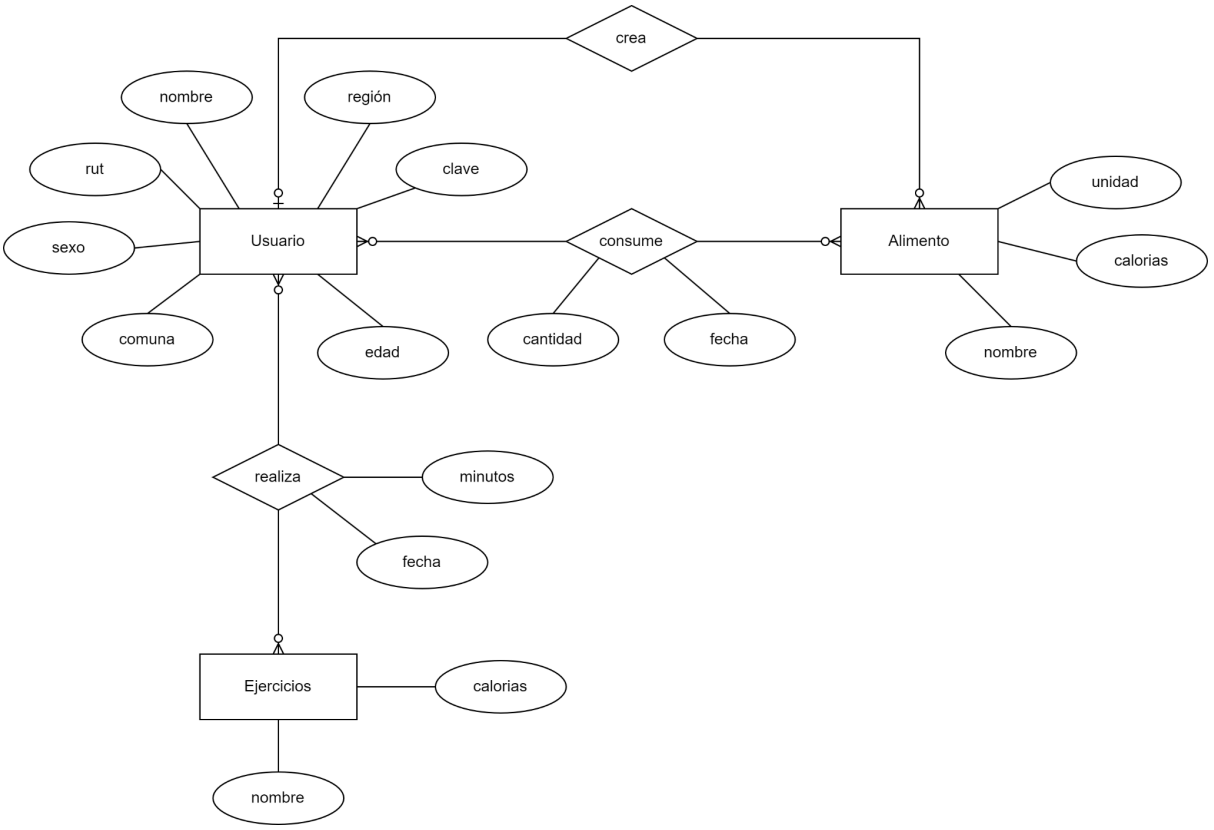
2.2. Lectura de un archivo JSON

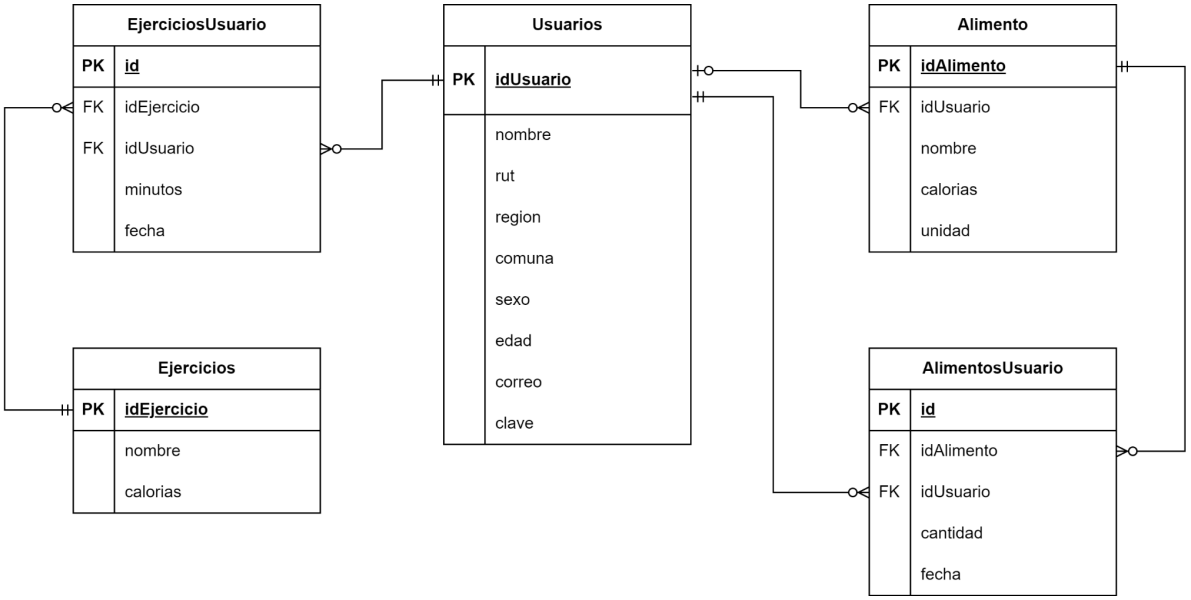



```
function get_regions(data: RegionData) {
    if (data == null)
        return (<option>Cargando</option>)

    return data.regiones.map((region: Region, index: number) => {
        return <option key={index+1} value={index+1}>{region.region}</option>
    })
}

function get_comunas(data: RegionData|null, region: number) {
    if (data == null )
        return (<option>Cargando</option>)
    else if (region == null) return (<option value={0}>Comuna</option>)

    else if (region != 0) return data.regiones[region-1].comunas.sort().map((comuna: string, index: number) => {
        return <option key={index+1} value={index+1}>{comuna}</option>
    })

}

export const SelectRegionComuna = () =>{

    const [data,setData] = useState(null)
    const [selection,setSelection] = useState(0)

    useEffect(() => {
            const response = fetch('https://gist.githubusercontent.com/juanbrujo/0fd2f4d126b3ce5a95a7dd1f28b3d8dd
            .then(response => response.json())
            .then(result => {
                setData(result)
            })
    }, [])

    useEffect(setSelection, [data])
```

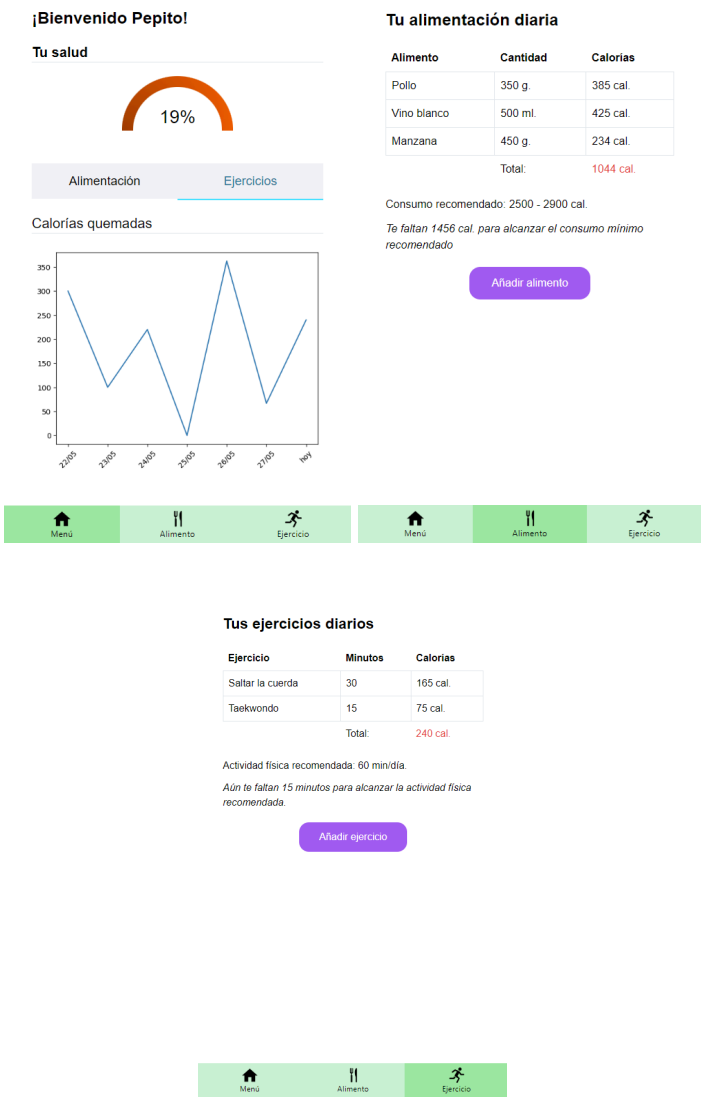
Se leerá un archivo JSON externo con los datos de las regiones y comunas.

2.3. Estructura de la base de datos



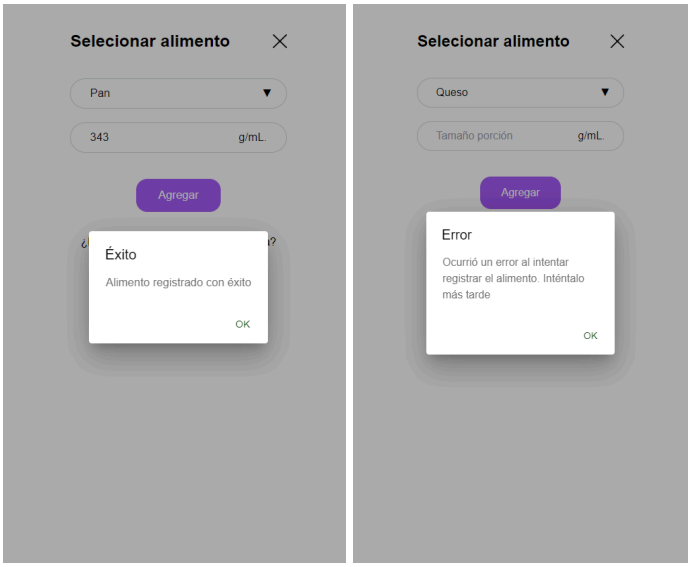


2.4. Patrones de diseño



Bottom Navigation: Consiste en colocar diferentes opciones de navegación en la parte inferior de la interfaz, para permitir al usuario acceder a las principales vistas de la aplicación de manera fácil y rápida.

Tab Bar: Similar al Bottom Navigation, pero permite cambiar entre diferentes vistas anidadas a partir de un menú no necesariamente ubicado en la parte inferior.

	**Ventanas emergentes**: En caso de querer comunicar un mensaje al usuario se utilizarán ventanas emergentes, ya sea para comunicar el éxito o fallo de una operación.

Entrega Final

EF.1 Autenticación y autorización

La autenticación de los usuarios se realiza por medio de su correo electrónico y contraseña, mediante una interfaz de inicio de sesión:

Para la autorización del usuario, la aplicación hace uso de la tecnología JSON Web Token (JWT). En caso de que las credenciales sean correctas, el servidor enviará al usuario un token codificado que contendrá la información relevante de la sesión, específicamente la ID del usuario y su rol. Este token será utilizado para validar la identidad del usuario en las futuras solicitudes que realice al servidor, con el fin de que solo los clientes autorizados que hayan iniciado sesión y que tengan los permisos correspondientes puedan acceder a las funcionalidades de la aplicación.

La aplicación cuenta con dos tipos de usuarios, básico y administrador, que pueden acceder a distintas funcionalidades como se muestra en la siguiente tabla:

Funcionalidad	Usuario
Registro	Básico
Iniciar sesión	Básico y administrador
Visualizar estadísticas	Básico
Registrar alimento consumido	Básico
Registrar ejercicio realizado	Básico
Crear nuevo alimento	Básico y administrador
Crear nuevo ejercicio	Administrador
Editar y eliminar alimentos	Administrador
Editar y eliminar ejercicios	Administrador

EF.2 Implementación del backend

El backend fue realizado en lenguaje Python mediante el framework Flask. A continuación se muestra una tabla donde se definen las rutas de la API:

Ruta	Método	Parámetros	Cuerpo	Descripción
/user/login	POST		correo, contraseña	Inicia sesión de un usuario.
/user/register	POST		usuario, rut, edad, género, correo, contraseña, región, comuna	Registra un nuevo usuario.
/user/health	GET	id		Obtiene el porcentaje de salud de un usuario.
/user/food	POST		idUsuario, idAlimento, cantidad	Registra un alimento consumido por el usuario.
/user/food	GET	id		Obtiene los alimentos consumidos por un usuario.
/user/exercise	POST		idUsuario, idEjercicio, minutos	Registra un ejercicio realizado por el usuario.
/user/exercise	GET	id		Obtiene los ejercicios realizados por un usuario.
/plot/food	GET	id		Obtiene un gráfico con el consumo de alimentos.
/plot/exercise	GET	id		Obtiene un gráfico con las calorías quemadas.
/food	POST		idUsuario, nombre, calorías, unidad, porción	Crea un nuevo alimento.
/food	GET	id		Obtiene los alimentos de la aplicación más los registrados por el usuario.

Ruta	Método	Parámetros	Cuerpo	Descripción
/food	PUT		idAlimento, nombre, calorías, unidad, porción	Actualiza un alimento.
/food	DELETE		idAlimento	Elimina un alimento.
/exercise	POST		nombre, calorías	Crea un nuevo ejercicio.
/exercise	GET			Obtiene todos los ejercicios.
/exercise	PUT		idEjercicio, nombre, calorías	Actualiza un ejercicio
/exercise	DELETE		idEjercicio	Elimina un ejercicio.

EF.4 Aspectos de seguridad

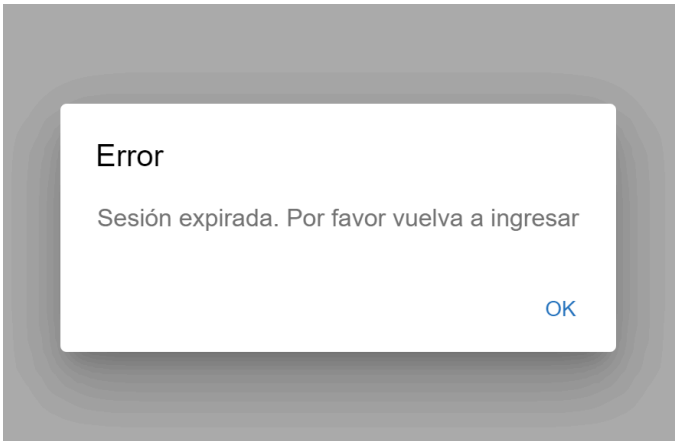
1. Autenticación y autorización de usuarios mediante JWT: Al iniciar sesión de manera exitosa, el servidor envía al usuario un token encriptado, que deberá ser utilizado por el cliente para realizar futuras solicitudes. En cada solicitud, el token deberá ser agregado al encabezado "Authorization" con el formato "Bearer [token]". Si el token no se encuentra, el servidor responderá con un estado de error "401 (UNAUTHORIZED)". También puede ocurrir que un usuario intente acceder indebidamente a funciones de un administrador; en este caso, el servidor responde con el código "403 (FORBIDDEN)".

2. Encriptación de contraseñas en la base de datos:

Para la encriptación de contraseñas se utiliza la función SHA2 nativa de SQL, la cual convierte un string de longitud variable en una cadena codificada de largo fijo. Para la implementación se utilizó el estándar SHA-256 que genera cadenas de 256 bits (64 caracteres).

3. Manejo de cierre de sesión:

El token de sesión cuenta con un tiempo de expiración de 15 minutos, sin embargo, cada solicitud al servidor permite actualizar el token generando uno nuevo, por lo cual, mientras el usuario se mantenga activo en la aplicación, la sesión permanecerá abierta. Una vez pasados 15 minutos de inactividad, la aplicación mostrará el siguiente mensaje y redirigirá al usuario nuevamente a la interfaz de inicio.



Por otra parte, la aplicación también cuenta con botones de cierre de sesión en caso de que el usuario desee borrar sus datos de sesión y volver a la interfaz de inicio.

4. Uso de variables de entorno:

Durante el desarrollo del proyecto se utilizaron variables de entorno, principalmente para resguardar datos sensibles y facilitar la manipulación de variables en los contextos de desarrollo y producción. Cabe decir que tanto backend como frontend cuentan con sus propios archivos para definir las variables de entorno. Para gestionar estas variables, en el frontend se utilizó la herramienta Vite, mientras que en el backend se utilizó la librería dotenv. Entre las variables almacenadas se encuentran las credenciales de la base de datos, la URL de la API y la clave secreta usada por JWT para generar los tokens.

Nota: No es recomendable subir a GitHub los archivos con las variables de entorno, especialmente si contienen datos sensibles. Esto solo se realizó con el propósito de entregar un código completamente funcional.