

DOCUMENTO

DOCUMENTACIÓN DE PÁGINA WEB

PROYECTO – RECOVGLOX

SEXTO CUATRIMESTRE

ALUMNOS:

BARRETO OCAMPO SANTIAGO

OJEDA HERNANDEZ MARIO ALBERTO

ASESOR:

DRA. BRENDA JUAREZ SANTIAGO

DR. NORMA ALEJANDRA LEDESMA URIBE

SAN JUAN DEL RÍO, QRO.

JULIO DE 2025

1. Introducción

- Objetivo del proyecto
- Público objetivo
- Alcance

2. Tecnologías Utilizadas

- Frontend:
- Backend:
- Estilos:
- Base de datos:
Firebase
- Librerías:

3. Estructura del Proyecto

- Organización de carpetas
- Archivos principales

4. Funcionalidades Principales

- Registro e inicio de sesión
- Gestión de pacientes
- Obtener Lista de Pacientes:
- Panel de Administración
- Diseño responsivo

5. Instalación y Ejecución del Proyecto

- Instalación de dependencias
- Ejecución en entorno local
- Despliegue
- Panel de fisioterapeuta
- Vista responsiva

1. Introducción

- **Objetivo del proyecto**

El proyecto tiene como finalidad apoyar terapias de rehabilitación mediante el uso de un guante inteligente, diseñado para recolectar datos en tiempo real durante las sesiones de fisioterapia. A través de una plataforma web, se busca facilitar el seguimiento del progreso de los pacientes y optimizar el trabajo de los fisioterapeutas mediante herramientas tecnológicas.

- **Público objetivo**

Está dirigido principalmente a dos tipos de usuarios:

- Pacientes en rehabilitación que requieren monitoreo constante de sus avances.
- Fisioterapeutas encargados de supervisar, analizar y retroalimentar el tratamiento de cada paciente.

- **Alcance**

El sistema incluye un guante con sensores que mide los movimientos de la mano y transmite los datos a una base de datos en Firebase mediante un script en Python ejecutado desde Arduino. Estos datos son visualizados en una plataforma web construida con Next.js. La página ofrece acceso personalizado: los pacientes pueden ver gráficas de su progreso, comentarios del especialista y descargar reportes en PDF; mientras que los fisioterapeutas pueden gestionar una lista de pacientes y revisar sus sesiones individuales con datos en tiempo real.

2. Tecnologías Utilizadas

- **Frontend:**

Next.js, React, JavaScript

El proyecto combina tecnologías modernas para el frontend, backend, estilos y manejo de datos. A continuación, se detallan:

Frontend

- Next.js: Framework de React que permite renderizado del lado del servidor (SSR) y generación de sitios estáticos (SSG). Se utiliza **para** construir la interfaz de usuario y manejar rutas dinámicas. Ejemplo de uso en el código:

```
import { useRouter } from 'next/navigation';  
const router = useRouter();
```

Este snippet muestra cómo se usa el enrutamiento dinámico para navegar entre páginas.

React: Biblioteca principal para construir componentes reutilizables y gestionar el estado de la aplicación. Hooks como useState, useEffect, y useCallback son fundamentales:

```
const [user, setUser] = useState(null);
useEffect(() => {
  const checkAuthState = async () => {
    const storedUser = localStorage.getItem('user');
    if (storedUser) setUser(JSON.parse(storedUser));
  };
  checkAuthState();
}, []);
```

● Backend:

API Routes de Next.js (Serverless Functions)

API Routes de Next.js (Serverless Functions): Las rutas API se implementan como funciones sin servidor dentro de Next.js, manejando solicitudes como registro, inicio de sesión y gestión de datos. Aunque el código compartido (usuariosRutas.js) usa Express, en el contexto de Next.js se integran rutas como /api/login:

```
const response = await fetch(`${API_BASE_URL}/api/login`, {
  method: 'POST',
  headers: { 'Content-Type': 'application/json' },
  body: JSON.stringify({ email, password }),
});
```

● Estilos:

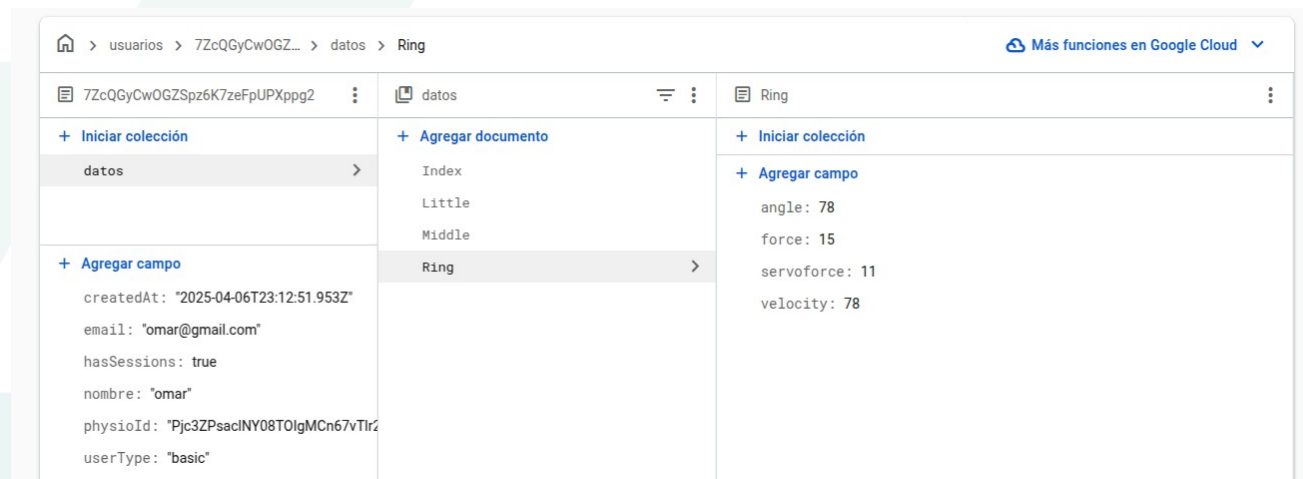
CSS, CSS Modules

CSS: Estilos globales y específicos se aplican mediante archivos CSS, como globalStyles.css.

CSS Modules: Aunque no se ve explícitamente en el código proporcionado, Next.js soporta módulos CSS para estilos encapsulados por componente.

● Base de datos:

Firestore



i Las siguientes funciones de autenticación dejarán de estar disponibles cuando se cierre Firebase Dynamic Links el 25 de agosto de 2025: la autenticación a través de vínculos de correo electrónico en apps para dispositivos móviles, así como la compatibilidad con OAuth de Cordova para apps web.

🔍 Buscar por dirección de correo electrónico, número de teléfono o UID de usuario

Agregar usuario



Identificador	Proveedores	Fecha de creación ↓	Fecha de acceso	UID de usuario
omar@gmail.com	✉	6 abr 2025		7ZcQGyCwOGZSpz6K7zeFpU...
fisio2@gmail.com	✉	6 abr 2025	6 abr 2025	vAIT2qHFXTXYsSOht6gB99...
trovatore@gmail.com	✉	6 abr 2025	6 abr 2025	UwoXFSC33USgjmM1NYikX8...
test@gmail.com	✉	6 abr 2025	6 abr 2025	iMbKE13ss6X3hKRxNLYR1jJt...
patient@gmail.com	✉	6 abr 2025	6 abr 2025	pvK6GsBJFXPiQbFG6Qelwgry...
verdi@gmail.com	✉	6 abr 2025	6 abr 2025	CQdsJaKab7QuDoOKASLxTJl5...
patient0@gmail.com	✉	6 abr 2025	6 abr 2025	OaZJc42gPudy6i4E9JqPkF26...
patient1@gmail.com	✉	6 abr 2025	6 abr 2025	XFDYs1OAxRUxz8ZJF57Edck...
francisco@gmail.com	✉	6 abr 2025	6 abr 2025	TPyp3z7aucbBIP5xgLMt84n1...
francis@gmail.com	✉	5 abr 2025	5 abr 2025	F64Rmn3uJES1eh8cHyyFLLaw...

HEMWICK

Plan Spark

🗨️ ¿Todo listo para comenzar? Cuéntale a Gemini sobre tu proyecto

13 apps

flexi (android)

flexi (ios)

login_ui (android)



Compilación

Firestore

Operaciones de lectura (actual)

22 k +956.4%



Operaciones de escritura (actual)

2.4 K +7,356.3%



— Esta semana — La semana pasada

Firebase: Utiliza Firestore para almacenamiento de datos y Firebase Authentication para la gestión de usuarios. Ejemplo de estructura en usuariosRutas.js:

```
await db.collection('usuarios').doc(userRecord.uid).set({
  nombre,
  email,
  userType: 'basic',
  createdAt: new Date().toISOString(),
  hasSessions: false,
});
```

● Librerías:

Axios, SWR, Formik, Yup

Axios: Para realizar solicitudes HTTP al backend (aunque en el código se usa fetch nativo).

SWR: Para manejo de datos en caché y actualizaciones en tiempo real (no visible en el código proporcionado, pero común en Next.js).

Formik y Yup: Para formularios y validación (no usado en el código actual, pero recomendado).

Highcharts: Generación de gráficos interactivos:

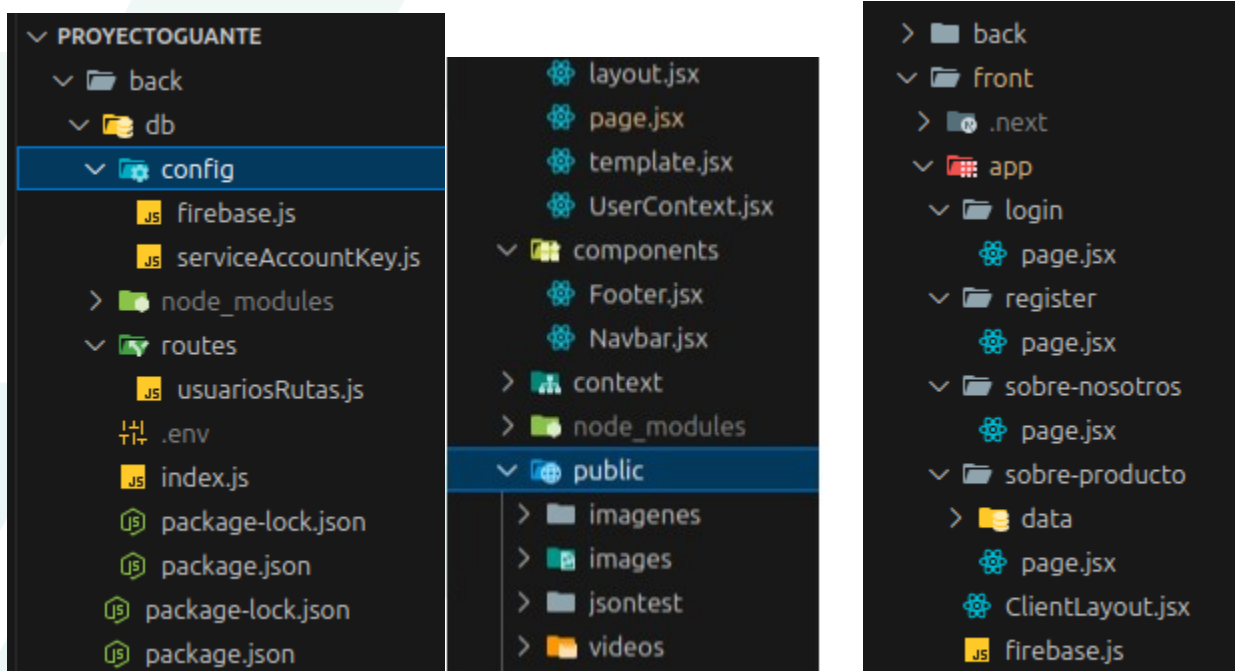
```
import Highcharts from 'highcharts';
import HighchartsReact from 'highcharts-react-official';
```

jsPDF: Creación de reportes en PDF


```
const doc = new jsPDF();
doc.text('Informe de Progreso', 105, 30, { align: 'center' });
```

3. Estructura del Proyecto

- Organización de carpetas



front/: Contiene el código del frontend, construido con Next.js.

- **app/:** Carpeta principal de Next.js (versión con App Router) que contiene las páginas y layouts.
 - **page.jsx:** Página principal del proyecto, probablemente equivalente a Home.js que me compartiste anteriormente.
 - **login/:** Carpeta que define la ruta /login.
 - **page.jsx:** Página de inicio de sesión, donde los usuarios ingresan sus credenciales.

- **register/**: Carpeta para la ruta /register.
 - **page.jsx**: Página de registro, donde los usuarios (pacientes o fisioterapeutas) crean una cuenta.
- **sobre-nosotros/**: Carpeta para la ruta /sobre-nosotros.
 - **page.jsx**: Página que muestra información sobre el equipo detrás del proyecto.
- **sobre-producto/**: Carpeta para la ruta /sobre-producto.
 - **page.jsx**: Página que describe el guante inteligente y sus características.
- **data/**: Carpeta que podría contener datos estáticos o componentes relacionados con la visualización de datos.
 - **page.jsx**: Posiblemente una página para mostrar datos generales o un dashboard.
- **ClientLayout.jsx**: Componente de layout que envuelve las páginas para agregar funcionalidad común, como un proveedor de contexto o autenticación.
- **firebase.js**: Archivo de configuración de Firebase, donde se inicializan los servicios de Firestore y Authentication.
- **layout.jsx**: Layout raíz de la aplicación, que define la estructura general (como el Navbar y el Footer).
- **page.jsx**: Página raíz (/), que actúa como el punto de entrada principal.
- **template.jsx**: Plantilla base para las páginas, posiblemente usada para aplicar estilos o componentes comunes.
- **UserContext.jsx**: Contexto de React para manejar el estado global del usuario autenticado.
- **components/**: Componentes reutilizables.
 - **Footer.jsx**: Componente del pie de página.

- **Navbar.jsx**: Componente de la barra de navegación, que incluye lógica para cerrar sesión y actualizar datos.
- **public/**: Recursos estáticos.
 - **images/**: Imágenes utilizadas en la aplicación, como las de los componentes del guante (raspberry.jpg, sensoresflexibles.png, etc.).
 - **iontest/**: Posiblemente una carpeta para pruebas o recursos adicionales.
 - **videos/**: Videos, como BACKGROUND.webm, usado en la página principal.
 - **globe.svg, next.svg, vercel.svg, window.svg**: Archivos SVG para íconos o gráficos.
- **src/**: Código fuente adicional (aunque parece vacía en la estructura).
- **styles/**: Archivos CSS para estilos globales y específicos, como globalStyles.css.

back/: Contiene el código del backend, implementado con Express.

- **db/**: Carpeta para la base de datos (probablemente vacía, ya que se usa Firebase).
- **config/**: Configuraciones del backend, como la inicialización de Firebase.
- **node_modules/**: Dependencias del backend instaladas con npm.
- **routes/**: Rutas del backend.
 - **usuariosRutas.js**: Define las rutas para registro, inicio de sesión, gestión de pacientes, notificaciones y más.
- **.env**: Archivo de variables de entorno, donde se almacenan las claves de Firebase y otras configuraciones.
- **index.js**: Punto de entrada del backend, donde se configura el servidor Express.

- **package-lock.json** y **package.json**: Archivos de configuración de dependencias para el backend.

- **Archivos principales**

front/app/page.jsx: Página principal que maneja la lógica de autenticación, visualización de gráficos y gestión de pacientes/usuarios. Equivalente a Home.js que me compartiste.

front/app/ClientLayout.jsx: Layout que envuelve las páginas para agregar funcionalidad común, como un proveedor de contexto.

front/app/firebase.js: Configuración de Firebase para Firestore y Authentication.

front/components/Navbar.jsx: Componente de navegación que incluye lógica de cierre de sesión y actualización de datos.

back/routes/usuariosRutas.js: Define las rutas del backend para registro, inicio de sesión, gestión de pacientes, notificaciones y obtención de datos de progreso.

back/index.js: Archivo principal del backend que inicializa el servidor Express y configura las rutas.

4. Funcionalidades Principales

Registro e inicio de sesión

El sistema permite a los usuarios registrarse e iniciar sesión con roles diferenciados (physio y basic). En usuariosRutas.js:

```
router.post('/register', async (req, res) => {
  const { email, password, nombre, userType } = req.body;
  if (!email || !password || !nombre || !userType) {
    return res.status(400).json({ success: false, error: 'Faltan campos requeridos.' });
  }
  userRecord = await auth.createUser({ email, password, displayName: nombre });
  if (userType === 'physio') {
    await db.collection('fisioterapeutas').doc(userRecord.uid).set({
      nombre,
      email,
      userType: 'physio',
      createdAt: new Date().toISOString(),
    });
  } else {
    await db.collection('usuarios').doc(userRecord.uid).set({
      nombre,
      email,
      userType: 'basic',
      createdAt: new Date().toISOString(),
      hasSessions: false,
    });
  }
  res.status(201).json({ success: true, message: 'Usuario registrado', uid: userRecord.uid });
});
```

Los pacientes (basic) solo pueden registrarse si están autorizados por un fisioterapeuta, lo que se verifica en la colección usuariosPermitidos. Si el

usuario es un paciente, se inicializan subcolecciones (datos) para almacenar las métricas de sus sesiones.

```
router.post('/login', async (req, res) => {
  const { email, password } = req.body;
  const userRecord = await auth.getUserByEmail(email);
  const userDoc = await db.collection('fisioterapeutas').doc(userRecord.uid).get();
  let userData = {};
  let userType = 'physio';
  if (!userDoc.exists) {
    const basicUserDoc = await db.collection('usuarios').doc(userRecord.uid).get();
    userData = basicUserDoc.data();
    userType = 'basic';
  } else {
    userData = userDoc.data();
  }
  res.json({
    success: true,
    uid: userRecord.uid,
    user: { ...userRecord.toJSON(), ...userData, userType },
  });
});
```

Verifica las credenciales mediante Firebase Authentication y devuelve los datos del usuario, incluyendo su tipo (physio o basic).

● Gestión de pacientes

Los fisioterapeutas pueden agregar pacientes y revisar sus datos:

```
router.post('/add-patient', async (req, res) => {
  const { physioId, patientName, patientEmail } = req.body;
  const patientData = {
    nombre: patientName,
    email: patientEmail,
    physioId,
    userId: patientUserId || null,
    createdAt: new Date().toISOString(),
    observaciones: [],
  };
  await db.collection('pacientes').doc(patientEmail).set(patientData);
  await db.collection('notifications').doc().set({
    recipientId: physioId,
    message: `Nuevo paciente registrado: ${patientName} (${patientEmail})`,
    timestamp: new Date().toISOString(),
    read: false,
  });
});
```

Los fisioterapeutas agregan pacientes, que se registran en las colecciones pacientes y physio_patients. También se crea una entrada en usuariosPermitidos para autorizar al paciente a registrarse.

Obtener Lista de Pacientes:


```
router.get('/patients/:physioId', async (req, res) => {
  const physioPatientsQuery = await
db.collection('physio_patients').where('physioId', '==', physioId).get();
  const patientsList = [];
  for (const docSnap of physioPatientsQuery.docs) {
    const patientEmail = docSnap.data().patientId;
    const userQuery = await db.collection('usuarios').where('email', '==',
patientEmail).get();
    let userId = null;
    if (!userQuery.empty) userId = userQuery.docs[0].id;
    patientsList.push({ id: patientEmail, email: patientEmail, ...patientData,
userId });
  }
  res.json({ success: true, patients: patientsList, totalSessions });
});
```

Panel de Administración

Para fisioterapeutas, el panel incluye:

- **Lista de Pacientes:** Muestra todos los pacientes registrados, con indicadores de si están registrados y cuántas sesiones tienen.
- **Gráficos de Progreso:** Visualización de métricas (ángulo, fuerza, fuerza del servo, velocidad) para cada paciente.
- **Observaciones:** Los fisioterapeutas pueden agregar observaciones personalizadas

```
router.post('/add-observation', async (req, res) => {
  const { physioId, patientEmail, observation } = req.body;
  const patientDocRef = db.collection('usuariosPermitidos').doc(patientEmail);
  const currentObservaciones = patientDoc.data().observaciones || [];
  const newObservacion = {
    text: observation,
    fechaObservacion: new Date().toISOString(),
    physioId,
  };
  await patientDocRef.update({ observaciones: [...currentObservaciones, newObservacion] });
});
```

● Diseño responsivo



RECOVGLOX

Una solución avanzada para la rehabilitación de manos. Monitorea tu progreso, mejora tu movilidad y recupera tu fuerza con tecnología de punta.

Registrarse

¿Ya tienes cuenta? [Inicia sesión](#)

RECOVGLOX 



RECOVGLOX

Una solución avanzada para la rehabilitación de manos. Monitorea tu progreso, mejora tu movilidad y recupera tu fuerza con tecnología de punta.

Registrarse



 **Registrarse**

5. Instalación y Ejecución del Proyecto

Requisitos previos

Node.js: Versión 16 o superior, necesario para ejecutar tanto el frontend como el backend.

Firebase CLI: Para configurar y desplegar Firebase Functions (si se usan) y gestionar la base de datos.

Cuenta de Firebase: Configurada con Firestore y Authentication habilitados.

Hardware: Raspberry Pi 4, sensores flexibles, MPU6050, servomotores SM-S4306R, batería recargable, y un guante de tela o neopreno para montar los componentes.

Python: Para el script que envía datos desde la Raspberry Pi a Firebase.

- **Instalación de dependencias**

```
cd front
npm install
npm install highcharts highcharts-react-official jspdf next react react-dom
```

```
cd back
npm install
npm install express firebase-admin
```

- **Ejecución en entorno local**

Crear un archivo `.env.local` en `front/` con las claves de Firebase:

```
NEXT_PUBLIC_FIREBASE_API_KEY=your-api-key  
NEXT_PUBLIC_FIREBASE_AUTH_DOMAIN=your-auth-domain  
NEXT_PUBLIC_FIREBASE_PROJECT_ID=your-project-id
```

Crear un archivo .env en back/ con las credenciales de Firebase Admin:

```
FIREBASE_PROJECT_ID=your-project-id  
FIREBASE_PRIVATE_KEY=your-private-key  
FIREBASE_CLIENT_EMAIL=your-client-email
```

● Despliegue

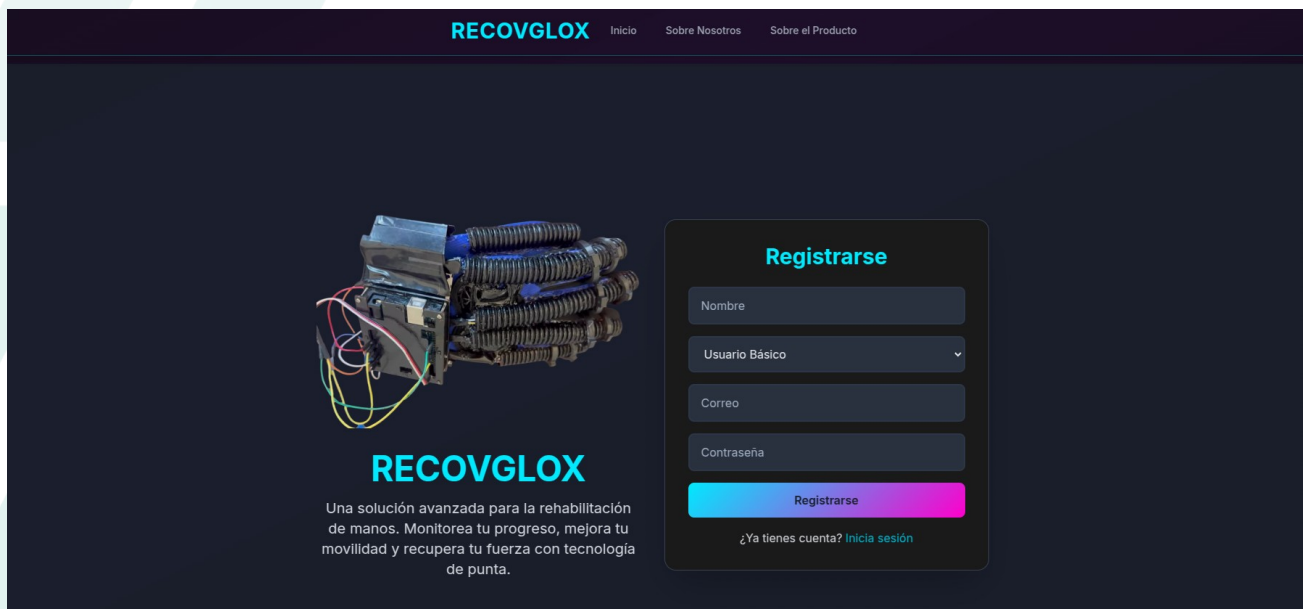
BACKEND:

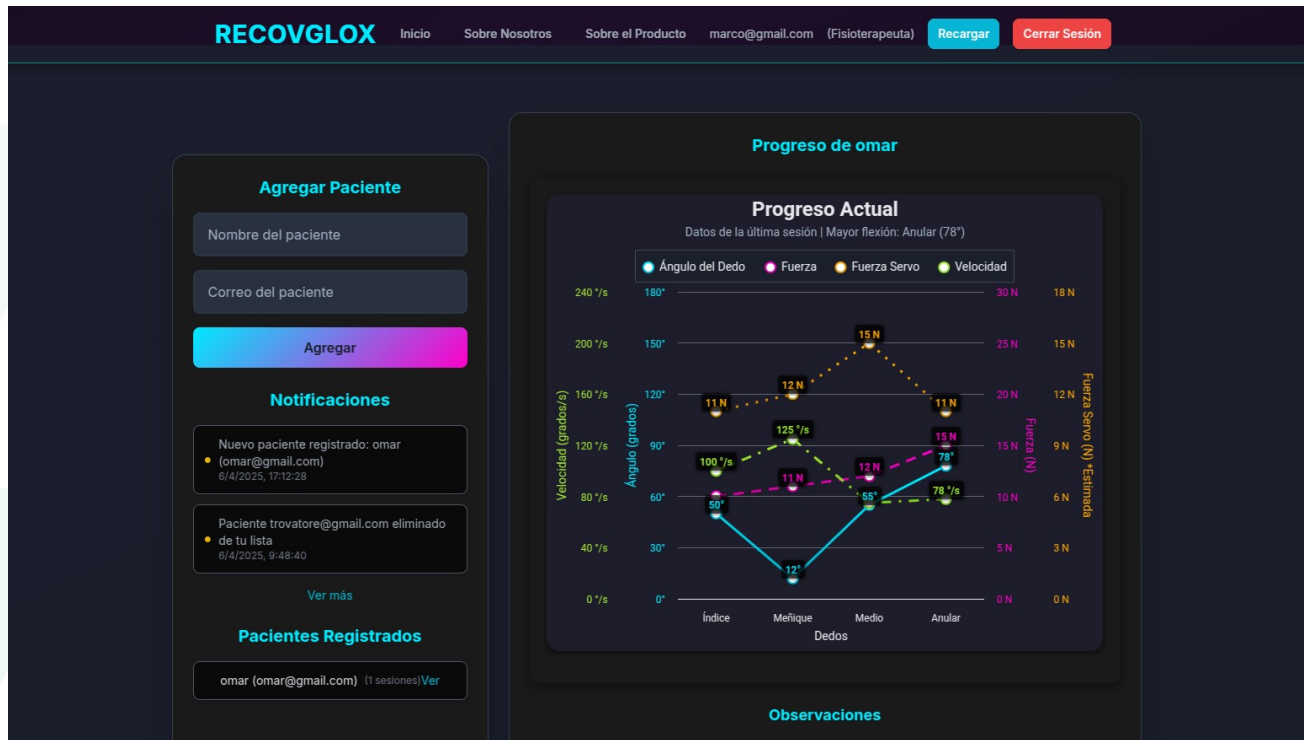
```
cd back  
node index.js
```

FRONTEND:

```
cd front  
npm run dev
```

6. Capturas de Pantalla






● Panel de fisioterapeuta

● Vista responsiva



RECOVGLOX 

Agregar Paciente

Nombre del paciente

Correo del paciente

Agregar

Notificaciones

Nuevo paciente registrado: omar
• (omar@gmail.com)
6/4/2025, 17:12:28

Paciente trovatore@gmail.com
• eliminado de tu lista
6/4/2025, 9:48:40

Ver más

Pacientes Registrados

omar (omar@gmail.com)
(1 sesiones)
Ver