Documentación Técnica AthletIA

1. Descripción del Diseño de la Interfaz

1.1. Estructura de la Aplicación

Arquitectura general:

- **Patrón de arquitectura:** Aplicación SPA (Single Page Application) basada en componentes React
- **Enrutamiento:** React Router DOM v6 con rutas protegidas
- **Gestión de estados:** React Context API (AuthContext, AccessibilityContext)
- **Cliente HTTP:** Axios con interceptores para manejo de tokens

Estructura de carpetas:

```
Frontend/src/
├── components/          # Componentes reutilizables
│   ├── layout/          # Layout principal (Sidebar, Layout)
│   ├── Users/           # Componentes específicos de usuarios
│   └── [otros]          # ThemeProvider, ProtectedRoute, etc.
├── context/             # Contextos de React (Auth, Accessibility)
├── lib/                 # Utilidades y configuración (api.ts, stores)
├── pages/               # Páginas/vistas de la aplicación
│   ├── Exercises/       # Módulo de ejercicios
│   └── Routines/        # Módulo de rutinas
└── App.tsx              # Configuración de rutas principal
```

Componentes principales:

- **Layout:** Wrapper principal que incluye Slidebar y área de contenido.
- **Slidebar:** Navegación lateral responsiva con menú hamburguesa en móvil.
- **ProtectedRoute:** Guard que protege rutas privadas verificando autenticación.
- **ThemeProvider/ThemeToggle:** Sistema de tema claro/oscuro.
- **AccessibilityButton:** Botón flotante para opciones de accesibilidad.

1.2. Estilos y Sistema de Diseño

Framework CSS: Tailwind CSS v3.4.15

Paleta de colores definida:

```
colors: {
  primary: '#13a4ec',              // Azul principal
  'background-light': '#f6f7f8',   // Fondo claro
  'background-dark': '#101c22',    // Fondo oscuro
}
```

Modo oscuro: Implementado con class strategy de Tailwind

- Persistencia en localStorage
- Detecta preferencia del sistema al primer render
- Toggle manual disponible

Tipografía:

- **Fuente principal:** Inter (sans-serif)
- **Escalado responsivo:** 95% en móviles, 100% por defecto, 105% en pantallas grandes (>1920px)

Espaciado y diseño responsivo:

- Mobile-first approach
- Breakpoints de Tailwind: sm (640px), md (786px), lg (1024px), xl (1280px)
- Slidebar colapsa en móvil con menú hamburguesa
- Padding adaptativo: 4(movil) -> 6(tablet) -> 10(desktop)

1.3. Dependencias Principales

Producción:

- react v19.2.0 y react-dom – Framework UI
- react-router-dom v6.28.0 – Enrutamiento
- axios v1.7.9 – Cliente HTTP
- react-hook-from v7.52.1 – Manejo de formularios
- zod v3.23.8 – Validacion de esquemas
- @hookfrom/resolvers – Integracion Zod + React Hook From
- lucide-react v.0.556.0 – Iconos
- sweetalert2 v11.14.5 – Alertas/modales

Desarrollo:

- vite v7.2.4 – Bundler y dev server
- typescript v5.9.3 – Tipado estático

- tailwindcss v3.4.15 – Framework CSS
- @tailwindcss/forms – Estilos para formularios
- @tailwindcss/container-queries – Container queries

2. Estándares de Diseño Aplicados

2.1. Nomenclatura CSS

Convenciones Tailwind:

- Utiliti-first approach
- Clases descriptivas inline: bg-white dark:bg-[#111c22]
- Prefijos para variants: dark, hover:, focus
- Colores custom con valores hexadecimales: dark:bg-[#1a2831]

2.2. Estructura de Componentes

Organización:

- Componentes funcionales con hooks
- Un componente por archivo
- Export default al final
- Props tipadas con TypeScript

Separación de concerns:

- Lógica de negocio en contextos/hooks
- Componentes de presentación en /components
- Paginas completes en /pages
- Utilidades en /lib

2.3. Accesibilidad (WCAG 2.1 Level AA)

Implementaciones:

- Labels en todos los inputs
- Atributos ARIA
- Focus visible con outline personalizado
- Navegación por teclado
- Contraste de colores adecuado
- Roles semánticos
- Soporte para lectores en pantalla

Funciones de accesibilidad avanzadas implementadas:

- Tamaños de letra
- Escala de grises
- Alto contraste
- Contraste negativo
- Fuente legible
- Enlaces subrayados
- Lectura de pagina

3. Instrucciones para Ejecutar Localmente

3.1. Requisitos Previos

- Node.js >= 18.0.0
- Npm >= 9.0.0

3.2. Instalación

3.2.1. Navegar al directorio del frontend

- cd Frontend

3.2.2. Instalar dependencias

- npm install

3.3. Configuración

3.3.1. Copiar archivos de ejemplo

- Copy .env.example .env

3.3.2. Editar .env con las URLs correctas

Nota: Si el Backend corre en un puerto diferente, actualiza VITE_API_URL

3.4. Ejecutar en Modo Desarrollo

npm run dev

- La aplicación se abrirá en https://localhost:5173
- Hot-reload activado (cambios se reflejan automáticamente)

3.5. Conectar con el Backend

Paso 1: Asegúrate de que el Backend este corriendo

- Cd Backend
- Npm run start:dev

Paso 2: Verifica la conexión

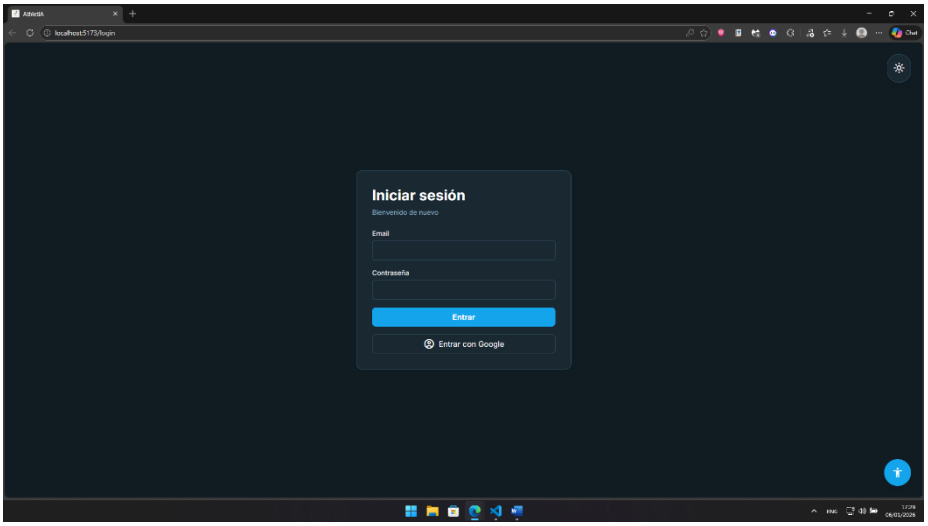
- Backend debe estar en http://localhost:3000
- Frontend debe tener VITE_API_URL=http://localhost:3000 en .env

Paso 3: Probar autenticación

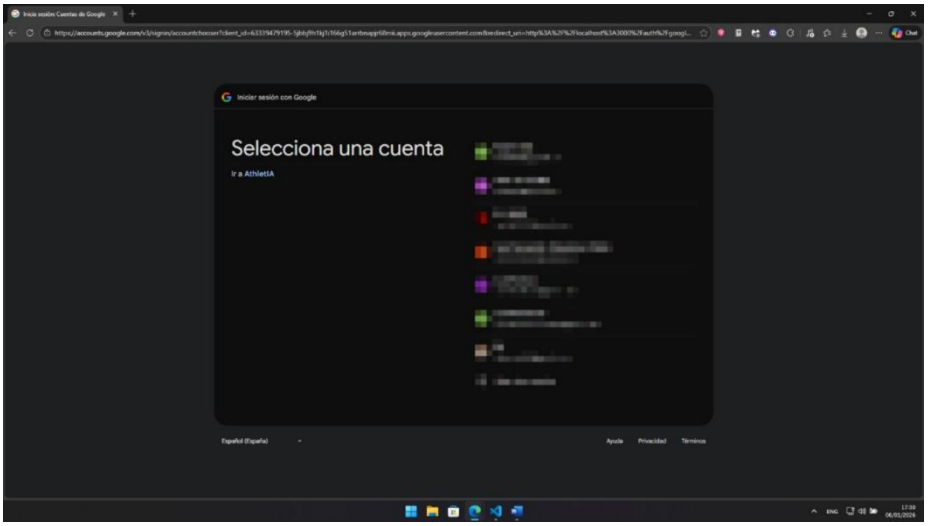
- Ve a https://localhost:5173/login
- Intenta loguear con credenciales o Google OAuth

4. Capturas de aplicación funcional

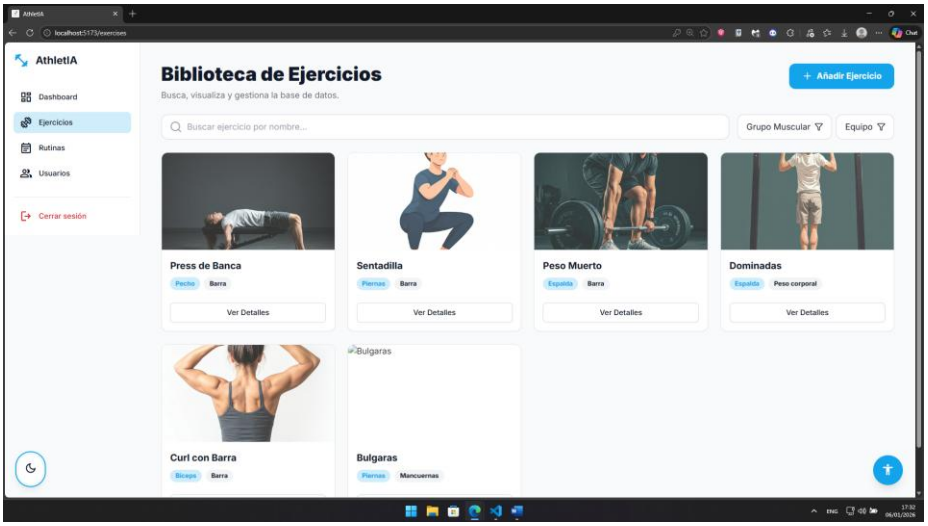
Login



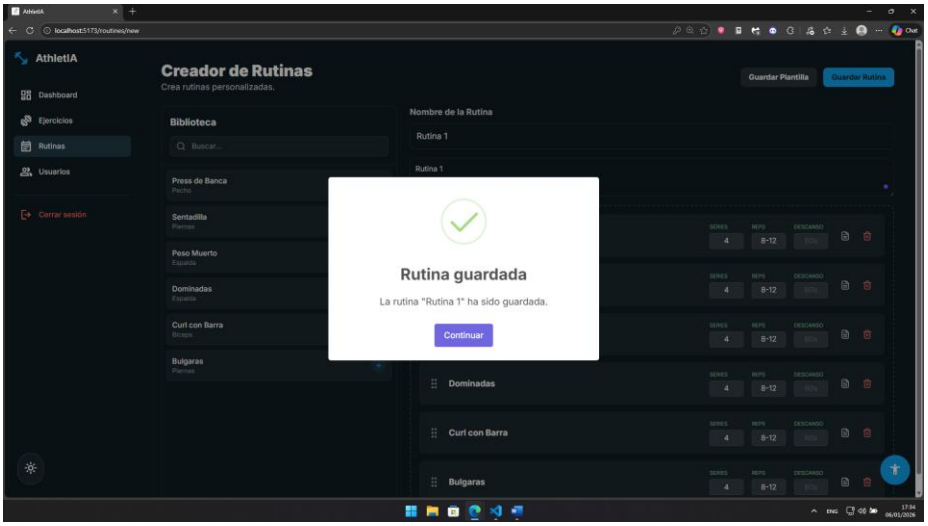
Autenticación con Google



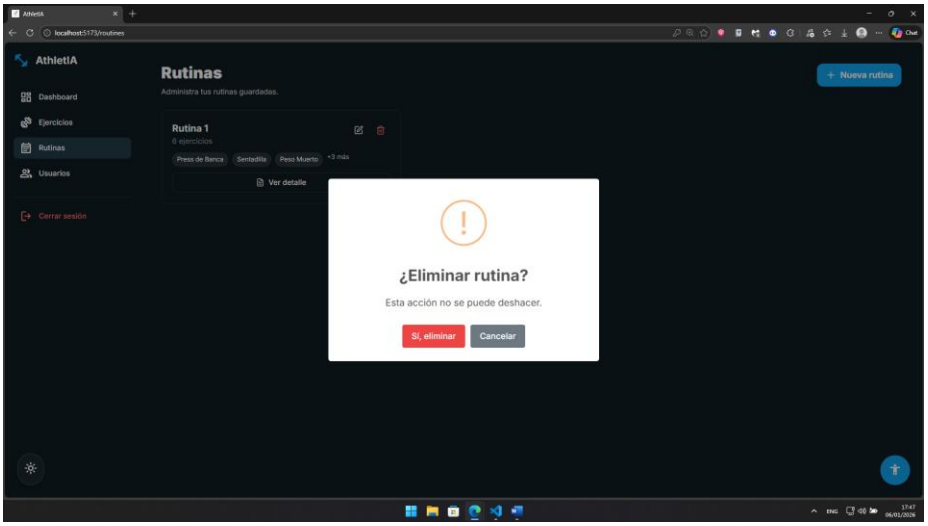
Pantalla de Ejercicios aplicando tema claro

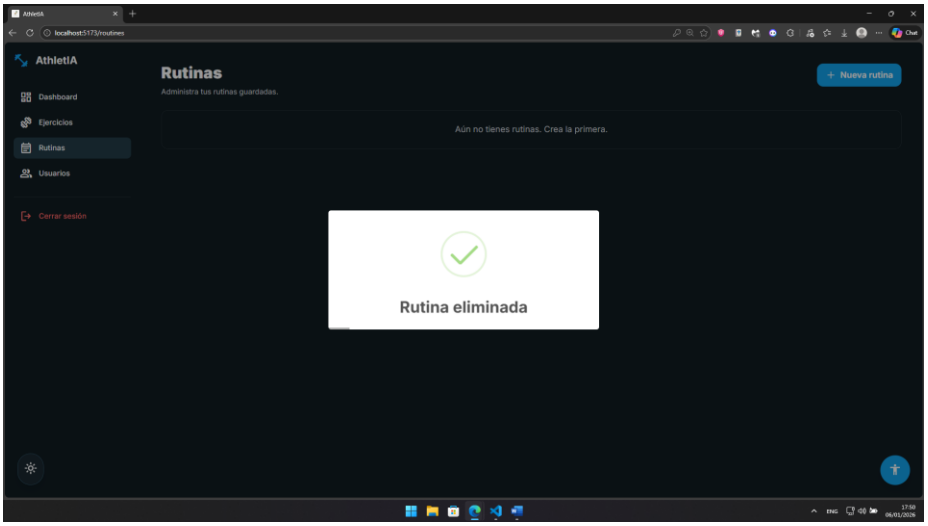


Creación de rutina para visualización de alerta

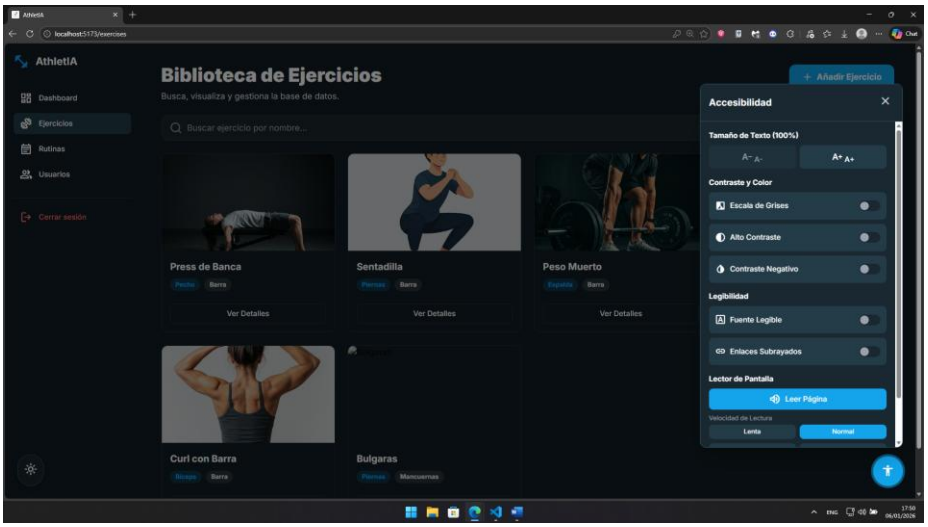


Eliminación de rutina para visualización de alerta

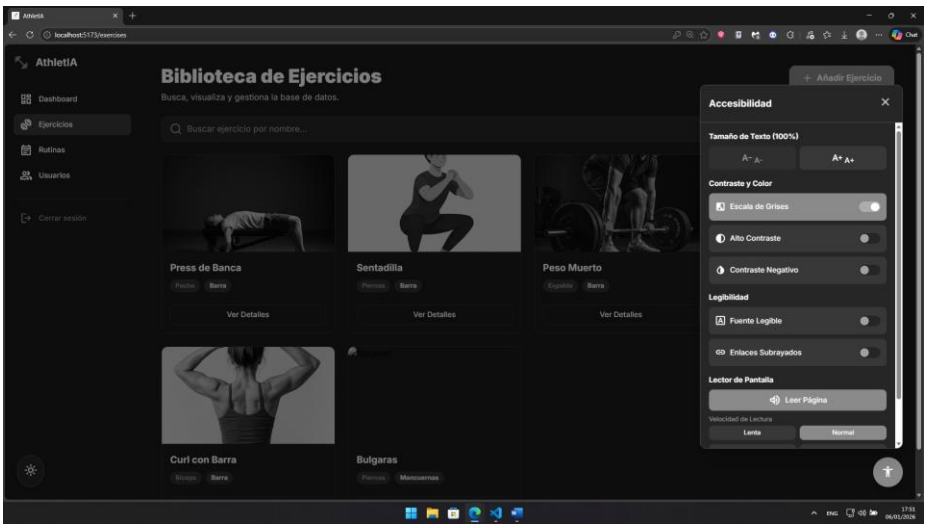




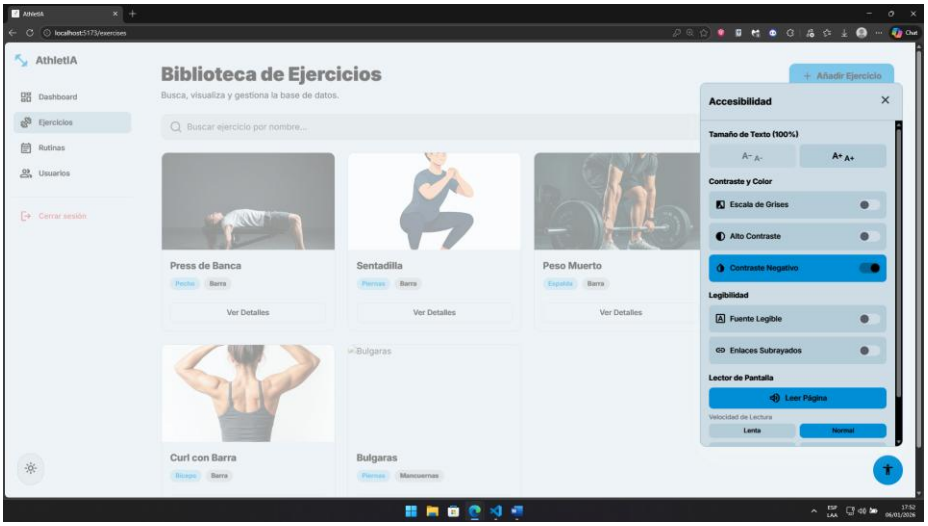
Menú de Accesibilidad



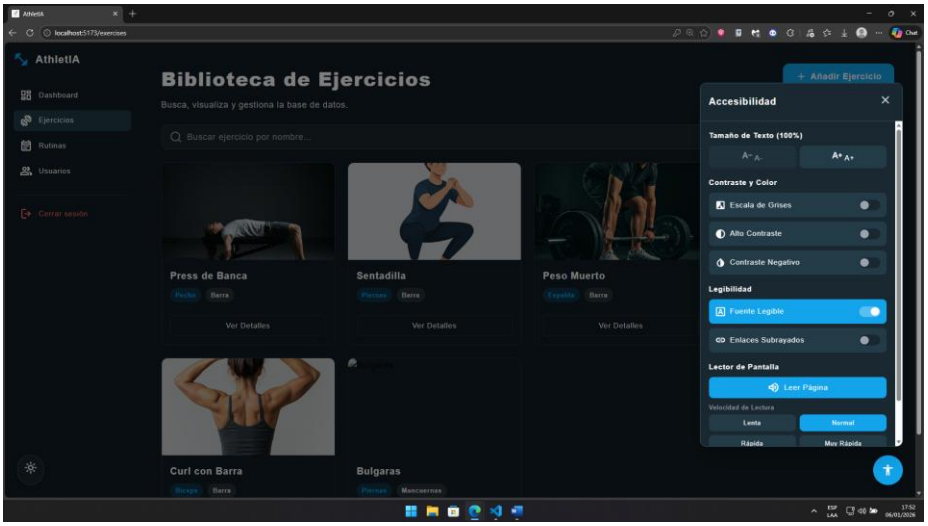
Escala de Grises



Contraste negativo



Fuente Legible



Enlaces Subrayados