Documentacion FronEnd

README.md

> Nombre del proyecto: Recicla DUOC (Frontend)
>
> **Stack principal:** React 19, React Router 7, Vite 7, Tailwind CSS 4, Axios, React Hook Form, Zod.

📌 Resumen

♻️ ReciclaDUOC – App de reciclaje para estudiantes de DuocUC

ReciclaDUOC es una aplicación diseñada para incentivar el reciclaje entre los estudiantes de **DuocUC**. Los usuarios pueden **iniciar sesión** y **registrar sus actividades de reciclaje** mediante una **foto del material reciclado**, la cual es analizada por un sistema de **inteligencia artificial** que **clasifica el tipo de residuo** y **asigna puntos** según su aporte ambiental.

Estos puntos se **acumulan en el perfil del usuario** y, en el futuro, podrán **canjearse por premios o beneficios** dentro de la comunidad estudiantil.

La SPA Integra autenticación con JWT y consumo de API REST.

- **Objetivo principal:** La aplicación busca **promover la conciencia ecológica** y **motivar hábitos sostenibles** a través de la gamificación y la tecnología.
- **Público objetivo:** Estudiantes de DuocUC
- **Estado:** Dev
- **Diseño (Figma):** *N/A*

🏗️ Arquitectura (alto nivel)

```
frontend (React + Vite)
 ├─ src/
 │  ├─ app/
 │  │  └─ routes/AppRoutes.jsx
 │  ├─ auth/
 │  │  ├─ api/AuthContext.jsx
 │  │  └─ pages/{Login.jsx, Register.jsx}
 │  ├─ features/
 │  │  ├─ MainMenu.jsx
 │  │  ├─ home/Home.jsx
 │  │  ├─ rewards/rewards.jsx
 │  │  └─ profile/profileScreen.jsx
 │  ├─ components/Statics/{header.jsx, navbar.jsx}
 │  ├─ assets/Icons/* (SVG/PNG)
 │  └─ App.jsx
```

```
├─ public/
└─ tests/
```

- **Gestión de estado:** Context API propio para auth (`AuthContext.jsx`).
- **Data fetching:** Axios con instancia configurada a `VITE_API_URL` y header `Authorization` con JWT.
- **Ruteo:** React Router 7 con rutas protegidas (`ProtectedRoute`).
- **UI:** Tailwind CSS 4 (utilidades) + íconos locales vía alias `@icons`.

⚙️ Requisitos

- Node: `>= 18.x` (LTS recomendado)
- Gestor de paquetes: PNPM/NPM/Yarn

🚀 Inicio rápido

```
# 1) Clonar
git clone https://github.com/CojitoJulio/Gutierrez_Cribillero_Lopez.git
cd Gutierrez_Cribillero_Lopez

# 2) Instalar deps
npm install   # o pnpm install / yarn

# 4) Ejecutar en dev
npm run dev   # abre http://localhost:5173

# 5) Build de producción
npm run build
npm run preview
```

🔑 Variables de entorno

Crea `.env` .

```
VITE_API_URL=http://localhost:3000
DATABASE_URL = "libsql://recicladuoc-fhantomdev.aws-us-west-2.turso.io"
DATABASE_TOKEN = "eyJhbGciOiJFZERTQSIsInR5cCI6IkpXVCJ9.eyJhIjoicnciLCJpYXQiOjE3NTc3MD
M2NTAsImlkIjoiZjg4NGM5MjMtYjE4OC00YWJkLWFmMWYtNTBiYzZkYzE3NTAxIiwicmlkIjoiYTkzNW
MyNTYtYzcwNS00N2I5LWJjZjUtN2EwMWU0NjQxZTJlIn0.DNtQcMmatRldurtW--MTmpaXUvBqS42L
MzhSBC9B4sknUT-6a9CA3213EDdwtnovp20SHXLnmCzCmXnOQNgeCQ"
JWT_SECRET = "gfb51f56b156fb15s6fb15sf1b5f1b561b61rewds"

SUPABASE_URL= "https://rpclohcrafputungoiya.supabase.co"
SUPABASE_KEY= "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzdXBhYmFzZSIsInJlZiI6InJw
```

```
Y2xvaGNyYWZwdXR1bmdvaXlhliwicm9sZSI6InNlcnZpY2Vfcm9sZSIsImlhdCI6MTc1OTIwMjA1MCwiZX
hwIjoyMDc0Nzc4MDUwfQ.uIEH1FrZSt1ftrMHzIsEIKOyiqvyB6NTnBFaCaVW9c8"
FRONT_END = "http://localhost:5173"
```

🧭 Ruteo

Rutas definidas en `src/app/routes/AppRoutes.jsx`:

Ruta	Componente	Protegida	Layout
`/login`	`auth/pages/Login`	No	—
`/register`	`auth/pages/Register`	No	—
`/`	`features/MainMenu`	Sí	MainMenu
`/(index)`	`features/home/Home`	Sí	MainMenu
`/rewards`	`features/rewards/rewards`	Sí	MainMenu
`/profile`	`features/profile/profileScreen`	Sí	MainMenu

- **Guard:** `ProtectedRoute` usa `useAuth()` para verificar `isAuthenticated` y redirige a `/login` conservando `location.state.from`.
- **Nota:** Dentro de `MainMenu` el contenido cambia con `<Outlet />`.

📂 Estructura de carpetas (detallada)

```
src/
  app/
    routes/
      AppRoutes.jsx
      ProtectedRoute.jsx
  auth/
    api/
      AuthContext.jsx  # contexto de auth, expone { user, token, isAuthenticated, login, logout, api }
    pages/
      Login.jsx        # pantalla de inicio de sesión
      Register.jsx     # pantalla de registro de usuario
  features/
    MainMenu.jsx       # layout móvil con Header + Outlet + Navbar
    home/
      Home.jsx         # pantalla principal: CTA "Reciclar", puntos, ranking
    rewards/
      rewards.jsx      # listado/canje de premios
    profile/
      profileScreen.jsx# perfil del usuario, con historial de reciclaje
  components/
    Statics/
      header.jsx       # obtiene perfil si hay token
```

```
    navbar.jsx       # navegación inferior con íconos
assets/
  Icons/           # SVG/PNG (HomeIcon, ChartIcon.png, etc.)
App.jsx
```

🔌 Servicios y API

Cliente Axios

- **Base URL:** `import.meta.env.VITE_API_URL || "http://localhost:3000"`.
- **Auth:** Si existe `token`, se inyecta en `Authorization: Bearer <token>`.

Endpoints usados por el front

- `POST /api/usuario/loginUsuario` — Autenticación. **Respuesta esperada:** `{ token, usuario }`.
- `GET /api/usuario/getPerfil` — Perfil del usuario autenticado. **Respuesta esperada:** `{ perfil: { ... } , reciclaje: [ { ... } ] }`.

> La instancia api se expone desde AuthContext y se reutiliza; header.jsx usa una instancia local con el mismo baseURL.

🔐 Autenticación (flujo)

1. **Login** (`AuthContext.login(email, password)`): hace POST a `/api/usuario/loginUsuario`.
2. **Guardar**: setea `token` en estado y `localStorage`; setea `api.defaults.headers.Authorization`.
3. **Sesión**: `isAuthenticated` es `!!token`.
4. **Perfil**: componentes pueden llamar `api.get('/api/usuario/getPerfil')` para datos del usuario.
5. **Logout**: limpia `user` y `token` y borra header Authorization.

🎨 UI

- **Tailwind CSS 4** para estilos utilitarios.
- Layout tipo "smartphone" en `MainMenu` con header fijo y navbar fija.
- Íconos: alias `@icons` definido en Vite (`vite.config.js`).

🧰 Scripts útiles (package.json)

```
{
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "lint": "eslint .",
    "preview": "vite preview"
```

```
  }
}
```

🧰 Convenciones y patrones

- Componentes en **PascalCase**, funciones y variables en **camelCase**.
- **Separación por features**: `features/*` con UI por dominio.
- **Context** para la autenticaci[on
- **Aliases**: `@` → `./src`, `@icons` → `./src/assets/Icons`.

Endpoints consumidos por el frontend

Recurso	Método	Ruta	Auth	Descripción	Respuesta esperada
LoginUsuario	POST	`/api/usuario/loginUsuario`	No	Autentica usuario y entrega JWT	`{ token, usuario }`
GetPerfil	GET	`/api/usuario/getPerfil`	Sí	Obtiene datos del usuario autenticado	`{ perfil: { ... } }`
Registro Usuario	POST	`/api/usuario/regsitroUsuario`	No	Registra al usuario en la BD	`{ mensaje: { ... } }`
UpdatePerfil	PUT	`/api/usuario/updatePerfil`	Si	Actualiza los datos del perfil autenticado	`{ mensaje: { ... }, perfil: { ... } }`
RegistroReciclaje	POST	`/api/reciclaje/registroReciclaje`	Si	Registra un nuevo reciclaje	`{ id_deposito: { ... }, foto: { ... }, productos: [ ... ]}`