Manual Técnico de ARASAAC Talk

Aplicación accesible de comunicación aumentativa que permite construir frases con pictogramas ARASAAC. Implementada con Flask, HTML/CSS/JS, IndexedDB y arquitectura PWA modular.

Estructura profesional del repositorio

ARASAAC-Talk/ app/ # Backend Flask modular **init**.py routes.py arasaac.py tts.py static/ # Frontend: CSS, JS, icons css/ style.css js/ main.js preload.js speech.js ui.js indexeddb.js sw-init.js service-worker.js icons/ icon-192.png icon-512.png maskable-512.png shortcut-frase.png shortcut-star.png templates/ # HTML base index.html preload/ # Pictogramas precargados pictos/ # Archivos PNG descargados categorias.json pictogramas.json metadata.json preload.py scripts/ # Herramientas externas preload_pictograms.py copiar_a_build.sh build_deb.sh start.sh manifest.webmanifest # PWA config app.py # Entrada principal de la app requirements.txt # Dependencias Python README.md # Documentación del proyecto .gitignore # Exclusión de archivos sensibles .gitattributes # Control de formato y binarios

---

Backend (Flask)

- Archivo principal: `app.py`
- Puerto dinámico vía `os.environ["PORT"]`
- Rutas:
  - `/` → Renderiza `index.html`
  - `/tts` → Genera audio con gTTS desde frase construida
  - `/preload/*` → Sirve pictogramas y JSON locales

Frontend Modular (`static/js/`)

Archivo	Rol
`main.js`	Orquestador global
`preload.js`	Carga pictogramas desde JSON
`speech.js`	Reproducción de audio (`/tts`)
`ui.js`	Manejo visual de botones y frase
`indexeddb.js`	Persistencia de favoritos
`sw-init.js`	Registro de Service Worker
`service-worker.js`	Cache, modo offline, fallback

Estilo y Accesibilidad

- CSS: `static/css/style.css` con diseño responsivo y accesible
- Etiquetas HTML con `aria-label`, roles semánticos y contraste visual
- Compatibilidad con lectores de pantalla y navegación por teclado

---

Progressive Web App

- `manifest.webmanifest` define:
  - Nombre, descripción, íconos, colores
  - Instalación en celulares y tablets
- Íconos adaptativos en `static/icons/`
  - `icon-192.png`, `icon-512.png`, `maskable-512.png`
  - `shortcut-frase.png`, `shortcut-star.png`
- `service-worker.js` para cache y carga sin red

Precarga Offline

- Script: `scripts/preload_pictograms.py`
  - Descarga pictogramas desde API ARASAAC
  - Genera: `categorias.json`, `pictogramas.json`, `metadata.json`
- Imágenes guardadas en `preload/pictos/`

Scripts de Desarrollo (`scripts/`)

Archivo	Descripción
`start.sh`	Arranque local con Flask
`preload_pictograms.py`	Precarga automática de pictogramas
`copiar_a_build.sh`	Prepara contenido para empaquetado
`build_deb.sh`	Crea `.deb` con `fpm`

Control de versión

- `.gitignore`: excluye `venv/`, caché, pictogramas crudos
- `.gitattributes`: normaliza `LF`, protege binarios `.png`, `.mp3`
- `requirements.txt`: instala Flask, gTTS y requests

Deploy en Railway

1. Subí el repo a GitHub
2. Vinculalo desde Railway
3. Railway detecta `requirements.txt`
4. Puerto gestionado automáticamente vía variable `PORT`
5. La app queda disponible en `https://arasaac-talk.up.railway.app`

Requisitos

- Python 3.9+
- Navegador compatible con PWA (Chrome, Firefox, Edge)
- Acceso a internet si no se usa precarga offline

Créditos

- Pictogramas: ARASAAC
- Voz: Google TTS
- Desarrollador: Hernán Luis Lang