LaunchIt Architecture Overview

Project

launchit / startuphunt

Generated

2025-11-20 08:40 UTC

Overview

Two-app architecture: a public-facing Vite/React frontend (startuphunt) and a Node/Express AI service (gptbackend). Supabase provides auth, storage, and the projects database; OpenAI and Microlink power auto-generated launch metadata.

Frontend (startuphunt)

- React + Vite SPA served via CDN
- Supabase anon client for auth/session
- Pages organized under Pages/, Components/, others/
- Lazy-load heavy routes; shared Header + Sidebar layout
- `/submit` form posts to AI backend; other views read directly from Supabase
- Analytics and performance helpers live in src/utils

Backend (gptbackend)

- Node 18 Express service with JSON + CORS middleware
- Rate limiting (per-IP, in-memory) and security headers
- `/generatelaunchdata` fetches target URL, calls Microlink for logo/thumbnail, then OpenAI gpt-4o-mini for structured data
- `/api/search/semantic` loads active projects from Supabase, ensures embeddings via OpenAI, and runs semanticSearch helper
- `/api/embeddings/generate` regenerates per-project vectors
- Uses Supabase service role key for DB access

Data Flow

1. User visits frontend on Netlify and downloads the React bundle.
2. Auth handled by Supabase anon key; sessions stored client-side.
3. `/submit` page POSTs to backend. Backend fetches site HTML, Microlink assets, and OpenAI for normalized JSON, then returns launch data.
4. Launch moderation/listing stored in Supabase `projects` table.
5. Search routes call backend semantic endpoint or direct Supabase queries.
6. Assets (logos/thumbnails) referenced via Microlink CDN and are not persisted locally.

Integrations & Env

- Supabase (auth, database, storage) via `VITE_SUPABASE_*` (frontend) and `SUPABASE_SERVICE_KEY` (backend).
- OpenAI via `OPENAI_API_KEY`.
- Microlink for screenshots and metadata.
- Deployment: frontend on Netlify (`netlify.toml`), backend on Render (`render.yaml`).

Performance & Reliability

- Lazy loading plus analytics/performance monitors on frontend.
- Backend bottlenecks: sequential Microlink/OpenAI calls and per-request Supabase fetches.
- Rate limit 50 requests per 15 minutes per IP.
- No job queue; long-running requests handled inline.
- Logging via `backend.log`; centralized tracing not yet set up.

Security Notes

- CORS restricted to launchit.site and Netlify preview domains.
- No user-level auth on backend endpoints yet; relies on obscurity.
- Supabase service key (full privileges) stored server-side.
- SSRF risk when fetching arbitrary URLs; prompt-injection surface when sending HTML to OpenAI.
- Microlink asset URLs expire; storing locally would improve reliability.