Ghost Empire / E-Forge

Dokumentacja dla developerów

Wielo-tenantowa, white-label platforma lojalnościowa dla streamerów. Next.js 16 · React 19 · Prisma 7 · Auth.js v5 · PostgreSQL · Vercel.

Wygenerowano automatycznie z kodu źródłowego · wersja Phase 3D

1. Przegląd

Widzowie zdobywają tokeny (GT) za czat, oglądanie, suby i donejty na Twitch/Kick/YouTube/Discord, a wydają je w sklepie, kasynie, predykcjach, eventach i battle passie. Jeden kod obsługuje wiele portali (white-label): założyciel + portale klientów, każdy z własnym brandingiem, ekonomią i planem.

2. Stack

Warstwa	Technologia
Framework	Next.js 16 (App Router, RSC)
UI	React 19 · Tailwind 4 · lucide-react
Język	TypeScript (strict)
ORM / DB	Prisma 7 + @prisma/adapter-pg · PostgreSQL (Supabase)
Auth	Auth.js v5 (next-auth@5) — sesje w bazie, tenant-aware adapter
i18n	next-intl (14 języków, deepMerge nad EN)
Płatności	Stripe (multi-currency, dry-wired)
Testy	Vitest (jednostkowe) · Playwright (E2E)
Hosting	Vercel (auto-deploy z gałęzi `main`)

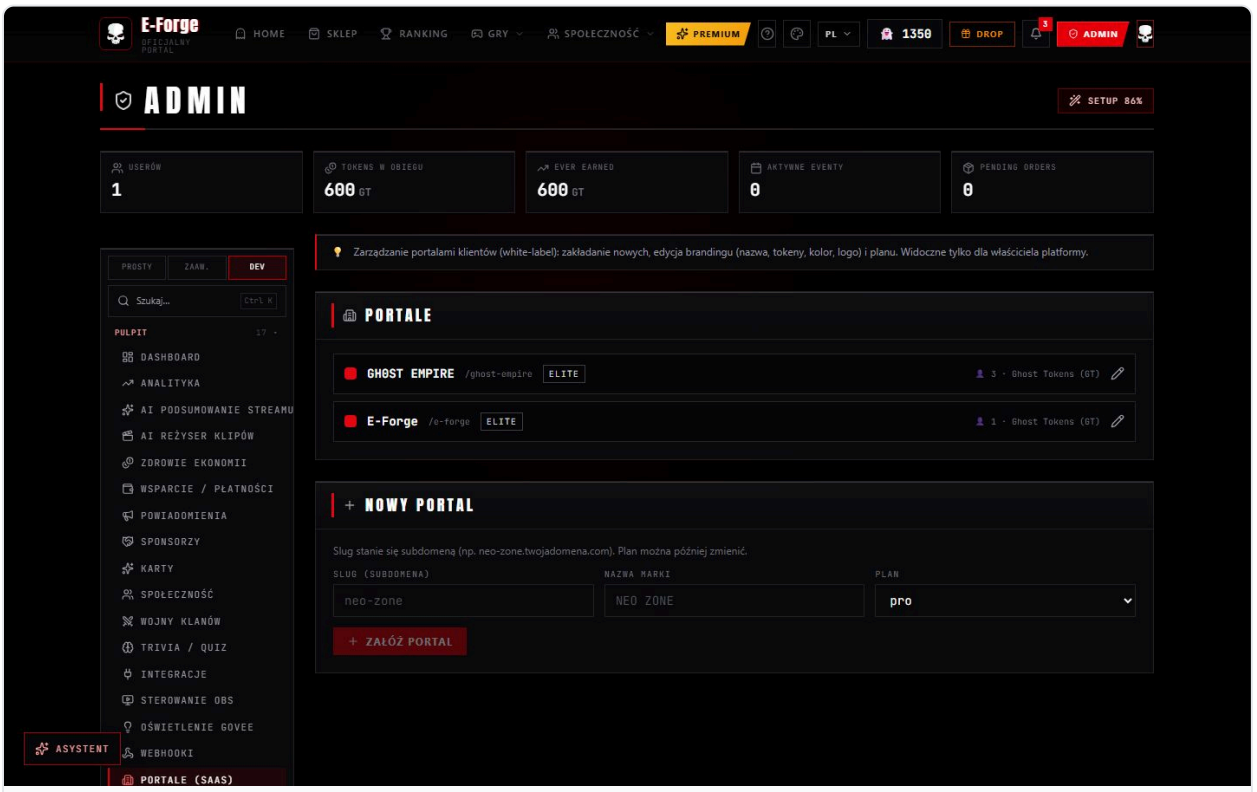
3. Layout repozytorium

```
ghost-empire-phase1/
├─ ghost-empire-web/        # aplikacja Next.js (99% pracy)
│  ├─ src/app/              # trasy: [locale]/ (UI), api/ (~200 endpointów), overlay/ (OBS)
│  ├─ src/components/       # UI + components/admin (~52 sekcje)
│  ├─ src/lib/              # auth, tenant, billing, premium, economy, alerts…
│  ├─ src/messages/         # i18n JSON (14 języków)
│  ├─ prisma/schema.prisma  # ~100 modeli (wszystko per-tenant)
│  └─ e2e/                  # testy Playwright
├─ docs/                    # ARCHITECTURE, ENDPOINTS, ENV, RLS…
├─ CHANGELOG.md ROADMAP.md CLAUDE.md
└─ ghost-empire-chat/       # bot Twitch/Kick/YouTube (osobny runtime)
```

4. Architektura multi-tenant

Tenant rozwiązywany po **hoście** (`lib/tenant.ts`): subdomena `<slug>.root` → po slug; custom domain → po `Tenant.domain` ; inaczej tenant domyślny (założyciel). Nagłówek `x-tenant-slug` jest forge-owalny i nigdy nie zaufany — proxy przelicza go sam.

Konwencja danych: prawie każdy model ma nullable `tenantId` ; odczyty/zapisy przez `...(tid ? { tenantId: tid } : {})` ; per-tenant composite unique (nigdy globalny unique na `code` / `name`). `currentTenantId()` jest cache'owane per request.



Panel „Portale (SaaS)" — zarządzanie portalami klientów

5. Model danych (kluczowe modele)

- **Tenant** — tożsamość portalu (slug, domain, plan, planExpiresAt, branding, Stripe IDs).
- **User / Account / Session** — tożsamość per portal (email @@unique([email, tenantId]); sesje w bazie).
- **Transaction** — księga GT (earn/spend/transfer); **User.tokens/level/xp/prestige**.
- **Shop/ShopItem/ShopOrder**, **Collectible/CardListing** (P2P), **Donation**.
- **Prediction/Poll/WheelSpin/GtGamePlay/Duel/Heist** — gry i zakłady.
- **Event/Bounty/Clan/ClanWar/Achievement/DailyTask/BattlePass** — zaangażowanie.
- **AlertTypeConfig/OverlayScene/CustomWidget/ObsRule/StreamGoal/Subathon** — overlaye.
- **IntegrationConfig** + tokeny platform — sekrety szyfrowane AES-256-GCM at rest.

6. Autoryzacja

Auth.js v5, `session.strategy = "database"`, `tenantAwarePrismaAdapter` (ta sama tożsamość OAuth → osobny User per portal). Providerzy: Twitch, Discord, Google (z YouTube scope), Kick (custom). Stałe adminy po e-mailu (`isPermanentAdminEmail`) są adminami wszędzie i przeżywają reset bazy.

Gatey (`lib/admin.ts`): `requireAdmin()`, `requirePermission(perm)`, `requirePlatformOwner()`. Feature-gate planów (`lib/entitlements.ts`): basic ⊂ pro ⊂ elite; wygasły plan → degradacja do basic; awaria DB → fail-open.

7. API i konwencje

Ścieżka	Auth	Cel
`/api/admin/*`	requireAdmin	konfiguracja panelu
`/api/internal/*`	Bearer BOT_SECRET	bot → portal (nagrody)
`/api/alerts/*`	?token=OVERLAY_TOKEN	feedy overlayu (SSE/polling)
`/api/webhooks/*`	podpis	Stripe/Twitch/Kick/YouTube
`/api/gt-games/*`	sesja + feature-gate	kasyno
`/api/cron/*`	Bearer CRON_SECRET	zadania Vercel Cron

Klient: `apiGet/apiPost` (lib/api-client) rzuca `ApiError` z komunikatem serwera. Rate-limit: sliding window (Redis → fallback pamięć).

8. Bramki weryfikacji (wszystkie muszą być zielone)

```
# z katalogu ghost-empire-web/
npx tsc --noEmit        # typy (strict)
npx vitest run          # testy jednostkowe
npx eslint <pliki>      # lint (flat config + React Compiler)
npx next build          # build produkcyjny
npm run docs:check      # spójność dokumentacji (CHANGELOG #NNN)
```

Reguła docs-sync (CLAUDE.md): każdy PR ze zmianą zachowania aktualizuje `CHANGELOG.md` (`[Unreleased]`, newest-first, ref `(#NNN)`) oraz odpowiednie `docs/`. CI blokuje merge, jeśli `docs:check` czerwone.

9. Zmienne środowiskowe (skrót)

Grupa	Zmienne
Rdzeń	`NEXTAUTH_SECRET` · `DATABASE_URL` (pooler:6543) · `DIRECT_URL` (5432) · `BOT_SECRET`
OAuth	`TWITCH_*` · `DISCORD_*` · `GOOGLE_*` · `KICK_*`
Sekrety at-rest	`ENCRYPTION_KEY` (fallback do NEXTAUTH_SECRET)
Billing	`STRIPE_SECRET_KEY` · `STRIPE_WEBHOOK_SECRET` · `STRIPE_PRICE_*`
Multi-tenant	`NEXT_PUBLIC_ROOT_DOMAIN` (subdomeny)
Opcjonalne	AI · VAPID (push) · UPSTASH_REDIS · GOVEE · STEAM · BACKUP_S3

Pełna referencja: `docs/ENV.md`.

10. Uruchomienie lokalne

```
cd ghost-empire-web
cp .env.example .env.local            # ustaw DATABASE_URL/DIRECT_URL/NEXTAUTH_SECRET/BOT_SECRET + 1 
npm install                           # postinstall: prisma generate
npm run db:push                       # schema → baza
npm run db:seed                       # dane startowe (achievementy, questy, sklep)
npm run dev                           # http://localhost:3000
# (opcjonalnie bot)
cd ../ghost-empire-chat && npm install && npm run auth:twitch && npm start
```

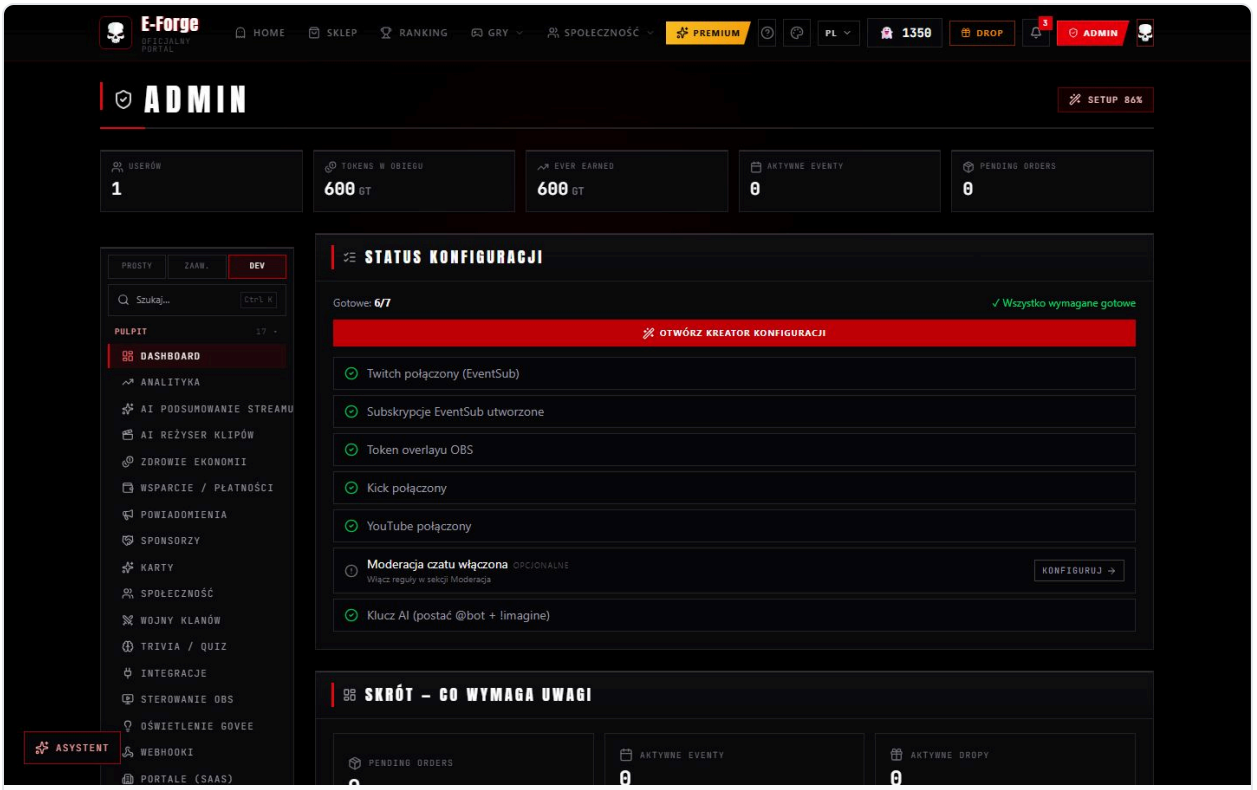
Node ≥ 22. Aby zostać adminem lokalnie: ustaw `User.isAdmin = true` w `npm run db:studio` albo zaloguj się e-mailem z `PERMANENT_ADMIN_EMAILS`.

11. Deployment

Vercel: każdy push na `main` auto-deployuje produkcję; PR-y dostają preview. Env-y ustawiasz w dashboardzie Vercel (lub przez API). Po zmianie env wymagany redeploy. Bot (`ghost-empire-chat`) to osobny runtime (Docker/VPS) — proces per portal przez `ENV_FILE`.

12. Billing (Stripe)

Dry-wired: bez `STRIPE_SECRET_KEY` checkout zwraca 503, trial 14 dni bez karty. `lib/premium.ts` trzyma czyste stałe (waluty PLN/EUR/USD, ceny lustrzane do `currency_options` Stripe). `POST /api/billing/checkout` tworzy sesję Checkout (mode=subscription, currency, trial_period_days=14); `POST /api/webhooks/stripe` aktualizuje `Tenant.plan/planExpiresAt` na zdarzeniach subskrypcji (idempotentnie).



Panel admina — Cinematic Mono, status konfiguracji portalu