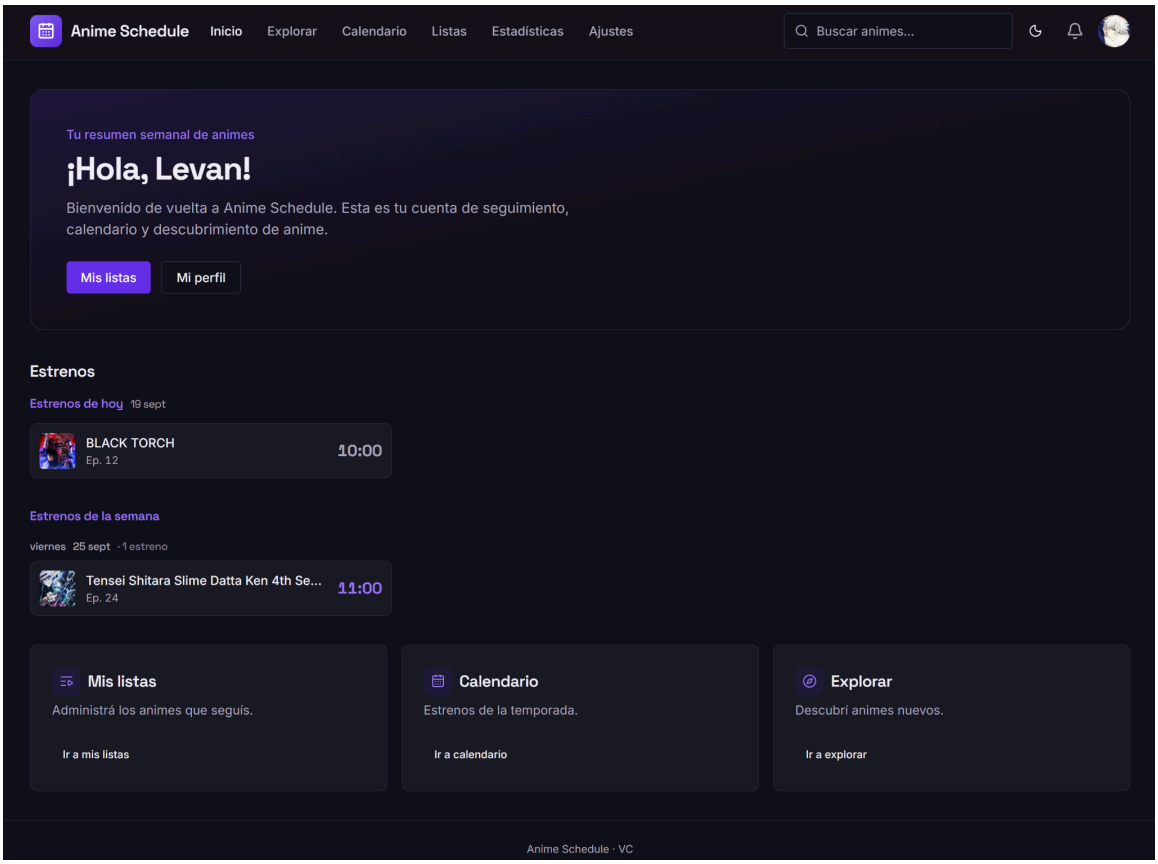
Anime Schedule

Aplicación web **full-stack** (responsive y PWA) en español para **organizar, ver el calendario y descubrir animes**.

Con una sola cuenta podés llevar el control de todo lo que ves: **listas personalizadas** (6 estados), **progreso de episodios**, **favoritos**, **calendario semanal de estrenos**, **estadísticas de consumo** y **notificaciones** (push, email e in-app). Los datos de los animes (títulos, sinopsis, imágenes, staff, personajes…) se obtienen de la **API pública de AniList** y se cachean en Redis.



Captura: Vista general de la app

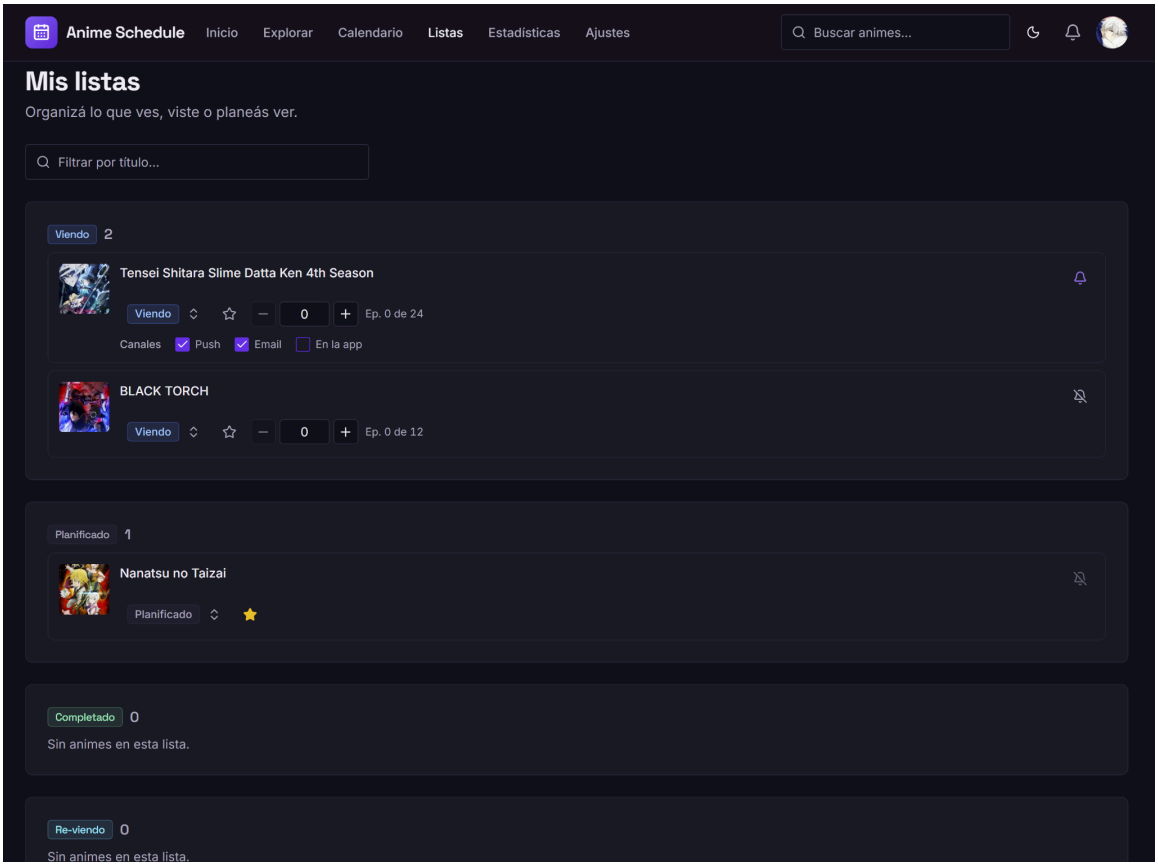
Tabla de contenidos

- Anime Schedule
  - Tabla de contenidos
  - Características
    - Mis listas y progreso
    - Calendario semanal
    - Exploración y búsqueda
    - Detalle de anime
    - Estrenos y estadísticas
    - Notificaciones
    - PWA y accesibilidad
  - Stack tecnológico
  - Arquitectura
  - Estructura del repositorio
  - Requisitos
  - Puesta en marcha (desarrollo local)
  - Scripts útiles
  - Variables de entorno
  - Páginas y funcionalidades
  - Testing y CI/CD
  - Documentación del proyecto

Características

Mis listas y progreso

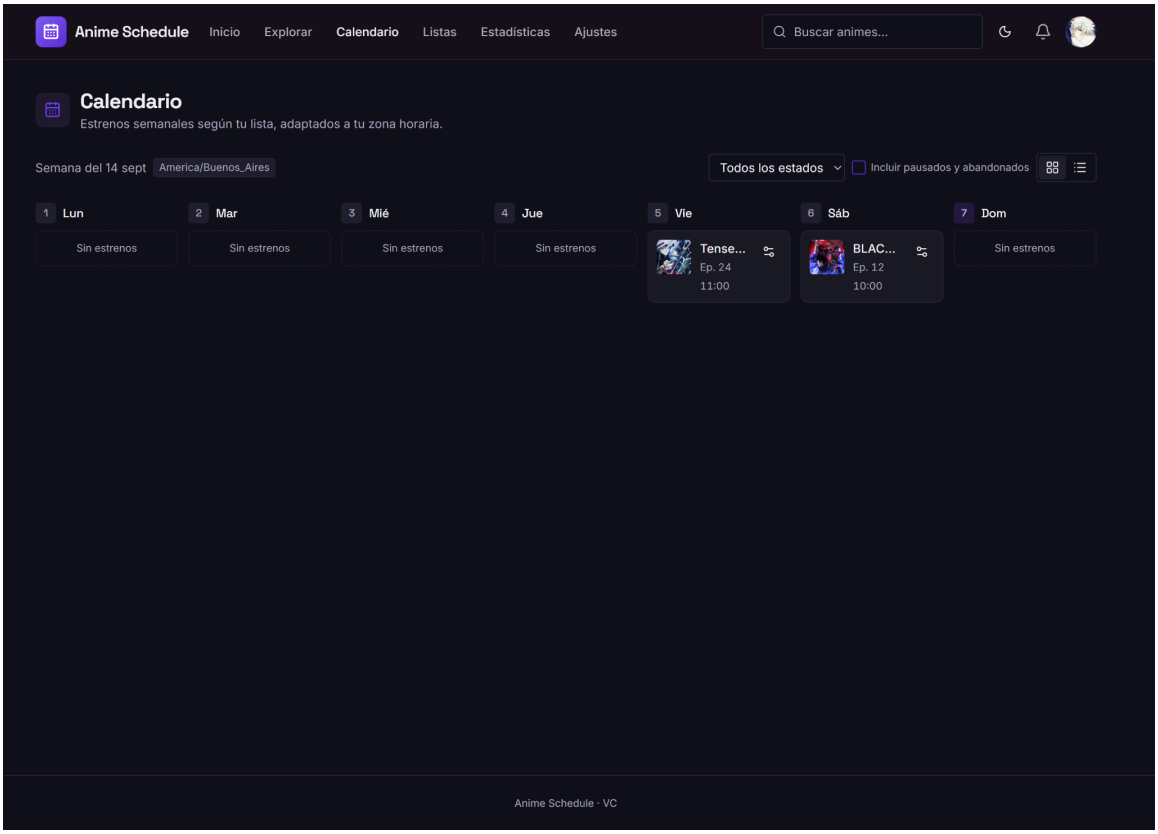
Organizá tus animes en los 6 estados clásicos (**Watching**, **Plan to Watch**, **Completed**, **Rewatching**, **Paused**, **Dropped**), controlá el progreso de episodios con + / − o entrada manual y marcá tus **favoritos**. Incluye un **buscador/filtro por título** que filtra al instante mientras escribís.



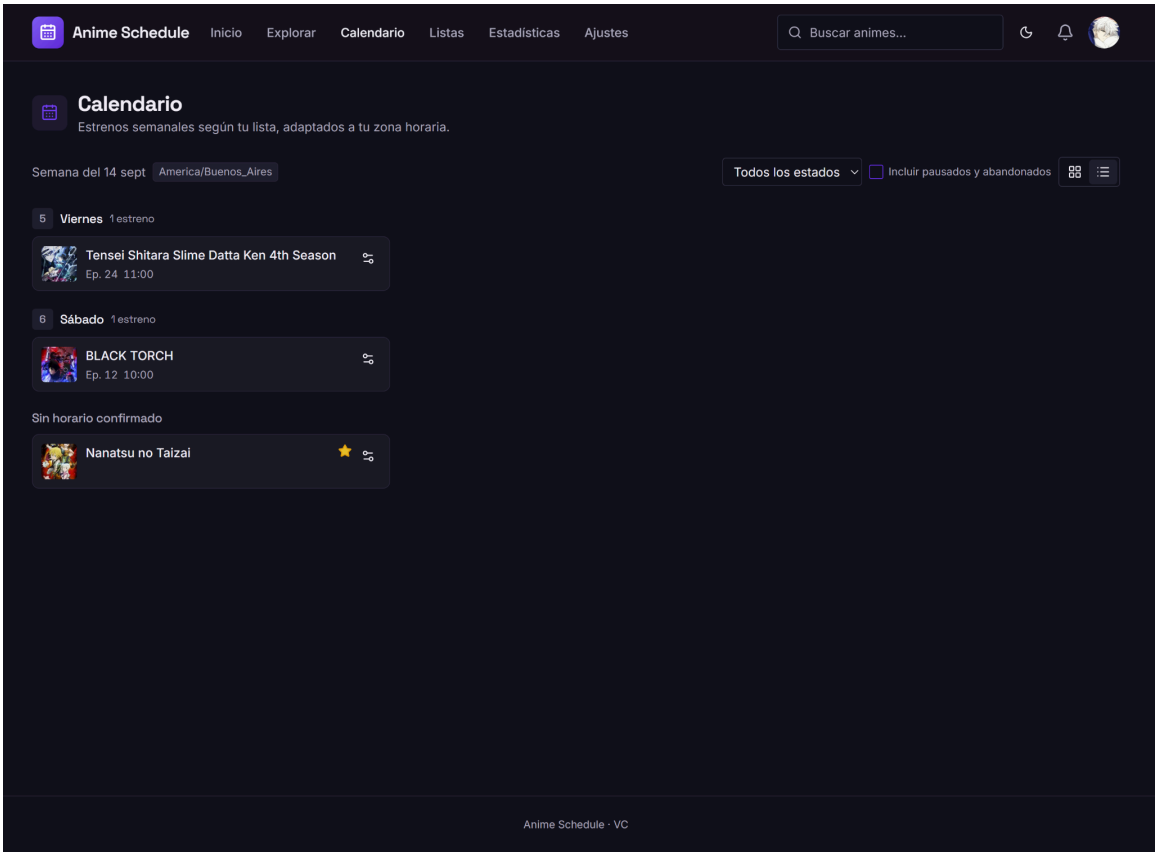
Captura: Mis listas

Calendario semanal

Vista **grid** (lunes a domingo) y **vista lista por día** de las emisiones de tus animes, con orden por hora, favoritos destacados, **override manual del horario** y ajuste de **zona horaria**.



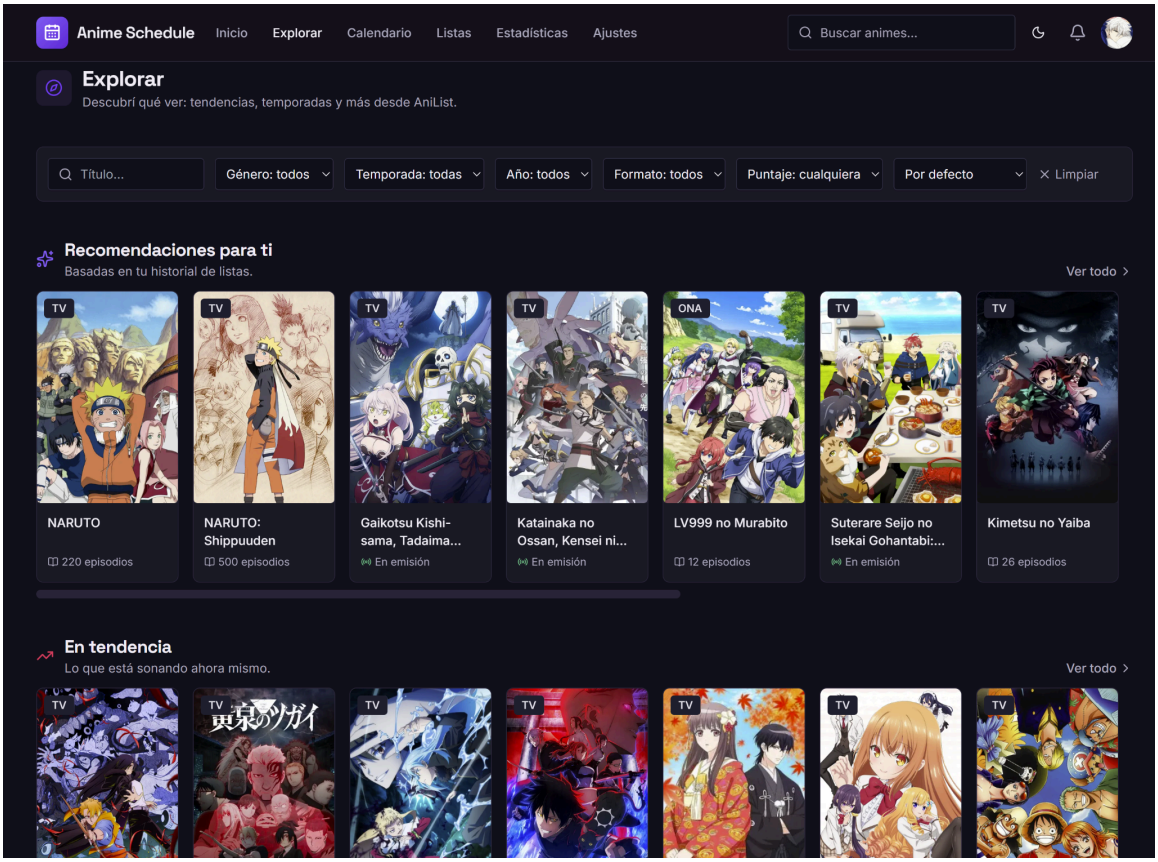
Captura: Calendario semanal



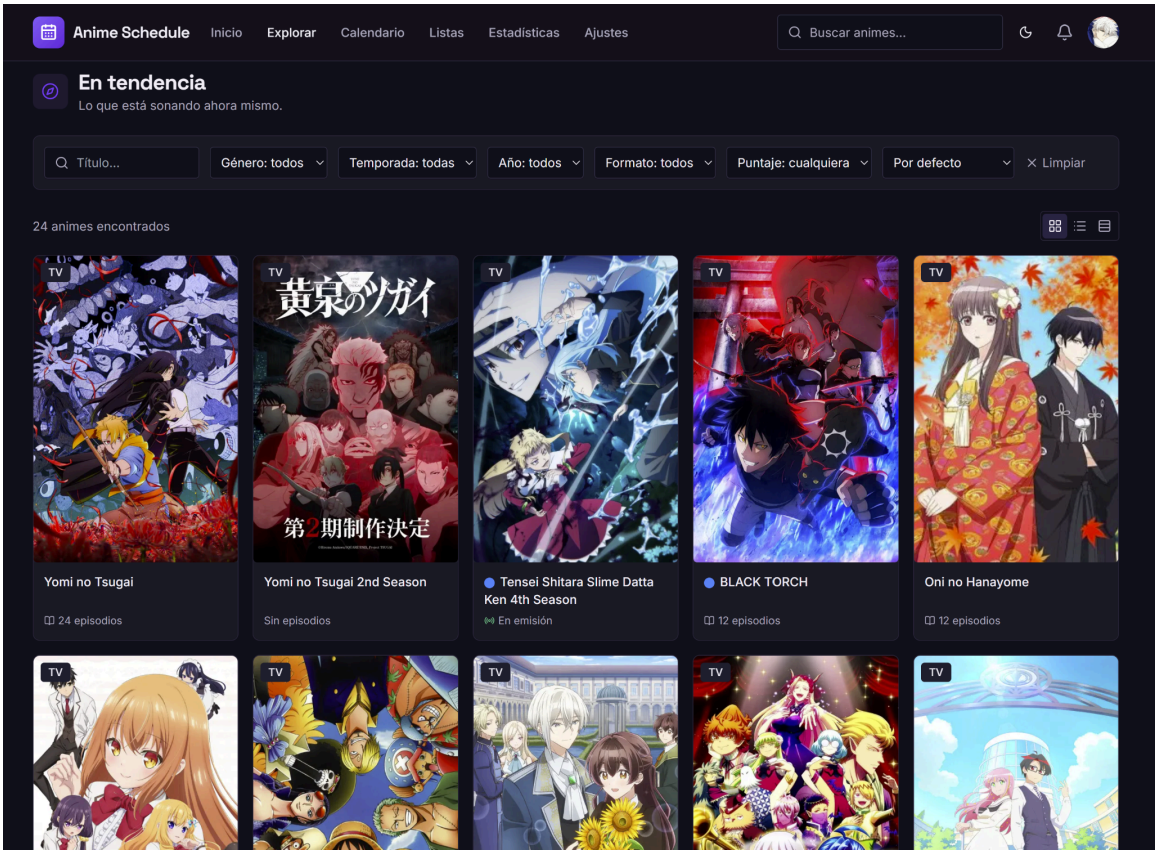
Captura: Calendario semanal segunda vista

Exploración y búsqueda

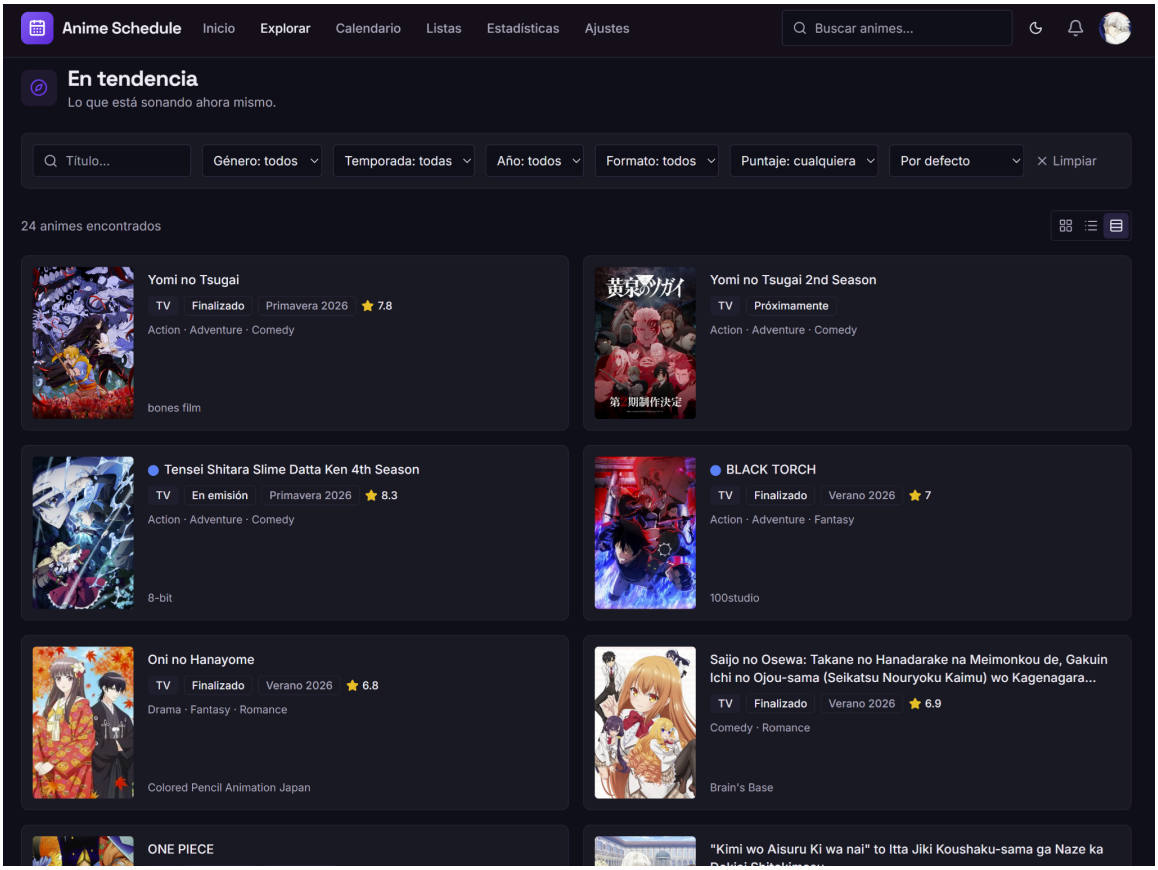
Top de la temporada, **recomendaciones personalizadas** basadas en tu historial y **buscador global con autocompletado** en la navbar, además de búsqueda con filtros (título, género, año, score, formato, estado) en Explorar.



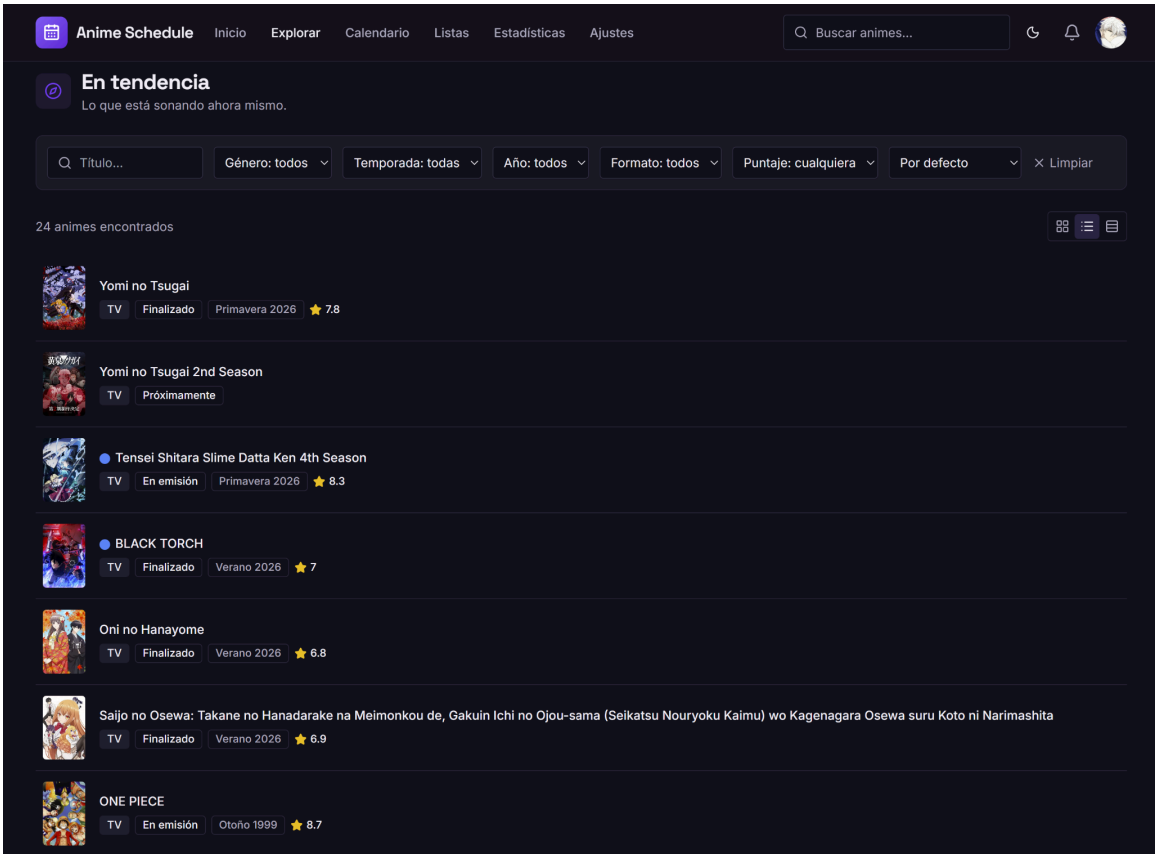
Captura: Explorar



Captura: Explorar tendecia



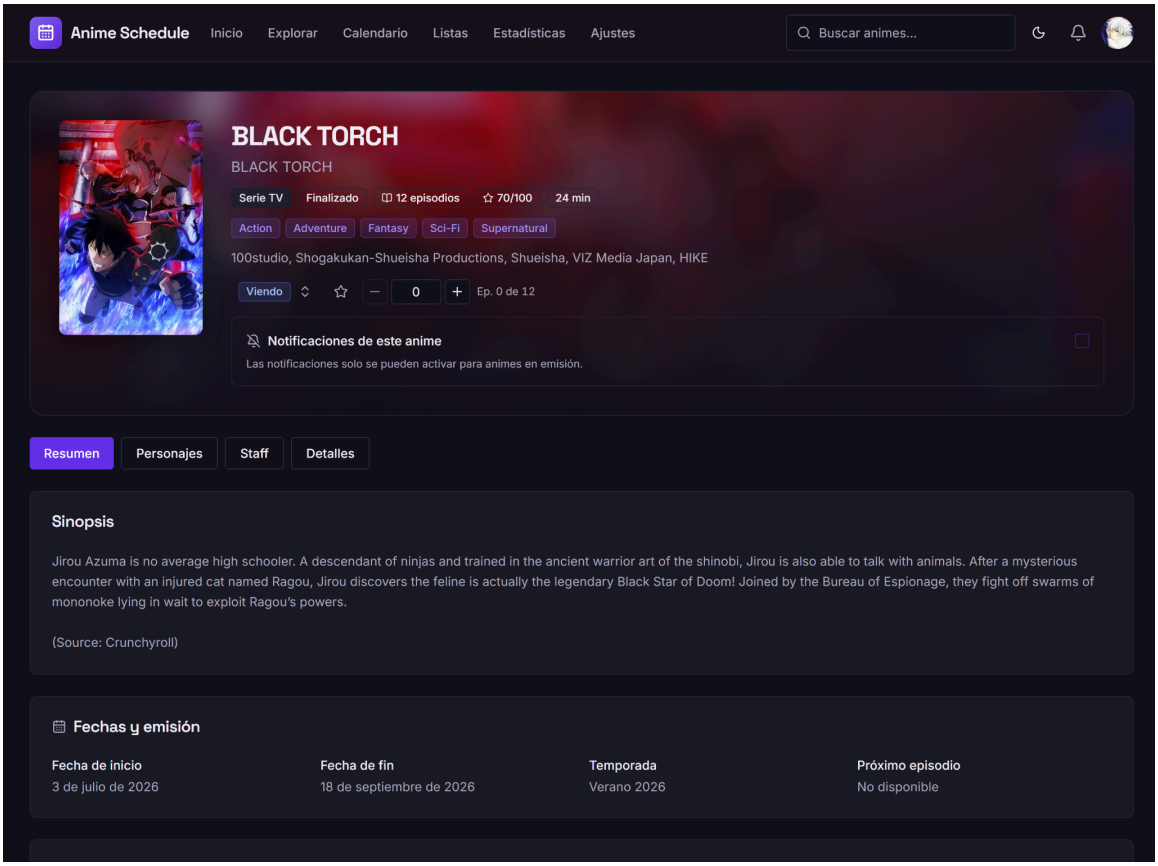
Captura: Explorar vista 2



Caputra: Explorar vista 3

Detalle de anime

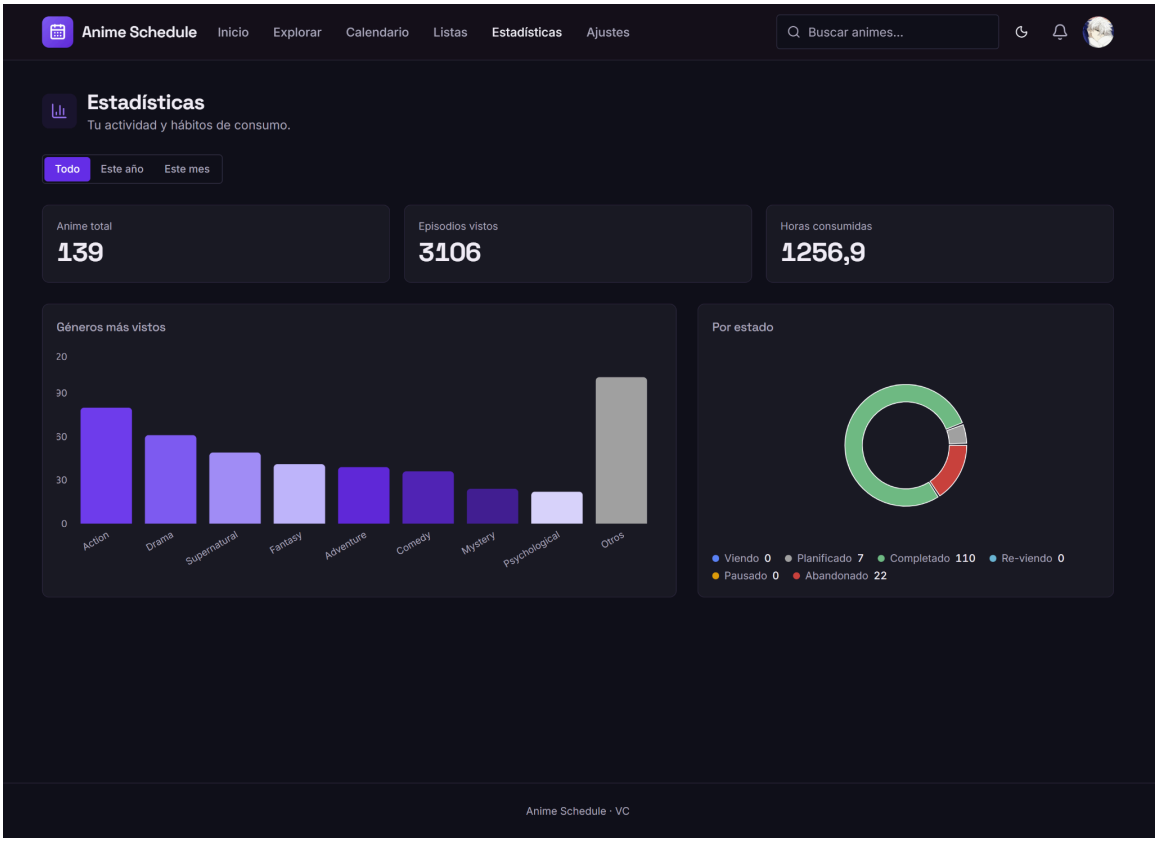
Ficha completa: sinopsis, rating, géneros, estudio, fechas, **staff, seiyuu y personajes**, plataformas de streaming y multimedia — con acciones de lista, favorito y notificaciones.



Captura: Detalle de anime

Estrenos y estadísticas

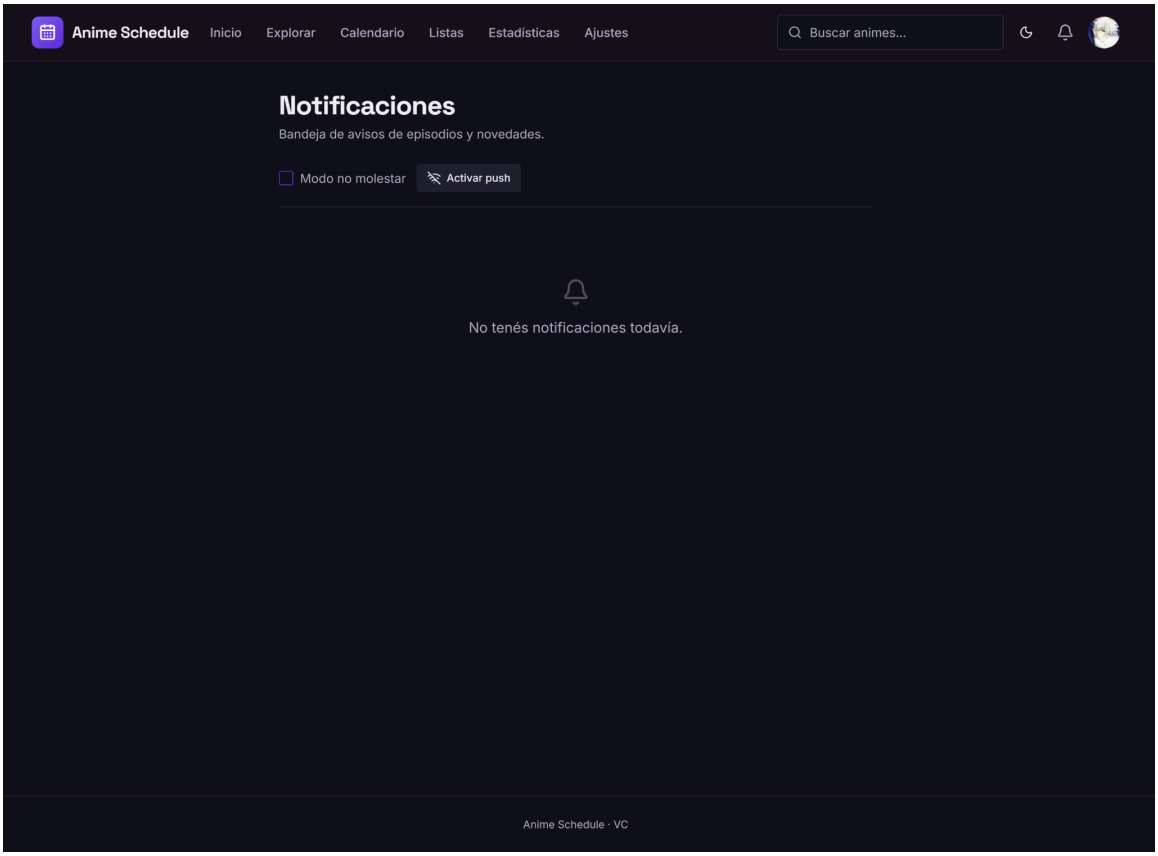
Dashboard de inicio con **"Qué se estrena hoy / esta semana"** y dashboard de **estadísticas** con gráficos de distribución por género y estado, episodios vistos y horas consumidas.



Captura: Estadísticas

Notificaciones

Configuración **por anime** con canales **push** (Web Push/VAPID), **email** (Resend) e **in-app** (centro de notificaciones), prioridad para favoritos y modo **no molestar**.



Captura: Notificaciones

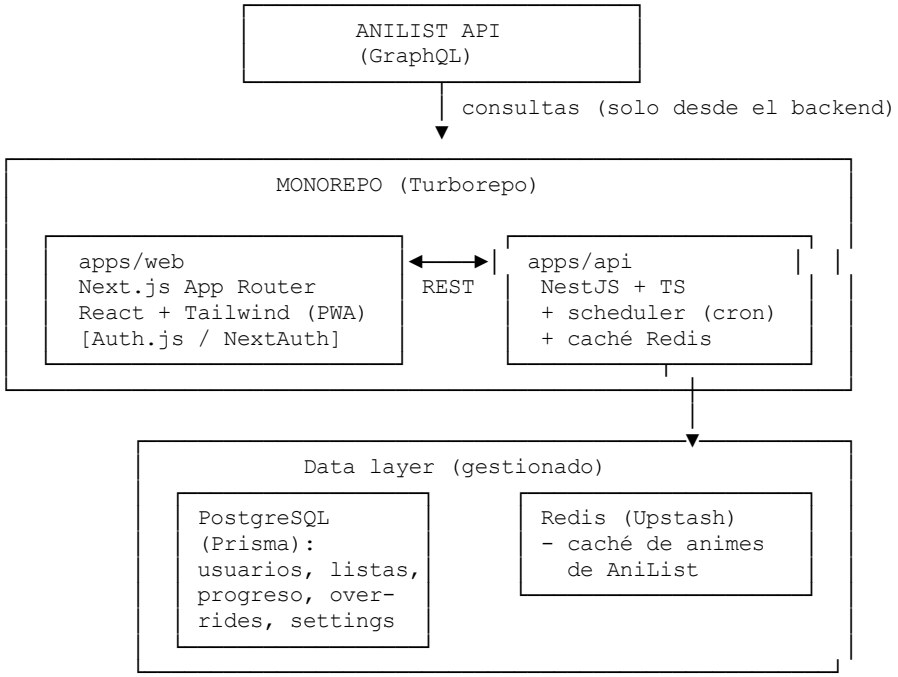
PWA y accesibilidad

Instalable como PWA, con **offline limitado**, tema oscuro/claro/sistema, interfaz **100% en español** y criterios de accesibilidad (WCAG 2.1 AA) aplicados.

Stack tecnológico

Capa	Tecnologías
Frontend	Next.js 15 (App Router) · React 19 · Tailwind CSS · TypeScript · Radix UI · `lucide-react` · `recharts` (estadísticas) · `sonner` (toasts) · `next-themes`
PWA	`@serwist/next` (service worker + manifest), offline limitado, push desde el SW
Backend	NestJS 11 · TypeScript · `@nestjs/schedule` (cron) · `class-validator` · `helmet` · `@nestjs/throttler` (rate limits)
Datos	PostgreSQL · Prisma ORM · Redis (vía `ioredis`) como caché de AniList
Integraciones	AniList GraphQL · MyAnimeList (proveedor alternativo) · Resend (email) · Cloudinary (avatares) · Web Push VAPID
Monorepo	Turborepo · pnpm · ESLint · Prettier · Husky/lint-staged

Arquitectura



Principios clave:

- El **frontend nunca llama a AniList directamente**: todas las consultas pasan por el backend, que centraliza la **caché en Redis** (read-through cache) y el manejo de errores / rate limits.
- Los **datos del usuario** (listas, progreso, favoritos, overrides, settings) viven en **PostgreSQL**, filtrados siempre por `userId` de la sesión (evita IDOR).
- Los datos de animes (cuerpos de AniList) se guardan **solo en caché (Redis)**: no se duplican en la base de datos.
- Los **overrides** del usuario prevalecen sobre los datos de la API al construir el calendario.

Estructura del repositorio

```
anime-schedule/
├── .github/workflows/ci.yml      # CI: typecheck, lint, format, tests, build
├── apps/
│   ├── web/                      # Frontend Next.js (PWA) — puerto 3000
│   │   └── src/app/              # App Router
│   │       ├── (app)/            #   landing /, inicio, listas, calendario,
│   │       │                     #   explorar, estadisticas, notificaciones,
│   │       │                     #   perfil, ajustes, anime/[animeId]
│   │       ├── (auth)/           #   login, registro, recuperación, etc.
│   │       └── ~offline/         #   fallback offline (service worker)
│   └── api/                      # Backend NestJS — puerto 4000
│       └── src/                  # auth, profile, lists, progress, favorites,
│                                 # calendar, explore, search, releases,
│                                 # notifications, stats, export, settings,
│                                 # catalog (AniList/MAL), anilist (caché)
├── packages/
│   ├── config/                   # Config ESLint / TS / Prettier compartida
│   ├── shared/                   # Tipos y DTOs de dominio compartidos
│   └── database/                 # Schema Prisma + cliente + migraciones
├── sdd/                          # Documentos de especificaciones (ver abajo)
├── screenshots/                  # Capturas de pantalla (pendientes)
├── docker-compose.yml            # PostgreSQL (5433) + Redis (6379) locales
└── .env.example                  # Plantilla de variables de entorno
```

Requisitos

- **Node.js ≥ 22**
- **pnpm 12.3.4** (`corepack enable` o `npm i -g pnpm@12.3.4`)
- **Docker** (para PostgreSQL y Redis locales) — o servicios remotos
- **Git**

Puesta en marcha (desarrollo local)

```
# 1. Instalar dependencias
pnpm install

# 2. Levantar PostgreSQL y Redis locales (opcional si usás servicios remotos)
docker compose up -d

# 3. Configurar variables de entorno
cp .env.example .env
#    → Generá un secreto para AUTH_SECRET: openssl rand -base64 32

# 4. Crear/aplicar las migraciones de la base de datos
pnpm --filter @anime-schedule/database migrate

# 5. Levantar web + api con Turborepo
pnpm dev
```

- **Web**: http://localhost:3000
- **API**: http://localhost:4000 (healthcheck en `/health`)

Sin claves de push/email, la app funciona igual: los canales **push** y **email** requieren `VAPID_*` y `RESEND_API_KEY`; podés deshabilitar push con `DISABLE_PUSH=1`. El catálogo de animes usa **AniList** por defecto (`CATALOG_PROVIDER=auto`) y no requiere credenciales para lectura pública.

Scripts útiles

Comando	Descripción
`pnpm install`	Instala dependencias (Husky + `prisma generate` automático)
`pnpm dev`	Levanta web y API en modo watch (Turborepo)
`pnpm build`	Build de producción de todos los paquetes
`pnpm lint`	ESLint en todo el monorepo
`pnpm typecheck`	TypeScript `--noEmit` en todo el monorepo
`pnpm test`	Tests (Jest) — la API tiene la suite activa
`pnpm format` / `pnpm format:check`	Prettier: formatear / verificar
`pnpm --filter @anime-schedule/database migrate`	`prisma migrate dev`
`pnpm --filter @anime-schedule/database db:deploy`	`prisma migrate deploy` (prod)
`pnpm --filter @anime-schedule/database studio`	Prisma Studio
`pnpm --filter @anime-schedule/api test`	Tests del backend

Variables de entorno

Todas las variables están documentadas en [`.env.example`](.env.example). Las principales:

Variable	Uso
`DATABASE_URL`	Conexión PostgreSQL (local: `localhost:5433`)
`REDIS_URL`	Conexión Redis (local: `localhost:6379`)
`AUTH_URL` / `AUTH_SECRET`	Autenticación Auth.js (v5)
`API_URL`	URL del backend desde el frontend (`http://localhost:4000`)
`CORS_ORIGINS`	Orígenes permitidos para la API
`ANILIST_API_URL`	Endpoint GraphQL de AniList (default `https://graphql.anilist.co`)
`CATALOG_PROVIDER`	`auto` \
`VAPID_PUBLIC_KEY` / `VAPID_PRIVATE_KEY` / `VAPID_SUBJECT`	Web Push
`DISABLE_PUSH`	`1` deshabilita el push (dev/tests)
`RESEND_API_KEY` / `RESEND_FROM`	Email transaccional
`CLOUDINARY_*`	Subida de avatares

Páginas y funcionalidades

Vista	Ruta	Qué incluye
Landing	`/`	Presentación de la app y acceso a registro/login
Inicio	`/inicio`	Saludo, estrenos de hoy / esta semana y accesos rápidos
Mis listas	`/listas`	6 listas por estado, progreso, favoritos, notificaciones por anime, buscador/filtro por título
Calendario	`/calendario`	Semana grid/lista, overrides de horario, vista preferida
Explorar	`/explorar`	Top de temporada, recomendaciones, búsqueda con filtros
Detalle de anime	`/anime/:animeId`	Info completa, staff/seiyuu/personajes, streaming, acciones
Estadísticas	`/estadisticas`	Gráficos por género y estado, episodios y horas consumidas
Notificaciones	`/notificaciones`	Centro in-app y configuración por anime
Perfil	`/perfil`	Avatar y datos personales con resumen de stats
Ajustes	`/ajustes`	Tema, zona horaria, no molestar, preferencias
Login / Registro	`/login`, `/registro`	Acceso y creación de cuenta
Recuperación	`/olvide-password`, `/reset-password`	Restablecer contraseña

La landing, Explorar y el detalle de anime son públicas; el resto requiere sesión.

Testing y CI/CD

- **Unit/integración**: suite **Jest** en `apps/api` (listas, progreso, favoritos, calendario, recomendaciones, catálogo AniList/MAL, notificaciones, stats, auth, etc.).
- **CI (GitHub Actions, `.github/workflows/ci.yml`)**: sobre cada push a `main` y cada PR ejecuta install → typecheck → lint → format check → tests de la API → build con Turborepo.
- **Calidad local**: Husky + lint-staged formatean y lintan en el pre-commit.