Syntropy

Professional web infrastructure built as an integral system

Idunn Freya

October 2024

STATUS COMPLETED **STARTED** October 2024 **VERSION** v1.0.0

REPO Private · available on request **URL** idunnfreya.com

SUMMARY

Syntropy is a professional web infrastructure designed and deployed end-to-end — backend, frontend, DevOps, security and SEO — as a cohesive system where every technical decision serves a real production criterion. Built from scratch because no template solves the problem of presenting work that crosses disciplines without flattening its complexity.

Professional portfolio | Web App | Django · PostgreSQL · Docker

TECHNOLOGY STACK

BACKEND	Django 4.2 LTS · Python 3.x
FRONTEND	Tailwind CSS · vanilla JavaScript
DATABASE	PostgreSQL 15
INFRASTRUCTURE	Hetzner VPS · GitHub Actions CI/CD
CONTAINERISATION	Docker · Docker Compose multi-environment
TESTING	pytest · 104+ tests

GOALS

- Modular Django architecture with seven independent applications
- Full bilingual system (ES/EN) with django-modeltranslation

- Automated CI/CD pipeline with production deployment
- Bilingual SEO: hreflang, Open Graph, Twitter Card, sitemap
- Production security: iptables, rate limiting, honeypot, anti-spam
- Automated daily backups with 14-day retention
- Custom analytics with no third-party dependencies
- Per-project visibility system with lead generation
- Full content population across all sections
- Image processing pipeline with brand colour coherence

ARCHITECTURE

A multidisciplinary portfolio has a structural problem: its content is not homogeneous. Code, CAD blueprints, audiovisual pieces and technical documents do not coexist well in templates designed for image galleries or blogs. Syntropy is built from scratch because the problem demands it — not as a technical indulgence.

The name is deliberate. Syntropy is the principle opposite to entropy: order emerging from complexity. That is both the project's thesis and the profile it presents.

MODULE / APP	DESCRIPTION	STATUS
`core`	Shared functionality, base templates, middleware, context processors	✓ Complete
`home`	Landing page with interactive orbital system	✓ Complete
`sections`	Five professional categories with bilingual slugs	✓ Complete
`projects`	Project management with content blocks and multimedia gallery	✓ Complete
`bio`	Professional information with expandable sections	✓ Complete
`contact`	Contact form and private content access request	✓ Complete
`legal`	Legal pages with Markdown rendering	✓ Complete

MILESTONES

Oct 2024

Design and architecture

Stack definition, Django app structure, data models and heterogeneous content system. Multi-environment Docker configuration.

Dec 2024

Functional core

Complete backend with seven apps, bilingual system, customised admin panel and 88+ tests. Development and staging environments operational.

Jan 2025

Production deployment

Deployed to Hetzner VPS. CI/CD pipeline with GitHub Actions, security hardening, automated backups, SSL and transactional email via Brevo.

Feb–Mar 2025

Refinement and SEO

Full responsive design, bilingual SEO (hreflang, OG, sitemap), custom analytics with admin dashboard, favicon, image pipeline and 104+ tests.

2025–present

Content population

Introducing projects, biography, legal texts and multimedia resources across the five portfolio sections.

TECHNICAL DECISIONS AND NOTES

DECISION

Django over SPA frameworks. The content is fundamentally static and multilingual. A framework like React or Vue would add complexity without real benefit: worse SEO, longer load times and an unnecessary build dependency for a site that requires no real-time interactivity.

DECISION

Tailwind CSS + vanilla JavaScript on the frontend. Utility-first allows rapid iteration without accumulating dead CSS. Vanilla JavaScript because interactions are punctual and do not justify the weight of a framework.

DECISION

Custom analytics instead of Google Analytics. Full control over visit data, no third-party cookies, no external dependency. Implemented as Django middleware with a dashboard in the admin panel.

NOTE

Docker Compose with per-environment overlays (dev, staging, production) allows exact replication of the production environment locally. Environment parity eliminates the entire category of “works on my machine” bugs.

WARNING

UFW and Docker have known conflicts with iptables. The solution is to manage rules directly in the DOCKER-USER chain, not through UFW.

INSTALLATION AND DEPLOYMENT

Private repository. To access the source code and detailed technical documentation, use the information request form at idunnfreya.com.

```
        # Development environment (requires Docker)
        docker compose -f docker-compose.yml -f docker-
compose.dev.yml --env-file .env.dev up -d --build

        # Production deployment (from local)
        ./scripts/deploy.sh

        # Tests
        docker compose exec web pytest --cov
```