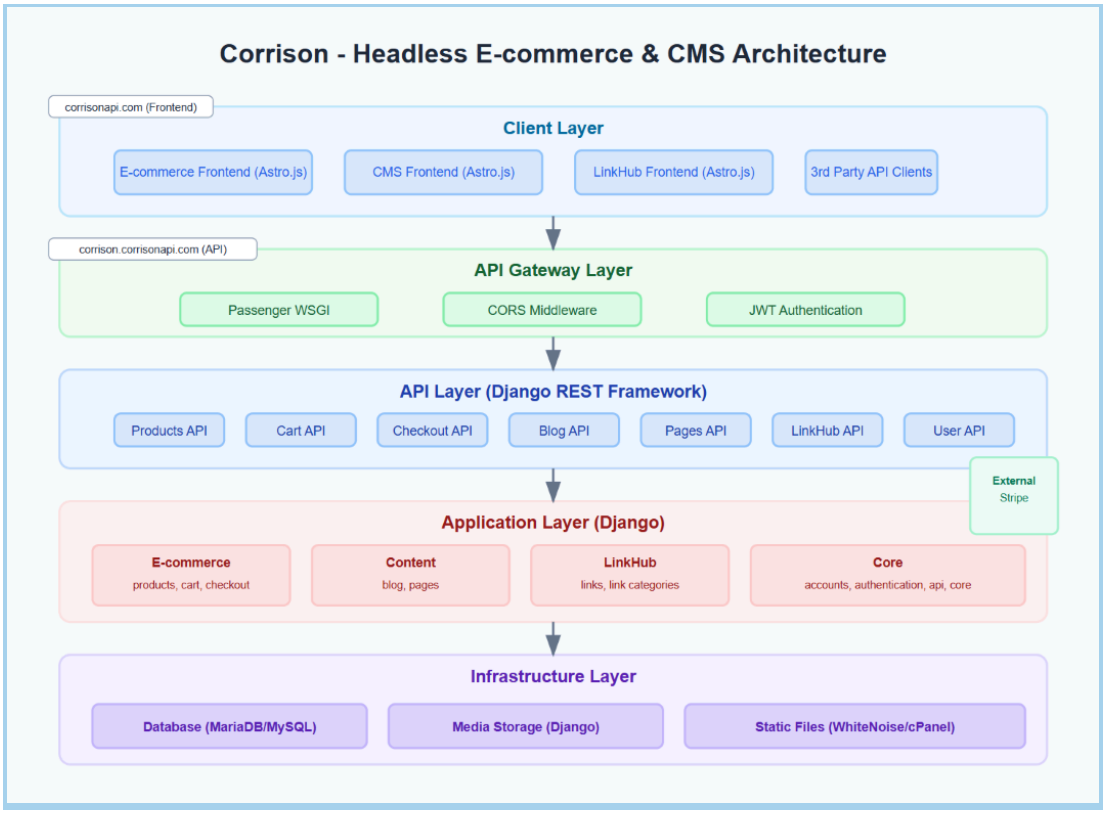
Introducing Corrison

The Multi-Functional Framework for Deployment Across Multiple Domains

By Diane Corriette May 19, 2025



The Corrison Platform

Building a multi-functional platform that can be deployed across multiple domains while maintaining code consistency is a challenge I set for myself and achieved.

I am excited to introduce Corrison, a Django-based platform designed specifically with reusability and tailoring in mind.

Corrison is designed to be fully headless and modular.

In my experience developing eCommerce solutions and websites, I repeatedly encountered the same challenge: clients wanted similar functionality but with unique branding.

The traditional approach of creating separate codebases for each project was time consuming and I went in search of a faster, more sustainable way.

This led to the development of Corrison - a platform that provides enterprise-level functionality while remaining flexible enough to adapt to different business needs.

Key Design Principles

Service-Oriented Architecture

At its core, Corrison separates business logic from presentation through a comprehensive service layer. This means:

Cleaner Code: Views remain thin and focused on presentation

Reusability: Business logic can be shared across different views and contexts

Testability: Services can be tested independently of the web framework

Maintainability: Changes to business rules don't require modifications to views

Modular Architecture

Corrison has been arranged into focused Django apps, each responsible for specific functionality:

corrison/

├── accounts/ # User management

├── cart/ # Shopping cart functionality

├── checkout/ # Order processing and payments

├── products/ # Product catalog

├── blog/ # Content management

├── Pages/ # Build websites and landing pages

├── LinkHub/ # Display links to your content around the web

├── Events/ # Coming soon - set up and manage events

├── Calendar/ # Coming soon - let your clients book appointments

There are plans to include a number of different apps. An events app for setting up and managing events. A calendar that can be used to book clients. Add it to your website with your available dates and clients can book and pay online.

One Platform - Multiple Uses

Multi-Domain Deployment

Corrison uses a sophisticated settings structure that makes deployment across different environments seamless:
Base Settings: Shared configuration across all environments
Environment-Specific: Development, staging, and production settings
Site Settings: Domain-specific configuration for multi-site deployment.

Business Website

Your website at potteryplace.com. Need a different UK and USA site? Create your site once, set it up on different domains. Need a promo page - promo.potteryplace.com - gives you one.

Need a documentation site docs.potterplace.com or a place to sell your pottery. Set up the ecommerce store.

Same Code Different Domains

Deploy the same codebase across different brand domains with unique:

- Visual styling and branding
- Product catalogs
- Pricing strategies
- Content and messaging

International Markets

Easily adapt the platform for different regions:

- Currency and language support
- Region-specific product availability
- Localised payment methods
- Compliance with local regulations

Headless API

Corrison has been created using a headless, API approach. One site. Many uses. It uses a Django-based API that serves content (blog posts, products, pages) to any front-end. Right now, I am using the Astro framework as the frontend.

It includes a CMS backend built with Django and Django REST Framework, powering multiple websites with a shared API infrastructure.

Single API Endpoint

Corrison uses a Single API endpoint. I deploy the Django app once (e.g. at corrison.yourdomain.com), and it exposes REST or GraphQL endpoints like:

GET /posts/
GET /products/
GET /pages/

The front-end then fetches the data.
Any front-end, whether it's Astro, React, Vue, Gatsby, Next.js, a mobile app, or even another Django instance. Your front-end framework simply calls those endpoints, receives JSON, and renders it however it likes.

CORS & Authentication

With a plug-and-play system like this one does that mean anyone can set up a site using my domain name?

NO! *And I'll tell you for why...*

The site has the ability to enable CORS on the API so that requests from domain-a.com, domain-b.com, etc. are allowed.

API keys or OAuth tokens are issued so that each front-end or client registers and authenticates when it asks for data.

So if a domain is not registered with your site it can't get access to your information.

Astro Front-End

By pairing Corrison's headless Django API with Astro's ultra-fast SSG and islands architecture, we get:

- **Data-first builds**
  – We use Astro's `getStaticPaths`/`getStaticProps` hooks to pull from Corrison's REST or GraphQL endpoints at build time. Every page—blog post, product listing, landing page—is generated as a lean, pre-rendered HTML file, with data baked in for instant load times.

- **Islands for interactivity**
  – Minimal client JavaScript is shipped by default; when you need a dynamic component (cart widget, live stock ticker, booking calendar), Astro lets you "island-hydrate" only that piece with React/Vue/Svelte, keeping the rest of the page static.

- **Plugin ecosystem**
  – We leverage Astro plugins for enhanced images, Tailwind CSS integration, and automatic sitemap/RSS feed generation from Corrison's content. That means your blog, product catalog, and event listings all flow straight from the API into a fully fledged site.

- **Incremental builds & preview deploys**
  – Astro's incremental-build support means only changed pages are rebuilt when you push updates to your Corrison backend. Your latest content goes live in seconds, and preview URLs let you take a look at feature updates before they hit production.

- **SEO & performance by default**
  – Pre-rendered output, built-in meta-tag controls, and CDN-friendly asset bundling give you top Lighthouse scores and search-engine visibility out of the box. Meanwhile, Corrison's canonical headers and cross-site linking dovetail perfectly with Astro's auto generated sitemap, keeping all your multi-domain sites unified in Google Search Console.

SEO tips to “unify” sites

You might think that having all these different sites will affect your SEO but there is a way to make sure that does not happen.

A central hub (potteryplace.com) links prominently to each sub-domain.

Cross-linking: In the Footer or Navigation bar of each sub-domain I point back to the central hub domain.

Shared branding: Use the same logo, meta tags, CSS so Google (and users) see them as part of one brand.

Sitemap & Search Console:

Create a root sitemap at potteryplace.com/sitemap.xml that references all sub-site sitemaps.

In Google Search Console, add the “Domain Property” (potteryplace.com) to cover all sub-domains automatically.

Use of canonical headers (if any content overlaps) so Google knows which URL you prefer indexed.

INITIAL SITES

Below are the two main sites that make this work. When you visit the main website you can check out any other sites we have included

https://www.corrisonapi.com - Astro Frontend

https://corrison.corrisonapi.com (the Django engine that makes this work)

https://corrisonapi.com/architecture/ The architecture

More Information

For any questions, please contact:

Diane Corriette

Email: djangify@djangify.com

Website: https://www.djangify.com