

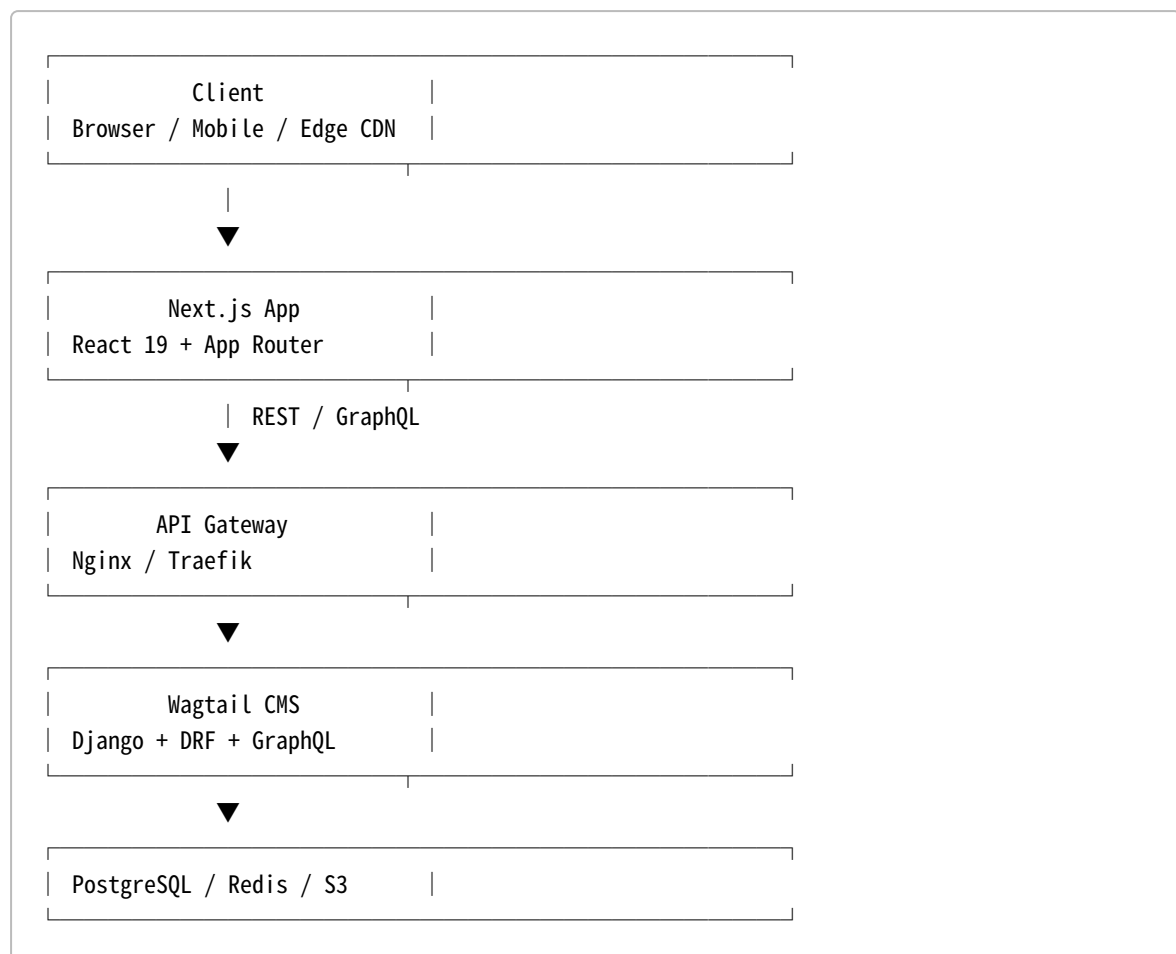
Next.js + Wagtail 企业级 Headless CMS 完整项目代码架构

1. 项目目标

构建一套企业级现代化前后端分离 CMS：

- Wagtail Headless CMS
- Next.js 企业官网前端
- REST + GraphQL API
- Docker/Kubernetes 部署
- 多语言
- SEO
- AI Ready
- 企业级可扩展架构

2. 整体系统架构



3. Monorepo 项目结构

推荐：

```
enterprise-platform/
├── apps/
│   ├── backend/
│   └── frontend/
├── packages/
│   ├── ui/
│   ├── config/
│   ├── types/
│   ├── eslint-config/
│   └── tsconfig/
├── infrastructure/
│   ├── docker/
│   ├── kubernetes/
│   ├── terraform/
│   └── nginx/
├── scripts/
├── docs/
├── .github/
├── package.json
└── turbo.json
```

推荐：

- Turborepo
- pnpm workspace

4. Backend 架构 (Wagtail)

Backend 目录结构

```
backend/
├── apps/
│   ├── core/
│   ├── accounts/
│   ├── pages/
│   ├── blog/
│   ├── products/
│   └── solutions/
```

```
| | | cases/
| | | forms/
| | | media/
| | | seo/
| | | search/
| | | analytics/
| | | workflows/
| | | ai/
| | | api/
|
| | blocks/
| | serializers/
| | graphql/
| | config/
| | requirements/
| | static/
| | media/
| | templates/
| | manage.py
| | pyproject.toml
```

5. Wagtail Content Modeling

推荐 Page Models

```
HomePage
ProductPage
SolutionPage
CaseStudyPage
BlogPage
LandingPage
CareerPage
ContactPage
```

示例 Page

```
from wagtail.models import Page
from wagtail.fields import StreamField
from wagtail.admin.panels import FieldPanel

from blocks.hero import HeroBlock
from blocks.feature import FeatureBlock

class HomePage(Page):
```

```
body = StreamField([
    ('hero', HeroBlock()),
    ('feature', FeatureBlock()),
], use_json_field=True)

content_panels = Page.content_panels + [
    FieldPanel('body')
]
```

6. StreamField Block 架构

Block 目录结构

```
blocks/
├── hero/
├── feature/
├── pricing/
├── faq/
├── stats/
├── cta/
├── video/
├── blog/
├── case-study/
└── shared/
```

HeroBlock 示例

```
from wagtail import blocks

class HeroBlock(blocks.StructBlock):
    title = blocks.CharBlock()
    subtitle = blocks.TextBlock()
    button_text = blocks.CharBlock()
    button_link = blocks.URLBlock()

    class Meta:
        icon = 'image'
```

7. API 架构

REST API

```
/api/v1/pages/  
/api/v1/blogs/  
/api/v1/products/  
/api/v1/solutions/  
/api/v1/cases/  
/api/v1/forms/  
/api/v1/search/
```

API 目录结构

```
api/  
├── v1/  
│   ├── pages/  
│   ├── blogs/  
│   ├── products/  
│   ├── solutions/  
│   ├── auth/  
│   └── search/
```

Serializer 示例

```
from rest_framework import serializers  
  
class BlogSerializer(serializers.Serializer):  
    title = serializers.CharField()  
    slug = serializers.CharField()  
    body = serializers.JSONField()
```

8. GraphQL 架构

推荐技术

Strawberry GraphQL

GraphQL 目录结构

```
graphql/  
├── schema.py  
├── queries/  
├── mutations/  
├── types/  
└── resolvers/
```

Query 示例

```
import strawberry  
  
@strawberry.type  
class Query:  
  
    @strawberry.field  
    def blogs(self) -> list:  
        return BlogPage.objects.live()
```

9. Frontend 架构 (Next.js)

Frontend 目录结构

```
frontend/  
├── app/  
├── components/  
├── blocks/  
├── services/  
├── hooks/  
├── stores/  
├── lib/  
├── styles/  
├── types/  
├── providers/  
├── middleware.ts  
└── next.config.ts
```

10. App Router 架构

```
app/
├── (marketing)/
│   ├── page.tsx
│   ├── products/
│   ├── solutions/
│   ├── blog/
│   └── about/
├── api/
├── layout.tsx
├── loading.tsx
├── error.tsx
└── globals.css
```

11. 前端组件架构

推荐结构

```
components/
├── ui/
├── layout/
├── navigation/
├── footer/
├── seo/
├── forms/
├── charts/
└── animations/
```

12. StreamField Renderer

Block Renderer

```
blocks/
├── hero/
│   ├── Hero.tsx
│   └── index.ts
├── feature/
└── faq/
```

Renderer 示例

```
import Hero from '@blocks/hero/Hero'
import Feature from '@blocks/feature/Feature'

const components = {
  hero: Hero,
  feature: Feature,
}

export function BlockRenderer({ blocks }) {
  return blocks.map((block) => {
    const Component = components[block.type]

    if (!Component) return null

    return <Component key={block.id} {...block.value} />
  })
}
```

13. 数据请求层

推荐结构

```
services/
├── api-client.ts
├── blog.service.ts
├── product.service.ts
├── page.service.ts
└── search.service.ts
```

API Client

```
export async function fetchAPI(url: string) {
  const res = await fetch(`${process.env.API_URL}${url}`)

  if (!res.ok) {
    throw new Error('API Error')
  }

  return res.json()
}
```

14. 状态管理

推荐方案

Zustand
React Query

Store 示例

```
import { create } from 'zustand'

export const useAppStore = create((set) => ({
  theme: 'light',
  setTheme: (theme) => set({ theme }),
}))
```

15. SEO 架构

SEO 目录

```
seo/
├── metadata.ts
├── schema.ts
├── sitemap.ts
└── robots.ts
```

Metadata 示例

```
export const metadata = {
  title: 'TechFlow AI Platform',
  description: 'Enterprise AI Platform',
}
```

16. 多语言架构

推荐方案

next-intl

目录结构

```
messages/  
├── en.json  
├── zh.json  
└── ja.json
```

17. 认证系统

后台

Wagtail Admin

前端用户

JWT
OAuth2
SSO

推荐:

- Keycloak
 - Auth0
-

18. 搜索系统

推荐架构

```
Wagtail
  ↓
OpenSearch
  ↓
Frontend Search UI
```

搜索目录

```
search/
├── indexing/
├── documents/
├── analyzers/
└── queries/
```

19. AI 模块架构

AI 目录结构

```
ai/
├── services/
├── prompts/
├── embeddings/
├── rag/
└── agents/
```

AI 功能

```
SEO Rewrite
Auto Summary
AI Translation
AI Search
RAG Knowledge Base
```

20. Docker 架构

Docker Compose

```
services:
  frontend:
  backend:
  postgres:
  redis:
  opensearch:
  minio:
  celery:
  nginx:
```

21. Kubernetes 架构

K8S 目录结构

```
kubernetes/
├── frontend/
├── backend/
├── postgres/
├── redis/
├── ingress/
└── monitoring/
```

推荐部署结构

```
Ingress
↓
Frontend Pods
Backend Pods
Worker Pods
```

22. CI/CD 架构

GitHub Actions

```
.github/  
├── workflows/  
│   ├── frontend.yml  
│   ├── backend.yml  
│   └── deploy.yml
```

CI/CD 流程

```
Git Push  
↓  
Lint  
↓  
Test  
↓  
Docker Build  
↓  
Deploy K8S
```

23. 数据库设计

PostgreSQL 模块

```
users  
pages  
blogs  
products  
solutions  
forms  
seo  
analytics
```

推荐扩展

```
pgvector
```

用于：

- AI Search
- RAG
- Embedding

24. Redis 架构

Redis 用途

Cache
Session
Queue
Rate Limit
ISR Revalidate

25. 文件存储架构

推荐方案

MinIO
AWS S3
Cloudflare R2

文件流

Upload
↓
Object Storage
↓
CDN

26. 安全架构

必须实现

HTTPS
CSP
Rate Limit
JWT Rotation
WAF
CSRF
XSS Protection

27. 日志与监控

推荐方案

Sentry
Prometheus
Grafana
OpenTelemetry

监控指标

API Latency
CPU
Memory
Error Rate
Search Performance

28. 企业级扩展能力

插件系统

```
plugins/  
├── ecommerce/  
├── crm/  
└── ai/
```

analytics/
marketing/

扩展方式

Signals
Hooks
Middleware
Webhook

29. 推荐开发规范

Frontend

ESLint
Prettier
Husky
Commitlint

Backend

Black
Ruff
Pytest
Mypy

30. 推荐企业级技术栈

Frontend

Next.js 15
React 19
TailwindCSS
Framer Motion
React Query
Zustand

Backend

Django 5
Wagtail 7
DRF
Strawberry GraphQL
Celery
Redis
PostgreSQL

Infrastructure

Docker
Kubernetes
OpenSearch
MinIO
Cloudflare CDN

31. 最终推荐架构

中小型企业官网

Next.js
+ Wagtail
+ PostgreSQL
+ Redis

企业级官网

Next.js
+ Wagtail
+ OpenSearch
+ Celery
+ S3
+ Kubernetes

SaaS 平台

```
Next.js
+ Wagtail Headless
+ AI Service
+ Event Bus
+ Multi-Tenant
+ Microservices
```

32. 最佳实践建议

推荐开发顺序

Phase 1

- Wagtail CMS
- Next.js 官网
- REST API
- SEO

Phase 2

- GraphQL
- OpenSearch
- 多语言
- AI 能力

Phase 3

- SaaS 化
- 多租户
- 微服务
- AI Agent

33. 总结

这套 Next.js + Wagtail 企业级代码架构具备：

- 真正前后端分离
- 企业级可扩展性
- AI Ready
- 云原生

- SEO 强化
- 高性能
- 多语言
- 多站点
- SaaS 演进能力
- 优秀开发体验

非常适合：

- 软件公司官网
- SaaS 平台
- AI 公司
- 企业门户
- 内容平台
- Headless CMS
- 多站点平台
- 企业数字化平台

后续还可以继续扩展：

1. Next.js 完整源码模板
2. Wagtail CMS 完整源码模板
3. StreamField 动态渲染引擎
4. 企业官网 Design System
5. SaaS 登录中心
6. AI 搜索系统
7. RAG 企业知识库
8. Kubernetes Helm Charts
9. Terraform 云部署模板
10. OpenAPI 规范
11. GraphQL Federation
12. 微服务拆分方案
13. 企业权限体系
14. 多租户 SaaS 架构
15. 企业级 DevOps 流水线