DongriGo Architecture Report

Phase 0 Baseline

Version	[e.g., v1.0]
Date	2025-12-25
Author	김동훈 (Donghun Kim)
Repository	https://github.com/lincoln-cmd/DongriGo/tree/main
Production URL	https://dongrigo.onrender.com/
Admin URL	https://dongrigo.onrender.com/admin/
Target	Operation(Render + PostgreSQL + Cloudinary) and Local develop environment

Table of Contents

1. Purpose and Scope

‘DongriGo’ is a content site which provides the history, culture, travel guide, and personal trip log.

The right for writing and editing of contents is only allowed to the administrator and users(visitors) of this site are only permitted to explore by having the right of reading.

This documentation is organized of system architecture and main flow that is standarized now available operating baseline (Phase 0 ~ 1).

2. Tech Stack Summary

- [Backend] Python 3.12, Django 6
- [Frontend] Django Templates + HTMX
- [3D Globe] globe.gl (based on Three.js)
- [Static] WhiteNoise (+ collectstatic in Render)
- [Media] Cloudinary (+ django-cloudinary-storage)
- [DB] in operating: Render PostgreSQL, in local: SQLite
- Deploy: GitHub push ⟶ Render automatically build and deploy, run Gunicorn

3. System Overview (Logical)

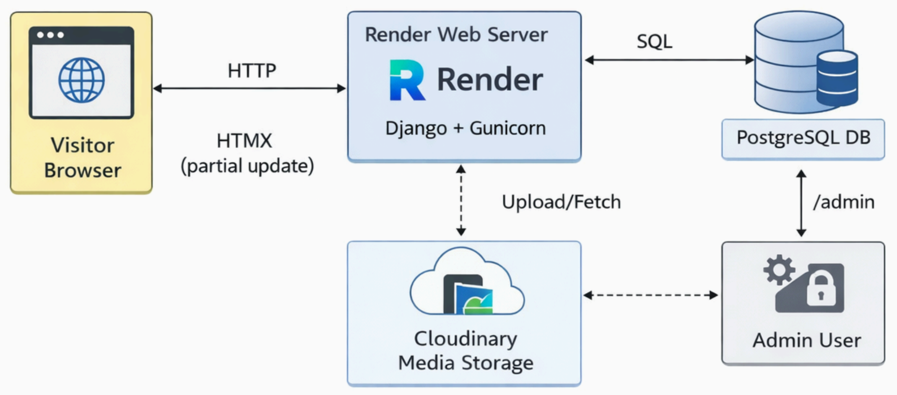

Figure 1. Logical Architecture

Mermaid (optional):

```
flowchart LR
  U[Visitor Browser] → |HTTP| W[Render Web Service \nGunicorn + Django + WhiteNoise]
  A[Admin User] → |/admin| W

  W → |SQL| DB[(Render PostgreSQL)]

  W → |Upload/Fetch| C[Cloudinary Media Storage]

  U → |HTMX request\n(country/category/search)| W

  U → |globe.gl click\nnselect country| U
```

Explanation:

- In the browser, if the user selects the country by globe.gl, the board section requests by HTMX to server in the page, then server returns HTML partial.
- The text and metadata are saved in DB(PostgreSQL/SQLite), and the image file is done in Cloudinary.

4. Data Model (Summary)

4.1 Country

- Identifier: slug
- ISO: iso_a2, iso_a3 (normalization, verification target)
- Display Name: name, name_ko, name_en
- Aliases: alternative designation string (normalization, verification target)
- Flag_image: Cloudinary storage possible

4.2 Post

- FK: country
- Category: Travel, History, Culture, My Log
- Content: Markdown + [[img:ID]] token method
- Cover_image: Representative image (Thumbnail)
- Deploy: is_published, published_at

4.3 PostImage

- FK: post
- Image: cloudinary image file.
- Order: display order (10-unit auto-alignment policy).

4.4 SeedMeta (Idempotent Seeding)

- By recording the fixture hash, make the idempotent the seed_prod.
- If there is an equivalent fixture, skip in restart (hash match).

4.5 PostSlugHistory (Link Preservation)

- Saving the previousURL key when changing the slug, country, category of a post.

5. Key User Flows

5.1 Visitor (Read-only)

1. Home entry → 3D Globe display
2. Counry select → Refresh board area by HTMX
3. Enret post details → Markdown render → Image token replacement
4. Click image → Lightbox zoom/move

5.2 Admin (Content Authoring)

5. '/admin' connect
6. Country/Post/PostImage management (create, edit, delete)
7. After image upload, insert the token([[img:ID]]) in the text
8. Check server render results with Live Preview

6. Content Rendering Rules

6.1 Markdown Rendering

- Post.content is written with Markdown.
- Markdown in the server → After transferring to HTML, sanitization based on allowed tags/attributes (for safety purposed)

6.2 Image Tokens [[img:ID]]

- Utilizing the id of PostImage, insert the image in the text.
- The images which are not employed in the text are printed at buttom gallery automatically.

7. Deployment and Operations

7.1 Render Deployment

- GitHub push → automatical deploy in Render
- Build: './build.sh' (intall requirements + collectstatic)
- Start: 'gunicorn config.wsgi:application –bind 0.0.0.0:$PORT'

Commands (Render Shell):

```
python manage.py migrate
python manage.py ops_check --json
python manage.py seed_prod
python manage.py check_integrity --fix --json
```

8. Operational Risks and Mitigations

- Preventing reseeding accidents: SeedMeta (Hash-based skip).
- Preventing broken links of slug changing: PostSlugHistory + 301 redirect
- Preventing mistakes of admin input: Nomalization of ISO/aliases and warning sign.
- Detecting missing of operating environment settings: ops_check(environment/static/DB/migration/seed status check)

9. Assumptions

- It is possible to utilize PostgreSQL(Render) in an operating environment and SQLite in a local one.
- The image files are stored in Cloudinary, and in DB, is saved the reference value of that.
- The user(visitor) has only read-right and the administrator can write/edit posts.

10. Update Triggers

If below items are altered, update this documentation

- URL structure (country, category, slug routing)
- Model expansion (city, region, tag, series etc.)
- Deploy structure (using other platform, add CDN/caching)
- The authority of contents (introduction of login, comment, community features)