Source: cppcc docs/p2t_Design_Notes.md @ 4e00d40 — curated 2026-08-10; edits land in the source first.

p2t — Design Notes

The reverse-engineered design that the translation was built from

2026-08-09 — frozen; this document records a completed project.

These are the design documents of the p2t experiment, which translated a working Python web service into TypeScript and had the result graded by a test suite written by strangers.

They are reproduced here in full, in reading order, with only their headings re-levelled to fit one document. Nothing has been rewritten.

Why they look the way they do

p2t did not translate Python syntax into TypeScript syntax. It reverse-engineered the upstream Python service back to a design — its entities, relationships, contracts and control flow — and then forward-engineered that design into the target language. These documents *are* that design.

They were written before any of the machinery existed:

```
2026-06-12   the seven realworld-uml documents        (sub-slice .B)
2026-06-12   fragment-mapping.md + the taxonomy       (sub-slice .C)
2026-06-12   the first grammar
2026-06-13   the decomposer
2026-06-13   the emitters
```

They have not been edited since — two commits, and then frozen. That is not neglect. What they describe is an existing service at a pinned commit and a public API specification, both fixed before the project started, so an accurate description of them has no reason to change.

The last section is the one that matters most. Sections 1–7 describe the service; **section 8 is the partition** — every source fragment assigned to one of ten fragment families. That assignment is what turned a coupled, whole-program translation problem into ten local ones, each solved once in a grammar rather than once per fragment.

Two proposed families were dropped during that work, and the reasoning is preserved in section 8 rather than tidied away. There is no `F6` and no `F9` in the shipped grammars; the gap is the visible scar of a decision that was made rather than assumed.

What is not here

`notes/setup.md`, a short local bring-up note, is omitted. It has been superseded by the published build guide and the archive's own `install_postgres.sh` / `install_upstream.sh`.

Companion documents

- **Adaptive Programming — the p2t use case** — what these documents are *for*, and the record of building on them.
- **Cross-Language Translation** — the full account of the experiment.
- **The `p2t` use case** — the pipeline and its ten grammars. These design notes ship inside its archive under `notes/`.

---

1. What this design layer is

Source document: `notes/realworld-uml/README.md` — original title: realworld-uml/ — reverse-engineered design notes

This tree is the **canonical design reference** for the upstream nsidnev / FastAPI implementation of the RealWorld ("Conduit") API spec, at the pin recorded in `source/PINNED_COMMIT.txt`.

Hand-authored during sub-slice .B of the p2t arc (see `steps/step.3000.B.txt`). Consumed by .C (mapping), .D (per-family grammar authoring), .E (decomposer), .F (emitters), .G (TS bring-up), and the later .H + .I sub-slices.

Files

```
+----+---------------------------+---------------------------+
|  # | File                      | What's in it              |
+----+---------------------------+---------------------------+
|  1 | entities.md               | ER diagram + per-entity   |
|    |                           | fields, types, validation,|
|    |                           | DB tables, cross-cutting   |
|    |                           | structural details         |
```

```
+----+----------------------------+---------------------------+
|  2 | routes.md                  | All 19 routes; auth class;|
|    |                            | error envelope; status    |
|    |                            | code histogram            |
+----+----------------------------+---------------------------+
|  3 | contracts.md               | Per-route request /       |
|    |                            | response DTO shapes; full |
|    |                            | Pydantic-to-route map     |
+----+----------------------------+---------------------------+
|  4 | layered-architecture.md    | Per-file -> layer ->      |
|    |                            | provisional fragment-     |
|    |                            | family map; layer summary;|
|    |                            | provisional taxonomy      |
|    |                            | histogram + open notes    |
|    |                            | for .C                    |
+----+----------------------------+---------------------------+
|  5 | auth-vertical.md           | Focused walkthrough of    |
|    |                            | POST /users + POST /users/|
|    |                            | login + GET /user (.E-.G  |
|    |                            | target)                   |
+----+----------------------------+---------------------------+
|  6 | data-flow.md               | Sequence diagrams for the |
|    |                            | three auth flows; failure |
|    |                            | branches called out       |
+----+----------------------------+---------------------------+
```

Read order

For an end-to-end first read, follow the order above. `README.md` is the index; each other file is independently readable.

For the .E-.G focus area, the minimum read is:

1. `entities.md` §"User"
2. `routes.md` (rows for `/api/users`, `/api/users/login`, `/api/user`)
3. `contracts.md` §"Authentication" + §"Current user"
4. `auth-vertical.md` (end-to-end)
5. `data-flow.md` (the three sequence diagrams)

For the .C (mapping) audit:

1. `layered-architecture.md` (the per-file column + "Notes for sub-slice .C")

2. `entities.md` cross-cutting details (F1 has 3 sources)
3. `routes.md` §"Routing composition" (F3 nesting)

Citation convention

Every claim about upstream behavior is cited as `path:line` where `path` is rooted at `source/upstream/`. This makes any factual question grep-friendly. If a future sub-slice corrects a claim here, the citation pin still identifies the source even after the note evolves.

Scope reminder

These notes cover the **entire** upstream API surface (all entities, all routes), not just the .E-.G auth vertical. The reason is that .H + .I + .J all consume these notes; authoring them piecemeal would mean re-walking the upstream multiple times.

The fragment-family column in `layered-architecture.md` is **provisional** (locked as such for this slice). .C is where the taxonomy gets validated and frozen.

Cross-references out

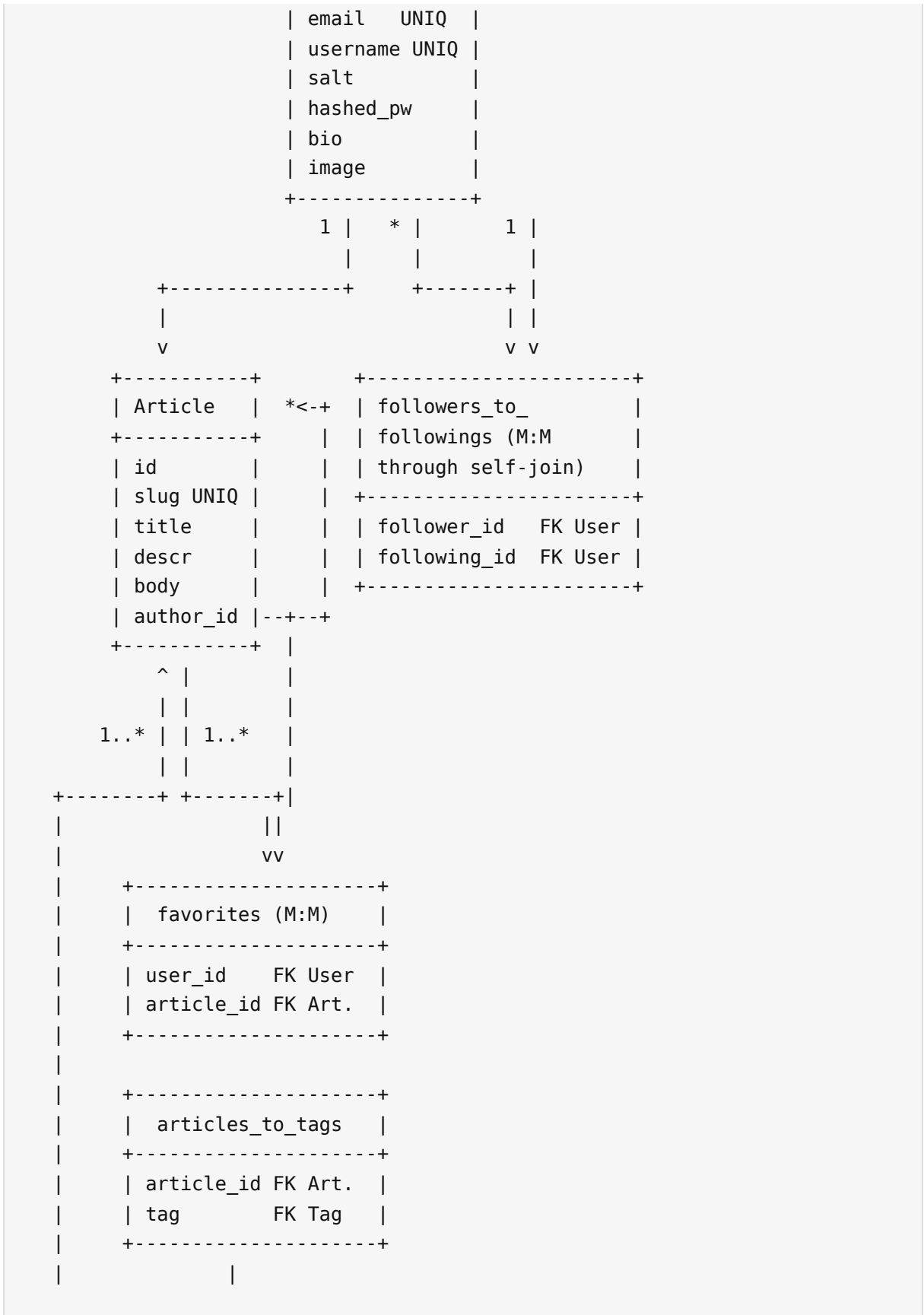
- `../setup.md` -- per-machine runbook (not data-model related).
- `../../steps/step.3000.txt` -- the arc plan, including the proposed F1..F11 fragment-family taxonomy in §(2g) that this `layered-architecture.md` first-pass tags against.
- `../../steps/step.3000.B.txt` -- the skeleton + LOCKs that govern these notes.

2. The data model

Source document: `notes/realworld-uml/entities.md` — original title: entities.md — RealWorld / nsidnev data model

Reverse-engineered from the upstream Python source at the pinned commit. Citations as `path:line`; paths are relative to `source/upstream/`.

ER diagram (overall)


| email UNIQ |
| username UNIQ |
| salt |
| hashed_pw |
| bio |
| image |
1 | * | 1 |
Article
| id |
| slug UNIQ |
| title |
| descr |
| body |
| author_id |
*<-+
followers_to_
followings (M:M
through self-join)
follower_id FK User
following_id FK User
1..* | | 1..*
favorites (M:M)
user_id FK User
article_id FK Art.
articles_to_tags
article_id FK Art.
tag FK Tag

```
   |              v
   |        +---------+
   |        |   Tag   |
   |        +---------+
   |        | tag PK  |
   |        +---------+
   |
   v
   +-------------------+
   |   Commentary      |  (a.k.a. "Comment" in API)
   +-------------------+
   | id                |
   | body              |
   | author_id  FK Usr |
   | article_id FK Art |
   +-------------------+
```

Profile is **not** a stored entity -- it is a projection of User joined with a per-requester "following" flag (`app/models/domain/profiles.py:6`).

Entities

User

Domain class: - `app/models/domain/users.py:7` -- `class User(RWModel)` with fields `username`, `email`, `bio = ""`, `image: Optional[str]`. - `app/models/domain/users.py:14` -- `class UserInDB(IDModelMixin, DateTimeModelMixin, User)` extends with `salt` + `hashed_password` plus `check_password()` / `change_password()` methods.

Database table (`users`, `app/db/migrations/versions/fdf8821871d7_main_tables.py:50`):

```
+----------------+----------+-------------+----------------------+
| Column         | Type     | Constraints | Notes                |
+----------------+----------+-------------+----------------------+
| id             | integer  | PK          |                      |
| username       | text     | NOT NULL,   | unique + indexed     |
|                |          | UNIQUE      |                      |
| email          | text     | NOT NULL,   | unique + indexed     |
|                |          | UNIQUE      |                      |
| salt           | text     | NOT NULL    | bcrypt salt          |
| hashed_password| text     | nullable    | bcrypt hash          |
```

```
| bio            | text     | NOT NULL,   | default ''           |
|                |          | DEFAULT ''  |                      |
| image          | text     | nullable    | URL (HttpUrl-typed   |
|                |          |             | at DTO layer)        |
| created_at     | timestmp | NOT NULL,   | server now()         |
|                |          | DEFAULT now |                      |
| updated_at     | timestmp | NOT NULL,   | trigger-updated      |
|                |          | DEFAULT now | (see §"updated_at    |
|                |          |             | triggers")           |
+----------------+----------+-------------+----------------------+
```

Validation rules (from the DTOs in `app/models/schemas/users.py`):

- `email` must be a valid email (`EmailStr`, `:7`).
- `username` non-empty in create; not constrained in update.
- `image` (update only) must be an `HttpUrl` (`:20`).
- Uniqueness for `email` + `username` checked in service layer before DB insert (`app/services/authentication.py:5`, `:14`) and reasserted by the DB's UNIQUE constraints.

Profile

Domain class (`app/models/domain/profiles.py:6`):

```
class Profile(RWModel):
    username: str
    bio: str = ""
    image: Optional[str] = None
    following: bool = False
```

NOT a table. Profile is a per-request projection of the requested User joined with the "is the requested viewer following this user" flag. The repository builds it in `app/db/repositories/profiles.py`.

The `following` flag is `false` when the request is anonymous (no JWT) or when the requester == the requested user.

Article

Domain class (`app/models/domain/articles.py:7`):

```
class Article(IDModelMixin, DateTimeModelMixin, RWModel):
    slug: str
    title: str
    description: str
    body: str
    tags: List[str]
    author: Profile         # NOTE: nested entity
    favorited: bool
    favorites_count: int
```

`tags`, `author`, `favorited`, `favorites_count` are **derived on read** (computed by the repository when loading the article), not stored on the `articles` row directly.

Database table (`articles`, migration `:99`):

```
+--------------+----------+-------------+----------------------+
| Column       | Type     | Constraints | Notes                |
+--------------+----------+-------------+----------------------+
| id           | integer  | PK          |                      |
| slug         | text     | NOT NULL,   | unique + indexed     |
|              |          | UNIQUE      | derived from title   |
|              |          |             | via python-slugify   |
|              |          |             | (services/articles.py|
|              |          |             | :15)                 |
| title        | text     | NOT NULL    |                      |
| description  | text     | NOT NULL    |                      |
| body         | text     | NOT NULL    |                      |
| author_id    | integer  | FK users.id | ON DELETE SET NULL   |
| created_at   | timestmp | NOT NULL    |                      |
| updated_at   | timestmp | NOT NULL    | trigger-updated      |
+--------------+----------+-------------+----------------------+
```

Tag

Domain shape: `List[str]` -- there is no dedicated Tag domain class. The DTO is just `schemas/tags.py:6` -- `TagsInList { tags: List[str] }`.

Database table (`tags`, migration `:121`):

```
+--------+------+-----+
| Column | Type | PK  |
```

```
+--------+------+-----+
| tag    | text | PK  |
+--------+------+-----+
```

Comment

Domain class (`app/models/domain/comments.py:6`):

```
class Comment(IDModelMixin, DateTimeModelMixin, RWModel):
    body: str
    author: Profile        # NOTE: nested entity
```

In the database the table is named `commentaries` (migration `:163`). The API exposes it as `comment` / `comments`.

Database table (`commentaries`, migration `:163`):

```
+------------+----------+-----------------+----------------+
| Column     | Type     | Constraints     | Notes          |
+------------+----------+-----------------+----------------+
| id         | integer  | PK              |                |
| body       | text     | NOT NULL        |                |
| author_id  | integer  | FK users.id,    |                |
|            |          | ON DELETE CASCDE|                |
| article_id | integer  | FK articles.id, |                |
|            |          | ON DELETE CASCDE|                |
| created_at | timestmp | NOT NULL        |                |
| updated_at | timestmp | NOT NULL        | trigger-updated|
+------------+----------+-----------------+----------------+
```

Favorite (M:M join, no domain class)

Pure join table; no domain class. Lives only in the DB.

Database table (`favorites`, migration `:145`):

```
+------------+----------+--------------------+
| Column     | Type     | Constraints        |
+------------+----------+--------------------+
| user_id    | integer  | FK users.id        |
```

```
|            |          | ON DELETE CASCADE  |
| article_id | integer  | FK articles.id     |
|            |          | ON DELETE CASCADE  |
+------------+----------+--------------------+
PK: (user_id, article_id)
```

Follow (M:M self-join on User, no domain class)

Pure join table; no domain class. The User-following-User relation.

Database table (`followers_to_followings`, migration `:80`):

```
+--------------+----------+--------------------+
| Column       | Type     | Constraints        |
+--------------+----------+--------------------+
| follower_id  | integer  | FK users.id        |
|              |          | ON DELETE CASCADE  |
| following_id | integer  | FK users.id        |
|              |          | ON DELETE CASCADE  |
+--------------+----------+--------------------+
PK: (follower_id, following_id)
```

Cross-cutting structural details

`updated_at` trigger pattern

All tables with `updated_at` use a Postgres trigger (`update_updated_at_column` function, migration `:14`) that sets `updated_at = now()` on every UPDATE. The Python ORM layer does NOT manage the timestamp; the database does.

Tables with the trigger: `users`, `articles`, `commentaries`.

IDModelMixin's `id_` field

`app/models/common.py:18` -- `id_: int = Field(0, alias="id")`. The pydantic field is `id_` (Python attribute) aliased to `id` on JSON / DB columns. This is to avoid shadowing the Python builtin.

Camel-case alias on the wire

`app/models/domain/rwmodel.py:17` -- `RWModel.Config` uses `alias_generator = convert_field_to_camel_case` so `favorites_count` (python) <-> `favoritesCount` (JSON). Every domain / schema class inherits from `RWModel` and gets this for free.

Datetime ISO-Z formatting

`rwmodel.py:5` -- a custom `json_encoder` formats datetimes as `...isoformat()` with the trailing `+00:00` replaced by `Z`. This is what the RealWorld spec expects.

DB schema source-of-truth

The complete schema lives in one alembic migration: `app/db/migrations/versions/fdf8821871d7_main_tables.py` (this is the only migration; the project has not evolved the schema since 2019). Sub-slice .D's per-family grammar for `F1 DataModelEntity` should pull from this migration as the canonical shape, not from the domain classes (which omit a few server-managed fields).

RealWorld spec references

External (for cross-check):

- Spec entities + fields: https://realworld-docs.netlify.app/specifications/backend/api-response-format/
- nsidnev's vendored Postman collection encodes the authoritative wire format for THIS implementation: `source/upstream/postman/Conduit.postman_collection.json`.

3. The route table

Source document: `notes/realworld-uml/routes.md` — original title: routes.md — RealWorld / nsidnev route table

All routes are mounted under `/api` (per `app/core/settings/app.py:31` -- `api_prefix: str = "/api"`). Top-level router composition is at `app/api/routes/api.py`.

All routes

```
+--------+---------------------------------------+---------+------------------+
| Method | Path                                  | Auth    | Source           |
```

POST	/api/users = register	none	authentication .py:53
POST	/api/users/login = login	none	authentication .py:21
GET	/api/user = retrieve current user	JWT	users.py:16
PUT	/api/user = update current user	JWT	users.py:36
GET	/api/profiles/{username} = retrieve a profile	JWT-opt	profiles.py:16
POST	/api/profiles/{username}/follow = follow a user	JWT	profiles.py:27
DELETE	/api/profiles/{username}/follow = unfollow a user	JWT	profiles.py:54
GET	/api/articles = list articles (filters: tag, author, favorited; pagination)	JWT-opt	articles_ resource.py:28
POST	/api/articles = create a new article	JWT	articles_ resource.py:54
GET	/api/articles/{slug} = retrieve an article by slug	JWT-opt	articles_ resource.py:80
PUT	/api/articles/{slug} = update an article	JWT	articles_ resource.py:85
DELETE	/api/articles/{slug} = delete an article	JWT	articles_ resource.py:104
GET	/api/articles/feed = articles by users I follow	JWT	articles_ common.py:21
POST	/api/articles/{slug}/favorite = favorite an article	JWT	articles_ common.py:48
DELETE	/api/articles/{slug}/favorite = unfavorite an article	JWT	articles_ common.py:75

```
+--------+---------------------------------------+---------+-----------------+
| GET    | /api/articles/{slug}/comments         | JWT-opt | comments.py:24  |
|        |   = list comments for an article      |         |                 |
+--------+---------------------------------------+---------+-----------------+
| POST   | /api/articles/{slug}/comments         | JWT     | comments.py:38  |
|        |   = create a comment                  |         |                 |
+--------+---------------------------------------+---------+-----------------+
| DELETE | /api/articles/{slug}/comments/{id}    | JWT     | comments.py:57  |
|        |   = delete a comment                  |         |                 |
+--------+---------------------------------------+---------+-----------------+
| GET    | /api/tags                             | none    | tags.py:9       |
|        |   = list all tags                     |         |                 |
+--------+---------------------------------------+---------+-----------------+
```

19 routes total. The Postman collection at `source/upstream/postman/Conduit.postman_collection.json` exercises every one with positive + selected negative tests (280 assertions).

Auth column legend

- `none` -- no Authorization header expected; routes are anonymous.
- `JWT` -- `Authorization: Token <jwt>` header required; 401/403 on absent / bad token.
- `JWT-opt` -- header optional; behavior may differ when authenticated (e.g. `following` flag on Profile, `favorited` flag on Article).

The auth header format is `Token <jwt>` (note: not `Bearer`). This is set by `app/core/settings/app.py:33` -- `jwt_token_prefix: str = "Token"` -- and validated in `app/api/dependencies/authentication.py:49`. **Deviation from RealWorld spec:** the spec says `Token <jwt>` too, so no deviation; just calling it out because `Bearer` is the more common modern choice and any TS port needs to match `Token`.

Routing composition

Per `app/api/routes/api.py`:

```
/api
 ├── /users      <- prefix for authentication router
 │     POST /         (register)
 │     POST /login    (login)
 │
 ├── /user       <- prefix for users router
```

```
  │     GET   /        (retrieve current)
  │     PUT   /        (update current)
  │
  ├── /profiles   <- prefix for profiles router
  │     GET    /{username}
  │     POST   /{username}/follow
  │     DELETE /{username}/follow
  │
  ├── /          <- no prefix for articles (paths embed /articles/)
  │     GET    /articles
  │     POST   /articles
  │     GET    /articles/feed
  │     GET    /articles/{slug}
  │     PUT    /articles/{slug}
  │     DELETE /articles/{slug}
  │     POST   /articles/{slug}/favorite
  │     DELETE /articles/{slug}/favorite
  │
  ├── /articles/{slug}/comments  <- prefix for comments router
  │     GET    /
  │     POST   /
  │     DELETE /{comment_id}
  │
  └── /tags      <- prefix for tags router
        GET    /
```

Note the two `articles` sub-routers (`articles_common` for feed + favorite, `articles_resource` for CRUD), composed in `app/api/routes/articles/api.py`. Both are mounted at `/articles`.

Error response shape

Per `app/api/errors/http_error.py:6`:

```
{ "errors": ["error message string"] }
```

with the HTTP status from the raised `HTTPException`.

Per `app/api/errors/validation_error.py:13` (422 responses):

```
{ "errors": [ { "loc": [...], "msg": "...", "type": "..." }, ... ] }
```

This is FastAPI / pydantic's standard validation-error format wrapped in the `errors` envelope.

Error-string registry

All human-readable error messages are constants in `app/resources/strings.py` (25 LOC). Examples:

- `USERNAME_TAKEN`, `EMAIL_TAKEN` -> 400.
- `INCORRECT_LOGIN_INPUT` -> 400.
- `AUTHENTICATION_REQUIRED`, `WRONG_TOKEN_PREFIX`, `MALFORMED_PAYLOAD` -> 403.
- `USER_DOES_NOT_EXIST_ERROR`, `ARTICLE_DOES_NOT_EXIST_ERROR`, `COMMENT_DOES_NOT_EXIST` -> 404.
- `UNABLE_TO_FOLLOW_YOURSELF`, `USER_IS_ALREADY_FOLLOWED`, `ARTICLE_ALREADY_EXISTS`, `ARTICLE_IS_ALREADY_FAVORITED`, etc. -> 400.

The TS port should keep these strings byte-identical (the Postman tests assert against them in some cases).

Status codes used

```
+---------+------+----------------------------------+
| HTTP    | Use  | Where                            |
+---------+------+----------------------------------+
| 200 OK  | std  | most GETs + PUTs + login         |
| 201     | Crtd | POST /api/users (register),      |
|         |      | POST /api/articles,              |
|         |      | POST /api/articles/{}/comments    |
| 204     | NoC  | DELETE /api/articles/{},         |
|         |      | DELETE /api/articles/{}/comments/{}|
| 400     | bad  | uniqueness, invalid logical     |
|         |      | state (already followed,         |
|         |      | already favorited, can't follow |
|         |      | self, etc.)                       |
| 403     | forb | wrong/missing JWT, not the      |
|         |      | author of an article/comment    |
| 404     | nf   | unknown user / article / comment|
| 422     | val  | DTO validation failure          |
+---------+------+----------------------------------+
```

Deviations from the abstract RealWorld spec

None observed at this pin. The Postman collection vendored with the upstream is aligned with the upstream's choices, so spec deviations (if any exist) would have been flagged by the Postman tests at .A baseline -- and the .A baseline is clean (280/280). The TS port targets the upstream behavior as authoritative.

If a future .H / .I sub-slice surfaces a deviation, it goes in this section as a new sub-heading.

4. Request and response contracts

Source document: `notes/realworld-uml/contracts.md` — original title: contracts.md — per-route request / response contracts

Pulled from the Pydantic schemas in `app/models/schemas/`. JSON-wire shapes use camelCase aliases via the `RWSchema.Config` inherited from `RWModel` (see `entities.md` §"Camel-case alias on the wire").

Authentication

POST /api/users (register) — 201

Source: `app/api/routes/authentication.py:53`, `schemas/users.py:13`.

Request body:

```
{
  "user": {
    "email": "<EmailStr>",
    "password": "<str>",
    "username": "<str>"
  }
}
```

Response body (= `UserInResponse` / `UserWithToken`, `schemas/users.py:24,28`):

```
{
  "user": {
    "email":    "<EmailStr>",
    "username": "<str>",
    "bio":      "<str>",          // default ""
```

```
    "image":    "<str|null>",
    "token":    "<jwt string>"
  }
}
```

POST /api/users/login — 200

Source: `app/api/routes/authentication.py:21`, `schemas/users.py:7`.

Request:

```
{ "user": { "email": "<EmailStr>", "password": "<str>" } }
```

Response: `UserInResponse` (same shape as register).

Current user

GET /api/user — 200

Source: `app/api/routes/users.py:16`.

Request: no body. Header `Authorization: Token <jwt>` required.

Response: `UserInResponse` (re-issues a fresh JWT in `.user.token` on every call -- see auth-vertical.md §3 for why).

PUT /api/user — 200

Source: `app/api/routes/users.py:36`, `schemas/users.py:16`.

Request:

```
{
  "user": {
    "username": "<str|missing>",
    "email":    "<EmailStr|missing>",
    "password": "<str|missing>",
    "bio":      "<str|missing>",
    "image":    "<HttpUrl|missing>"
```

```
  }
}
```

All fields optional; missing fields leave existing values untouched.

Response: `UserInResponse`.

Profiles

GET /api/profiles/{username} — 200

Source: `app/api/routes/profiles.py:16`, `schemas/profiles.py:7`.

Response (= `ProfileInResponse`):

```
{
  "profile": {
    "username":  "<str>",
    "bio":       "<str>",          // default ""
    "image":     "<str|null>",
    "following": <bool>            // false if anon
  }
}
```

POST /api/profiles/{username}/follow — 200

DELETE /api/profiles/{username}/follow — 200

Both: empty request body; response = `ProfileInResponse` with `following` set to true / false respectively.

Articles

GET /api/articles — 200

Source: `app/api/routes/articles/articles_resource.py:28`, `schemas/articles.py` (filters at `:35`).

Query parameters:

```
+-------------+--------+------------+----------+
| Param       | Type   | Default    | Required |
+-------------+--------+------------+----------+
| tag         | str    | -          | no       |
| author      | str    | -          | no       |
| favorited   | str    | -          | no       |
| limit       | int>=1 | 20         | no       |
| offset      | int>=0 | 0          | no       |
+-------------+--------+------------+----------+
```

Response (= `ListOfArticlesInResponse`, `schemas/articles.py:29`):

```
{
  "articles":       [ <ArticleForResponse>, ... ],
  "articlesCount":  <int>
}
```

POST /api/articles — 201

Source: `articles_resource.py:54`, `schemas/articles.py:13`.

Request:

```
{
  "article": {
    "title":       "<str>",
    "description": "<str>",
    "body":        "<str>",
    "tagList":     [ "<str>", ... ]    // default []
  }
}
```

Response (= `ArticleInResponse`):

```
{
  "article": <ArticleForResponse>
}
```

`ArticleForResponse` shape

(`schemas/articles.py:10`, extends domain Article with `tagList` alias for `tags`):

```
{
  "slug":           "<str>",
  "title":          "<str>",
  "description":    "<str>",
  "body":           "<str>",
  "tagList":        [ "<str>", ... ],
  "createdAt":      "<ISO Z>",
  "updatedAt":      "<ISO Z>",
  "favorited":      <bool>,
  "favoritesCount": <int>,
  "author": {
    "username":  "<str>",
    "bio":       "<str>",
    "image":     "<str|null>",
    "following": <bool>
  }
}
```

GET /api/articles/{slug} — 200

Response: `ArticleInResponse`.

PUT /api/articles/{slug} — 200

Source: `articles_resource.py:85`, `schemas/articles.py:21`.

Request:

```
{
  "article": {
    "title":       "<str|missing>",
    "description": "<str|missing>",
    "body":        "<str|missing>"
  }
}
```

Updating `title` re-slugs (`services/articles.py:15`).

Response: `ArticleInResponse`.

DELETE /api/articles/{slug} — 204

No body either direction.

GET /api/articles/feed — 200

Source: `articles_common.py:21`.

Query params:

```
+--------+--------+---------+
| Param  | Type   | Default |
+--------+--------+---------+
| limit  | int>=1 | 20      |
| offset | int>=0 | 0       |
+--------+--------+---------+
```

Response: `ListOfArticlesInResponse`.

POST /api/articles/{slug}/favorite — 200

DELETE /api/articles/{slug}/favorite — 200

Both: empty request body; response = `ArticleInResponse` with `favorited` flipped and `favoritesCount` adjusted.

Comments

GET /api/articles/{slug}/comments — 200

Source: `app/api/routes/comments.py:24`, `schemas/comments.py:7`.

Response (= `ListOfCommentsInResponse`):

```
{
  "comments": [ <Comment>, ... ]
}
```

`Comment` shape:

```
{
  "id":        <int>,
  "createdAt": "<ISO Z>",
  "updatedAt": "<ISO Z>",
  "body":      "<str>",
  "author":    <Profile>
}
```

POST /api/articles/{slug}/comments — 201

Source: `comments.py:38`, `schemas/comments.py:15`.

Request:

```
{ "comment": { "body": "<str>" } }
```

Response: `CommentInResponse` -- `{ "comment": <Comment> }`.

DELETE /api/articles/{slug}/comments/{comment_id} — 204

No body either direction. Author-only (403 if not author).

Tags

GET /api/tags — 200

Source: `app/api/routes/tags.py:9`, `schemas/tags.py:6`.

Response: `{ "tags": [ "<str>", ... ] }`.

Error responses

Per `routes.md` §"Error response shape":

- non-422: `{ "errors": [ "<message>" ] }` with the status code from the raised HTTPException.
- 422: `{ "errors": [ { "loc": [...], "msg": "...", "type": "..." }, ... ] }` -- pydantic / FastAPI default, wrapped in `errors`.

DTO map (Pydantic class -> route)

```
+----------------------------+--------------------------+
| Pydantic class             | Used by                  |
+----------------------------+--------------------------+
| UserInCreate               | POST   /api/users        |
| UserInLogin                | POST   /api/users/login  |
| UserInUpdate               | PUT    /api/user         |
| UserInResponse             | response for above + GET /api/user |
| ProfileInResponse          | profiles routes          |
| ArticleInCreate            | POST   /api/articles     |
| ArticleInUpdate            | PUT    /api/articles/{slug} |
| ArticleInResponse          | response for single-article routes |
| ArticleForResponse         | response item shape      |
| ListOfArticlesInResponse   | GET    /api/articles, feed |
| ArticlesFilters            | query-param binding for  |
|                            | GET /api/articles        |
| CommentInCreate            | POST   /api/articles/{slug}/comments |
| CommentInResponse          | single-comment response  |
| ListOfCommentsInResponse   | GET    .../comments      |
| TagsInList                 | GET    /api/tags         |
+----------------------------+--------------------------+
```

15 DTOs total -- a useful sizing data point for sub-slice .D (per-family grammar authoring).

5. How a request moves through the layers

Source document: `notes/realworld-uml/data-flow.md` — original title: data-flow.md — sequence diagrams for the auth vertical

Classic vertical-lifeline form. Each diagram corresponds to one endpoint of the auth vertical (see `auth-vertical.md` for narrative + line citations).

1. POST /api/users (register) -- happy path

```
client       authn route        users repo       security svc       jwt svc       DB
   |             |                  |                  |              |          |
   |---POST----->|                  |                  |              |          |
   |  {user:...}  |                  |                  |              |          |
   |             |---get_user_by_username---->         |              |          |
   |             |                  |---SELECT-------------------------|--------->|
```

```
   |              |                 |<---EntityDoesNotExist-----------|---empty--|
   |              |  (== not taken, OK to continue)                   |          |
   |              |                 |                  |               |          |
   |              |---get_user_by_email------>         |               |          |
   |              |                 |---SELECT-------------------------|--------->|
   |              |                 |<---EntityDoesNotExist-----------|---empty--|
   |              |                 |                  |               |          |
   |              |---create_user(u,e,p)----->         |               |          |
   |              |                 |---generate_salt-->               |          |
   |              |                 |<--salt_str-------                |          |
   |              |                 |---hash(salt+p)-->                |          |
   |              |                 |<--hashed_str-----                |          |
   |              |                 |---INSERT INTO users--------------|--------->|
   |              |                 |<---row { id, created_at, ... }--|<---row---|
   |              |<---UserInDB-----|                  |               |          |
   |              |                 |                  |               |          |
   |              |---create_access_token_for_user(user, secret)----->|          |
   |              |                                                   |          |
   |              |  jwt.encode({username, exp, sub}, HS256)          |          |
   |              |<-------------------------------------------token----|          |
   |              |                                                   |          |
   |              |  UserInResponse(user=UserWithToken(...token=token))|          |
   |<--201--------|                                                   |          |
   |  {user:...}  |                                                   |          |
```

2. POST /api/users/login -- happy path

```
client       authn route        users repo       security svc       jwt svc       DB
   |              |                 |                  |               |          |
   |---POST-----> |                 |                  |               |          |
   |  {user:...}  |                 |                  |               |          |
   |              |---get_user_by_email(e)--->         |               |          |
   |              |                 |---SELECT-------------------------|--------->|
   |              |                 |<---row { ...salt, hashed_pw }---|<---row---|
   |              |<---UserInDB-----|                  |               |          |
   |              |                 |                  |               |          |
   |              |---user.check_password(plain)---------------------->|          |
   |              |                 |                  |               |          |
   |              |  internally: security.verify_password(salt+plain, hash)       |
   |              |                 |                                  |          |
   |              |<---True---------|                  |               |          |
   |              |                 |                  |               |          |
   |              |---create_access_token_for_user(user, secret)----->|          |
   |              |<-------------------------------------------token----|          |
   |              |                                                   |          |
```

```
   |              |  UserInResponse(user=UserWithToken(...))           |          |
   |<--200--------|                                                    |          |
```

Login failure (wrong password) -- collapsed

```
client       authn route        users repo       security svc
   |              |                 |                  |
   |---POST----->|                |                 |
   |              |---get_user_by_email(e)--->         |
   |              |<---UserInDB-----|                  |
   |              |---check_password(plain)------------|
   |              |<---False--------|                  |
   |              |                                    |
   |              |  raise HTTPException(400, INCORRECT_LOGIN_INPUT)
   |<--400--------|
   |  {"errors": ["Incorrect email or password."]}
```

Login failure (no such user)

```
client       authn route        users repo
   |              |                 |
   |---POST----->|                |
   |              |---get_user_by_email(e)--->
   |              |<---EntityDoesNotExist--|   (caught, re-raised
   |              |                            as the SAME 400)
   |              |
   |              |  raise HTTPException(400, INCORRECT_LOGIN_INPUT)
   |<--400--------|
```

Same error string for both branches -- prevents account enumeration via response-message diff.

3. GET /api/user -- happy path

```
client     api key dep      authz_header_dep    current_user_dep    users repo    jwt svc      DB
   |           |                 |                 |                 |          |          |
   |---GET----->|                 |                 |                 |          |          |
   |  Authoriz: |                 |                 |                 |          |          |
   |  Token jwt |                 |                 |                 |          |          |
   |           |                 |                 |                 |          |          |
   |           |---read "Authorization" header----->|                |          |          |
   |           |<---api_key str----------|          |                 |          |          |
   |           |                 |--split "Token "-->                |          |          |
```

```
   |           |                 |<-- (prefix, jwt)--|                 |           |          |
   |           |                 |  (prefix == "Token", OK)            |           |          |
   |           |                 |<--<jwt>-----------|                 |           |          |
   |           |                 |                   |                 |           |          |
   |           |                 |                   |---get_username_from_token(jwt, secret)->|
   |           |                 |                   |  jwt.decode(HS256) ->                |
   |           |                 |                   |  JWTUser.username                   |
   |           |                 |                   |<------------------------username------|
   |           |                 |                   |                                      |
   |           |                 |                   |---get_user_by_username(username)----->|
   |           |                 |                   |                 |---SELECT---------->|
   |           |                 |                   |                 |<--row { ...}-------|
   |           |                 |                   |<-----UserInDB---|                    |
   |           |                 |                   |                                       |
   |           |  (User now bound as dependency)                                             |
   |           |                                                                             |
   |           |---route handler body:                                                       |
   |           |     token = create_access_token_for_user(user, secret)  (re-issue)          |
   |           |     return UserInResponse(user=UserWithToken(...token=token))              |
   |<--200-----|                                                                             |
   |  {user:...}|                                                                            |
```

Failure: missing Authorization header

```
client     api key dep
   |            |
   |---GET----->|
   |            |---read "Authorization" -> KeyError
   |            |
   |            |  RWAPIKeyHeader catches Starlette's
   |            |  HTTPException(401) and re-raises with
   |            |  detail = AUTHENTICATION_REQUIRED, status 401
   |<--401------|
   |  {"errors": ["Authentication required."]}
```

(Actually -- check: nsidnev's RWAPIKeyHeader re-raises with the **same** status code as the original Starlette exception, which is 403 in the strict-required path. Refer to `app/api/dependencies/authentication.py:18`. The Postman tests record the actual return for the canonical answer.)

Failure: bad JWT prefix

```
client     api key dep   authz header dep
   |            |             |
   |---GET----->|             |
```

```
   |  Bearer jwt|             |
   |            |---value---->|
   |            |             |---split " "--> (Bearer, jwt)
   |            |             |  prefix != "Token"
   |            |             |  raise 403 WRONG_TOKEN_PREFIX
   |<--403--------------------|
   |  {"errors": ["Unsupported authorization type."]}
```

Failure: bad JWT (expired / tampered)

```
client     api key dep   authz_header   current_user_dep   jwt svc
   |            |             |              |               |
   |---GET----->|             |              |               |
   |            |---value---->|              |               |
   |            |             |--<jwt>------>|               |
   |            |             |              |---decode----->|
   |            |             |              |<-PyJWTError-(.expired/...)
   |            |             |              |
   |            |             |              |  raise ValueError("unable to decode")
   |            |             |              |  caught -> raise 403 MALFORMED_PAYLOAD
   |<--403--------------------|--------------|
   |  {"errors": ["Could not validate credentials."]}
```

Notes for sub-slice .F (per-family emitters)

Each diagram step is a candidate **execution edge** in the hierarchy DAG:

- "route reads body" -> F3 RouteContract executed.
- "service helper called" -> F8 ValidationRule or F6 ServiceOrchestration.
- "repo method called" -> F5 RepositoryQuery.
- "JWT issued / verified" -> F7 AuthPolicy.
- "exception caught -> HTTPException raised" -> F10 ErrorHandling.

The per-family emitters in .F should preserve this sequencing in the generated TS -- the assertions in the Postman tests will catch a permuted order if it changes observable behavior (e.g. uniqueness checks BEFORE password hash, to surface the user-visible 400 promptly rather than do extra crypto work).

6. The authentication vertical, end to end

Source document: `notes/realworld-uml/auth-vertical.md` — original title: auth-vertical.md — User-auth vertical end-to-end

Focused walkthrough of the three endpoints that sub-slices .E-.G will exercise end-to-end. These are the smallest non-toy slice of the upstream that touches the entire stack (routing, DTOs, service helpers, repository, DB, crypto, JWT).

```
+----+------------------+--------+---------+---------------+
| #  | Endpoint         | Method | Auth    | Status        |
+----+------------------+--------+---------+---------------+
| 1  | /api/users       | POST   | none    | 201           |
|    | (register)       |        |         |               |
+----+------------------+--------+---------+---------------+
| 2  | /api/users/login | POST   | none    | 200 / 400     |
+----+------------------+--------+---------+---------------+
| 3  | /api/user        | GET    | JWT     | 200 / 403     |
+----+------------------+--------+---------+---------------+
```

1. POST /api/users -- register

Source: `app/api/routes/authentication.py:53`.

Request

```
POST /api/users
Content-Type: application/json

{
  "user": {
    "email":    "jake@example.com",
    "password": "hunter2",
    "username": "jake"
  }
}
```

The `embed=True, alias="user"` on the Body parameter (`authentication.py:62`) is what nests the DTO under `user`.

Handler flow

```
1. Parse body  -> UserInCreate                  (schemas/users.py:13)
2. check_username_is_taken(repo, username)      (services/authentication.py:5)
   -> if True: raise 400 USERNAME_TAKEN
3. check_email_is_taken(repo, email)            (services/authentication.py:14)
   -> if True: raise 400 EMAIL_TAKEN
4. repo.create_user(username, email, password)  (db/repositories/users.py:32)
   a. instantiate UserInDB with no salt/hash
   b. user.change_password(password)            (models/domain/users.py:21)
       -> security.generate_salt() + bcrypt hash
                                                (services/security.py:6,15)
   c. async transaction:
      queries.create_new_user(connection, ...)  (aiosql; SQL in
                                                 db/queries/sql/)
   d. user.copy(update=dict(user_row))           -- merge DB return cols
5. create_access_token_for_user(user, secret)   (services/jwt.py:23)
   -> JWT signed HS256, payload = {username, exp, sub="access"}
6. assemble UserInResponse / UserWithToken      (schemas/users.py:24,28)
7. return 201 with body
```

Response (201)

```
{
  "user": {
    "email":    "jake@example.com",
    "username": "jake",
    "bio":      "",
    "image":    null,
    "token":    "<jwt>"
  }
}
```

Error cases

- 400 `{"errors": ["User with this username already exists."]}`
- 400 `{"errors": ["User with this email already exists."]}`
- 422 (pydantic validation) on bad email, missing fields, etc.

2. POST /api/users/login

Source: `app/api/routes/authentication.py:21`.

Request

```
POST /api/users/login
Content-Type: application/json

{
  "user": { "email": "jake@example.com", "password": "hunter2" }
}
```

Handler flow

```
1. Parse body -> UserInLogin                    (schemas/users.py:7)
2. repo.get_user_by_email(email)                (db/repositories/users.py:9)
   - on EntityDoesNotExist (db/errors.py:1) ->
     raise INCORRECT_LOGIN_INPUT (400)
3. user.check_password(plain)                   (models/domain/users.py:18)
   -> security.verify_password(salt + plain, hash)
                                                (services/security.py:11)
   - on False -> raise INCORRECT_LOGIN_INPUT (400)
4. create_access_token_for_user(user, secret)
5. assemble UserInResponse
6. return 200 with body
```

The same INCORRECT_LOGIN_INPUT message is returned for both "no such user" and "wrong password" -- intentional, to prevent username enumeration via timing or message diff.

Response (200)

Same shape as register response.

Error cases

- 400 `{"errors": ["Incorrect email or password."]}`
- 422 on bad request shape.

3. GET /api/user (current user)

Source: `app/api/routes/users.py:16`.

Request

```
GET /api/user
Authorization: Token <jwt>
```

Handler flow

```
1. Auth dependency chain runs BEFORE the handler:
   a. RWAPIKeyHeader (deps/auth.py:18) reads "Authorization"
   b. _get_authorization_header (:43):
      - split "Token <jwt>" -> if not 2 parts: 403 WRONG_TOKEN_PREFIX
      - if prefix != "Token": 403 WRONG_TOKEN_PREFIX
      - return raw <jwt>
   c. _get_current_user (:80):
      - jwt.get_username_from_token(token, secret)
                                                (services/jwt.py:32)
        -> jwt.decode(HS256) -> JWTUser.username
        -> on PyJWTError / ValidationError: ValueError
      - on ValueError: 403 MALFORMED_PAYLOAD
      - repo.get_user_by_username(username)     (db/repositories/users.py:18)
        -> on EntityDoesNotExist: 403 MALFORMED_PAYLOAD
      - return User
2. Handler body:
   - create_access_token_for_user(user, secret)  (re-issues JWT)
   - assemble UserInResponse
   - return 200
```

Why re-issue the JWT on every GET /api/user? It is the same JWT factory the auth routes use; for the RealWorld spec clients expect the `user.token` field populated on every read; sliding the expiry forward by re-issuing is a side effect, not the goal. The TS port should preserve this behavior (Postman tests rely on a non-null token in the response).

Response (200)

Same shape as register / login response.

Error cases

- 403 `{"errors": ["Authentication required."]}` on missing header (`RWAPIKeyHeader` intercept).
- 403 `{"errors": ["Wrong token prefix."]}` on malformed header.
- 403 `{"errors": ["Could not validate credentials."]}` on bad JWT / unknown username.

Cross-cutting details

JWT shape

`app/services/jwt.py:11-:22`:

```
Algorithm:  HS256
Secret:     settings.secret_key (from .env SECRET_KEY)
Expiry:     7 days from issue (ACCESS_TOKEN_EXPIRE_MINUTES = 60*24*7)
Payload:    { "username": "<str>", "exp": <unix-ts>, "sub": "access" }
Prefix:     "Token " (NOT "Bearer ")
```

Password storage

`app/db/repositories/users.py:40` -- the DB columns are `salt` + `hashed_password`. `User.change_password(p)` does:

```
salt = bcrypt.gensalt().decode()                    # services/security.py:6
hashed = pwd_context.hash(salt + p)                  # services/security.py:15 with bcrypt scheme
```

The salt is stored separately even though bcrypt embeds the salt in the hash. The DB has `salt` for legacy reasons; the TS port can keep the column (parity with upstream's data) or collapse it.

Verification (`security.py:11`):

```
pwd_context.verify(salt + plain, hashed_password)
```

i.e. salt is prepended to the plaintext before bcrypt verifies. **Important:** the TS implementation must replicate this exact ordering or existing rows (carried across the wire from upstream) won't verify.

For sub-slice .G (fresh TS service running independently) the database is freshly migrated, so cross-implementation row compatibility is not a v0 requirement -- but the semantic equivalence is,

since the Postman tests issue register -> login flows and expect them to round-trip.

Pool acquisition for DB access

`app/api/dependencies/database.py:9` -- the request handler acquires a connection from the asyncpg pool via:

```
pool = request.app.state.pool      # set at startup
async with pool.acquire() as conn:
    yield conn                       # FastAPI Depends
```

Each request gets its own connection, returned on completion.

Cross-references

- Sequence diagrams for the three flows: `data-flow.md`.
- DTO definitions: `contracts.md` §Authentication + §"Current user".
- Entity definitions: `entities.md` §User.
- Route table row for each: `routes.md`.

7. The per-file layer map

Source document: `notes/realworld-uml/layered-architecture.md` — original title: layered-architecture.md — per-file map + provisional fragment-family

> **NOTE (sub-slice .C, 2026-06-12):** The "Family" column in the table below is **superseded** by the FROZEN per-fragment map in `../fragment-mapping.md`. The .B content here is preserved as the historic record; .C's per-fragment ground truth (with refined F1..F12 taxonomy, F6/F9 dropped, F12 added) is the canonical reference for sub-slices .D / .E / .F. The family *definitions* live in `../../sgdl/families.md`.

Reverse-engineered from the upstream tree at the pinned commit. The "Family" column is a **PROVISIONAL** first-pass hypothesis (per the slice's lock). Sub-slice .C is where the taxonomy gets validated and frozen; this column is .B's input to that pass.

Family codes refer to step.3000.txt §(2g): F1 DataModelEntity, F2 DataModelRelationship, F3 RouteContract, F4 RouteHandler, F5 RepositoryQuery, F6 ServiceOrchestration, F7 AuthPolicy, F8 ValidationRule, F9 ImportDependency, F10 ErrorHandling, F11 Configuration.

Per-file table

File	Layer	Role	Family
app/main.py	entry	FastAPI app factory; CORS; error handlers; router mount; startup/shutdwn	(boot, out of taxon.)
app/core/config.py	core	settings factory by APP_ENV	F11
app/core/settings/{base,app, development,production,test}.py	core	env-bound settings	F11
app/core/logging.py	core	loguru intercept	(infra)
app/core/events.py	core	startup / shutdown hooks	(infra)
app/api/routes/api.py	api	top-level router composition	F3 (the route tree)
app/api/routes/authentication.py	api	POST /users, POST /login	F3 + F4
app/api/routes/users.py	api	GET /user, PUT /user	F3 + F4
app/api/routes/profiles.py	api	GET/POST/DELETE /profiles/{u}/*	F3 + F4
app/api/routes/articles/api.py	api	sub-router composition	F3
app/api/routes/articles/ articles_resource.py	api	article CRUD	F3 + F4
app/api/routes/articles/ articles_common.py	api	feed + favorite	F3 + F4
app/api/routes/comments.py	api	comment CRUD	F3 + F4

app/api/routes/tags.py	api	GET /tags	F3 + F4
app/api/dependencies/database.py	api	DI: connection from pool	(DI infra)
app/api/dependencies/authentication.py	api	DI: JWT parse + current user resolution	F7 (AuthPo- licy)
app/api/dependencies/profiles.py	api	DI: profile by-username path param	F8 (Valid. Rule)? or part of F4
app/api/dependencies/articles.py	api	DI: article by-slug; filters builder; modify check	F8 / F4 (split provis.)
app/api/dependencies/comments.py	api	DI: comment by-id; modify check	F8 / F4
app/api/errors/http_error.py	api	global 4xx/5xx envelope	F10
app/api/errors/validation_error.py	api	422 envelope	F10
app/services/authentication.py	svc	check_username _is_taken etc.	F8 (Valid. Rule)
app/services/security.py	svc	bcrypt wrappers	F7
app/services/jwt.py	svc	JWT issue + verify	F7
app/services/articles.py	svc	slug gen, modify check	F6 (orch) + F8
app/services/comments.py	svc	modify check	F8
app/db/repositories/base.py	db	repo base class	(infra)
app/db/repositories/users.py	db	user CRUD	F5
app/db/repositories/profiles.py	db	profile build	F5 + F2

		+ follow mgmt	(relat.)	
app/db/repositories/articles.py	db	article CRUD, favorite mgmt, filtering, feed	F5 + F2	
app/db/repositories/comments.py	db	comment CRUD	F5	
app/db/repositories/tags.py	db	tag list + upsert	F5	
app/db/queries/queries.py	db	aiosql load	(infra)	
app/db/queries/tables.py	db	SQL table constants	(infra)	
app/db/migrations/...	db	alembic schema	F1 (the canonical data-model source)	
app/db/errors.py	db	EntityDoesNotExist exception	F10	
app/db/events.py	db	asyncpg pool startup / shutdown	(infra)	
app/models/common.py	model	IDModelMixin, DateTimeModel-Mixin	(mixin)	
app/models/domain/{rwmodel,users,profiles,articles,comments}.py	model	pydantic domain classes	F1	
app/models/schemas/{rwschema,users,profiles,articles,comments,tags,jwt}.py	model	request / response DTOs	F1 + F3 (DTO is contract + entity shape)	
app/resources/strings.py	res	error message string consts	(string table)	

Layer summary

Layer	Files	What it does
entry	1	wires the FastAPI app
core	8	settings + logging + lifecycle
api	16	HTTP surface (routes + deps + error handlers)
svc	5	business logic + auth crypto
db	10	persistence + migrations
model	10	data shapes (domain + DTOs)
res	1	strings
TOTAL	51	

Provisional fragment-family histogram

Fam	Code	Files (incl. partial overlap)
F1	DataModel Entity	9 (alembic + domain + DTO schemas)
F2	Relation	2 (profiles repo, articles repo -- relationship management methods)
F3	Route Contract	8 (route files + DTO schemas)
F4	Route Handler	7 (route handler bodies)
F5	Repo Query	5 (per-entity repository)
F6	Service Orchstr.	1 (services/articles.py:15) -- only `get_slug_for_article` is a true orchstr here; this is small data!
F7	AuthPol	3 (security, jwt, deps/auth)

```
+-----+----------+-------------------------------+
| F8  | Valid    |  5 (services/auth check_*,    |
|     | Rule     |     deps/profiles,            |
|     |          |     deps/articles,            |
|     |          |     deps/comments, dto field  |
|     |          |     validators in schemas)    |
+-----+----------+-------------------------------+
| F9  | Import   | ~ every file. Pure structural |
|     |  Dep     | edges; one-line fragments.    |
+-----+----------+-------------------------------+
| F10 | Error    |  3 (errors/http_error,        |
|     | Handling |     errors/validation_error,  |
|     |          |     db/errors)                |
+-----+----------+-------------------------------+
| F11 | Config   |  6 (core/config +             |
|     |          |     core/settings/*)          |
+-----+----------+-------------------------------+
```

Notes for sub-slice .C

Things .C should verify / refine:

1. **F6 ServiceOrchestration appears thin.** nsidnev's service layer is mostly tiny helpers (uniqueness checks, slug generation, modify-permission checks). The "compose multiple repository calls" pattern actually lives in the ROUTE HANDLERS (F4) -- e.g. `authentication.py:53` register: check username taken -> check email taken -> create -> issue JWT -> assemble response. If F6 is defined to require "2+ repo / external calls", many route handlers qualify.

.C should either: (a) accept that route handlers are F4 even when they orchestrate, and reserve F6 for non-route orchestrations; or (b) reclassify orchestrating handlers as F6 + F3 (contract).

2. **F1 DataModelEntity has THREE sources for the same entity**: the alembic migration (database truth), the domain class (in-memory shape), the schemas (wire shape with camelCase + aliases). For sub-slice .D the per-family grammar should pick ONE as canonical -- proposed: the migration (most stable; covers server-default fields).

3. **F8 ValidationRule is fragmented.** It appears in: pydantic `EmailStr` / `HttpUrl` types (declarative); pydantic field validators (`models/common.py:9`); service-layer `check_*` functions (procedural); dependency-layer path/query constraints; route-handler conditional

checks (`profiles.py:34 -- if user.username == profile.username: raise ...`). .C should decide whether F8 covers all of these or only the declarative subset.

4. **F2 DataModelRelationship is sparse.** Only profiles repo (follow / unfollow) and articles repo (favorite / unfavorite, tag M:M) really use it. .C may decide F2 is a sub-pattern of F5 rather than a peer family.

5. **F9 ImportDependency is everywhere.** Either it is the universal background-radiation fragment family (every node has one) or it is structural metadata rather than a fragment. .C should make a call.

These are .C's first-day questions; .B does not pre-empt them.

8. The partition: every fragment to its family

Source document: `notes/fragment-mapping.md` — original title: fragment-mapping.md — FROZEN per-fragment family map

This file supersedes the "provisional Family" column in `realworld-uml/layered-architecture.md`. Frozen at sub-slice .C (see `steps/step.3000.C.txt`). The taxonomy itself (definitions, surface forms, SGDL design notes) is in `sgdl/families.md`; this file is the per-source-fragment ASSIGNMENT TABLE.

Notation:

- Paths are rooted at `source/upstream/`.
- Family codes (F1, F2, ...) reference `sgdl/families.md`.
- A row with `F4 + F8 + F10` means the fragment is a composite — its body contains instances of the listed families inlined. The PRIMARY family appears first.
- "n/a" in the Family column = framework glue with no per-fragment family. Either drives F12 alongside neighboring boot code, or is structurally trivial.

Models layer

```
+---------------------------------------+--------+--------------------------+
| Fragment                              | Family | Notes                    |
+---------------------------------------+--------+--------------------------+
| models/domain/rwmodel.py:5            | n/a    | json_encoder helper      |
| - convert_datetime_to_realworld       |        | (used by RWModel.Config) |
```

```
+--------------------------------------+--------+--------------------------+
| models/domain/rwmodel.py:9           | n/a    | alias_generator helper   |
| - convert_field_to_camel_case        |        |                          |
+--------------------------------------+--------+--------------------------+
| models/domain/rwmodel.py:15          | F1     | F1 base class for all    |
| - RWModel                            |        | domain entities (Config  |
|                                      |        | with camelCase + ISO-Z)  |
+--------------------------------------+--------+--------------------------+
| models/common.py:5                   | F8     | declarative_validator    |
| - DateTimeModelMixin + default_datetime|      | sub-variant + F1 mixin   |
+--------------------------------------+--------+--------------------------+
| models/common.py:18                  | F1     | F1 mixin (id alias)      |
| - IDModelMixin                       |        |                          |
+--------------------------------------+--------+--------------------------+
| models/domain/users.py:7             | F1     | User in-memory facet     |
| - class User                         |        |                          |
+--------------------------------------+--------+--------------------------+
| models/domain/users.py:14            | F1     | User in-memory facet     |
| - class UserInDB                     |        | (extends User with       |
|                                      |        | salt/hash + crypto       |
|                                      |        | methods)                 |
+--------------------------------------+--------+--------------------------+
| models/domain/users.py:18            | F7     | password verify method   |
| - UserInDB.check_password            |        | (delegates to F7 security)|
+--------------------------------------+--------+--------------------------+
| models/domain/users.py:21            | F7     | password set + salt gen  |
| - UserInDB.change_password           |        |                          |
+--------------------------------------+--------+--------------------------+
| models/domain/profiles.py:6          | F1     | Profile in-memory facet  |
| - class Profile                      |        | (projection-only;        |
|                                      |        | no persistence table)    |
+--------------------------------------+--------+--------------------------+
| models/domain/articles.py:7          | F1     | Article in-memory facet  |
| - class Article                      |        |                          |
+--------------------------------------+--------+--------------------------+
| models/domain/comments.py:6          | F1     | Comment in-memory facet  |
| - class Comment                      |        |                          |
+--------------------------------------+--------+--------------------------+
| models/schemas/rwschema.py:4         | F1     | F1 base class for wire   |
| - class RWSchema                     |        | facets (orm_mode=True)   |
+--------------------------------------+--------+--------------------------+
| models/schemas/users.py:7            | F1     | User wire facet:         |
| - UserInLogin                        |        | login request            |
+--------------------------------------+--------+--------------------------+
| models/schemas/users.py:12           | F1     | User wire facet:         |
```

```
| - UserInCreate                          |        | register request          |
+-----------------------------------------+--------+---------------------------+
| models/schemas/users.py:16              | F1     | User wire facet:          |
| - UserInUpdate                          |        | update request            |
+-----------------------------------------+--------+---------------------------+
| models/schemas/users.py:24              | F1     | User wire facet:          |
| - UserWithToken                         |        | response payload          |
+-----------------------------------------+--------+---------------------------+
| models/schemas/users.py:28              | F1     | User wire facet:          |
| - UserInResponse                        |        | response envelope         |
+-----------------------------------------+--------+---------------------------+
| models/schemas/profiles.py:7            | F1     | Profile wire facet        |
| - ProfileInResponse                     |        |                           |
+-----------------------------------------+--------+---------------------------+
| models/schemas/articles.py:10           | F1     | Article wire facet:       |
| - ArticleForResponse                    |        | response payload          |
+-----------------------------------------+--------+---------------------------+
| models/schemas/articles.py:14           | F1     | Article wire facet:       |
| - ArticleInResponse                     |        | response envelope         |
+-----------------------------------------+--------+---------------------------+
| models/schemas/articles.py:18           | F1     | Article wire facet:       |
| - ArticleInCreate                       |        | create request            |
+-----------------------------------------+--------+---------------------------+
| models/schemas/articles.py:25           | F1     | Article wire facet:       |
| - ArticleInUpdate                       |        | update request            |
+-----------------------------------------+--------+---------------------------+
| models/schemas/articles.py:31           | F1     | Article wire facet:       |
| - ListOfArticlesInResponse              |        | list response             |
+-----------------------------------------+--------+---------------------------+
| models/schemas/articles.py:35           | F1+F8  | F1 query-params shape +   |
| - ArticlesFilters                       |        | F8 param_constraint       |
|                                         |        | (limit ge=1, offset ge=0) |
+-----------------------------------------+--------+---------------------------+
| models/schemas/comments.py:7            | F1     | Comment wire facet:       |
| - ListOfCommentsInResponse              |        | list response             |
+-----------------------------------------+--------+---------------------------+
| models/schemas/comments.py:11           | F1     | Comment wire facet:       |
| - CommentInResponse                     |        | response envelope         |
+-----------------------------------------+--------+---------------------------+
| models/schemas/comments.py:15           | F1     | Comment wire facet:       |
| - CommentInCreate                       |        | create request            |
+-----------------------------------------+--------+---------------------------+
| models/schemas/tags.py:6                | F1     | Tag wire facet            |
| - TagsInList                            |        |                           |
+-----------------------------------------+--------+---------------------------+
```

```
| models/schemas/jwt.py:6                | F7     | JWT payload shape         |
| - JWTMeta                              |        |                           |
+----------------------------------------+--------+---------------------------+
| models/schemas/jwt.py:11               | F7     | JWT payload shape         |
| - JWTUser                              |        |                           |
+----------------------------------------+--------+---------------------------+
```

DB layer

```
+----------------------------------------+--------+---------------------------+
| Fragment                               | Family | Notes                     |
+----------------------------------------+--------+---------------------------+
| db/errors.py:1                         | F10    | EntityDoesNotExist        |
| - class EntityDoesNotExist             |        | exception                 |
+----------------------------------------+--------+---------------------------+
| db/events.py (whole file)              | F12    | asyncpg pool startup +    |
|                                        |        | shutdown handlers         |
+----------------------------------------+--------+---------------------------+
| db/queries/queries.py:3                | F12    | aiosql query loader       |
+----------------------------------------+--------+---------------------------+
| db/queries/tables.py                   | F12    | SQL table-name constants  |
+----------------------------------------+--------+---------------------------+
| db/migrations/versions/                | F1     | persistence facet of      |
| fdf8821871d7_main_tables.py:50         |        | User entity (users table  |
| - create_users_table                   |        | + trigger)                |
+----------------------------------------+--------+---------------------------+
| db/migrations/.../fdf...py:80          | F2     | followers_to_followings   |
| - create_followers_to_followings_table |        | junction table            |
+----------------------------------------+--------+---------------------------+
| db/migrations/.../fdf...py:99          | F1     | persistence facet of      |
| - create_articles_table                |        | Article entity            |
+----------------------------------------+--------+---------------------------+
| db/migrations/.../fdf...py:121         | F1     | persistence facet of      |
| - create_tags_table                    |        | Tag entity                |
+----------------------------------------+--------+---------------------------+
| db/migrations/.../fdf...py:127         | F2     | articles_to_tags          |
| - create_articles_to_tags_table        |        | junction table            |
+----------------------------------------+--------+---------------------------+
| db/migrations/.../fdf...py:145         | F2     | favorites junction table  |
| - create_favorites_table               |        |                           |
+----------------------------------------+--------+---------------------------+
| db/migrations/.../fdf...py:163         | F1     | persistence facet of      |
| - create_commentaries_table            |        | Comment entity (has own   |
|                                        |        | id; not a junction)       |
```

db/migrations/.../fdf...py:14 - create_updated_at_trigger	F10	update_updated_at trigger function (side-effect of F1 persistence facets)
db/repositories/base.py	F12	BaseRepository class (DI infra)
db/repositories/users.py:9	F5	get_user_by_email
db/repositories/users.py:18	F5	get_user_by_username
db/repositories/users.py:32	F5	create_user (calls F7 change_password inside)
db/repositories/users.py:53	F5	update_user
db/repositories/profiles.py - (per-method)	F5+F2	per-method split: profile reads = F5; follow / unfollow = F2 (junction)
db/repositories/articles.py - (per-method)	F5+F2	per-method split: article CRUD = F5; favorite / tag-management = F2
db/repositories/comments.py	F5	comment CRUD
db/repositories/tags.py	F5	tag list + upsert

Services layer

Fragment	Family	Notes
services/security.py:1-16 (whole module)	F7	bcrypt password hash + verify + salt
services/jwt.py:11	F7	create_jwt_token
services/jwt.py:23	F7	create_access_token_for_ user
services/jwt.py:32	F7	get_username_from_token

services/authentication.py:5	F8	check_username_is_taken (procedural_function)
services/authentication.py:14	F8	check_email_is_taken (procedural_function)
services/articles.py:9	F8	check_article_exists (procedural_function)
services/articles.py:18	n/a	get_slug_for_article (pure helper; library wrapper around python-slugify)
services/articles.py:22	F8	check_user_can_modify_article (procedural_function returning bool)
services/comments.py	F8	check_user_can_modify_comment

API errors layer

Fragment	Family	Notes
api/errors/http_error.py:6	F10	http_error_handler
api/errors/validation_error.py:13	F10	http422_error_handler
resources/strings.py (whole file)	F10	error string registry (~25 constants)

API dependencies layer

Fragment	Family	Notes
api/dependencies/database.py:11	F12	_get_db_pool

```
| api/dependencies/database.py:15        | F12    | _get_connection_from_pool |
+----------------------------------------+--------+---------------------------+
| api/dependencies/database.py:22        | F12    | get_repository factory    |
+----------------------------------------+--------+---------------------------+
| api/dependencies/authentication.py:21  | F7     | RWAPIKeyHeader            |
+----------------------------------------+--------+---------------------------+
| api/dependencies/authentication.py:35  | F7     | get_current_user_         |
|                                        |        | authorizer factory        |
+----------------------------------------+--------+---------------------------+
| api/dependencies/authentication.py:43  | F7     | _get_authorization_header |
| (+ F10 inlined raises)                 | +F10   |                           |
+----------------------------------------+--------+---------------------------+
| api/dependencies/authentication.py:65  | F7     | _get_authorization_header_|
|                                        |        | optional                  |
+----------------------------------------+--------+---------------------------+
| api/dependencies/authentication.py:80  | F7     | _get_current_user         |
| (+ F10 inlined raises)                 | +F10   |                           |
+----------------------------------------+--------+---------------------------+
| api/dependencies/authentication.py:106 | F7     | _get_current_user_optional|
+----------------------------------------+--------+---------------------------+
| api/dependencies/profiles.py:16        | F8     | get_profile_by_username_  |
| (+ F10 raise)                          | +F10   | from_path: F8 (Path       |
|                                        |        | param_constraint +        |
|                                        |        | EntityDoesNotExist->404)  |
+----------------------------------------+--------+---------------------------+
| api/dependencies/articles.py:20        | F8     | get_articles_filters      |
|                                        |        | (param_constraint)        |
+----------------------------------------+--------+---------------------------+
| api/dependencies/articles.py:34        | F8     | get_article_by_slug_      |
| (+ F10 raise)                          | +F10   | from_path                 |
+----------------------------------------+--------+---------------------------+
| api/dependencies/articles.py:50        | F8     | check_article_            |
| (+ F10 raise)                          | +F10   | modification_permissions  |
+----------------------------------------+--------+---------------------------+
| api/dependencies/comments.py:17        | F8     | get_comment_by_id_from_   |
| (+ F10 raise)                          | +F10   | path                      |
+----------------------------------------+--------+---------------------------+
| api/dependencies/comments.py:38        | F8     | check_comment_            |
| (+ F10 raise)                          | +F10   | modification_permissions  |
+----------------------------------------+--------+---------------------------+
```

API routes layer

Fragment	Family	Notes
api/routes/api.py (whole file)	F12	top-level router composition (mount tree)
api/routes/articles/api.py (whole file)	F12	sub-router composition for /articles
api/routes/authentication.py:21 - login	F3+F4 +F8 +F10	POST /users/login: F3 decorator + signature; F4 body; F8 password check inlined; F10 raises
api/routes/authentication.py:53 - register	F3+F4 +F8 +F10	POST /users (register): F3 decorator + sig; F4 body orchestrates 2x F8 (uniqueness) + F5 create + F7 token; F10 raises
api/routes/users.py:16 - retrieve_current_user	F3+F4	GET /user
api/routes/users.py:36 - update_current_user	F3+F4 +F8 +F10	PUT /user
api/routes/profiles.py:16 - retrieve_profile_by_username	F3+F4	GET /profiles/{username}
api/routes/profiles.py:27 - follow_for_user	F3+F4 +F8 +F10	POST /profiles/{}/follow: F8 inlined preconditions (can't follow self; already following); F2 call to add membership
api/routes/profiles.py:54 - unsubscribe_from_user	F3+F4 +F8 +F10	DELETE /profiles/{}/follow
api/routes/articles/ articles_resource.py:28 - list_articles	F3+F4	GET /articles

```
| api/routes/articles/                    | F3+F4  | POST /articles             |
| articles_resource.py:54                 | +F8    |                            |
| - create_new_article                    | +F10   |                            |
+-----------------------------------------+--------+----------------------------+
| api/routes/articles/                    | F3+F4  | GET /articles/{slug}       |
| articles_resource.py:80                 |        |                            |
| - retrieve_article_by_slug              |        |                            |
+-----------------------------------------+--------+----------------------------+
| api/routes/articles/                    | F3+F4  | PUT /articles/{slug}       |
| articles_resource.py:85                 |        |                            |
| - update_article_by_slug                |        |                            |
+-----------------------------------------+--------+----------------------------+
| api/routes/articles/                    | F3+F4  | DELETE /articles/{slug}    |
| articles_resource.py:104                |        |                            |
| - delete_article_by_slug                |        |                            |
+-----------------------------------------+--------+----------------------------+
| api/routes/articles/                    | F3+F4  | GET /articles/feed         |
| articles_common.py:21                   |        |                            |
| - get_articles_for_user_feed            |        |                            |
+-----------------------------------------+--------+----------------------------+
| api/routes/articles/                    | F3+F4  | POST /articles/{}/favorite|
| articles_common.py:48                   | +F8    | F2 add to favorites        |
| - mark_article_as_favorite              | +F10   |                            |
+-----------------------------------------+--------+----------------------------+
| api/routes/articles/                    | F3+F4  | DELETE                     |
| articles_common.py:75                   | +F8    | /articles/{}/favorite      |
| - remove_article_from_favorites         | +F10   |                            |
+-----------------------------------------+--------+----------------------------+
| api/routes/comments.py:24               | F3+F4  | GET /articles/{}/comments  |
| - list_comments_for_article             |        |                            |
+-----------------------------------------+--------+----------------------------+
| api/routes/comments.py:38               | F3+F4  | POST .../comments          |
| - create_comment_for_article            |        |                            |
+-----------------------------------------+--------+----------------------------+
| api/routes/comments.py:57               | F3+F4  | DELETE .../comments/{id}   |
| - delete_comment_from_article           |        |                            |
+-----------------------------------------+--------+----------------------------+
| api/routes/tags.py:9                    | F3+F4  | GET /tags                  |
| - get_all_tags                          |        |                            |
+-----------------------------------------+--------+----------------------------+
```

Core / entry / config layer

Fragment	Family	Notes
main.py:9 - get_application	F12	get_application factory (CORS, error handlers, startup/shutdown, router mount)
main.py:43 - app = get_application()	F12	module-level app instance
core/config.py:14	F11	environments map
core/config.py:20	F11	get_app_settings factory
core/events.py (whole file)	F12	startup / shutdown hooks
core/logging.py (whole file)	F12	InterceptHandler
core/settings/base.py	F11	AppEnvTypes enum + BaseAppSettings
core/settings/app.py:12 - class AppSettings	F11	AppSettings (12 fields + 1 derived property + configure_logging method)
core/settings/development.py	F11	per-env override
core/settings/production.py	F11	per-env override
core/settings/test.py	F11	per-env override

Summary

Family	Name	Count
F1	DataModelEntity	28
F2	DataModelRelationship (3 junction tables + per-method splits	6

```
|        |  inside profiles/articles  |         |
|        |  repos)                    |         |
+--------+----------------------------+---------+
| F3     | RouteContract              |    19   |
|        | (one per endpoint;         |         |
|        |  inlined in F3+F4 rows)    |         |
+--------+----------------------------+---------+
| F4     | RouteHandler               |    19   |
|        | (paired with F3 rows)      |         |
+--------+----------------------------+---------+
| F5     | RepositoryQuery            |   12+   |
|        | (per-method count;         |         |
|        |  profiles + articles       |         |
|        |  split with F2)            |         |
+--------+----------------------------+---------+
| F7     | AuthPolicy                 |    13   |
|        | (security.py + jwt.py +    |         |
|        |  deps/auth.py +            |         |
|        |  schemas/jwt.py +          |         |
|        |  domain/users password     |         |
|        |  methods)                  |         |
+--------+----------------------------+---------+
| F8     | ValidationRule             |    11+  |
|        | (services check_*,         |         |
|        |  deps/profiles/articles/   |         |
|        |  comments, common.py       |         |
|        |  default_datetime,         |         |
|        |  schemas type constraints) |         |
+--------+----------------------------+---------+
| F10    | ErrorHandling              |     5+  |
|        | (2 handlers + 1 exception  |         |
|        |  type + 1 trigger + 1      |         |
|        |  string registry; inlined  |         |
|        |  raises counted with F4)   |         |
+--------+----------------------------+---------+
| F11    | Configuration              |     8   |
|        | (5 settings files +        |         |
|        |  config.py factory + map)  |         |
+--------+----------------------------+---------+
| F12    | ApplicationBootstrap       |    13   |
|        | (main + core/events +      |         |
|        |  db/events + core/logging  |         |
|        |  + db/queries/queries +    |         |
```

```
|        |  db/queries/tables +       |         |
|        |  db/repositories/base +    |         |
|        |  api/deps/database +       |         |
|        |  api/routes/api +          |         |
|        |  api/routes/articles/api)  |         |
+--------+----------------------------+---------+
```

(F8 + F10 inline-counts under F4 fragments add to the totals; the actual surface-level fragment count is roughly 110, with composites contributing multiple family assignments per fragment.)

Comparison to .B provisional histogram

```
+------+-------+--------+--------------------------------+
| Fam  | .B    | .C     | Delta                          |
+------+-------+--------+--------------------------------+
| F1   |   9   |   28   | refined to per-class granular |
|      |       |        | (each domain + schema + each  |
|      |       |        | persistence facet counted)    |
+------+-------+--------+--------------------------------+
| F2   |   2   |    6   | refined to per-method (follow,|
|      |       |        | unfollow, favorite, unfav.,   |
|      |       |        | tag-add, tag-remove)          |
+------+-------+--------+--------------------------------+
| F3   |   8   |   19   | 19 routes from routes.md      |
|      |       |        | (one F3 per endpoint, NOT     |
|      |       |        | one per file)                 |
+------+-------+--------+--------------------------------+
| F4   |   7   |   19   | one per endpoint              |
+------+-------+--------+--------------------------------+
| F5   |   5   |  12+   | per-method, not per-file      |
+------+-------+--------+--------------------------------+
| F6   |   1   |    -   | DROPPED                       |
+------+-------+--------+--------------------------------+
| F7   |   3   |   13   | per-function under security + |
|      |       |        | jwt + deps/auth + the JWT     |
|      |       |        | schemas + UserInDB password   |
|      |       |        | methods                       |
+------+-------+--------+--------------------------------+
| F8   |   5   |  11+   | per-rule (counted across      |
|      |       |        | services + deps + schemas)    |
+------+-------+--------+--------------------------------+
| F9   |  ~∞   |    -   | DROPPED (DAG edges, not nodes)|
```

```
+------+-------+--------+-------------------------------+
| F10  |   3   |   5+   | added trigger fn + string     |
|      |       |        | registry; per-handler raises  |
|      |       |        | counted as composites of F4   |
+------+-------+--------+-------------------------------+
| F11  |   6   |    8   | per-class count                |
+------+-------+--------+-------------------------------+
| F12  |   -   |   13   | NEW family                    |
+------+-------+--------+-------------------------------+
```

The .C counts are higher than .B's because .C counts at **per-fragment** granularity (functions, classes, route endpoints) while .B counted at **per-file** granularity. Both views are correct at their respective levels; .B is the layered-arch overview and .C is the per-fragment ground truth.

Cross-references

- Family definitions, surface forms, SGDL design notes: `sgdl/families.md`.
- Layered architecture per-file overview (now superseded for the Family column): `realworld-uml/layered-architecture.md`.
- Entity field details: `realworld-uml/entities.md`.
- Route + DTO details: `realworld-uml/routes.md`
- `realworld-uml/contracts.md`.
- Auth-vertical walkthroughs: `realworld-uml/auth-vertical.md` + `realworld-uml/data-flow.md`.