Forms App - Comprehensive Documentation

Purpose

The `forms` app provides a full-featured dynamic form builder and workflow engine. It lets creators build forms with fields, categorize entities, orchestrate forms into linear or free processes, collect responses and answers, and expose a clean, versioned REST API with proper documentation and tests.

Architecture Overview

- **API Versioning**: All endpoints live under `api/v1/forms/`.
- **Layers**:
  - `repositories`: thin data-access abstractions over ORM queries and updates
  - `services`: business logic and transactional operations
  - `api/v1/views.py`: DRF ViewSets and actions exposing services
  - `serializers.py`: DRF serializers with validation
  - `admin.py`: Django admin customization
  - `management/commands/seed_data.py`: data seeding
  - `tests/`: API and service tests
- **Documentation**: DRF Spectacular generates OpenAPI/Swagger at `/api/docs/`.

Models (forms/models.py)

- `Form`:
  - Fields: `id (UUID)`, `title`, `description`, `created_by (User)`, `is_public`, `access_password`, `is_active`, `created_at`, `updated_at`.
  - Properties: `view_count`, `response_count`.
  - Purpose: Core container of fields; can be public or password-protected.
- `Field`:
  - Fields: `id (UUID)`, `form (FK)`, `label`, `field_type ('text'|'select'|'checkbox')`, `is_required`, `options (JSON)`, `order_num`, timestamps.
  - Validation: `select` and `checkbox` require `options.choices`.
  - Purpose: Question/attribute definition for a form.
- `Process`:
  - Fields: `id (UUID)`, `title`, `description`, `process_type ('linear'|'free')`, `is_public`, `access_password (optional)`, `is_active`, `created_by`, timestamps.
  - Validation: private processes require password.
  - Purpose: Orchestrates multiple forms into a workflow.
- `ProcessStep`:

    - Fields: `id (UUID)`, `process (FK)`, `form (FK)`, `step_name`, `step_description`, `order_num`, `is_mandatory`, timestamps.
    - Note: `is_required` was removed (redundant). Required answers are enforced per `Field.is_required`.
    - Purpose: Associates a form to a process step with ordering and mandatory gate.

- `Category` / `EntityCategory`:
    - Category owns `name`, `description`, `created_by`, timestamps.
    - EntityCategory links `entity_type ('form'|'process')`, `entity_id`, `category` with unique `(entity_type, entity_id, category)`.
    - Purpose: Flexible categorization of forms and processes.

- `Response` (aliased as `FormResponse` in code paths):
    - Fields: `id`, `form`, `submitted_by (optional)`, `ip_address`, `user_agent`, `submitted_at`.
    - Purpose: A single submission to a form.

- `Answer`:
    - Fields: `id`, `response`, `field`, `value (text)`.
    - Purpose: Value for a specific field within a response.

- `FormView`:
    - Fields: `id`, `form`, `user (nullable)`, `viewed_at`, `ip_address (optional)`, `user_agent (optional)`.
    - Purpose: View analytics per form.

Serializers (forms/serializers.py)

- Forms:
    - `FormSerializer`, `FormCreateSerializer`, `FormUpdateSerializer`, `FormListSerializer`, `PublicFormSerializer`, `PublicFormAccessSerializer`.
    - Validation for private forms: `access_password` required when `is_public = False`.

- Fields:
    - `FieldSerializer`, `FieldCreateSerializer`, `FieldUpdateSerializer`, `FieldListSerializer`, `FieldReorderSerializer`.
    - Validation ensures `options.choices` for `select`/`checkbox` and proper ordering.

- Processes / Steps:
    - `ProcessSerializer`, `ProcessCreateSerializer`, `ProcessUpdateSerializer`, `ProcessListSerializer`.
    - `ProcessStepSerializer`, `ProcessStepCreateSerializer`, `ProcessStepUpdateSerializer`, `ProcessStepListSerializer`.
    - Validation enforces ownership and proper sequencing.

- Categories:
  - `CategorySerializer`, `CategoryCreateSerializer`, `CategoryUpdateSerializer`, `CategoryListSerializer`.
  - `EntityCategorySerializer`, `EntityCategoryCreateSerializer`.
- Responses / Answers:
  - `ResponseSerializer`, `ResponseCreateSerializer` (accepts list of answers), `ResponseListSerializer`.
  - `AnswerSerializer`, `AnswerCreateSerializer`, `AnswerListSerializer`.
  - Validation: answers must include `field_id` and `value`; required fields must be answered.

Repositories (forms/repositories/repositories.py)

- Provide data-access helpers for each entity (Form, Field, Process, ProcessStep, Category, EntityCategory, Response, Answer, FormView).
- Common capabilities:
  - Get by id, list by owner, public listing, reordering using `F` expressions, calculating max order, and query helpers for categories.
- Benefit: Centralized, testable ORM queries separated from business logic.

Services (forms/services/services.py)

- Encapsulate business logic and transactions.
- Major services:
  - `FieldService`: create/update/reorder fields, enforce type/choices rules.
  - `FormService`: CRUD, validate access for private forms (password), compute analytics (views/responses), public/my listings.
  - `ProcessService`: CRUD, list types, public/my listings.
  - `ProcessStepService`: CRUD, list by process, reorder within process.
  - `CategoryService` (via repository composition): manage categories and entity associations.
  - `ResponseService`: create responses with answers atomically; validate required fields and field/form consistency; field statistics aggregation; answers by field/response.
- Transactional boundaries use `transaction.atomic()` for multi-write operations.

API Layer (forms/api/v1/views.py)

- Implemented as DRF ViewSets with `@action` for custom endpoints.
- Spectacular schema tagging via `@extend_schema_view` for accurate API docs categories.

Main ViewSets and Notable Actions

- `FormViewSet` (registered at empty prefix `''`):
  - CRUD: `list`, `retrieve`, `create`, `update`, `partial_update`, `destroy`.

    - Custom: `my_forms`, `public_forms`.
- `FieldViewSet` (`fields/`):
    - CRUD + `by_form`, `reorder`, `field_types`, `my_fields`.
- `ProcessViewSet` (`processes/`):
    - CRUD + `my_processes`, `public_processes`, `process_types`.
- `ProcessStepViewSet` (`process-steps/`):
    - CRUD + `by_process`, `my_steps`, `reorder`.
- `CategoryViewSet` (`categories/`) and `EntityCategoryViewSet` (`entity-categories/`): CRUD + ownership filters.
- `ResponseViewSet` (`responses/`):
    - CRUD + `my_responses`.
- `AnswerViewSet` (`answers/`):
    - CRUD + `by_response`, `by_field`, `field_statistics`, `my_answers`.
- `PublicFormViewSet` (`public/forms/`):
    - `list`, `retrieve`, `submit_response` (no auth; password validated for private forms).
- `PrivateFormViewSet` (`private/forms/`):
    - `validate_access` to verify password for private forms.
- `ProcessWorkflowViewSet` (`workflow/...`):
    - `get_process_steps`, `get_current_step`, `complete_step`, `get_process_progress`.
    - Linear processes block next steps until mandatory prior steps completed and required fields answered.

URLs (forms/api/v1/urls.py)

- Router registrations (order matters):
    1. `fields/`
    2. `processes/`
    3. `process-steps/`
    4. `categories/`
    5. `entity-categories/`
    6. `responses/`
    7. `answers/`
    8. `''` (forms root)
- Additional paths:

- Public forms: `public/forms/`, `public/forms/<uuid:pk>/`, `public/forms/<uuid:pk>/submit/`
- Private forms: `private/forms/validate/`
- Workflow: `workflow/process-steps/`, `workflow/current-step/`, `workflow/complete-step/`, `workflow/progress/`

Admin (forms/admin.py)

- Inlines: `FieldInline`, `ProcessStepInline` to edit related items within parent admin.
- List displays with counts and quick links using `reverse`.
- Read-only analytics fields and ordering configured.
- `ProcessStepAdmin` updated to remove deprecated `is_required`.

Management Command (forms/management/commands/seed_data.py)

- `python manage.py seed_data [--clear]` seeds:
  - Users (test1, test2, test3, admin)
  - Categories and mapped forms/processes via `EntityCategory`
  - Forms with fields (text/select/checkbox)
  - Processes with ordered steps (linear/free)
  - Responses with answers and `FormView` records
- Useful for local demos, Postman testing, and analytics endpoints.

Tests (forms/tests/ and comprehensive_api_test.py)

- Unit and integration coverage:
  - `forms/tests/test_api.py`: core API flows (forms, fields)
  - `forms/tests/test_process_api.py`: processes and steps CRUD + ordering
  - `forms/tests/test_category_api.py`: categories and entity categories
  - `forms/tests/test_response_api.py`: responses and answers creation and validation
  - `forms/tests/test_services.py`: service-layer validation and business logic
  - `forms/tests/test_new_features.py`: public/private access, workflow checks
  - `comprehensive_api_test.py`: combined live-server style checks (requests) and Django client tests with verbose reporting, covering all endpoints and error handling

---

Security & Permissions

- Default: Auth required for creator-owned resources; public endpoints are explicitly marked.
- Private forms/processes require correct password for access.
- JWT authentication via Accounts app; protected endpoints require `Authorization: Bearer <token>`.

Analytics & Reporting

- Form-level: `view_count`, `response_count` computed properties.
- Field statistics API: counts, unique values, most common values, answers timeline (`answers/field_statistics/`).
- View tracking via `FormView` with `ip_address` and `user_agent` metadata.

Workflow Semantics

- Linear processes enforce ordered completion and block skipping mandatory steps.
- Required answers are enforced per `Field.is_required` during `ResponseService` submission.
- Free processes allow completing steps in any order.

Public/Private Access

- Public forms/processes can be accessed without auth (public endpoints).
- Private forms: `access_password` required; validated on public submission/validation endpoints.

Error Handling & Validation

- Consistent `serializers.ValidationError` raised for invalid operations.
- Common 400/401/404 responses covered in tests and visible in docs.
- Schema tags ensure endpoints are correctly categorized (e.g., Form Answers, Process Steps, Processes).

Example Flows

- Create a form → add fields → share public link or require password → collect responses → analyze field statistics.
- Build a linear process → add mandatory steps with forms → enforce step completion order → track progress per respondent.

API Documentation

- OpenAPI schema at `/api/schema/` with Swagger UI at `/api/docs/` and ReDoc at `/api/redoc/`.
- Schema customizations with `@extend_schema` and `@extend_schema_view` for summaries, descriptions, and tags.

Maintenance Notes

- Keep router registrations ordered: specific routes (e.g., `processes`) before the forms root (`''`).
- When adding new actions, tag them with Spectacular so they appear under the correct group in docs.

- Prefer services for business logic; keep views thin.
- Use repositories for complex or reused ORM operations.
- Update tests and seeders alongside schema or behavior changes.

Deep Dive: Business Logic and Invariants

Forms

- Creation: only authenticated users can create; `created_by` is set to the requester.
- Visibility:
  - Public: accessible via public list/detail endpoints without auth.
  - Private: requires `access_password` on creation/update; respondents must provide it to view/submit.
- Analytics:
  - `FormView` row inserted on public retrieval endpoints (with `ip_address`, `user_agent` if available).
  - `response_count` derived from `Response` rows per form.
- Invariants:
  - Fields belong to exactly one form; `order_num` is 1..N without gaps after reorder operations.
  - Deleting a form cascades to fields, responses, answers, and views (via FK on_delete).

Fields

- Types supported: `text`, `select`, `checkbox`.
- Validation:
  - `select` and `checkbox` require `options.choices` as a non-empty list of strings.
  - `is_required=True` means a response must provide non-empty `value`.
  - Reorder uses atomic updates and `F` expressions to avoid race conditions.
- Answer semantics:
  - `checkbox` answers store a single selected value per answer row (simple model). Multiple choices can be captured with multiple answers or comma-separated strings depending on UI; statistics aggregate over values.

Processes & Steps

- Types: `linear` or `free`.
- Linear:
  - Current step is the lowest `order_num` not yet completed.
  - Cannot advance if any prior step marked `is_mandatory=True` is incomplete.
- Free:
  - Steps can be completed in any order; progress reflects completed count.
- Step completion: submitting a `Response` for the step's `form`. Required field validation occurs at submission time via `ResponseService`.
- Invariants:
  - A step's `form.created_by` must match the `process.created_by` (ownership constraint).
  - `order_num` is unique per process and normalized on reorder.

Categories

- One `Category` may be applied to many entities via `EntityCategory`.
- `EntityCategory` unique index ensures no duplicate mapping for the same `(entity_type, entity_id, category)`.
- Ownership: users can only manage categories they created; entity assignments require ownership of the entity or admin privileges.

Responses & Answers

- Submission is transactional: creating a `Response` and all `Answer` rows is wrapped in `transaction.atomic()`.
- Validation matrix (simplified):
  - Field exists and belongs to the target form.
  - All `is_required` fields present in payload with non-empty values.
  - For `select`/`checkbox`, provided `value` must be in `options.choices`.
- Security: IP and User-Agent captured for audit/analytics; user association optional for anonymous submissions (public forms).

Repositories: Important Methods (Selected)

Note: Names reflect patterns used in `forms/repositories/repositories.py`.

- FormRepository
  - `list_by_user(user_id)` → QuerySet of forms owned by user
  - `list_public()` → public forms
  - `max_order_for_form(form_id)` → support ordering logic (used indirectly by fields)
- FieldRepository
  - `get_form_fields(form_id)` → fields ordered by `order_num`
  - `reorder(field_id, new_order)` → adjusts others via `F('order_num') ± 1`
- ProcessRepository
  - `list_by_user(user_id)`, `list_public()`, etc.
- ProcessStepRepository
  - `list_by_process(process_id)` (ordered)
  - `reorder(step_id, new_order)` (gap-free, atomic)
- Category/EntityCategoryRepository
  - `list_user_categories(user_id)`
  - `list_entity_categories(entity_type, ids)`
- Response/AnswerRepository

    - `list_by_form(form_id)`, `list_by_user(user_id)`
    - `answers_by_field(field_id)`
    - Aggregations for statistics (counts, distincts, date truncations)

- FormViewRepository
    - `track_view(form_id, ip, ua, user)` → inserts row
    - `count_by_form(form_id)`

Services: Responsibilities and Key Flows

FieldService

- `create_field(user, form_id, data)`
    - Ensures ownership, assigns `order_num` if omitted (max+1), validates type/options.
- `update_field(user, field_id, data)`
    - Prevents cross-form move unless allowed, re-validates constraints.
- `reorder_field(user, field_id, new_order)`
    - Delegates to repository; raises on invalid range.

FormService

- `create_form(user, data)`, `update_form(user, form_id, data)`, `delete_form`.
- Access control helpers:
    - `validate_access(form, password)`
    - `get_public_forms()`, `get_user_forms(user)`
- Analytics helpers: compute view/response counts.

ProcessService

- `create_process`, `update_process`, `delete_process`.
- `get_public_processes()`, `get_user_processes(user)`.
- `list_types()` → returns configured process types.

ProcessStepService

- `create_process_step`, `update_process_step`, `delete_process_step`.
- `get_process_steps(user, process_id)` → ownership enforced.
- `reorder_step(user, step_id, new_order)`

ResponseService

- `create_response(user, form_id, answers_data, ip, ua)`
    - Validates required fields, field types, and value membership.
    - Creates `Response` then bulk-creates `Answer` rows atomically.
- `get_field_statistics(user, field_id)`
    - Returns counts, distincts, most-common, per-day series using `TruncDate`.
- `get_field_answers(user, field_id)`, `get_response_answers(user, response_id)`

ViewSets and Route-to-Action Map (Key)

Forms (`''`)

- `GET /api/v1/forms/` → `FormViewSet.list`
- `POST /api/v1/forms/` → create
- `GET /api/v1/forms/{id}/` → retrieve
- `PATCH /api/v1/forms/{id}/` → partial_update
- `DELETE /api/v1/forms/{id}/` → destroy
- `GET /api/v1/forms/my_forms/` → current user's forms
- `GET /api/v1/forms/public_forms/` → public forms

Fields (`fields/`)

- `GET /fields/` list, `POST /fields/` create
- `GET /fields/{id}/`, `PATCH /fields/{id}/`, `DELETE /fields/{id}/`
- `GET /fields/by_form/?form_id=`
- `POST /fields/{id}/reorder/` with `{"new_order": N}`
- `GET /fields/field_types/`
- `GET /fields/my_fields/`

Processes (`processes/`) and Steps (`process-steps/`)

- Processes: CRUD + `my_processes`, `public_processes`, `process_types`
- Steps: CRUD + `by_process?process_id=`, `my_steps`, `reorder {id}`

Categories (`categories/`, `entity-categories/`)

- CRUD; lists scoped to owner; entity links enforced by ownership.

Responses/Answers (`responses/`, `answers/`)

- Responses: CRUD + `my_responses`
- Answers: CRUD + `by_response?response_id=`, `by_field?field_id=`, `field_statistics?field_id=`

Public/Private Forms

- Public: `/public/forms/` list, `/{id}/`, `/{id}/submit/`
- Private: `/private/forms/validate/` {form_id, password}

Workflow

- `/workflow/process-steps/?process_id=` → list
- `/workflow/current-step/?process_id=` → linear current
- `/workflow/complete-step/` → payload {process_id, step_id, answers}
- `/workflow/progress/?process_id=` → completion summary

Error Catalogue (Representative)

- 400 Bad Request
  - Missing query params (e.g., `field_id`, `process_id`)
  - Invalid `new_order` type/range
  - Private resource without password or wrong password
  - Required field missing in submission
- 401 Unauthorized
  - Missing/invalid JWT on protected endpoints
- 403 Forbidden
  - Attempt to access resources not owned by user (when enforced)
- 404 Not Found
  - Non-existent IDs, or filtered out by ownership

Error responses use a consistent shape `{ "detail": "message" }` or DRF serializer error dicts.

End-to-End Examples (Condensed)

A) Public Survey

1. Creator builds a public `Form` and adds `select` and `text` fields.
2. Respondent opens `/public/forms/{form_id}/` — a `FormView` is recorded.
3. Respondent submits answers → `ResponseService.create_response()` validates and writes.
4. Creator views `/answers/field_statistics/?field_id=` to see analytics.

B) Linear Onboarding Process

1. Creator builds `Process (linear)` with steps pointing to existing forms.
2. Respondent requests `/workflow/current-step/?process_id=` → gets first mandatory step.
3. Respondent completes step by POSTing `/workflow/complete-step/` with form answers.
4. On success, current step advances; progress endpoint reflects status.

Security Notes

- JWT authentication required for creator-managed endpoints; public endpoints are explicitly unauthenticated.
- Passwords for private forms/processes validated server-side; not stored in plain text in clients.
- Minimal PII: only IP/User-Agent collected for analytics; user association optional.

Extensibility Guide

- Adding a new field type:
  1. Extend `Field.field_type` choices and validation in `FieldService`/serializers.
  2. Update response validation in `ResponseService` and statistics logic if needed.
  3. Add tests and schema examples.
- Adding a new workflow rule:

    1. Extend `ProcessService`/`ProcessWorkflowViewSet` logic.
    2. Consider a strategy object per `process_type` for cleaner separation.
- Adding new analytics:
    1. Create a service method that aggregates with ORM annotations.
    2. Expose through a read-only action with `@extend_schema` tagging.

Glossary

- Form: container of fields/questions.
- Field: question definition with type/validation/options.
- Response: a submission instance for a form.
- Answer: a value for a single field within a response.
- Process: an orchestration of forms into steps.
- Step: a link between a process and a form, with ordering and mandatory flag.
- Category: a label to group forms/processes via `EntityCategory`.
- Public/Private: access model controlled by `is_public` and optional password.