Отчёт по проекту «Система управления дефектами на строительных объектах» (Лазарев)

1. Анализ

1.1 Бизнес-цели

- Полный цикл работы с дефектами: от регистрации, назначения и исполнения — до контроля статусов и управленческой отчётности.
- Снижение потери информации и повышение прозрачности взаимодействия **инженеров**, **менеджеров**, **руководителей/заказчика**.
- Сокращение сроков устранения и соблюдение SLA на строительных объектах.

1.2 Роли и разграничение доступа

- **Инженер** — регистрирует/редактирует дефекты, добавляет фото и комментарии, изменяет статусы в рамках своего пула.
- **Менеджер** — назначает исполнителей и сроки, управляет проектами, отслеживает соблюдение SLA, формирует отчёты.
- **Руководитель/заказчик (наблюдатель)** — читает сводную аналитику и статусные отчёты, без прав изменений.

1.3 Предметная область и жизненный цикл дефекта

- **Статусы:** *Новая → В работе → На проверке → Закрыта/Отменена* (жёсткий workflow с аудитом переходов).
- **Состав дефекта:** заголовок, описание, приоритет/серьёзность, исполнитель, срок, вложения, история изменений, комментарии.

1.4 Функциональные требования (FR) — приоритизация MoSCoW

№	Требование	Детали реализации	Приоритет	Трассировка
FR-1	Регистрация пользователей и аутентификация	JWT (access/refresh), восстановление пароля (one-time token), audit login	Must	Users/Auth; юнит-тесты токенов
FR-2	Роли и права	RBAC: инженер/менеджер/наблюдатель; middleware в Gin	Must	Политики доступа; интеграционные тесты ролей
FR-3	Управление проектами/объектами	CRUD проектов, этапов; привязка дефектов	Must	Projects; CRUD-тесты
FR-4	Управление дефектами	CRUD, статусы, приоритеты, сроки, назначение исполнителя, теги/участки	Must	Defects; тесты workflow
FR-5	Вложения	Фото/документы, лимиты размера, S3-совместимое хранилище/локальный store	Should	Интеграционные тесты загрузки
FR-6	Комментарии и история	Комментарии; полная история изменений (audit trail)	Should	Тесты на аудит и сортировку

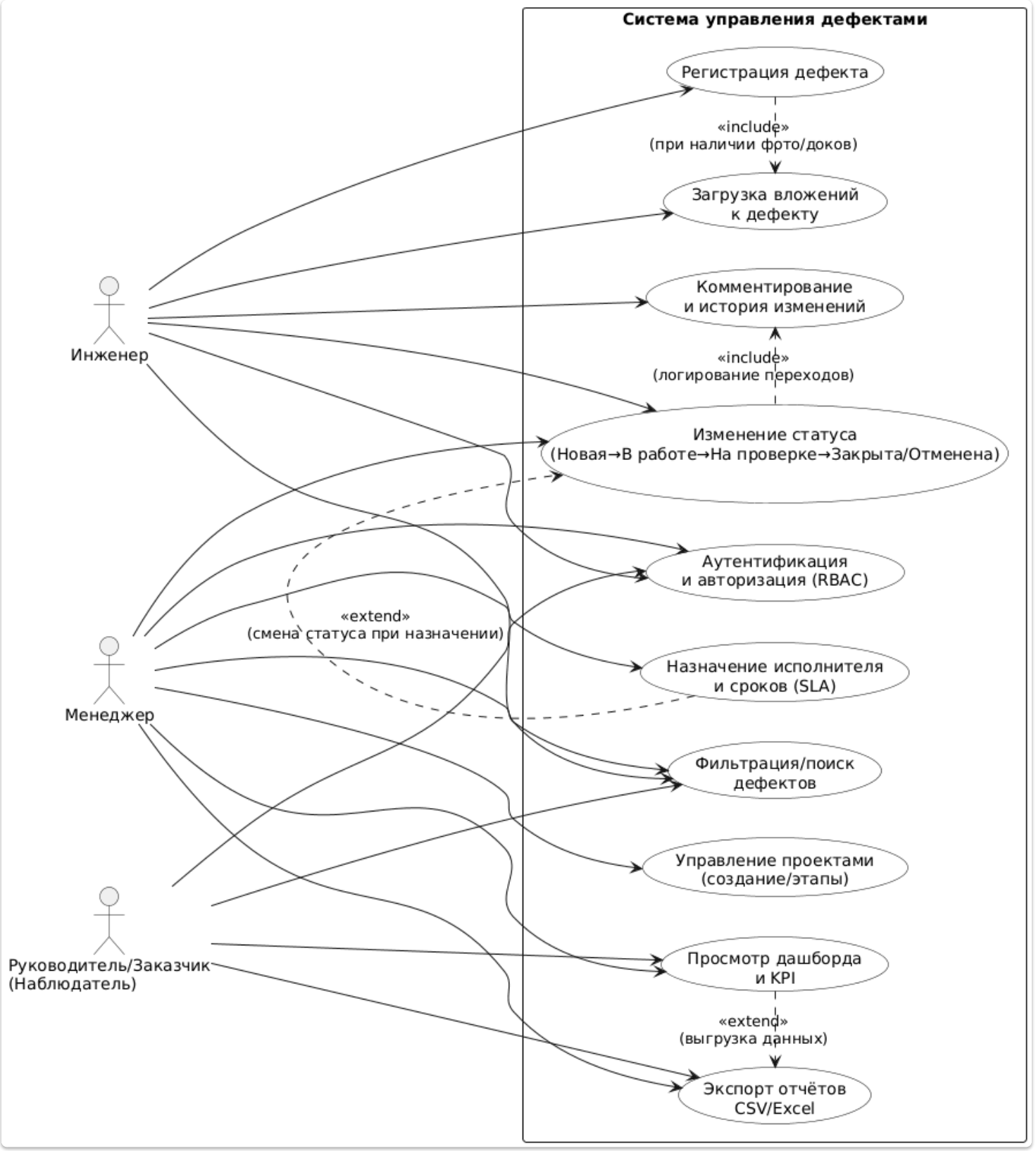
№	Требование	Детали реализации	Приоритет	Трассировка
FR-7	Поиск/фильтры/ сортировка	По статусу, приоритету, проекту, срокам; пагинация	Must	Нагрузочные тесты (≤1 сек)
FR-8	Отчётность CSV/Excel	Экспорт выборок; агрегации	Could	Интеграционные тесты экспорта
FR-9	Аналитические панели	KPI, графики статусов/сроков	Could	Визуальные и API-тесты

1.5 Нефункциональные требования (NFR)

NFR	Требование	Цель и контроль
Производительность	Отклик ≤ 1 сек при 50 активных пользователях	JMeter/profiling, кэш select-ов, индексы БД
Безопасность	bcrypt/argon2 для паролей, защита от SQLi/XSS/CSRF	OWASP чек-лист, middleware, input validation
Надёжность	Ежедневные бэкапы БД + проверка восстановления	Cron/pg_dump, тест восстановления
UX/i18n	Русский интерфейс, адаптив (ПК/ планшет)	UI-гайд, e2e-проверки
Совместимость	Chrome/Firefox/Edge (актуальные)	Smoke по браузерам

1.6 Use Cases (ключевые)

- **UC-1 Регистрация дефекта (инженер)**: выбрать проект → заполнить форму → приложить фото → «Сохранить» → статус *Новая*.
- **UC-2 Назначение и контроль сроков (менеджер)**: фильтр «просроченные/рисковые» → назначить исполнителя/срок → уведомления.
- **UC-3 Изменение статуса (инженер/менеджер)**: *Новая → В работе → На проверке → Закрыта/ Отменена* с логом переходов.
- **UC-4 Просмотр прогресса/отчёт (наблюдатель/менеджер)**: дашборд KPI, экспорт CSV/Excel.



1.7 Риски и допущения

- Смена требований на поздних этапах — снижать через согласование SRS и трассировку «требование → задача → тест» .
- Риски производительности — индексация, кэш, профилирование запросов.
- Конфликты ролей — строгая RBAC-матрица и интеграционные тесты ролей.

---

2. Проектирование

2.1 Целевая архитектура (монолит)

- UI: Single Page App (любой современный фреймворк; аналитика/графики).

- **API**: Go 1.22+ / Gin (REST), валидация, RBAC middleware, обработка ошибок.
- **БД**: PostgreSQL (рекомендация: 14+), миграции (golang-migrate), транзакции, индексы.
- **Хранилище файлов**: локальный диск или S3-совместимое (MinIO) — абстраговано интерфейсом.
- **Аутентификация**: JWT (HS256/RS256), refresh-токены, blacklist (optional) или короткий TTL + ротирование.
- **Логи и аудит**: JSON-логи, корреляция по request-id; таблица audit_log.
- **Конфигурация**: 12-factor (ENV), встроенный конфиг-пакет.

Слои монолита:

- `transport/http` (Gin handlers, DTO, auth checks)
- `service` (бизнес-правила, workflow статусов)
- `repo` (доступ к БД: `gorm`)
- `domain` (модели, ошибки, константы)
- `pkg` (utils: jwt, hashing, валидация, storage)

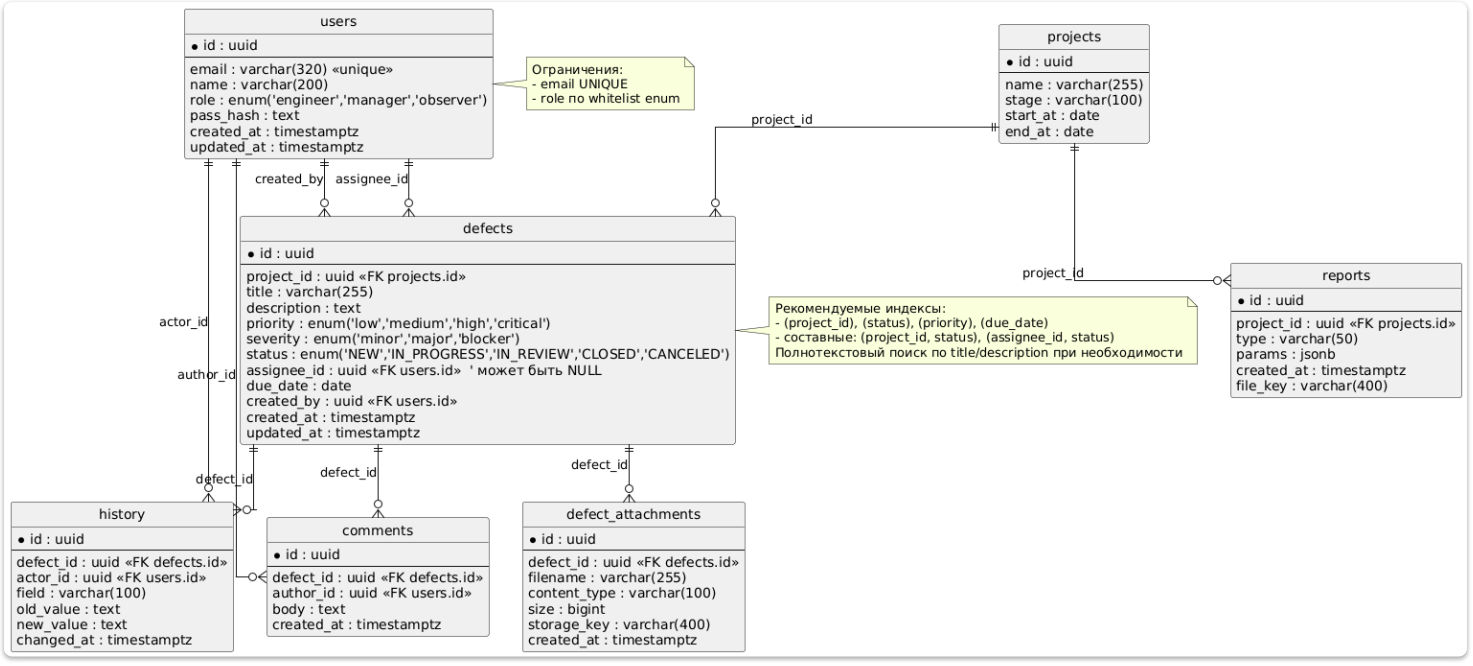
2.2 ER-модель (ключевые сущности)

- **users**: id, email, name, role, pass_hash, created_at, updated_at
- **projects**: id, name, stage, start_at, end_at
- **defects**: id, project_id(FK), title, description, priority, severity, status, assignee_id(FK users), due_date, created_by, created_at, updated_at
- **defect_attachments**: id, defect_id, filename, content_type, size, storage_key, created_at
- **comments**: id, defect_id, author_id, body, created_at
- **history**: id, defect_id, actor_id, field, old_value, new_value, changed_at
- **reports** (опционально): id, project_id, type, params(json), created_at, file_key

Индексация:

- `defects(project_id)`, `defects(status)`, `defects(priority)`, `defects(due_date)`; комбинированные на частые фильтры.
- Полнотекстовый индекс для `title/description` при потребности поиска.

ER-диаграмма:



2.3 REST API (выдержка)

- `POST /api/v1/auth/login` — логин → JWT.

- `GET /api/v1/projects` / `POST /api/v1/projects` — список/создание (роль: менеджер).
- `GET /api/v1/defects?projectId&status&priority&assignee&dueFrom&dueTo&page&size` — поиск/фильтры/сортировка.
- `POST /api/v1/defects` — создать дефект (инженер/менеджер).
- `PATCH /api/v1/defects/:id/status` — переход статусов (RBAC + валидация workflow).
- `POST /api/v1/defects/:id/attachments` — загрузка файла.
- `POST /api/v1/defects/:id/comments` — комментарий.
- `GET /api/v1/reports/export?projectId&type=csv|xlsx` — экспорт.

2.4 Workflow статусов (валидация в сервисе)

Разрешённые переходы:
`NEW → IN_PROGRESS → IN_REVIEW → CLOSED` и альтернативный `ANY → CANCELED` (только менеджер, с комментарием-основанием).
При каждом переходе — запись в `history` + валидация роли.

2.5 Нефункциональные решения

- **Производительность**: пул соединений `pgx`, N+1-контроль, кэш частых справочников (in-memory), пагинация, лимиты выборок.
- **Безопасность**: bcrypt (`cost=12` по умолчанию), строгая валидация DTO, sanitization, ограничение типов контента и размера файлов, CSRF не актуален для чистого API при корректном CORS/JWT.
- **Логи/мониторинг**: correlation id, уровень WARN для бизнес-ошибок, алерты по 5xx и времени ответа >1с.
- **Миграции**: `golang-migrate up/down` при деплое.

3. Разработка (под Go / Gin)

3.1 Технический стек

- **Язык/рантайм:** Go 1.22+
- **Web-фреймворк:** Gin Gonic
- **ORM/драйвер:** GORM
- **Миграции:** golang-migrate
- **Auth:** JWT (`github.com/golang-jwt/jwt/v5`), `bcrypt` (`golang.org/x/crypto/bcrypt`)
- **Валидация:** `github.com/go-playground/validator/v10`
- **Логи:** `zerolog` / `zap` (JSON)
- **Файлы:** локальный `nfs` или MinIO SDK (feature-flag)
- **Тестирование:** `testing`, `testify`, интеграционные с `dockertest` / `testcontainers-go`
- **CI:** GitHub Actions (test, build, миграции)

3.2 Структура репозитория (монолит)

```
/cmd/api/main.go
/internal/domain/...            // модели, константы, ошибки
/internal/transport/http/...    // Gin handlers, DTO, middleware (authz, logging)
/internal/service/...           // бизнес-логика, workflow статусов
/internal/repo/...              // PostgreSQL (pgx/sqlc или GORM), транзакции
/internal/pkg/auth/...          // jwt manager, password hasher
/internal/pkg/storage/...       // файловое хранилище (fs/S3)
internal/migrations/...         // sql миграции
/config/...                     // конфиги (sample .env), OpenAPI
```

3.3 Примеры контрактов (DTO)

```
POST /api/v1/defects
{
  "projectId": "e3b0c4...",
  "title": "Трещина в панели",
  "description": "Обнаружена трещина на 3 этаже, секция Б",
  "priority": "HIGH",
  "severity": "MAJOR",
  "assigneeId": "u_123",
  "dueDate": "2025-10-10"
}
```

Ответ:

```
{ "id":"d_789", "status":"NEW", "createdAt":"2025-09-25T10:10:00Z" }
```

3.4 Бизнес-правила

- Создание дефекта — только **инженер/менеджер**; обязательны `projectId`, `title`, `priority`; `dueDate`.
- Переход `IN_REVIEW → CLOSED` — доступен менеджеру или автору, если настроено правило «самоприёмка» = false → запрещено.
- Отмена (`CANCELED`) — только менеджер с причиной (комментарий обязателен).
- Любое изменение — запись в `history` (actor, field, old→new, timestamp).

3.5 Тестирование (на уровне разработки)

- **Юнит-тесты**: сервисы (workflow, валидации), JWT/пароли, репозитории (с моками).
- **Интеграционные**: цепочка `create → status changes → comment → export`.
- **Нагрузочные**: поиск/фильтры по дефектам при 50 rps, SLO p95 ≤ 1 сек.