ИТ Школа CRM

Сопроводительная документация

- README.md
- docs/current-functionality.md
- docs/development.md
- docs/architecture.md

Содержание

ИТ Школа CRM 4

Технологический стек . . . 4
Быстрый старт . . . 4
Проверки . . . 5
Документация . . . 5

Текущий функционал CRM 6

Что делает система . . . 6
Роли . . . 6
Главная страница и реестр процессов . . . 6
Запуск и исполнение workflow . . . 7
Формы этапа . . . 7
Документы . . . 7
Ручные действия этапа . . . 8
Конструктор workflow . . . 8
Каталоги и карточки . . . 8
Интеграции . . . 8
Отчётность и аудит . . . 9
Референсный workflow . . . 9
Справка и API . . . 9

Руководство разработчика 10

Предварительные требования . . . 10
Быстрый старт . . . 10
Локальный запуск кода . . . 11
База данных . . . 11
Конфигурация и секреты . . . 12
Проверки . . . 12
Демонстрационный workflow . . . 13

Архитектура 14

Общая схема . . . 14
Технологический стек . . . 14
Границы backend . . . 14
Данные и доступ . . . 15

Workflow 15
Формы и структура БД 15
Файлы 16
Интеграции и действия этапа 16
Отчёты и аудит 16
Frontend 17
Ошибки и конкурентные изменения 17
Правила изменений 17

README.md

ИТ Школа CRM

CRM для настройки и исполнения процессов взаимодействия ИТ Школы с вузами.

Текущая поставка содержит один редактируемый workflow «Взаимодействие с вузом» из 14 этапов. Менеджер проходит назначенные линейные этапы, работает с предметными данными, комментариями, документами и ручными действиями интеграции. Руководитель назначает КАМов и ответственных в доступных вузах и согласует настроенные этапы; администратор имеет те же полномочия глобально и управляет каталогами, доступами, шаблонами и интеграциями.

Реализованы:

- реестр процессов с фильтрами, списком и канбаном;
- предметные каталоги, связи и карточки вузов, направлений, продуктов и контактов;
- XLS/XLSX-импорт каталогов с предпросмотром и атомарным применением;
- data-first конструктор линейных workflow и неизменяемые опубликованные версии;
- три типа блоков в формах этапов: корневые записи (Entity Card), связанные записи по проверенным связям внешних ключей (Entity List) и значения контекста, расширяющие модель данных (Process Field);
- контроль времени прохождения этапов (SLA);
- общий черновик активного этапа, блокирование (optimistic locking) при изменениях, комментарии и аудит;
- типизированные документы этапов в S3 хранилище с поддержкой версионирования;
- ручные действия импорта, экспорта и получение данных из внешних информационных систем по API;
- регистрация интеграций и импорт схем OpenAPI 3.0/3.1;
- историческая отчётность, диаграммы и выгрузки XLS/XLSX/PDF/JSON/PNG;
- авторизация через Keycloak OIDC/JWT и роли user, manager, admin;
- единый production-контейнер FastAPI + React SPA без Nginx и CORS.

Полное описание фактически доступного поведения: docs/current-functionality.md.

Технологический стек

Backend: Python 3.12, FastAPI, psycopg, СУБД: PostgreSQL. Frontend: React, TypeScript, Vite.

Полное окружение: Docker Compose, Keycloak и S3-хранилище (Silo: открытый форк MinIO).

Быстрый старт

```
cp .env.example .env
docker compose up --build
```

Откройте http://localhost:8000 и войдите как admin / admin. Эти данные предназначены только для локальной демонстрации.

На чистой базе один референсный workflow создаётся из schema.sql. Для синхронизации его управляемого определения через публичный API используйте команду подготовки демонстрационного workflow.

Проверки

```
cd backend
.venv/bin/python -m pytest -q

cd ../frontend
npm test
npm run build
```

Документация

- Текущий функционал
- Разработка и запуск
- Архитектура

docs/current-functionality.md

Текущий функционал CRM

Документ описывает возможности текущей версии, доступные в пользовательском интерфейсе и публичном API.

Что делает система

CRM ведёт процессы взаимодействия ИТ Школы с вузами. Пользователь запускает опубликованный workflow для выбранного вуза, проходит его этапы, работает с предметными данными и документами. Руководитель контролирует процессы своей области, администратор настраивает каталоги, workflow и интеграции.

В поставку входит редактируемый референсный workflow «Взаимодействие с вузом» из 14 последовательных этапов.

Роли

Роль	Возможности
Менеджер (user)	Работает с назначенными процессами и данными доступных вузов, запускает workflow, заполняет этапы, загружает документы и выполняет ручные действия.
Руководитель (manager)	Видит процессы доступных вузов, назначает КАМов, меняет ответственного и согласует этапы.
Администратор (admin)	Видит все данные, управляет каталогами, доступом, workflow, интеграциями и аудитом.

Keycloak хранит учётные записи и базовые роли. CRM хранит бизнес-права, назначения и область видимости. Создавать и удалять пользователей через CRM нельзя.

Когда администратор предоставляе пользователю роли user доступ к вузу, CRM одновременно назначает его КАМом этого вуза. Состав КАМов и снятие назначения настраиваются в разделе Каталоги.

Главная страница и реестр процессов

Главная страница показывает данные в пределах прав пользователя:

- менеджеру — показатели своей работы и активные процессы;
- руководителю — сводку по доступным вузам;
- администратору — сводку по всей системе.

Реестр процессов предназначен для оперативной работы. В нём есть фильтры по workflow, версии, вузу, направлению, продукту и дате запуска. Руководитель и администратор также могут фильтровать по ответственному.

Доступны два представления:

- таблица с серверной пагинацией;
- канбан для одной опубликованной версии workflow.

Перетаскивание карточки в канбане вызывает обычное завершение текущего этапа. Если этап требует заполнить форму, загрузить документ или пройти согласование, система открывает страницу процесса. Пропустить несколько этапов через канбан нельзя.

Фильтры, страница и режим отображения сохраняются в URL.

Запуск и исполнение workflow

На странице запуска процессов пользователь выбирает опубликованный шаблон и доступную корневую запись. Менеджер назначает процесс себе; руководитель и администратор могут выбрать КАМа соответствующего вуза.

Запущенный процесс фиксирует версию workflow. Новая публикация шаблона на него не влияет.

Страница процесса содержит три раздела:

- **Основное** — этапы, форма и действия;
- **Обсуждение** — общая лента комментариев;
- **Файлы** — документы, сгруппированные по этапам.

Редактировать активный этап может только назначенный КАМ. Руководитель и администратор могут переназначить процесс или принять решение по согласованию, но не редактируют форму за исполнителя.

Черновик активного этапа общий для процесса и сохраняется отдельно от завершения. При переназначении он переходит новому ответственному. Optimistic locking защищает его от конфликтов между вкладками и от одновременных изменений.

Если этап требует согласования, руководитель может подтвердить завершение или отправить на доработку.

Назначенный КАМ может переименовать активный этап только в текущем процессе. Название опубликованного шаблона и группировка отчётов не меняются.

Формы этапа

Поддерживаются три блока:

- **Entity Card** — карточка корневой предметной записи;
- **Entity List** — связанные по внешним ключам записи с разрешёнными операциями чтения, создания, изменения и выбора;
- **Process Field** — значение в контексте процесса: текст, число, дата, логическое значение, выбор или пользователь.

Список связанных сущностей (Entity List) использует компактную таблицу, серверные поиск, сортировку и пагинацию. Настроенный одиночный выбор требует ровно одну запись, множественный — не менее одной. Выбранные ID сохраняются в контексте процесса(Workflow) и могут ограничивать другой список того же или последующего этапа. Завершённый этап показывает только выбранные записи.

Физического удаления предметных записей через форму нет. Backend повторно проверяет права, типы, обязательные поля и достижимость связанных записей при сохранении и завершении.

Документы

Администратор задаёт для этапа тип документа, обязательность, количество, допустимые расширения и MIME-типы, максимальный размер и возможность замены.

Загрузка и замена доступны только в активном этапе. Обязательные документы проверяются при завершении. Замена создаёт новую версию, а предыдущая остаётся в истории.

Файлы хранятся в S3-хранилище, метаданные — в БД. Скачивание идёт по короткоживущей ссылке после проверки доступа. Встроенной антивирусной проверки нет.

Ручные действия этапа

Этап может содержать ручной импорт, экспорт или чтение внешнего API.

Импорт принимает XLSX, JSON или ответ зарегистрированной OpenAPI-операции. Сначала пользователь выполняет предварительный просмотр в формате текущее → новое, затем явно выбирает поля для применения. Изменения выполняются одной транзакцией.

Экспорт формирует XLSX или JSON из доступных записей. Результат сохраняется в хранилище S3 либо отправляется зарегистрированной API-операции.

Read-only API вызывает зарегистрированную операцию и сохраняет нормализованный ответ в журнале запусков, не изменяя предметные таблицы.

Журнал запусков и сохранённые заявки/зачисления видны только в том экземпляре workflow, в котором были получены. Один и тот же обучающийся переиспользуется между процессами без копирования.

В текущей версии действия не запускаются по расписанию и не выполняются автоматически при переходе этапа.

Конструктор workflow

Администратор может:

- создать шаблон и выбрать корневую таблицу;
- добавить, удалить или переставить этапы;
- настроить форму, документы и ручные действия;
- сохранить черновик и опубликовать неизменяемую версию.

Конструктор строит только линейную последовательность. Таблицы, колонки и сваязи внешних ключей читаются из СУБД и ограничиваются белым списком из таблицы workflow_allowed_tables.

Каталоги и карточки

В системе есть каталоги вузов, ИТ-направлений, продуктов, вендоров, контактов вузов и пользователей Keycloak. Пользователи доступны только руководителю и администратору и только для чтения.

Каталоги используют серверные пагинацию, фильтрацию и сортировку. Администратор может создавать и изменять записи, а также импортировать XLS/XLSX после обязательного предпросмотра. Импорт применяется атомарно.

Интеграции

Администратор регистрирует интеграцию, указывает базовый URL, способ авторизации и ссылку на секрет, затем импортирует схему OpenAPI 3.0/3.1 в формате JSON или YAML. Значение секрета в CRM не хранится.

Внешний API можно вызвать только из настроенного ручного действия. Production принимает только HTTPS-ссылки и разрешённые хосты; перенаправления запрещены, время и размер ответа ограничены.

Локальный стенд содержит синтетические CMS и LMS с данными по направлениям демонстрационных вузов. Реальные контракты и доступы заказчика в поставку не входят.

Отчётность и аудит

Отчётность строит исторический срез на выбранный момент. Фильтры, KPI, диаграммы, проблемные процессы, таблица и выгрузки используют один набор данных и одну область доступа.

Полный срез выгружается в XLS, XLSX, PDF или JSON; диаграммы — в PNG или PDF.

Администратор может просматривать и выгружать журнал аудита. В нём фиксируются значимые изменения, работа с персональными данными, файлами и интеграциями, а также входы в систему, выходы и ошибки входа Keycloak. Персональные значения, токены и содержимое документов в записи журнала аудита не копируются.

Референсный workflow

Поставка содержит редактируемый workflow «Взаимодействие с вузом» из 14 этапов:

1. Актуализация ответственных лиц.
2. Коммуникация и актуальность программ.
3. Организация встречи.
4. Обмен документами.
5. Корректировка документов.
6. Подписание документов.
7. Передача материалов и лицензии.
8. Сопровождение внедрения.
9. Обучение преподавателей.
10. Актуализация учебной программы.
11. Ведение занятий.
12. Актуализация документации.
13. Повышение квалификации.
14. Контроль исполнения.

На различных этапах показаны разные функциональные возможности управления данными:

- обновление данных в БД CRM
- загрузка файлов в хранилище S3
- импорт данных из CMS и LMS (mock-данные)

Справка и API

Раздел Справка содержит руководства для менеджера, руководителя и администратора. Доступ к типу справки зависит от роли.

Также доступноописание методов API CRM через Swagger UI (/docs). Ошибки возвращают стабильный код и request_id.

docs/development.md

Руководство разработчика

В этом разделе собраны команды для запуска и проверки проекта.

Предварительные требования

В системе должны быть установлены:

- Docker с Compose plugin - для запуска всего стенда;
- Python 3.12 - для разработки backend;
- Node.js 24 и npm - для разработки frontend.

Быстрый старт

Из корня репозитория:

```
cp .env.example .env
docker compose up --build
```

После запуска:

Сервис	Адрес
CRM	http://localhost:8000
Swagger UI	http://localhost:8000/docs
Keycloak	http://localhost:8081
MinIO Console	http://localhost:9001
CMS mock	http://localhost:8082/docs
LMS mock	http://localhost:8083/docs

Учётные записи локального стенда:

Роль	Логин	Пароль
Администратор	admin	admin
Руководитель	manager	manager
Менеджер	user	user

Эти пароли предназначены только для локальной разработки.

Управление стендом

```
# Запустить в фоне или пересобрать после изменений
docker compose up --build -d

# Проверить состояние контейнеров
docker compose ps

# Смотреть логи приложения
docker compose logs -f app

# Остановить стенд, сохранив данные
docker compose down
```

Чтобы удалить локальные данные PostgreSQL и S3-хранилища и начать с чистого стенда:

```
docker compose down -v
```

Эта команда удаляет Docker volumes без возможности восстановления.

Порты можно изменить в .env: APP_PORT, POSTGRES_PORT, KEYCLOAK_PORT, MINIO_API_PORT, MINIO_CONSOLE_PORT, CMS_MOCK_PORT и LMS_MOCK_PORT.

Локальный запуск кода

Для разработки backend и frontend достаточно поднять PostgreSQL и S3-хранилище:

```
docker compose up -d postgres minio
```

Backend

```
cd backend
python3.12 -m venv .venv
.venv/bin/pip install -r requirements.txt
.venv/bin/uvicorn src.main:app --reload
```

В .env.example для такого запуска установлено AUTH_DISABLED=true: backend использует тестового администратора без Keycloak. Чтобы выполнить запрос от другого тестового пользователя, передайте заголовок X-Dev-User со значением external_subject из backend/schema.sql.

Для проверки интеграций также запустите заглушки:

```
docker compose up -d cms-mock lms-mock
```

В разделе «Интеграции» укажите http://localhost:8082 для CMS и http://localhost:8083 для LMS.

Frontend

```
cd frontend
npm ci
npm run dev
```

Vite откроет приложение на http://localhost:5173 и будет проксировать /api на http://localhost:8000.

База данных

backend/schema.sql — единственный актуальный снимок схемы. Миграций в проекте нет: файл применяется только при создании нового PostgreSQL volume.

После несовместимого изменения схемы:

1. Обновите schema.sql, backend, frontend, тесты и документацию в одной задаче.
2. Пересоздайте локальную базу командой docker compose down -v.

3. Поднимите стенд заново и проверьте изменение на пустой базе.

Таблицы, доступные конструктору workflow, перечислены в workflow_allowed_tables. Названия таблиц и полей для интерфейса задаются через PostgreSQL comments:

```
COMMENT ON TABLE universities IS 'Вузы';
COMMENT ON COLUMN universities.full_name IS 'Полное название';
```

Конфигурация и секреты

Локальные значения находятся в .env.example. Пароли, токены, API keys и строки подключения для рабочих окружений в репозиторий не коммитятся.

Основные параметры:

Переменная	Назначение
DATABASE_URL	Подключение backend к PostgreSQL
REPORT_TIMEZONE	Часовой пояс отчётов; по умолчанию Europe/Moscow
ENTITY_ALLOWED_SCHEMAS	Схемы PostgreSQL, доступные конструктору
OPENAPI_ALLOWED_HOSTS	Разрешённые хосты внешних API
INTEGRATION_TIMEOUT_SECONDS	Тайм-аут внешнего запроса
INTEGRATION_MAX_RESPONSE_MB	Максимальный размер ответа интеграции

Интеграция хранит только имя переменной secret_reference. Сам секрет передаётся через окружение или Docker secret. В production внешние интеграции работают только по HTTPS.

Проверки

Backend:

```
cd backend
.venv/bin/python -m pytest -q
```

Frontend:

```
cd frontend
npm test
npm run build
```

Запускайте полный набор перед слиянием изменений. Во время разработки можно передать pytest путь к нужному файлу или имя теста, а Vitest — путь после npm test --.

Сквозной smoke-тест требует запущенного Compose-стенда:

```
cd backend
CRM_SMOKE_BASE_URL=http://localhost:8000 \
CRM_SMOKE_KEYCLOAK_URL=http://localhost:8081 \
.venv/bin/python -m pytest -q tests/test_smoke_api.py
```

Без этих двух переменных smoke-тест пропускается.

При изменении публичного API проверьте Swagger UI и обновите backend/src/api/documentation.py. У прикладных endpoint должны оставаться актуальные описания прав, побочных эффектов и ошибок.

Демонстрационный workflow

На чистой базе schema.sql создаёт workflow «Взаимодействие с вузом» из 14 этапов. Чтобы обновить его определение через публичный API локального стенда:

```
cd backend
CRM_DEMO_BASE_URL=http://localhost:8000 \
CRM_DEMO_KEYCLOAK_URL=http://localhost:8081 \
.venv/bin/python scripts/bootstrap_spec_demo.py
```

Скрипт предназначен только для локального окружения. Повторный запуск без изменений не создаёт новую версию и не меняет уже запущенные процессы.

docs/architecture.md

Архитектура

Данный раздел описывает устройство системы и правила, которые нужно соблюдать при изменении кода.

Общая схема

```
Browser
 |
 v
FastAPI ──────────> PostgreSQL
 |              |
 |              └─ предметные данные, workflow, права и аудит
 ├──────────────────> Keycloak
 |              └─ аутентификация и базовые роли
 ├──────────────────> MinIO
 |              └─ документы и файлы импорта/экспорта
 └──────────────────> External APIs
                └─ зарегистрированные OpenAPI-операции
```

Технологический стек

Backend: Python 3.12, FastAPI, psycopg, СУБД: PostgreSQL. Frontend: React, TypeScript, Vite.

Полное окружение: Docker Compose, Keycloak и S3-хранилище (Silo: открытый форк MinIO).

Границы backend

Направление вызовов:

```
HTTP route -> feature handler -> repository / external adapter
```

HTTP-слой

Код в backend/src/api отвечает только за транспорт:

- маршрутизацию и HTTP-схемы;
- зависимости FastAPI;
- преобразование известных ошибок в стабильный API-ответ.

В маршрутах не должно быть бизнес-правил и SQL. Все API-маршруты находятся в ветке /api; SPA fallback регистрируется последним и не перехватывает их.

Особенности реализации

Бизнес-сценарии находятся в папках backend/src/features/<feature>. Каждый обработчик (handler) получает явную request-модель, проверяет бизнес-правила, управляет транзакцией и возвращает явную response-модель.

Репозиторий (Repository) содержит SQL и преобразование строк. Он использует переданное соединение и не вызывает подтверждений транзакций (commit): один обработчки может атомарно работать с несколькими репозиториями.

Абстракции вводятся только на реальных границах — например, для объектного хранилища или внешнего API. Универсальный уровень репозитория и пользовательские SQL запросы в системе не нужны.

Данные и доступ

Единственный снимок схемы — backend/schema.sql; миграций нет. Несовместимое изменение выполняется одновременно в схеме, коде, тестах и документации и проверяется на новой базе.

Основные предметные каталоги: вузы, ИТ-направления, вендоры, продукты и контакты вузов. Пользователи получаются из Keycloak и доступны в CRM только для чтения.

Предметный контекст нормализован:

```
University
└── UniversityDirection
    └── UniversityDirectionProduct
```

Keycloak подтверждает пользователя и роль user, manager или admin. Права на конкретные вузы, назначения КАМов и область видимости хранит CRM. Каждый обработчик повторно проверяет доступ; скрытие кнопки во frontend не считается защитой.

Workflow

Шаблон, опубликованная версия и запущенный экземпляр — это разные сущности.

- Черновик шаблона можно менять.
- Опубликованная версия неизменяема.
- Запущенный экземпляр остаётся на той версии, с которой был создан.
- Текущий конструктор создаёт только линейный процесс с одним активным этапом.

Шаблон задаёт корневую таблицу. Экземпляр хранит ID корневой записи и вычисленные ссылки на вуз, направление и продукт. Эти ссылки используются для проверки прав доступа, назначения ответственного и отчётности.

Сохранение черновика и завершение этапа — разные операции, которые защищены блокировками (optimistic locking); при завершении повторно проверяются поля, выбранные записи и обязательные документы.

Этап может включать настройку обязательности подтверждения руководителем. После отправки руководитель может подтвердить и завершить этап или вернуть его на доработку с обязательным комментарием.

Комментарии образуют общую ленту процесса. История завершённых этапов не переписывается при смене ответственного или публикации новой версии.

Формы и структура БД

Форма этапа хранится в JSONB как упорядоченный список form_blocks. Поддерживаются три блока:

- entity_card — корневая предметная запись;
- entity_list — связанные записи по FK-пути;
- process_field — значение в контексте процесса.

Backend компилирует блоки в runtime-схему и остаётся единственным источником правил валидации. Frontend только отображает форму и может заранее подсказать ожидаемую ошибку.

Список сущностей читается страницами с сервера; поиск и сортировка применяются до пагинации. Настроенный выбор обязателен при завершении этапа: режим single принимает ровно один ID, multiple — непустой набор. Связь selected_ids_binding переносит ID в Workflow Context, а пара scope_binding + scope_relation_path ограничивает другой список ранее выбранным набором. Backend проверяет тип

context-поля, происхождение набора, префикс связи по внешним ключам и принадлежность каждой выбранной записи области процесса. Для завершённого этапа запрос списка дополнительно ограничивается сохранёнными ID самого блока.

Конструктор видит только таблицы из белого списка (таблица workflow_allowed_tables). Колонки, типы, ограничения и связи он читает из системных каталогов БД. Связь доступна, только если обе таблицы входят в белый список. В workflow сохраняются полные имена таблиц, колонок и ограничений внешнего ключа.

Человекочитаемые названия берутся из метаданных таблиц (COMMENT ON TABLE и COMMENT ON COLUMN). Комментарии влияют только на подписи; типы, права и связи определяет схема.

Любая связь внешнего ключа повторно проверяется перед чтением или изменением данных. Имена SQL-объектов собираются через psycopg.sql, значения всегда параметризуются. Физического удаления записей через workflow нет.

Файлы

S3-хранилище хранит файлы, БД — метаданные и связи. Один файл имеет атрибуты: документ этапа, источник импорта или результат экспорта.

При настройке этапа можно указать тип, обязательность, количество, размер и разрешённые форматы загружаемых файлов. Документ не является полем формы и не меняет контекст процесса (workflow).

Доступ к файлу проверяется через связанный workflow и вуз. Backend выдаёт короткоживущую ссылку на конкретную версию объекта; публичных постоянных URL нет. Замена создаёт новую версию и сохраняет историю.

Интеграции и действия этапа

Администратор регистрирует интеграцию и импортирует неизменяемую версию OpenAPI-схемы. Workflow ссылается на параметр методов API (operationId).

Значение секрета получается по названию параметра из настроек окружения только во время вызова. Хост должен входить в список допустимых (OPENAPI_ALLOWED_HOSTS); перенаправления запрещены; запрос ограничен по времени и размеру ответа. В production разрешён только HTTPS.

Импорт предполагает предварительныцй просмотр. Перед ручным действием система сериализованно сохраняет общий черновик этапа. Для параметров, хранящихся в контексте процесса, обработчик строит эффективный контекст: сохранённый контекст экземпляра, затем значения текущего этапа, затем его черновик. Поэтому выбор, сделанный на активном этапе, доступен действию до завершения этапа. Применение повторно проверяет исходные значения и одной транзакцией меняет только выбранные пользователем поля.

CMS/LMS-профили эталонного workflow передают выбранные параметры university_directions.id как повторяющийся query-параметр direction_ids и отбрасывают строки по программам вне выбранного набора.

Отчёты и аудит

Экран, экспорт и главная страница строятся из одного среза данных. Исторический срез восстанавливает состояние процесса на фиксированный момент и поддерживается экспорт в форматы XLSX, PDF, PNG и JSON.

Значимые изменения, обращения к персональным данным, операции с файлами и интеграциями фиксируются в журнале аудита. Данные содержат техническую информацию, но не копии персональных данных, токенов или содержимого документов. События входа, выхода и ошибки входа поступают из Keycloak.

Структурированные персональные данные проходят через сервис CryptoMediator на границе взаимодействия с БД. Текущий PassthroughCryptoMediator — точка расширения, а не готовое крипторешение. Подключение реального шифрования потребует отдельного решения о ключах, алгоритмах, поиске, индексах и формате данных.

Frontend

Frontend организован по функциональным возможностям:

- pages собирает веб-страницы и формы;
- features реализует пользовательский сценарий и работу с API;
- shared содержит только переиспользуемые примитивы.

При разработке использован Atomaro Gen2, если требуемого компонента нет в библиотеке, разрабатывается собственный компонент. Цвета, типографика и отступы для него берутся из css-токенов проекта. Общие элементы управления находятся в frontend/src/shared/Kit.tsx, семантические токены — в frontend/src/styles.css.

Состояния загрузки, пустого результата, ошибки и недоступного действия проектируются явно. Фильтры, вкладки и режимы представления, которые должны переживать обновление страницы, хранятся в URL.

Frontend не повторяет серверные бизнес-правила. Любое изменение workflow, прав, файлов или предметных данных выполняется через backend-обработчики.

Ошибки и конкурентные изменения

Ожидаемые ошибки возвращаются в стабильном envelope с code, безопасным сообщением, деталями полей и request_id:

- 409 — конфликт версии, unique/exclusion или изменение схемы;
- 422 — неверный тип, FK, NOT NULL, CHECK или бизнес-валидация;
- 500 — непредвиденная ошибка без внутренних деталей в ответе.

Полный traceback остаётся в серверном журнале. Изменяемые сущности защищаются версией записи или другой явной блокировкой (optimistic concurrency).

Правила изменений

- Разрабатывается только актуальный контракт. Переходные endpoint, aliases, адаптеры, двойное чтение и двойная запись запрещены.
- Бизнес-правило реализуется один раз — в handler, а не одновременно в API, UI и repository.
- Публичный API остаётся описан в Swagger/OpenAPI.
- Архитектурное или поведенческое изменение сопровождается обновлением тестов и документации.