

# Отчёт по домашним работам Р01–Р05

## Р01 — Работа с секретами

- Приложение больше не хранит токен в коде: ожидаем APP\_API\_TOKEN из окружения и сравниваем его с X-API-Key без утечек (app/main.py:38).
- Если ключ не передан или неверный, клиент сразу получает понятный ответ 401 с шаблоном RFC 7807 (tests/test\_items.py:91).
- При отсутствии переменной окружения возвращаем аккуратную ошибку с correlation id, поэтому проблему легко найти (app/main.py:52, app/errors.py:39).
- Все проверки ruff/black/isort/pytest крутятся в CI и не дадут сломать стиль или тесты (.github/workflows/ci.yml:30).
- Тот же механизм токена используется во всех изменяющих запросах, а тесты автоматически подставляют ключ через фикстуру (app/main.py:112, tests/conftest.py:21).

## Р02 — Проверка входных данных

- Валидация теперь режет управляющие символы и угловые скобки, чтобы в задачи нельзя было подложить скрипты (app/schemas.py:26).
- Pytest-проверка убеждается, что <script> в названии даёт 422 с понятным описанием (tests/test\_items.py:103).
- Все ошибки валидации уходят в общий обработчик и приходят клиенту в одном формате с correlation id (app/errors.py:92, tests/test\_items.py:41).
- Lint, форматтеры и тесты всё так же запускаются в CI.
- Эти правила работают и при создании задач, и при частичном обновлении (app/main.py:112, app/main.py:136).

## Р03 — Единый формат ошибок

- Любое исключение превращается в ответ RFC 7807 с заголовком и полем X-Correlation-ID, поэтому внутренностей приложения наружу не видно (app/errors.py:39, app/errors.py:67).
- Тесты проверяют, что 404 и 422 выглядят как задумано и содержат все поля (tests/test\_errors.py:12, tests/test\_errors.py:21).
- Перед отправкой мы вычищаем поле input, чтобы в ответах не появлялись сырье пользовательские данные (app/errors.py:26, tests/test\_errors.py:33).
- CI продолжает следить за качеством без изменений.
- Обработчики подключены в самом приложении и покрывают все основные типы исключений (app/main.py:21).

## Р04 — Безопасные загрузки файлов

- Функция secure\_save проверяет размер, сигнатуру, симлинки и пути, поэтому файлы нельзя подложить в чужую папку или засунуть мусор (app/upload.py:25).
- Тесты пробуют все неприятные случаи: слишком большой файл, подделка формата и симлинки (tests/test\_upload.py:6, tests/test\_upload.py:33).
- Функция возвращает только статус и код причины, не раскрывая путь к файлу в логах или ответах (app/upload.py:28, app/upload.py:63).
- Контроль качества опять же на стороне CI.
- Модуль уже лежит в приложении и готов для подключения эндпоинта загрузки (app/upload.py:1).

## Р05 — Хранение данных в базе

- Вместо временного словаря теперь полноценная база через SQLAlchemy, так что данные не теряются и гонок нет (app/main.py:105, app/models.py:8).
- Тесты гоняются поверх настоящей таблицы; фикстура сбрасывает состояние до и после проверки (tests/test\_items.py:21, tests/conftest.py:12).
- Ошибочные запросы (например, пустой PATCH) по-прежнему возвращают стандартный problem-details (app/main.py:147, tests/test\_items.py:67).

- Настройка CI и правила качества остались прежними.
- Добавлены модуль `app/db.py` и модель `Item`, а все CRUD-операции используют сессии (`app/db.py:32`, `app/main.py:94`).