ЧАСТЬ 6 · ИИ В РАБОТЕ

ИТ-Коктель с Анной Фединой

Одна функция. Рабочий код. Проверенный итог.

Как довести изменение от сценария до приёмки на примере реестра



Опыт · Шаблоны · Практика / Сентябрь 2026

СЛОВО АВТОРА

Быстро собрать - только начало

Рабочая версия появилась за пару недель. Дальше важнее стало точно понимать, что именно работает.



Анна Федина
CIO · Автор канала

За разработку MVP мне называли 10 млн рублей. С ИИ я получила рабочее решение за пару недель. Это моя история и внешняя оценка, а не расчёт чистой экономии или обещание такого срока для любого проекта.

В реальном проекте регистрации баллонов есть реестр, генерация меток и чтение. В этом выпуске я беру одну операцию: перевод учебной записи в активное состояние. Так можно показать весь цикл, не пересказывать приложение целиком.

Что было до кода

В выпуске 2 я описала пользователя и критерии приёмки. В выпуске 3 - сущности, поля и допустимые значения. Здесь эти решения превращаются в изменение, которое можно проверить отдельно от камеры, интерфейса и интеграций.

> Учебная задача: объект DEMO-001 существует в состоянии NEW. Разрешено сделать его ACTIVE. Повтор операции со старым состоянием должен вернуть конфликт. Неизвестный идентификатор не должен выглядеть как успешное изменение.

Это упрощённый контракт, а не описание всех производственных правил. Учебная SQLite-база находится в памяти; рабочая база в упражнениях не используется.

> **Результат: работающая операция, примеры отказов, инструкция воспроизведения и честный список того, чего тест ещё не доказывает.**

ВЫПУСК 6 · ШАГ 1

Минимальный срез проходит через систему

Кнопка, сервис и база должны договориться об одном результате.



У операции есть три входа: ID объекта, ожидаемое состояние и новое состояние. Ожидаемое состояние позволяет заметить, что другой шаг уже изменил запись. Это полезнее, чем молча ещё раз записать ACTIVE.

> Учебный контракт
> Вход: id=1, before=NEW, after=ACTIVE.
> Успех: изменена ровно одна запись; после чтения status=ACTIVE.
> Отказ: переход не разрешён либо запись уже не в состоянии NEW.
> Побочный эффект: никаких новых меток и сообщений.

Промпт: подготовить изменение

> В учебной модели есть cylinder(id,status) и функция transition. Нужно разрешить NEW→ACTIVE и ACTIVE→WITHDRAWN. Найди минимальное место изменения. До правки перечисли вход, выход, ошибки и затронутые тесты. Не меняй схему, генератор меток и интерфейс без необходимости. Рабочую базу не открывай.

Ожидаемый результат - небольшой план с файлами и проверками. Если агент предлагает переписать весь сервис, я возвращаю его к контракту одной операции. Большой объём кода не является признаком завершения.

Сохранить исходное состояние

Перед правкой смотрю изменения в рабочей папке и определяю свою область. Если там уже есть незавершённая работа, её нельзя стереть командой «вернуть всё». Исправление должно быть отделимо и объяснимо.

ВЫПУСК 6 · ШАГ 2

Обновление, которое замечает конфликт

Ниже фрагмент учебной реализации из практики к выпуску 3. Он проверяет состояние при записи.

```
with db:
    cur = db.execute(
        'UPDATE cylinder SET status=? '
        'WHERE id=? AND status=?',
        (after, cylinder_id, before))
    if cur.rowcount != 1:
        raise ValueError('STATE_CONFLICT_OR_NOT_FOUND')
```

Перед этим фрагментом функция проверяет, что пара before/after входит в список разрешённых переходов. Параметры передаются отдельно от SQL. Условие status=? не позволяет выдать повтор со старым состоянием за новое успешное действие.

Зачем здесь транзакция

В используемой учебной конфигурации sqlite3 блок with db завершает транзакцию или откатывает её при исключении. Он не создаёт отдельный сервер и не закрывает соединение. Для других настроек автокоммита поведение нужно проверить отдельно.

> Проверка вручную
> 1. Создать DEMO-001 со статусом NEW.
> 2. Вызвать transition(id, NEW, ACTIVE).
> 3. Прочитать запись: ACTIVE.
> 4. Повторить тот же вызов: получить конфликт.
> 5. Убедиться, что статус остался ACTIVE.

Этот фрагмент не решает все конкурентные сценарии приложения. Он показывает конкретную защиту от устаревшего ожидаемого состояния. Авторизация и правила доступа проверяются на другом уровне.

> **«Ни одной строки не изменено» - отдельный исход. Его нельзя превращать в зелёную галочку только потому, что SQL не упал.**

Python: управление транзакциями sqlite3

ВЫПУСК 6 · ШАГ 3

Я проверяю успех и отказ

Проверка должна подтверждать правило, а не повторять строку реализации.

T1: NEW→ACTIVE - состояние изменено.
T2: повтор NEW→ACTIVE - конфликт, данные прежние.
T3: ACTIVE→WITHDRAWN - переход разрешён.
T4: WITHDRAWN→ACTIVE - отказ по правилу.
T5: неизвестный ID - отказ, чужие записи не затронуты.
T6: ошибка второй записи в транзакции - первая откатилась.

В последнем сценарии моделируется операция из нескольких записей: создать объект и связанную запись. Если вторая нарушает ограничение, проверяю число объектов после отката. Это доказывает атомарность именно этой учебной транзакции.

Промпт: проверить без самообмана

Для transition составь независимые входы T1–T5 выше и проверь итоговое состояние базы. Для транзакции из двух INSERT добавь ошибку второй записи и проверь отсутствие первой. Запиши команды и результаты. Если тест не запускался, не ставь PASS. Не считай проверку SQLite испытанием рабочего сервера.

В пакете к выпуску 3 есть demo_model.py и schema.sql: python3 demo_model.py запускает 22 проверки. Из них 20 проверяют сценарии и две демонстрируют известные границы: прямой SQL может обойти календарную проверку и правило переходов приложения.

Сильный отрицательный результат

Если тест показал обход, я не скрываю его ради красивого отчёта. Нужно либо закрыть обход на нужном уровне, либо ограничить способ записи и честно описать границу. Для учебного примера выбрано второе.

Приёмочный отчёт объясняет, что проверено и что осталось за пределами, а не только число зелёных строк.

ВЫПУСК 6 · ШАГ 4

Сбой - это вход для следующей итерации

Полезный отчёт об ошибке позволяет другому человеку повторить её без догадок.

> Учебная иллюстрация дефекта
> Не утверждение о текущей версии.
> Вход: повтор NEW→ACTIVE для ID=1.
> Ожидание: конфликт.
> Факт: интерфейс показывает успех при нуле обновлённых строк.
> Граница: сервисный сценарий; камера не участвует.
> Проверка исправления: тот же вход + чтение статуса.

Промпт: исправить причину

> Сначала проверь, есть ли описанный дефект в текущей учебной версии. Если rowcount уже проверяется, подтверди защиту тестом: повторная правка не нужна. Иначе воспроизведи дефект на учебной базе. Найди место, где ноль изменённых строк трактуется как успех. Предложи минимальное исправление и тест, который падает до него. После правки запусти этот тест и соседние сценарии переходов. Не переписывай несвязанные модули и не меняй ожидаемое поведение ради прохождения теста.

Первый удачный запуск после правки ещё не объясняет причину. Я хочу видеть связь: конкретный вход воспроизводил дефект, изменение устранило его, соседние правила сохранились.

Как вернуть изменение

Откат кода и откат данных - разные операции. Для учебной базы достаточно нового запуска. В живом проекте нужно отдельно понимать миграции, совместимость версий и резервную копию. Универсальная команда отката без контекста здесь была бы опасным советом.

Если другая работа уже изменила тот же файл, агент должен сохранить её и показать конфликт. Вернуть весь каталог к старой версии ради одной функции - потерять чужой результат.

> **В карточку проекта попадает причина исправления, проверенная версия и ограничение. Полный лог остаётся рядом с тестом.**

ВЫПУСК 6 · ШАГ 5

Передать функцию, а не переписку

Рабочий результат должен быть понятен следующему человеку без вашего присутствия.

Пакет передачи

> Что изменено: правило перехода и обнаружение конфликта.
> Как запустить: python3 demo_model.py в папке учебной практики.
> Вход: DEMO-001, NEW→ACTIVE.
> Ожидаемый итог: ACTIVE; повтор отклонён.
> Доказательство: отчёт проверок с датой и версией.
> Граница: SQLite в памяти; без камеры, авторизации и нагрузки.

В этой серии готовый учебный набор начинается в выпуске 3. В комплекте отдельно приложены demo_model.py и schema.sql: сохраните их рядом в одной папке. Это самостоятельный учебный набор, а не рабочая база приложения.

Промпт: приёмка изменения

> Прими функцию по T1–T6 из этого выпуска. Проверь, что инструкция воспроизводится с чистого учебного состояния. Сопоставь отчёт с текущими файлами. Верни: принято / не принято, конкретные причины, ограничения и один следующий шаг. Не принимай результат по фразе исполнителя «всё готово».

Самостоятельное задание: добавьте разрешённый переход только после изменения требований. Сначала запишите, зачем он нужен, затем обновите набор примеров. Сравните, какие старые тесты должны сохраниться и какие теперь описывают отменённое правило.

> **Готово, когда операция повторяется, неверные входы дают ожидаемый отказ, а другой человек может запустить проверку. В выпуске 7 усложняю среду: файл, камера и физическая метка.**

Рабочий MVP - начало проверяемого развития. Срок первой версии не отменяет последующие испытания и поддержку.

ПРОДОЛЖИМ РАЗГОВОР

ИТ-Коктель с Анной Фединой

Про ИТ, управление, людей и жизнь вокруг работы.
Выберите площадку, на которой вам удобно.

Telegram
Заметки, обсуждения и новые материалы
Открыть >

ВК Видео
Выпуски и короткие фрагменты
Открыть >

Дзен
Тексты и видео
Открыть >

RuTube
Разговоры и выпуски подкаста
Открыть >

Сетка
Профессиональные темы и общение
Открыть >

Яндекс Музыка
ИТ-коктейль в аудиоформате
Открыть >

Названия площадок и карточки кликабельны.

