ВЫСШАЯ ШКОЛА ЭКОНОМИКИ · ФКН

FCS Task

бэкенд, инфраструктура и бизнес-логика учебной платформы

Групповой исследовательский проект · 2026

Босякова Яна Сергеевна, БПМИ 2413

Яндекс

Зачем вообще нужна была эта работа

В отчёте много backend-деталей, но по сути задача была простой: сделать учебную платформу управляемой и предсказуемой.

Курсы не должны «жить сами по себе»: у каждого курса есть владелец, участники, права, статусы и правила видимости.

Студенту важно видеть понятную картину: задания, дедлайны, баллы и свой прогресс — без лишней технической магии.

Администратору нужна уверенность: если действие запрещено, backend действительно не даст обойти ограничение.

Карта работы по отчёту

Если смотреть на документ как на карту проекта, больше всего внимания ушло на инфраструктуру, бизнес-логику и контроль качества.

Приложение развивалось не только “вширь”, но и в сторону правил, доступа и стабильности.

Путь запроса: где живёт логика

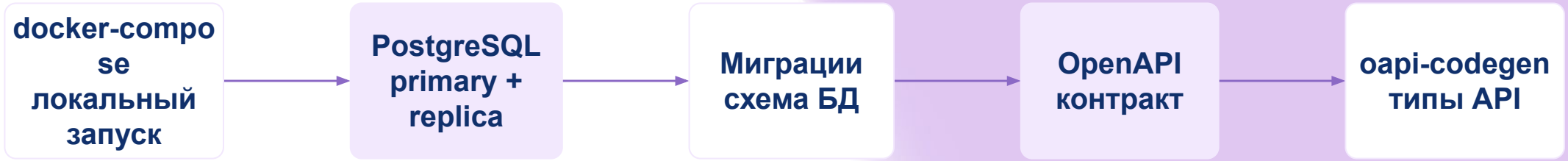
Главный принцип — не смешивать уровни. HTTP-слой принимает запрос, service принимает решения, repository работает с данными.

В service-слое проверяются роли, membership, статусы курса и запреты на действия. Поэтому правила не размазаны по ручкам.

Repository не решает продуктовые вопросы — он возвращает данные по понятным правилам выборки.

Инфраструктура: чтобы команда запускалась одинаково

Часть работы была не про новые фичи, а про то, чтобы разработка перестала зависеть от “у меня локально как-то иначе”.



Итог: меньше ручной настройки, меньше случайных расхождений и понятнее вход в проект для команды.

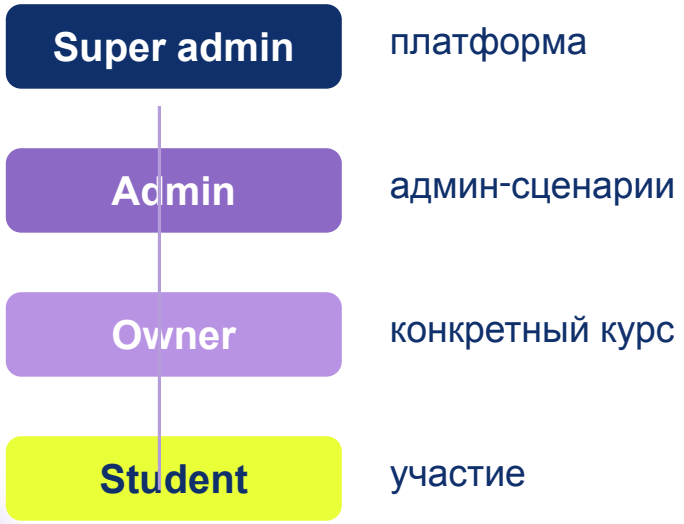
OpenAPI стал “договором” между frontend и backend: если меняется контракт, это видно сразу.

Права доступа: не только роль, но и правила

Пользователь может быть студентом в одном курсе и владельцем в другом. Поэтому права разделены на глобальные и course-scoped.

Отдельно выделены права вроде просмотра hidden-курса. Это важно: “видеть” и “управлять” — разные действия.

Так backend может отвечать не только на вопрос “кто ты?”, но и “что именно ты можешь сделать в этом курсе?”.



Hidden и finished: два статуса — две разные логики

Оба статуса ограничивают поведение, но делают это по-разному. Это хороший пример, где backend должен понимать продуктовый смысл.

hidden

Курс скрывается из общих списков, но остаётся доступен владельцу и администраторам.

главная мысль: не показывать лишним людям

finished

Курс можно просматривать, но нельзя менять участие, баллы и учебные результаты.

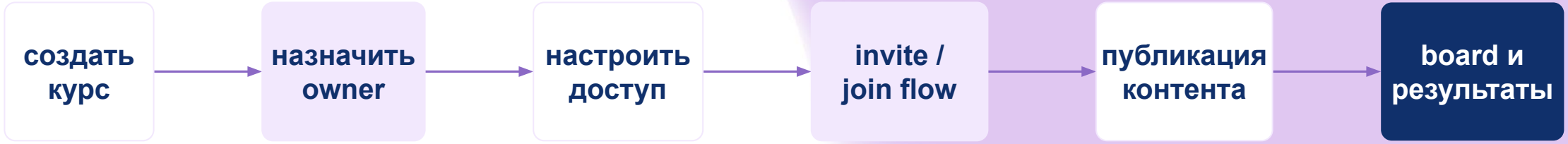
главная мысль: оставить историю, закрыть изменения

Разница кажется небольшой, но если её не заложить в сервисах, появятся обходные пути и странное поведение API.

business flow

Жизненный цикл курса: от создания до участия

Курс — это не только запись в таблице. Вокруг него сразу появляются владелец, участники, приглашения, задачи и правила доступа.



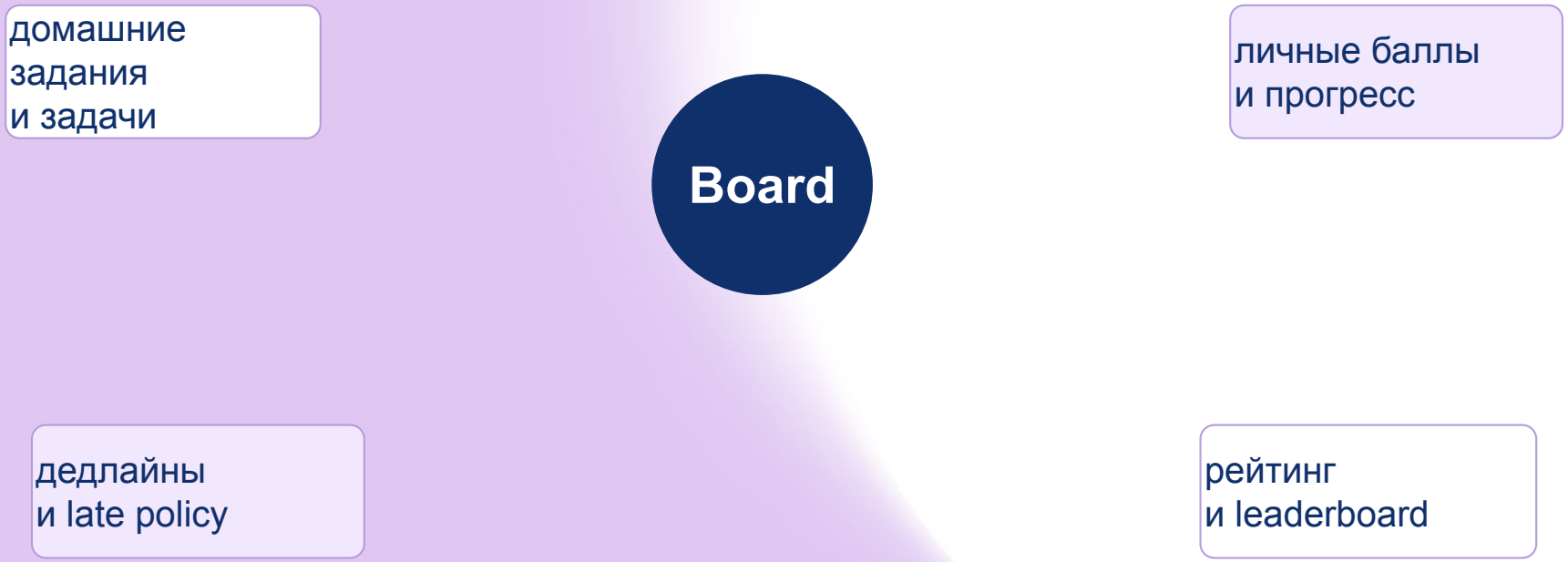
Ownership нужен, чтобы курс не оставался “ничейным”.

Membership отвечает на вопрос: пользователь внутри курса или только пытается попасть?

Slug и UUID ведут к одному правилу: как ни нашли курс, доступ проверяется одинаково.

Board, leaderboard и дедлайны: это уже опыт студента

На этом уровне backend перестаёт быть “сервером с ручками” и начинает собирать для пользователя понятную картину курса.



Важно: студент видит опубликованный контент, администратор — ещё и рабочие материалы. Это часть модели доступа.

Надёжность: защита от “почти сохранилось”

Несколько связанных изменений должны проходить как одна операция. Иначе в платформе легко получить курс без корректных задач, баллов или связей.

1

Транзакции

Если одна часть операции падает, откатывается всё. Данные не остаются в промежуточном состоянии.

2

Консистентность

Проверяем, что task действительно относится к нужному homework и course.

3

Visibility в DB

Repository сразу возвращает данные с учётом hidden-правил и административных сценариев.

Как работа превращалась в задачи для команды

Большие требования приходилось разбивать так, чтобы каждому было понятно: что делаем, где меняем код и что блокирует следующий шаг.

Так команда видела не случайный список тикетов, а цепочку: сначала права и хранение, потом board, score и course info.

Ревью: проверять не только код, но и смысл

Отдельный фокус был на board, leaderboard, course info, invite code, дедлайнах и публикации задач.

Permission

API-контракт

Миграции и backfill

Пользовательский сценарий

Repository запросы

Смысл ревью: поймать ошибку в модели раньше, чем она станет ошибкой в данных.

ИИ в проекте: ускоритель, но не автопилот

ИИ помогал быстрее входить в чужие изменения, разбирать запросы, готовить каркас и группировать задачи. Но решения всё равно нужно было проверять руками.

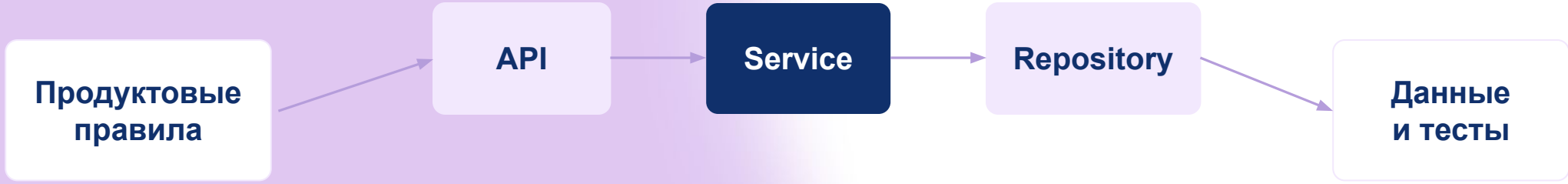
Где помогал
- быстро читать PR и diff
- искать идеи для декомпозиции
- накидывать каркас кода
- помогать с дебагом

Где нужен контроль
- права доступа
- миграции и backfill
- транзакции
- API-контракты
- архитектурные границы

ИИ ускоряет рутину, человек держит архитектуру и ответственность за поведение системы.

Личный вклад: соединить продуктовую логику и backend

Моя роль была в том, чтобы не дать проекту распасться на отдельные endpoint-ы. Правила курса, оценки, доступы и сценарии должны сходиться в одну модель.



Декомпозиция задач, уточнение требований и ревью помогали держать проект в рабочем состоянии, а не только “закрывать тикеты”.

Что получилось в итоге

Backend стал не просто набором ручек, а основанием для учебной платформы: с понятными ролями, безопасными изменениями и предсказуемым поведением для студентов и администраторов.

- архитектура разложена по слоям
- права и статусы стали частью модели
- данные защищены транзакциями и проверками

Спасибо!