Курсовой проект: «Сравнительный анализ подходов к разработке баз данных: классический SQL vs BaaS-платформа»

1. Цель проекта

Сравнить традиционный подход к разработке реляционной базы данных "вручную" с использованием BaaS-платформы (Backend-as-a-Service) на примере Back4app (https://blog.back4app.com/ru/как-использовать-искусственный-инте/). Оценить сложность разработки, производительность, гибкость и особенности каждого подхода.

2. Задачи проекта

1. Придумать сложную предметную область с нетривиальными связями и бизнес-логикой.
2. Реализовать базу данных классическим способом: спроектировать и создать локальную SQL БД (PostgreSQL/MySQL), наполнить её тестовыми данными.
3. Реализовать ту же базу данных с использованием платформы Back4app через её веб-интерфейс и, опционально, SDK.
4. Разработать простое консольное приложение (на Python/Node.js), которое выполняет идентичные операции с обеими базами.
5. Провести сравнительный анализ двух подходов по заданным критериям.
6. Сделать выводы о целесообразности использования каждого подхода в разных сценариях.

3. Варианты сложных предметных областей (на выбор студента)

Вариант 1: Система управления умным домом с иерархией прав

- **Сущности:** Пользователи, Дома, Комнаты, Устройства (лампы, датчики, розетки), Сценарии автоматизации, События (история изменений).
- **Сложность:**
    - Иерархия пользователей: владелец, резидент, гость (разные права на управление устройствами).
    - Сценарии автоматизации: "Если датчик движения в коридоре сработал И время между 18:00 и 06:00, ТО включить свет в коридоре на 2 минуты".
    - Хранение истории показаний датчиков (температура, влажность) и событий.

Вариант 2: Платформа для организации онлайн-образования

- **Сущности:** Студенты, Преподаватели, Курсы, Модули, Уроки (видео, текст, тест), Домашние задания, Проверка ДЗ (с системой рецензий), Прогресс студента.
- **Сложность:**
    - Сложная структура курса (курс → модули → уроки).
    - Система проверки ДЗ: студент сдал → преподаватель проверил → оставил комментарии → студент исправил → новая проверка.
    - Хранение прогресса по каждому уроку для каждого студента.

Вариант 3: Система логистики и управления доставками

- **Сущности:** Заказы, Товары, Склады, Курьеры, Транспортные средства, Маршруты, Трекинг-события (геолокация, статусы).
- **Сложность:**
    - Оптимизация маршрутов (связь между заказами, курьерами и точками доставки).
    - Расчет стоимости доставки на основе веса, габаритов, расстояния и срочности.
    - История перемещений курьеров в реальном времени.

4. Детализация этапов выполнения

Этап 1: Проектирование и реализация "ручной" SQL БД

1. **Проектирование:** Создать детальную ER-диаграмму, нормализовать схему до 3НФ.
2. **Реализация:** Написать SQL-скрипты для создания всех таблиц, первичных и внешних ключей, индексов.
3. **Наполнение:** Создать скрипты (или использовать GUI) для наполнения базы реалистичными тестовыми данными (~100-200 записей в основных таблицах).
4. **Бизнес-логика:** Реализовать 3-5 сложных запросов (с JOIN, агрегирующими функциями, подзапросами), отражающих ключевую бизнес-логику.
    - *Пример для умного дома: "Выбрать все устройства в доме 'X', которыми имеет право управлять пользователь 'Y'."*
    - *Пример для образования: "Найти студентов, которые не сдали более 2 домашних заданий на курсе 'Z'."*

Этап 2: Реализация той же БД в Back4app

1. **Изучение платформы:** Зарегистрироваться на Back4app, изучить интерфейс панели управления.

2. **Создание классов (таблиц):** Перенести схему из Этапа 1 в Back4app, создавая классы и поля. Продемонстрировать работу с указателями (Pointers) для связей.
3. **Наполнение данных:** Загрузить те же самые тестовые данные через панель администратора или с помощью API.
4. **Реализация "запросов":**
    - Использовать "Cloud Code" (функции на Node.js) для реализации той же бизнес-логики, что и в SQL-запросах.
    - Или продемонстрировать, как строятся аналогичные запросы через REST API с использованием where, include, order и других параметров.

Этап 3: Сравнительный анализ

Разработать и заполнить сводную таблицу с анализом по следующим критериям:

Критерий	Классический SQL (PostgreSQL/MySQL)	Back4app (BaaS)	Выводы и комментарии
Сложность начальной настройки			
Скорость разработки			
Гибкость схемы данных			
Сложность реализации связей			
Сложность бизнес-логики (запросы vs Cloud Code)			
Производительность (на примере 3-5 операций)			
Масштабируемость			

Критерий	Классический SQL (PostgreSQL/MySQL)	Back4app (BaaS)	Выводы и комментарии
Безопасность и управление доступом			
Порог входа для новичка			
Стоимость владения (time/money)			

5. Отчётные материалы

1. **Исходный код:**
    - SQL-скрипты создания и наполнения базы.
    - Код Cloud Code функций для Back4app.
    - Код консольного приложения для тестирования.
2. **Документация:**
    - ER-диаграмма SQL-базы.
    - Скриншоты структуры классов в Back4app.
    - Заполненная таблица сравнительного анализа.
3. **Пояснительная записка:**
    - Описание предметной области.
    - Подробное описание проблем, с которыми столкнулся студент при реализации каждым из способов.
    - **Итоговый вывод:** Для каких типов проектов и на каких этапах предпочтительнее использовать классический SQL, а когда выбор в пользу BaaS (Back4app) является оправданным.