ЛЕКЦИЯ 4

Архитектурный синтез программной системы

CSE842 Capstone Project
ОП 6B06102 Computer Science



План лекции

1. Архитектура
2. Драйверы
3. Сценарии качества
4. Стили
5. Тактики
6. C4
7. Порты и адаптеры
8. ADR

Что такое архитектура



Элементы

модули, процессы, хранилища



Связи

вызовы, события, данные



Свойства

и обоснование решений

Всё в архитектуре — компромисс; «почему» важнее «как» (Ричардс, Форд)

Архитектурные драйверы LabBook

FR-2
нет пересечений

NFR-1
p95 ≤ 300 мс

FR-3
письмо без задержки ответа



Ограничение
один сервер

Модифицируемость
SQLite → PostgreSQL

Команда
параллельная работа

Сценарий атрибута качества

Источник	Стимул	Артефакт	Среда	Отклик	Мера
пользователи	50 броней/с	API	нормальная работа	бронь сохранена	p95 ≤ 300 мс

Архитектурные стили

Стиль	Сильная сторона	Слабая сторона
Монолит	простота	«большой ком грязи»
Модульный монолит	границы модулей	дисциплина
Микросервисы	независимость	сложность сети
Событийная	слабая связь	трудная отладка

Для команды до 4 человек — модульный монолит + очередь

Тактики достижения качества



Производительность

индекс, кэш, асинхронность



Доступность

очередь, повтор, heartbeat



Модифицируемость

интерфейсы, конфигурация



Тестируемость

внедрение зависимостей

Тактики конфликтуют: кэш против согласованности

Связность и сцепление



Высокая связность

- меняется вместе — лежит вместе
- модуль «бронирование»



Низкое сцепление

- минимум знаний о соседях
- обмен через контракт

Делить по предметной области, а не по техническим слоям

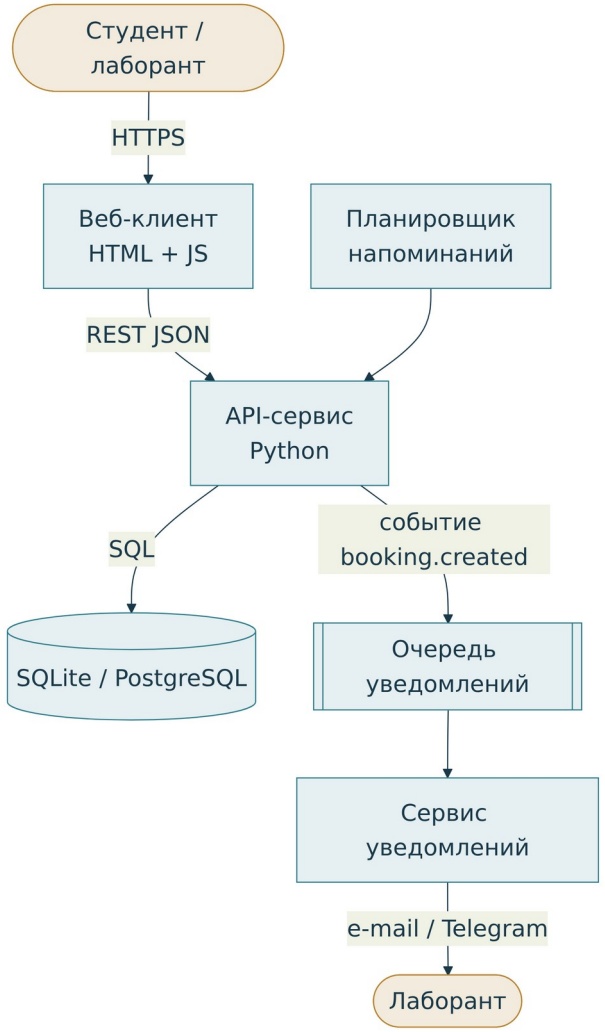
Модель C4: четыре уровня



Контекст	Контейнеры	Компоненты	Код
система, пользователи, внешние системы	API, БД, очередь, клиент	контроллеры, сервисы, репозитории	классы — по необходимости

Для РК1: контекст, контейнеры и компоненты ключевого контейнера

Диаграмма контейнеров LabBook



- синхронно: клиент → API → БД
- асинхронно: событие → очередь
- подписи: технология и протокол
- диаграмма как код: Mermaid

Порты и адаптеры

l04_ports.py

```python
from typing import Protocol

class BookingRepo(Protocol):          # порт
    def overlaps(self, eq: int, s: int, e: int) -> bool: ...
    def add(self, eq: int, s: int, e: int) -> int: ...

class MemoryRepo:                     # адаптер
    def __init__(self): self.rows = []
    def overlaps(self, eq, s, e):
        return any(r[0] == eq and s < r[2] and r[1] < e
                   for r in self.rows)
    def add(self, eq, s, e):
        self.rows.append((eq, s, e)); return len(self.rows)

class BookingService:                 # ядро не знает БД
    def __init__(self, repo: BookingRepo): self.repo = repo
    def book(self, eq, s, e):
        if self.repo.overlaps(eq, s, e): return "409 conflict"
        return f"201 id={self.repo.add(eq, s, e)}"

svc = BookingService(MemoryRepo())
print(svc.book(3, 10, 12), svc.book(3, 11, 13), svc.book(3, 12, 14))
```

```
$ python l04_ports.py
201 id=1 409 conflict 201 id=2
```

- порт — Protocol
- адаптер — MemoryRepo
- замена на SQLite без правки ядра

11–13 — конфликт; 12–14 — смежно, принято

Алгоритм архитектурного синтеза



Architecture Decision Record



Контекст

проблема и драйверы



Альтернативы

что рассмотрели



Решение

и последствия, статус

ADR-0003: уведомления через outbox — почта отвечает 1–5 с

Проверка архитектуры сценариями

Сценарий	Решение	Проверка
Производительность	индекс, лёгкий путь	эксперимент, нагрузка
Целостность	транзакция, ограничение БД	параллельные запросы
Доступность	outbox, повтор	отключить почту
Модифицируемость	порт репозитория	тесты с MemoryRepo

Результат недели 4



Схема C4

контекст, контейнеры, компоненты



Потоки данных

для сквозного сценария



≥ 3 ADR

драйверы и сценарии качества

Вопросы для самопроверки

1. Определение архитектуры
2. Драйверы
3. Сценарий качества
4. Монолит и микросервисы
5. Тактики
6. Уровни C4
7. Порты и адаптеры
8. Разделы ADR



Неделя 4: архитектура (3 балла)