ЛЕКЦИЯ 5

Проектирование данных и интерфейсов подсистем

CSE842 Capstone Project
ОП 6B06102 Computer Science



План лекции

1. Модель области
2. Модель данных
3. Целостность
4. Контракт
5. REST
6. OpenAPI
7. Ошибки
8. События

От архитектуры к контрактам



Модель данных

что хранит подсистема



Контракт

правила обмена

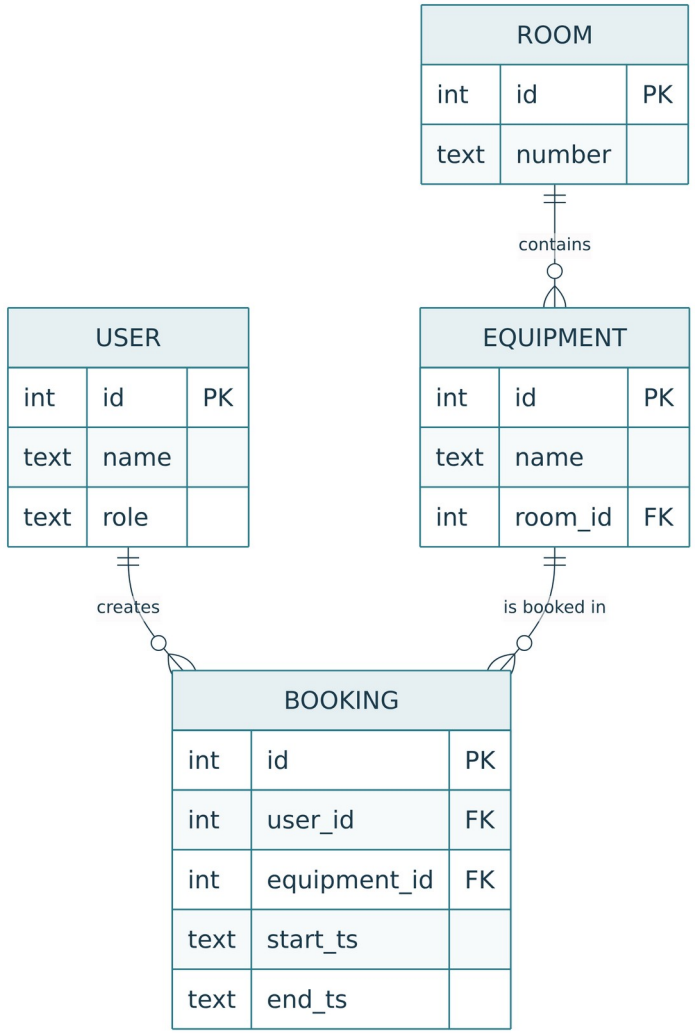


Параллельная работа

клиент и сервер по контракту

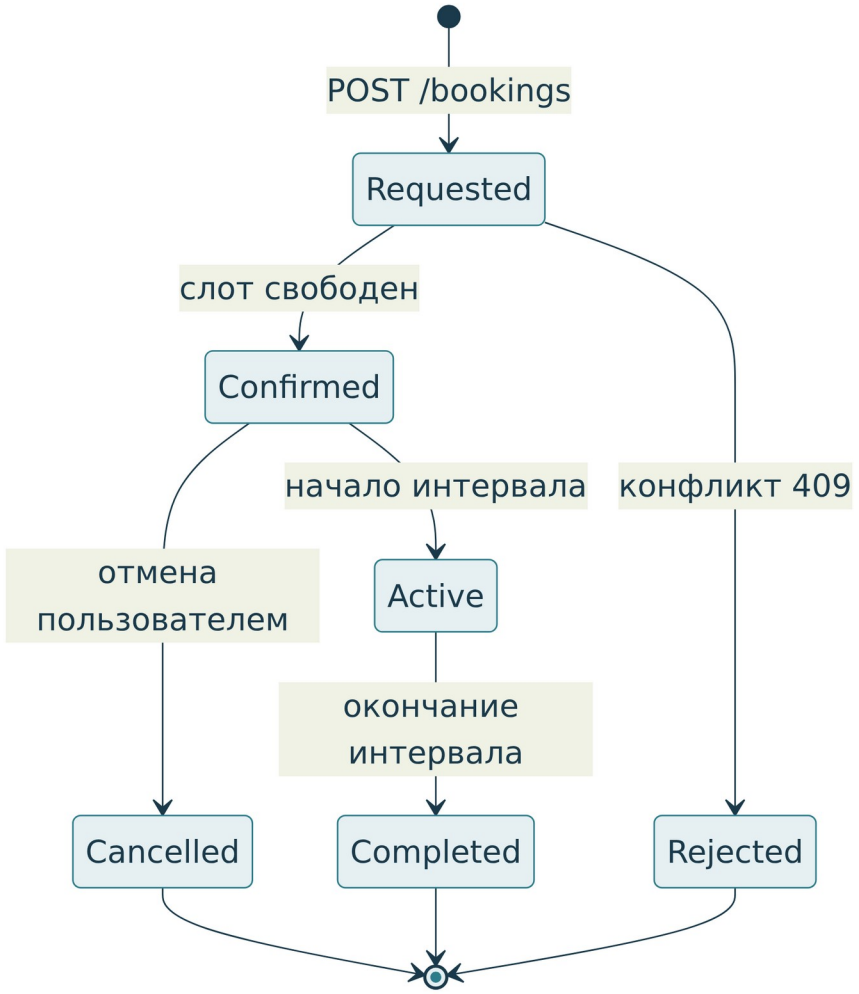
Ошибки контракта дороже всего находятся на интеграции (неделя 10)

Модель предметной области (ER)



- помещение → приборы
- пользователь → брони
- прибор → брони
- инвариант: брони не пересекаются

Жизненный цикл брони



- явные состояния вместо is_ok
- отклонена при конфликте 409
- отменённую нельзя активировать
- каждый переход — тест

Физическая модель: SQL

l05_schema.sql

```sql
CREATE TABLE equipment (
  id    INTEGER PRIMARY KEY,
  name  TEXT NOT NULL,
  room  TEXT NOT NULL);
CREATE TABLE booking (
  id           INTEGER PRIMARY KEY,
  equipment_id INTEGER NOT NULL REFERENCES equipment(id),
  user_id      INTEGER NOT NULL,
  start_ts     TEXT NOT NULL,
  end_ts       TEXT NOT NULL,
  CHECK (start_ts < end_ts));
CREATE INDEX ix_booking_eq_time
  ON booking(equipment_id, start_ts, end_ts);
```

- CHECK (start < end)
- внешний ключ
- индекс для поиска пересечений
- время в UTC, ISO 8601

Целостность на трёх уровнях



Без транзакции — состояние гонки

Контракт «создать бронь»



Вход

equipment_id, start, end



Предусловие

вход, прибор исправен



Постусловие

бронь + событие



Ошибки

400, 401, 404, 409, 422

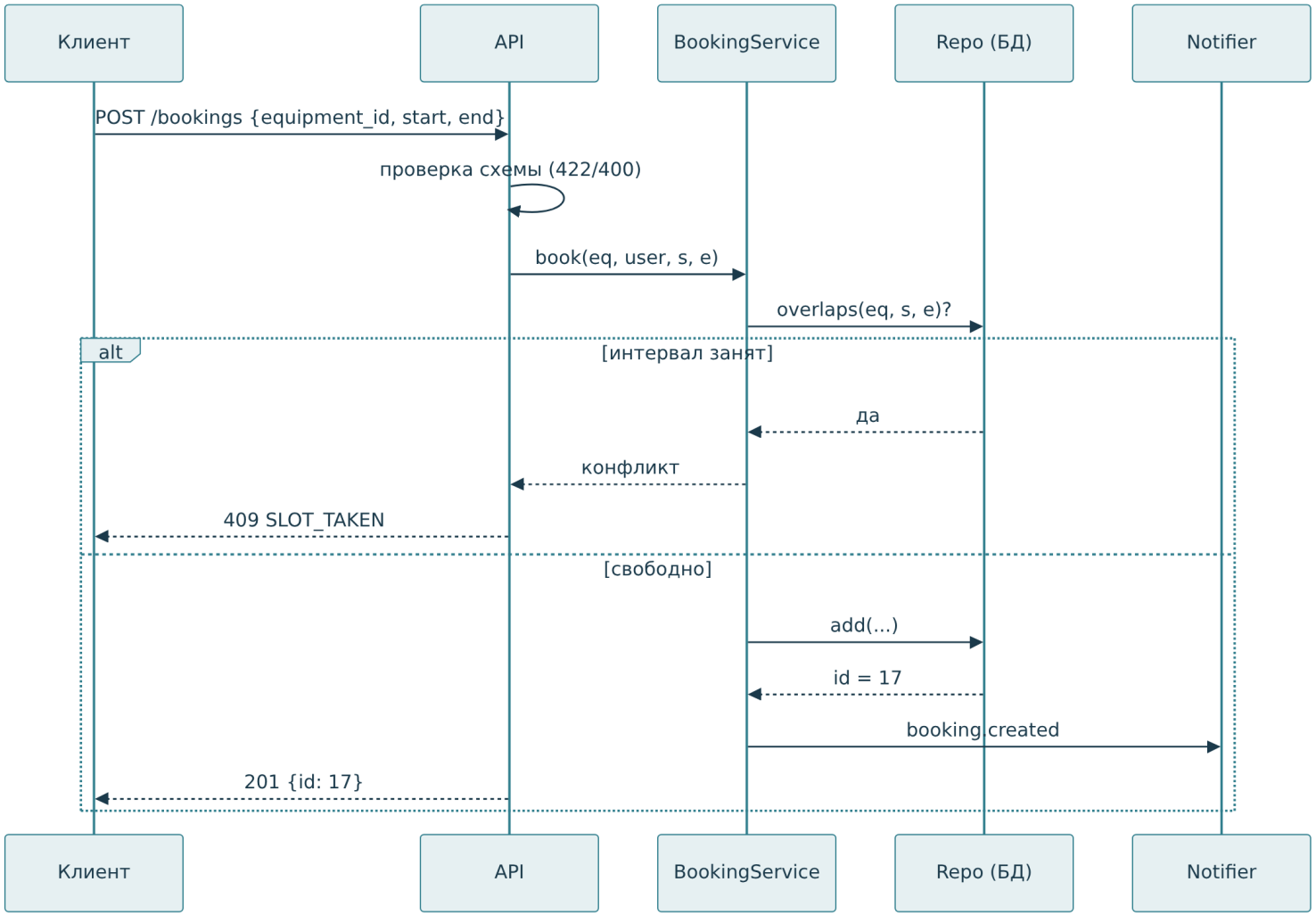
Б. Мейер: проектирование по контракту

Ресурсы и методы API

Метод	URL	Успех
GET	/equipment?room=301	200
GET	/equipment/{id}/slots	200
POST	/bookings	201
GET	/bookings/{id}	200
DELETE	/bookings/{id}	204

POST не идемпотентен: заголовок Idempotency-Key

Диаграмма последовательности



- проверка схемы
- alt: 409 или 201
- событие после записи

Проверка запроса по контракту

l05_contract.py

```python
import json
SCHEMA = {"equipment_id": int, "start": str, "end": str}

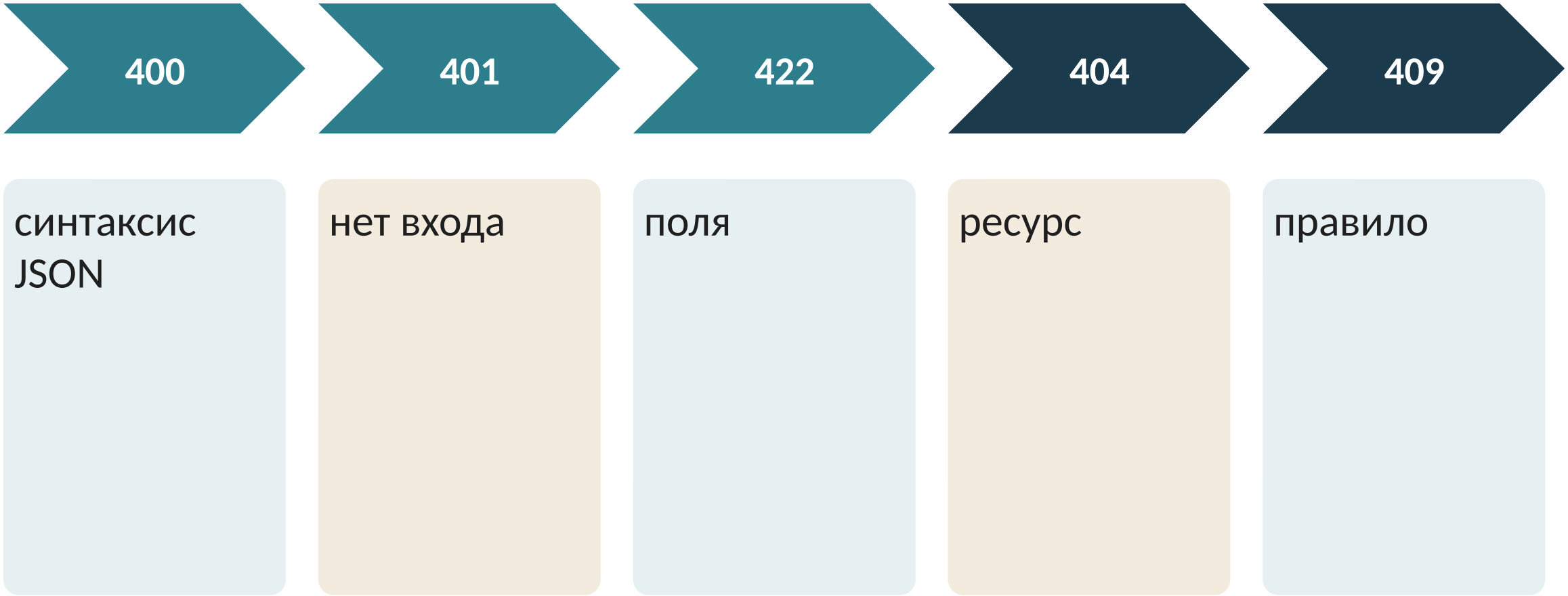
def validate(body: str):
    try: data = json.loads(body)
    except json.JSONDecodeError:
        return 400, {"error": "INVALID_JSON"}
    for k, t in SCHEMA.items():
        if not isinstance(data.get(k), t):
            return 422, {"error": "FIELD", "field": k}
    if data["start"] >= data["end"]:
        return 422, {"error": "EMPTY_INTERVAL"}
    return 201, {"id": 17, **data}

ok = ('{"equipment_id": 3, "start": "2027-03-12T10:00",'
      ' "end": "2027-03-12T12:00"}')
for b in [ok, '{"equipment_id": "3"}', "{oops"]:
    print(*validate(b))
```

```
$ python l05_contract.py
201 {'id': 17, 'equipment_id': 3, 'start': '2027-03-12T10:00', 'end': '2027-03-12T12:00'}
422 {'error': 'FIELD', 'field': 'equipment_id'}
400 {'error': 'INVALID_JSON'}
```

- 201 — корректно
- 422 — тип поля
- 400 — синтаксис

Порядок проверок запроса



Одинаковый порядок — одинаковые коды у всех реализаций

OpenAPI и подход API-first



Спецификация

YAML: пути, схемы, ответы



Документация

Swagger UI



Генерация

клиенты, заглушки, тесты

Контракт события booking.created



Имя и версия

booking.created v1



Поля

booking_id, eq, user, start, end



Доставка

как минимум один раз

Получатель идемпотентен; outbox в той же транзакции

Версионирование контракта

Совместимо

- новое необязательное поле
- новый ресурс
- новый код ошибки

Несовместимо

- удалить или переименовать
- сменить тип
- новое обязательное поле

Несовместимое — в /v2; контрактные тесты в CI

Вопросы для самопроверки

1. Модель области и физическая
2. Инвариант
3. Состояние гонки
4. Пред- и постусловие
5. Правила REST
6. Идемпотентность
7. OpenAPI
8. Совместимость



Неделя 5: модель данных и контракт (3 балла)