Техническое задание: CRUD-сервис расчета депозитов

1. Описание проекта

Необходимо разработать CRUD REST API сервис для управления депозитами с расчетом ежемесячной капитализации процентов. Сервис должен предоставлять функционал для создания, получения, обновления и удаления пользователей и их депозитов, а также отдельный метод для расчета депозита по заданным параметрам.

2. Стек технологий

- Язык программирования: Python 3.10+
- Фреймворк: FastAPI
- ORM: SQLAlchemy
- База данных: PostgreSQL
- Контейнеризация: Docker
- Тестирование: pytest (покрытие не менее 80%)

3. Структура базы данных и связи

Таблица users:
  - id (PK)
  - name (string)
  - email (string, уникальный)
  - created_at (timestamp)

Таблица deposits:
  - id (PK)
  - user_id (FK -> users.id, ON DELETE CASCADE)
  - date (date) — дата заявки
  - periods (int) — количество месяцев
  - amount (numeric) — сумма вклада
  - rate (float) — годовой процент
  - created_at (timestamp)

Техническое задание: CRUD-сервис расчета депозитов

Связь: Один пользователь может иметь много депозитов (One-to-Many). При удалении пользователя удаляются все его депозиты.

4. API спецификация

Пользователи (Users):

POST /users — создать пользователя

GET /users — список пользователей

GET /users/{id} — получить пользователя по ID

PUT /users/{id} — обновить данные пользователя

DELETE /users/{id} — удалить пользователя

Депозиты (Deposits):

POST /deposits — создать депозит (привязка к user_id)

GET /deposits — список депозитов

GET /deposits/{id} — получить депозит по ID

PUT /deposits/{id} — обновить депозит

DELETE /deposits/{id} — удалить депозит

Расчет депозита:

POST /calculate-deposit — принимает входные параметры и возвращает расчет с капитализацией.

Вход (JSON):

- date: string (dd.mm.yyyy)
- periods: integer (1-60)
- amount: integer (10000-3000000)
- rate: float (1-8)

Выход (JSON): даты и суммы на конец каждого месяца или ошибка.

5. Алгоритм расчета

Техническое задание: CRUD-сервис расчета депозитов

1. Принять входные данные и выполнить валидацию.
2. Преобразовать дату в объект datetime.
3. Для каждого месяца рассчитать сумму с учетом капитализации процентов.
4. Вернуть словарь с ключами-датами и значениями-суммами.

6. Требования к разработке

- Репозиторий хранить на github.com
- Создать Dockerfile для сборки приложения
- Покрытие unit-тестами не менее 80%
- Первая итерация — 1 неделя, доработка — 2 дня