МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ   
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение   
высшего образования «Самарский национальный исследовательский университет   
имени академика С.П. Королева»

(Самарский университет)

Институт информатики и кибернетики

Кафедра технической кибернетики

**ОТЧЕТ**

по лабораторной работе №0

Дисциплина «Технологии сетевого проектирования»

«Проектирование приложения»

Выполнили студенты   
 группы 6301

Шкалей Илья

Зотов Антон

Самара 2025Задание на лабораторную работу.

1. **Проектирование архитектуры**:
   * Разработать схему взаимодействия компонентов (сервер, БД, клиент);
   * Разработать логическую схему базы данных (использование, как минимум, 3-х сущностей и 1-ой связи many-to-many);
   * Определить структуру API (REST, методы, URL, параметры, форматы запросов и ответов).
2. **Создание Git-репозитория**:
   * Инициализировать репозиторий (рекомендуется использовать GitHub);
   * Настроить базовую структуру проекта.
   * Добавить .gitignore для исключения ненужных файлов (виртуальное окружение, логи, артефакты сборки и т. д.).
3. **Список используемых технологий**

**Ход работы**

1. Проектирование архитектуры

**Схема взаимодействия компонентов (сервер, БД, клиент):**

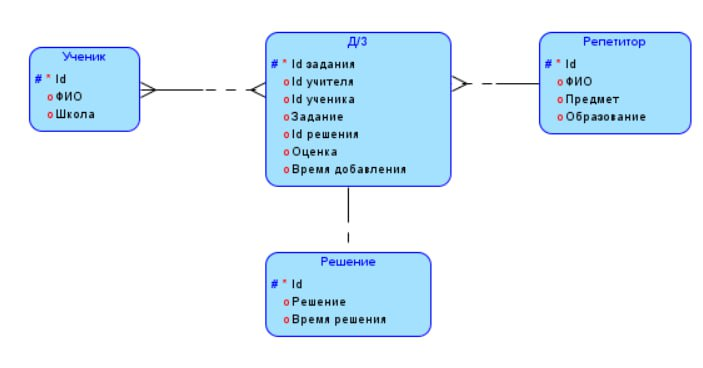
Учитель отправляет дз на сервер, которое доставляется определенному ученику. Когда на сервер приходит решение дз от ученика, сервер отправляет его учителю для возможности оценить данную работу. После проверки сервер отправляет оценку и комментарий к работе ученику, чтобы тот смог посмотреть отзыв о своей работе.

В качестве БД будет журнал с домашним заданием и отзывом на него (оценка + комментарий).

Ученик, может зайти в журнал и посмотреть, есть ли у него дз. Если у него есть дз, то он может его выполнить и загрузить решение для проверки, а после посмотреть результат.

Учитель, может загрузить дз для своего ученика и после получения решения сможет оценить работу.

1. **Логическая схема базы данных** (использование, как минимум, 3-х сущностей и 1-ой связи many-to-many):



**Структура API** (REST, методы, URL, параметры, форматы запросов и ответов):

Основные ресурсы: пользователи системы (репетиторы и ученики),

сообщения между репетитором и учеником(отправление дз и отзыв на него),

Формат запросов и ответов - JSON.

Предполагается наличие системы аутентификации (например, JWT) для защиты API. Авторизация должна проверять права доступа пользователя к ресурсам и действиям.

Детальная структура API по ресурсам:

**Методы**:

POST /api/register - Регистрация нового пользователя (ученика или репетитора).

Запрос (JSON):

GET /api/users/{user\_id} - Получение информации о пользователе.

PUT /api/users/{user\_id} - Обновление информации о пользователе (только для авторизованного пользователя).

GET /api/mentors - Получение списка репетиторов (с фильтрацией).

**Сообщения**:

POST /api/messages/{user\_id} - Отправка нового сообщения(дз или отзыв).

GET /api/messages - Получение списка сообщений для текущего пользователя (с фильтрацией и пагинацией).

Форматы запросов:

• Запросы на создание и обновление (POST, PUT, PATCH): application/json (за исключением /api/materials для загрузки файлов).

• Запросы на получение (GET): параметры могут передаваться как query параметры в URL.

• Обработка ошибок: валидация входных данных, обработку ситуаций, когда ресурс не найден, и т.д. В ответах об ошибках будут возвращаться информативные сообщения и соответствующие HTTP статусы ошибок.

1. **Список используемых технологий:**

* Git, GitHub
* Spring, Spring Web MVC, Spring Security
* PostgreSQL, Hibernate, Jackson
* Postman
* IntelliJ IDEA
* Markdown для документации