МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА №33

ОТЧЕТ ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

Старший преподаватель	23.06.2025	Жиданов К.А.
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЁТ О ЛАБОРАТОРОНОЙ РАБОТЕ №1

«vibe todo list: web + auth + bot + deploy»

по дисциплине: ТЕХНОЛОГИИ И МЕТОДЫ ПРОГРАММИРОВАНИЯ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ гр. №	3333	23.06.2025	Киселев С. И.
		подпись, дата	инициалы, фамилия

Санкт-Петербург 2025

Отчёт о разработке проекта «To-Do List + Telegram-бот»

1 Цель работы

Целью данной лабораторноя работы является вайб-формирование полноценного приложения «To-Do List» с поддержкой:

1. базовых CRUD-операций (создание, чтение, обновление, удаление задач);
2. авторизации пользователей;
3. управления задачами через Telegram-бота;
4. контейнеризации всех компонентов в Docker.

2 Структура итогового репозитория

Файл	Назначение	Ключевые особенности
.env	конфигурация	логин, пароль, DB, HOST, PORT, BOT_TOKEN
server.js	чистый Node.js-сервер	статическая раздача, сессии, REST API
bot.js	Telegram-бот	команды `/add /help /list / edit /delete`
db.sql	инициализация БД	таблица items
Dockerfile	образ Node.js	установка зависимостей, за-пуск
docker-compose.yml	оркестрация, контейнеризация	контейнеры: app, bot, db, nginx
nginx.conf	реверс-прокси	proxy → app:3000, порт host NGINX_PORT
index.html	UI списка задач	AJAX fetch, credentials = include
login.html	форма входа	передача username / password

3 Ход работы

3.1 Инициализация репозитория и базовый CRUD

1. Создана таблица `items` в MySQL, поля: id INT AUTO_INCREMENT, text VARCHAR.

2. Реализованы маршруты:

2.1. `GET /api/items` → список задач.

2.2. `POST /api/add` → добавление новой записи.

2.3. `POST /api/delete` → удаление по id.

2.4. `POST /api/edit` → обновление текста по id.

3.2 Авторизация и сессии

1. `POST /login` проверяет учётные данные из .env.

2. При успехе создаётся cookie `sid`; сессии хранятся в Map на памяти.

3. `GET /logout` удаляет cookie и запись Map.

4. Все маршруты `/api/*` возвращают 401, если нет действующей сессии.

3.3 Фронтенд

1. `index.html` формирует таблицу задач.

2. AJAX-запросы выполняются с `credentials: "include"` → cookie отправляется автоматически.

3. При ответе 401 страница перенаправляется на `/login.html`.

3.4 Telegram-бот

1. Библиотека node-telegram-bot-api.

2. Подключается к той же БД:

2.1. `/add текст` → INSERT.

2.2. `/list` → SELECT.

2.3. `/edit id новый текст` → UPDATE.

2.4. `/delete id` → DELETE.

3. Бот работает в режиме polling, чат определяется по `msg.chat.id`.

3.5 Контейнеризация Docker

1. **db** → mysql:8.0, том `db-data`, примонтирован `db.sql` в `/docker-entrypoint-initdb.d/`.

2. **app** → образ Node.js 18 alpine, запускает `server.js`.

3. **bot** → отдельный образ Node.js, запускает `bot.js`.

4. **nginx** → ревёрс-прокси, пробрасывает `$NGINX_PORT:8080`.

5. Переменные окружения подключаются через файл `.env`.

3.6 Рефакторинг

1. Заново реализованы:

1.1. парсинг URL и тела запросов;

1.2. статическая отдача файлов;

1.3. куки и сессионный механизм;

4 Выводы

Работа полностью реализовала цели:

1. Сервер написан на чистом Node.js, что позволило минимизировать зависимости и лучше понять механизмы HTTP.

2. Реализована защищённая авторизация, CRUD API и удобный фронтенд.

3. Telegram-бот расширил каналы взаимодействия.

4. Docker-контейнеры обеспечили воспроизводимость и простое развёртывание.

Система готова к дальнейшему развитию — добавлению регистрации, роли пользователей, разграничению доступа и CI/CD-деплоя на VPS.

Весь код и сам Typst отчёт навайблены с помощью ChatGPT и доработаны вручную