

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

К.А. Жиданов

\_\_\_\_\_  
должность, уч. степень,  
звание

\_\_\_\_\_  
подпись, дата

\_\_\_\_\_  
инициалы, фамилия

**ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №1**

по дисциплине: Технологии и методы программирования

СТУДЕНТ ГР. №

3331

А.А. Коновалова

\_\_\_\_\_  
подпись, дата

\_\_\_\_\_  
инициалы, фамилия

Санкт-Петербург 2025

## **1. Введение**

### **Цель работы:**

Разработка To-Do приложения с интеграцией с Telegram ботом при помощи взаимодействия с ИИ-ассистентом и поддержкой работы сервера через Яндекс Облако.

### **Функционал приложения:**

- Добавление/удаление/редактирование задач
- Аутентификация пользователей
- Интеграция с Telegram (управление задачами)

### **Используемый стек технологии:**

- Сервер: Node.js, Express
- База данных: MySQL
- Клиент: HTML, CSS, JavaScript
- Аутентификация/безопасность: JWT, bcrypt
- Конфигурация окружения: dotenv
- Интеграция: Telegram Bot API
- Управление процессами: PM2
- Прочее: Axios (HTTP-клиент), node-telegram-bot-api

## **2. Этапы разработки с использованием ИИ**

### **Настройка базы данных**

#### **I. Первоначальная настройка проекта**

**Запрос:** Создай структуру To-Do приложения на Node.js с:

- Express сервером
- MySQL базой данных
- JWT аутентификацией
- Telegram интеграцией

#### **II. Настройка базы данных**

**Запрос:** Напиши SQL-скрипты для создания:

1. Базы данных todolist
2. Таблицы users с полями: id, username, password\_hash, telegram\_chat\_id
3. Таблицы items с полями: id, user\_id, text, created\_at  
Включи внешний ключ между items.user\_id и users.id.

### III. Подключение к MySQL и отладка

**Запрос:** Покажи пример db.js (пул соединений, dotenv, обработка ошибок). После npm start получаю Access denied → помоги исправить параметры подключения.

### IV. Реализация backend-маршрутов

**Запрос:** Добавь в index.js маршруты для:

- GET /list → список задач
- POST /add → добавление задачи
- PUT /edit/:id → редактирование
- DELETE /delete/:id → удаление

После запуска GET /list выдаёт 404 → исправь отдачу статических файлов и шаблон index.html.

### V. Клиентская часть и AJAX

**Запрос:** Напиши index.html и client.js для:

- Отображения таблицы задач
- Добавления, редактирования, удаления через fetch/AJAX

Кнопка Remove не появляется → проверь рендеринг кнопок в DOM.

### VI. Аутентификация пользователей

**Запрос:** Реализуй регистрацию и логин через JWT и bcrypt:

- /register → хеширование и сохранение
- /login → проверка хеша, выдача токена
- Защити CRUD-маршруты middleware, передавай userId через token

При попытке добавления задачи получаю 500 (Missing userId) → обнови клиентские запросы, передавай токен.

### VII. Интеграция Telegram-бота

**Запрос:** Напиши bot.js с node-telegram-bot-api:

- Многошаговая регистрация/логин
- Команды: My To-Do List, Add Task, Edit Task, Delete Task, Logout
- Храни сессии и передавай JWT к API

Бот отвечает пусто → исправь URL (используй localhost:3000).

## VIII. Тестирование внешнего доступа

**Запрос:** Настрой внешний доступ через serveo и localtunnel:

- ssh -R 80:localhost:3000 serveo.net
- lt --port 3000

На публичном URL таблица пуста → проверь импорт db.sql на сервере.

## IX. Развёртывание в Yandex Cloud

**Запрос:** Пошагово в консоли Yandex Cloud:

- Сгенерируй SSH-ключи в Windows, добавь публичный ключ в метаданные VM
- Создай VM, укажи публичный ключ и метаданные
- Открой Security Group (3000/TCP, 0.0.0.0/0)

## X. Перенос и запуск на VM

**Запрос:** Клонировать репозиторий на VM, переключись на нужную ветку, git pull, npm install, npm run start через PM2.

При npm install ошибка jsonwebtoken → замени версию на существующую.

## XI. Импорт и проверка базы на VM

**Запрос:** Импортируй схему: mysql -u root -p todolist < db.sql, проверь SHOW TABLES;, создай пользователя appuser.

## XII. Мониторинг и оптимизация

**Запрос:** Управление процессами PM2:

- pm2 start index.js --name todolist-api
- pm2 start bot.js --name todolist-bot
- pm2 list, pm2 delete <name>, pm2 restart <name>
- Очистить лишние зависимости (npm prune).

### **ХIII. Финальное тестирование и документация**

**Запрос:** Проверь работу веб-интерфейса и бота по публичному IP, задеплой Nginx в качестве обратного прокси, составь краткий список использованных технологий и логи запросов.

#### **3. Хостинг сервера**

Мы развернули весь стек на виртуальной машине в Yandex Cloud: там запущены Node.js/Express, MySQL, PM2 и (при необходимости) Nginx. Такой подход даёт нам:

- **Гибкость и контроль.** Через веб-консоль или CLI можно быстро менять размер VM, открывать порты и настраивать бэкапы.
- **Простая инфраструктура.** Один инстанс держит и API, и Telegram-бота, соединяясь с локальным MySQL; PM2 следит за процессами и автоматически перезапускает упавшие сервисы.
- **Безопасность доступа.** Подключение — по SSH-ключу, внешний порт 3000 открыт только в Security Group (настройка CIDR).
- **Лёгкий деплой и апгрейд.** Любые изменения из GitHub подтягиваются командой `git pull`; пересборка через `npm install`, затем `pm2 restart`.

В итоге мы получаем надёжную и управляемую среду, где всё - от API до бота - держится в едином месте с понятным контролем версий и автоматическим мониторингом.

#### **4. Решённые проблемы**

##### **I. Ошибка доступа к MySQL (ER\_ACCESS\_DENIED\_ERROR / ER\_BAD\_DB\_ERROR)**

– Причина: на локальной машине и на VM были разные учётные данные и базы («root» без пароля против «appuser»/«todolist»).

– Решение: в `.env` и в коде задать единые параметры подключения (DB\_USER, DB\_PASS, DB\_NAME), экспортировать их через `dotenv`; на VM создать БД todolist, пользователя appuser с паролем и дать ему права, затем в

mysql2.createPool использовать именно эти настройки.

## II. Ошибка “Invalid URL” в Telegram-боте

– Причина: бот указывал жёстко `http://158.160.136.211:3000`, а при локальном тесте обращался к неверному хосту.

– Решение: в `bot.js` завести константу

```
const API = process.env.API_URL || 'http://localhost:3000';
```

и вызывать `axios.post(\${API}/login`, ...)`. Это позволяет гибко переключать между `localhost`` и публичным IP.

## III. Редактирование через бота “Item not found or not yours”

– Причина: бот для редактирования передавал неправильный `id` (использовал номер строки вместо реального идентификатора из БД).

– Решение: сохранять при получении списка массив объектов `lastItems`, а при вводе номера задачи вычислять `realId = lastItems[number-1].id`, и уже по нему отправлять `PUT /items/:realId`.

## IV. Конфликты зависимостей и ошибки npm (No matching version, deprecations)

– Причина: в `package.json` были указаны несуществующие версии (например, `jsonwebtoken@^9.2.0`).

– Решение: выполнить `npm view jsonwebtoken version`, заменить на актуальную (`^9.0.0`), затем `npm install` или явно установить нужную версию — после этого `npm audit fix` завершается без ошибок.

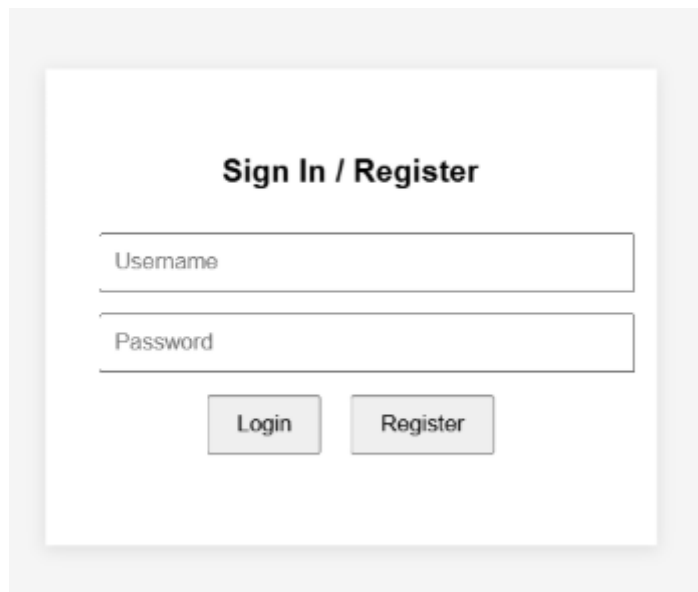
## 5. Вывод

В рамках лабораторной работы была достигнута поставленная цель — разработано полнофункциональное To-Do приложение с интеграцией Telegram-бота, реализующего взаимодействие с пользователем. Основные функции включают регистрацию, авторизацию, добавление, редактирование, удаление и просмотр задач. Управление задачами осуществляется через Telegram-интерфейс с использованием API-запросов к серверу.

Особое внимание было уделено корректному хранению сессий

пользователей, обработке состояний и устранению возможных ошибок, связанных с устаревшими данными. В процессе разработки использовались современные технологии: Node.js, Telegram Bot API, REST API и хостинг серверной части в Яндекс Облаке.

Также важным аспектом работы стало активное использование ИИ-ассистента в процессе проектирования, отладки и улучшения логики взаимодействия клиента и сервера. Это позволило ускорить разработку и повысить качество архитектуры приложения.



A centered white box with a light gray border and shadow. Inside, the title "Sign In / Register" is centered at the top. Below it are two input fields: "Username" and "Password". At the bottom are two buttons: "Login" and "Register".

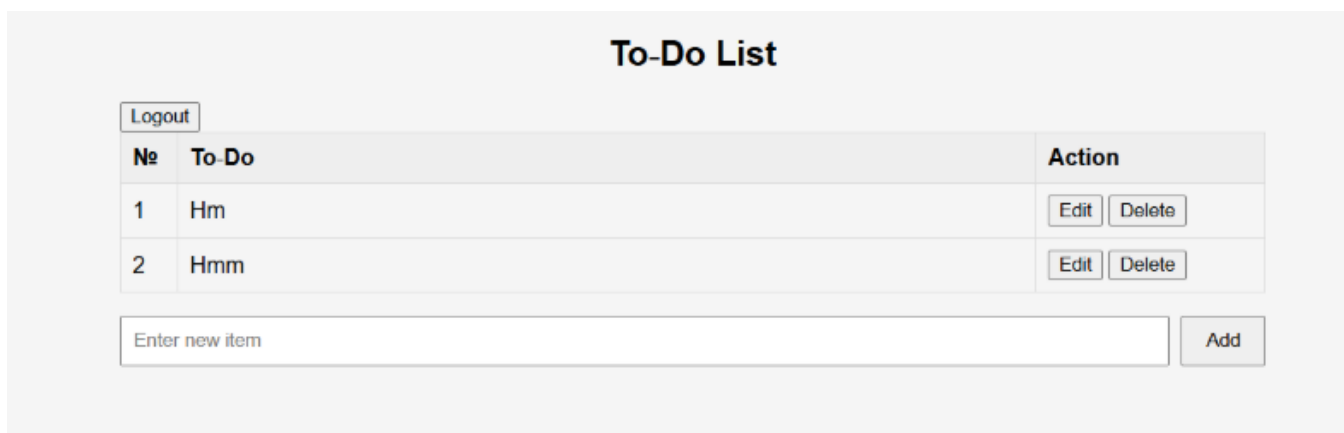
**Sign In / Register**

Username

Password

Login Register

Рисунок 1 – Аутентификация вход/регистрация.



The interface has a title "To-Do List" at the top center. Below it is a "Logout" button. A table with three columns: "№", "To-Do", and "Action". The table contains two rows of items. Below the table is an input field labeled "Enter new item" and an "Add" button.

**To-Do List**

Logout

№	To-Do	Action
1	Hm	Edit Delete
2	Hmm	Edit Delete

Enter new item Add

Рисунок 2 – Интерфейс списка.



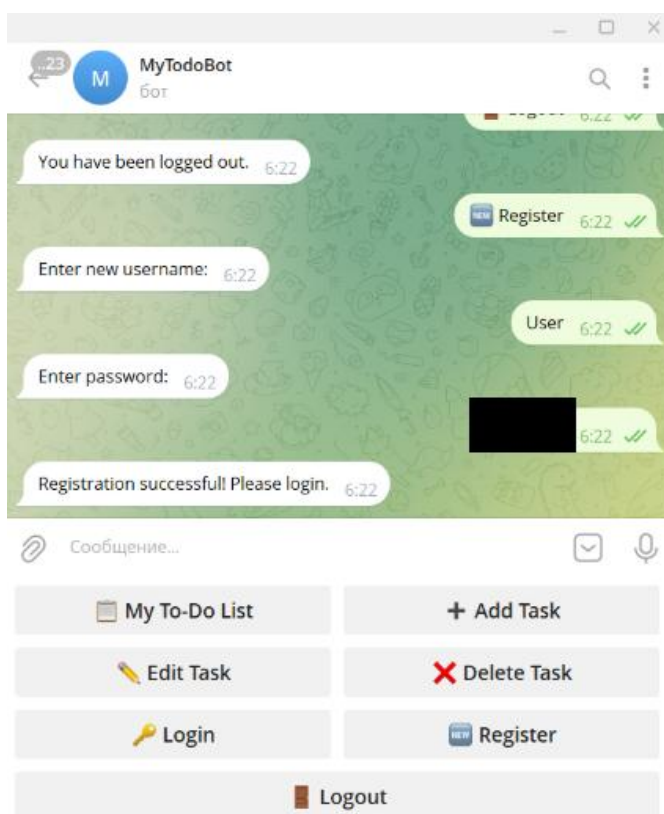


Рисунок 3 – регистрация аккаунта через бота.

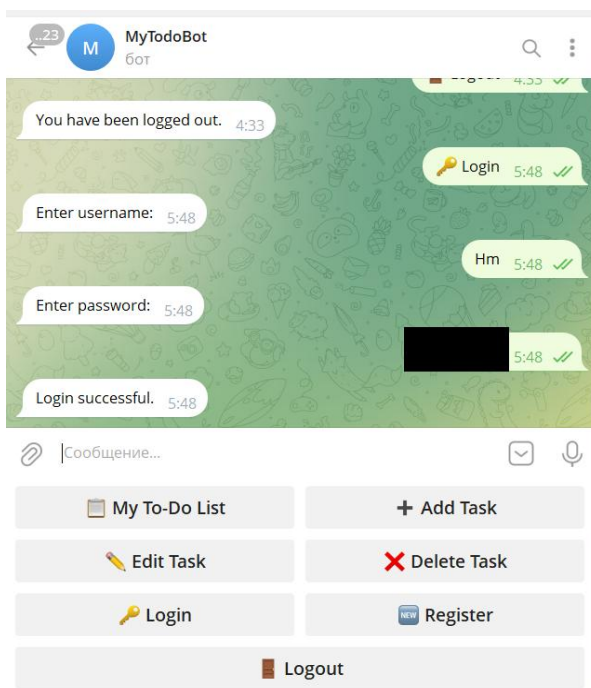


Рисунок 4 – вход в аккаунт через бота.

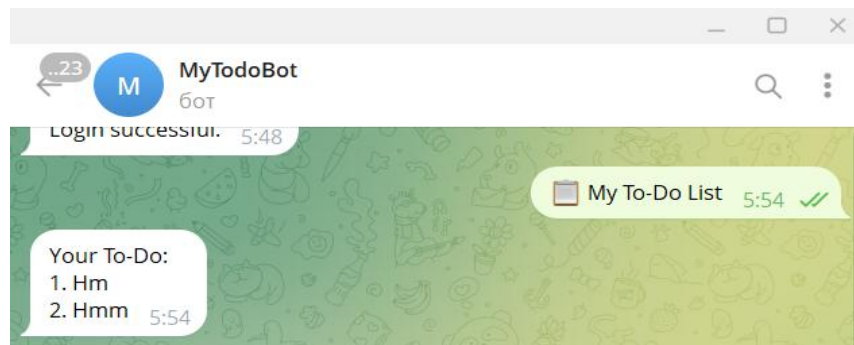


Рисунок 5 – Список задач

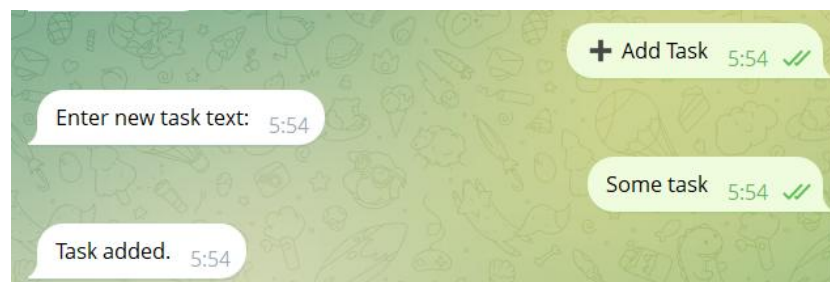


Рисунок 6 – Функция «добавить задачу»

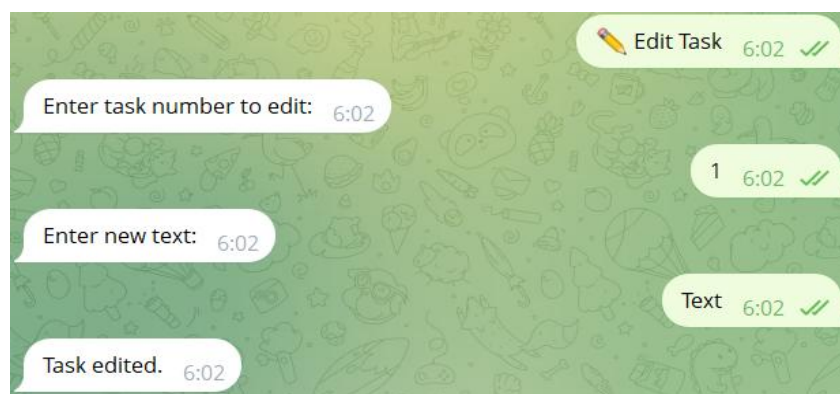


Рисунок 7 – Функция «редактировать задачу»

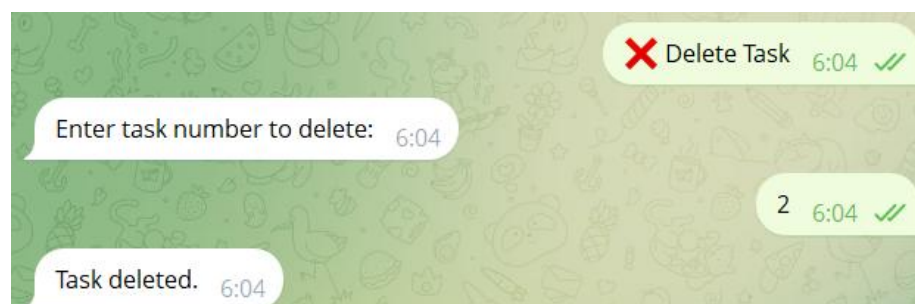


Рисунок 8 – Функция «удалить задачу»

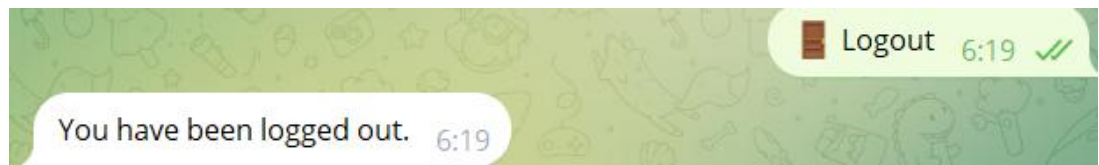


Рисунок 9 - Функция «Logout»