Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ

УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра автоматизированной обработки информации (АОИ)

**ВНЕДРЕНИЕ WEBSOCKET**

Отчет о выполнении лабораторной работы

по дисциплине «Разработка интернет-приложений»

Студент гр. 429-3

\_\_\_\_\_\_\_\_\_\_Бабец А. А.

«\_\_\_» \_\_\_\_\_\_\_\_\_ 2021 г.

Ассистент каф. АОИ

\_\_\_\_\_\_\_\_\_\_ Владимиров М. В.

«\_\_\_» \_\_\_\_\_\_\_\_\_ 2021 г.

Томск 2021

**Оглавление**

[1 Введение 3](#_Toc83149604)

[2 Основная часть 4](#_Toc83149605)

[2.1 Описание используемой среды программирования 4](#_Toc83149606)

[2.2 Описание метода решения задачи 4](#_Toc83149607)

[2.3 Описание фрагментов кода 5](#_Toc83149608)

[2.4 Описание функциональных возможностей разработанного приложения. 6](#_Toc83149609)

[3 Заключение 7](#_Toc83149611)

[Приложения 8](#_Toc83149612)

# Введение

Целью лабораторной работы является интеграция технологии WebSocket в проект.

# Основная часть

## Описание стека технологий

Для реализации Websocket в проекте использовалась библиотека для NodeJs под названием ws.

Также писать код будем в Visual Studio Code.

## Описание метода решения задачи

Для реализации приложения необходимо реализовать api для взаимодействия с сервером.

## Описание фрагментов кода

Для реализации необходимо импортировать ws:

const WebSocket = require('ws');

Далее необходимо начать слушать порт через websocket:

wss.on('connection', function connection(ws) {

console.log('A new client Connected!');

ws.send('Welcome New Client!');

ws.on('message', function incoming(message) {

console.log('received: %s', message);

wss.clients.forEach(function each(client) {

if (client !== ws && client.readyState === WebSocket.OPEN) {

client.send(message);

}

});

});

});

server.listen(5000, () => console.log(`Lisening on port :5000`))

При удачном подключении каждому клиенту будет отправлено сообщение об удачном подключении.

Листинг кода представлен в приложении A.

## Описание функциональных возможностей разработанного приложения

Запустив приложение, мы можем через Postman отправлять GET,POST, PUT,DELETE запросы на наш сервер, и сервер возвращает json с результатом запросов.

Скриншоты приложения представлен в приложении Б.

# Заключение

В практической работе были достигнуты поставленные задачи. Мы познакомились с библиотекой ws и внедрили с помощью нее технологию WebSocket.

# Приложения

# Приложение А

# (обязательное)

# Листинг кода

index.js

const express = require('express')

const mongoose = require('mongoose')

const WebSocket = require('ws');

const app = express()

const server = require('http').createServer(app);

const wss = new WebSocket.Server({server:server});

app.use(express.json({extended: true}))

app.use('/api/todo', require('./routes/todo.route'))

async function start() {

    try {

        await mongoose.connect('mongodb+srv://kot:adept@mycluster.flb6q.mongodb.net/todoBase?retryWrites=true&w=majority')

        wss.on('connection', function connection(ws){

            console.log('New Client Connected');

            ws.send('Welcome to server');

            ws.on('message', function connection(ws){

                console.log('received: %s', message);

                wss.clients.forEach(function each(client){

                    if (client !== ws && client.readyState === WebSocket.OPEN){

                        client.send(message);

                    }

                });

            });

        });

        server.listen(5000, () => console.log(`Listening on port: 5000`))

    } catch (err) {

        console.error(err)

    }

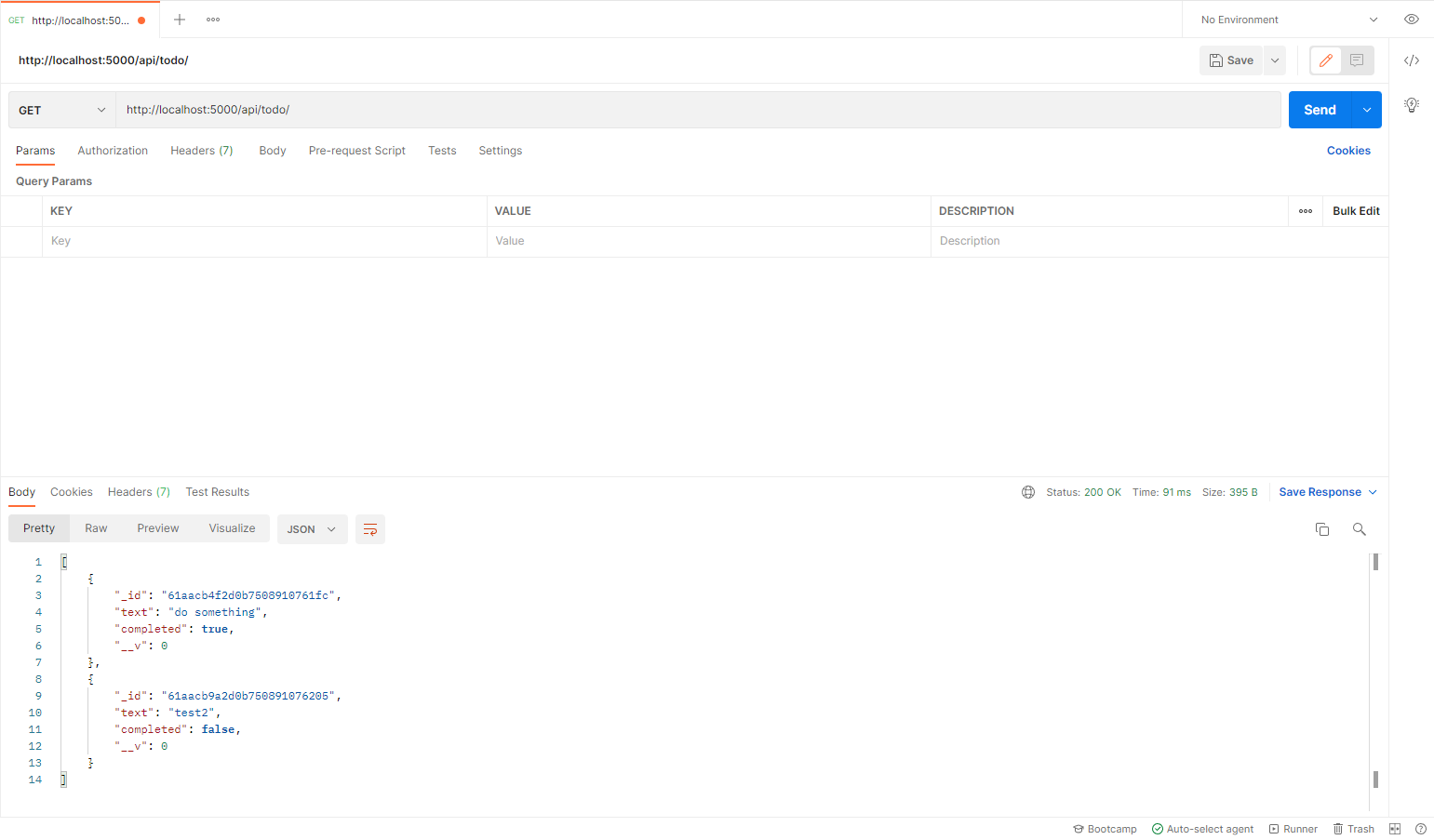
}

start()

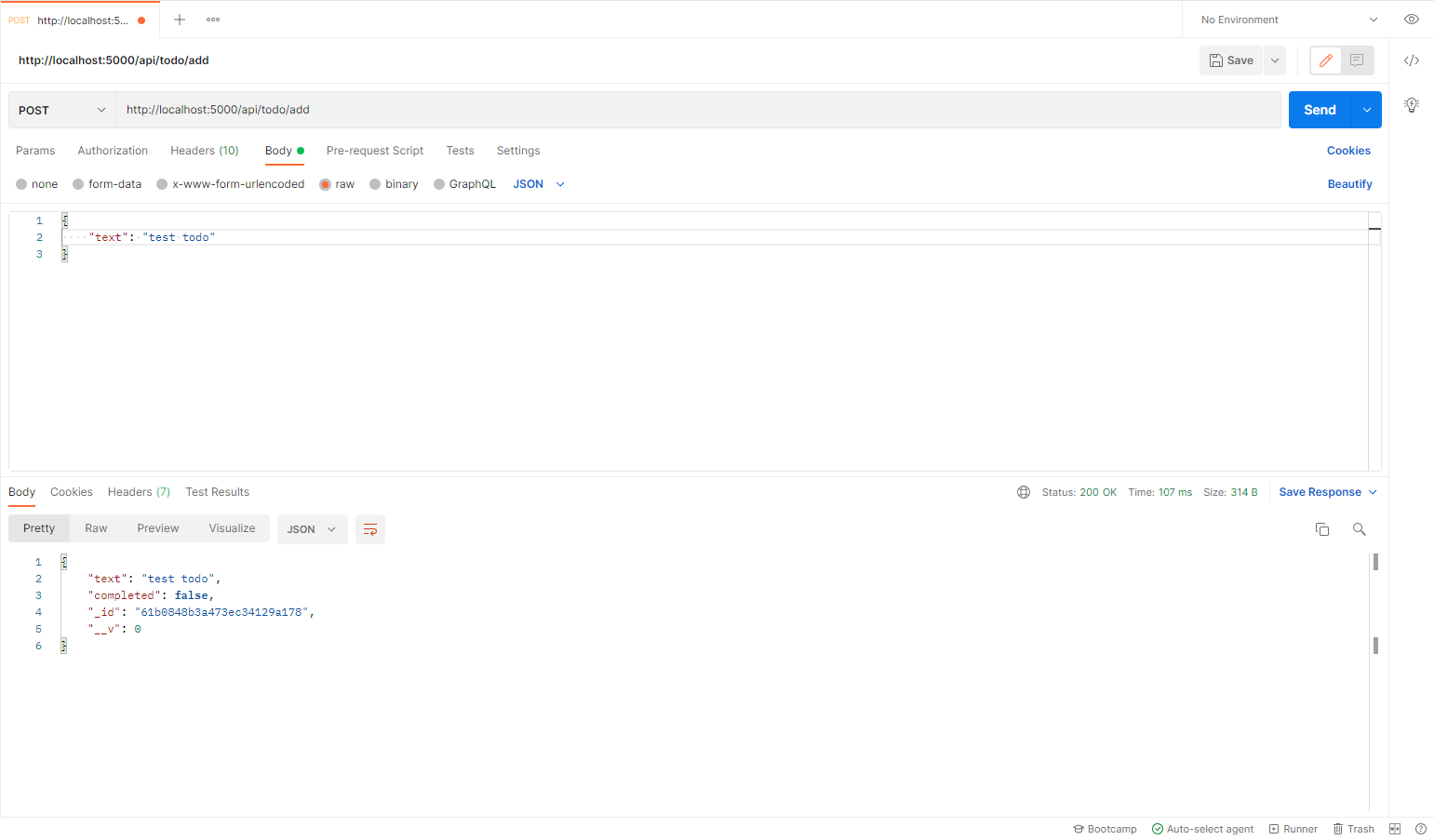
# Приложение Б

# (обязательное)

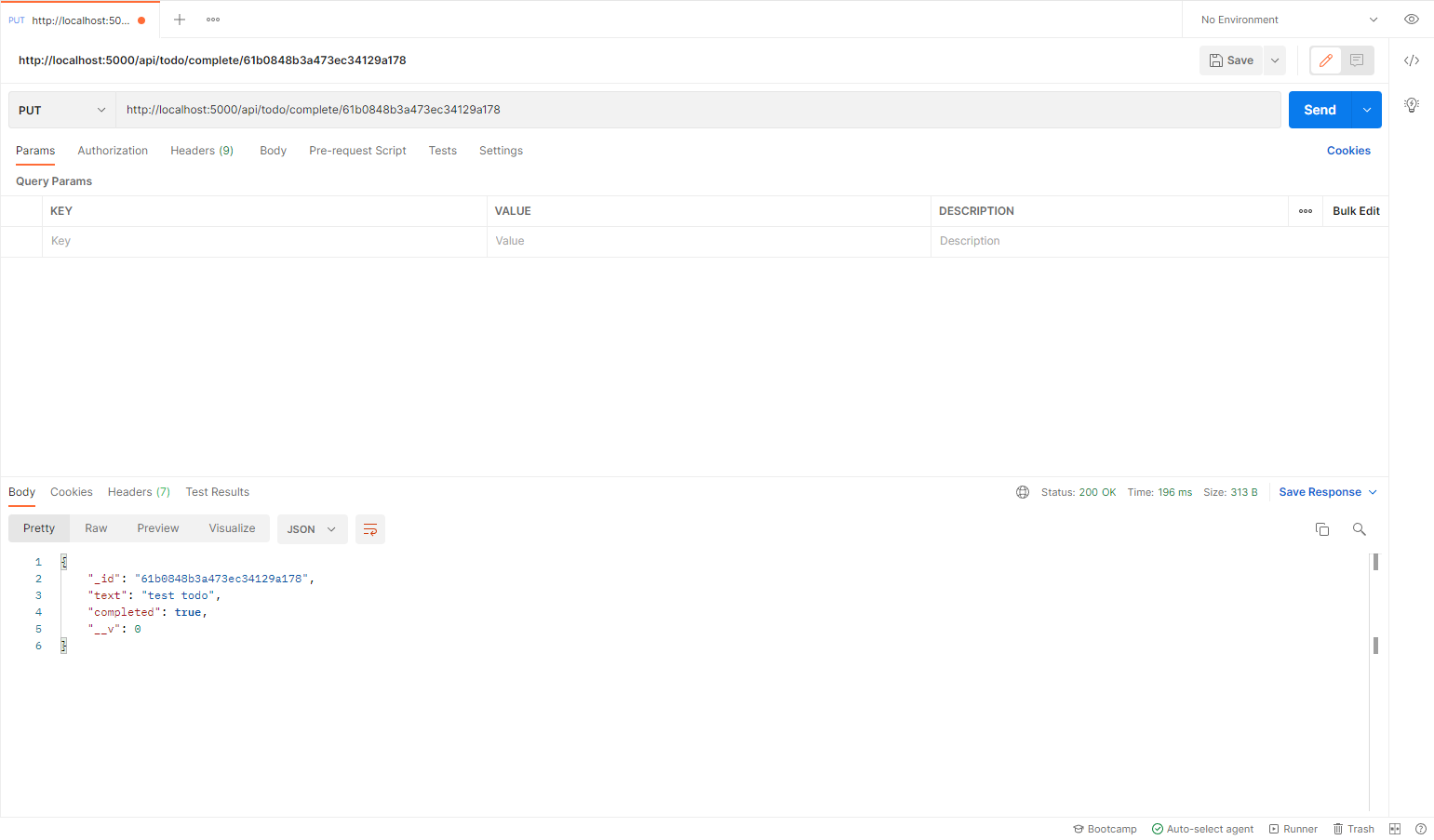
# GET запрос



# POST запрос



# PUT запрос

**DELETE запрос**

