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)

**BACKEND-ПРИЛОЖЕНИЕ «СПИСОК ДЕЛ»**

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 разработка backend-приложения для списка дел.

Задачи лабораторной работы следующие:

* Выбор современного фреймворка;
* Изучение основ фреймворка;
* Реализовать CRUD api.

# Основная часть

## Описание стека технологий

Для разработки backend-приложения было принято решение воспользоваться фреймворком express для NodeJs. Express представляет собой популярный веб-фреймворк, написанный на JavaScript и работающий внутри среды исполнения NodeJs. Язык программирования был выбран NodeJs. NodeJs –Node или Node.js — программная платформа, основанная на движке V8, превращающая JavaScript из узкоспециализированного языка в язык общего назначения.

Также писать код будем в Visual Studio Code, так как это бесплатная IDE с широким функционалом.

## Описание метода решения задачи

Для реализации магазина техники были реализованы контроллеры, которые обрабатывают запросы. Роутеры для перенаправления запросов, и методы обработки запросов.

## Описание фрагментов кода

Перейдём к описанию основных функций. Создадим роутер router, который отвечает за направление запроса:

const {Router} = require('express')

const router = Router()

Создадим метод post, который создает объект в базе данных:

router**.**post**(**'/add'**,** **async** **(**req**,** res**)** **=>** **{**

**try** **{**

const **{**text**}** **=** req**.**body

const todo **=** **await** **new** Todo**({**

text**,**

completed**:** **false**

**})**

**await** todo**.**save**()**

res**.**json**(**todo**)**

**}**

**catch** **(**error**)** **{**

console**.**log**(**error**)**

**}**

**})**

Далее напишем метод get для получения всех задач, имеющихся в базе данных:

router**.get(**'/'**,** **async** **(**req**,** res**)** **=>** **{**

**try** **{**

const todo **=** **await** Todo**.**find**()**

res**.**json**(**todo**)**

**}**

**catch** **(**error**)** **{**

console**.**log**(**error**)**

**}**

**})**

Создадим метод put, который изменяет объект с нужным id (ставит или убирает флаг «выполнено»):

router**.**put**(**'/complete/:id'**,** **async** **(**req**,** res**)** **=>** **{**

**try** **{**

const todo **=** **await** Todo**.**findOne**({**\_id**:** req**.params.**id**})**

todo**.**completed **=** **!**todo**.**completed

**await** todo**.**save**()**

res**.**json**(**todo**)**

**}**

**catch** **(**error**)** **{**

console**.**log**(**error**)**

**}**

**})**

Создадим метод delete, который удаляет объект с нужным id:

router**.**delete**(**'/delete/:id'**,** **async** **(**req**,** res**)** **=>** **{**

**try** **{**

const todo **=** **await** Todo**.**findOneAndDelete**({**\_id**:** req**.params.**id**})**

res**.**json**(**todo**)**

**}**

**catch** **(**error**)** **{**

console**.**log**(**error**)**

**}**

**})**

Листинг кода представлен в приложении Б.

## Описание функциональных возможностей разработанного приложения

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

Скриншоты приложения представлен в приложении Б.

# Заключение

В лабораторной работе были достигнуты поставленные задачи. Мы познакомились с фреймворком Express и изучили его основы. Разработали проект для интернет магазина с таким функционалом с CRUD api.

# Приложения

# Приложение А

# (обязательное)

# Листинг кода

todo.route.js

const {Router} = require('express')

const router = Router()

const Todo = require('../models/Todo')

router.post('/add', async (req, res) => {

try {

const {text} = req.body

const todo = await new Todo({

text,

completed: false

})

await todo.save()

res.json(todo)

}

catch (error) {

console.log(error)

}

})

router.get('/', async (req, res) => {

try {

const todo = await Todo.find()

res.json(todo)

}

catch (error) {

console.log(error)

}

})

router.delete('/delete/:id', async (req, res) => {

try {

const todo = await Todo.findOneAndDelete({\_id: req.params.id})

res.json(todo)

}

catch (error) {

console.log(error)

}

})

router.put('/complete/:id', async (req, res) => {

try {

const todo = await Todo.findOne({\_id: req.params.id})

todo.completed = !todo.completed

await todo.save()

res.json(todo)

}

catch (error) {

console.log(error)

}

})

module.exports = router

Todo.js

const { Schema, model, Types} = require('mongoose')

const schema = new Schema({

text: {type: String},

completed: false

})

module.exports = model('Todo', schema)

index.js

const express **=** require**(**'express'**)**

const mongoose **=** require**(**'mongoose'**)**

const app **=** express**()**

const PORT **=** process**.**env**.**PORT **||** 5000

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'**)**

app**.**listen**(**PORT**,** **()** **=>** **{**

console**.**log**(**'Server started on port ' **+** PORT**)**

**})**

**}** **catch** **(**err**)** **{**

console**.**error**(**err**)**

**}**

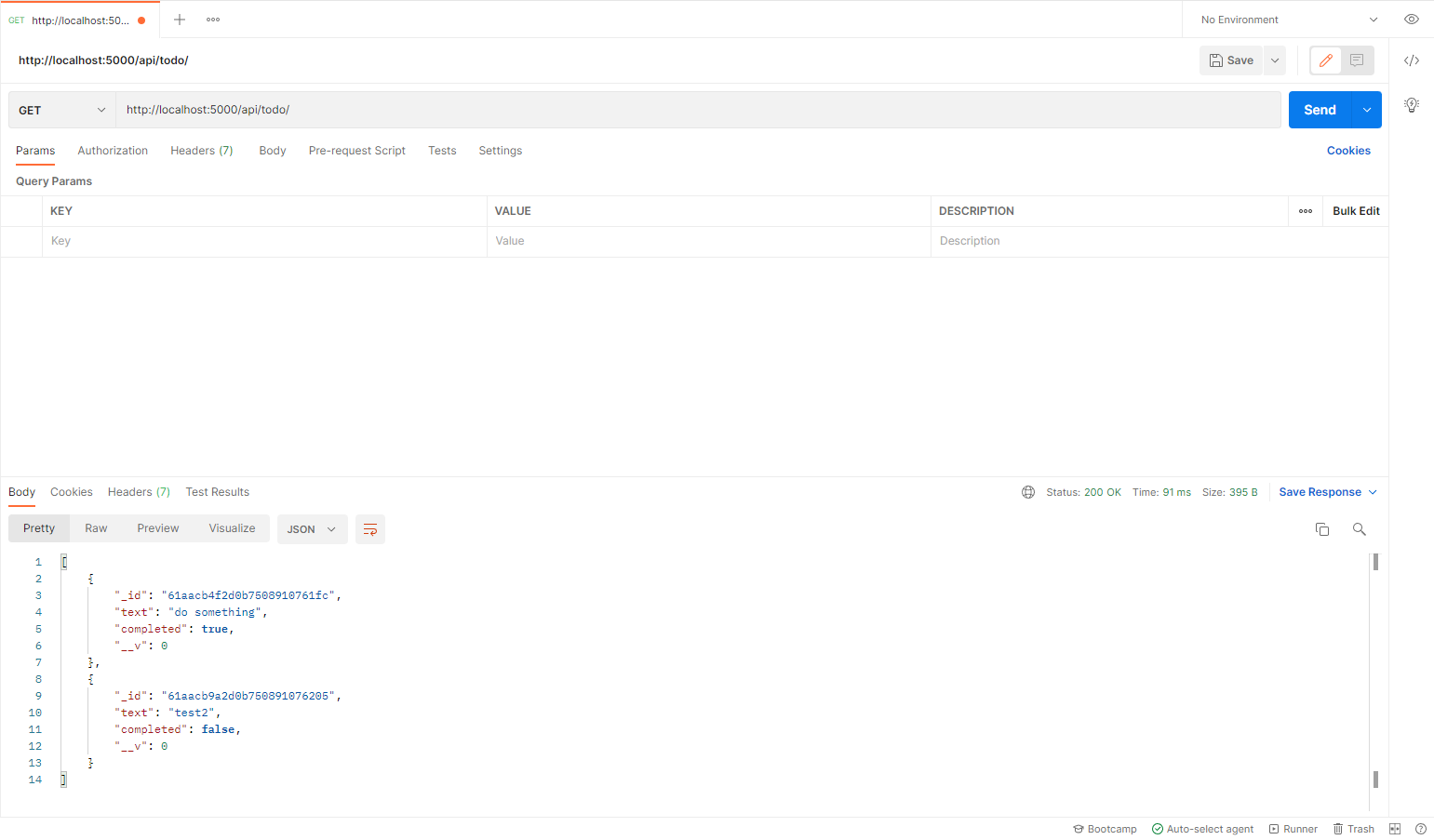
**}**

start**()**

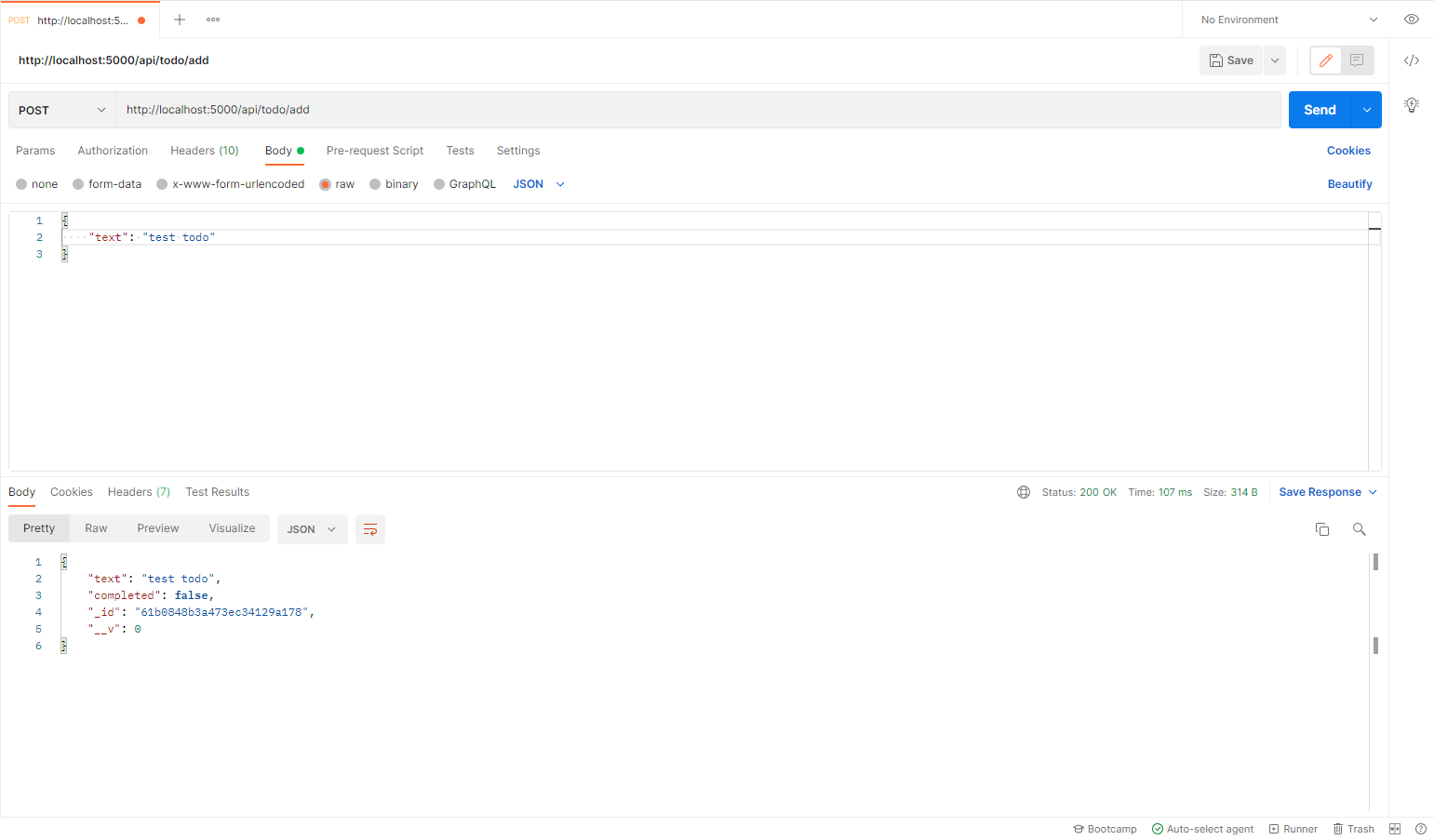
# Приложение Б

# (обязательное)

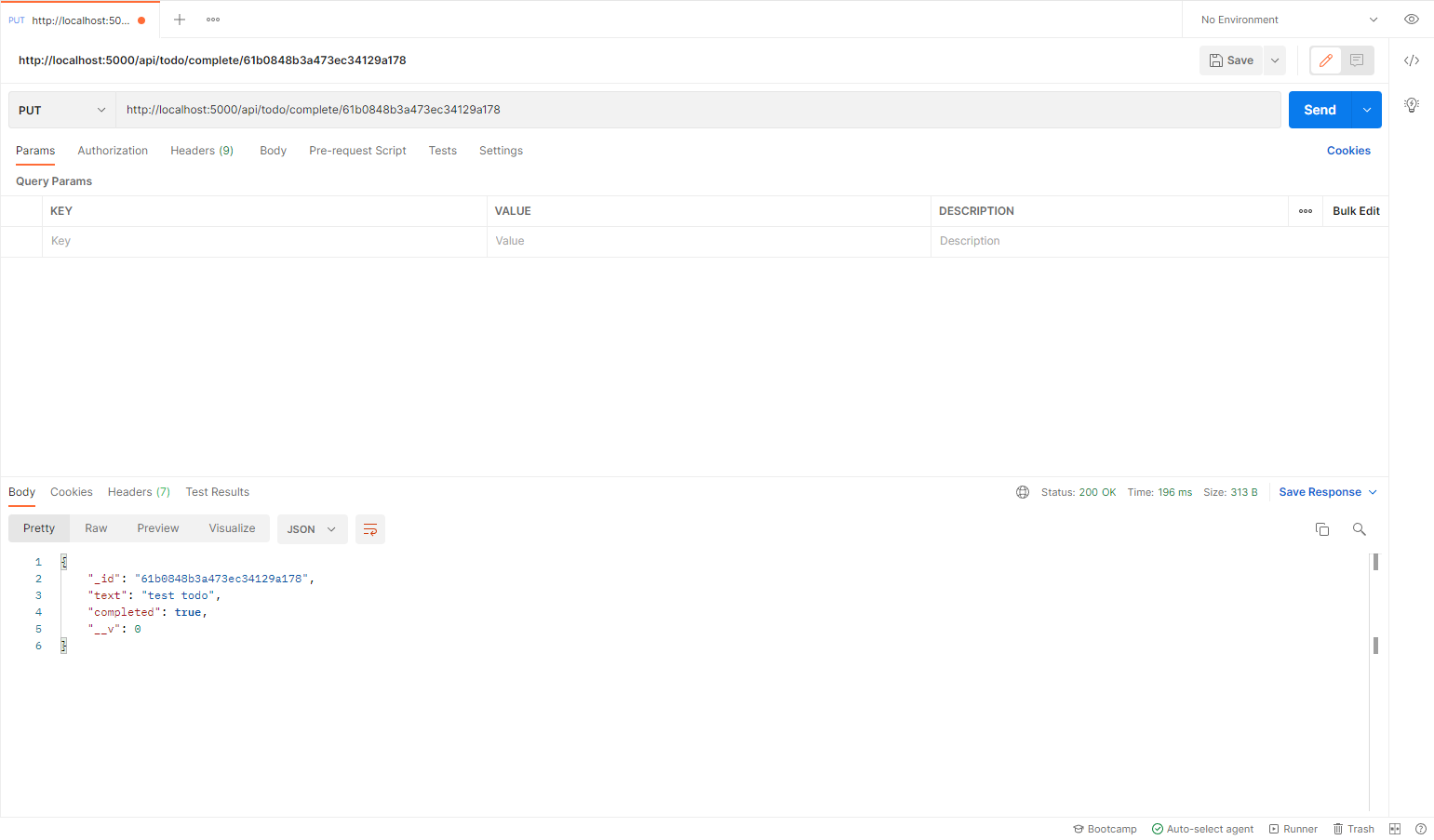
# GET запрос



# POST запрос



# PUT запрос

**DELETE запрос**

