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)

**АВТОРИЗАЦИЯ И РЕГИСТРАЦИЯ**

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 авторизации и регистрации пользователей в интернет приложении

Задачи лабораторной работы следующие:

* Реализация аутентификации через JWT токен в заголовке;
* Ограничит API по доступу для неавторизованных пользователей.

# Основная часть

## Описание стека технологий

Для разработки frontend-приложения было принято решение воспользоваться библиотекой ReactJS для Javascript. React – это инструмент для создания пользовательских интерфейсов. Его главная задача – обеспечение вывода на экран того, что можно видеть на веб-страницах. React значительно облегчает создание интерфейсов благодаря разбиению каждой страницы на небольшие фрагменты. Мы называем эти фрагменты компонентами.

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 WebStorm, так как это бесплатная IDE для студентов с широким функционалом.

## Описание метода решения задачи

Для реализации приложения необходимо реализовать api для регистрации и авторизации пользователя, контроллер для api, и api для работы фронтенда с бекендом.

## Описание фрагментов кода

Для реализации авторизации и регистрации необходимо реализовать routes для пользователя:

const {Router} = require('express')

const router = new Router()

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

const {check, validationResult} = require('express-validator')

const bcrypt = require('bcrypt')

const jwt = require('jsonwebtoken')

router.post('/registration',

    [

        check('email', 'Некорректный email').isEmail(),

        check('password', 'Некорректный пароль').isLength({ min: 6 })

    ],

    async (req, res) => {

        try {

            const errors = validationResult(req)

            if(!errors.isEmpty()) {

                return res.status(400).json({

                    errors: errors.array(),

                    message: "Некорректные данные"

                })

            }

            const { email, password } = req.body

            const isUsed = await User.findOne({ email })

            if (isUsed) {

                return res.status(300).json({ message: 'Данный email уже занят, попробуйте другой.'})

            }

            const hashedPassword = await bcrypt.hash(password, 12)

            const user = new User({

                email, password: hashedPassword

            })

            await user.save()

            res.status(201).json({ message: 'Пользователь создан'})

        }

        catch (error) {

            console.log(error)

        }

})

router.post('/login',

    [

        check('email', 'Некорректный email').isEmail(),

        check('password', 'Некорректный пароль').exists()

    ],

    async (req, res) => {

        try {

            const errors = validationResult(req)

            if(!errors.isEmpty()) {

                return res.status(400).json({

                    errors: errors.array(),

                    message: "Некорректные данные"

                })

            }

            const user = await User.findOne({email: req.body.email})

            if (!user) {

                return res.status(400).json({

                    message: "Такого email нет в базе"

                })

            }

            const isMatch = await bcrypt.compare(req.body.password, user.password)

            if (!isMatch) {

                return res.status(400).json({

                    message: "Неверный пароль"

                })

            }

            const jwtSecret = 'dhf8d7g32g9fg932sfdsf81D90ASSAgf8hdfhaofhia939129138'

            const token = jwt.sign(

                {userId: user.id},

                jwtSecret,

                {expiresIn: '1h'}

            )

            res.json({

                token,

                userId: user.id,

            })

        }

        catch (error) {

            console.log(error)

        }

})

module.exports = router

Здесь выполняется валидация полученных данных. В регистрации после проверки, пароль хэшируется и пользователь сохраняется в базе данных. В авторизации происходит проверка пароля, при удачной авторизации генерируется токен.

На стороне клиента необходимо реализовать функции для регистрации и авторизации пользователя:

const Authpage = () => {

    const [form, setForm] = useState({

        email: '',

        password: ''

    })

    const { login } = useContext(AuthContext)

    const changeHandler = (event) => {

        setForm({...form, [event.target.name]: event.target.value})

    }

    const registerHandler = async (event) => {

        try {

            await axios.post('/api/auth/registration', {...form}, {

                headers: {

                    'Content-Type': 'application/json'

                }

            })

            .then(response => console.log(response))

        } catch (error) {

            console.log(error)

        }

    }

    const loginHandler = async () => {

        try {

            await axios.post('/api/auth/login', {...form}, {

                headers: {

                    'Content-Type': 'application/json'

                }

            })

            .then(response => {

                console.log(response.data)

                login(response.data.token, response.data.userId)

            })

        }

        catch (error) {

            console.error(error)

        }

    }

Также необходимо создать форму для регистрации и входа:

 return (

        <BrowserRouter>

            <Switch>

                <React.Fragment>

                    <div className="container">

                        <div className="auth-page">

                            <Route path="/login">

                                <h3>Авторизация</h3>

                                <form className="form form-login" onSubmit={e => e.preventDefault()}>

                                    <div className="row">

                                        <div className="input-field col s12">

                                            <input type="email" name="email" className="validate" onChange={changeHandler} />

                                            <label htmlFor="email">Email</label>

                                        </div>

                                        <div className="input-field col s12">

                                            <input type="password" name="password" className="validate" onChange={changeHandler} />

                                            <label htmlFor="password">Пароль</label>

                                        </div>

                                    </div>

                                    <div className="row">

                                        <button className="wawes-effect wawes-light btn black" onClick={loginHandler}>Войти</button>

                                        <Link to="/registration" className="btn-outline btn-reg">Нет аккаунта?</Link>

                                    </div>

                                </form>

                            </Route>

                            <Route path="/registration">

                                <h3>Регистрация</h3>

                                <form className="form form-login" onSubmit={e => e.preventDefault()}>

                                    <div className="row">

                                        <div className="input-field col s12">

                                            <input type="email" name="email" className="validate" onChange={changeHandler} />

                                            <label htmlFor="email">Email</label>

                                        </div>

                                        <div className="input-field col s12">

                                            <input type="password" name="password" className="validate" onChange={changeHandler} />

                                            <label htmlFor="password">Пароль</label>

                                        </div>

                                    </div>

                                    <div className="row">

                                        <button className="wawes-effect wawes-light btn black" onClick={registerHandler} >Регистрация</button>

                                        <Link to="/login" className="btn-outline btn-reg">Уже есть аккаунт?</Link>

                                    </div>

                                </form>

                            </Route>

                        </div>

                    </div>

                </React.Fragment>

            </Switch>

        </BrowserRouter>

    )

}

export default Authpage

Листинг кода представлен в приложении А.

## Описание функциональных возможностей разработанного приложения

При запуске приложения, неавторизованный пользователь попадает на страницу авторизации, по нажатию на кнопку «Нет аккаунт» должен быть перенаправлен на страницу регистрации.

При удачной регистрации будет создан новый пользователь.

При удачной авторизации пользователь попадает на страницу со своим списком задач.

По нажатию кнопки «Выйти» пользователь становится неавторизованным и попадает на страницу авторизации.

# Заключение

В практической работе были достигнуты поставленные задачи. Мы добавили регистрацию и авторизацию с jwt токеном.

# Приложения

# Приложение А

# (обязательное)

# Листинг кода

AuthPage.jsx

import React, { useState, useContext } from 'react'

import {BrowserRouter, Switch, Route, Link} from 'react-router-dom'

import axios from 'axios'

import { AuthContext } from '../../context/AuthContext'

import './AuthPage.scss'

const Authpage = () => {

    const [form, setForm] = useState({

        email: '',

        password: ''

    })

    const { login } = useContext(AuthContext)

    const changeHandler = (event) => {

        setForm({...form, [event.target.name]: event.target.value})

    }

    const registerHandler = async (event) => {

        try {

            await axios.post('/api/auth/registration', {...form}, {

                headers: {

                    'Content-Type': 'application/json'

                }

            })

            .then(response => console.log(response))

        } catch (error) {

            console.log(error)

        }

    }

    const loginHandler = async () => {

        try {

            await axios.post('/api/auth/login', {...form}, {

                headers: {

                    'Content-Type': 'application/json'

                }

            })

            .then(response => {

                console.log(response.data)

                login(response.data.token, response.data.userId)

            })

        }

        catch (error) {

            console.error(error)

        }

    }

    return (

        <BrowserRouter>

            <Switch>

                <React.Fragment>

                    <div className="container">

                        <div className="auth-page">

                            <Route path="/login">

                                <h3>Авторизация</h3>

                                <form className="form form-login" onSubmit={e => e.preventDefault()}>

                                    <div className="row">

                                        <div className="input-field col s12">

                                            <input type="email" name="email" className="validate" onChange={changeHandler} />

                                            <label htmlFor="email">Email</label>

                                        </div>

                                        <div className="input-field col s12">

                                            <input type="password" name="password" className="validate" onChange={changeHandler} />

                                            <label htmlFor="password">Пароль</label>

                                        </div>

                                    </div>

                                    <div className="row">

                                        <button className="wawes-effect wawes-light btn black" onClick={loginHandler}>Войти</button>

                                        <Link to="/registration" className="btn-outline btn-reg">Нет аккаунта?</Link>

                                    </div>

                                </form>

                            </Route>

                            <Route path="/registration">

                                <h3>Регистрация</h3>

                                <form className="form form-login" onSubmit={e => e.preventDefault()}>

                                    <div className="row">

                                        <div className="input-field col s12">

                                            <input type="email" name="email" className="validate" onChange={changeHandler} />

                                            <label htmlFor="email">Email</label>

                                        </div>

                                        <div className="input-field col s12">

                                            <input type="password" name="password" className="validate" onChange={changeHandler} />

                                            <label htmlFor="password">Пароль</label>

                                        </div>

                                    </div>

                                    <div className="row">

                                        <button className="wawes-effect wawes-light btn black" onClick={registerHandler} >Регистрация</button>

                                        <Link to="/login" className="btn-outline btn-reg">Уже есть аккаунт?</Link>

                                    </div>

                                </form>

                            </Route>

                        </div>

                    </div>

                </React.Fragment>

            </Switch>

        </BrowserRouter>

    )

}

export default Authpage

auth.route.js

const {Router} = require('express')

const router = new Router()

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

const {check, validationResult} = require('express-validator')

const bcrypt = require('bcrypt')

const jwt = require('jsonwebtoken')

router.post('/registration',

    [

        check('email', 'Некорректный email').isEmail(),

        check('password', 'Некорректный пароль').isLength({ min: 6 })

    ],

    async (req, res) => {

        try {

            const errors = validationResult(req)

            if(!errors.isEmpty()) {

                return res.status(400).json({

                    errors: errors.array(),

                    message: "Некорректные данные"

                })

            }

            const { email, password } = req.body

            const isUsed = await User.findOne({ email })

            if (isUsed) {

                return res.status(300).json({ message: 'Данный email уже занят, попробуйте другой.'})

            }

            const hashedPassword = await bcrypt.hash(password, 12)

            const user = new User({

                email, password: hashedPassword

            })

            await user.save()

            res.status(201).json({ message: 'Пользователь создан'})

        }

        catch (error) {

            console.log(error)

        }

})

router.post('/login',

    [

        check('email', 'Некорректный email').isEmail(),

        check('password', 'Некорректный пароль').exists()

    ],

    async (req, res) => {

        try {

            const errors = validationResult(req)

            if(!errors.isEmpty()) {

                return res.status(400).json({

                    errors: errors.array(),

                    message: "Некорректные данные"

                })

            }

            const user = await User.findOne({email: req.body.email})

            if (!user) {

                return res.status(400).json({

                    message: "Такого email нет в базе"

                })

            }

            const isMatch = await bcrypt.compare(req.body.password, user.password)

            if (!isMatch) {

                return res.status(400).json({

                    message: "Неверный пароль"

                })

            }

            const jwtSecret = 'dhf8d7g32g9fg932sfdsf81D90ASSAgf8hdfhaofhia939129138'

            const token = jwt.sign(

                {userId: user.id},

                jwtSecret,

                {expiresIn: '1h'}

            )

            res.json({

                token,

                userId: user.id,

            })

        }

        catch (error) {

            console.log(error)

        }

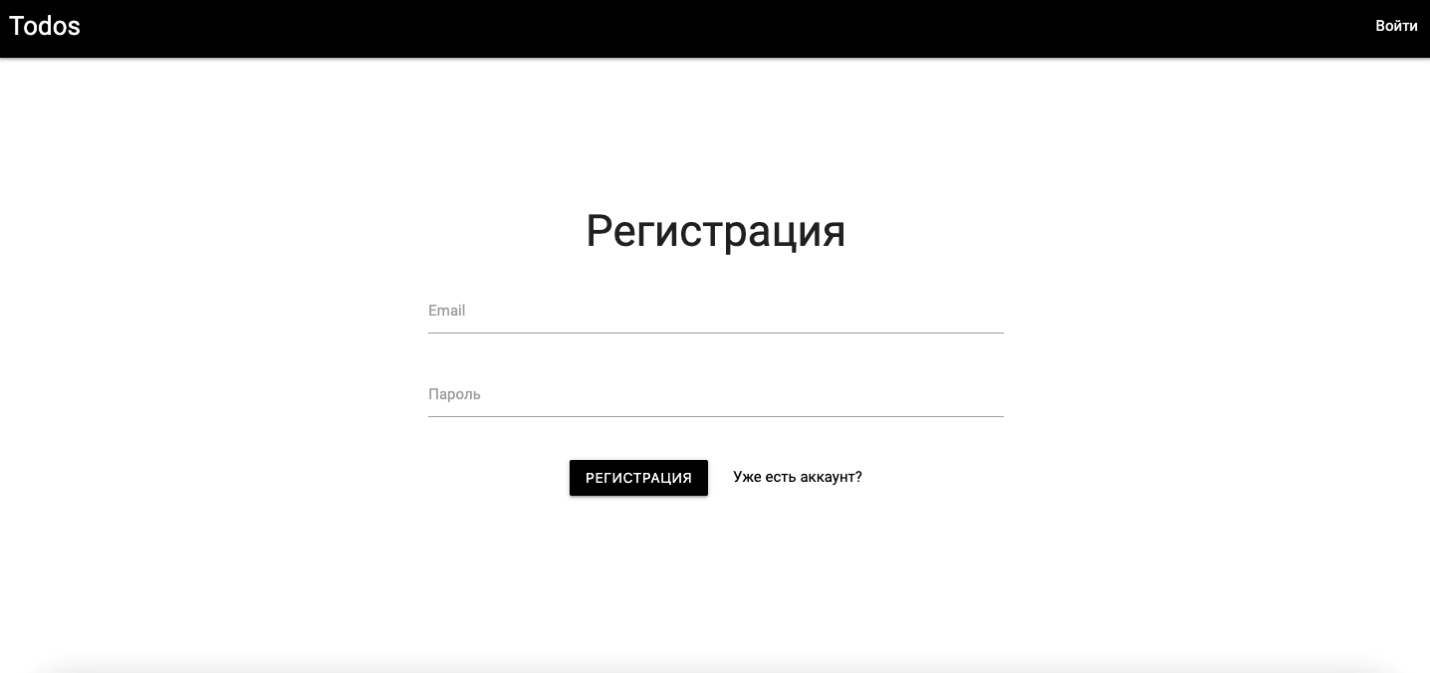
})

module.exports = router

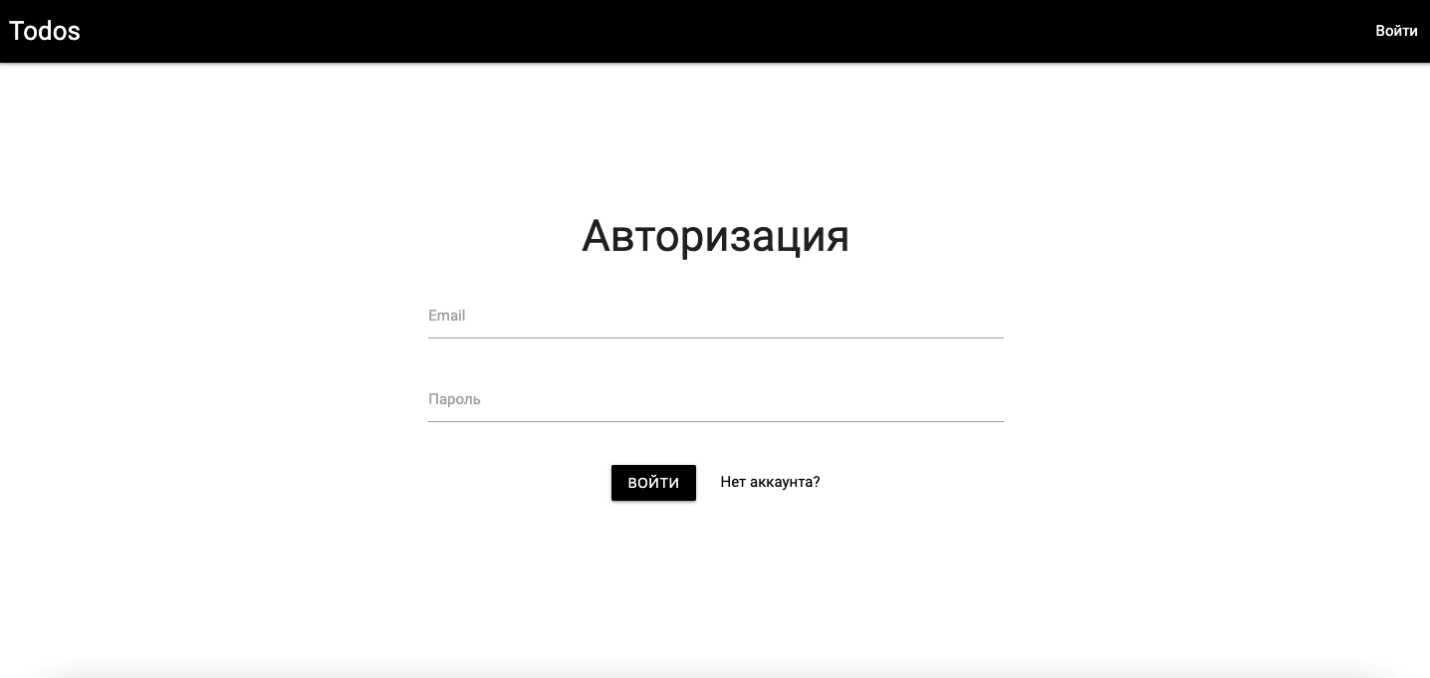
# Приложение Б

# (обязательное)

# Страница ргеистрации



# Страница авторизации



# Страница авторизированного пользователя

