Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ

КАФЕДРА СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ

Домашнее задание

Студент Смыслов Максим Алексеевич
фамилия, имя, отчество

Группа ИУ5-55Б

Студент 18.12.2021 **Смыслов М.А.**
подпись, дата фамилия, и.о.

Преподаватель 18.12.2021 **Гапанюк Ю.Е.**
подпись, дата фамилия, и.о.

2021г.

Описание:
Создал сервис для очередей. Изначально задумка была организовать сдачу лабораторных работ. Сервер был написан на Node.js (express) по принципу REST API. Клиент на React'е, а база на Postgres. Была создана связь многие-ко-многим, с сущностями очередь и юзер. Реализован поиск очередей, их создание и т.д. А также динамическое обновление людей в очереди с помощью Ajax запросов. Также реализована регистрация и авторизация с помощью jwt токена. GitHub: https://github.com/Juwume/Queue

Текст программы:
queueController.js

```
class queueController{
    async getUsers(req, res,next){
        try {
            const {id} = req.query

            let query = await Queue.findAll({

                where:{
                    id
                }
                ,
                include: [{
                    model: User,
                    attributes: ['id','username'],
                    required: true
                }]
            })

            if(query.length == 0){
                query = await Queue.findOne({
                    where:{
                        id
                    }
                })
                return res.json({message: "В очереди никого нет", queueName: query.name, queueDesc: query.description})
            }
            return res.json(query)

        } catch (error) {
            return next(ApiError.internal('Не удалось выполнить запрос'))
        }

    }

   async addToQueue(req,res){
       try {
            const {userId, queueId} = req.body
            console.log(userId + ' ' + queueId)
            const adding = await QueueUser.create({
                queueId,
                userId,
            })
            res.json(adding)
```

```
async searchByName(req, res,next){
    try {
         const {name} = req.query
         let candidate = await Queue.findAll({
             where:{
                 name:{
                     [Op.substring]: name
                 }
             }
         })
         if(candidate.length == 0){
             return res.json({message: "Ничего не найдено"})
         }

         return res.json(candidate)

     } catch (error) {
         return next(ApiError.internal('Не удалось выполнить запрос'))
     }

 }

 async searchById(req, res,next){
     try {
         const {id} = req.query
         let candidate = await Queue.findAll({
             where:{
                 id
             }
         })
         if(!candidate){
             return res.json({message: "Ничего не найдено"})
         }

         return res.json(candidate)

     } catch (error) {
             return next(ApiError.internal('Не удалось выполнить запрос'))
         }



 }

 async createQueue(req, res){
```

userController.js

```
7
8   class UserController{
9       async registration(req, res,next){
10          const {username, email, password, role} = req.body
11          if (!username || !password || !email){
12              return next(ApiError.badRequest('Неправильные данные при регистрации'))
13          }
14
15          let candidate = await User.findOne({where:{email}})
16
17          if (candidate){
18              return next(ApiError.badRequest('Такой Email уже есть в системе'))
19          }
20          candidate = await User.findOne({where:{username}})
21          if (candidate){
22              return next(ApiError.badRequest('Такой username уже есть в системе'))
23          }
24
25          const hashPassword = await bcrypt.hash(password, 5)
26          const newUser = await User.create({username,password:hashPassword,email,role})
27
28          const token = jwt.sign(
29
30              {id:newUser.id, username:newUser.username, role:newUser.role},
31              process.env.SECRET_KEY,
32              {expiresIn: '24h'}
33
34          )
35
36          return res.json({token})
37
38
39      }
40
41      async login(req, res,next){
42
43          const {username, password} = req.body
44
45          let candidate = await User.findOne({where:{username}})
46          if(!candidate){
47              return next(ApiError.internal("Пользователь не найден"))
48          }
49
50          let cmpPass = bcrypt.compareSync(password,candidate.password)
51          if(!cmpPass){
52              return next(ApiError.badRequest("Неправильный пароль"))
```

```
    )

    return res.json({token})

}

async check(req, res){
    const token = jwt.sign(

        {id: req.user.id, username: req.user.username, role: req.user.role},
        process.env.SECRET_KEY,
        {expiresIn: '24h'}

    )
    return res.json({token})
}

async getAllQs(req, res, next){
    try {
        const {id} = req.query

        let query = await User.findAll({
            attributes: ['id','username'],
            where:{
                id
            }
            ,
            include: [{
                model: Queue,
                attributes: ['id','name', 'description'],
                required: true
            }]
        })
        if(query.length == 0){
            res.json({message:"Очередей нет"})
        }
        return res.json(query)

    } catch (error) {
        return next(ApiError.internal('Не удалось выполнить запрос'))
    }

}
async getUser(req, res, next){
    try {
```

Models.js

```
const sequelize = require('../db')

const {DataTypes} = require('sequelize')

const Queue = sequelize.define('queues',{
    id: {type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true},
    name: {type: DataTypes.STRING},
    description: {type: DataTypes.STRING}
})

const User = sequelize.define('user',{
    id: {type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true},
    username: {type: DataTypes.STRING, unique: true},
    password: {type: DataTypes.STRING},
    email: {type: DataTypes.STRING, unique: true},
    role: {type: DataTypes.STRING, defaultValue: 'USER'}
})

const QueueUser = sequelize.define('queue_user',{
    id: {type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true},

})

Queue.belongsToMany(User, {through: QueueUser})
User.belongsToMany(Queue, {through: QueueUser})

module.exports = {
    Queue,
    User,
    QueueUser
}
```

queueRouter.js

```
const Router = require('express')
const queueController = require('../controllers/queueController')
const authMiddleware = require('../middleware/authMiddleware')
const router = new Router()

router.get('/', queueController.getUsers)
router.post('/addToQ',authMiddleware, queueController.addToQueue)
router.post('/leave',authMiddleware, queueController.leaveQueue)
router.post('/create',authMiddleware, queueController.createQueue)
router.post('/delete',authMiddleware, queueController.deleteQueue)
router.get('/searchById',queueController.searchById)
router.get('/searchByName',queueController.searchByName)

module.exports = router
```

userRouter.js

12 lines (10 sloc) | 485 Bytes

```
const Router = require('express')
const router = new Router()
const userController = require('../controllers/userController')
const authMiddleware = require('../middleware/authMiddleware')

router.post('/registration',  userController.registration)
router.post('/login', userController.login)
router.get('/auth', authMiddleware,userController.check)
router.get('/getAllQs',authMiddleware, userController.getAllQs)
router.get('/getUser', userController.getUser)

module.exports = router
```

About.js

```
import { observer } from 'mobx-react-lite'
import React from 'react'
import { Card, Container } from 'react-bootstrap'

const About = observer(() => {
    return (
        <Container>
            <Card className='align-items-center mt-3 mb-2'>

                    <h1 className='mt-2'>О сервисе</h1>
                    <div className='p-4 pt-0 pb-0' style={{fontSize:'20px', textAlign:'justify'}}>
                        <p >
                        <span style={{fontWeight:'bold'}}>Добрый день или вечер!</span>  Вы попали на сайт сервиса с очередями.
                        Для того, чтобы полностью пользоваться его функционалом пожалуйста зарегистрируйтесь или авторизуйтесь.
                        </p>
                        <h4>О функционале</h4>
                        <p>
                        Этот сервис поможет вам и вашим коллегам по несчастью с гордостью отстоять возникшую очередь.
                        Здесь вы можете найти нужную вам очередь и встать в неё.
                        Пожалуйста не забывайте выходить из неё когда закончите.
                        </p>
                        <h4>Примечание</h4>
                        <p>
                        Данный сайт пока находится в далёкой pre alpha версии,
                        и, если вы нашли какие-то недочёты, обязательно свяжитесь со мной.
                        </p>
                        <h4>Контакты</h4>
                        <p>
                            Почта: <span style={{fontStyle:'italic'}}>test@mail.ru</span><br></br>
                            Телефон: <span style={{fontStyle:'italic'}}>+7(999)555-55-55</span><br></br>

                        </p>
                    </div>

            </Card>
        </Container>
```

Create.js

```
import { observer } from 'mobx-react-lite'
import React, {useState} from 'react'
import { Container,Form,Card, Button } from 'react-bootstrap'
import { useHistory } from 'react-router-dom'
import { createQ } from '../http/queueAPI'
import { QUEUE_ROUTE } from '../utils/consts'

const clearInput = (idElem) =>{
    try {
        document.getElementById(idElem).value = ""
    } catch (error) {
       console.log(error)
    }
}

const Create = observer(() => {

    const [name, setName] = useState('')
    const [description, setDescription] = useState('')
    const history = useHistory()

    const checkInputs = () => {

        if (name && description) return false
        else return true
    }

    const create = async () => {
        try {
            const res = await createQ(name,description)
            history.push(QUEUE_ROUTE + `/${res.id}`)
        } catch (error) {
            alert(error.response.data.message)
        }

    }
```

```
return (
    <Container>
        <Card className='align-items-center mt-4 p-4'>
            <Form style={{width:'75%'}}>
                <Form.Label>Название</Form.Label>
                <Form.Control
                    value={name}
                    onChange={str=>setName(str.target.value)}
                    id='nameInput'
                    style={{width:'98.5%'}}
                    placeholder={'Введите название'}
                    className='mb-3'
                />
                <Form.Label>Описание</Form.Label>
                <Form.Control
                    value={description}
                    onChange={str=>setDescription(str.target.value)}
                    id='descriptionInput'
                    style={{width:'98.5%'}}
                    placeholder={'Напишите описание для вашей очереди'}
                    as="textarea"
                    className='mb-3'
                />
                <Container className='d-flex justify-content-end'>
                <Button

                        variant={'outline-secondary'}
                        className='m-2'
                        onClick={() => {
                            clearInput('descriptionInput')
                            clearInput('nameInput')
                        }}
                    >
                        Очистить
                </Button>
                <Button
```

Find.js

```
    const [number, setNumber] = useState('')
    const [foundVisible, setFoundVisible] = useState(false)

    const findByName = async () => {
        try {

            searchByName(name).then(data=>{
                queues.setQueues(data)
                console.log(queues.queues)
            })

        } catch (error) {
            alert(error.response.data.message)
        }
    }
    const findByNumber = async () => {
        try {

            searchById(number).then(data=>{
                queues.setQueues(data)
                console.log(queues.queues)
            })

        } catch (error) {
            alert(error.response.data.message)
        }
    }


    return (
        <Container >
            <Card className='align-items-center mt-4 p-4'>
                <h4 className='ml-0'>Поиск по номеру очереди</h4>
                <Form style={{width:'75%'}}>
                    <Form.Control
                        id='number'
                        value={number}
                        onChange={str=>setNumber(str.target.value)}
                        placeholder='Введите номер'
                        style={{width:'98.5%'}}
                    />
                    <Container className='d-flex justify-content-end'>
                    <Button
```

MyQs.js

```
const MyQs = observer(() => {

    const history = useHistory()
    const {queues,user} = useContext(Context)
    useEffect(() => {
        getQs(user.user.id).then((data)=>{
            try {
                queues.setQueues(data[0].queues)
                console.log(queues.queues)
            } catch (error) {

            }

        })
    }, [])

    return (
        <Container>
            <Card className='p-4 pt-2 mt-3'>
                <h1>Мои очереди</h1>
                {queues.queues.length > 0? queues.queues.map(data=>{
                    return(
                        <Card
                        style={{cursor:'pointer'}}
                        className='m-1'
                        key={data.id}
                        onClick={()=>history.push(QUEUE_ROUTE+ '/' + data.id)}
                        >
                            <h4 className='m-2 mt-0'>{data.name}</h4>
                            <h5 className='m-2 mt-0' style={{font:'italic 20px  Arial', opacity:'0.75'}}>{data.description}</h5>
                        </Card>
                    )
                }): <h3>У тебя нет очередей, попробуй
                    <Button
                        onClick={() => history.push(FIND_ROUTE)}
                    >поискать</Button>
                        </h3>}
            </Card>

        </Container>
    )
})

export default MyQs
```

Profile.js

```
import React, {useContext, useEffect, useState} from 'react'
import { Card, Container } from 'react-bootstrap'
import { useParams } from 'react-router-dom/cjs/react-router-dom.min'
import { Context } from '..'
import { getUser } from '../http/userAPI'

const Profile = () => {
    const {id} = useParams()
    const {curUsers} = useContext(Context)
    const [username, setUsername] = useState()

    useEffect(() => {
       getUser(id).then((data)=>{
        curUsers.setUsers(data)
        setUsername(curUsers.users.username)
        console.log(curUsers.users.username)
       })
    }, [curUsers, id])
    return (
        <Container>
            <Card className='mt-3 p-2'>
                <h1>Профиль пользователя</h1>
                <h2>{username}</h2>

            </Card>
        </Container>
    )
}

export default Profile
```

Queue.js

```
:onst Queue = observer(() => {

    const [nameQueue, setNameQueue] = useState('')
    const [descQueue, setDescQueue] = useState('')
    const {user, curUsers} = useContext(Context)
    const {id} = useParams()
    const history = useHistory()
    const update = async () => {
        let response = await fetch(process.env.REACT_APP_API_URL + 'api/queue?id='+id)

        let parse = await response.json()
       try {
           console.log(parse)
           if(!parse.message){
            document.getElementById('nobody').innerHTML =''
            if(!(parse[0].users.length === curUsers.users.length) || !(parse[0].users.every(function(element, index) {
                return curUsers.users.filter((data)=>{
                    return data.id === element.id
                }).length > 0
            }))) {
                clearInterval(timerId);
                curUsers.setUsers(parse[0].users)
            }
           } else {
               document.getElementById('nobody').innerHTML = "В очереди никого нет, станьте первым!"
               clearInterval(timerId);
                curUsers.setUsers([])
            }
       } catch (error) {
           return
       }




    }

    const getIn = async () => {
        try {
            const res = await addToQ(user.user.id, id)
            await update()
            return res
        } catch (error) {
            alert(error.response.data.message)
        }
```

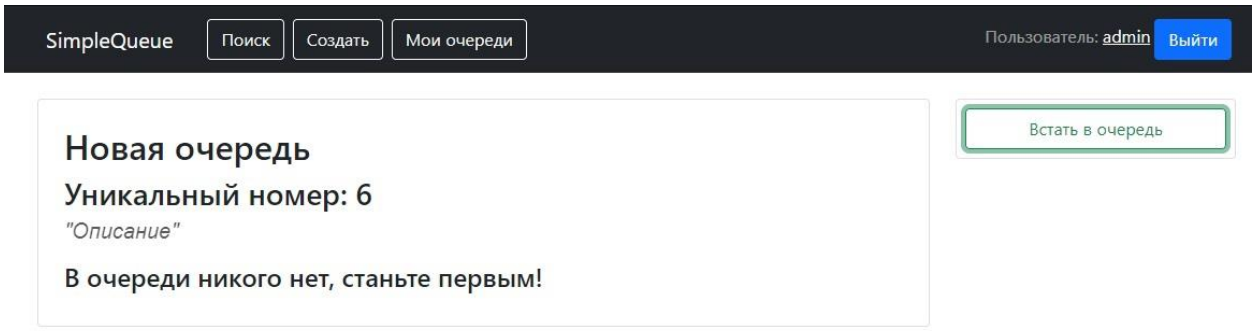
```
        <Row>
            <Col className='mt-4' sm={9}>
                <Card className='p-4'>
                    <h2 >{nameQueue}</h2>
                    <h3>Уникальный номер: {id}</h3>
                    {descQueue? <p style={{font:'italic 20px  Arial', opacity:'0.75'}}>"{descQueue}"</p> : null}

                    {curUsers.users? curUsers.users.map((data,index)=>{
                        return(
                            <Card
                            onClick={()=>history.push(PROFILE_ROUTE+'/'+data.id)}
                            key={index}
                            style={{height:'50px', background:'#212529', color:'white', textAlign:'center', fontSize:'20px', border:'2px solid white', cursor:'pointer'
                            >{`${index+1}. ${data.username}`}</Card>
                        )
                    }) : null}
                    <h4 id='nobody'></h4>
                </Card>
            </Col>
            <Col className='mt-4' sm={3}>
                <Card className='p-2'>
                    {() =>{
                        if(!user.isAuth) return <h4>Вы не можете встать в очередь, пожалуйста авторизуйтесь</h4>
                        else if(curUsers.users.filter((data)=>{
                                return data.id === user.user.id
                            }).length > 0)
                            return(
                                <Button
                                    variant='outline-danger'
                                    onClick={getOut}
                                >
                                    Выйти
                                </Button>
                            )
                            else return(
                                <Button
                                    variant='outline-success'
                                    onClick={getIn}
                                >
                                    Встать в очередь
                                </Button>
                            )
```

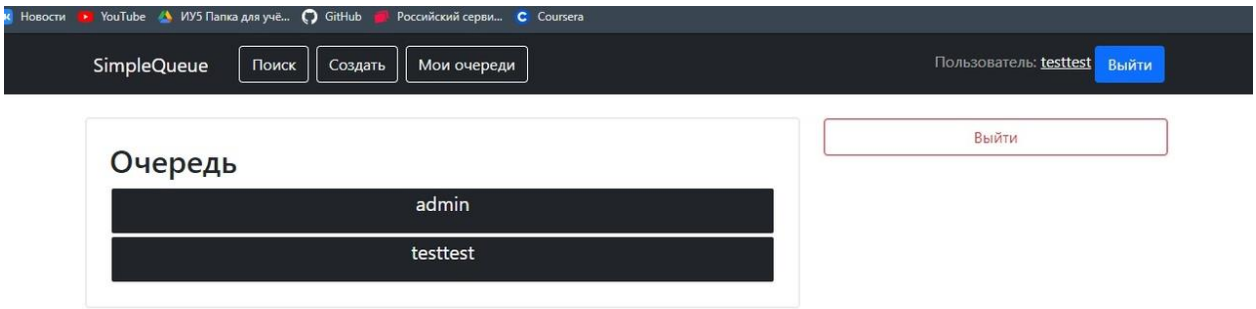
userAPI.js

```
import { $authHost, $host } from "./index"


export const getUsers = async (id) => {
    const {data} = await $host.get('api/queue',{
        params:{
            id
        }
    })
    return data

}
export const searchById = async (id) => {
    const {data} = await $host.get('api/queue/searchById',{
        params:{
            id
        }
    })
    return data

}
export const searchByName = async (name) => {
    const {data} = await $host.get('api/queue/searchByName',{
        params:{
            name
        }
    })
    return data

}

export const addToQ = async (userId, queueId) => {
    const {data} = await $authHost.post('api/queue/addToQ',{
            userId,
            queueId
        })
    return data

}
export const createQ = async (name, description) => {
```

queueAPI.js

```
import { $authHost, $host } from "./index"


export const getUsers = async (id) => {
    const {data} = await $host.get('api/queue',{
        params:{
            id
        }
    })
    return data

}
export const searchById = async (id) => {
    const {data} = await $host.get('api/queue/searchById',{
        params:{
            id
        }
    })
    return data

}
export const searchByName = async (name) => {
    const {data} = await $host.get('api/queue/searchByName',{
        params:{
            name
        }
    })
    return data

}

export const addToQ = async (userId, queueId) => {
    const {data} = await $authHost.post('api/queue/addToQ',{
            userId,
            queueId
        })
    return data

}
export const createQ = async (name, description) => {
```




Новости
YouTube
ИУ5 Папка для учё...
GitHub
Российский серви...
Coursera
SimpleQueue
Поиск
Создать
Мои очереди
Пользователь: testtest
Выйти
Очередь
admin
testtest
Выйти