Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»
(Университет ИТМО)

Факультет **Инфокоммуникационных технологий**

Образовательная программа **Мобильные и сетевые технологии**

Направление подготовки (специальность) **09.03.03 Прикладная информатика**

О Т Ч Е Т

по дисциплине «Проектирование и реализация баз данных»

на тему: РАЗРАБОТКА ИНТЕРФЕЙСОВ ДЛЯ ВЫПОЛНЕНИЯ CRUD-ОПЕРАЦИЙ НАД БАЗОЙ ДАННЫХ

Обучающийся: Олейникова Полина Леонидовна, K32402

Преподаватель: Говорова М. М.

Дата 15.05.2023

Санкт-Петербург 2023

1 Выполнение

Ссылка: https://github.com/OleinikovaPolina/ProjectControl

Стек:

- Node.js
- MongoDB
- Express
- Vue
- …

Запуск сервера, подключение к БД.

```
const express = require("express");
const cors = require("cors");
const mongoose = require('mongoose');

//Server port
const port = 4242;

const app = express();

//Options to parse body
app.use(express.json());
app.use(express.urlencoded({extended: true}));

//Disable CORS
const corsOptions = {
    origin: 'http://localhost:8080',
    credentials: true,
    optionSuccessStatus: 200
}

app.use(cors(corsOptions));

const {authMiddleware} = require('./utils');

//Router import and connection.
app.use('/user', require('./routers/user'));
app.use('/user/check', authMiddleware, require('./routers/userCheck'));
app.use('/participant', authMiddleware, require('./routers/participant'));
app.use('/project', authMiddleware, require('./routers/project'));
app.use('/task', authMiddleware, require('./routers/task'));
app.use('/application', authMiddleware, require('./routers/application'));
app.use('/skill', require('./routers/skill'));
app.use('/role', authMiddleware, require('./routers/role'));

app.listen(port, () => console.log(`Server is on ${port}`));

//Database connection (You need to start DB before connection by CMD or
MongoDBCompass)
mongoose.connect(
    'mongodb://localhost:27017/project_control',
    {useNewUrlParser: true, useUnifiedTopology: true},
    (error => console.log(`MongoDB connection ${error ? 'error' :
```

```
'success'}`))
);
```

Запросы получения задачи, создания, изменения, удаления.

```
const {Router} = require('express');
const TaskModel = require('../models/task');
const ProjectModel = require('../models/project');
const router = Router();

router.get('/', (req, res) => {
    const {id: project} = req.query;

    return TaskModel
        .find({project})
        .populate({
            path: 'participants',
            select: '_id roles participant',
            populate: [{path: 'roles', select: 'name'}, {path: 'participant',
select: 'name'}]
        })
        .populate({path: 'project', select: 'leader'})
        .then((data) => {
            return res.send(data.map(({
                                          _id,
                                          title,
                                          short_description,
                                          description,
                                          participants,
                                          deadline,
                                          times,
                                          type_task,
                                          project,
                                          createdAt
                                      }) => ({
                id: _id,
                title,
                short_description,
                description,
                participants: participants.map(
                    ({_id, roles, participant}) => ({
                        id: _id,
                        roles,
                        participant: {id: participant.id, name:
participant.name}
                    })
                ),
                deadline,
                times,
                type_task,
                project: {leader: project.leader.toString()},
                createdAt
            })))
        })
        .catch(() => {
            return res.sendStatus(400);
        })
})

router.post('/', (req, res) => {
    const {title, short_description, description, participants, deadline,
project} = req.body;
```

```
    const {id: userId} = req.user;

    return ProjectModel
        .findById({_id: project})
        .populate({
            path: 'leader',
            match: userId
        })
        .then((data) => {
            if (data.leader) return TaskModel
                .create({
                    title,
                    short_description,
                    description,
                    participants,
                    deadline,
                    times: [],
                    project,
                    type_task:0
                })
                .then(() => {
                    return res.sendStatus(200)
                })
                .catch(() => {
                    return res.sendStatus(400);
                })
        })
        .catch(() => {
            return res.sendStatus(400);
        })
})

router.post('/update', (req, res) => {
    const {id: _id, title, short_description, description, participants,
deadline,  type_task} = req.body;
    const {id: userId} = req.user;
    return TaskModel
        .findById({_id})
        .populate({path: 'project', populate: {path: 'leader', match: {_id:
userId}, select: 'name'}})
        .populate({path: 'participants', populate:{path: 'participant',
match: {_id: userId}}})
        .then((data) => {
            if (data.project.leader || data.participants.length > 0) {
                let times2 = data.times
                if (data.type_task !== type_task) {
                    times2.push(new Date())
                }
                return TaskModel
                    .findByIdAndUpdate({_id}, {
                        title,
                        short_description,
                        description,
                        participants,
                        deadline,
                        times: times2,
                        type_task
                    })
                    .then((data) => {
                        if (!data) return res.sendStatus(400);
                        return res.sendStatus(200)
                    })
                    .catch(() => {
```

```
                        return res.sendStatus(400);
                    })
            }
            return res.sendStatus(400);
        })
        .catch(() => {
            return res.sendStatus(400);
        })
})

router.delete('/', async (req, res) => {
    const {id: _id} = req.query;
    const {id: userId} = req.user;
    return TaskModel
        .findById({_id})
        .populate({path: 'participants', match: {_id: userId}})
        .populate({
            path: 'project',
            populate: {path: 'leader', match: {_id: userId}}
        })
        .then((data) => {
            if (data.project.leader || data.participants)
                return TaskModel
                    .findByIdAndDelete({_id})
                    .then((data) => {
                        if (!data) {
                            return res.sendStatus(400)
                        }
                        return res.send({id:data._id})
                    })
                    .catch(() => {
                        return res.sendStatus(400);
                    })
        })
        .catch(() => {
            return res.sendStatus(400);
        })
})

module.exports = router;
```

Отправка запросов на сервер.

```
import { instance } from '@/utils/api'

export default class TaskService {
  /**
   * Function to get all tasks of the project
   * Authentication is required
   * @param payload
   * @return Promise
   * @example TaskService.getTasks({ id:'Project Id'})
   */
  static async getTasks(payload: { id: string }) {
    return await instance.get('task', { params: payload }).then((response) =>
{
      if (200 <= response.status && response.status < 300) {
        return response.data
      }
      return Promise.reject(response.status)
    })
  }

  /**
```

```
  * Add Task function
  * Authentication is required
  * @param payload
  * @return Promise
  * @example TaskService.addTask({
  *  title: 'Title',
  *  short_description: 'Short description',
  *  description: 'Description',
  *  participants: ["User Id"],
  *  deadline: Date,
  *  project: 'Project Id',
  *  type_task: 0,
  * })
  */
 static async addTask(payload: {
   title?: string;
   short_description?: string;
   description?: string;
   participants?: Array<string>;
   deadline?: Date|string;
   project?: string;
 }) {
   return await instance
     .post('task', JSON.stringify(payload))
     .then((response) => {
       if (200 <= response.status && response.status < 300) {
         return response.data
       }
       return Promise.reject(response.status)
     })
 }

 /**
  * Change Task function
  * Authentication is required
  * @param payload
  * @return Promise
  * @example TaskService.changeTask({
  *  title: 'Title',
  *  short_description: 'Short description',
  *  description: 'Description',
  *  participants: ["User Id"],
  *  deadline: Date,
  *  type_task: 0,
  * })
  */
 static async changeTask(payload: {
   id?: string;
   title?: string;
   short_description?: string;
   description?: string;
   participants?: Array<string>;
   deadline?: Date|string;
   type_task?: number;
 }) {
   return await instance
     .post('task/update', JSON.stringify(payload))
     .then((response) => {
       if (200 <= response.status && response.status < 300) {
         return response.data
       }
       return Promise.reject(response.status)
     })
```

```
  }

  /**
   * Delete task function
   * Authentication is required
   * @param payload
   * @return Promise
   * @example TaskService.deleteTask({id:'Participant Id'})
   */
  static async deleteTask(payload: { id: string }) {
    return await instance
      .delete('task', { params: payload})
      .then((response) => {
        if (200 <= response.status && response.status < 300) {
          return response.data
        }
        return Promise.reject(response.status)
      })
  }
}
```

Вывод задачи и удаление.

```
<template>
  <v-dialog
    v-if="task"
    v-model="dialog"
    persistent
    max-width="600px"
  >
    <v-card>
      <v-card-title
        :style="{borderBottom: '3px solid '+
getColor(titles[task.type_task])}"
      >
        <v-sheet
          class="rounded mr-1"
          :color="getColor(titles[task.type_task])"
          width="15"
          height="15"
        />
        <span>{{ task.title }}</span>
        <v-spacer />
        <v-btn
          icon
          @click="changeDialog(false)"
        >
          <v-icon>mdi-close</v-icon>
        </v-btn>
      </v-card-title>
      <v-card-text
        class="pt-3 pb-1"
        v-html="task.description"
      />
      <v-card-text class="pb-1">
        <v-list-item
          class="pa-0"
        >
          <v-list-item-avatar
            dark
            class="grey lighten-1"
            size="36"
          >
            <v-icon>mdi-clock-outline</v-icon>
```

```
          </v-list-item-avatar>

          <v-list-item-content>
            <v-list-item-subtitle v-if="task.createdAt">
              <v-icon small>
                mdi-calendar-arrow-right
              </v-icon>
              <span>{{ $moment(task.createdAt).format("DD.MM.YYYY") }}</span>
            </v-list-item-subtitle>
            <v-list-item-subtitle v-if="task.deadline">
              <v-icon small>
                mdi-calendar-alert
              </v-icon>
              <span>{{ $moment(task.deadline).format("DD.MM.YYYY") }}</span>
            </v-list-item-subtitle>
          </v-list-item-content>
        </v-list-item>
      </v-card-text>
      <v-card-title
        v-if="task.participants && task.participants.length>0"
        class="font-weight-regular"
      >
        Participants
      </v-card-title>
      <v-card-text v-if="task.participants && task.participants.length>0">
        <v-list-item
          v-for="participant in task.participants"
          :key="participant.id"
          class="pa-0"
        >
          <v-list-item-avatar>
            <AvatarBase
              size="36"
              :name="participant.participant.name"
            />
          </v-list-item-avatar>

          <v-list-item-content>
            <v-list-item-title>
              <router-link
                style="color: inherit"
                class="text-decoration-none"
                :to="'/profile/'+participant.participant.id"
              >
                {{ participant.participant.name }}
              </router-link>
            </v-list-item-title>
            <v-list-item-subtitle>
              {{ participant.roles.map(x => x.name).join(", ") }}
            </v-list-item-subtitle>
          </v-list-item-content>
        </v-list-item>
      </v-card-text>
      <v-card-actions>
        <v-spacer />
        <v-btn
          v-if="status"
          text
          @click="deleteTask"
        >
          Delete
        </v-btn>
        <v-btn
```

```
          color="blue darken-1"
          text
          @click="changeDialog(false)"
        >
          Close
        </v-btn>
        <v-btn
          v-if="status"
          text
          color="blue darken-1"
          @click="changeDialogForm(true)"
        >
          Change
        </v-btn>
      </v-card-actions>
    </v-card>
  </v-dialog>
</template>

<script lang="ts">
import { Component, Emit, Mixins, Prop } from 'vue-property-decorator'
import { TaskInterface } from '@/types'
import AvatarMixin from '@/mixins/AvatarMixin'
import { taskModule } from '@/store'


@Component({
  name: 'DialogTask',
  components: { AvatarBase:()=>import('@/components/Common/BaseAvatar.vue') }
})
export default class DialogTask extends Mixins(AvatarMixin) {
  @Prop() dialog!: boolean
  @Prop() task!: TaskInterface
  @Prop() titles!: boolean
  @Prop() status!:boolean

  @Emit()
  changeDialog(val: boolean) {
    return val
  }
  @Emit()
  changeDialogForm(val: boolean) {
    return val
  }

  private async deleteTask() {
    await taskModule.actions.taskDelete({id:this.task.id})
    this.changeDialog(false)
  }
}
</script>
```

Форма для создания\изменения задачи.

```
<template>
  <div>
    <v-dialog
      v-model="dialog"
      persistent
      max-width="600px"
    >
      <v-card>
        <v-card-title class="text-h6 font-weight-regular justify-space-
between">
```

```
          <span>{{ form.id ? "Change" : "Create" }} task</span>
          <v-spacer />
          <v-btn
            icon
            @click="changeDialog(false)"
          >
            <v-icon>mdi-close</v-icon>
          </v-btn>
        </v-card-title>
        <v-form
          ref="form"
          @submit.prevent="submit"
        >
          <v-card-text>
            <v-text-field
              v-model="form.title"
              :rules="rules.string"
              label="Title"
              required
              outlined
              dense
            />
            <v-text-field
              v-model="form.short_description"
              :rules="rules.string"
              label="Short description"
              required
              outlined
              dense
            />
            <v-autocomplete
              v-model="form.participants"
              :rules="rules.array"
              :items="participants"
              item-value="id"
              item-text="participant.name"
              label="Participants"
              outlined
              :dense="!($vuetify.breakpoint.xs ||$vuetify.breakpoint.sm)"
              small-chips
              multiple
              no-data-text="The project has no participants"
            >
              <template #item="data">
                <v-list-item-content>
                  <v-list-item-title>{{ data.item.participant.name }}</v-
list-item-title>
                  <v-list-item-subtitle style="height: 100%">
                    {{ data.item.roles.map(x => x.name).join(", ") }}
                  </v-list-item-subtitle>
                </v-list-item-content>
              </template>
            </v-autocomplete>
            <v-checkbox
              v-model="checkbox"
              class="my-0"
              label="Deadline"
              :hide-details="!!form.deadline && checkbox"
              @click="checkboxChanged"
            />
            <div
              v-if="checkbox&&form.deadline"
              class="mb-3 mt-1 d-flex align-center"
```

```
            >
              <div
                class="text-subtitle-1"
              >
                {{ $moment(form.deadline).format("DD.MM.YYYY") }}
              </div>
              <v-btn
                icon
                @click="dialog2=true"
              >
                <v-icon>mdi-square-edit-outline</v-icon>
              </v-btn>
            </div>
            <div class="text--secondary">
              Description
            </div>
            <VueEditor
              v-model="form.description"
              :editor-toolbar="customToolbar"
            />
          </v-card-text>
          <v-card-actions>
            <v-spacer />
            <v-btn
              text
              @click="resetForm"
            >
              {{ form.id ? "Revoke" : "Clear" }}
            </v-btn>
            <v-btn
              color="blue darken-1"
              text
              @click="changeDialog(false)"
            >
              Close
            </v-btn>
            <v-btn
              :disabled="!formIsValid()"
              text
              color="primary"
              type="submit"
            >
              {{ form.id ? "Change" : "Add" }}
            </v-btn>
          </v-card-actions>
        </v-form>
      </v-card>
    </v-dialog>
    <v-dialog
      v-model="dialog2"
      max-width="300px"
    >
      <v-date-picker
        v-if="checkbox"
        v-model="form.deadline"
        :min="task&&task.id?
          ($moment(task.deadline).format('YYYY-MM-DD')):
          ($moment().format('YYYY-MM-DD')) "
      />
    </v-dialog>
  </div>
</template>
```

```
<script lang="ts">
import { Component, Emit, Mixins, Prop, Watch } from 'vue-property-decorator'
import { VueEditor } from 'vue2-editor'
import { participantMapper, participantModule, taskModule } from '@/store'
import { TaskInterface } from '@/types'
import FormMixin from '@/mixins/FormMixin'

interface FormInterface {
  id?: string;
  project: string;
  title?: string;
  short_description?: string;
  description?: string;
  participants?: Array<string>;
  deadline?: Date | string;
}


@Component({
  name: 'DialogTaskForm',
  components: { VueEditor },
  computed: {
    ...participantMapper.mapState({ participants: 'participants' })
  }
})
export default class DialogTaskForm extends Mixins(FormMixin) {
  @Prop() dialog!: boolean
  @Prop() task!: TaskInterface

  private dialog2 = false
  private checkbox = false
  private defaultForm: FormInterface = {
    project: this.$route.params.id.toString(),
    title: '',
    short_description: '',
    description: '',
    participants: []
  }
  private form: FormInterface = Object.assign({}, this.defaultForm)

  async mounted() {
    await participantModule.actions.participantGetForProject({ id:
this.form.project })
  }

  private checkboxChanged() {
    if (this.checkbox) {
      this.dialog2 = true
    } else {
      this.form.deadline = undefined
    }
  }

  @Watch('dialog')
  dialogChanged(newVal: boolean) {
    if (newVal) {
      this.resetForm()
    }
  }

  @Watch('dialog2')
  dialog2Changed(newVal: boolean) {
    if (!newVal && !this.form.deadline) {
```

```
      this.checkbox = false
    }
  }

  @Emit()
  changeDialog(val: boolean) {
    return val
  }

  @Emit()
  changeTask(val: TaskInterface) {
    return val
  }

  private formIsValid() {
    return (
      this.form.title &&
      this.form.short_description &&
      this.form.description &&
      this.form.participants &&
      this.form.participants.length > 0
    )
  }

  private resetForm() {
    if (this.task) {
      this.form = {
        id: this.task.id,
        project: this.$route.params.id.toString(),
        title: this.task.title,
        short_description: this.task.short_description,
        description: this.task.description,
        participants: this.task.participants?.map(x => x.id),
        deadline: this.$moment(this.task.deadline).format('YYYY-MM-DD')
      }
      if (this.task.deadline) {
        this.checkbox = true
      }
    } else {
      this.form = Object.assign({}, this.defaultForm)
      if (this.$refs.form) {
        this.$refs.form.reset()
      }
    }
  }

  private async submit() {
    if (!this.form.id) {
      await taskModule.actions.taskCreate(this.form)
    } else {
      await taskModule.actions.taskChange({ data: this.form, project:
this.$route.params.id })
      this.changeTask(this.task)
    }
    this.changeDialog(false)
  }
}
</script>
```

ЗАКЛЮЧЕНИЕ

В данной работе я создала интерфейс для выполнения CRUD операция для задач.