Frontend-часть

|  |
| --- |
| auth/api/group.js |
| import ky from "ky";  import { API\_URL } from "@/constants";  const groupAPI = ky.create({    prefixUrl: `${API\_URL}/groups`,  });  export async function getGroupNames() {    const response = await groupAPI      .get("names")      .then((response) => response)      .catch((error) => console.log(error));    return response.json();  } |

|  |
| --- |
| auth/models/newUser.js Остальные модели реализованы аналогично |
| export default class User {    static get fields() {      return {        login: null,        password: null,        Curator: {          surname: null,          name: null,          patronymic: null,        },        Group: {          groupNumber: null,          groupName: null,        },      };    }  } |

|  |
| --- |
| auth/services/InMemoryJWT.js |
| import { AuthClient } from "../stores/AuthContext.js";  const inMemoryJWTService = () => {    let inMemoryJWT = null;    let refreshTimeoutId;    const refreshToken = (expiration) => {      const timeoutTrigger = expiration - 10000;      refreshTimeoutId = setTimeout(() => {        AuthClient.post("/refresh")          .then((res) => {            const { accessToken, accessTokenExpiration } = res.data;            setToken(accessToken, accessTokenExpiration);          })          .catch(console.error);      }, timeoutTrigger);    };    const abortRefreshToken = () => {      if (refreshTimeoutId) {        clearInterval(refreshTimeoutId);      }    };    const getToken = () => inMemoryJWT;    const setToken = (token, tokenExpiration) => {      inMemoryJWT = token;      refreshToken(tokenExpiration);    };    const deleteToken = () => {      inMemoryJWT = null;      abortRefreshToken();    };    return { getToken, setToken, deleteToken };  };  export default inMemoryJWTService(); |

|  |
| --- |
| auth/stores/AuthContext.js |
| import { defineStore } from "pinia";  import { useHomeStore } from "./Home.js";  import axios from "axios";  import { API\_URL } from "@/constants.js";  import InMemoryJWT from "../services/InMemoryJWT.js";  import { onMounted, ref } from "vue";  import { useRouter } from "vue-router";  import { useGroupListStore } from "@/views/GroupList/stores/groupList.js";  import { useToastStore } from "@/stores/toasts.js";  export const AuthClient = axios.create({    baseURL: `${API\_URL}/auth`,    withCredentials: true,  });  export const ResourceClient = axios.create({    baseURL: `${API\_URL}/resource`,    withCredentials: true,  });  ResourceClient.interceptors.request.use(    (config) => {      const accessToken = InMemoryJWT.getToken();      if (accessToken) {        config.headers["Authorization"] = `Bearer ${InMemoryJWT.getToken()}`;      }      return config;    },    (error) => {      Promise.reject(error);    }  );  export const useAuthProvider = defineStore("AuthStore", () => {    const { setData } = useHomeStore();    const { fetchStudents } = useGroupListStore();    const { successToast, warningToast } = useToastStore();    const router = useRouter();    const isAppReady = ref(false);    const setIsAppReady = (value) => {      isAppReady.value = value;    };    const isUserLogged = ref(false);    const setIsUserLogged = (value) => {      isUserLogged.value = value;    };    const handleFetchProtected = async () => {      await ResourceClient.get("/protected", {        headers: { Authorization: `Bearer ${InMemoryJWT.getToken()}` },      })        .then((res) => {          setData(res.data);        })        .catch((err) => {          console.log(err);        });    };    const handleLogOut = () => {      AuthClient.post("/logout")        .then(() => {          InMemoryJWT.deleteToken();          setIsUserLogged(false);          router.push({ name: "login" });        })        .catch((err) => {          console.log(err);        });    };    const handleSignUp = (data) => {      AuthClient.post("/sign-up", data)        .then((res) => {          const { accessToken, accessTokenExpiration } = res.data;          InMemoryJWT.setToken(accessToken, accessTokenExpiration);          setIsUserLogged(true);          router.push({ name: "Students" });        })        .catch((error) => {          warningToast(error.response.data.error);        });    };    const handleSignIn = (data) => {      AuthClient.post("/sign-in", data)        .then((res) => {          const { accessToken, accessTokenExpiration } = res.data;          InMemoryJWT.setToken(accessToken, accessTokenExpiration);          setIsUserLogged(true);          router.push({ name: "Students" });          fetchStudents();        })        .catch((error) => {          warningToast(error.response.data.error);        });    };    const refresh = async () => {      await AuthClient.post("/refresh")        .then((res) => {          const { accessToken, accessTokenExpiration } = res.data;          InMemoryJWT.setToken(accessToken, accessTokenExpiration);          setIsAppReady(true);          setIsUserLogged(true);          fetchStudents();        })        .catch(() => {          setIsAppReady(true);          setIsUserLogged(false);          router.push({ name: "login" });        });    };    onMounted(async () => {      await refresh();      await handleFetchProtected();    });    return {      handleFetchProtected,      handleLogOut,      handleSignUp,      handleSignIn,      refresh,      isUserLogged,      isAppReady,    };  }); |

|  |
| --- |
| auth/stores/Home.js |
| import { defineStore } from "pinia";  import { ref } from "vue";  export const useHomeStore = defineStore("HomeStore", () => {    const data = ref();    const curator = ref();    const group = ref();    function setData(responseData) {      data.value = responseData;      curator.value =        responseData.curator.surname + " " + responseData.curator.name;      group.value =        responseData.group.groupNumber + " " + responseData.group.groupName;    }    return { data, setData, curator, group };  }); |

|  |
| --- |
| routers/mainRouter.js. Остальные роутеры реализованы аналогично |
| import { createRouter, createWebHistory } from "vue-router";  import GroupListView from "@/views/GroupList/GroupListView.vue";  import StartView from "@/views/Start/StartView.vue";  import SocialPassportView from "@/views/SocialPassport/SocialPassportView.vue";  import ParentMeetingsView from "@/views/ParentMeetings/ParentMeetingsView.vue";  import GroupMeetingsView from "@/views/GroupMeetings/GroupMeetingsView.vue";  import HomeroomsView from "@/views/Homerooms/HomeroomsView.vue";  import ProfileView from "@/views/Profile/ProfileView.vue";  import profileRouter from "./profileRouter.js";  import socialPassportRouter from "./socialPassportRouter.js";  import HomeView from "@/auth/views/HomeView.vue";  import RegisterForm from "@/auth/views/RegisterForm.vue";  import LoginForm from "@/auth/views/LoginForm.vue";  const routes = [    {      path: "/",      component: StartView,    },    {      path: "/home",      name: "home",      component: HomeView,    },    {      path: "/sign-up",      name: "register",      component: RegisterForm,    },    {      path: "/sign-in",      name: "login",      component: LoginForm,    },    {      path: "/students",      name: "Students",      component: GroupListView,    },    {      path: "/social-passport",      name: "SocialPassport",      component: SocialPassportView,      children: socialPassportRouter,    },    {      path: "/parent-meetings",      name: "ParentMeetings",      component: ParentMeetingsView,    },    {      path: "/group-meetings",      name: "GroupMeetings",      component: GroupMeetingsView,    },    {      path: "/homerooms",      name: "Homerooms",      component: HomeroomsView,    },    {      path: "/profile/:id(\\d+)",      name: "Profile",      component: ProfileView,      redirect: { name: "GeneralInformation" },      children: profileRouter,    },  ];  const router = createRouter({    history: createWebHistory(import.meta.env.BASE\_URL),    routes,  });  export default router; |

|  |
| --- |
| stores/confirms.js |
| import { defineStore } from "pinia";  import { useConfirm } from "primevue/useconfirm";  import { useToastStore } from "./toasts";  export const useConfirmStore = defineStore("confirms", () => {    const { successToast, infoToast, warningToast } = useToastStore();    const confirm = useConfirm();    async function confirmAdd({ funcIf, funcElse = () => {}, invalid }) {      if (!invalid) {        funcIf();        successToast("Данные были успешно добавлены");      } else {        warningToast("Заполните все обязательные поля");        funcElse();      }    }    async function confirmEdit({ funcAccept, funcReject, invalid }) {      confirm.require({        message: "Сохранить изменения?",        header: "Подтверждение",        icon: "pi pi-info-circle",        rejectLabel: "Отмена",        acceptLabel: "Изменить",        rejectClass: "p-button-secondary p-button-outlined",        acceptClass: "p-button-info",        accept: () => {          if (!invalid) {            funcAccept();            infoToast("Изменения были успешно сохранены");          } else {            warningToast("Заполните все обязательные поля");          }        },        reject: funcReject,      });    }    async function confirmDelete({ funcAccept, funcReject = () => {} }) {      confirm.require({        message: "Вы точно хотите удалить?",        header: "Подтверждение",        icon: "pi pi-info-circle",        rejectLabel: "Отмена",        acceptLabel: "Удалить",        rejectClass: "p-button-secondary p-button-outlined",        acceptClass: "p-button-danger",        accept: () => {          funcAccept();          successToast("Удаление прошло успешно");        },        reject: funcReject,      });    }    return { confirmEdit, confirmDelete, confirmAdd };  }); |

|  |
| --- |
| stores/search.js |
| import { defineStore } from "pinia";  import { FilterMatchMode } from "primevue/api";  import { ref } from "vue";  export const useSearchStore = defineStore("search", () => {    const filters = ref({      global: { value: null, matchMode: FilterMatchMode.CONTAINS },    });    return { filters };  }); |

|  |
| --- |
| stores/toasts.js |
| import { defineStore } from "pinia";  import { computed, ref } from "vue";  import { useToast } from "primevue/usetoast";  export const useToastStore = defineStore("toasts", () => {    const toast = useToast();    const TOAST\_LIFE = 2000;    function successToast(message) {      toast.add({        severity: "success",        summary: "Успех",        detail: message,        life: TOAST\_LIFE,      });    }    function infoToast(message) {      toast.add({        severity: "info",        summary: "Инфо",        detail: message,        life: TOAST\_LIFE,      });    }    function warningToast(message) {      toast.add({        severity: "warn",        summary: "Предупреждение",        detail: message,        life: TOAST\_LIFE,      });    }    return { successToast, infoToast, warningToast };  }); |

|  |
| --- |
| utils/dialog.js |
| import { ACTIONS } from "@/constants";  export default class DialogForm {    action = ACTIONS.VIEW;    isShowDialog = false;    formHeaders = { add: "", edit: "", view: "" };    currentHeader = "";    constructor(formHeaders) {      this.formHeaders = formHeaders;      this.currentHeader = formHeaders.view;    }    get isShowDialog() {      return this.isShowDialog;    }    setHeaderForm(action) {      switch (action) {        case ACTIONS.ADD:          this.currentHeader = this.formHeaders.add;          break;        case ACTIONS.EDIT:          this.currentHeader = this.formHeaders.edit;          break;        default:          this.currentHeader = this.formHeaders.view;          break;      }    }    openDialog(action) {      this.action = action;      this.setHeaderForm(action);      this.isShowDialog = true;    }    closeDialog() {      this.isShowDialog = false;    }  } |

|  |
| --- |
| utils/export.js |
| import \* as XLSX from "xlsx";  export function exportXLSX(data, tableColumns, filename, isGroupList = false) {    const columns = tableColumns.concat([]);    if (isGroupList) {      columns.unshift({ field: "fullname", header: "ФИО" });    }    const filterData = data.map((student) => {      const item = {};      columns.forEach((column) => {        const fieldSplit = column.field.split(".");        if (fieldSplit.length == 1) {          item[column.header] = student[column.field];        } else {          item[column.header] = student[fieldSplit[0]][fieldSplit[1]];        }      });      return item;    });    const workSheet = XLSX.utils.json\_to\_sheet(filterData);    const workBook = XLSX.utils.book\_new();    XLSX.utils.book\_append\_sheet(workBook, workSheet, "data");    XLSX.writeFile(workBook, `${filename}.xlsx`);  } |

|  |
| --- |
| validators/studentFormRules.js Остальные файлы с валидацией реализованы аналогичным образом |
| import { required, maxLength, email } from "@vuelidate/validators";  export default {    surname: { required, maxLength: maxLength(50) },    name: { required, maxLength: maxLength(50) },    patronymic: { required, maxLength: maxLength(50) },    sex: { required },    PersonalDatum: {      birthday: { required },      reportCardNumber: { required },      phoneNumber: { required },      email: { required, email },      residentialAddress: { required },    },  }; |

|  |
| --- |
| views/ParentMeetings/api/parentMeetings.js Все файлы с обращением к API реализованы идентичным либо похожим способом |
| import ky from "ky";  import { API\_URL } from "@/constants";  import InMemoryJWT from "@/auth/services/InMemoryJWT.js";  const parentMeetingAPI = ky.create({    prefixUrl: `${API\_URL}/protocols/parent-meetings`,  });  export async function getParentMeetings() {    const response = await parentMeetingAPI      .get("", {        headers: {          Authorization: `Bearer ${InMemoryJWT.getToken()}`,        },      })      .then((response) => response)      .catch((error) => console.log(error));    return response.json();  }  export async function getParents() {    const response = await parentMeetingAPI      .get("parents", {        headers: {          Authorization: `Bearer ${InMemoryJWT.getToken()}`,        },      })      .then((response) => response)      .catch((error) => console.log(error));    return response.json();  }  export async function createParentMeeting(parentMeeting) {    const response = await parentMeetingAPI.post("", {      json: parentMeeting,      headers: {        Authorization: `Bearer ${InMemoryJWT.getToken()}`,      },    });    return response.json();  }  export async function updateParentMeeting({ id, parentMeeting }) {    await parentMeetingAPI.put(`${parentMeeting.id}`, {      json: parentMeeting,    });  }  export async function deleteParentMeeting(id) {    await parentMeetingAPI.delete(`${id}`);  } |

|  |
| --- |
| views/ParentMeetings/stores/parentMeetings.js Остальные файлы-хранилища реализованы аналогичным либо похожим способом |
| import DialogForm from "@/utils/dialog";  import { defineStore } from "pinia";  import { ref, computed } from "vue";  import { useVuelidate } from "@vuelidate/core";  import parentMeetingRules from "@/validators/parentMeetingRules.js";  import { useConfirmStore } from "@/stores/confirms";  import \* as API from "../api/parentMeetings.js";  import { GROUP\_ID } from "@/constants";  import parentMeetingModel from "../models/parentMeeting.js";  // import InMemoryJWT from "@/auth/services/InMemoryJWT";  export const useParentMeetingsStore = defineStore("parentMeetings", () => {    const { confirmAdd, confirmEdit, confirmDelete } = useConfirmStore();    const dialog = ref(      new DialogForm({        add: "Добавление протокола",        edit: "Редактирование протокола",        view: "Просмотр протокола",      })    );    const parentMeeting = ref(parentMeetingModel.fields);    const v$ = useVuelidate(parentMeetingRules, parentMeeting);    function resetParentMeeting() {      parentMeeting.value = parentMeetingModel.fields;    }    const selectedParentMeetings = ref([]);    const parentMeetings = ref([]);    const presentParents = ref([]);    const presentParentsComputed = computed(() => {      try {        return parentMeeting.value.Attendanceparentmeetings.map(          (attendance) => attendance.Relative        );      } catch (error) {        return [];      }    });    const parents = ref([]);    async function fetchParentMeetings() {      const response = await API.getParentMeetings();      parentMeetings.value = response;    }    async function fetchParents() {      const response = await API.getParents();      parents.value = response;    }    const isSubmit = ref(false);    async function addParentMetting() {      isSubmit.value = true;      confirmAdd({        invalid: v$.value.$invalid,        funcIf: async () => {          const response = await API.createParentMeeting({            groupId: GROUP\_ID,            ...parentMeeting.value,          });          parentMeetings.value.push(response);          dialog.value.closeDialog();          isSubmit.value = false;        },      });    }    async function editParentMetting() {      await API.updateParentMeeting({        id: parentMeeting.value.id,        parentMeeting: {          ...parentMeeting.value,        },      });      await fetchParentMeetings();    }    const confirmEditParentMeeting = () => {      confirmEdit({        invalid: v$.value.$invalid,        funcAccept: async () => {          await editParentMetting();          isSubmit.value = false;          dialog.value.closeDialog();        },        funcReject: async () => {          await fetchParentMeetings();          dialog.value.closeDialog();        },      });    };    async function deleteParentMeeting(id) {      await API.deleteParentMeeting(id);      parentMeetings.value = parentMeetings.value.filter(        (meet) => meet.id !== id      );    }    const confirmDeleteParentMeeting = () => {      confirmDelete({        funcAccept: () => {          selectedParentMeetings.value.forEach((meet) => {            deleteParentMeeting(meet.id);          });        },        funcReject: () => {          selectedParentMeetings.value = [];        },      });    };    return {      dialog,      parentMeeting,      v$,      selectedParentMeetings,      parentMeetings,      fetchParentMeetings,      confirmDeleteParentMeeting,      addParentMetting,      confirmEditParentMeeting,      resetParentMeeting,      isSubmit,      presentParents,      presentParentsComputed,      parents,      fetchParents,    };  }); |

|  |
| --- |
| views/ParentMeetings/utils/export.js Остальные файлы с экспортом данных реализованы аналогичным либо похожим способом |
| import { Document, Packer, Paragraph, TextRun } from "docx";  import { saveAs } from "file-saver";  import { FONT\_SIZE, DEFAULT\_PAGE\_MARGINS } from "@/utils/docx\_elements.js";  export function exportToParentMeetingInDOCX(parentMeeting) {    const title = new Paragraph({      children: [        new TextRun({          text: "Протокол родительского собрания №" + parentMeeting.id,          size: FONT\_SIZE,          bold: true,        }),      ],      alignment: "center",    });    const date = new Paragraph({      children: [        new TextRun({          text: "Дата: ",          size: FONT\_SIZE,          break: 1,        }),        new TextRun({          text: parentMeeting.meetingDate,          size: FONT\_SIZE,        }),      ],    });    const theme = new Paragraph({      children: [        new TextRun({          text: "Тема: ",          size: FONT\_SIZE,        }),        new TextRun({          text: parentMeeting.theme,          size: FONT\_SIZE,        }),      ],    });    const familyMembers = parentMeeting.FamilyMembers.map((familyMember) => {      return new TextRun({        text: familyMember.fullname,        size: FONT\_SIZE,        break: 1,      });    });    const presentPeople = new Paragraph({      children: [        new TextRun({          text: "ФИО присутствующих родителей:",          size: FONT\_SIZE,          break: 1,        }),        ...familyMembers,      ],    });    const content = [      new Paragraph({        children: [          new TextRun({            text: "ПОВЕСТКА:",            size: FONT\_SIZE,            break: 2,          }),        ],        alignment: "center",      }),      new Paragraph({        children: [          new TextRun({            text: parentMeeting.content,            size: FONT\_SIZE,          }),        ],        run: {          size: FONT\_SIZE,        },      }),    ];    const doc = new Document({      sections: [        {          properties: {            page: {              margin: DEFAULT\_PAGE\_MARGINS,            },          },          children: [title, date, theme, presentPeople, ...content],        },      ],    });    Packer.toBlob(doc).then((blob) => {      saveAs(blob, "test.docx");    });  } |

Backend-часть

|  |
| --- |
| index.js |
| import express from "express";  import { generateDB } from "./commands/syncDB.js";  import dotenv from "dotenv";  import cors from "cors";  import swaggerUI from "swagger-ui-express";  import swaggerJSDoc from "swagger-jsdoc";  import swaggerOptions from "./swaggerOptions.js";  import Fingerprint from "express-fingerprint";  import cookieParser from "cookie-parser";  import AuthRootRouter from "./routers/Auth/AuthRouter.js";  import TokenService from "./services/Auth/Token.js";  import GroupRouter from "./routers/Group/GroupRouter.js";  import StudentRouter from "./routers/Group/StudentRouter.js";  import ProfileRouter from "./routers/Profile/ProfileRouter.js";  import ProtocolRouter from "./routers/Protocols/ProtocolRouter.js";  import SocialPassportRouter from "./routers/SocialPassport/SocialPassportRouter.js";  import ImageRouter from "./routers/Profile/ImageRouter.js";  import { IS\_DOCKER\_RUNNING } from "./constants.js";  if (IS\_DOCKER\_RUNNING) {    await generateDB();  }  dotenv.config();  const PORT = process.env.PORT || 5000;  const app = express();  app.use("/static", express.static("./static"));  app.use(cookieParser());  app.use(express.json());  app.use(cors({ credentials: true, origin: process.env.CLIENT\_URL }));  app.use(express.static("uploads"));  app.use(    Fingerprint({      parameters: [Fingerprint.useragent, Fingerprint.acceptHeaders],    })  );  const specs = swaggerJSDoc(swaggerOptions);  app.use("/api-docs", swaggerUI.serve, swaggerUI.setup(specs));  app.use("/groups", GroupRouter);  app.use("/students", StudentRouter);  app.use("/profile", ProfileRouter);  app.use("/protocols", ProtocolRouter);  app.use("/images", ImageRouter);  app.use("/social-passport", TokenService.checkAccess, SocialPassportRouter);  app.use("/auth", AuthRootRouter);  app.get("/resource/protected", TokenService.checkAccess, (req, res) => {    return res.status(200).json({ curator: req.curator, group: req.group });  });  app.get("/hello", (req, res) => {    res.send("Hello World!");  });  app.listen(PORT, () => {    console.log(`Server is running`);  }); |

|  |
| --- |
| controllers/Group/StudentController.js Остальные контроллеры реализованы похожим способом |
| import StudentService from "../../services/Group/StudentService.js";  import ErrorUtils from "../../utils/Errors.js";  export default class StudentController {    static async getAllStudentByGroupId(req, res) {      try {        console.log("getAllStudentByGroupId");        console.log(req.groupId);        const response = await StudentService.getAllStudentByGroupId(req.groupId);        res.status(200).json(response);      } catch (error) {        console.log(error);        res.status(500).json(error);      }    }    static async getStudentById(req, res) {      try {        const { id } = req.params;        const response = await StudentService.getStudentById(id);        res.status(200).json(response);      } catch (error) {        console.log(error);        res.status(500).json(error);      }    }    static async createStudent(req, res) {      try {        console.log("createStudent");        console.log(req.groupId);        const response = await StudentService.createStudent({          data: req.body,          groupId: req.groupId,        });        res.status(201).json(response);      } catch (error) {        console.log(error);        return ErrorUtils.catchError(res, error);      }    }    static async updateStudent(req, res) {      try {        const { id } = req.params;        await StudentService.updateStudent({ id, data: req.body });        res.sendStatus(204);      } catch (error) {        console.log(error);        res.status(500).json(error);      }    }    static async deleteStudent(req, res) {      try {        const { id } = req.params;        await StudentService.deleteStudent(id);        res.sendStatus(204);      } catch (error) {        console.log(error);        res.status(500).json(error);      }    }  } |

|  |
| --- |
| models/Group/Student.js Остальные модели реализованы аналогичным способом |
| import { DataTypes } from "sequelize";  import sequelize from "../../config/connectingDB.js";  const Student = sequelize.define(    "Student",    {      surname: {        type: DataTypes.STRING(50),        allowNull: false,      },      name: {        type: DataTypes.STRING(50),        allowNull: false,      },      patronymic: {        type: DataTypes.STRING(50),        allowNull: false,      },      fullname: {        type: DataTypes.VIRTUAL,        get() {          return `${this.surname} ${this.name} ${this.patronymic}`;        },      },      sex: {        type: DataTypes.ENUM,        values: ["Мужской", "Женский"],        allowNull: false,      },      note: DataTypes.TEXT,    },    { timestamps: false }  );  export default Student; |

|  |
| --- |
| repositories/Group/StudentRepository.js Остальные репозитории реализованы похожим образом |
| import initModels from "../../models/initModels.js";  const {    Student,    PersonalData,    FamilyMember,    StudentAttitudes,    StudentPersonality,  } = initModels();  export default class StudentRepository {    static async getAllStudentByGroupId(groupId) {      let students = await Student.findAll({        where: {          groupId,        },        include: [PersonalData],      });      if (!students) {        return null;      }      const expelledStudents = students.filter((student) => {        if (student.note) {          return student.note.toLowerCase().includes("отчислен");        }        return false;      });      if (expelledStudents.length > 0) {        students = students.filter(          (student) => !expelledStudents.includes(student)        );        students = students.concat(expelledStudents);      }      return students;    }    static async getStudentById(id) {      const student = await Student.findByPk(id, {        include: [PersonalData],      });      if (!student) {        return null;      }      return student;    }    static async getStudentsByPhoneNumber(phoneNumber) {      const student = await Student.findOne({        include: {          model: PersonalData,          where: {            phoneNumber,          },        },      });      if (!student) {        return null;      }      return student;    }    static async getStudentBySNILS(SNILS) {      const student = await Student.findOne({        include: {          model: PersonalData,          where: {            SNILS,          },        },      });      if (!student) {        return null;      }      return student;    }    static async getStudentByMedicalPolicy(medicalPolicy) {      const student = await Student.findOne({        include: {          model: PersonalData,          where: {            medicalPolicy,          },        },      });      if (!student) {        return null;      }      return student;    }    static async getStudentByReportCardNumber(reportCardNumber) {      const student = await Student.findOne({        include: {          model: PersonalData,          where: {            reportCardNumber,          },        },      });      if (!student) {        return null;      }      return student;    }    static async getStudentByEmail(email) {      const student = await Student.findOne({        include: {          model: PersonalData,          where: {            email,          },        },      });      if (!student) {        return null;      }      return student;    }    static async createStudent({ student, personalData }) {      const createdStudent = await Student.create(student);      const createdPersonalData = await PersonalData.create(personalData);      await createdStudent.setPersonalDatum(createdPersonalData);      const studentAttitudes = await StudentAttitudes.create();      const studentPersonality = await StudentPersonality.create();      await createdStudent.setStudentAttitude(studentAttitudes);      await createdStudent.setStudentPersonality(studentPersonality);      const newStudent = await Student.findByPk(createdStudent.id, {        include: [PersonalData],      });      return newStudent;    }    static async updateStudent({ id, student, personalData }) {      await Student.update(student, { where: { id } });      await PersonalData.update(personalData, {        where: { studentId: id },      });    }    static async deleteStudent(id) {      const student = await Student.findByPk(id, { include: [FamilyMember] });      student.FamilyMembers.forEach((familyMember) => {        familyMember.destroy({          force: true,        });      });      student.destroy({ force: true });    }    static async updateImage({ id, image }) {      await PersonalData.update({ image }, { where: { studentId: id } });    }  } |

|  |
| --- |
| routes/Group/StudentRouter.js Остальные роутеры выглядят аналогичным образом |
| import { Router } from "express";  import StudentController from "../../controllers/Group/StudentController.js";  import TokenService from "../../services/Auth/Token.js";  const studentRouter = Router();  studentRouter.get(    "/",    TokenService.checkAccess,    StudentController.getAllStudentByGroupId  );  studentRouter.get("/:id", StudentController.getStudentById);  studentRouter.post(    "/",    TokenService.checkAccess,    StudentController.createStudent  );  studentRouter.put("/:id", StudentController.updateStudent);  studentRouter.delete("/:id", StudentController.deleteStudent);  export default studentRouter; |

|  |
| --- |
| services/Group/StudentService.js Остальные сервисы реализованы похожим образом |
| import StudentRepository from "../../repositories/Group/StudentRepository.js";  import { Conflict } from "../../utils/Errors.js";  export default class StudentService {    static async getAllStudentByGroupId(groupId) {      const response = await StudentRepository.getAllStudentByGroupId(groupId);      return response;    }    static async getStudentById(id) {      const response = await StudentRepository.getStudentById(id);      return response;    }    static async createStudent({ data, groupId }) {      const { surname, name, patronymic, sex, note, PersonalDatum } = data;      if (PersonalDatum.SNILS) {        const studentBySNILS = await StudentRepository.getStudentBySNILS(          PersonalDatum.SNILS        );        if (studentBySNILS) {          throw new Conflict("Студент с таким СНИЛС уже существует");        }      }      if (PersonalDatum.phoneNumber) {        const studentByPhoneNumber =          await StudentRepository.getStudentsByPhoneNumber(            PersonalDatum.phoneNumber          );        if (studentByPhoneNumber) {          throw new Conflict("Студент с таким номером телефона уже существует");        }      }      if (PersonalDatum.reportCardNumber) {        const studentByReportCartNumber =          await StudentRepository.getStudentByReportCardNumber(            PersonalDatum.reportCardNumber          );        if (studentByReportCartNumber) {          throw new Conflict("Студент с таким табельным номером уже существует");        }      }      if (PersonalDatum.medicalPolicy) {        const studentByMedicalPolicy =          await StudentRepository.getStudentByMedicalPolicy(            PersonalDatum.medicalPolicy          );        if (studentByMedicalPolicy) {          throw new Conflict(            "Студент с таким медицинским полисом уже существует"          );        }      }      if (PersonalDatum.birthday) {        const studentByEmail = await StudentRepository.getStudentByEmail(          PersonalDatum.email        );        if (studentByEmail) {          throw new Conflict("Студент с такой почтой уже существует");        }      }      if (PersonalDatum.residentialAddress) {        const newStudent = await StudentRepository.createStudent({          student: { surname, name, patronymic, sex, groupId, note },          personalData: PersonalDatum,        });        return newStudent;      }    }    static async updateStudent({ id, data }) {      const { surname, name, patronymic, sex, note, PersonalDatum } = data;      await StudentRepository.updateStudent({        id,        student: { surname, name, patronymic, sex, note },        personalData: PersonalDatum,      });    }    static async deleteStudent(id) {      await StudentRepository.deleteStudent(id);    }  } |

|  |
| --- |
| utils/Errors.js |
| class WebError {    constructor(status, error) {      this.status = status;      this.error = error;    }  }  // «необрабатываемый экземпляр»  export class Unprocessable extends WebError {    constructor(error) {      super(422, error);    }  }  // «конфликт»  export class Conflict extends WebError {    constructor(error) {      super(409, error);    }  }  // «не найдено»  export class NotFound extends WebError {    constructor(error) {      super(404, error);    }  }  // «запрещено»  export class Forbidden extends WebError {    constructor(error) {      super(403, error);    }  }  // «не авторизован»  export class Unauthorized extends WebError {    constructor(error) {      super(401, error);    }  }  // «неправильный, некорректный запрос»  export class BadRequest extends WebError {    constructor(error) {      super(400, error);    }  }  class ErrorUtils {    static catchError(res, error) {      console.log(error);      return res.status(error.status || 500).json(error);    }  }    export default ErrorUtils; |