МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

«ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ АГРАРНО-ЭКОНОМИЧЕСКИЙ КОЛЛЕДЖ»

Специальность 2-40 01 01 «Программное обеспечение

информационных технологий»

**КУРСОВОЙ ПРОЕКТ**

по предмету «Конструирование программ и языки программирования»

**на тему Информационная система «Журнал куратора»**

Пояснительная записка

Разработал обучающийся

группы ПО-33 \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (ФИО)

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (ФИО)

Дата защиты \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Отметка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Гомель, 2023

**СОДЕРЖАНИЕ**

[Введение 4](#_Toc138852426)

[1 Обзор и анализ методов и средств разработки базы данных 5](#_Toc138852427)

[1.1 Описание предметной области и круга задач для создания приложения 5](#_Toc138852428)

[1.2 Описание используемых технологий и языков программирования 6](#_Toc138852429)

[2 Проектирование программного продукта 7](#_Toc138852430)

[2.1 Концептуальное проектирование 7](#_Toc138852431)

[2.2 Логическая модель данных 10](#_Toc138852432)

[3 Описание технологии реализации 14](#_Toc138852433)

[3.1 Разработка графического интерфейса 14](#_Toc138852434)

[3.2 Программная реализация 16](#_Toc138852435)

[4 Тестирование программного продукта 28](#_Toc138852436)

[Заключение 31](#_Toc138852437)

[Список использованных источников 32](#_Toc138852438)

[Приложение А Листинг программы 33](#_Toc138852439)

[Приложение Б Руководство пользователя 52](#_Toc138852440)

[Приложение В Руководство программиста 57](#_Toc138852441)

# Введение

В современном обществе информация играет важную роль во всех сферах деятельности. Особенно актуальна проблема обработки и хранения информации в образовательных учреждениях, где необходимо отслеживать успеваемость, посещаемость, активность и достижения студентов. Для этого традиционно используется бумажный вариант журнала куратора, который является основным документом учета идеологической и воспитательной работы в группе. Аккуратное и своевременное ведение записей в нем является обязательным для каждого куратора. Однако такой способ ведения журнала имеет ряд недостатков, таких как:

* большой объем бумажной документации, занимающей много места;
* риск потери или повреждения журнала;
* неудобство поиска и анализа информации;
* низкая скорость и точность обработки данных;
* невозможность удаленного доступа к журналу.

Для решения этих проблем возникает необходимость создания информационной системы «Журнал куратора», которая позволит перевести бумажный вариант журнала в электронный формат и обеспечить эффективное управление информацией о студентах и их учебной деятельности.

Целью проекта является разработка приложения, которое позволит хранить, обрабатывать и анализировать данные журнала куратора и предоставит возможность постоянного доступа к ним.

Актуальность выбранной темы обусловлена необходимостью цифровизации, которая способствует более быстрой и точной обработке данных, снижает вероятность ошибок, упрощает процесс анализа информации и составления отчетности.

Задачи исследования:

* автоматизировать процессы заполнения отчетности;
* создать возможность постоянного доступа к информации.

Объект исследования — журнал куратора, его структура и содержание.

Предмет исследования — использование журнала куратора в работе современного образовательного процесса.

Журнал куратора — полезный инструмент для администрации и кураторов, который облегчает их каждодневную бумажную рутину, чтобы иметь своевременную информацию о своих учащихся.

Система была создана на основе бумажной версии журнала куратора, который хорошо знает каждый преподаватель, и повторяет его внешний вид. Простой в использовании интерфейс помогает кураторам быстро адаптироваться к новшеству, а также сокращать время на заполнение и упрощать подготовку регулярных отчетов.

# 1 Обзор и анализ методов и средств разработки базы данных

## 1.1 Описание предметной области и круга задач для создания приложения

Анализ предметной области — это первый шаг этапа системного анализа, с которого начинается разработка программной системы. Разработчик должен:

* определить информационные объекты и связи между ними;
* определить цели деятельности работы приложения;
* определить набор решаемых им задач;
* построить концептуальную модель предметной области и спроектировать концептуальную схему БД.

Журнал куратора — это документ, в котором фиксируются сведения об успеваемости, посещаемости, поведении и других аспектах учебной и внеучебной деятельности студентов, находящихся под курированием определенного преподавателя.

В основе разработки лежала бумажная копия журнала куратора, знакомая каждому педагогу, который и стал образцом внешнего вида системы. Интуитивно понятный интерфейс позволяет преподавателям с легкостью освоить нововведение, а главное — затрачивать минимум времени на заполнение, и экономить время и силы на составлении регулярной отчетности (ежемесячные пропуски, социальный паспорт группы и колледжа, социально-педагогическая характеристика группы и колледжа).

При разработке информационной системы журнала куратора необходимо учитывать следующие задачи:

* учет (создание, редактирование, удаление, анализ) специальностей, кураторов, групп и учащихся;
* валидация данных на стороне клиента и сервера;
* защита от распространенных атак, таких как «XSS», «CSRF» и «SQL-инъекции»;
* управление доступом на основе ролей;
* адаптация под все разрешения экрана;
* возможность экспорта данных в различные форматы для сдачи отчетности.

Таким образом, предметная область связана с проектированием, разработкой, внедрением и эксплуатацией информационной системы, которая позволяет автоматизировать различные аспекты работы кураторов в учебных заведениях.

## 1.2 Описание используемых технологий и языков программирования

Приложение разработано на «Laravel» — одном из самых популярных PHP-фреймворков для создания приложений. «Laravel» обладает высокой производительностью и простотой использования, а также имеет множество готовых компонентов для быстрого и удобного создания приложений.

Для создания интерфейса приложения использовался фреймворк «Vue JS», который является одним из наиболее популярных JavaScript-фреймворков для создания интерактивных пользовательских интерфейсов. «Vue JS» обладает высокой производительностью, простотой использования и широким функционалом [1].

Для надежного и безопасного хранения пользовательских данных приложение использует базу данных «MySQL». «MySQL» — это одна из самых популярных реляционных систем управления базами данных в мире, которая обеспечивает высокую производительность, надежность и безопасность.

Все данные, которые вводятся на сайте, хранятся в базе данных «MySQL». Приложение использует современные методы шифрования для защиты конфиденциальной информации, такие как пароли пользователей и персональные данные.

Данная система СУБД подходит для самых различных проектов: от небольших приложений до больших высоконагруженных проектов. «MySQL» характеризуется такими особенностями как:

* производительность;
* надежность и безопасность. «MySQL» предоставляет шифрование данных;
* простота. С данной СУБД относительно легко работать и вести администрирование.

В качестве среды разработки программного продукта был выбран редактор «Visual Studio Code».

«Visual Studio Code (VS Code)» — это бесплатный редактор исходного кода, который позволяет работать с разными языками и технологиями веб- и облачных приложений. Он имеет функции «IDE», такие как подсветка синтаксиса, автокоррекция, отладка, поддержка «git» и другие. Он преимущественно используется в веб-разработке, но не является полноценной средой разработки [2].

В ходе рассмотрения средств разработки языками были выбраны «Typescript» и «PHP», так как они просты в освоении, среда разработки «Visual Studio Code» была выбрана из-за простого интерфейса, и «MySQL» в качестве базы данных, так как эта СУБД имеет высокую производительность.

# 2 Проектирование программного продукта

## 2.1 Концептуальное проектирование

Разработка любого программного продукта подразумевает его моделирование и проектирование.

В процессе проектирования программного средства после изучения предметной области по теме индивидуального задания были разработаны:

* диаграмма «IDEF0»;
* декомпозиция первого уровня диаграммы «IDEF0»;
* диаграмма потоков данных («DFD»).

Данные диаграммы предназначены для определения структуры будущей базы данных, последовательности выполнения процесса, а также последовательности работы программного средства.

Диаграмма IDEF0 представляет собой автоматизируемый бизнес-процесс в общем виде. Она состоит из:

* входных данных;
* выходных данных;
* блока бизнес-процесса;
* аппарата управления (документы, контролирующие выполнение бизнес-процесса);
* механизма решения бизнес-процесса (сотрудник или сотрудники, выполняющие бизнес-процесс).

Диаграмма «IDEF0» представлена на рисунке 2.1.

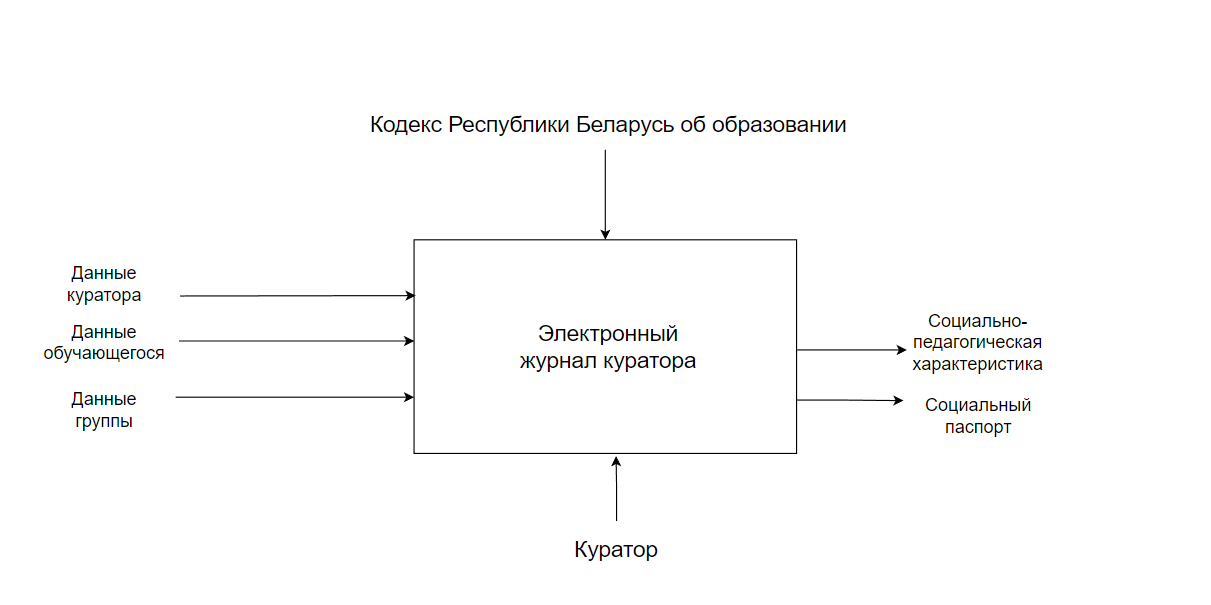


Рисунок 2.1 — Диаграмма «IDEF0»

Исходя из темы «Информационная система «Журнал куратора»» в качестве входных данных выступают объекты: данные куратора, данные обучающегося и данные группы, в качестве выходных данных: социально-педагогическая характеристика и социальный паспорт. Механизмом служит — куратор. Управляет данным процессом — «Кодекс Республики Беларусь об образовании».

Декомпозиция первого уровня диаграммы «IDEF0» представлена на рисунке 2.2.

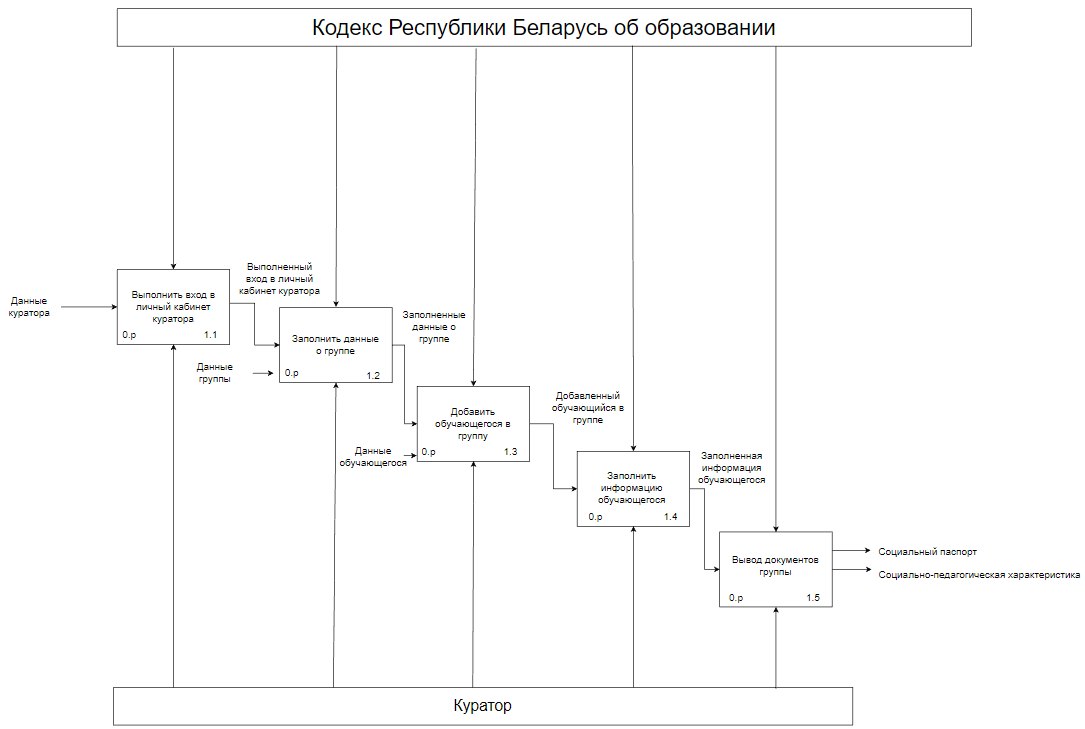


Рисунок 2.2 — Декомпозиция первого уровня диаграммы «IDEF0»

На данной диаграмме более подробно отражен процесс учета учащихся. На диаграмме отражены пять основных функциональных блока и связи между ними. Они отражают задачи и функции моделируемой системы, которые происходят в течении определенного времени и имеют распознаваемые результаты, которые являются исходными данными для последующих блоков во всей цепочке программного приложения.

1. Функциональный блок «Выполнить вход в личный кабинет куратора»:

* вход — данные куратора;
* выход — выполненный вход в личный кабинет куратора.

1. Функциональный блок «Заполнить данные о группе»:

* вход — данные группы;
* выход — заполненные данные о группе.

1. Функциональный блок «Добавить обучающегося в группу»:

* вход — данные обучающегося;
* выход — добавленный обучающийся в группе.

1. Функциональный блок «Заполнить информацию обучающегося»:

* вход — добавленный обучающийся в группе;
* выход — заполненная информация обучающегося.

1. Функциональный блок «Вывод документов в группы»:

* вход — данные об обучающихся группы;
* выход — социальный-паспорт и социально-педагогическая характеристика (отчетность).

«DFD» — это нотация, которая используется при моделировании информационных систем с точки зрения хранения, обработки и передачи данных.

Основными компонентами диаграмм потоков данных являются:

* внешние сущности;
* системы и подсистемы;
* процессы;
* накопители данных;
* потоки данных.

К главным преимуществам можно отнести:

* способность нотации точно определить внешние сущности, при этом используя анализ потоков информации внутри и за пределами системы;
* способность проектирование сверху вниз;
* описание процессов нижнего уровня.

Диаграмма потоков данных «DFD» представлена на рисунке 2.3.

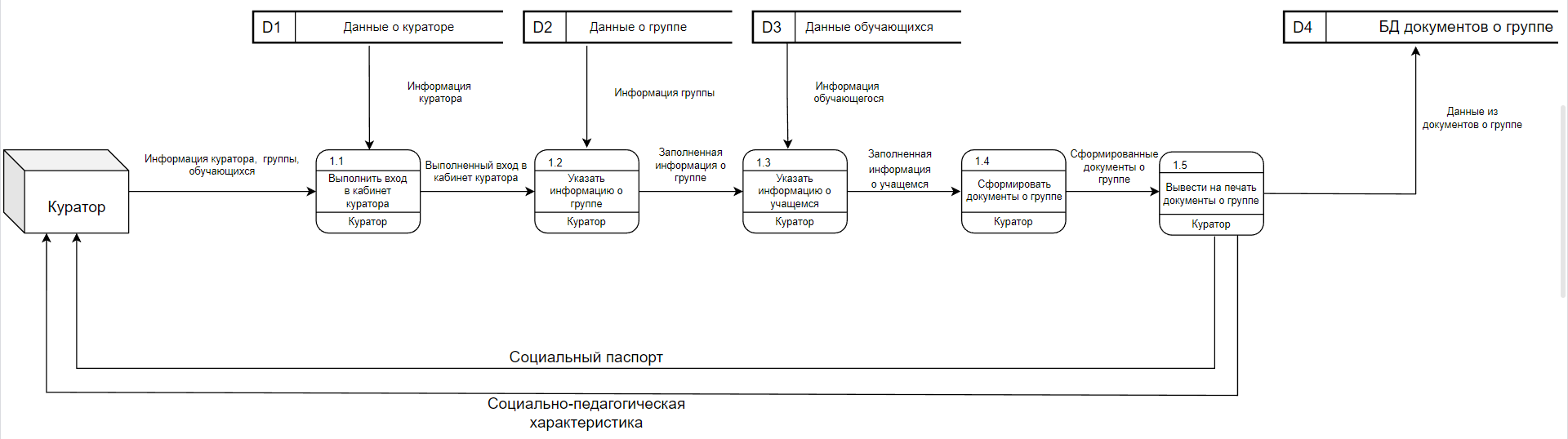


Рисунок 2.3 — Диаграмма потоков данных «DFD»

Данная диаграмма отображает процесс управления учета учащихся, а также процесс формирования документов по учету учебных групп. В качестве функциональных блоков используются блоки «Выполнить вход в кабинет куратора», «Указать информацию о группе», «Указать информацию о учащемся», «Сформировать документы о группе», «Вывести на печать документы о группе». Для описания потоков данных использовались хранилища «Данные о кураторе», «Данные о группе», «Данные обучающихся», «БД документов о группе». Куратор осуществляет обработку информации с помощью реализованных функций.

В ходе построения диаграмм был подробно описан ход работы для облегчения создания программы.

## 2.2 Логическая модель данных

База данных — это совокупность данных, организованных по определённым правилам. Она представляет собой набор информации, которая хранится упорядоченно в электронном виде [3].

Обычно база данных представляется в виде совокупности взаимосвязанных файлов или таблиц, предназначенных для решения конкретной задачи.

Схема системы базы данных — это план или архитектура того, как будут выглядеть наши данные, он не содержит самих данных, а вместо этого описывает форму данных и то, как они могут быть связаны с другими таблицами или моделями.

Таблица «absences» используется для хранения данных о пропусках учащихся.

Поля, которые находятся в этой таблице, показаны в таблице 2.1.

Таблица 2.1 — Структура таблицы «absences»

|  |  |  |
| --- | --- | --- |
| Имя поля | Тип данных поля | Описание |
| id | Числовой | Первичный ключ |
| student\_id | Числовой | Вторичный ключ |
| group\_id | Числовой | Вторичный ключ |
| date | Дата и время | Дата пропуска |
| respectful | Числовой | Количество уважительных часов |
| disrespectful | Числовой | Количество неуважительных часов |
| created\_at | Дата и время | Дата создания записи |
| updated\_at | Дата и время | Дата обновления записи |

Таблица «users» используется для хранения данных о пользователях системы.

Поля, которые находятся в этой таблице, показаны в таблице 2.2.

Таблица 2.2 — Структура таблицы «users»

|  |  |  |
| --- | --- | --- |
| Имя поля | Тип данных поля | Описание |
| 1 | 2 | 3 |
| id | Числовой | Первичный ключ |
| login | Текстовый | Текстовый идентификатор для входа в аккаунт |
| password | Текстовый | Пароль для входа в аккаунт |
| is\_admin | Логический | Является ли пользователь администратором |

Продолжение таблицы 2.2.

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| remember\_token | Текстовый | Идентификатор для запоминания системой вошедшего пользователя |
| created\_at | Дата и время | Дата создания записи |
| updated\_at | Дата и время | Дата обновления записи |

Таблица «groups» используется для хранения данных об учебных группах.

Поля, которые находятся в этой таблице, показаны в таблице 2.3.

Таблица 2.3 — Структура таблицы «groups»

|  |  |  |
| --- | --- | --- |
| Имя поля | Тип данных поля | Описание |
| id | Числовой | Первичный ключ |
| number | Числовой | Номер группы |
| start\_education | Дата и время | Дата начала обучения |
| end\_education | Дата и время | Дата окончания обучения |
| specialty\_id | Числовой | Вторичный ключ |
| curator\_id | Числовой | Вторичный ключ |
| created\_at | Дата и время | Дата создания записи |
| updated\_at | Дата и время | Дата обновления записи |

Таблица «students» используется для хранения данных об учащихся.

Поля, которые находятся в этой таблице, показаны в таблице 2.4.

Таблица 2.4 — Структура таблицы «students»

|  |  |  |
| --- | --- | --- |
| Имя поля | Тип данных поля | Описание |
| 1 | 2 | 3 |
| id | Числовой | Первичный ключ |
| user\_id | Числовой | Вторичный ключ |
| surname | Текстовый | Фамилия |
| name | Текстовый | Имя |
| patronymic | Текстовый | Отчество |
| sex | Перечисление | Пол |
| birthday | Дата и время | Дата рождения |
| citizenship | Текстовый | Гражданство |
| home\_phone | Текстовый | Домашний телефон |
| phone | Текстовый | Телефон |
| region | Текстовый | Область проживания |
| district | Текстовый | Район проживания |
| residenceType | Текстовый | Тип места проживания |
| residence | Текстовый | Место проживания |

Продолжение таблицы 2.4.

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| street | Текстовый | Улица проживания |
| created\_at | Дата и время | Дата создания записи |
| updated\_at | Дата и время | Дата обновления записи |

Таблица «characteristics» используется для хранения данных об характеристиках учащихся.

Поля, которые находятся в этой таблице, показаны в таблице 2.5.

Таблица 2.5 — Структура таблицы «characteristics»

|  |  |  |
| --- | --- | --- |
| Имя поля | Тип данных поля | Описание |
| id | Числовой | Первичный ключ |
| title | Текстовый | Название характеристики |
| group | Текстовый | Категория характеристики (учащийся, семья) |
| shorthand\_name | Текстовый | Краткое название характеристики |
| type | Перечисление | Тип характеристики (одиночный, множественный выбор) |
| created\_at | Дата и время | Дата создания записи |
| updated\_at | Дата и время | Дата обновления записи |

Таблица «relatives» используется для хранения данных об родственниках учащихся.

Поля, которые находятся в этой таблице, показаны в таблице 2.6.

Таблица 2.6 — Структура таблицы «relatives»

|  |  |  |
| --- | --- | --- |
| Имя поля | Тип данных поля | Описание |
| id | Числовой | Первичный ключ |
| surname | Текстовый | Фамилия |
| name | Текстовый | Имя |
| patronymic | Текстовый | Отчество |
| sex | Перечисление | Пол |
| birthday | Дата и время | Дата рождения |
| job | Текстовый | Место работы |
| position | Текстовый | Должность |
| phone | Текстовый | Номер телефона |
| address | Текстовый | Адрес проживания |

Таблица «marks» используется для хранения данных о оценках учащихся за месяц.

Поля, которые находятся в этой таблице, показаны в таблице 2.7.

Таблица 2.7 — Структура таблицы «marks»

|  |  |  |
| --- | --- | --- |
| Имя поля | Тип данных поля | Описание |
| id | Числовой | Первичный ключ |
| student\_id | Числовой | Вторичный ключ |
| group\_id | Числовой | Вторичный ключ |
| date | Дата и время | Дата выставления оценки |
| mark | Числовой | Оценка |
| created\_at | Дата и время | Дата создания записи |
| updated\_at | Дата и время | Дата обновления записи |

# 3 ОПИСАНИЕ ТЕХНОЛОГИИ РЕАЛИЗАЦИИ

## 3.1 Разработка графического интерфейса

Интерфейс — это набор инструментов, который позволяет пользователю взаимодействовать с программой. В более широком смысле термин обозначает любые инструменты для соприкосновения между разными системами и сущностями [4].

Проектирование интерфейсов — это всегда поиск наиболее эффективного решения, которое основано на понимании задач, мотиваций и обстоятельств пользователей и в то же время учитывает цели, возможности и ограничения со стороны бизнеса и технологий.

Результатом проектирования являются макеты и сопроводительные материалы, которые можно передать команде разработки.

Шаблон приложения в данной курсовой работе представлен на рисунке 3.1.

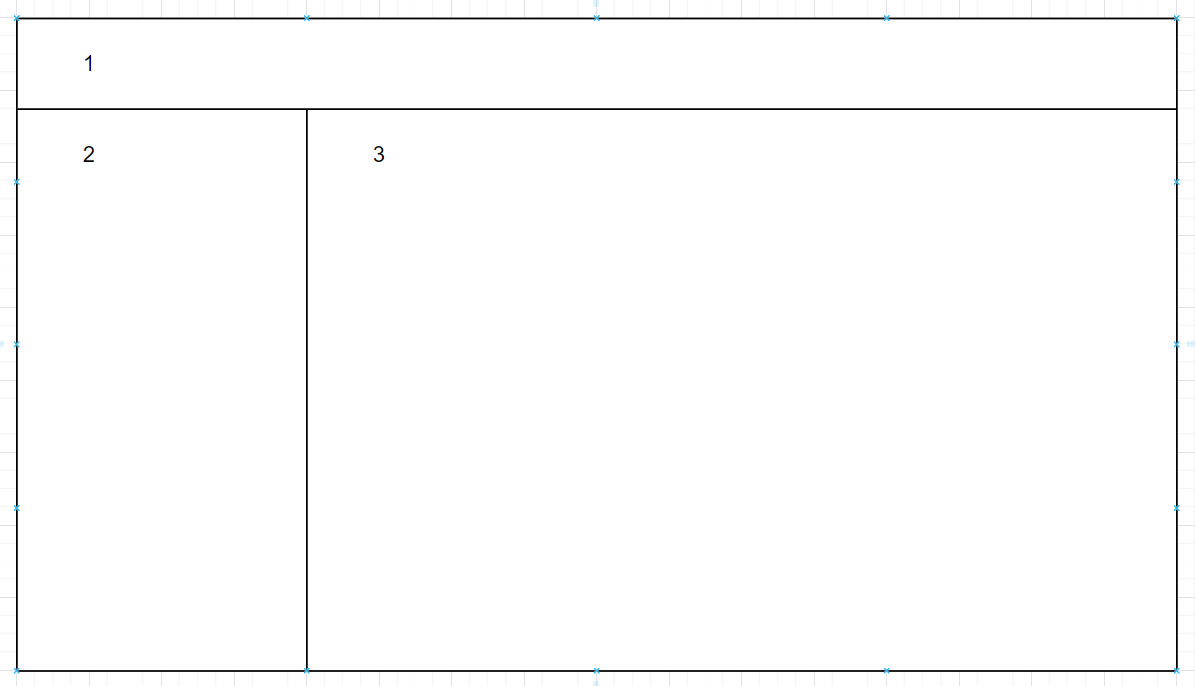


Рисунок 3.1 — Шаблон приложения:

1 — Область шапки приложения; 2 — Область навигации;   
3 — Область основного содержимого

Макет главной формы представлен на рисунке 3.2.

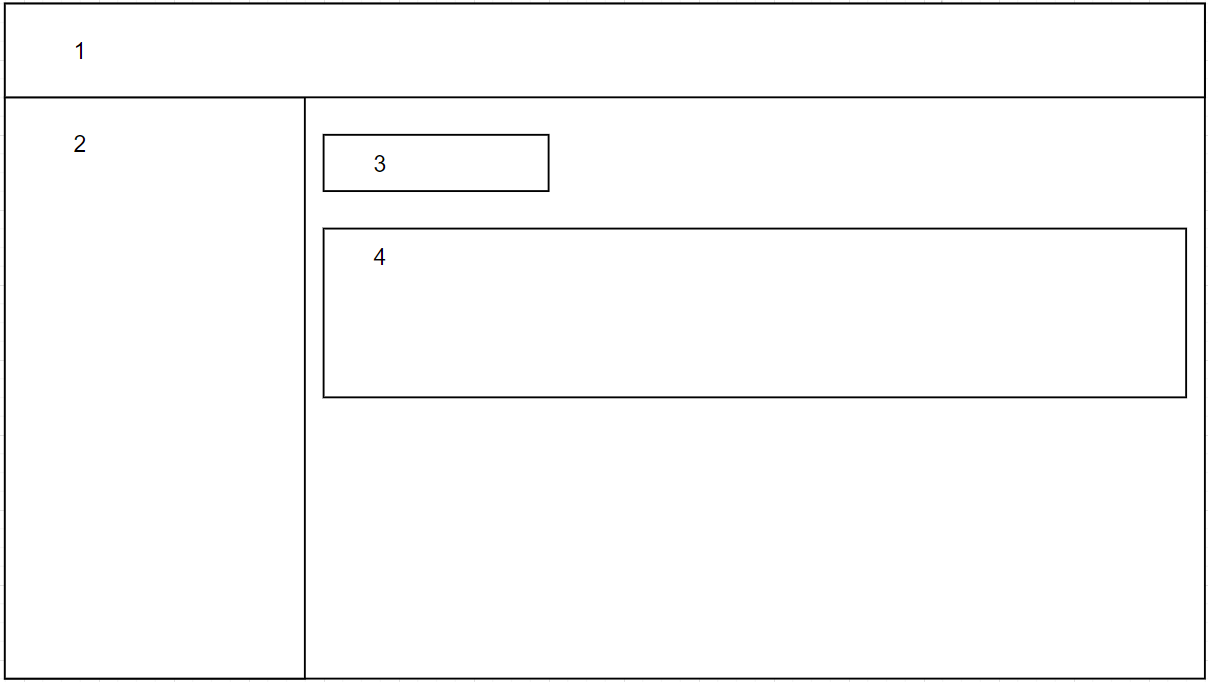


Рисунок 3.2 — Макет главной формы:

1 — Область шапки приложения; 2 — Область навигации; 3 — Область основного содержимого; 3 — Кнопка для формирования отчетности;   
4 — Секция диаграмм

Макеты форм составления пропусков и оценок представлены на рисунке 3.3.

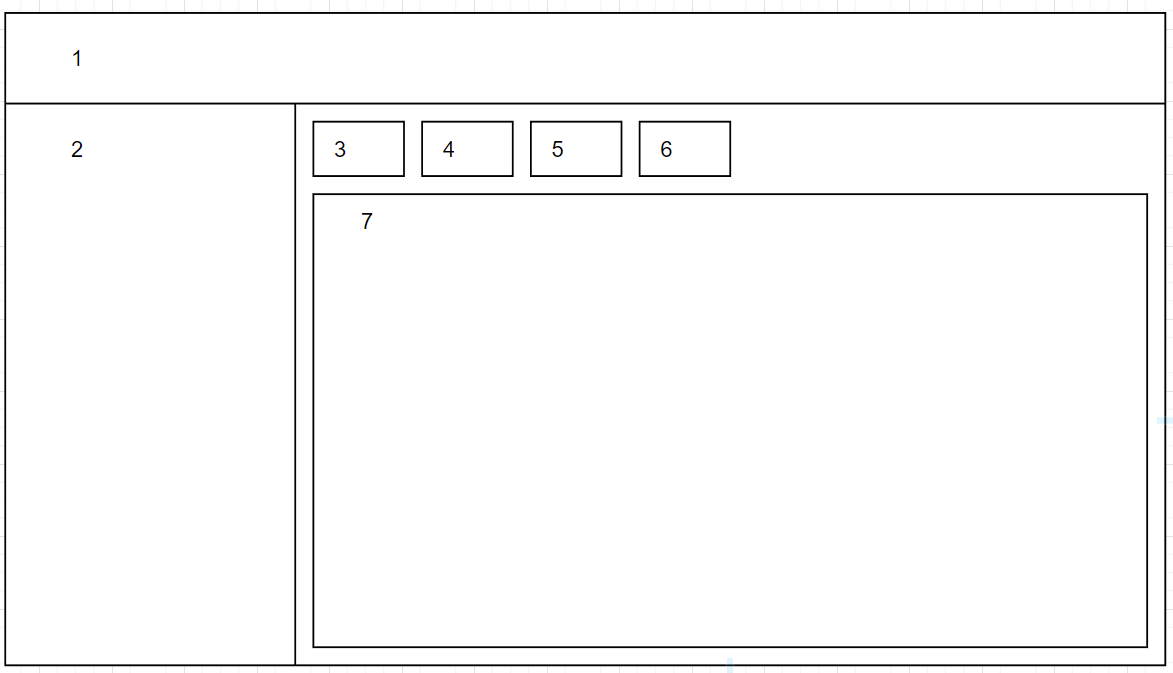


Рисунок 3.3 — Макеты форм составления пропусков и оценок:

1 — Область шапки приложения; 2 — Область навигации; 3 — Кнопка сохранения; 4 — Навигация по времени; 5 — Кнопка для запуска полноэкранного режима; 6 — Кнопка экспорта в Excel

## 3.2 Программная реализация

При первом запуске приложения открывается страница с формой входа, которая позволяет войти в аккаунт. Страница с формой входа представлена на рисунке 3.4.

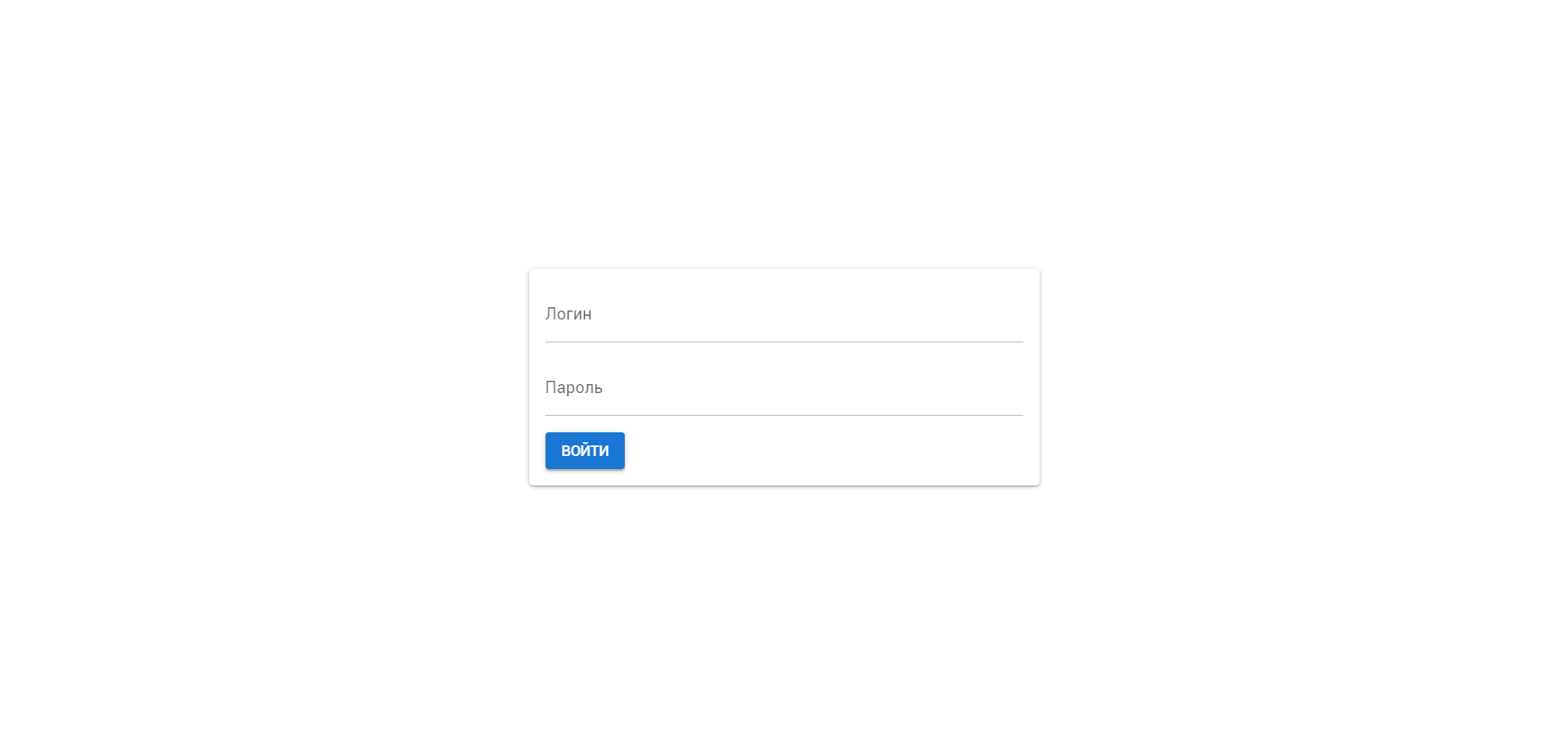


Рисунок 3.4 — Страница с формой входа

После аутентификации система запомнит вошедшего пользователя и при последующем запуске веб-приложения входить в аккаунт не требуется.

Аутентификация — процедура проверки подлинности, например: проверка подлинности пользователя путём сравнения введённого им логина и пароля с данными, хранящимся в базе данных [5].

В приложении предусмотрено 2 роли: администратор (заместитель директора по воспитательной работе) и кураторы групп. Регистрацию новых пользователей осуществляет администратор.

На каждой форме производится первичная проверка введенных данных, которая существенно улучшает взаимодействие пользователя с интерфейсом. Построение пользовательского интерфейса было вдохновлено принципом «Не заставляйте меня думать!», когда каждый элемент с точки зрения человеческих возможностей должен быть самоочевидным. Обнаружение некорректных данных на стороне клиента позволяет пользователю немедленно их исправить. Результаты проверки введенных данных представлены на рисунке 3.5 и 3.6 соответственно [6].

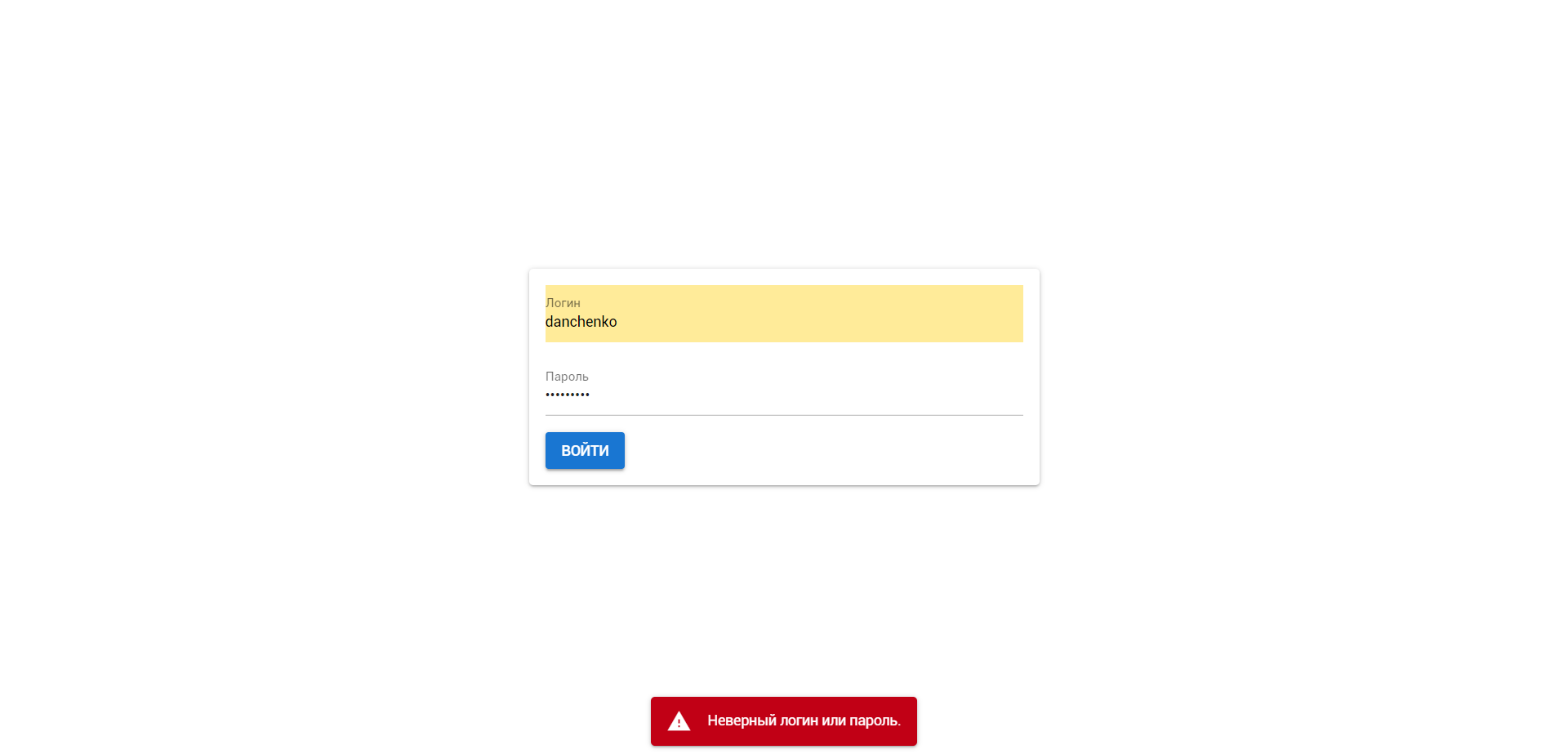


Рисунок 3.5 — Результат проверки введенных данных

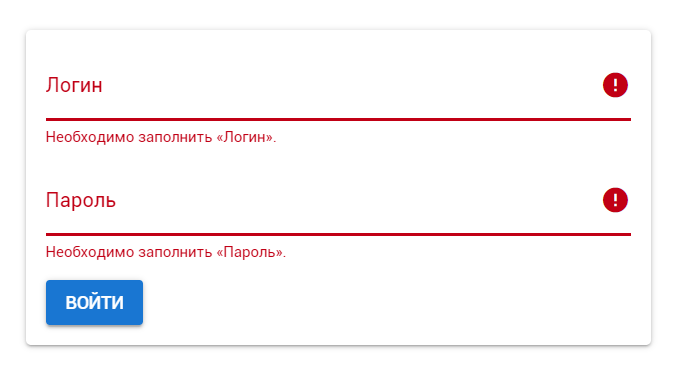


Рисунок 3.6 — Результат проверки заполнения обязательных полей

При входе в аккаунт пользователь вводит свои данные и, в случае успеха, переходит на главную форму. Главная форма содержит общую информацию о деятельности колледжа: выходные формы, характеристики учащихся и их семей в виде диаграмм и боковую навигацию, позволяющую переходить на другие страницы. Главная форма представлена на рисунке 3.7.

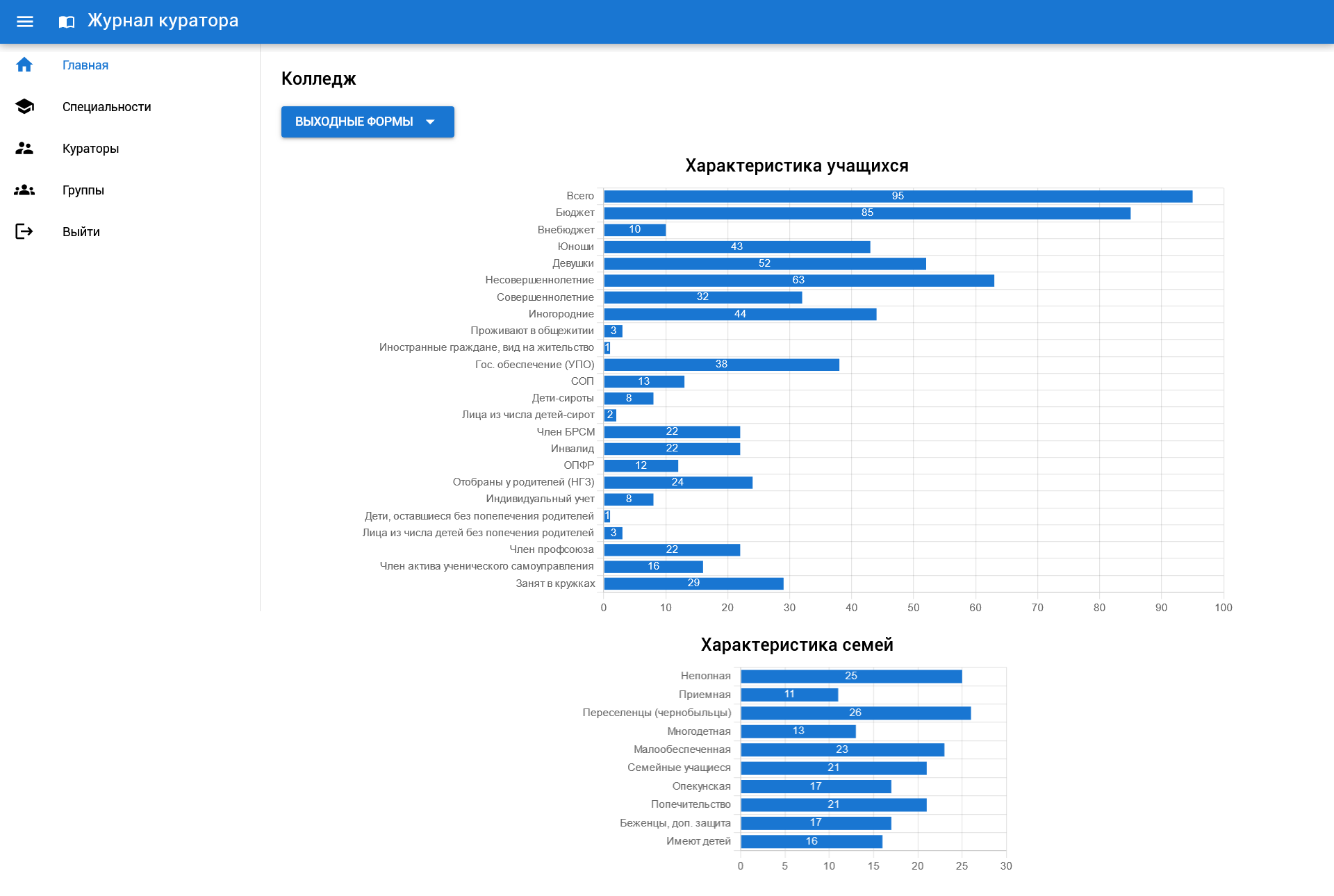


Рисунок 3.7 — Главная форма

При переходе на форму «Специальности» отобразится таблица, которая позволяет сортировать, создавать, редактировать или удалять специальности. Форма «Специальности» представлена на рисунке 3.8.

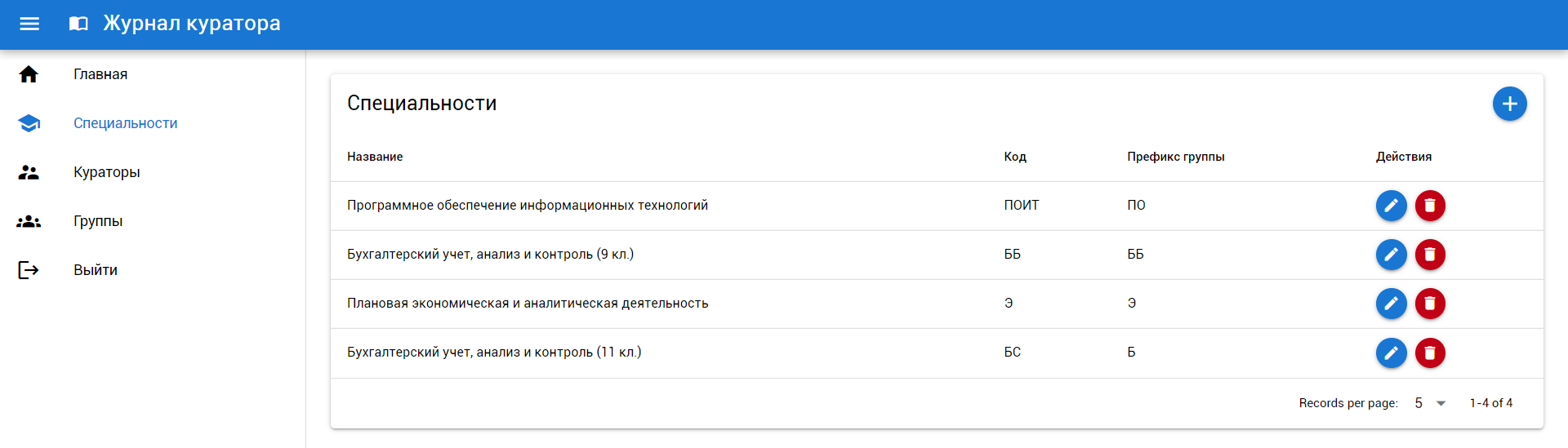


Рисунок 3.8 — Форма «Специальности»

При создании или редактировании специальности открывается форма, которая представлена на рисунке 3.9.

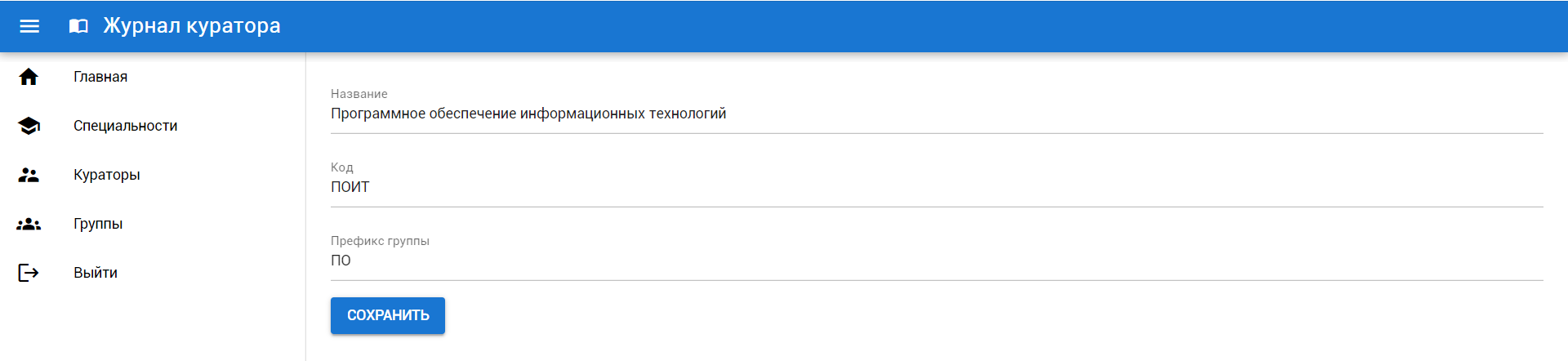


Рисунок 3.9 — Форма создания, редактирования специальности

При переходе на форму «Кураторы» отобразится таблица, которая позволяет сортировать, создавать, редактировать или удалять кураторов. Форма «Кураторы» представлена на рисунке 3.10.

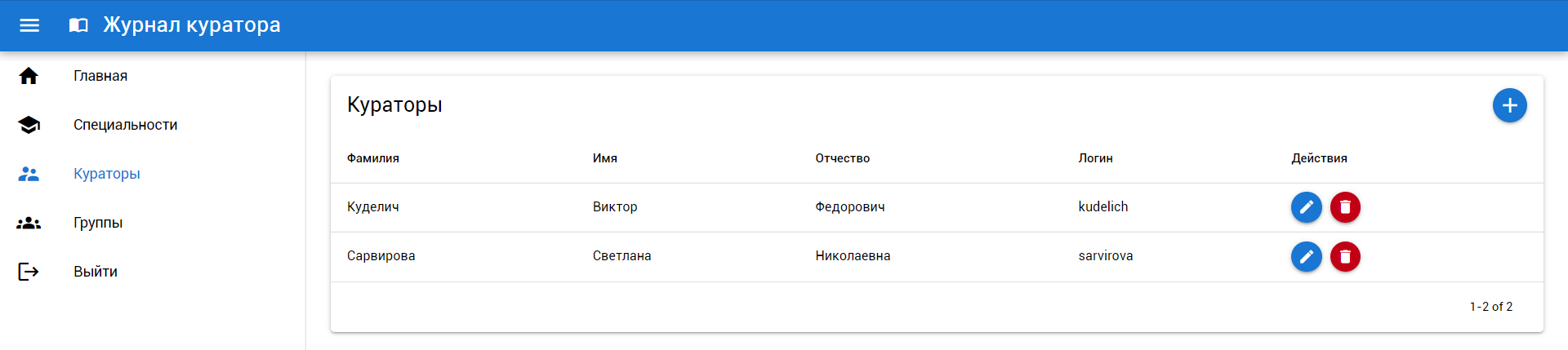


Рисунок 3.10 — Форма «Кураторы»

При создании или редактировании куратора открывается форма, которая представлена на рисунке 3.11.

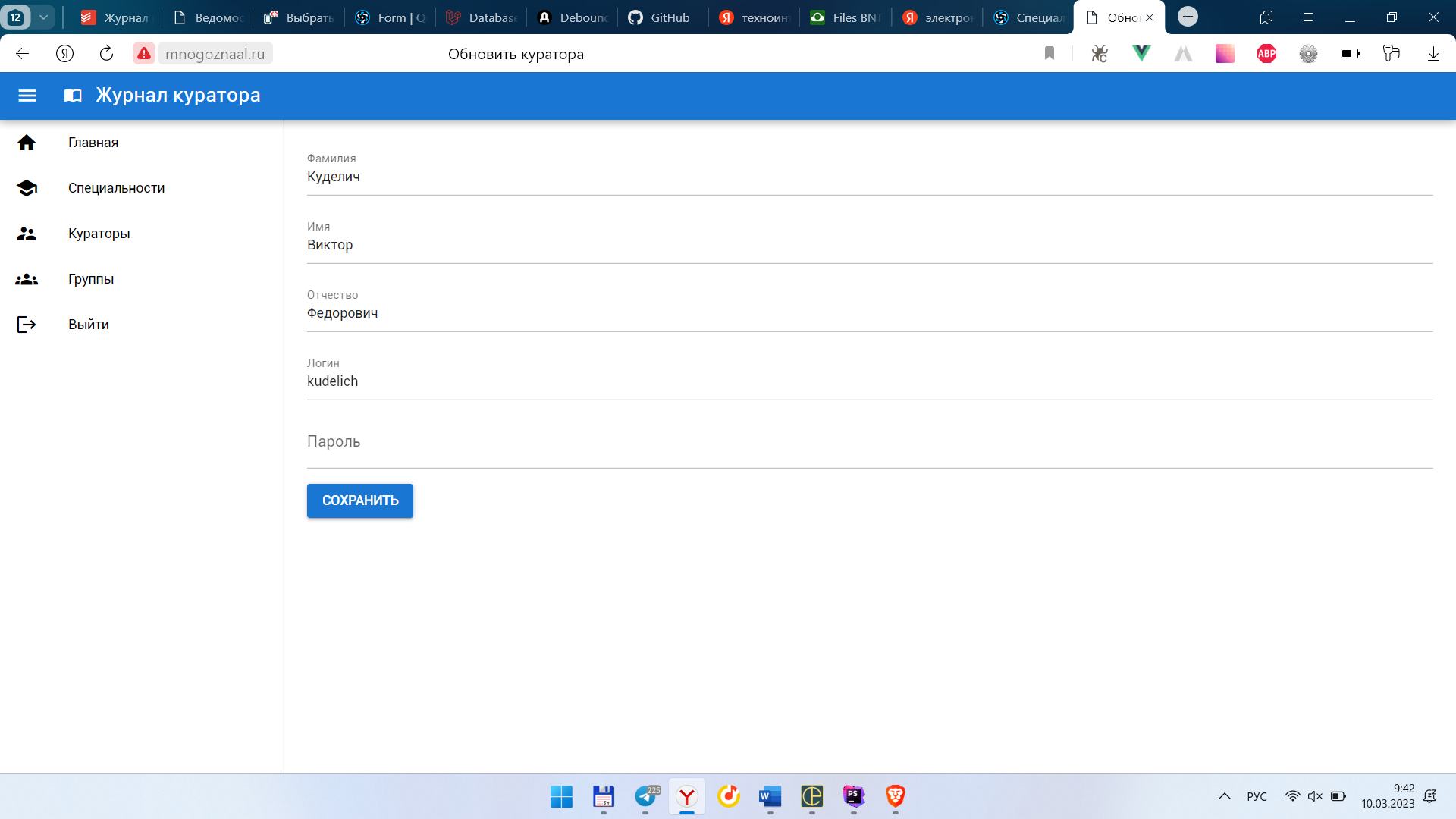


Рисунок 3.11 — Форма создания, редактирования куратора

При регистрации нового куратора и редактировании данных существующего куратора предусмотрена проверка на уникальность поля «Логин».

При переходе на форму «Группы» отобразится таблица, которая позволяет создавать, редактировать или удалять группы. Форма «Группы» представлена на рисунке 3.12.

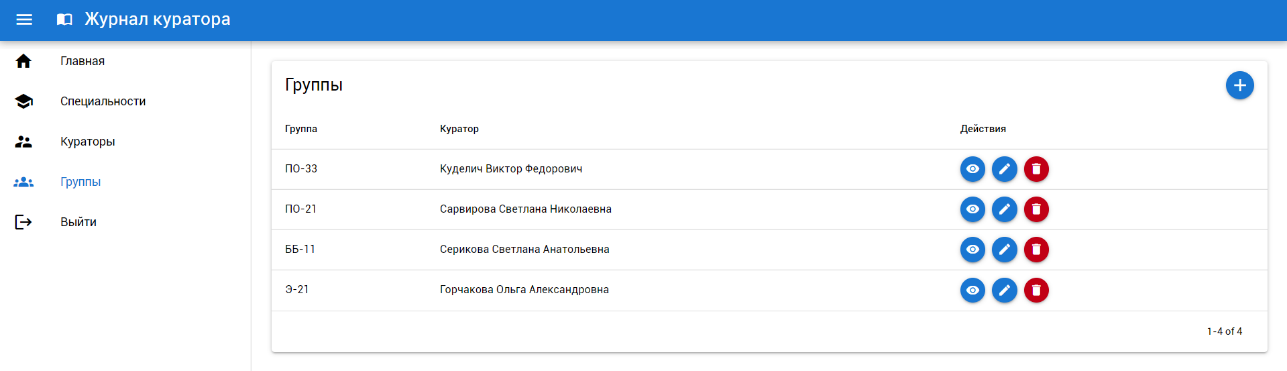


Рисунок 3.12 — Форма «Группы»

При создании или редактировании группы открывается форма, которая представлена на рисунке 3.13.

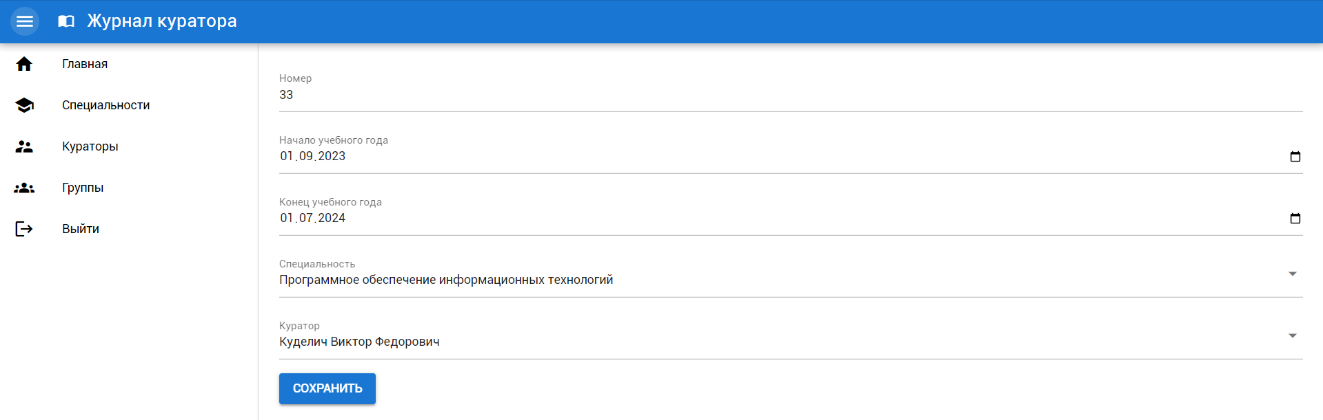


Рисунок 3.13 — Форма создания, редактирования группы

При посещении группы, путем нажатия на кнопку , открывается форма с учащимися, которая представлена на рисунке 3.14. Для удобства можно выбрать только определенные колонки для отображения, применять сортировку.

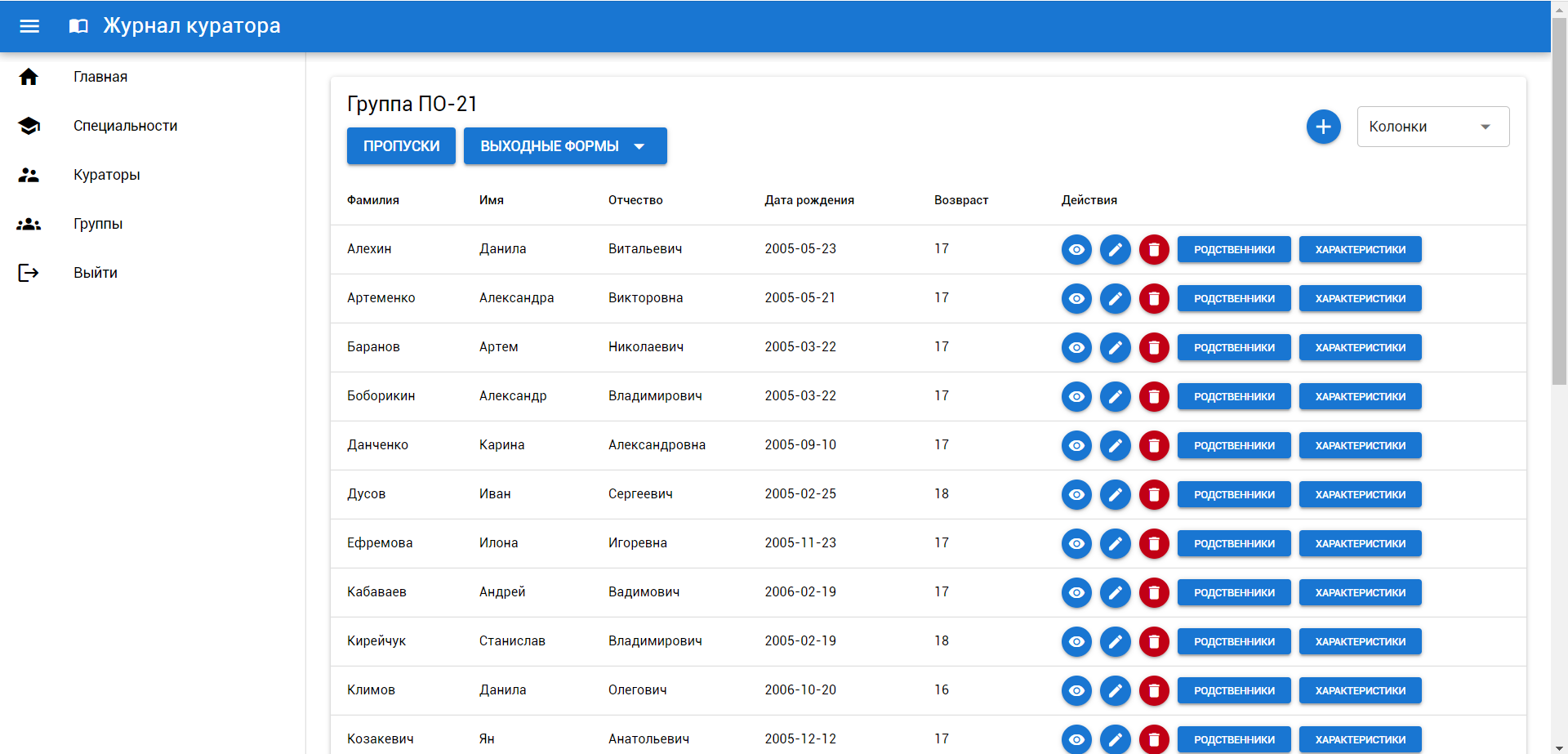


Рисунок 3.14 — Форма с учащимися

Для просмотра анкеты учащегося необходимо нажать на кнопку , после чего откроется форма анкеты учащегося, которая представлена на рисунке 3.15. Форма анкеты полностью соответствует форме «Карта персонифицированного учета» бумажной версии журнала куратора.

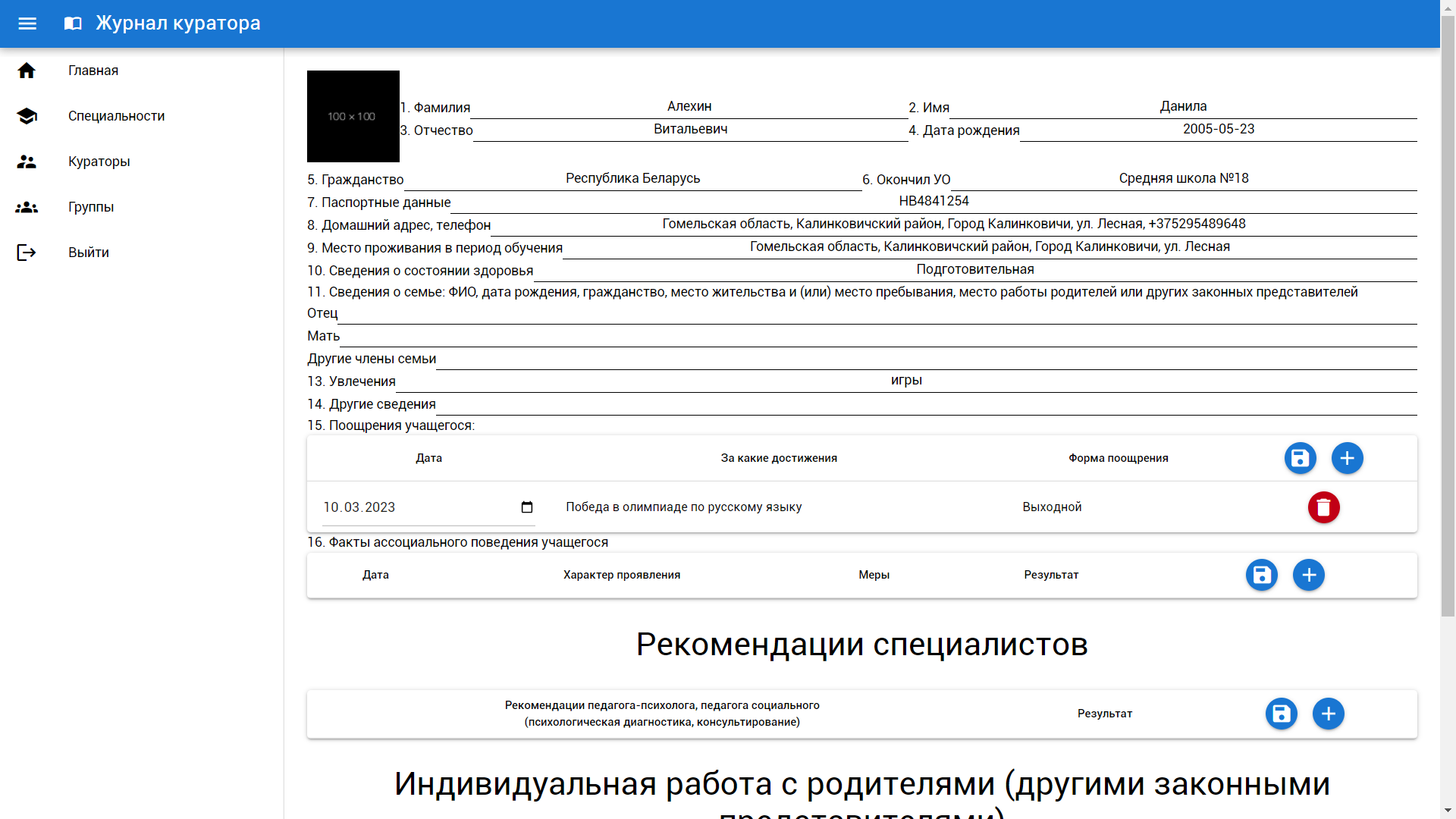


Рисунок 3.15 — Форма анкеты учащегося

Для добавления учащегося в группу необходимо нажать на кнопку , после чего откроется форма, которая представлена на рисунке 3.16.

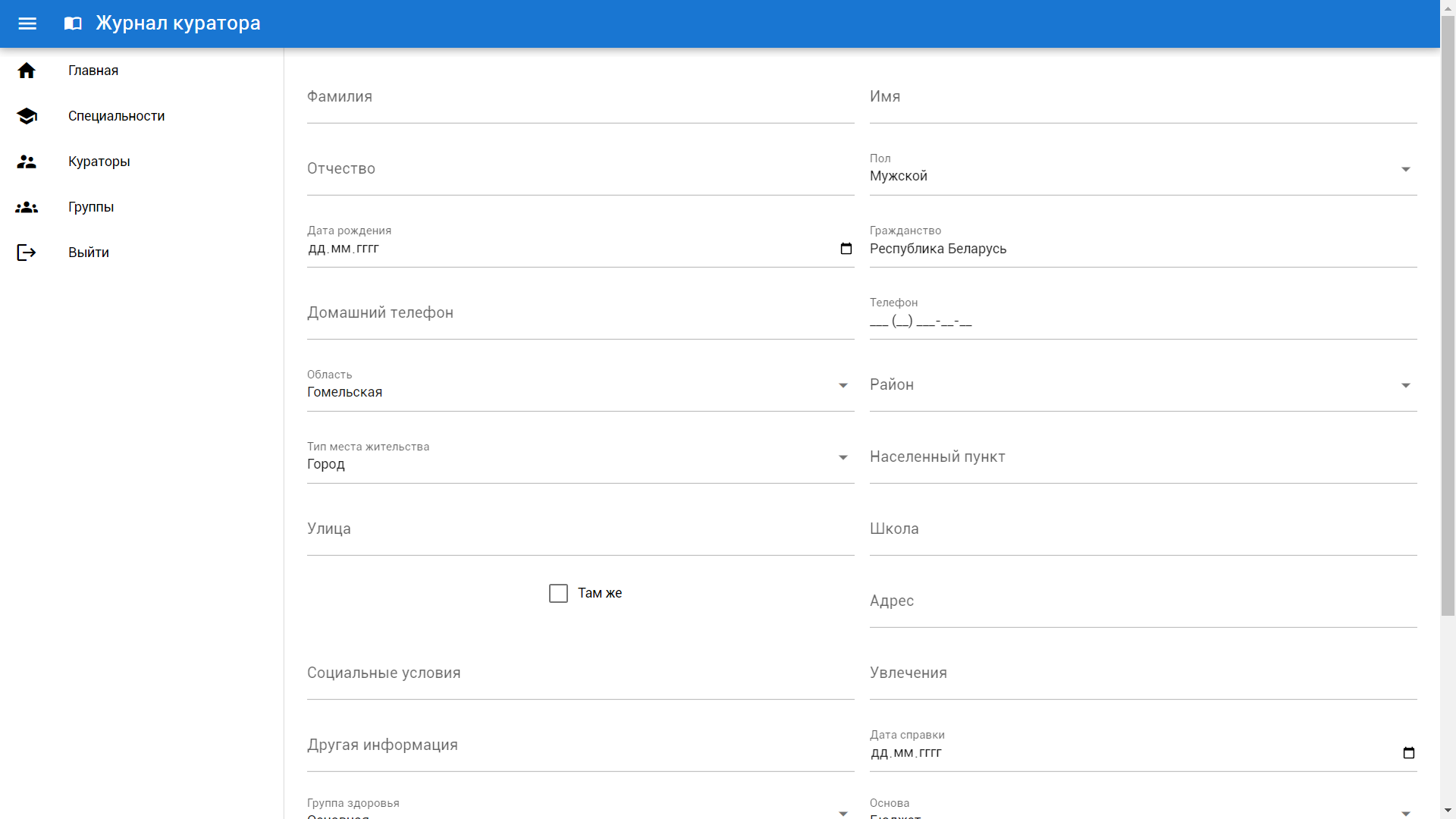


Рисунок 3.16 — Форма добавления учащегося

Журнал ведет учет родственников учащегося. Для перехода к форме «Родственники учащегося» необходимо нажать на кнопку «Родственники» на форме с учащимися. Форма «Родственники учащегося» представлена на рисунке 3.17.

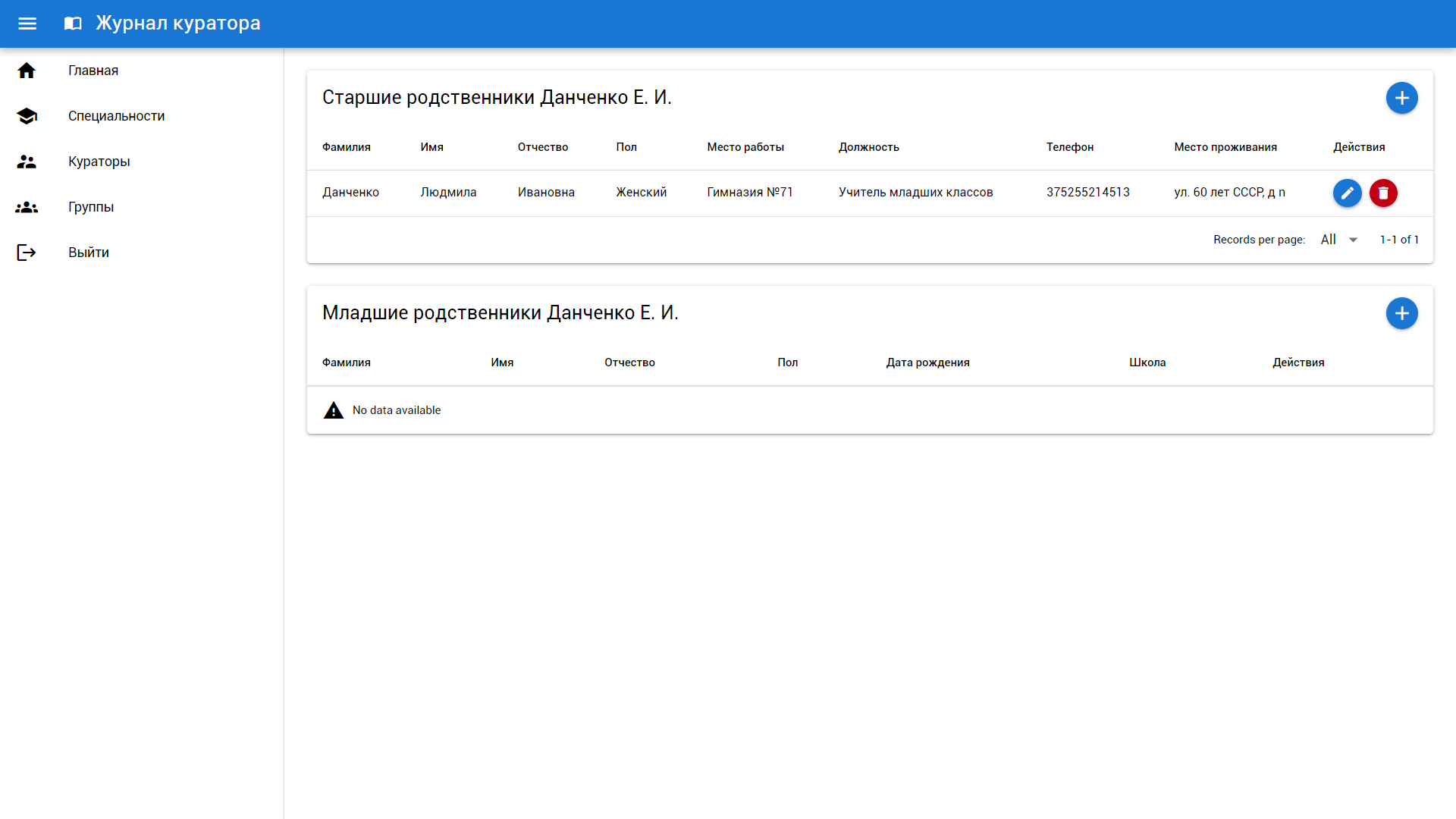


Рисунок 3.17 — Форма «Родственники учащегося»

Шаги создания, редактирования и удаления аналогичны для предыдущих форм и описаны выше.

Для работы с характеристиками учащегося и его семьи необходимо нажать на кнопку «Характеристики» на форме с учащимися. Форма «Характеристика учащегося» представлена на рисунке 3.18.

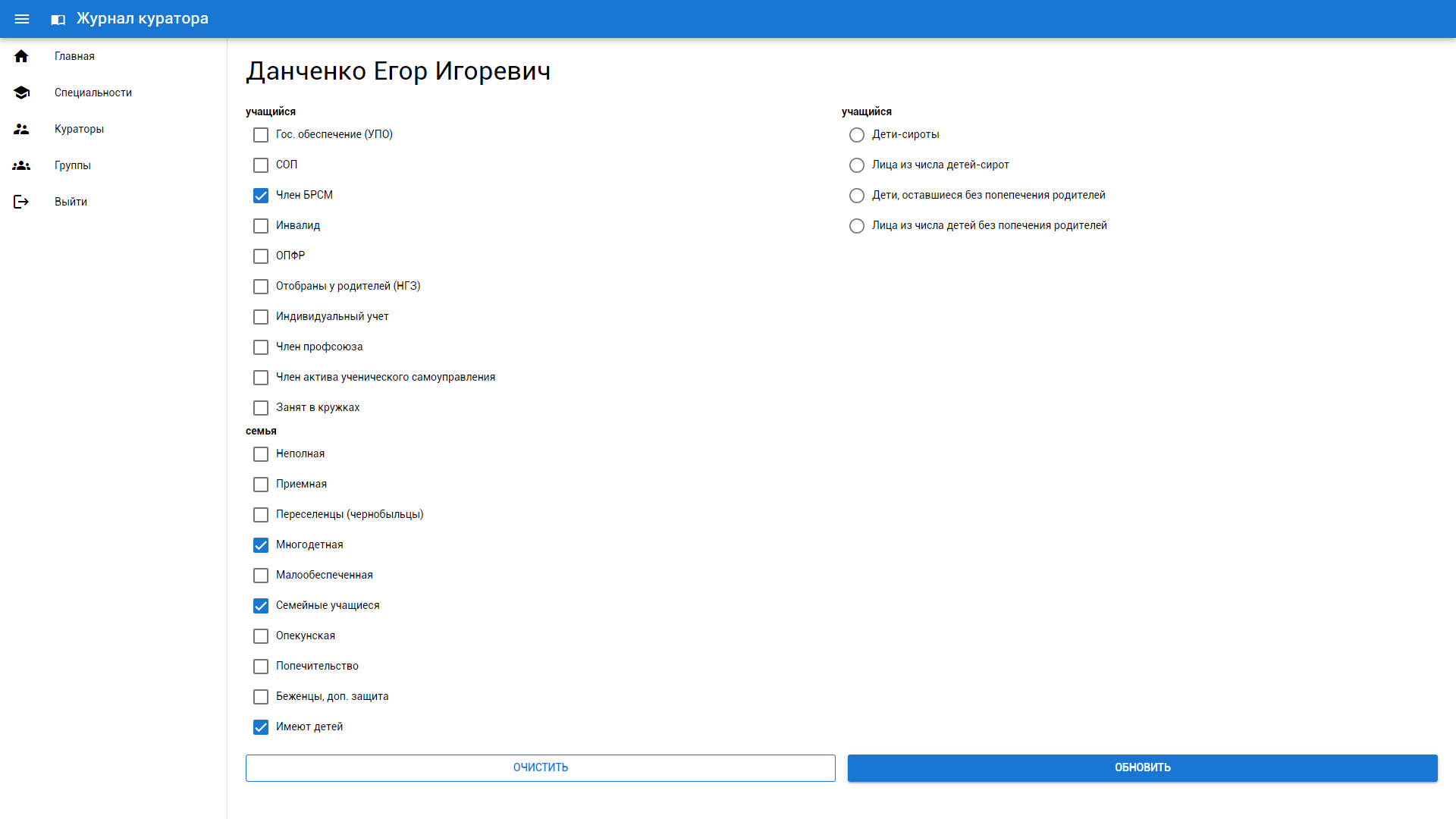


Рисунок 3.18 — Форма «Характеристика учащегося»

Приложение позволяет ускорить работу с отчетностью. Для этого существуют «Выходные формы», которые заполняются на основе существующих данных. Выходные формы строятся в разрезе группы и в разрезе колледжа. Пример выходных форм для группы представлен на рисунке 3.19.

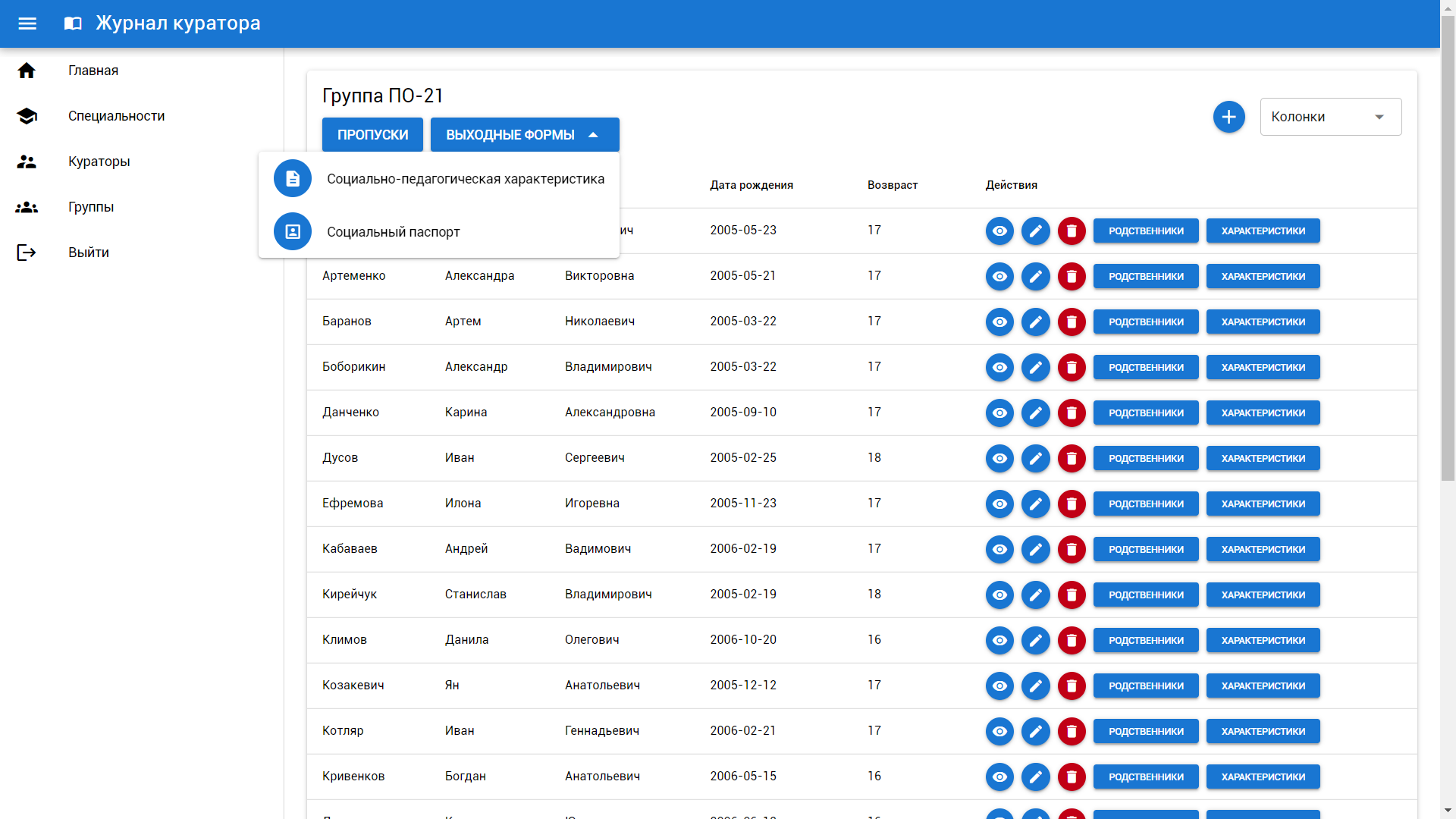


Рисунок 3.19 — Выходные формы для группы

При нажатии на пункт выпадающего меню начинается загрузка документа. Выходная форма «Социально-педагогическая характеристика» представлена на рисунке 3.20.

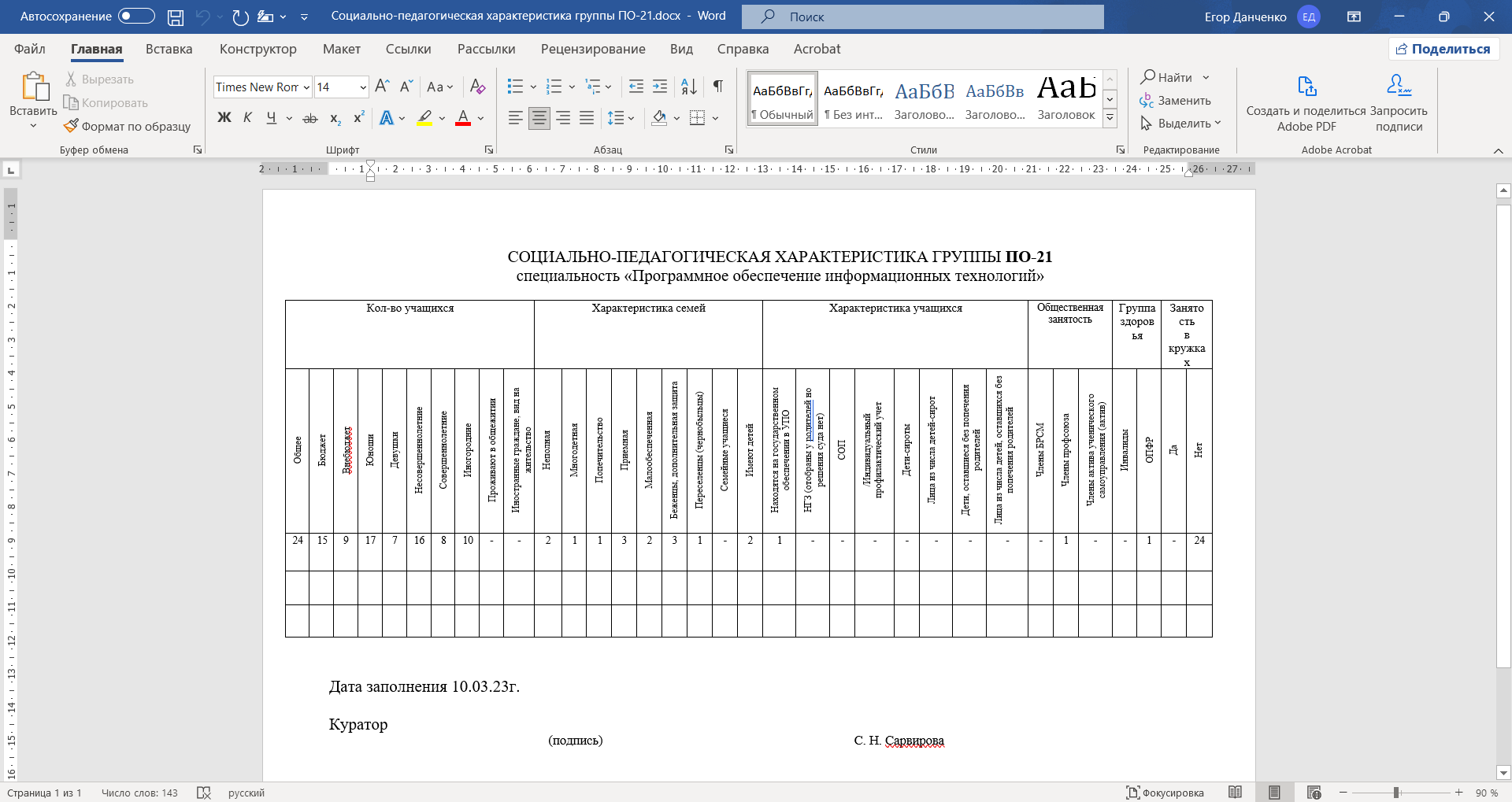


Рисунок 3.20 — Выходная форма «Социально-педагогическая характеристика»

Выходная форма «Социальный паспорт» представлена на рисунке 3.21.

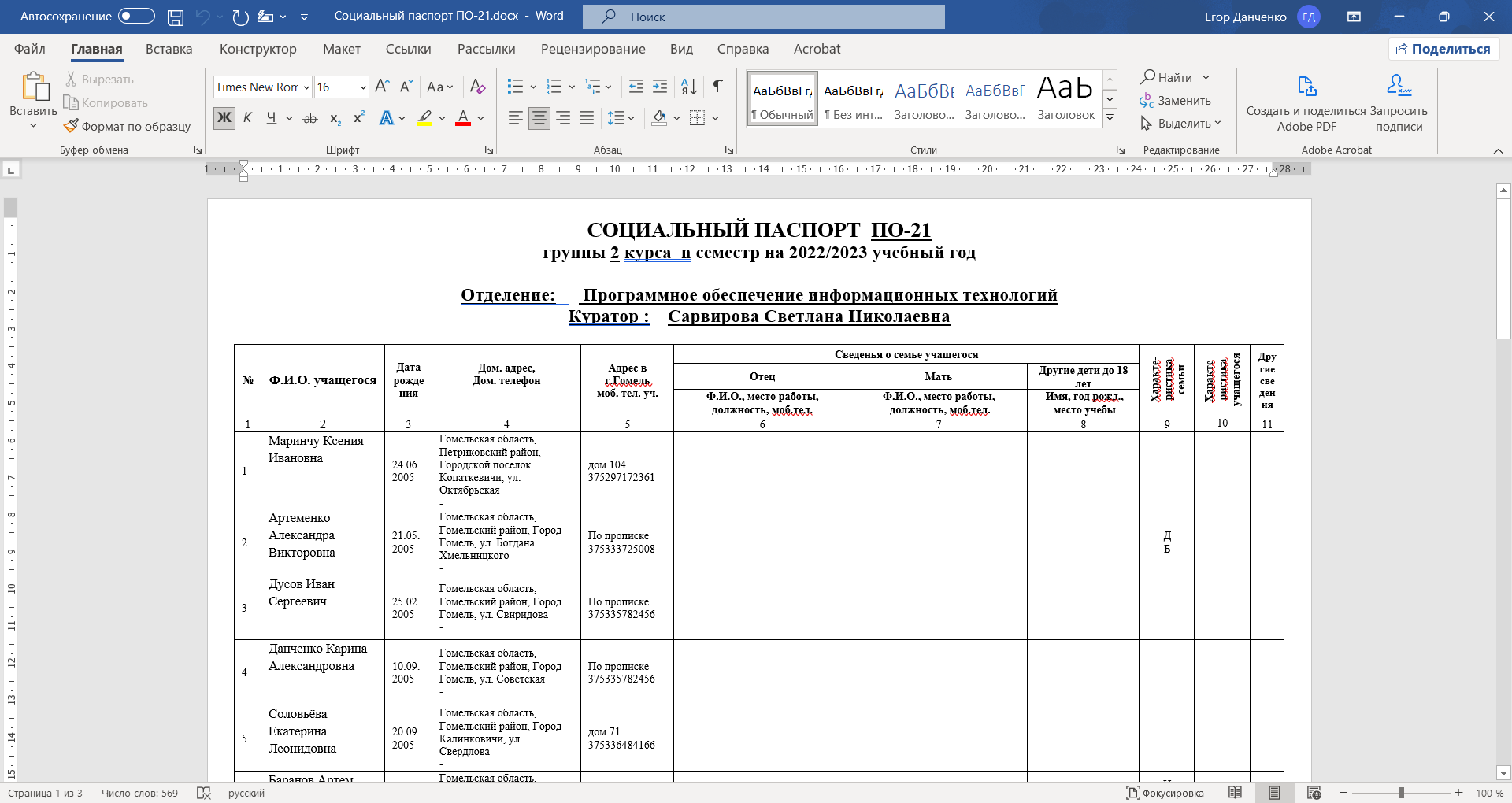


Рисунок 3.21 — Выходная форма «Социальный паспорт»

В электронной версии журнала куратора предусмотрен модуль «Пропуски», который позволяет ежедневно учитывать пропуски занятий учащихся и подсчитывать количество уважительных и неуважительных часов. Для этого куратору необходимо ежедневно вносить данные о пропусках.

Для учета пропусков необходимо нажать на кнопку «Пропуски» на странице с учащимися, после чего открываются пропуски на текущий месяц. Для перехода на предыдущий или следующий месяц используются кнопки со стрелками влево и вправо соответственно. Пропуски учитывают количество уважительных и неуважительных часов. Каждый месяц автоматически высчитанные пропуски сдаются в администрацию для учета. Страница с пропусками на текущий месяц представлена на рисунке 3.22.

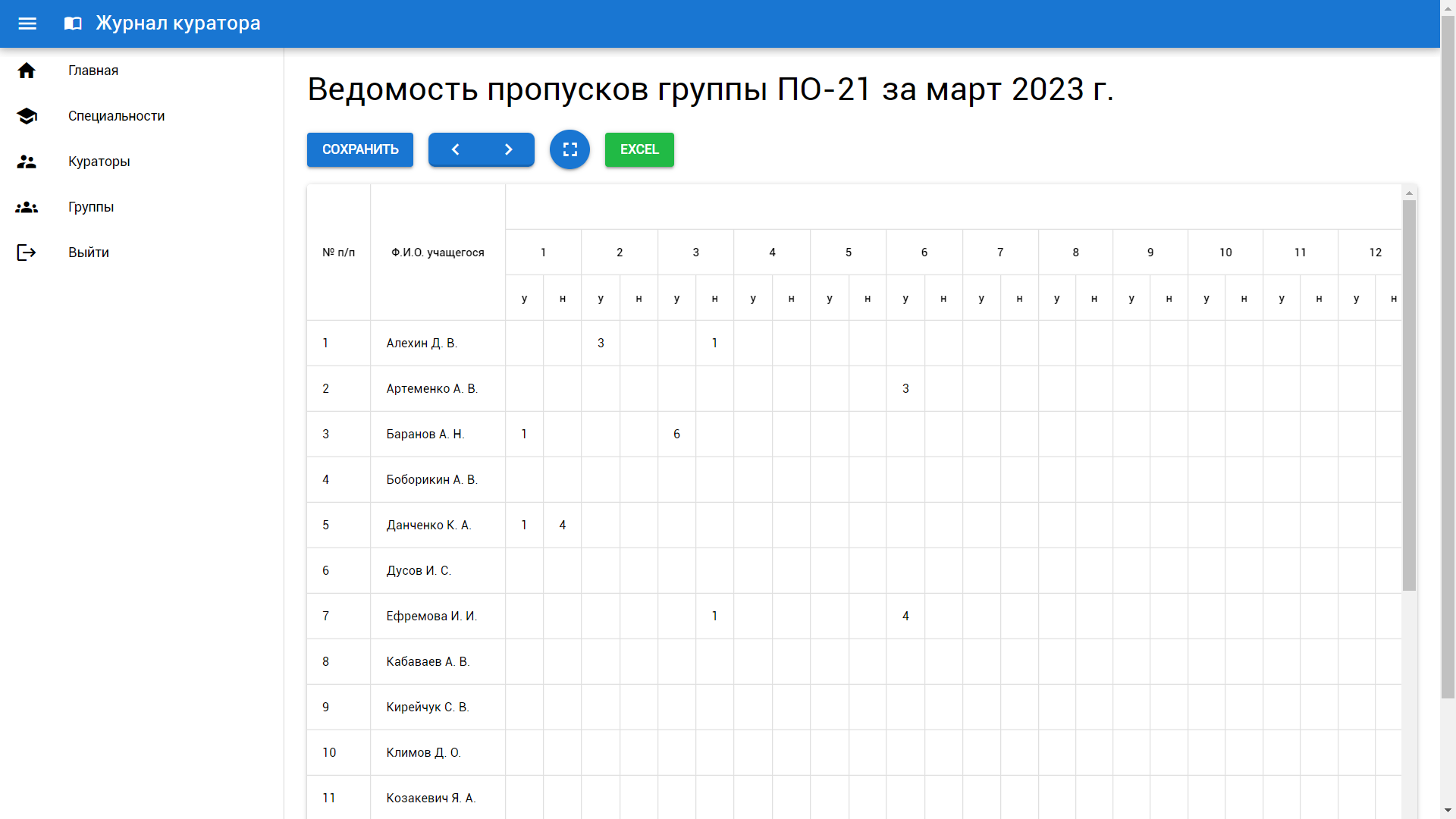


Рисунок 3.22 — Страница с пропусками

Для удобства создан полноэкранный режим, который позволяет развернуть таблицу на весь экран. Запускается нажатием на соответствующую кнопку. Таблица пропусков в полноэкранном режиме представлена на рисунке 3.23.

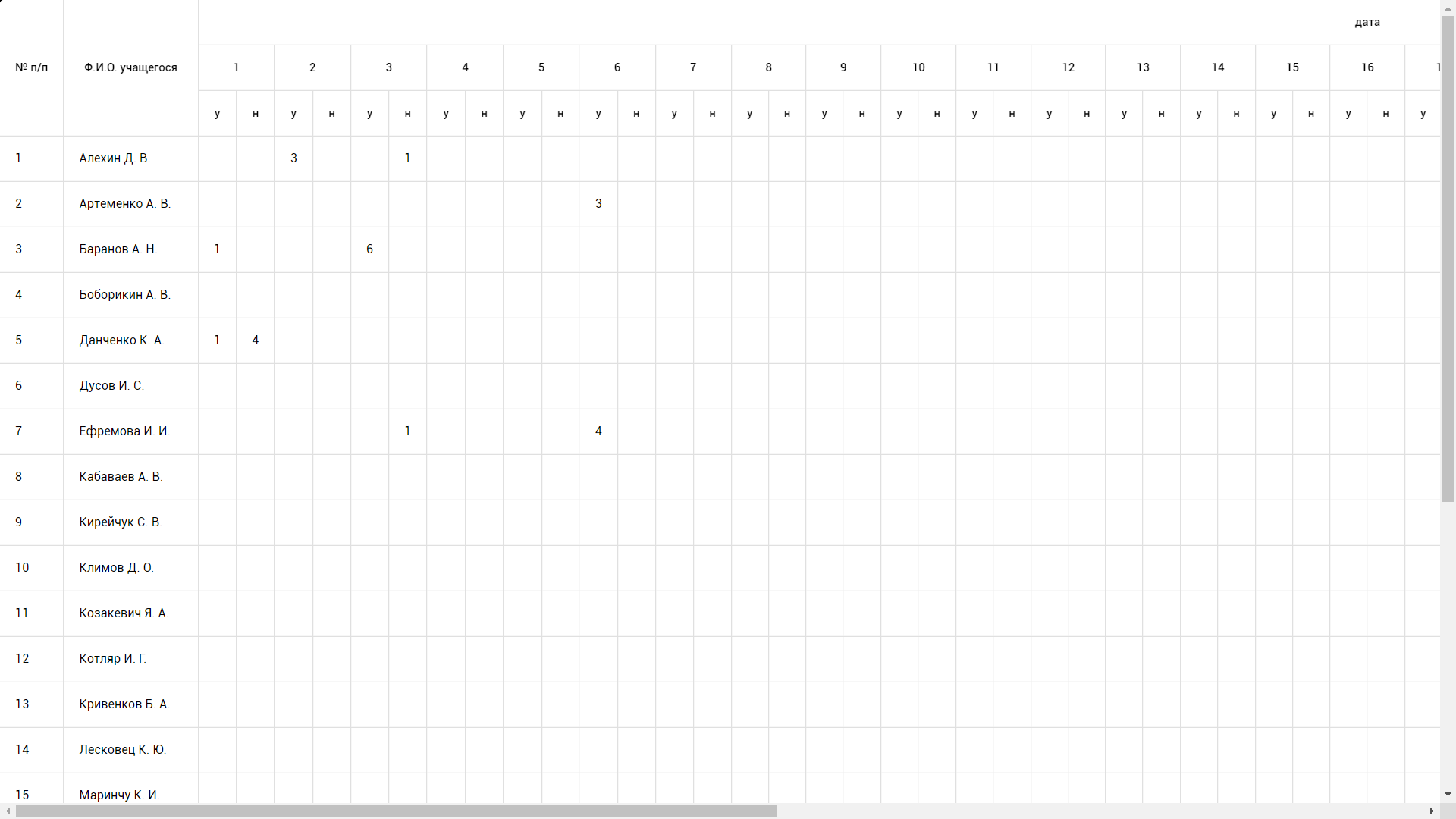


Рисунок 3.23 — Таблица пропусков в полноэкранном режиме

Для экспорта пропусков из веб-приложения в Excel необходимо нажать на кнопку «Excel», после чего документ загрузится на устройство. Загруженный документ представлен на рисунке 3.24.

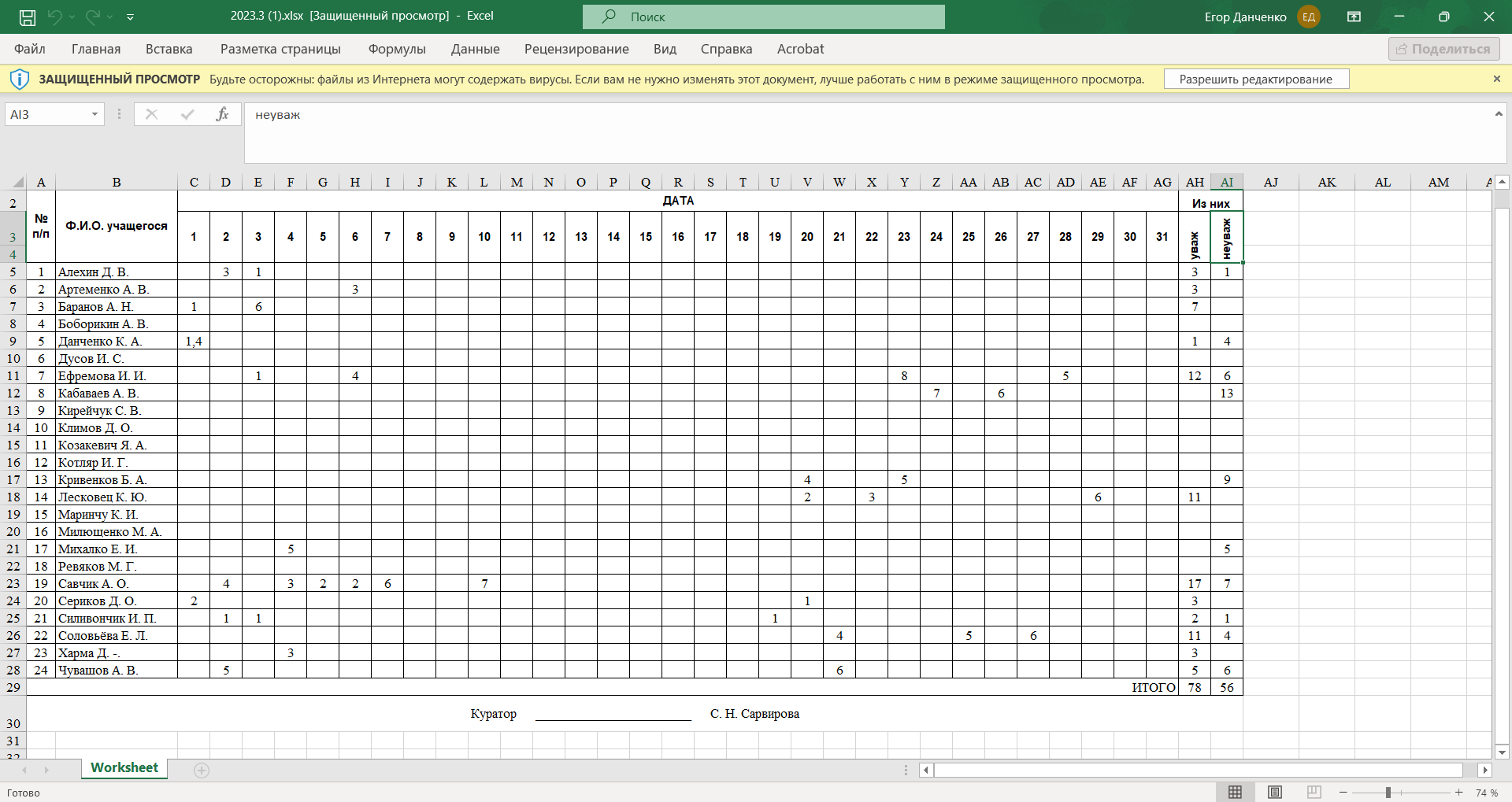


Рисунок 3.24 — Учет пропусков в Excel

Для учета оценок за месяц необходимо нажать на кнопку «Оценки» на странице с учащимися, после чего открываются оценки на текущий месяц. Работа с оценками аналогична с работой формы «Пропуски». Форма оценок представлена на рисунке 3.25.

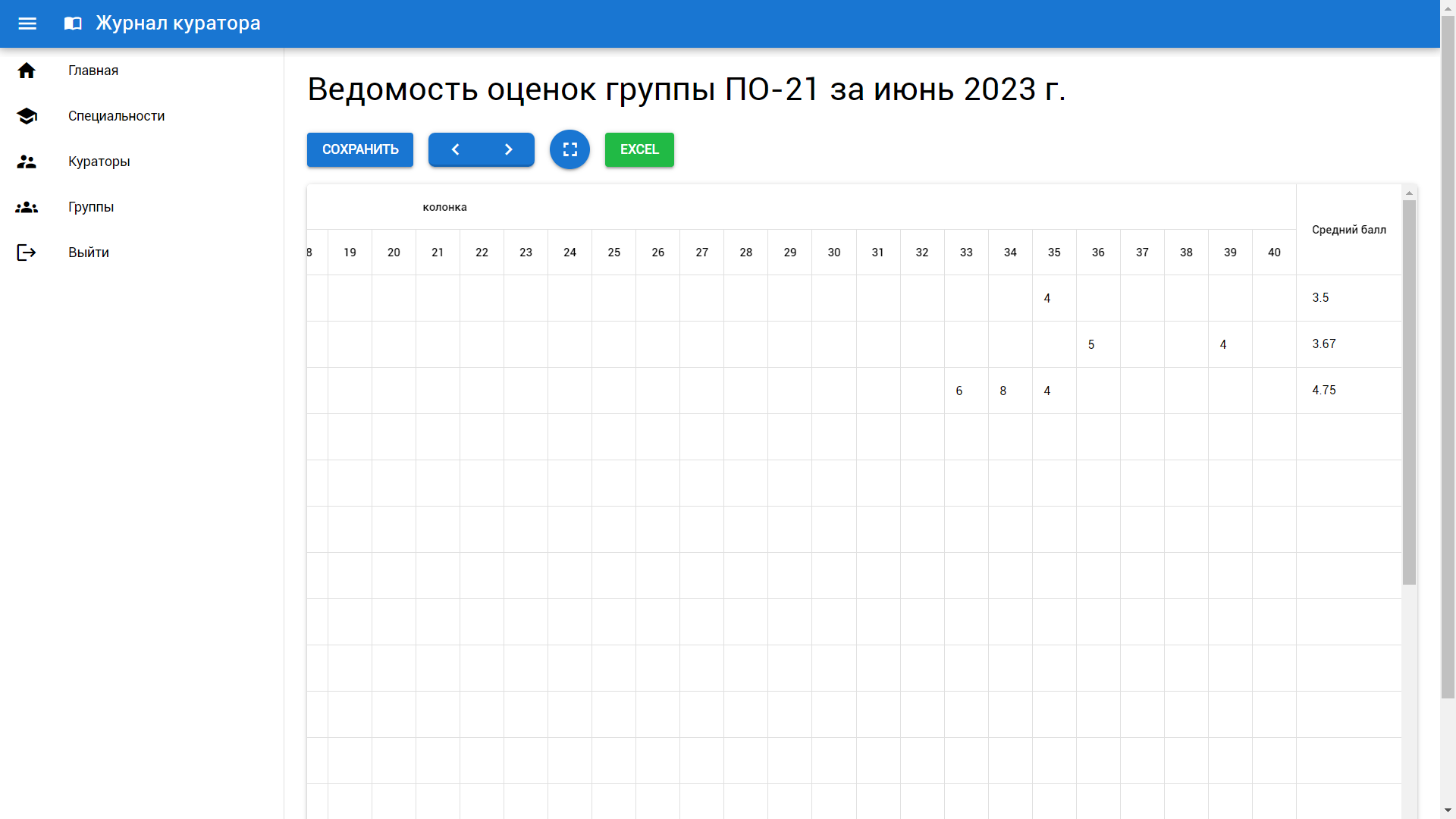


Рисунок 3.25 — Форма оценок

Каждая группа имеет разделы «Актив учебной группы» и «Состав совета группы», которые содержат в себе положения учащихся. Перейти к данным разделам можно через соответствующую кнопку на форме с учащимися. Форма с вышеперечисленными разделами представлена на рисунке 3.26.

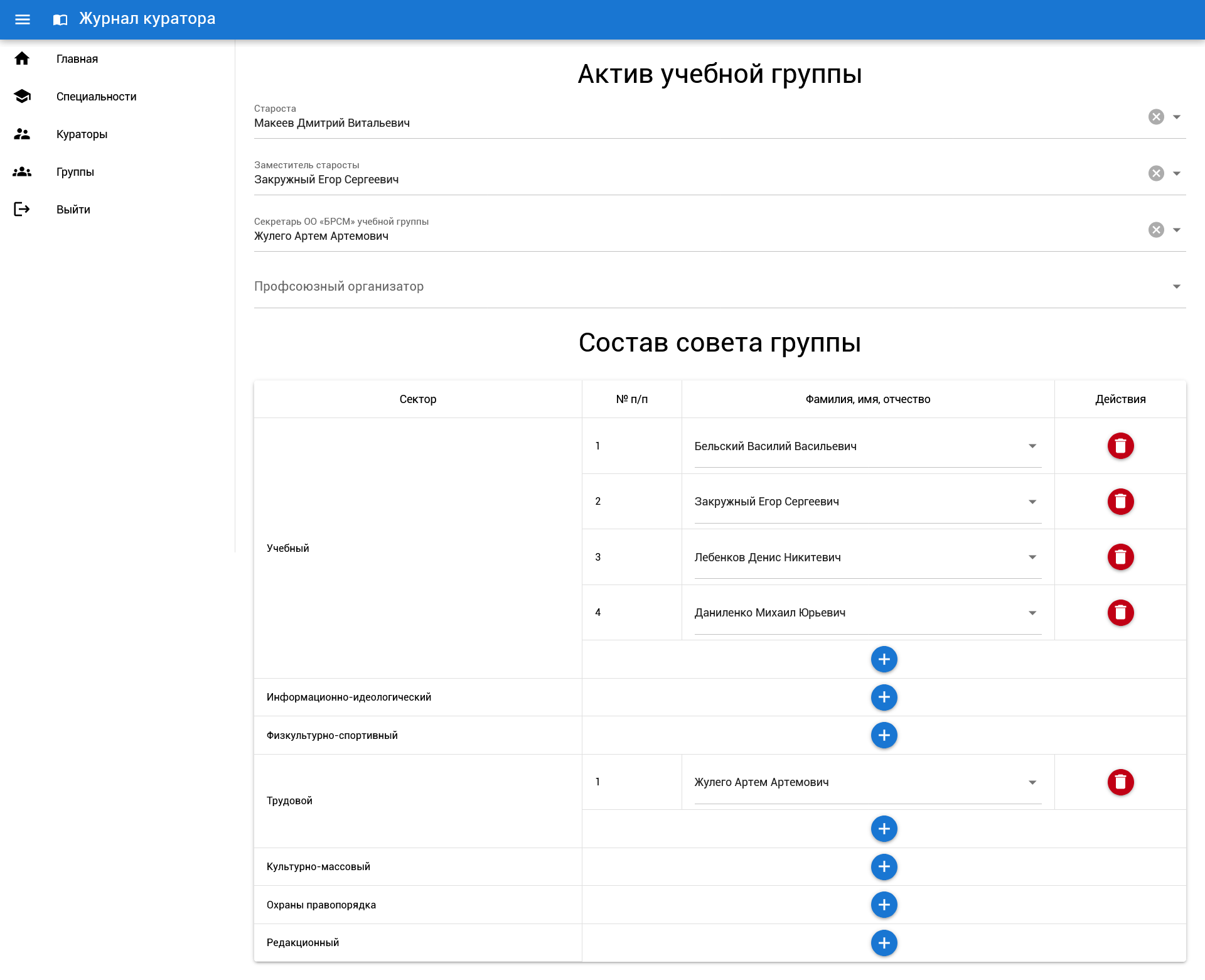


Рисунок 3.26 — Форма, содержащая «Актив учебной группы» и «Состав совета группы»

В журнале куратора предусмотрено наличие плана воспитательной работы и отчет о проделанной работе.

Для составления плана идеологической и воспитательной работы на форме «Учащиеся» нужно выбрать месяц в выпадающем списке, на который составляется план. Возможность выбора месяца из выпадающего списка представлена на рисунке 3.27.

Планы на месяц составляет куратор группы.

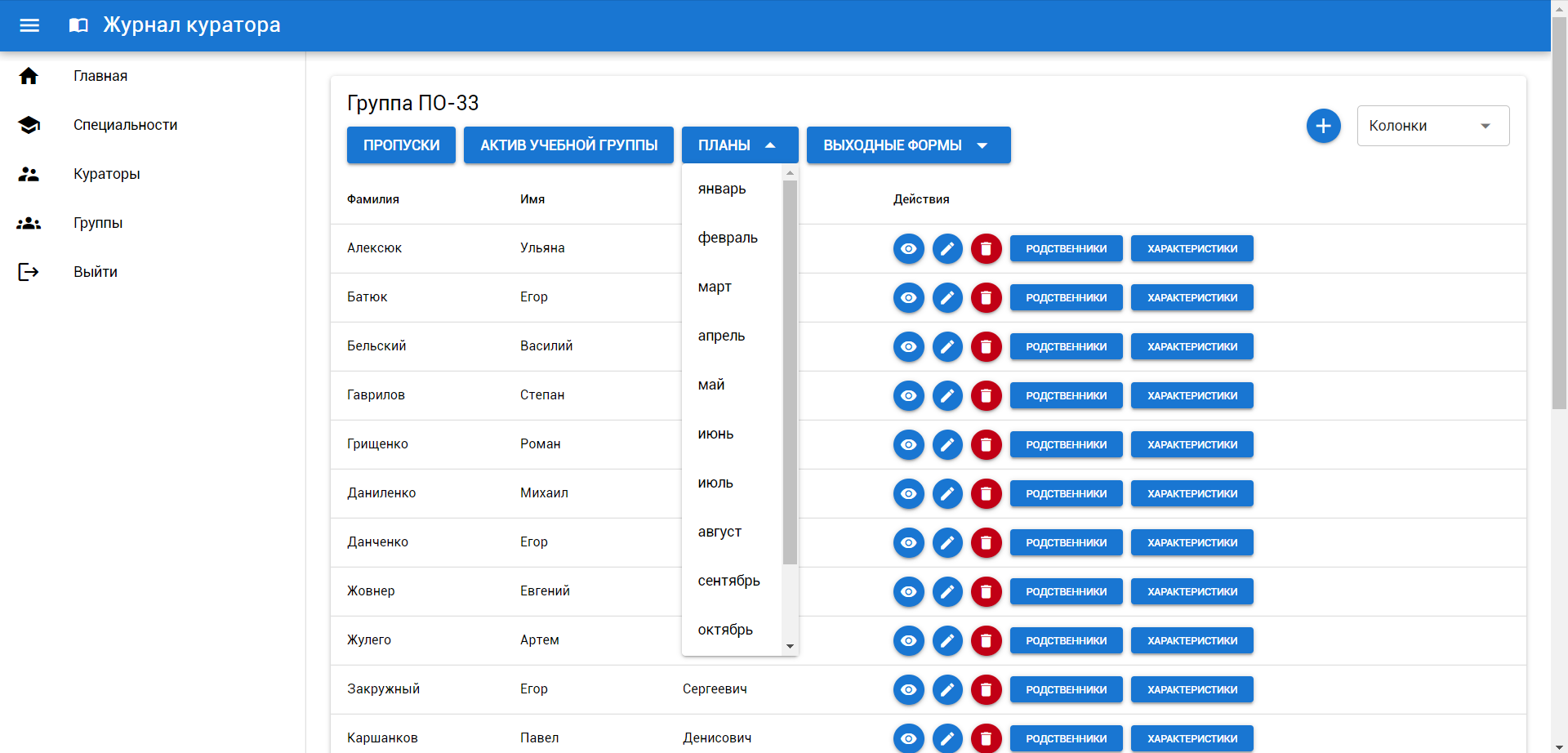


Рисунок 3.27 — Выпадающий список «Планы»

После выбора месяца открывается страница плана на выбранный месяц. Страница плана представлена на рисунке 3.28.

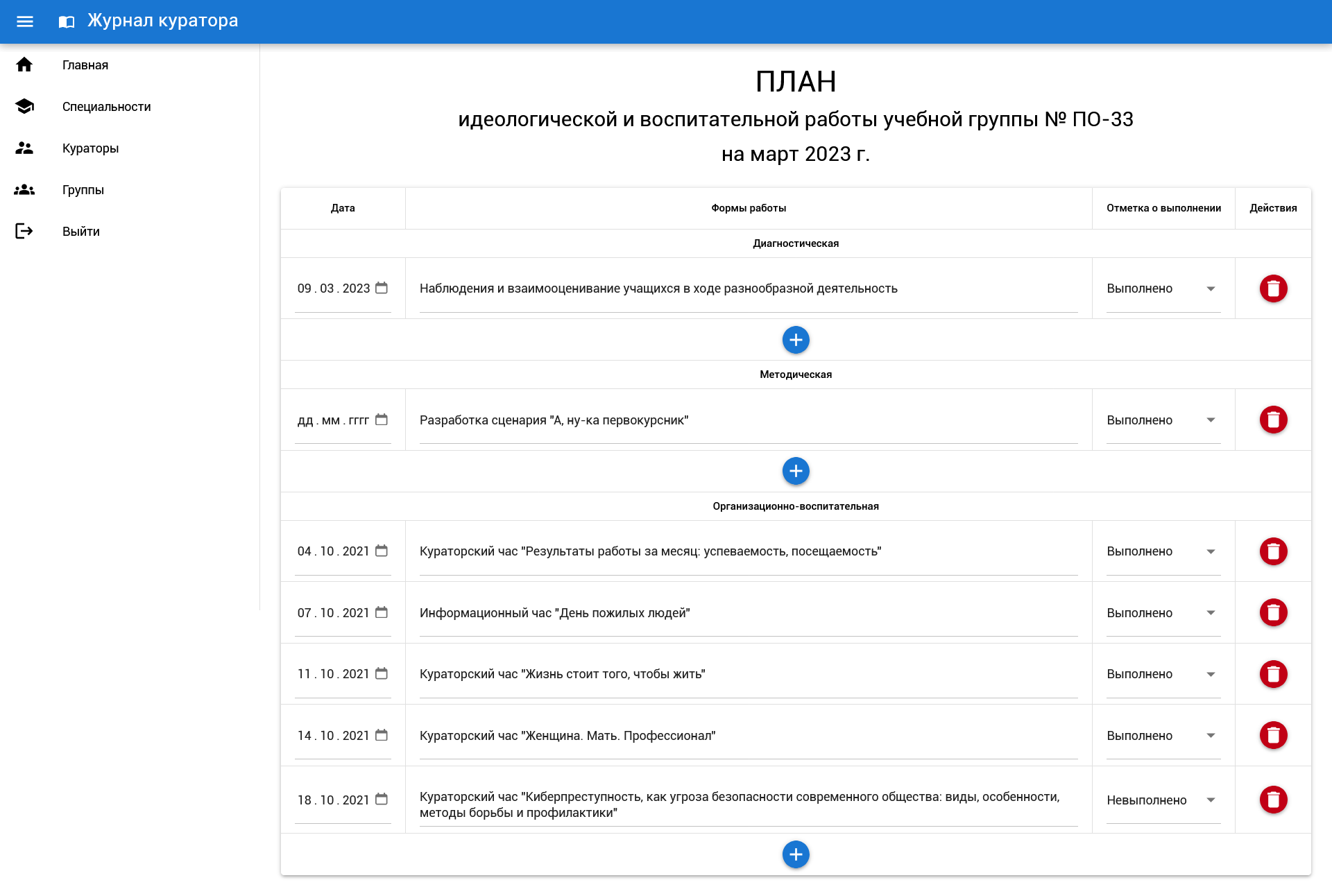


Рисунок 3.28 — Страница плана

# 4 Тестирование программного продукта

Тестирование программного обеспечения — процесс исследования, испытания программного продукта, имеющий своей целью проверку соответствия между реальным поведением программы и её ожидаемым поведением на конечном наборе тестов, выбранных определенным образом.

Для начала тестирования необходимо открыть программное средство. Появится главная форма программы. Тестирование будет проведено по всем таблицам и функциям. Для проверки работоспособности были произведены всевозможные манипуляции с данными, в ходе которые аномальное поведение или ошибки не обнаружены.

Важным приоритетом системы является безопасность. Приложение обеспечивает высокий уровень безопасности, благодаря реализации множества мер защиты от распространенных атак, таких как «XSS», «CSRF» и «SQL-инъекции», а также реализует управление доступом на основе ролей.

«XSS (Cross-Site Scripting)» — это тип атаки на веб-приложения, при которой злоумышленник внедряет вредоносный скрипт (обычно на JavaScript) на страницу веб-приложения, которую видит другой пользователь. Когда другой пользователь посещает эту страницу, вредоносный скрипт может собирать конфиденциальную информацию, такую как пароли, данные о сессии и другие личные данные [7].

Для защиты от «XSS-атак» используется технология фильтрации вывода данных, чтобы предотвратить вставку вредоносного кода на страницах сайта.

«CSRF (Cross-Site Request Forgery)» — это атака на приложение, при которой злоумышленник создает запрос к приложению от имени другого пользователя, который уже прошел аутентификацию на сайте.

Для защиты от «CSRF-атак» используется специальный механизм токенов, который создает уникальный идентификатор для каждого действия пользователя, что позволяет эффективно блокировать любые несанкционированные попытки изменения данных на сайте.

«SQL-инъекция» — атака, при которой заменяется параметр запроса, полученный, например, из «URL» строки [8].

Для предотвращения «SQL-инъекции» используются параметризованные запросы к базе данных и тщательно проверяются все входящие данные от пользователя. Это гарантирует, что никакой вредоносный код не будет выполнен в контексте базы данных.

При вводе некорректных данных программа предупреждает об этом пользователя и не производит никаких действий до исправления ошибки, результат действия представлен на рисунке 4.1.

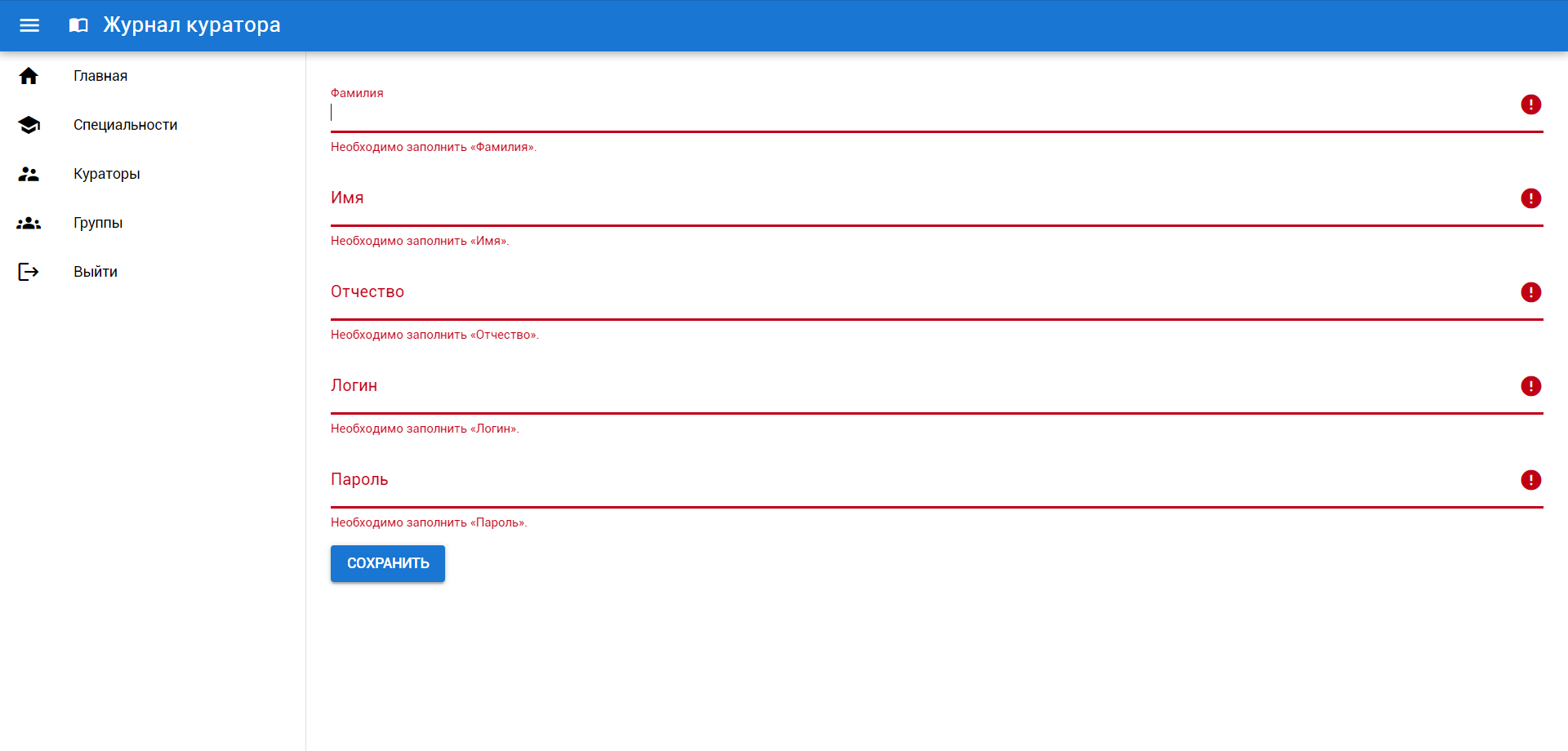


Рисунок 4.1 — Попытка добавления пустых значений

При попытке добавления или редактирования записи, где присутствует уникальное поле и используется значение, которое уже занято, программа предупреждает об этом пользователя и не производит никаких действий до исправления ошибки. Результат действия показан на рисунке 4.2.

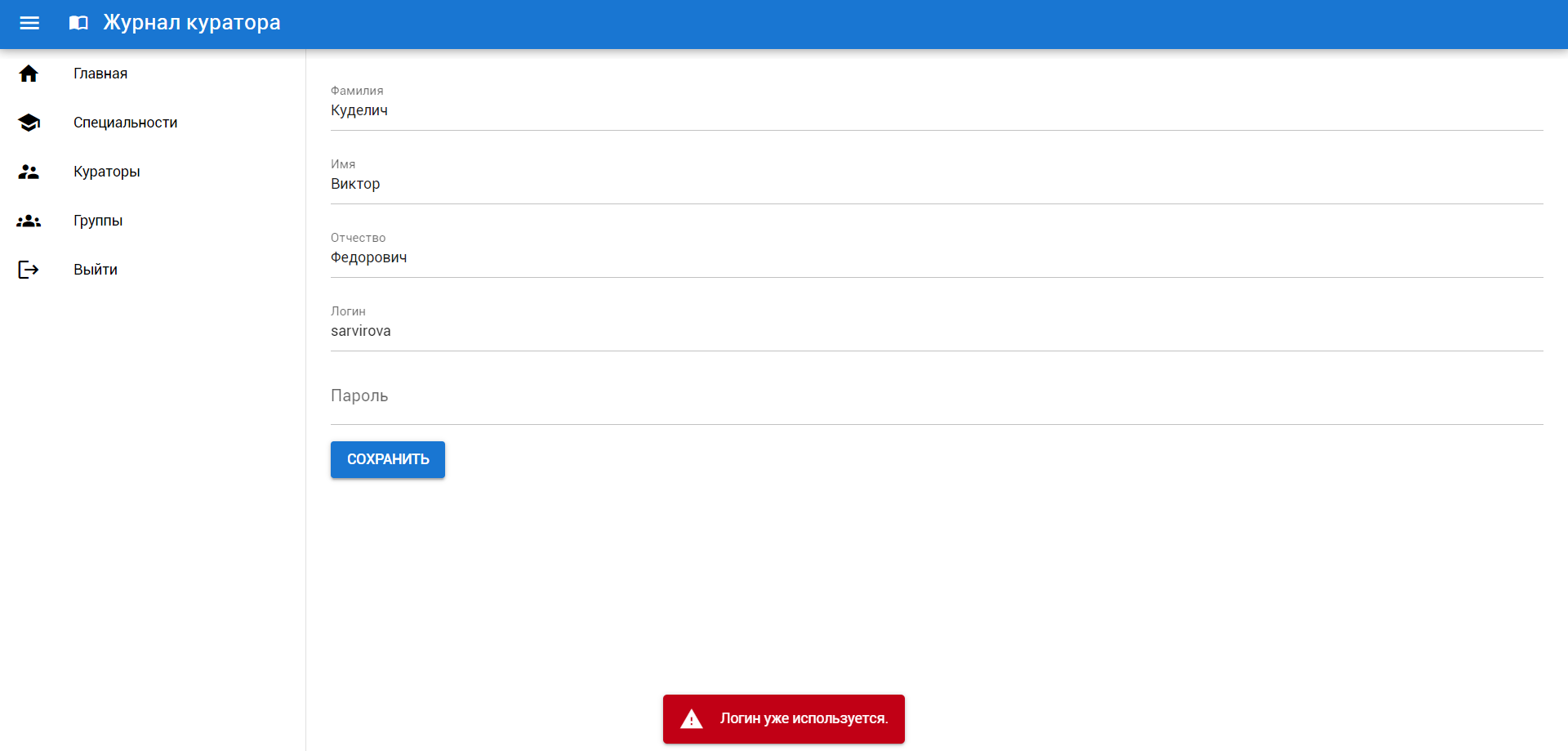


Рисунок 4.2 — Ошибка уникального поля

При попытке удаления записи, которые связаны с другими, выдается ошибка, которая показана на рисунке 4.3.

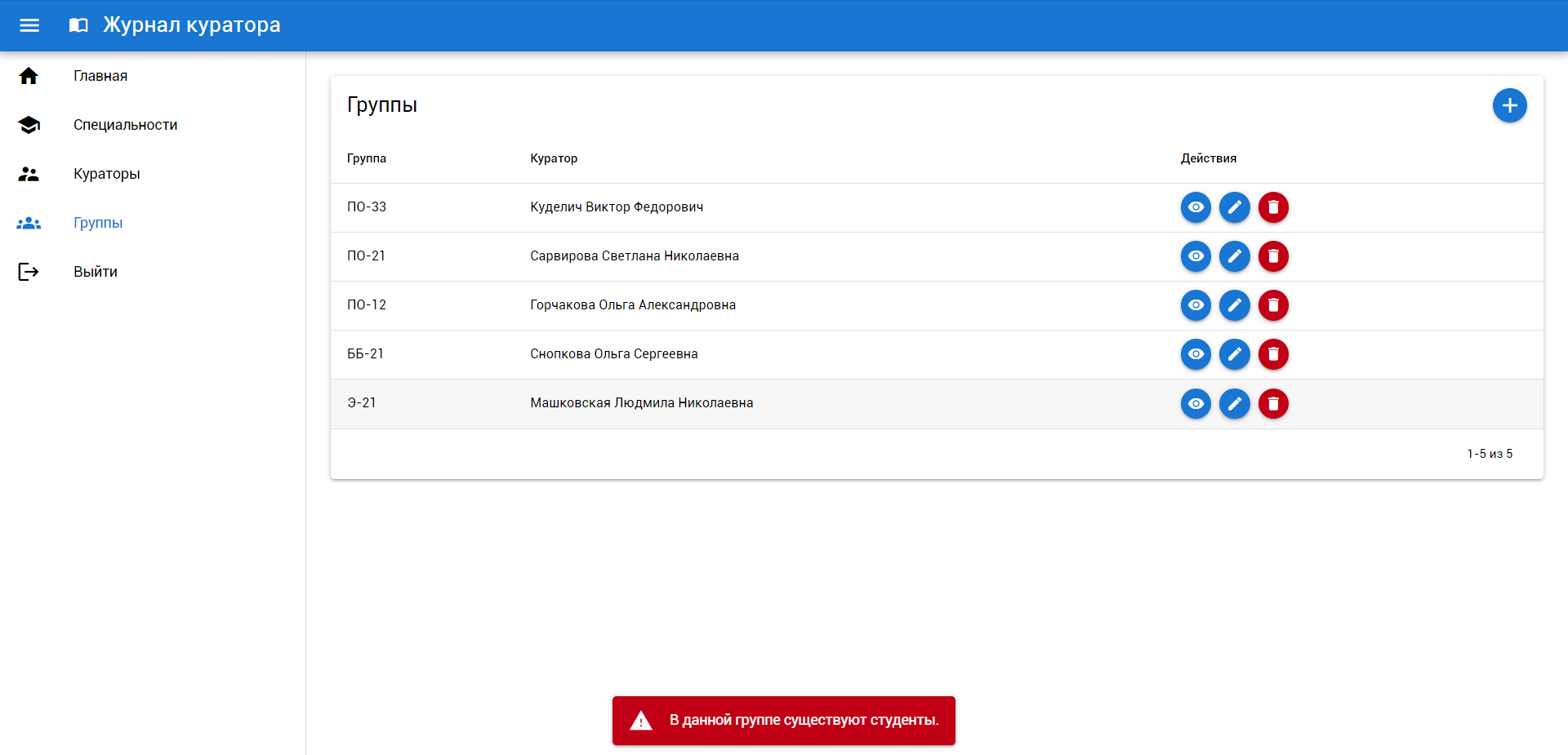


Рисунок 4.3 — Попытка удаления записи, которая связана с другими записями

# Заключение

В заключении стоит отметить, что перевод бумажной документации в электронный вид является актуальным направлением в современной образовательной среде.

В данной работе были рассмотрены проблемы перехода от бумажных документов к цифровым и возможности использования цифрового журнала куратора для более быстрой и точной обработки данных и составления отчетности. Были поставлены задачи по изучению структуры бумажных документов, их переводу в электронный формат и разработке приложения.

Приложение содержит следующие функции:

* учет (создание, редактирование, удаление, анализ) специальностей, кураторов, групп и учащихся;
* валидация данных на стороне клиента и сервера;
* защита от распространенных атак, таких как «XSS», «CSRF» и «SQLинъекции»;
* управление доступом на основе ролей;
* адаптация под все разрешения экрана;
* возможность экспорта данных в различные форматы для сдачи отчетности.

# Список использованных источников

1. Документация «Vue JS 3.0» [Электронный ресурс]. — Режим доступа: https://vuejs.org. — Дата доступа: 25.06.2023.
2. Visual Studio Code [Электронный ресурс]. — Режим доступа: https://code.visualstudio.com. — Дата доступа: 27.06.2023.
3. Коннолли, Т. Базы данных. Проектирование, реализация и сопровождение. / Т. Коннолли. — М.: Издательский дом «Вильяме», 2003. —  1440 с.
4. Медиа Skillfactory [Электронный ресурс]. — Режим доступа: https://blog.skillfactory.ru/glossary/interface. — Дата доступа: 27.06.2023.
5. Стаффер, М. Laravel Полное руководство / М. Стаффер. — СПб.: Питер, 2020. — 514 с.
6. Круг, C. Не заставляйте меня думать. / С. Круг. — Москва: Издательство «Э», 2017. — 256 c.
7. Веллинг, Л. Разработка веб-приложений с помощью PHP и MySQL. / Л. Веллинг. — Москва: Компьютерное издательство «Диалектика». — 770 с.
8. Фленов, М.Е. PHP глазами хакера. / М.Е. Фленов. — СПб.: БХВ-Петербург, 2016. —  247 c.

# Приложение а

(обязательное)

**Листинг программы**

Файл «AbsenceController.php».

<?php

namespace App\Http\Controllers;

use App\Models\Absence;

use App\Models\Group;

use Illuminate\Http\Request;

use Inertia\Inertia;

class AbsenceController extends Controller

{

    public function index()

    {

        abort(404);

    }

    public function create()

    {

        abort(404);

    }

    public function store(Request $request)

    {

        $rules = [

            'student\_id' => 'required',

            'group\_id' => 'required',

            'date' => 'required|date',

            'respectful' => 'required|numeric',

            'disrespectful' => 'required|numeric',

        ];

        if ($request->has('array')) {

            $arrayRules = [];

            foreach ($rules as $key => $value) {

                $arrayRules['array.\*.' . $key] = $value;

            }

            $validated = $request->validate(array\_merge(['array' => 'required|array'], $arrayRules));

            $foundGroups = [];

            foreach ($validated['array'] as $absence) {

                if (!isset($foundGroups[$absence['group\_id']])) {

                    $foundGroups[$absence['group\_id']] = Group::find($absence['group\_id']);

                }

                $this->authorize('view', $foundGroups[$absence['group\_id']]);

                Absence::updateOrCreate(

                    [

                        'student\_id' => $absence['student\_id'],

                        'group\_id' => $absence['group\_id'],

                        'date' => $absence['date'],

                    ],

                    $absence,

                );

            }

        } else {

            $validated = $request->validate($rules);

            $this->authorize('view', Group::find($validated['group\_id']));

            Absence::updateOrCreate(

                [

                    'student\_id' => $validated['student\_id'],

                    'group\_id' => $validated['group\_id'],

                    'date' => $validated['date'],

                ],

                $validated,

            );

        }

        return $request->headers->has('referer')

            ? Inertia::location($request->headers->get('referer'))

            : Inertia::location(route('home'));

    }

    public function show($id)

    {

        abort(404);

    }

    public function edit($id)

    {

        abort(404);

    }

    public function update(Request $request, $id)

    {

        abort(404);

    }

    public function destroy($id)

    {

        abort(404);

    }

}

Файл «CharacteristicsController.php».

<?php

namespace App\Http\Controllers;

use App\Models\Characteristic;

use App\Models\Group;

use App\Models\Student;

use App\Models\User;

use App\Services\CharacteristicService;

use Inertia\Inertia;

use Illuminate\Http\Request;

class CharacteristicController extends Controller

{

    public function show(Request $request, $characteristic)

    {

        $validated = $request->validate([

            'group\_id' => 'numeric',

        ]);

        if (isset($validated['group\_id'])) {

            Group::findOrFail($validated['group\_id']);

        }

        $isCharacteristicVirtual = !is\_numeric($characteristic);

        if ($isCharacteristicVirtual) {

            if (!CharacteristicService::hasVirtualCharacteristic($characteristic)) {

                abort(404);

            }

            $characteristicTitle = CharacteristicService::getVirtualCharacteristic($characteristic)['label'];

            $studentQuery = Student::orderBy('surname')

                ->orderBy('name')

                ->orderBy('patronymic');

            if (isset($validated['group\_id'])) {

                $studentQuery->whereHas('groups', function ($query) use ($validated) {

                    $query->where('groups.id', $validated['group\_id']);

                });

            }

            $students = CharacteristicService::getCharacteristicMapItemCollection(

                $studentQuery->get(),

                $characteristic,

            )->values();

        } else {

            $characteristic = Characteristic::find($characteristic);

            if (!$characteristic) {

                abort(404);

            }

            $characteristicTitle = $characteristic->title;

            $students = $characteristic

                ->students()

                ->with('groups')

                ->orderBy('surname')

                ->orderBy('name')

                ->orderBy('patronymic');

            if (isset($validated['group\_id'])) {

                $students->whereHas('groups', function ($query) use ($validated) {

                    $query->where('groups.id', $validated['group\_id']);

                });

            }

            $students = $students->get()->values();

        }

        return Inertia::render('Characteristic/Show', compact('characteristicTitle', 'students'));

    }

}

Файл «CollegeController.php».

<?php

namespace App\Http\Controllers;

use App\Models\Student;

use App\Services\StudentService;

use PhpOffice\PhpWord\TemplateProcessor;

class CollegeController extends Controller

{

    public function socialPedagogicalCharacteristics()

    {

        $students = Student::with('characteristics')->get();

        $budgetApprenticeship = StudentService::getBudget($students);

        $boys = StudentService::getBoys($students);

        $adult = StudentService::getAdult($students);

        $studentsInClubs = $students->filter(function ($student) {

            return $student->characteristics->where('id', 14)->count() > 0;

        });

        $data = [

            $students->count(),

            $budgetApprenticeship->count(),

            $students->count() - $budgetApprenticeship->count(),

            $boys->count(),

            $students->count() - $boys->count(),

            $students->count() - $adult->count(),

            $adult->count(),

            StudentService::getOutOfTown($students)->count(),

            StudentService::getDorm($students)->count(),

            StudentService::getForeign($students)->count(),

            StudentService::withSingleParent($students)->count(),

            StudentService::withLargeFamily($students)->count(),

            StudentService::withTrusteeship($students)->count(),

            StudentService::withFosterFamily($students)->count(),

            StudentService::withLowIncomeFamily($students)->count(),

            StudentService::withRefugeeFamily($students)->count(),

            StudentService::withSettlerFamily($students)->count(),

            StudentService::withFamilyEducation($students)->count(),

            StudentService::withChildrenFamily($students)->count(),

            $students

                ->filter(function ($student) {

                    return $student->characteristics->where('id', 1)->count() > 0;

                })

                ->count(),

            $students

                ->filter(function ($student) {

                    return $student->characteristics->where('id', 8)->count() > 0;

                })

                ->count(),

            $students

                ->filter(function ($student) {

                    return $student->characteristics->where('id', 2)->count() > 0;

                })

                ->count(),

            $students

                ->filter(function ($student) {

                    return $student->characteristics->where('id', 9)->count() > 0;

                })

                ->count(),

            $students

                ->filter(function ($student) {

                    return $student->characteristics->where('id', 3)->count() > 0;

                })

                ->count(),

            $students

                ->filter(function ($student) {

                    return $student->characteristics->where('id', 4)->count() > 0;

                })

                ->count(),

            $students

                ->filter(function ($student) {

                    return $student->characteristics->where('id', 10)->count() > 0;

                })

                ->count(),

            $students

                ->filter(function ($student) {

                    return $student->characteristics->where('id', 11)->count() > 0;

                })

                ->count(),

            $students

                ->filter(function ($student) {

                    return $student->characteristics->where('id', 5)->count() > 0;

                })

                ->count(),

            $students

                ->filter(function ($student) {

                    return $student->characteristics->where('id', 12)->count() > 0;

                })

                ->count(),

            $students

                ->filter(function ($student) {

                    return $student->characteristics->where('id', 13)->count() > 0;

                })

                ->count(),

            $students

                ->filter(function ($student) {

                    return $student->characteristics->where('id', 6)->count() > 0;

                })

                ->count(),

            $students

                ->filter(function ($student) {

                    return $student->characteristics->where('id', 7)->count() > 0;

                })

                ->count(),

            $studentsInClubs->count(),

            $students->count() - $studentsInClubs->count(),

        ];

        $templateProcessor = new TemplateProcessor(

            base\_path('resources/templates/college-social-pedagogical-characteristics.docx'),

        );

        $templateProcessor->setValues([

            'fillDate' => now()->format('d.m.y') . 'г.',

        ]);

        foreach ($data as $i => $column) {

            $templateProcessor->setValue($i, $column <= 0 ? '-' : $column);

        }

        $tempFile = tempnam(sys\_get\_temp\_dir(), 'phpword');

        $templateProcessor->saveAs($tempFile);

        return response()

            ->download($tempFile, 'Социально-педагогическая характеристика учащихся колледжа.docx')

            ->deleteFileAfterSend();

    }

}

Файл «CuratorController.php».

<?php

namespace App\Http\Controllers;

use App\Models\Curator;

use App\Models\User;

use Illuminate\Http\Request;

use Illuminate\Support\Arr;

use Illuminate\Support\Facades\DB;

use Illuminate\Support\Facades\Hash;

use Illuminate\Support\Facades\Redirect;

use Illuminate\Validation\ValidationException;

use Inertia\Inertia;

class CuratorController extends Controller

{

    public function \_\_construct()

    {

        $this->authorizeResource(Curator::class, 'curator');

    }

    public function index()

    {

        $query = Curator::with('user');

        if (!request()->user()->is\_admin) {

            $query->where('id', request()->user()->curator->id);

        }

        $curatorPagination = $query->paginate();

        return Inertia::render('Curator/Index', [

            'curatorPagination' => $curatorPagination,

            'can' => [

                'create\_curators' => request()

                    ->user()

                    ->can('create', Curator::class),

            ],

        ]);

    }

    public function create()

    {

        return Inertia::render('Curator/Create');

    }

    public function store(Request $request)

    {

        $validated = $request->validate([

            'surname' => 'required',

            'name' => 'required',

            'patronymic' => 'required',

            'login' => 'required',

            'password' => 'required|min:8',

        ]);

        if (User::where('login', $validated['login'])->exists()) {

            throw ValidationException::withMessages(['Логин уже используется.']);

        }

        DB::transaction(function () use ($validated) {

            $user = User::create([

                'login' => $validated['login'],

                'password' => Hash::make($validated['password']),

            ]);

            Curator::create(array\_merge(Arr::except($validated, ['login', 'password']), ['user\_id' => $user->id]));

        });

        return Inertia::location(route('curators.index'));

    }

    public function edit(Curator $curator)

    {

        $curator->load('user');

        return Inertia::render('Curator/Update', compact('curator'));

    }

    public function update(Request $request, Curator $curator)

    {

        $validated = $request->validate([

            'surname' => 'required',

            'name' => 'required',

            'patronymic' => 'required',

            'login' => 'required',

            'password' => 'min:8',

        ]);

        if (User::where('login', $validated['login'])->exists() && $validated['login'] !== $curator->user->login) {

            throw ValidationException::withMessages(['Логин уже используется.']);

        }

        if (isset($validated['password'])) {

            $validated['password'] = Hash::make($validated['password']);

        }

        DB::transaction(function () use ($curator, $validated) {

            $curator->update(Arr::only($validated, ['surname', 'name', 'patronymic']));

            $curator->user()->update(Arr::only($validated, ['login', 'password']));

        });

        return Inertia::location(route('curators.index'));

    }

    public function destroy(Curator $curator)

    {

        if ($curator->groups()->exists()) {

            return Redirect::back()->withErrors(['Удалите связь куратора и группы.']);

        }

        $curator->delete();

        return Inertia::location(route('curators.index'));

    }

}

Файл «GroupController.php».

<?php

namespace App\Http\Controllers;

use App\Exports\GroupAbsencesExport;

use App\Models\Curator;

use App\Models\Group;

use App\Models\GroupCouncil;

use App\Models\GroupPlan;

use App\Models\Specialty;

use App\Models\StudentEducationLevel;

use App\Services\CharacteristicService;

use App\Services\StudentService;

use Illuminate\Database\Eloquent\Builder;

use Illuminate\Database\Eloquent\Collection;

use Illuminate\Database\Eloquent\Relations\BelongsToMany;

use Illuminate\Http\Request;

use Illuminate\Support\Carbon;

use Illuminate\Support\Facades\DB;

use Illuminate\Validation\Rule;

use Illuminate\Validation\ValidationException;

use Inertia\Inertia;

use Maatwebsite\Excel\Facades\Excel;

use PhpOffice\PhpWord\TemplateProcessor;

class GroupController extends Controller

{

    public function \_\_construct()

    {

        $this->authorizeResource(Group::class, 'group');

    }

    public function index()

    {

        $query = Group::with('curator');

        if (!request()->user()->is\_admin) {

            $query->where('curator\_id', request()->user()->curator->id);

        }

        $groupPagination = $query->paginate();

        return Inertia::render('Group/Index', [

            'groupPagination' => $groupPagination,

            'can' => [

                'create\_groups' => request()

                    ->user()

                    ->can('create', Group::class),

            ],

        ]);

    }

    public function create()

    {

        $curators = Curator::all();

        $specialties = Specialty::all();

        return Inertia::render('Group/Create', compact('curators', 'specialties'));

    }

    public function store(Request $request)

    {

        $validated = $request->validate([

            'number' => 'required|numeric|min:10|max:99',

            'start\_education' => 'required|date',

            'end\_education' => 'required|date',

            'specialty\_id' => 'required|numeric',

            'curator\_id' => 'required|numeric',

        ]);

        Group::create($validated);

        return Inertia::location(route('groups.index'));

    }

    public function edit(Group $group)

    {

        $curators = Curator::all();

        $specialties = Specialty::all();

        return Inertia::render('Group/Update', compact('group', 'curators', 'specialties'));

    }

    public function update(Request $request, Group $group)

    {

        $validated = $request->validate([

            'number' => 'required|numeric|min:10|max:99',

            'start\_education' => 'required|date',

            'end\_education' => 'required|date',

            'specialty\_id' => 'required|numeric',

            'curator\_id' => 'required|numeric',

        ]);

        $group->update($validated);

        return Inertia::location(route('groups.index'));

    }

    public function destroy(Group $group)

    {

        if ($group->students()->exists()) {

            throw ValidationException::withMessages(['В данной группе существуют студенты.']);

        }

        $group->delete();

        return Inertia::location(route('groups.index'));

    }

    public function absences(Group $group, $year, $month)

    {

        $year = (int) $year;

        $month = (int) $month;

        $students = $group

            ->students()

            ->orderBy('surname')

            ->orderBy('name')

            ->orderBy('patronymic')

            ->get()

            ->load([

                'absences' => function ($query) use ($group, $month, $year) {

                    $query->where('group\_id', $group->id);

                    $query->whereMonth('date', $month);

                    $query->whereYear('date', $year);

                },

            ]);

        $daysInMonth = cal\_days\_in\_month(CAL\_GREGORIAN, $month, $year);

        return Inertia::render('Absence/Index', compact('group', 'students', 'daysInMonth', 'year', 'month'));

    }

    public function printAbsences(Group $group, $year, $month)

    {

        return Excel::download(new GroupAbsencesExport($group, (int) $year, (int) $month), "$year.$month.xlsx");

    }

    public function marks(Group $group, int $year, int $month)

    {

        $students = $group

            ->students()

            ->orderBy('surname')

            ->orderBy('name')

            ->orderBy('patronymic')

            ->get()

            ->load([

                'marks' => function ($query) use ($group, $month, $year) {

                    $query->where('group\_id', $group->id);

                    $query->whereMonth('date', $month);

                    $query->whereYear('date', $year);

                },

            ]);

        $daysInMonth = 40;

        return Inertia::render('Group/Marks', compact('group', 'students', 'daysInMonth', 'year', 'month'));

    }

    public function students(Group $group)

    {

        $students = $group

            ->students()

            ->orderBy('surname')

            ->orderBy('name')

            ->orderBy('patronymic')

            ->get();

        $characteristicMapQuery = CharacteristicService::buildCharacteristicMap()->leftJoin(

            'group\_student',

            'characteristic\_student.student\_id',

            '=',

            'group\_student.student\_id',

        );

        $characteristicMapQuery

            ->select(

                DB::raw(

                    'characteristics.id characteristic\_id, characteristics.title, (select count(\*) from characteristic\_student join students on characteristic\_student.student\_id=students.id join group\_student on group\_student.student\_id=students.id where characteristic\_id=characteristics.id and group\_student.group\_id=?) connection\_strength',

                ),

            )

            ->setBindings([$group->id]);

        return Inertia::render('Student/Index', [

            'group' => $group,

            'students' => $students,

            'studentCharacteristicMap' => array\_merge(

                [

                    CharacteristicService::buildVirtualCharacteristicMapItem($students, 'total'),

                    CharacteristicService::buildVirtualCharacteristicMapItem($students, 'budget'),

                    CharacteristicService::buildVirtualCharacteristicMapItem($students, 'not-budget'),

                    CharacteristicService::buildVirtualCharacteristicMapItem($students, 'boys'),

                    CharacteristicService::buildVirtualCharacteristicMapItem($students, 'girls'),

                    CharacteristicService::buildVirtualCharacteristicMapItem($students, 'minors'),

                    CharacteristicService::buildVirtualCharacteristicMapItem($students, 'adult'),

                    CharacteristicService::buildVirtualCharacteristicMapItem($students, 'out-of-town'),

                    CharacteristicService::buildVirtualCharacteristicMapItem($students, 'dorm'),

                    CharacteristicService::buildVirtualCharacteristicMapItem($students, 'foreign'),

                ],

                $characteristicMapQuery

                    ->clone()

                    ->where('characteristics.group', 'учащийся')

                    ->get()

                    ->toArray(),

            ),

            'familyCharacteristicMap' => $characteristicMapQuery

                ->clone()

                ->where('characteristics.group', 'семья')

                ->get(),

        ]);

    }

    public function printSocialPedagogicalCharacteristics(Group $group)

    {

        $phpWord = new TemplateProcessor(base\_path('resources/templates/social-pedagogical-characteristics.docx'));

        $phpWord->setValues([

            'specialty' => $group->specialty->title,

            'group' => $group->title,

            'initials' => $group->curator->initials,

            'fillDate' => now()->format('d.m.y') . 'г.',

        ]);

        $adultStudentsCount = $group

            ->students()

            ->whereDate(

                'birthday',

                '<=',

                now()

                    ->subYears(18)

                    ->format('Y-m-d'),

            )

            ->count();

        $studentsInClubsCount = $group

            ->students()

            ->with('characteristics')

            ->whereHas('characteristics', function (Builder $query) {

                $query->where('characteristic\_id', 14);

            })

            ->count();

        $data = [

            $group->students()->count(),

            $group

                ->students()

                ->where('apprenticeship', '=', 'Бюджет')

                ->count(),

            $group

                ->students()

                ->where('apprenticeship', '=', 'Внебюджет')

                ->count(),

            $group

                ->students()

                ->where('sex', '=', 'Мужской')

                ->count(),

            $group

                ->students()

                ->where('sex', '=', 'Женский')

                ->count(),

            $group->students()->count() - $adultStudentsCount,

            $adultStudentsCount,

            $group

                ->students()

                ->where('district', '<>', 'Гомельский')

                ->count(),

            // @TODO

            $group

                ->students()

                ->where('residence\_at\_the\_time\_of\_study', 'REGEXP', 'гомель.\*речицкая.\*д[ом]?.\*4')

                ->count(),

            $group

                ->students()

                ->where('citizenship', '<>', 'Республика Беларусь')

                ->count(),

            $group

                ->students()

                ->with('characteristics')

                ->whereHas('characteristics', function (Builder $query) {

                    $query->where('characteristic\_id', 15);

                })

                ->count(),

            $group

                ->students()

                ->with('characteristics')

                ->whereHas('characteristics', function (Builder $query) {

                    $query->where('characteristic\_id', 18);

                })

                ->count(),

            $group

                ->students()

                ->with('characteristics')

                ->whereHas('characteristics', function (Builder $query) {

                    $query->where('characteristic\_id', 22);

                })

                ->count(),

            $group

                ->students()

                ->with('characteristics')

                ->whereHas('characteristics', function (Builder $query) {

                    $query->where('characteristic\_id', 16);

                })

                ->count(),

            $group

                ->students()

                ->with('characteristics')

                ->whereHas('characteristics', function (Builder $query) {

                    $query->where('characteristic\_id', 19);

                })

                ->count(),

            $group

                ->students()

                ->with('characteristics')

                ->whereHas('characteristics', function (Builder $query) {

                    $query->where('characteristic\_id', 23);

                })

                ->count(),

            $group

                ->students()

                ->with('characteristics')

                ->whereHas('characteristics', function (Builder $query) {

                    $query->where('characteristic\_id', 17);

                })

                ->count(),

            $group

                ->students()

                ->with('characteristics')

                ->whereHas('characteristics', function (Builder $query) {

                    $query->where('characteristic\_id', 20);

                })

                ->count(),

            $group

                ->students()

                ->with('characteristics')

                ->whereHas('characteristics', function (Builder $query) {

                    $query->where('characteristic\_id', 24);

                })

                ->count(),

            $group

                ->students()

                ->with('characteristics')

                ->whereHas('characteristics', function (Builder $query) {

                    $query->where('characteristic\_id', 1);

                })

                ->count(),

            $group

                ->students()

                ->with('characteristics')

                ->whereHas('characteristics', function (Builder $query) {

                    $query->where('characteristic\_id', 8);

                })

                ->count(),

            $group

                ->students()

                ->with('characteristics')

                ->whereHas('characteristics', function (Builder $query) {

                    $query->where('characteristic\_id', 2);

                })

                ->count(),

            $group

                ->students()

                ->with('characteristics')

                ->whereHas('characteristics', function (Builder $query) {

                    $query->where('characteristic\_id', 9);

                })

                ->count(),

            $group

                ->students()

                ->with('characteristics')

                ->whereHas('characteristics', function (Builder $query) {

                    $query->where('characteristic\_id', 3);

                })

                ->count(),

            $group

                ->students()

                ->with('characteristics')

                ->whereHas('characteristics', function (Builder $query) {

                    $query->where('characteristic\_id', 4);

                })

                ->count(),

            $group

                ->students()

                ->with('characteristics')

                ->whereHas('characteristics', function (Builder $query) {

                    $query->where('characteristic\_id', 10);

                })

                ->count(),

            $group

                ->students()

                ->with('characteristics')

                ->whereHas('characteristics', function (Builder $query) {

                    $query->where('characteristic\_id', 11);

                })

                ->count(),

            $group

                ->students()

                ->with('characteristics')

                ->whereHas('characteristics', function (Builder $query) {

                    $query->where('characteristic\_id', 5);

                })

                ->count(),

            $group

                ->students()

                ->with('characteristics')

                ->whereHas('characteristics', function (Builder $query) {

                    $query->where('characteristic\_id', 12);

                })

                ->count(),

            $group

                ->students()

                ->with('characteristics')

                ->whereHas('characteristics', function (Builder $query) {

                    $query->where('characteristic\_id', 13);

                })

                ->count(),

            $group

                ->students()

                ->with('characteristics')

                ->whereHas('characteristics', function (Builder $query) {

                    $query->where('characteristic\_id', 6);

                })

                ->count(),

            $group

                ->students()

                ->with('characteristics')

                ->whereHas('characteristics', function (Builder $query) {

                    $query->where('characteristic\_id', 7);

                })

                ->count(),

            $studentsInClubsCount,

            $group->students()->count() - $studentsInClubsCount,

        ];

        foreach ($data as $i => $column) {

            $phpWord->setValue($i, $column <= 0 ? '-' : $column);

        }

        $tempFile = tempnam(sys\_get\_temp\_dir(), 'phpword');

        $phpWord->saveAs($tempFile);

        return response()

            ->download(

                $tempFile,

                'Социально-педагогическая характеристика группы ' . htmlspecialchars($group->title) . '.docx',

            )

            ->deleteFileAfterSend();

    }

    public function printSocialPassport(Group $group)

    {

        /\*\* @var Collection $students \*/

        $students = $group->students;

        $students->load([

            'relatives',

            'characteristics' => function (BelongsToMany $query) {

                /\*\* Загружаем только необходимые характеристики для документа \*/

                $query->whereIn('characteristics.id', [18, 15, 23, 19, 17, 20, 24, 21, 1, 2, 5, 6, 9, 3, 4, 10, 11]);

            },

        ]);

        $tableRows = $students->map(function ($student, $studentIndex) {

            $father = $student->father;

            $mother = $student->mother;

            $minorRelatives = $student->relatives->filter(function ($relative) {

                if (!$relative->birthday) {

                    return false;

                }

                return Carbon::createFromFormat('Y-m-d', $relative->birthday)->diffInYears(now()) < 18;

            });

            return [

                'number' => $studentIndex + 1,

                'fullName' => $student->fullName,

                'birthday' => Carbon::createFromFormat('Y-m-d', $student->birthday)->format('d.m.Y'),

                'address' => $student->address,

                'home\_phone' => $student->home\_phone ?? '',

                'residence\_at\_the\_time\_of\_study' => $student->residence\_at\_the\_time\_of\_study ?? 'По прописке',

                'phone' => $student->phone,

                'father' => $father ? "$father->fullName, $father->job, $father->position, $father->phone" : '',

                'mother' => $mother ? "$mother->fullName, $mother->job, $mother->position, $mother->phone" : '',

                'minorRelatives' => $minorRelatives

                    ->map(function ($relative) {

                        return "$relative->fullname, $relative->birthday, $relative->educational\_institution";

                    })

                    ->join(', '),

                'familyCharacteristics' => $student->characteristics

                    ->where('group', 'семья')

                    ->where('id', '<>', 21)

                    ->map(function ($characteristic) {

                        return $characteristic->shorthand\_name;

                    })

                    ->join('<w:br/>'),

                'studentCharacteristics' => $student->characteristics

                    ->whereNotIn('id', [3, 4, 10, 11])

                    ->filter(function ($characteristic) {

                        return $characteristic->group === 'учащийся' || $characteristic->id === 21;

                    })

                    ->map(function ($characteristic) {

                        return $characteristic->shorthand\_name;

                    })

                    ->join('<w:br/>'),

                'otherDetails' => $student->other\_details ?? '',

            ];

        });

        $templateProcessor = new TemplateProcessor(base\_path('resources/templates/social-passport.docx'));

        $templateProcessor->setValue('group', $group->title);

        $templateProcessor->setValue('course', $group->course);

        $templateProcessor->setValue(

            'education\_duration',

            Carbon::createFromFormat('Y-m-d', $group->start\_education)->year .

                '/' .

                Carbon::createFromFormat('Y-m-d', $group->end\_education)->year .

                ' ',

        );

        $templateProcessor->setValue('specialty', $group->specialty->title);

        $templateProcessor->setValue('curator', $group->curator->fullName);

        $templateProcessor->setValue('largeFamilies', StudentService::withLargeFamily($students)->count());

        $templateProcessor->setValue('orphans', StudentService::getOrphans($students)->count());

        $templateProcessor->setValue('boys', StudentService::getBoys($students)->count());

        $templateProcessor->setValue('girls', StudentService::getGirls($students)->count());

        $templateProcessor->setValue('cities', StudentService::getUrban($students)->count());

        $templateProcessor->setValue('outOfTown', StudentService::getOutOfTown($students)->count());

        $templateProcessor->setValue('singleParentFamilies', StudentService::withSingleParent($students)->count());

        $templateProcessor->cloneRow('fullName', $tableRows->count());

        foreach ($tableRows as $tableRow) {

            foreach ($tableRow as $tableCol => $tableValue) {

                $templateProcessor->setValue("$tableCol#{$tableRow['number']}", $tableValue);

            }

        }

        $tempFile = tempnam(sys\_get\_temp\_dir(), 'phpword');

        $templateProcessor->saveAs($tempFile);

        return response()

            ->download($tempFile, 'Социальный паспорт ' . htmlspecialchars($group->title) . '.docx')

            ->deleteFileAfterSend();

    }

    public function leadership(Group $group)

    {

        $activity = $group->leadership;

        $councils = $group->council->map(function ($councilRow) {

            return $councilRow->pivot;

        });

        $studentNamesList = $group

            ->students()

            ->orderBy('surname')

            ->orderBy('name')

            ->orderBy('patronymic')

            ->get(['students.id', 'surname', 'name', 'patronymic'])

            ->map(function ($student) {

                return ['value' => $student->id, 'label' => $student->full\_name];

            });

        return Inertia::render('Group/Activity', compact('studentNamesList', 'group', 'activity', 'councils'));

    }

    public function syncCouncil(Request $request, Group $group)

    {

        $validated = $request->validate([

            '\*.sector' => [

                'required',

                Rule::in(

                    'Учебный',

                    'Информационно-идеологический',

                    'Физкультурно-спортивный',

                    'Трудовой',

                    'Культурно-массовый',

                    'Охраны правопорядка',

                    'Редакционный',

                ),

            ],

            '\*.student\_id' => 'required|numeric|exists:students,id',

        ]);

        if (

            collect($validated)

                ->unique()

                ->count() !== count($validated)

        ) {

            throw ValidationException::withMessages(['Удалите повторяющихся студентов из секции.']);

        }

        DB::transaction(function () use ($validated, $group) {

            GroupCouncil::where('group\_id', $group->id)->delete();

            foreach ($validated as $council) {

                GroupCouncil::create(array\_merge($council, ['group\_id' => $group->id]));

            }

        });

        return back();

    }

    public function plan(Group $group, $year, $month)

    {

        $plan = GroupPlan::whereYear('date', $year)

            ->whereMonth('date', $month)

            ->first();

        if (!$plan) {

            $plan = GroupPlan::create([

                'date' => Carbon::create($year, $month)->format('Y-m-d'),

                'group\_id' => $group->id,

            ]);

        }

        $planItems = $plan->items;

        return Inertia::render('Group/Plan', compact('group', 'plan', 'planItems'));

    }

    public function savePlanItems(Request $request, Group $group, $year, $month)

    {

        $validated = $request->validate([

            '\*.date' => 'nullable|date',

            '\*.working\_forms' => 'required',

            '\*.done' => 'required|boolean',

            '\*.category' => [

                'required',

                Rule::in(['Диагностическая', 'Методическая', 'Организационно-воспитательная']),

            ],

        ]);

        $plan = GroupPlan::whereYear('date', $year)

            ->whereMonth('date', $month)

            ->first();

        if (!$plan) {

            $plan = GroupPlan::create([

                'date' => Carbon::create($year, $month)->format('Y-m-d'),

                'group\_id' => $group->id,

            ]);

        }

        DB::transaction(function () use ($validated, $plan) {

            $plan->items()->forceDelete();

            $plan->items()->createMany($validated);

        });

        return back();

    }

    public function studentEducationLevel(Group $group)

    {

        foreach ($group->students as $student) {

            StudentEducationLevel::firstOrCreate(

                ['student\_id' => $student->id, 'group\_id' => $group->id],

                [

                    'student\_id' => $student->id,

                    'group\_id' => $group->id,

                    'meta1' => '1',

                    'meta2' => '1',

                    'meta3' => '1',

                    'meta4' => '1',

                    'meta5' => '1',

                    'meta6' => '1',

                    'meta7' => '1',

                    'meta8' => '1',

                    'meta9' => '1',

                    'meta10' => '1',

                    'meta11' => '1',

                    'meta12' => '1',

                    'meta13' => '1',

                    'meta14' => '1',

                    'meta15' => '1',

                ],

            );

        }

        $studentEducationLevelList = StudentEducationLevel::where('group\_id', $group->id)

            ->with('student')

            ->get();

        $columns = StudentEducationLevel::$columns;

        return Inertia::render('Group/StudentEducationLevel', compact('studentEducationLevelList', 'columns'));

    }

    public function updateStudentEducationLevel(

        Request $request,

        Group $group,

        StudentEducationLevel $studentEducationLevel,

    ) {

        $metaRule = ['required', Rule::in('1', '2', '3', '4')];

        $request = $request->validate([

            'meta1' => $metaRule,

            'meta2' => $metaRule,

            'meta3' => $metaRule,

            'meta4' => $metaRule,

            'meta5' => $metaRule,

            'meta6' => $metaRule,

            'meta7' => $metaRule,

            'meta8' => $metaRule,

            'meta9' => $metaRule,

            'meta10' => $metaRule,

            'meta11' => $metaRule,

            'meta12' => $metaRule,

            'meta13' => $metaRule,

            'meta14' => $metaRule,

            'meta15' => $metaRule,

        ]);

        $studentEducationLevel->update($request);

        return back();

    }

}

# Приложение Б

(обязательное)

**Руководство пользователя**

1. **Введение**

Позволяет хранить, обрабатывать и анализировать данные журнала куратора и предоставляет возможность постоянного доступа к ним.

1. **Назначение и условия применения.**

Разработанное приложение обеспечивает эффективное управление информацией о студентах и их учебной деятельности.

Условия, необходимые для работы приложения:

* веб-браузер;
* установленный «OpenServer»;
* клавиатура, мышь, монитор и принтер (рекомендуется).

1. **Подготовка к работе.**

Запуск приложения производится на ноутбуке или компьютере, который обладает минимальными требованиями для работы веб-браузера. Для работы с данным приложением пользователь должен обладать средними навыками работы с ноутбуком или компьютером.

1. **Описание операций.**

В программе предусмотрены такие функции, как:

* добавление;
* удаление;
* изменение;
* сортировка;
* поиск;
* вывод в «word», «excel»;
* аналитика.

При первом запуске приложения открывается страница с формой входа, которая позволяет войти в аккаунт. Страница с формой входа представлена на рисунке Б.1.

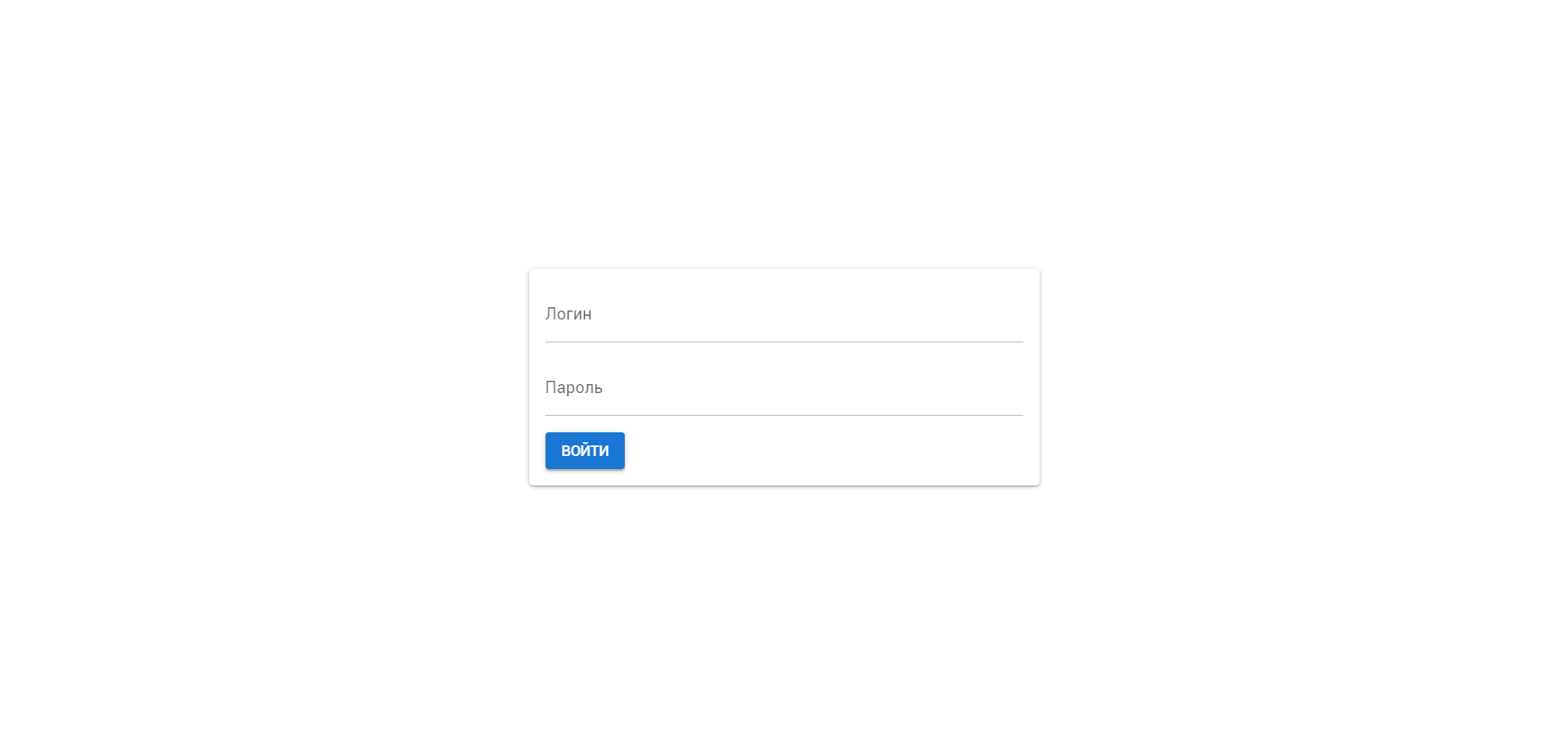


Рисунок Б.1 — Страница с формой входа

При входе в аккаунт пользователь вводит свои данные и, в случае успеха, переходит на главную форму. Главная форма содержит общую информацию о деятельности колледжа: выходные формы, характеристики учащихся и их семей в виде диаграмм и боковую навигацию, позволяющую переходить на другие страницы. Главная форма представлена на рисунке Б.2.

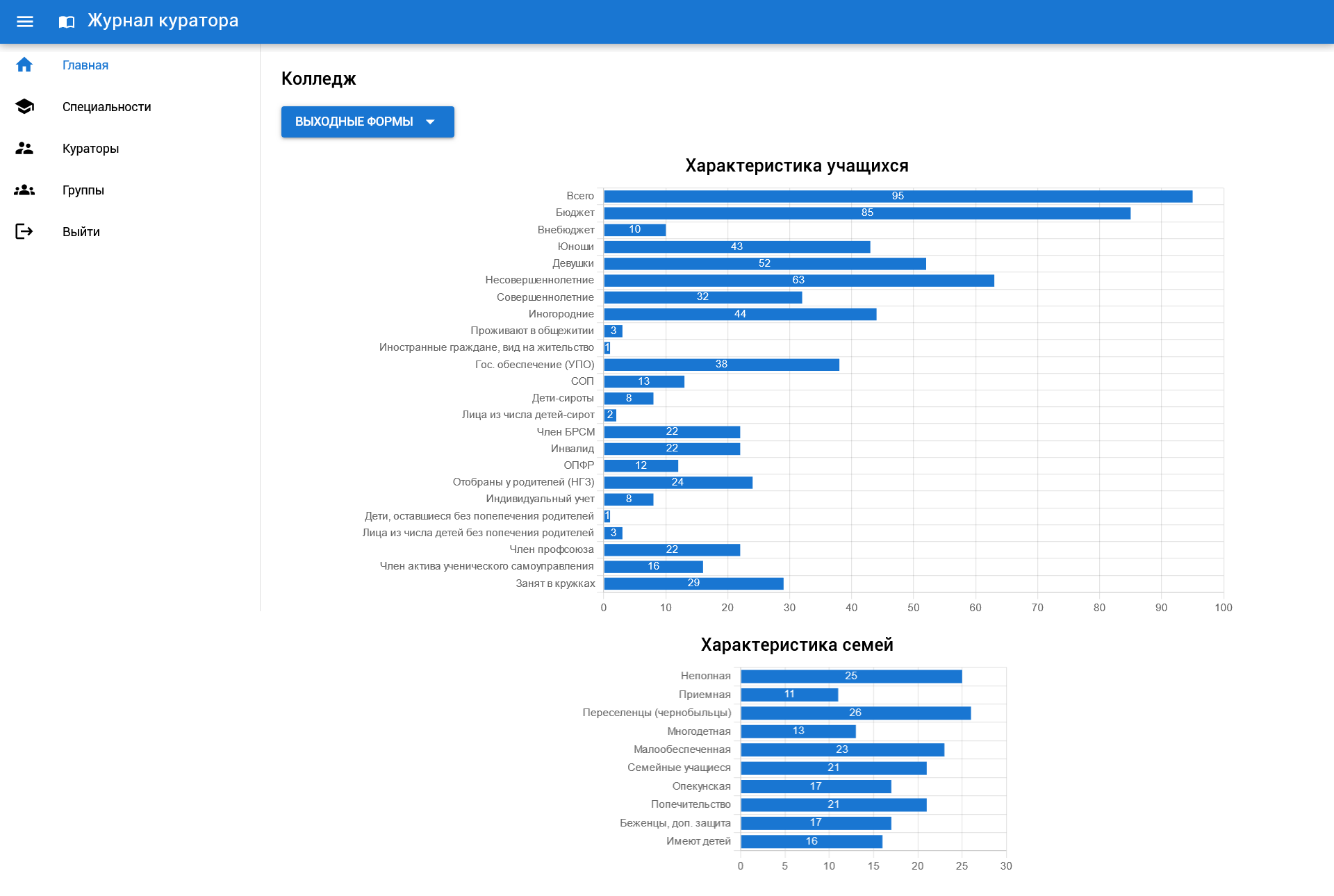


Рисунок Б.2 — Главная форма

Для работы с сущностями существуют отдельные формы: таблица с выводом данных, формы добавления и редактирования.

Таблица с выводом данных представлена на рисунке Б.3.

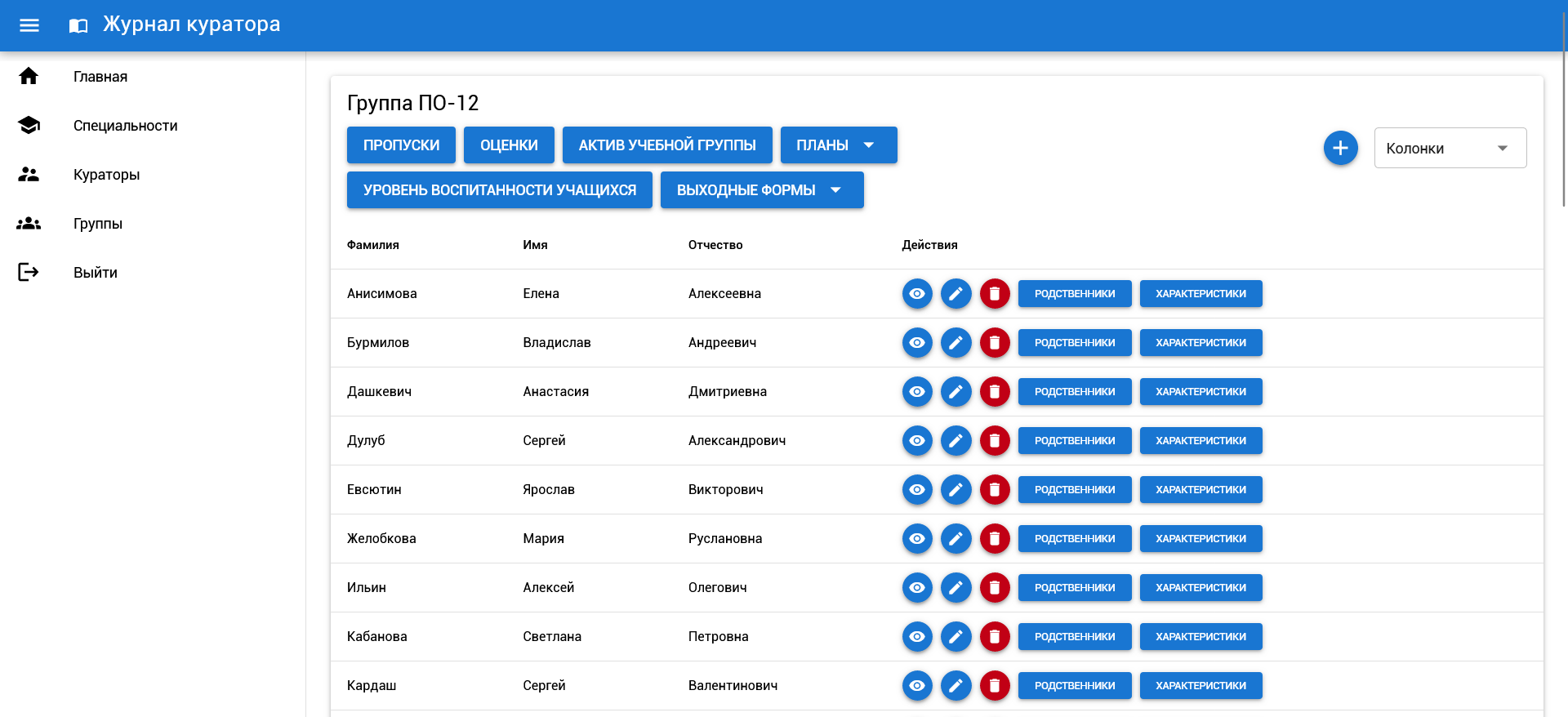


Рисунок Б.3 — Таблица с выводом данных

Сортировка данных активизируется по нажатию на один из заголовков таблицы. Сортировка по полю «Имя» представлена на рисунке Б.4.

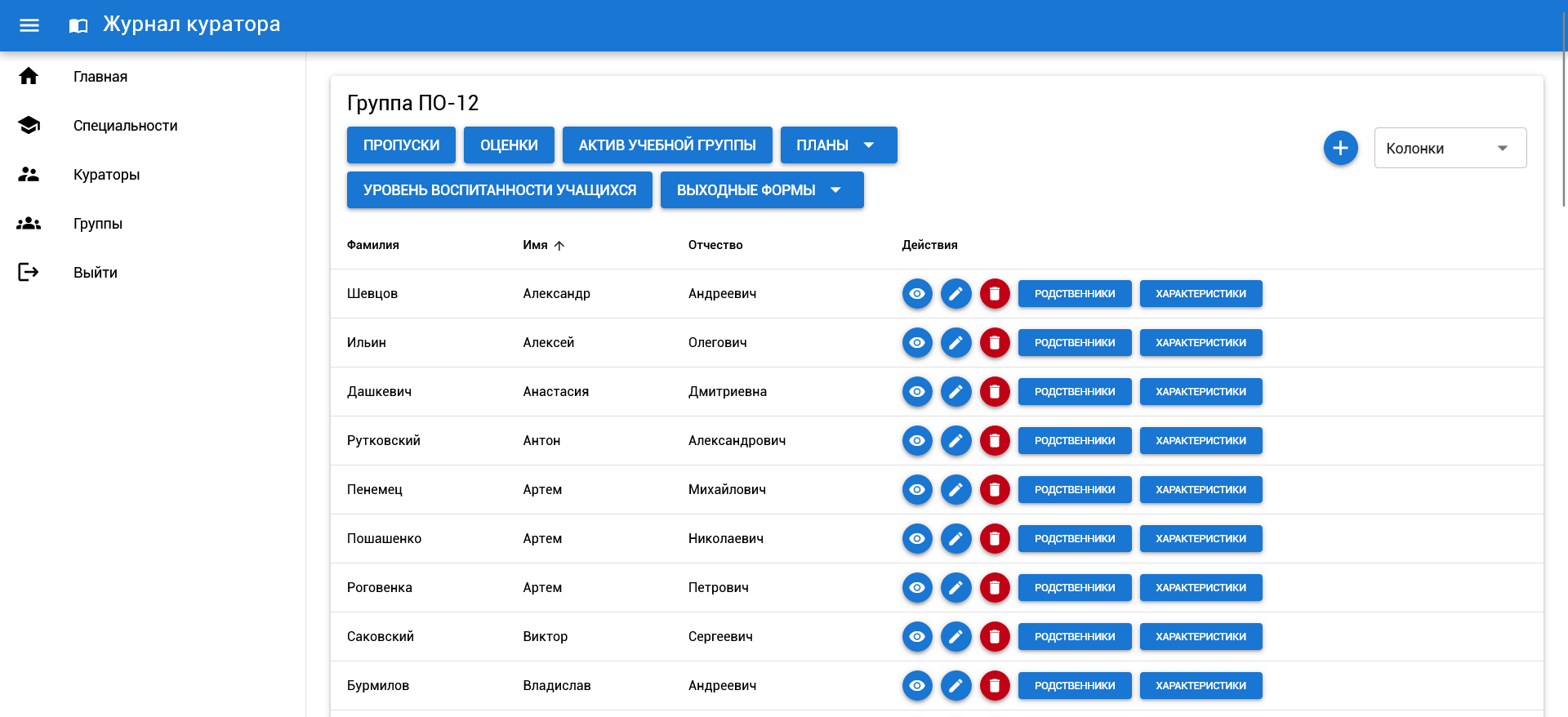


Рисунок Б.4 — Сортировка по полю «Имя»

Поиск данных активируется при нажатии на клавиши «Ctrl + F», после чего появляется окно с текстовым полем. Поиск данных представлен на рисунке Б.5.

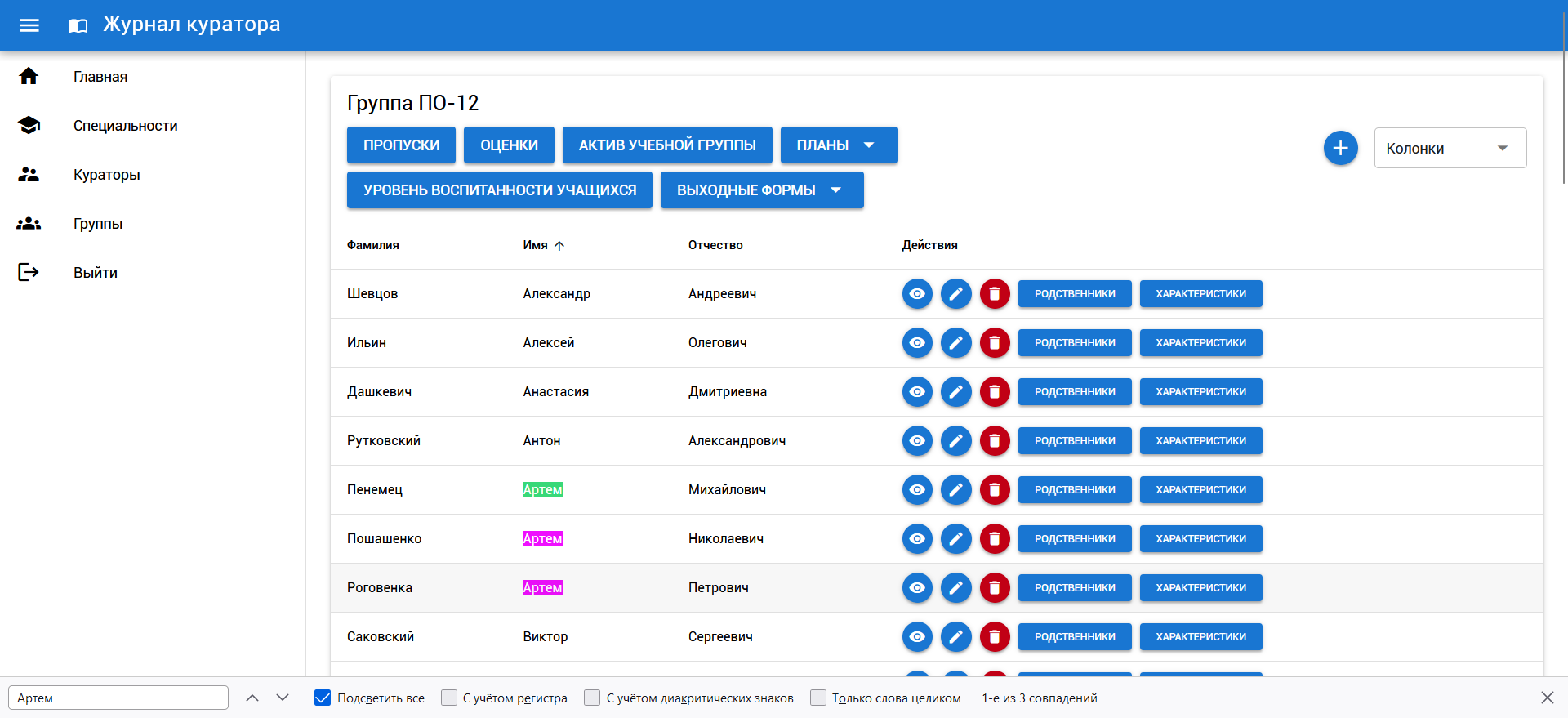


Рисунок Б.5 — Поиск данных

Форма добавления и редактирования представлена на рисунке Б.6.

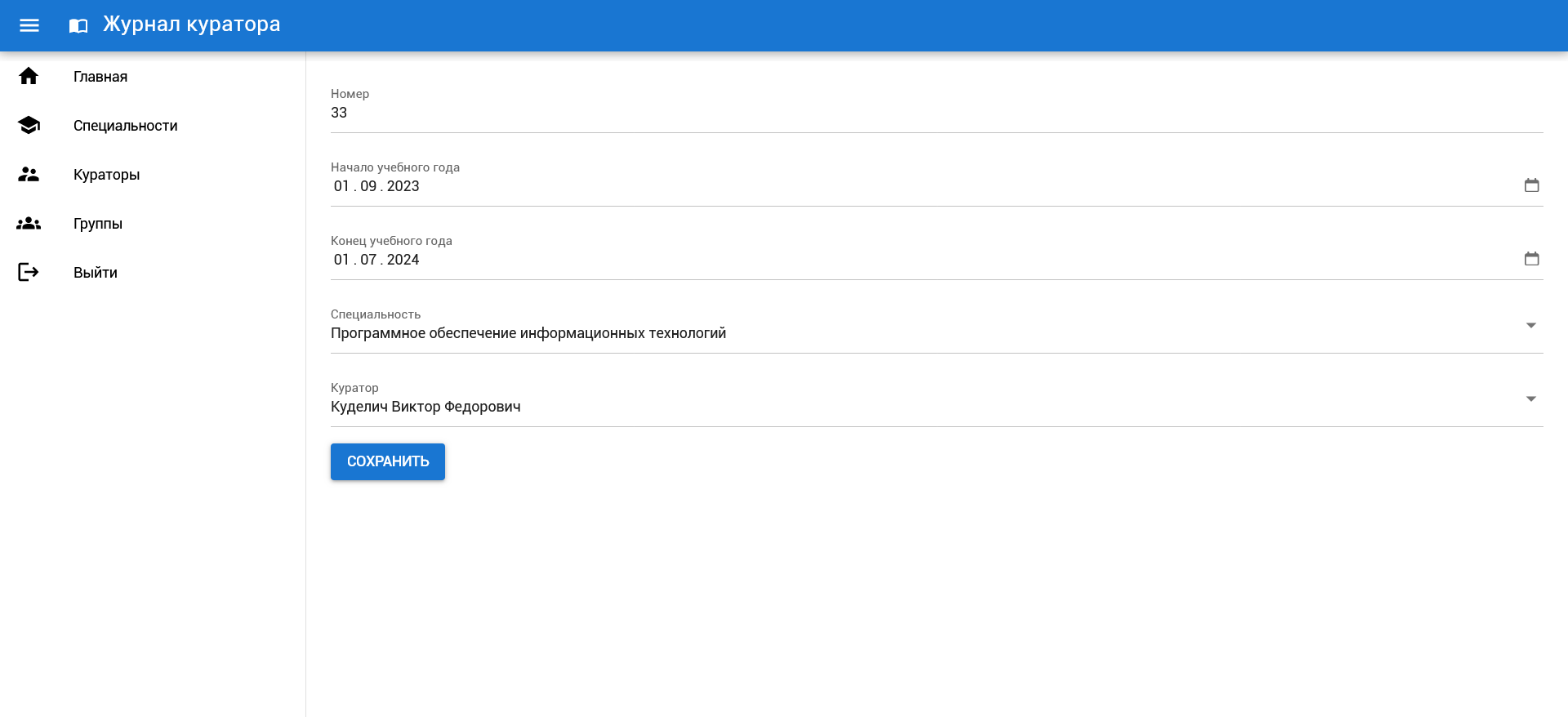


Рисунок Б.6 — Форма добавления и редактирования

Для аналитики данных существуют выходные формы и диаграммы. Пример выходной формы «Социально-педагогическая характеристика» представлена на рисунке Б.7.

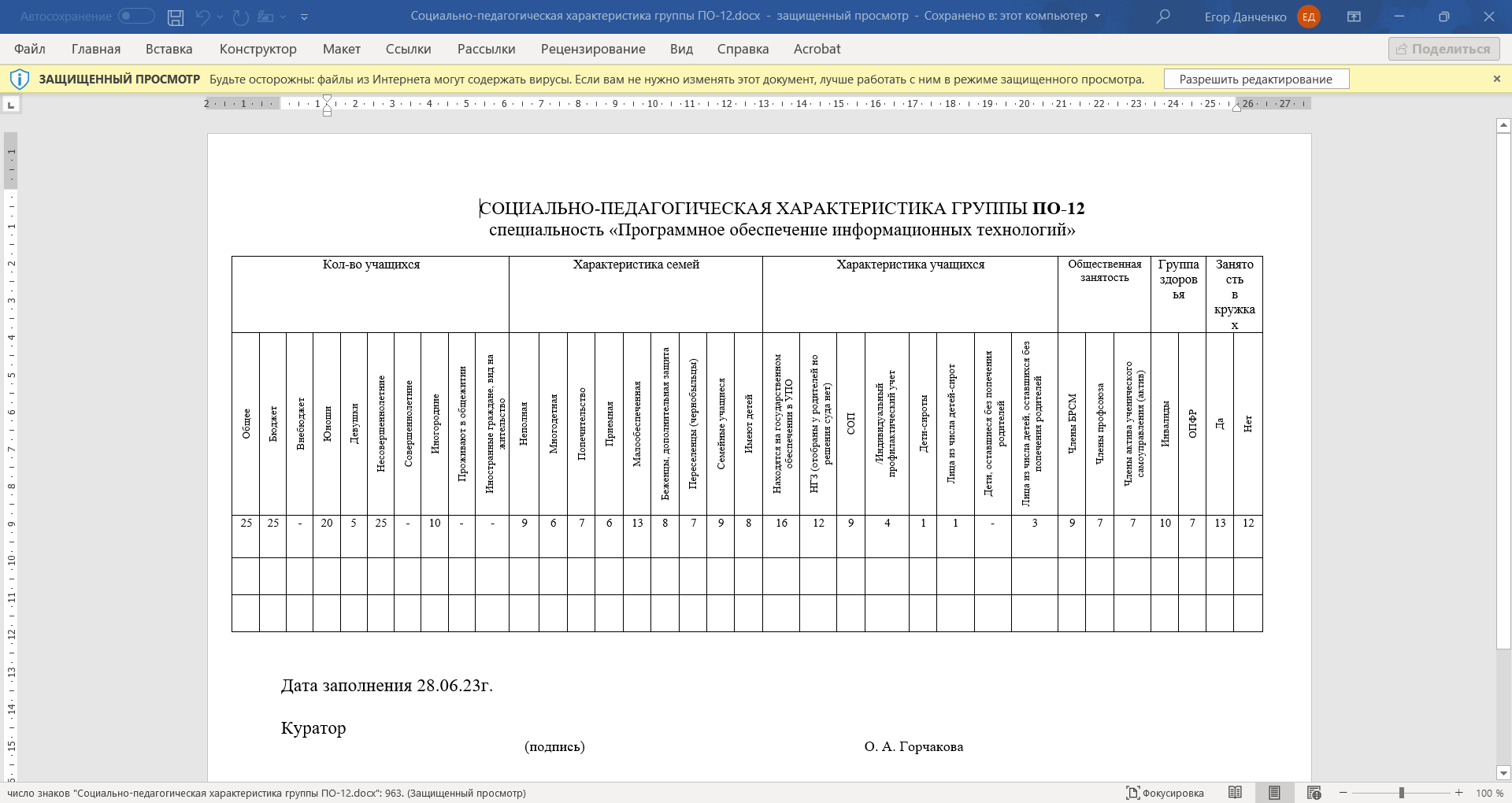


Рисунок Б.7 — Выходная форма «Социально-педагогическая характеристика»

Диаграммы «Характеристика учащихся» и «Характеристика семей» представлены на рисунке Б.8.

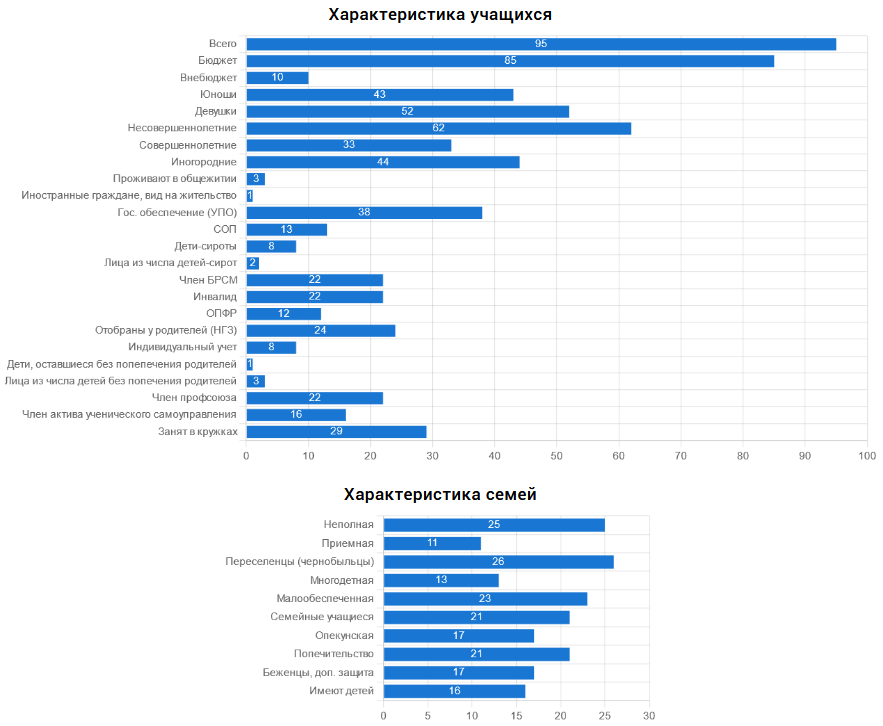


Рисунок Б.8 — Диаграммы «Характеристика учащихся»  
и «Характеристика семей»

1. **Аварийные ситуации.**

При случайном выключении света или перезагрузке ПК, и других аварийных ситуаций, приложение закроется, но все изменения, которые были выполнены во время работы программы, сохранятся в базе данных.

1. **Рекомендации по освоению.**

Интерфейс приложения прост в управлении и использовании, но требует специальных знаний с введением бумажного варианта журнала куратора.

# Приложение В

(обязательное)

**Руководство программиста**

1. **Назначение и условия применения программы.**

Функции программы:

* учет (создание, редактирование, удаление, анализ) специальностей, кураторов, групп и учащихся;
* валидация данных на стороне клиента и сервера;
* защита от распространенных атак, таких как «XSS», «CSRF» и «SQLинъекции»;
* управление доступом на основе ролей;
* адаптация под все разрешения экрана;
* возможность экспорта данных в различные форматы для сдачи отчетности.

1. **Характеристики программы.**

Электронная версия журнала куратора решает проблему с необходимостью цифровизации, которая способствует более быстрой и точной обработке данных, снижает вероятность ошибок, упрощает процесс анализа информации и составления отчетности.

1. **Обращение к программе.**

Для входа в приложение необходимо перейти по специальному адресу в браузере. Адрес выбирает веб-администратор, который выполнит установку приложения на устройство. Адрес по умолчанию: «http://log.cu58644.tw1.ru».

1. **Входные и выходные данные.**

Входные данные — это данные, которые куратор вводит вручную.

Выходные данные — это данные, которые получаются в результате работы программы.

Входными данными считаются информация о учащихся и группа. Входные данные вносятся куратором вручную.

Выходными данными являются: социально-педагогическая характеристика, социальный паспорт, пропуски и успеваемость учащихся. Документы формируются для программ «Excel» и «Word».

1. **Сообщения.**

Приложение имеет множество сообщений об ошибочных операциях. При некорректной работе приложения рекомендуется ее перезапустить и связаться с разработчиком продукта.

В программе могут встретиться следующие сообщения:

* сообщения при некорректном введении данных;
* сообщения при попытке выполнить неверную операцию.