**ОГЛАВЛЕНИЕ**

[ВВЕДЕНИЕ 4](#_Toc120528941)

[I. СИСТЕМНЫЙ АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 6](#_Toc120528942)

[1.1 Состав, параметры объектов (сущностей) и ограничения предметной области 6](#_Toc120528943)

[1.2 Состав групп пользователей системы 7](#_Toc120528944)

[1.3 Обязанности и права доступа групп пользователей к информации в базе данных. 8](#_Toc120528945)

[1.4 Формирование отчётности 8](#_Toc120528946)

[II. ИНФОЛОГИЧЕСКОЕ ПРОЕКТИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ 9](#_Toc120528947)

[III. КОНЦЕПТУАЛЬНОЕ ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ 12](#_Toc120528948)

[3.1 Преобразование сущностей в отношения (связи) 12](#_Toc120528949)

[3.2 Определение свойств атрибутов отношений (столбцов таблиц) 15](#_Toc120528950)

[3.3 Определение внешних ключей 17](#_Toc120528951)

[3.4 Определение внешних ключей 17](#_Toc120528952)

[3.5 Концептуальная схема БД 18](#_Toc120528953)

[IV. РЕАЛИЗАЦИЯ БАЗЫ ДАННЫХ 18](#_Toc120528954)

[V. ЗАПРОСЫ К БАЗЕ ДАННЫХ 30](#_Toc120528955)

[VI. РУКОВОДСТВО ПРОГРАММИСТА 31](#_Toc120528956)

[VII. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 34](#_Toc120528957)

[ЗАКЛЮЧЕНИЕ 42](#_Toc120528958)

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 43](#_Toc120528959)

# ВВЕДЕНИЕ

**Актуальность**

Знание иностранных языков чрезвычайно важно в современное время. Именно поэтому существует множество школ иностранного языка. Разрабатываемая в данной работе система поможет в управлении школой и автоматизации её бизнес-процессов.

**Цели и задачи курсовой работы:**

Целью курсовой работы является приобретение студентами навыков: применения теоретических знаний и практических умений для самостоятельного анализа и формулирования задачи повышения эффективности деятельности организации или предприятия; разработки информационного и программного обеспечения автоматизации деятельности специалистов организации или предприятия; грамотного и последовательного изложения материала проведенных исследований.

Задачи курсовой работы состоят в овладении практическими навыками выполнения следующих этапов проектирования и реализации проекта:

1. Системный анализ и словесное описание информационных объектов предметной области.
2. Проектирование инфологической модели предметной области - формализованное описание объектов предметной области в терминах некоторой семантической модели, например, в терминах ЕR-модели.
3. Концептуальное проектирование базы данных в рамках реляционной модели данных.
4. Реализация проекта и создание клиентского приложения для работы с базой данных.

**Требования к приложению:**

Приложение базы данных должно:

* обеспечивать авторизацию и разделение прав выделенных пользователей;
* обеспечивать выполнение всех задач пользователей, описанных в системном анализе предметной области;
* позволять вводить новые записи в любую таблицу базы данных, редактировать и удалять записи таблиц с обеспечением целостности данных;
* позволять выполнять поиск и/или фильтрацию данных таблиц;
* позволять выполнять некоторые запросы (не менее 5 запросов);
* иметь дружественный, интуитивно- и профессионально-понятный интерфейс.

**Краткое содержание разделов курсовой:**

*Раздел «1. Системный анализ предметной области»* содержит словесное описание предметной области.

*Раздел «2. Инфологическое проектирование предметной области»* содержит результаты проектирования по методике, изученной на практическом занятии 1: сущности, атрибуты каждой сущности, ограничения на информацию каждой сущности, связи между сущностями.

*Раздел «3. Концептуальное проектирование базы данных»* содержит результаты проектирования по методике, изученной на практическом занятии 2.

*Раздел «4. Реализация базы данных»* содержит обоснование выбора СУБД для реализации базы данных, а также команды SQL для создания базы данных и всех ее объектов – таблиц, индексов, представлений, триггеров и т.д.

*Раздел «5. Запросы к базе данных»* должен содержать словесную формулировку каждого запроса на выборку данных из таблиц, соответствующую команду SQL и результаты ее выполнения.

*Раздел «6. Руководство программиста»* содержит список программных и инструментальных средств, выбранных для реализации приложения, а также описание архитектуры и основных модулей разработанного приложения.

*Раздел «7. Руководство пользователя»* содержит описание работы пользователя с разработанным приложением, включая скриншоты форм.

*Раздел «Заключение»* содержит основные выводы о проделанной работе и достигнутым результатам.

*Раздел «Список литературы»* содержит библиографические записи, оформленные в соответствии с требованиями ГОСТ 7.1-2003.

# I. СИСТЕМНЫЙ АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

## 1.1 Состав, параметры объектов (сущностей) и ограничения предметной области

Школа иностранных языков должна вести учёт учеников, преподавателей, хранить данные о курсах, их уровнях, языке, а также составлять расписание занятий.

Школа может проводить курсы. Каждый курс имеет:

* название;
* описание;
* язык;
* уровень языка;
* категорию.

На один курс может записаться несколько учеников. Курс состоит из серии занятий. Одно занятие проводится только одним учителем в определённый день и время и имеет тип занятия (лекция, урок с носителем и т.п.). Следующая информация хранится о занятиях:

* курс, частью которого является занятие;
* ответственный преподаватель;
* день недели;
* время начала;
* время конца;
* тип занятия.

В школе хранятся следующие данные об учениках:

* фамилия;
* имя;
* отчество;
* почта;
* телефон;
* дата рождения.

Ученик может записаться на несколько курсов одновременно.

Об учителях хранится похожая информация:

* фамилия;
* имя;
* отчество;
* почта;
* телефон;
* дата рождения;
* опыт работы;
* дополнительные сведения.

Преподаватель, ученик и администратор имеют аккаунты с соответствующими правами доступами, описанными в пункте 1.2.

Дополнительно вводятся следующие ограничения:

1. Поля описание у преподавателя и курса могут быть не заполнены.
2. У ученика или преподавателя может не быть фамилии.
3. День рождения заносится в формате YYYY-MM-DD
4. Время заносится в формате HH:MM

## 1.2 Состав групп пользователей системы

С информацией работают следующие группы пользователей:

* администратор;
* преподаватель;
* ученик.

## 1.3 Обязанности и права доступа групп пользователей к информации в базе данных.

Администратор выполняет следующие действия:

1. При поступлении студента заносит его информацию в базу. Может удалять, изменять и просматривать её.
2. При найме преподавателя заносит его информацию в базу. Может удалять, изменять и просматривать её.
3. Создаёт категории, возможные уровни языка, сами язык.
4. По требованию третьей стороны создаёт курсы. Может удалять, изменять и просматривать их.
5. Вносит информацию о рабочих днях и типах занятий.
6. Составляет расписание курсов: создаёт занятие, назначает время, день недели, преподавателя, устанавливает тип занятия. Может удалять, изменять и просматривать расписание.
7. Создаёт аккаунты с различными правами доступа.
8. Зачисляет студентов на курсы.

Преподаватель может:

1. Просмотреть информацию о своём расписании.

Ученик может:

1. Просмотреть информацию о курсах, на которые он зачислен.
2. Просмотреть информацию о своём расписании.

## 1.4 Формирование отчётности

Система может выводить следующие отчёты:

1. Вывод всех учеников курса.
2. Вывод всех курсов ученика
3. Вывод расписания учителя.
4. Вывод расписания ученика
5. Состояние таблиц

# II. ИНФОЛОГИЧЕСКОЕ ПРОЕКТИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

Результаты инфологического проектирования предметной области представлены в таблицах 2.1 – 2.3 и на рисунке 2.1.

**Таблица 2.1 - Перечень атрибутов**

|  |  |  |
| --- | --- | --- |
| Обозначение атрибута | Атрибут | Примечание |
| X1 | ID преподавателя |  |
| X2 | Имя преподавателя |  |
| X3 | Фамилия преподавателя |  |
| X4 | Отчество преподавателя | Может не быть |
| X5 | Почта преподавателя |  |
| X6 | Телефон преподавателя |  |
| X7 | День рождения преподавателя | YYYY-MM-DD |
| X8 | Дополнительная информация о преподавателе | Может не быть |
| X9 | ID ученика |  |
| X10 | Имя ученика |  |
| X11 | Фамилия ученика |  |
| X12 | Отчество ученика | Может не быть |
| X13 | Почта ученика |  |
| X14 | Телефон ученика |  |
| X15 | День рождения ученика |  |
| X16 | ID курса |  |
| X17 | Название курса | Уникальное |
| X18 | Описание курса | Может не быть |
| X19 | Язык курса |  |
| X20 | Уровень курса |  |
| X21 | Категория курса |  |
| X22 | ID уровня курса |  |
| X23 | Название уровня курса | Уникальное |
| X24 | ID категории курса |  |
| X25 | Название категории курса | Уникальное |
| X26 | ID дня недели занятия |  |
| X27 | Название дня недели занятия | Уникальное |
| X28 | ID типа занятия |  |
| X29 | Название типа занятия | Уникальное |
| X30 | ID занятия |  |
| X31 | Курс занятия |  |
| X32 | Преподаватель занятия |  |
| X33 | День недели занятия |  |
| X34 | Время начала | HH:MM |
| X35 | Время конца | HH:MM |
| X36 | Тип занятия |  |
| X37 | ID языка курса |  |
| X38 | Название языка курса |  |
| X39 | Аккаунт преподавателя | Может не быть |
| X40 | Аккаунт студента | Может не быть |
| X41 | Логин аккаунта | Уникальное |
| X42 | Пароль аккаунта |  |
| X43 | Роль аккаунта |  |
| X44 | Опыт преподавателя |  |

**Таблица 2.2 - Определение сущностей**

|  |  |  |  |
| --- | --- | --- | --- |
| Обозначение сущности | Название | Состав атрибутов | Первичный ключ |
| Y1 | Преподаватель | X1-X8, X39, X44 | X1 |
| Y2 | Ученик | X9-X15, X40 | X9 |
| Y3 | Курс | X16-X21 | X16 |
| Y4 | Занятие | X30-X36 | X30 |
| Y5 | Язык курса | X37-X38 | X37 |
| Y6 | Уровень курса | X22-X23 | X22 |
| Y7 | Категория курса | X24-X25 | X24 |
| Y8 | День недели занятия | X26-X27 | X26 |
| Y9 | Тип занятия | X28-X29 | X28 |
| Y10 | Аккаунт | X41-X43 | X40 |

**Таблица 2.3 - Определение связи между сущностями**

|  |  |  |  |
| --- | --- | --- | --- |
| Сущность 1 | Сущность 2 | Тип связи | Обязательность |
| Y3 | Y5 | M:1 | 1-0 |
| Y3 | Y6 | M:1 | 1-0 |
| Y3 | Y7 | M:1 | 1-0 |
| Y3 | Y2 | M:M | 0-0 |
| Y3 | Y4 | 1:M | 0-1 |
| Y4 | Y8 | M:1 | 1-0 |
| Y4 | Y9 | M:1 | 1-0 |
| Y4 | Y1 | M:1 | 1-0 |
| Y1 | Y10 | 1:1 | 0-0 |
| Y2 | Y10 | 1:1 | 0-0 |

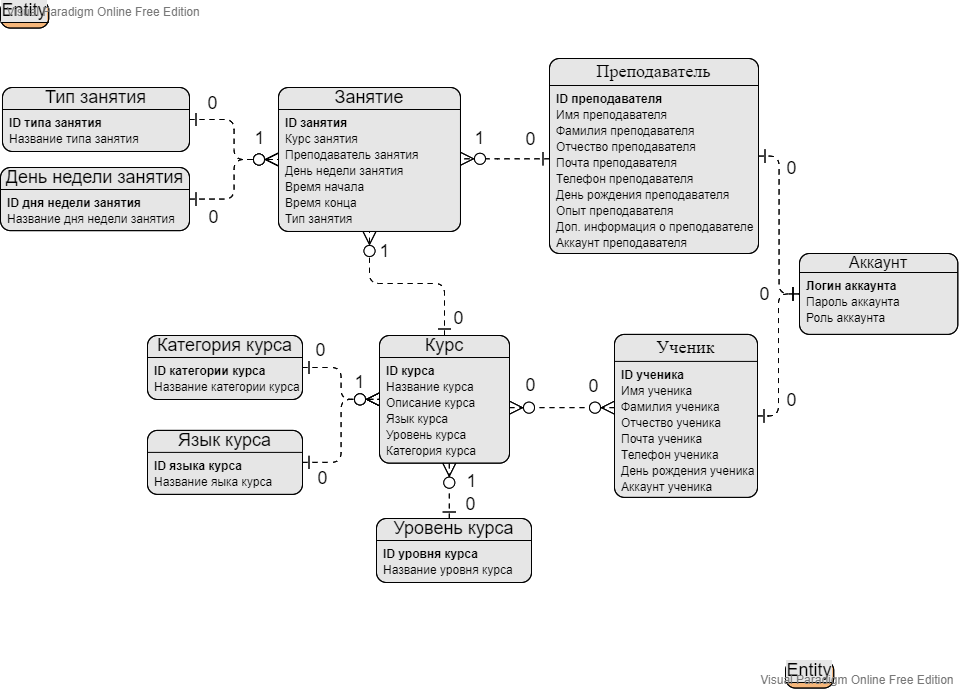
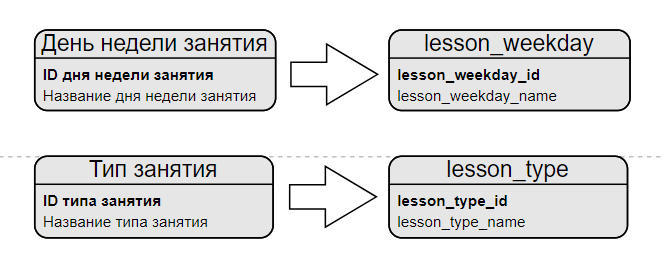


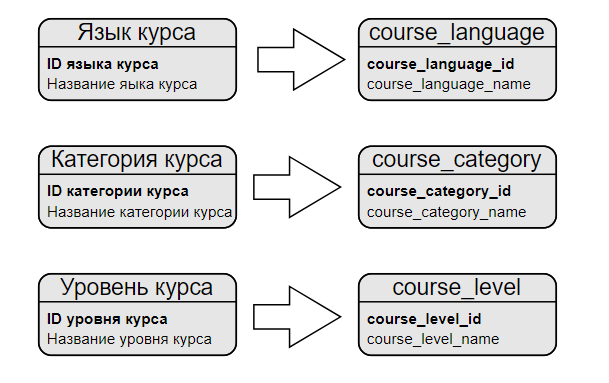
Рисунок 2.1 – Инфологическая схема

# III. КОНЦЕПТУАЛЬНОЕ ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ

## 3.1 Преобразование сущностей в отношения (связи)

Результат преобразования сущностей в отношения представлен на рис. 3.1.





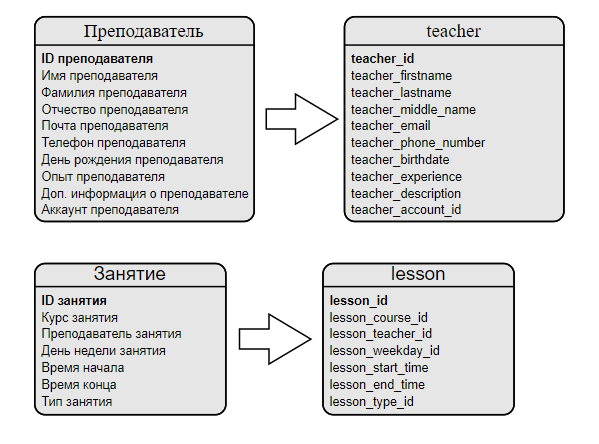
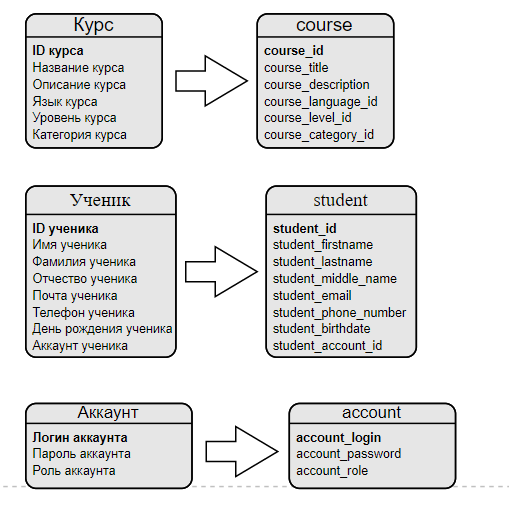


Рис. 3.1 – Преобразование сущностей в отношения

## 3.2 Определение свойств атрибутов отношений (столбцов таблиц)

Первичные ключи всех отношений, кроме account, по умолчанию имеют значение nextval('генератор последовательности'::regclass). Свойства атрибутов отношений представлены в таблицах 3.2.1-3.2.10.

**Таблица 3.2.1 - Свойства атрибутов отношения «course\_category»**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Атрибут | Тип (размер) | Уникальность | Null | Ограничения |
| course\_category\_id | integer | PK | Not null |  |
| course\_category\_name | text |  | Not null | Unique |

**Таблица 3.2.2 - Свойства атрибутов отношения «course\_language»**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Атрибут | Тип (размер) | Уникальность | Null | Ограничения |
| course\_language\_id | integer | PK | Not null |  |
| course\_language\_name | text |  | Not null | Unique |

**Таблица 3.2.3 - Свойства атрибутов отношения «course\_level»**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Атрибут | Тип (размер) | Уникальность | Null | Ограничения |
| course\_level\_id | integer | PK | Not null |  |
| course\_level\_name | text |  | Not null | Unique |

**Таблица 3.2.4 - Свойства атрибутов отношения таблицы «lesson\_type»**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Атрибут | Тип (размер) | Уникальность | Null | Ограничения |
| lesson\_type\_id | integer | PK | Not null |  |
| lesson\_type\_name | text |  | Not null | Unique |

**Таблица 3.2.5 - Свойства атрибутов отношения таблицы «lesson\_weekday»**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Атрибут | Тип (размер) | Уникальность | Null | Ограничения |
| lesson\_weekday\_id | integer | PK | Not null |  |
| lesson\_weekday\_name | text |  | Not null | Unique |

**Таблица 3.2.6 - Свойства атрибутов отношения таблицы «course»**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Атрибут | Тип (размер) | Уникальность | Null | Ограничения |
| course\_id | integer | PK | Not null |  |
| course\_description | text |  | Null |  |
| course\_title | text |  | Not null |  |
| course\_language\_id | integer | FK | Not null |  |
| course\_level\_id | integer | FK | Not null |  |
| course\_category\_id | integer | FK | Not null |  |

**Таблица 3.2.7 - Свойства атрибутов отношения таблицы «lesson»**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Атрибут | Тип (размер) | Уникальность | Null | Ограничения |
| lesson\_id | integer | PK | Not null |  |
| lesson\_start\_time | time |  | Not null |  |
| lesson\_end\_time | time |  | Not null |  |
| lesson\_teacher\_id | integer | FK | Not null |  |
| lesson\_course\_id | integer | FK | Not null |  |
| lesson\_weekday\_id | integer | FK | Not null |  |
| lesson\_type\_id | integer | FK | Not null |  |

**Таблица 3.2.8 - Свойства атрибутов отношения таблицы «account»**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Атрибут | Тип (размер) | Уникальность | Null | Ограничения |
| account\_login | text | PK | Not null |  |
| account\_password | text |  | Not null |  |
| account\_role | text |  | Not null |  |

**Таблица 3.2.9 - Свойства атрибутов отношения таблицы «teacher»**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Атрибут | Тип (размер) | Уникальность | Null | Ограничения |
| teacher\_id | integer | PK | Not null |  |
| teacher\_firstname | text |  | Not null |  |
| teacher\_lastname | text |  | Not null |  |
| teacher\_middle\_name | text |  | Null |  |
| teacher\_email | text |  | Not null |  |
| teacher\_phone\_number | text |  | Not null |  |
| teacher\_birthdate | date |  | Not null |  |
| teacher\_experience | integer |  | Not null |  |
| teacher\_description | text |  | Null |  |
| teacher\_account\_id | text | FK | Null | Unique |

**Таблица 3.2.10 - Свойства атрибутов отношения таблицы «student»**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Атрибут | Тип (размер) | Уникальность | Null | Ограничения |
| student\_id | integer | PK | Not null |  |
| student\_firstname | text |  | Not null |  |
| student\_lastname | text |  | Not null |  |
| student\_middle\_name | text |  | Null |  |
| student\_email | text |  | Not null |  |
| student\_phone\_number | text |  | Not null |  |
| student\_birthdate | date |  | Not null |  |
| student\_account\_id | text | FK | Null | Unique |

## 3.3 Определение внешних ключей

Перечень внешних ключей представлен в таблице 3.3.1.

**Таблица 3.3.1 – Перечень внешних ключей**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Дочернее отношение | Внешний ключ | Тип (размер) | Null | Ссылка |
| course | course\_category\_id | integer | Not null | course\_category |
| course | course\_language\_id | integer | Not null | course\_language |
| course | course\_level\_id | integer | Not null | course\_level |
| lesson | lesson\_course\_id | integer | Not null | lesson\_course |
| lesson | lesson\_weekday\_id | integer | Not null | lesson\_weekday |
| lesson | lesson\_type\_id | integer | Not null | lesson\_type |
| lesson | lesson\_teacher\_id | integer | Not null | lesson\_teacher |

## 3.4 Определение внешних ключей

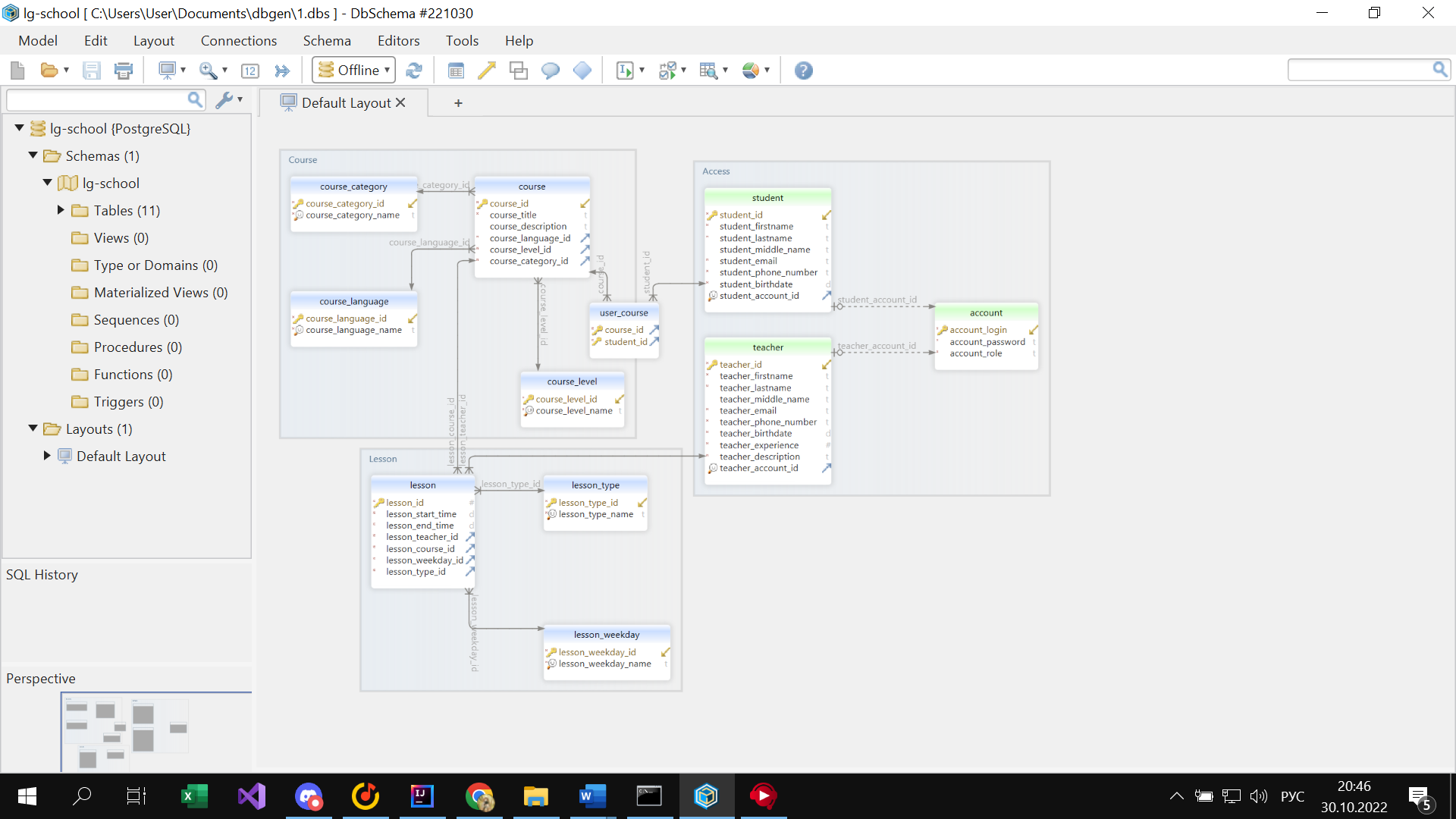
Свойства атрибутов отношения-связки «user\_course» представлены в таблице 3.4.1.

**Таблица 3.4.1 - Свойства атрибутов отношения-связки «user\_course»**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Атрибут | Тип (размер) | Уникальность | Null | Ссылка |
| course\_id | integer | PK FK | Not null | course |
| student\_id | integer | PK FK | Not null | student |

## 3.5 Концептуальная схема БД

Концептуальная модель реализованной базы данных представлена на рис. 3.5.1.

Рисунок 3.5.1 – Концептуальная модель базы данных

# IV. РЕАЛИЗАЦИЯ БАЗЫ ДАННЫХ

В данной работе используется PostgreSQL. Выбор был обусловлен бесплатностью, поддержкой БД неограниченного размера, мощными и надёжными механизмами транзакций и репликации, а также огромным количеством документации.

Для создания базы данных используется Java Persistence API и её реализация Hibernate. Описание атрибутов и таблиц происходит в классах сущностей.

Формирование таблиц базы данных происходит путём создания классов POJO и специальных аннотаций.

@Entity – даёт программе понять, что должны формироваться сущности.

@Id – отвечает за первичный ключ.

@Column – определяет имя, уникальность, обязательность и тип атрибута.

@ManyToOne – формирует отношение «многие к одному»

@OneToMany – формирует отношение «один ко многому»

@JoinColumn – определяет внешний ключ.

@JoinTable – создаёт таблицу для связи «многие ко многим» и определяет её поля.

@Getter, @Setter, @NoArgsContructor автоматически создают геттеры, сеттеры, конструктор без параметров.

В свойства @OneToOne @ManyToMany cascade = CascadeType.REMOVE означает каскадное удаление записей.

Программная реализация показана на рисунках 4.1 – 4.10.

На рисунке 4.11 указано исходное состояние базы данных после создания сущностей.

На рисунке 4.12 показаны исходные данные базы данных.

На рисунках 4.13 – 4.23 показано состояние таблиц базы данных после загрузки исходных данных.

@Data   
@Entity  
@Table(name = "account")  
public class Account {  
  
 @Id   
 @Column(columnDefinition="TEXT", name = "account\_login")  
 private String login;  
  
 @Column(columnDefinition="TEXT", name = "account\_password", nullable = false)  
 private String password;  
  
 @Column(columnDefinition="TEXT", name = "account\_role", nullable = false)  
 private String role;  
  
}

Рисунок 4.1. Программная реализация таблицы «account»

@Getter   
@Setter   
@NoArgsConstructor   
@Entity   
@Table(name = "course")  
public class Course {  
  
 @Id   
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 @Column(name = "course\_id")  
 private Integer id;  
  
 @Column(columnDefinition="TEXT", name = "course\_title", nullable = false)  
 private String title;  
  
 @Column(columnDefinition="TEXT", name = "course\_description", nullable = true)  
 private String description;  
  
 @ManyToOne   
 @JoinColumn(name="course\_language\_id", nullable=false)  
 private CourseLanguage language;  
  
 @ManyToOne   
 @JoinColumn(name="course\_level\_id", nullable=false)  
 private CourseLevel level;  
  
 @ManyToOne   
 @JoinColumn(name="course\_category\_id", nullable=false)  
 private CourseCategory category;  
  
 @OneToMany(mappedBy = "course", cascade = CascadeType.REMOVE)  
 private Set<Lesson> lessons;  
  
 @ManyToMany   
 @JoinTable(  
 name = "user\_course",  
 joinColumns = @JoinColumn(name = "course\_id"),  
 inverseJoinColumns = @JoinColumn(name = "student\_id"))  
 Set<Student> students;  
}

Рисунок 4.2. Программная реализация таблицы «course»

@Data   
@Entity   
@Table(name = "course\_category")  
public class CourseCategory {  
  
 @Id   
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 @Column(name = "course\_category\_id")  
 private Integer id;  
  
 @Column(columnDefinition="TEXT", name = "course\_category\_name", unique = true, nullable = false)  
 private String name;  
  
 @OneToMany(mappedBy="category")  
 private Set<Course> categories;  
  
}

Рисунок 4.3. Программная реализация таблицы «course\_category»

@Data   
@Entity   
@Table(name = "course\_language")  
public class CourseLanguage {  
  
 @Id   
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 @Column(name = "course\_language\_id")  
 private Integer id;  
  
 @Column(columnDefinition="TEXT", name = "course\_language\_name", unique = true, nullable = false)  
 private String name;  
  
 @OneToMany(mappedBy="language")  
 private Set<Course> courses;  
  
}

Рисунок 4.4. Программная реализация таблицы «course\_language»

@Data   
@Entity   
@Table(name = "course\_level")  
public class CourseLevel {  
  
 @Id   
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 @Column(name = "course\_level\_id")  
 private Integer id;  
  
 @Column(columnDefinition="TEXT", name = "course\_level\_name", unique = true, nullable = false)  
 private String name;  
  
 @OneToMany(mappedBy="level")  
 private Set<Course> levels;  
  
}

Рисунок 4.5. Программная реализация таблицы «course\_level»

@Getter   
@Setter   
@NoArgsConstructor   
@Entity   
@Table(name = "lesson")  
public class Lesson {  
  
 @Id   
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 @Column(name = "lesson\_id")  
 private Integer id;  
  
 @Column(name = "lesson\_start\_time", nullable = false)  
 private LocalTime startTime;  
  
 @Column(name = "lesson\_end\_time", nullable = false)  
 private LocalTime endTime;  
  
 @ManyToOne   
 @JoinColumn(name = "lesson\_teacher\_id", nullable = false)  
 private Teacher teacher;  
  
 @ManyToOne   
 @JoinColumn(name="lesson\_course\_id", nullable = false)  
 private Course course;  
  
 @ManyToOne   
 @JoinColumn(name = "lesson\_weekday\_id", nullable = false)  
 private LessonWeekDay weekday;  
  
 @ManyToOne   
 @JoinColumn(name = "lesson\_type\_id", nullable = false)  
 private LessonType lessonType;  
   
}

Рисунок 4.6. Программная реализация таблицы «lesson»

@Data   
@Entity   
@Table(name = "lesson\_type")  
public class LessonType {  
  
 @Id   
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 @Column(name = "lesson\_type\_id")  
 private Integer id;  
  
 @Column(columnDefinition="TEXT", name = "lesson\_type\_name", unique = true, nullable = false)  
 private String name;  
  
 @OneToMany(mappedBy = "lessonType")  
 private Set<Lesson> lessons;  
  
}

Рисунок 4.7. Программная реализация таблицы «lesson\_type»

@Data   
@Entity   
@Table(name = "lesson\_weekday")  
public class LessonWeekDay {  
  
 @Id   
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 @Column(name = "lesson\_weekday\_id")  
 private Integer id;  
  
 @Column(columnDefinition="TEXT", name = "lesson\_weekday\_name", unique = true, nullable = false)  
 private String name;  
  
 @OneToMany(mappedBy = "weekday")  
 private Set<Lesson> lessons;  
  
}

Рисунок 4.8. Программная реализация таблицы «lesson\_weekday»

@Data   
@Entity   
@Table(name = "student")  
public class Student {  
  
 @Id   
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 @Column(name = "student\_id")  
 private Integer id;  
  
 @Column(columnDefinition="TEXT", name = "student\_firstname", nullable = false)  
 private String firstName;  
  
 @Column(columnDefinition="TEXT", name = "student\_lastname", nullable = false)  
 private String lastName;  
  
 @Column(columnDefinition="TEXT", name = "student\_middle\_name", nullable = true)  
 private String middleName;  
  
 @Column(columnDefinition="TEXT", name = "student\_email", nullable = false)  
 private String email;  
  
 @Column(columnDefinition="TEXT", name = "student\_phone\_number", nullable = false)  
 private String phoneNumber;  
  
 @Column(name = "student\_birthdate", nullable = false)  
 private Date birthdate;  
  
 @OneToOne(cascade = CascadeType.REMOVE)  
 @JoinColumn(name = "student\_account\_id", referencedColumnName = "account\_login", nullable = true, unique = true)  
 private Account account;  
  
 @ManyToMany(mappedBy = "students", cascade = CascadeType.REMOVE)  
 Set<Course> courses;  
}

Рисунок 4.9. Программная реализация таблицы «student»

@Data   
@Entity   
@Table(name = "teacher")  
public class Teacher {  
  
 @Id   
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 @Column(name = "teacher\_id")  
 private Integer id;  
  
 @Column(columnDefinition="TEXT", name = "teacher\_firstname", nullable = false)  
 private String firstName;  
  
 @Column(columnDefinition="TEXT", name = "teacher\_lastname", nullable = false)  
 private String lastName;  
  
 @Column(columnDefinition="TEXT", name = "teacher\_middle\_name", nullable = true)  
 private String middleName;  
  
 @Column(columnDefinition="TEXT", name = "teacher\_email", nullable = false)  
 private String email;  
  
 @Column(columnDefinition="TEXT", name = "teacher\_phone\_number", nullable = false)  
 private String phoneNumber;  
  
 @Column(name = "teacher\_birthdate", nullable = false)  
 private Date birthdate;  
  
 @Column(name = "teacher\_experience", nullable = false)  
 private Integer experience;  
  
 @Column(columnDefinition="TEXT", name = "teacher\_description", nullable = true)  
 private String description;  
  
 @OneToOne(cascade = CascadeType.REMOVE)  
 @JoinColumn(name = "teacher\_account\_id", referencedColumnName = "account\_login", nullable = true, unique = true)  
 private Account account;  
  
 @OneToMany(mappedBy = "teacher")  
 private Set<Lesson> lessons;  
  
}

Рисунок 4.10. Программная реализация таблицы «teacher»

List of relations

Schema | Name | Type | Owner

--------+----------------------------------------+----------+--------------------

public | account | table | j79080372\_lg\_admin  
 public | course | table | j79080372\_lg\_admin  
 public | course\_category | table | j79080372\_lg\_admin  
 public | course\_category\_course\_category\_id\_seq | sequence | j79080372\_lg\_admin  
 public | course\_course\_id\_seq | sequence | j79080372\_lg\_admin  
 public | course\_language | table | j79080372\_lg\_admin  
 public | course\_language\_course\_language\_id\_seq | sequence | j79080372\_lg\_admin  
 public | course\_level | table | j79080372\_lg\_admin  
 public | course\_level\_course\_level\_id\_seq | sequence | j79080372\_lg\_admin  
 public | lesson | table | j79080372\_lg\_admin  
 public | lesson\_lesson\_id\_seq | sequence | j79080372\_lg\_admin  
 public | lesson\_type | table | j79080372\_lg\_admin  
 public | lesson\_type\_lesson\_type\_id\_seq | sequence | j79080372\_lg\_admin  
 public | lesson\_weekday | table | j79080372\_lg\_admin  
 public | lesson\_weekday\_lesson\_weekday\_id\_seq | sequence | j79080372\_lg\_admin  
 public | student | table | j79080372\_lg\_admin  
 public | student\_student\_id\_seq | sequence | j79080372\_lg\_admin  
 public | teacher | table | j79080372\_lg\_admin  
 public | teacher\_teacher\_id\_seq | sequence | j79080372\_lg\_admin  
 public | user\_course | table | j79080372\_lg\_admin

Рисунок 4.11.Состояние базы данных

INSERT INTO course\_category (course\_category\_name) VALUES ('Школьники');  
INSERT INTO course\_category (course\_category\_name) VALUES ('Взрослые');  
INSERT INTO course\_category (course\_category\_name) VALUES ('Дошкольники');  
  
INSERT INTO course\_language (course\_language\_name) VALUES ('Английский');  
INSERT INTO course\_language (course\_language\_name) VALUES ('Немецкий');  
INSERT INTO course\_language (course\_language\_name) VALUES ('Китайский');  
  
INSERT INTO course\_level (course\_level\_name) VALUES ('А1');  
INSERT INTO course\_level (course\_level\_name) VALUES ('А2');  
INSERT INTO course\_level (course\_level\_name) VALUES ('B1');  
INSERT INTO course\_level (course\_level\_name) VALUES ('B2');  
INSERT INTO course\_level (course\_level\_name) VALUES ('C1');  
INSERT INTO course\_level (course\_level\_name) VALUES ('C2');  
  
INSERT INTO lesson\_type (lesson\_type\_name) VALUES ('Практика');  
INSERT INTO lesson\_type (lesson\_type\_name) VALUES ('Носитель');  
INSERT INTO lesson\_type (lesson\_type\_name) VALUES ('Лекция');  
  
INSERT INTO lesson\_weekday (lesson\_weekday\_name) VALUES ('Пн');  
INSERT INTO lesson\_weekday (lesson\_weekday\_name) VALUES ('Вт');  
INSERT INTO lesson\_weekday (lesson\_weekday\_name) VALUES ('Ср');  
INSERT INTO lesson\_weekday (lesson\_weekday\_name) VALUES ('Чт');  
INSERT INTO lesson\_weekday (lesson\_weekday\_name) VALUES ('Пт');  
  
INSERT INTO course (course\_title, course\_category\_id, course\_language\_id, course\_level\_id) VALUES ('Course 1', 2, 1, 1);  
INSERT INTO course (course\_title, course\_category\_id, course\_language\_id, course\_level\_id) VALUES ('Course 2', 2, 1, 2);  
INSERT INTO course (course\_title, course\_category\_id, course\_language\_id, course\_level\_id) VALUES ('Course 3', 2, 2, 3);  
INSERT INTO course (course\_title, course\_category\_id, course\_language\_id, course\_level\_id) VALUES ('Course 4', 2, 3, 4);  
INSERT INTO course (course\_title, course\_category\_id, course\_language\_id, course\_level\_id) VALUES ('Course 5', 1, 1, 5);  
INSERT INTO course (course\_title, course\_category\_id, course\_language\_id, course\_level\_id) VALUES ('Course 6', 1, 1, 6);  
  
INSERT INTO account (account\_login, account\_password, account\_role) VALUES ('admin', 'admin', 'ROLE\_ADMIN');  
INSERT INTO account (account\_login, account\_password, account\_role) VALUES ('student', 'student', 'ROLE\_STUDENT');  
INSERT INTO account (account\_login, account\_password, account\_role) VALUES ('teacher', 'teacher', 'ROLE\_TEACHER');  
INSERT INTO account (account\_login, account\_password, account\_role) VALUES ('e\_student', 'e\_student', 'ROLE\_STUDENT');  
INSERT INTO account (account\_login, account\_password, account\_role) VALUES ('e\_teacher', 'e\_teacher', 'ROLE\_TEACHER');  
  
INSERT INTO account (account\_login, account\_password, account\_role) VALUES ('KrutinAI', 'a174deb3f', 'ROLE\_STUDENT');  
INSERT INTO account (account\_login, account\_password, account\_role) VALUES ('LaptevaYD', 'b5f16b066', 'ROLE\_STUDENT');  
INSERT INTO account (account\_login, account\_password, account\_role) VALUES ('KamenevPV', '2229cd752', 'ROLE\_STUDENT');  
INSERT INTO account (account\_login, account\_password, account\_role) VALUES ('GrishkinaSK', 'adf247800', 'ROLE\_STUDENT');  
INSERT INTO account (account\_login, account\_password, account\_role) VALUES ('VeselovNF', 'ec1e061c5', 'ROLE\_STUDENT');  
  
INSERT INTO teacher (teacher\_birthdate, teacher\_email, teacher\_experience, teacher\_firstname, teacher\_lastname, teacher\_middle\_name, teacher\_phone\_number) VALUES ('06.02.1994', 'prohor6610@outlook.com', 3, 'Прохор', 'Ишутин', 'Прокльев ', '+7 (976) 364-85-87');  
INSERT INTO teacher (teacher\_birthdate, teacher\_email, teacher\_experience, teacher\_firstname, teacher\_lastname, teacher\_middle\_name, teacher\_phone\_number, teacher\_account\_id) VALUES ('10.13.1978', 'gerasim.belov@outlook.com', 20, 'Герасим', 'Белов', 'Феликсович', '+7 (927) 459-84-84', 'admin');  
INSERT INTO teacher (teacher\_birthdate, teacher\_email, teacher\_experience, teacher\_firstname, teacher\_lastname, teacher\_middle\_name, teacher\_phone\_number, teacher\_account\_id) VALUES ('10.28.1996', 'nadejda1996@ya.ru', 4, 'Надежда', 'Хабенская', 'Саввановна', '+7 (949) 612-47-52', 'teacher');  
  
INSERT INTO student (student\_birthdate, student\_email, student\_firstname, student\_lastname, student\_middle\_name, student\_phone\_number, student\_account\_id) VALUES ('05.27.1998', 'anton.krutin@hotmail.com', 'Антон', 'Крутин', 'Ираклиевич', '+7 (931) 506-27-41', 'KrutinAI');  
INSERT INTO student (student\_birthdate, student\_email, student\_firstname, student\_lastname, student\_middle\_name, student\_phone\_number, student\_account\_id) VALUES ('09.06.1999', 'yaroslava.lapteva@hotmail.com', 'Ярослава', 'Лаптева', 'Давидовна', '+7 (974) 833-51-65', 'LaptevaYD');  
INSERT INTO student (student\_birthdate, student\_email, student\_firstname, student\_lastname, student\_middle\_name, student\_phone\_number, student\_account\_id) VALUES ('03.11.2000', 'petr65@yandex.ru', 'Петр', 'Каменев', 'Валерьевич', '+7 (993) 349-20-39', 'KamenevPV');  
INSERT INTO student (student\_birthdate, student\_email, student\_firstname, student\_lastname, student\_middle\_name, student\_phone\_number, student\_account\_id) VALUES ('06.08.2001', 'serafima06021962@ya.ru', 'Серафима', 'Гришкина', 'Кирилловна', '+7 (998) 391-79-64', 'GrishkinaSK');  
INSERT INTO student (student\_birthdate, student\_email, student\_firstname, student\_lastname, student\_middle\_name, student\_phone\_number, student\_account\_id) VALUES ('06.02.1998', 'nikita22041978@outlook.com', 'Никита', 'Веселов', 'Феодосивич', '+7 (986) 660-85-83', 'VeselovNF');  
  
INSERT INTO user\_course (student\_id, course\_id) VALUES (1, 1);  
INSERT INTO user\_course (student\_id, course\_id) VALUES (1, 2);  
  
INSERT INTO lesson (lesson\_end\_time, lesson\_start\_time, lesson\_course\_id, lesson\_type\_id, lesson\_teacher\_id, lesson\_weekday\_id) VALUES ('13:30', '12:30', 1, 1, 1, 1);  
INSERT INTO lesson (lesson\_end\_time, lesson\_start\_time, lesson\_course\_id, lesson\_type\_id, lesson\_teacher\_id, lesson\_weekday\_id) VALUES ('13:30', '12:30', 1, 1, 2, 2);  
INSERT INTO lesson (lesson\_end\_time, lesson\_start\_time, lesson\_course\_id, lesson\_type\_id, lesson\_teacher\_id, lesson\_weekday\_id) VALUES ('13:30', '12:30', 2, 1, 2, 2);

Рисунок 4.12. Файл для загрузки изначальных данных

j79080372\_lg\_school=> SELECT \* FROM account;  
  
account\_login | account\_password | account\_role

---------------+------------------+--------------

admin | admin | ROLE\_ADMIN  
 student | student | ROLE\_STUDENT  
 teacher | teacher | ROLE\_TEACHER  
 e\_student | e\_student | ROLE\_STUDENT  
 e\_teacher | e\_teacher | ROLE\_TEACHER  
 KrutinAI | a174deb3f | ROLE\_STUDENT  
 LaptevaYD | b5f16b066 | ROLE\_STUDENT  
 KamenevPV | 2229cd752 | ROLE\_STUDENT  
 GrishkinaSK | adf247800 | ROLE\_STUDENT  
 VeselovNF | ec1e061c5 | ROLE\_STUDENT

Рисунок 4.13. Состояние таблицы «account» после загрузки данных

j79080372\_lg\_school=> \x  
Expanded display is on.  
j79080372\_lg\_school=> SELECT \* FROM course;  
-[ RECORD 1 ]------+---------  
course\_id | 1  
course\_description |  
course\_title | Course 1  
course\_category\_id | 2  
course\_language\_id | 1  
course\_level\_id | 1  
-[ RECORD 2 ]------+---------  
course\_id | 2  
course\_description |  
course\_title | Course 2  
course\_category\_id | 2  
course\_language\_id | 1  
course\_level\_id | 2  
-[ RECORD 3 ]------+---------  
course\_id | 3  
course\_description |  
course\_title | Course 3  
course\_category\_id | 2  
course\_language\_id | 2  
course\_level\_id | 3  
-[ RECORD 4 ]------+---------  
course\_id | 4  
course\_description |  
course\_title | Course 4  
course\_category\_id | 2  
course\_language\_id | 3  
course\_level\_id | 4  
-[ RECORD 5 ]------+---------  
course\_id | 5  
course\_description |  
course\_title | Course 5  
course\_category\_id | 1  
course\_language\_id | 1  
course\_level\_id | 5  
-[ RECORD 6 ]------+---------  
course\_id | 6  
course\_description |  
course\_title | Course 6  
course\_category\_id | 1  
course\_language\_id | 1  
course\_level\_id | 6

Рисунок 4.14. Состояние таблицы «course» после загрузки данных

j79080372\_lg\_school=> SELECT \* FROM teacher;

-[ RECORD 1 ]--------+--------------------------

teacher\_id | 1  
teacher\_birthdate | 1994-06-02  
teacher\_description |  
teacher\_email | prohor6610@outlook.com  
teacher\_experience | 3  
teacher\_firstname | Прохор  
teacher\_lastname | Ишутин  
teacher\_middle\_name | Прокльев  
teacher\_phone\_number | +7 (976) 364-85-87  
teacher\_account\_id |  
-[ RECORD 2 ]--------+--------------------------  
teacher\_id | 2  
teacher\_birthdate | 1978-10-13  
teacher\_description |  
teacher\_email | gerasim.belov@outlook.com  
teacher\_experience | 20  
teacher\_firstname | Герасим  
teacher\_lastname | Белов  
teacher\_middle\_name | Феликсович  
teacher\_phone\_number | +7 (927) 459-84-84  
teacher\_account\_id | admin  
-[ RECORD 3 ]--------+--------------------------  
teacher\_id | 3  
teacher\_birthdate | 1996-10-28  
teacher\_description |  
teacher\_email | nadejda1996@ya.ru  
teacher\_experience | 4  
teacher\_firstname | Надежда  
teacher\_lastname | Хабенская  
teacher\_middle\_name | Саввановна  
teacher\_phone\_number | +7 (949) 612-47-52  
teacher\_account\_id | teacher

Рисунок 4.15. Состояние таблицы «teacher» после загрузки данных

j79080372\_lg\_school=> SELECT \* FROM STUDENT;

-[ RECORD 1 ]--------+------------------------------  
  
student\_id | 1  
student\_birthdate | 1998-05-27  
student\_email | anton.krutin@hotmail.com  
student\_firstname | Антон  
student\_lastname | Крутин  
student\_middle\_name | Ираклиевич  
student\_phone\_number | +7 (931) 506-27-41  
student\_account\_id | KrutinAI  
-[ RECORD 2 ]--------+------------------------------  
student\_id | 2  
student\_birthdate | 1999-09-06  
student\_email | yaroslava.lapteva@hotmail.com  
student\_firstname | Ярослава  
student\_lastname | Лаптева  
student\_middle\_name | Давидовна  
student\_phone\_number | +7 (974) 833-51-65  
student\_account\_id | LaptevaYD  
-[ RECORD 3 ]--------+------------------------------  
student\_id | 3  
student\_birthdate | 2000-03-11  
student\_email | petr65@yandex.ru  
student\_firstname | Петр  
student\_lastname | Каменев  
student\_middle\_name | Валерьевич  
student\_phone\_number | +7 (993) 349-20-39  
student\_account\_id | KamenevPV  
-[ RECORD 4 ]--------+------------------------------  
student\_id | 4  
student\_birthdate | 2001-06-08  
student\_email | serafima06021962@ya.ru  
student\_firstname | Серафима  
student\_lastname | Гришкина  
student\_middle\_name | Кирилловна  
student\_phone\_number | +7 (998) 391-79-64  
student\_account\_id | GrishkinaSK  
-[ RECORD 5 ]--------+------------------------------  
student\_id | 5  
student\_birthdate | 1998-06-02  
student\_email | nikita22041978@outlook.com  
student\_firstname | Никита  
student\_lastname | Веселов  
student\_middle\_name | Феодосивич  
student\_phone\_number | +7 (986) 660-85-83  
student\_account\_id | VeselovNF

Рисунок 4.16. Состояние таблицы «student» после загрузки данных

 j79080372\_lg\_school=> SELECT \* FROM lesson;

-[ RECORD 1 ]-----+---------

lesson\_id | 1  
lesson\_end\_time | 13:30:00  
lesson\_start\_time | 12:30:00  
lesson\_course\_id | 1  
lesson\_type\_id | 1  
lesson\_teacher\_id | 1  
lesson\_weekday\_id | 1  
-[ RECORD 2 ]-----+---------  
lesson\_id | 2  
lesson\_end\_time | 13:30:00  
lesson\_start\_time | 12:30:00  
lesson\_course\_id | 1  
lesson\_type\_id | 1  
lesson\_teacher\_id | 2  
lesson\_weekday\_id | 2  
-[ RECORD 3 ]-----+---------  
lesson\_id | 3  
lesson\_end\_time | 13:30:00  
lesson\_start\_time | 12:30:00  
lesson\_course\_id | 2  
lesson\_type\_id | 1  
lesson\_teacher\_id | 2  
lesson\_weekday\_id | 2

Рисунок 4.17. Состояние таблицы «lesson» после загрузки данных

 j79080372\_lg\_school=> SELECT \* FROM course\_category;  
  
 course\_category\_id | course\_category\_name  
--------------------+----------------------  
 1 | Школьники  
 2 | Взрослые  
 3 | Дошкольники  
(3 rows)

Рисунок 4.18. Состояние таблицы «course\_category» после загрузки данных

j79080372\_lg\_school=> SELECT \* FROM course\_language;  
  
 course\_language\_id | course\_language\_name  
--------------------+----------------------  
 1 | Английский  
 2 | Немецкий  
 3 | Китайский  
(3 rows)

Рисунок 4.19. Состояние таблицы «course\_language» после загрузки данных

j79080372\_lg\_school=> SELECT \* FROM course\_level;  
  
 course\_level\_id | course\_level\_name  
-----------------+-------------------  
 1 | А1  
 2 | А2  
 3 | B1  
 4 | B2  
 5 | C1  
 6 | C2  
(6 rows)

Рисунок 4.20. Состояние таблицы «course\_level» после загрузки данных

j79080372\_lg\_school=> SELECT \* FROM lesson\_type;  
  
 lesson\_type\_id | lesson\_type\_name  
----------------+------------------  
 1 | Практика  
 2 | Носитель  
 3 | Лекция  
(3 rows)

Рисунок 4.21. Состояние таблицы «lesson\_type» после загрузки данных

j79080372\_lg\_school=> SELECT \* FROM lesson\_weekday;  
  
 lesson\_weekday\_id | lesson\_weekday\_name  
-------------------+---------------------  
 1 | Пн  
 2 | Вт  
 3 | Ср  
 4 | Чт  
 5 | Пт  
(5 rows)

Рисунок 4.22. Состояние таблицы «lesson\_weekday» после загрузки данных

j79080372\_lg\_school=> SELECT \* FROM user\_course;  
  
 course\_id | student\_id  
-----------+------------  
 1 | 1  
 2 | 1  
(2 rows)

Рисунок 4.23. Состояние таблицы «user\_course» после загрузки данных

# V. ЗАПРОСЫ К БАЗЕ ДАННЫХ

Основные запросы формируются фреймворком. Для создания репозитория конкретной сущности нужно создать новый класс и унаследовать его от CrudRepository<T, ID>. При необходимости можно создать дополнительные запросы прямо из имени методов. Для этого используется механизм префиксов find…By, read…By, query…By, count…By, и get…By, далее от префикса метода начинает разбор остальной части. Вводное предложение может содержать дополнительные выражения, например, Distinct. Далее первый By действует как разделитель, чтобы указать начало фактических критериев. Можно определить условия для свойств сущностей и объединить их с помощью And и Or. Интерфейс стандартной реализации показан на рисунке 5.1.

public interface CrudRepository<T, ID> extends Repository<T, ID> {  
 <S extends T> S save(S entity); // Сохраняет экземпляр сущности  
 <S extends T> Iterable<S> saveAll(Iterable<S> entities); // Сохраняет коллекцию экземпляров  
 Optional<T> findById(ID id); // Возвращает экземпляр по ключу  
 boolean existsById(ID id); // Проверяет на существование экземпляра  
 Iterable<T> findAll(); // Возвращает коллекцию экземпляров  
 Iterable<T> findAllById(Iterable<ID> ids); // Возвращает коллекцию экземпляров с заданным ключом  
 long count(); // Возвращает количество записей  
 void deleteById(ID id); // Удаляет по ключу  
 void delete(T entity); // Удаляет переданный экземпляр  
 void deleteAllById(Iterable<? extends ID> ids); // Удаляет все экземпляры с заданным ключом  
 void deleteAll(Iterable<? extends T> entities); // Удаляет коллекцию экземпляров  
 void deleteAll(); // Удаляет все записи в таблице  
}

Рисунок 5.1. Интерфейс CrudRepository

Дополнительные запросы:

Для формирования дополнительных запросов используется JPQL - платформонезависимый объектно-ориентированный язык запросов, являющийся частью спецификации JPA. JPQL используется для написания запросов к сущностям, хранящимся в реляционной базе данных.

JPQL во многом похож на SQL, но в отличие от последнего, оперирует запросами, составленными по отношению к сущностям JPA, в отличие от прямых запросов к таблицам базы данных. В дополнение к получению объектов (запросы SELECT), JPQL поддерживает запросы, основанные на операторах UPDATE и DELETE.

Запросы:

«SELECT CASE WHEN COUNT(s) > 0 THEN 'true' ELSE 'false' END FROM Student s WHERE s.account.login = ?1» - проверяет на наличие записей с аккаунтом ?1 в таблице учеников.

«SELECT CASE WHEN COUNT(t) > 0 THEN 'true' ELSE 'false' END FROM Teacher t WHERE t.account.login = ?1» - проверяет на наличие записей с аккаунтом ?1 в таблице преподавателей.

# VI. РУКОВОДСТВО ПРОГРАММИСТА

Приложение написано на Java версии 17. Для реализации веб-приложения использованы следующие библиотеки и фреймворки:

* Spring Boot 2.7.5.
* Spring Security
* Spring MVC
* Lombok
* Thymeleaf
* Bootstrap 5
* Simple-DataTables

Приложение состоит из следующих уровней:

* Уровень представления. Представляет контент конечному пользователю через графический интерфейс. Веб-страницы отправляются клиенту по запросу через контроллеры. Клиентом является веб-браузер.
* Уровень бизнес-логики. Управляет обработкой и валидацией данных. Состоит из сервисных классов. Может обращаться к уровню данных.
* Уровень данных. Организует хранение и извлечение данных из базы данных. Определяет реализацию основных CRUD операций. Представлен в виде репозиториев.

Приложение состоит из следующих пакетов:

* controller – содержит классы контроллеров.
* domain.model – содержит POJO (простые java объекты) с аннотациями для создания сущностей в базе данных.
* domain.enums – содержит перечисления.
* domain. repository – содержит репозитории для работы с базой данных.
* exception – содержит исключения, выбрасываемые при работе сервисов.
* security – содержит классы для настройки Spring Security и создания собственной системы авторизации.

Помимо пакетов приложение содержит статические html страницы и их стили, а также js скрипты для придания интерактивности в папке static.

В папке templates лежат шаблоны, обрабатываемые thymeleaf и выдаваемые клиенту.

В файле application.properties содержатся настройки для подключения к базе данных, а также дополнительные настройки, необходимые для корректной работы приложения.

Структура проекта:

├───java

│ └───com

│ └───pshandy

│ └───lgschool

│ ├───controller

│ │ ├───admin

│ │ ├───student

│ │ └───teacher

│ ├───domain

│ │ ├───enums

│ │ ├───model

│ │ └───repository

│ ├───exception

│ ├───security

│ │ └───configuration

│ └───service

└───resources

├───images

├───static

│ ├───assets

│ │ └───img

│ ├───css

│ ├───error

│ └───js

└───templates

├───admin

│ ├───account

│ ├───course

│ ├───course\_category

│ ├───course\_language

│ ├───course\_level

│ ├───lesson

│ ├───lesson\_type

│ ├───lesson\_weekday

│ ├───student

│ └───teacher

├───fragments

├───login

├───student

└───teacher

# VII. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

С целью приобретения навыков разработки информационного и программного обеспечения автоматизации деятельности специалистов организации или предприятия в рамках данного проекта было создано клиентское приложение для работы с базой данных "Школа иностранного языка".

Согласно техническому заданию, база данных позволяет вводить новые записи в любую таблицу базы данных, редактировать и удалять записи таблиц с обеспечением целостности данных; позволяет выполнять поиск и фильтрацию данных таблиц. Для облегчения взаимодействия пользователей с базой данных, программа также имеет дружественный интерфейс.

База данных содержит информацию о учениках, преподавателях, их аккаунтов, занятиях, их дней недели и тип, курсах, их категориях, уровнях и языках. Каждая из этих категорий имеют свою таблицу. Вы можете добавлять в каждую таблицу новые записи, редактировать и удалять старые при условии обеспечения целостности данных. Все действия, совершаемые в приложении, совершаются и над таблицами в самой базе данных.

Для получения доступа к функционалу, нужно перейти на сайт, где развернуто веб-приложение в персонального компьютера или другого устройства, дизайн адаптивный. В случае отсутствия авторизации будет автоматическая переадресация на страницу входа. (Рис. 7.1).

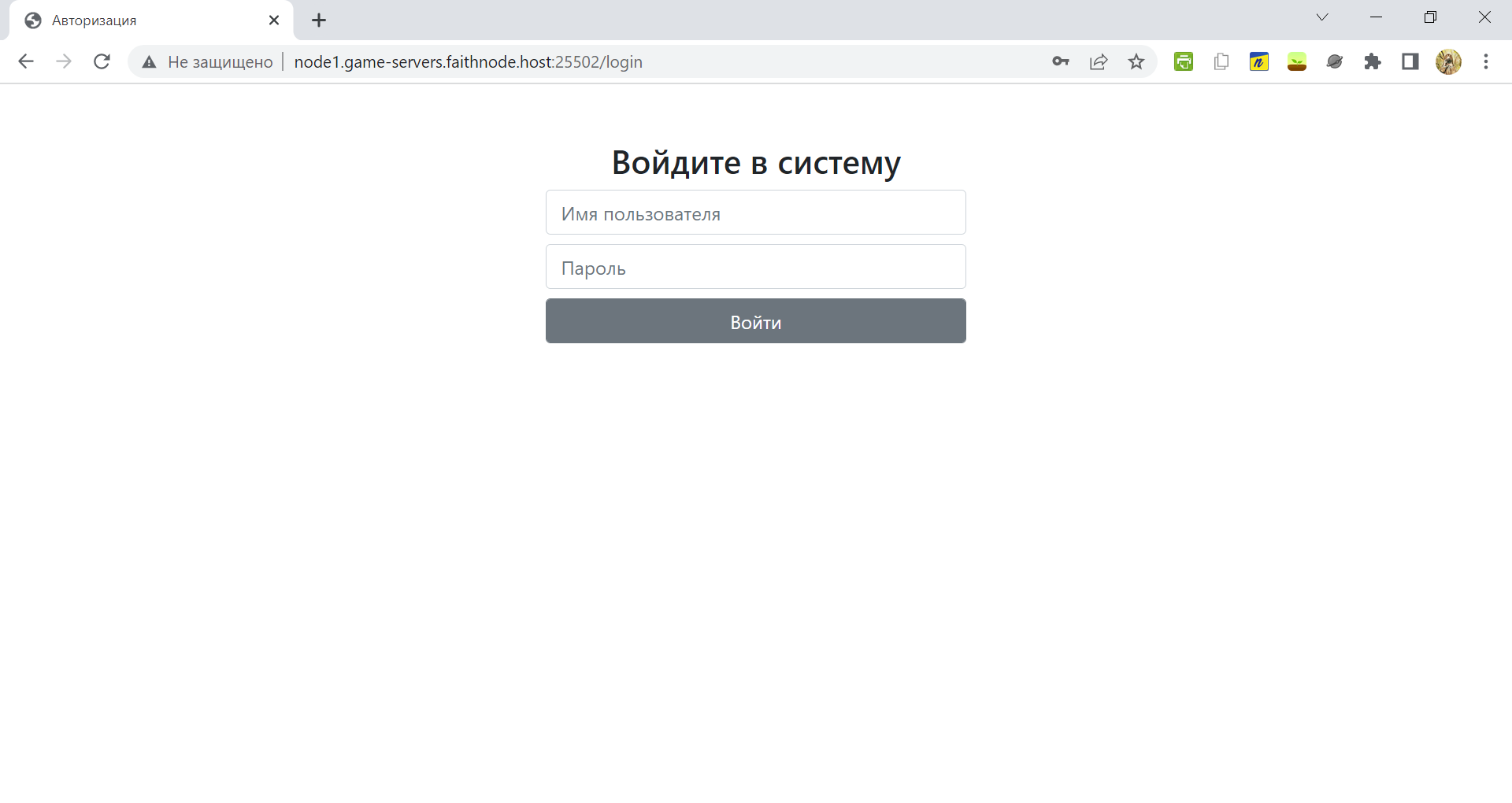


Рисунок 7.1. Страница входа в систему

В зависимости от роли будут выводиться разные страницы (рис. 7.2 – 7.4).

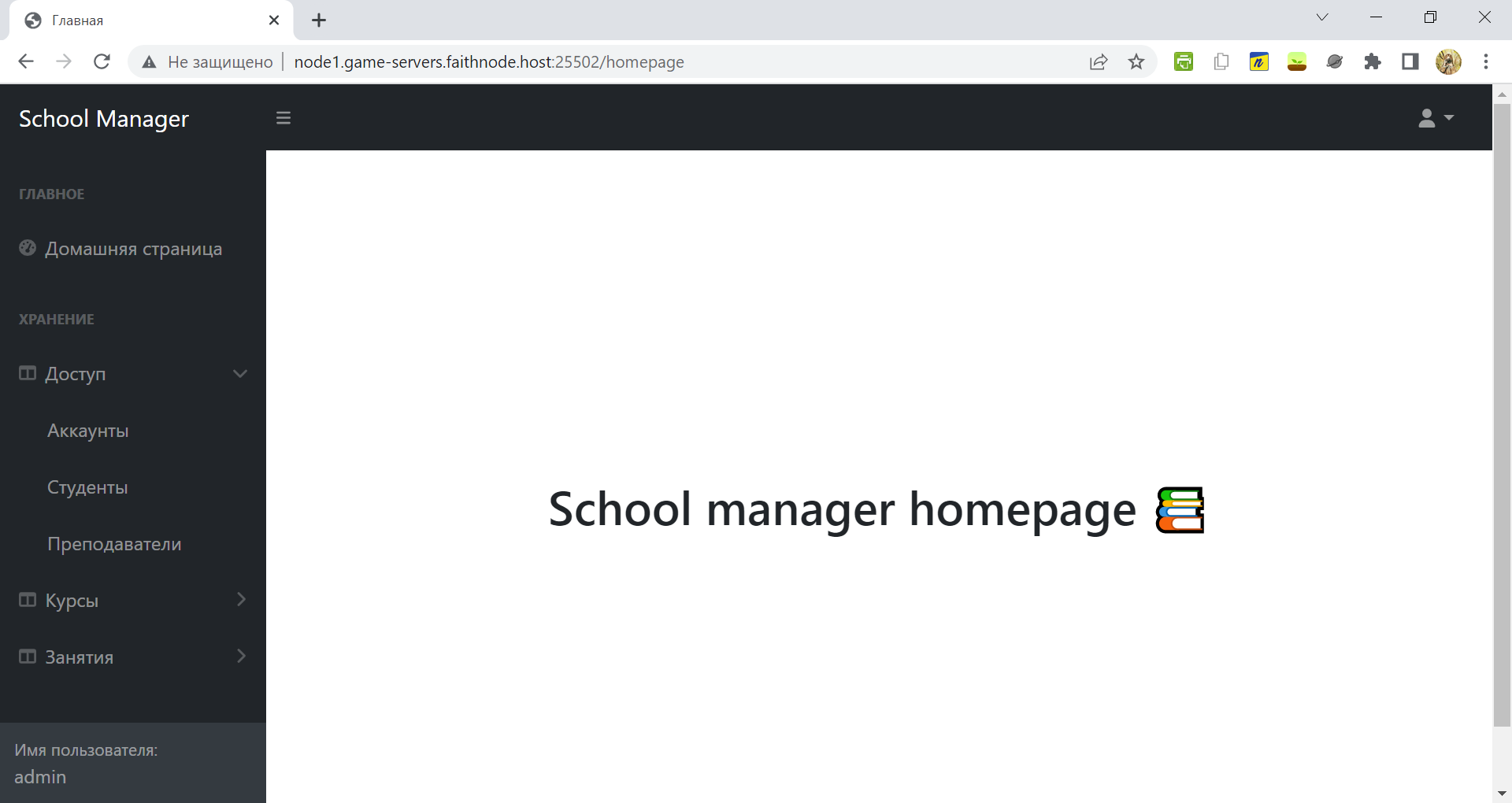


Рисунок 7.2. Интерфейс администратора

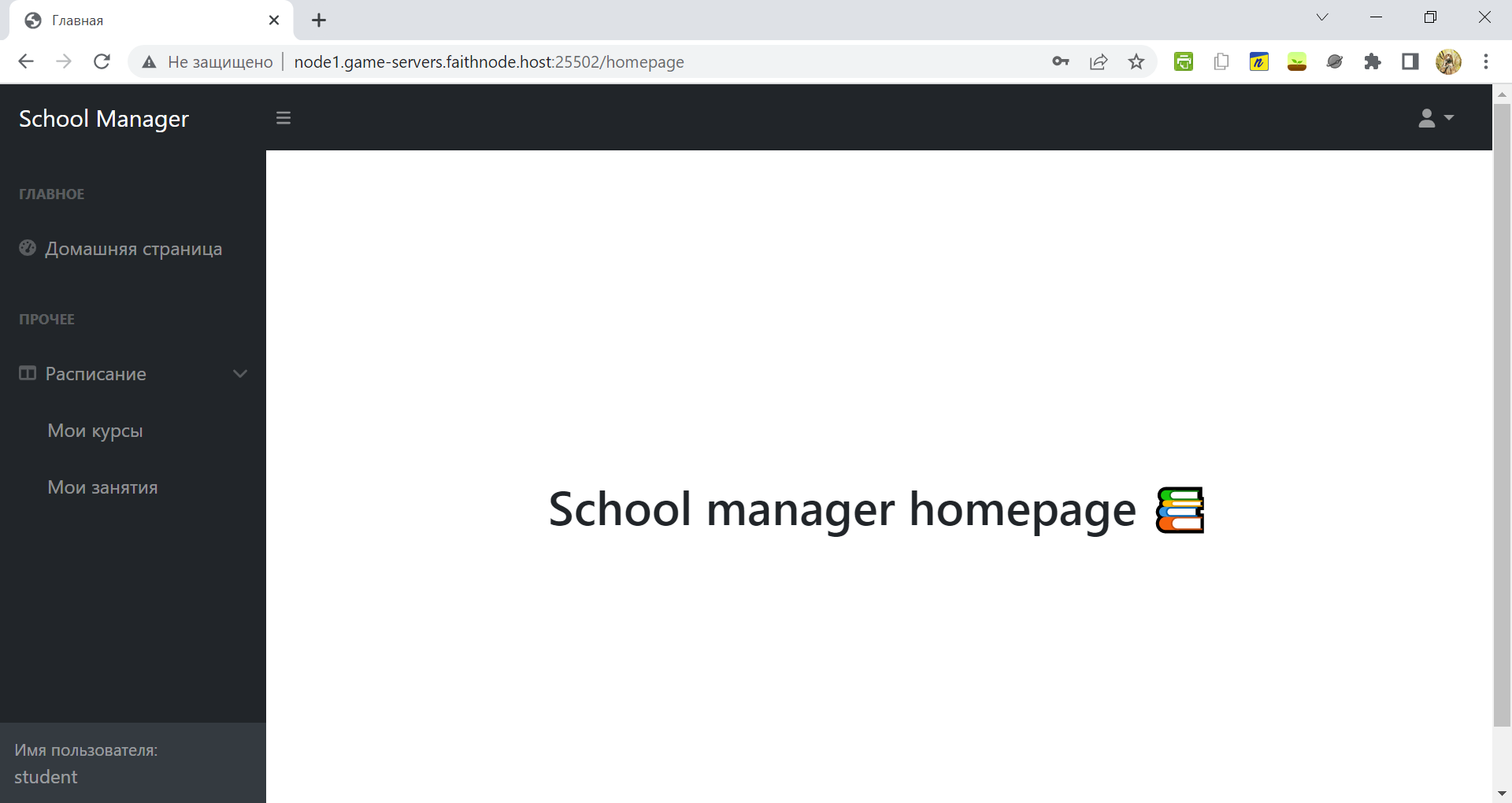


Рисунок 7.3. Интерфейс ученика

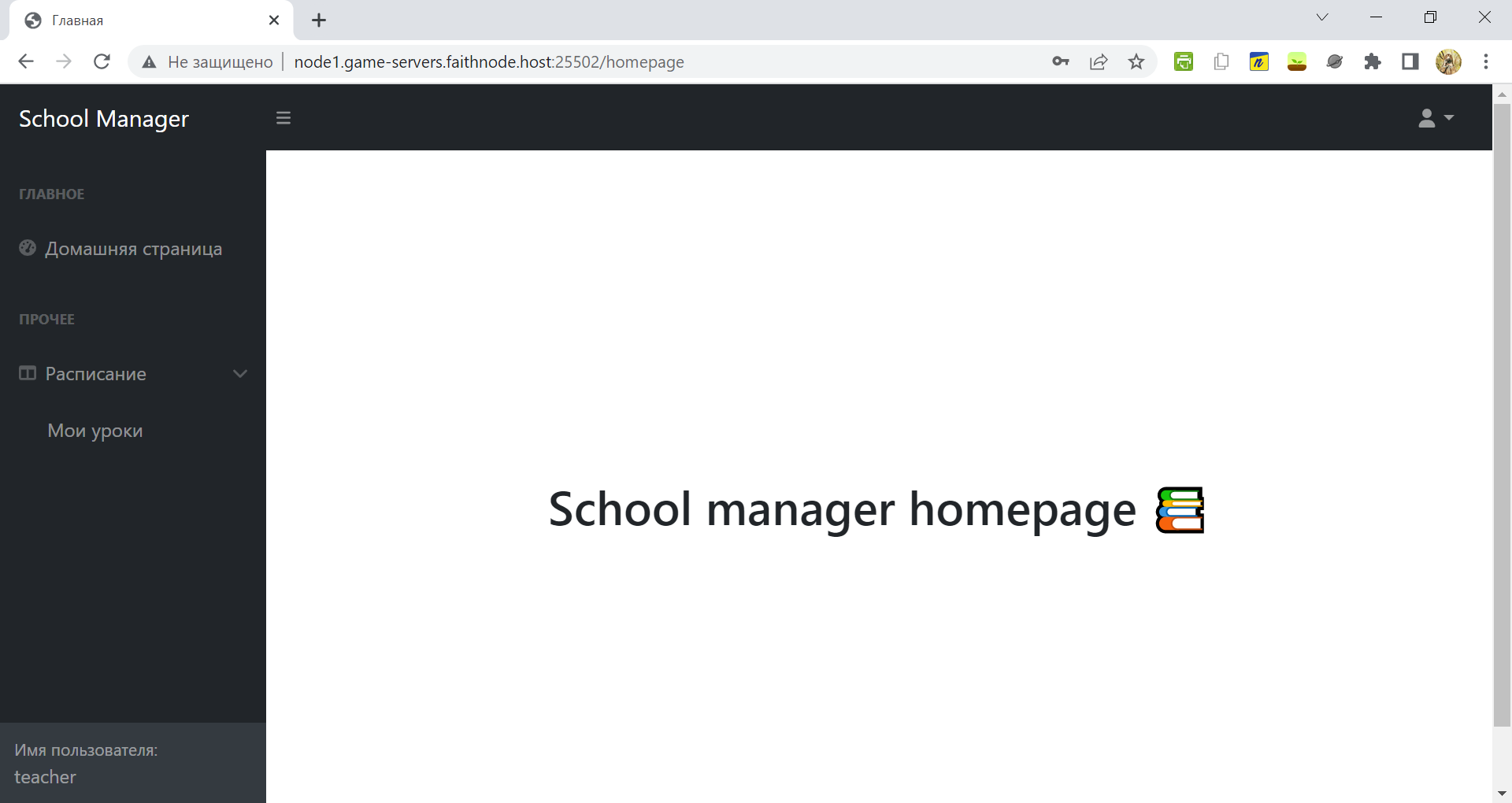


Рисунок 7.4. Интерфейс преподавателя

Рассмотрим возможности администратора. Администратор может удалять, изменять и редактировать любые записи в любых таблицах. Для поиска по таблице достаточно ввести в поле поиска ключевое слово и записи сами отсортируются (рис. 7.5 – 7.6).

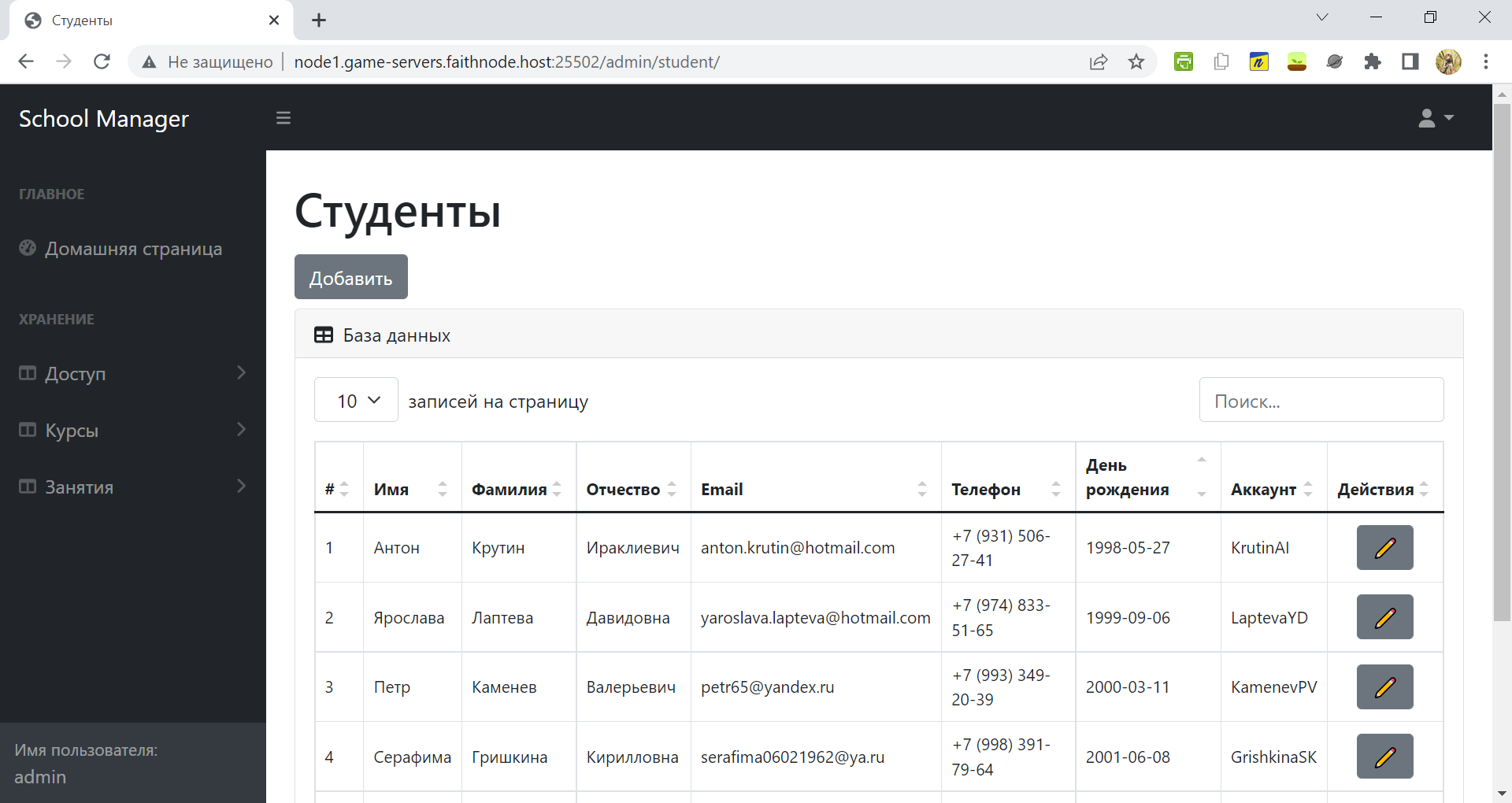


Рисунок 7.5. Таблица «Студенты»

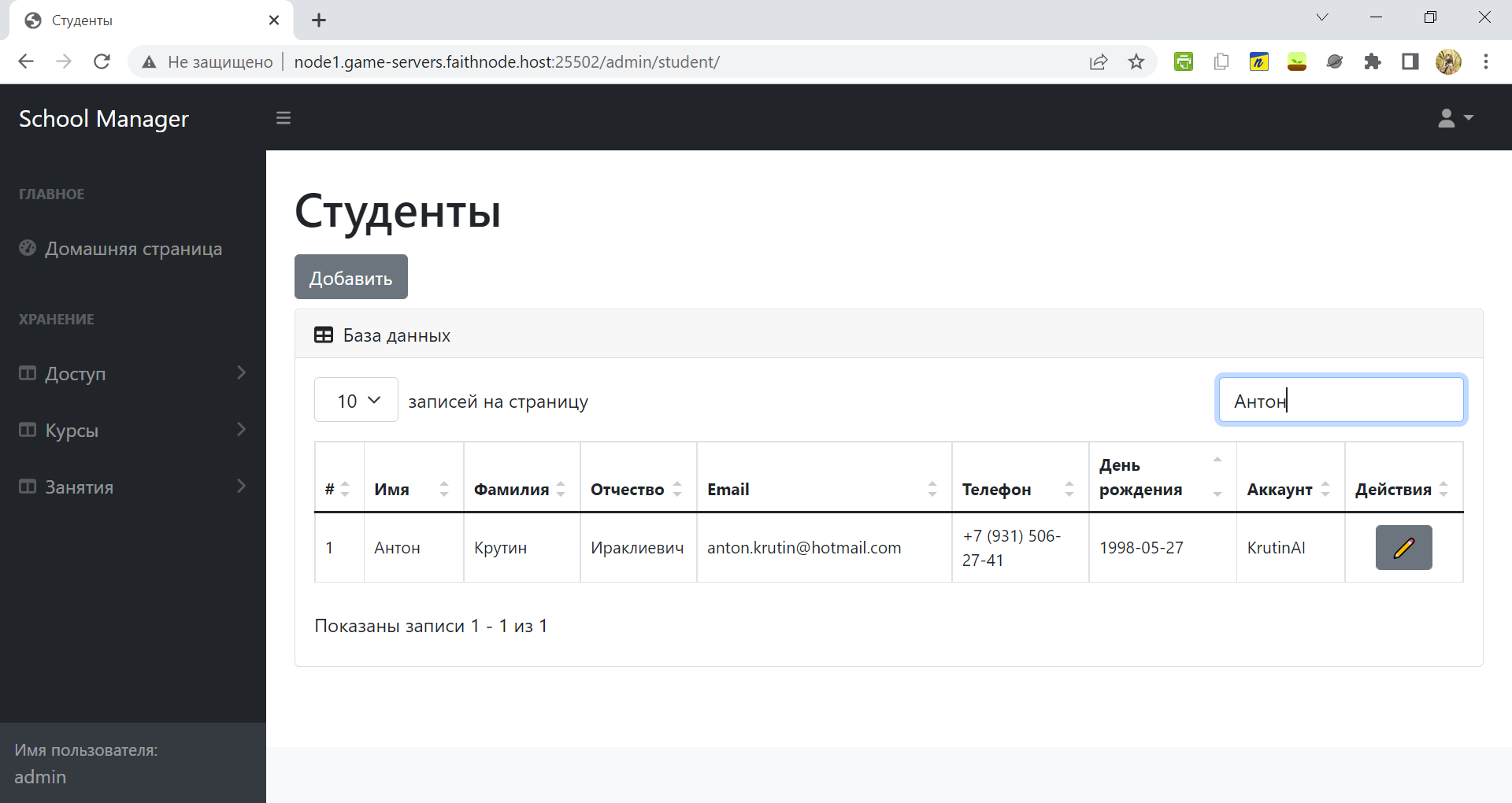


Рисунок 7.6. Поиск по таблице

При нажатии кнопки добавить появится форма для добавления нового студента (рис. 7.7).

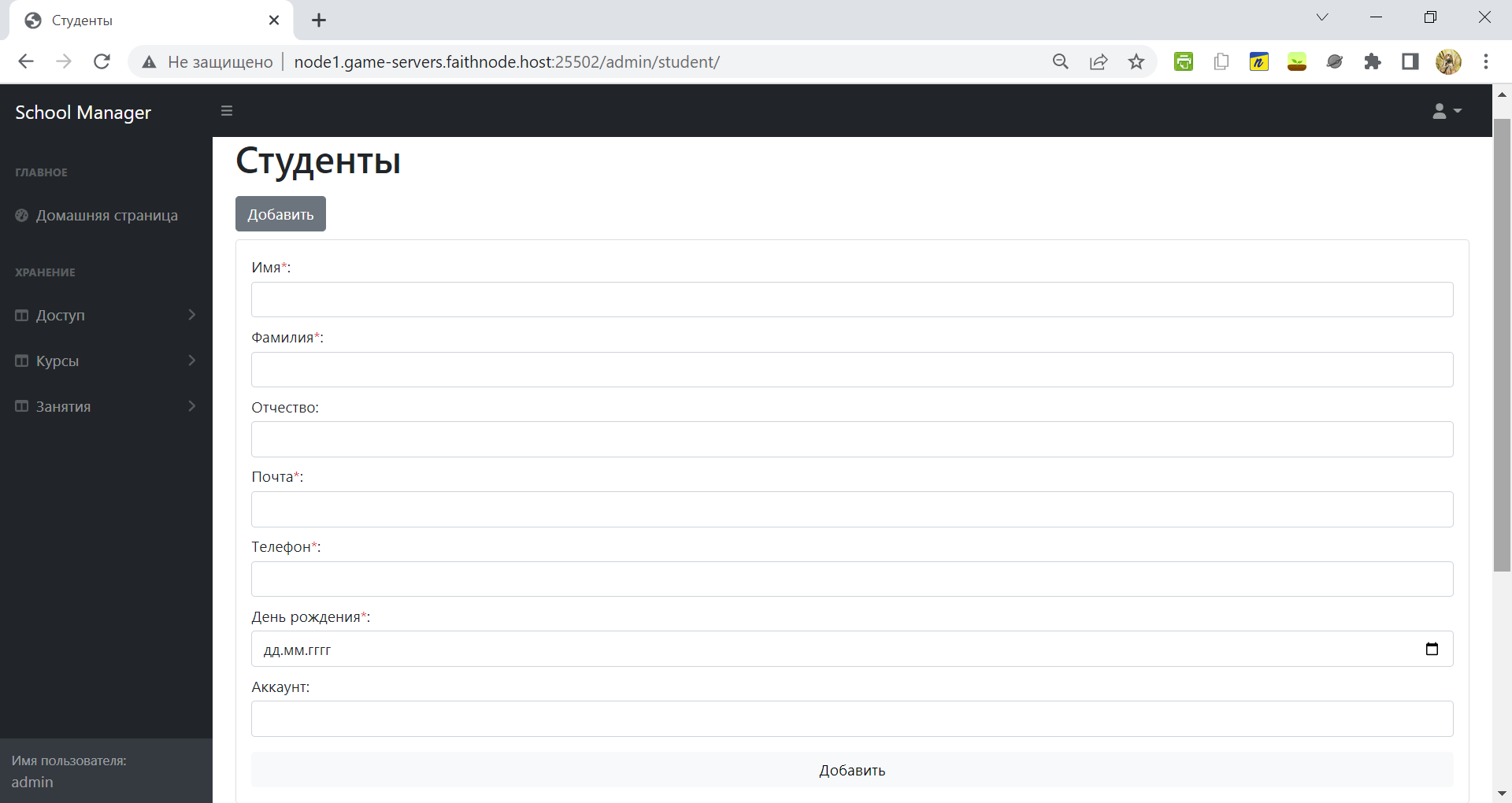


Рисунок 7.7. Добавление нового студента

В случае ошибок при заполнении на экране появится уведомляющее сообщение (рис. 7.8).

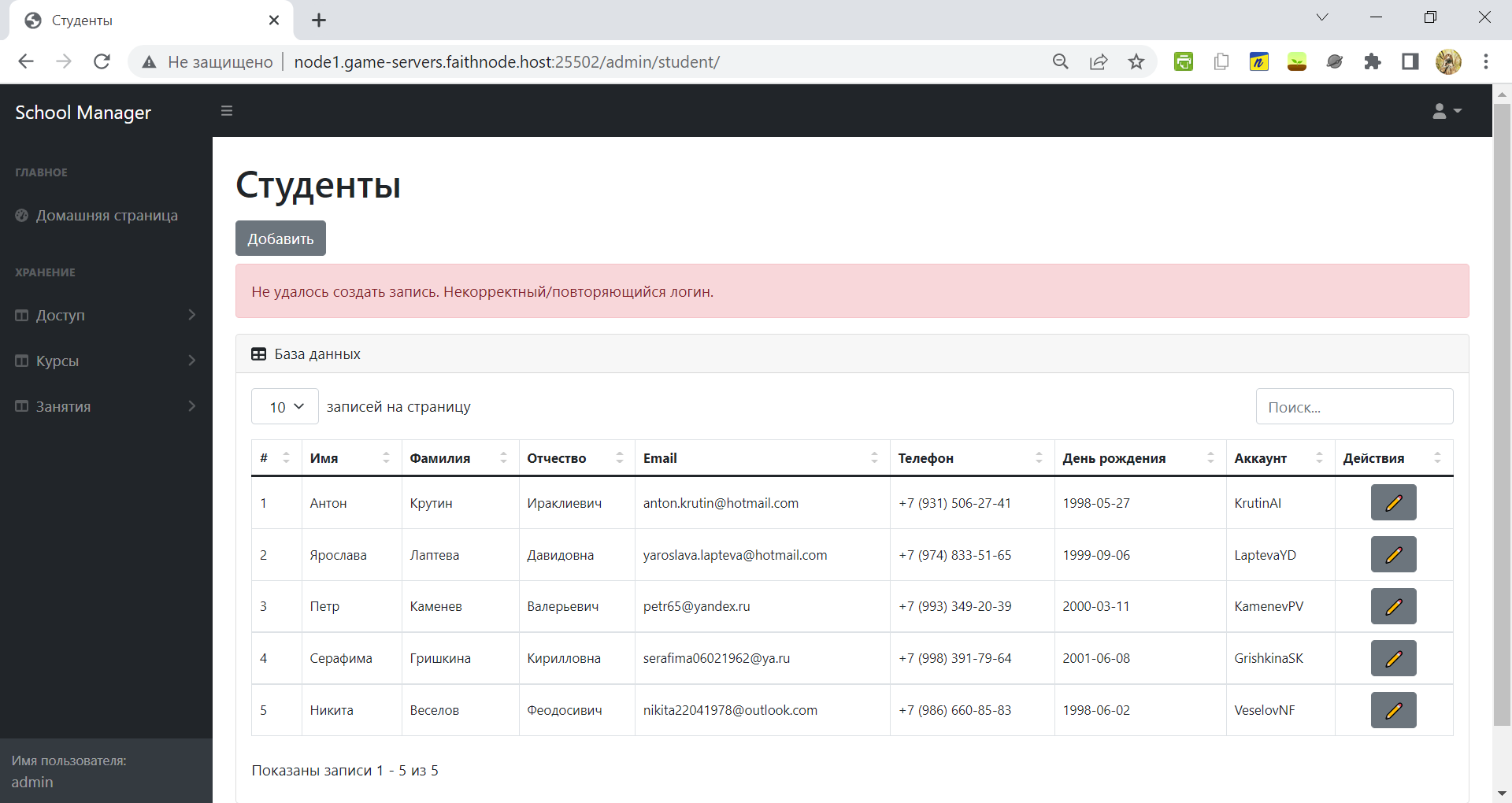


Рисунок 7.8. Уведомление об ошибке

Для изменения записи необходимо нажать на карандаш рядом с ней. Пользователя перенаправит на страницу редактирования (рис. 7.9).

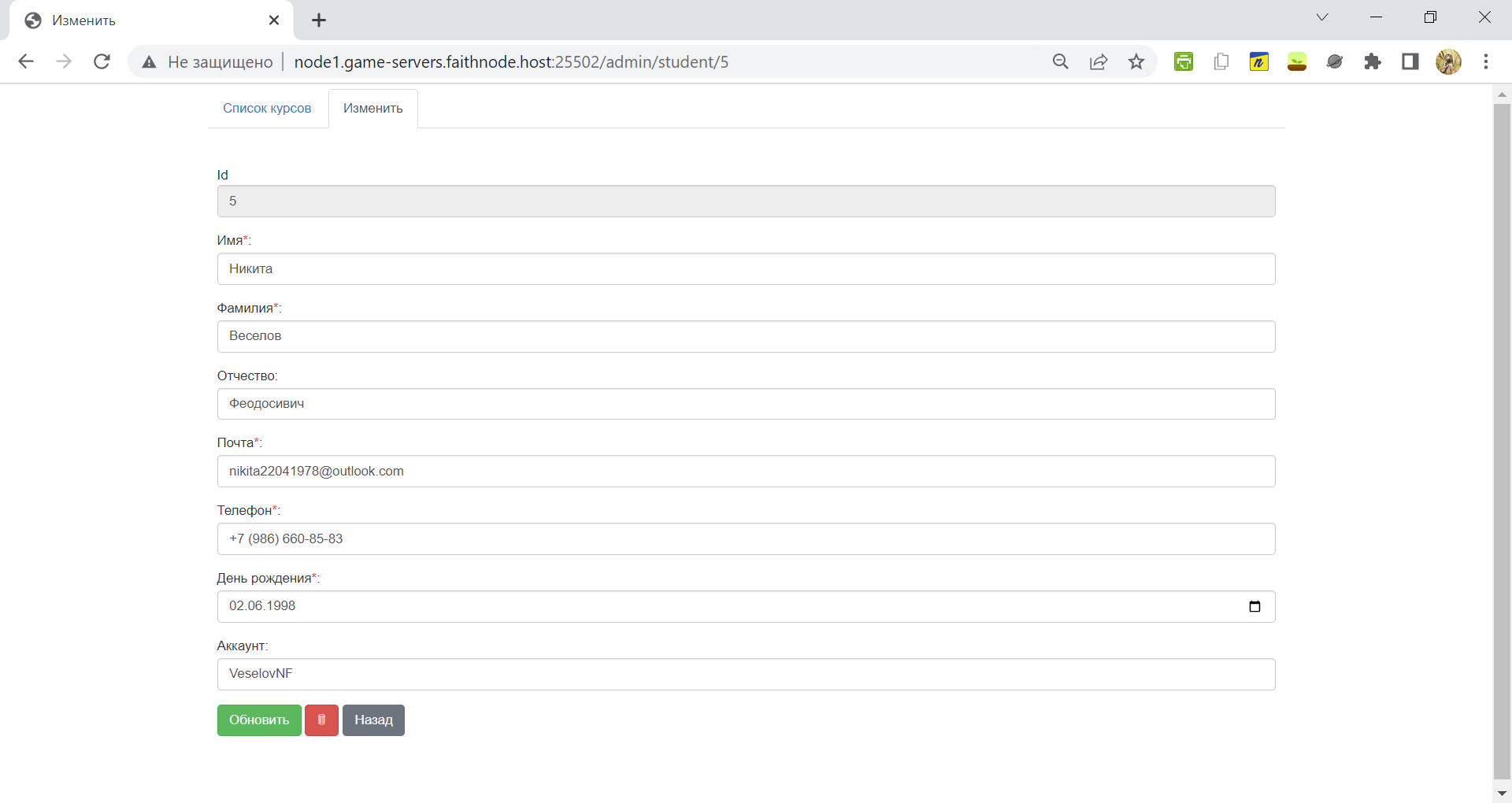


Рисунок 7.9. Редактирование записи в таблице «студенты»

Для сохранения изменений нужно нажать обновить, для удаления красную иконку с корзиной, а оставить без изменений – назад. В случае ошибок также будет выведено уведомляющее сообщение.

Большая часть страниц администратора построена похожим образом, за исключением студентов и курсов (рис. 7.10 – 7.11).

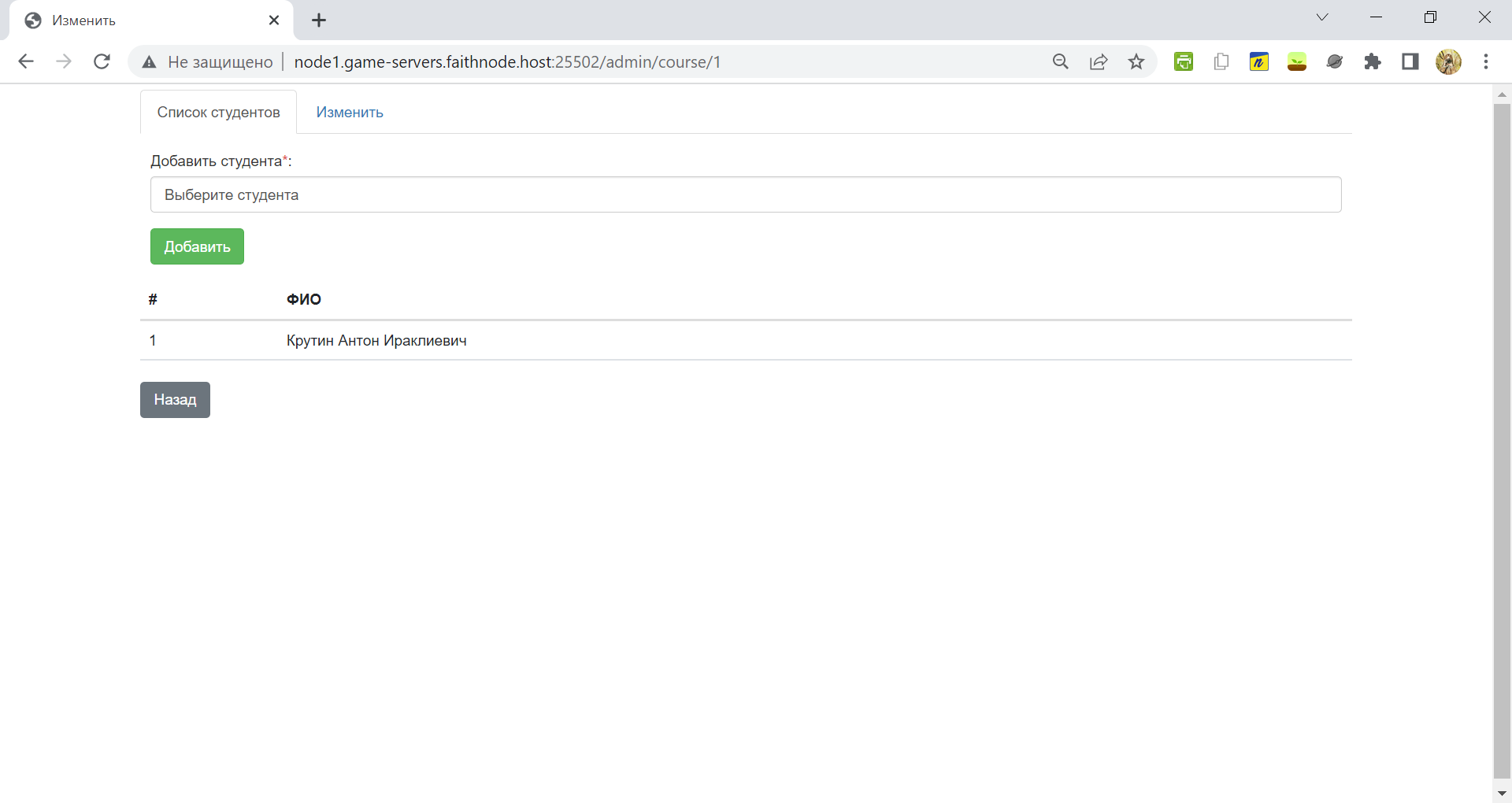


Рисунок 7.10. Редактирование списка студентов на курсе

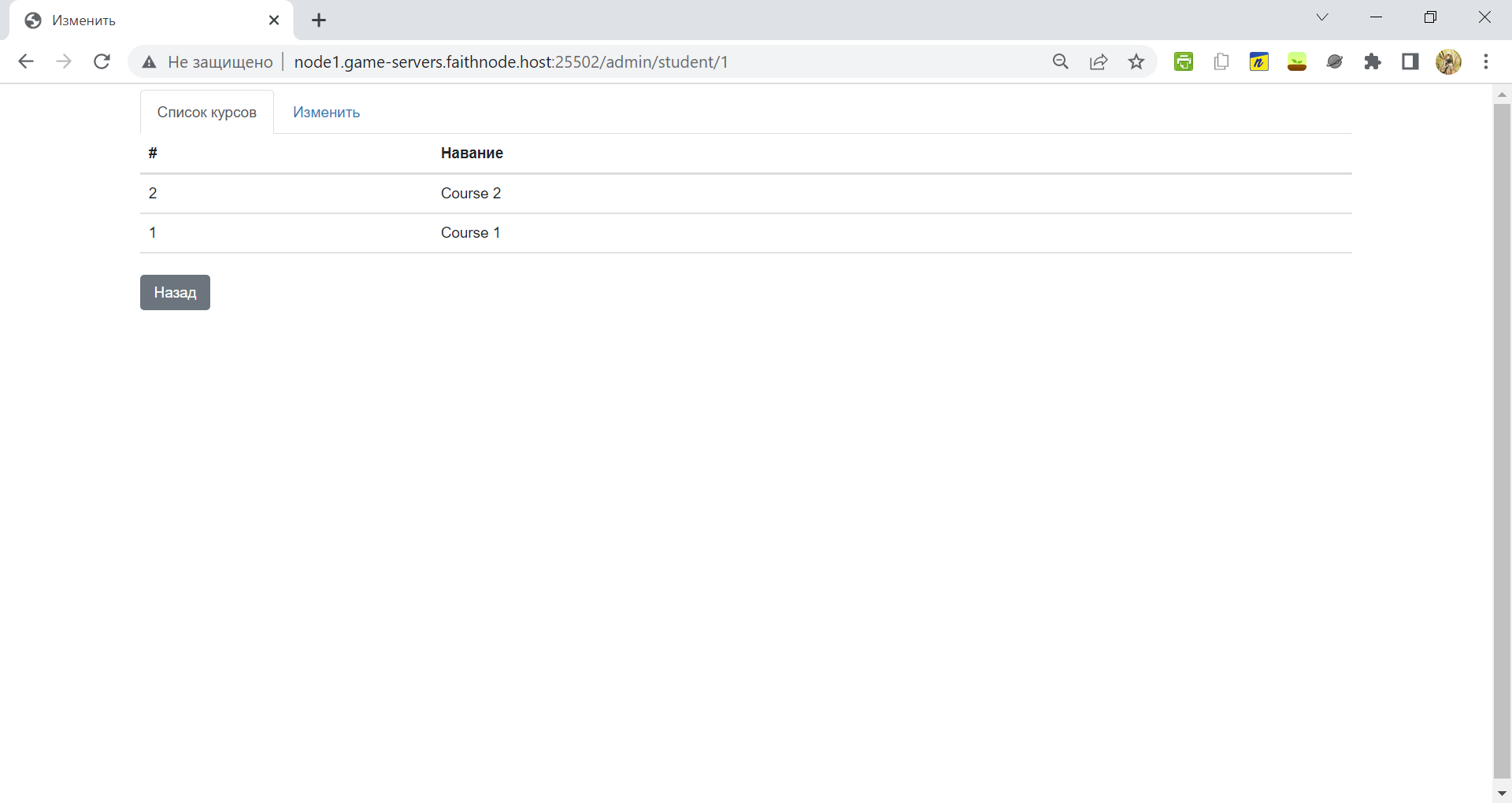


Рисунок 7.11. Просмотр списка курсов студента

Рассмотрим возможности ученика. У него имеется только права на чтение своего расписания и своих курсов (рис. 7.12 – 7.13).

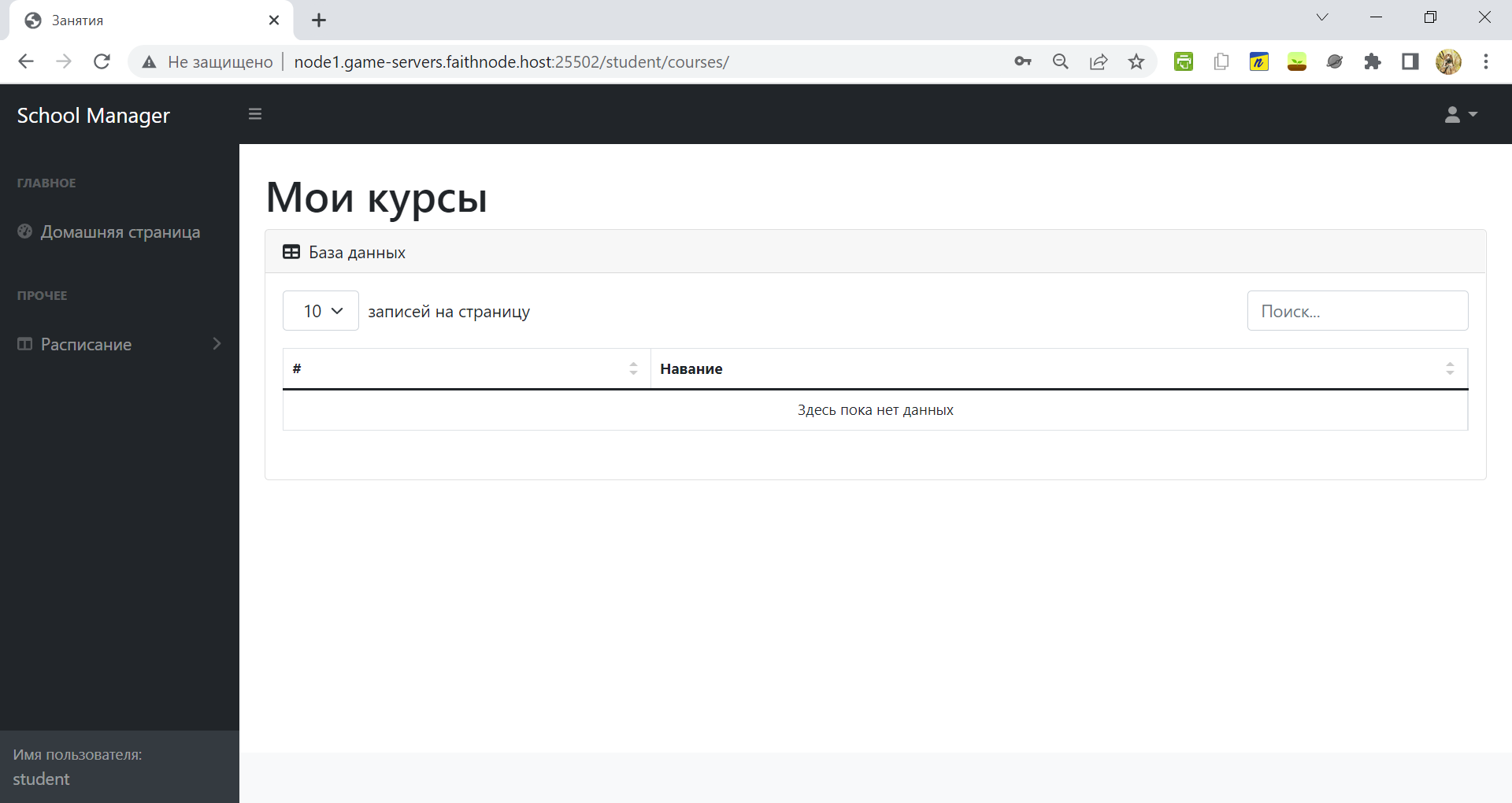


Рисунок 7.12. Список курсов студента

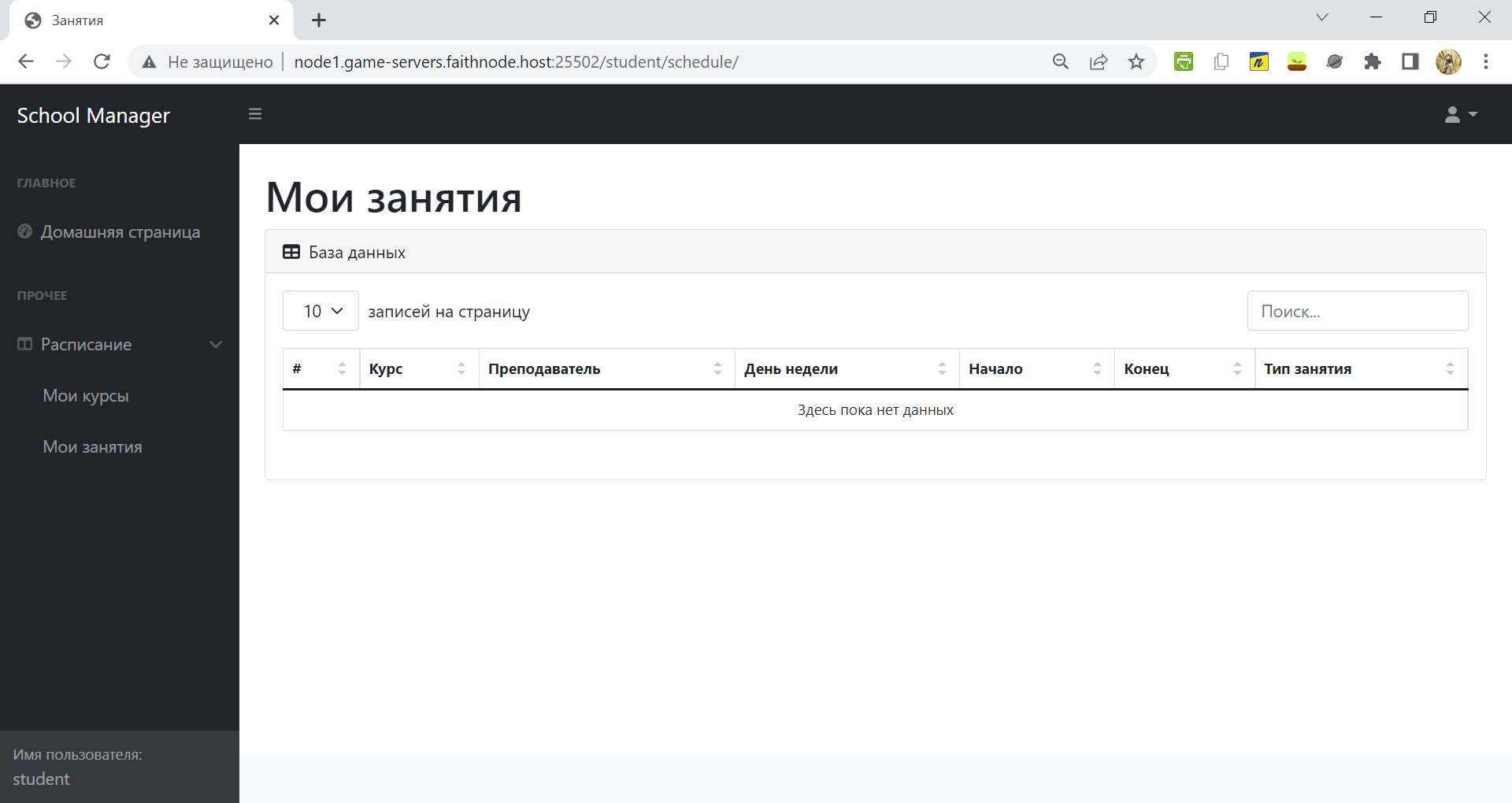


Рисунок 7.13. Список занятий преподавателя

У преподавателя же доступ только с своим занятиям (рис. 7.14).

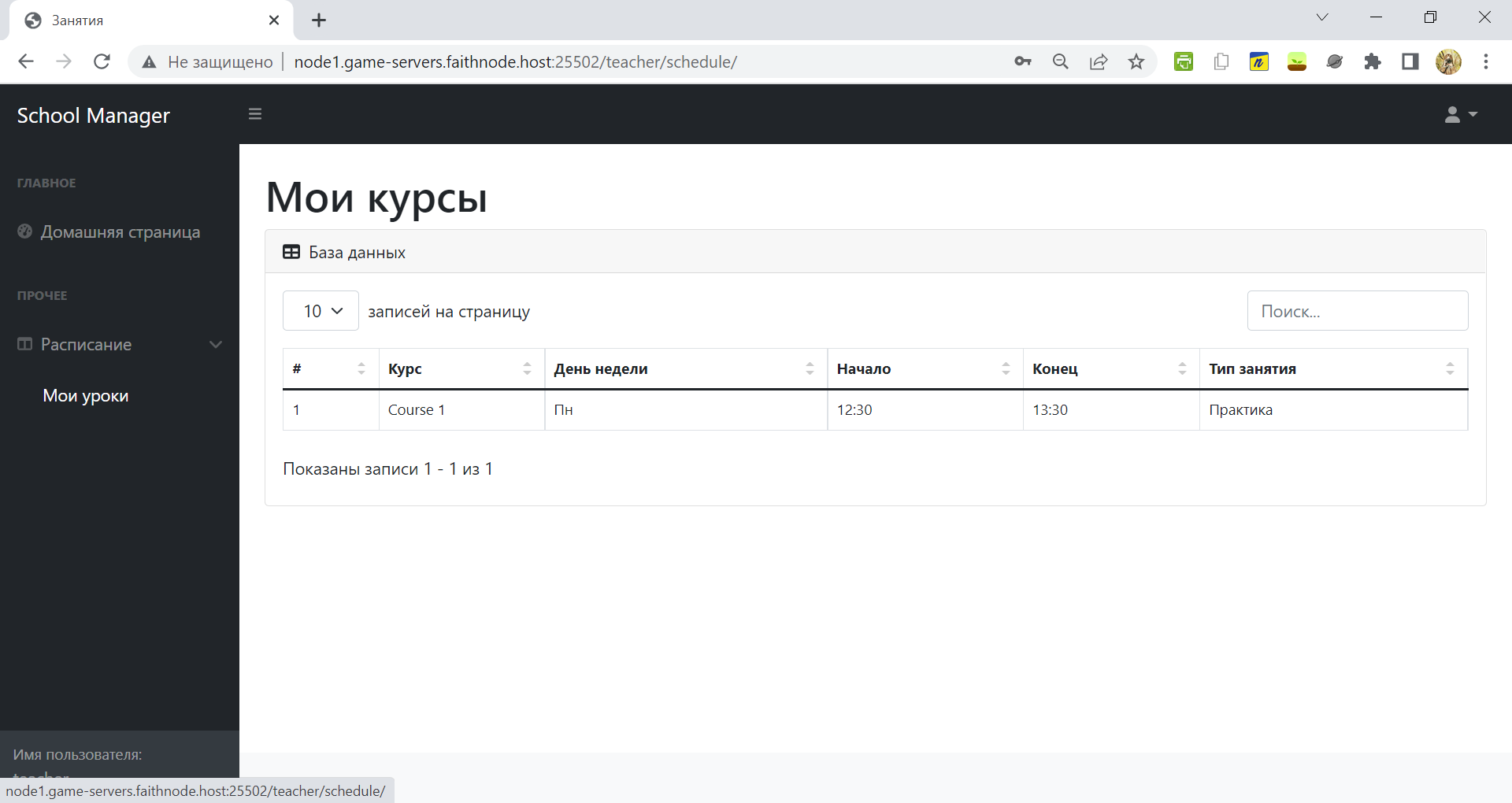


Рисунок 7.14. Список занятий преподавателя

# ЗАКЛЮЧЕНИЕ

В результате выполнения данной курсовой работы была создана программа для работы с базой данных. Были приобретены навыки применения теоретических знаний и практических умений для самостоятельного анализа и формулирования задачи повышения эффективности деятельности организации или предприятия; разработки информационного и программного обеспечения автоматизации деятельности специалистов организации или предприятия; грамотного и последовательного изложения материала проведенных исследований. а также были выработаны новые навыки владения языком программирования Java.

На основе исходных данных выполнены все требования к реализации программы, получен ожидаемый результат.

# СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Карнелл, Д. Микросервисы Spring / Д. Карнелл, И. У. Санчес ; перевод с английского А. Н. Киселева. — Москва : ДМК Пресс, 2022. — 490 с. — ISBN 978-5-97060-971-2. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/241172 (дата обращения: 14.10.2022).
2. Бауэр, К. Java Persistence API и Hibernate / К. Бауэр, Г. Кинг, Г. Грегори ; под редакцией А. Н. Киселева ; перевод с английского Д. А. Зинкевич. — Москва : ДМК Пресс, 2017. — 632 с. — ISBN 978-5-97060-180-8. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/111435 (дата обращения: 14.10.2022).