



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

---

ФАКУЛЬТЕТ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ (ИУ)

КАФЕДРА ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ЭВМ И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ (ИУ7)

# **РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

## ***К КУРСОВОЙ РАБОТЕ***

### ***НА ТЕМУ:***

***Разработка базы данных курсовых проектов по  
дисциплине «Базы данных»***

Студент группы ИУ7-66Б

\_\_\_\_\_

**Виноградов А. О.**

Руководитель курсовой работы

\_\_\_\_\_

**Кузнецов Д. А.**

2023 г.

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

---

УТВЕРЖДАЮ

Заведующий кафедрой ИУ7

\_\_\_\_\_ Рудаков И. В.

«3» марта 2023 г.

**ЗАДАНИЕ**  
**на выполнение курсовой работы**

по дисциплине

**Базы данных**

Студент группы **ИУ7-66Б**

**Виноградов Алексей Олегович**

Тема курсовой работы

**Разработка базы данных курсовых проектов по дисциплине «Базы данных»**

График выполнения работы: 25% к 4 нед., 50% к 8 нед., 75% к 13 нед., 100% к 15 нед.

**Задание**

*Провести анализ предметной области. Сформулировать требования к базе данных и приложению. Сформулировать описание пользователей проектируемого приложения. . Спроектировать архитектуру базы данных и ограничения целостности. Спроектировать ролевую модель на уровне базы данных. . Выбрать средства реализации. Реализовать спроектированную БД и необходимый интерфейс для взаимодействия с ней. Исследовать характеристики разработанного программного обеспечения.*

**Оформление курсовой работы:**

Расчетно-пояснительная записка на 25-40 листах формата А4. Презентация на 12-18 слайдах.

Дата выдачи задания «3» марта 2023 г.

Руководитель курсовой работы

\_\_\_\_\_ Кузнецов Д. А.

Студент

\_\_\_\_\_ Виноградов А. О.

# СОДЕРЖАНИЕ

<b>ВВЕДЕНИЕ</b>	<b>5</b>
<b>1 Аналитический раздел</b>	<b>6</b>
1.1 Определение базы данных . . . . .	6
1.2 Определение СУБД . . . . .	6
1.3 Классификация СУБД по модели данных . . . . .	7
1.4 Классификация СУБД по способу доступа к базе данных . .	8
1.5 Классификация СУБД по организации хранения данных . .	9
1.6 Формализация данных . . . . .	9
1.7 Формализация пользователей . . . . .	10
1.8 Анализ существующих решений . . . . .	10
1.9 Выбор модели данных . . . . .	11
1.10 Вывод . . . . .	11
<b>2 Конструкторский раздел</b>	<b>12</b>
2.1 Формализация сущностей системы . . . . .	12
2.2 Ролевая модель . . . . .	14
2.3 Разработка триггера и функции . . . . .	14
2.4 Вывод . . . . .	15
<b>3 Технологический раздел</b>	<b>16</b>
3.1 Выбор СУБД . . . . .	16
3.2 Средства реализации . . . . .	17
3.3 Создание сущностей базы данных . . . . .	17
3.4 Создание триггера . . . . .	18
3.5 Создание ролевой модели . . . . .	19
3.6 Интерфейс ПО . . . . .	20

<b>4</b>	<b>Исследовательский раздел</b>	<b>22</b>
4.1	Демонстрация работы программы . . . . .	22
4.2	Технические характеристики . . . . .	23
4.3	Время выполнения реализации функции . . . . .	23
	<b>СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ</b>	<b>24</b>

# ВВЕДЕНИЕ

Курсовая работа[?] — практическая работа по всему курсу соответствующей научной дисциплины, выполняемая студентом в качестве формы контроля полученных и усвоенных знаний. Ежегодно десятки курсовых проектов по предмету «Базы данных» создаются студентами МГТУ им. Н.Э. Баумана.

Электронное хранение, обработка и анализ информации о курсовых проектах могут быть полезными для различных систем, например автоматизированной проверки на плагиат для выявления студентов, недобросовестно выполняющих практическую работу.

Целью данной работы является разработка программного обеспечения для хранения, редактирования и удаления данных о курсовых проектах по предмету «Базы данных».

Для достижения поставленной цели необходимо выполнить следующие задачи:

- 1) проанализировать существующие методы представления данных и определить подходящий для выполнения работы вариант;
- 2) провести анализ существующих систем управления базами данных и определить подходящий для выполнения работы вариант;
- 3) спроектировать базу данных, описать ее структуру сущностей и связей между ними;
- 4) реализовать описанную базу данных и программное обеспечение для работы с ней.

# 1 Аналитический раздел

В данном разделе представлен анализ существующих систем управления базами данных, проведена формализация информации, подлежащей хранению.

## 1.1 Определение базы данных

**База данных**[1] — это упорядоченный набор структурированных данных, хранящихся в электронном виде.

По способу применения базы данных разделяют на два типа.

- 1) **OLAP**(online analytical processing) — система, используемая для обработки больших объемов данных.
- 2) **OLTP**(online transactional processing) — система, основной упор которой делается на быструю обработку запросов, операции в режиме реального времени.

В связи с необходимостью работать с базой данных в режиме реального времени в разрабатываемом программной обеспечении будет использована технология OLTP.

## 1.2 Определение СУБД

**Система управления базами данных (СУБД)** — это приложение, обеспечивающее создание, хранение, обновление и поиск информации в базе данных.

Существует множество классификация СУБД, использующие разные признаки классификации. Далее приведены некоторые из них.

## 1.3 Классификация СУБД по модели данных

### Дореляционные СУБД

- 1) **Инвертированные списки.** Базы данных, основанные на использовании инвертированных списков представляют собой совокупность файлов, содержащих записи. Инвертированный список — это организованный в вид списка специального вида индекс файла, позволяющий получить всю совокупность указателей на записи, которым соответствует заданное значение ключа индексации. В общем случае ограничения целостности базы данных отсутствуют. В некоторых системах поддерживаются ограничения уникальности значений некоторых полей, но в основном все возлагается на прикладную программу.
- 2) **Иерархическая модель.** Иерархические модели имеют древовидную структуру, где каждому узлу соответствует один сегмент, представляющий собой запись (кортеж полей) базы данных. Каждому сегменту может соответствовать несколько дочерних сегментов, однако запись-потомок должна иметь в точности одного предка.
- 3) **Сетевые СУБД.** Сетевой подход к организации данных является расширением иерархического. На формирование связи ограничений не накладывается, в отличие от иерархической модели, предполагающей ровно одного предка у записи-потомка.

### Реляционные СУБД

Реляционная модель предполагает организацию данных в виде таблиц (отношений), содержащих информацию о сущностях. Каждая запись таблицы содержит уникальный индекс (ключ), используемый для поиска связанных данных в разных таблицах.

Реляционная модель состоит из трех частей:

- 1) **структурная часть** фиксирует, что база данных использует только одну структуру данных —  $n$ -арное отношение;

- 2) **целостная часть** описывает ограничения, накладываемые на отношения реляционной модели;
- 3) **манипуляционная часть** описывает два эквивалентных способа манипулирования реляционными данными — реляционную алгебру и реляционное исчисление.

## Постреляционные СУБД

Постреляционная модель представляет собой обобщение реляционной модели. Она допускает многозначные поля таблиц, каждое из которых рассматривается как самостоятельное отношение, встроенное в главное отношение.

### 1.4 Классификация СУБД по способу доступа к базе данных

- 1) **Файл-серверные СУБД.** Значительная часть вычислений выполняется на стороне клиента, сервер отвечает только за извлечение данных из файлов и отправку пользователю.
- 2) **Клиент-серверные СУБД.** Основная вычислительная нагрузка ложится на сервер базы данных, задача клиента заключается в предварительной обработке данных и организации доступа к серверу.
- 3) **Встраиваемые СУБД.** СУБД представляет собой библиотеку, позволяющую структурировать и хранить большие объемы данных на локальной машине.
- 4) **Сервис-ориентированные СУБД.** База данных представляет собой хранилище сообщений и метаданных о сервисах и очередях сообщений.
- 5) **Другие СУБД.**



## 1.5 Классификация СУБД по организации хранения данных

- 1) **Локальные СУБД.** Все части СУБД располагаются на одной машине
- 2) **Распределенные СУБД.** Части СУБД распределены на двух или более машинах.

## 1.6 Формализация данных

База данных состоит из следующих сущностей:

- 1) таблица курсовых проектов Project;
- 2) таблица литературных источников Source;
- 3) таблица академических групп Group;
- 4) таблица студентов Student;
- 5) таблица тем курсовых проектов Theme;

На рисунке 1 представлена ER-диаграмма сущностей в нотации Чена.

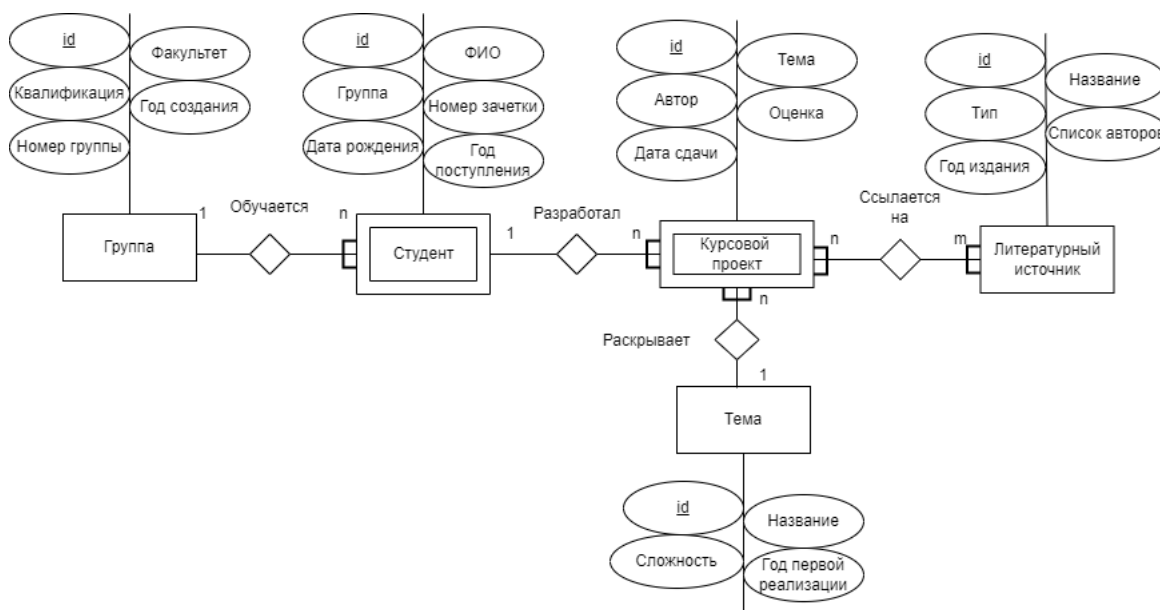


Рисунок 1 – ER-диаграмма в нотации Чена

## 1.7 Формализация пользователей

В системе присутствуют три уровня пользователей.

- 1) **Студент** — пользователь, обладающий возможностями просмотра сущностей академических групп и тем курсовых проектов.
- 2) **Преподаватель** — пользователь, обладающий возможностями просмотра всех сущностей, перечисленных в разделе «Формализация данных».
- 3) **Администратор** — пользователь, обладающий возможностями изменения сущностей и полей базы данных, просмотра всех сущностей.

## 1.8 Анализ существующих решений

Среди уже существующих проектов были выделены 3 аналога, частично решающие поставленную задачу. Их сравнительный анализ представлен в таблице 1.

Для сравнения были выбраны следующие критерии:

- 1) группы — возможность просмотра информации об академических группах студентов;
- 2) антиплагиат — невозможность приобретения готового проекта другого человека;
- 3) темы — возможность просмотра тем уже существующих работ для поиска вдохновения.

Таблица 1 – Существующие решения поставленной задачи

Название проекта	Группы	Антиплагиат	Темы
ЭУ МГТУ[?]	+	+	-
Studynote [?]	-	+	+
Workspay [?]	-	-	+

Из таблицы можно сделать вывод, что ни один из перечисленных аналогов не обладает функционалом, удовлетворяющим требованиям или обладает лазейками для недобросовестного выполнения курсовой работы.

Создаваемое программное обеспечение будет предоставлять описанный функционал, являясь некоммерческим продуктом и не предоставляя возможности для плагиата.

## 1.9 Выбор модели данных

Для реализации программного продукта была выбрана реляционная модель данных, так как она обладает рядом преимуществ в рамках проекта:

- 1) позволяет реализовать связи между выделенными сущностями и исключить дублирование за счет использования механизмов первичного и внешнего ключа;
- 2) подразумевает хранение в виде таблиц, понятных конечному пользователю;
- 3) имеет возможность произвольного доступа ко всем элементам сущностей.

## 1.10 Вывод

В данном разделе представлен анализ существующих систем управления базами данных, проведена формализация информации, подлежащей хранению.

## 2 Конструкторский раздел

В данном разделе рассматриваются сущности проектируемой базы данных, спроектированы триггер и функция.

### 2.1 Формализация сущностей системы

На основе ER-диаграммы (рис. 1) спроектированы таблицы базы данных.

#### Таблица Group

Содержит информацию об академических группах и включает следующие поля

- 1) id — идентификатор группы, являющийся первичным ключом;
- 2) group\_num — номер группы (уникальный для потока), целочисленный тип;
- 3) faculty — название факультета, символьный тип;
- 4) qualification — квалификация (бакалавр, специалист и т.д.), символьный тип;
- 5) creation — год создания группы, целочисленный тип.

#### Таблица Student

Содержит информацию о студентах и включает следующие поля:

- 1) id — идентификатор студента, являющийся первичным ключом;

- 2) group — группа студента, внешний ключ;
- 3) FIO — ФИО студента, символьный тип;
- 4) book\_num — номер зачетной книжки студента (уникальный для года поступления), целочисленный тип;
- 5) birth — дата рождения, тип-дата;
- 6) enrollment — год поступления, целочисленный тип.

## Таблица Theme

Содержит информацию о темах курсовых проектов и включает следующие поля:

- 1) id — идентификатор темы, являющийся первичным ключом;
- 2) name — название темы, символьный тип;
- 3) complexity — сложность темы по 10-бальной шкале, целочисленный тип;
- 4) first\_time — год первой реализации темы, целочисленный тип;

## Таблица Source

Содержит информацию о литературных источниках курсовых проектов и включает следующие поля:

- 1) id — идентификатор источника, являющийся первичным ключом;
- 2) name — название источника, символьный тип;
- 3) type — тип источника (учебник, статья и т.д.), символьный тип;
- 4) author — список авторов, символьный тип;
- 5) creation — год создания источника, целочисленный тип;

## Таблица Project

Содержит информацию о курсовых проектах и включает следующие поля:

- 1) id — идентификатор проекта, являющийся первичным ключом;
- 2) theme\_id — тема проекта, внешний ключ;
- 3) author\_id — автор проекта, внешний ключ;
- 4) mark — оценка проекта, целочисленный тип;
- 5) passed — дата сдачи проекта, тип-дата;

## 2.2 Ролевая модель

Для обеспечения работы пользователей с системой управления базами данных, выделена следующая ролевая модель.

### Студент StUser

- 1) SELECT над таблицей Theme.
- 2) SELECT над таблицей Group.

### Преподаватель TeUser

SELECT над таблицами Theme, Group, Student, Source, Project.

### Администратор AdmUser

Все права над Theme, Group, Student, Source, Project.

## 2.3 Разработка триггера и функции

Также в системе представлен INSERT/UPDATE триггер, который при добавлении в базу нового курсового проекта или изменения уже существующего, проверяет, что год его сдачи больше или равен году первой ре-

лизации соответствующей темы. В противном случае триггер меняет год первой реализации данной темы на год сдачи проекта.

## 2.4 Вывод

В данном разделе рассматриваются сущности проектируемой базы данных, спроектированы триггер и функция.

## 3 Технологический раздел

В данном разделе рассмотрены средства реализации программного продукта, приведены листинги кода реализованных функций и сценариев, а также пользовательский интерфейс

### 3.1 Выбор СУБД

В качестве наиболее популярных реляционных СУБД чаще всего выделяют SQLite [?], MySQL [?] и PostgreSQL [?]. Рассмотрим их сильные и слабые стороны.

- 1) SQLite. Данная СУБД является файловой, то есть все данные хранятся в одном файле, что облегчает перемещение и использование в небольших приложениях. Но в связи с этим данная СУБД не оптимизирована для работы с многопользовательскими приложениями и большими объемами данных.
- 2) MySQL. Среди преимуществ данной СУБД часто выделяют ее простоту, масштабируемость и сравнительно высокую скорость работы. К минусам данной СУБД относят неполную SQL-совместимость, так как MySQL реализует не весь функционал SQL, и сравнительно невысокую оптимизацию процессов параллельного чтения.
- 3) PostgreSQL. В качестве главных преимуществ данной СУБД приводят полную SQL-совместимость, расширяемость и поддержку многими сторонними инструментами, связанными с СУБД. Среди недостатков выделяют операции чтения, в которых PostgreSQL может проигрывать конкурентам.

Для реализации программного продукта была выбрана СУБД PostgreSQL, так как она обладает всем необходимым для проекта функционалом.



## 3.2 Средства реализации

Для реализации программного продукта был выбран язык программирования Python [?]. Данный язык программирования предоставляет весь необходимый для реализации проекта функционал.

## 3.3 Создание сущностей базы данных

Создание сущностей в соответствии с результатами проектирования базы данных представлены на листинге 1.

Листинг 1 – Создание сущностей базы данных

```
1 class Group(BASE):
2     __tablename__ = 'grouptab'
3     id = Column(Integer, Identity(always=True), primary_key=True)
4     group_num = Column(Integer)
5     faculty = Column(Text)
6     qualification = Column(Text)
7     creation = Column(Integer)
8
9 class Student(BASE):
10    __tablename__ = 'student'
11    id = Column(Integer, Identity(always=True), primary_key=True)
12    fio = Column(Text)
13    group_id = Column(Integer)
14    book_num = Column(Integer)
15    birth = Column(Date)
16    enrollment = Column(Integer)
17
18 class Theme(BASE):
19    __tablename__ = 'theme'
20    id = Column(Integer, Identity(always=True), primary_key=True)
21    name = Column(Text)
22    complexity = Column(Integer)
23    first_time = Column(Integer)
24
25 class Source(BASE):
26    __tablename__ = 'source'
27    id = Column(Integer, Identity(always=True), primary_key=True)
28    name = Column(Text)
```

```

29     type = Column(Text)
30     authors = Column(Text)
31     creation = Column(Integer)
32
33 class Project(BASE):
34     __tablename__ = 'project'
35     id = Column(Integer, Identity(always=True), primary_key=True)
36     theme_id = Column(Integer, ForeignKey('theme.id'),
37                          primary_key=True)
38     author_id = Column(Integer, ForeignKey('student.id'),
39                          primary_key=True)
40     mark = Column(Integer)
41     passed = Column(Date)
42
43 class SourceProject(BASE):
44     __tablename__ = 'source_project'
45     id = Column(Integer, Identity(always=True), primary_key=True)
46     source_id = Column(Integer, ForeignKey('source.id'),
47                          primary_key=True)
48     project_id = Column(Integer, ForeignKey('project.id'),
49                          primary_key=True)

```

### 3.4 Создание триггера

Создание INSERT/UPDATE триггера над таблицей Project в соответствии с результатами проектирования представлено в листинге 2.

Листинг 2 – Создание INSERT/UPDATE триггера над таблицей Project

```

1 create TRIGGER check_first_date_trigger
2 AFTER INSERT or UPDATE ON "project"
3 FOR EACH ROW
4 EXECUTE FUNCTION check_first_date();

```

Для работы триггера также была написана функция *check\_first\_date*, выполняющей проверку года первой сдачи темы курсовой работы в соответствии с результатами проектирования. Ее код приведен в листинге 3

Листинг 3 – Код функции проверки

```

1 CREATE FUNCTION check_first_date() RETURNS TRIGGER AS $$
2 declare first_t int;

```

```

3 begin
4     select  first_time
5         from theme
6         where id = new.theme_id
7         into first_t;
8     IF first_t > extract('Year' from new.passed) then
9         RAISE NOTICE 'Previous_ first_time:_%', first_t;
10        RAISE NOTICE 'New_ first_time:_%', extract('year' from
            new.passed);
11        RAISE NOTICE 'Updating_table_theme_with_new_value:_%',
            extract('year' from new.passed);
12        update theme set first_time = extract('year' from new.passed)
            where id = new.theme_id;
13        Return NEW;
14    ELSE
15        RETURN NULL;
16    END IF;
17 END;
18 $$ LANGUAGE plpgsql;

```

### 3.5 Создание ролевой модели

В соответствии с разработанной в конструкторском разделе ролевой моделью был написан SQL-сценарий, выполняющий создание ролей в базе данных и выделение им прав. Код сценария представлен в листинге 4.

Листинг 4 – Код сценария создания ролей базы данных и выделения им прав

```

1 CREATE ROLE StUser LOGIN PASSWORD 'postgres';
2 GRANT SELECT ON TABLE "theme", "grouptab" TO StUser;
3
4 CREATE ROLE TeUser LOGIN PASSWORD 'postgres';
5 GRANT SELECT ON TABLE "theme", "grouptab", "student", "source",
    "project", "source_project" TO TeUser;
6
7 CREATE ROLE AdmUser LOGIN PASSWORD 'postgres';
8 GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO AdmUser;

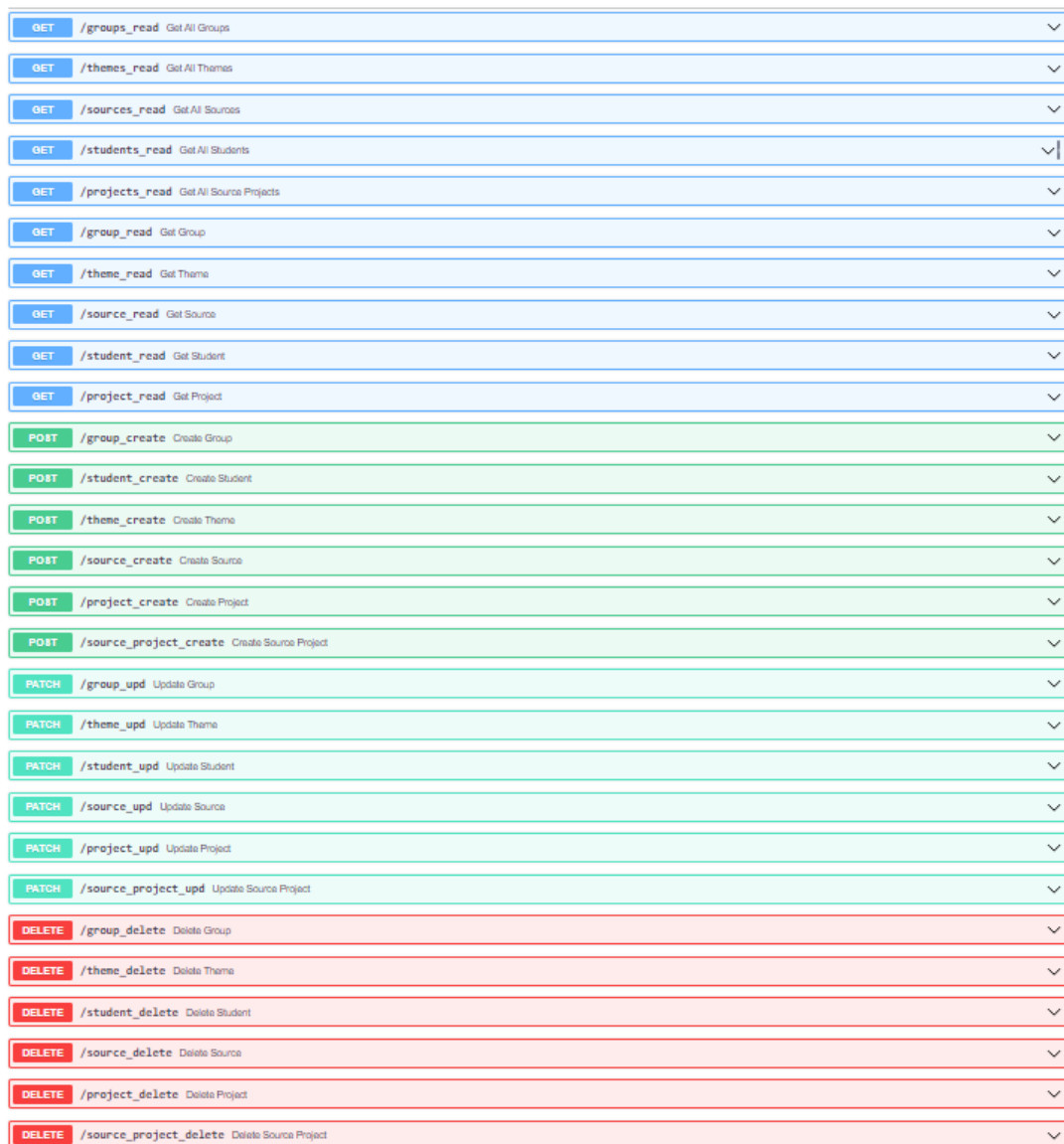
```

## 3.6 Интерфейс ПО

Для работы с базой данных был реализован интерфейс взаимодействия API [?]. Для создания интерфейса была использована библиотека fastapi [?].

Для каждой сущности базы данных были реализованы операции чтения, добавления, обновления и удаления.

Интерфейс приведен на рисунке 2.



GET	/groups_read	Get All Groups	▼
GET	/themes_read	Get All Themes	▼
GET	/sources_read	Get All Sources	▼
GET	/students_read	Get All Students	▼
GET	/projects_read	Get All Source Projects	▼
GET	/group_read	Get Group	▼
GET	/theme_read	Get Theme	▼
GET	/source_read	Get Source	▼
GET	/student_read	Get Student	▼
GET	/project_read	Get Project	▼
POST	/group_create	Create Group	▼
POST	/student_create	Create Student	▼
POST	/theme_create	Create Theme	▼
POST	/source_create	Create Source	▼
POST	/project_create	Create Project	▼
POST	/source_project_create	Create Source Project	▼
PATCH	/group_upd	Update Group	▼
PATCH	/theme_upd	Update Theme	▼
PATCH	/student_upd	Update Student	▼
PATCH	/source_upd	Update Source	▼
PATCH	/project_upd	Update Project	▼
PATCH	/source_project_upd	Update Source Project	▼
DELETE	/group_delete	Delete Group	▼
DELETE	/theme_delete	Delete Theme	▼
DELETE	/student_delete	Delete Student	▼
DELETE	/source_delete	Delete Source	▼
DELETE	/project_delete	Delete Project	▼
DELETE	/source_project_delete	Delete Source Project	▼

Рисунок 2 – Интерфейс взаимодействия с базой данных

## Вывод

В данном разделе были рассмотрены средства реализации программного продукта, приведены листинги кода реализованных функций и сценариев, а также пользовательский интерфейс

## 4 Исследовательский раздел

В данном разделе будут приведены примеры работы разработанной программы и исследована зависимость времени выполнения ...

### 4.1 Демонстрация работы программы

На рисунке 3 продемонстрирован запрос, позволяющий получить информацию из таблицы групп базы данных.

The screenshot shows a REST client interface with the following sections:

- Parameters:** A table with two rows: 'limit' (integer, query) with value 10, and 'skip' (integer, query) with value 0. There are 'Execute' and 'Clear' buttons below.
- Responses:** A section containing:
  - Curl:** A code block with the command: `curl -X 'GET' \ 'http://127.0.0.1:5000/groups_read?limit=10&skip=0' \ -H 'accept: application/json'`
  - Request URL:** A text field containing: `http://127.0.0.1:5000/groups_read?limit=10&skip=0`
  - Server response:** A section with a 'Code' tab showing '200' and a 'Details' tab showing the response body.
- Response body:** A JSON array of three objects, each representing a group with fields: faculty, creation, id, qualification, and group\_num.

Рисунок 3 – Пример работы программы

Для операций чтения и удаления необходимо указать идентификатор

записи в соответствующей таблицы. Для операций обновления и добавления необходимо указать значения полей, соответствующих атрибутам целевой таблицы.

Для операций массового чтения также необходимо указать количество элементов на возвращаемой странице и количество страниц, которые необходимо пропустить при выводе.

## 4.2 Технические характеристики

Ниже приведены технические характеристики устройства, на котором было проведено измерение времени работы ПО:

- операционная система Windows 10 Домашняя Версия 21H1 [?] x86\_64;
- оперативная память 8 Гбайт 2133 МГц;
- процессор Intel Core i5-8300H с тактовой частотой 2.30 ГГц [?], 4 физических ядра, 8 логических ядер.

## 4.3 Время выполнения реализации функции

### Вывод

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- [1] DATABASE SYSTEMS The Complete Book Second Edition Hector Garcia-Molina Jeffrey D. Ullman.