

**МИНОБРНАУКИ РОССИИ  
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ  
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ «ЛЭТИ»  
ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**КУРСОВАЯ РАБОТА**

**по дисциплине «Управление данными»**

**Тема: Разработка базы данных для автоматизации  
деятельности приемной комиссии**

Студентка гр. 2374  
Преподаватель

\_\_\_\_\_  
\_\_\_\_\_

Воронина К.Д.  
Татарникова Т.М.

Санкт-Петербург  
2024

## **ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ**

Студент: Воронина К.Д.

Группа: 2374

Тема работы: Разработка базы данных для автоматизации деятельности приемной комиссии

Исходные данные: Вариант 5. Спроектировать базу данных, построить программу, обеспечивающую взаимодействие с ней в режиме диалога для работников приемной комиссии. В БД должны храниться сведения об абитуриентах, датах экзаменов и консультаций, номерах аудиторий.

Содержание пояснительной записки: содержание, введение, анализ предметной области, обоснование модели данных, обоснование выбора СУБД, описание функций групп пользователей, описание функций управления БД, организация защиты БД, заключение, список использованных источников.

Предполагаемый объем пояснительной записки: Не менее 30 страниц.

Дата выдачи задания: 07.09.2024

Дата сдачи работы: 23.12.2024

Дата защиты работы:

Студент

Воронина К.Д.

Преподаватель

Татарникова Т.М.

## **АННОТАЦИЯ**

В рамках курсовой работы реализуется реляционная база данных для автоматизации деятельности приемной комиссии. На практике создается логическая схема базы данных, интерфейс пользователя для взаимодействия с базой данных, в виде клиентского приложения.

## **SUMMARY**

As part of the course work, a relational database is implemented to automate the activities of the selection committee. In practice, a logical database schema is created, a user interface for interacting with the database, in the form of a client application.

## **ВВЕДЕНИЕ**

Работа приемной комиссии – сложный процесс, в ходе которого используется большое количество данных таких, как данные об абитуриенте, его экзаменах и отметках, данные об аудиториях, в которых будут приниматься экзамены, и данные о датах экзаменов по предметам.

Стоит отметить, что хранить всю необходимую информацию с помощью отдельных носителей является неоптимальным решением из-за того, что будет отсутствовать взаимосвязь между разными данными. Таким образом возникает потребность в создании базы данных для удобного конфигурирования информацией.

В данной работе будет разработана система автоматизации учета данных о работе приемной комиссии. В рамках данной задачи необходимо будет решить следующие основные задачи:

- хранение данных об абитуриентах, экзаменационных аудиториях, датах экзаменов по разным предметам, оценках абитуриентов;
- управлять имеющимися данными: добавлять и удалять записи о абитуриентах;
- получать требуемую информацию об оценках, полученных студентами; аудиториях, где будут проходить экзамены; о датах проведения экзаменов по определенному предмету.

При проектировании БД будут проведены следующие этапы:

- Системный анализ предметной области;
- Проектирование инфологической модели;
- Логическое проектирование БД;
- Физическое проектирование БД.

В ходе данной работы будут пройдены все перечисленные этапы и реализована требуемая база данных с соответствующим клиентским приложением.

## Оглавление

АННОТАЦИЯ .....	3
SUMMARY.....	3
ВВЕДЕНИЕ .....	4
1. СИСТЕМНЫЙ АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ .....	6
2. МОДЕЛЬ ДАННЫХ .....	7
3. ОБОСНОВАНИЕ ВЫБОРА СУБД .....	9
4. КОД ПРИЛОЖЕНИЯ.....	10
5. РАБОТА В ПРИЛОЖЕНИИ.....	23
ЗАКЛЮЧЕНИЕ.....	35
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ.....	36

# **1. СИСТЕМНЫЙ АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ**

## **1.1. Выбор подхода к проектированию БД**

В первую очередь нужно выбрать, какой из двух подходов к проектированию базы данных в большей степени подходит для данного случая: функциональный или предметный.

На практике чаще всего реализуется некоторый компромиссный вариант, однако, так как функции некоторой группы лиц и комплексов решаемых задач уже оговорены в задании к курсовой работе, возьмем за основу функциональный подход.

## **1.2. Выделение и фиксирование бизнес-требований к функциональности системы**

Перед началом разработки базы данных следует выделить из поставленных условий бизнес-требования к информационной системе. Необходимо закрепить набор требований к функциональности системы с точки зрения поддержки необходимых процессов.

Проанализировав предметную область, можно выделить следующие группы субъектов – потенциальных пользователей системы:

- Пользователь (сотрудник приемной комиссии)
- Администратор

Разрабатываемая система предназначена для поддержки деятельности выбранных пользователей. База данных должна удовлетворять их потребности, исходя из этого можно выделить следующие требования к проектируемой системе.

Администратор БД может вносить следующие изменения:

- ☐ ввести информацию о новом абитуриенте;
- ☐ изменить оценку абитуриенту;
- ☐ удалить запись об абитуриенте.

Пользователь не может ничего изменять в базе данных, но может получить данные:

- ☐ список абитуриентов на заданный факультет;
- ☐ полученные оценки для абитуриента;
- ☐ дата консультации и экзамена для абитуриента по данному предмету;
- ☐ номера аудиторий, где будут экзамены у заданной группы;
- ☐ список групп, которые будут заниматься в заданной аудитории в заданное время.

## 2. МОДЕЛЬ ДАННЫХ

Проведя анализ предметной области, получаем следующую концептуальную ER-модель (см. рисунок 1).

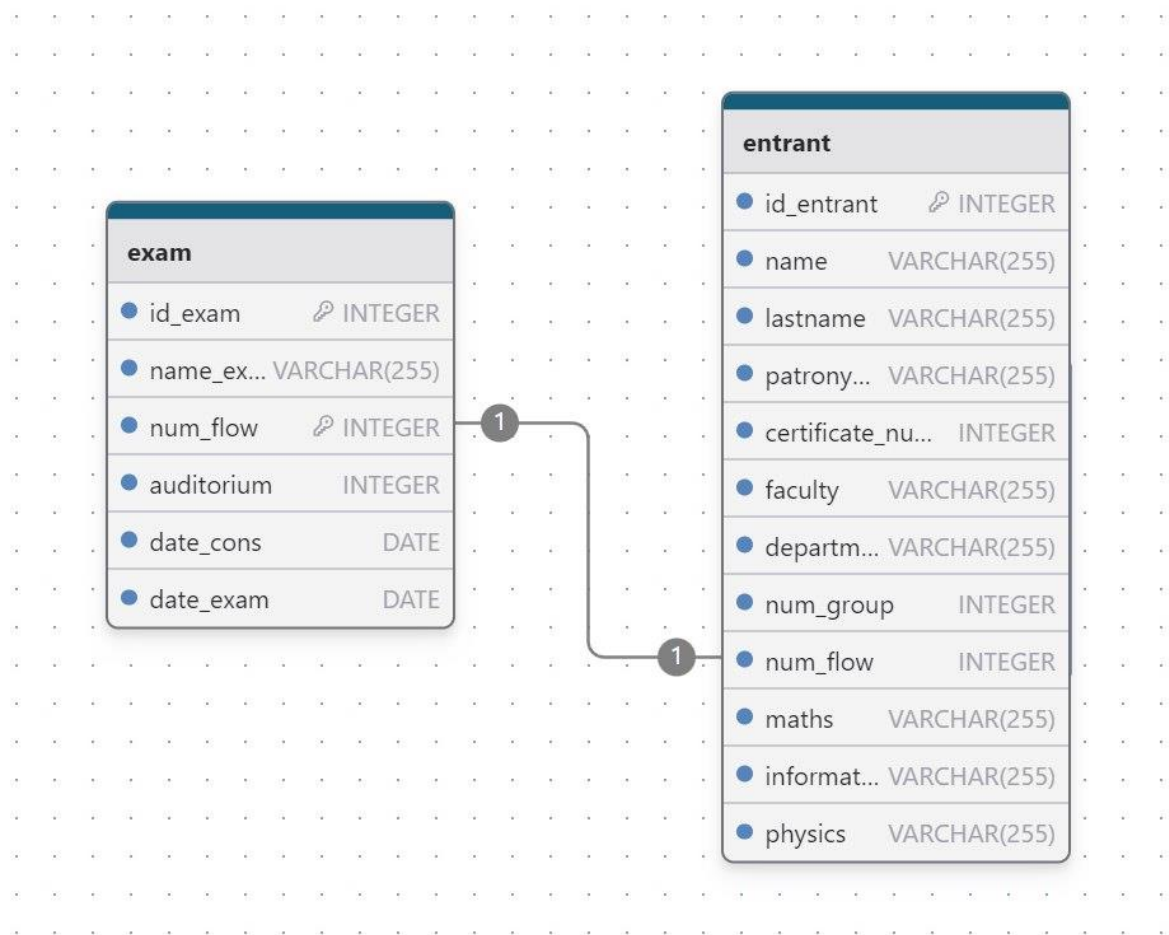


Рис. 1 – ER-модель













Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
 id_entrant	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
 name	VARCHAR(30)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 lastname	VARCHAR(30)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 patronymic	VARCHAR(30)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 certificate_number	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 faculty	VARCHAR(100)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 department	VARCHAR(100)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 num_group	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 num_flow	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 maths	INT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
 informatics	INT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
 physics	INT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Рис. 2 – Столбцы в базе данных “entrant”

- id\_entrant – идентификатор абитуриента
- name – имя абитуриента
- lastname – фамилия абитуриента
- patronymic – отчество абитуриента
- certificate\_number – номер экзаменационного листа абитуриента
- faculty – факультет
- department – кафедра
- num\_group – номер группы
- num\_flow – номер потока
- maths – оценка за экзамен по математике
- informatics – оценка за экзамен по информатике
- physics – оценка за экзамен по физике







Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
 id_exam	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
 name_exam	VARCHAR(30)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 num_flow	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 auditorium	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 date_cons	DATE	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 date_exam	DATE	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Рис. 3 – Столбцы в базе данных “exam”



- id\_exam – идентификатор экзамена
- name\_exam – название экзамена (предмет)
- num\_flow – номер потока
- auditorium – аудитория для экзамена
- date\_cons – дата консультации по экзамену
- date\_exam – дата экзамена

Для корректной работы всех функций приложения, изначально в базе данных уже будет храниться некоторая информация:

	id_entrant	name	lastname	patronymic	certificate_number	faculty	department	num_group	num_flow	maths	informatics	physics
►	1	Анна	Рассказова	Николаевна	121212	ФИБС	БТС	1	1	5	4	5
	2	Ксения	Воронина	Дмитриевна	232323	ФКТИ	ИС	2	2	5	5	3
	3	Елизаб...	Заярная	Викторовна	343434	ФЭА	ЭА	4	1	4	4	3
	4	Семен	Чернорезкий	Алексеевич	676767	ФРТ	РТ	3	2	3	3	3
	5	Артем	Иванов	Иванович	585858	ФРТ	РТ	3	2	3	4	3
	6	Ренат	Хисматулин	Амирович	969696	ФКТИ	САПР	2	2	4	4	4

Рис. 4 – База данных “entrant”

	id_exam	name_exam	num_flow	auditorium	date_cons	date_exam
►	1	Математика	1	1234	2025-01-12	2025-01-14
	2	Математика	2	4321	2025-01-15	2025-01-17
	3	Информатика	1	1122	2025-01-09	2025-01-10
	4	Информатика	2	3344	2025-01-11	2025-01-13
	5	Физика	1	2323	2025-01-19	2025-01-20
	6	Физика	2	3434	2025-01-22	2025-01-23

Рис. 5 – База данных “exam”

### 3. ОБОСНОВАНИЕ ВЫБОРА СУБД

В качестве СУБД выберем MySQL. MySQL — объектно- реляционная СУБД. Это значит, что она поддерживает и объектный, и реляционный подход. Другими преимуществами выбранной СУБД являются: работа с большими объемами, поддержка сложных запросов, поддержка множества типов данных, высокая мощность и широкая функциональность, кроссплатформенность, а также открытость (ПО имеет открытый исходный код).

Для написания клиентского приложения будет использоваться язык программирования Python.

## 4. КОД ПРИЛОЖЕНИЯ

```
from tkinter import *
from tkinter.ttk import *
import mysql.connector
from threading import Timer

# подключение базы данных
db = mysql.connector.connect(
    host="localhost",
    port="3306",
    user="root",
    password="ksu12345!",
    database="dm"
)

window = Tk()
window.geometry('800x500')
window['background'] = "lightblue"
window.title("Приемная комиссия")

# создание объекта стилей
style = Style()

style.configure('TButton',
                font = ('calibri', 15, 'bold'),
                foreground = 'darkblue')

##### Добавление абитуриента
#####

def add_entr():
    global window
    window.destroy()
    window=Tk()
    window.geometry('800x500')
    window['background'] = "lightblue"
    window.title("Добавление абитуриента в базу данных")

    style = Style()

    style.configure('TButton',
                    font = ('calibri', 14, 'bold'),
                    foreground = 'darkblue')

    lbl_name = Label(window, text="Введите имя абитуриента")
    lbl_name.place(relx=0.5, rely=0.05, anchor='center', width=400, height=20)
    txt_name = Entry(window,width=20)
    txt_name.place(relx=0.5, rely=0.10, anchor='center', width=400, height=20)

    lbl_lname = Label(window, text="Введите фамилию абитуриента")
    lbl_lname.place(relx=0.5, rely=0.20, anchor='center', width=400,
height=20)
    txt_lname = Entry(window,width=20)
    txt_lname.place(relx=0.5, rely=0.25, anchor='center', width=400,
height=20)

    lbl_patr = Label(window, text="Введите отчество абитуриента")
    lbl_patr.place(relx=0.5, rely=0.35, anchor='center', width=400, height=20)
    txt_patr = Entry(window,width=20)
    txt_patr.place(relx=0.5, rely=0.40, anchor='center', width=400, height=20)
```

```

lbl_num_ex = Label(window, text="Введите номер экзаменационного листа")
lbl_num_ex.place(relx=0.5, rely=0.50, anchor='center', width=400,
height=20)
txt_num_ex = Entry(window,width=20)
txt_num_ex.place(relx=0.5, rely=0.55, anchor='center', width=400,
height=20)

lbl_fac = Label(window, text="Введите факультет абитуриента")
lbl_fac.place(relx=0.5, rely=0.65, anchor='center', width=400, height=20)
txt_fac = Entry(window,width=20)
txt_fac.place(relx=0.5, rely=0.70, anchor='center', width=400, height=20)

lbl_dep = Label(window, text="Введите кафедру абитуриента")
lbl_dep.place(relx=0.5, rely=0.80, anchor='center', width=400, height=20)
txt_dep = Entry(window,width=20)
txt_dep.place(relx=0.5, rely=0.85, anchor='center', width=400, height=20)

def save_to_db():
    a = txt_name.get()
    b = txt_lname.get()
    c = txt_patr.get()
    d = txt_num_ex.get()
    e = txt_fac.get()
    f = txt_dep.get()

    mycursor = db.cursor()

    sql = """INSERT INTO entrant (name, lastname, patronymic,
certificate_number, faculty, department, num_group, num_flow)
VALUES (%s, %s, %s, %s, %s, %s, %s, %s)"""
    val = (a, b, c, d, e, f, 1, 1)

    mycursor.execute(sql, val)

    db.commit()

def close_window():
    window.destroy()

window = Tk()
window.title("Уведомление")
window.geometry("300x100")

label = Label(window, text="Данные успешно добавлены")
label.pack(pady=20)

timer = Timer(3.0, close_window)
timer.start()

clicked_adm()

window.mainloop()

btn_add = Button(window, text="Добавить", command=save_to_db)
btn_add.place(relx=0.5, rely=0.95, anchor='center', width=200, height=40)

##### удаление абитуриента
#####

def del_entr():
    global window
    window.destroy()
    window=Tk()

```

```

window.geometry('800x500')
window['background'] = "lightblue"
window.title("Удаление абитуриента из базы данных")

style = Style()

style.configure('TButton',
                font = ('calibri', 14, 'bold'),
                foreground = 'darkblue')

lbl_name = Label(window, text="Введите имя абитуриента")
lbl_name.place(relx=0.5, rely=0.2, anchor='center', width=400, height=20)
txt_name = Entry(window, width=20)
txt_name.place(relx=0.5, rely=0.25, anchor='center', width=400, height=20)
lbl_lname = Label(window, text="Введите фамилию абитуриента")
lbl_lname.place(relx=0.5, rely=0.35, anchor='center', width=400,
height=20)
txt_lname = Entry(window, width=20)
txt_lname.place(relx=0.5, rely=0.4, anchor='center', width=400, height=20)
lbl_patr = Label(window, text="Введите отчество абитуриента")
lbl_patr.place(relx=0.5, rely=0.50, anchor='center', width=400, height=20)
txt_patr = Entry(window, width=20)
txt_patr.place(relx=0.5, rely=0.55, anchor='center', width=400, height=20)

def del_from_db():
    a = txt_name.get()
    b = txt_lname.get()
    c = txt_patr.get()

    mycursor = db.cursor()

    sql = "DELETE FROM entrant WHERE name = %s and lastname = %s and
patronymic = %s"
    val = (a, b, c)

    mycursor.execute(sql, val)

    db.commit()

def close_window():
    window.destroy()

window = Tk()
window.title("Уведомление")
window.geometry("300x100")

label = Label(window, text="Данные успешно удалены")
label.pack(pady=20)

timer = Timer(3.0, close_window)
timer.start()

clicked_adm()

window.mainloop()

btn_del = Button(window, text="Удалить", command=del_from_db)
btn_del.place(relx=0.5, rely=0.7, anchor='center', width=200, height=40)

##### Изменение оценки
#####

def change_mark():
    global window

```

```

window.destroy()
window=Tk()
window.geometry('800x500')
window['background'] = "lightblue"
window.title("Изменение оценки по предмету у абитуриента")

style = Style()

style.configure('TButton',
                font = ('calibri', 14, 'bold'),
                foreground = 'darkblue')

lbl_name = Label(window, text="Введите имя абитуриента")
lbl_name.place(relx=0.5, rely=0.1, anchor='center', width=400, height=20)
txt_name = Entry(window,width=20)
txt_name.place(relx=0.5, rely=0.15, anchor='center', width=400, height=20)
lbl_lname = Label(window, text="Введите фамилию абитуриента")
lbl_lname.place(relx=0.5, rely=0.25, anchor='center', width=400,
height=20)
txt_lname = Entry(window,width=20)
txt_lname.place(relx=0.5, rely=0.3, anchor='center', width=400, height=20)
lbl_patr = Label(window, text="Введите отчество абитуриента")
lbl_patr.place(relx=0.5, rely=0.4, anchor='center', width=400, height=20)
txt_patr = Entry(window,width=20)
txt_patr.place(relx=0.5, rely=0.45, anchor='center', width=400, height=20)
lbl_sub = Label(window, text="Выберете предмет, по которому нужно изменить
оценку у абитуриента")
lbl_sub.place(relx=0.5, rely=0.55, anchor='center', width=400, height=20)
combo_sub = Combobox(window)
combo_sub['values'] = ("Математика", "Информатика", "Физика")
combo_sub.place(relx=0.5, rely=0.6, anchor='center', width=400, height=20)
lbl_mark = Label(window, text="Выберете оценку, которую получил
абитуриент")
lbl_mark.place(relx=0.5, rely=0.7, anchor='center', width=400, height=20)
combo_mark = Combobox(window)
combo_mark['values'] = ("2", "3", "4", "5")
combo_mark.place(relx=0.5, rely=0.75, anchor='center', width=400,
height=20)

def change_db():
    a = txt_name.get()
    b = txt_lname.get()
    c = txt_patr.get()
    d = combo_sub.get()
    e = combo_mark.get()

    mycursor = db.cursor()

    if d == "Математика":
        d = "maths"
    elif d == "Информатика":
        d = "informatics"
    elif d == "Физика":
        d = "physics"

    sql = f"UPDATE entrant SET {d} = %s WHERE name = %s and lastname = %s
and patronymic = %s"
    val = (e, a, b, c)

    mycursor.execute(sql, val)

    db.commit()

def close_window():
    window.destroy()

```

```

window = Tk()
window.title("Уведомление")
window.geometry("300x100")

label = Label(window, text="Данные успешно изменены")
label.pack(pady=20)

timer = Timer(3.0, close_window)
timer.start()

clicked_adm()

window.mainloop()

btn_change = Button(window, text="Изменить оценку", command=change_db)
btn_change.place(relx=0.5, rely=0.9, anchor='center', width=200,
height=40)

##### Возвращение в главное меню
#####

def menu():
    global window
    window.destroy()
    window=Tk()
    window.geometry('800x500')
    window['background'] = "lightblue"

    style = Style()

    style.configure('TButton',
                    font = ('calibri', 14, 'bold'),
                    foreground = 'darkblue')

    # кнопка для входа администратора
    btn_adm = Button(window, text="Войти как администратор",
command=clicked_adm)
    btn_adm.place(relx=0.5, rely=0.4, anchor='center', width=400, height=40)

    # кнопка для входа сотрудника
    btn_user = Button(window, text="Войти как сотрудник приемной комиссии",
command=clicked_user)
    btn_user.place(relx=0.5, rely=0.5, anchor='center', width=400, height=40)

##### Администратор
#####

def clicked_adm():
    global window
    window.destroy()
    window=Tk()
    window.geometry('800x500')
    window['background'] = "lightblue"
    window.title("Администратор")

    style = Style()

    style.configure('TButton',
                    font = ('calibri', 14, 'bold'),
                    foreground = 'darkblue')

    # кнопка добавления абитуриента

```

```

    btn_add_entr = Button(window, text="Добавить нового абитуриента",
command=add_entr)
    btn_add_entr.place(relx=0.5, rely=0.3, anchor='center', width=400,
height=40)

    # кнопка удаления абитуриента
    btn_del_entr = Button(window, text="Удалить нового абитуриента",
command=del_entr)
    btn_del_entr.place(relx=0.5, rely=0.4, anchor='center', width=400,
height=40)

    # кнопка для изменения оценки абитуриента
    btn_change_mark = Button(window, text="Изменение оценки абитуриента по
предмету", command=change_mark)
    btn_change_mark.place(relx=0.5, rely=0.5, anchor='center', width=400,
height=40)

    # кнопка возврата в главное меню
    btn_menu = Button(window, text="Назад", command=menu)
    btn_menu.place(relx=0.5, rely=0.7, anchor='center', width=200, height=40)

##### Вывод списка абитуриентов #####

def list_entr():
    global window
    window.destroy()
    window=Tk()
    window.geometry('800x500')
    window['background'] = "lightblue"
    window.title("Список абитуриентов")

    style = Style()

    style.configure('TButton',
                    font = ('calibri', 14, 'bold'),
                    foreground = 'darkblue')

    lbl_fac = Label(window, text="Введите нужный факультет")
    lbl_fac.place(relx=0.5, rely=0.35, anchor='center', width=400, height=20)
    txt_fac = Entry(window,width=20)
    txt_fac.place(relx=0.5, rely=0.4, anchor='center', width=400, height=20)

    def print_list_fac():
        a = txt_fac.get()

        mycursor = db.cursor()

        sql = """SELECT name, lastname, patronymic, department
FROM entrant WHERE faculty = %s"""
        val = (a,)

        mycursor.execute(sql, val)

        columns = [desc[0] for desc in mycursor.description]
        rows = mycursor.fetchall()

        db.commit()

        new_window = Tk()
        new_window.title("Список абитуриентов")
        new_window.geometry("800x500")

        tree = Treeview(new_window)
        tree.pack(side=LEFT, fill=BOTH, expand=True)

```



```

tree["columns"] = columns
tree["show"] = "headings"

for col in columns:
    tree.column(col, anchor='center', width=len(col)*15)
    tree.heading(col, text=col, anchor='center')

for row in rows:
    tree.insert("", "end", values=row)

scrollbar = Scrollbar(new_window, orient="horizontal",
command=tree.xview)
tree.configure(xscrollcommand=scrollbar.set)

new_window.mainloop()

btn_list = Button(window, text="Назад", command=clicked_user)
btn_list.place(relx=0.35, rely=0.6, anchor='center', width=200, height=40)
btn_list = Button(window, text="Вывести список", command=print_list_fac)
btn_list.place(relx=0.65, rely=0.6, anchor='center', width=200, height=40)

##### Вывод оценки абитуриента
#####

def mark_entr():
    global window
    window.destroy()
    window=Tk()
    window.geometry('800x500')
    window['background'] = "lightblue"
    window.title("Оценки абитуриента")

    style = Style()

    style.configure('TButton',
                    font = ('calibri', 14, 'bold'),
                    foreground = 'darkblue')

    lbl_name = Label(window, text="Введите имя абитуриента")
    lbl_name.place(relx=0.5, rely=0.2, anchor='center', width=400, height=20)
    txt_name = Entry(window,width=20)
    txt_name.place(relx=0.5, rely=0.25, anchor='center', width=400, height=20)

    lbl_lname = Label(window, text="Введите фамилию абитуриента")
    lbl_lname.place(relx=0.5, rely=0.35, anchor='center', width=400,
height=20)
    txt_lname = Entry(window,width=20)
    txt_lname.place(relx=0.5, rely=0.4, anchor='center', width=400, height=20)

    lbl_patr = Label(window, text="Введите отчество абитуриента")
    lbl_patr.place(relx=0.5, rely=0.5, anchor='center', width=400, height=20)
    txt_patr = Entry(window,width=20)
    txt_patr.place(relx=0.5, rely=0.55, anchor='center', width=400, height=20)

    def print_mark():
        a = txt_name.get()
        b = txt_lname.get()
        c = txt_patr.get()

        mycursor = db.cursor()

        sql = """SELECT name, lastname, patronymic, department, maths,
informatics, physics
FROM entrant

```



```

WHERE name = %s and lastname = %s and patronymic = %s"""
val = (a, b, c)

mycursor.execute(sql, val)

columns = [desc[0] for desc in mycursor.description]
rows = mycursor.fetchall()

db.commit()

new_window = Tk()
new_window.title("Список оценок")
new_window.geometry("800x500")

tree = Treeview(new_window)
tree.pack(side=LEFT, fill=BOTH, expand=True)

tree["columns"] = columns
tree["show"] = "headings"

for col in columns:
    tree.column(col, anchor='center', width=len(col)*10)
    tree.heading(col, text=col, anchor='center')

for row in rows:
    tree.insert("", "end", values=row)

scrollbar = Scrollbar(new_window, orient="horizontal",
command=tree.xview)
tree.configure(xscrollcommand=scrollbar.set)

new_window.mainloop()

btn_list = Button(window, text="Назад", command=clicked_user)
btn_list.place(relx=0.35, rely=0.75, anchor='center', width=200,
height=40)
btn_list = Button(window, text="Вывести список", command=print_mark)
btn_list.place(relx=0.65, rely=0.75, anchor='center', width=200,
height=40)

##### Вывод даты консультации и
экзамена для абитуриента
#####

def cons_exam():
    global window
    window.destroy()
    window=Tk()
    window.geometry('800x500')
    window['background'] = "lightblue"
    window.title("Даты консультаций и экзаменов абитуриента")

    style = Style()

    style.configure('TButton',
                    font = ('calibri', 14, 'bold'),
                    foreground = 'darkblue')

    lbl_name = Label(window, text="Введите имя абитуриента")
    lbl_name.place(relx=0.5, rely=0.1, anchor='center', width=400, height=20)
    txt_name = Entry(window,width=20)
    txt_name.place(relx=0.5, rely=0.15, anchor='center', width=400, height=20)

    lbl_lname = Label(window, text="Введите фамилию абитуриента")

```

```

lbl_lname.place(relx=0.5, rely=0.25, anchor='center', width=400,
height=20)
txt_lname = Entry(window,width=20)
txt_lname.place(relx=0.5, rely=0.3, anchor='center', width=400, height=20)

lbl_patr = Label(window, text="Введите отчество абитуриента")
lbl_patr.place(relx=0.5, rely=0.4, anchor='center', width=400, height=20)
txt_patr = Entry(window,width=20)
txt_patr.place(relx=0.5, rely=0.45, anchor='center', width=400, height=20)

lbl_sub = Label(window, text="Выберете предмет")
lbl_sub.place(relx=0.5, rely=0.55, anchor='center', width=400, height=20)
combo_sub = Combobox(window)
combo_sub['values'] = ("Математика", "Информатика", "Физика")
combo_sub.place(relx=0.5, rely=0.6, anchor='center', width=400, height=20)

def print_exam():
    a = txt_name.get()
    b = txt_lname.get()
    c = txt_patr.get()
    d = combo_sub.get()

    if d == "Математика":
        d = "maths"
    elif d == "Информатика":
        d = "informatics"
    elif d == "Физика":
        d = "physics"

    mycursor = db.cursor()

    sql = """SELECT entrant.name, entrant.lastname, entrant.patronymic,
exam.name_exam, exam.date_cons, exam.date_exam
FROM entrant
JOIN exam ON entrant.num_flow = exam.num_flow
WHERE entrant.name = %s and exam.name_exam = %s"""
    val = (a, b)

    mycursor.execute(sql, val)

    columns = [desc[0] for desc in mycursor.description]
    rows = mycursor.fetchall()

    db.commit()

    new_window = Tk()
    new_window.title("Список консультаций и экзаменов у абитуриентов")
    new_window.geometry("800x500")

    tree = Treeview(new_window)
    tree.pack(side=LEFT, fill=BOTH, expand=True)

    tree["columns"] = columns
    tree["show"] = "headings"

    for col in columns:
        tree.column(col, anchor='center', width=len(col)*10)
        tree.heading(col, text=col, anchor='center')

    for row in rows:
        tree.insert("", "end", values=row)

    scrollbar = Scrollbar(new_window, orient="horizontal",
command=tree.xview)

```

```

tree.configure(xscrollcommand=scrollbar.set)

new_window.mainloop()

btn_list = Button(window, text="Назад", command=clicked_user)
btn_list.place(relx=0.35, rely=0.75, anchor='center', width=200,
height=40)
btn_list = Button(window, text="Вывести список", command=print_exam)
btn_list.place(relx=0.65, rely=0.75, anchor='center', width=200,
height=40)

##### Вывод номеров аудиторий для
экзаменов у группы #####

def num_audit():
    global window
    window.destroy()
    window=Tk()
    window.geometry('800x500')
    window['background'] = "lightblue"
    window.title("Номера аудиторий для экзаменов")

    style = Style()

    style.configure('TButton',
                    font = ('calibri', 14, 'bold'),
                    foreground = 'darkblue')

    lbl_group = Label(window, text="Введите номер группы")
    lbl_group.place(relx=0.5, rely=0.3, anchor='center', width=400, height=20)
    txt_group = Entry(window,width=20)
    txt_group.place(relx=0.5, rely=0.35, anchor='center', width=400,
height=20)

    def print_aud():
        a = txt_group.get()

        mycursor = db.cursor()

        sql = """SELECT DISTINCT entrant.num_group, exam.name_exam,
exam.auditorium
                FROM exam
                JOIN entrant ON entrant.num_flow = exam.num_flow
                WHERE entrant.num_group = %s"""
        val = (a,)

        mycursor.execute(sql, val)

        columns = [desc[0] for desc in mycursor.description]
        rows = mycursor.fetchall()

        db.commit()

        new_window = Tk()
        new_window.title("Список номеров аудиторий")
        new_window.geometry("800x500")

        tree = Treeview(new_window)
        tree.pack(side=LEFT, fill=BOTH, expand=True)

        tree["columns"] = columns
        tree["show"] = "headings"

        for col in columns:

```

```

        tree.column(col, anchor='center', width=len(col)*10)
        tree.heading(col, text=col, anchor='center')

    for row in rows:
        tree.insert("", "end", values=row)

    scrollbar = Scrollbar(new_window, orient="horizontal",
command=tree.xview)
    tree.configure(xscrollcommand=scrollbar.set)

    new_window.mainloop()

    btn_list = Button(window, text="Назад", command=clicked_user)
    btn_list.place(relx=0.35, rely=0.55, anchor='center', width=200,
height=40)
    btn_list = Button(window, text="Вывести список", command=print_aud)
    btn_list.place(relx=0.65, rely=0.55, anchor='center', width=200,
height=40)

##### Вывод списка групп в аудитории
#####

def list_audit():
    global window
    window.destroy()
    window=Tk()
    window.geometry('800x500')
    window['background'] = "lightblue"
    window.title("Список групп в аудитории")

    style = Style()

    style.configure('TButton',
                    font = ('calibri', 14, 'bold'),
                    foreground = 'darkblue')

    lbl_audit = Label(window, text="Введите номер аудитории")
    lbl_audit.place(relx=0.5, rely=0.3, anchor='center', width=400, height=20)
    txt_audit = Entry(window,width=20)
    txt_audit.place(relx=0.5, rely=0.35, anchor='center', width=400,
height=20)

    lbl_time = Label(window, text="Введите дату и время")
    lbl_time.place(relx=0.5, rely=0.45, anchor='center', width=400, height=20)
    txt_time = Entry(window,width=20)
    txt_time.place(relx=0.5, rely=0.5, anchor='center', width=400, height=20)

    def print_group():
        a = txt_audit.get()
        b = txt_time.get()

        mycursor = db.cursor()

        sql = """SELECT DISTINCT entrant.num_group, exam.num_flow,
exam.auditorium
                FROM exam
                JOIN entrant ON entrant.num_flow = exam.num_flow
                WHERE exam.auditorium = %s and (exam.date_cons = %s or
exam.date_exam = %s)"""
        val = (a, b, b)

        mycursor.execute(sql, val)

        columns = [desc[0] for desc in mycursor.description]

```

```

rows = mycursor.fetchall()

db.commit()

new_window = Tk()
new_window.title("Список групп в аудитории")
new_window.geometry("800x500")

tree = Treeview(new_window)
tree.pack(side=LEFT, fill=BOTH, expand=True)

tree["columns"] = columns
tree["show"] = "headings"

for col in columns:
    tree.column(col, anchor='center', width=len(col)*10)
    tree.heading(col, text=col, anchor='center')

for row in rows:
    tree.insert("", "end", values=row)

scrollbar = Scrollbar(new_window, orient="horizontal",
command=tree.xview)
tree.configure(xscrollcommand=scrollbar.set)

new_window.mainloop()

btn_list = Button(window, text="Назад", command=clicked_user)
btn_list.place(relx=0.35, rely=0.7, anchor='center', width=200, height=40)
btn_list = Button(window, text="Вывести список", command=print_group)
btn_list.place(relx=0.65, rely=0.7, anchor='center', width=200, height=40)

```

```

##### Сотрудник
#####

```

```

def clicked_user():

```

```

    global window
    window.destroy()
    window=Tk()
    window.geometry('800x500')
    window['background'] = "lightblue"
    window.title("Сотрудник")

    style = Style()

    style.configure('TButton',
                    font = ('calibri', 14, 'bold'),
                    foreground = 'darkblue')

    # кнопка для списка абитуриентов
    btn_list_entr = Button(window, text="Вывести список абитуриентов",
command=list_entr)
    btn_list_entr.place(relx=0.5, rely=0.2, anchor='center', width=400,
height=40)

    # кнопка для оценок абитуриента
    btn_mark_entr = Button(window, text="Вывести оценки абитуриента",
command=mark_entr)
    btn_mark_entr.place(relx=0.5, rely=0.3, anchor='center', width=400,
height=40)

    # кнопка для даты консультации и экзамена для абитуриента по данному
предмету

```

```

    btn_cons_exam = Button(window, text="Вывести даты консультации и экзамена
для абитуриента", command=cons_exam)
    btn_cons_exam.place(relx=0.5, rely=0.4, anchor='center', width=400,
height=40)

    # кнопка для номера аудиторий, где будут экзамены у заданной группы
    btn_num_audit = Button(window, text="Вывести номера аудиторий для
экзаменов у группы", command=num_audit)
    btn_num_audit.place(relx=0.5, rely=0.5, anchor='center', width=400,
height=40)

    # кнопка для списка групп в аудитории
    btn_list_audit = Button(window, text="Вывести список групп в аудитории",
command=list_audit)
    btn_list_audit.place(relx=0.5, rely=0.6, anchor='center', width=400,
height=40)

    # кнопка возврата в главное меню
    btn_menu = Button(window, text="Назад", command=menu)
    btn_menu.place(relx=0.5, rely=0.8, anchor='center', width=200, height=40)

##### Главное меню
#####

# кнопка для входа администратора
btn_adm = Button(window, text="Войти как администратор", command=clicked_adm)
btn_adm.place(relx=0.5, rely=0.4, anchor='center', width=400, height=40)

# кнопка для входа сотрудника
btn_user = Button(window, text="Войти как сотрудник приемной комиссии",
command=clicked_user)
btn_user.place(relx=0.5, rely=0.5, anchor='center', width=400, height=40)

window.mainloop()

```

## 5. РАБОТА В ПРИЛОЖЕНИИ

После запуска приложения открывается окно, где можно выбрать под какой ролью будет использоваться приложение (администратор или сотрудник приемной комиссии – пользователь).

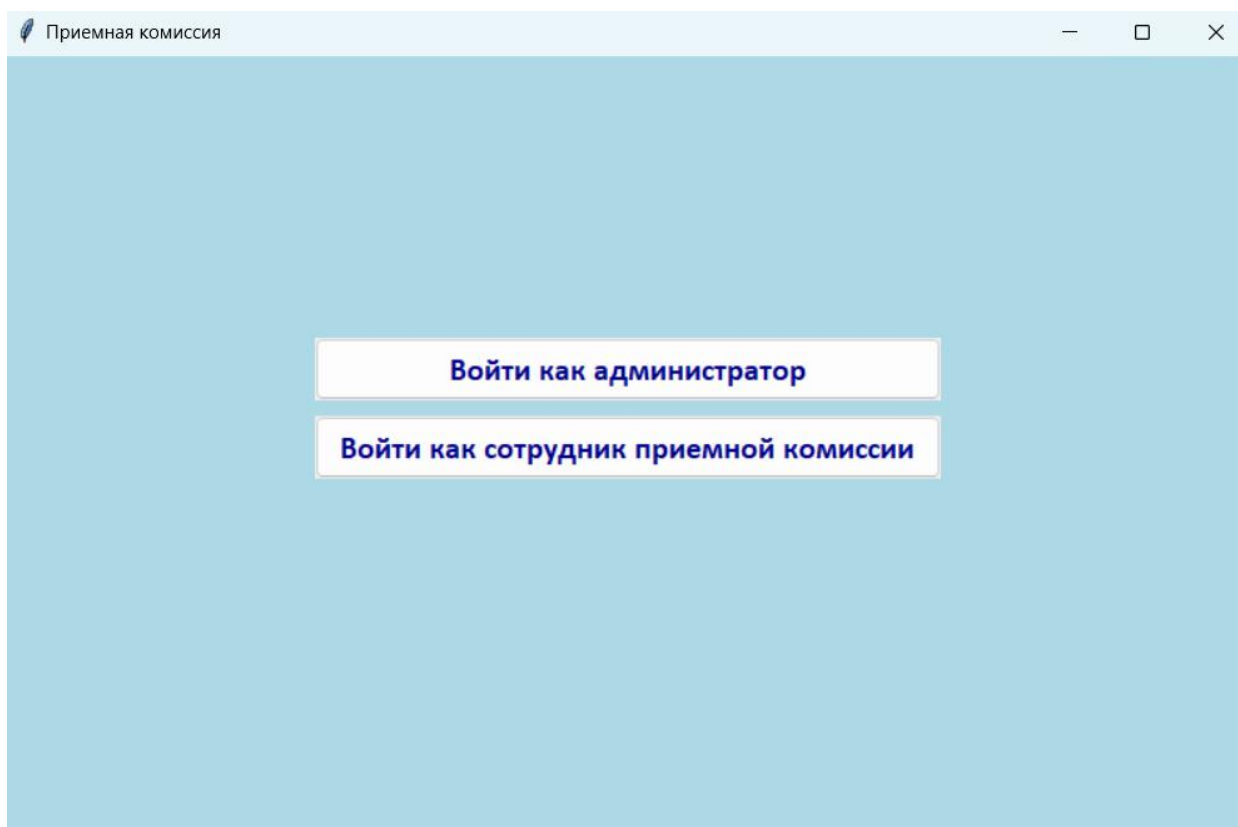


Рис. 6 – Главное окно приложения

Рассмотрим сначала работу приложения со стороны администратора.

После выбора данной роли (нажатия соответствующей кнопки) открывается окно с выбором функций, который доступны администратору.

Во многих окнах приложения есть кнопка «Назад», которая позволяет вернуться к предыдущему окну и сделать другой выбор при нажатии на кнопку.

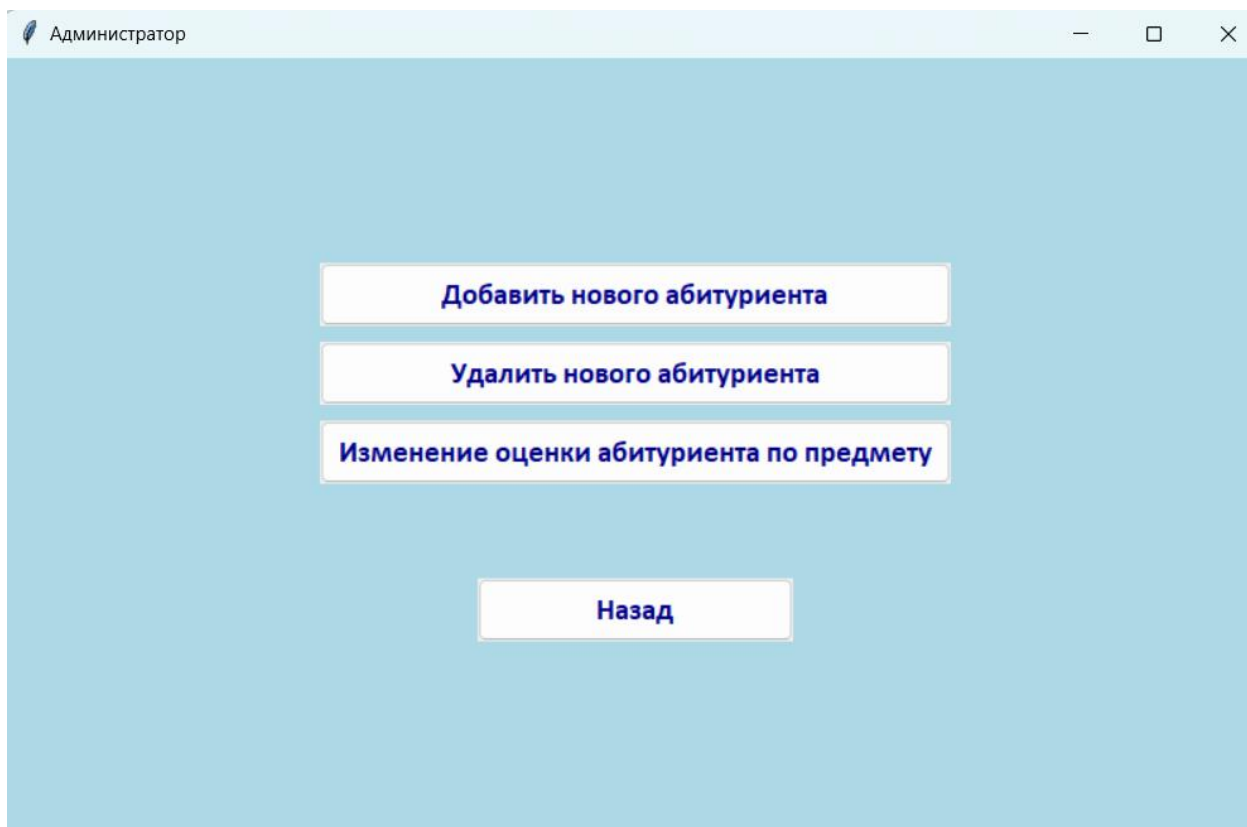


Рис. 7 – Окно администратора

Рассмотрим первую функцию, которая доступна администратору – добавление нового абитуриента.

После нажатия кнопки открывается окно, где нужно ввести информацию о новом абитуриенте (имя, фамилия, отчество, номер экзаменационного листа, факультет и кафедра).

Если данные введены корректно, то окно ввода данных закроется, вновь появится окно с выбором функций для абитуриента, также появится уведомление о том, что данные успешно добавились.



Добавление абитуриента в базу данных

Введите имя абитуриента  
Вадим

Введите фамилию абитуриента  
Шнырев

Введите отчество абитуриента  
Олегович

Введите номер экзаменационного листа  
171717

Введите факультет абитуриента  
ФРТ

Введите кафедру абитуриента  
ПС

**Добавить**

Рис. 8 – Окно ввода информации для добавления нового абитуриента

Администратор

Уведомление

Данные успешно добавлены

**Добавить нового абитуриента**

**Удалить нового абитуриента**

**Изменение оценки абитуриента по предмету**

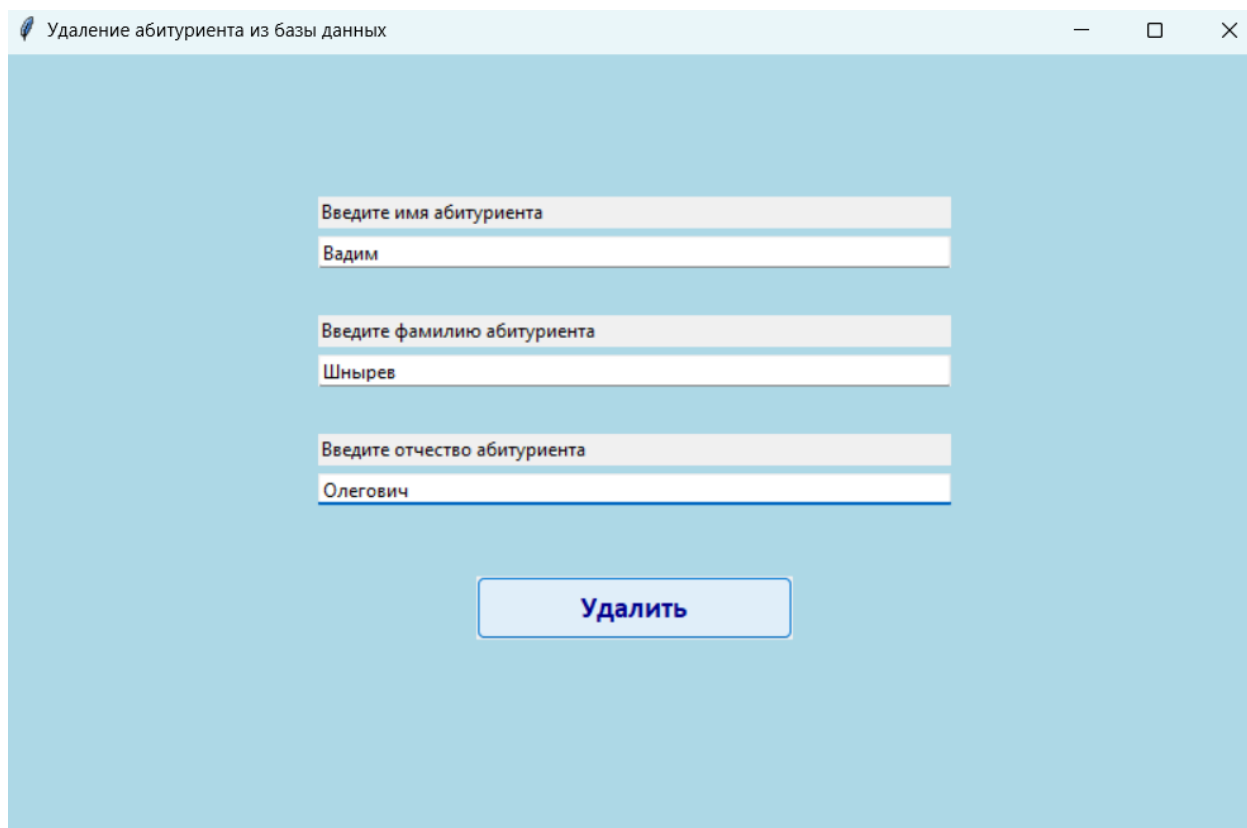
**Назад**

Рис. 9 – Уведомление об успешном добавлении данных

После трех секунд уведомление исчезнет (или его можно закрыть раньше самостоятельно) и можно будет вновь выбирать функцию.

Рассмотрим функцию удаления абитуриента.

После нажатия на кнопку появляется окно, где нужно ввести данные абитуриента, которого нужно удалить из базы данных.



Удаление абитуриента из базы данных

Введите имя абитуриента  
Вадим

Введите фамилию абитуриента  
Шнырев

Введите отчество абитуриента  
Олегович

Удалить

Рис. 10 – Окно ввода информации для удаления абитуриента

При успешном удалении закроется окно ввода, появится уведомление об успешном удалении данных и будут доступны функции для администратора.

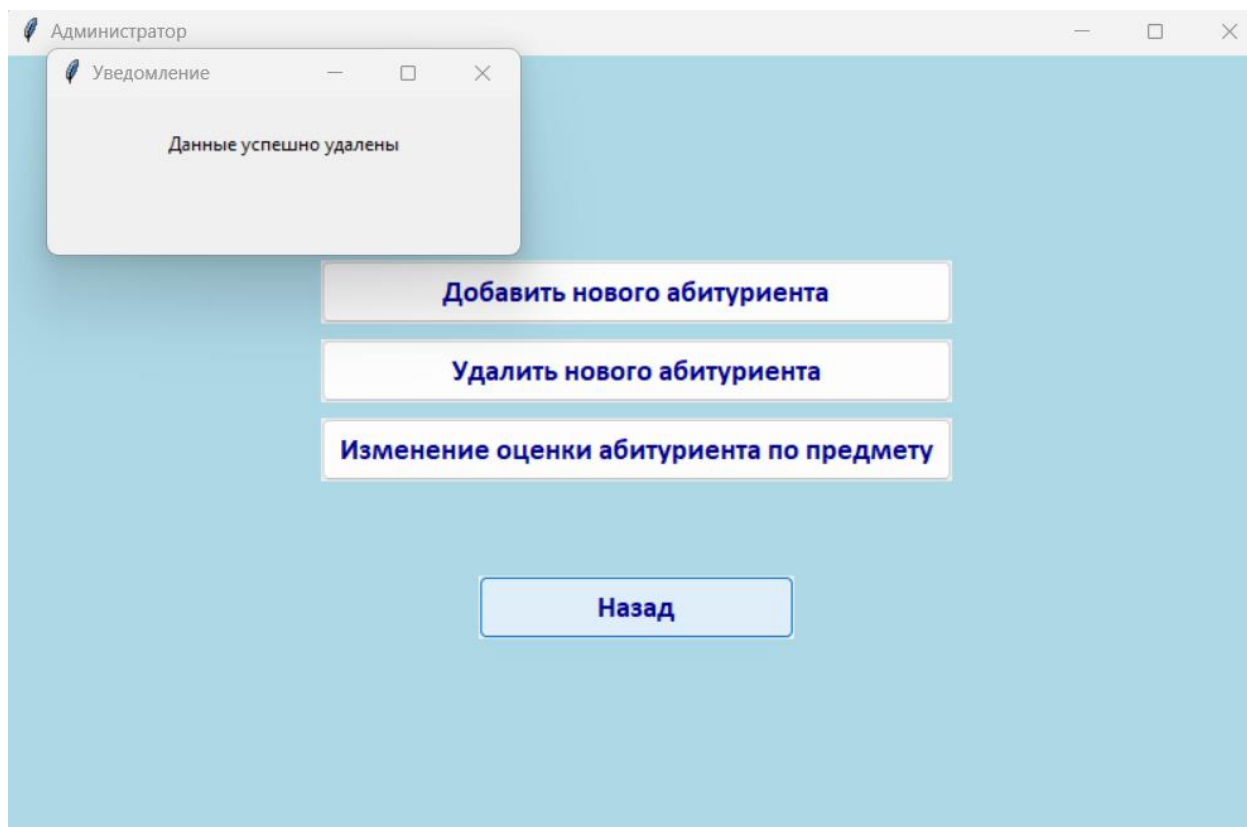


Рис. 11 – Уведомление об успешном удалении данных

Последняя функция для администратора – изменение оценки по какому-либо экзамену у абитуриента.

После нажатия соответствующей кнопки появляется окно для ввода информации об абитуриенте, которому нужно изменить оценку, также администратору нужно выбрать предмет, по которому будет производиться изменение и ту оценку, которую получил абитуриент.

При успешном изменении оценки появится уведомление об этом, так же окно ввода закроется и на его месте появится окно с выбором функций.

Изменение оценки по предмету у абитуриента

Введите имя абитуриента  
Анна

Введите фамилию абитуриента  
Рассказова

Введите отчество абитуриента  
Николаевна

Выберете предмет, по которому нужно изменить оценку у абитуриента  
Информатика

Выберете оценку, которую получил абитуриент  
4

**Изменить оценку**

Рис. 12 – Окно ввода данных для изменения оценки абитуриента

Администратор

Уведомление

Данные успешно изменены

**Добавить нового абитуриента**

**Удалить нового абитуриента**

**Изменение оценки абитуриента по предмету**

**Назад**

Рис. 13 – Уведомление об успешном изменении данных

Дальше рассмотрим роль сотрудника приемной комиссии, у которого чуть больше доступных функций, однако изменять данные в базе он не может, так что все функции только выводят определенную информацию по запросу.

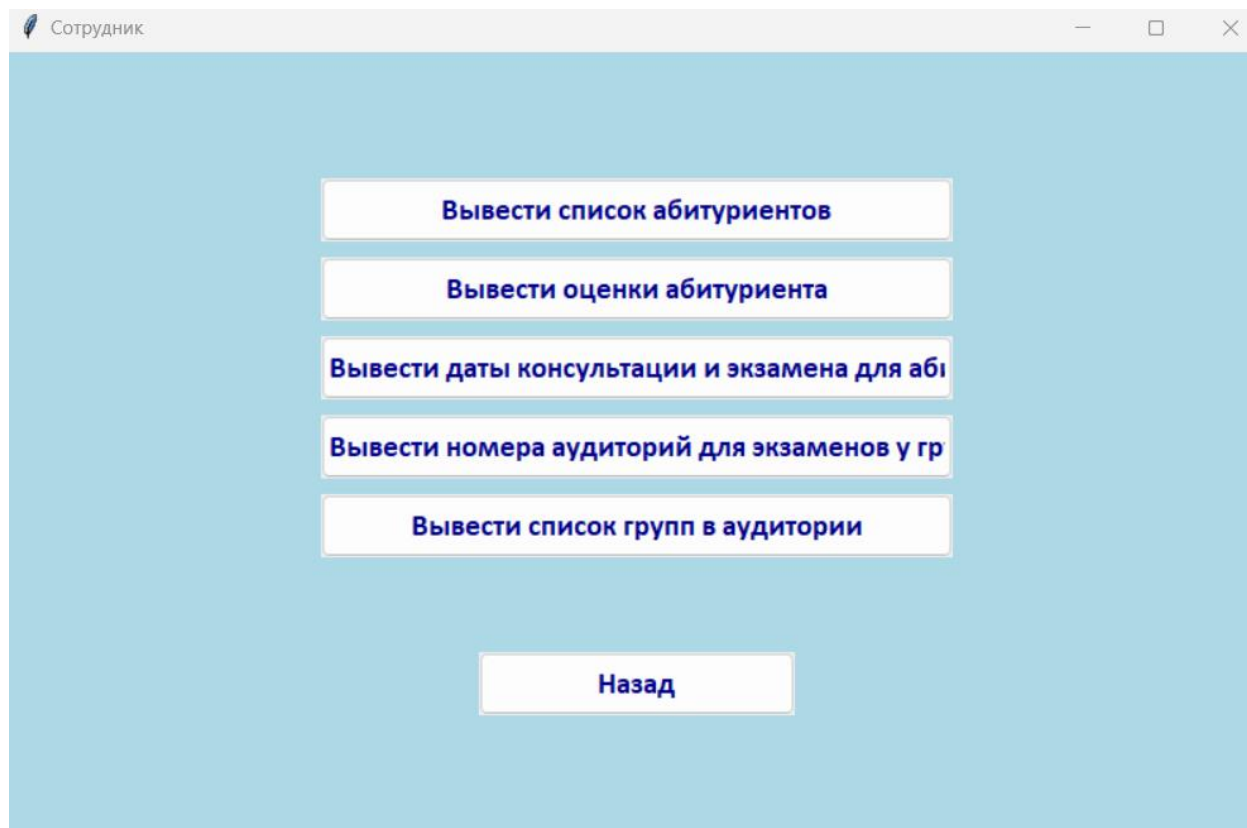


Рис. 14 – Окно сотрудника приемной комиссии

Рассмотрим первую доступную функцию для сотрудника приемной комиссии – вывод списка абитуриентов для определенного факультета.

После выбора данной функции открывается окно для ввода названия факультета.

После этого в новом окне выведутся все абитуриенты из базы данных, которые поступают на этот факультет (ФИО абитуриента и кафедра с этого факультета).

Список абитуриентов

Введите нужный факультет

ФРТ

Назад

Вывести список

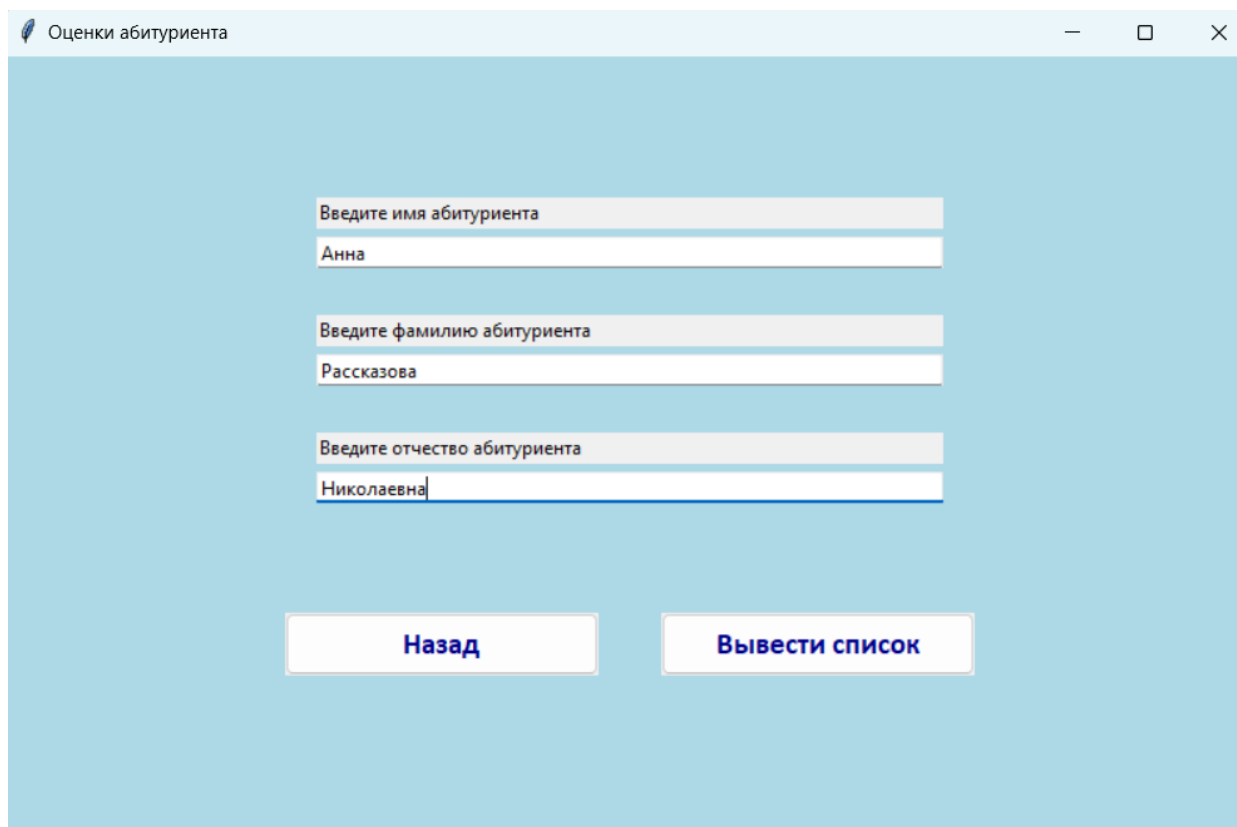
Рис. 15 – Окно ввода данных для вывода списка абитуриентов на определенный факультет

name	lastname	patronymic	department
Семен	Чернорецкий	Алексеевич	РТ
Артем	Иванов	Иванович	РТ

Рис. 16 – Таблица со списком абитуриентов на факультет (ФРТ)

Еще сотрудник приемной комиссии может вывести оценки определенного абитуриента.

Для этого ему нужно ввести ФИО интересующего его абитуриента.



Оценки абитуриента

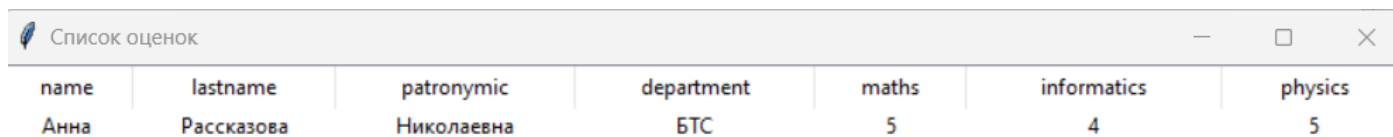
Введите имя абитуриента  
Анна

Введите фамилию абитуриента  
Рассказова

Введите отчество абитуриента  
Николаевна

Назад Вывести список

Рис. 17 – Окно ввода данных для вывода оценок абитуриента



name	lastname	patronymic	department	maths	informatics	physics
Анна	Рассказова	Николаевна	БТС	5	4	5

Рис. 18 – Таблица с оценками, полученными абитуриентом на экзаменах

Сотрудник приемной комиссии так же может вывести даты консультаций и экзаменов для абитуриента.

Для этого ему нужно ввести ФИО абитуриента и выбрать тот предмет, по которому он хочет получить информацию.

Даты консультаций и экзаменов абитуриента

Введите имя абитуриента  
Ксения

Введите фамилию абитуриента  
Воронина

Введите отчество абитуриента  
Дмитриевна

Выберете предмет  
Информатика

Назад Вывести список

Рис. 19 – Окно ввода данных для вывода дат консультаций и экзаменов по предмету

Можно посмотреть в каких аудиториях будут проходить экзамены у группы. Для этого просто нужен номер интересующей группы.



Номера аудиторий для экзаменов

Введите номер группы

3

Назад Вывести список

Рис. 20 – Окно ввода данных для вывода аудиторий экзаменов у группы

num_group	name_exam	auditorium
3	Математика	4321
3	Информатика	3344
3	Физика	3434

Рис. 21 – Таблица с номерами аудиторий

Последняя функция, которая доступна сотруднику приемной комиссии – вывод списка групп, которые находятся в аудитории в данное время.

Для этого нужно ввести в окне номер группы и дату, которая интересует.

На выходе в таблице получается список с номерами групп и потоков.

Список групп в аудитории

Введите номер аудитории

1234

Введите дату и время

2025-01-12

Назад

Вывести список

Рис. 22 – Окно ввода данных для вывода списка групп в аудитории в определенное время

num_group	num_flow	auditorium
1	1	1234
4	1	1234

Рис. 23 – Таблица с номерами групп и потоков

## **ЗАКЛЮЧЕНИЕ**

В ходе данной курсовой работы была спроектирована база данных для автоматизации деятельности приемной комиссии. Был разработан пакет функций для обращения к базе данных и написана программа для взаимодействия с данными в удобном формате.

## **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. SQL в примерах и задачах; Учеб. пособие / И.Ф. Астахова, А.П. Толстобров, В.М. Мельников. — Мн.: Новое знание, 2002. — 176 с.
2. Ризаев И.С., Яхина З.Т. Базы данных: Учебное пособие. Казань.: Изд-во Казан. гос. техн. ун-та. 2008. 240 с.
3. К. Дж. Дейт SQL и реляционная теория. Как грамотно писать код на SQL. — Пер. с англ. — СПб.: Символ-Плюс, 2010. — 480 с., ил.