



**ПОЛИТЕХ**

Санкт-Петербургский  
политехнический университет  
Петра Великого

## **Курсовая работа**

### **Разработка прототипа приложения для автоматизации работы магазина по дисциплине “Базы данных”**

Студент гр. 5130202/20202

*Гамин*

*Гамин В.И.*

Руководитель

*Степина Н.О.*

«12» декабря 2024 г.

## Оглавление

Введение.....	3
1. Описание интерфейса, ролей пользователей и работы с базой данных .....	6
1.1 Описание работы интерфейса .....	6
1.2 Описание работы с базой данных .....	7
2. Описание и тестирование работы интерфейса.....	8
2.1 Авторизация и управление правами доступа .....	8
2.2 Работа с товарными и расходными данными .....	11
2.3 Работа с журналами продаж и расходов .....	17
2.4. Генерация отчетов .....	20
2.5 Тестирование триггеров .....	22
Решенные проблемы: .....	24
3. Обзор кода .....	25
3.1 Модуль `auth.py` .....	25
3.2 Модуль `database.py` .....	26
3.4 Главный модуль `main.py` .....	27
1. `__init__` .....	27
2. `show_login_screen` .....	28
3. `create_main_menu` .....	29
4. Управление данными:.....	30
5. Пользовательские права.....	32
6. Работа с таблицами.....	32
Вывод.....	32

## **Введение**

**Актуальность темы** автоматизации работы магазина не вызывает сомнений в условиях современной экономики и высококонкурентной среды. Технологии автоматизации играют ключевую роль в оптимизации бизнес-процессов, повышении эффективности и точности учета, а также в снижении человеческого фактора. Управление товарооборотом, учетом расходов и продажами на основе автоматизированных информационных систем позволяет значительно улучшить контроль за процессами, ускорить обработку данных и обеспечить высокий уровень качества обслуживания. В связи с этим внедрение эффективных программных решений для автоматизации деятельности магазина, включая учет товарных операций и финансов, становится важной задачей как с теоретической, так и с практической точек зрения.

**Проблема**, рассматриваемая в рамках курсовой работы, заключается в автоматизации работы магазина с использованием базы данных PostgreSQL и разработки клиентского приложения. Для эффективного управления товарооборотом, расходами и продажами необходимо обеспечить точность учета, а также минимизировать ошибки при вводе данных. Важным аспектом является соблюдение целостности данных, управление их модификациями с помощью транзакций и триггеров, а также реализация вычислений через хранимые процедуры. В процессе реализации проекта также используется язык программирования Python для выполнения операций, взаимодействующих с базой данных, что позволяет автоматизировать различные процессы, такие как обработка данных, выполнение расчетов и выполнение запросов к базе данных.

**Проблематика автоматизации** работы магазина, а именно разработка программного обеспечения для эффективного управления учетной документацией и данными, активно изучается в области информационных технологий и компьютерных наук. Существующие решения, такие как

учетные системы для торговли и управления запасами, включают различные подходы к обработке данных. Однако остаются задачи, связанные с улучшением интерфейсов, повышением безопасности данных, а также с интеграцией современных технологий для работы с большими объемами информации.

**Целью курсовой работы** является автоматизация процессов учета в магазине с использованием базы данных PostgreSQL для хранения данных о товарах, продажах и расходах, а также разработка клиентского приложения для взаимодействия с этой базой данных. Это решение позволит улучшить процессы учета, минимизировать ошибки, ускорить обработку данных и повысить уровень прозрачности.

#### **Задачи работы:**

1. Разработка структуры базы данных для хранения данных о товарах, продажах и расходах с учетом требований целостности данных.
2. Реализация механизмов контроля целостности данных с помощью связей и транзакций.
3. Разработка триггеров для управления логикой работы системы, ограничивающих ввод некорректных данных.
4. Создание хранимых процедур для вычисления ключевых показателей (прибыли, самых доходных товаров и т.д.).
5. Реализация интерфейсов для ввода, модификации и удаления данных о товарах и статьях расходов.
6. Разработка интерфейса для ведения журнала продаж и расходов.
7. Реализация функционала для вычисления прибыли магазина и отображения доходных товаров за заданный интервал времени.
8. Использование языка Python для автоматизации процессов взаимодействия с базой данных и выполнения расчетов.

**Объектом исследования** является процесс автоматизации учета и контроля в магазине, включая взаимодействие с базой данных и клиентским приложением.

**Предметом исследования** является разработка и реализация механизмов работы с данными в системе учета товаров, продаж и расходов магазина.

**Методика исследования** включает анализ существующих решений для автоматизации учета в розничной торговле, проектирование базы данных и клиентского приложения с использованием языка SQL для работы с PostgreSQL, ADO.NET или JDBC для разработки интерфейса, а также применение Python для автоматизации работы с базой данных и выполнения вычислений.

В ходе работы будет проведено проектирование базы данных, реализация логики работы приложения с использованием триггеров и хранимых процедур, а также создание интерфейсов для взаимодействия с пользователем, что позволит оценить эффективность предложенной системы в условиях реальной торговли.

## **1. Описание интерфейса, ролей пользователей и работы с базой данных**

### **1.1 Описание работы интерфейса**

Интерфейс приложения был разработан с использованием библиотеки Tkinter для Python, что позволяет создать простой и интуитивно понятный графический интерфейс для взаимодействия с пользователем. Он включает несколько окон, каждое из которых соответствует определенному функциональному разделу системы. Основные компоненты интерфейса включают окна для управления товарами, статьями расходов, продажами и отчетами.

**Экран входа:** На экране входа пользователи вводят свои логин и пароль для доступа к системе. В зависимости от введенных данных, пользователю предоставляется доступ как администратору (с полными правами) или как обычному пользователю (с ограниченными правами). Это помогает контролировать доступ к ключевым операциям, таким как изменение данных о товарах и расходах.

**Главное меню:** После успешного входа пользователю отображается главное меню, в котором представлены кнопки для навигации по основным разделам приложения:

- **Управление товарами:** Здесь можно добавлять, редактировать, удалять товары и просматривать весь ассортимент на складе.
- **Управление статьями расходов:** В этом разделе осуществляется добавление и редактирование категорий расходов.
- **Продажи и расходы:** Журнал продаж и расходов, где отображаются все завершенные операции.
- **Отчеты:** Генерация отчетов по прибыли за месяц и по самым доходным товарам.

**Работа с данными:** В каждом разделе интерфейса предусмотрены формы для ввода, редактирования и удаления данных. Для выполнения операций используются текстовые поля, выпадающие списки и кнопки. Важно, что при добавлении или изменении данных выполняются проверки на корректность ввода, что помогает избежать ошибок в базе данных.

**Права доступа:** Интерфейс ограничивает доступ в зависимости от роли пользователя. Например, обычный пользователь может только просматривать данные о товарах, но не имеет права их редактировать, в то время как администратор имеет полный доступ ко всем разделам и функционалу.

**Проверка работоспособности интерфейса:** Интерфейс был протестирован на удобство использования и корректность отображения данных. Все формы работают в тесной связи с базой данных и обновляются после каждой операции. Также была проведена проверка на устойчивость интерфейса при многократном выполнении операций с одинаковыми данными, в ходе которой были обнаружены и исправлены незначительные ошибки, связанные с обновлением таблиц после изменения данных.

## 1.2 Описание работы с базой данных

Для хранения данных о товарах, статьях расходов, продажах и расходах используется база данных PostgreSQL. В процессе разработки была выполнена следующая структура базы данных:

**Структура базы данных:** База данных включает несколько таблиц, связанных между собой через внешние ключи. Основные таблицы:

- **Товары:** Хранят данные о наименованиях товаров, их ценах и количестве на складе.
- **Статьи расходов:** Содержат данные о категориях расходов, таких как аренда, зарплаты и другие.
- **Продажи и расходы:** Таблицы для учета всех операций, включая дату, сумму и товар.

- **Журналы:** Хранят информацию о выполненных операциях в системе.

**Триггеры и хранимые процедуры:** Для обеспечения целостности данных и автоматизации вычислений в базе данных были реализованы триггеры и хранимые процедуры. Например, триггеры используются для проверки корректности данных при добавлении или удалении записей, а хранимые процедуры — для вычисления прибыли магазина и генерации отчетов по самым доходным товарам.

**Контроль целостности данных:** В базе данных используются механизмы контроля целостности данных, такие как внешние ключи, ограничения и транзакции, что позволяет избежать ошибок, таких как дублирование записей или потеря данных.

**Автоматизация работы с базой данных через Python:** Взаимодействие с базой данных осуществляется с помощью языка программирования Python. Это позволяет автоматизировать процессы добавления и обновления данных, а также выполнения вычислений и генерации отчетов по ключевым показателям.

## **2. Описание и тестирование работы интерфейса**

Интерфейс приложения был разработан с акцентом на удобство и простоту взаимодействия пользователя с системой. В этом разделе подробно описан процесс тестирования интерфейса, разделённый на этапы для двух типов пользователей: администратора и обычного пользователя. Мы рассмотрим, какие функции доступны каждому типу, а также выявленные проблемы и способы их устранения.

### **2.1 Авторизация и управление правами доступа**

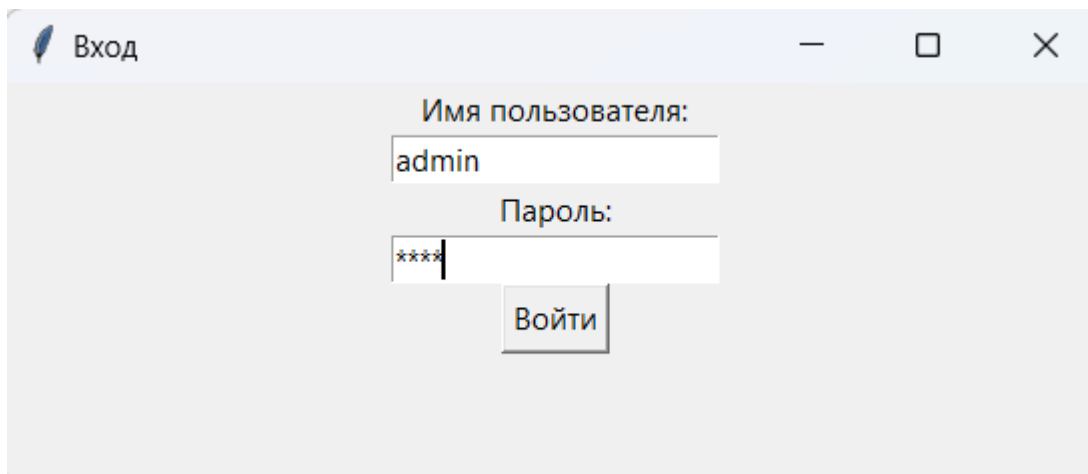
#### **Авторизация администратора:**

При успешной авторизации администратор попадает на главную панель управления.



### Проверка функциональности:

- Ввод правильных данных (логин и пароль).

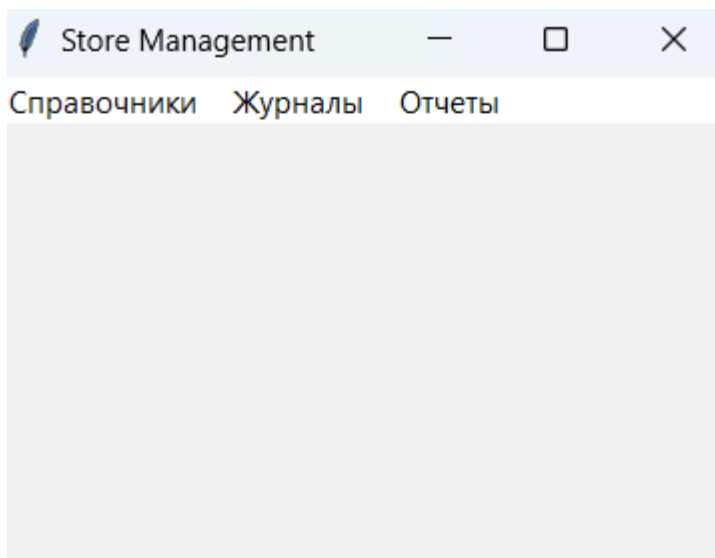


Вход

Имя пользователя:  
admin

Пароль:  
\*\*\*\*

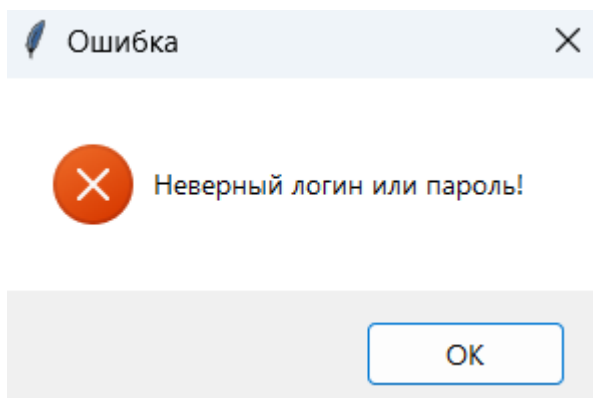
Войти



Store Management

Справочники Журналы Отчеты

- Авторизация с ошибочными данными (неправильный логин или пароль) и проверка корректности отображения ошибки.



### Результаты:

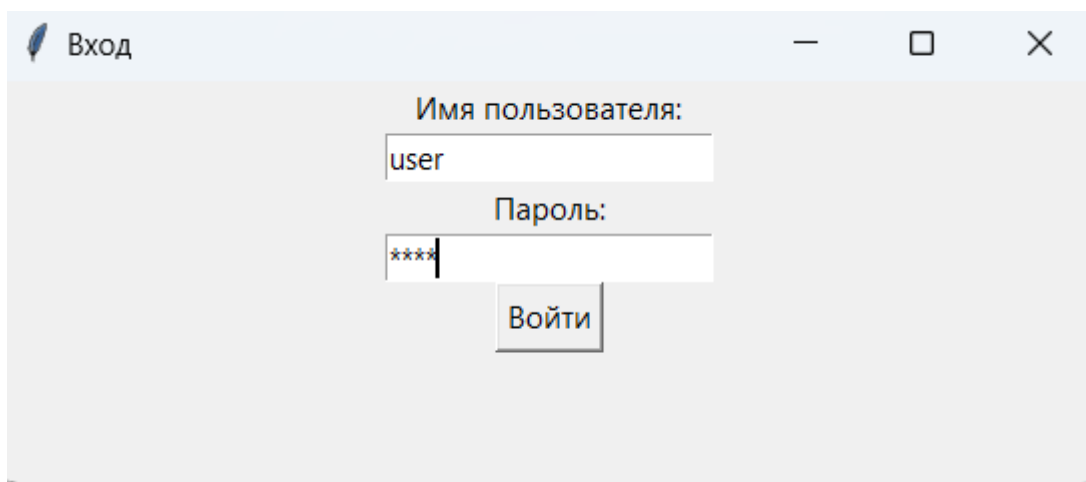
- Администратор успешно заходит в систему с полным доступом, может управлять всеми разделами.

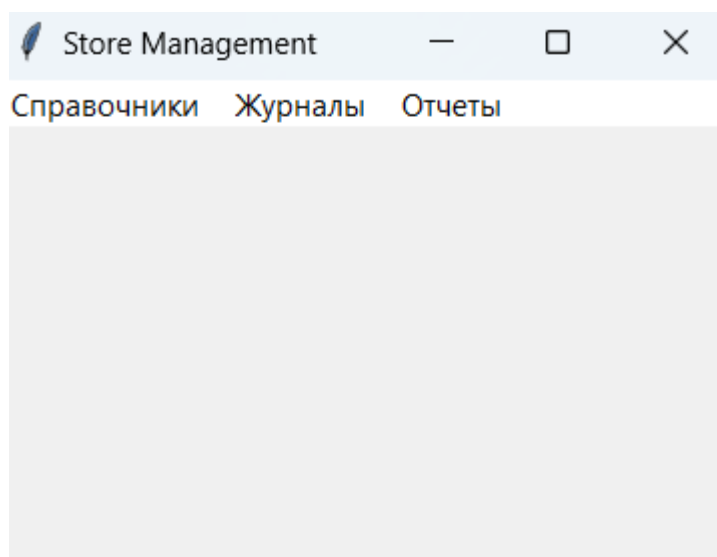
### Авторизация пользователя:

Обычный пользователь имеет ограниченный доступ, который позволяет только просматривать данные без возможности их редактирования или удаления.

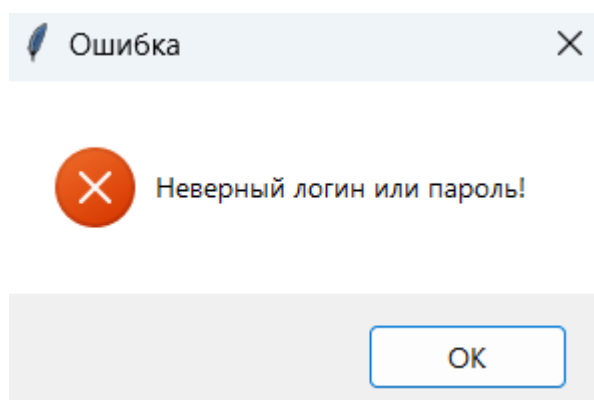
### Проверка функциональности:

- Ввод правильных данных (логин и пароль).





- Авторизация с ошибочными данными и проверка отображения ошибки.



### Результаты:

- Пользователь успешно заходит в систему с ограниченными правами.

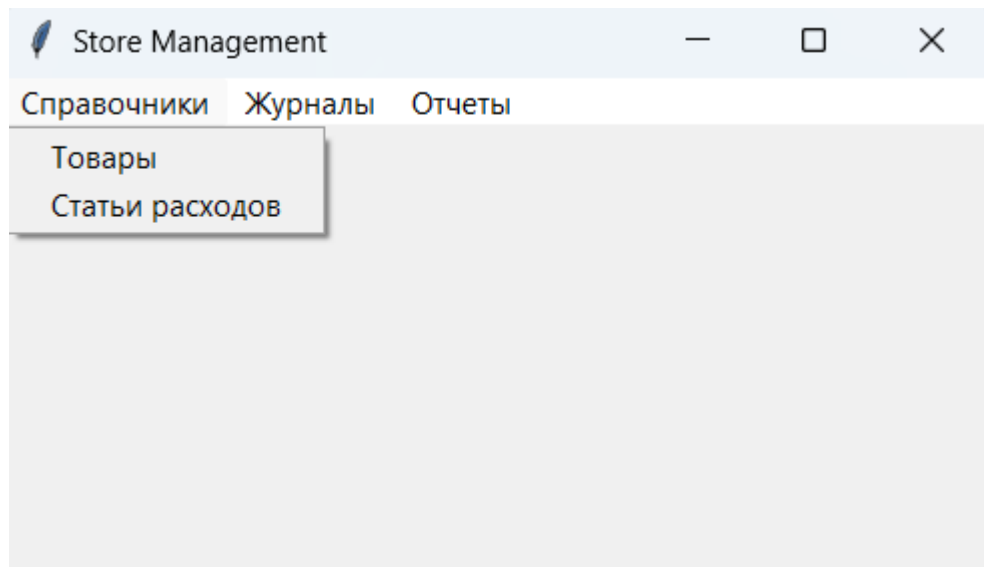
## 2.2 Работа с товарными и расходными данными

### Действия пользователя:

Обычный пользователь имеет только права для **просмотра** данных о товарах и расходах, но не может их редактировать, добавлять или удалять.

### Проверка функциональности:

- Пользователь может просматривать список товаров и расходов, но не видит кнопки для их редактирования или удаления.



Товары				
ID	Наименование	Количество	Цена	
39	Product A	50	110	
40	Product B	30	220	
41	Product C	70	165	
42	Product D	40	275	
43	Product E	45	330	

Статьи расходов	
ID	Наименование
25	Rent
26	Utilities
27	Salaries
28	Marketing

### Результаты:

- Обычный пользователь успешно видит список товаров и расходов, но все кнопки для изменения данных, такие как "Редактировать" и "Удалить", недоступны.

Действия администратора:

Администратор может добавлять, редактировать и удалять товары и статьи расходов.

Проверка функциональности:

- Добавление нового товара/расхода.

Товары					—	□	×
	ID	Наименование	Количество	Цена			
39		Product A	50	110			
40		Product B	30	220			
41		Product C	70	165			
42		Product D	40	275			
43		Product E	45	330			

Наименование	Количество	Цена
Добавить	Удалить	Редактировать

Товары					—	□	×
	ID	Наименование	Количество	Цена			
39		Product A	50	110			
40		Product B	30	220			
41		Product C	70	165			
42		Product D	40	275			
43		Product E	45	330			
60		test	60	200			

test	60	200
Добавить	Удалить	Редактировать

Статьи расходов		—	□	×
ID	Наименование			
25	Rent			
26	Utilities			
27	Salaries			
28	Marketing			

Test	
Добавить	Удалить Редактировать

Статьи расходов		—	□	×
ID	Наименование			
25	Rent			
26	Utilities			
27	Salaries			
28	Marketing			
45	Test			

Test	
Добавить	Удалить Редактировать

- Редактирование существующего товара/расхода.

Товары

ID	Наименование	Количество	Цена
39	Product A	50	110
40			220
41			165
42			275
43			330
60			200

Редакти...

Наименование

test

Количество

50

Цена

300

Сохранить изменения

Наименование

Количество

Цена

Добавить

Удалить

Редактировать

ID	Наименование	Количество	Цена
39	Product A	50	110
40	Product B	30	220
41	Product C	70	165
42	Product D	40	275
43	Product E	45	330
60	test	50	300

Наименование

Количество

Цена

Добавить

Удалить

Редактировать

Статьи расходов

ID	Наименование
25	Rent
26	Utilities
27	Salaries
28	Marketing
47	Tess

Tess

Добавить

Удалить

Редактировать

Статьи расходов	
ID	Наименование
25	Rent
26	Utilities
27	Salaries
28	Marketing
47	Tess

Редакти...

Наименование

Test

Сохранить изменения

Наименование

Добавить

Удалить

Редактировать

Статьи расходов	
ID	Наименование
25	Rent
26	Utilities
27	Salaries
28	Marketing
47	Test

Добавить

Удалить

Редактировать

- Удаление товара/расхода.




Статьи расходов	
ID	Наименование
25	Rent
26	Utilities
27	Salaries
28	Marketing

Наименование			
	Добавить	Удалить	Редактировать

### Проблемы:

- При редактировании или удалении данных, если поля формы не были заполнены корректно, система не всегда отображала ошибку.

Ошибка	✕
--------	---


Поле 'Наименование' обязательно для заполнения!

OK

### Решение:

- Добавлены проверки обязательных полей.

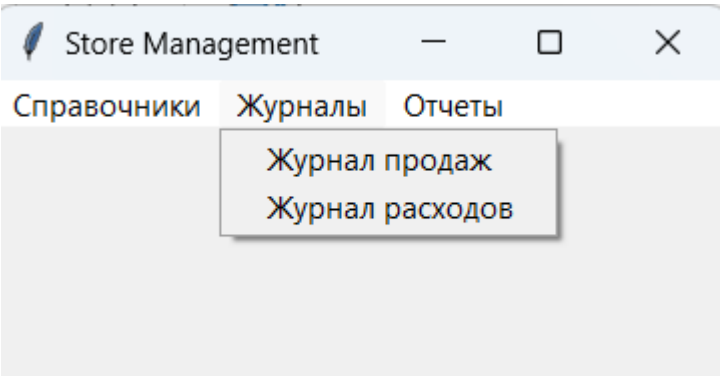
## 2.3 Работа с журналами продаж и расходов

### Действия пользователя:

Обычный пользователь может только просматривать записи в журналах, но не имеет прав на добавление, редактирование или удаление записей.

Проверка функциональности:

- Проверка доступа к журналам для пользователя.



Журнал продаж					
ID	Товар	Количество	Цена	Дата	
75	39	15	110	2024-09-10 00:00:00	
76	40	10	220	2024-09-22 00:00:00	
77	41	20	165	2024-10-15 00:00:00	
78	42	8	275	2024-10-18 00:00:00	
80	43	15	320	2024-10-25 00:00:00	

Журнал расходов				
ID	Сумма	Статья расхода	Дата	
29	700	27	2024-09-18 00:00:00	
30	400	28	2024-10-01 00:00:00	
28	300	26	2024-09-10 00:00:00	
27	600	25	2024-09-10 00:00:00	
36	123	28	2024-08-02 00:00:00	

Результаты:

- Пользователь может только просматривать журнал, но не может изменять его.

Действия администратора:

Администратор может добавлять записи в журналы продаж и расходов, а также редактировать и удалять записи.

Проверка функциональности:

- Добавление записи в журнал.

Журнал продаж

ID	Товар	Количество	Цена	Дата
75	39	15	110	2024-09-10 00:00:00
76	40	10	220	2024-09-22 00:00:00
77	41	20	165	2024-10-15 00:00:00
78	42	8	275	2024-10-18 00:00:00
80	43	15	320	2024-10-25 00:00:00
82	40	14	320	2024-10-26 00:00:00
40		14	320	2024-10-26
Добавить		Удалить		Редактировать

Журнал расходов

ID	Сумма	Статья расхода	Дата
29	700	27	2024-09-18 00:00:00
30	400	28	2024-10-01 00:00:00
28	300	26	2024-09-10 00:00:00
27	600	25	2024-09-10 00:00:00
36	123	28	2024-08-02 00:00:00
39	500	28	2024-10-12 00:00:00

500

28

2024-10-12

Добавить

Удалить

Редактировать

- Редактирование и удаление записей.

Редактирование...

Товар

Количество

Цена

Дата

43

15

320

2024-10-25 00:00:00

Сохранить изменения

Редакти...

Сумма

Статья расхода

Дата

500

28

2024-10-12 00:00:00

Сохранить изменения

**Проблемы:**

- Иногда при массовом добавлении записей система не успевала обновить отображение данных.

**Решение:**

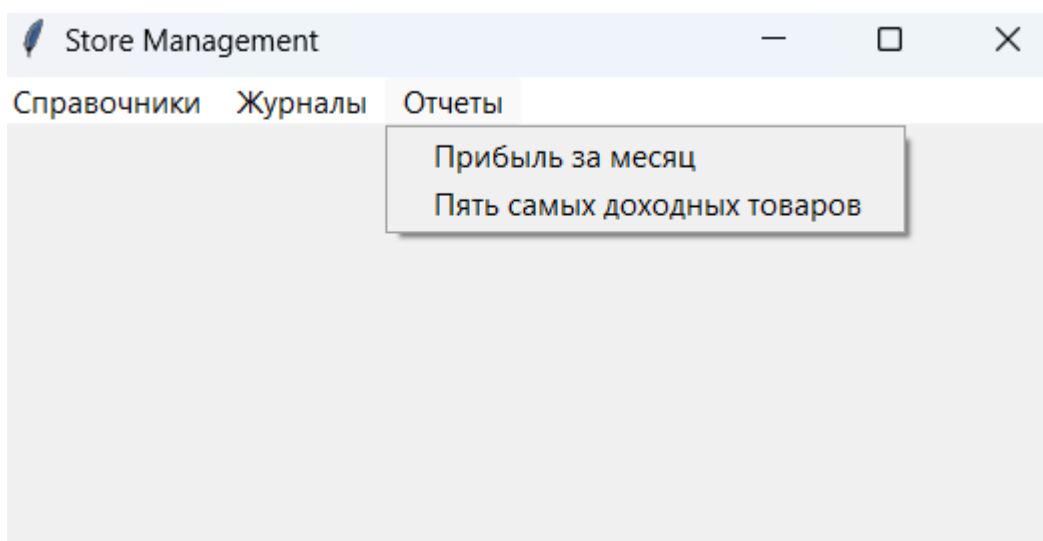
- Оптимизирована синхронизация с базой данных для более быстрой загрузки данных.

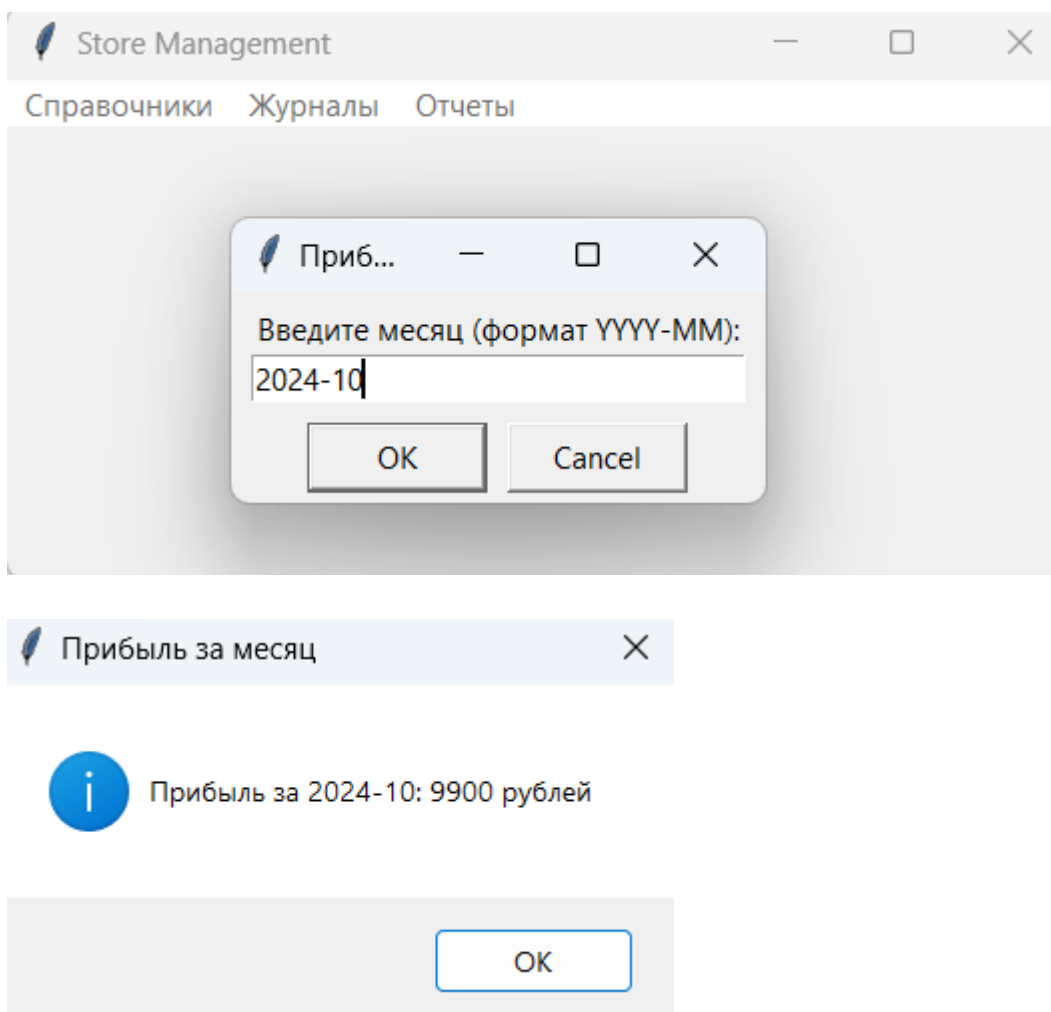
**2.4. Генерация отчетов****Действия администратора или пользователя:**

Администратор и пользователь имеет возможность генерировать отчеты по прибыли, продажам и популярным товарам.

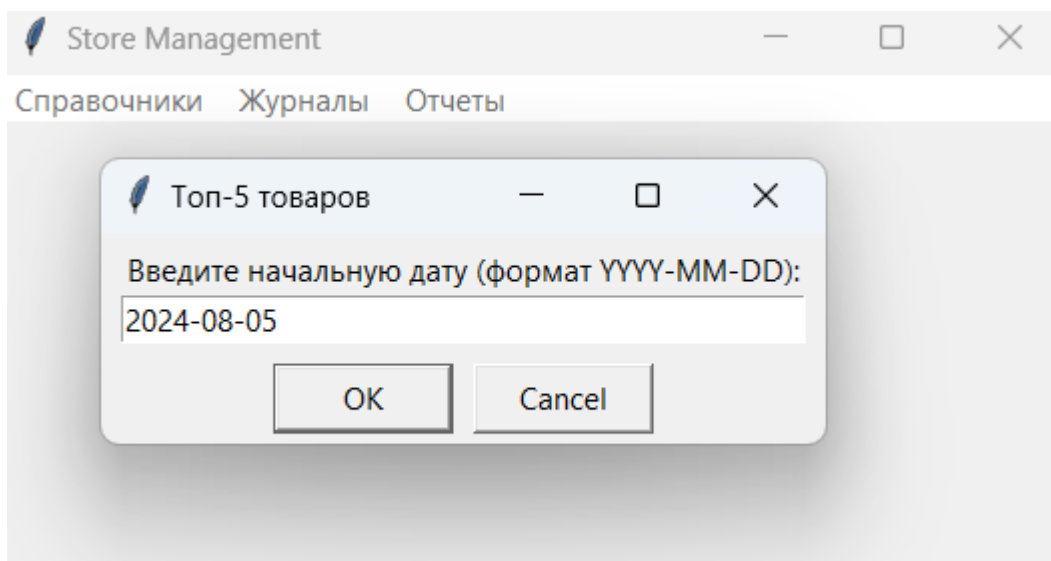
**Проверка функциональности:**

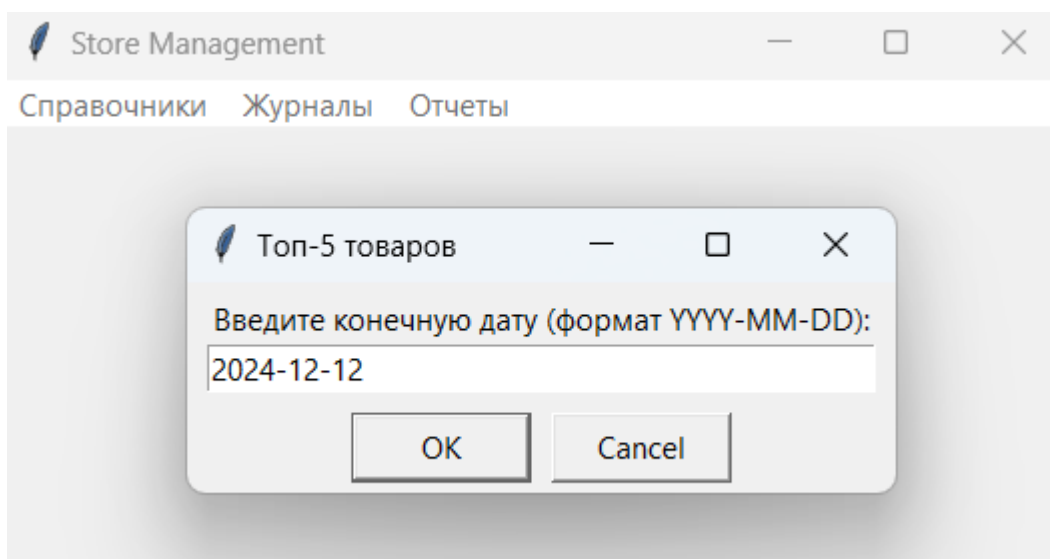
- Проверка отчета “Прибыль за месяц”





- Проверка генерации отчета “пять самых доходных товаров” в виде txt файла





Топ-5 самых доходных товаров  
Период: 2024-08-05 - 2024-12-12

Product E: 4800 рублей  
Product C: 3300 рублей  
Product B: 2200 рублей  
Product D: 2200 рублей  
Product A: 1650 рублей

- **Проблемы:**

- Некорректное отображение сумм в отчетах при большом объеме данных.

**Решение:**

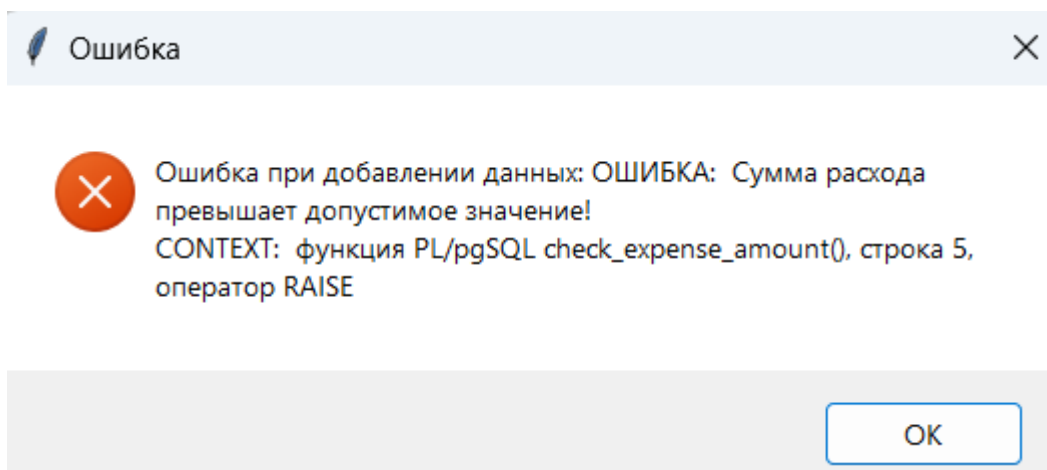
- Оптимизирован алгоритм расчета отчетов и исправлены ошибки в отображении данных.

## 2.5 Тестирование триггеров

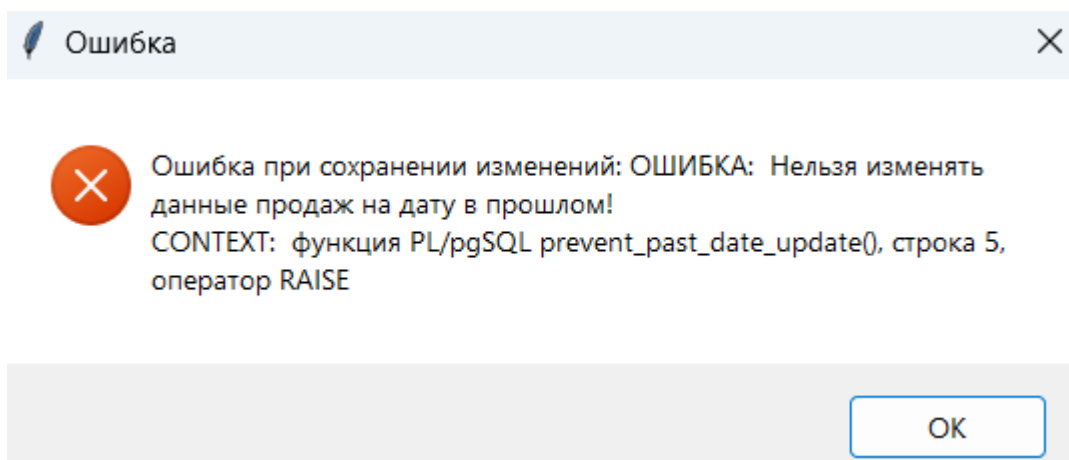
Триггеры были настроены для автоматической обработки данных при добавлении, изменении и удалении записей в базе данных. Их задача — поддержание целостности данных и выполнение необходимых операций при изменении данных.

### Проверка функциональности:

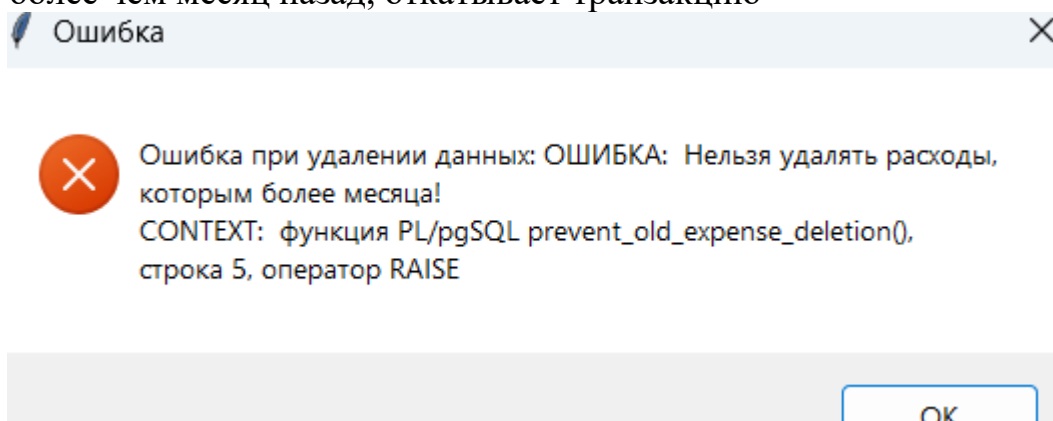
- Триггер, который не позволяет добавлять расход, с суммой большей заданной



- Триггер, который не позволяет изменять данные в таблице продаж задним числом от сегодняшней даты



- Триггер, который при удалении расхода в случае, если расход был более чем месяц назад, откатывает транзакцию



**Проблемы:**

- В случае некорректного добавления или редактирования данных, триггер иногда не срабатывал вовремя, что приводило к некорректным данным в базе.

**Решение:**

- Триггеры были настроены для более тщательной проверки данных, включая проверку всех полей на заполненность и корректность.

**Итоги тестирования и результаты**

По результатам тестирования интерфейс приложения продемонстрировал высокую стабильность и надежность. Все выявленные проблемы были успешно решены, и интерфейс теперь работает корректно при выполнении различных операций.

**Решенные проблемы:**

- Ошибки в обработке пустых полей и некорректного ввода данных.
- Проблемы с синхронизацией данных при массовом добавлении записей.
- Ошибки при генерации отчетов (некорректное отображение сумм).
- Ошибки срабатывания триггеров при добавлении, редактировании и удалении данных.

После внесения изменений и оптимизаций приложение прошло успешную проверку на работоспособность и стабильность работы для разных типов пользователей. Улучшены механизмы синхронизации данных, повысилась производительность системы и оптимизировано отображение данных.



### 3. Обзор кода

Код курсового проекта представляет собой приложение для управления данными магазина, разработанное с использованием языка Python. Приложение реализует графический интерфейс пользователя (GUI) на основе библиотеки `tkinter` и взаимодействует с базой данных PostgreSQL. Ниже приведено описание ключевых компонентов и функциональных модулей программы.

#### 3.1 Модуль `auth.py`

Этот модуль отвечает за аутентификацию пользователей. Основная функция `authenticate\_user` выполняет проверку имени пользователя и пароля. Она использует следующие шаги:

- Подключается к базе данных с помощью функции `get\_db\_connection`.
- Выполняет SQL-запрос для получения данных пользователя (имя, хэш пароля, роль).
- Проверяет правильность введенного пароля с использованием утилиты `check\_password`.
- Возвращает данные о пользователе, если проверка прошла успешно, иначе — `None`.

```
from .database import get_db_connection
from utils import check_password

def authenticate_user(username: str, password: str):
    """Проверяет логин и пароль пользователя."""
    conn = get_db_connection()
    cur = conn.cursor()
    cur.execute(
        "SELECT username, password_hash, role FROM users WHERE username = %s",
        (username,)
    )
    user = cur.fetchone()
    conn.close()

    if user and check_password(password, user[1]):
        return {"username": user[0], "role": user[2]}
    return None
```

### 3.2 Модуль `database.py`

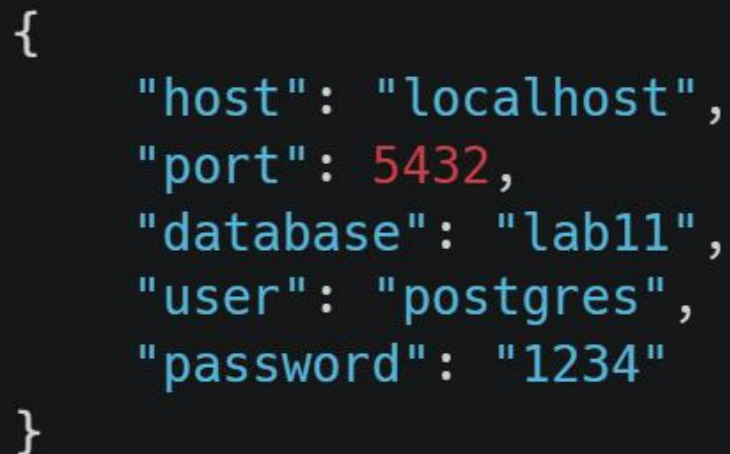
Этот модуль отвечает за установление соединения с базой данных PostgreSQL. Основная функция `get\_db\_connection` выполняет следующие действия:

- Загружает параметры подключения из файла `db\_config.json`.
- Создает соединение с базой данных с помощью библиотеки `psycopg2`.
- Возвращает объект подключения для использования в других модулях.

```
import psycopg2
import json
import os
def get_db_connection():
    """
    Устанавливает соединение с базой данных PostgreSQL.
    """
    with open('db_config.json', 'r', encoding='utf-8') as config_file:
        db_params = json.load(config_file)

    conn = psycopg2.connect(
        host=db_params['host'],
        port=db_params['port'],
        database=db_params['database'],
        user=db_params['user'],
        password=db_params['password']
    )
    return conn
```

**3.3 Файл `db\_config.json`** содержит параметры конфигурации базы данных, такие как хост, порт, имя базы, пользователь и пароль. Это позволяет легко настраивать приложение для работы с разными базами данных.



```
{  
    "host": "localhost",  
    "port": 5432,  
    "database": "lab11",  
    "user": "postgres",  
    "password": "1234"  
}
```

### 3.4 Главный модуль `main.py`

Этот модуль реализует графический интерфейс приложения и основную бизнес-логику. Основной класс `App` содержит следующие ключевые методы:

#### 1. `\_\_init\_\_`

- Инициализирует главное окно приложения.
- Запускает экран входа через метод `show\_login\_screen`.



```
class App:
    def __init__(self, root):
        self.root = root
        self.root.title("Store Management")
        self.user_authenticated = False
        self.current_user_role = None # Роль текущего пользователя
        self.show_login_screen()
```

## 2. `show\_login\_screen`

- Отображает окно авторизации с полями ввода для имени пользователя и пароля.
- Использует функцию `authenticate\_user` для проверки данных пользователя.
- При успешной аутентификации сохраняет роль пользователя и открывает главное меню.

```

def show_login_screen(self):
    """Экран входа."""
    login_window = tk.Toplevel(self.root)
    login_window.title("Вход")
    login_window.geometry("550x200")

    tk.Label(login_window, text="Имя пользователя:").pack()
    username_var = tk.StringVar()
    tk.Entry(login_window, textvariable=username_var).pack()

    tk.Label(login_window, text="Пароль:").pack()
    password_var = tk.StringVar()
    tk.Entry(login_window, textvariable=password_var, show="*").pack()

    def login():
        username = username_var.get()
        password = password_var.get()
        user_data = authenticate_user(username, password)
        if user_data:
            self.user_authenticated = True
            self.current_user_role = user_data['role'] # Сохраняем роль
            login_window.destroy()
            self.create_main_menu()
        else:
            messagebox.showerror("Ошибка", "Неверный логин или пароль!")

    tk.Button(login_window, text="Войти", command=login).pack()

```

### 3. `create\_main\_menu`

- Создает главное меню приложения, содержащее три категории: “Справочники”, “Журналы” и “Отчеты”.
- Каждая категория содержит пункты для управления соответствующими данными.

```

def create_main_menu(self):
    """Создает главное меню приложения."""
    menu = tk.Menu(self.root)
    self.root.config(menu=menu)

    # Справочники
    directories_menu = tk.Menu(menu, tearoff=0)
    directories_menu.add_command(label="Товары", command=self.manage_warehouses)
    directories_menu.add_command(label="Статьи расходов", command=self.manage_expense_items)
    menu.add_cascade(label="Справочники", menu=directories_menu)

    # Журналы
    journals_menu = tk.Menu(menu, tearoff=0)
    journals_menu.add_command(label="Журнал продаж", command=self.manage_sales_log)
    journals_menu.add_command(label="Журнал расходов", command=self.manage_charges_log)
    menu.add_cascade(label="Журналы", menu=journals_menu)

    # Отчеты
    reports_menu = tk.Menu(menu, tearoff=0)
    reports_menu.add_command(label="Прибыль за месяц", command=self.show_monthly_profit)
    reports_menu.add_command(label="Пять самых доходных товаров",
command=self.show_top_5_profitable_products)
    menu.add_cascade(label="Отчеты", menu=reports_menu)

```

#### 4. Управление данными:

- Методы `manage\_warehouses`, `manage\_expense\_items`, `manage\_sales\_log`, и `manage\_charges\_log` открывают окна для работы с различными таблицами базы данных. Эти методы используют общий подход для отображения данных и добавления новых записей.

- Метод `generic\_management\_window` предоставляет обобщенный функционал для управления записями (отображение таблиц, добавление, удаление, редактирование).

```

def manage_warehouses(self):
    """Управление справочником товаров."""
    self.generic_management_window("Товары", "warehouses",
                                   [("id", "ID"), ("name", "Наименование"), ("quantity",
"Количество"),
                                   ("amount", "Цена")])

def manage_expense_items(self):
    """Управление справочником статей расходов."""
    self.generic_management_window("Статьи расходов", "expense_items", [("id", "ID"), ("name",
"Наименование")])

def manage_sales_log(self):
    """Управление журналом продаж."""
    self.generic_management_window(
        "Журнал продаж", "sales",
        [("id", "ID"), ("warehouse_id", "Товар"), ("quantity", "Количество"), ("amount", "Цена"),
        ("sale_date", "Дата")])

def manage_charges_log(self):
    """Управление журналом расходов."""
    self.generic_management_window(
        "Журнал расходов", "charges",
        [("id", "ID"), ("amount", "Сумма"), ("expense_item_id", "Статья расхода"), ("charge_date",
"Дата")])

def create_input_field(self, parent, label, default_text):
    """Создает поле ввода с начальным текстом, который исчезает при фокусе и восстанавливается при
    потере фокуса"""
    var = tk.StringVar()

    # Вставляем начальный текст
    entry = tk.Entry(parent, textvariable=var)
    entry.insert(0, default_text)

    # При фокусе очищаем поле
    def on_focus_in(event):
        if var.get() == default_text:
            var.set('')

    # Если поле теряет фокус и пустое, восстанавливаем текст
    def on_focus_out(event):
        if var.get() == '':
            var.set(default_text)

    entry.bind("<FocusIn>", on_focus_in)
    entry.bind("<FocusOut>", on_focus_out)

    return entry, var

def generic_management_window(self, title, table, columns):
    """Обобщенный метод для управления записями таблиц."""
    window = tk.Toplevel(self.root)
    window.title(title)

    tree = ttk.Treeview(window, columns=[col[0] for col in columns], show="headings")
    for col in columns:
        tree.heading(col[0], text=col[1])
    tree.grid(row=0, column=0, columnspan=3)

    # Функция для обновления таблицы
    def refresh_table():
        try:
            with get_db_connection() as conn:
                cur = conn.cursor()
                for row in tree.get_children():
                    tree.delete(row)
                cur.execute(f"SELECT {'', ' '.join(col[0] for col in columns)} FROM {table}")
                for row in cur.fetchall():
                    tree.insert("", "end", values=row)
        except Exception as e:
            messagebox.showerror("Ошибка", f"Ошибка при обновлении данных: {e}")

    # Если текущий пользователь – это user, то отображаем только таблицу, без дополнительных полей
    if self.current_user_role == 'user':
        refresh_table() # Показываем только таблицу с данными
        return # Для пользователя нет кнопок и текстовых полей, поэтому завершить функцию

    # Для администраторов создаем текстовые поля и кнопки
    entry_vars = {} # Словарь для хранения переменных ввода
    for i, col in enumerate(columns[1:]):
        entry, var = self.create_input_field(window, col[1], f"{col[1]}")
        entry.grid(row=1, column=i)
        entry_vars[col[0]] = var

```

## 5. Пользовательские права

- Приложение поддерживает разграничение прав доступа. Пользователи с ролью “admin” могут добавлять, редактировать и удалять записи, а пользователи с ролью “user” имеют доступ только к просмотру данных.

```
def add_entry():
    """Добавить новую запись в таблицу."""
    if self.current_user_role != 'admin':
        messagebox.showerror("Ошибка", "У вас нет прав для добавления данных.")
    return

    .
    .
    .
    //////////////////////////////////

def delete_entry():
    if self.current_user_role != 'admin':
        messagebox.showerror("Ошибка", "У вас нет прав для удаления данных.")
    return

    .
    .
    .
    //////////////////////////////////

def edit_entry():
    if self.current_user_role != 'admin':
        messagebox.showerror("Ошибка", "У вас нет прав для редактирования данных.")
    return

    .
    .
    .
    //////////////////////////////////

if self.current_user_role == 'admin':
    tk.Button(window, text="Добавить", command=add_entry).grid(row=2, column=0)
    tk.Button(window, text="Удалить", command=delete_entry).grid(row=2, column=1)
    tk.Button(window, text="Редактировать", command=edit_entry).grid(row=2, column=2)

refresh_table()
```

## 6. Работа с таблицами

- Для отображения данных используется компонент `tk.Treeview`.
- Данные извлекаются из базы данных с помощью SQL-запросов, выполняемых через подключение, предоставляемое модулем `database.py`.

## Вывод

Код организован модульно, что упрощает его поддержку и расширение. Модули `auth.py` и `database.py` отвечают за ключевые аспекты работы с базой данных и авторизации, а `main.py` — за взаимодействие с пользователем и основную логику приложения. Такой подход обеспечивает гибкость системы и ее надежность.



## Список литературы

1. PostgreSQL для начинающих: Руководство. 1-е изд. – 2016. – PostgreSQL 9.5. – Электронный ресурс.
2. PostgreSQL для начинающих: Руководство. 2-е изд. – 2016. – PostgreSQL 9.5. – Электронный ресурс.
3. Баруздин, С.А. Основы работы с реляционными базами данных: Учебное пособие. – М.: Лаборатория знаний, 2018. – 256 с.
4. Кузнецов, А.С. Управление базами данных в PostgreSQL: Практическое пособие. – СПб.: Невский Институт, 2020. – 298 с.
5. Орлов, П.С., Новиков, И.В. Python для анализа данных: Руководство по применению. – М.: ДМК Пресс, 2019. – 416 с.