

Министерство образования и науки РФ
Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и кибербезопасности
Высшая школа программной инженерии

КУРСОВОЙ ПРОЕКТ

**Разработка системы автоматизации работы оптовой фирмы
по дисциплине «Базы данных»**

Выполнил:
Студент гр. 5130904/10103



Шульгин И. К.

Проверил:

Юркин В. А.

Оглавление

Введение.....	3
Задание	4
Описание работы.....	6
Подключение к БД	7
Авторизация в приложении.....	8
Главное окно приложения	9
Взаимодействие с таблицами.....	10
Журналы и логирование	11
Информация о пользователе и аналитические данные	12
Отчёты	14
Заключение.....	15
Список литературы	16

Введение

Необходимо реализовать систему для автоматизации оптовой фирмы, используя базу данных PostgreSQL.

Оптовая фирма (далее организация) занимается оптовыми и мелкооптовыми поставками различных товаров в магазины Санкт-Петербурга. Требуется автоматизировать рабочее место менеджера по распределению товаров между двумя складами организации. Два склада несут функционально различную нагрузку в целях уменьшения арендной платы организации. Первый склад находится на территории Санкт-Петербурга и сильно меньше второго, который находится в значительном удалении. Первый склад выполняет функции КЭШа множества самых популярных товаров. Задачей менеджера по распределению является изучение спроса и выделения множества приоритетных товаров, которые будут завезены на первый склад. Менеджер должен иметь доступ к состоянию обоих складов, справочнику товаров и журналу реализации. При появлении заявки на определенный товар менеджер заносит заявку в журнал и списывает необходимое количество товара с первого склада. В случае если на первом складе товара недостаточно – остаток списывается со второго. Такой вариант приносит лишние расходы фирме, так как в этом случае она занимается доставкой товара на территорию заказчика, чтобы не потерять клиента. Завоз товаров на первый склад производится ночью каждого дня в соответствии с приоритетами, выставленными менеджером. Второй склад содержит достаточное число товаров, и его работа выходит за рамки данного проекта.

Задание

База данных должна удовлетворять следующим требованиям:

1. Контроль целостности данных, используя механизм связей
2. Операции модификации групп данных и данных в связанных таблицах должны быть выполнены в рамках транзакций.
3. Логика работы приложения должна контролироваться триггерами. В частности:
 - Триггер должен не позволять списать товар со второго склада при наличии товара на первом
 - Триггер должен контролировать, чтобы вводимая заявка не превышала суммарное количество товара на первом и втором складах
4. Все операции вычисления различных показателей (из требований к клиентскому приложению) должны реализовываться хранимыми процедурами.

Требования к клиентскому приложению:

1. Необходимо реализовать интерфейсы для ввода, модификации и удаления
 - Товаров, включая задание приоритета
 - Заявки, с автоматическим списанием товара со складов
2. В главном окне приложения должен быть реализован журнал менеджера с просмотром количества товаров на складах.
3. Необходимо реализовать возможность просмотра менеджером следующих показателей:
 - Пять самых популярных товаров.
 - Изменение спроса данного товара за некоторый промежуток времени.
 - Графическое отображение изменения спроса заданного товара

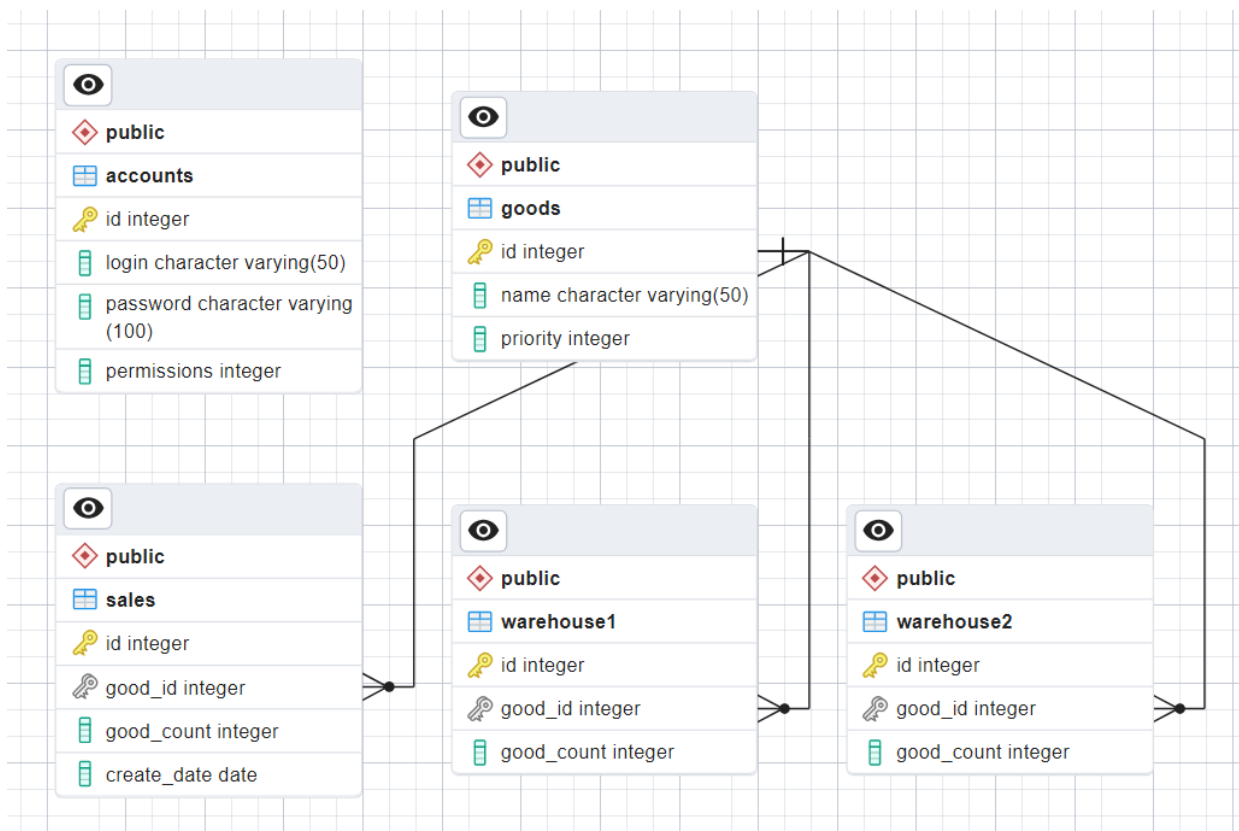


Рис. 1. Диаграмма базы данных

Описание работы

Приложение написано на языке C++ версии 17. Для взаимодействия с базой данных используется библиотека libpqxx. Графический интерфейс создан с помощью библиотеки Dear ImGui и ImPlot (для графиков), а также GLFW для взаимодействия с OpenGL. Также используется библиотека boost для чтения конфигурационного файла.

Взаимодействие с большей частью базы данных (за исключением таблицы accounts) происходит через шаблонный класс Table. Использование шаблона и пакета параметров позволило написать унифицированный класс для взаимодействия с таблицами из базы данных. Структура TableQueries определяет SQL-запросы, которые используют методы при однотипном взаимодействии с БД.

```
template <typename... T>
class Table
{
public:
    using RowType          = std::tuple<int, T...>;
    using RowTypeWithoutId = std::tuple<T...>;

    Table(pqxx::connection& con, const TableQueries&
tableQueries);

    void updateData();

    const std::vector<RowType>& getData() const;
    RowType getById(int id) const;

    template <typename CT>
    RowType getByColumnValue(int cNumber, CT value) const;

    std::string create(T... args);
    std::string modify(int id, T... args);
    std::string deleteById(int id);

    std::string_view getTableName() const;

    void setDataNotMoified();
    bool isDataModified() const;

private:
    pqxx::connection* con_;
    std::unique_ptr<TableQueries> tableQueries_;
    std::vector<RowType> data_{};
    bool dataModified_{};
};
```

Подключение к БД

Перед подключением к БД считываются параметры из конфигурационного файла с помощью функции из библиотеки boost, затем из этих параметров формируется строка для подключения.

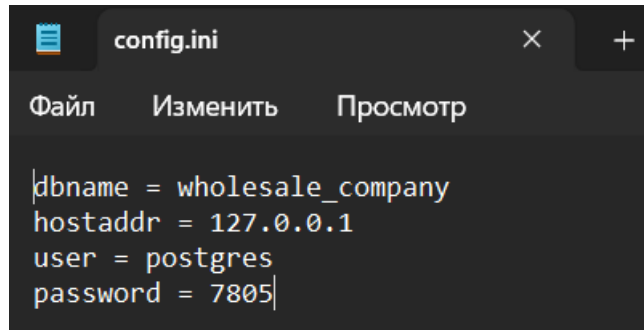


Рис. 2. Конфигурационный файл

```
namespace WHCSys = wholesaleCompanySystem;

po::parsed_options WHCSys::getConfigOptions()
{
    po::options_description desc("All options");
    desc.add_options()
        ("dbname", po::value<std::string>())
        ("hostaddr", po::value<std::string>())
        ("user", po::value<std::string>())
        ("password", po::value<std::string>());

    return po::parse_config_file(INI_FILE_NAME.data(), desc);
}

std::string WHCSys::getConnectionString(const
po::parsed_options& parsedOptions)
{
    std::string connectionString{};
    for (const auto& option : parsedOptions.options)
        connectionString += option.string_key + '=' +
option.value[0] + ' ';
    return connectionString;
}
```

Авторизация в приложении

Для хранения информации о пользователях имеется таблица accounts с полями для логина, пароля и прав доступа. Права доступа представляют собой число, каждый бит которого отвечает за определённую область доступа. Это позволяет задавать различные комбинации прав для пользователей.

Для хэширования паролей используется функционал PostgreSQL, а именно функция crypt с параметром gen_salt('md5'), для использования алгоритма хэширования MD5. MD5 – это 128-битный алгоритм хэширования, который позволяет получить 16-байтное уникальное значение хэш-функции из входных данных любой длины (в моём случае паролей любой длины).

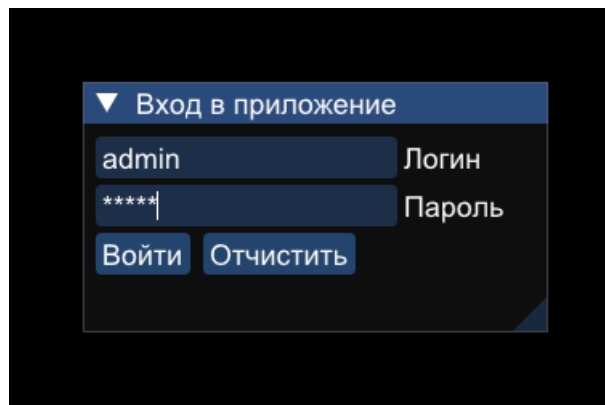


Рис. 3. Окно входа

Имеется главный пользователь – admin, его нельзя удалить, а также изменить ему права (по умолчанию имеет полный набор разрешений). При этом только admin имеет возможность создавать других пользователей и изменять им права доступа через соответствующие пункты в Меню.

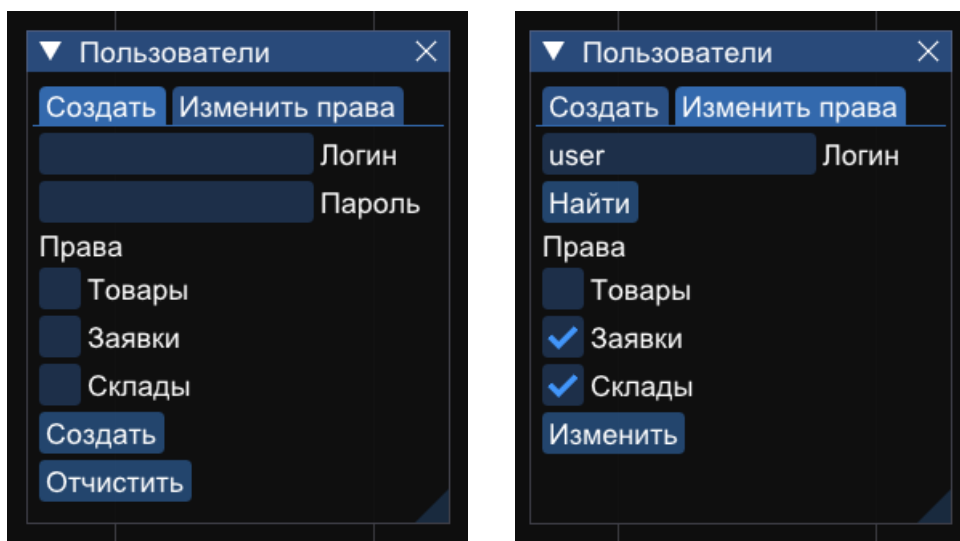


Рис. 4. Возможности admin

Права доступа позволяют или, наоборот, запрещают пользователю взаимодействовать с таблицами.

Главное окно приложения

Главное окно приложения можно условно разделить на три части:

- Взаимодействие с таблицами (слева)
- Журналы таблиц (в центре)
- Информация о пользователе и аналитика (справа)

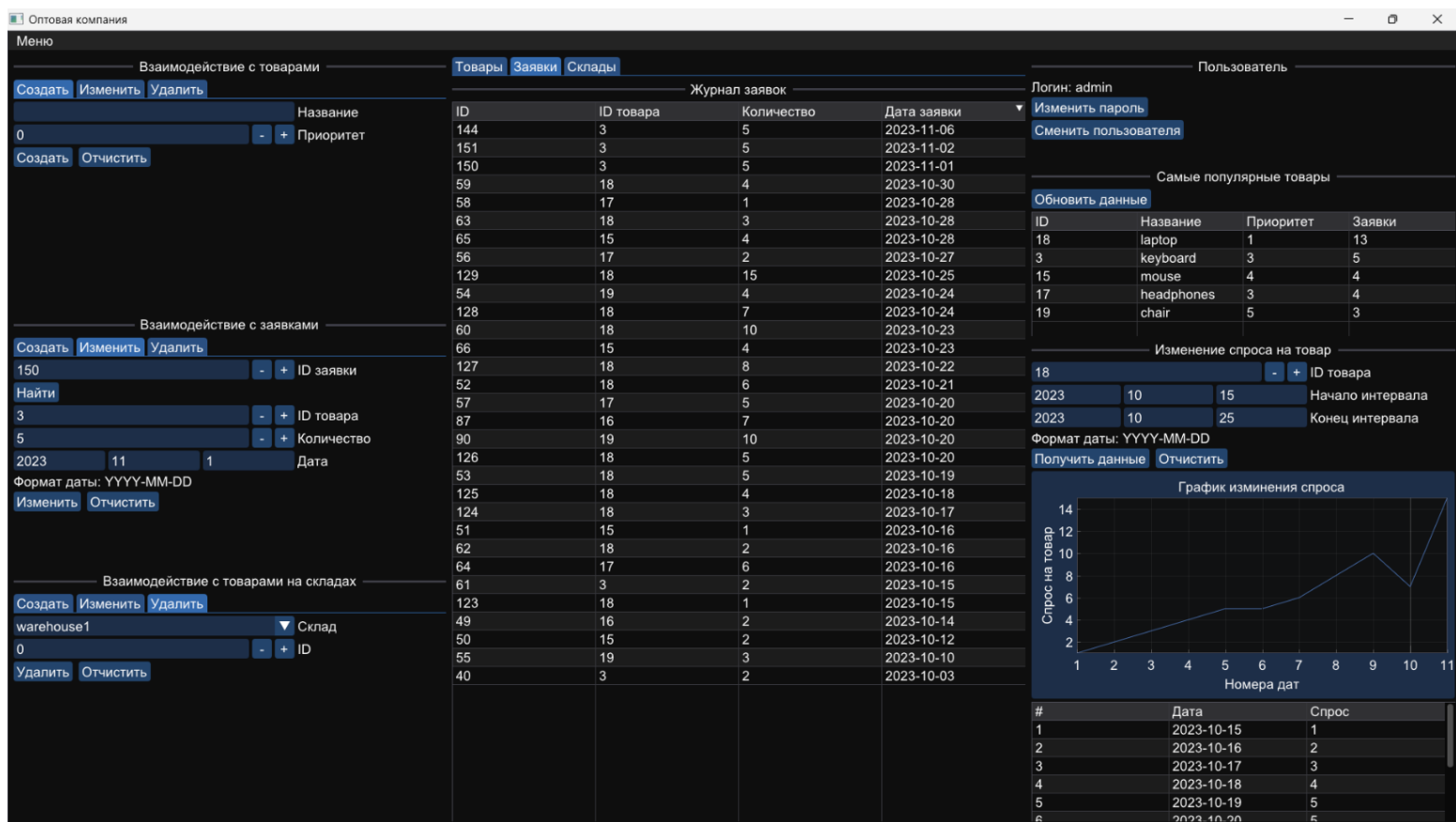


Рис. 5. Главное окно приложения

Взаимодействие с таблицами

При взаимодействии с таблицами можно создавать, изменять и удалять данные о таблицах. Однако эта часть экрана может быть недоступна пользователя из-за отсутствия необходимых прав доступа. При этом в случае ошибки взаимодействия с таблицей действие совершенно не будет, и пользователь получит информацию об ошибке.

Взаимодействие с товарами

Создать Изменить Удалить

3 - + ID товара

Найти

keyboard Название

4 - + Приоритет

Изменить Отчистить

Успешно!

Рис. 6. Успешное взаимодействие

Взаимодействие с товарами

Создать Изменить Удалить

3 - + ID товара

Удалить Отчистить

Ошибка!

ERROR: References still exist
CONTEXT: PL/pgSQL function checkifreferencesexist() line 9 at RAISE

Рис. 7. Вывод информации об ошибке

Взаимодействие с товарами

Создать Изменить Удалить

Название

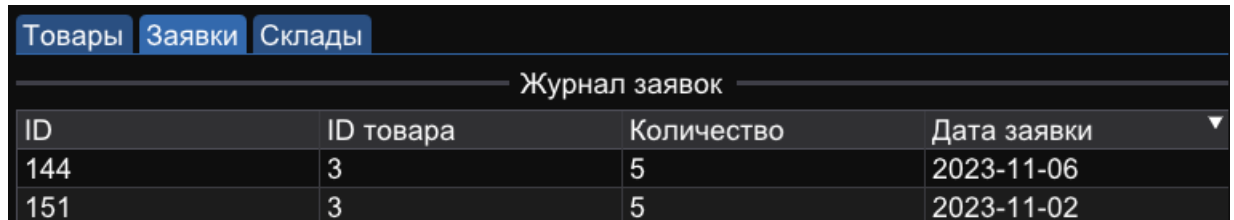
0 - + Приоритет

Создать Отчистить

Рис. 8. Взаимодействие с товарами недоступно

Журналы и логирование

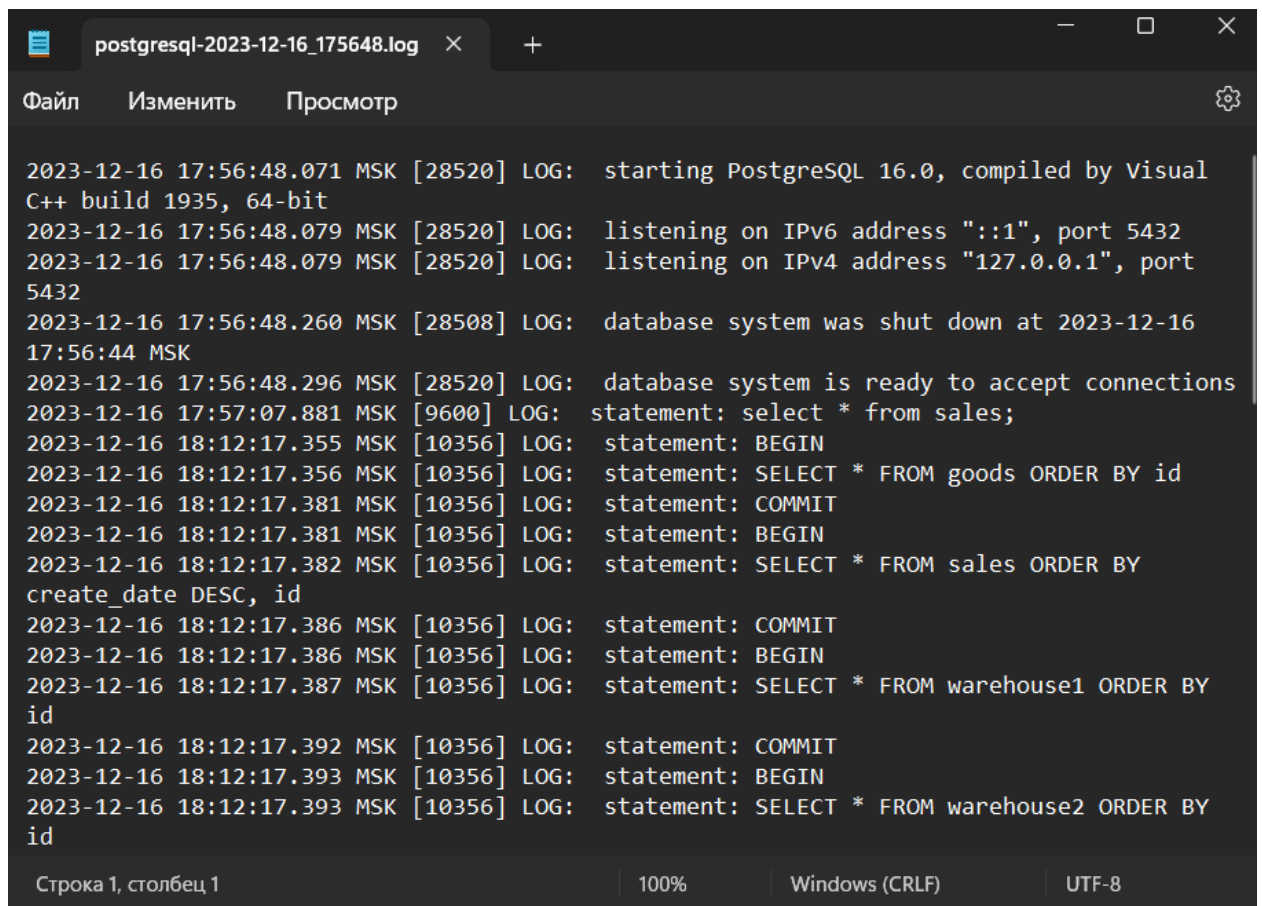
Журналы можно отсортировать по возрастанию и убыванию значений столбцов, также они автоматически обновляются при изменении данных. Переключение между журналами происходит с помощью вкладок сверху.



Журнал заявок			
ID	ID товара	Количество	Дата заявки
144	3	5	2023-11-06
151	3	5	2023-11-02

Рис. 9. Вкладки журналов

Логирование базы данных происходит с помощью средств PostgreSQL. Логи сохраняются в файл с названием 'postgresql-%Y-%m-%d_%H%M%S.log'.



```
2023-12-16 17:56:48.071 MSK [28520] LOG:  starting PostgreSQL 16.0, compiled by Visual
C++ build 1935, 64-bit
2023-12-16 17:56:48.079 MSK [28520] LOG:  listening on IPv6 address "::1", port 5432
2023-12-16 17:56:48.079 MSK [28520] LOG:  listening on IPv4 address "127.0.0.1", port
5432
2023-12-16 17:56:48.260 MSK [28508] LOG:  database system was shut down at 2023-12-16
17:56:44 MSK
2023-12-16 17:56:48.296 MSK [28520] LOG:  database system is ready to accept connections
2023-12-16 17:57:07.881 MSK [9600] LOG:  statement: select * from sales;
2023-12-16 18:12:17.355 MSK [10356] LOG:  statement: BEGIN
2023-12-16 18:12:17.356 MSK [10356] LOG:  statement: SELECT * FROM goods ORDER BY id
2023-12-16 18:12:17.381 MSK [10356] LOG:  statement: COMMIT
2023-12-16 18:12:17.381 MSK [10356] LOG:  statement: BEGIN
2023-12-16 18:12:17.382 MSK [10356] LOG:  statement: SELECT * FROM sales ORDER BY
create_date DESC, id
2023-12-16 18:12:17.386 MSK [10356] LOG:  statement: COMMIT
2023-12-16 18:12:17.386 MSK [10356] LOG:  statement: BEGIN
2023-12-16 18:12:17.387 MSK [10356] LOG:  statement: SELECT * FROM warehouse1 ORDER BY
id
2023-12-16 18:12:17.392 MSK [10356] LOG:  statement: COMMIT
2023-12-16 18:12:17.393 MSK [10356] LOG:  statement: BEGIN
2023-12-16 18:12:17.393 MSK [10356] LOG:  statement: SELECT * FROM warehouse2 ORDER BY
id
```

Рис. 10. Логи базы данных

Информация о пользователе и аналитические данные

В дочернем окне информации о пользователе можно изменить или сменить пользователя. Последнее действие возвращает к окну входа в приложение.

Рис. 11. Изменение пароля

Справа доступна таблица 5 самых популярных товаров. Она создаётся с помощью представления в базе данных и шаблонного класса Table с ограниченными возможностями (например, нельзя изменять данные в ней).

Самые популярные товары			
Обновить данные			
ID	Название	Приоритет	Заявки
18	laptop	1	13
3	keyboard	3	5
15	mouse	4	4
17	headphones	3	4
19	chair	5	3

Рис. 12. Самые популярные товары

Использованное представление:

```
CREATE VIEW PopularGoodsOfMonth AS
SELECT *,
    (
        SELECT COUNT(*)
        FROM sales
        WHERE sales.good_id = goods.id
    ) AS sales_count
FROM goods
ORDER BY sales_count DESC
LIMIT 5;
```

Чуть ниже доступна информация об изменении спроса на товар, которая выдаётся при вводе id товара и исследуемого интервала. Изменение спроса показано как в виде графика, так и в виде соответствующей таблицы. При этом с графиком можно взаимодействовать, например, изменять масштаб.

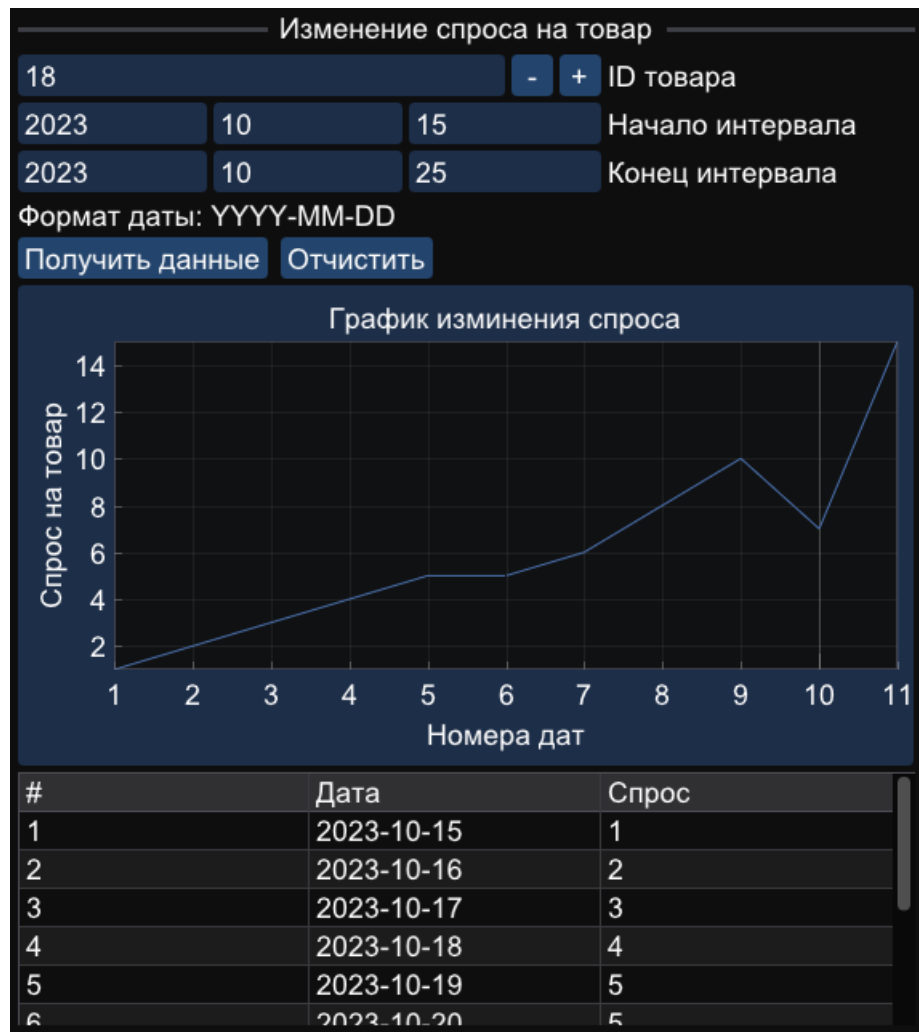


Рис. 13. Изменение спроса на товар

Отчёты

Пользователь имеет возможность формировать отчёты по таблицам в виде csv файла. Для этого есть соответствующий пункт в Меню, при нажатии на который можно выбрать таблицу, отчёт для которой необходимо сформировать.

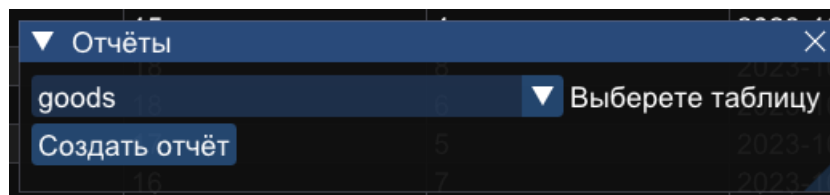


Рис. 14. Окно формирования отчётов

	A	B	C
1	3	keyboard	3
2	15	mouse	4
3	16	monitor	2
4	17	headphone	3
5	18	laptop	1
6	19	chair	5
7	20	arduino	7
8	29	cringe	4

Рис. 15. Сформированная таблица

Заключение

В результате выполнения курсовой работы я научился:

1. Взаимодействовать с базой данных через SQL-запросы: создавать, изменять и удалять данные в таблицах, использовать представления, триггера, функции и т.д.
2. Использовать шаблонные классы и пакеты параметров для оптимизации процесса написания кода.
3. Использовать библиотеки для взаимодействия с БД, чтения конфигурационных параметров.
4. Создавать графический интерфейс немедленного режима с помощью Dear ImGui.

Возникавшие трудности решались путём поиска необходимой информации в документации библиотек и в Интернете.

Список литературы

1. PostgreSQL 16.1 Documentation
2. Dear ImGui wiki (<https://github.com/ocornut/imgui/wiki>)
3. Interactive ImGui Manual
(https://pthom.github.io/imgui_manual_online/manual/imgui_manual.html)
4. Interactive ImPlot Demo
(https://traineq.org/implot_demo/src/implot_demo.html)
5. libpqxx 7.8.1 Documentation
6. Boost Library Documentation
7. Как безопасно хранить пароли с помощью PostgreSQL | Timeweb Cloud
(<https://timeweb.cloud/tutorials/postgresql/kak-bezopasno-hranit-paroli-s-postgresql>)