

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ
ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение
высшего образования

**«Южно-Уральский государственный университет
(национальный исследовательский университет)»**

Высшая школа электроники и компьютерных наук

Кафедра системного программирования

РАБОТА ПРОВЕРЕНА

Рецензент

заведующий кафедрой компьютерной
безопасности и прикладной алгебры

ЧелГУ,

кандидат физ.-мат. наук.

_____ А.Н. Ручай

“ ___ ” _____ 2019 г.

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой, д.ф.-
м.н., профессор

_____ Л.Б. Соколинский

“ ___ ” _____ 2019 г.

**РАЗРАБОТКА НАСТОЛЬНОГО ПРИЛОЖЕНИЯ ДЛЯ
УЧЕТА СОВМЕСТНЫХ ЗАКУПОК**

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 09.03.02.2019.115-005.ВКР

Научный руководитель

к.ф.-м.н., доцент

_____ Т.Ю. Маковецкая

Автор работы,

студент группы КЭ-405

_____ Ю.М. Жидяева

Ученый секретарь

(нормоконтролер)

_____ О.Н. Иванова

“ ___ ” _____ 2019 г.

Челябинск 2019

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ
ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение
высшего образования

**«Южно-Уральский государственный университет
(национальный исследовательский университет)»**

**Высшая школа электроники и компьютерных наук
Кафедра системного программирования**

УТВЕРЖДАЮ

Зав. кафедрой СП

_____ Л.Б. Соколинский

09.02.2019

ЗАДАНИЕ

на выполнение выпускной квалификационной работы бакалавра
студенту группы КЭ-405
Жидяевой Юлии Максимовной,
обучающемуся по направлению
09.03.02 «Информационные системы и технологии»

1. Тема работы (утверждена приказом ректора от 25.03.2019 № 899)

Разработка настольного приложения для учета совместных закупок.

2. Срок сдачи студентом законченной работы: 05.06.2019.

3. Исходные данные к работе

3.1. Лабор В.В. «С#. Создание приложений для Windows»

3.2. О. Н. Евсеева А. Б. Шамшев «Работа с базами данных на языке С#»

4. Перечень подлежащих разработке вопросов

4.1. Изучить существующие методы разработки настольных приложений

4.2. Выполнить анализ требований и разработать внешние спецификации

4.3. Выполнить проектирование настольного приложения и базы данных

4.4. Реализовать приложение для ОС Windows

4.5. Провести тестирование приложения

5. Дата выдачи задания: 09.02.2019.

Научный руководитель

Доцент кафедры СП, кандидат физ.-мат. наук Т.Ю. Маковецкая

Задание принял к исполнению

Ю.М. Жидяева

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ	4
1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ.....	6
1.1. Предметная область.....	6
1.2. Обзор аналогов	6
1.3. Обзор существующий решений.....	9
2. ПРОЕКТИРОВАНИЕ	13
2.1. Функциональные и нефункциональные требования	13
2.2. Диаграмма вариантов использования.....	13
2.3. Проектирование интерфейса.....	15
2.4. Схема базы данных.....	18
3. РЕАЛИЗАЦИЯ СИСТЕМЫ	20
3.1. Инструменты, используемые для реализации	20
3.2. Диаграмма последовательности	21
3.3. Реализация компонентов системы.....	22
4. ТЕСТИРОВАНИЕ	25
ЗАКЛЮЧЕНИЕ	30
СПИСОК ЛИТЕРАТУРЫ	31
ПРИЛОЖЕНИЕ	32

ВВЕДЕНИЕ

Актуальность

В настоящее время пользователи интернета имеют возможность выбора товаров на любой вкус и цвет. В связи с этим, онлайн-шоппинг стал привычным для нас явлением и продолжает набирать популярность по сей день. Самыми быстрорастущими сегментами в онлайн-торговле стали электроника и одежда. И для того чтобы приобрести понравившуюся вещь нужно сделать всего пару кликов.

Одним из самых экономичных способов онлайн-покупок являются совместные покупки (СП). Этот вид шоппинга особенно популярен в России среди городов с населением более миллиона жителей. Главные плюсы совместных покупок – низкая стоимость доставки, скидки на оптовый заказ, а также низкая итоговая стоимость товаров, за счёт малого количество посредников в цепочке.

Чаще всего подобные закупки совершаются через социальные сети. Клиенты связываются с организатором, делают свои заказы согласно с условиями закупки и ожидают доставки своих товаров. Однако, для организатора закупки составляет большую сложность просматривать все данные о заказах и клиентах в социальных сетях, также самостоятельно просчитывать все денежные операции и формировать закупки. Поэтому было принято решение разработать настольное приложение администратора для учета совместных покупок.

Цель и задачи работы

Целью проекта является разработка настольного Windows приложения для учета совместных закупок.

Для реализации данной цели были поставлены следующие задачи.

1. изучить существующие методы разработки настольных приложений;
2. выполнить анализ требований и разработать внешние спецификации;

3. выполнить проектирование настольного приложения и базы данных;
4. реализовать приложение для ОС Windows;
5. провести тестирование приложения.

Структура и объем работы

Работа состоит из введения, четырех глав, заключения, списка литературы и приложения. Объем работы составляет 35 страниц, объем списка литературы – 15 источников, объем приложения – 4 страницы.

Содержание работы

В первой главе производится анализ предметной области и обзор существующих аналогов в данной области. Также здесь приведены методы создания настольных приложений.

Во второй главе «Проектирование» содержит требования к разрабатываемой системе, подробности проектирования интерфейса приложения, спроектирована диаграмма последовательности, приведена схема базы данных.

В третьей главе «Реализация» описаны детали реализации приложения: инструментальные средства разработки, код некоторых функций приложения.

В четвертой главе «Тестирование» результаты функционального и нефункционального тестирования настольного приложения.

В заключении представлены результаты проделанной работы и направление дальнейшего развития.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Предметная область проекта

В данной работе рассматривается создание настольного приложения для администратора совместных закупок с функционалом, направленным на помощь организаторам в формировании закупок, отслеживании статуса оплаты, доставки и расчете денежной сводки. Организатор совместных покупок является посредником между покупателем и поставщиком, в его обязанности входит:

1. поиск поставщиков (магазинов);
2. реклама ассортимента;
3. сбор заказов;
4. формирование закупки;
5. сбор оплаты от покупателей;
6. получение товара;
7. доставка товара покупателю.

1.2. Обзор аналогов

На сегодняшний день существует несколько решений для ведения учета совместных закупок. В основном они представлены в качестве веб-сервисов. Ниже приведены примеры подобных сервисов.

Полка СП

Полка СП — программа для организаторов совместных покупок в социальных сетях Вконтакте и Одноклассники, работающих с комментариями. Позволяет формировать, редактировать и группировать новые заказы. Полка СП предоставляет отчеты по покупателям и товарам в формате таблиц. Приложение реализовано в качестве веб-сервиса. Интерфейс данного сервиса представлен на рис.1.

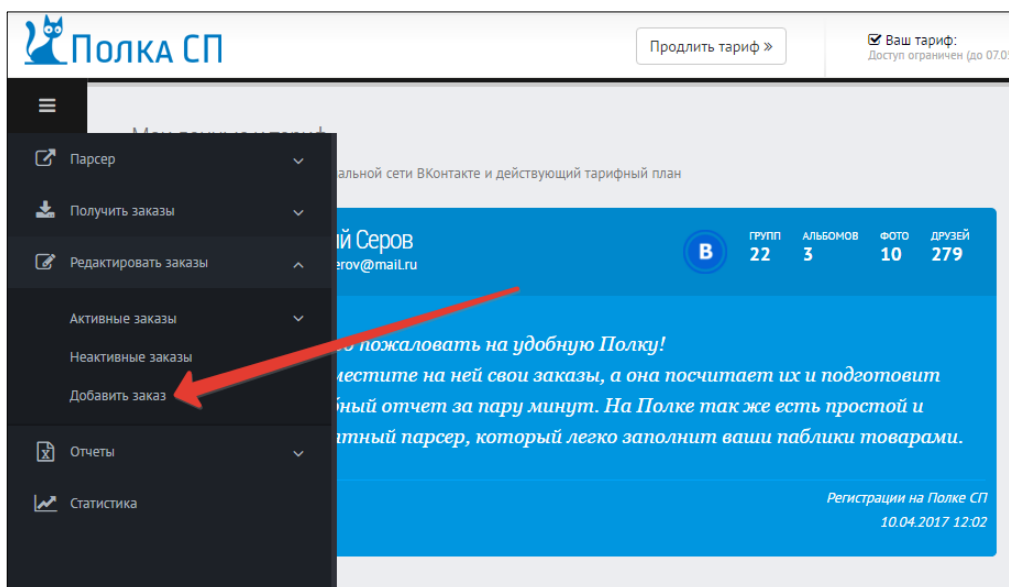


Рис. 1. «Полка СП»

Приложение удобно в использовании и действительно облегчает работу организатору совместных покупок, однако Полка СП работает только в режиме онлайн и доступно при авторизации в соц. сетях Вконтакте и Одноклассники. Приложением платное и для продолжения работы нужно оплачивать тарифный план.

СП Помощник

СП Помощник – веб-сервис, синхронизирующий данные из соц. сети Вконтакте, позволяющий сформировать заказы из комментариев в альбомах группы, оповестить клиента о выставлении счета в личных сообщениях и хранить историю закупок. Выполняет функцию органайзера для ведения закупок. Скриншот веб-сервиса представлен на рис.2.

Данное приложение позволяет формировать закупки и счета, имеет возможность экспортировать данные в таблицу Excel, однако сервис также является платным и не имеет функции анализа данных и отслеживания доставки товара. Также весомый недостаток сервиса СП Помощник - невозможность работы без подключения к соц. сети.

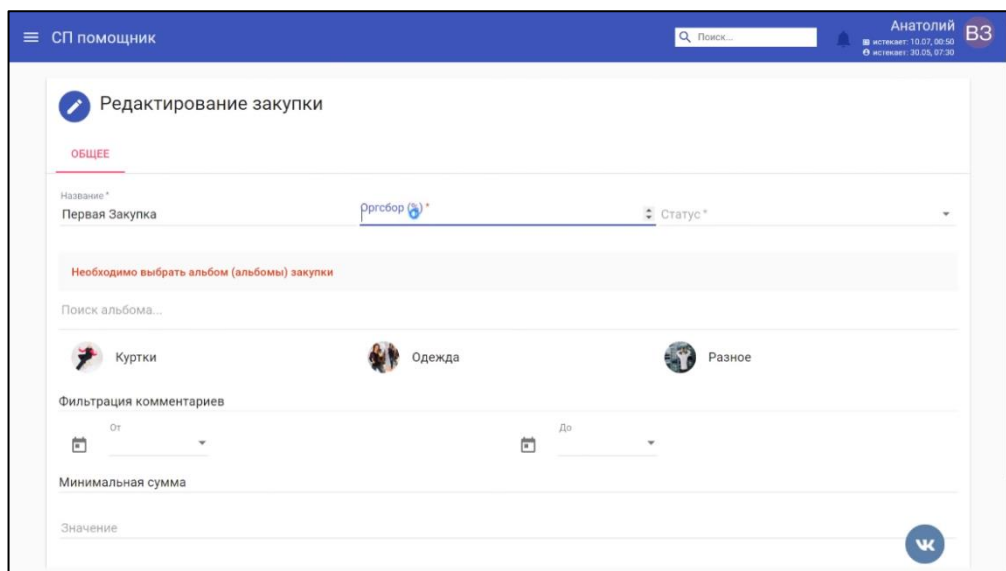


Рис. 2. «СП Помощник»

СП Комфорт

СП Комфорт – программа для организаторов совместных покупок, позволяет осуществлять выгрузку данных о заказанных товарах в таблицу Excel, формирует уведомления для клиентов, счета на основании параметров закупки, имеет возможность распределения заказов по пунктам выдачи. На рис.3 представлен интерфейс данной системы.

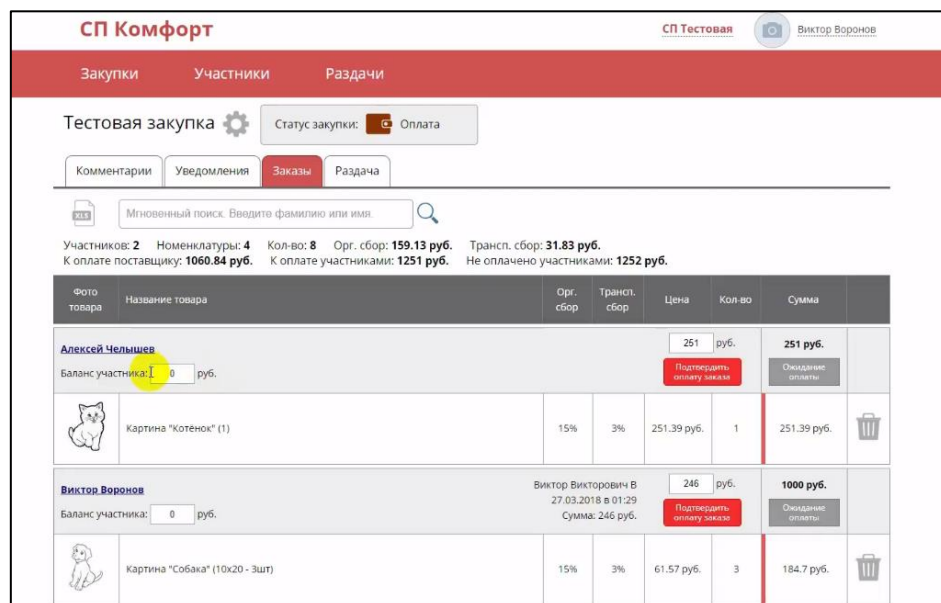


Рис. 3. «СП Комфорт»

СП Комфорт позволяет экономить время организаторов закупок на подсчет суммы всех заказов и позволяет автоматически оповещать клиентов об успешной оплате заказа. Из недостатков этого приложения - отсутствие общей статистики по сформированным и завершенным закупкам.

Вывод

Рассмотрев аналоги можно сделать вывод, что в данной сфере есть конкурентоспособные решения, однако все из предложенных выполнены в качестве платных веб-сервисов, а полного аналога настольного приложения для администратора не существует.

1.3. Обзор существующих решений

Разработка настольных (или десктопных) приложений – это процесс, при котором приложения разрабатываются для настольного компьютера [2]. В данной главе рассмотрим примеры инструментов для реализации настольного приложения под операционные системы Windows.

На сегодняшний день существует множество платформ для создания Windows приложений, проанализируем несколько, самых известных и актуальных:

1. Windows Forms;
2. WPF;
3. Qt;
4. WxWidgets;
5. Swing.

Windows Forms

Windows Forms – это интерфейс программирования приложений (API), отвечающий за графический интерфейс пользователя и являющийся частью платформы *Microsoft .NET Framework*. Классы, реализующие интерфейс программирования приложения, не зависят от языка разработки [4].

В Windows Forms форма – это визуальная поверхность, на которой выводится информация для пользователя. Приложение можно создать с помощью перетаскивания и вставки конструктор Windows Forms в Visual Studio. Также Windows Forms включает широкий набор элементов управления, которые можно добавлять на формы: текстовые поля, кнопки, раскрывающиеся списки, переключатели, веб-страницы [1].

WPF

WPF – система для построения клиентских приложений Windows. В основе WPF лежит векторная система визуализации, которая не зависит от разрешения устройства и которая создана с учётом возможностей современного графического оборудования [3].

Для работы с WPF требуется любой .NET-совместимый язык. WPF предоставляет средства для создания визуального интерфейса, элементы управления, привязку данных, макеты, двухмерную и трёхмерную графику, анимацию, стили, шаблоны, документы, текст, мультимедиа и оформление [9].

Qt

Qt – кроссплатформенный фреймворк для разработки программного обеспечения на языке программирования C++. Существуют также «привязки» ко многим другим языкам программирования:

Python – PyQt;

Ruby – QtRuby;

Java – QtJambi;

PHP – PHP-Qt.

Отличительная особенность этого фреймворка – использование предварительной обработки исходного кода, что позволяет запускать приложения в большинстве ОС. Qt полностью объектно-ориентированный и включает в себя все основные классы, которые требуются при разработке прикладного программного обеспечения, начиная от

элементов графического интерфейса и заканчивая классами для работы с сетью, базами данных и XML.

WxWidgets

WxWidgets (wxWindows) – библиотека инструментов с открытым исходным кодом для разработки кроссплатформенных на уровне исходного кода приложений. Основное применение данной библиотеки является построение графического интерфейса пользователя [10].

WxWidgets позволяет легко реализовать стандартные интерфейсные функции и тем самым сконцентрировать внимание разработчика на функциональных особенностях приложения [6].

Библиотека написана на языке программирования C++, но может подключаться ко множеству других языков, таких, как Ruby, Python и др.

Swing

Swing – библиотека для создания графического интерфейса для программ на языке Java. Данная библиотека содержит ряд графических составляющих, таких как кнопки, поля ввода, таблицы и т.д. [7].

Swing относится к библиотеке классов JFC, которая представляет собой набор библиотек для разработки графических оболочек. К этим библиотекам относятся Java 2D, Accessibility-API, Drag & Drop-API и AWT.

Архитектура Swing разработана таким образом, что разработчики могут изменять внешний вид компонентов вашего приложения и их поведение.

Вывод

В ходе анализа представленных аналогов, были выявлены преимущества и недостатки рассматриваемых систем.

В качестве главных достоинств можно отметить удобство графического интерфейса, возможность сохранения истории покупок, возможность создания покупки и просмотра общего баланса.

Однако, среди недостатков некоторых приложений особенно выделяются невозможность просмотра и анализа баланса организатора, невозможность наблюдения за доставкой товаров и платный тарифный план. Из чего следует вывод, что в разрабатываемом настольном приложении хотелось бы видеть денежную сводку, которая включает в себя подробно описанные расходы и выгоду с закупок, подсчет общего баланса и отслеживание невыплат клиентов. Также приложение должно иметь функцию просмотра статуса доставки товара. Приложение должно работать, не имея доступа к социальным сетям.

Исходя из анализа существующих решений было принято решение разрабатывать настольное приложение при помощи Windows Forms на языке программирования C#.

2. ПРОЕКТИРОВАНИЕ

2.1. Функциональные и нефункциональные требования

В результате анализа предметной области были сформулированы функциональные и нефункциональные требования к приложению.

Настольное приложение должно предоставлять пользователю функциональные возможности.

1. Авторизация.
2. Просмотр списка закупок.
3. Фильтр списков закупок.
4. Формирование новой закупки.
5. Возможность редактировать информацию о товаре в закупке.
6. Возможность редактировать статус закупки.
7. Просмотр списка клиентов.
8. Фильтр списка клиентов.
9. Функция добавления новых клиентов.
10. Возможность редактировать информацию о клиенте.
11. Регистрация клиента.
12. Просмотр денежной сводки.
13. Нефункциональные требования:
14. Настольное приложение должно быть написано на языке программирования C#.
15. Настольное приложение должно функционировать на устройствах с ОС Windows.
16. Настольное приложение должно работать без доступа к социальным сетям.

2.2. Диаграмма вариантов использования

В данной системе существует только один актер, взаимодействующий с приложением, – Администратор, использующий приложение для учета совместных покупок.

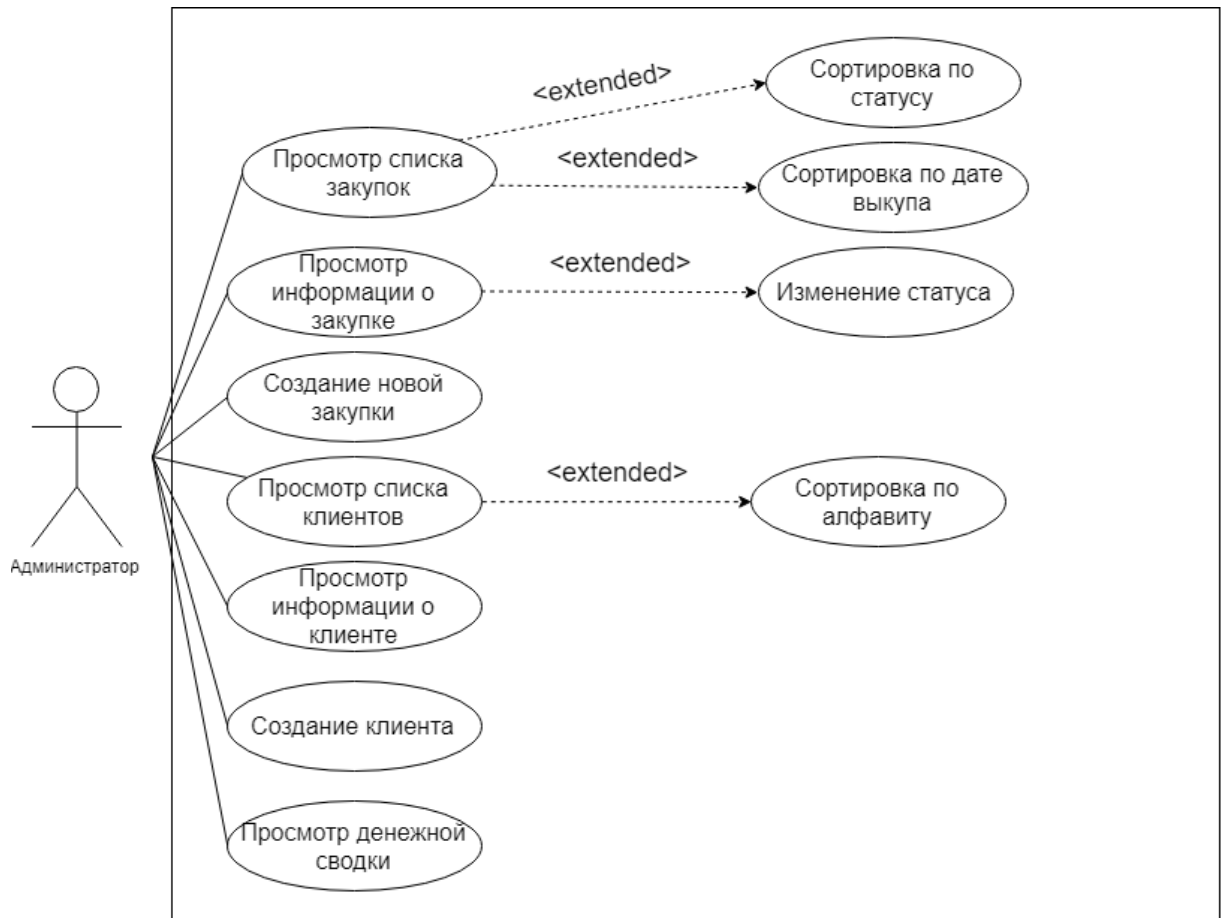


Рис. 4. Диаграмма вариантов использования

В разрабатываемом приложении возможны следующие варианты использования:

1. Просмотр списка закупок – просмотр всех существующих закупок в виде списка.
2. Сортировка по статусу – просмотр списка закупок, отсортированных по статусам: формирующиеся, выкупленные, завершённые.
3. Сортировка по дате выкупа – просмотр списка закупок, отсортированных по дате выкупа закупки.
4. Просмотр информации о закупке - просмотр товаров в закупке, клиентов, участвующих в закупке, статуса доставки и оплаты товара клиентом.

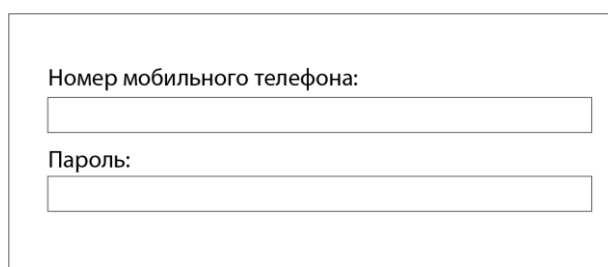
5. Создание новой закупки – формирование новой закупки.
6. Просмотр списка клиентов - просмотр всех клиентов в виде списка
7. Сортировка по алфавиту – просмотр списка клиентов, отсортированных по алфавиту.
8. Просмотр информации о клиенте – просмотр контактов клиента и статуса оплаты его товаров.
9. Авторизация клиента – присвоение логина и пароля клиенту.
10. Просмотр денежной сводки – просмотр баланса по закупкам и общего баланса.

2.3. Проектирование интерфейса

Интерфейс приложения является точкой взаимодействия пользователя с программной системой [5].

Одним из этапов разработки дизайна было создание макетов окон приложения.

Окно 1 «Авторизация» – стартовый экран при запуске, в котором осуществляется вход в приложение. Макет окна 1 представлен на рисунке 5.



Номер мобильного телефона:

Пароль:

Рис. 5. «Авторизация»

Окно 2 «Меню» – окно, появляющееся при верном заполнении формы входа. Меню является основным окном приложения, через которое осуществляется вход во все последующие окна. Макет представлен на рисунке 6.

Текущий пользователь:

Имя:

[Выйти](#) [Изменить](#)

Рис. 6. «Меню»

Окно 3 «Пользователи» – окно, при переходе в которое открывается список клиентов и данные о каждом из них. В данном окне администратор имеет возможность создать нового пользователя, изменить информацию о существующем пользователе и создать для него заказ. Далее представлен макет окна «Пользователи» (рисунок 7).

Создать		Изменить		Создать заказ		Заказы		Обновить	
	Фамилия	Имя	Телефон	Дата регистрации	Администратор				

Рис. 7. «Пользователи»

Окно 4 «Магазины» – окно, переход в которое осуществляется через «Меню». В данном окне отображается список созданных магазинов с ссылкой на сайт. Здесь администратор добавляет новые магазины, также

имеет возможность просмотреть заказы и товары, привязанные к конкретному магазину. Макет окна представлен на рисунке 8.

Создать		Изменить		Товары		Закупки		Обновить	
	Имя	Страна	Сайт						

Рис. 8. «Магазины»

Окно 5 «Закупки» – форма, переход в которую осуществляется с главного экрана «Меню». Открыв это окно, администратор просматривает список существующих закупок с возможностью выборки по видам закупок. Здесь же редактируется и отслеживается статус доставки покупки. Также через данное окно происходит формирование новых закупок и добавление в них новых товаров. Макет окна «Закупки» изображен на рисунке 9.

Создать		Изменить		Создать заказ		Заказы		Товары		Обновить	
	Магазин	Дата выкупа	Цена доставки	Дата доставки	Дата доставки	Дата доставки	Дата доставки	Дата доставки	Дата доставки	Доставлено	

Рис. 9. «Закупки»

Окно 6 «Заказы» – форма, переход в которую выполняется с главного экрана «Меню». В окне «Заказы» отображается таблица со всеми существующими заказами клиентов. В данном окне администратор может отслеживать что осталось доплатить клиенту и что уже оплачено. Также через данное окно происходит редактирование заказов и их создание. Макет окна «Заказы» изображен на рисунке 10.

Создать		Изменить		Обновить							
	Создан	Пользователь	Закупка	Товар	Количество	Итоговая цена	Итоговая цена оплачена	Аванс оплачен	Доставка оплачена	Дата доставки	Доставлено

Рис. 10. «Заказы»

Окно 7 «Денежная сводка» – окно, переход в которое осуществляется через «Меню». Данная форма отражает текущий баланс по разным видам закупок, имеет возможность просмотра баланса за выбранный год, месяц или вид закупки. Позволяет администратору провести анализ расходов и доходов за разные периоды времени. Макет данного окна представлен на рисунке 11.

Год_____		Месяц_____		За всё время		Обновить			
	Год	Месяц	Закупка	Потрачено на товары	Потрачено на доставку	Получено за товары	Получено за доставку	Выгода	

Рис. 11. «Денежная сводка»

2.4. Схема базы данных

Схема базы данных включает в себя описания содержания, структуры и ограничений целостности, используемые для создания и поддержки базы данных. Система управления базами данных (СУБД) использует определения данных в схеме для обеспечения доступа и управления доступом к данным в базе данных [11].

База данных для приложения учета закупок содержит таблицы с данными о закупках, товарах, клиентах и заказах клиентов. Также в базе данных описаны таблицы доставки и магазинов. На схеме продемонстрированы связи для каждой из таблиц и их элементы.

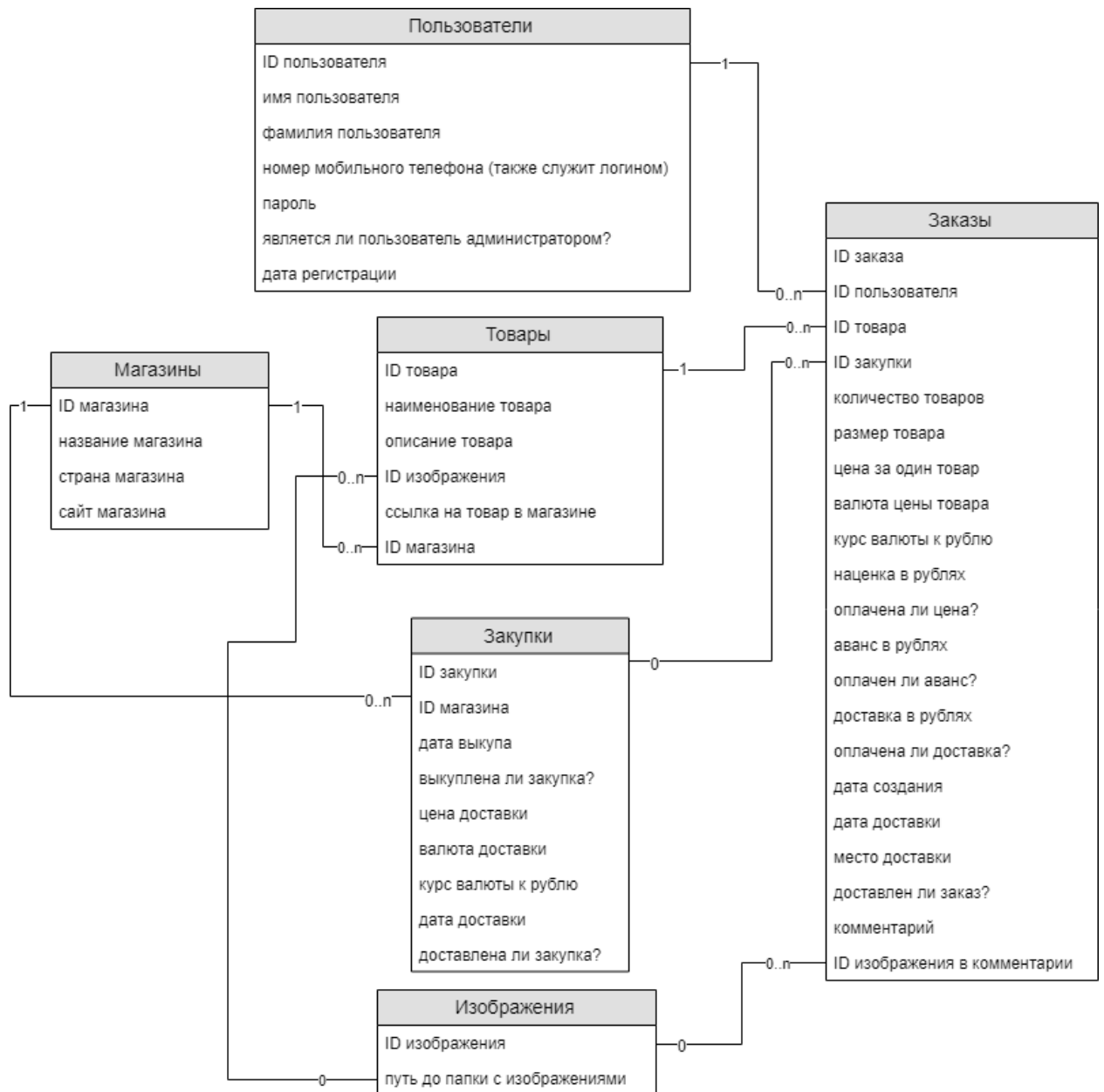


Рис. 12. «Схема базы данных»

Выводы

В данной главе были рассмотрены основные функциональные и нефункциональные требования к разрабатываемой системе, которые были учтены в процессе реализации. В соответствии с требованиями были разработаны диаграмма вариантов использования, интерфейс приложения и схема базы данных.

3. РЕАЛИЗАЦИЯ СИСТЕМЫ

3.1. Инструменты, используемые для реализации

Для разработки настольного приложения был выбран язык программирования C# и среда разработки Visual Studio 2017.

Язык программирования C# – язык, разработанный в 1998-2001 годах командой инженеров компании Microsoft. C# – объектно-ориентированный язык, который является основным языком разработки приложений для платформы Microsoft .NET [3].

Visual Studio – продукт компании Microsoft, представляющий собой интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. С помощью Visual Studio можно разрабатывать консольные приложения, приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения, веб-службы и т.д.

Для реализации интерфейса было принято решение использовать Windows Forms, так как данная технология имеет широкий набор элементов управления и с помощью конструктора в среде разработки Visual Studio позволяет легко создавать приложения. Построение приложения Windows forms осуществляется путем помещения элементов на форму и написания кода для реагирования на действия пользователя.

Основные возможности Windows Forms, необходимые в реализации приложения для учета совместных покупок.

1. Возможность создавать интерактивные пользовательские интерфейсы.
2. Возможность создания форм и элементов управления.
3. Отображение данных и управление ими.
4. Развертывание приложений на клиентских компьютерах.

3.2. Диаграмма последовательности

На рисунке 13 изображена диаграмма последовательности, которая отображает процесс отправки запроса на обработку данных сервером, включающий следующие шаги.

1. Пользователь нажимает на кнопку при взаимодействии с приложением.
2. Приложение формирует запрос и отправляет его на сервер.
3. Сервер принимает запрос.
4. В ответ сервер отправляет массив байт, который анализирует приложение.
5. Проанализировав ответ сервера, приложение выводит данные в форму.
6. Если обнаружена ошибка, приложение уведомляет об этом пользователя в диалоговом окне.

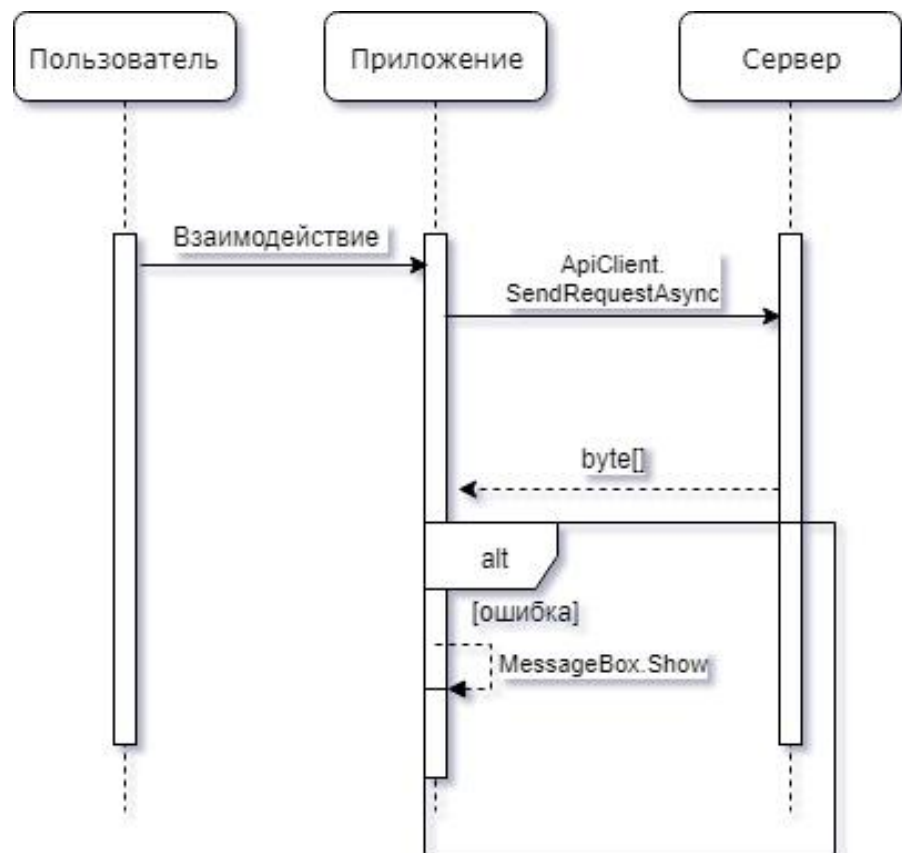


Рис. 13. Процесс отправки запроса на сервер

3.3. Реализация компонентов системы

Реализация некоторых основных функций приложения представлена в листингах.

В листинге 1 представлен код формы входа в приложение. LoginForm – главная форма приложения, так как через нее открывается все последующие формы и осуществляется авторизация административная.

Листинг 1. Форма входа в приложение

```
partial class LoginForm
{
    private System.ComponentModel.IContainer components = null;

    protected override void Dispose(bool disposing)
    {
        if (disposing && (components != null))
        {
            components.Dispose();
        }
        base.Dispose(disposing);
    }
}
```

В листинге 2 представлена отправка запроса на сервер. ApiClient.SendRequestAsync – ключевой метод системы, так как через него проходят все запросы на сервер приложения.

Листинг 2. Отправка запроса на сервер

```
public static async Task<byte[]> SendRequestAsync(
    string path,
    string method,
    string token = null,
    HttpContent data = null)
{
    var requestUri = new Uri(new Uri(ApiConst.API_ENDPOINT),
path);
    var request = new HttpRequestMessage(new
HttpMethod(method), requestUri);
    if (token != null)
    {
        request.Headers.Authorization = new
AuthenticationHeaderValue("Bearer", token);
    }
    if (method.ToLower() != "get" || method.ToLower() !=
"delete")
    {
        request.Content = data;
    }

    var client = new HttpClient();
    var response = await client.SendAsync(request);
    if (response.IsSuccessStatusCode)
    {

```

```

        var result = await
response.Content.ReadAsByteArrayAsync();
        return result;
    }
    else
    {
        var result = await
response.Content.ReadAsStringAsync();
        var apiError =
JsonConvert.DeserializeObject<ApiError>(result);
        apiError.Status = response.StatusCode;
        throw new ApiException(apiError);
    }
}

```

В листинге 3 представлена реализация создания нового пользователя. Класс `NewUserForm` позволяет администратору приложения добавлять в форму «Клиенты» новых пользователей и присваивать им логин и пароль.

Листинг 3. Создание нового пользователя

```

partial class NewUserForm
{
    private System.ComponentModel.IContainer components = null;

    protected override void Dispose(bool disposing)
    {
        if (disposing && (components != null))
        {
            components.Dispose();
        }
        base.Dispose(disposing);
    }
}

```

В листинге 4 представлен код создания нового магазина. При формировании закупок администратор выбирает из какого магазина она будет осуществляться. В данном листинге изображена реализация формы создания нового магазина.

Листинг 4. Создание нового магазина

```

partial class NewShopForm
{
    private System.ComponentModel.IContainer components = null;

    protected override void Dispose(bool disposing)
    {

```

```

        if (disposing && (components != null))
        {
            components.Dispose();
        }
        base.Dispose(disposing);
    }

```

В листинге 5 представлена реализация получения закупки по ID. При просмотре списка администратор выбирает одну из существующих закупок для внесения изменений. В случае ошибки приложение оповещает администратора посредством диалогового окна.

Листинг 5. Получение закупки

```

    public static async Task<Order> GetByIdAsync(string id, string token)
    {
        try
        {
            var result = await
                ApiClient.SendRequestAsync($"/api/v0/orders/{id}", "GET", token);
            return
                JObject.Parse(Encoding.UTF8.GetString(result))["data"].ToObject<Order>();
        }
        catch (ApiException e)
        {
            if (e.ApiError.Status == HttpStatusCode.NotFound)
            {
                throw new Exception("заказ не найден!");
            }
            throw e;
        }
        catch (Exception e)
        {
            throw e;
        }
    }

```

Реализация расчета денежной сводки по закупкам и заказам осуществляется на сервере, который в свою очередь хранится на облачной платформе Heroku. Осуществляется расчет через SQL запрос к базе данных приложения, который возвращает баланс, отсортированный по виду закупки или дате. С реализацией расчета данной статистики можно ознакомиться в приложении 1.

4.ТЕСТИРОВАНИЕ

Для проведения тестирования системы была создана тестовая база данных, включающая около 30 записей.

Для тестирования системы был выбран метод функционального тестирования.

Функциональное тестирование – это один из ключевых видов тестирования программного обеспечения, при котором определяется качество реализуемости системы. Главная задача функционального тестирования – проверить способность программного продукта в определенных условиях решить задачи необходимые пользователю [15].

Часть разработанных тестов, проверяющих функциональность приложения, приведен в таблице 1.

Табл. Функциональное тестирование приложения

№	Название теста	Шаги	Ожидаемый результат	Тест пройден?
1	Отображение «LoginForm»	Запустить приложение	Отображение окна авторизации	Да
2	Вход в приложение	Ввести логин и пароль администратора	Переход в «Меню»	Да
3	Изменение данных администратора	Нажать на кнопку изменить в окне «Меню»	Отображение окна «Изменить пользователя»	Да

Продолжение табл.

№	Название теста	Шаги	Ожидаемый результат	Тест пройден?
4	Просмотр списка клиентов	Нажать на кнопку «Пользователи» в окне «Меню»	Отображение списка клиентов в окне «Пользователи»	Да
5	Создание нового пользователя	Нажать на кнопку «Создать» в окне «Пользователи»	Форма создания нового пользователя	Да
6	Просмотр списка магазинов	Нажать на кнопку «Магазины» в окне «Меню»	Отображение списка магазинов в окне «Магазины»	Да
7	Создание нового магазина	Нажать на кнопку «Создать» в окне «Магазины»	Форма создания нового пользователя	Да
8	Просмотр списка закупок	Нажать на кнопку «Закупки» в окне «Меню»	Отображение списка закупок в окне «Закупки»	Да
9	Создание новой закупки	Нажать на кнопку «Создать» в окне «Закупки»	Форма создания новой закупки	Да

Продолжение табл.

№	Название теста	Шаги	Ожидаемый результат	Тест пройден?
10	Изменение данных закупки	Нажать на кнопку «Изменить» в окне «Закупки»	Форма изменения текущей закупки	Да
11	Просмотр списка заказов	Нажать на кнопку «Заказы» в окне «Меню»	Отображение списка заказов в окне «Заказы»	Да
12	Создание нового заказа	Нажать на кнопку «Создать» в окне «Заказы»	Форма создания нового заказа	Да
13	Изменение данных заказа	Нажать на кнопку «Изменить» в окне «Заказы»	Форма изменения текущего заказа	Да
14	Просмотр списка товаров	Нажать на кнопку товары в окне «Магазины»	Окно со списком товаров выбранного магазина	Да
15	Создание нового товара	Нажать на кнопку «Создать» в окне «Товары»	Форма создания нового товара	Да

Продолжение табл.

№	Название теста	Шаги	Ожидаемый результат	Тест пройден?
16	Изменение информации о товаре	Нажать на кнопку «Изменить» в окне «Товары»	Форма изменения выбранного товара	Да
17	Просмотр денежной сводки	Нажать на кнопку «Денежная сводка» в окне «Меню»	Отображение окна со статистикой	Да
18	Сортировка баланса по дате	В окне «Денежная сводка» выбрать необходимый год и месяц	Отображение статистики, отсортированной по дате	Да
19	Сортировка баланса по виду закупки	В окне «Денежная сводка» выбрать необходимый вид закупки	Отображение статистики по текущему виду закупки	Да
20	Просмотр статистики за всё время	В окне «Денежная сводка» нажать на кнопку «За всё время»	Отображение расходов и выгоды за все время использования приложения	Да

Вывод

В результате проведенного тестирования было установлено соответствие разработанной системы исходным функциональным требованиям. Все тесты пройдены успешно.

ЗАКЛЮЧЕНИЕ

В рамках выпускной квалификационной работы было реализовано настольное Windows приложение для учета совместных закупок. Объем кода составил 4100 строк. Для достижения данной цели были решены следующие задачи:

1. анализ предметной области проекта
2. изучение методов существующих решений в разработке настольных приложений
3. анализ требований к разрабатываемой системе
4. проектирование интерфейса настольного приложения
5. реализация приложения
6. тестирование разработанной системы

ЛИТЕРАТУРА

1. Hacking PostgreSQL. Курс для разработчиков СУБД. [Электронный ресурс] URL: <https://postgrespro.ru/education/courses/hacking> (дата обращения: 26.04.2019).
2. Rob Miles R. C# Programming Yellow. Edition 8.2 (cheese). – Rob Miles, 2016. – 216 p.
3. Администрирование PostgreSQL. Базовый курс. [Электронный ресурс] URL: <https://postgrespro.ru/education/courses/DBA1> (дата обращения: 05.04.2019).
4. Документация СУБД PostgreSQL. [Электронный ресурс] URL: <https://postgrespro.ru/docs> (дата обращения: 11.05.2019).
5. Евсеева О. Н., Шамшев А. Б. Работа с базами данных на языке C#. – СПб.:БХВ-Петербург, 2011. – 696 с.
6. Кариев Ч.А. Создание Windows-приложений на основе Visual C#. 2-е изд. – М.: НОУ "Интуит", 2016. - 978 с.
7. Константайн Л., Локвуд Л. Разработка программного обеспечения. - Пер. с англ. – СПб.: Питер, 2004. – 592с.
8. Лабор В.В. C#. Создание приложений для Windows. – Минск: Харвест, 2003. – 385 с.
9. Лузанов П., Рогов Е., Лёвшин И. PostgreSQL для начинающих. - Postgres Professional, 2017. – 118 с.
10. Петцольд Ч. Программирование для Microsoft Windows на C#. В 2-х томах. Том 1. / Пер. с англ. – М.: Издательско-торговый дом «Русская Редакция», 2002. – 576 с.: ил.
11. Пугачев С., Шериев А., Кичинский К. Разработка приложений для Windows 8 на языке C# – СПб.: БХВ-Петербург, 2013. – 416 с.
12. Спольски Д.Х. Лучшие примеры разработки ПО. Сборник статей. – Пер. с англ. - СПб.: Питер, 2007. – 208с.

13. Стиллмен Э., Грин Дж. Изучаем С#. 3-е изд. – СПб.: Питер, 2014. – 816 с.
14. Тестирование программного обеспечения – основные понятия и определения. [Электронный ресурс] URL: <http://www.protesting.ru/testing/> (дата обращения: 12.04.2019).
15. Шилдт Г. С# 4.0. Полное руководство. Пер. с англ. –М.: ООО "И.Д. Вильямс", 2011. – 1056 с.

ПРИЛОЖЕНИЕ

Ниже представлена реализация SQL запроса, который осуществляет расчет денежной сводки для администратора приложения по учету закупок.

```
import {celebrate} from "celebrate";
import Router from "express-promise-router";
import {QueryTypes} from "sequelize";
import {accessLevel, Joi, Sequelize} from "src";
const router = Router();
export default router;
router.use(
  accessLevel("admin"),
  celebrate({
    query: Joi.object({
      tz: Joi.pgInterval().default("0"),
    }),
  }),
  async (req, res) => {
    // language=PostgreSQL
    const result = await Sequelize.instance.query<{
      year: number | null;
      month: number | null;
      id: string | null;
      spent_products: string | null;
      spent_delivery: string | null;
      received_products: string | null;
      received_delivery: string | null;
      profit: string | null;
      redemption: Date | null;
      redemption_status: boolean | null;
      delivered_status: boolean | null;
      shop_id: string | null;
      shop_name: string | null;
      shop_country: string | null;
      shop_site: string | null;
    }>
    WITH temp_orders AS (SELECT orders.purchase_id,
                                SUM(CASE
                                  WHEN orders.price IS NULL THEN 0
                                  ELSE orders.price END *

```

```

CASE
WHEN orders.exchange_rate IS NULL THEN 0
                                ELSE orders.exchange_rate
END *
                                orders.amount) AS
spent_products,
SUM(orders.delivery) AS spent_delivery,
SUM(CASE
WHEN orders.price_paid THEN
CASE
WHEN orders.price IS NULL THEN 0
ELSE orders.price END *
CASE
WHEN orders.exchange_rate IS NULL THEN 0
ELSE orders.exchange_rate END *orders.amount +
orders.markup
WHEN orders.advance_paid THEN orders.advance
ELSE 0 END) AS received_products,
SUM(CASE
WHEN orders.delivery_paid THEN orders.delivery
ELSE 0 END) AS received_delivery
FROM orders
GROUP BY orders.purchase_id),
temp_purchases AS (SELECT EXTRACT(YEAR FROM
purchases.redemption AT TIME ZONE $1) AS year,
EXTRACT(MONTH FROM
purchases.redemption AT TIME ZONE $1) AS month,
purchases.id,
SUM(CASE
WHEN temp_orders.spent_products IS NULL THEN 0
ELSE temp_orders.spent_products
END) AS spent_products,
SUM(CASE
WHEN temp_orders.spent_delivery IS NULL THEN 0
ELSE temp_orders.spent_delivery END +
CASE
WHEN purchases.delivery IS NULL THEN 0
ELSE purchases.delivery END *
CASE
WHEN purchases.exchange_rate IS NULL THEN 0
ELSE purchases.exchange_rate
END) AS spent_delivery,

```

```

SUM(CASE
    WHEN temp_orders.received_products IS NULL THEN 0
    ELSE temp_orders.received_products
END) AS received_products,
SUM(CASE
    WHEN temp_orders.received_delivery IS NULL THEN 0
    ELSE temp_orders.received_delivery
END) AS received_delivery
FROM purchases
LEFT JOIN temp_orders ON purchases.id =
temp_orders.purchase_id
GROUP BY year, month, purchases.id),
temp_purchases_cube AS (SELECT temp_purchases.year,
    temp_purchases.month,
    temp_purchases.id,
SUM(temp_purchases.spent_products) AS
spent_products,
SUM(temp_purchases.spent_delivery) AS
spent_delivery,
SUM(temp_purchases.received_products) AS
received_products,
SUM(temp_purchases.received_delivery) AS
received_delivery,
SUM(temp_purchases.received_products
+ temp_purchases.received_delivery -temp_purchases.spent_products -
temp_purchases.spent_delivery) AS profit
FROM temp_purchases
temp_purchases_cube AS (SELECT temp_purchases.year,
    temp_purchases.month,
    temp_purchases.id,
SUM(temp_purchases.spent_products) AS
spent_products,
(temp_purchases.spent_delivery) AS
spent_delivery,
SUM(temp_purchases.received_products) AS
received_products,
SUM(temp_purchases.received_delivery) AS
received_delivery,
SUM(temp_purchases.received_products +
temp_purchases.received_delivery -
temp_purchases.spent_products -

```

```

temp_purchases.spent_delivery) AS
profit
FROM temp_purchases
GROUP BY ROLLUP (temp_purchases.year,
temp_purchases.month, temp_purchases.id))
SELECT temp_purchases_cube.year,
temp_purchases_cube.month,
temp_purchases_cube.id,
temp_purchases_cube.spent_products,
temp_purchases_cube.spent_delivery,
temp_purchases_cube.received_products,
temp_purchases_cube.received_delivery,
temp_purchases_cube.profit,
purchases.redemption,
purchases.redemption_status,
purchases.delivered_status,
purchases.shop_id,
shops.name AS shop_name,
shops.country AS shop_country,
shops.site AS shop_site
FROM temp_purchases_cube
LEFT JOIN purchases ON temp_purchases_cube.id =
purchases.id
LEFT JOIN shops ON purchases.shop_id = shops.id
ORDER BY temp_purchases_cube.year DESC NULLS FIRST,
temp_purchases_cube.month DESC NULLS FIRST,
purchases.redemption DESC NULLS FIRST;`,
{bind: [req.query.tz],
type: QueryTypes.SELECT,
},
);
res.status(200).json({
data: result.map((row) => ({
deliveredStatus: row.delivered_status,
id: row.id,
month: row.month,
profit: row.profit,
receivedDelivery: row.received_delivery,
receivedProducts: row.received_products,
redemption: row.redemption,
redemptionStatus: row.redemption_status,
shop:

```

```
row.shop_id === null
  ? null
  : {country: row.shop_country,
     id: row.shop_id,
     name: row.shop_name,
     site: row.shop_site,
     },shopId: row.shop_id,
    spentDelivery: row.spent_delivery,
    spentProducts: row.spent_products,
    year: row.year,
  })),
});
},
);
```