

Министерство науки и высшего образования Российской Федерации  
Федеральное государственное автономное образовательное учреждение  
высшего образования «Южно-Уральский государственный университет»  
(национальный исследовательский университет)  
Высшая школа экономики и управления  
Кафедра «Цифровая экономика и информационные технологии»

РАБОТА ПРОВЕРЕНА

Рецензент, директор  
ООО «УралГидроМаш»

\_\_\_\_\_ А.Р. Хакимов  
«\_\_\_» \_\_\_\_\_ 2022 г.

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой, д.э.н.  
доцент

\_\_\_\_\_ Т.А. Худякова  
«\_\_\_» \_\_\_\_\_ 2022 г.

Разработка приложения для расчета стоимости ремонта  
помещений

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА  
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ  
ЮУрГУ– 09.03.03.2022.123.ПЗ ВКР

Руководитель работы  
к.т.н, доцент

\_\_\_\_\_ В.А. Конов  
«\_\_\_» \_\_\_\_\_ 2022 г.

Автор работы  
студент группы ЭУ–431

\_\_\_\_\_ М.В. Петров  
«\_\_\_» \_\_\_\_\_ 2022 г.

Нормоконтролёр,  
ст. преподаватель

\_\_\_\_\_ Н.В. Тихонова  
«\_\_\_» \_\_\_\_\_ 2022 г.

Челябинск 2022

## АННОТАЦИЯ

Петров М.В. Разработка приложения для расчета стоимости ремонта помещений. – Челябинск: ЮУрГУ, ЭУ-431, 86 с., 49 ил., 5 табл., библиографический список – 42 наим., 3 прил.

Выпускная квалификационная работа на тему «Разработка приложения для расчета стоимости ремонта помещений» выполнена с целью проектирования и разработки десктопного приложения «CASTOPARSER».

В работе рассматриваются вопросы разработки десктопного приложения «CASTOPARSER» с использованием реляционных баз данных и высокоуровневых языков программирования. Для решения поставленной задачи производится анализ предметной области, рассматриваются способы возможности разработки, производится проектирование и описываются непосредственно результаты разработки десктопного приложения «CASTOPARSER».

В качестве результата работы предоставляется разработанное десктопное приложение, готовая к тестированию.

## ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ .....	4
1 ПОДГОТОВКА К РЕАЛИЗАЦИИ ПРИЛОЖЕНИЯ.....	6
1.1 ПОСТАНОВКА ЗАДАЧИ И ВЫБОР ИНСТРУМЕНТОВ ДЛЯ РЕАЛИЗАЦИИ ПРОГРАММЫ.....	6
1.2 НАСТРОЙКА VPN.....	9
1.3 АНАЛИЗ ИМЕЮЩИХСЯ СИСТЕМ .....	12
2 РАЗРАБОКА СТРУКТУРЫ ПРОГРАМММЫ .....	25
2.1 ДИАГРАММА ПОТОКОВ ДАННЫХ (DFD) .....	25
2.1.1 Моделирование .....	25
2.1.2 Процессы, потоки данных и хранилища данных .....	28
2.2 БАЗА ДАННЫХ СЕРВЕРНОЙ И ПОЛЬЗОВАТЕЛЬСКОЙ ЧАСТЕЙ .....	31
2.2.1 Таблицы и представления.....	31
2.2.2 Хранимые процедуры и триггеры.....	36
2.3 КЛАССЫ И ИНТЕРФЕЙСЫ.....	40
3 РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЯ «CASTOPARSER».....	46
3.1 MDI-ФОРМА.....	47
3.2 ФОРМА «HABRAPARSER» .....	50
3.3 ФОРМА «ARRAGEMENT».....	55
3.4 ФОРМА «CONSTRUCTION ESTIMATE» .....	65
ЗАКЛЮЧЕНИЕ .....	79
БИБЛИОГРАФИЧЕСКИЙ СПИСОК .....	82
ПРИЛОЖЕНИЯ .....	87
ПРИЛОЖЕНИЕ А .....	87
ПРИЛОЖЕНИЕ Б .....	90
ПРИЛОЖЕНИЕ В .....	94

## ВВЕДЕНИЕ

В связи с развитием интернетом, всё большую популярность приобретают сервисы и приложения, позволяющие пользователю, экономя время и денежные средства, приобретать товары и услуги онлайн.

Люди меньше ходят по магазинам в поисках необходимых ресурсов, они желают получить товар или услугу, не выходя из дома.

Безопасное и удобное жильё – это одна из первых и важных потребностей человека. Обустройство жилого помещения зачастую требует большое количество денежных вложений, а также затрачивает время и усилия.

Рынок программных продуктов, которые предоставляют пользователю, различное разнообразие строительных материалов, инструментов и оборудования, актуальные и выгодные цены, постоянно растёт. Всё больше появляются приложения, борющиеся за своих клиентов. Они каждым разом обновляются новым функционалом, тем самым их интерфейс становится узконаправлен не на обычных пользователей-любителей, а на профессиональных квалифицированных специалистов.

Избыточный и сложный для понимания функционал отталкивает обычных пользователей. Зачастую им необходимо получить разнообразие товаров, выгодную цену, подсчитанную стоимость и возможность выгрузить в Excel или Word, тем самым предоставив им упрощенную версию строительной сметы.

Цель выпускной квалификационной работы – создание приложения «CASTOPARSER», которое предоставит пользователю:

- Выбор строительных материалов, инструментов и оборудования как для покупки, так и для приобретения в аренду;
- Возможность сортировать стройматериалы;
- Возможность делать поиск по записям;
- Фильтрацию по различным категориям;

- Движение цены строительных материалов по графику для того, чтобы понимать на основе неё выгоду или невыгоду покупки в определённые дни;
- Подсчёт суммы на основе введенного количества;
- Занесение выбранных записей в строительную смету;
- Итоги по занесённым записям,
- Возможность редактировать смету;
- Возможность очищать смету её от записей;
- Вывод полученные результаты в Word и Excel.

Всё вышеописанное доступно для пользователя, для того чтобы это работало, основная задача приложения «CASTOPARSER» заключается в создании парсера, который будет собирать необходимые данные с разных источников.

В качестве основных данных будут записаны следующие критерии:

- Наименования строительных материалов, оборудования и инструментов;
- Стоимость;
- Характеристики и спецификации;
- Фотографии.

Выше описанные критерии должны храниться в базе данных, расположенной на серверной части приложения. Эти данные с помощью парсера будут обновляться в определённые дни.

Парсер будет запускаться администратором, у которого будет доступ к серверу.

В данной выпускной квалификационной работе была реализована программа «CASTOPARSER» по сбору данных строительных материалов с интернет-магазинов «Castorama» и «Hilti». Полученные данные записываются и хранятся на локальном сервере для их дальнейшего отображения и анализа. Во избежание блокировки с сервера магазина из-за частых запросов была использована программа OpenVPN.

## 1 ПОДГОТОВКА К РЕАЛИЗАЦИИ ПРИЛОЖЕНИЯ

### 1.1 Постановка задачи и выбор инструментов для реализации программы

Требуется создать программу для сбора данных строительных материалов и оборудования. Источником данных должны служить интернет ресурсы и сайты по продажам строительных материалов и инструментов, а также сдача в аренду строительного оборудования и помещений.

Полученные сведения следует занести в базу данных для хранения, редактирования и дальнейшего использования. Отобразить имеющиеся записи пользователю в виде таблицы. Таблица должна содержать наименование товара, характеристики, категории и его стоимость, иметь возможность сортировки столбцов по возрастанию и убыванию, фильтрацию по отдельным критериям и поиск по записям.

К таблице должна быть привязана область с фотографией выбранного товара или оборудования, поле с описанием характеристик, а также поля и кнопки для занесения выбранных товаров или оборудования в строительную смету с автоматическим подсчётом итоговой суммы.

В качестве дополнения следует построить график движения цены, в котором будут отображаться само движение цены и средняя стоимость товара или оборудования.

Необходимо составить строительную смету [21] по отдельным таблицам, для покупки и аренды отдельно, и общую. Общая смета должна иметь возможность вывода добавленных ранее записей в Excel и Word. Отдельная смета для аренды и покупки помимо вывода в Excel и Word, должна давать пользователю возможность очищать таблицу, удалять записи и редактировать отдельные записи, а именно менять количество товара или оборудования и срок аренды оборудования или помещения. Всё это должно суммироваться для каждой сметы, а также высчитывать общую итоговую сумму, которая тоже должна выводиться в Excel и Word.

Алгоритм работы программы:

- Приложение с помощью парсинга [29] по сайтам подключенных строительных интернет-магазинов считывает и записывает имеющиеся материалы в базу данных (название магазина, наименование строительного материала или оборудования, параметры (вес, высота, ширина, длина и так далее), фото (если имеется));
- Пользователь в приложении выбирает категорию работы;
- Программа представляет пользователю материалы для выбранного вида ремонта;
- Пользователь из представленного списка выбирает подходящие материалы;
- Алгоритм анализирует желаемые материалы, подсчитывает их необходимое количество и производит расчёт итоговой суммы, после которого данные записываются в строительную смету;
- Пользователю выдаётся табличный отчёт в виде списка материалов, названия магазинов, цены за единицу, количества, срок аренды и итоговой суммы.

Для разработки программы «CASTOPARSER» следует определиться с выбором инструментов для реализации программы, а также взять два источника в роли интернет-магазинов, которые предоставляют строительные материалы, инструменты и оборудования как для продажи, так и для предоставления их в аренду.

Средой разработки приложения будет служить Visual Studio 2022 [38]. Она отличается своей профессиональностью и хорошо подходит для создания приложений на основе Windows Forms [14]. Используется версия Visual Studio 2022 Community.

Средой для работы с базой данных серверной части будет служить SQL Server Management Studio [40]. У неё удобный интерфейс и понятная древовидная архитектура проекта [3].

Средой для работы с базой данных пользовательской части будет служить Access 2016. Простой и понятный инструмент для работы с базами данных, который будет располагаться на персональном компьютере пользователя и хранить в себе таблицы строительной сметы.

Для составления диаграммы потоков [10] будет использована программа BPwin AllFusion Process Modeler [33], которая отличается своей простотой и низкими системными требованиями, позволяет производить декомпозицию диаграммы.

Во время работы программы, а именно парсинга, возникнет необходимость использования средств VPN [41], которые позволяют обходить запрет на доступ к сайту со стороны сервера, на котором расположен интернет-магазин. Для этого хорошо подходит программа OpenVPN [27], которая удобна в управлении и имеет бесплатный пробный период в течение одного месяца и обладает реферальной системой привлечения клиентов.

В качестве интернет-магазинов строительных материалов и оборудования были использованы известные на территории РФ сайты магазинов «Castorama» и «Hilti».

«Castorama» предоставляет различные категории строительных материалов и инструментов. Проанализировав несколько сайтов, выбор был остановлен на данном варианте, так как он хорошо подходит для демонстрации и примера, потому что имеет меньшую вероятность закрытия доступа к серверу магазина во время парсинга.

«Hilti» предоставляет различные виды инструментов, оборудования и помещений для сдачи в аренду. Выбор пал на него, так как на данном сайте есть разнообразие профессионального оборудования и скорость парсинга в данном сайте высока, так как сайты, предоставляющие аренду инструментов, имеют меньшую нагрузку, чем сайты по продажам.



## 1.2 Настройка VPN

Чтобы во время парсинга не произошла блокировка доступа к сайту со стороны сервера из-за частых запросов, необходимо установить VPN, который позволит менять адрес сети.

VPN – это обобщённое название технологий, позволяющих обеспечить одно или несколько сетевых соединений поверх другой сети, например, Интернет.

Для данной работы следует скачать и настроить программу OpenVPN. Данная программа обеспечит надёжное подключение к технологиям VPN, даст возможность избежать блокировки со стороны сервера сайта и позволит безопасно собрать необходимые данные с сайтов о стоительных материалах и оборудований, подлежащих как для продажи, так и для сдачи в аренду.

Использование в OpenVPN стандартных протоколов TCP [17] и UDP [18] позволяет ему стать альтернативой IPsec [36] в ситуациях, когда Интернет-провайдер блокирует некоторые VPN-протоколы.

OpenVPN необязательно отключать, программа автоматически делает переподключение через определённое время. Для переподключения необходимо от 20 до 30 секунд.

На рисунке 1.1 представлен процесс подготовки и настройки OpenVPN.

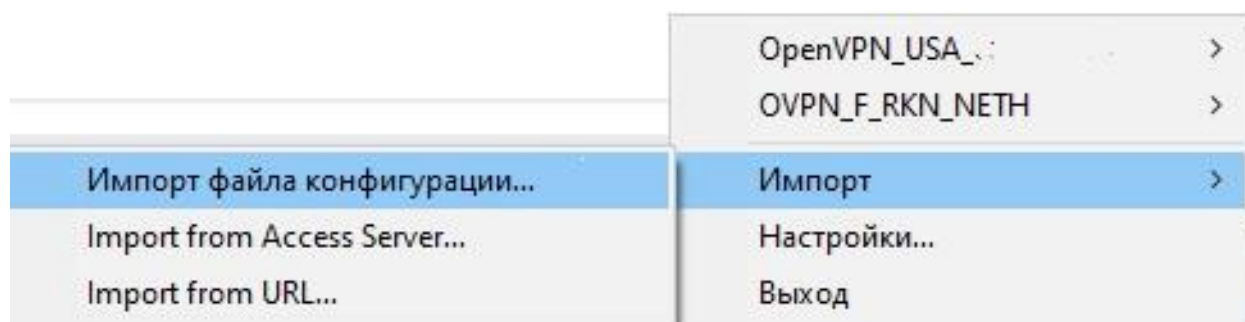


Рисунок 1.1 – Импорт файла конфигурации

После установки и запуска OpenVPN нужно импортировать файл конфигурации. Это делается только при первом подключении. Файл конфигурации можно найти и скачать на разных интернет-ресурсах.

Файл конфигурации представляет из себя набор необходимых команд и настроек, которые нужны для подключения к серверу. На выбор доступны сервера

из таких стран, как Казахстан, Германия, Швеция, Нидерланды, США, Турция, Чехия, Финляндия.

Помимо файла конфигурации, необходим логин с паролем, позволяющий войти на выбранный сервер. За определённую плату можно приобрести подписку к выбранному серверу и пользоваться OpenVPN.

Процесс подключения к серверу изображён на рисунке 1.2.

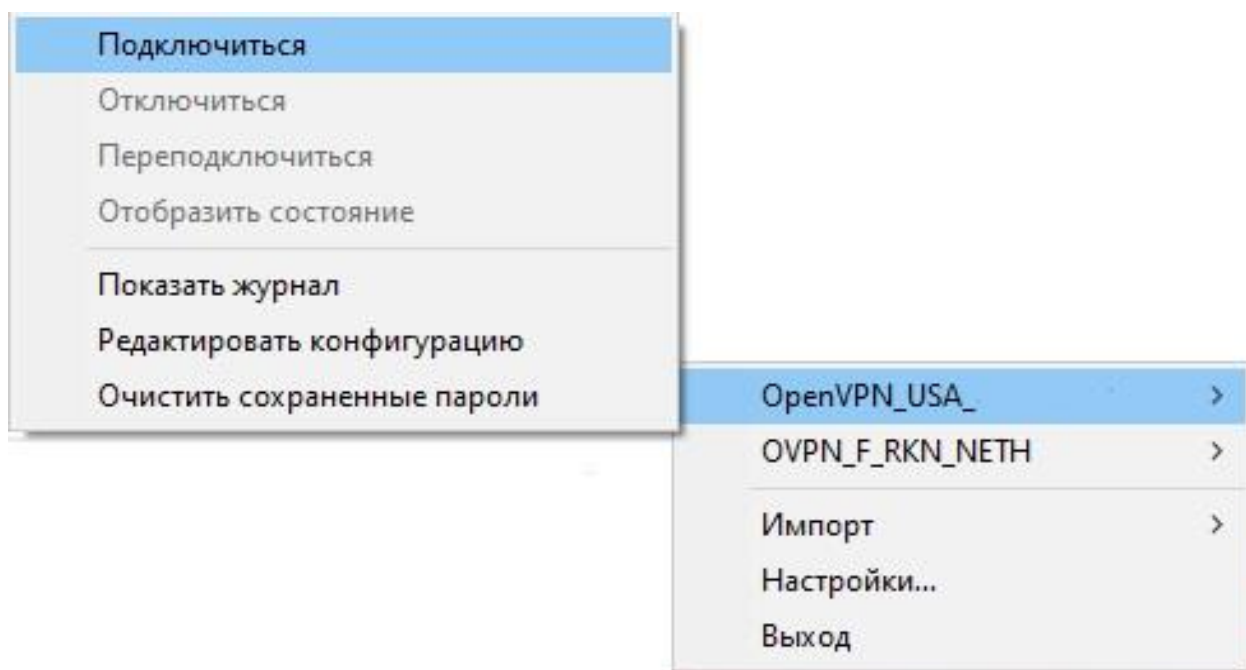


Рисунок 1.2 – Подключение к серверу

После импорта файла конфигурации станет доступным нужный сервер. В данном случае это «OpenVPN\_USA\_...». Процесс ввода пароля показан на рисунке 1.3.

При первом подключении необходимо ввести логин и пароль, позволяющие подключиться к выбранному серверу и имеющие временное ограничение в виде подписки. Цены подписки зависят от сервиса, который предоставляет данную услугу. Чтобы повторно не вводить логин и пароль можно поставить галочку в пункте «Запомнить», тогда при повторном подключении программа пропустит шаг авторизации.

Программа время от времени автоматически делает переподключение и меняет адрес сети, это зависит от времени использования OpenVPN и трафика, который проходит через данную программу.

На рисунке 1.4 показано удачное подключение.

Во время работы программа OpenVPN автоматически сворачивается в трей. Следует через определённый промежуток времени проверять состояние подключения. Зелёный экран значка программы OpenVPN, расположенный в трее, означает, что с подключением всё в порядке, жёлтый – подключение к серверу не установлено.

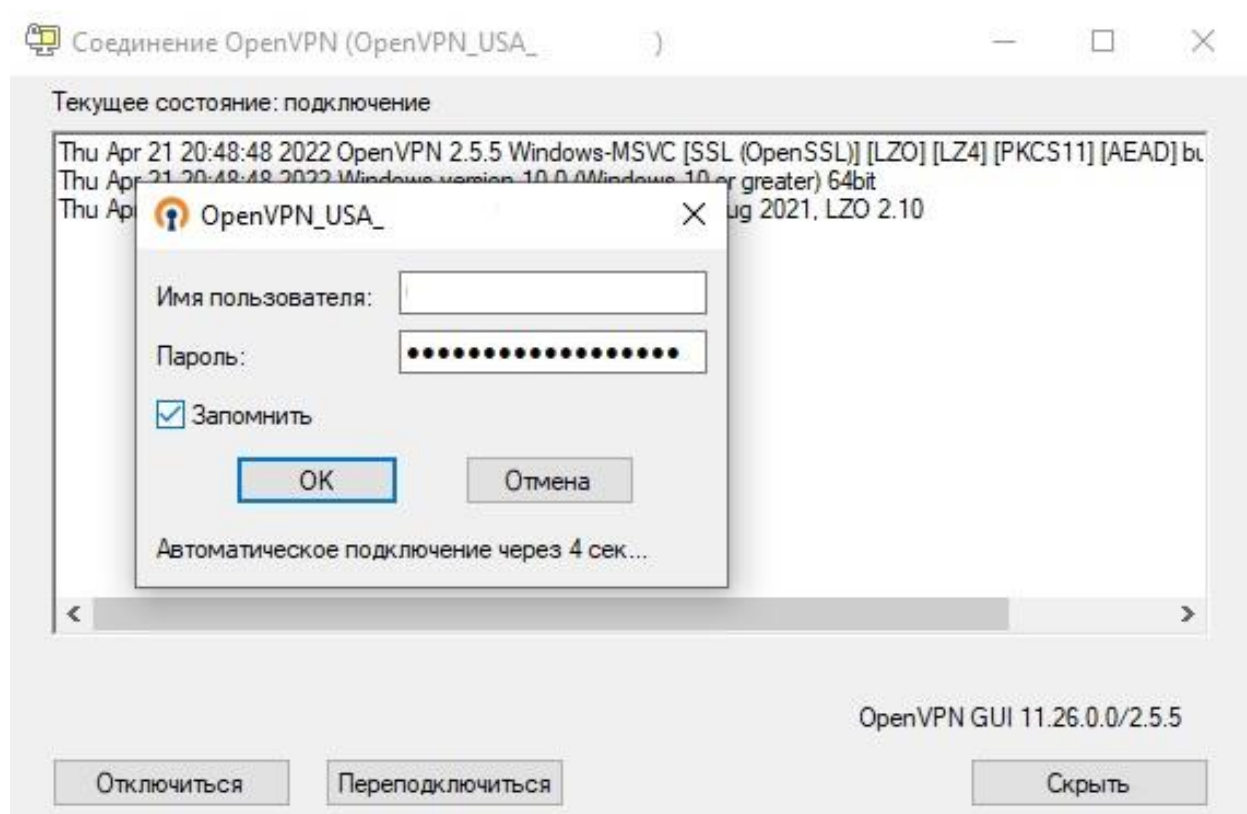


Рисунок 1.3 – Ввод логина и пароля



Рисунок 1.4 – Удачное подключение OpenVPN

### 1.3 Анализ имеющихся систем

Для сравнения представленной идеи программы с уже имеющимися на рынке приложениями был проведён анализ, целью которого является выявление наличия конкурентных систем, их спецификаций, достоинств и недостатков.

Для данного анализа было подобрано несколько приложений с бесплатным и платным (демоверсии) доступом.

Список анализируемых систем:

- Оптим-РемСтрой;
- Дизайн Интерьера 3D;
- ГРАНД-Смета;
- АванСмета;
- АванСмета;
- Смета+;
- WinСмета;
- Smeta.ru;
- «Сметный калькулятор».

Оптим-РемСтрой позволяет создавать многоуровневые сметы, а также отслеживать процесс ремонта и оплату каждого этапа. Софт поддерживает перенос информации из Excel, так что пользователю не придется заново вводить замеры и формировать отчеты с нуля. Программа обладает большим набором полезных функций: можно создавать формы КС-2 и КС-3, устанавливать скидки или надбавки на каждый раздел, отслеживать складские операции. Также присутствует автоматическое заполнение, которое базируется на введенных данных. Денежный подсчет можно вести в нескольких валютах. Оптим-РемСтрой доступен в стандартной десктопной и сетевой версии [13].

Интерфейс программы «Оптим-РемСтрой» показан на рисунке 1.5.

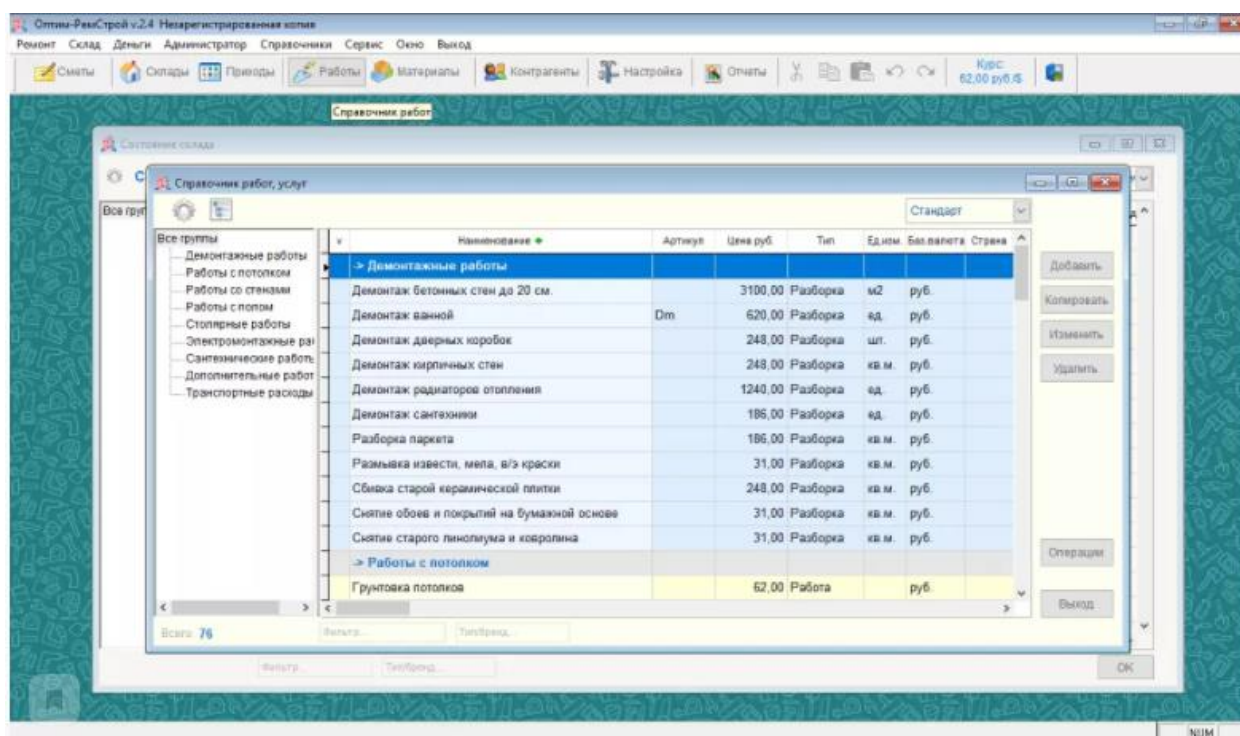


Рисунок 1.5 – Оптим-РемСтрой

Плюсы:

- Неограниченное количество создаваемых проектов;
- Отслеживание остатков.

Минусы:

- В пробной версии доступно только 10 позиций.

Дизайн Интерьера 3D совмещает в себе сметный калькулятор и функции для проектирования помещения. В приложении можно составить смету, вручную указав размеры или воспользовавшись библиотекой типовых планировок. Программа позволяет включать в бюджет дополнительные услуги – электромонтажные, отделочные, прочее. Также есть встроенный справочник, в которых можно составлять план и добавлять комментарии для монтажных и демонтажных работ. Процесс разбит на пошаговую инструкцию, так что освоение будет лёгким даже новичкам [2].

Интерфейс программы «Дизайн Интерьера 3D» показан на рисунке 1.6.

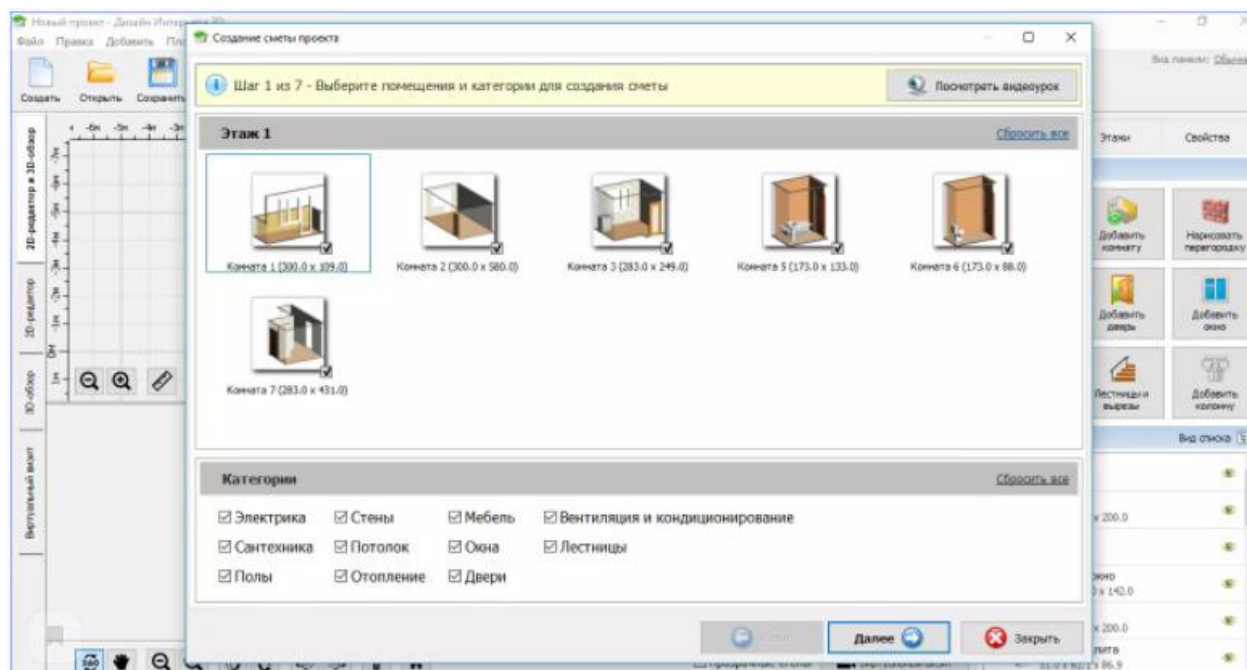


Рисунок 1.6 – Дизайн Интерьера 3D

Плюсы:

- Поддержка русского языка;
- Библиотека типовых помещений;
- Коллекция материалов;
- Создание 3D модели объекта.

Минусы:

- Пробный период длится 10 дней.

ГРАНД-Смета подходит для планирования общего бюджета строительства и включает в себя справочную библиотеку с документацией и словарями. С ее помощью можно проверить правильность расчетов, отследить расход материалов за определенный временной промежуток. ГРАНД-Смета отличается удобным управлением, понятным интерфейсом в стиле Microsoft Word и доступна полностью на русском языке. Стоит отметить, что программа не предоставляет пробного периода для ознакомления [16].

Интерфейс программы «ГРАНД-Смета» показан на рисунке 1.7.



Гранд-Смета - Локальная смета (1)

Файл

Главная

Вставка

Документ

Форматы

Ресурсы

Выполнение

Выделение

Фильтр

Операции

Данные

Проверка расценок

Проверка текущих цен

Проверка индексов

Проверка НР и СП

Экспертиза

Упорядочить данные

Сортировать данные

Перенумеровать позиции

Специальное удаление позиций

Групповые операции

Макросы

Объединение смет

Загрузка индексов

Обновить расценки

Регион

База

Объекты

Локальная смета (1)

Поиск

№ п.п.	Обозначение	Наименование	Ед. изм.	Количество		Всего	Стоимость единицы				Общая стоимость				ТЗ	
				На единицу	Всего		основ. з.п.	эксп. накл.	з.п. нек.	материалы	Всего	основ. з.п.	эксп. накл.	з.п. нек.	материалы	На единицу
Раздел 1. Фундаменты																
1	ФЕР06-01-001-01 Приказ Минстроя Р... от 30.01.14 №31/пр	Устройство бетонной подготовки	100 м3 бетона, бутобетона	0,6 (3*4*5(проект)) / 100	58 585,02	1 404,00	1 590,53	243,00	55 590,49	35 151	842	954	146	33 355	180	108
2	ФСЦЛ-401-0061 Приказ Минстроя России от 24.01.17 №41/пр	Бетон тяжелый, крупность заполнителя 20 мм, класс В3,5 (М50)	м3	-61,2 Ф1.p1	520,00				520,00	-31 824				-31 824	0	0
3	ФСЦЛ-401-0063 Приказ Минстроя России от 24.01.17 №41/пр	Бетон тяжелый, крупность заполнителя 20 мм, класс В7,5 (М100)	м3	61,2 Ф1.p1	550,00				550,00	33 660				33 660	0	0
Ведомость ресурсов по разделу 1 "Фундаменты"																
Итого прямые затраты по разделу в базисных ценах										36 987	842	954	146	35 191		108
Итого прямые затраты по разделу с учетом индексов, в текущих ценах										132 490	7 174	4 913	752	120 403		108
Накладные расходы										6 341						
Сметная прибыль										3 487						
Итого по разделу 1 Фундаменты										142 318,00						108
Раздел 2. Полы																
4	ФЕРр57-2-2 Приказ Минстроя Р... от 30.01.14 №31/пр	Разборка покрытий полов: из плиток поливинилхлоридная	100 м2 покрытия	0,2 20 / 100	280,88	276,82	4,06	1,76	56	55	1			35,49	7,1	
5	ФЕРр57-2-3 Приказ Минстроя Р... от 30.01.14 №31/пр	Разборка покрытий полов: из керамических плиток	100 м2 покрытия	0,4 40 / 100	641,00	595,99	45,01	19,44	256	238	18	8		69,87	27,95	
6	ФЕР11-01-011-03 Приказ Минстроя Р... от 30.01.14 №31/пр	Устройство стяжки: бетонных толщиной 20 мм	100 м2 стяжки	0,6 Ф2+Ф3	1 591,66	317,07	42,05	17,15	1 232,54	955	190	25	10	740	40,65	24,39

ГЭСН, ФЕР-2001 (в ред.2014г. с Изм.1-3) Базовый федеральный район

Баз.-индексный расчет

Итого: 154 918,00р.

Сообщений: 0

Рисунок 1.7 – ГРАНД-Смета

Плюсы:

- Многопользовательский режим;
- Встроенная таблица актуальных цен;
- Перенос в Word, Excel, OpenOffice.

Минусы:

- Бывают ошибки при расчёте грузовых перевозок;
- Для запуска нужно установить дополнительные драйвера;
- Частые обновления могут мешать работе приложения;
- Нет пробного периода.

АванСмета позволяет рассчитать стоимость и количество требуемого для строительства или ремонта материала. Кроме этого, можно составить трудовое соглашение и акты о выполненной работе и создать 3D-модель ремонтируемого помещения. Софт включает в себя библиотеку справочных материалов, которые помогут не ошибиться при ремонте и отвечают законодательным требованиям проведения ремонта и строительства. АванСмета доступна в десктопной и сетевой

версии. Вы можете опробовать программу для составления сметы бесплатно в течение 14 дней [1].

Интерфейс программы «АванСмета» показан на рисунке 1.8.

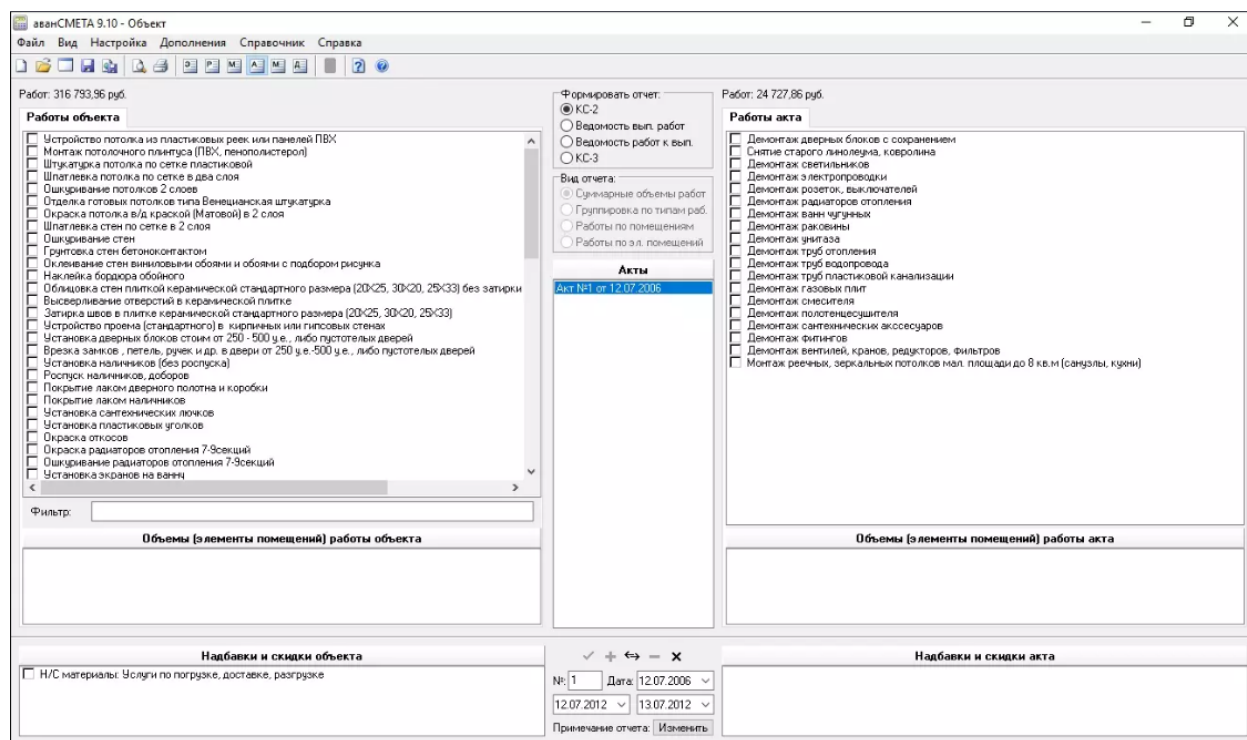


Рисунок 1.8 – АванСмета

Плюсы:

- Простое управление, подходит новичкам;
- Создание виртуальной модели помещения.

Минусы:

- Подходит только для небольших расчетов.

Смета+ поддерживается буквально всеми версиями Windows, начиная с 95, и подходит для 32 и 64-bit. Софт можно использовать для различного рода калькуляций, учета выполненных действий и расчета процента прибыли. Также он позволяет хранить информацию о клиентах и подсчитывать НДС для всего проекта сразу или отдельных услуг. При необходимости в программе можно провести переиндексацию с учетом скидок. Доступна библиотека с формами актов, утвержденными Госкомитетом РФ [19].

Интерфейс программы «Смета+» показан на рисунке 1.9.



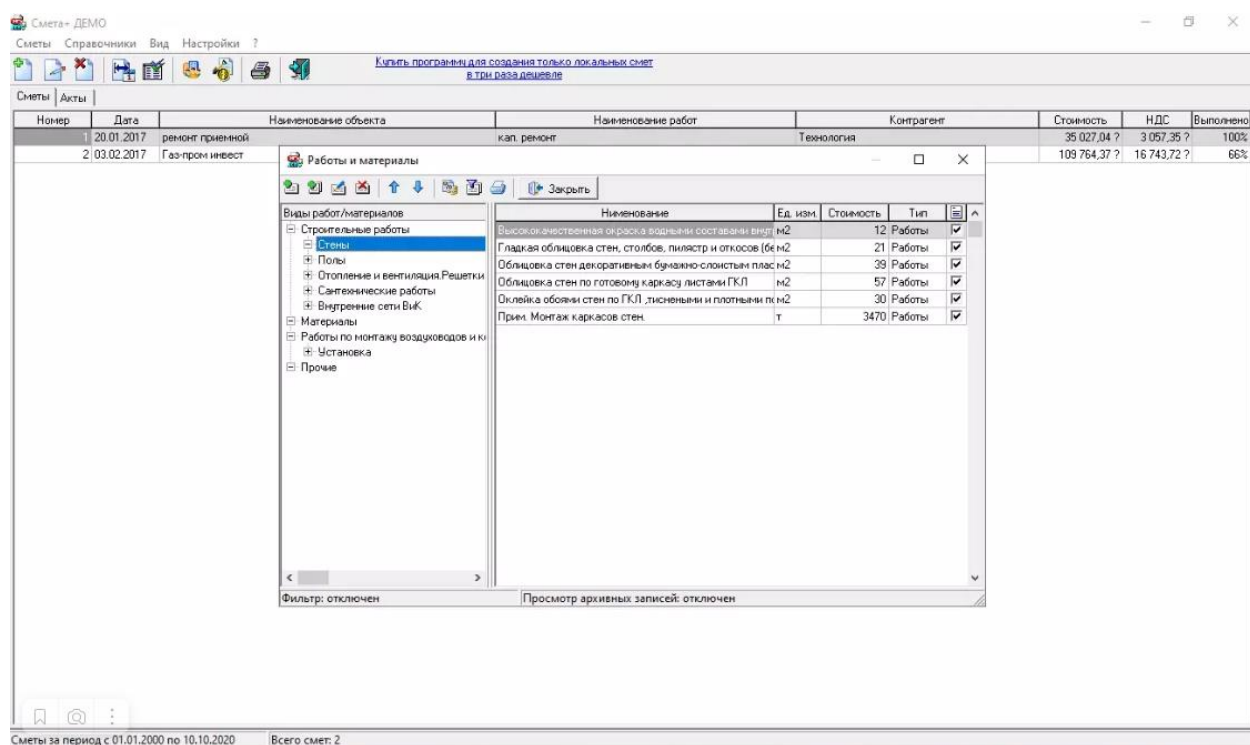


Рисунок 1.9 – Смета+

Плюсы:

- Большая коллекция готовых документов;
- Контроль остатков;
- Можно добавлять и удалять печатные формы.

Минусы:

- Ручная корректировка итога при замене позиций;
- Ошибки в расчетах при наложении коэффициентов.

WinСмета впервые была представлена пользователям еще в 1992 году и до сих пор считается одной из самых удобных программ. Ее используют для составления планов используемых ресурсов, цен на материалы и общей стоимости ремонта. Пользователям доступны готовые шаблоны документов, которые помогут ускорить рабочий процесс. Есть возможность настраивать отображение таблиц и окон под свои нужды. Можно размещать неограниченное количество строк и позиций, добавлять описание и комментарии. Функции в WinСмета сгруппированы по тематическим разделам [42].

Интерфейс программы «WinСмета» показан на рисунке 1.10.

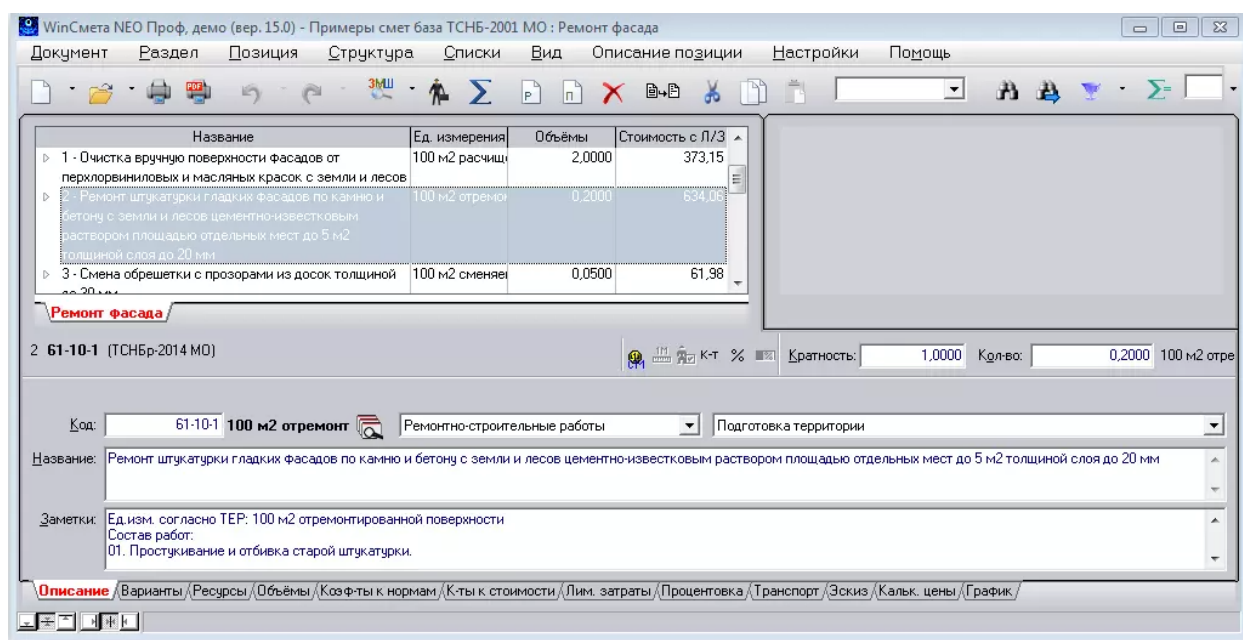


Рисунок 1.10 – WinСмета

Плюсы:

- Удобная сортировка;
- Настройка интерфейса под ваши нужды;
- Внутренняя база с актуальными ценами, документами и пр.

Минусы:

- Может подвисать на последних версиях Windows 10;
- Платные обновления.

Smeta.RU — разработка российской компании СтройСофт для составления и проверки строительных документов. Программа отлично подойдет для компаний подрядчиков любого масштаба, от небольших фирм до крупных организаций с большим количеством клиентов. В ней можно составить акты для монтажных, ремонтных, проектных, изыскательских работ, а также создать проектную документацию. Smeta.RU помогает проверять готовые бумаги и оптимизировать расходы. Софт также используют для обмена внутренней информацией между подразделениями. Приложение поддерживает Windows XP, 7, Vista, 8. 10 с разрядность 32 и 64 бита [39].

Интерфейс программы «Smeta.ru» показан на рисунке 1.11.

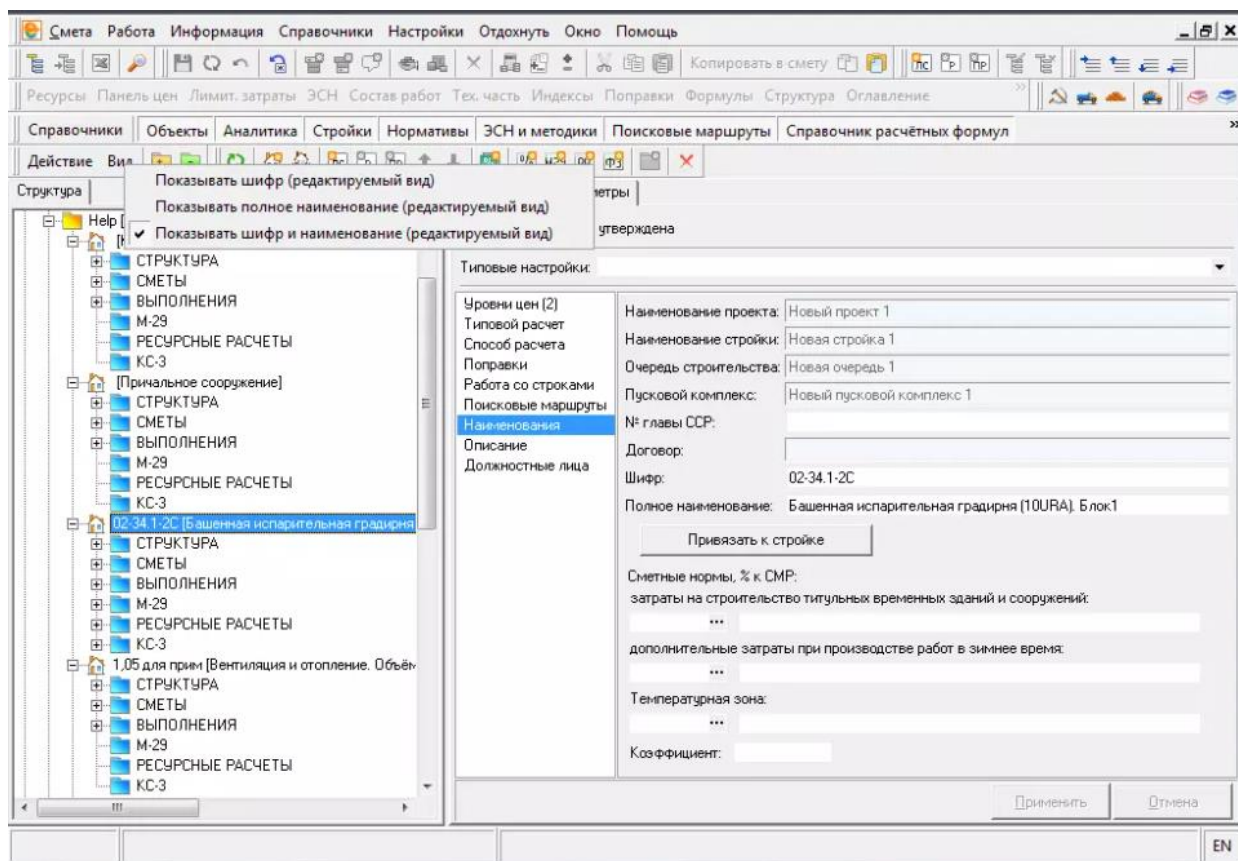


Рисунок 1.11 – Smeta.ru

#### Плюсы:

- Работа с несколькими документами одновременно;
- Установка прав пользователей;
- Ручной ввод данных или использование шаблонов;
- Автоматическая проверка документов на соответствие государственным нормам.

#### Минусы:

- Подвисает при хранении большого количества информации;
- Резервное копирование занимает много времени.

«Сметный калькулятор» представляет из себя инструмент для расчета разного типа данных. Пользователям предоставляется доступ к базе актуальных расценок, можно вести журнал, отслеживать наличие материалов и создавать дефектные ведомости. Готовый проект можно отправить в Word или Excel или на корпоративный сайт. Среди прочих функции: поиск по документам, расчет по

всему проекту или по отдельным пунктам, выбор коэффициента расценки. Вся введенная информация сохраняется в автоматическом режиме, что исключает риск потери важных данных при сбое [20].

Интерфейс программы «Сметный калькулятор» показан на рисунке 1.12.

Наименование работ и затрат, материалов, изделий и конструкций	Единица измер.	Кол-во	ВСЕГО з/п рабочих	экспл. машин т.ч з/п маш-то	ВСЕГО	з/п рабочих	экспл. машинист	затраты труда на единицу	затраты труда всего	Материал
1. Устройство оснований подстилающих и выравнивающих слоев оснований из песка толщ. слоя 45 см к з/п рабочих: 4.7 к стоим. и экспл. машин: 2.95 к стоим. мат-ов: 4.28	100 м3 мате	14.386	6968.01 591.82	6323.97 523.89	100241.79	8513.92	90976.63 7536.68	15.72 13.88	226.15 199.68	751.24
408-9040-010 Песок для строительных работ природный, карьерный намывной	м3	1582.42	(393.19)		(622191.72)					
2. Устройство оснований толщиной 15 см из щебня фракции 40-70 мм (при укатке каменных материалов с пределом прочности на сжатие свыше 98,1 (1000) МПа (кгс/см2)) однослойных к з/п рабочих: 4.7 к стоим. и экспл. машин: 2.95 к стоим. мат-ов: 4.33	1000 м2 осн	3.1968	109883.15 1430.16	15400.42 1916.7	351274.45	4571.94	49232.06 6127.31	37.29 49.92	119.21 159.58	297470.4
3. Устройство оснований городских проездов толщиной слоя 16 см к з/п рабочих: 4.7 к стоим. и экспл. машин: 2.95 к стоим. мат-ов: 3.13	1000 м2 осн	3.1968	456790.62 12393.66	6956.57 527.25	1460268.25	39620.05	22238.76 1685.51	301.71 16.15	964.51 51.63	1398409.
8. При изменении толщины покрытия на 0,5 см добавлять расценке 27-06-020-3 до 6 см к прямым затратам: 4 к з/п рабочих: 4.7 к стоим. и экспл. машин: 2.95 к стоим. мат-ов: 2.71	1000 м2 пок	3.1968	66729.49 16.36	34.46 0	213320.83	52.3	110.16 0	0.36 0	1.15 0	213158.
9. Устройство покрытия толщиной 4 см из горячих асфальтобетонных смесей плотных мелкозернистых типа АБВ, плотность каменных материалов 2,5-2,9 т/м3 к з/п рабочих: 4.7 к стоим. и экспл. машин: 2.95 к стоим. мат-ов: 2.71	1000 м2 пок	3.1968	149629.33 1731.71	7037.96 774.52	478335.04	5535.93	22498.95 2475.99	38.3 19.12	122.44 61.12	450300.1
10. При изменении толщины покрытия на 0,5 см добавлять до 5 см к прямым затратам: 2 к з/п рабочих: 4.7 к стоим. и экспл. машин: 2.95 к стоим. мат-ов: 2.71	1000 м2 пок	3.1968	35158.47 8.18	18.29 0	112394.6	26.15	58.47 0	0.18 38.36	0.58 122.63	112310.0
										9753305    204865    5121

Рисунок 1.12 – «Сметный калькулятор»

Плюсы:

- Простой интерфейс и удобное управление;
- Поддерживает все типы баз;
- Проверка документов на соответствие нормам.

Минусы:

- Перед печатью документ нужно переводить в Word;
- Доступны только два вида актов: КС-2, КС-6;
- Ручной ввод коэффициентов для каждой позиции.

Для понимания целей и представлений для разработки приложения, необходимо составить сравнительную таблицу для выявления плюсов и минусов просмотренных ранее приложений.

На таблице 1.1 и таблице 1.2 представлена сравнительная таблица анализируемых выше приложений. Данные приложения сравниваются по таким критериям, как простота и удобство интерфейса, руссификация интерфейса, наличие базы строительных материалов для покупок, предоставление инструментов в аренду, возможность вывода информации в Excel и Word, сортировка и фильтрация строительных материалов и оборудования, автоматический подсчёт суммы, а также актуальность цен.

Таблица 1.1 – Сравнительная таблица часть первая

Критерии	Оптим-РемСтрой	Дизайн Интерьера 3D	ГРАНД-Смета	АванСмета
Простой и удобный интерфейс	Нет	Нет	Нет	Нет
Руссификация интерфейса	Да	Да	Да	Да
Наличие базы строительных материалов для покупок	Нет	Нет	Нет	Нет
Предоставление инструментов в аренду	Нет	Нет	Нет	Нет

Окончание таблицы 1.1

Критерии	Оптим-РемСтрой	Дизайн Интерьера 3D	ГРАНД-Смета	АванСмета
Вывод информации в Excel и Word	Да	Нет	Да	Нет
Сортировка и фильтрация строительных материалов и оборудования	Да	Да	Да	Да
Автоматический подсчёт суммы	Нет	Нет	Да	Да
Автоматический подсчёт итоговой суммы	Да	Да	Да	Да
Актуальность цен	Нет	Нет	Нет	Нет

Таблица 1.2 – Сравнительная таблица часть вторая

Критерии	Смета+	WinСмета	Smeta.RU	Сметный калькулятор
Простой и удобный интерфейс	Да	Да	Нет	Да

## Окончание таблицы 1.2

Критерии	Смета+	WinСмета	Smeta.RU	Сметный калькулятор
Руссификация интерфейса	Да	Да	Да	Да
Наличие базы строительных материалов для покупок	Да	Да	Нет	Нет
Предоставление инструментов в аренду	Нет	Нет	Нет	Нет
Вывод информации в Excel и Word	Нет	Да	Да	Да
Сортировка и фильтрация строительных материалов и оборудования	Нет	Да	Да	Да
Автоматический подсчёт суммы	Нет	Да	Да	Да
Автоматический подсчёт итоговой суммы	Нет	Да	Да	Да
Актуальность цен	Нет	Да	Нет	Нет

Данная таблица показывает достоинства и недостатки приведённых приложений. Исходя из полученных результатов, можно сделать вывод, что рассмотренные приложения не имеют онлайн базу актуальных цен, и большинство из них предназначены для специалистов, так как имеют сложный для понимания интерфейс и лишний функционал.

#### Выводы по первой части

Представленные в анализе программы и сервисы имеют схожий функционал с подставленной задачей. Они производят расчёты, предоставляют товары и оборудования, составляют смету, некоторые из них позволяют строить 3D модель.

Большинство из вышеописанных программ предназначены для профессионального использования. Они имеют сложный для понимания интерфейс. Такие программы подходят для квалифицированных специалистов, фирм и компаниям занимающимся ремонтом и строительством.

Не все программы, представленные выше, предлагают пользователю товары и оборудования из различных магазинов, некоторые такого функционала не имеют. В основном товары, которые имеются в базах данных таких приложений, это либо стандартные наименования, где пользователь сам указывает их стоимость и характеристики, либо товары и оборудования одного конкретного поставщика или магазина. Иными словами, у пользователя отсутствует свобода выбора как по маркам материалов, так и по магазинам, продающих данные стройматериалы. А это значит пользователь теряет не только разнообразие товаров, но и возможную выгоду и экономию.

Разрабатывая приложение «CASTOPARSER» необходимо предоставить пользователю простой и понятный интерфейс. Показать динамику движения цены конкретного товара. Дать возможность пользователю выбирать товары и оборудования из разных магазинов в одном приложении. Упростить смету до уровня понимания непрофессиональными пользователями. Но при этом оставить функционал вывода данных в Word и Excel.



## 2 РАЗРАБОКА СТРУКТУРЫ ПРОГРАМММЫ

### 2.1 Диаграмма потоков данных (DFD)

DFD (Data Flow Diagram) – диаграмма потоков данных. Диаграммы потока данных (DFD) наглядно отображают, каким образом информация перемещается от задачи к задаче в рамках процесса. DFD модель представляет физические характеристики информационной системы, т.к. она показывает движение информационных объектов и хранилища данных [7].

Для построения модели будет использован программный продукт BPwin.

Моделирование потока данных позволяет сконцентрировать внимание на обмене данными между различными задачами.

Для анализа работы организации в комплексе, и построения больших моделей, в BPwin предусмотрена детализация [37]. Модели могут быть разбиты на группы. Каждая модель представляется на более низком уровне детализации. При этом взаимосвязь между моделями и их элементами сохраняется. С помощью BPwin модель можно разделить на составляющие части, провести работу отдельно с каждой из них, а затем интегрировать обратно в единую модель.

#### 2.1.1 Моделирование

Диаграмма потоков данных будет состоять из трёх вложенных друг в друга уровней детализации.

Диаграмма 0 уровня отображает обобщенный поток данных с относящимися к ней внешними сущностями.

Диаграмма 0 уровня состоит из трёх внешних сущностей [4]:

- Администратор. Определённый человек, который заносит наименования магазинов в программу;
- Магазины строительных материалов. Программа путём парсинга будет получать данные о товарах и оборудовании в виде запросов;
- Пользователь. Некий человек, который будет запрашивать определённые товары и оборудования через приложения и получать ответ в виде

Наименований товаров и оборудования, цены, характеристик и их изображений.

Диаграмма 0 уровня изображена на рисунке 2.1.

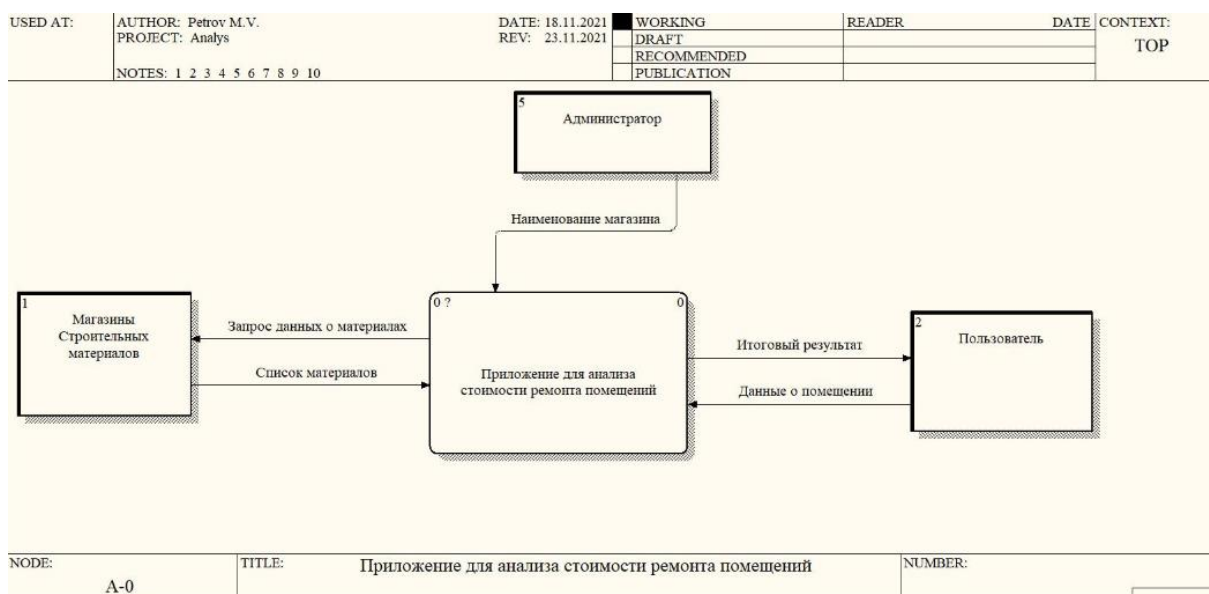


Рисунок 2.1 – Диаграмма 0 уровня

Диаграмма 1 уровня отображает потоки данных в приложении для анализа стоимости ремонта помещений («CASTOPARSER») обобщенный поток данных. Диаграмма 1 уровня состоит из четырёх процессов и одного блока хранилища данных:

- База материалов. Хранилище данных, в котором хранится вся необходимая информация о различных товарах и оборудованиях, и их характеристики. Данные поступают после парсинга и отображаются пользователю в блоке анализа и подсчёта;
- Парсинг. Процесс сбора данных о строительных товарах и оборудованиях, источниками данных которого являются различные строительные магазины. Полученные результаты записываются в базу материалов;
- Блок запросов. Блок процесса, выполняющий отправку запроса о наличии необходимых товаров в подключённую в приложение магазины;
- Анализ и подсчёт. Блок процесса, выполняющий одну из основных функций данного приложения, а именно анализирует полученные

данные о товарах и оборудований, производит необходимые подсчёты и подготавливает данные для их дальнейшего отображения пользователю. Данные получает из базы материалов и отправляет их в интерфейс программы;

- Интерфейс программы. Процесс, при котором пользователь может выбирать необходимые товары, производить поиск по всей базе или по различным критериям, а также в дальнейшем записывать их в смету.

Диаграмма 1 уровня изображена на рисунке 2.2.

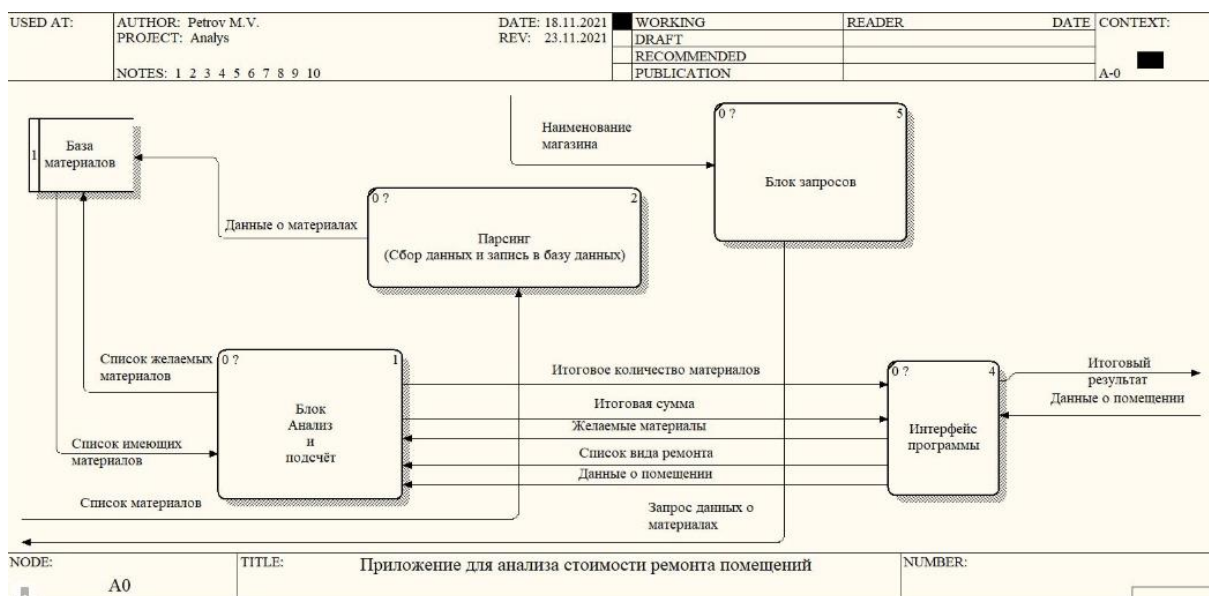


Рисунок 2.2 – Диаграмма 1 уровня

Диаграмма 2 уровня отображает потоки данных процесса анализа и подсчёта. Диаграмма 2 уровня состоит из трёх процессов:

- Выбор ремонта. Процесс выбора критериев и поиск необходимых товаров, которые пользователь вводит в интерфейсе программы и отправляется в блок поиска необходимых материалов;
- Поиск нужных материалов. Получает от пользователя запрос о желаемых материалах, отправляет этот запрос базу материалов и полученный ответ передаёт в блок калькулятор;
- Калькулятор. Процесс, при котором производится подсчёт необходимого количества товара и оборудования и их итоговой стоимости. В дальнейшем результат данного процесса отправляется в смету.

Диаграмма 2 уровня изображена на рисунке 2.3.

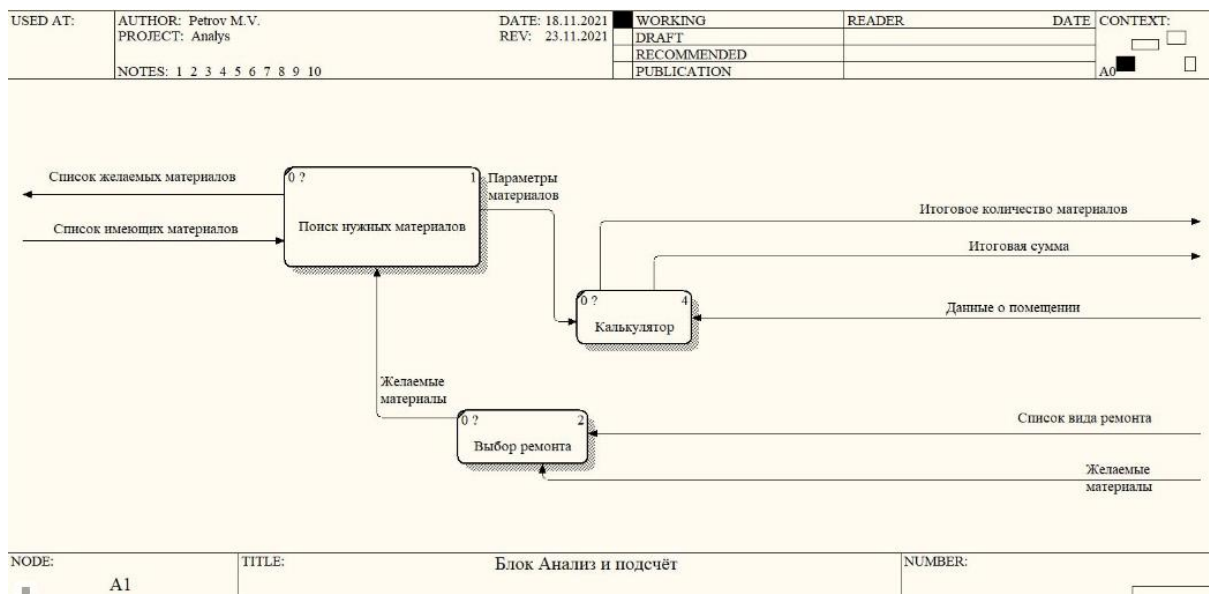


Рисунок 2.3 – Диаграмма 2 уровня

### 2.1.2 Процессы, потоки данных и хранилища данных

Все процессы [4], используемые в данной модели изображены в таблице 2.1.

Таблица 2.1 – Процессы

Наименование	Описание для пользователя	Описание для программиста
Блок запросов	Наименование магазина	Код магазина(Int) Наименование магазина(String)
Выбор ремонта	Наименование ремонта	Наименование ремонта(String)
Интерфейс программы	—	—
Калькулятор	—	—
Парсинг (Сбор данных и запись в базу данных)	Наименование материала Цена за единицу Количество в наличии Длина материала Ширина материала Вес Магазин	Наименование материала(String) Цена за единицу(Float) Количество в наличии(Int) Длина материала(Float) Ширина материала(Float) Вес(Float) Магазин(String)
Поиск нужных материалов	Наименование материала	Наименование материала(String)

## Окончание таблицы 2.1

Наименование	Описание для пользователя	Описание для программиста
Приложение для анализа стоимости ремонта помещений	Поле выбора ремонтных работ Поле ввода площади помещения Поле выбора материалов Фото объекта Чек Поле итоговой стоимости	ComboBox выбора ремонтных работ(string) TextBox площади помещения(string) ComboBox материалов(string) PictureBox материала(.png, .jpg) DataGrid(материал, цена, количество, общая сумма) TextBox итоговой стоимости(string)

Все потоки данных, используемые в данной модели видны в таблице 2.2.

Таблица 2.2 – Потоки данных

Наименование	Описание для программиста	Описание для пользователя
Данные о материалах	Наименование материала(String) Цена за единицу(Float) Количество в наличии(Int) Длина материала(Float) Ширина материала(Float) Магазин(String) Фото(jpeg, png)	Наименование материала Цена за единицу Количество в наличии Длина материала Ширина материала Наименование магазина Фото
Данные о помещении	Высота(Float) Длина(Float) Ширина(Float)	Высота Длина Ширина
Желаемые материалы	Наименование материала(String)	Наименование материала
Запрос данных о материалах	Наименование материалов(String)	Наименование материалов
Итоговое количество материалов	Количество(Int)	Количество
Итоговая сумма	Сумма(Float)	Сумма
Итоговый результат	Список материалов(String) Итоговое количество(Int) Итоговая сумма(Float)	Список материалов Итоговое количество Итоговая сумма

## Окончание таблицы 2.2

Наименование	Описание для программиста	Описание для пользователя
Наименование магазина	Код магазина(Int) Наименование магазина(String)	Наименование магазина
Параметры материалов	Наименование материала(String) Цена за единицу(Float) Количество в наличии(Int) Длина материала(Float) Ширина материала(Float) Вес(Float) Магазин(String) Фото(jpeg, png)	Наименование материала Цена за единицу Количество в наличии Длина материала Ширина материала Вес Магазин Фото
Получение имеющихся материалов	Наименование материала(String)	Наименование материала
Список вида ремонта	Вид ремонта(String)	Вид ремонта
Список видов ремонта	Наименование работы(String)	Наименование работы
Список желаемых материалов	Наименование материала(String)	Наименование материала
Список имеющихся материалов	Наименование материала(String) Цена за единицу(Float) Количество в наличии(Int) Длина материала(Float) Ширина материала(Float) Вес(Float) Магазин(String) Фото(jpeg, png)	Наименование материала Цена за единицу Количество в наличии Длина материала Ширина материала Вес Магазин Фото
Список материалов	Наименование материала(String) Цена за единицу(Float) Количество в наличии(Int) Длина материала(Float) Ширина материала(Float) Магазин(String) Фото(jpeg, png)	Наименование материала Цена за единицу Количество в наличии Длина материала Ширина материала Магазин Фото

Все хранилища данных [26], используемые в данной модели изображены в таблице 2.3.

Таблица 2.3 – Хранилища данных

Наименование	Описание для программиста	Описание для пользователя
База материалов	Код материала(Int) Код магазина(Int) Наименование магазина(String) Вес(Float) Высота(Float) Ширина(Float) Количество в наличии(Int) Цена(Float) Фото	Код материала Наименование магазина Вес Высота Ширина Количество в наличии Цена Фото

## 2.2 База данных серверной и пользовательской частей

Прежде чем приступать к разработке самого парсера, надо подготовить базу данных, а именно:

- Создать таблицы, где будут храниться данные;
- Создать представления (запросы), которые будут использованы как для парсера, так и для дальнейшего отображения пользователю;
- Создать хранимые процедуры для добавления, изменения или удаления данных в таблицах;
- Реализовать триггеры для записи даты в архив.

### 2.2.1 Таблицы и представления

При чтении, добавлении или изменении данных таблица становится доступна для чтения или записи. И если несколько пользователей или процессов попытаются обратиться к таблице, то это не только замедлит работу, но и вызовет получение некорректных данных. Для этого существуют представления [15], выполняющие функцию запросов, которые единожды обращаются к таблице для совершения определённой операции. На рисунках ниже показаны таблицы, их представления и другие представления необходимые для данной работы.

На рисунке 2.4 показана таблица «Каталог» в режиме разработки. На рисунке 2.5 показано представление «Каталог» в режиме разработки. На рисунке

2.6 показано представление «Каталог» в режиме показа данных. Они необходимы для хранения основного каталога товаров, а именно наименования и ссылки каталога.

Имя столбца	Тип данных	Разрешить ...
ID_catalog	int	<input type="checkbox"/>
Name_catalog	nvarchar(100)	<input checked="" type="checkbox"/>
Link_catalog	nvarchar(300)	<input checked="" type="checkbox"/>

Рисунок 2.4 – Таблица «Каталог» в режиме разработки

Столбец	Псевдо...	Таблица	Выход	Тип сортиро...
ID_catalog		Castoram...	<input checked="" type="checkbox"/>	
Name_catalog		Castoram...	<input checked="" type="checkbox"/>	
Link_catalog		Castoram...	<input checked="" type="checkbox"/>	

SELECT ID\_catalog, Name\_catalog, Link\_catalog  
FROM dbo.Castorama\_catalog

Рисунок 2.5 – Представление «Каталог» в режиме разработки

Столбец	Псевдо...	Таблица	Выход	Тип сортиро...
ID_catalog		Castoram...	<input checked="" type="checkbox"/>	
Name_catalog		Castoram...	<input checked="" type="checkbox"/>	
Link_catalog		Castoram...	<input checked="" type="checkbox"/>	

SELECT ID\_catalog, Name\_catalog, Link\_catalog  
FROM dbo.Castorama\_catalog

Рисунок 2.6 – Представление «Каталог» в режиме показа данных

На рисунке 2.7 показана таблица «Категории» в режиме разработки. На рисунке 2.8 показано представление «Категории» в режиме разработки. На рисунке 2.9 показано представление «Категории» в режиме показа данных. Они



необходимы для хранения категорий товаров, а именно наименования, ссылка на открытие страницы с товарами, количество страниц в каждой категории и наименование каталога, к которому принадлежит категория.

Имя столбца	Тип данных	Разрешить ...
ID	int	<input type="checkbox"/>
Name	nvarchar(100)	<input checked="" type="checkbox"/>
Link	nvarchar(300)	<input checked="" type="checkbox"/>
Pages	int	<input checked="" type="checkbox"/>
Name_catalog	nvarchar(100)	<input checked="" type="checkbox"/>

Рисунок 2.7 – Таблица «Категории» в режиме разработки

Столбец	Псевдо...	Таблица	Выход
ID		Castoram...	<input checked="" type="checkbox"/>
Name		Castoram...	<input checked="" type="checkbox"/>
Link		Castoram...	<input checked="" type="checkbox"/>
Pages		Castoram...	<input checked="" type="checkbox"/>

```

SELECT ID, Name, Link, Pages, Name_catalog
FROM
    dbo.Castorama_catalog_categories
    
```

Рисунок 2.8 – Представление «Категории» в режиме разработки

	ID	Name	Link	Pages	Name_catalog
1	1	Обои	https://www.castorama.ru/decoration/wallpaper	64	Обои, шторы и декор
2	2	Шторы и ткани	https://www.castorama.ru/decoration/curtains-and-f...	39	Обои, шторы и декор
3	3	Карнизы и аксессуары	https://www.castorama.ru/decoration/curtain-rods-a...	24	Обои, шторы и декор
4	4	Домашний текстиль	https://www.castorama.ru/decoration/textiles	7	Обои, шторы и декор
5	5	Ковры и придверные коврики	https://www.castorama.ru/decoration/rugs-and-door...	15	Обои, шторы и декор
6	6	Жалюзи и рулонные шторы	https://www.castorama.ru/decoration/zhaljuzi-i-ru-lon...	12	Обои, шторы и декор
7	7	Декор для потолка	https://www.castorama.ru/decoration/ceiling-decor...	6	Обои, шторы и декор
8	8	Постеры и картины	https://www.castorama.ru/decoration/posters-and-c...	8	Обои, шторы и декор
9	9	Рамки и фоторамки	https://www.castorama.ru/decoration/frames-and-p...	6	Обои, шторы и декор
10	10	Освещение помещений	https://www.castorama.ru/lighting/interior-lighting	61	Освещение
11	11	Уличное освещение	https://www.castorama.ru/lighting/outdoor-lighting	6	Освещение

Рисунок 2.9 – Представление «Категории» в режиме показа данных

На рисунке 2.10 показана таблица «Товары» в режиме разработки. На рисунке 2.11 показано представление «Товары» в режиме разработки. На рисунке 2.12 показано представление «Товары» в режиме показа данных. Они необходимы для хранения товаров, а именно наименование товара, ссылка на открытие страницы с подробным описанием товара, цена товара, ссылка на фотографию

товара, спецификации (характеристики) товара, наименование категории, к которой относится товар, артикул товара в магазине и наименование каталога, к которому принадлежит товар.

Имя столбца	Тип данных	Разрешить ...
ID	int	<input type="checkbox"/>
Name	nvarchar(100)	<input checked="" type="checkbox"/>
Link	nvarchar(300)	<input checked="" type="checkbox"/>
Price	nvarchar(100)	<input checked="" type="checkbox"/>
Photo	nvarchar(900)	<input checked="" type="checkbox"/>
Specifications	nvarchar(4000)	<input checked="" type="checkbox"/>
Categories	nvarchar(300)	<input checked="" type="checkbox"/>
Vendor_code	nvarchar(100)	<input checked="" type="checkbox"/>
Name_catalog	nvarchar(100)	<input checked="" type="checkbox"/>
		<input type="checkbox"/>

Рисунок 2.10 – Таблица «Товары» в режиме разработки

Столбец	Псевдо...	Таблица	Выход	Тип сортиро...	Порядок сор...
ID		Castoram...	<input checked="" type="checkbox"/>		
Name		Castoram...	<input checked="" type="checkbox"/>		
Link		Castoram...	<input checked="" type="checkbox"/>		
Price		Castoram...	<input checked="" type="checkbox"/>		
Photo		Castoram...	<input checked="" type="checkbox"/>		
Specifications		Castoram...	<input checked="" type="checkbox"/>		
Categories		Castoram...	<input checked="" type="checkbox"/>		
Vendor_code		Castoram...	<input checked="" type="checkbox"/>		
Name catalog		Castoram...	<input checked="" type="checkbox"/>		

SELECT ID, Name, Link, Price, Photo, Specifications, Categories, Vendor\_code, Name\_catalog  
FROM  
dbo.Castorama\_catalog\_categories\_products

Рисунок 2.11 – Представление «Товары» в режиме разработки

	ID	Name	Link	Price	Photo	Specifications	Categories	Vendor_code	Name_catalog
1	1	Обои Rasch Tiles & More 2014 824506, 10 x 0,53 м,...	https://www.c...	1 148	https://...	Тип продукта ...	Обои	1001424463	Обои, шторы и декор
2	2	Клей для обоев Quelyd Стеклообои, 500 г	https://www.c...	813	https://...	Тип продукта ...	Обои	1001424788	Обои, шторы и декор
3	3	Клей для обоев Quelyd Raccordo, 80 мл	https://www.c...	419	https://...	Тип продукта ...	Обои	1001424785	Обои, шторы и декор
4	4	Клей для обоев Metylan Универсал Премиум, 250 г	https://www.c...	283	https://...	Тип продукта ...	Обои	1001412920	Обои, шторы и декор
5	5	Клей для обоев Metylan Винил Премиум, 300 г	https://www.c...	403	https://...	Тип продукта ...	Обои	1001412921	Обои, шторы и декор
6	6	Грунтовка Момент, 150 г	https://www.c...	148	https://...	Тип продукта ...	Обои	1001412915	Обои, шторы и декор
7	7	Клей для обоев Quelyd Экспресс, 180 г	https://www.c...	280	https://...	Тип продукта ...	Обои	1001424783	Обои, шторы и декор
8	8	Клей для обоев Quelyd Спецвинил, 300 г	https://www.c...	338	https://...	Тип продукта ...	Обои	1001424786	Обои, шторы и декор
9	9	Клей для обоев Quelyd Спецфлизелин, 300 г	https://www.c...	436	https://...	Тип продукта ...	Обои	1001424789	Обои, шторы и декор
10	10	Клей для обоев Quelyd индикатор, 150 г	https://www.c...	365	https://...	Тип продукта ...	Обои	1001424782	Обои, шторы и декор
11	11	Клей для обоев Quality универсальный, 200 г	https://www.c...	103	https://...	Тип продукта ...	Обои	1001418737	Обои, шторы и декор

Рисунок 2.12 – Представление «Товары» в режиме показа данных

Для детального анализа товаров нам нужна таблица, где будет храниться архив товаров с указанием даты добавления записей.

На рисунке 2.13 показана таблица «Архив товаров» в режиме разработки. На рисунке 2.14 показано представление «Архив товаров» в режиме разработки. На

рисунке 2.15 показано представление «Архив товаров» в режиме показа данных. Они необходимы для хранения архива товаров. Таблица «Архив товаров» имеет те же столбцы, что таблица «Товары», и добавлен столбец «Дата».

Имя столбца	Тип данных	Разрешить ...
ID	int	<input type="checkbox"/>
Name	nvarchar(100)	<input checked="" type="checkbox"/>
Link	nvarchar(300)	<input checked="" type="checkbox"/>
Price	nvarchar(100)	<input checked="" type="checkbox"/>
Photo	nvarchar(900)	<input checked="" type="checkbox"/>
Date_	date	<input checked="" type="checkbox"/>
Specifications	nvarchar(4000)	<input checked="" type="checkbox"/>
Categories	nvarchar(300)	<input checked="" type="checkbox"/>
Vendor_code	nvarchar(100)	<input checked="" type="checkbox"/>

Рисунок 2.13 – Таблица «Архив товаров» в режиме разработки

Столбец	Псевдо...	Таблица	Выход	Тип сортиро...	Порядок сор...
ID		Castoram...	<input checked="" type="checkbox"/>		
Name		Castoram...	<input checked="" type="checkbox"/>		
Link		Castoram...	<input checked="" type="checkbox"/>		
Price		Castoram...	<input checked="" type="checkbox"/>		
Photo		Castoram...	<input checked="" type="checkbox"/>		
Date_		Castoram...	<input checked="" type="checkbox"/>		
Specifications		Castoram...	<input checked="" type="checkbox"/>		
Categories		Castoram...	<input checked="" type="checkbox"/>		
Vendor_code		Castoram...	<input checked="" type="checkbox"/>		

SELECT ID, Name, Link, Price, Photo, Date\_, Specifications, Categories, Vendor\_code  
FROM dbo.Castorama\_catalog\_categories\_products\_archive

Рисунок 2.14 – Представление «Архив товаров» в режиме разработки

	ID	Name	Link	Price	Photo	Date_	Specifications	Categories	Vendor_code
1...	244278	Зажимы для выравнивания пли...	https://www....	...	https://www....	2022-04-21	Тип продук...	Крестики и клинья для плитки	1001422701
1...	244279	Земляника садовая	https://www....	...	https://www....	2022-04-21	Тип продук...	Садовые растения и саженцы	1001404703
1...	244280	Земляника садовая, кассета, 1...	https://www....	...	https://www....	2022-04-21	Тип продук...	Садовые растения и саженцы	1001404727
1...	244281	Зеркало для ванной 1Мака Пр...	https://www....	...	https://www....	2022-04-21	Тип продук...	Мебель для ванной комнаты	1001409551
1...	244282	Зеркало косметическое увелич...	https://www....	...	https://www....	2022-04-21	Тип продук...	Мебель для ванной комнаты	1001420960
1...	244283	Желоб водосточный стальной б...	https://www....	...	https://www....	2022-04-21	Тип продук...	Водостоки и дренажные сист...	1001415796
1...	244284	Жгут Порилекс НПЗ, 6 мм х 30 м	https://www....	...	https://www....	2022-04-21	Тип продук...	Изоляция	1001414424
1...	244285	Жидкие гвозди для панелей Qu...	https://www....	...	https://www....	2022-04-21	Тип продук...	Клей и клейкие ленты	1001405706
1...	244286	Жидкие гвозди экспресс Qualit...	https://www....	...	https://www....	2022-04-21	Тип продук...	Клей и клейкие ленты	1001405703
1...	244287	Жидкие гвозди экспресс Qualit...	https://www....	...	https://www....	2022-04-21	Тип продук...	Клей и клейкие ленты	1001405705
1...	244288	Жидкие гвозди универсальные ...	https://www....	...	https://www....	2022-04-21	Тип продук...	Клей и клейкие ленты	1001405704

Рисунок 2.15 – Представление «Архив товаров» в режиме показа данных

Также свой архив имеют таблицы «Каталог» и «Категории».

Записей товаров будет много, основная причина, с которой можно бороться, это повторение одних и тех же товаров, которые присутствуют в разных

категориях. Для этого было создано представление «Уникальные товары», которое выводит только неповторяющиеся товары. Фильтрация происходит по неповторяющимся ссылкам на товары.

На рисунке 2.16 показано представление «Уникальные товары» в режиме разработки. На рисунке 2.17 показано представление «Уникальные товары» в режиме показа данных.

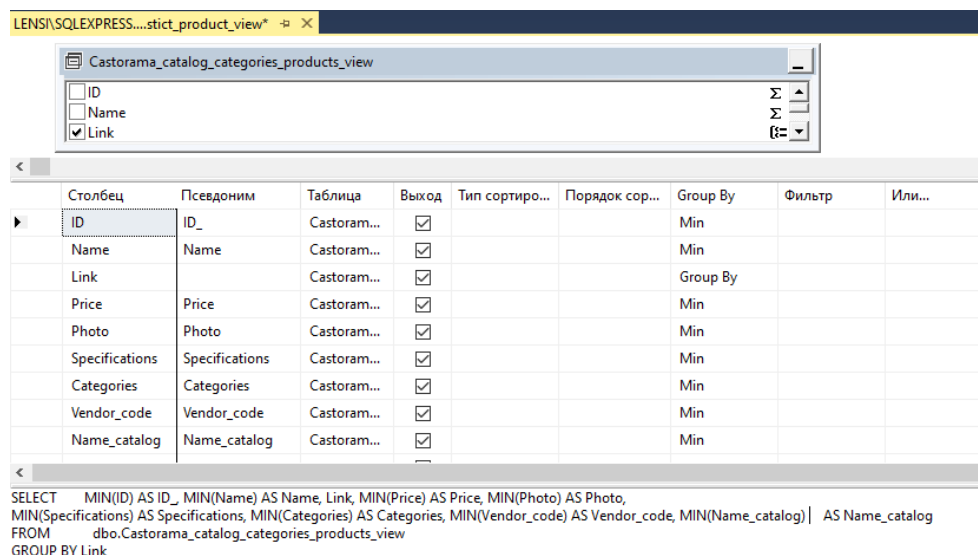


Рисунок 2.16 – Представление «Уникальные товары» в режиме разработки

	ID_	Name	Link	Price	Photo	Specifications	Categories	Vendor_code	Name_catalog
457	37366	Фонарь Navigator NPT-SP18 L...	https://www....	2 ...	https://www...	Тип продукта Фо...	Фонари	1001442584	Электротовары
458	37371	Фонарь-подсветка Navigator ...	https://www....	150	https://www...	Тип продукта Фо...	Фонари	1001442576	Электротовары
459	37374	Фонарь-прожектор ЭРА LED ...	https://www....	2 ...	https://www...	Тип продукта Фо...	Фонари	1001436651	Электротовары
460	37375	Фонарь прожекторный ЭРА Р...	https://www....	3 ...	https://www...	Тип продукта Руч...	Фонари	1001436648	Электротовары
461	37353	Фонарь ручной Эра C0027253 ...	https://www....	257	https://www...	Тип продукта Руч...	Фонари	1001436652	Электротовары
462	37356	Фонарь с карабином Эра Б00...	https://www....	660	https://www...	Тип продукта Пох...	Фонари	1001436642	Электротовары
463	37376	Фонарь-трансформер 3 в 1 Э...	https://www....	1 ...	https://www...	Тип продукта Руч...	Фонари	1001436649	Электротовары
464	37114	Ф-разъем RG6 Партнер Элект...	https://www....	44	https://www...	Тип продукта Ште...	Антенны и ТВ-аксессуары	1001443543	Электротовары
465	24625	Фреза дисковая Bosch 26086...	https://www....	1 ...	https://www...	Тип продукта Фр...	Оснастка для электроинструмент...	1001444641	Инструменты
466	24624	Фреза карнизная Bosch 2608...	https://www....	1 ...	https://www...	Тип продукта Фр...	Оснастка для электроинструмент...	1001444635	Инструменты
467	13967	Комплект от комаров (электр...	https://www....	294	https://www...	Тип продукта Сре...	Средства защиты от насекомых и...	1001407257	Дача и сад

Рисунок 2.17 – Представление «Уникальные товары» в режиме показа данных

### 2.2.2 Хранимые процедуры и триггеры

Хранимые процедуры [31], показанные и описанные на листинге 2.1–2.6, имеют общие принципы работы для добавления, изменения и записи в архив данных. Поэтому, в качестве примера для добавления, изменения и добавления записей в архив, будут представлены хранимые процедуры, относящиеся к таблице «Товары».

На листинге 2.1 показана процедура, которая записывает в таблицу «Категории» количество страниц товаров, относящихся к этой категории.

#### Листинг 2.1 – Добавление страниц

```
ALTER PROCEDURE [dbo].[Castorama_catalog_categories_insert_pages]

    @id_ int,
    @pages int
AS
BEGIN
    SET NOCOUNT ON;
    Update Castorama_catalog_categories
    set [Pages]=@pages
    where ID=@id_
END
```

На листинге 2.2 показана процедура, которая очищает таблицу «Продукты» от записей перед добавлением новых записей.

#### Листинг 2.2 – Очищение таблицы

```
ALTER PROCEDURE [dbo].[Castorama_catalog_categories_products_clear_]
AS
BEGIN
    SET NOCOUNT ON;
    TRUNCATE TABLE Castorama_catalog_categories_products;
END
```

На листинге 2.3 показана процедура, которая получает и записывает такие данные, как наименование товара, ссылка на товар, цена товара, ссылка на фотографию товара, наименование каталога и наименование категории, к которым относится товар.

#### Листинг 2.3 – Запись основных данных

```
ALTER PROCEDURE [dbo].[Castorama_catalog_categories_products_insert]
    @name_ nvarchar(100),
    @link_ nvarchar(300),
    @price_ nvarchar(100),
    @photo_ nvarchar(900),
    @name_cat nvarchar(900),
    @name_categ nvarchar(900)
AS
BEGIN
    SET NOCOUNT ON;
    Insert Into Castorama_catalog_categories_products ([Name],Link,Price, Photo,
Categories,Name_catalog )
    Values (@name_,@link_,@price_,@photo_,@name_cat, @name_categ)
END
```

На листинге 2.4 показана процедура, которая записывает спецификации (характеристики) товара, а также его стоимость. Цена записывается второй раз, так как на некоторых товарах указаны две цены, за товар и, к примеру, за квадратный

метр. Во избежание некорректных данных, цена берётся со страницы с подробной информацией о товаре.

#### Листинг 2.4 – Запись спецификаций

```
ALTER PROCEDURE [dbo].[Castorama_catalog_categories_products_insert_spec]
    @id_ int,
    @spec nvarchar(4000),
    @ven_cod nvarchar(4000),
    @pr_ nvarchar(4000)
AS
BEGIN
    SET NOCOUNT ON;
    Update Castorama_catalog_categories_products
    set [Specifications]=@spec, [Vendor_code]=@ven_cod, [Price]=@pr_
    where ID=@id_
END
```

После всех записей процедура, описанная на листинге 2.5, необходима для правильного отображения информации, а именно спецификаций (характеристик) товара и его артикула. Во время парсинга попадают ненужные символы, такие как, лишние пробелы и отступы. Фильтровать их сразу, означает повышение нагрузки и снижение скорости самого парсинга, поэтому это делается в конце, когда парсер выполнил свою задачу.

#### Листинг 2.5 – Обработка спецификаций

```
ALTER PROCEDURE [dbo].[Castorama_catalog_categories_products_update_spec]
    @id_ int,
    @spec nvarchar(4000),
    @vc nvarchar(4000)
AS
BEGIN
    SET NOCOUNT ON;
    UPDATE Castorama_catalog_categories_products
    SET Specifications = @spec, [Vendor_code]=@vc
    WHERE ID = @id_
END
```

На листинге 2.6 показана процедура, которая заносит уникальные (неповторяющиеся) записи в архив.

#### Листинг 2.6 – Запись в архив

```
ALTER PROCEDURE [dbo].[Castorama_catalog_categories_products_archive_]
AS
BEGIN
    SET NOCOUNT ON;
    INSERT INTO Castorama_catalog_categories_products_archive([Name],Link,Price,
    Photo, Specifications, Categories, Vendor_code)
    SELECT [Name], Link,Price,Photo, Specifications, Categories, Vendor_code
    From Distict_product_view
END
```

В данной работе триггер [24] используется для добавления даты при создании архива. Соответственно, триггер имеют три таблицы, а именно, «Архив каталога», «Архив категорий» и «Архив товаров». На листинге 2.7 представлен триггер для таблицы «Архив товаров».

#### Листинг 2.7 – Запись даты в архив

```
ALTER TRIGGER [dbo].[Castorama_catalog_archive_categories_products_trigger]
ON [dbo].[Castorama_catalog_archive_categories_products_archive]
AFTER INSERT
AS
BEGIN
    SET NOCOUNT ON;
    Update Castorama_catalog_archive_categories_products_archive
    set Date_=getDate()
    from inserted, Castorama_catalog_archive_categories_products_archive
    where inserted.ID=Castorama_catalog_archive_categories_products_archive.ID
END
```

Из вышесказанного следует, что при добавлении записи в столбец «Дата» записывается текущая дата.

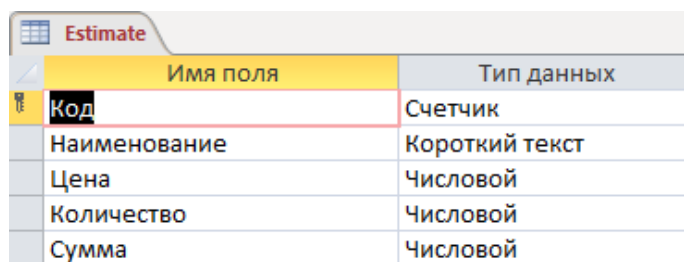
Подобные таблицы, представления, хранимые процедуры и триггеры были созданы для магазина «Hilti».

Для работы со строительной сметой необходимо создать локальную базу данных на персональном компьютере с использованием Access.

В данной базе данных будут храниться две таблицы «Estimate» и «Estimate\_rent».

Таблица «Estimate» предназначена для хранения данных сметы на покупку товаров или оборудования. И состоит из столбцов «Код», «Наименование», «Цена», «Количество» и «Сумма».

На рисунке 2.18 показана таблица «Estimate» в режиме конструктора, а на рисунке 2.19 таблица «Estimate» показана в режиме отображения.



Имя поля	Тип данных
Код	Счетчик
Наименование	Короткий текст
Цена	Числовой
Количество	Числовой
Сумма	Числовой

Рисунок 2.18 – Таблица «Estimate» в режиме конструктора

Estimate				
Код	Наименова	Цена	Количество	Сумма
22	Ручка-защелк	1197	5	5985
23	Патрон для др	406	1	406
24	Патрон для ла	47	10	470
25	Семена Томат	21	100	2100
26	Ручка-скоба К	103	13	1339
27	Стакан для зуб	460	2	920
29	Тавр алюмин	515	10	5150
*	(№)	0	0	0

Рисунок 2.19 – Таблица «Estimate» в режиме отображения

Таблица «Estimate\_rent» предназначена для хранения данных сметы на аренду рабочих помещений или оборудования. И состоит из столбцов «Код», «Наименование», «Цена», «Количество», «Продолжительность» и «Сумма».

На рисунке 2.20 показана таблица «Estimate\_rent» в режиме конструктора, а на рисунке 2.21 таблица «Estimate\_rent» показана в режиме отображения.

Estimate_rent	
Имя поля	Тип данных
Код	Счетчик
Наименование	Короткий текст
Цена	Числовой
Количество	Числовой
Продолжительность	Числовой
Сумма	Числовой

Рисунок 2.20 – Таблица «Estimate\_rent» в режиме конструктора

Estimate_rent					
Код	Наименова	Цена	Количество	Продолжит	Сумма
9	Коленчатый д	9700	1	2	19400
10	Ножничный д	5250	2	7	257250
11	Мотобур Stihl	2250	1	3	6750
12	Виброплита D	1350	1	3	4050
13	Строительная	560	1	3	1680
15	Установка алм	2200	1	3	6600
16	Офисный мод	670	1	3	2010
*	(№)	0	0	0	0

Рисунок 2.21 – Таблица «Estimate\_rent» в режиме отображения

### 2.3 Классы и интерфейсы

Для приложения потребуется создание вспомогательных классов [8] и интерфейсов [6].

Для парсинга надо подключить библиотеку AngleSharp.



AngleSharp – это библиотека .NET, которая дает возможность анализировать гипертексты на основе угловых скобок, такие как HTML, SVG и MathML. XML без проверки также поддерживается библиотекой. Важным аспектом AngleSharp является то, что CSS также можно анализировать. Включенный парсер построен на официальной спецификации W3C. Это создает совершенно портативное представление HTML5 DOM данного исходного кода и обеспечивает совместимость с результатами в вечнозеленых браузерах. Также стандартные функции DOM, такие как `querySelector` или `querySelectorAll`, работают для обхода дерева [32].

Преимущество перед подобными библиотеками, такими как `HtmlAgilityPack`, заключается в том, что открытый DOM использует официальный указанный W3C API, т.е. Даже такие вещи, как `querySelectorAll` доступны в AngleSharp. Также парсер использует спецификацию HTML 5.1, которая определяет обработку ошибок и исправление элементов. Библиотека AngleSharp фокусируется на соответствии стандартам, интерактивности и расширяемости. Поэтому он дает веб-разработчикам, работающим с C #, все возможности, которые они знают из использования DOM в любом современном браузере [32].

Производительность AngleSharp довольно близка к производительности браузеров. Даже очень большие страницы могут быть обработаны в течение миллисекунд. AngleSharp пытается минимизировать выделение памяти и повторно использует элементы внутри, чтобы избежать ненужного создания объектов [32].

Класс «`HtmlLoader`» необходим для загрузки html-документа [12] по его URL [28]. Ниже на листинге 2.8-2.10 представлены фрагменты кода.

На листинге 2.8 представлена инициализация переменных и создание конструктора получения и отправления URL.

#### Листинг 2.8 –Инициализация

```
readonly HttpClient client;  
    string url;  
    public string URL  
    {  
        get { return url; }  
        set { url = value; }  
    }
```

На листинге 2.9 представлен конструктор класса [9] «HtmlLoader», который присваивает url её стандартное значение

#### Листинг 2.9 – Конструктор

```
public HtmlLoader(IParserSettings settings)
{
    client = new HttpClient();
    url = $"{settings.BaseUrl}";
}
```

На листинге 2.10 представлена функция получения html-документа. Данная функция корректирует ссылку, добавляя в конце номер нужной страницы. Добавляет в заголовок параметры клиента, для имитации работы живого человека. Получает html-документ и преобразуем его в кодировку UTF8.

#### Листинг 2.10 – Получение html-документа

```
public string getResponse(int id)
{
    var currentUrl = url.Replace("{CurrentId}", "?p=" + id.ToString());
    currentUrl = url + "?p=" + id.ToString();
    StringBuilder sb = new StringBuilder();
    byte[] buf = new byte[10192];
    int count = 0;
    WebClient client = new WebClient();
    client.Headers.Add("user-agent", "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.2; .NET CLR 1.0.3705;)");
    client.UseDefaultCredentials = true;
    client.Proxy.Credentials = System.Net.CredentialCache.DefaultCredentials;
    Stream resStream = client.OpenRead(currentUrl);
    do {count = resStream.Read(buf, 0, buf.Length);
        if (count != 0)
        { sb.Append(Encoding.UTF8.GetString(buf, 0, count)); }
    }
    while (count > 0);
    return sb.ToString();
}
```

Класс «HabraParser» выполняет запросы по html-документу.

Существует несколько разновидностей класса «HabraParser». Они принимают переменные [30], которые хранят различные теги и различные имена классов. Суть данного класса заключается в том, чтобы правильно выбрать необходимые данные и по возможности скорректировать для их дальнейшей записи в базу данных. На листинге 2.11 представлен код класса «HabraParser».

#### Листинг 2.11 – HabraParser

```
internal class HabraParser_ : IParser_<string[]>
{
    public string[] Parse(IHtmlDocument document, string teg, string class_name)
    {
        var list2 = new List<string>();
        var L = new List<string>();
    }
}
```

```

        var items = document.QuerySelectorAll(teg.ToString()).Where(item =>
item.ClassName != null && item.ClassName.Contains(class_name.ToString()));
        foreach (var item in items)
        {
            L.Add(item.TextContent);
            string t = item.TextContent.Replace("\n", String.Empty);
            t = t.Replace("py6.", String.Empty);
            t = t.Replace("за нор. м", String.Empty);
            int c = t.Length;
            if (c > 0)
            {
                while (t[0].ToString() == " " && c>1)
                {
                    t = t.Substring(1);
                    c = t.Length;
                }
                while (t[c - 1].ToString() == " " && c > 1)
                {
                    t = t.Substring(0, c - 1);
                    c = t.Length;
                }
            }
            if (teg.ToString() == "div" && class_name.ToString() == "current-price")
            {
                try{ float x = Convert.ToSingle(t);
                    list2.Add(t); }
                catch{ list2.Add("0"); }
            }
            else { }
            if (teg.ToString() == "a")
            { list2.Add(t); }
            if (teg.ToString() == "a")
            { list2.Add(item.Attributes["href"].Value); }
            if (teg.ToString() == "img")
            { list2.Add(item.Attributes["data-src"].Value); }
        }
        return list2.ToArray();
    }
}

```

Здесь идёт запрос о необходимых данных, в данном случае мы получаем тег [22] и наименование класса, и в зависимости какой это тег, будет выполняться та или иная операция. Общие операции, это очистка от лишних символов, которые не предоставляют необходимые сведения.

Класс «HabraSettings» задаёт стандартные параметры для парсера. Код представлен на листинге 2.12.

#### Листинг 2.12 – HabraSettings

```

internal class HabraSettings : IParseSettings
{
    public HabraSettings(int start, int end)
    {
        StartPoint = start;
        EndPoint = end;
    }
    public string BaseUrl { get; set; } = "https://www.castorama.ru/catalogue/";
    public string Prefix { get; set; } = "page{CurrentId}";
    public int StartPoint { get; set; }
    public int EndPoint { get; set; }
}

```

В классе «ParserWorker» происходит рабочий процесс самого парсинга. Здесь идут обращения к классам, указанным выше. Основная функция «Worker», имеет различные вариации и нужна для сбора результатов парсера и её дальнейшую передачу в класс формы. Ниже на листинге 2.13–2.15 описаны основные фрагменты класса «ParserWorker».

На листинге 2.13 показано создание событий.

#### Листинг 2.13 – События

```
public event Action<object, T> OnNewData;  
public event Action<object, T, int, string> OnNewData2;  
public event Action<object, T, int, T, T, string, string> OnNewData3;  
public event Action<object, T,int> OnNewData4;  
public event Action<object, T, int, T, T> OnNewData5;  
public event Action<object> OnCompleted;
```

На листинге 2.14 показана функция «Start». Функция [25] «Start» принимает id [35], тег и класс и передает данные в функцию «Worker».

#### Листинг 2.14 – Start

```
public void Start3(int id_,string class_name, string teg_)  
{  
    isActive = true;  
    Worker3(id_,class_name, teg_);  
}
```

На листинге 2.15 показана функция «Worker». Функция «Worker» получает html-документ, передает функции «Parser». И полученный результат передает событию «OnCompleted».

#### Листинг 2.15 – Worker

```
private void Worker3(int id_,string class_name, string teg_)  
{  
    for (int i = parserSettings.StartPoint; i <= parserSettings.EndPoint; i++)  
    {  
        if (!isActive)  
        {  
            OnCompleted?.Invoke(this);  
            return;  
        }  
        var source = loader.getResponse(i);  
        var domParser = new HtmlParser();  
        Thread.Sleep(1000);  
        var document = domParser.ParseDocument(source);  
        var result = parser33.Parse(document, teg_, class_name, " ");  
        OnNewData4?.Invoke(this, result, id_);  
    }  
    OnCompleted?.Invoke(this);  
    isActive = false;  
}
```

Ниже на листинге 2.16 – 2.17 описаны интерфейсы, используемые для работы.

На листинге 2.16 описан интерфейс «IParserSettings».

## Листинг 2.16 – IParserSettings

```
internal interface IParserSettings
{
    string BaseUrl { get; set; }
    string Prefix { get; set; }
    int StartPoint { get; set; }
    int EndPoint { get; set; }
}
```

На листинге 2.17 описаны интерфейсы «IParser».

## Листинг 2.17 – IParser

```
interface IParser<T> where T : class
{
    T Parse(IHtmlDocument document);
}
interface IParser_<T> where T : class
{
    T Parse(IHtmlDocument document, string nn, string n);
}
interface IParser_3<T> where T : class
{
    T Parse(IHtmlDocument document, string nn, string n, string var);
}
```

### Выводы по второй части

Разработка данной главы является начальной стадией реализации приложения «CASTOPARSER».

В программе BPwin была составлена модель потоков данных будущего приложения.

Модель приложения «CASTOPARSER» имеет трёхуровненную архитектуру [23] (нумерация начинается с 0). В данном разделе подробно был описан каждый уровень структуры.

В таблицах были показаны и расписаны все процессы, хранилища и потоки данных. Расписаны были для двух субъектов, для программиста, который будет программировать данное приложение, и для обычного пользователя, это может быть роль заказчика.

Модель может изменяться и дополняться в ходе реализации приложения «CASTOPARSER».

Было создано 10 таблиц серверной части, 2 таблицы пользовательской части, 8 представлений, 17 хранимых процедур, 4 триггера. Все они нужны для правильной работы приложения и хранят в себе необходимые данные.

Классы и интерфейсы играют важную роль, а именно реализацию парсинга.

### 3 РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЯ «CASTOPARSER»

Приложение «CASTOPARSER» будет содержать две основные части.

Первая часть включает в себя сбор данных (парсинг) и подготовку данных для их дальнейшего использования. Данная часть программы устанавливается на сервер и запускает парсер в определённый промежуток времени (ежедневно, еженедельно и т.д.).

Вторая часть предназначена для пользователя и устанавливается на его персональное устройство. Она в свою очередь разделяется на два раздела.

Первый раздел включает в себя отображение пользователю записей из базы данных, фильтрацию полученных записей, поиск по записям, отображение графика движения цены, отображение характеристик и фотографий выбранных товаров и оборудования, и добавление выбранных товаров, оборудования или помещений в строительную смету.

Второй раздел включает в себя работу со строительной сметой, редактирование количества товаров или оборудования, изменение срока аренды, очищение сметы и вывод данных в Excel и Word.

Для простоты реализации и демонстрации первая и вторая часть будут соединены в одном приложении и на одном устройстве.

Для этого были созданы 4 основные формы:

- MDI-форма. Эта общая форма, позволяющая внутри себя открывать дочерние формы;
- NabraParser. Эта форма будет содержать в себе всю первую часть работы, которая была описана выше;
- Arrangement. Форма, включающая в себя первый раздел второй части работы;
- Construction Estimate. Форма, включающая в себя второй раздел второй части работы.

### 3.1 MDI-форма

Как уже было написано выше MDI-форма [11] позволяет открывать внутри себя одновременно несколько дочерних форм.

На рисунке 3.1 изображена MDI-форма в режиме запуска.

На рисунке 3.2 изображена MDI-форма в режиме открытия окон.

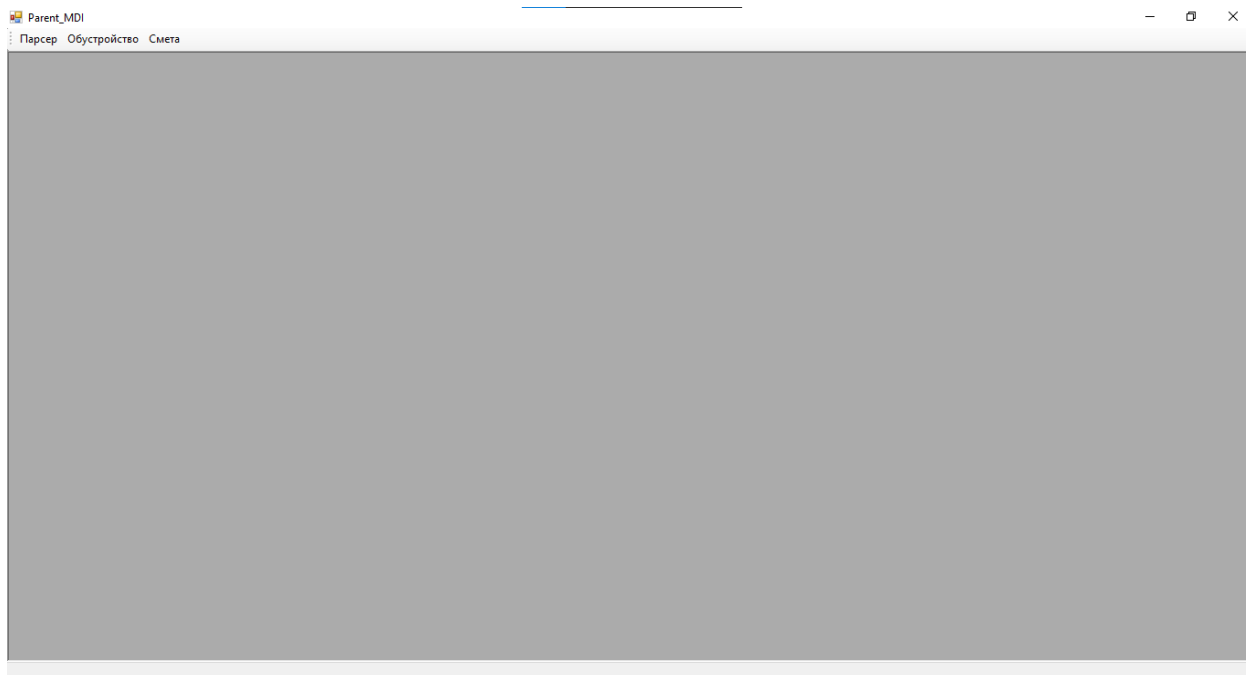


Рисунок 3.1 – MDI-форма в режиме запуска

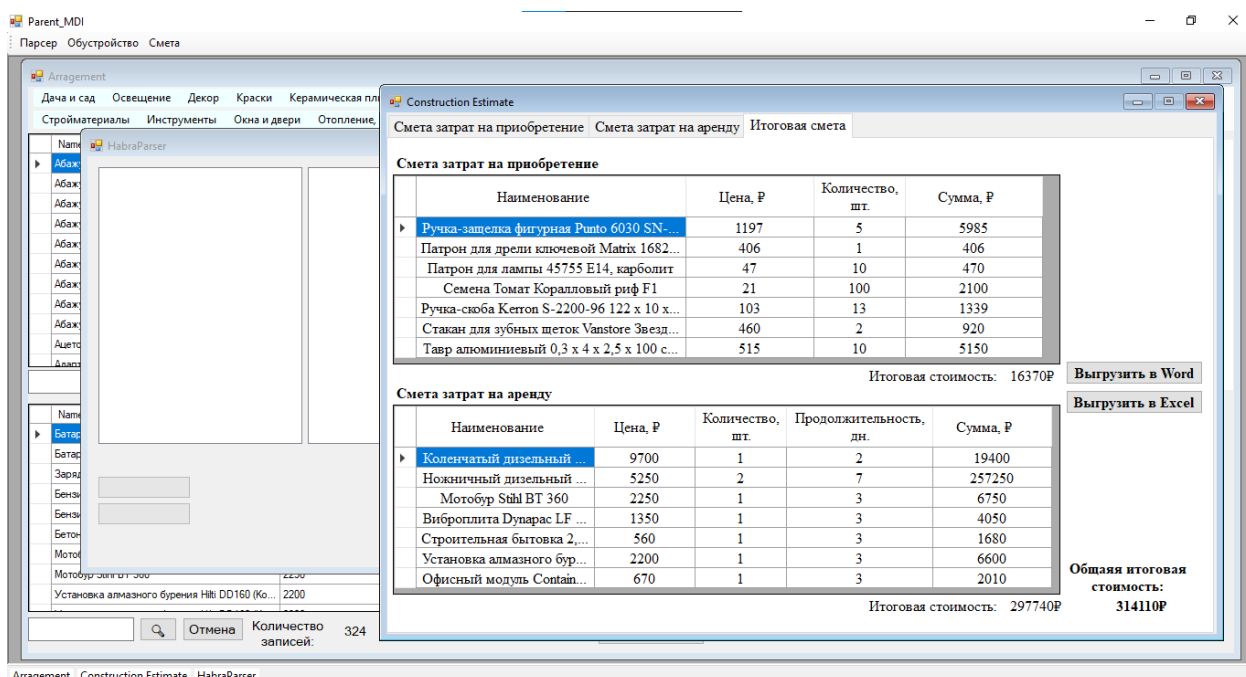


Рисунок 3.2 – MDI-форма в режиме открытия окон

Интерфейс MDI-формы интуитивно прост. В верхней части расположено меню с формами, такие как «Парсер», «Обустройство» и «Смета», которые можно запустить.

В основной области запускаются нужные пользователю окна.

В нижней части располагается меню уже запущенных окон, таких как «HabraParser», «Arrangement» и «Construction Estimate». При нажатии на неё определённое окно развернётся (если она ранее была свёрнута) и отобразится на переднем плане.

Ниже будут описаны основные события принципа работы данной формы.

На листинге 3.1 происходит инициализация [5] форм и переменных. Переменные «flag» отображают состояние формы, открыта она или нет.

#### Листинг 3.1 – Инициализация

```
Form1 f1 = new Form1();

Form3 f3= new Form3();
public int form_parser_flag=0;
public int form_arrangement_flag = 0;
public Form4 f4;
public int construction_estimate_flag = 0;
```

На листинге 3.2 описано событие на запуск формы «HabraParser». Запускается форма с парсером. И в нижней части MDI-формы динамически отображается запущенная форма. Подобный принцип действия на события при запуске формы «Arrangement» и «Construction Estimate» отображены на листингах 3.3 – 3.4 соответственно.

#### Листинг 3.2 – Событие «toolStripButton1\_Click»

```
private void toolStripButton1_Click(object sender, EventArgs e)
{
    if (f1.form_parser_flag == 0)
    {
        f1 = new Form1();
        f1.MdiParent = this;
        f1.Show();
        form_parser_flag = 1;
        ToolStripButton l1 = new ToolStripButton();
        l1.Name= f1.Text;
        l1.Text = f1.Text;
        statusStrip1.Items.Add(l1);
        l1.Click += ButtonOnClick;
        l1.BackColor = Color.White;
    }
}
```



### Листинг 3.3 – Событие «toolStripButton2\_Click\_1»

```
private void toolStripButton2_Click_1(object sender, EventArgs e)
{
    if (f3.form_arrangement_flag == 0)
    {
        f3 = new Form3();
        f3.MdiParent = this;
        f3.Show();
        form_arrangement_flag = 1;
        ToolStripButton l1 = new ToolStripButton();
        l1.Name = f3.Text;
        l1.Text = f3.Text;
        statusStrip1.Items.Add(l1);
        l1.Click += ButtonOnClick;
        l1.BackColor = Color.White;
    }
}
```

### Листинг 3.4 – Событие «toolStripButton3\_Click»

```
private void toolStripButton3_Click(object sender, EventArgs e)
{
    if ( construction_estimate_flag==0)
    {
        f4 = new Form4();
        f4.MdiParent = this;
        f4.Show();
        f4.Tabcon = 2;
        construction_estimate_flag = 1;
        ToolStripButton l1 = new ToolStripButton();
        l1.Name = f4.Text.Replace(" ", "_");
        l1.Text = f4.Text;
        statusStrip1.Items.Add(l1);
        l1.Click += ButtonOnClick;
        l1.BackColor = Color.White;
    }
}
```

На листинге 3.5 описан принцип работы события, происходящее при нажатии на нижнюю часть MDI-формы. Если нужная форма была свёрнута, то она разворачивается и выходит на передний план.

### Листинг 3.5 – Событие «ButtonOnClick»

```
private void ButtonOnClick(object sender, EventArgs eventArgs)
{
    ToolStripButton cb = (ToolStripButton)sender;
    string name_ = cb.Name;
    foreach (Form frm in this.MdiChildren)
    {
        if (frm.Text==name_)
        {
            Form frmChild = frm;
            frmChild.WindowState = FormWindowState.Normal;
            frmChild.Visible = true;
            frmChild.Activate();
            cb.BackColor = Color.White;
        }
    }
}
```

### 3.2 Форма «HabraParser»

Данная форма содержит в себе всю первую часть работы. Сам процесс парсинга и подготовку данных. Форма представлена на рисунке 3.3.

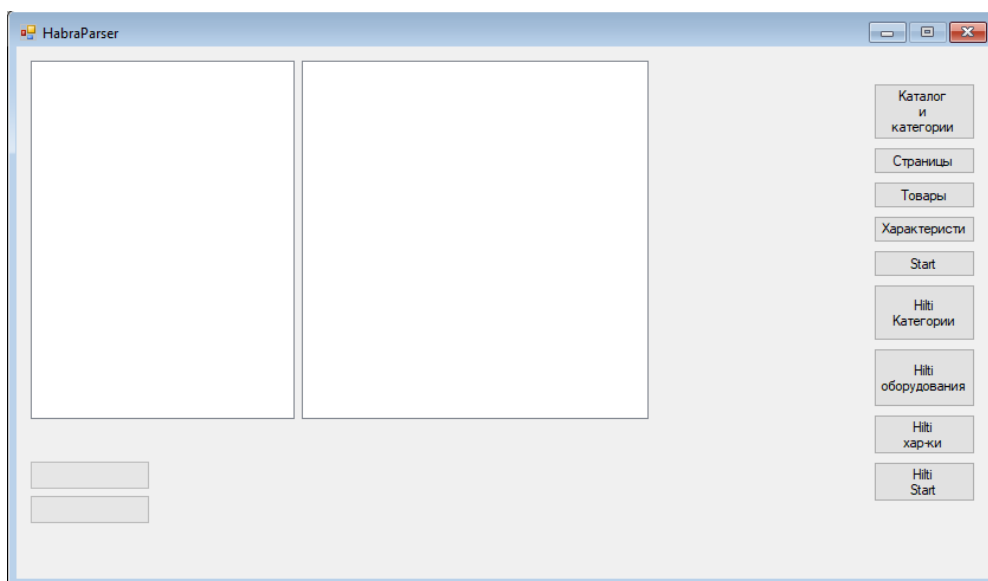


Рисунок 3.3 – Парсер-форма

Данная форма имеет минималистичный интерфейс.

Два listbox'а, предназначенных для вывода категорий и относящихся к ним ссылок.

Две строки, показывающие прогресс операций. Первая показывает прогресс спарсенных категорий товаров, второй – прогресс парсинга самих товаров.

Первая часть кнопок относится к магазину «Castorama»:

- Кнопка «Каталог и категории» собирают данные о всех категориях товаров;
- Кнопка «Страницы» производит подсчёт страниц для каждой категории;
- Кнопка «Товары» запускает процесс парсинга всех имеющихся товаров по всем категориям;
- Кнопка «Характеристики» заносит цену, артикул товара и подробные характеристики о каждом товаре;
- Кнопка «Start» включает в себя выполнение всех предыдущих операций и дополнительную подготовку данных, а именно убирает все ненужные символы и отступы в столбцах «Цена», «Характеристики» и «Артикул».

Вторая часть кнопок относится к магазину «Hilti»:

- Кнопка «Hilti категории» собирают данные о всех категориях товаров;
- Кнопка «Hilti оборудования» запускает процесс парсинга всего имеющегося оборудования по всем категориям;
- Кнопка «Hilti хар-ки» заносит подробные характеристики о каждом оборудовании, если такие имеются;
- Кнопка «Hilti Start» включает в себя выполнение всех предыдущих операций и дополнительную подготовку данных, а именно убирает все ненужные символы и отступы в столбцах «Цена» и «Характеристики».

Ниже будут описаны основные события принципа работы данной формы.

События [34] типа «Parser\_OnNewData» имеют схожий принцип работы.

Получить данные после парсера и занести их базу данных.

Событий «Parser\_OnNewData» несколько разновидностей, так как для парсинга определённых данных, нужны свои исходные переменные, которые отличаются не только данными, но и их количеством. В итоге каждый по своей сути индивидуален, но суть остаётся одна, записать полученные данные в базу данных. В приложении А приведены остальные события «Parser\_OnNewData».

На листинге 3.6 описано событие, которое записывает данные каталога в базу данных и заносит их в архив

Листинг 3.6 – Событие «Parser\_OnNewData»

```
private void Parser_OnNewData(object arg1, string[] arg2)
{
    int count = arg2.Length;
    int flag=0;
    this.queriesTableAdapter1.Castorama_catalog_archive();
    for (int i = 0; i < count;i++)
    {
        if (flag == 0)
        {
            ListTitles.Items.Add(arg2[i]);
            this.queriesTableAdapter1.Castorama_catalog_insert(arg2[i], arg2[i + 1]);
            flag = 1;
        }
        else
        {
            ListTitles2.Items.Add(arg2[i]);
            flag = 0;
        }
    }
}
```

Событие «Parser\_OnNewData2» записывает все категории. Код события представлен на листинге А.1.

Событие «Parser\_OnNewData3» заносит основные данные о товаре или оборудовании в таблицу «Товары». Код события представлен на листинге А.2.

Событие «Parser\_OnNewData33» заносит количество страниц для каждой категории. Код события представлен на листинге А.3.

Отправка данных парсеру происходит при нажатии на определённую кнопку. Принцип действия кнопок «Каталог и категории», «Страницы», «Товары» и «Характеристики» одинаков, поэтому приведён пример кнопки «Товары». Принцип действия прост: получаем необходимые данные из представления «Категории», а именно наименование каталога, наименование категории, количество страниц и ссылку на товары. С помощью функции «Catalog2» получаем эти данные, в цикле проходимся по ним и отправляем на парсинг.

На листинге 3.7 приведён код кнопки «Товары».

На листинге 3.8 приведён код функции «Catalog2».

#### Листинг 3.7 – Событие «button3\_Click»

```
private void button3_Click(object sender, EventArgs e)
{
    string[] cat_ = new string[500];
    cat_ = new string[500];
    progressBar1.Value = 0;
    cat_ = Catalog2();
    int count = 0;
    for (int m = 2; m < cat_.Length; m = m + 5)
    {
        try { count = count + Convert.ToInt32(cat_[m]); }
        catch { }
    }
    int pi = 0;
    int len = cat_.Length;
    this.queriesTableAdapter1.Castorama_catalog_categories_products_clear();
    for (int i = 0; i < cat_.Length; i=i+5)
    {
        string l = "";
        string name_cat = "";
        string categ = "";
        int page_ = 1;
        try{
            l = cat_[i + 1];
            name_cat = cat_[i + 3];
            categ = cat_[i + 4];
            if (cat_[i + 2].ToString() == "")
            {
                page_ = 1;
            }
            else { page_ = Convert.ToInt32(cat_[i + 2]); }
            for(int j=1;j<=page_;j++)
            {
                var ppp = new HabraSettings(j, j);
                ppp.BaseUrl = l;
                parser3.Settings = ppp;
                parser3.Start("product-card__name ga-product-card-name", "a", 14, "current-
price", "div", "product-card__img js-lazy-image lazy-image", "img", name_cat,categ);
                pi++;
                progressBar2.Value = (int)(pi * 100 / count);
            }
        }
    }
}
```

```

        label4.Text = progressBar2.Value.ToString();
        label5.Text = pi.ToString();
        Thread.Sleep(100);
    }
}
catch
{
    l = cat_[i + 1];
    page_ = 1;
    name_cat = cat_[i + 3];
    categ = cat_[i + 4];
    for (int j = 1; j <= page_; j++)
    {
        var ppp = new HabraSettings(j, j);
        ppp.BaseUrl = l;
        parser3.Settings = ppp;
        parser3.Start("product-card__name ga-product-card-name", "a", 14, "current-
price", "div", "product-card__img js-lazy-image lazy-image", "img", name_cat, categ);
        pi++;
        progressBar2.Value = (int)(pi * 100 / count);
        label4.Text = progressBar2.Value.ToString();
        label5.Text = pi.ToString();
        Thread.Sleep(100);
    }
}
progressBar1.Value = (int)(i * 100 / len);
label3.Text = progressBar1.Value.ToString();
Thread.Sleep(100);
}
progressBar1.Value = 100;
}

```

Описание кнопки «Товары». В функцию «Start» мы передаём тег, название класса и другую необходимую нам информацию.

### Листинг 3.8 – Функция «Catalog2»

```

public string[] Catalog2()
{
    string connectionString = @"Data Source=.\SQLEXPRESS;Initial
Catalog=DIPL0M;Integrated Security=True";
    var l = new List<string>();
    string sqlExpression = "SELECT * FROM Castorama_catalog_categories_view";
    using (SqlConnection connection = new SqlConnection(connectionString))
    {
        connection.Open();
        SqlCommand command = new SqlCommand(sqlExpression, connection);
        SqlDataReader reader = command.ExecuteReader();
        if (reader.HasRows) // если есть данные
        {
            while (reader.Read()) // построчно считываем данные
            {
                string id_ = reader.GetValue(0).ToString();
                string name_ = reader.GetValue(1).ToString();
                string link_ = reader.GetValue(2).ToString();
                string page_ = reader.GetValue(3).ToString();
                string cat_name = reader.GetValue(4).ToString();
                l.Add(id_);
                l.Add(link_);
                l.Add(page_);
                l.Add(name_);
                l.Add(cat_name);
            }
        }
        reader.Close();
    }
    return l.ToArray();
}

```

Описание функции «Catalog2». Построчно считываем данные из представления «Категории», записываем их в массив и возвращаем массив.

На листинге 3.9 описано событие при закрытии формы, которое удаляет из нижней части MDI-формы кнопку состояния.

#### Листинг 3.9 – Событие «Form1\_FormClosed»

```
private void Form1_FormClosed(object sender, FormClosedEventArgs e)
{
    form_parser_flag = 0;
    Form q= this.MdiParent;
    ToolStripItem[] toolStripItem;
    toolStripItem =
((Parent_MDI)ParentForm).statusStrip1.Items.Find("HabraParser",true);
    toolStripItem[0].Dispose();
}
```

На листинге 3.10 описано событие при сворачивании формы. Задний фон нижней части MDI-формы меняет цвет, тем самым сообщая, что данная форма свёрнута.

Для избавления от лишних символов, усугубляющих чтение была реализована функция «Specifics», которая представлена на листинге 3.11.

#### Листинг 3.10 – Событие «Form1\_Resize»

```
private void Form1_Resize(object sender, EventArgs e)
{
    if (this.WindowState == FormWindowState.Minimized)
    {
        this.Visible = false;
        ToolStripItem[] toolStripItem;
        toolStripItem = ((Parent_MDI)ParentForm).statusStrip1.Items.Find("HabraParser", true);
        toolStripItem[0].BackColor = Color.Gray;
    }
}
```

#### Листинг 3.11 – Функция «Specifics»

```
public void Specifics()
{
    string[] cat_ = new string[100000];
    cat_ = Catalog5();
    string spec;
    string vc;
    for (int pi = 0; pi < cat_.Length; pi=pi+3)
    {
        spec = cat_[pi+1];
        vc = cat_[pi + 2];
        if (spec.Length != 0)
        {
            while (spec[0].ToString() == " ")
            {
                spec = spec.Substring(1);
            }
        }
        string[] specif;
        specif = new string[1000];
        int o = 0;
        int flag = 0;
        for (int i = 0; i < spec.Length; i++)
        {
            if (i + 2 < spec.Length)
            {
                if (spec[i].ToString() == " " && spec[i + 1].ToString() == " " &&
spec[i + 2].ToString() == " ")
                {
                    flag = 1;
                }
            }
        }
    }
}
```

```

        specif[o] = spec.Substring(0, i);
        o++;
        spec = spec.Substring(i);
        i = 0;
        int j = i;
        while (spec[j].ToString() == " ")
        {
            spec = spec.Remove(j, 1);
            if (spec.Length == 0)
            {
                break;
            }
        }
        i = -1;
    }
}
string stroka = "";
for (int i = 0; i < specif.Length; i++)
{
    if (specif[i] == null)
    {
        break;
    }
    if (stroka == "")
    {
        stroka = stroka + specif[i];
    }
    else { stroka = stroka + "\n" + specif[i]; }
}

if (vc.Length != 0)
{
    while (!Char.IsDigit(vc[0]))
    {
        vc = vc.Substring(1);
    }
}

if (flag == 1)
{
    this.queriesTableAdapter1.Castorama_catalog_categories_products_update_spec(Convert.ToInt32(cat_[pi]), stroka, vc);
}
else {
    this.queriesTableAdapter1.Castorama_catalog_categories_products_update_spec(Convert.ToInt32(cat_[pi]), spec, vc);
}
}
}

```

### 3.3 Форма «Arrangement»

Данная форма содержит в себе первый раздел второй части работы. Форма представлена на рисунке 3.4.

The screenshot shows the 'Arrangement' application window with two main sections: 'Товары для покупки' (Goods for purchase) and 'Инструменты для аренды' (Tools for rent).

**Товары для покупки:**

Name	Price
Абажур для светильника Банные штучки 321	472
Абажур Vitaluce VL0388 нить золото	226
Абажур Vitaluce VL0452P текстиль, коричневый	498
Абажур Vitaluce VL0455 текстиль, черный	754
Абажур Vitaluce VL0457P текстиль, черный	560
Абажур Vitaluce VL0458P текстиль, зеленый	476
Абажур Vitaluce VL0460P текстиль, белый	352
Абажур Vitaluce VL0461P текстиль, бежевый	352
Абажур Vitaluce VL0471P текстиль, бежевый	606
Ацетон Вершина, 0,5 л	155
Адаптер для аккумуляторов MAC ALLISTER	149

Selected item: **Абажур для светильника Банные штучки 32142, угловой, липа, 31 x 17 x 25 см**. Price: 472P.

**Количество:** 1  
**Итого:** 472P

**Добавить в смету**  
**Открыть смету затрат на приобретение**

**Технические характеристики:**

- Тип продукта: Абажур банный
- Артикул производителя: 32142
- Высота, см: 31
- Цвет: коричневый
- Материал: липа
- Вес, кг: 7.7
- Цветовая гамма: коричневый
- Бренд: Банные штучки
- Страна-производитель: Россия
- Размеры упаковки (ШхВхГ), см: 32.1 x 26.6 x 10.3

**График:** Bar chart showing the price of 472P over time (24.04.2022 to 15.05.2022). Average price: 472P.

**Инструменты для аренды:**

Name	Price
Батарея Hiti B 22/3.0 Li-ion	250
Батарея Hiti B 22/5.2 Li-ion	250
Зарядное устройство Hiti C 4/36-350 220V	250
Бензиновый генератор SDMO SH 10000 E	1250
Бензиновый генератор SDMO SH 6000 TE	1000
Бетономешалка Minmix 150	350
Мотофор Husqvarna 143AE15	1150
Мотофор Shih BT 360	2250
Установка алмазного бурения Hiti DD160 (Ko...	2200

Selected item: **Батарея Hiti B 22/3.0 Li-ion**. Price: 250P/сутки.

**Количество оборудования:** 1  
**Сутки:** 1  
**Итого:** 250P/сутки

**Добавить в смету**  
**Открыть смету затрат на аренду**

**Технические характеристики:**

- Вес, кг: 0.58
- Диапазон рабочей температуры, °C: -17 - 60
- Состояние индикатора зарядки: Да
- Температурный диапазон хранения, °C: -40 - 40
- Размеры (ДхШхВ), мм: 135 x 84 x 60
- Емкость аккумулятора, Ач: 165
- Время зарядки на 3V C4/36-90, мин: 64
- Время зарядки на 3V C4/36-350, мин: 37

**Общее количество записей:** 14  
**Итоговая сумма:** 314110  
**Открыть итоговую смету затрат**

Рисунок 3.4 – Форма «Arrangement»

В верхней части формы динамически располагаются категории товаров, которые позволяют фильтровать данные для первой таблицы с товарами.

Основная часть формы разделена на два раздела. Первый раздел существует для покупки товаров из магазинов, в данной работе в качестве магазина используется сайт магазина «Castorama». Второй раздел используется для аренды оборудования и рабочих помещений, данный взят с магазина «Hilti». Далее ниже будет подробно описаны основные принципы работы.

На рисунке 3.5 показана часть формы «Arragement», которая предназначена для покупки и аренда строительных материалов.

Name	Price
Аэратор для смесителя пластиковый ИС.130...	390
Аэратор для смесителя с двойной струей пла...	545
Аэратор для смесителя пластиковый ИС.130...	636
Аэратор для смесителя пластиковый ИС.130...	480
Аэратор для смесителя пластиковый OLIVE'...	116
Аэратор электрический Al-Ko Combi Care 38E ...	16 990
Аэратор электрический M1P-ZP-300A	2 598
Аэратор коньковый сплошной Техноколь...	695
Аэратор коньковый Техноколь Pilot, диаме...	2 860
Аэратор с 5 шипами Verve	882
Аэратор-скрипачик электрический Паев...	14 990

Name	Price
Батарея Hilti B 22/3.0 Li-Ion	250
Батарея Hilti B 22/5.2 Li-Ion	250
Зарядное устройство Hilti C 4/36-350 220V	250
Бензиновый генератор SDMO SH 10000 E	1250
Бензиновый генератор SDMO SH 6000 TE	1000
Бетономешалка Minmix 150	350
Мотобур Husqvarna 143AE15	1150
Мотобур Stihl BT 360	2250
Установка алмазного бурения Hilti DD160 (Ко...	2200

Рисунок 3.5 – Форма «Arragement. Покупка и аренда»

Принцип работы для покупок и для аренды схожий. И имеет схожий интерфейс.

В левой части располагается таблица со всеми товарами и оборудованием. Сразу под ней расположено поле для поиска необходимых товаров или оборудования. Чуть правее находится надпись найденных записей.

В средней части пользователю представлена фотография выбранного товара или оборудования и форма для добавления записи в смету и открытия сметы на определенном разделе. Поля «Итого» высчитываются автоматически при



заполнении предыдущих полей, для покупки это «Количество», а для аренды это «Количество оборудования» и «Сутки».

В правой части располагаются характеристики выбранного товара или оборудования.

На рисунке 3.6 показана часть формы «Arrangement», которая отражает график движения цены.

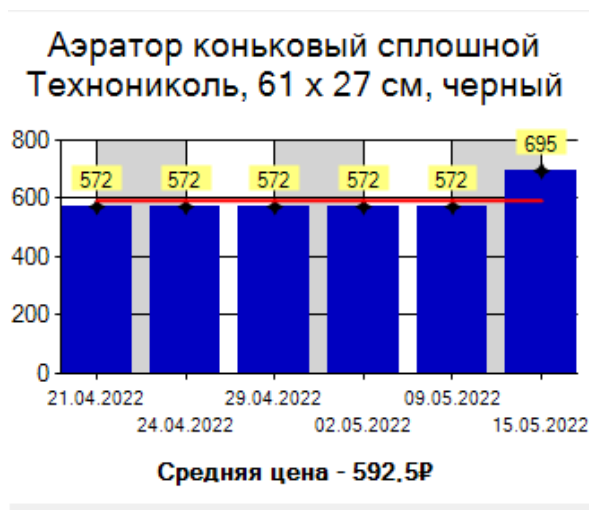


Рисунок 3.6 – Форма «Arrangement. График цены»

В правой верхней части формы пользователю видна диаграмма движения цены. График имеет столбчатый тип. Сверху графика указано наименование выбранного товара. Ось X – Дата, ось Y – Стоимость. Снизу указана средняя цена. Также на график добавлена линия средней цены, которая показывает насколько выгодно пользователю купить выбранный товар. Как правило выгодна та цена, которая находится ниже уровня средней цены. Для наглядности средняя линии показана только там, где цена хоть как-то изменилась. Если цена не менялась, то средняя линия цены не отображается, при этом сама подпись под графиком с расчётом средней цены никуда не исчезает.

На рисунке 3.7 показана часть формы «Arrangement», которая отражает общую информацию.

Рисунок 3.7 – Форма «Arragement. Общая информация»

В правой нижней части содержится информационная форма с общим количеством записей и общей суммой для обеих таблиц. А также содержит кнопку для открытия сметы в общем разделе.

Ниже будут описаны основные события принципа работы данной формы.

На листинге 3.12 показано событие «Form3\_Load», которое происходит во время загрузки формы «Arragement». Идёт загрузка записей из баз данных. Запускается асинхронно процесс динамического создания критериев для фильтров, который расположен в верхней части формы. Его код будет указан ниже. Записываются количество найденных записей для каждой таблицы. Запускается функция «chet», её код показан на листинге 3.16.

Листинг 3.12 – Событие «Form3\_Load»

```
private void Form3_Load(object sender, EventArgs e)
{
    this.estimate_rentTableAdapter.Fill(this.construction_estimateDataSet.Estimate_rent);
    this.estimateTableAdapter.Fill(this.construction_estimateDataSet.Estimate);
    this.hilti_distict_product_viewTableAdapter.Fill(this.dIPLOMDataSet1.Hilti_distict_product_view);
    this.distict_product_viewTableAdapter.Fill(this.dIPLOMDataSet1.Distict_product_view);
    if (!backgroundWorker1.IsBusy)
    { backgroundWorker1.RunWorkerAsync(); }
    label2.Text = dataGridView1.RowCount.ToString();
    label8.Text = dataGridView2.RowCount.ToString();
    form_arrangement_flag = 1;
    chet();
}
```

В событии «backgroundWorker1\_DoWork» происходит очистка от ненужных символов наименований критериев, которые будут использованы дальше в качестве фильтрации. Код события представлен на листинге Б.1.

На листинге 3.13 описана функция «mass», которая берёт все наименования критерий и записывает их в массив.

На листинге 3.14 показано событие «backgroundWorker1\_RunWorkerCompleted», которое срабатывает после завершения события «backgroundWorker1\_DoWork». В процессе данного события происходит подгрузка полученного результата на форму. Критерии располагаются поровну в две строки. Также происходит привязывания события нажатия на кнопку «menustrip».

#### Листинг 3.13 – Функция «mass»

```
public string[] mass()
{
    string connectionString = @"Data Source=.\SQLEXPRESS;Initial
Catalog=DIPL0M;Integrated Security=True";
    var l = new List<string>();
    string sqlExpression = "SELECT * FROM Castorama_catalog_view";
    using (SqlConnection connection = new SqlConnection(connectionString))
    {
        connection.Open();
        SqlCommand command = new SqlCommand(sqlExpression, connection);
        SqlDataReader reader = command.ExecuteReader();
        if (reader.HasRows) // если есть данные
        {
            while (reader.Read()) // построчно считываем данные
            {
                string link_ = reader.GetValue(1).ToString();
                l.Add(link_);
            }
            reader.Close();
        }
        return l.ToArray();
    }
}
```

#### Листинг 3.14 – Событие «backgroundWorker1\_RunWorkerCompleted»

```
private void backgroundWorker1_RunWorkerCompleted(object sender,
RunWorkerCompletedEventArgs e)
{
    string[] results = (string[])e.Result;
    int pol = results.Length / 2;
    for (int i = 0; i < results.Length; i++)
    {
        ToolStripMenuItem l1 = new ToolStripMenuItem();
        l1.Name = results[i].Replace(" ", String.Empty); ;
        l1.Text = results[i];
        l1.Click += menustrip;
        l1.BackColor = Color.Azure;
        if (i < pol)
        {
            menuStrip2.Items.Add(l1);
        }
        else { menuStrip1.Items.Add(l1); }
    }
}
```

На листинге 3.15 показано событие «menustrip», которое происходит после клика на критерий. Если критерий не активен, то он активируется, меняя цвет и в таблице с товарами пользователь видит и может искать только те записи, которые соответствуют выбранному критерию. Если критерий был уже активирован, то после повторного клика, он отменяется и возвращает стандартные настройки.

#### Листинг 3.15 – Событие «menustrip»

```
private void menustrip(object sender, EventArgs e)
{
    string name_ = "";
    name_ = (sender as ToolStripMenuItem).Text;
    if ((sender as ToolStripMenuItem).BackColor == Color.MediumSpringGreen)
    {
        (sender as ToolStripMenuItem).BackColor = Color.Azure;
        name_filter = "";
        distict_product_viewBindingSource.Filter = "";
        textBox1.Text="";
        label2.Text=dataGridView1.RowCount.ToString();
    }
    else
    {
        for (int i = 0; i < menuStrip1.Items.Count; i++)
        {
            menuStrip1.Items[i].BackColor = Color.Azure;
        }
        for (int i = 0; i < menuStrip2.Items.Count; i++)
        {
            menuStrip2.Items[i].BackColor = Color.Azure;
        }
        (sender as ToolStripMenuItem).BackColor = Color.MediumSpringGreen;
        name_filter = "[Name_catalog] = '" + name_ + "'";
        distict_product_viewBindingSource.Filter = name_filter;
        label2.Text = dataGridView1.RowCount.ToString();
        textBox1.Text = "";
    }
}
```

На листинге 3.16 показана функция «chet», которая необходима для пересчёта общего числа записей и общей суммы обеих таблиц в информационной форме, расположенной в правой нижней части формы.

#### Листинг 3.16 – Функция «chet»

```
void chet()
{
    this.estimate_rentTableAdapter.Fill(this.construction_estimateDataSet.Estimate_rent);
    this.estimateTableAdapter.Fill(this.construction_estimateDataSet.Estimate);
    textBox8.Text = Convert.ToString(estimateBindingSource.Count +
estimate_rentBindingSource.Count);
    float summ_itog = 0;
    int dr_count = estimateBindingSource.Count;
    int dr_count2 = estimate_rentBindingSource.Count;

    for (int i = 0; i < dr_count; i++)
    {
```

```

        DataGridView dr = (DataGridView)estimateBindingSource[i];
        summ_itog = summ_itog + Convert.ToSingle(dr[4]);
    }
    for (int i = 0; i < dr_count2; i++)
    {
        DataGridView dr = (DataGridView)estimate_rentBindingSource[i];
        summ_itog = summ_itog + Convert.ToSingle(dr[5]);
    }
    textBox9.Text = summ_itog.ToString();
}

```

На листинге 3.17 показано событие «Form3\_Resize», которое срабатывает при изменении состояния формы. Если форма сворачивается, то в строку состояния, расположенную снизу родительской формы «Parent\_MDI», записывается имя формы и выделяется серым цветом. При разворачивании формы вызывается функция «chet», описанная на листинге 3.16.

#### Листинг 3.17 – Событие «Form3\_Resize»

```

private void Form3_Resize(object sender, EventArgs e)
{
    if (this.WindowState == FormWindowState.Minimized)
    {
        this.Visible = false;
        ToolStripItem[] toolStripItem;
        toolStripItem =
((Parent_MDI)ParentForm).statusStrip1.Items.Find("Arrangement", true);
        toolStripItem[0].BackColor = Color.Gray;
    }
    if (this.WindowState == FormWindowState.Normal)
    {
        chet();
    }
}

```

На листинге 3.18 показано событие «Form3\_Activated», которое срабатывает при активации формы. Если форма находится в развёрнутом состоянии, то вызывается функция «chet», описанная на листинге 3.16.

#### Листинг 3.18 – Событие «Form3\_Activated»

```

private void Form3_Activated(object sender, EventArgs e)
{
    if (this.WindowState == FormWindowState.Normal)
    {
        chet();
    }
}

```

На листинге 3.19 показано событие «Form3\_FormClosing», которое происходит при закрытии формы «Arrangement». из строки состояния, расположенной снизу родительской формы «Parent\_MDI», удаляется соответствующая надпись.

### Листинг 3.19 – Событие «Form3\_FormClosing»

```
private void Form3_FormClosing(object sender, FormClosingEventArgs e)
{
    form_arrangement_flag = 0;
    Form q = this.MdiParent;
    ToolStripItem[] toolStripItem;
    toolStripItem = ((Parent_MDI)ParentForm).statusStrip1.Items.Find("Arrangement", true);
    toolStripItem[0].Dispose();
}
```

Событие «dataGridView1\_SelectionChanged» происходит при изменении выбранной записи в первой таблице. Происходит загрузка фотографии и характеристик соответствующих выбранной записи. А также заполняется график движения цены и подсчёт средней линии стоимости. Код события представлен на листинге Б.3.

Вторая таблица имеет такое же событие, за исключением записи в график, поэтому код не будет представлен.

Функция «mass2» берёт все цены и даты выбранного товара по его артиклю и записывает их в массив. Данная функция используется для отображения графика. Код события представлен на листинге Б.2.

На листинге 3.20 показано событие «textBox1\_TextChanged», которое срабатывает во время поиска по таблицы, либо при нажатии на отдельную кнопку поиска по таблице. Тем самым происходит фильтрация по таблице и подсчёт найденных записей.

### Листинг 3.20 – Событие «textBox1\_TextChanged»

```
private void textBox1_TextChanged(object sender, EventArgs e)
{
    string name_;
    name_ = textBox1.Text;
    if (name_filter == "")
    {
        distict_product_viewBindingSource.Filter = "[Name] LIKE'" + "%" + name_ + "%'";
    }
    else {
        distict_product_viewBindingSource.Filter = name_filter + " And " + "[Name] LIKE'" + "%" + name_ + "%'";
    }
    label2.Text = dataGridView1.RowCount.ToString();
}
```

Такое же событие имеет вторая таблица, поэтому код не имеет смысла приводить.

На листинге 3.21 представлен код кнопки «Отмена» для верхней таблицы. Клик на данную кнопку отменяет поиск и очищает поле поиска

Такое же событие имеет вторая таблица, поэтому код не имеет смысла приводить.

#### Листинг 3.21 – Событие «button3\_Click»

```
private void button3_Click(object sender, EventArgs e)
{
    textBox1.Text = "";
    if (name_filter == "")
    { distict_product_viewBindingSource.Filter = ""; }
    else { distict_product_viewBindingSource.Filter = name_filter + ""; }
}
```

На листинге 3.22 представлен код события «textBox2\_TextChanged», который срабатывает при изменении поля количества для верхней таблицы. Происходит подсчёт итоговой стоимости, путём умножения количества на цену.

Подобный код используется и для подсчёта итоговой стоимости аренды, разница лишь в том, что там переумножаются три значения, количество оборудования, продолжительность и цена за сутки аренды.

#### Листинг 3.22 – Событие «textBox2\_TextChanged»

```
private void textBox2_TextChanged(object sender, EventArgs e)
{
    int ID;
    ID = dataGridView1.CurrentRow.RowIndex;
    string price_;
    price_ = dataGridView1[3, ID].Value.ToString();
    float Summ_ = 0;
    if (textBox2.Text == "")
    { Summ_ = Convert.ToSingle(price_) * 0; }
    else
    { Summ_ = Convert.ToSingle(price_) * Convert.ToInt32(textBox2.Text); }
    textBox3.Text = Summ_.ToString();
}
```

На листинге 3.23 представлен код события «textBox2\_KeyPress», позволяющий вводить только цифровые значения в поле «Количество».

Такой же код используется для полей «Количество оборудования» и «Продолжительность».

#### Листинг 3.23 – Событие «textBox2\_KeyPress»

```
private void textBox2_KeyPress(object sender, KeyPressEventArgs e)
{
    char number = e.KeyChar;
    if (!Char.IsDigit(number) && number != 8)
    { e.Handled = true; }
}
```

На листинге 3.24 представлен код события «button4\_Click» нажатие на кнопку «Добавить в смету». Данные, в виде наименования количества и суммы

записываются в смету покупок, в таблицу «Estimate» и вызывается функция «chet», описанная на листинге 3.16.

Такой же код используется при записи в смету аренды, в таблицу «Estimate\_rent», только добавляется ещё один столбец «Продолжительность».

#### Листинг 3.24 – Событие «button4\_Click»

```
private void button4_Click(object sender, EventArgs e)
{
    if (textBox2.Text != "" && textBox3.Text != "")
    {
        string name_="";
        string price_ = "";
        string count_ = "";
        string sum_ = "";
        int ID;
        ID = dataGridView1.CurrentCell.RowIndex;
        name_=dataGridView1[1, ID].Value.ToString();
        price_ = dataGridView1[3, ID].Value.ToString();
        count_ = textBox2.Text;
        sum_ = textBox3.Text;
        form4 = new Form4();
        form4.Insert_Estimate(name_, price_, count_, sum_);
        textBox8.Text = form4.Count_r.ToString();
        chet();
        textBox2.Text = "";
    }
}
```

На листинге 3.25 представлен код события «button5\_Click» нажатие на кнопку «Открыть смету затрат на приобретение». Открывается форма «Construction Estimate» на вкладке «Смета затрат на приобретение». Далее в нижней части формы «Parent\_MDI» создается запись, что запущена форма «Construction Estimate».

Такой же код используется при открытии общей сметы и сметы затрат на аренду. Открывается та же самая форма, но на разных вкладках.

#### Листинг 3.25 – Событие «button5\_Click»

```
private void button5_Click(object sender, EventArgs e)
{
    if (((Parent_MDI)ParentForm).construction_estimate_flag == 0)
    {
        form4 = new Form4();
        form4.MdiParent = ((Parent_MDI)ParentForm);
        form4.Tabcon = 0;
        form4.Show();
        ToolStripButton l1 = new ToolStripButton();
        l1.Name = form4.Text.Replace(" ", "_");
        l1.Text = form4.Text;
        Form q = this.MdiParent;
        ToolStripItem[] toolStripItem;
        ((Parent_MDI)ParentForm).statusStrip1.Items.Add(l1);
        l1.Click += ((Parent_MDI)ParentForm).ButtonOnClick;
        l1.BackColor = Color.White;
    }
}
```



### 3.4 Форма «Construction Estimate»

Данная форма отвечает за второй раздел второй части работы. Она содержит в себе строительную смету. Форма «Construction Estimate» изображена на рисунке 3.8.

Construction Estimate

Смета затрат на приобретение | Смета затрат на аренду | Итоговая смета

**Смета затрат на приобретение**

Наименование	Цена, Р	Количество, шт.	Сумма, Р
Ручка-зашелка фигурная Punto 6030 SN-...	1197	5	5985
Патрон для дрели ключевой Matrix 1682...	406	1	406
Патрон для лампы 45755 E14, карболит	47	10	470
Семена Томат Коралловый риф F1	21	100	2100
Ручка-скоба Keron S-2200-96 122 x 10 х...	103	13	1339
Стакан для зубных щеток Vanstore Звезд...	460	2	920
Тавр алюминиевый 0,3 x 4 x 2,5 x 100 с...	515	10	5150

Итоговая стоимость: 16370Р

**Смета затрат на аренду**

Наименование	Цена, Р	Количество, шт.	Продолжительность, дн.	Сумма, Р
Коленчатый дизельный ...	9700	1	2	19400
Ножничный дизельный ...	5250	2	7	257250
Мотобур Stihl BT 360	2250	1	3	6750
Виброплита Дунарас LF ...	1350	1	3	4050
Строительная бытовка 2,...	560	1	3	1680
Установка алмазного бур...	2200	1	3	6600
Офисный модуль Contain...	670	1	3	2010

Итоговая стоимость: 297740Р

Общая итоговая стоимость: 314110Р

Выгрузить в Word

Выгрузить в Excel

Рисунок 3.8 – Форма «Construction Estimate»

Данная форма состоит из трёх вкладок. На первой вкладке отображается «Смета затрат на приобретение», на второй отображается «Смета затрат на аренду» и на третьей – «Итоговая смета».

Вкладка «Смета затрат на приобретение» отображает таблицу с товарами или оборудованием, который пользователь желает купить.

Таблица состоит из четырёх столбцов: «Наименование», «Цена, Р», «Количество, шт.», «Сумма, Р». Под таблицей располагается надпись с указанием итоговой стоимости. Справа от таблицы находятся пять кнопок: «Выгрузить в Word», «Выгрузить в Excel», «Изменить количество, шт.», «Удалить запись», «Очистить смету».

При попытке изменить количество товара или оборудования, на форме под кнопками появляется панель, в которой будет предложено выбрать число, желаемого количества выбранного товара или оборудования. При подтверждении изменения в выбранную строку перезаписывается новое число, идёт перерасчет суммы в последнем столбце, снизу перерасчитывается итоговая сумма, данные изменения сохраняются в базу данных и в итоговой смете происходит перерасчёт новых данных.

Вышеописанные действия позволяют пользователю редактировать смету затрат на приобретение, выводить её в Word или Excel и сохранять на своём персональном компьютере. «Смета затрат на приобретении» в действии указана на рисунке 3.9 – 3.10.

Наименование	Цена, Р	Количество, шт.	Сумма, Р
Ручка-зашелка фигурная Punto 6030 SN-P...	1197	5	5985
Патрон для дрели ключевой Matrix 1682...	406	1	406
Патрон для лампы 45755 E14, карболит	47	10	470
Семена Томат Коралловый риф F1	21	100	2100
Ручка-скоба Keron S-2200-96 122 x 10 x ...	103	13	1339
Стакан для зубных щеток Vanstore Звезда...	460	2	920
Тавр алюминиевый 0,3 x 4 x 2,5 x 100 см...	515	10	5150
Сиденье для унитаза Eurocomfort Океан ...	624	1	624

Итоговая стоимость: 16994Р

Рисунок 3.9 – Вкладка «Смета затрат на приобретение»

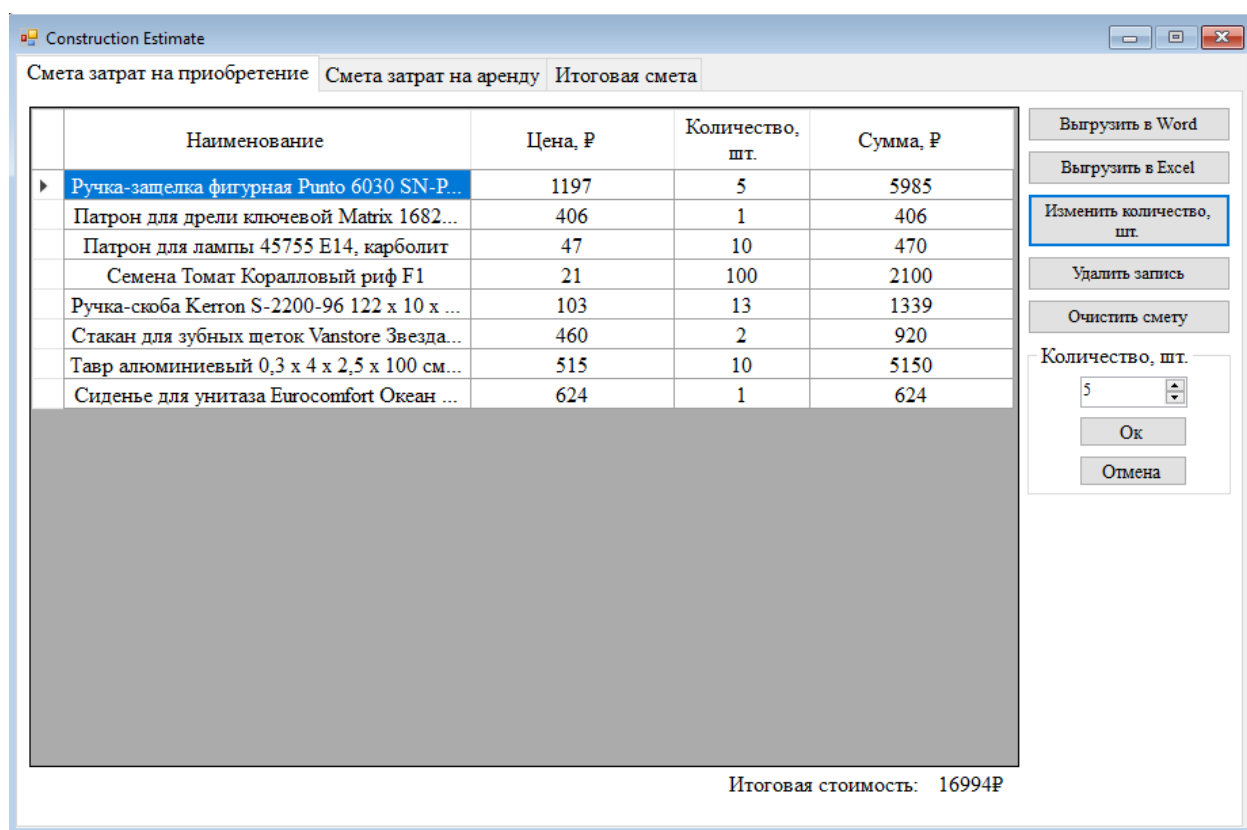


Рисунок 3.10 – Вкладка «Смета затрат на приобретение» в режиме редактирования количества

Вкладка «Смета затрат на аренду» отображает таблицу с оборудованием или помещением, которые пользователь желает арендовать на определённый промежуток времени.

Вкладка выглядит почти как предыдущая, за исключением некоторых деталей, таких как дополнительный столбец «Продолжительность, дн.» и дополнительной кнопкой «изменить продолжительность» с привязанной к ней панели редактирования.

Действия позволяют пользователю редактировать смету затрат на аренду, выводить её в Word или Excel и сохранять на своём персональном компьютере. «Смета затрат на аренду» в действии указана на рисунке 3.11 – 3.12.

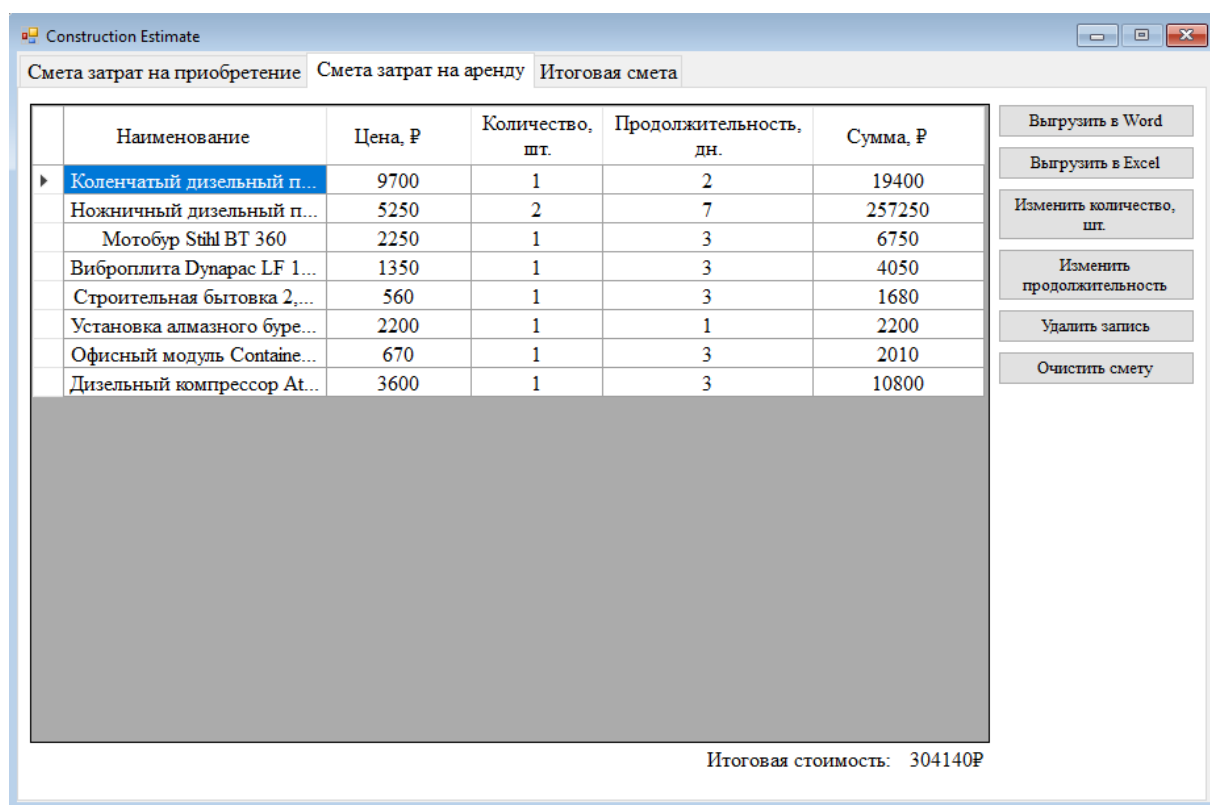


Рисунок 3.11 – Вкладка «Смета затрат на аренду»

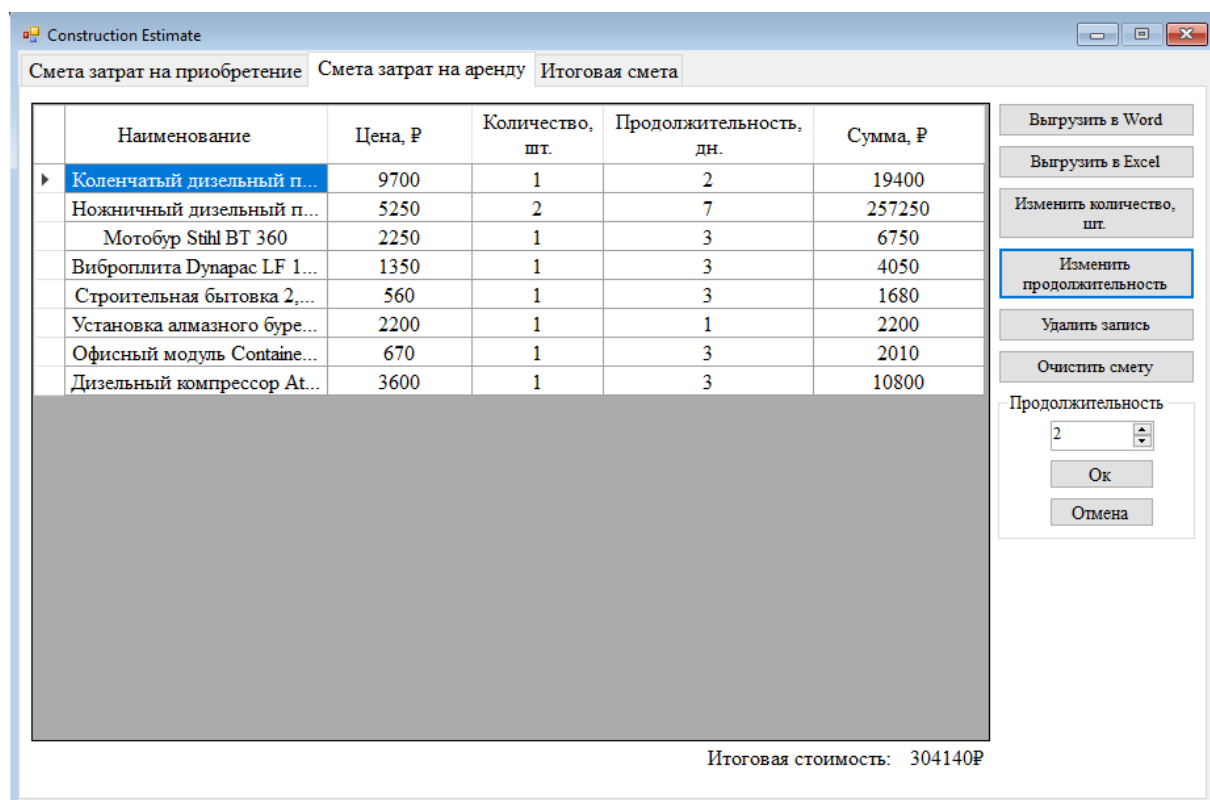


Рисунок 3.12 – Вкладка «Смета затрат на аренду» в режиме редактирования продолжительности

Вкладка «Итоговая смета» отображает две предыдущие таблицы, расположенные на вкладках «Смета затрат на приобретение» и «Смета затрат на аренду».

Здесь доступна общая информация итоговые суммы каждой таблицы и общая итоговая смета. На данной вкладке доступны только функции вывода в Word и Excel. «Итоговая смета» в действии указана на рисунке 3.13.

Construction Estimate

Смета затрат на приобретение   Смета затрат на аренду   **Итоговая смета**

**Смета затрат на приобретение**

Наименование	Цена, Р	Количество, шт.	Сумма, Р
Ручка-зашелка фигурная Punto 6030 SN-...	1197	5	5985
Патрон для дрели ключевой Matrix 1682...	406	1	406
Патрон для лампы 45755 E14, карболит	47	10	470
Семена Томат Коралловый риф F1	21	100	2100
Ручка-скоба Kercon S-2200-96 122 x 10 x...	103	13	1339
Стакан для зубных щеток Vanstore Звезд...	460	2	920
Тавр алюминиевый 0,3 x 4 x 2,5 x 100 с...	515	10	5150

Итоговая стоимость: 16994Р

**Смета затрат на аренду**

Наименование	Цена, Р	Количество, шт.	Продолжительность, дн.	Сумма, Р
Коленчатый дизельный ...	9700	1	2	19400
Ножничный дизельный ...	5250	2	7	257250
Мотобур Stihl BT 360	2250	1	3	6750
Виброплита Дунарас LF ...	1350	1	3	4050
Строительная бытовка 2,...	560	1	3	1680
Установка алмазного бур...	2200	1	1	2200
Одноразовый контейнер	670	1	2	2010

Итоговая стоимость: 304140Р

**Общая итоговая стоимость: 321134Р**

Выгрузить в Word

Выгрузить в Excel

Рисунок 3.13 – Вкладка «Итоговая смета»

На всех трёх вкладках доступны выводы в Word и Excel. Принцип действия у всех один. Разница лишь в том, что первые две вкладки выводят данные по одной таблице, относящейся к конкретной вкладке. В третьей же вкладке вывод записей осуществляется сразу по двум таблицам с дополнительным подсчётом общей итоговой стоимости.

В качестве примеров на рисунке 3.14 – 3.16 изображены вывод в Word для вкладки «Смета затрат на приобретение» и вывод в Excel для вкладки «Итоговая смета».



### Листинг 3.26 – Функции «Tabcon и Count\_r»

```
public int Tabcon
{
    get { return tabControl1.SelectedIndex; }
    set { tabControl1.SelectedIndex = value; }
}
public int Count_r
{
    get { return estimateBindingSource.Count+estimatorrentBindingSource.Count; }
    set { }
}
```

На листинге 3.27 показан код функций, которые добавляют новые записи для таблиц «Esimate» и «Estimate\_rent» соответственно.

### Листинг 3.27 – Функции «Insert\_Estimate и Insert\_Estimate\_rent»

```
public void Insert_Estimate(string name_, string price, string count, string summ)
{
    estimateTableAdapter.Insert(name_, Convert.ToSingle(price), Convert.ToInt32(count),
    Convert.ToSingle(summ));
}
public void Insert_Estimate_rent(string name_, string price, string count, string count2,
string summ)
{
    estimate_rentTableAdapter.Insert(name_, Convert.ToSingle(price),
    Convert.ToInt32(count), Convert.ToInt32(count2), Convert.ToSingle(summ));
}
```

На листинге 3.28 показан код события «dataGridView1\_RowsAdded», которое срабатывает при добавлении новой записи в таблицу, находящуюся на вкладке «Смета затрат на приобретение». Происходит перерасчёт итоговой суммы, а также общей итоговой суммы на вкладке «Итоговая смета».

Аналогичное событие имеется для добавления записи на вкладке «Смета затрат на аренду».

На листинге 3.29 показан код события «Form4\_FormClosing», которое срабатывает при закрытии текущей формы. В нижней части формы «Parent\_MDI» вкладка с названием текущей формы удаляется. И все данные в обеих таблицах сохраняются в базе данных

### Листинг 3.28 – Событие «dataGridView1\_RowsAdded»

```
private void dataGridView1_RowsAdded(object sender, DataGridViewRowsAddedEventArgs e)
{
    float itog = 0;
    for (int i = 0; i < dataGridView1.Rows.Count; i++)
    {
        string summ = dataGridView1[4, i].Value.ToString();
        try {
            float sum_ = Convert.ToSingle(summ);
            itog=itog+sum_;
        }
    }
}
```

```

        catch { }
    }
    label3.Text = itog.ToString() + "Р";
    label5.Text = itog.ToString() + "Р";
    for (int i = 0; i < dataGridView2.Rows.Count; i++)
    {
        string summ = dataGridView2[5, i].Value.ToString();
        try {
            float sum_ = Convert.ToSingle(summ);
            itog = itog + sum_;
        }
        catch { }
    }
    label11.Text = itog.ToString() + "Р";
}

```

### Листинг 3.29 – Событие «Form4\_FormClosing»

```

private void Form4_FormClosing(object sender, FormClosingEventArgs e)
{
    button6.PerformClick();
    construction_estimate_flag = 0;
    Form q = this.MdiParent;
    ((Parent_MDI)ParentForm).construction_estimate_flag = 0;
    ToolStripItem[] toolStripItem;
    toolStripItem =
    ((Parent_MDI)ParentForm).statusStrip1.Items.Find("Construction_Estimate", true);
    toolStripItem[0].Dispose();
    this.estimateBindingSource.EndEdit();
    this.estimateTableAdapter.Update(this.construction_estimateDataSet.Estimate);
    this.estimaterentBindingSource.EndEdit();

    this.estimate_rentTableAdapter.Update(this.construction_estimateDataSet.Estimate_rent);
}

```

На листинге 3.30 показан код события «Form4\_Resize», которое срабатывает при сворачивании или разворачивании формы. добавлении новой записи в таблицу, находящуюся на вкладке «Смета затрат на приобретение». Происходит перерасчёт итоговой суммы для каждой сметы, а также общей итоговой суммы на вкладке «Итоговая смета». В нижней части формы «Parent\_MDI» вкладка с названием текущей формы выделяется серым, если она находится в свёрнутом состоянии.

На листинге 3.31 показан код события «Form4\_Load», которое срабатывает при загрузке формы. Происходит подгрузка данных для всех таблиц.

### Листинг 3.30 – Событие «Form4\_Resize»

```

private void Form4_Resize(object sender, EventArgs e)
{
    if (this.WindowState == FormWindowState.Minimized)
    {
        this.Visible = false;
        ToolStripItem[] toolStripItem;
        toolStripItem =
        ((Parent_MDI)ParentForm).statusStrip1.Items.Find("Construction_Estimate", true);
        toolStripItem[0].BackColor = Color.Gray;
    }
}

```



```

        button6.PerformClick();
    }
    if (this.WindowState == FormWindowState.Normal)
    {
        float itog = 0;
        for (int i = 0; i < dataGridView1.Rows.Count; i++)
        {
            string summ = dataGridView1[4, i].Value.ToString();
            try
            {
                float sum_ = Convert.ToSingle(summ);
                itog = itog + sum_;
            }
            catch { }
        }
        label3.Text = itog.ToString() + "₽";
        label5.Text = itog.ToString() + "₽";
        label3.Refresh();
        label5.Refresh();
        itog = 0;
        for (int i = 0; i < dataGridView2.Rows.Count; i++)
        {
            string summ = dataGridView2[5, i].Value.ToString();
            try {
                float sum_ = Convert.ToSingle(summ);
                itog = itog + sum_;
            }
            catch { }
        }
        label2.Text = itog.ToString() + "₽";
        label9.Text = itog.ToString() + "₽";
        label2.Refresh();
        label9.Refresh();
        for (int i = 0; i < dataGridView2.Rows.Count; i++)
        {
            string summ = dataGridView2[5, i].Value.ToString();
            try {
                float sum_ = Convert.ToSingle(summ);
                itog = itog + sum_;
            }
            catch { }
        }
        label11.Text = itog.ToString() + "₽";
        label11.Refresh();
        dataGridView1.Columns[0].Visible = false;
        dataGridView2.Columns[0].Visible = false;
        dataGridView3.Columns[0].Visible = false;
        dataGridView4.Columns[0].Visible = false;
    }
}

```

### Листинг 3.31 – Событие «Form4\_Load»

```

private void Form4_Load(object sender, EventArgs e)
{
    this.estimate_rentTableAdapter.Fill(this.construction_estimateDataSet.Estimate_rent);
    this.estimate_rentTableAdapter.Fill(this.construction_estimateDataSet.Estimate_rent);
    this.estimateTableAdapter.Fill(this.construction_estimateDataSet.Estimate);
    construction_estimate_flag = 1;
    ((Parent_MDI)ParentForm).construction_estimate_flag = 1;
    dataGridView1.Columns[0].Visible = false;
    dataGridView2.Columns[0].Visible = false;
    dataGridView3.Columns[0].Visible = false;
    dataGridView4.Columns[0].Visible = false;
}

```

На листинге 3.32 показан код события, которое происходит при нажатии на кнопку «Удалить запись». Удаляется выбранная запись, изменения заносятся в базу данных, делается перерасчёт итоговой суммы для каждой сметы, а также общей итоговой суммы на вкладке «Итоговая смета».

#### Листинг 3.32 – Событие «button2\_Click»

```
private void button2_Click(object sender, EventArgs e)
{
    button6.PerformClick();
    if (dataGridView1.SelectedCells.Count > 0)
    {
        int ID;
        ID = dataGridView1.CurrentCell.RowIndex;
        dataGridView1.Rows.Remove(dataGridView1.Rows[ID]);
        label3.Text = "";
        float itog = 0;
        for (int i = 0; i < dataGridView1.Rows.Count; i++)
        {
            string summ = dataGridView1[4, i].Value.ToString();
            try {
                float sum_ = Convert.ToSingle(summ);
                itog = itog + sum_;
            }
            catch { }
        }
        label3.Text = itog.ToString() + "P";
        label5.Text = itog.ToString() + "P";
        for (int i = 0; i < dataGridView2.Rows.Count; i++)
        {
            string summ = dataGridView2[5, i].Value.ToString();
            try {
                float sum_ = Convert.ToSingle(summ);
                itog = itog + sum_;
            }
            catch { }
        }
        label11.Text = itog.ToString() + "P";
        this.estimateBindingSource.EndEdit();
        this.estimateTableAdapter.Update(this.construction_estimateDataSet.Estimate);
        this.estimaterentBindingSource.EndEdit();
        this.estimate_rentTableAdapter.Update(this.construction_estimateDataSet.Estimate_rent);
    }
}
```

На листинге 3.33 показан код события «Form4\_Activated», которое срабатывает при активации формы. Происходит перерасчёт итоговой суммы для каждой сметы, а также общей итоговой суммы на вкладке «Итоговая смета».

#### Листинг 3.33 – Событие «Form4\_Activated»

```
private void Form4_Activated(object sender, EventArgs e)
{
    this.estimateTableAdapter.Fill(this.construction_estimateDataSet.Estimate);
    if (this.WindowState == FormWindowState.Normal)
    {
        float itog = 0;
        for (int i = 0; i < dataGridView1.Rows.Count; i++)
```

```

    {
        string summ = dataGridView1[4, i].Value.ToString();
        try { float sum_ = Convert.ToSingle(summ);
            itog = itog + sum_;
        }
        catch { }
    }
    label3.Text = itog.ToString() + "₽";
    label3.Refresh();
    label5.Text = itog.ToString() + "₽";
    label5.Refresh();
    itog = 0;
    for (int i = 0; i < dataGridView2.Rows.Count; i++)
    {
        string summ = dataGridView2[5, i].Value.ToString();
        try {
            float sum_ = Convert.ToSingle(summ);
            itog = itog + sum_;
        }
        catch { }
    }
    label2.Text = itog.ToString() + "₽";
    label9.Text = itog.ToString() + "₽";
    label2.Refresh();
    label9.Refresh();
    for (int i = 0; i < dataGridView1.Rows.Count; i++)
    {
        string summ = dataGridView1[4, i].Value.ToString();
        try {
            float sum_ = Convert.ToSingle(summ);
            itog = itog + sum_;
        }
        catch { }
    }
    label11.Text = itog.ToString() + "₽";
    label11.Refresh();
    dataGridView1.Columns[0].Visible = false;
    dataGridView2.Columns[0].Visible = false;
    dataGridView3.Columns[0].Visible = false;
    dataGridView4.Columns[0].Visible = false;
}
}

```

На листинге 3.34 описано событие, которое происходит при нажатии на кнопку «Очистить смету». Очищается текущая смета, изменения заносятся в базу данных, делается перерасчёт итоговой суммы для каждой сметы, а также общей итоговой суммы на вкладке «Итоговая смета»

#### Листинг 3.34 – Событие «button1\_Click»

```

private void button1_Click(object sender, EventArgs e)
{
    button6.PerformClick();
    while (dataGridView1.RowCount != 0)
    { dataGridView1.Rows.Remove(dataGridView1.Rows[0]); }
    label3.Text = "";
    float itog = 0;
    for (int i = 0; i < dataGridView1.Rows.Count; i++)
    {
        string summ = dataGridView1[4, i].Value.ToString();
        try {
            float sum_ = Convert.ToSingle(summ);

```

```

        itog = itog + sum_;
    }
    catch { }
}
label3.Text = itog.ToString() + "₽";
label5.Text = itog.ToString() + "₽";
for (int i = 0; i < dataGridView2.Rows.Count; i++)
{
    string summ = dataGridView2[5, i].Value.ToString();
    try {
        float sum_ = Convert.ToSingle(summ);
        itog = itog + sum_;
    }
    catch { }
}
label11.Text = itog.ToString() + "₽";
this.estimateBindingSource.EndEdit();
this.estimateTableAdapter.Update(this.construction_estimateDataSet.Estimate);
this.estimaterentBindingSource.EndEdit();
this.estimate_rentTableAdapter.Update(this.construction_estimateDataSet.Estimate_rent);
}

```

На листинге 3.35 показано событие, которое происходит при нажатии на кнопку «Изменить количество, шт.». Пользователь видит панель изменения с полем текущим количеством, которое он может изменить, кнопки «Ок» и «Отмена».

#### Листинг 3.35 – Событие «button4\_Click»

```

private void button4_Click(object sender, EventArgs e)
{
    if (dataGridView1.SelectedCells.Count > 0)
    {
        int ID;
        ID = dataGridView1.CurrentCell.RowIndex;
        groupBox1.Visible = true;
        try {
            float count = 1;
            count = Convert.ToSingle(dataGridView1[3, ID].Value);
            numericUpDown1.Value = Convert.ToDecimal(count);
        }
        Catch { numericUpDown1.Value = 1; }
    }
}

```

На листинге 3.36 показано событие, которое происходит при нажатии на кнопку «Ок» на панели изменения, которое появляется при попытке изменить количество. Новое число перезаписывается в столбец «Количество, шт.» выбранной строки, пересчитывается и заносится сумма в столбец «Сумма, ₽», делается перерасчёт итоговой суммы для вкладки «смета затрат на приобретение», а также общей итоговой суммы на вкладке «Итоговая смета». Панель изменения количества становится невидимым

### Листинг 3.36 – Событие «button5\_Click»

```
private void button5_Click(object sender, EventArgs e)
{
    if (dataGridView1.SelectedCells.Count > 0)
    {
        int ID;
        ID = dataGridView1.CurrentCell.RowIndex;
        dataGridView1[3, ID].Value = numericUpDown1.Value.ToString();
        float s = 0;
        s = Convert.ToSingle(numericUpDown1.Value) *
Convert.ToSingle(dataGridView1[2, ID].Value);
        dataGridView1[4, ID].Value = s.ToString();
        label3.Text = "";
        float itog = 0;
        for (int i = 0; i < dataGridView1.Rows.Count; i++)
        {
            string summ = dataGridView1[4, i].Value.ToString();
            try {
                float sum_ = Convert.ToSingle(summ);
                itog = itog + sum_;
            }
            catch { }
        }
        label3.Text = itog.ToString() + "Р";
        label5.Text = itog.ToString() + "Р";
        for (int i = 0; i < dataGridView2.Rows.Count; i++)
        {
            string summ = dataGridView2[5, i].Value.ToString();
            try {
                float sum_ = Convert.ToSingle(summ);
                itog = itog + sum_;
            }
            catch { }
        }
        label11.Text = itog.ToString() + "Р";
    }
    button6.PerformClick();
    this.estimateBindingSource.EndEdit();
    this.estimateTableAdapter.Update(this.construction_estimateDataSet.Estimate);
    this.estimaterentBindingSource.EndEdit();
    this.estimate_rentTableAdapter.Update(this.construction_estimateDataSet.Estimate_rent);
}
```

На листинге 3.37 показ код кнопки «Отмена», которая делает невидимым панель изменения количества.

### Листинг 3.37 – Событие «button6\_Click»

```
private void button6_Click(object sender, EventArgs e)
{
    numericUpDown1.Value = 1;
    groupBox1.Visible = false;
}
```

На листинге В.1 описан код нажатия на кнопку «Выгрузить в Word». Происходит выгрузка текущей таблицы в Word с названием «Смета затрат на приобретение» с жирным шрифтом размером 16, даты выгрузки с размером

шрифта 14. Сама таблица с размером шрифта 12, с выделением жирным шрифтом заголовки столбцов и итоговой суммы.

На листинге В.2 описан код нажатия на кнопку «Выгрузить в Excel». Происходит выгрузка текущей таблицы в Excel с названием «Смета затрат на приобретение» с жирным шрифтом размером 12, даты выгрузки с размером шрифта 12. Сама таблица с размером шрифта 11, с выделением жирным шрифтом заголовки столбцов и итоговой суммы с размером шрифта 12.

Выше приведён код для вкладки «Смета затрат на приобретение». Вторая вкладка «Смета затрат на аренду» имеет подобный код с добавлением кнопки «Изменить продолжительность», которая имеет такой же принцип работы, что и кнопка «Изменить количество».

На вкладке «Итоговая смета» расположены две кнопки «Выгрузить в Word» и «Выгрузить в Excel», которые работают по такому же принципу, что и те же кнопки на предыдущих вкладках, но они выгружают данные сразу с обеих таблиц. В Word идёт выгрузка данных последовательно друг под другом, сначала «Смета затрат на приобретение» потом «Смета затрат на аренду» и снизу, в конце общая итоговая сумма. В Excel выгрузка происходит тоже последовательно, но не друг под другом, а с отступом в один столбец.

#### Выводы по третьей части

В ходе данной главы были реализованы 4 формы:

- MDI-форма;
- HabraParser;
- Arrangement;
- Construction Estimate.

Было подробно объяснено их принцип работы, описан программный код. На формах расположен весь необходимый функционал для сбора данных, отображения, фильтраций и вывод данных в Word и Excel.

Формы в рабочем состоянии и готовы к тестированию как отдельно, так и в целом со всем приложением «CASTOPARSER».

## ЗАКЛЮЧЕНИЕ

За время выполнения дипломной квалификационной работы были закреплены и усовершенствованы знания и навыки языков программирования и СУБД.

Проведён анализ имеющихся на рынке программ в количестве восьми штук, в ходе которого была составлена сравнительная таблица, на основании которой были выведены достоинства и недостатки рассматриваемых приложений. Как показал анализ, программы на рынке имеют сложный для понимания интерфейс и в большинстве случаев они предназначены для специализированных и профессиональных пользователей.

Исходя из анализа, целью данной работы стала создание программы, имеющее простой для понимания интерфейс, удобный и необходимый функционал. Программа должна предназначаться обычному не специализированному пользователю, которым может выступать лицо, планирующее провести ремонт помещений различной сложности. Приложение должно предоставить пользователю необходимый функционал, а именно:

- Выбор строительных материалов, инструментов и оборудования как для покупки, так и для приобретения в аренду;
- Возможность сортировать стройматериалы;
- Возможность делать поиск по записям;
- Фильтрацию по различным категориям;
- Движение цены строительных материалов по графику для того, чтобы понимать на основе неё выгоду или невыгоду покупки в определённые дни;
- Подсчёт суммы на основе введенного количества;
- Занесение выбранных записей в строительную смету;
- Итоги по занесённым записям,
- Возможность редактировать смету;
- Возможность очищать смету её от записей;
- Вывод полученные результаты в Word и Excel.

В качестве первых образцов были выбраны два магазина. Первый магазин называется «Castorama», он выступает в роли магазина продающий свои товары и оборудования. Вторым магазином называется «Hilti», данный магазин занимается сдачей в аренду инструментов, оборудования и помещений.

Перед выполнением работы была описана трёхуровневая модель, показывающая движения потоков данных.

Отталкиваясь от построенной модели была реализована программа под названием «CASTOPARSER».

Первым этапом данной программы было создание баз данных серверной и пользовательской частей. Серверная часть записывает и хранит в себе данные о товарах и оборудовании. Пользовательская часть, располагаемая на персональном компьютере пользователя, записывает и хранит в себе строительную смету, в которую заносятся товары, оборудование и другие вещи в ходе использования приложения «CASTOPARSER».

Вторым этапом стала разработка парсера для сбора необходимой информации и записи их в серверную часть базы данных.

Третий этап разработки включает в себя создание формы для отображения имеющихся записей; возможность фильтровать по критериям товары и оборудования; поиск необходимых товаров и оборудования; характеристики выбранных записей; возможность выбрать необходимое число товаров и оборудования, указать срок аренды и получить подсчитанную стоимость выбранных продуктов; занести данные товаров и оборудования в строительную смету; а также данная форма включает в себя построение графика движения цен.

Четвёртый этап заключается в создании формы строительной сметы. Данная форма отображает две таблицы, первая таблица выводит товары и оборудование с учётом их количества, предназначенных для совершения покупок. Вторая таблица выводит оборудование и помещения с учётом их количества, предназначенных для приобретения в аренду. Обе таблицы позволяют пользователю удалить или



редактировать записи, а также выводить как отдельные таблицы, так и все таблицы в Word и в Excel.

Дополнительные магазины для программы «CASTOPARSER» добавляются программистом вручную.

В данной выпускной квалификационной работе были реализованы все цели и задачи, поставленные перед студентом. В ходе работы удалось спарсить данные, скорректировать их и отобразить полученные результаты пользователю с возможностью их анализировать и записывать в строительную смету.

Программа полностью реализована и готова к дальнейшему тестированию.

## БИБЛИОГРАФИЧЕСКИЙ СПИСОК

- 1 АванСМЕТА — эффективное использование — Текст электронный // StavropolRemont [сайт] — URL: <https://stavropolremont.ru/avansmeta-e-ffektivnoe-ispol-zovanie/> (дата обращения: 30.05.2022).
- 2 ДИЗАЙН ИНТЕРЬЕРА 3D — Текст электронный // ДИЗАЙН ИНТЕРЬЕРА [сайт] — URL: <https://interior3d.su/> (дата обращения: 30.05.2022).
- 3 Древовидная структура - Tree structure — Текст электронный // Wiki5 [сайт] — URL: [https://wiki5.ru/wiki/Tree\\_structure](https://wiki5.ru/wiki/Tree_structure) (дата обращения: 29.05.2022).
- 4 Зачем вам DFD-диаграммы или как описать движение потоков данных в бизнес-процессах — Текст электронный // BABOKSCHOOL [сайт] — URL: <https://babok-school.ru/blogs/dfd-diagram-practical-example/#:~:text=Внешняя%20сущность%20-%20объект%20за,данных%20и%20пр.%20Обозначается%20квадратом> (дата обращения: 31.05.2022).
- 5 Инициализация в современном C++ — Текст электронный // Хабр [сайт] — URL: <https://habr.com/ru/company/jugru/blog/469465/> (дата обращения: 29.05.2022).
- 6 Интерфейс (объектно-ориентированное программирование) — Текст электронный // Wikipedia [сайт] — URL: [https://ru.wikipedia.org/wiki/Интерфейс\\_\(объектно-ориентированное\\_программирование\)](https://ru.wikipedia.org/wiki/Интерфейс_(объектно-ориентированное_программирование)) (дата обращения: 02.06.2022).
- 7 Использование DFD: как описать движение данных в бизнес-процессах — Текст электронный // Systems Education [сайт] — URL: <https://systems.education/data-flow-diagrams> (дата обращения: 31.05.2022).
- 8 Класс (программирование) — Текст электронный // Wikipedia [сайт] — URL: [https://ru.wikipedia.org/wiki/Класс\\_\(программирование\)](https://ru.wikipedia.org/wiki/Класс_(программирование)) (дата обращения: 02.06.2022).
- 9 Конструктор (объектно-ориентированное программирование) — Текст электронный // Википедия [сайт] — URL:

- [https://ru.wikipedia.org/wiki/Конструктор\\_\(объектно-ориентированное\\_программирование\)](https://ru.wikipedia.org/wiki/Конструктор_(объектно-ориентированное_программирование)) (дата обращения: 29.05.2022).
- 10 Методология SADT. Функциональное моделирование бизнес-процессов в AllFusion Process Modeler (BPwin) – Текст электронный // Quizlet [сайт] – URL: <https://quizlet.com/604044186/28-Методология-sadt-Функциональное-моделирование-бизнес-процессов-в-allfusion-process-modeler-bpwin-flash-cards/> (дата обращения: 31.05.2022).
- 11 Многодокументный интерфейс – Текст электронный // Wikipedia [сайт] – URL: [https://ru.wikipedia.org/wiki/Многодокументный\\_интерфейс](https://ru.wikipedia.org/wiki/Многодокументный_интерфейс) (дата обращения: 03.06.2022).
- 12 Начало работы с HTML – Текст электронный // Mdn Web Doc [сайт] – URL: [https://developer.mozilla.org/ru/docs/Learn/HTML/Introduction\\_to\\_HTML/Getting\\_started](https://developer.mozilla.org/ru/docs/Learn/HTML/Introduction_to_HTML/Getting_started) (дата обращения: 29.05.2022).
- 13 Оптим-РемСтрой – Текст электронный // КОРС-СОФТ [сайт] – URL: <https://www.kors-soft.ru/oprem3.htm> (дата обращения: 30.05.2022).
- 14 Основы Windows.Forms – Текст электронный // Esate [сайт] – URL: <http://esate.ru/uroki/OpenGL/uroki-OpenGL-c-sharp/osnovi-windows-forms/> (дата обращения: 29.05.2022).
- 15 Представления – Текст электронный // Microsoft Docs [сайт] – URL: <https://docs.microsoft.com/ru-ru/sql/relational-databases/views/views?view=sql-server-ver15> (дата обращения: 29.05.2022).
- 16 Программа ГРАНД-Смета – Текст электронный // ГРАНД-Смета [сайт] – URL: <https://www.grandsmeta.ru/product/pk-grand-smeta#first> (дата обращения: 30.05.2022).
- 17 Протокол TCP – Текст электронный // PC [сайт] – URL: <https://pc.ru/docs/network/tcp> (дата обращения: 29.05.2022).
- 18 Протокол UDP — преимущества, недостатки и применение – Текст электронный // ZvonDozvon [сайт] – URL: <https://zvondozvon.ru/tehnologii/protokoli/udp> (дата обращения: 29.05.2022).

- 19 Смета+ – Текст электронный // FileToGo [сайт] – URL: <https://fileto.net/app/smeta> (дата обращения: 30.05.2022).
- 20 Сметная программа "Сметный Калькулятор 3.5.40 (17.05.2022) " с формой по приказу Минстроя 421 – Текст электронный // Scalc [сайт] – URL: <https://www.scalc.ru/> (дата обращения: 30.05.2022).
- 21 Строительная смета - что это такое и как составляется – Текст электронный // УютСтройПроект [сайт] – URL: <https://usproekt.ru/stroitel'naya-smeta/> (дата обращения: 29.05.2022).
- 22 Тег (языки разметки) – Текст электронный // Википедия [сайт] – URL: [https://ru.wikipedia.org/wiki/Тег\\_\(языки\\_разметки\)](https://ru.wikipedia.org/wiki/Тег_(языки_разметки)) (дата обращения: 29.05.2022).
- 23 Трёхуровневая архитектура – Текст электронный // Википедия [сайт] – URL: [https://ru.wikipedia.org/wiki/Трёхуровневая\\_архитектура](https://ru.wikipedia.org/wiki/Трёхуровневая_архитектура) (дата обращения: 29.05.2022).
- 24 Триггер (базы данных) – Текст электронный // CodeWiki [сайт] – URL: [https://codewiki.imagetube.xyz/code/Триггер\\_\(базы\\_данных\)](https://codewiki.imagetube.xyz/code/Триггер_(базы_данных)) (дата обращения: 02.06.2022).
- 25 Функция (программирование) – Текст электронный // Wikipedia [сайт] – URL: [https://ru.wikipedia.org/wiki/Функция\\_\(программирование\)](https://ru.wikipedia.org/wiki/Функция_(программирование)) (дата обращения: 02.06.2022).
- 26 Хранилище данных – Краткое руководство – Текст электронный // CoderLessons.com [сайт] – URL: <https://coderlessons.com/tutorials/akademicheskii/learn-data-warehouse/khranilishche-dannykh-kratkoe-rukovodstvo> (дата обращения: 31.05.2022).
- 27 Что такое OpenVPN – Текст электронный // cPanel хостинг [сайт] – URL: <https://cpanelhosting.ru/wiki/article/openvpn> (дата обращения: 01.06.2022).
- 28 Что такое URL Комплекс услуг для вашего бизнеса – Текст электронный // SEMANTICA [сайт] – URL: <https://semantica.in/blog/chto-takoe-url.html> (дата обращения: 29.05.2022).

- 29 Что такое парсинг? – Текст электронный // КОМЬЮНИТИ [сайт] – URL: <https://timeweb.com/ru/community/articles/chto-takoe-parser#:~:text=Парсинг%20–%20это%20метод%20индексирования,может%20оказаться%20под%20рукой%20HTML-файл> (дата обращения: 29.05.2022).
- 30 Что такое переменная в программировании – Текст электронный // PROGKIDS [сайт] – URL: <https://progkids.com/blog/what-is-a-variable> (дата обращения: 29.05.2022).
- 31 Что такое хранимая процедура в sql – Текст электронный // САМОУЧИТЕЛЬ [сайт] – URL: <https://programka.com.ua/dokument/administrator/chto-takoe-hranimaja-procedura-v-sql> (дата обращения: 02.06.2022).
- 32 AngleSharp – Текст электронный // AngleSharp Docs [сайт] – URL: <https://anglesharp.github.io/general/introduction> (дата обращения: 02.06.2022).
- 33 Bpwin – Текст электронный // Менеджмент качества [сайт] – URL: <https://www.kpms.ru/Automatization/BPwin.htm> (дата обращения: 29.05.2022).
- 34 Event (computing) – Текст электронный // Wikipedia [сайт] – URL: [https://en.wikipedia.org/wiki/Event\\_\(computing\)#Event\\_handler](https://en.wikipedia.org/wiki/Event_(computing)#Event_handler) (дата обращения: 29.05.2022).
- 35 ID – Текст электронный // Википедия [сайт] – URL: <https://ru.wikipedia.org/wiki/ID> (дата обращения: 29.05.2022).
- 36 IPSec — протокол защиты сетевого трафика на IP-уровне – Текст электронный // IXBT [сайт] – URL: <https://www.ixbt.com/comm/ipsecure.shtml> (дата обращения: 29.05.2022).
- 37 Level of detail (computer graphics) – Текст электронный // Wikipedia [сайт] – URL: [https://translated.turbopages.org/proxy\\_u/en-ru.ru.d6d20072-6299db08-3acc5ad9-74722d776562/https/en.wikipedia.org/wiki/Level\\_of\\_detail\\_\(computer\\_graphics\)](https://translated.turbopages.org/proxy_u/en-ru.ru.d6d20072-6299db08-3acc5ad9-74722d776562/https/en.wikipedia.org/wiki/Level_of_detail_(computer_graphics)) (дата обращения: 29.05.2022).

- 38 Microsoft Visual Studio – Текст электронный // Wikipedia [сайт] – URL: [https://ru.wikipedia.org/wiki/Microsoft\\_Visual\\_Studio](https://ru.wikipedia.org/wiki/Microsoft_Visual_Studio) (дата обращения: 29.05.2022).
- 39 Smeta.RU версия 11 – Текст электронный // СтройСофт [сайт] – URL: <https://www.smeta.ru/products/smeta-ru> (дата обращения: 30.05.2022).
- 40 SQL Server Management Studio – Текст электронный // Wikipedia [сайт] – URL: [https://ru.wikipedia.org/wiki/SQL\\_Server\\_Management\\_Studio](https://ru.wikipedia.org/wiki/SQL_Server_Management_Studio) (дата обращения: 29.05.2022).
- 41 VPN – Текст электронный // Wikipedia [сайт] – URL: <https://ru.wikipedia.org/wiki/VPN> (дата обращения: 01.06.2022).
- 42 WinСмета NEO – Текст электронный // WinСмета [сайт] – URL: <https://winsmeta.com/neo/> (дата обращения: 30.05.2022).

## ПРИЛОЖЕНИЯ

### ПРИЛОЖЕНИЕ А

#### Листинг А.1 – Событие «Parser\_OnNewData2»

```
private void Parser_OnNewData2(object arg1, string[] arg2, int arg3, string arg4)
{
    int count = arg2.Length;
    int flag = 0;
    for (int i = 0; i < count; i++)
    {
        if (flag == 0)
        {
            ListTitles.Items.Add(arg2[i]);
            if (arg3 == 3)
            {
                try
                {
                    if (arg4 == "Hilti")
                    {
                        this.queriesTableAdapter1.Hilti_catalog_categories_insert(arg2[i],
                        "https://arenda.maxipro.ru" + arg2[i + 1], 1, arg4);
                    }
                    else
                    {
                        this.queriesTableAdapter1.Castorama_catalog_categories_insert(arg2[i], arg2[i + 1],
                        arg4);
                    }
                }
                catch { }
            }
            else { }
            flag = 1;
        }
        else
        {
            ListTitles2.Items.Add(arg2[i]);
            flag = 0;
        }
    }
}
```

#### Листинг А.2 – Событие «Parser\_OnNewData3»

```
private void Parser_OnNewData3(object arg1, string[] arg2, int arg3, string[] arg4,
string[] arg5, string name_catalog, string categor)
{
    int count = arg2.Length;
    int flag = 0;
    int j = 0;
    for (int i = 0; i < count; i++)
    {
        if (flag == 0)
        {
            if (arg3 == 2)
            {
                try { j++; }
                catch { }
            }
            else if (arg3 == 14)
            {
                try
                {
                    if (categor == "Hilti")
```

```

        {
this.queriesTableAdapter1.Hilti_catalog_categories_products_insert(arg2[i],
"https://arenda.maxipro.ru" + arg2[i + 1], arg4[j], "https://arenda.maxipro.ru"+ arg5[j],
name_catalog, categor);
        }
        else
        {
this.queriesTableAdapter1.Castorama_catalog_categories_products_insert(arg2[i], arg2[i +
1], arg4[j], arg5[j], name_catalog, categor);
        }
        j++;
    }
    catch { }
    }
    flag =1;
}
else
{
    flag =flag+ 1;
    if (flag == 2)
    {
        flag = 0;
    }
}
}
v = j;
}

```

#### Листинг А.3 – Событие «Parser\_OnNewData33»

```

private void Parser_OnNewData33(object arg1, string[] arg2, int id_)
{
    int count = arg2.Length;
    int p = 1;
    try
    {
        if (count == 0)
        {
            p = 1;
        }
        else { p = Convert.ToInt32(arg2[0]); }
        this.queriesTableAdapter1.Castorama_catalog_categories_insert_pages(id_,p );
    }
    catch { }
}

```

#### Листинг А.4 – Событие «Parser\_OnNewData5»

```

private void Parser_OnNewData5(object arg1, string[] arg2, int id_, string[] arg3,
string[] arg4)
{
    if (arg2.Length != 0 && arg3 == null && arg4 == null)
    {
        int count = arg2.Length;
        string p = "";
        try
        {
            if (count == 2)
            {
                p = arg2[0].ToString();
            }
            else { p = arg2[0].ToString(); }
            string pp = p.Replace(" ", "");
            if (p == null || p==" " ||p==" " ||pp=="")
            {
                p = "Подробных данных нет";
            }
        }
    }
}

```



```

this.queriesTableAdapter1.Hilti_catalog_categories_products_insert_spec(id_, p);
    }
    catch { }
}
else if (arg2.Length == 0 && arg3 == null && arg4 == null)
{
    string p = "";
    p = "Подробных данных нет";
    this.queriesTableAdapter1.Hilti_catalog_categories_products_insert_spec(id_, p);
}
else
{
    int count = arg2.Length;
    string p = "";
    string pp = "";
    string ppp = "";
    try {
        if (count == 2)
        {
            p = arg2[1].ToString();
        }
        else { p = arg2[0].ToString(); }
        pp = arg3[0].ToString();
        ppp = arg4[0].ToString();
        this.queriesTableAdapter1.Castorama_catalog_categories_products_insert_spec(id_, p, pp,
        ppp);
    }
    catch { }
}
}
}

```

Данное событие «Parser\_OnNewData5» заносит в таблицу «Товары» характеристики товара, его цену и артикул или просто характеристики для магазина «Hilti».

## ПРИЛОЖЕНИЕ Б

### Листинг Б.1 – Событие «backgroundWorker1\_DoWork»

```
private void backgroundWorker1_DoWork(object sender, DoWorkEventArgs e)
{
    string[] cat_ = new string[100000];
    string[] cat_2 = new string[100000];
    cat_ = new string[100000];
    cat_2 = new string[100000];
    cat_ = mass();
    List<string> list = new List<string>();
    string[] workerResult = new string[1];
    for (int i = 0; i < cat_.Length; i++)
    {
        string t = cat_[i].Replace("\n", String.Empty);
        int c = t.Length;
        while (t[0].ToString() == " ")
        {
            t = t.Substring(1);
            c = t.Length;
        }
        while (t[c - 1].ToString() == " ")
        {
            t = t.Substring(0, c - 1);
            c = t.Length;
        }
        int o = 0;
        for (int y = 0; y < t.Length; y++)
        {
            if (t[y].ToString() != " ")
            {
                o++;
            }
            else if (t[y].ToString() == " ")
            {
                string h = t;
                h = t.Substring(y - 2, 2);
                if (h != "ая" && h != "ое" && h != "ий" && h != "ый" && h != "ые")
                { o++; }
                else { o++; }
            }
        }
        t = t.Substring(0, o);
        cat_[i] = t;
    }
    int index = 0;
    for (int i = 0; i < cat_.Length; i++)
    {
        string t = cat_[i];
        int flag = 0;
        for (int k = 0; k < cat_2.Length; k++)
        {
            if (t == cat_2[k])
            {
                flag = 1;
                break;
            }
        }
        if (flag != 1)
        {
            cat_2[index] = cat_[i];
            index++;
        }
    }
}
```

```

    }
    int l = 0;
    for (int i = 0; i < cat_2.Length; i++)
    {
        if (cat_2[i] == null)
        {
            l = i ;
            break;
        }
        workerResult[0] = cat_2[i];
        backgroundWorker1.ReportProgress(i, workerResult);
    }
    string[] cat_3 = new string[l];
    for (int i = 0; i < cat_3.Length; i++)
    {
        cat_3[i] = cat_2[i];
    }
    e.Result = cat_3;
}

```

### Листинг Б.2 – Функция «mass»

```

public string[] mass2(string code_)
{
    string connectionString = @"Data Source=.\SQLEXPRESS;Initial
Catalog=DIPL0M;Integrated Security=True";
    var l = new List<string>();
    string pp=code_;
    string sqlExpression = "SELECT * FROM
Castorama_catalog_categories_products_archive_view WHERE Vendor_code='" + pp+"'";
    using (SqlConnection connection = new SqlConnection(connectionString))
    {
        connection.Open();
        SqlCommand command = new SqlCommand(sqlExpression, connection);
        SqlDataReader reader = command.ExecuteReader();
        if (reader.HasRows) // если есть данные
        {
            while (reader.Read()) // построчно считываем данные
            {
                string price_ = reader.GetValue(3).ToString();
                string date_ = reader.GetValue(5).ToString();
                l.Add(price_);
                l.Add(date_);
            }
        }
        reader.Close();
    }
    return l.ToArray();
}

```

### Листинг Б.3 – Событие « dataGridView1\_SelectionChanged »

```

private void dataGridView1_SelectionChanged(object sender, EventArgs e)
{
    try{
        if (dataGridView1.SelectedCells.Count > 0)
        {
            int ID;
            ID = dataGridView1.CurrentCell.RowIndex;
            string photo_link;
            string vendor_code;
            string name_;
            string spec;

```

```

string price_;
photo_link = dataGridView1[4, ID].Value.ToString();
price_ = dataGridView1[3, ID].Value.ToString();
while (price_[0].ToString() == " ")
{ price_ = price_.Substring(1); }
int len = price_.Length;
while (price_[len-1].ToString() == " ")
{
    price_ = price_.Substring(0, len-1);
    len = price_.Length;
}
price_lb.Text = price_ + "₽";
vendor_code = dataGridView1[7, ID].Value.ToString();
name_ = dataGridView1[1, ID].Value.ToString();
spec = dataGridView1[5, ID].Value.ToString();
pictureBox1.Load(photo_link);
textBox3.Clear();
textBox2.Clear();
if (vendor_code != "")
{
    string[] cat_ = new string[100000];
    cat_ = mass2(vendor_code);
    int count;
    count = cat_.Length;
    this.chart1.Series[0].Points.Clear();
    this.chart1.Series[1].Points.Clear();
    chart1.Titles[1].Text = "";
    richTextBox1.Text = spec;
    double sr = 0;
    int a = 0;
    int flag_pov = 0;
    string pov = "";
    if (count != 0)
    {
        pov = cat_[0];
        for (int i = 0; i < count; i = i + 2)
        {
            if (pov != cat_[i])
            {
                flag_pov = 1;
                break;
            }
        }
    }
    for (int i = 0; i < count; i = i + 2)
    {
        sr = sr + Convert.ToDouble(cat_[i]);
        a++;
    }
    sr = sr / a;
    for (int i = 0; i < count; i = i + 2)
    {
        this.chart1.Series[0].Points.AddXY( Convert.ToString(cat_[i + 1].Substring(0, 10)), Convert.ToDouble(cat_[i]) );
        if (flag_pov == 1)
        {
            this.chart1.Series[1].Points.AddXY( Convert.ToString(cat_[i + 1].Substring(0, 10)), sr );
            chart1.Titles[1].Text = "Средняя цена - " + Math.Round(sr, 2).ToString() + "₽";
            chart1.Titles[0].Text = name_;
            if (count > 2)
            {
                chart1.Series[0].ChartType =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Column;
            }
        }
    }
}

```

```
                else{chart1.Series[0].ChartType =  
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Column;  
                }  
            }  
        }  
    }  
    catch{ }  
}
```

## ПРИЛОЖЕНИЕ В

### Листинг В.1 – Выгрузка в Word

```
private void button7_Click(object sender, EventArgs e)
{
    button6.PerformClick();
    WordApp = new Word.Application();
    WordApp.Visible = true;
    WordDocuments = WordApp.Documents;
    WordDocument = WordDocuments.Add();
    WordParagraphs = WordDocument.Content.Paragraphs;
    WordParagraph = WordParagraphs[1];
    WordRange = WordParagraph.Range;
    WordRange.InsertAfter("Смета затрат на приобретение");
    WordRange.Font.Bold = 1;
    WordRange.Font.Size = 16;
    WordParagraph.Alignment = Word.WdParagraphAlignment.wdAlignParagraphCenter;
    WordRange.Collapse(Word.WdCollapseDirection.wdCollapseEnd);
    WordParagraph = WordParagraphs.Add();
    WordParagraph = WordParagraphs[2];
    WordRange = WordParagraph.Range;
    WordRange.InsertAfter("по состоянию на " + DateTime.Now.ToLongDateString() + "\n");
    WordRange.Font.Bold = 0;
    WordRange.Font.Size = 14;
    WordParagraph.Alignment = Word.WdParagraphAlignment.wdAlignParagraphCenter;
    WordRange.Collapse(Word.WdCollapseDirection.wdCollapseEnd);
    WordParagraph = WordParagraphs.Add();
    WordParagraph = WordParagraphs[3];
    WordRange.Font.Bold = 0;
    WordRange.Font.Size = 14;
    WordParagraph.Alignment = Word.WdParagraphAlignment.wdAlignParagraphCenter;
    WordRange = WordParagraph.Range;
    WordApp.ActiveDocument.Tables.Add(WordRange, dataGridView1.RowCount+2,5);
    var table = WordApp.ActiveDocument.Tables[WordApp.ActiveDocument.Tables.Count];
    table.set_Style("Сетка таблицы");
    table.ApplyStyleHeadingRows = true;
    table.ApplyStyleLastRow = false;
    table.ApplyStyleFirstColumn = true;
    table.ApplyStyleLastColumn = false;
    table.ApplyStyleRowBands = true;
    table.ApplyStyleColumnBands = false;
    Thread.Sleep(1000);
    string[] c = { "%", "Наименование", "Цена, ₽", "Количество, шт.", "Сумма, ₽" };
    try {
        for (int i = 1; i < c.Length + 1; i++)
        {
            table.Cell(1, i).Range.Text = c[i - 1];
            if (i == 1)
            { table.Cell(1, i).Width = 25; }
            if (i == 2)
            { table.Cell(1, i).Width = 250; }
            if (i == 4)
            { table.Cell(1, i).Width = 90; }
            if (i == 3 || i>4)
            { table.Cell(1, i).Width = 65; }
            table.Cell(1, i).Range.Font.Bold=1;
            table.Cell(1, i).Range.Font.Size = 12;
        }
    }
    catch { }
    int c_ = 0;
    for (int i = 0; i < dataGridView1.RowCount; i++)
```

```

{
    c_++;
    for (int j = 0; j < dataGridView1.ColumnCount; j++)
    {
        if (j == 0)
        {
            table.Cell(i+2, j+1).Range.Text = c_.ToString();
            table.Cell(i + 2, j + 1).Width = 25;
        }
        else {
            table.Cell(i + 2, j+1).Range.Text = dataGridView1.Rows[i].Cells[j].Value.ToString();
            if (j == 1)
            { table.Cell(i+2, j+1).Width = 250; }
            else if (j == 3)
            { table.Cell(i + 2, j + 1).Width = 90; }
            else if (j != 3 && j!=1)
            { table.Cell(i + 2, j + 1).Width = 65; }
        }
        table.Cell(i + 2, j + 1).Range.Font.Size = 12;
    }
}
try {
    for (int i = 1; i < c.Length + 1; i++)
    {
        int y = dataGridView1.RowCount + 2;
        if (i == 1)
        {
            table.Cell(y, i).Width = 25;
            table.Cell(y, i).Borders[Word.WdBorderType.wdBorderLeft].LineStyle =
            Word.WdLineStyle.wdLineStyleNone;
            table.Cell(y, i).Borders[Word.WdBorderType.wdBorderBottom].LineStyle =
            Word.WdLineStyle.wdLineStyleNone;
        }
        if (i == 2)
        {
            table.Cell(y, i).Width = 250;
            table.Cell(y, i).Borders[Word.WdBorderType.wdBorderLeft].LineStyle =
            Word.WdLineStyle.wdLineStyleNone;
            table.Cell(y, i).Borders[Word.WdBorderType.wdBorderBottom].LineStyle =
            Word.WdLineStyle.wdLineStyleNone;
        }
        if (i == 4)
        { table.Cell(y, i).Width = 90; }
        if (i == 3 || i > 4)
        { table.Cell(y, i).Width = 65; }
        if (i == 3)
        {
            table.Cell(y, i).Borders[Word.WdBorderType.wdBorderLeft].LineStyle =
            Word.WdLineStyle.wdLineStyleNone;
            table.Cell(y, i).Borders[Word.WdBorderType.wdBorderBottom].LineStyle =
            Word.WdLineStyle.wdLineStyleNone;
        }
        if (i == 4)
        { table.Cell(y, i).Range.Text = "Итого:"; }
        if (i == 5)
        {
            float itog = 0;
            for (int p = 0; p < dataGridView1.Rows.Count; p++)
            {
                string summ = dataGridView1[4, p].Value.ToString();
                try {

```

```

        float sum_ = Convert.ToSingle(summ);
        itog = itog + sum_;
    }
    catch { }
    }
    table.Cell(y, i).Range.Text = itog.ToString();
}
table.Cell(y, i).Range.Font.Bold = 1;
table.Cell(y, i).Range.Font.Size = 12;
}
}
finally { }
}

```

## Листинг В.2 – Выгрузка в Excel

```

private void button3_Click(object sender, EventArgs e)
{
    button6.PerformClick();
    excelapp = new Excel.Application();
    excelapp.Visible = true;
    excelapp.SheetsInNewWorkbook = 1;
    excelapp.Workbooks.Add(Type.Missing);
    excelappworkbooks = excelapp.Workbooks;
    excelappworkbook = excelappworkbooks[1];
    excelsheets = excelappworkbook.Worksheets;
    excelworksheet = (Excel.Worksheet)excelsheets[1];
    excelworksheet.Activate();
    excelcells = excelworksheet.get_Range("A3", "A3");
    excelcells.Value2 = "%";
    excelcells = excelworksheet.get_Range("B3", "B3");
    excelcells.Value2 = "Наименование";
    excelcells = excelworksheet.get_Range("C3", "C3");
    excelcells.Value2 = "Цена, ₺";
    excelcells = excelworksheet.get_Range("D3", "D3");
    excelcells.Value2 = "Количество, шт.";
    excelcells = excelworksheet.get_Range("E3", "E3");
    excelcells.Value2 = "Сумма, ₺";
    excelcells = excelworksheet.get_Range("A3", "E3");
    excelcells.Font.Bold = true;
    excelcells.Font.Size = 12;
    string[] c = { "A", "B", "C", "D", "E" };
    int c_ = 0;
    for (int i = 0; i < dataGridView1.RowCount; i++)
    {
        c_++;
        for (int j = 0; j < dataGridView1.ColumnCount; j++)
        {
            if (j == 0)
            {
                excelcells = excelworksheet.get_Range(c[j] + Convert.ToString(i + 4),
c[j] + Convert.ToString(i + 4));
                excelcells.Value2 = c_;
            }
            else
            {
                excelcells = excelworksheet.get_Range(c[j] + Convert.ToString(i + 4),
c[j] + Convert.ToString(i + 4));
                excelcells.Value2 = dataGridView1.Rows[i].Cells[j].Value.ToString();
            }
        }
    }
}

```



```

    }
    excelcells = excelworksheet.get_Range("A3", "E3");
    excelcells.Font.Bold = true;
    excelcells.Font.Size = 12;
    excelcells.HorizontalAlignment = Excel.Constants.xlCenter;
    excelcells = excelworksheet.get_Range("A3", "E" +
Convert.ToSingle(dataGridView1.RowCount + 3));
    excelcells.Columns.AutoFit();
    excelcells = excelworksheet.get_Range("A3", "E" +
Convert.ToSingle(dataGridView1.RowCount + 3));
    excelcells.HorizontalAlignment = Excel.Constants.xlCenter;
    Excel.XlBordersIndex bi = Excel.XlBordersIndex.xlInsideVertical;
    excelcells.Borders[bi].LineStyle = 1;
    bi = Excel.XlBordersIndex.xlInsideHorizontal;
    excelcells.Borders[bi].LineStyle = 1;
    bi = Excel.XlBordersIndex.xlEdgeLeft;
    excelcells.Borders[bi].LineStyle = 1;
    bi = Excel.XlBordersIndex.xlEdgeTop;
    excelcells.Borders[bi].LineStyle = 1;
    bi = Excel.XlBordersIndex.xlEdgeBottom;
    excelcells.Borders[bi].LineStyle = 1;
    bi = Excel.XlBordersIndex.xlEdgeRight;
    excelcells.Borders[bi].LineStyle = 1;
    float itog = 0;
    for (int i = 0; i < dataGridView1.Rows.Count; i++)
    {
        string summ = dataGridView1[4, i].Value.ToString();
        try
        {
            float sum_ = Convert.ToSingle(summ);
            itog = itog + sum_;
        }
        catch { }
    }
    excelcells = excelworksheet.get_Range("D"+Convert.ToSingle(dataGridView1.RowCount + 4),
"D" + Convert.ToSingle(dataGridView1.RowCount + 4));
    excelcells.Value2 = "Итого:";
    excelcells = excelworksheet.get_Range("E" + Convert.ToSingle(dataGridView1.RowCount + 4),
"E" + Convert.ToSingle(dataGridView1.RowCount + 4));
    excelcells.Value2 = itog;
    excelcells = excelworksheet.get_Range("D" + Convert.ToSingle(dataGridView1.RowCount + 4),
"E" + Convert.ToSingle(dataGridView1.RowCount + 4));
    excelcells.Font.Bold = true;
    excelcells.Font.Size = 12;
    excelcells.HorizontalAlignment = Excel.Constants.xlCenter;
    bi = Excel.XlBordersIndex.xlInsideVertical;
    excelcells.Borders[bi].LineStyle = 1;
    bi = Excel.XlBordersIndex.xlInsideHorizontal;
    excelcells.Borders[bi].LineStyle = 1;
    bi = Excel.XlBordersIndex.xlEdgeLeft;
    excelcells.Borders[bi].LineStyle = 1;
    bi = Excel.XlBordersIndex.xlEdgeTop;
    excelcells.Borders[bi].LineStyle = 1;
    bi = Excel.XlBordersIndex.xlEdgeBottom;
    excelcells.Borders[bi].LineStyle = 1;
    bi = Excel.XlBordersIndex.xlEdgeRight;
    excelcells.Borders[bi].LineStyle = 1;
    excelcells = excelworksheet.get_Range("A1", "E1").Cells;
    excelcells.Merge(Type.Missing);
    excelworksheet.Cells[1, 1] = "Смета затрат на приобретение";
    excelcells.Font.Bold = true;

```

```

        excelcells.Font.Size = 12;
        excelcells.HorizontalAlignment = Excel.Constants.xlCenter;
        excelcells = excelworksheet.get_Range("A2", "E2").Cells;
        excelcells.Merge(Type.Missing);
    excelworksheet.Cells[2, 1] = "по состоянию на " + DateTime.Now.ToLongDateString();
    excelcells.Font.Bold = false;
    excelcells.Font.Size = 12;
    excelcells.HorizontalAlignment = Excel.Constants.xlCenter;
    excelcells = excelworksheet.get_Range("A1", "E2").Cells;
    bi = Excel.XlBordersIndex.xlInsideVertical;
    excelcells.Borders[bi].LineStyle = 1;
    bi = Excel.XlBordersIndex.xlInsideHorizontal;
    excelcells.Borders[bi].LineStyle = 1;
    bi = Excel.XlBordersIndex.xlEdgeLeft;
    excelcells.Borders[bi].LineStyle = 1;
    bi = Excel.XlBordersIndex.xlEdgeTop;
    excelcells.Borders[bi].LineStyle = 1;
    bi = Excel.XlBordersIndex.xlEdgeBottom;
    excelcells.Borders[bi].LineStyle = 1;
    bi = Excel.XlBordersIndex.xlEdgeRight;
    excelcells.Borders[bi].LineStyle = 1;
    }

```