Частное учреждение образования

«Колледж бизнеса и права»

РЕАЛИЗАЦИЯ ПРОГРАММНОГО СРЕДСТВА

ДЛЯ СОСТАВЛЕНИЯ СМЕТЫ РЕМОНТНО-

ОТДЕЛОЧНЫХ РАБОТ ПЛИТОЧНОГО МАСТЕРА

Пояснительная записка

к курсовому проекту по дисциплине

«Основы алгоритмизации и программирование»

КП Т.319026.401

Руководитель проекта (А.В.Рогалевич)

Учащийся (З.Н.Шабуневич)

2024

Изм.

Лист

№ докум.

Подпись

Датаа

Лист

3

КП Т.319026.401

Разраб.

Шабуневич З.Н

Провер.

Рогалевич А.В

Реценз.

Н. Контр.

Утверд.

*«Реализация программного средства для составления сметы ремонтно-отделочных работ плиточного мастера»*

Лит.

Листов

КБиП

57

у

**Содержание**

[Введение 4](#_Toc168677653)

[1 Описание задачи 5](#_Toc168677654)

[1.1 Анализ предметной области 5](#_Toc168677655)

[1.2 Постановка задачи 5](#_Toc168677656)

[2 Проектирование системы 7](#_Toc168677657)

[2.1 Требования к приложению 7](#_Toc168677658)

[2.2 Проектирование модели 7](#_Toc168677659)

[2.3 Проектирование структуры базы данных 8](#_Toc168677660)

[2.4 Концептуальный прототип 11](#_Toc168677661)

[3 Описание реализации программного средства 19](#_Toc168677662)

[3.1 Инструменты разработки и применяемые технологии 19](#_Toc168677663)

[3.2 Функции: логическая и физическая организация 19](#_Toc168677664)

[3.3 Входные и выходные данные 22](#_Toc168677665)

[3.4 Функциональное тестирование 23](#_Toc168677666)

[3.5 Описание справочной системы 25](#_Toc168677667)

[4 Применение 26](#_Toc168677668)

[4.1 Назначение программного средства 26](#_Toc168677669)

[4.2 Условия применения 26](#_Toc168677670)

[Заключение 27](#_Toc168677671)

[Список информационных источников 28](#_Toc168677672)

[Приложение А Текст программных модулей 29](#_Toc168677673)

[Приложение Б Формы входных и выходных документов 52](#_Toc168677674)

[Приложение В Результаты работы программы 54](#_Toc168677675)

# Введение

Целью курсового проектирования на тему «Реализация программного средства для составления сметы ремонтно-отделочных работ плиточного мастера» является автоматизация процесса составления сметы, увеличение эффективности работы плиточного мастера, повышение точности расчётов и сокращение времени, затрачиваемого на подготовку сметы. При помощи данного программного средства плиточный мастер сможет быстро и точно оценить затраты на выполнение работ, определить объёмы материалов и оценить общую стоимость ремонтно-отделочных работ.

Актуальностью создания приложения является возможность облегчения расчётов, а также составление документации.

Решение поставленных задач отражено в пояснительной записке.

Пояснительная записка состоит из четырёх разделов и содержит необходимую и достаточную информацию по использованию данного приложения.

В первом разделе «Описание задачи» описываются задачи и цели создания приложения, происходит определение функций, которые должны быть автоматизированы. Проводится анализ предметной области, задачи, которые планируется решать посредством приложения.

Во втором разделе «Проектирование системы» указаны средства защиты будущего приложения, требования к аппаратным и операционным ресурсам. Раскрывается суть спроектированных диаграмм работы алгоритмов и описание внешнего пользовательского интерфейса.

В третьем разделе «Описание реализации программного средства» перечисляются инструменты разработки, необходимые для разработки приложения, организация данных, а также применяемые технологии, описываются основные функции и возможности, структура программы, включая компоненты, модули и их взаимодействие, а также интерфейс, проводится функциональное тестирование и описание справочной системы. проводится проверка работы основных функций программного средства.

В четвёртом разделе «Применение» описывается назначение приложения, анализируется программное обеспечение, которое необходимо для функционирования программы, также описаны условия применения приложения.

В заключении будут выводы, которые были сделаны в течение разработки и создания курсового проекта.

В списке использованных источников перечисляются информационные источники, которые были использованы в процессе разработки приложения.

В ПРИЛОЖЕНИИ А – представлен текст программных модулей приложения.

В ПРИЛОЖЕНИИ Б – представлены формы входных и выходных документов.

В ПРИЛОЖЕНИИ B – представлен результат работы приложения.

# Описание задачи

## Анализ предметной области

Предметной областью решаемой задачи является любое Общество с Ограниченной Ответственностью связанная с работой плиточного мастера. Объектом решаемой задачи является программное средство для составления сметы ремонтно-отделочных работ.

Бизнес-процессы, происходящие в данной области, включают в себя оценку объёма работ, выбор материалов, расчёт сметы, взаимодействие с заказчиками и поставщиками.

Для составления сметы ремонтно-отделочных работ плиточного мастера необходимо учитывать такие параметры, как площадь поверхности, вид используемой плитки, сложность монтажа, стоимость материалов и трудозатраты.

Алгоритм работы программного средства включает в себя следующие шаги: ввод данных о параметрах поверхности и работе, расчёт объёма работ и материалов, учёт стоимости материалов и трудозатрат, формирование сметы.

Компьютерная обработка позволяет значительно повысить эффективность работы плиточного мастера. Автоматизация процесса составления сметы позволяет сократить время на выполнение расчётов, уменьшить вероятность ошибок, упростить взаимодействие с заказчиками и поставщиками, а также повысить качество предоставляемой услуги.

Потенциальная аудитория включает в себя профессиональных строителей, дизайнеров интерьера, архитекторов, заказчиков, а также всех, кто занимается ремонтно-отделочными работами и нуждается в точных и быстрых расчётах стоимости работ.

Самыми популярными аналогами приложения является приложение «Kerama Marazzi», «AllCalc», «Сметчик».

## Постановка задачи

Основная задача создания программы для составления сметы ремонтно-отделочных работ плиточного мастера заключается в разработке удобного и многофункционального приложения для автоматизирования работы плиточного мастера по составлению ремонтно-отделочных работ.

При помощи разрабатываемого приложения были решены следующие бизнес-задачи исследуемой предметной области:

* подсчёт количества материалов: программа должна автоматически вычислять необходимое количество плитки, выбранного размера;
* расчёт стоимости работ: программа должна автоматически рассчитывать общую стоимость работ, включая материалы, затраты на труд и прочие расходы;
* создание сметы: на основе полученных данных программа должна генерировать готовую смету, которую можно предоставить клиенту или использовать для закупки материалов;
* учёт выполненных работ: программа должна позволять вносить изменения в смету при необходимости.

Для решения поставленных задач в настоящее время существует множество аналогичных решений, представленных другими приложениями для составления сметы ремонтно-отделочных работ плиточного мастера. Некоторые из них используют инновационные технологии, такие как аналитика данных для оптимизации бизнес-процессов.

Одним из таких приложений является "Сметчик", программа позволяет автоматически подсчитывать количество материалов и стоимость работ, а также генерировать смету.

Однако, разработка собственного приложения позволит более точные расчёты материалов, возможность выбора специфических параметров для работ и материалов, учёт скрытых затрат и возможность предоставления клиенту интерактивной сметы с визуализацией результатов.

# Проектирование системы

## Требования к приложению

Для обеспечения безопасности и защиты программного средства необходимо реализовать следующие средства защиты:

* аутентификация. Приложение должно обеспечивать возможность аутентификации пользователей для контроля доступа к функционалу и данным через механизм проверки аккаунтов с использованием паролей.

Важно также учесть следующие требования к интерфейсу программного средства:

* интуитивный интерфейс. Приложение должно иметь интуитивно понятный и легко осваиваемый интерфейс пользователя;
* реактивность интерфейса. Интерфейс приложения должен быть отзывчивым и отражать результат операций пользователя моментально;
* соответствие стандартам организации интерфейса. Использование многооконного подхода, реализация управления работой программного приложения с помощью элементов управления;
* требования к выбранной цветовой схеме рабочего экрана; выполнение одной функции с помощью разных элементов управления;
* адаптивность интерфейса. Приложение должно поддерживать адаптивность интерфейса к различным экранам.

Для удобства работы пользователя с программным средством необходимо при разработке форм придерживаться единого стиля оформления. Формы не должны быть перегружены излишней информацией или содержать информацию, не относящуюся к данной форме. Также необходимо предусмотреть защиту данных от удаления и изменения, а также от ввода некорректных данных. В случае ввода некорректных данных или попытке совершить запрещённые действия, пользователь должен быть проинформирован о своих действиях с помощью диалоговых окон.

Пользователю необходимо установить пакет Microsoft Office для переноса сметы из программы в документ Word. Также компоненты последней версии .Net Framework. Установщика программного средства не будет, в связи с тем, что программа сразу будет идти как исполняемый файл. Пользователь сможет скачать папку с программой и всеми необходимыми для неё файлами, уже находящиеся в папке.

## Проектирование модели

Суть диаграммы работы самой сложной реализованной разработчиком функции состоит в следующем: данная функция отвечает за расчет стоимости материалов и работы, необходимых для выполнения ремонтно-отделочных работ. В процессе работы функция учитывает выбранные материалы, их стоимость, количество, которые влияют на итоговую стоимость. Функция выводит в таблицу разбивку стоимости по каждому материалу и виду работ.

Соответствующая диаграмма работы функции представлена в графической части на листе 1.

Суть работы всей программы в целом состоит в следующем: программа предоставляет пользователю возможность составления и изменения сметы, выбора видов работ, материалов, ввода количества и расчет стоимости ремонтно-отделочных работ. После ввода всех необходимых данных программа выполняет расчёт и хранит информацию о стоимости и виде работ.

Диаграмма классов для программы представлена в графической части на листе 2.

## Проектирование структуры базы данных

Главной целью проектирования моделей является отображения функциональной структуры объекта, то есть производимые ими действия и связи между этими действиями. Наиболее распространённым средством моделирования данных являются диаграммы «Сущность-связь», которые предназначены для графического представления моделей данных 10 разрабатываемой программной системы и предлагают некоторой набор стандартных обозначений для определения данных и отношения между ними. С помощью этого вида диаграмм можно описать отдельные компоненты концептуальной модели данных и совокупность взаимосвязей между ними, имеющих важное значение для разрабатываемой системы.

Основными понятиями данной нотации являются понятия сущности и связи. При этом под сущностью понимается произвольное множество реальных и абстрактных объектов, каждый из которых обладает одинаковыми свойствами и характеристиками. В этом случае каждый рассматриваемый объект может иметь уникальное имя или идентификатор, а также отличаться от других экземпляров данной сущности.

Связь определяется как отношение или некоторая ассоциация между отдельными сущностями. Примерами связей могут являться родственные отношения типа «отец-сын».

Графическая модель строится таким образом, чтобы связи между отдельными сущностями отражали не только семантический характер соответствующего отношения, но и дополнительные аспекты обязательности связей, а также кратность участвующих в данных отношениях экземпляров сущностей.

Определим сущности для данного программного средства и построим диаграмму «Сущность-связь». Исследовав предметную область, можно выделить следующие сущности, относящиеся к данному проекту: «Authorization», «Work», «WorkZakup», « », «MaterialZakup», «SaveSmeta», «SaveTableMaterial», «SaveTableWork», «SaveTextBox».

Для сущности «Authorization» атрибутами будут являться:

* идентификационный номер пользователя;
* логин пользователя;
* пароль пользователя.

Для сущности «Work» атрибутами будут являться:

* идентификационный номер;
* наименование работы;
* артикул работы;
* цена в рублях работы;
* тип работы;
* единицы измерения работы;
* страна, где возможно выполнение работы;
* цена, указанная в долларах.

Для сущности «WorkZakup» атрибутами будут являться:

* идентификационный номер;
* наименование работы;
* артикул работы;
* цена в рублях работы;
* тип работы;
* единицы измерения работы;
* страна, где возможно выполнение работы;
* цена, указанная в долларах.

Для сущности «Material» атрибутами будут являться:

* идентификационный номер;
* наименование материала;
* артикул материалов;
* цена в рублях материалов;
* единицы измерения материалов;
* страна производства материала;
* цена, указанная в долларах;
* бренд, в котором указывается оригинал материалов.

Для сущности «MaterialZakup» атрибутами будут являться:

* идентификационный номер;
* наименование материала;
* артикул материалов;
* цена в рублях материалов;
* единицы измерения материалов;
* страна производства материала;
* цена, указанная в долларах;
* бренд, в котором указывается оригинал материалов.

Для сущности «SaveSmeta» атрибутами будут являться:

* идентификационный номер;
* №, который присвоен смете;
* дата составления сметы;
* наименование организации или человека;
* телефон для связи;
* итоговая сумма в рублях всех работ и материалов;
* дата начала выполнения работ;
* плановая дата окончания работ;
* фактическая дата окончания работ.

Для сущности «SaveTableMaterial» атрибутами будут являться:

* идентификационный номер пользователя;
* наименование материала;
* артикул материалов;
* цена в рублях материалов;
* единицы измерения материалов;
* страна производства материала;
* цена, указанная в долларах;
* бренд, в котором указывается оригинал материалов;
* выбранное количество материалов.

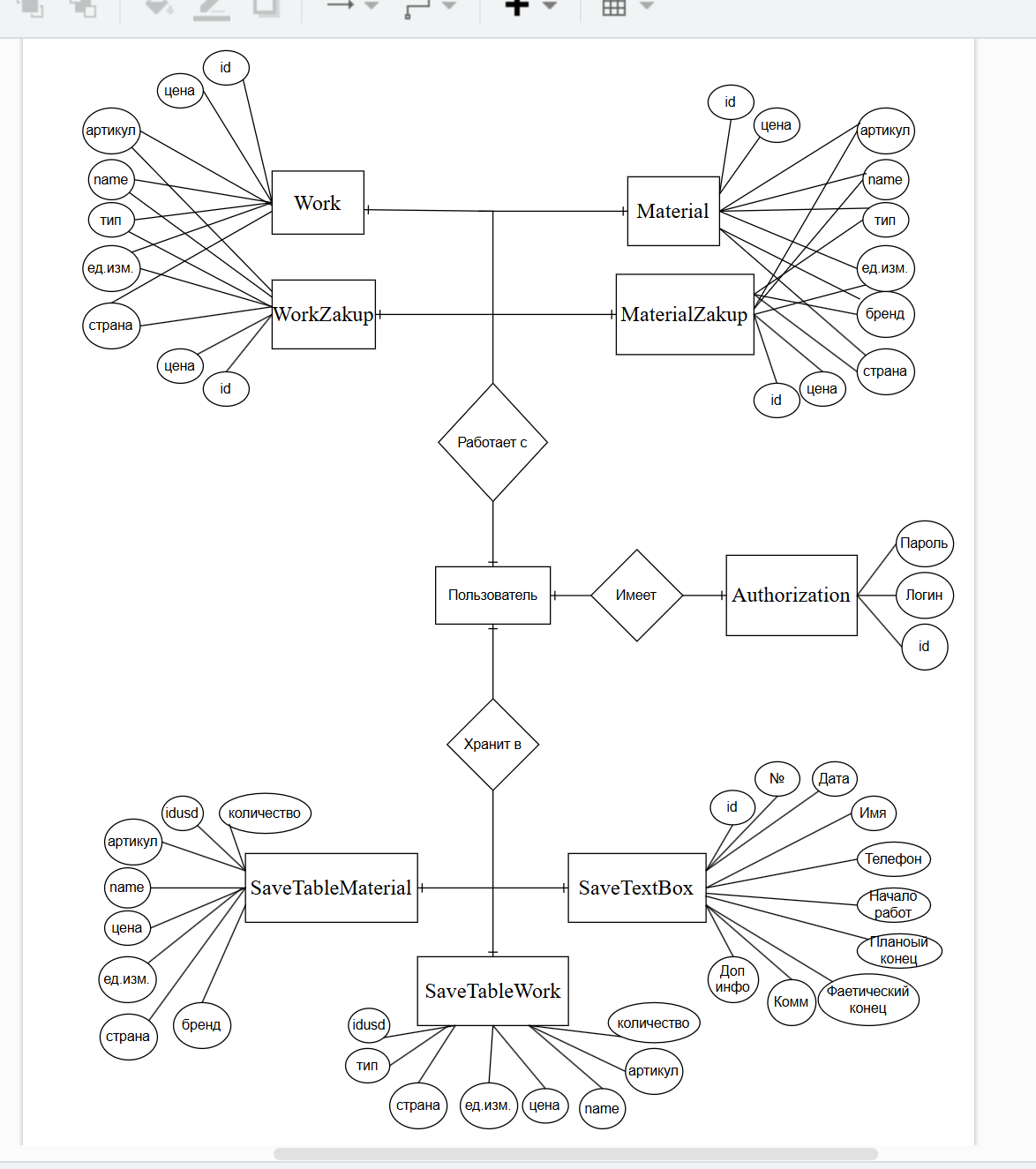
Для сущности «SaveTableWork» атрибутами будут являться:

* идентификационный номер пользователя;
* наименование работы;
* артикул работы;
* цена в рублях работы;
* тип работы;
* единицы измерения работы;
* страна, где возможно выполнение работы;
* цена, указанная в долларах;
* выбранное количество материалов.

Для сущности «SaveTextBox» атрибутами будут являться:

* идентификационный номер пользователя
* номер сметы;
* дата создания сметы;
* имя клиента;
* номер телефона;
* дата начала работ;
* плановая дата окончания работ;
* фактическая дата окончания работ;
* дополнительная информация;
* комментарии;
* рекомендации.

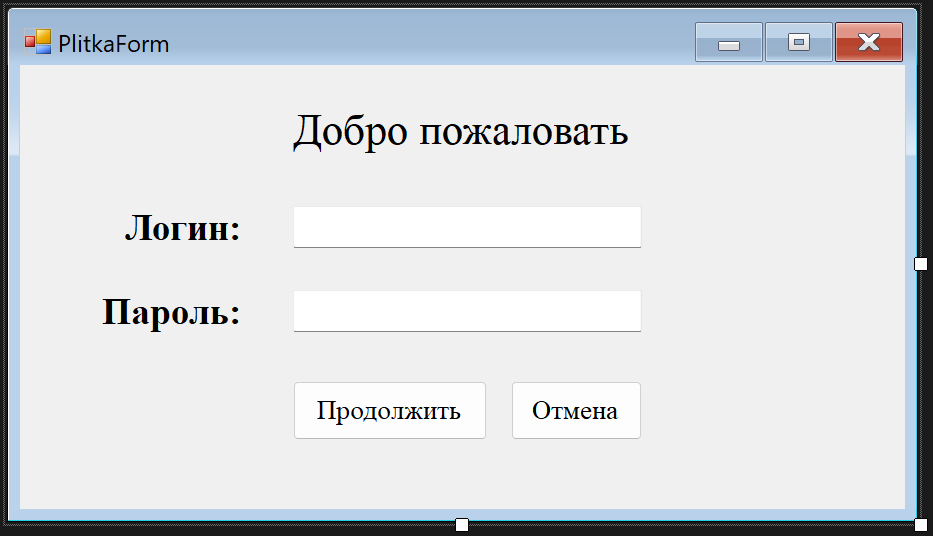
Диаграмма «Сущность-связь» представлена на рисунке 1.



## Концептуальный прототип

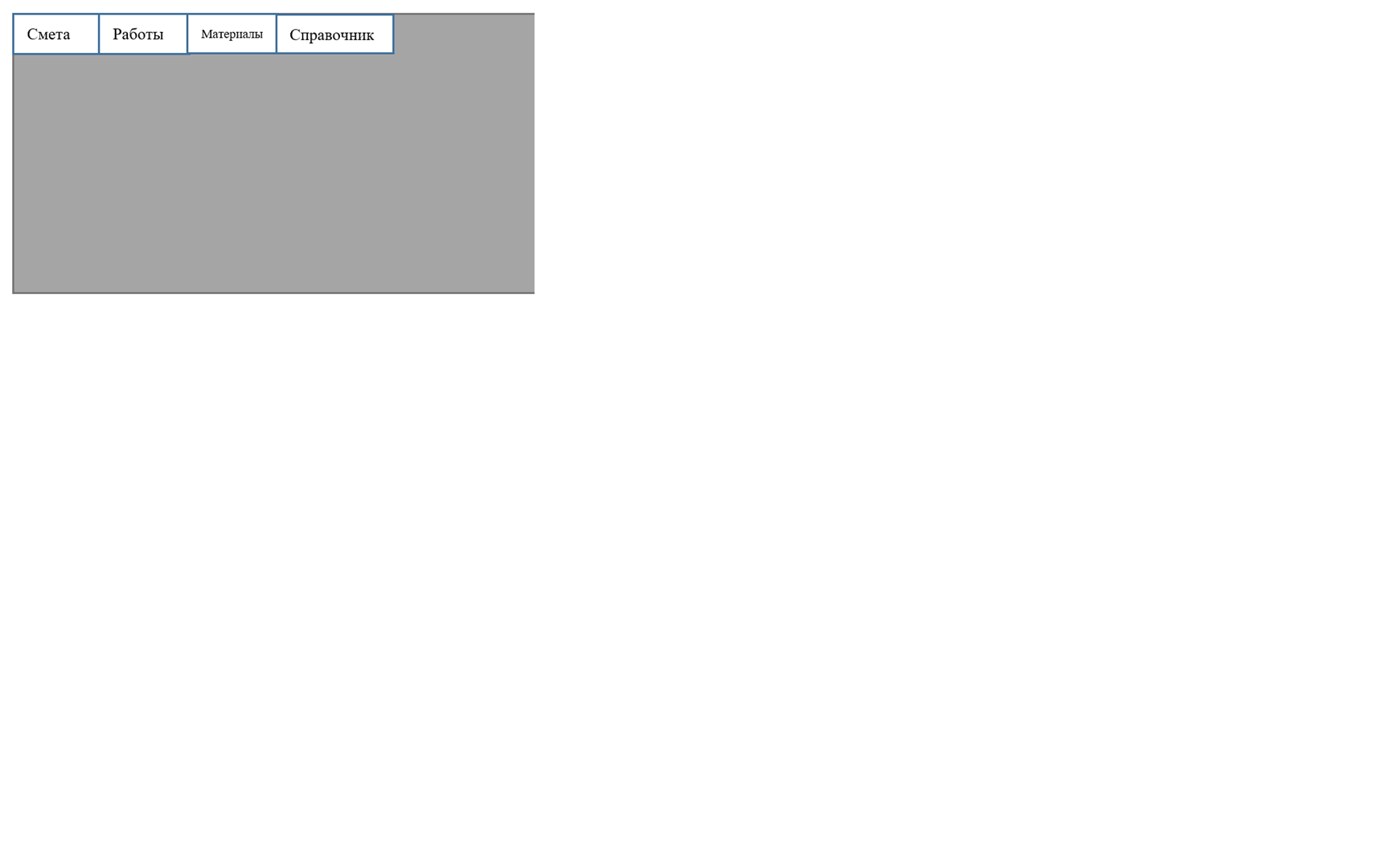
Концептуальный прототип приложения состоит из описания внешнего пользовательского интерфейса, а именно, элементов управления. При создании данного приложения важную роль играют формы, так как они являются основным средством работы с пользователем. Разрабатываемое приложение будет содержать несколько форм. Все формы будут содержать стандартные пользовательские элементы управления. В рабочем режиме программы, пользователю, для удобной навигации, будет предоставлено меню. При входе в программу пользователю необходимо авторизоваться в аккаунт. В окне «Авторизация» пользователю требуется ввести логин и пароль.

Макет формы «Авторизация» представлен на рисунке 2.



При нажатии на кнопку «Продолжить», будет загружена форма «MenuForm», которая представляет собой главное меню, а также задним фоном для всех дочерних окон приложения.

Макет формы «MenuForm» представлен на рисунке 3.



При взаимодействии с разделом меню «Главная», будут доступны кнопки «Настройки рабочего места», «Изменения курса валют». При нажатии на кнопку «Настройки рабочего места» загрузится форма «SettingForm», в которой появятся параметры для настройки интерфейса. При нажатии на кнопку «Изменения курса валют» открывается та же форма.

Макет формы «SettingForm» представлен на рисунке 4.



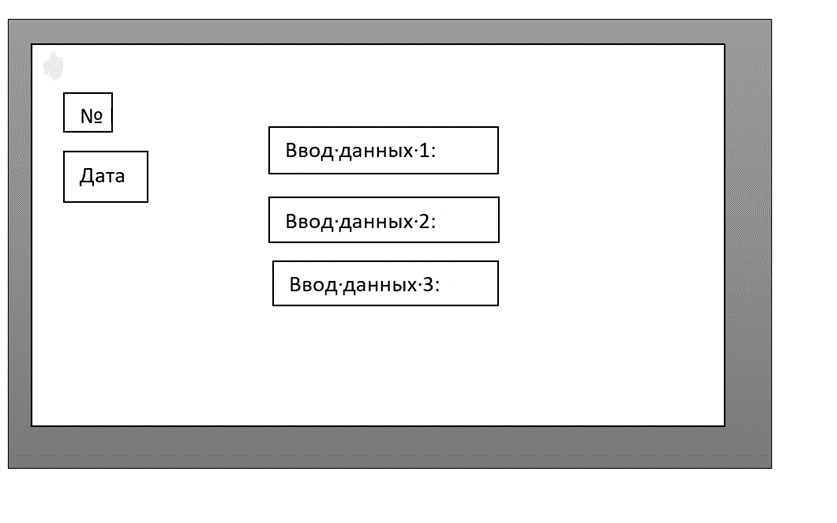
При взаимодействии с разделом меню «Сметы», будет загружена форма «SmetaForm», в которой доступны кнопки «Добавить», «Удалить», «Очистка таблиц», «Выход».

Макет формы «SmetaForm» представлен на рисунке 5.



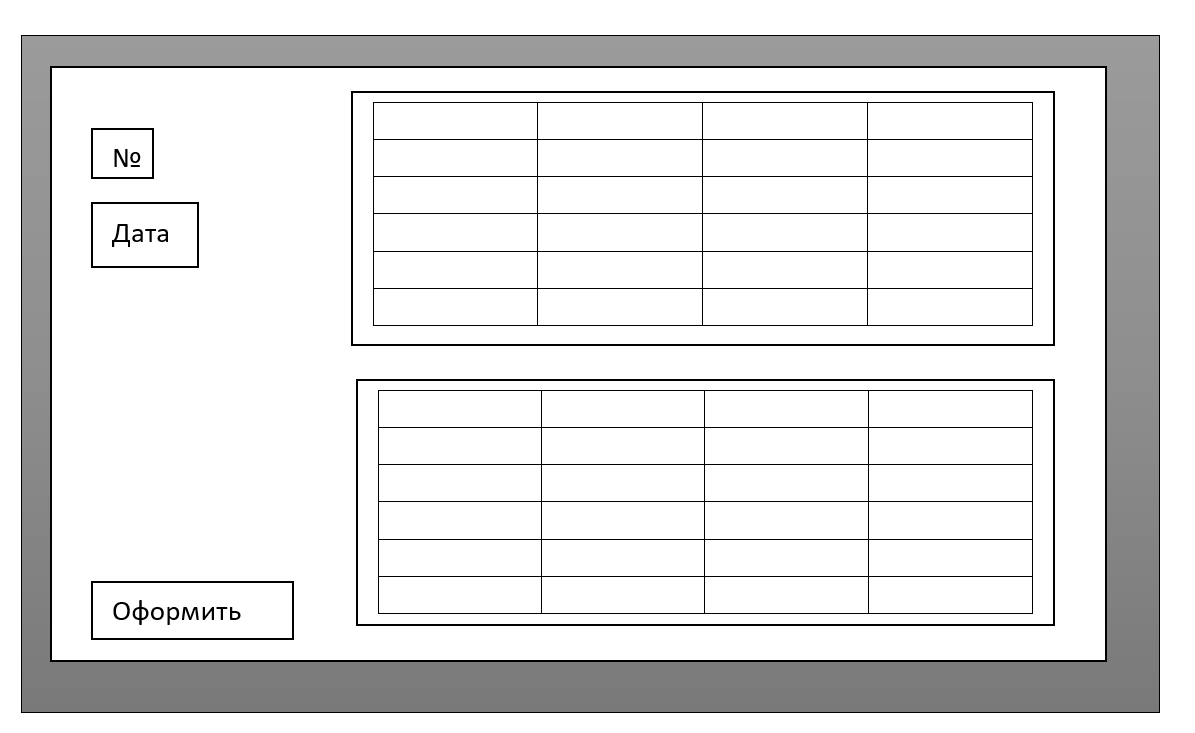
При нажатии на кнопку «Добавить», будет загружена форма «ADDForm», в которой будут доступны поля для ввода данных для составления сметы.

Макет формы «ADDForm» представлен на рисунке 6.



При нажатии на вкладку «Работы/Материалы», будет загружена вкладка для заполнения данных выбранных работах и материалов.

Макет вкладки «Работы/материалы» представлен на рисунке 7.



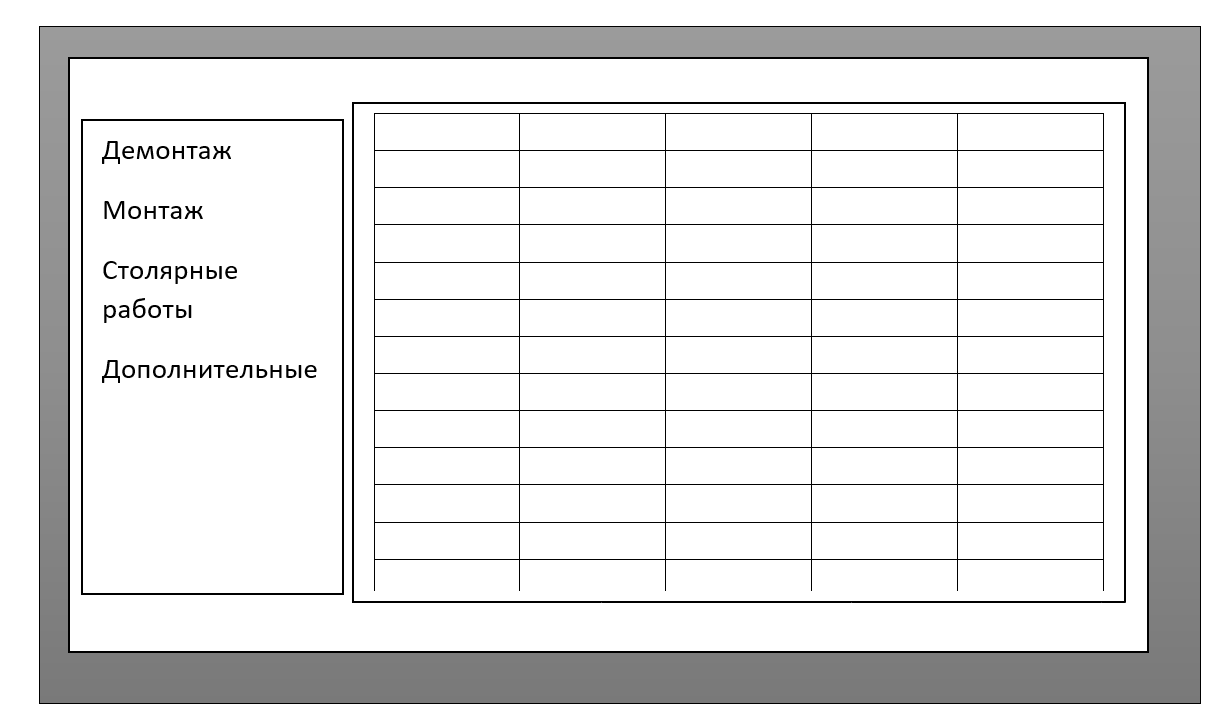
При нажатии на вкладку «Расчёт обкатываемой поверхности», будет загружена вкладка для заполнения данных обкатываемой поверхности, автоматический подсчёт нужного количества плитки.

Макет вкладки «Работы/материалы» представлен на рисунке 8.



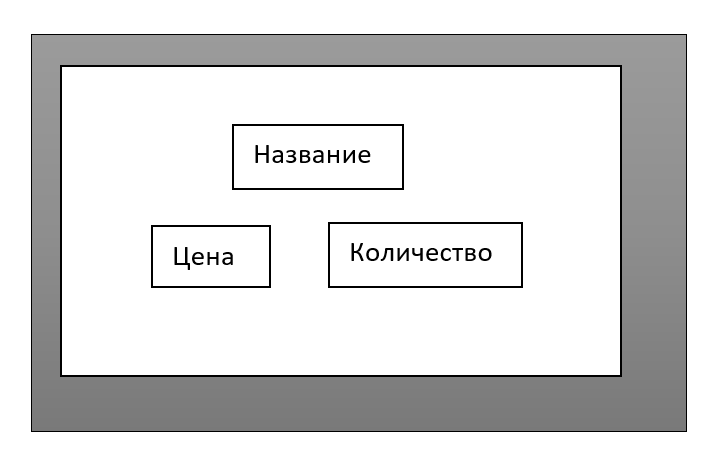
При нажатии на кнопку «Работы», будет загружена форма «AddWorkForm», в которой будет доступна таблица, в которой нужно выбрать необходимые позиции работ.

Макет формы «AddWorkForm» представлен на рисунке 9.



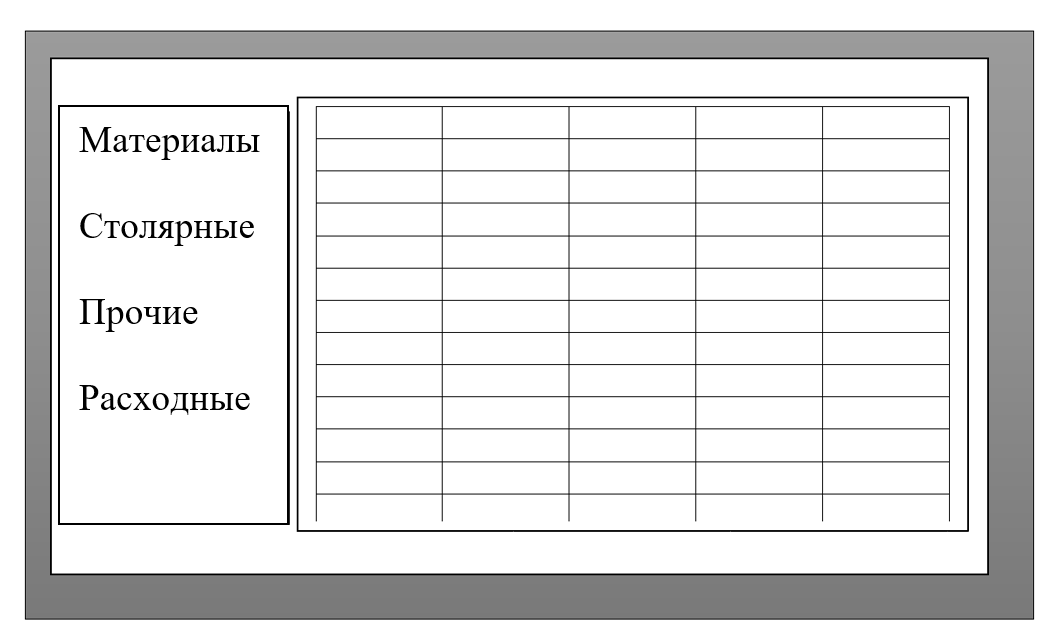
При двойном нажатии на необходимую строку, будет загружена форма «CountForm», в которой будет отображаться название и цена услуги, а также предоставлен выбор количества. При нажатии кнопки «Ок» произойдёт добавление нужной работы/услуги в первую таблицу в форме «ADDForm».

Макет формы «CountForm» представлен на рисунке 10.



При нажатии на кнопку «Материалы», будет загружена форма «AddMaterialForm», в которой будет доступна таблица, в которой нужно выбрать необходимые позиции материалов.

Макет формы «AddMaterialForm» представлен на рисунке 11.



При двойном нажатии на необходимую строку, будет загружена форма «CountForm», в которой будет отображаться название и цена услуги, а также предоставлен выбор количества. При нажатии кнопки «Ок» произойдёт добавление нужной работы/услуги во вторую таблицу в форме «ADDForm».

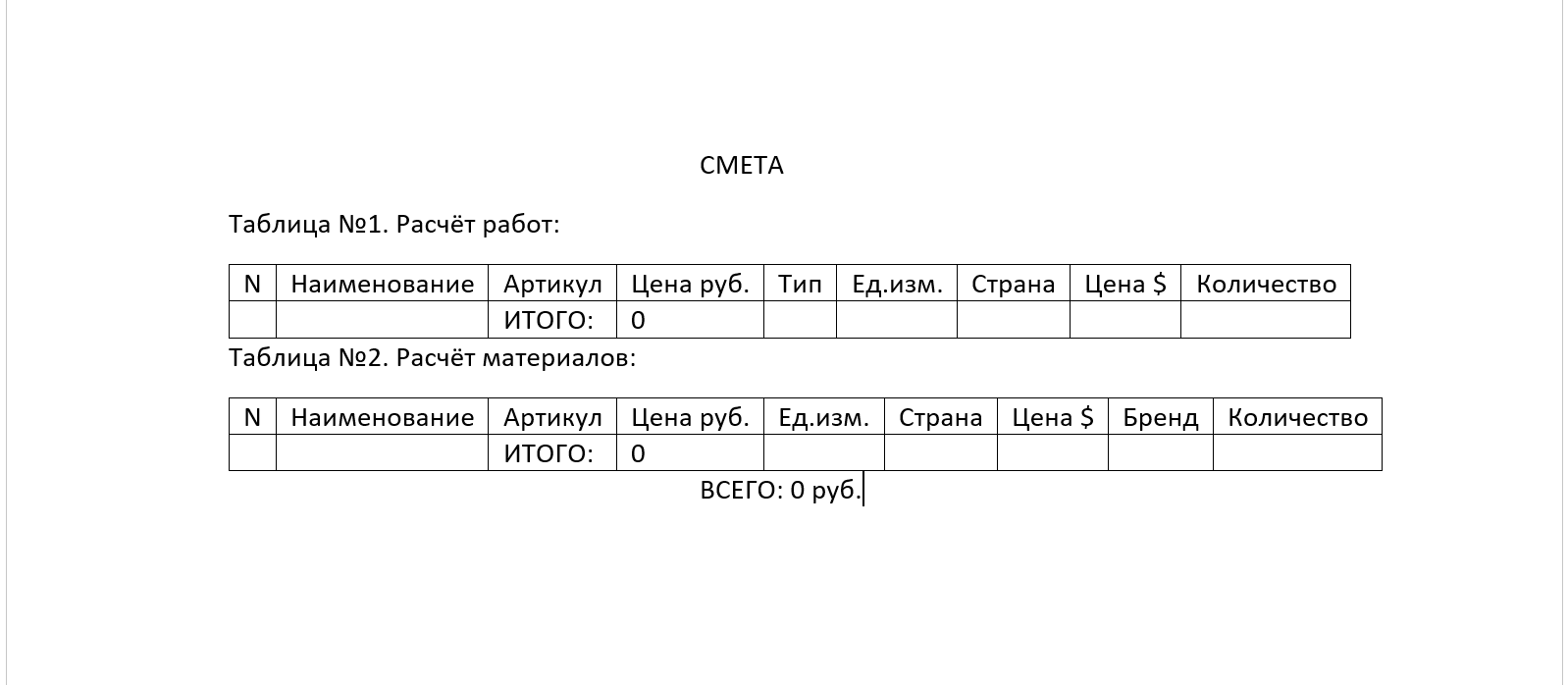
Макет формы «CountForm» представлен на рисунке 10.

Перед нажатием на кнопку «Оформить» на форме «ADDForm» нужно заполнить поля и таблицы данными. После чего при нажатии произойдёт оформление сметы, она будет записана в таблице на форме «SmetaForm».

При двойном нажатии на оформленную строку будет открыта «ADDForm» с ранее заполненными данными. Чтобы изменить данные нужно использовать кнопку «Изменить», которая перезапишет данные.

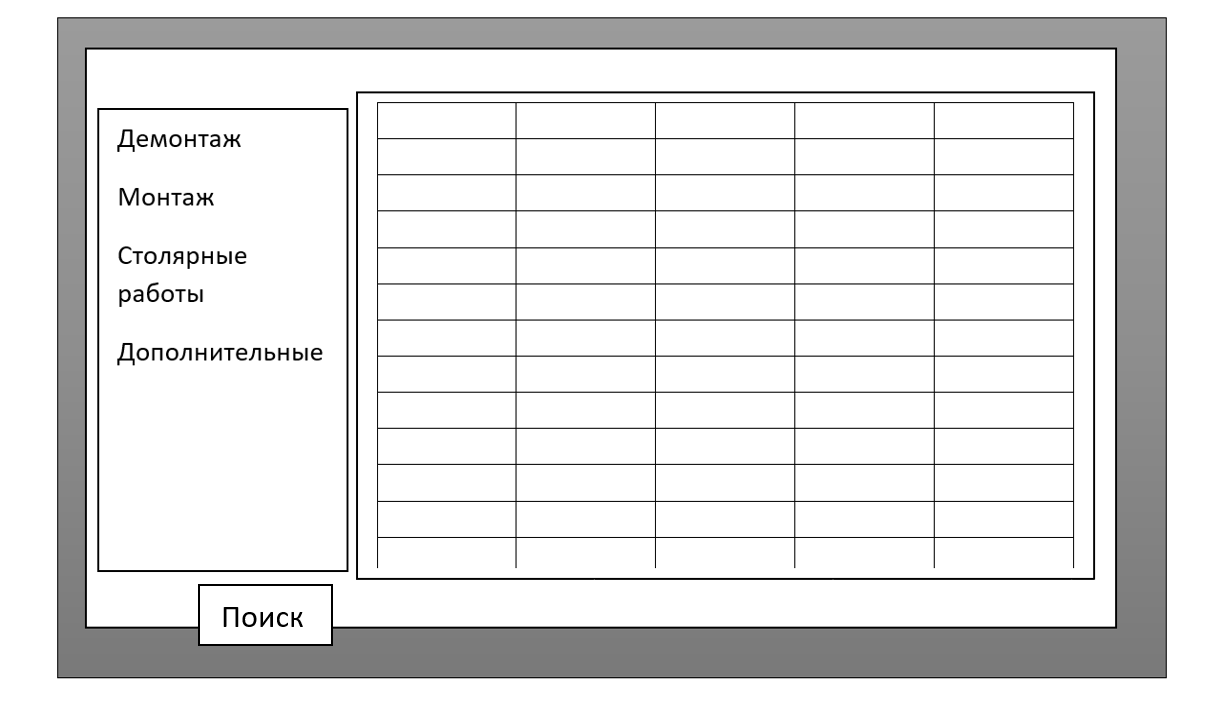
При нажатии на кнопку «Печать» произойдёт автоматическое составление сметы ремонтно-отделочных работ в формате doxc.

Макет сметы ремонтно-отделочных работ представлен на рисунке 12.



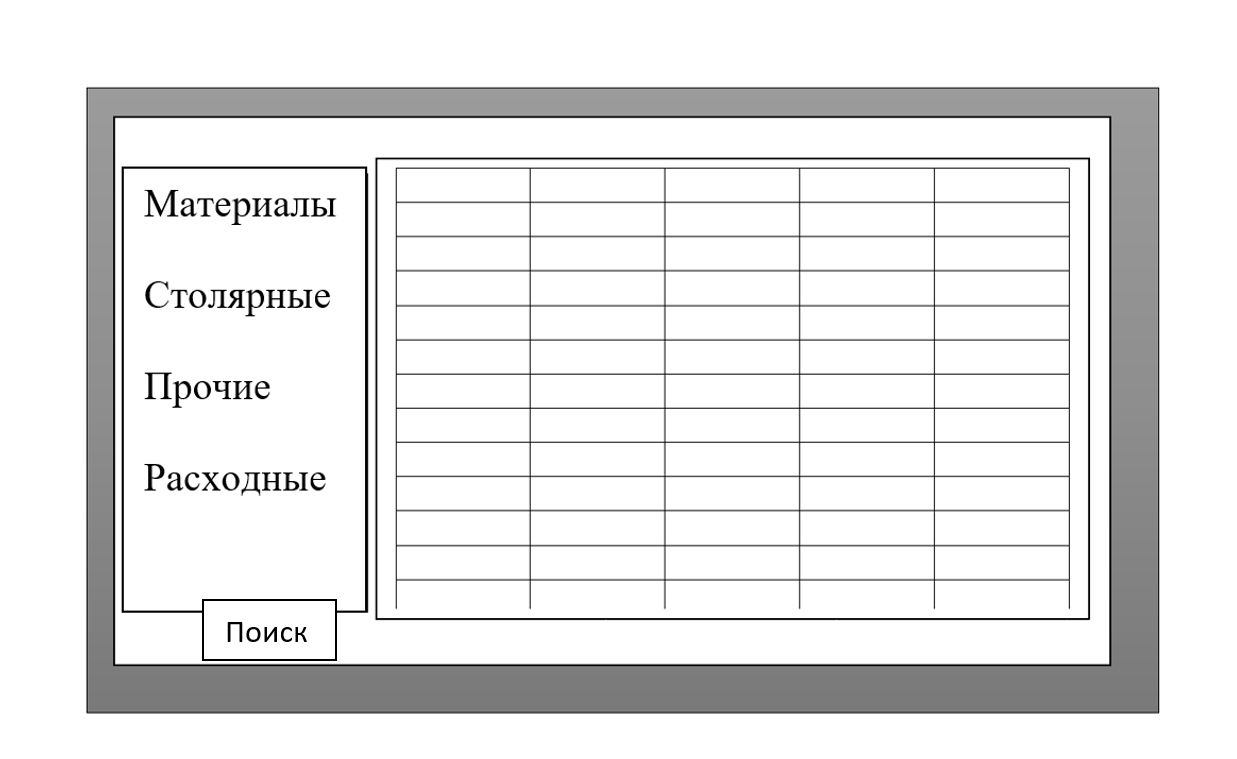
При взаимодействии с разделом меню «Работы», будет загружена форма «WorkForm», в которой доступны услуги работ, представленные в таблице. При нажатии на кнопку «Удалить» будет удалена выбранная строка таблицы.

Макет формы «WorkForm» представлен на рисунке 13.



При взаимодействии с разделом меню «Материалы», будет загружена форма «MaterialForm», в которой доступны услуги материалов, представленные в таблице. При нажатии на кнопку «Удалить» будет удалена выбранная строка таблицы.

Макет формы «MaterialForm» представлен на рисунке 14.



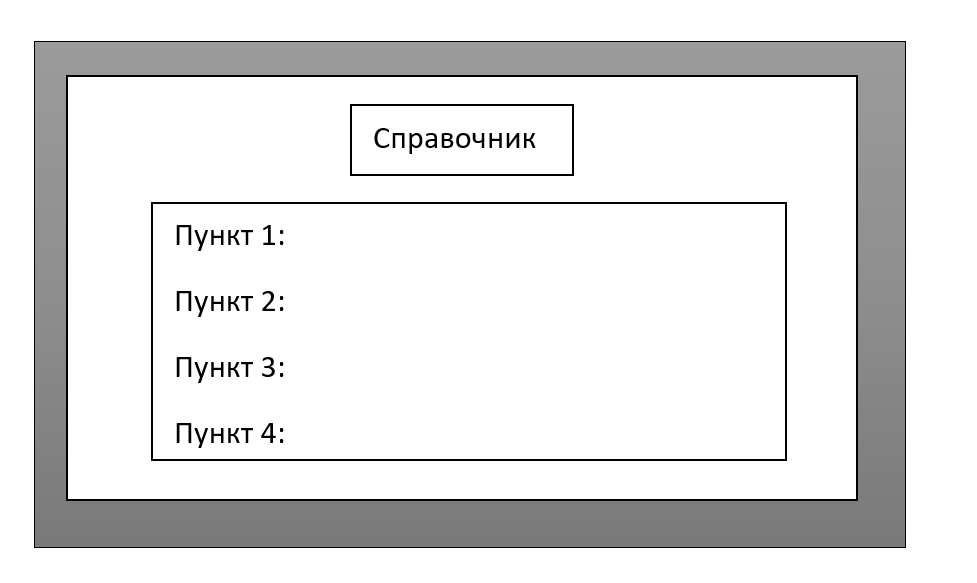
При взаимодействии с разделом меню «Настройки», будет загружена форма «SettingForm», в которой доступны параметры для изменения рабочего места.

Макет формы «SettingForm» представлен на рисунке 3.

При взаимодействии с разделом меню «Выход», произойдёт закрытие всего приложение.

При взаимодействии с разделом меню «Справка», будет отображено всплывающее окно «Справка», в котором находится вся информация по использованию программы.

Макет всплывающего окно «Справка» представлен на рисунке 15.



# Описание реализации программного средства

## Инструменты разработки и применяемые технологии

При разработке программного средства будут использованы следующие программы:

1. операционная система MS Windows 11 64-х разрядная;
2. редактор кода MS Visual Studio Community 2022;
3. язык программирования C++;
4. фреймворк MS Windows Form;
5. система управления базами данных (СУБД) MS SQL;
6. офисный пакет MS Office.

MS Windows 11 Pro была выбрана из-за необходимости тестирования приложения.

MS Visual Studio Community 2022 это лучший инструмент для разработки любого приложения под любую платформу.

Для написания программы был выбран язык программирования C++. Это обусловлено его скоростью работы и целью закрепления изученного материала по данному языку программирования в ходе разработки программного средства.

Windows Form – фреймворк программирования приложений, отвечающий за графический фреймворк пользователя и являющийся частью Microsoft .NET Framework. Данный фреймворк упрощает доступ к элементам интерфейса Microsoft Windows за счёт создания обёртки для существующего Win32 API в управляемом коде.

MS SQL – это система управления базами данных, отличительной особенностью которой является её встраиваемость в приложения. Это значит, что большинство СУБД являются самостоятельными приложениями, взаимодействие с которыми организовано по принципу клиент-сервер. Программа-клиент посылает запрос на языке SQL, СУБД, которая в том числе может находиться на удалённом компьютере, возвращает результат запроса

Microsoft Office – офисный пакет приложений, созданных корпорацией Microsoft для операционных систем Microsoft Windows, Windows Phone, Android, OS X, IOS. В состав этого пакета входит программное обеспечение для работы с различными типами документов: текстами, электронными таблицами, базами данных.

## Функции: логическая и физическая организация

Для реализации графического интерфейса программного средства для составления сметы ремонтно-отделочных работ плиточного мастера была создана единая структура окон Windows Form, которые содержат все необходимые функции программного средства. За каждым пунктом навигационного меню закреплён модуль информационной части. При выборе требуемого пункта меню подключается закреплённый за этим пунктом модуль.

Обязательными функциями программного средства являются добавление данных. Функции, которые выполняет программа – ведение данных в базе данных, поиск данных по указанному критерию. Для успешного выполнения вышеперечисленных функций программное средство необходимо запустить на персональном компьютере (ПК). После запуска оно отобразит окно с авторизацией. Где пользователю потребуется ввести свои логин и пароль. Затем, после успешной авторизации, перед ним откроется окно главного меню, где будет доступ ко всем функциям. При разработке программного средства были созданы запросы, позволяющие выполнять ряд действий в приложении.

Функция для создания сметы и отображения в таблице. Ниже представлен код.

if (!String::IsNullOrWhiteSpace(textBox9->Text))

{

String^ textBoxValue = textBox9->Text;

bool matchFound = true;

for (int i = 0; i < dataNumId->Length; i++) {

if (dataNumId[i]->Trim() == textBoxValue->Trim()) {

matchFound = false;

break;

}

}

if (matchFound)

{

try {

db->openConnection();

db->InsertIntoSaveWork(dataGridView1, textBox9);

db->InsertIntoSaveMaterials(dataGridView3, textBox9);

String^ data1 = textBox9->Text;//Номер сметы

String^ data2 = textBox10->Text;//Дата

String^ data3 = textBox1->Text;//наименование

String^ data4 = textBox2->Text;//тел

String^ data5 = textBox3->Text;//начало работ

String^ data6 = textBox4->Text;//окончание план

String^ data7 = textBox5->Text;//окончание работ

String^ data8 = textBox6->Text;//Доп инфа

String^ data9 = textBox7->Text;//Комм

String^ data10 = textBox8->Text;//Рекомендации

double total1 = Convert::ToDouble(label9->Text);

db->InsertIntoSaveTextBox(data1, data2, data3, data4, data5, data6, data7, data8, data9, data10, textBox9);

deleg(data1, data2, data3, data4, data5, data6, data7, data8, data9, data10, total1);

db->closeConnection();

MessageBox::Show("Успешно добавлен");

this->Hide();

}

catch (Exception^ ex) {

MessageBox::Show(ex->Message);

}

}

else

{

MessageBox::Show("Вы уже добавили смету с таким номером.\n Если хотите добавить изменения выберите кпоку изменить.");

}

}

else

{

MessageBox::Show("Введите номер сметы чтобы оформить её.");

}

Для добавления сметы был создан запрос «InsertIntoSaveSmeta». Ниже представлен код запроса «InsertIntoSaveSmeta».

INSERT INTO SaveSmeta ([№],Дата, Наименование, Телефон, [Сумма, руб.], [Начало работ], [Окончание план], [Конец работ]) Values (@value1,@value2,@value3,@value4,@value5,@value6,@value7,@value8

Для добавления материалов в смету был создан запрос «InsertIntoSaveMaterials». Ниже представлен код запроса «InsertIntoSaveMaterials».

INSERT INTO SaveTableMaterial (Наименование, Артикул, [Цена руб.],[Ед.изм.],Страна,[Цена $], Бренд, [idusd], Количество) Values (@value1,@value2,@value3,@value4,@value5,@value6,@value7,@value8,@value9

Для добавления работ в смету был создан запрос «InsertIntoSaveWork». Ниже представлен код запроса «InsertIntoSaveWork».

INSERT INTO SaveTableWork (Наименование, Артикул, [Цена руб.],Тип,[Ед.изм.],Страна,[Цена $], [idusdt], Количество) Values (@value1,@value2,@value3,@value4,@value5,@value6,@value7,@value8,@value9

Для добавления иных полей в смету был создан запрос «InsertIntoSaveTextBox». Ниже представлен код запроса «InsertIntoSaveTextBox».

INSERT INTO SaveTextBoxs (data1,data2,data3,data4,data5,data6,data7,data8,data9,data10,idusd) Values (@value1,@value2,@value3,@value4,@value5,@value6,@value7,@value8,@value9,@value10,@value11

Для отображения выбора материалов был создан запрос «getTableMaterial». Ниже представлен код запроса «getTableMaterial».

select Наименование, Артикул, [Цена руб.],[Ед.изм.],Страна,[Цена $],Бренд from Material

Для отображения выбора работ был создан запрос «getTableWork». Ниже представлен код запроса «getTableWork».

select Наименование, Артикул, [Цена руб.], Тип, [Ед.изм.], Страна, [Цена $] from Worki

Весь код программы представлен в приложении А.

## Входные и выходные данные

Входные данные – это данные, которые передаются в программное средство для ведения базы данных. Входные данные могут быть представлены в различных форматах, таких как строки, числа, булевы значения, таблицы и т.д. Эти данные могут быть получены от пользователя через интерфейс программы или базу данных. Все данные, введённые пользователем, хранятся в базе данных.

Входными являются данные, введённые пользователем в таблицы базы данных: «Authorization», «SaveSmeta», «SaveTableMaterial», «SaveTableWork», «SaveTextBox». Описание входных данных представлено в таблице 1.

Таблица 1 – Входные и выходные данные программного средства

|  |  |
| --- | --- |
| Наименование таблицы | Наименование полей для заполнения |
| «Authorization» | Surname, Password |
| «SaveSmeta» | №, Дата, Наименование, Телефон, Сумма, руб., Начало работ, Окончание план, Конец работ |
| «SaveTableMaterial» | Наименование, Артикул, Цена руб., Ед.изм., Страна, Цена $, Бренд, idusd, Количество |
| «SaveTableWork» | Наименование, Артикул, Цена руб., Тип, Ед.изм. ,Страна, Цена $, Количество |
| «SaveTextBox» | Номер, дата создания, имя; телефон, дата начала работ, плановая дата окончания, фактическая дата окончания, дополнительная информация, комментарии, рекомендации. |

Выходные данные – это результат обработки входных данных программного средства. Эти данные также могут быть представлены в различных форматах, в зависимости от того, что программа, или функция, должна вернуть. Например, это может быть число, строка, булево значение, массив или таблица.

В приложении выходными данными являются заполненные таблицы с выбранными данными.

## Функциональное тестирование

Таблица 2 – Функциональное тестирование

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Модуль / Функция | Шаги выполнения | Ожидаемый результат | Фактический результат |
| 1 | 2 | 3 | 4 | 5 |
| 11 | Вход в приложение | 1. Запустить приложение. 2. Заполнить поле «Логин» и «Пароль». 3. Нажать на кнопку продолжить. | Открывается страница авторизации. Произведён вход в аккаунт. Открывается главная форма приложения | Результат представлен на рисунке 20 |
| 2 | Создание новой сметы | 1. Нажать кнопку «Сметы». 2. Нажать кнопку «Добавить». 3. Заполнить вкладку «Основное» данными. | Открывается форма со сметами. Открывается окно заполнения данных. | Результат представлен на рисунке 21, 22 |

Продолжение таблицы 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| 2 |  | 1. Заполнить вкладку «Работы/Материалы» данными. 2. Заполнить вкладку «Расчёт обкатываемой поверхности» данными. 3. Нажать кнопку «Оформить» для составления сметы. | После полного заполнения данными происходит оформление сметы. |  |
| 3 | Изменение ранее созданной сметы | 1. Двойным нажатием в таблице выбрать нужную смету. 2. Ввести или изменить данные. 3. Нажать кнопку «Изменить». | Откроется форма для заполнения данными. После внесения изменений и нажатии соответствующей кнопки изменения запишутся в ранее добавленную строку в таблице. | Результат представлен на рисунке 23 |
| 4 | Печать сметы | 1. Двойным нажатием в таблице выбрать смету. 2. Нажать кнопку «Печать». | Откроется форма заполнения. Запуститься автозаполнение сметы в формате doxc. | Результат представлен на рисунке 24 |
| 5 | Очистка таблице | 1. Нажать кнопку «Сметы». 2. Нажать кнопку «Очистка таблиц». | Сброс данных всех таблице | Результат представлен на рисунке 25 |
| 6 | Удаление одной сметы | 1. Нажать кнопку «Сметы». 2. Выбрать строку, которую хотите удалить. 3. Нажать кнопку «Удалить». | Сброс данных одной строки. | Результат представлен на рисунке 26 |
| 7 | Открытие справочника работ | 1. Нажать кнопку «Работы». | Открывается окно со справочником работ | Результат представлен на рисунке 27 |

Продолжение таблицы 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| 8 | Поиск в справочнике работ | 1. Нажать кнопку «Работы». 2. Ввести в поле данные о поиске | В таблице справочника будет выделена нужная строка | Результат представлен на рисунке 28 |
| 9 | Открытие справочника материалов | 1. Нажать кнопку «Материалы». | Открывается окно со справочником материалов | Результат представлен на рисунке 29 |
| 10 | Поиск в справочнике работ | 1. Нажать кнопку «Материалы». 2. Ввести в поле данные о поиске | В таблице справочника будет выделена нужная строка | Результат представлен на рисунке 30 |

## Описание справочной системы

Справочная система представлена в виде сообщений, которые отображаются с помощью MessageBox в приложении. Для вызова справочника требуется нажать кнопку «Справочник».

Страница справочной системы представлена на рисунке 15

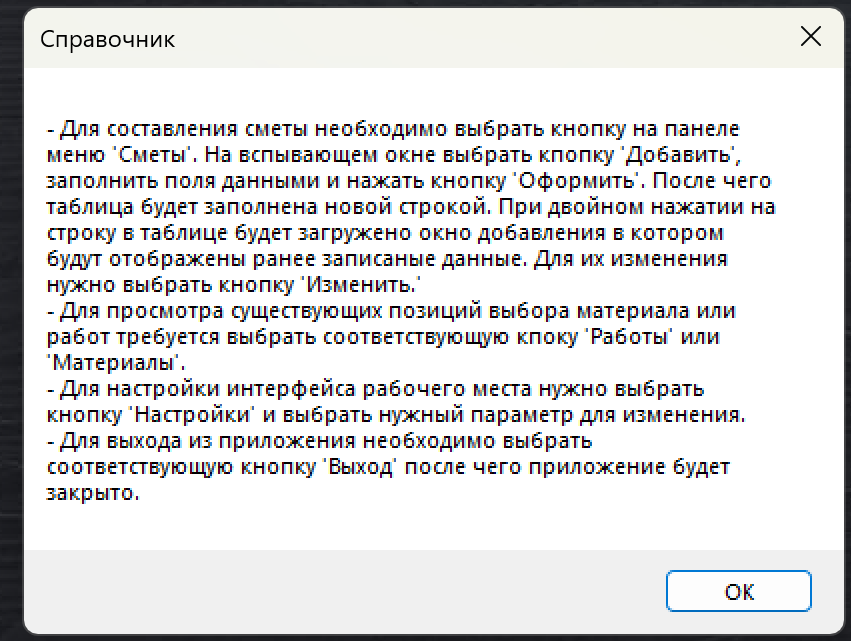


Рисунок 15

# Применение

## Назначение программного средства

Программное средство для составления сметы ремонтно-отделочных работ плиточного мастера предназначено для автоматизации и упрощения процесса планирования, увеличение эффективности работы плиточного мастера, повышение точности расчётов и сокращение времени, затрачиваемого на подготовку сметы, расчёта и составления сметы на ремонтные работы и материалы, связанные с укладкой плитки и обычными работами.

## Условия применения

Десктопное приложение предусмотрено для работы на устройствах с операционной системой Windows. Для экспорта отчётов необходимо иметь программное средство Microsoft Word. Для корректной работы приложение необходимо библиотеки и фреймворки: .NET Framework 4.5 и выше. Для работы компонентов, написанных на C++, необходимо наличие установленного пакета Microsoft Visual C++.

# Заключение

Задача курсового проекта заключалась в разработке программного средства для составления сметы ремонтно-отделочных работ плиточного мастера.

Выполнение: Программное средство было разработано с использованием Windows Form. Все задачи, поставленные перед разработкой, были выполнены. Были реализованы необходимые функции.

Описание функций: Приложение позволяет создавать новые сметы, редактировать ранее созданные сметы. Просматривать в любой момент необходимые данные. Возможен выбор различных работ и материалов, а также расчёт обкатываемой поверхности с автоматическим подсчётом количества плиточек. Реализовано автоматическое составление сметы в формате docx.

Программа реализована в полном объёме и в соответствии с заданными требованиями, полностью отлажена и протестирована. Поставленные задачи выполнены.

К достоинствам данного программного средства можно отнести: интуитивно понятный интерфейс и простоту в использовании.

К недостаткам данного программного средства можно отнести: ограниченный выбор материалов и работ.

К направлению для усовершенствования программного средства можно отнести: расширение доступных работ и материалов, добавление новых методов подсчёта, а также улучшить графический интерфейс. Для достижения этих целей можно использовать более современные технологии и фреймворки.

# Список информационных источников

1. Багласова, Т.Г. Методические указания по выполнению дипломного проекта для учащихся по специальности 2-40 01 01 «Программное обеспечение технологий» / Т.Г. Багласова. – Минск : КБП, 2017. – 30 c.
2. Багласова, Т.Г. Методические указания по оформлению курсовых и дипломных проектов / Т.Г. Багласова, К.О. Якимович. – Минск : КБП, 2013. – 29 c.
3. Тепляков, С. Паттерны проектирования на платформе .NET / С.Тепляков. – СПб. : Питер, 2015. – 320 с.
4. Общие требования к тестовым документам : ГОСТ 2.105-95. – Введ. 01.01.1996. – Минск : Межгос. совет по стандартизации, метрологии и сертификации, 1995. – 84 с.
5. Обзор обновлений и новых функций Windows 10 [Электронный ресурс]. – Microsoft, 2020. – Режим доступа : https://www.microsoft.com/ru-ru/windows/features. – Дата доступа : 22.05.2024.
6. Visual Studio 2019 [Электронный ресурс]. – Microsoft, 2020. – Режим доступа : https://visualstudio.microsoft.com/ru/vs. – Дата доступа : 22.05.2024.
7. Осипов Е.А. Word 2021 в примерах: учебное пособие / Е.А Осипов, Е.М. Карчевский, И.Е. Филиппов. – Казань: Казанский федеральный университет, 2022. – 120 с.
8. C++ – Википедия [Электронный ресурс]. – Microsoft, 2020. – Режим доступа : <https://shorturl.at/rAQTY> – Дата доступа : 20.05.2024.
9. Объектно-ориентированный анализ и проектирование с примерами приложений / Гради Буч [и др.]. – 3-е изд. – М. : ООО «И.Д. Вильямс», 2008. – 720 с.
10. Руководство по MS SQL [Электронный ресурс]. – Режим доступа: https://metanit.com/SQL. – Дата доступа: 01.06.2025.
11. Руководство по расчётам [Электронный ресурс]. – Режим доступа: <https://lb-ceramics.ru/ideas/basics/kak-rasschitat-kolichestvo-plitki/> − Дата доступа: 05.06.2024

# Приложение А Текст программных модулей

MenuForm.h

#pragma endregion

private: System::Void сметыToolStripMenuItem1\_Click(System::Object^ sender, System::EventArgs^ e) {

SmetaForm^ f3 = gcnew SmetaForm();

f3->Show();

}

private: System::Void MenuForm\_Load(System::Object^ sender, System::EventArgs^ e) {

this->Text = "Меню";// Задаёт название формы

this->Top = 0; //Задаёт расстояние верхней границы

this->Left = 0; //левой границы

this->Width = Screen::PrimaryScreen->WorkingArea.Width;

this->Height = Screen::PrimaryScreen->WorkingArea.Height;

PlitkaForm^ f4 = gcnew PlitkaForm();

f4->Show();

}

private: System::Void выходToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) {

auto result = MessageBox::Show(

"Вы действительно хотите выйти из программы?",

"Подтвердите выход",

MessageBoxButtons::YesNo,

MessageBoxIcon::Question);

if (result == System::Windows::Forms::DialogResult::Yes) {

this->Close();

}

}

private: System::Void работыToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) {

WorkForm^ f1 = gcnew WorkForm();

f1->Show();;

}

private: System::Void материалыToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) {

MaterialsForm^ f2 = gcnew MaterialsForm();

f2->Show();

}

private: System::Void настройкиToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) {

SettingsForm^ f5 = gcnew SettingsForm();

if (f5->Visible == true)

{

f5->Close();

}

else f5->Show();

}

private: System::Void настройкаРабочегоМестаToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) {

SettingsForm^ f5 = gcnew SettingsForm();

if (f5->Visible == true)

{

f5->Close();

}

else f5->Show();

}

private: System::Void изменениеКурсаВалютToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) {

SettingsForm^ f5 = gcnew SettingsForm();

if (f5->Visible == true)

{

f5->Close();

}

else f5->Show();

}

private: System::Void menuStrip1\_ItemClicked(System::Object^ sender, System::Windows::Forms::ToolStripItemClickedEventArgs^ e) {

}

private: System::Void справкаToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) {

MessageBox::Show("- Для составления сметы необходимо выбрать кнопку на панеле меню 'Сметы'. На вспывающем окне выбрать кпопку 'Добавить', заполнить поля данными и нажать кнопку 'Оформить'. После чего таблица будет заполнена новой строкой. При двойном нажатии на строку в таблице будет загружено окно добавления в котором будут отображены ранее записаные данные. Для их изменения нужно выбрать кнопку 'Изменить.'"

+ "\n- Для просмотра существующих позиций выбора материала или работ требуется выбрать соответствующую кпоку 'Работы' или 'Материалы'."

+ "\n- Для настройки интерфейса рабочего места нужно выбрать кнопку 'Настройки' и выбрать нужный параметр для изменения."

+ "\n- Для выхода из приложения необходимо выбрать соответствующую кнопку 'Выход' после чего приложение будет закрыто."

, "Справочник");

}

};

}

MenuForm.cpp

#include "MenuForm.h"

#include <Windows.h>

using namespace CLRStroyBat;

[STAThread]

int WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int)

{

Application::EnableVisualStyles();

Application::SetCompatibleTextRenderingDefault(false);

Application::Run(gcnew MenuForm());

return 0;

}

PlitkaForm.h

private: System::Void PlitkaForm\_Load(System::Object^ sender, System::EventArgs^ e) {

this->Text = "Авторизация";// Задаёт название формы

if (textBox2->Text == "")

{

textBox1->MaxLength = 50;

textBox2->MaxLength = 50;

}

try {

SqlConnection^ Con = gcnew SqlConnection("Data Source=TIZKIZOY\\SQLEXPRESS;Integrated Security=True;Connect Timeout=30;Encrypt=True;TrustServerCertificate=True;ApplicationIntent=ReadWrite;MultiSubnetFailover=False");

Con->Open();

}

catch (SqlException^ e) {

MessageBox::Show("Ошибка при подключении к базе данных: " + e->Message);

Application::Exit();

}

}

private: System::Void button1\_Click(System::Object^ sender, System::EventArgs^ e) {

SqlConnection^ Con = gcnew SqlConnection("Data Source=TIZKIZOY\\SQLEXPRESS;Initial Catalog=Plitka;Integrated Security=True;Connect Timeout=30;Encrypt=True;TrustServerCertificate=True;ApplicationIntent=ReadWrite;MultiSubnetFailover=False");

Con->Open();

String^ login = textBox1->Text;

String^ password = textBox2->Text;

String^ querystring = "select \* from Registr where login = \'" + login + "\' and password = \'" + password + "\'";

SqlCommand^ command = gcnew SqlCommand(querystring, Con);

SqlDataAdapter^ adapter = gcnew SqlDataAdapter(command);

DataTable^ dtR = gcnew DataTable();

adapter->Fill(dtR);

if (dtR->Rows->Count == 1)

{

MessageBox::Show("Вы успешно зашли!", "Успешно!", MessageBoxButtons::OK, MessageBoxIcon::Information);

this->Close();

}

else MessageBox::Show("Такого аккаунта не существует!", "Ошибка!", MessageBoxButtons::OK, MessageBoxIcon::Warning);

Con->Close();

}

private: System::Void textBox2\_TextChanged(System::Object^ sender, System::EventArgs^ e) {

textBox2->PasswordChar = '\*';

}

private: System::Void button2\_Click(System::Object^ sender, System::EventArgs^ e) {

Application::Exit();

}

};

}

SmetaForm.h

Database^ db = gcnew Database();

public:void SumAndDisplayCosts() {

double total = 0.0;

// Суммирование стоимости из первого DataGridView

for each (DataGridViewRow ^ row in dataGridView1->Rows) {

if (row->Cells["Сумма, руб."]->Value != nullptr) {

total += Convert::ToDouble(row->Cells["Сумма, руб."]->Value);

}

}

// Отображение общей стоимости в Label

label5->Text = total.ToString();

}

public: void CountTable()

{

int count = 0;

for (int i = 0; i < dataGridView1->RowCount; i++)

{

count++;

}

label4->Text = count.ToString();

}

public: void ChengeDataTextBox(String^ data1, String^ data2, String^ data3, String^ data4, String^ data5, String^ data6, String^ data7, String^ data8, String^ data9, String^ data10, double total)

{

array<String^>^ row = gcnew array<String^> {

data1,//номер

data2,//дата

data3,//наименование

data4,//тел

data5,//начало работ

data6,//окончание план

data7,//окончание работ

data8,//Доп инфа

data9,//Комы

data10,//Рекомендации

};

total;

db->openConnection();

db->ChangeSaveSmetaTextBoxs(data1, data2, data3, data4, total, data5, data6, data7);

dataGridView1->DataSource = db->getTableSmeta(dataGridView1);

db->closeConnection();

CountTable();

}

public: void GetDataTextBox(String^ data1, String^ data2, String^ data3, String^ data4, String^ data5, String^ data6, String^ data7, String^ data8, String^ data9, String^ data10, double total)

{

array<String^>^ row = gcnew array<String^> {

data1,//номер

data2,//дата

data3,//наименование

data4,//тел

data5,//начало работ

data6,//окончание план

data7,//окончание работ

data8,//Доп инфа

data9,//Комы

data10,//Рекомендации

};

total;

db->openConnection();

db->InsertIntoSaveSmetaTextBoxs(data1, data2, data3, data4, total, data5, data6, data7);

dataGridView1->DataSource = db->getTableSmeta(dataGridView1);

db->closeConnection();

CountTable();

SumAndDisplayCosts();

}

private: System::Void SmetaForm\_Load(System::Object^ sender, System::EventArgs^ e) {

this->Text = "Сметы";// Задаёт название формы

SmetaForm::Width = 1010; //Установка пользовательского размера формы

SmetaForm::Height = 490;

dataGridView1->RowHeadersWidth = 5; // уменьшение крайнего левого столбца по ширине

dataGridView1->Rows->Clear();

db->openConnection();

dataGridView1->DataSource = db->getTableSmeta(dataGridView1);

db->closeConnection();

CountTable();

SumAndDisplayCosts();

}

private: System::Void button5\_Click(System::Object^ sender, System::EventArgs^ e) {

Close(); //закрывает форму

}

private: System::Void button1\_Click(System::Object^ sender, System::EventArgs^ e) {

array<String^>^ dataNum = gcnew array<String^>(dataGridView1->RowCount);

for (int i = 0; i < dataGridView1->RowCount; i++)

{

if (dataGridView1->Rows[i]->Cells[0]->Value != nullptr)

{

dataNum[i] = dataGridView1->Rows[i]->Cells[0]->Value->ToString();

}

}

array<String^>^ data = gcnew array<String^>(dataGridView1->RowCount);

for (int i = 0; i < dataGridView1->RowCount; i++)

{

if (dataGridView1->Rows[i]->Cells[0]->Value != nullptr)

{

data[i] = dataGridView1->Rows[i]->Cells[0]->Value->ToString();

}

}

ADDForm^ f1 = gcnew ADDForm(gcnew returnTextBoxData(this, &SmetaForm::GetDataTextBox), data, dataNum);

f1->ShowDialog();

}

private: System::Void button3\_Click(System::Object^ sender, System::EventArgs^ e) {

auto result = MessageBox::Show(

"Вы действительно хотите удалить выбранный фрагмент?",

"Подтвердите удаление",

MessageBoxButtons::YesNo,

MessageBoxIcon::Question);

if (result == System::Windows::Forms::DialogResult::Yes) {

dataGridView1->Rows->RemoveAt(dataGridView1->CurrentRow->Index);

dataGridView1->Refresh();

}

}

private: System::Void button6\_Click(System::Object^ sender, System::EventArgs^ e) {

AddMaterialForm^ f5 = gcnew AddMaterialForm();

f5->ShowDialog();

}

private: System::Void dataGridView1\_CellMouseDoubleClick(System::Object^ sender, System::Windows::Forms::DataGridViewCellMouseEventArgs^ e) {

array<String^>^ dataNum = gcnew array<String^>(dataGridView1->RowCount);

for (int i = 0; i < dataGridView1->RowCount; i++)

{

if (dataGridView1->Rows[i]->Cells[0]->Value != nullptr)

{

dataNum[i] = dataGridView1->Rows[i]->Cells[0]->Value->ToString();

int a = 5;

}

}

array<String^>^ data = gcnew array<String^>(dataGridView1->RowCount);

for (int i = 0; i < dataGridView1->RowCount; i++)

{

if (dataGridView1->Rows[i]->Cells[0]->Value != nullptr)

{

data[i] = dataGridView1->Rows[i]->Cells[0]->Value->ToString();

int a = 5;

}

}

if (e->RowIndex >= 0)

{

String^ id = Convert::ToString(dataGridView1->Rows[e->RowIndex]->Cells["№"]->Value);

ADDForm^ f2 = gcnew ADDForm(gcnew returnTextBoxData(this, &SmetaForm::ChengeDataTextBox), id, data, dataNum);

f2->ShowDialog();

}

else

{

MessageBox::Show("Повторите попытку.");

}

}

private: System::Void button2\_Click(System::Object^ sender, System::EventArgs^ e) {

auto result = MessageBox::Show(

"Вы действительно хотите выполнить очистку таблиц?",

"Подтвердите удаление",

MessageBoxButtons::YesNo,

MessageBoxIcon::Question);

if (result == System::Windows::Forms::DialogResult::Yes) {

db->openConnection();

db->clearEverythingTable();

db->getTableSmeta(dataGridView1);

db->closeConnection();

MessageBox::Show("Очистка выполнена.\nПерезапустите данное окно.");

}

}

};

}

ADDForm.h

#pragma endregion

int countW = 0;

int countM = 0;

Database^ db = gcnew Database();

private: void СonfirmationOfTheDeleteDataGrid1()

{

auto result = MessageBox::Show(

"Вы действительно хотите удалить выбранный фрагмент?",

"Подтвердите удаление",

MessageBoxButtons::YesNo,

MessageBoxIcon::Question);

if (result == System::Windows::Forms::DialogResult::Yes) {

dataGridView1->Rows->RemoveAt(dataGridView1->CurrentRow->Index);

dataGridView1->Refresh();

SumAndDisplayCosts();

}

}

private: void СonfirmationOfTheDeleteDataGrid3()

{

auto result = MessageBox::Show(

"Вы действительно хотите удалить выбранный фрагмент?",

"Подтвердите удаление",

MessageBoxButtons::YesNo,

MessageBoxIcon::Question);

if (result == System::Windows::Forms::DialogResult::Yes) {

dataGridView3->Rows->RemoveAt(dataGridView3->CurrentRow->Index);

dataGridView3->Refresh();

SumAndDisplayCosts();

}

}

private: void ShowCMS(System::Windows::Forms::MouseEventArgs^ e) {

if (e->Button == System::Windows::Forms::MouseButtons::Right) {

contextMenuStrip1->Show(

this->MousePosition);

}

}

public:void SumAndDisplayCosts() {

double total = 0.0;

int columnIndexPrice = 2; // Индекс столбца 'цена'

int columnIndexQuantity = 7; // Индекс столбца 'количество'

// Проходим по всем строкам DataGridView

for each (DataGridViewRow ^ row in dataGridView1->Rows)

{

// Извлекаем значения из ячеек

double price = Convert::ToDouble(row->Cells[columnIndexPrice]->Value);

double quantity = Convert::ToDouble(row->Cells[columnIndexQuantity]->Value);

// Перемножаем и добавляем к общей сумме

total += price \* quantity;

}

int columnIndexPrice2 = 2; // Индекс столбца 'цена'

int columnIndexQuantity2 = 7; // Индекс столбца 'количество'

// Проходим по всем строкам DataGridView

for each (DataGridViewRow ^ row in dataGridView3->Rows)

{

// Извлекаем значения из ячеек

double price = Convert::ToDouble(row->Cells[columnIndexPrice2]->Value);

double quantity = Convert::ToDouble(row->Cells[columnIndexQuantity2]->Value);

// Перемножаем и добавляем к общей сумме

total += price \* quantity;

}

// Отображение общей стоимости в Label

label9->Text = total.ToString();

}

// Метод для загрузки данных из базы данных и заполнения формы

public: void DataFromDatabaseTextBoxs() {

SqlConnection^ Con = gcnew SqlConnection("Data Source=TIZKIZOY\\SQLEXPRESS;Initial Catalog=Plitka;Integrated Security=True;Connect Timeout=30;Encrypt=True;TrustServerCertificate=True;ApplicationIntent=ReadWrite;MultiSubnetFailover=False");

try {

Con->Open();

String^ query = "SELECT \* FROM SaveTextBoxs WHERE [idusd] = @dataId";

SqlCommand^ command = gcnew SqlCommand(query, Con);

command->Parameters->AddWithValue("@dataId", dataId);

SqlDataReader^ reader = command->ExecuteReader();

while (reader->Read()) {

textBox9->Text = reader["data1"]->ToString();

textBox10->Text = reader["data2"]->ToString();

textBox1->Text = reader["data3"]->ToString();

textBox2->Text = reader["data4"]->ToString();

textBox3->Text = reader["data5"]->ToString();

textBox4->Text = reader["data6"]->ToString();

textBox5->Text = reader["data7"]->ToString();

textBox6->Text = reader["data8"]->ToString();

textBox7->Text = reader["data9"]->ToString();

textBox8->Text = reader["data10"]->ToString();

}

}

catch (SqlException^ ex) { // Обрабатывайте конкретные исключения

MessageBox::Show("Ошибка базы данных: " + ex->Message);

}

catch (Exception^ ex) {

MessageBox::Show("Общая ошибка: " + ex->Message);

}

finally {

Con->Close();

}

}

public: void DataFromDatabaseWorks()

{

SqlConnection^ Con = gcnew SqlConnection("Data Source=TIZKIZOY\\SQLEXPRESS;Initial Catalog=Plitka;Integrated Security=True;Connect Timeout=30;Encrypt=True;TrustServerCertificate=True;ApplicationIntent=ReadWrite;MultiSubnetFailover=False");

try {

Con->Open();

// SQL запрос для получения данных

String^ query = "SELECT \* FROM SaveTableWork WHERE idusdt = @dataId";

SqlCommand^ command = gcnew SqlCommand(query, Con);

command->Parameters->AddWithValue("@dataId", dataId);

// Выполнение запроса и получение данных

SqlDataReader^ reader = command->ExecuteReader();

// Очистка существующих данных в DataGridView

dataGridView1->Rows->Clear();

// Чтение данных из SqlDataReader и заполнение DataGridView

while (reader->Read()) {

array<Object^>^ row = gcnew array<Object^> {

reader["Наименование"]->ToString(),

reader["Артикул"]->ToString(),

reader["Цена руб."]->ToString(),

reader["Тип"]->ToString(),

reader["Ед.изм."]->ToString(),

reader["Страна"]->ToString(),

reader["Цена $"]->ToString(),

reader["Количество"]->ToString(),

};

dataGridView1->Rows->Add(row);

}

reader->Close();

}

catch (Exception^ ex) {

MessageBox::Show(ex->Message);

}

finally {

// Закрытие соединения с базой данных, если оно было открыто

if (Con->State == ConnectionState::Open) {

Con->Close();

}

}

}

public: void DataFromDatabaseMaterials()

{

SqlConnection^ Con = gcnew SqlConnection("Data Source=TIZKIZOY\\SQLEXPRESS;Initial Catalog=Plitka;Integrated Security=True;Connect Timeout=30;Encrypt=True;TrustServerCertificate=True;ApplicationIntent=ReadWrite;MultiSubnetFailover=False");

try {

Con->Open();

// SQL запрос для получения данных

String^ query = "SELECT \* FROM SaveTableMaterial WHERE idusd = @dataId";

SqlCommand^ command = gcnew SqlCommand(query, Con);

command->Parameters->AddWithValue("@dataId", dataId);

// Выполнение запроса и получение данных

SqlDataReader^ reader = command->ExecuteReader();

// Очистка существующих данных в DataGridView

dataGridView3->Rows->Clear();

// Чтение данных из SqlDataReader и заполнение DataGridView

while (reader->Read()) {

array<Object^>^ row = gcnew array<Object^> {

reader["Наименование"]->ToString(),

reader["Артикул"]->ToString(),

reader["Цена руб."]->ToString(),

reader["Ед.изм."]->ToString(),

reader["Страна"]->ToString(),

reader["Цена $"]->ToString(),

reader["Бренд"]->ToString(),

reader["Количество"]->ToString(),

};

dataGridView3->Rows->Add(row);

}

reader->Close();

}

catch (Exception^ ex) {

MessageBox::Show(ex->Message);

}

finally {

// Закрытие соединения с базой данных, если оно было открыто

if (Con->State == ConnectionState::Open) {

Con->Close();

}

}

}

public: void GetDataWork(DataGridViewRow^ returnedRow, String^ dataNum)

{

bool FoundRow = true;

array<String^>^ columnData = gcnew array<String^>(dataGridView1->Rows->Count - 1);

String^ str = returnedRow->Cells[1]->Value->ToString();

// Извлекаем данные из столбца

for (int i = 0; i < dataGridView1->Rows->Count; ++i) {

if (dataGridView1->Rows[i]->IsNewRow) continue; // Пропускаем строку для нового ввода

columnData[i] = dataGridView1->Rows[i]->Cells[1]->Value->ToString();

if (columnData[i]->Trim() == str->Trim())

{

FoundRow = false;

}

}

if (FoundRow)

{

dataGridView1->Rows->Add(returnedRow->Cells[0]->Value, returnedRow->Cells[1]->Value, returnedRow->Cells[2]->Value, returnedRow->Cells[3]->Value,

returnedRow->Cells[4]->Value, returnedRow->Cells[5]->Value, returnedRow->Cells[6]->Value, dataNum);

}

else

{

MessageBox::Show("Эта позиция уже добавлена");

}

SumAndDisplayCosts();

}

public: void GetDataMaterial(DataGridViewRow^ returnedRow, String^ dataNum)

{

bool FoundRow = true;

array<String^>^ columnData = gcnew array<String^>(dataGridView3->Rows->Count - 1);

String^ str = returnedRow->Cells[1]->Value->ToString();

// Извлекаем данные из столбца

for (int i = 0; i < dataGridView3->Rows->Count; ++i) {

if (dataGridView3->Rows[i]->IsNewRow) continue; // Пропускаем строку для нового ввода

columnData[i] = dataGridView3->Rows[i]->Cells[1]->Value->ToString();

if (columnData[i]->Trim() == str->Trim())

{

FoundRow = false;

}

}

if (FoundRow)

{

dataGridView3->Rows->Add(returnedRow->Cells[0]->Value, returnedRow->Cells[1]->Value, returnedRow->Cells[2]->Value, returnedRow->Cells[3]->Value,

returnedRow->Cells[4]->Value, returnedRow->Cells[5]->Value, returnedRow->Cells[6]->Value, dataNum);

}

else

{

MessageBox::Show("Эта позиция уже добавлена");

}

SumAndDisplayCosts();

}

private: System::Void ADDForm\_Load(System::Object^ sender, System::EventArgs^ e) {

this->Text = "Новый документ";

ADDForm::Width = 1000; //Установка пользовательского размера формы

ADDForm::Height = 530;

SumAndDisplayCosts();

dataGridView3->AutoResizeColumns();//автоподбор ширины

dataGridView3->RowHeadersWidth = 5; // уменьшение крайнего левого столбца по ширине

dataGridView3->AutoResizeColumnHeadersHeight();

dataGridView1->AutoResizeColumns();//автоподбор ширины

dataGridView1->RowHeadersWidth = 5; // уменьшение крайнего левого столбца по ширине

textBox1->MaxLength = 30;

textBox2->MaxLength = 20;

textBox3->MaxLength = 15;

textBox4->MaxLength = 15;

textBox5->MaxLength = 15;

textBox6->MaxLength = 200;

textBox7->MaxLength = 200;

textBox8->MaxLength = 500;

textBox9->MaxLength = 15;

textBox10->MaxLength = 15;

}

private: System::Void button5\_Click(System::Object^ sender, System::EventArgs^ e) {

this->Close();

}

private: System::Void button2\_Click(System::Object^ sender, System::EventArgs^ e) {

AddWorkForm^ f2 = gcnew AddWorkForm(gcnew returnMaterialData(this, &ADDForm::GetDataWork));

f2->ShowDialog();

}

private: System::Void button3\_Click(System::Object^ sender, System::EventArgs^ e) {

AddMaterialForm^ f = gcnew AddMaterialForm(gcnew returnMaterialData(this, &ADDForm::GetDataMaterial));

f->ShowDialog();

}

private: System::Void button4\_Click(System::Object^ sender, System::EventArgs^ e) {

if (!String::IsNullOrWhiteSpace(textBox9->Text))

{

String^ textBoxValue = textBox9->Text;

bool matchFound = true;

for (int i = 0; i < dataNumId->Length; i++) {

if (dataNumId[i]->Trim() == textBoxValue->Trim()) {

matchFound = false;

break;

}

}

if (matchFound)

{

try {

db->openConnection();

db->InsertIntoSaveWork(dataGridView1, textBox9);

db->InsertIntoSaveMaterials(dataGridView3, textBox9);

String^ data1 = textBox9->Text;//Номер сметы

String^ data2 = textBox10->Text;//Дата

String^ data3 = textBox1->Text;//наименование

String^ data4 = textBox2->Text;//тел

String^ data5 = textBox3->Text;//начало работ

String^ data6 = textBox4->Text;//окончание план

String^ data7 = textBox5->Text;//окончание работ

String^ data8 = textBox6->Text;//Доп инфа

String^ data9 = textBox7->Text;//Комм

String^ data10 = textBox8->Text;//Рекомендации

double total1 = Convert::ToDouble(label9->Text);

db->InsertIntoSaveTextBox(data1, data2, data3, data4, data5, data6, data7, data8, data9, data10, textBox9);

deleg(data1, data2, data3, data4, data5, data6, data7, data8, data9, data10, total1);

db->closeConnection();

MessageBox::Show("Успешно добавлен");

this->Hide();

}

catch (Exception^ ex) {

MessageBox::Show(ex->Message);

}

}

else

{

MessageBox::Show("Вы уже добавили смету с таким номером.\n Если хотите добавить изменения выберите кпоку изменить.");

}

}

else

{

MessageBox::Show("Введите номер сметы чтобы оформить её.");

}

}

public: System::Void textBox9\_KeyPress(System::Object^ sender, System::Windows::Forms::KeyPressEventArgs^ e) {

// Проверяем нажатую клавишу

Char keyPressed = e->KeyChar;

String^ textBoxValue = textBox9->Text;

// Проверяем, является ли символ числом или управляющим символом (например, Backspace)

if (!Char::IsDigit(keyPressed) && keyPressed != 0x08)

{

e->Handled = true; // Отменяем ввод символа

MessageBox::Show("Введите целое число.");

}

else

{

// Строим предполагаемое число с учётом нового символа

String^ expectedNum = textBoxValue + Convert::ToString(keyPressed);

// Проверяем, содержится ли уже введенное число в массиве строк

bool numExists = false;

for each (String ^ str in dataNumId)

{//Trim() - убирает лишние пробелы для корректного сравнивания!

if (str->Trim() == expectedNum->Trim())

{

numExists = true;

break;

}

}

if (numExists)

{

e->Handled = true; // Отменяем ввод символа

MessageBox::Show("Это число уже используется.");

}

}

}

private: System::Void ADDForm\_MouseUp(System::Object^ sender, System::Windows::Forms::MouseEventArgs^ e) {

this->ShowCMS(e);

}

private: System::Void dataGridView1\_CellMouseUp(System::Object^ sender, System::Windows::Forms::DataGridViewCellMouseEventArgs^ e) {

if (e->Button == System::Windows::Forms::MouseButtons::Right)

{

this->ShowCMS(e);

}

}

private: System::Void dataGridView3\_CellMouseUp(System::Object^ sender, System::Windows::Forms::DataGridViewCellMouseEventArgs^ e) {

if (e->Button == System::Windows::Forms::MouseButtons::Right)

{

this->ShowCMS(e);

}

}

private: System::Void button7\_Click(System::Object^ sender, System::EventArgs^ e) {

double sum = 0.0;

int columnIndexPrice = 2; // Индекс столбца 'цена'

int columnIndexQuantity = 7; // Индекс столбца 'количество'

// Проходим по всем строкам DataGridView

for each (DataGridViewRow ^ row in dataGridView1->Rows)

{

// Извлекаем значения из ячеек

double price = Convert::ToDouble(row->Cells[columnIndexPrice]->Value);

double quantity = Convert::ToDouble(row->Cells[columnIndexQuantity]->Value);

// Перемножаем и добавляем к общей сумме

sum += price \* quantity;

}

double sum1 = 0.0;

int columnIndexPrice2 = 2; // Индекс столбца 'цена'

int columnIndexQuantity2 = 7; // Индекс столбца 'количество'

// Проходим по всем строкам DataGridView

for each (DataGridViewRow ^ row in dataGridView3->Rows)

{

// Извлекаем значения из ячеек

double price = Convert::ToDouble(row->Cells[columnIndexPrice2]->Value);

double quantity = Convert::ToDouble(row->Cells[columnIndexQuantity2]->Value);

// Перемножаем и добавляем к общей сумме

sum1 += price \* quantity;

}

try

{

array<String^>^ product = {

"№", "Наименование", "Артикул", "Цена руб." ,"Тип","Ед.изм.","Страна","Цена $,Количество"

};

try

{

auto word3 = gcnew Microsoft::Office::Interop::Word::Application();//всегда будет подсвечивать ошибку но компилирует

word3->Visible = true;//отображать прогу ворд

auto t = Type::Missing;// t иницилизируется пустым

auto document2 = word3->Documents->Add(t, t, t, t); //создаём пустую страницу ворда(документ)

word3->Selection->TypeText("\t\t\t\t\tСМЕТА");//в текущую позицию нахождения курсора записываем текст и курсор смещается каждый раз за последний введенный символ

word3->Selection->TypeText("\nТаблица №1. Расчёт работ:");

Object^ t1 = Microsoft::Office::Interop::Word::WdDefaultTableBehavior::wdWord9TableBehavior;//в параметрах документа ворд указываем настройку отображать границы ячеек таблицы

Object^ t2 = Microsoft::Office::Interop::Word::WdAutoFitBehavior::wdAutoFitContent; // в параметрах документа Ворд указываем настройки автоматически изменять размер ячеек таблицы под

word3->ActiveDocument->Tables->Add(word3->Selection->Range, dataGridView1->RowCount + 1, dataGridView1->ColumnCount + 1, t1, t2);//записываемое в них содеримое и визуальным показом всех границ ячеек таблицы

try

{

word3->ActiveDocument->Tables[1]->Cell(1, 1)->Range->Text = "N";

word3->ActiveDocument->Tables[1]->Cell(1, 2)->Range->Text = "Наименование";

word3->ActiveDocument->Tables[1]->Cell(1, 3)->Range->Text = "Артикул";

word3->ActiveDocument->Tables[1]->Cell(1, 4)->Range->Text = "Цена руб.";

word3->ActiveDocument->Tables[1]->Cell(1, 5)->Range->Text = "Тип";

word3->ActiveDocument->Tables[1]->Cell(1, 6)->Range->Text = "Ед.изм.";

word3->ActiveDocument->Tables[1]->Cell(1, 7)->Range->Text = "Страна";

word3->ActiveDocument->Tables[1]->Cell(1, 8)->Range->Text = "Цена $";

word3->ActiveDocument->Tables[1]->Cell(1, 9)->Range->Text = "Количество";

int a = 1;

for (int i = 0; i < dataGridView1->RowCount - 1; i++)

{

word3->ActiveDocument->Tables[1]->Cell(i + 2, 1)->Range->Text = a++.ToString();

for (int j = 0; j < dataGridView1->ColumnCount; j++)

{

word3->ActiveDocument->Tables[1]->Cell(i + 2, j + 2)->Range->Text = dataGridView1->Rows[i]->Cells[j]->Value->ToString();

}

}

int lastRow = word3->ActiveDocument->Tables[1]->Rows->Count;

int lastColumn = word3->ActiveDocument->Tables[1]->Columns->Count;

word3->ActiveDocument->Tables[1]->Cell(lastRow, 3)->Range->Text = "ИТОГО:";

word3->ActiveDocument->Tables[1]->Cell(lastRow, 3 + 1)->Range->Text = sum.ToString();

}

catch (System::Runtime::InteropServices::COMException^ eva)

{

MessageBox::Show(eva->Message);

}

Object^ t3 = Microsoft::Office::Interop::Word::WdUnits::wdLine; ////назначаем единицей измерения текста втекущем документе ОднУ строку

Int32^ str10 = 50; // динамическая целочисленная переменная декларируется и инициализируется 10 - ю, чтобы указывать на 10 - ю строку таблицы

word3->Selection->MoveDown(t3, str10, t);///переводим текущую позицию курсоря Selection за пределы табл в 10-ю строку, куда введем текст

word3->Selection->TypeText("Таблица №2. Расчёт материалов:\n");

word3->ActiveDocument->Tables->Add(word3->Selection->Range, dataGridView3->RowCount + 1, dataGridView3->ColumnCount + 1, t1, t2);//записываемое в них содеримое и визуальным показом всех границ ячеек таблицы

try

{

word3->ActiveDocument->Tables[2]->Cell(1, 1)->Range->Text = "N";

word3->ActiveDocument->Tables[2]->Cell(1, 2)->Range->Text = "Наименование";

word3->ActiveDocument->Tables[2]->Cell(1, 3)->Range->Text = "Артикул";

word3->ActiveDocument->Tables[2]->Cell(1, 4)->Range->Text = "Цена руб.";

word3->ActiveDocument->Tables[2]->Cell(1, 5)->Range->Text = "Ед.изм.";

word3->ActiveDocument->Tables[2]->Cell(1, 6)->Range->Text = "Страна";

word3->ActiveDocument->Tables[2]->Cell(1, 7)->Range->Text = "Цена $";

word3->ActiveDocument->Tables[2]->Cell(1, 8)->Range->Text = "Бренд";

word3->ActiveDocument->Tables[2]->Cell(1, 9)->Range->Text = "Количество";

int b = 1;

for (int i = 0; i < dataGridView1->RowCount - 1; i++)

{

word3->ActiveDocument->Tables[2]->Cell(i + 2, 1)->Range->Text = b++.ToString();

for (int j = 0; j < dataGridView1->ColumnCount; j++)

{

word3->ActiveDocument->Tables[2]->Cell(i + 2, j + 2)->Range->Text = dataGridView3->Rows[i]->Cells[j]->Value->ToString();

}

}

int lastRow2 = word3->ActiveDocument->Tables[2]->Rows->Count;

int lastColumn2 = word3->ActiveDocument->Tables[2]->Columns->Count;

word3->ActiveDocument->Tables[2]->Cell(lastRow2, 3)->Range->Text = "ИТОГО:";

word3->ActiveDocument->Tables[2]->Cell(lastColumn2, 3 + 1)->Range->Text = sum1.ToString();

}

catch (System::Runtime::InteropServices::COMException^ eva)

{

MessageBox::Show(eva->Message);

}

int overSum = sum + sum1;

Int32^ str100 = 100; // динамическая целочисленная переменная декларируется и инициализируется 10 - ю, чтобы указывать на 10 - ю строку таблицы

word3->Selection->MoveDown(t3, str100, t);///переводим текущую позицию курсоря Selection за пределы табл в 10-ю строку, куда введем текст

word3->Selection->TypeText("\t\t\t\t\tВСЕГО: " + overSum.ToString() + " руб.");

Object^ filename = "C:\\Users\\Захар\\source\\repos\\Laba27\\Шабуневич Захар Т-319.docx";///текучему у дркументу (написал полный путь файлу и указал имя расширение

word3->ActiveDocument->SaveAs(filename, t, t, t, t, t, t, t, t, t, t, t, t, t, t, t);//Если пользователь сохранит то он сохранит архивированных файла по нашему пути

}

catch (System::Exception^ eva)

{

MessageBox::Show(eva->Message);

}

}

catch (Exception^ eva)

{

MessageBox::Show(eva->Message);

}

}

private: System::Void button1\_Click(System::Object^ sender, System::EventArgs^ e) {

String^ textBoxValue1 = textBox9->Text;

bool matchFound1 = true;

for (int i = 0; i < dataNumId->Length; i++) {

if (dataNumId[i]->Trim() == textBoxValue1->Trim()) {

matchFound1 = false;

break;

}

}

if (!matchFound1)

{

try

{

db->openConnection();

db->RepeatChecFromSaveTableWork(dataGridView1, textBox9);

db->RepeatChecFromSaveTableMaterial(dataGridView3, textBox9);

String^ data1 = textBox9->Text;//Номер сметы

String^ data2 = textBox10->Text;//Дата

String^ data3 = textBox1->Text;//наименование

String^ data4 = textBox2->Text;//тел

String^ data5 = textBox3->Text;//начало работ

String^ data6 = textBox4->Text;//окончание план

String^ data7 = textBox5->Text;//окончание работ

String^ data8 = textBox6->Text;//Доп инфа

String^ data9 = textBox7->Text;//Комм

String^ data10 = textBox8->Text;//Рекомендации

double total1 = Convert::ToDouble(label9->Text);

db->ChangeSaveTextBox(data1, data2, data3, data4, data5, data6, data7, data8, data9, data10, textBox9);

deleg(data1, data2, data3, data4, data5, data6, data7, data8, data9, data10, total1);

db->closeConnection();

MessageBox::Show("Успешно изменён");

this->Hide();

}

catch (Exception^ eva)

{

MessageBox::Show(eva->Message);

}

}

else

{

MessageBox::Show("Вы не оформляли смету с таким номер.\n Чтобы внести изменения нужно сначала оформить смету.");

}

}

private: System::Void button6\_Click(System::Object^ sender, System::EventArgs^ e) {

double length = Convert::ToDouble(textBox11->Text);

double height = Convert::ToDouble(textBox12->Text);

double plitka = Convert::ToDouble(textBox15->Text);

double area = length \* height;

double result = area / plitka;

result = result \* 10;

textBox13->Text = result.ToString();

textBox14->Text = area.ToString();

int centerX = pictureBox1->Width / 2;

int centerY = pictureBox1->Height / 2;

Pen^ myPen = gcnew Pen(Color::Blue, 2);

Graphics^ myGraphics = pictureBox1->CreateGraphics();

myGraphics->DrawRectangle(myPen,

centerX, centerY,

length \* 10 , height \* 10);

}

};

}

MaterialsForm.cpp

#pragma endregion

Database^ db = gcnew Database();

DataTable^ dtR;

int count = 0;

private: void СonfirmationOfTheExit()

{

auto result = MessageBox::Show(

"Вы действительно хотите выйти из программы?",

"Подтвердите выход",

MessageBoxButtons::YesNo,

MessageBoxIcon::Question);

if (result == System::Windows::Forms::DialogResult::Yes) {

this->Close();

}

}

private: void СonfirmationOfTheDelete()

{

auto result = MessageBox::Show(

"Вы действительно хотите удалить выбранный фрагмент?",

"Подтвердите удаление",

MessageBoxButtons::YesNo,

MessageBoxIcon::Question);

if (result == System::Windows::Forms::DialogResult::Yes) {

dataGridView2->Rows->RemoveAt(dataGridView2->CurrentRow->Index);

dataGridView2->Refresh();

count--;

label4->Text = count.ToString();

}

}

private: void ShowCMS(System::Windows::Forms::MouseEventArgs^ e) {

if (e->Button == System::Windows::Forms::MouseButtons::Right) {

contextMenuStrip1->Show(

this->MousePosition);

}

}

private: System::Void MaterialsForm\_Load(System::Object^ sender, System::EventArgs^ e) {

this->Text = "Справочник Материалов";

MaterialsForm::Width = 990; //Установка пользовательского размера формы

MaterialsForm::Height = 545;

dataGridView2->AutoResizeColumns();//автоподбор ширины

dataGridView2->RowHeadersWidth = 5; // уменьшение крайнего левого столбца по ширине

db->openConnection();

dataGridView2->DataSource = db->getTableMaterial();

db->closeConnection();

for (int i = 0; i < dataGridView2->Rows->Count - 6; i++)

{

count++;

}

label4->Text = count.ToString();

}

private: System::Void button5\_Click(System::Object^ sender, System::EventArgs^ e) {

СonfirmationOfTheExit();

}

private: System::Void treeView1\_AfterSelect(System::Object^ sender, System::Windows::Forms::TreeViewEventArgs^ e) {

int id = Convert::ToInt32(treeView1->SelectedNode->Index);

switch (id)

{

case 0:

{

dataGridView2->ClearSelection();

dataGridView2->Rows[0]->Selected = true;

break;

}

case 1:

{

dataGridView2->ClearSelection();

dataGridView2->Rows[9]->Selected = true;

break;

}

case 2:

{

dataGridView2->ClearSelection();

dataGridView2->Rows[23]->Selected = true;

break;

}

case 3:

{

dataGridView2->ClearSelection();

dataGridView2->Rows[30]->Selected = true;

break;

}

case 4:

{

dataGridView2->ClearSelection();

dataGridView2->Rows[33]->Selected = true;

break;

}

case 5:

{

dataGridView2->ClearSelection();

dataGridView2->Rows[36]->Selected = true;

break;

}

default:

{

MessageBox::Show("Ты где-то ошибся в коде");

}

}

}

private: System::Void button5\_Click\_1(System::Object^ sender, System::EventArgs^ e) {

СonfirmationOfTheExit();

}

private: System::Void button6\_Click(System::Object^ sender, System::EventArgs^ e) {

for (int i = 0; i < dataGridView2->RowCount; i++)

{

dataGridView2->Rows[i]->Selected = false;

for (int j = 0; j < dataGridView2->ColumnCount; j++)

if (dataGridView2->Rows[i]->Cells[j]->Value != nullptr)

if (dataGridView2->Rows[i]->Cells[j]->Value->ToString()->Contains(textBox1->Text))

{

dataGridView2->Rows[i]->Selected = true;

break;

}

}

}

private: System::Void button3\_Click(System::Object^ sender, System::EventArgs^ e) {

СonfirmationOfTheDelete();

}

private: System::Void MaterialsForm\_MouseUp(System::Object^ sender, System::Windows::Forms::MouseEventArgs^ e) {

this->ShowCMS(e);

}

private: System::Void удалитьToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) {

СonfirmationOfTheDelete();

}

private: System::Void dataGridView2\_CellMouseUp(System::Object^ sender, System::Windows::Forms::DataGridViewCellMouseEventArgs^ e) {

if (e->Button == System::Windows::Forms::MouseButtons::Right)

{

this->ShowCMS(e);

}

}

private: System::Void comboBox2\_SelectedIndexChanged\_1(System::Object^ sender, System::EventArgs^ e) {

String^ selectedValue = this->comboBox2->SelectedItem->ToString();

if (selectedValue == "Закупка")

{

db->openConnection();

dataGridView2->DataSource = db->getTableMaterialZakup();

db->closeConnection();

}

else if (selectedValue == "Стандарт")

{

db->openConnection();

dataGridView2->DataSource = db->getTableMaterial();

db->closeConnection();

}

}

};

}

WorkForm.h

#pragma endregion

DataTable^ dtR;

int count = 0;

Database^ db = gcnew Database();

private: void СonfirmationOfTheExit()

{

auto result = MessageBox::Show(

"Вы действительно хотите выйти из программы?",

"Подтвердите выход",

MessageBoxButtons::YesNo,

MessageBoxIcon::Question);

if (result == System::Windows::Forms::DialogResult::Yes) {

this->Close();

}

}

private: void СonfirmationOfTheDelete()

{

auto result = MessageBox::Show(

"Вы действительно хотите удалить выбранный фрагмент?",

"Подтвердите удаление",

MessageBoxButtons::YesNo,

MessageBoxIcon::Question);

if (result == System::Windows::Forms::DialogResult::Yes) {

dataGridView2->Rows->RemoveAt(dataGridView2->CurrentRow->Index);

dataGridView2->Refresh();

count--;

label4->Text = count.ToString();

}

}

private: void ShowCMS(System::Windows::Forms::MouseEventArgs^ e) {

if (e->Button == System::Windows::Forms::MouseButtons::Right) {

contextMenuStrip1->Show(

this->MousePosition);

}

}

private: System::Void WorkForm\_Load(System::Object^ sender, System::EventArgs^ e) {

this->Text = "Справочник работ";

WorkForm::Width = 1090; //Установка пользовательского размера формы

WorkForm::Height = 490;

dataGridView2->AutoResizeColumns();//автоподбор ширины

dataGridView2->RowHeadersWidth = 5; // уменьшение крайнего левого столбца по ширине

db->openConnection();

dataGridView2->DataSource = db->getTableWork();

db->closeConnection();

for (int i = 0; i < dataGridView2->Rows->Count - 9; i++)

{

count++;

}

label4->Text = count.ToString();

}

private: System::Void button5\_Click(System::Object^ sender, System::EventArgs^ e) {

СonfirmationOfTheExit();

}

private: System::Void treeView1\_AfterSelect(System::Object^ sender, System::Windows::Forms::TreeViewEventArgs^ e) {

int id = Convert::ToInt32(treeView1->SelectedNode->Index);

switch (id)

{

case 0:

{

dataGridView2->ClearSelection();

dataGridView2->Rows[0]->Selected = true;

break;

}

case 1:

{

dataGridView2->ClearSelection();

dataGridView2->Rows[12]->Selected = true;

break;

}

case 2:

{

dataGridView2->ClearSelection();

dataGridView2->Rows[24]->Selected = true;

break;

}

case 3:

{

dataGridView2->ClearSelection();

dataGridView2->Rows[43]->Selected = true;

break;

}

case 4:

{

dataGridView2->ClearSelection();

dataGridView2->Rows[57]->Selected = true;

break;

}

case 5:

{

dataGridView2->ClearSelection();

dataGridView2->Rows[63]->Selected = true;

break;

}

case 6:

{

dataGridView2->ClearSelection();

dataGridView2->Rows[70]->Selected = true;

break;

}

case 7:

{

dataGridView2->ClearSelection();

dataGridView2->Rows[74]->Selected = true;

break;

}

case 8:

{

dataGridView2->ClearSelection();

dataGridView2->Rows[78]->Selected = true;

break;

}

default:

{

MessageBox::Show("Ты где-то ошибся в коде");

}

}

}

private: System::Void button6\_Click(System::Object^ sender, System::EventArgs^ e) {

for (int i = 0; i < dataGridView2->RowCount; i++)

{

dataGridView2->Rows[i]->Selected = false;

for (int j = 0; j < dataGridView2->ColumnCount; j++)

if (dataGridView2->Rows[i]->Cells[j]->Value != nullptr)

if (dataGridView2->Rows[i]->Cells[j]->Value->ToString()->Contains(textBox1->Text))

{

dataGridView2->Rows[i]->Selected = true;

break;

}

}

}

private: System::Void button3\_Click(System::Object^ sender, System::EventArgs^ e) {

СonfirmationOfTheDelete();

}

private: System::Void WorkForm\_MouseUp(System::Object^ sender, System::Windows::Forms::MouseEventArgs^ e) {

this->ShowCMS(e);

}

private: System::Void удалитьToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) {

СonfirmationOfTheDelete();

}

private: System::Void dataGridView2\_CellMouseUp(System::Object^ sender, System::Windows::Forms::DataGridViewCellMouseEventArgs^ e) {

if (e->Button == System::Windows::Forms::MouseButtons::Right)

{

this->ShowCMS(e);

}

}

private: System::Void comboBox2\_SelectedIndexChanged(System::Object^ sender, System::EventArgs^ e) {

String^ selectedValue = this->comboBox2->SelectedItem->ToString();

if (selectedValue == "Закупка")

{

db->openConnection();

dataGridView2->DataSource = db->getTableWorkZakup();

db->closeConnection();

}

else if (selectedValue == "Стандарт")

{

db->openConnection();

dataGridView2->DataSource = db->getTableWork();

db->closeConnection();

}

}

SettingForm.h

#pragma endregion

private: System::Void SettingsForm\_Load(System::Object^ sender, System::EventArgs^ e) {

this->Text = "Настройки рабочего места";

}

private: System::Void button1\_Click(System::Object^ sender, System::EventArgs^ e) {

System::Windows::Forms::DialogResult eva = colorDialog1->ShowDialog();

if (eva == System::Windows::Forms::DialogResult::OK) {

this->BackColor = colorDialog1->Color;

}

}

private: System::Void splitContainer1\_Panel1\_Paint(System::Object^ sender, System::Windows::Forms::PaintEventArgs^ e) {

this->Text = "Себестоимость";

}

private: System::Void label25\_Click(System::Object^ sender, System::EventArgs^ e) {

}

private: System::Void button4\_Click(System::Object^ sender, System::EventArgs^ e) {

this->Close();

}

private: System::Void button3\_Click(System::Object^ sender, System::EventArgs^ e) {

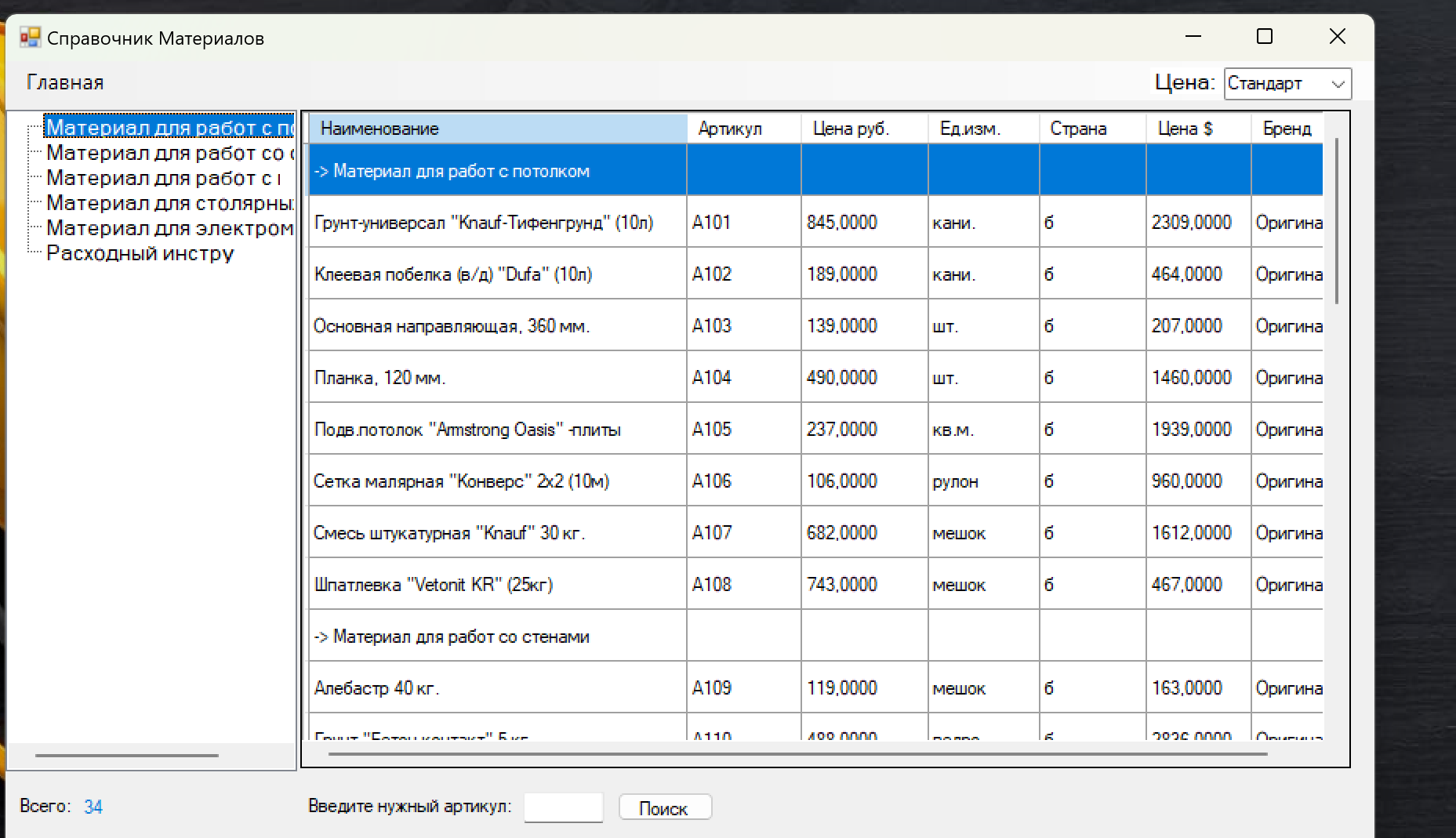
this->Close();

}

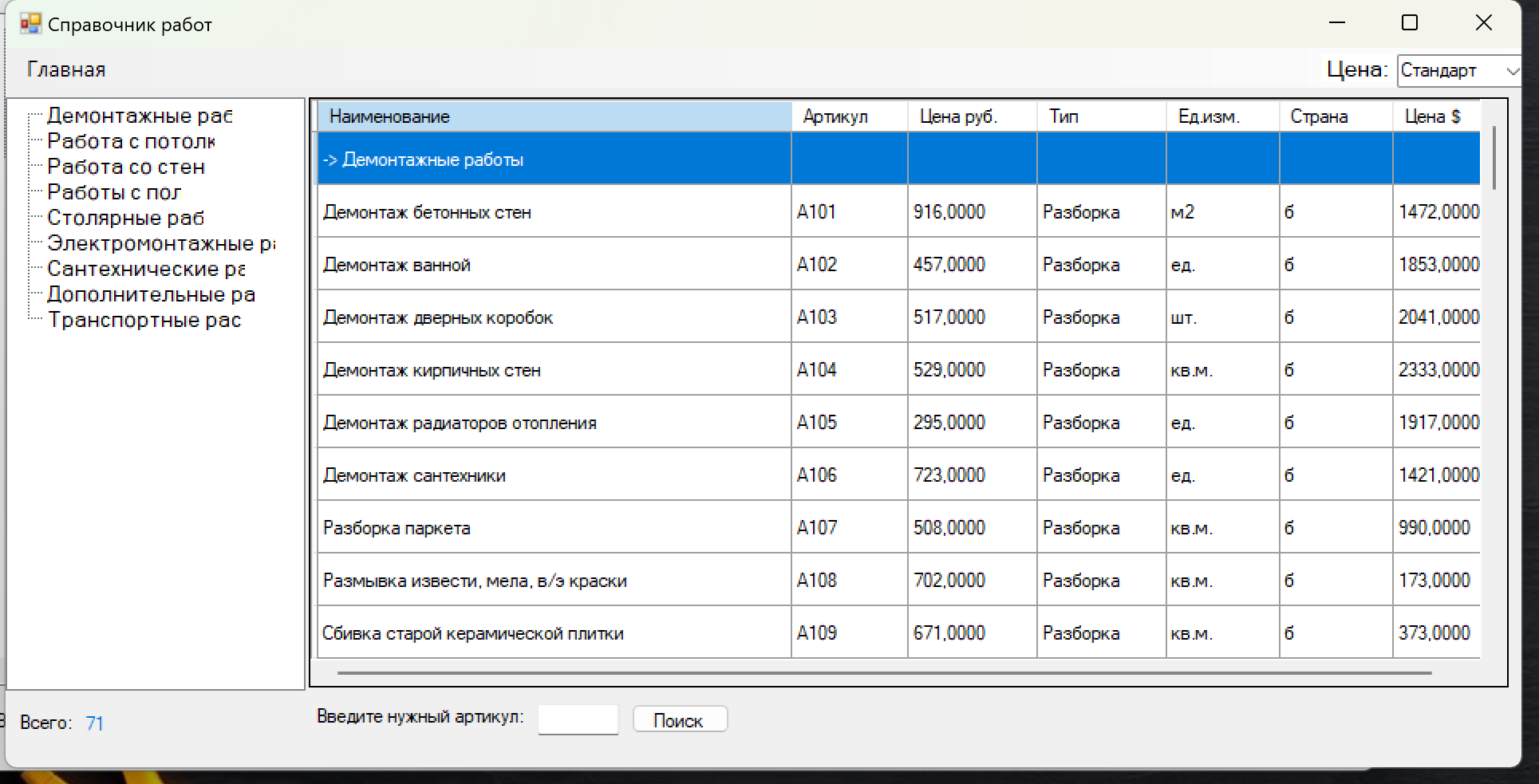
};

}

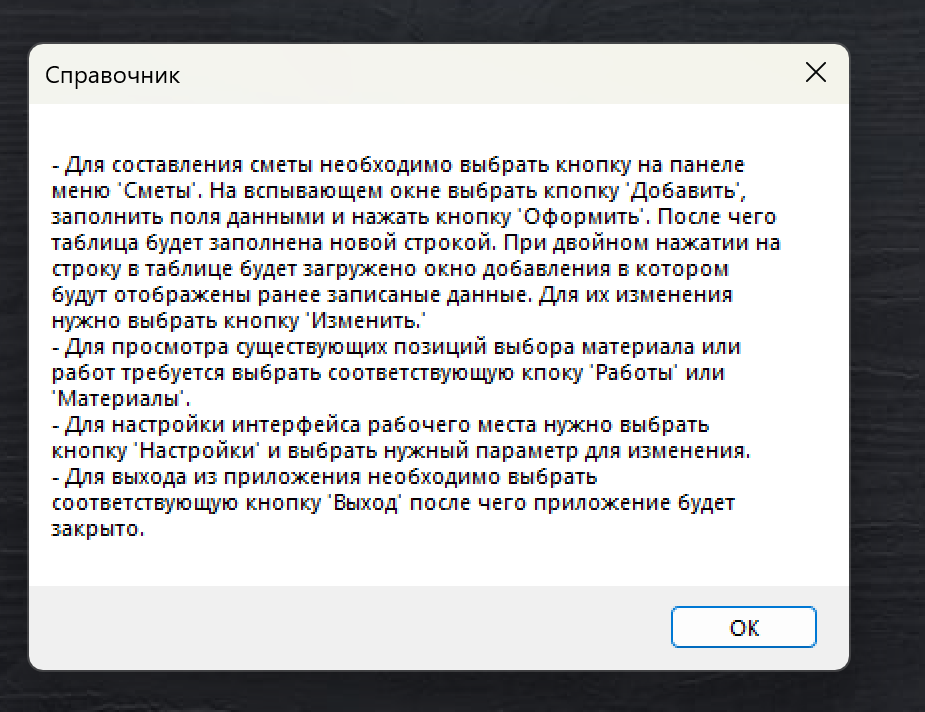
# Приложение Б Формы входных и выходных документов



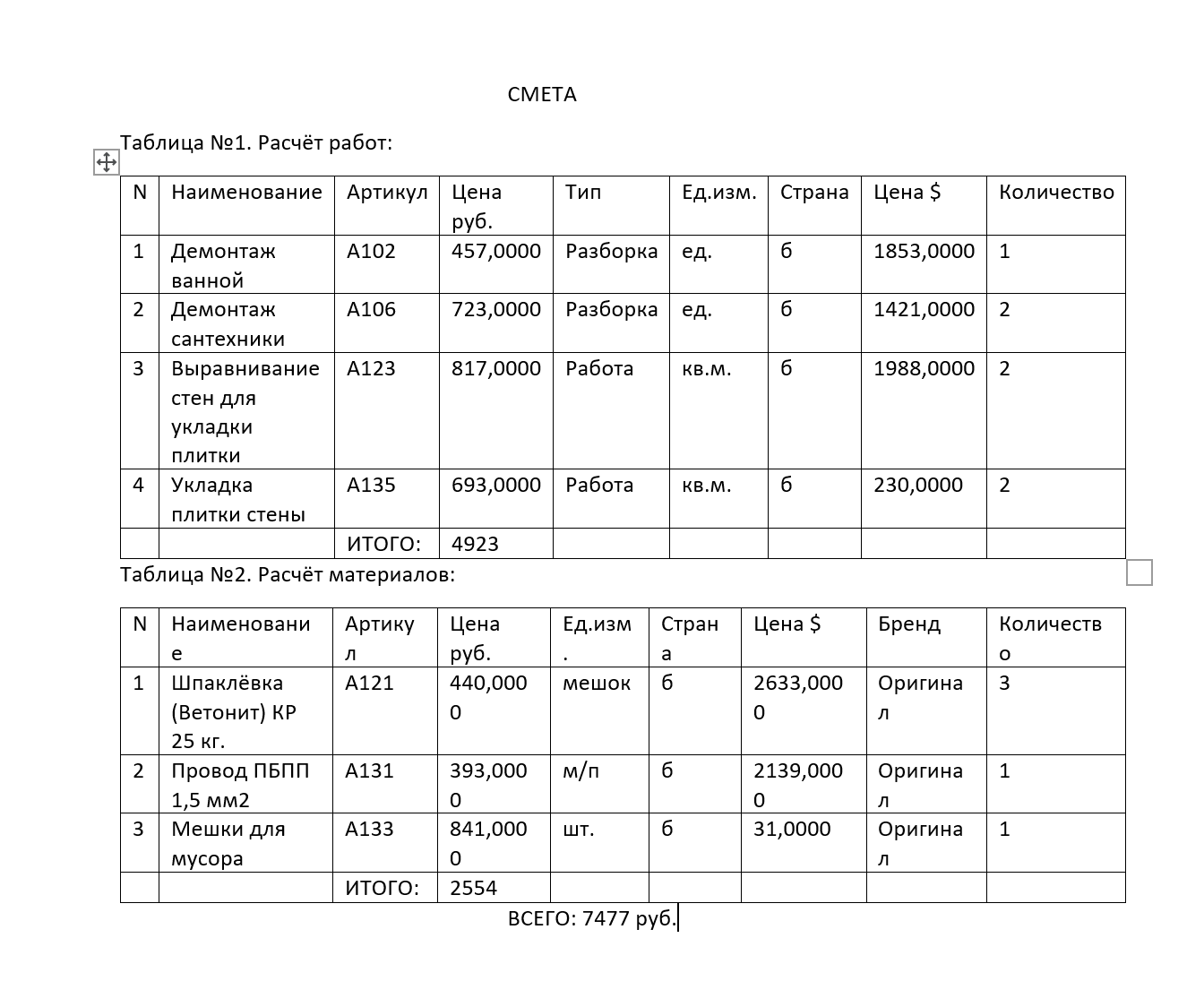
1. − Справочник материалов



1. − Справочник работ

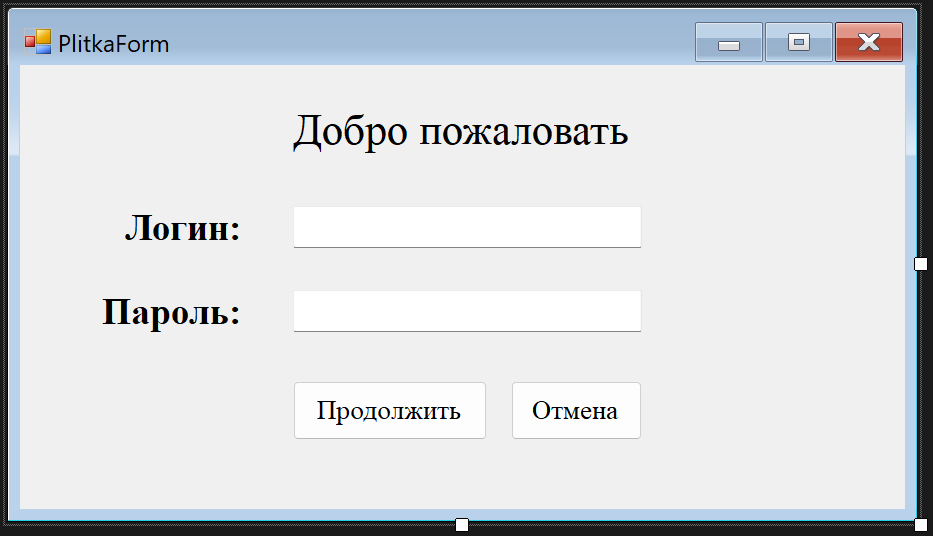


1. − Справочная система для пользователя

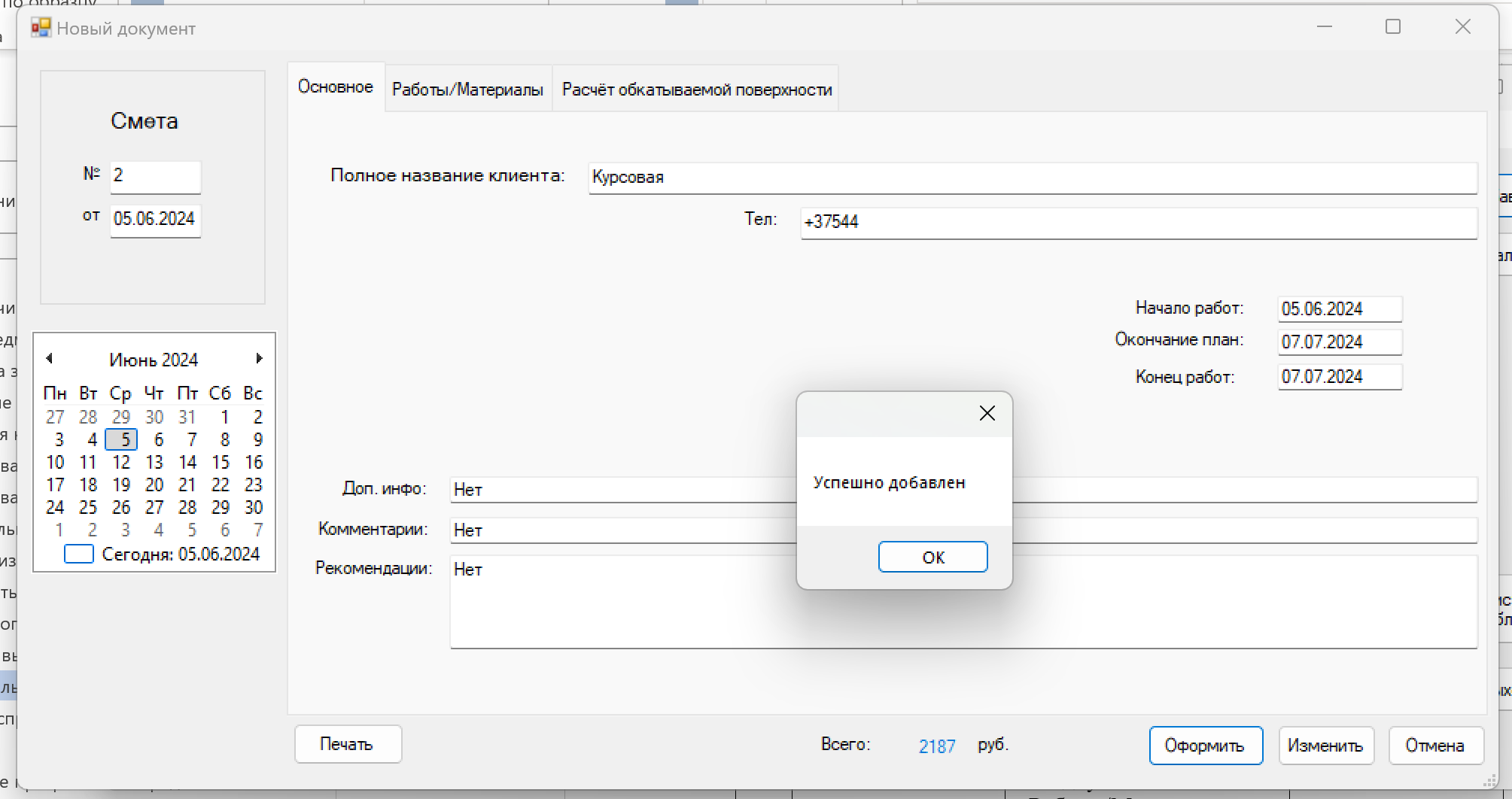


1. − Смета ремонтно-отделочных работ

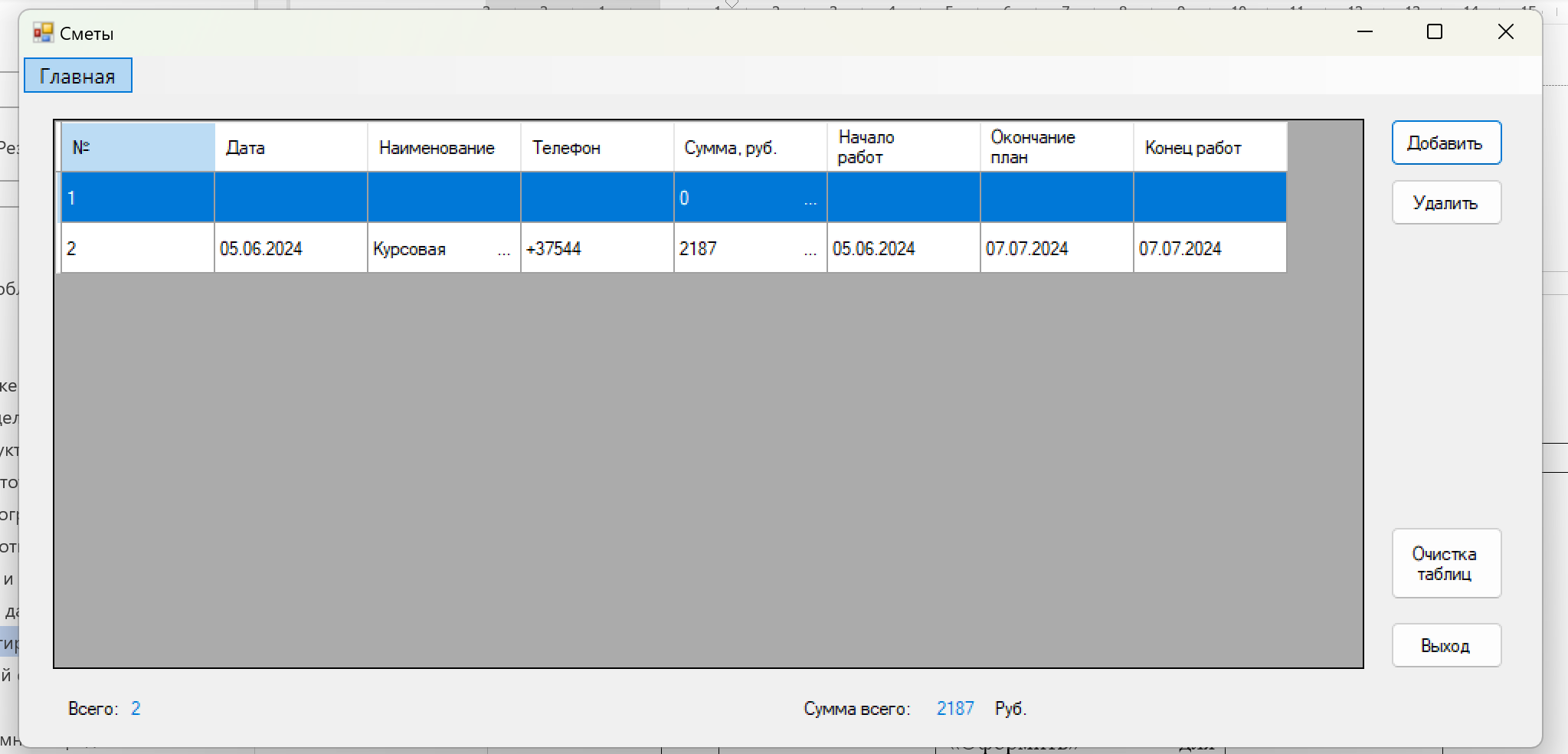
# Приложение В Результаты работы программы



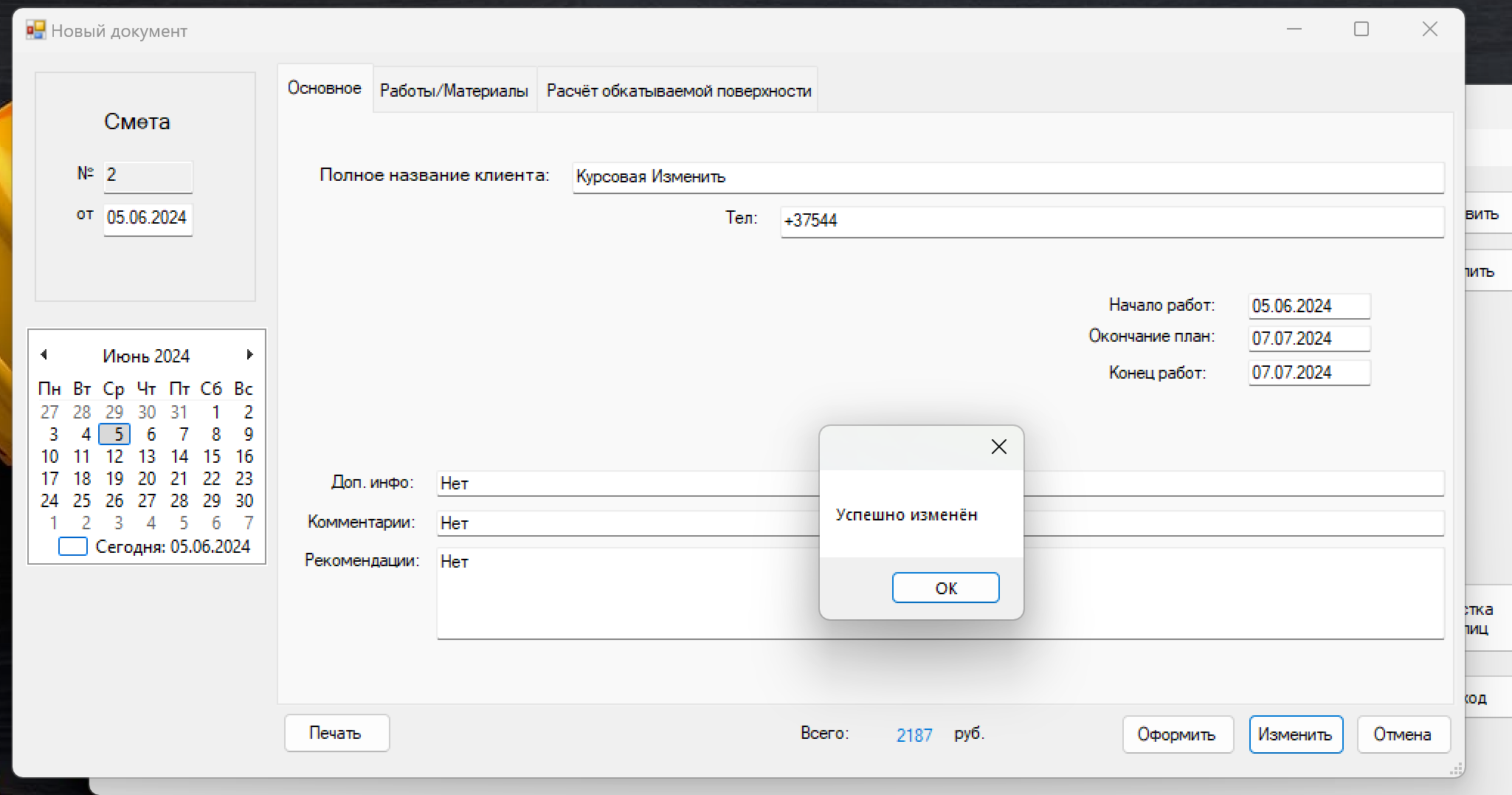
1. – Вход в приложение



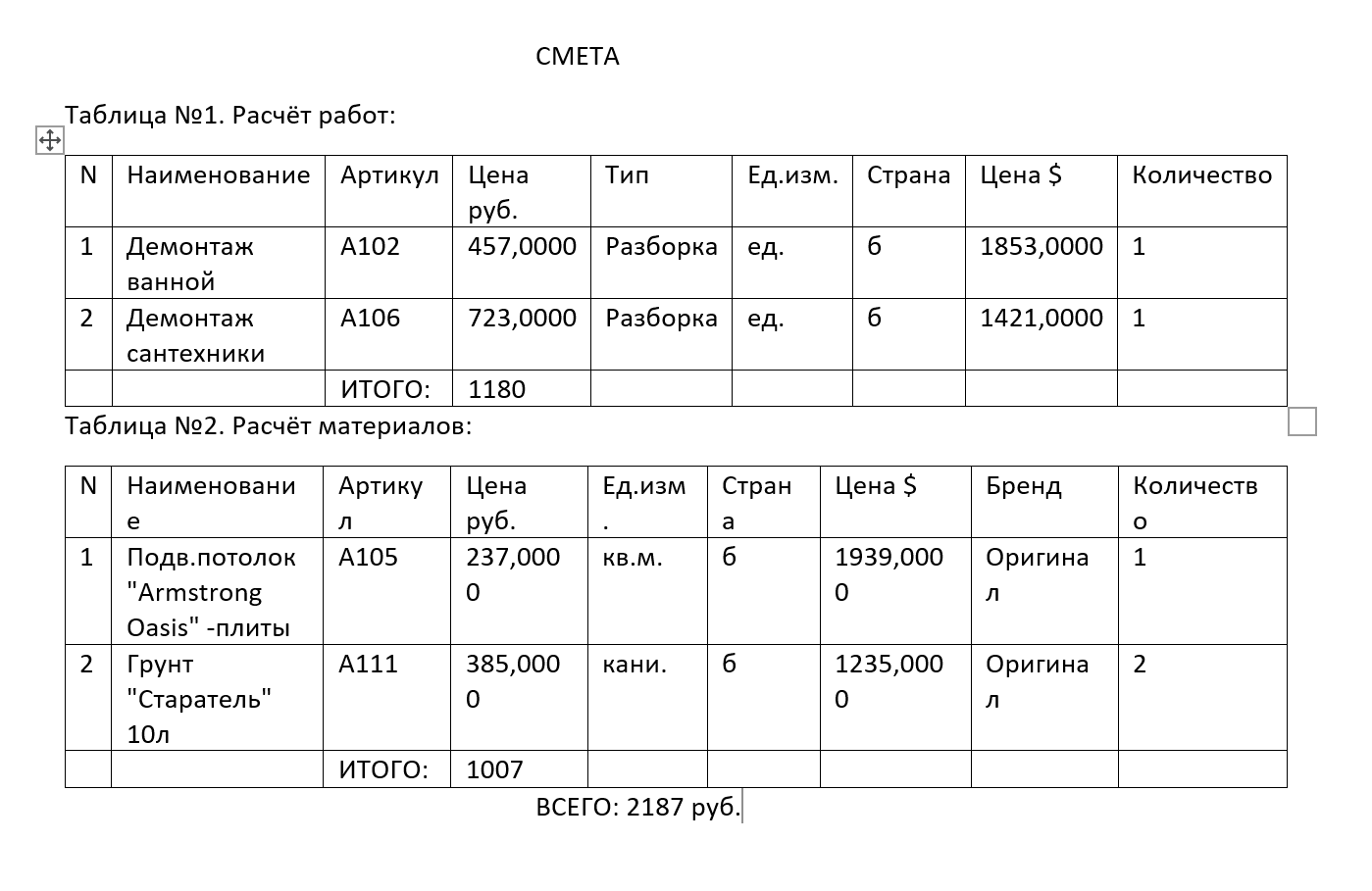
1. – Окно заполнения данных.



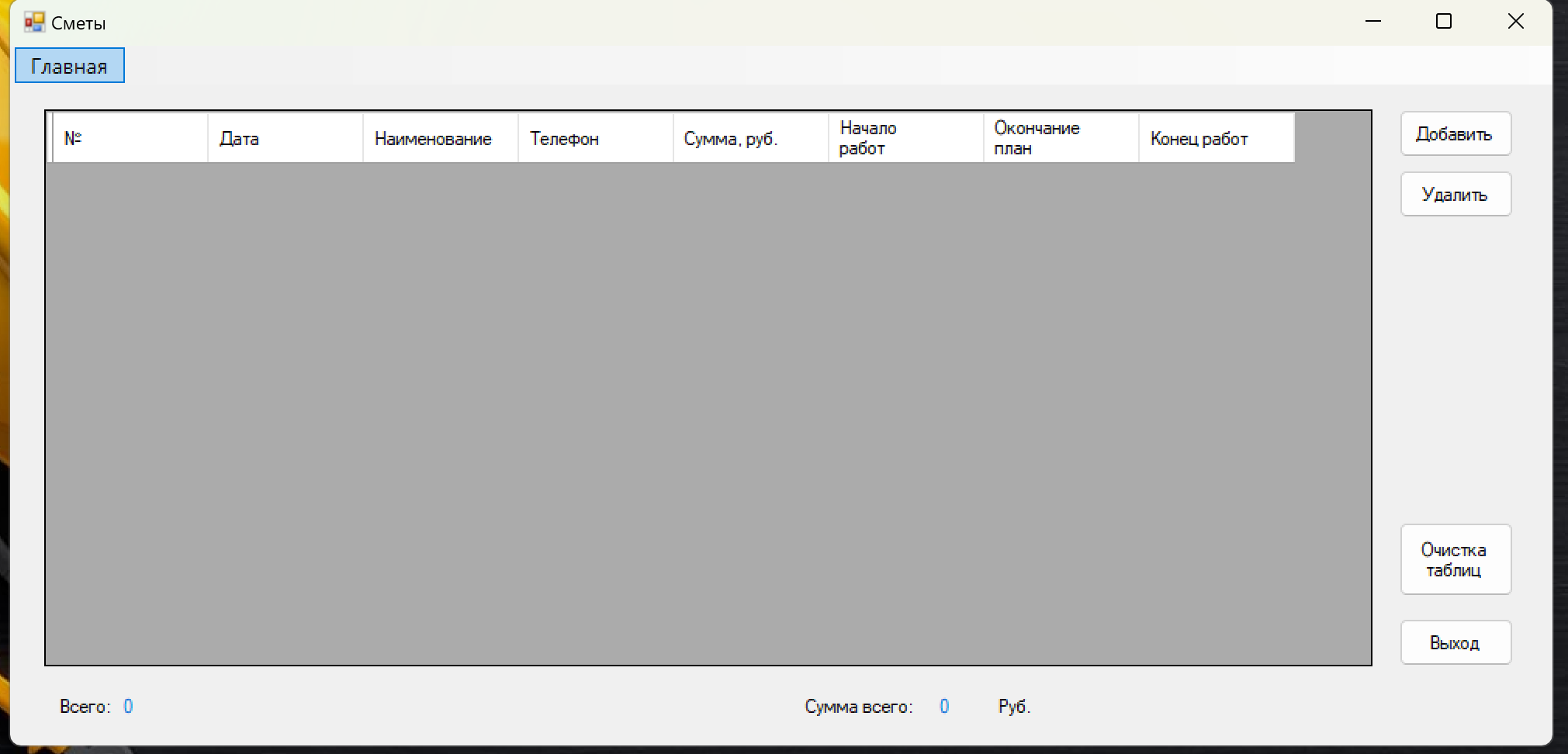
1. – Оформление сметы



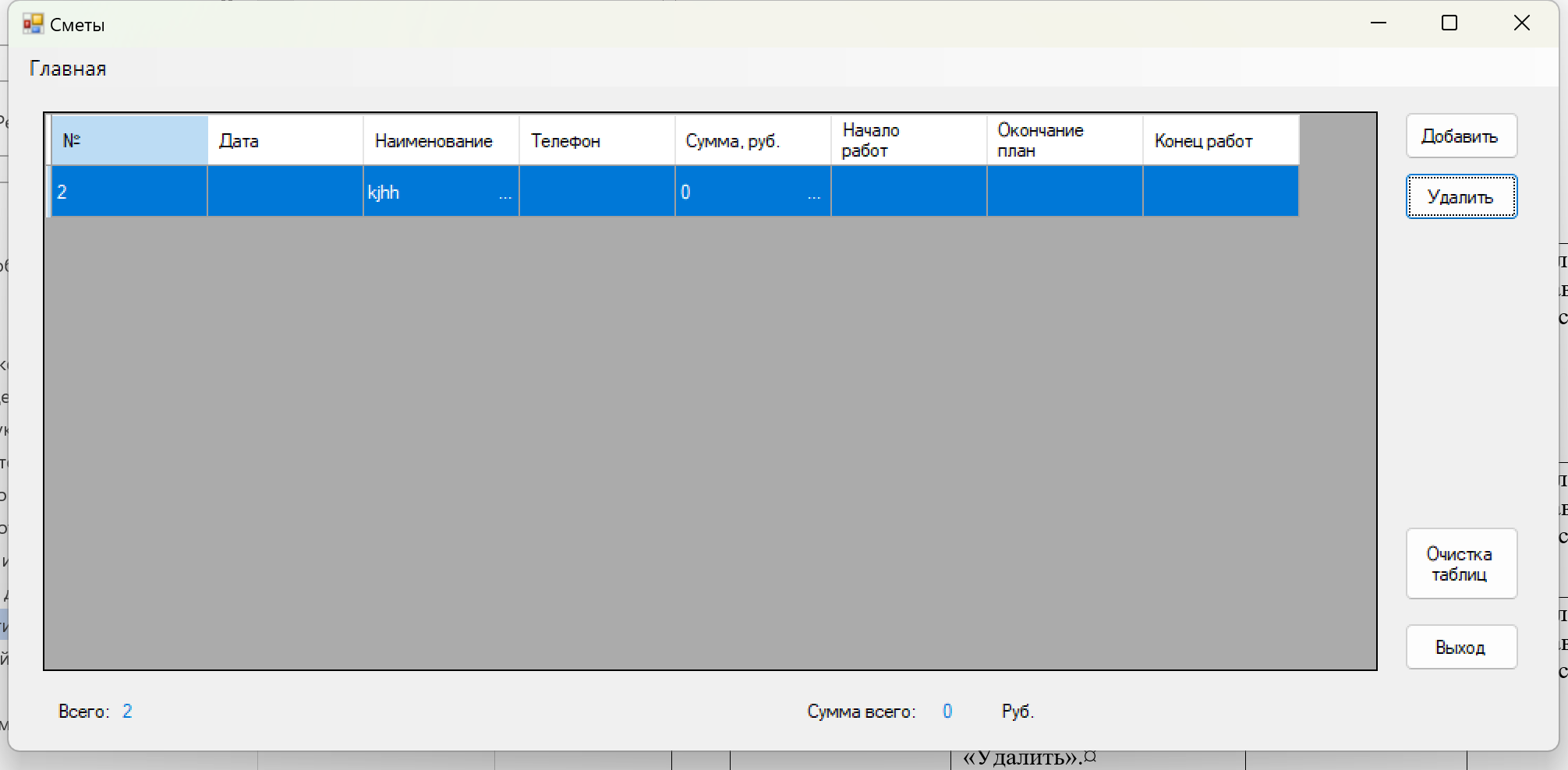
1. – Изменения раннее созданной сметы



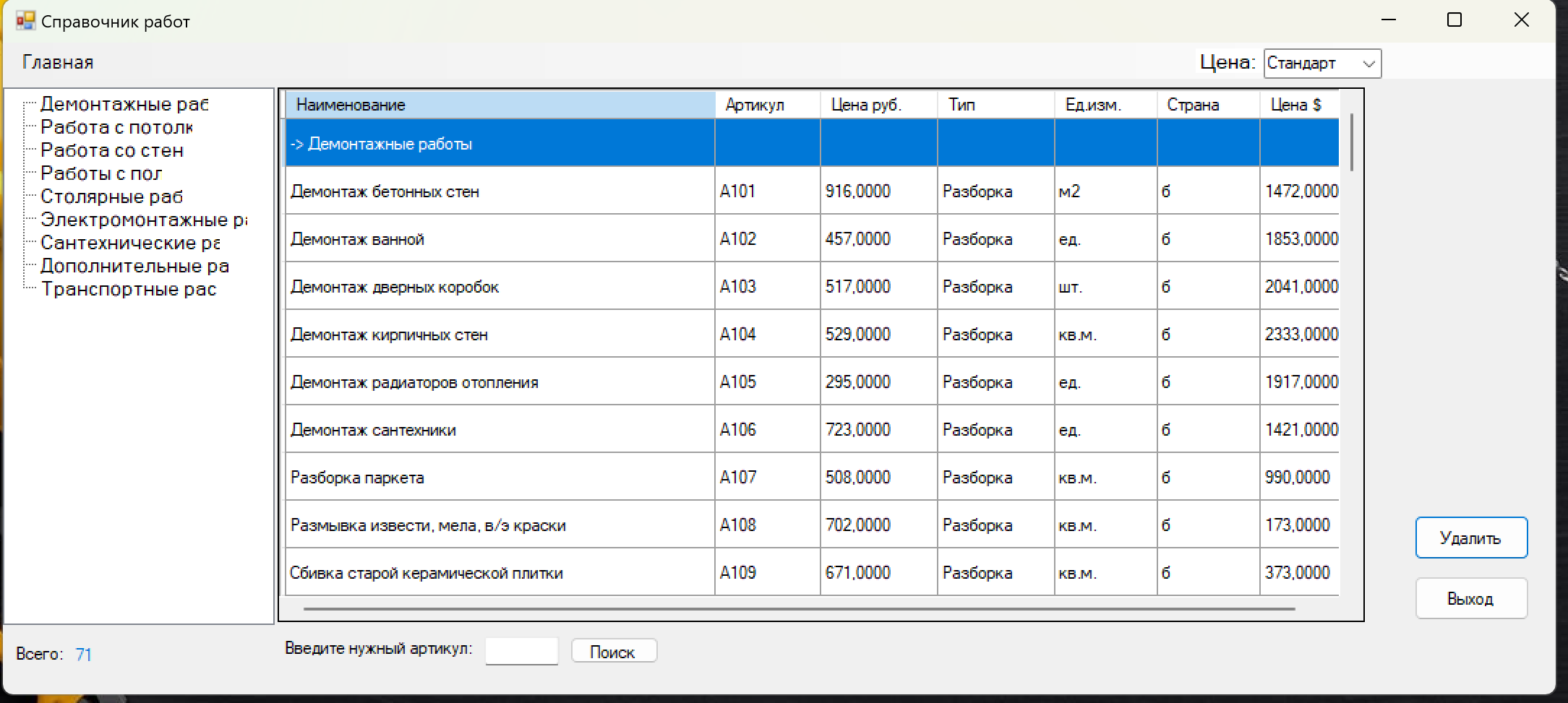
1. – Автозаполнение сметы в формате doxc



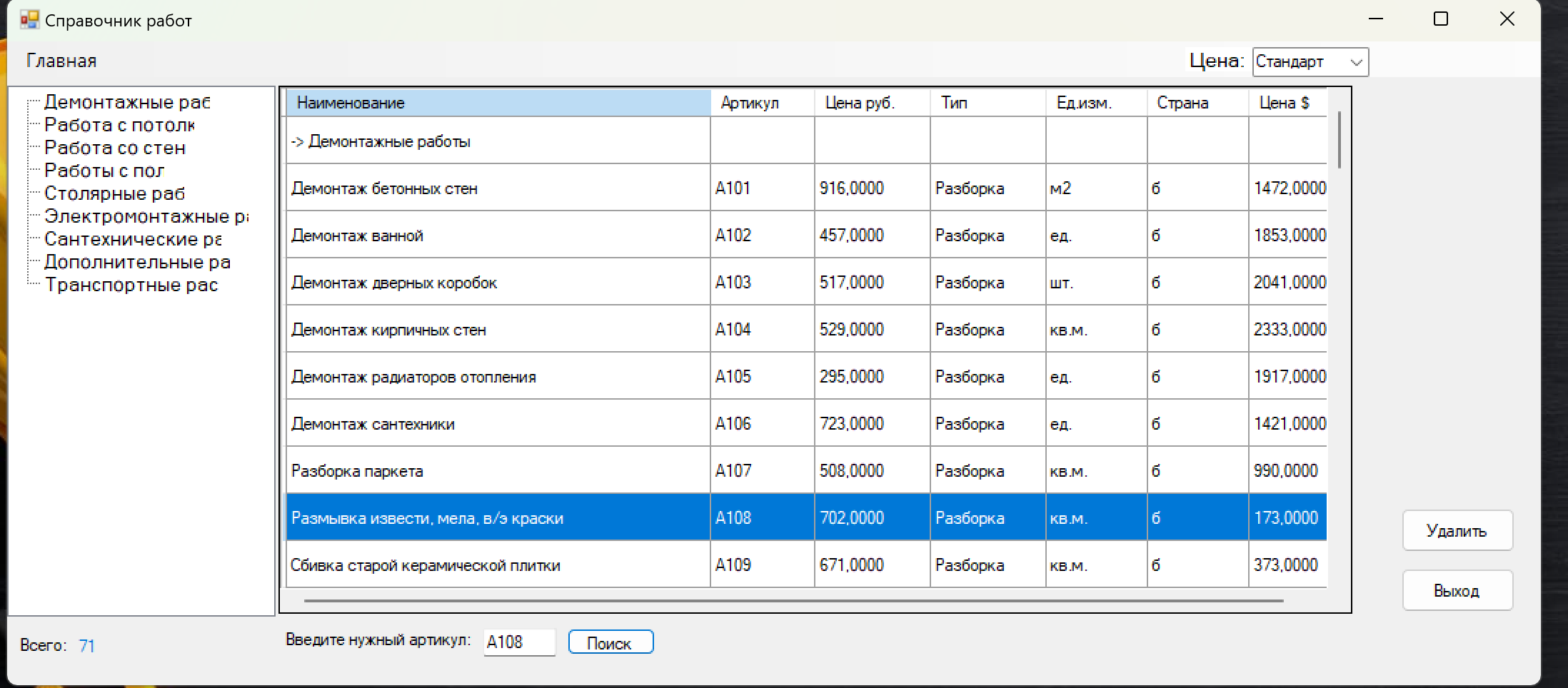
1. – Сброс данных таблиц



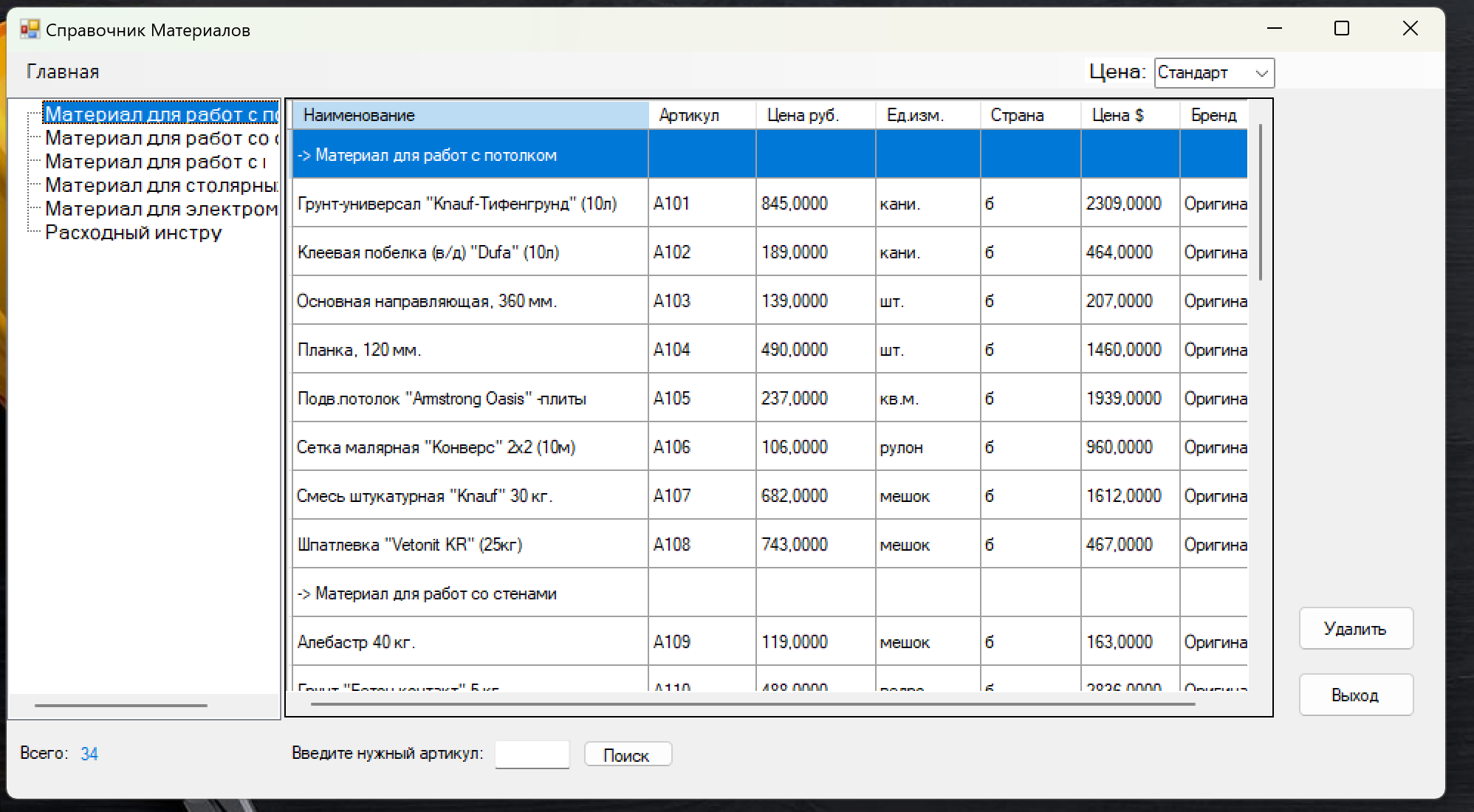
1. – Сброс одной строки



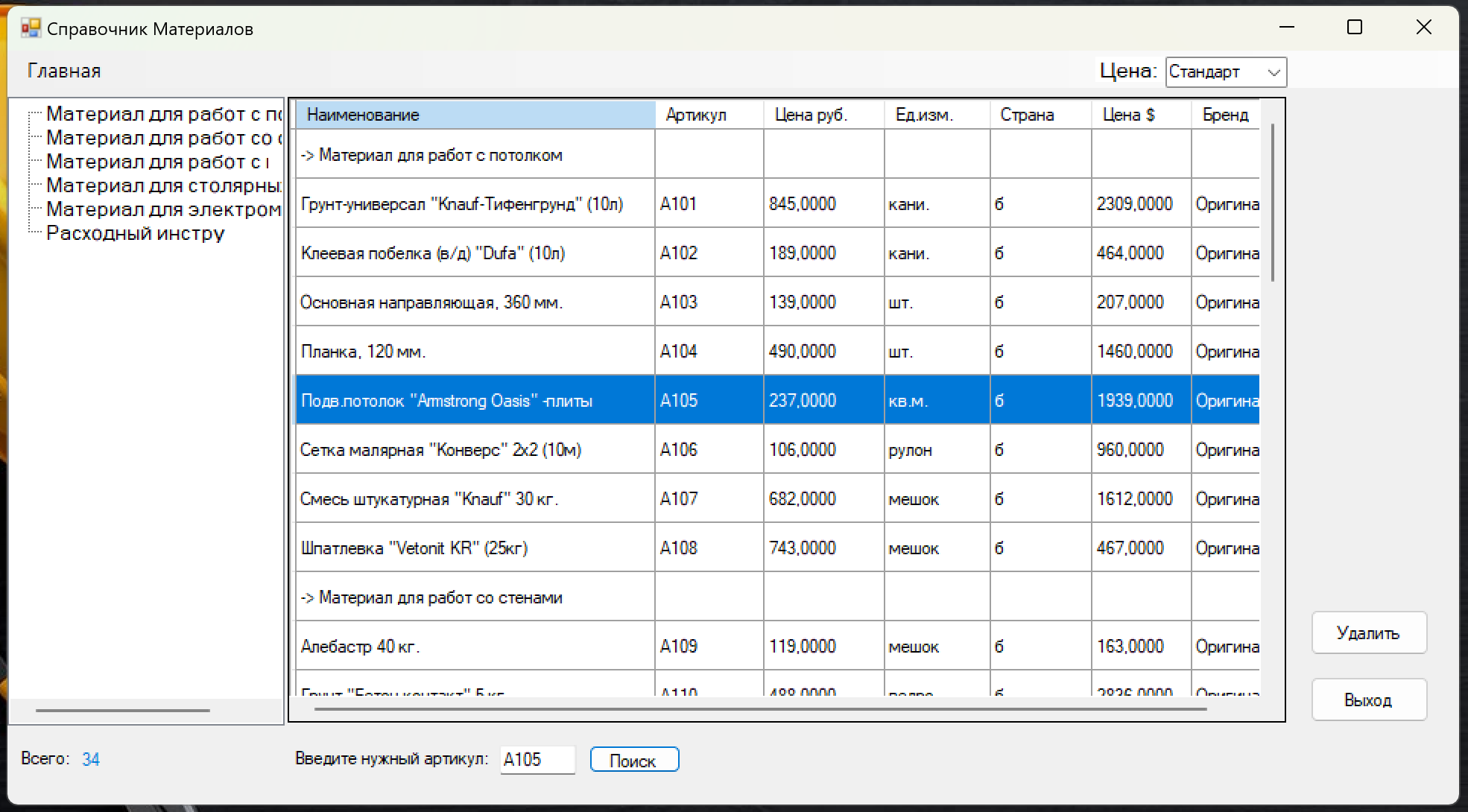
1. – Окно со справочником работ



1. – Поиск в справочнике работ



1. – Окно со справочником материалов



1. – Поиск в справочнике материалов

КБП

У

Т. Контр.

Разраб. Разраб.

Шабуневич З.Н

Провер. Провер.

Рогалевич А.В.

Реценз.

Утверд.

Н. Контр.

Схема работы алгоритма расчёта обкатываемой поверхности

№ докум.

Подпись

Дата

Изм.

Лист

Масса

Лит.

Масштаб

Лист 1

Листов 2

Инв.№подл.

Подп. и дата

Взам.инв.№

Инв.№дубл.

Подп. и дата

КП Т.319026.401 ГЧ

КП Т.319026.401 ГЧ

*«Реализация программного средства для составления сметы ремонтно-отделочных работ плиточного мастера»*

Отображение всех результатов

Графическое отображение примерной поверхности

Подсчёт площади обкатываемой поверхности

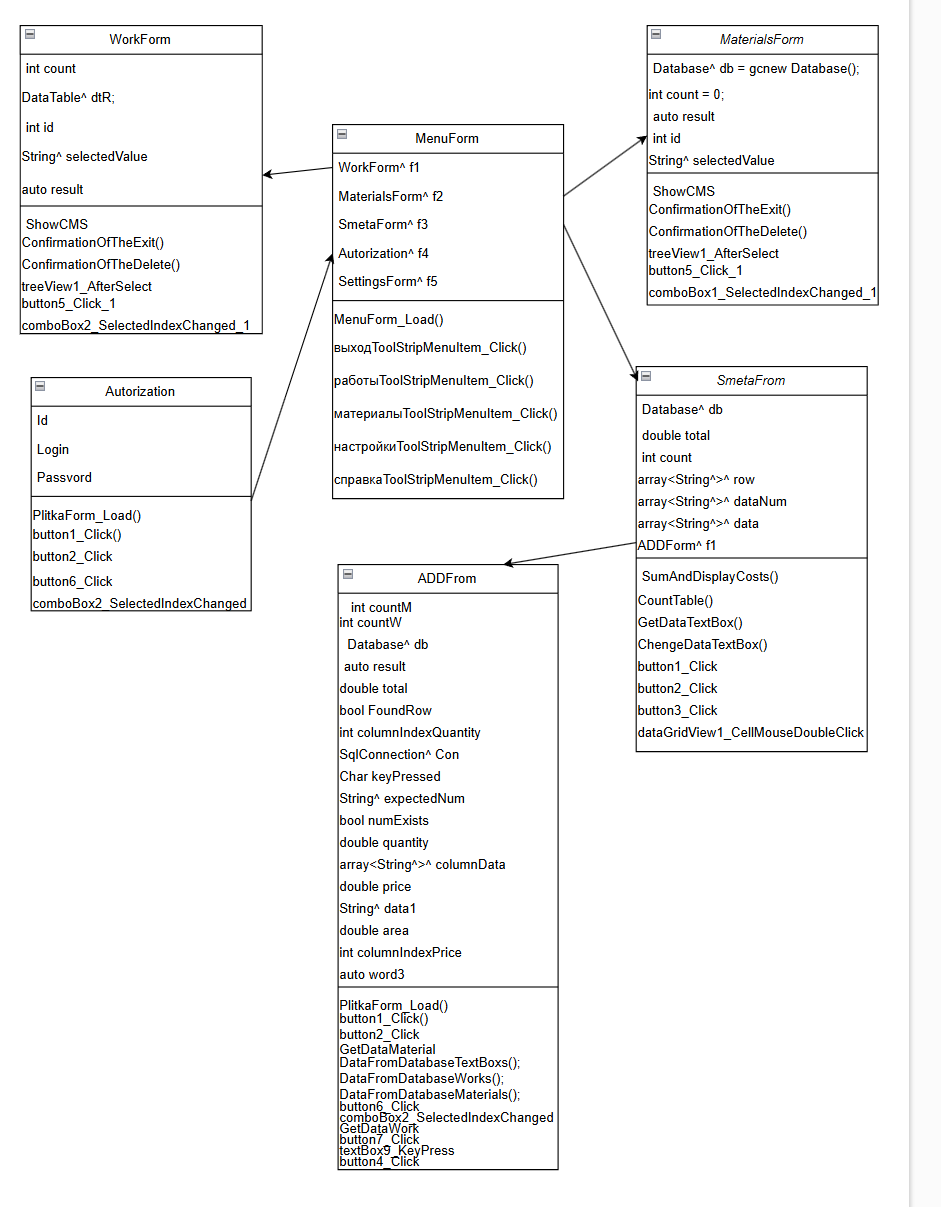
Подсчёт количества плитка

Обработка данных

Чтение данных и текстовых полей

Ввод данных в текстовые поля для расчёта поверхности

Ввод необходимых материалов и работ



КП Т.319026.401 ГЧ

КБП

У

Т. Контр.

Разраб. Разраб.

Шабуневич З.Н

Провер. Провер.

Рогалевич А.В.

Реценз.

Утверд.

Н. Контр.

№ докум.

Подпись

Дата

Изм.

Лист

Масса

Лит.

Масштаб

Лист 2

Листов 2

Инв.№подл.

Подп. и дата

Взам.инв.№

Инв.№дубл.

Подп. и дата

КП Т.319026.401 ГЧ

*«Реализация программного средства для составления сметы ремонтно-отделочных работ плиточного мастера»*

Диаграмма классов

**Удостоверяющий лист**

электронного документа – курсовой проект

Тема «Реализация программного средства для составления сметы ремонтно-отделочных работ плиточного мастера»

Обозначение КП Т.319026.401

Разработчик Шабуневич З.Н. Руководитель Рогалевич А. В.

(Ф.И.О.) (Ф.И.О.)

Подписи лиц, ответственных за разработку электронного документа

|  |  |  |
| --- | --- | --- |
| Состав электронного документа | Разработчик | Руководитель |
| Пояснительная записка (на бумажном носителе формата А4), ПЗ\_Шабуневич\_Т319.docх |  |  |
| Папка с проектом «CLRStroyBat» |  |  |
| Файл базы данных «PlitkaBase.mdf» |  |  |
| Тип носителя: USB-flash накопитель |  |  |

**Этикетка**

для курсового проекта

**Курсовой проект**

Тема «Реализация программного средства для составления сметы ремонтно-отделочных работ плиточного мастера»

КП Т.319026.401

Разработан

Утвержден

Разработчик Шабуневич З.Н

Руководитель: Рогалевич А.В.

Технические средства: Компьютер с процессором AMD A12-9720P от 2.6 ГГц, ОЗУ от 4 ГБ, 600 МБ свободного места на диске, монитор, мышь Программные средства: Операционная система MS Windows 11, MS Visual Studio 2022, текстовые редакторы Microsoft Word 2019

**Состав документа:**

Пояснительная записка – ПЗ\_Шабуневич\_Т319.docх

Папка с проектом – CLRStroyBat

Файл базы данных – PlitkaBase.mdf

Сведения о защите информации: логин отсутствует, пароль отсутствует