

Министерство образования Республики Беларусь
Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ
Институт информационных технологий
Кафедра микропроцессорных систем и сетей

К защите допустить:

Заведующий кафедрой

_____ И.В. Кашникова

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к дипломному проекту

на тему

**ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ЗАПИСИ КЛИЕНТОВ В
МУЖСКОЙ САЛОН СТРИЖКИ И БРИТЬЯ**

Дипломник _____

Н.П. Матюшенко

Руководитель _____

И.И. Гламаздин

Консультанты:

по ЕСПД и ЕСКД _____

В.Н. Видничук

по специальности

А.А. Москалев

Рецензент _____

2024

Содержание

Условные обозначения и сокращения.....	3
Введение	4
1 Обзор и постановка задач	6
1.1 . Обзор существующих решений по теме проекта и постановка задачи.....	6
1.2 Сравнение программных продуктов	10
1.3 Постановка задач.....	11
2 Разработка методов и моделей	12
2.1 Моделирование предметной области.....	12
2.2 Разработка структуры базы данных	13
3 Разработка проекта программного обеспечения	15
3.1 Функциональный анализ предметной области	15
3.2 Разработка схемы ресурсов системы	18
3.3 Разработка схемы базы данных	19
3.4 Разработка диаграммы классов	24
3.5 Разработка графического интерфейса пользователя	25
4 Разработка алгоритмов и их описание	31
4.1 Алгоритм свободного времени.....	31
4.2 Алгоритм свободного мастеров.....	32
4.3 Алгоритм свободной даты	33
4.4 Алгоритм отчета.....	34
4.5 Алгоритм сохранения данных	36
5 Тестирование полученного программного продукта	37
Заключение.....	45
Список использованных источников	46
Приложение А.....	47
Приложение Б	53
Приложение В.....	55
Приложение Г	56
Приложение Д.....	57
Приложение Е	59

Условные обозначения и сокращения

В пояснительной записке применяются следующие определения и сокращения.

Программное обеспечение (программное средство, программный комплекс) – совокупность программ системы обработки информации и программных документов, необходимых для эксплуатации этих программ.

Программирование – научная и практическая деятельность по созданию программ.

Программный модуль – программа или функционально завершенный фрагмент программы, предназначенный для хранения, объединения с другими программными модулями и загрузки в оперативную память.

Система программирования – система, образуемая языком программирования компиляторами или интерпретаторами программ, представленных на этом языке, соответствующей документацией, а также вспомогательными средствами для подготовки программ к форме, пригодной для выполнения.

Фреймворк (англ. Framework) – заготовки, шаблоны для программной платформы, определяющие структуру программной системы; программное обеспечение, облегчающее разработку и объединение разных модулей программного проекта.

Программный интерфейс приложения (англ. application programming interface, сокр. API) – Интерфейс приложения — это среда, которую пользователь видит и с которой взаимодействует при использовании приложения. Он включает в себя все элементы управления, кнопки, меню, диалоговые окна и другие элементы, которые помогают пользователю управлять приложением и выполнять с его помощью различные задачи.

SQL (Structured Query Language) – язык программирования, предназначенный для управления данными в реляционной СУБД.

Введение

Как показывает практика, деятельность салонов красоты, парикмахерских нацелена большего всего на клиента. Оказать максимально качественные услуги. Но порой никто не задумывается как ведется учет записи всех клиентов.

По моим наблюдениям могу сделать вывод, что не все организации по оказанию бытовых услуг могут позволить себя полноценное программное обеспечение.

Для успешной работы необходимо постоянно владеть информацией о времени и работе мастера, так же о виде оказания услуг и их стоимости. С этим и помогают справляться системы учета записи клиентов и работы мастеров.

В настоящее время в распоряжении организации имеются программные средства, позволяющие вести учет всех процессов на предприятии. Существующие программные средства являются достаточно затратными по внедрению и сопровождению для микро-организаций и индивидуальных предпринимателей или самозанятых, обладают избыточным функционалом, требуют дополнительного оборудования и обучения.

В большинстве случаев был замечен способ ведения такого учета - это ручной учет в журналах, либо ведение таблиц Excel, что очень трудоёмко и не исключает большого количества ошибок из-за человеческого фактора.

Выходом можем стать разработка автоматизированной системы, которая позволит вести учет клиентов и оказание услуг в режиме реального времени, отслеживать работу мастера и свободного времени для записи клиентов.

Актуальность работы: данное исследование направлено на решение одной из важнейших проблем микро-организаций – учет записи клиентов и работу мастеров, который является залогом точного и своевременного отражения оказания услуг.

Целью данного проекта является повышение эффективности труда сотрудников, путем минимизации ручного труда с бумажными документами.

Для достижения поставленной цели необходимо решить следующие задачи:

- провести обзор аналогов, выявить их достоинства и недостатки, провести анализ литературных источников и сформировать требования к

проектируемому программному средству;

- смоделировать предметную область и разработать функциональные требования;
- разработать программное средство с использованием выбранных средств разработки;
- провести функциональное тестирование программного средства и предоставить руководство пользователя;
- оценить экономическую эффективность реализации программного средства.

В результате проектирования дипломного проекта разработанное программное средство представляет собой клиент-серверную систему с использованием технологий баз данных. Эксплуатируемой базой данных является SQL-ориентированная база данных, расположенная на выделенном сервере Microsoft SQL Server 2017. Языком реализации программного средства является C#, среда разработки Microsoft Visual Studio 2022.

.

1 Обзор и постановка задач

1.1 Обзор существующих решений по теме проекта и постановка задачи

В настоящее время на рынке программного обеспечения (далее ПО) для такого направления как бытовые услуги выбор достаточно скромный, а точнее по моим анализам их только два.

Они отличаются друг от друга набором функций, стабильностью, удобством работы, стоимостью.

Приступая к анализу основных из них, следует отметить, что на рынке отсутствуют решения, предназначенные исключительно для учета записи клиентов для оказания услуг в направлении парикмахерских услуг.

Для анализа выбраны следующие системы: 1С: Управление небольшой фирмой для Беларуси и YCLIENS - Сотни задач. Одна экосистема

Примеры программных окон представлены на рисунке 1-6.

1С: Управление небольшой фирмой для Беларуси [1]

Комплексное готовое решение для учета на предприятиях малого и среднего бизнеса. На рисунках 1-2 представлены примеры окон программы.

Барбершоп (1С:Предприятие, учебная версия)

Продажа (создание)

Клиент: Дата: 26.01.2024 0:00:00

Услуга: Сумма: 0.00

Сотрудник: Салон:

Запись: Комментарий:

Рисунок 1 - 1С: Управление небольшой фирмой для Беларуси:

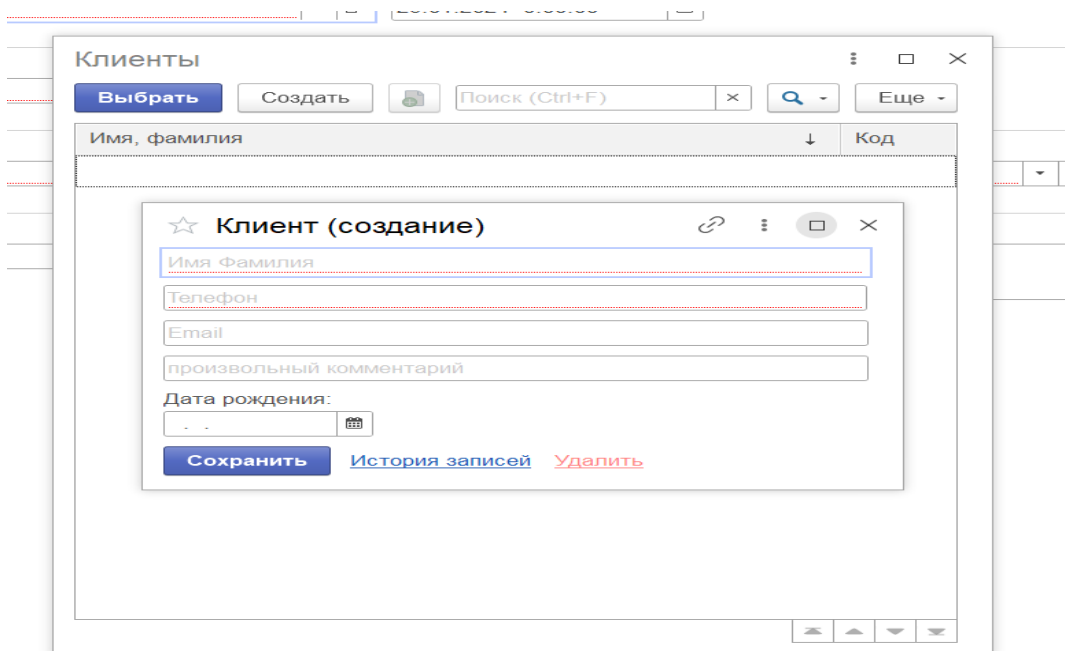


Рисунок 2 – Интерфейс приложения - 1С: Управление небольшой фирмой для Беларуси: справочник клиентов

Данная программа позволяет работать одновременно большому количеству пользователей, осуществлять учет по работе мастеров и записи клиентов. Создавать отчет по финансовым показателям работы предприятия и отражение заработной платы мастерам за выполненные работы.

«YCLIENS - Сотни задач. Одна экосистема» [2]

Программа «YCLIENS» — специализированное программное оснащение для салонов красоты. На рисунках 3-5 представлены примеры окон программы.

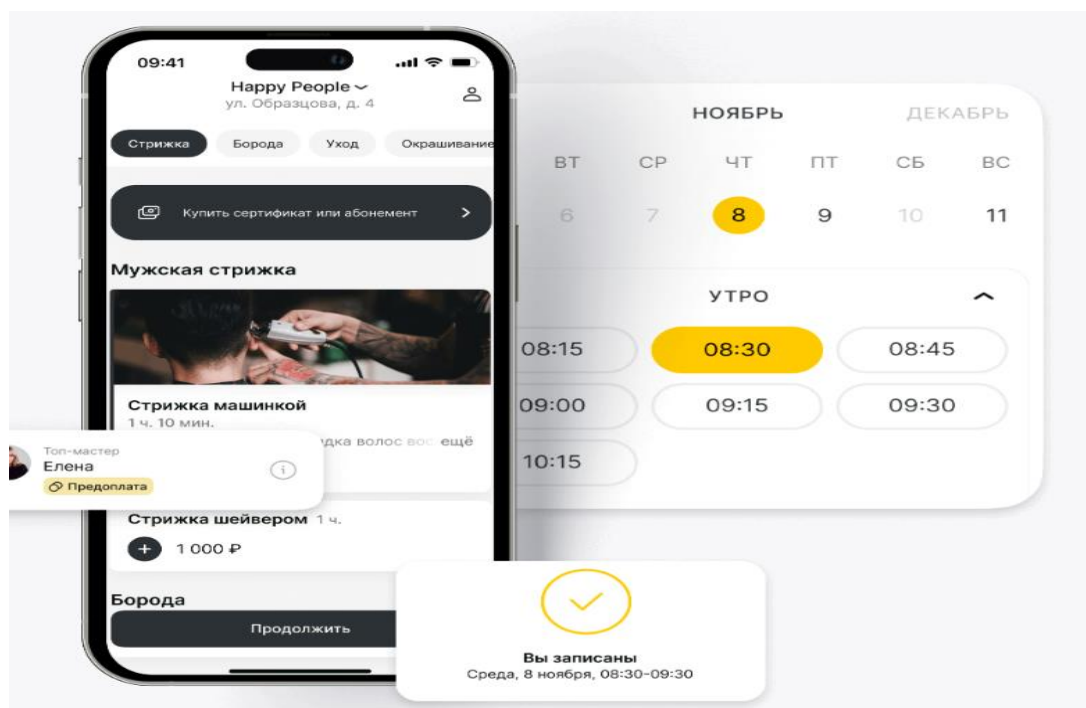


Рисунок 3 – Интерфейс программы - YCLIENS онлайн запись

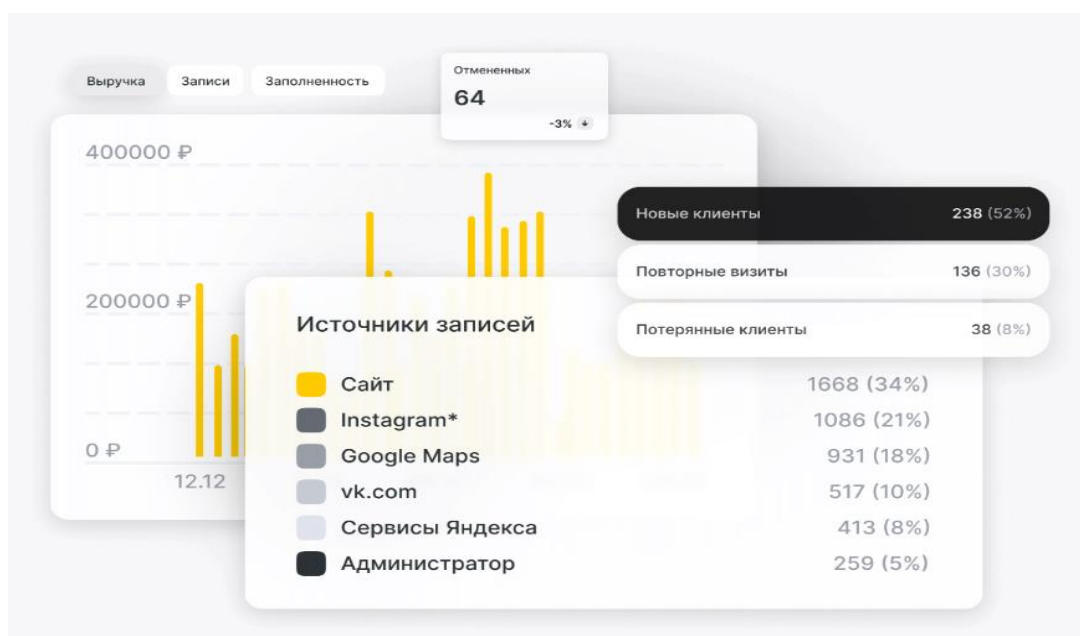


Рисунок 4 – Интерфейс приложения YCLIENS: Карточка клиента с полной историей и информацией

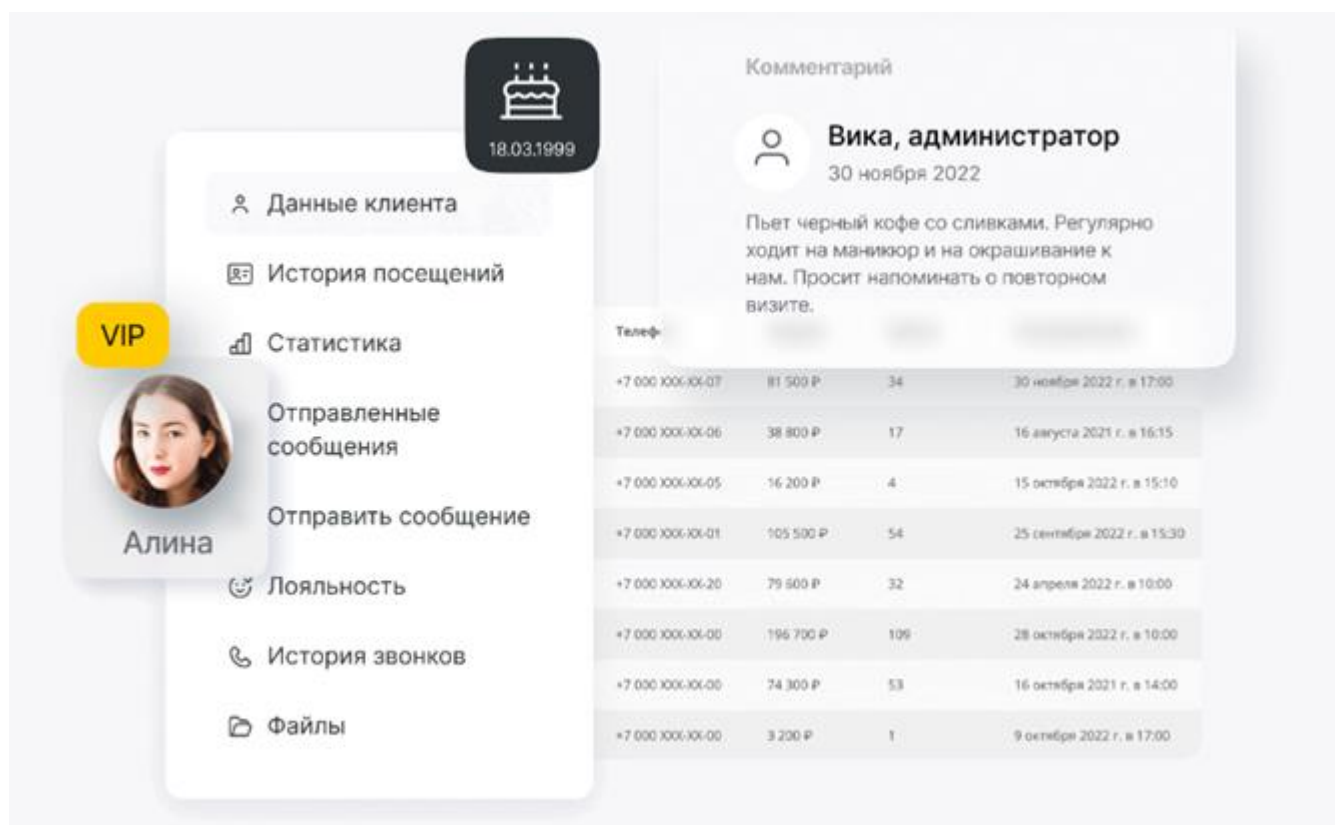


Рисунок 5 - Интерфейс приложения YCLIENS: Аналитика работы предприятия

Данное решение предназначено является полностью клиенто - ориентированным приложением. Понятный интерфейс, максимальное наполнение функций для удобной работы. Не требует особого обучения.

1.2 Сравнение программных продуктов

В таблице 1 проанализированы основные сравнительные характеристики данных программ. Сравнение проведено по минимальным пакетам, представленным разработчиками данных программ. В минимальный пакет должно входить: работа 1-2 сотрудников, 2 и более точки продаж, возможность подключения дополнительных опций, возможность работать в сети.

Таблица 1 – Сравнительные характеристики программ-аналогов

	1С: типовая конфигурация	YCLIENS
Количество сотрудников	До 5	До 5
Стоимость подключения, руб.	Лицензия бессрочно 1410	Метод лицензия 1000 в год
Подключение доп. мест	+	+
Излишний функционал	+	+
Необходимость обучения	Есть инструкция	Есть инструкция
Подключение доп. опций	+	+

Как видим из таблицы - эти программы обладают существенными недостатками для микро-организаций (ограниченность в финансовых ресурсах): достаточно высокая стоимость, необходимость обучения, платное сопровождение, оплата подключения дополнительного рабочего места и приобретение в пакете излишнего функционала, так же существует проблема для малого бизнеса с оплатой за продукт, так как в основном в предоставленных программах является разработчик из Российской Федерации.

Таким образом, принято решение в ходе данной работы реализовать собственное ПО для учета записи клиентов, максимально адаптированную

под финансовые возможности малого бизнеса. ПО позволит владельцу бизнеса получить необходимый функционал с минимальными затратами.

1.3 Постановка задач

Планируется разработать ПО для учета записи клиентов согласно выбранной услуге и мастеру. В разрабатываемой программе должны быть реализованы следующие функции:

- авторизация пользователя
- добавление и удаление клиента;
- выбор даты и времени;
- выбор услуги и стоимости;
- выбор дополнительных услуг и стоимости;
- формирование отчета по работе мастеров и вывод данных в табличную форму Excel;
- поиск по заданным признакам и вывод данных в табличную форму Excel;
- отслеживание статуса заявки;
- расчет заработной платы мастера за заданный период

ПО должно обладать простым, удобным и легко осваиваемым интерфейсом. Также должна быть разработана база данных (далее БД), в которой предусмотрено хранение информации о дате, времени, имени клиента, мастер, который выполнил услуг, либо будет оказывать услуг, вид услуги и стоимость услуги либо услуг, в случае выбора дополнительно. Также в БД должна храниться информация о пользователях. БД должна обеспечивать надежное хранение и доступ к информации. Вход в приложение должен осуществляться по логину и паролю.

Приложение не должно быть перегружено излишним функционалом.

Программа должна быть написана с использованием языка программирования C# с использованием Entity Framework. Графический интерфейс реализуется с использованием технологии Windows Presentation Foundation Microsoft.NET Framework (WPF). В качестве СУБД будет использоваться MS SQL SERVER [3].

2 Разработка методов и моделей

2.1 Моделирование предметной области

Разрабатываемое приложение должно включать возможность регистрации и авторизации пользователей, добавления клиентов и услуги, стоимость услуги и стоимость дополнительной услуги, а также удаление добавленных элементов. Но кроме этого программа должна производить расчеты стоимости услуг по каждому клиенту.

Для выполнения расчетов применяются следующие математические модели:

Расчет суммы за услуги производится по формуле:

Если выбрана основная услуга и дополнительная услуга,

$$\text{Price} = \text{Price1} + \text{Price2}. \quad (1)$$

Если выбрана только одна основная услуга,

$$\text{Price} = \text{Price1}, \quad (2)$$

где Price — сумма общая по всем услугам, которые выбрал клиент;

Price1 — стоимость основной услуги,

Price2 — стоимость дополнительной услуги.

Расчет суммы процента за выполненные работы мастеру по формуле

$$\text{Price1} = \text{Price1} - (\text{Price1} * \text{Procent}), \quad (3)$$

где Price1 – Сумма за основную услугу,

Procent – процент скидки.

Расчет суммы процента за выполненные работы мастеру по формуле:

$$\text{Total} = (\text{Summa1} * \text{Procent1}) + (\text{Summa2} * \text{Procent2}), \quad (4)$$

где Total – Итого начислена заработная плата мастерам,

Summa1 – Сумма за выполненные услуги мастера за выбранный период,

Summa2 - Сумма за выполненные дополнительные услуги мастера за выбранный период,

Procent1 – процент за основные услуги,

Procent2 – процент за дополнительные услуги.

2.2 Разработка структуры базы данных

В результате анализа в предыдущей главе этого анализа принято решение о выделении следующих сущностей:

Таблица 2 – Таблица - характеристики сущностей

Номер п/п	Наименование сущности	Характеристика
1	Вход(регистрация)	сущность содержит информацию о логине и пароле пользователя;
2	Услуга	сущность содержит информацию о виде услуги
3	Дополнительно	сущность содержит информацию о дополнительных средствах по уходу либо услугах
4	Расписание	сущность содержит информацию о времени, о дате, о мастере и услуги для записи
5	Мастер	сущность содержит информацию о имени мастера выполняющего услугу;
6	Ведомость заработной платы	сущность содержит информацию о начисленной заработной плате за выбранный период
7	Отчет/поиск	Сущность содержит сгруппированную информацию по выбранным параметрам

Таблица 2.1 – Таблица справочники

Номер п/п	Наименование справочника	Характеристика
1	Услуга	сущность содержит информацию об вариантах услуг и их стоимости;
2	Справочник работы мастеров	Сущность содержит информацию о мастерах, их имени, дате и времени работы
3	Справочник скидки/проценты мастерам	Сущность содержит о действующих скидках и проценты для расчета заработной платы мастерам

Исходя из приведенных выше сущностей, построена диаграмма сущностей, которая представлена на рисунке 6.

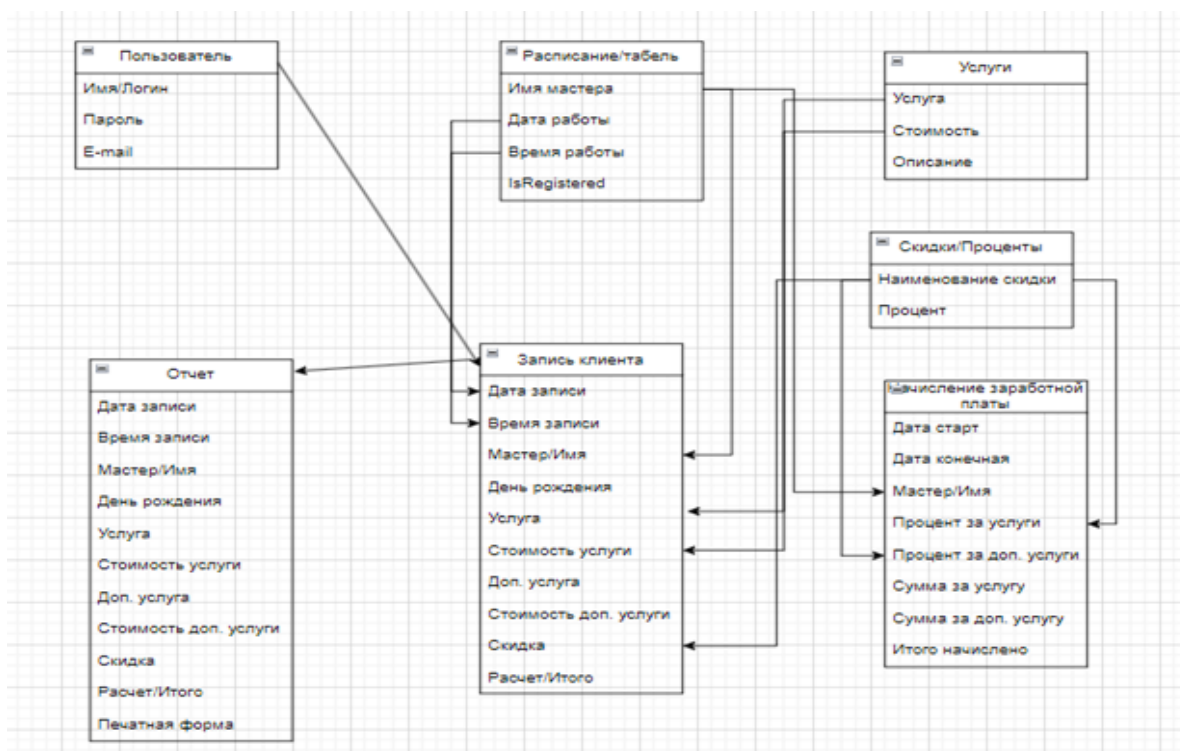


Рисунок 6 – Диаграмма сущностей

3 Разработка проекта программного обеспечения

3.1 Функциональный анализ предметной области

Анализ предметной области позволяет выделить ее сущности, определить первоначальные требования к функциональности и определить границы проекта. Модель предметной области должна быть документирована, храниться и поддерживаться в актуальном состоянии до этапа реализации.

Для визуализации связей между элементами системы разработаем трассировочную матрицу. По определению матрица трассируемой — двумерная таблица, содержащая соответствие функциональных требований продукта и подготовленных тестовых сценариев. Трассировочная матрица проекта представлена в таблице 2.

Таблица 3 – Трассировочная матрица

№	Пожелания заказчика	№ функции	Функциональное требование	Use cases
1.	Программа должна обеспечивать администратору работу с учетными записями.	1.1	Вход в систему	Вход в систему через логин и пароль
2.	Программа должна обеспечивать владельцу работу со справочником услуг	2.1	Добавление новой услуги	Отображение справочника услуг и его редактирования
3.	Программа должна обеспечивать владельцу работу с расписание мастеров	3.1 3.2 3.3	Добавление мастера Добавление даты Добавление времени	Отображение расписания мастеров и его редактирования

Продолжение таблицы 3 – Трассировочная матрица

№	Пожелания заказчика	№ функции	Функциональное требование	Use cases
4.	Программа должна обеспечивать владельцу работу со справочником скидки/проценты	4.1 4.2	Ввод новой скидки Ввод размера скидки	Просмотр справочника проценты/скидки и его редактирования
5.	Программа должна обеспечивать владельцу работу с журналом записи клиентов	5.0 5.1 5.2 5.3 5.4 5.5 5.6 5.7	Отображение журнала Выбор даты в списке Выбор времени в выбранной дате Выбор услуги Выбор мастера Ввод стоимости Выбор дополнительных услуг	Отображение журнала записи клиентов и его редактирования
6.	Программа должна выводить отчет по работе мастера	6.1	Выбор мастера и его работу согласно записи	Формирование отчета по параметрам и вывод в табличную форму Excel
7.	Программа должна быть простой и интуитивно понятной пользователю		Абстрактное пожелание	

Продолжение таблицы 3 –Трассировочная матрица

№	Пожелания заказчика	№ функции	Функциональное требование	Use cases
8.	Программа должна обеспечивать администратору работу с журналом начисления заработной платы	8.1 8.2 8.3	Заполнение ведомости Расчет заработной платы Запись в журнал	Отображение общего журнала зарплата и его редактирования

Функциональные требования хорошо иллюстрирует диаграмма вариантов использования. Модели вариантов использования изображены на рисунках 6-7



Рисунок 7 – Диаграмма вариантов использования программы администратором

3.2 Разработка схемы ресурсов системы

На чертеже ГУИР 2033113.001 представлена схема ресурсов разрабатываемого программного средства.

Схема ресурсов была построена на основании следующих принципов обработки данных:

- система представляет собой приложение по шаблону WPF;
- все пользовательские данные хранятся в таблицах базы данных;
- приложение использует Entity Framework;
- работа приложений .NET обеспечивается за счет наличия такого механизма, как общезыковая среда исполнения (Common Language Runtime, CLR), функции ОС и микропрограммы процессора.

В качестве базы данных для хранения информации выбран Microsoft SQL Server Express Edition (64-bit) 2019, параметры сортировки - SQL_Latin1_General_CP1_CI_AS, номер версии продукта - 15.0.4153 RTM. SQL Server является основой всесторонней платформы данных Microsoft, демонстрирует высочайшую производительность критически важных приложений благодаря встроенным технологиям обработки в оперативной памяти OLTP[4].

3.3 Разработка схемы базы данных

Схема БД должна включать набор всех схем ее таблиц, т.е. описание всех колонок этих таблиц (их типов, допустимых значений, связей между таблицами, типа внешних ключей и т.д.), без учета конкретных данных. При составлении схемы БД разрабатываемой программы создано необходимое количество таблиц для хранения нужной информации и представлена в следующей схеме[5].

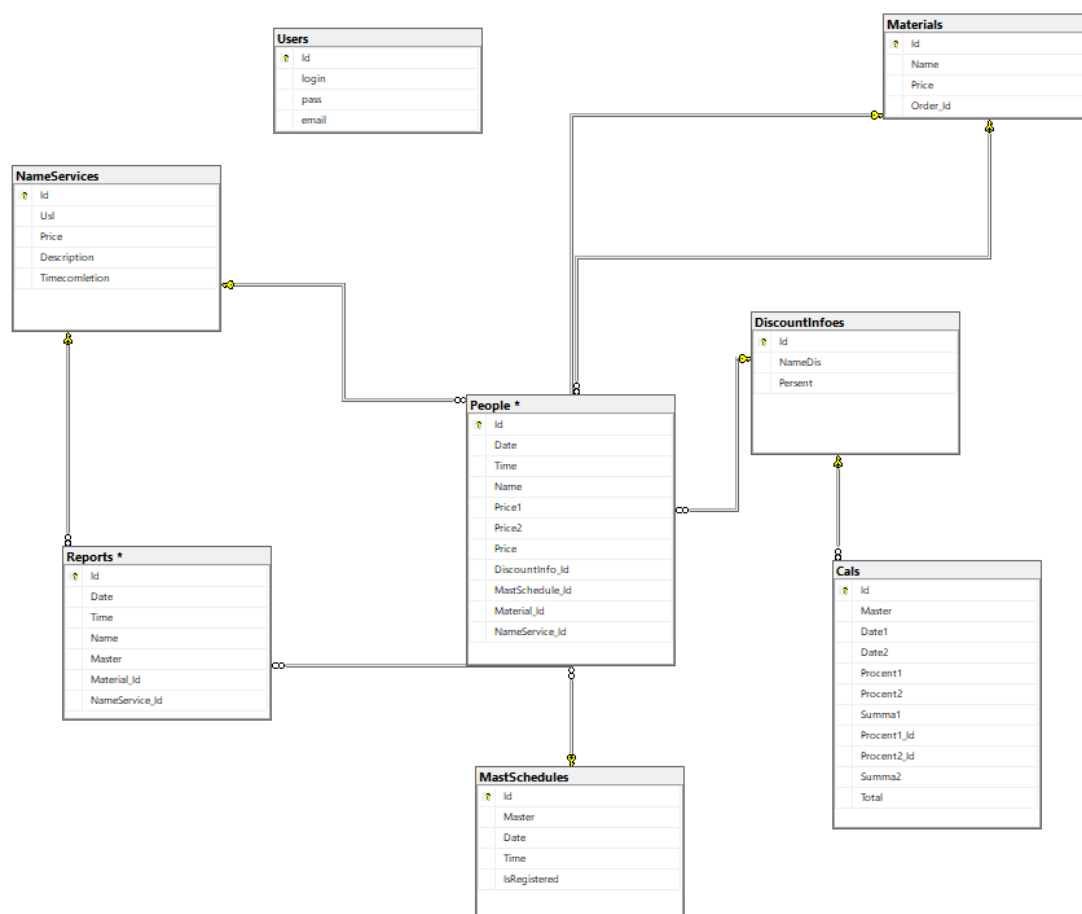


Рисунок 8 – Схема базы данных

Данные таблицы соответствуют одноименным сущностям. Значения атрибутов сущностей представлены в следующих таблицах.

Сущность «Пользователь» описана в таблице и предназначена для хранения информации о клиенте. Описание атрибутов представлено в таблице 4.

Таблица 4 – Таблица пользователи

Атрибут	Описание
[Id] [int] IDENTITY(1,1) NOT NULL , [login] [nvarchar](max) NULL , [pass] [nvarchar](max) NULL , [email] [nvarchar](max) NULL	
	Логин
	Пароль
	E-mail

Сущность «Person/People» описана в таблице и предназначена для хранения информации о записи клиента, времени, дате и мастере. Описание атрибутов представлено в таблице 5.

Таблица 5 – Таблица запись клиентов

Атрибут	Описание
[Id] [int] IDENTITY(1,1) NOT NULL [Date] [nvarchar](max) NULL [Time] [nvarchar](max) NULL [Name] [nvarchar](max) NULL [Price1] [nvarchar](max) NULL [Price2] [nvarchar](max) NULL [Price] [nvarchar](max) NULL [DiscountInfo_Id] [int] NULL [MastSchedule_Id] [int] NULL [Material_Id] [int] NULL [NameService_Id] [int] NULL	
	Дата визита
	Время визита
	Имя клиента
	Стоимость основной услуги
	Стоимость дополнительной
	Общая стоимость
	Скидка при
	Мастер
	Доп услуга
	Услуга

Сущность «Услуга» описана в таблице и предназначена для хранения информации о наличии видов услуг и описание. Описание атрибутов представлено в таблице 5.

Таблица 6 – Таблица -Услуга

Атрибут	Описание
[Id] [int] IDENTITY(1,1) NOT NULL, [Usl] [nvarchar](max) NULL, [Price] [decimal](18, 2) NOT NULL, [Description] [nvarchar](max) NULL,	
	Наименование услуги
	Стоимость услуги
	Описание
	Время исполнения

Сущность «Расписание» описана в таблице и предназначена для хранения информации о свободном времени для записи клиента. Описание атрибутов представлено в таблице 6.

Таблица 7 – Таблица - Расписание

Атрибут	Описание
[Id] [int] IDENTITY(1,1) NOT NULL, [Master] [nvarchar](max) NULL, [Date] [nvarchar](max) NULL, [Time] [nvarchar](max) NULL, [IsRegistered] [bit] NOT NULL	
	Мастер
	Дата работы
	Время работы

Сущность «Скидки/проценты» описана в таблице и предназначена для хранения информации о скидках на услуги и дополнительных услуг, а также проценты для мастеров за выполненную работу. Описание атрибутов представлено в таблице 7.

Таблица 7 – Таблица – Скидки/проценты

Атрибут	Описание
[Id] [int] IDENTITY(1,1) NOT NULL , [NameDis] [nvarchar](max) NULL , [Persent] [decimal](18, 2) NOT NULL	
	Наименование скидки
	Размер скидки в процентах

Сущность «Материалы/Дополнительные услуги» описана в таблице и предназначена для хранения информации об остатках товаров по пунктам хранения. Описание атрибутов представлено в таблице 8.

Таблица 8 – Таблица - Материалы

Атрибут	Описание
Id] [int] IDENTITY(1,1) NOT NULL , [Name] [nvarchar](max) NULL , [Price] [nvarchar](max) NULL ,	
	Наименование доп. услуги
	Стоимость услуги

Сущность «Сервис» описана в таблице и предназначен для хранения списка записи клиентов по датам и видам услуг. Описание атрибутов представлено в таблице 9.

Таблица 9 – Таблица - Сервис

Атрибут	Описание
[Id] [int] IDENTITY(1,1) NOT NULL , [Date] [nvarchar](max) NULL , [Time] [nvarchar](max) NULL , [Name] [nvarchar](max) NULL , [Master] [nvarchar](max) NULL , [Material_Id] [int] NULL ,	
	Дата записи
	Время клиента
	Имя клиента
	Имя мастера
	Доп услуга
	Наименование услуги

Сущность «Начисление заработной платы» предназначен для хранения начислений заработной платы мастерам. Описание атрибутов представлено в таблице 10.

Таблица – 10 – Таблица – Начисление заработной платы

Атрибут	Описание
[Id] [int] IDENTITY(1,1) NOT NULL , [Master] [nvarchar](max) NULL , [Date1] [nvarchar](max) NULL , [Date2] [nvarchar](max) NULL , [Summa1] [nvarchar](max) NULL , [Summa2] [nvarchar](max) NULL , [Total] [nvarchar](max) NULL , [Procent1_Id] [int] NULL , [Procent2_Id] [int] NULL	
	Мастер
	Дата начала периода
	Дата конец периода
	Сумма за услуги
	Сумма за доп услуги
	Итого начислено
	Процент за услуги
	Процент за доп услуги

3.4 Разработка диаграммы классов

В соответствии с диаграммой сущностей предметной области были спроектированы следующие классы, представленные на диаграмме классов (рисунок 11).

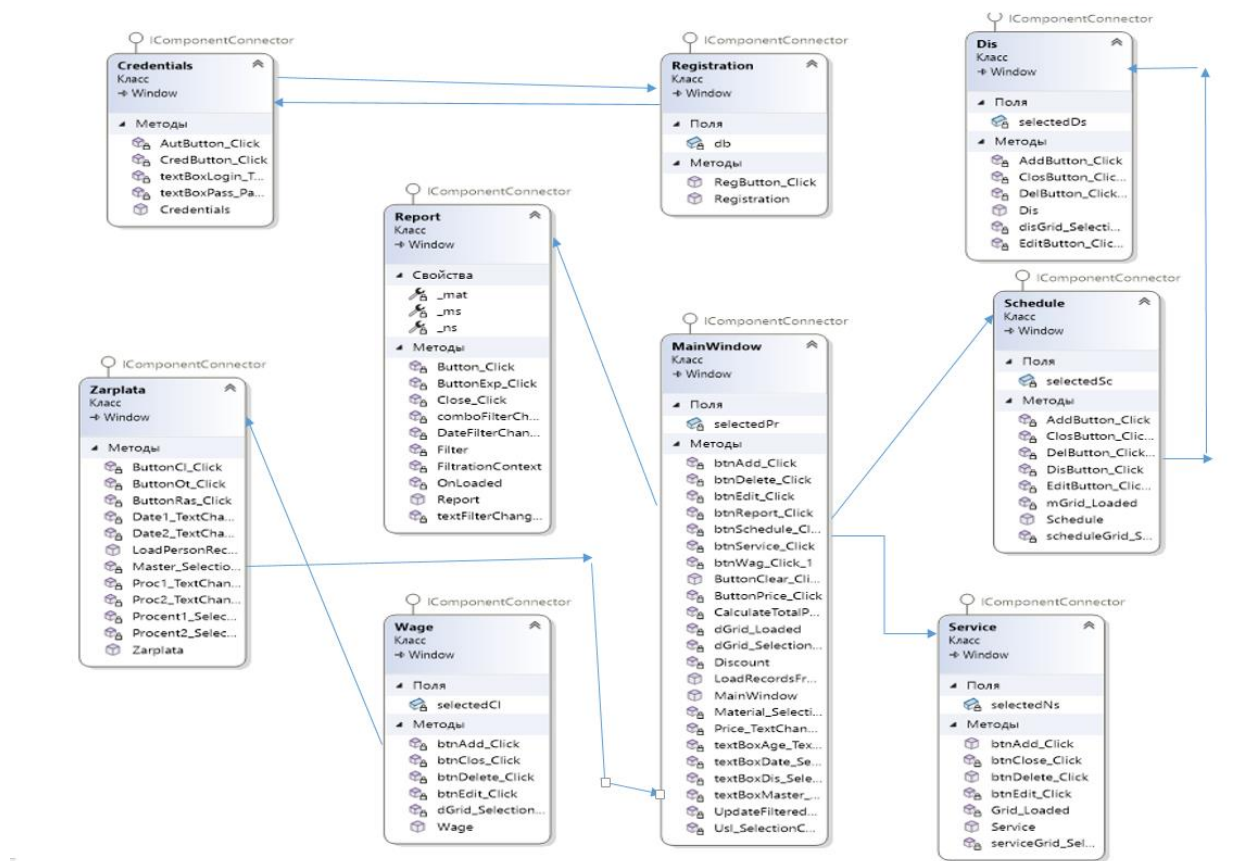


Рисунок 11 – диаграмма классов

3.5 Разработка графического интерфейса пользователя

Исходя из диаграммы вариантов использования и этапа анализа требований был спроектирован простой и интуитивно понятный графический пользовательский интерфейс. Макеты разрабатываемых окон и описание полей приведены на рисунках 12 – 21.

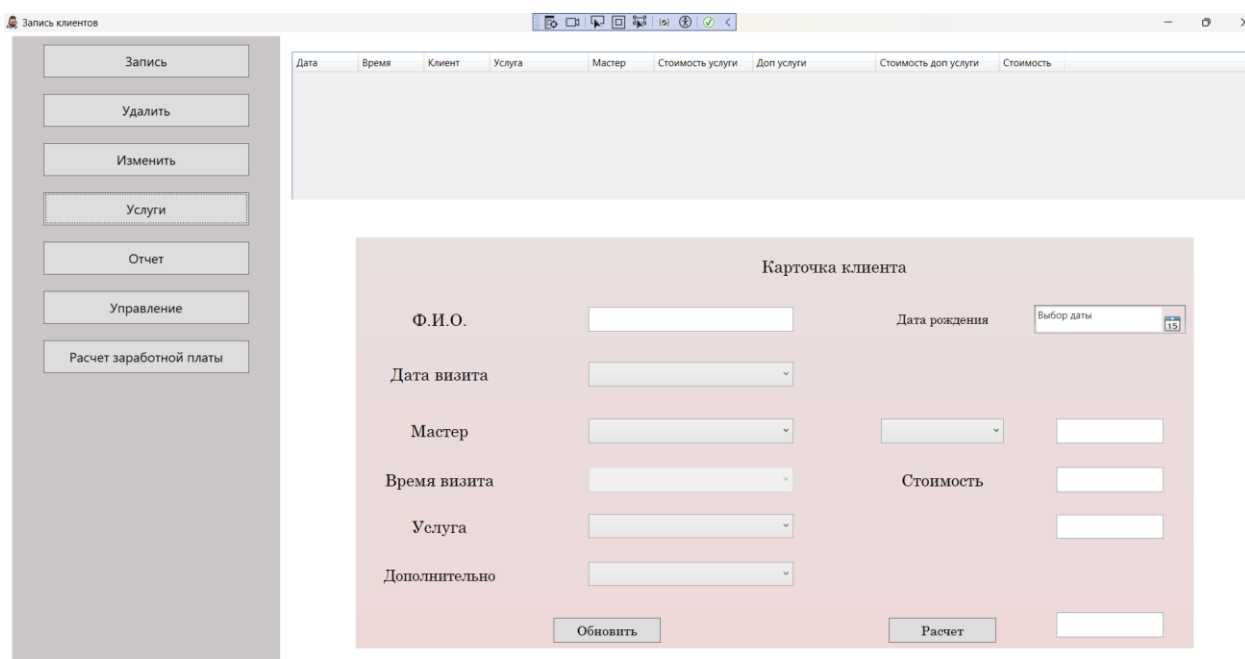


Рисунок 12 – Макет главного окна приложения записи клиентов

1. кнопка «Запись» для начала работы для записи клиента
2. кнопка «Удалить» для удаления записи
3. кнопка «Изменить» для изменения ранее внесенных данных
4. кнопка «Услуги» вызов таблицы для внесения услуг и их стоимость
5. кнопка «Отчет» для поиска и просмотра записей по определенным параметрам
6. кнопка «Управление» вызов расписания/табеля мастеров и справочника «Скидки/Проценты»
7. кнопка «Расчет заработной платы» вызов журнала учета начисления заработной платы и ведомости для расчета

Макет окна для записи новых услуг и редактирования существующих записей на рисунке 13.

Наименование скидки	Процент
Скидка на день рождения	20.0%
Скидка персональная	10.0%
Процент мастеру	40.0%
Процент мастеру за доп у	10.0%

Наименование скидки/процента

Процент скидки Формат указания скидки: 0,2 это 20%

Рисунок 13

1. текстовое поле для ввода наименование услуги;
2. текстовое поле для ввода стоимости услуги;
3. текстовое поле для ввода описания услуги;
4. текстовое поле для ввода информации время исполнения;
5. кнопка «Добавить» добавляет новую услугу;
6. кнопка «Удалить» удаляет выбранную услугу/строчку;
7. кнопка «Изменить» изменяет выбранную услугу;
8. кнопка «Заккрыть» закрывает окно;

Макет окна для записи новых наименований скидок и процентов и редактирования существующих записей на рисунке 14.

Справочник услуг

Добавить

Удалить

Изменить

Закрыть

Наименование услуги	Стоимость	Описание	Время исполнения
Бритье	35.00	Бритва стандартная	35 минут
Стрижка	35.00	Бритва стандартная	35 минут
Королевское бритье	35.00	Бритва стандартная	35 минут

Наименование услуги

Стоимость

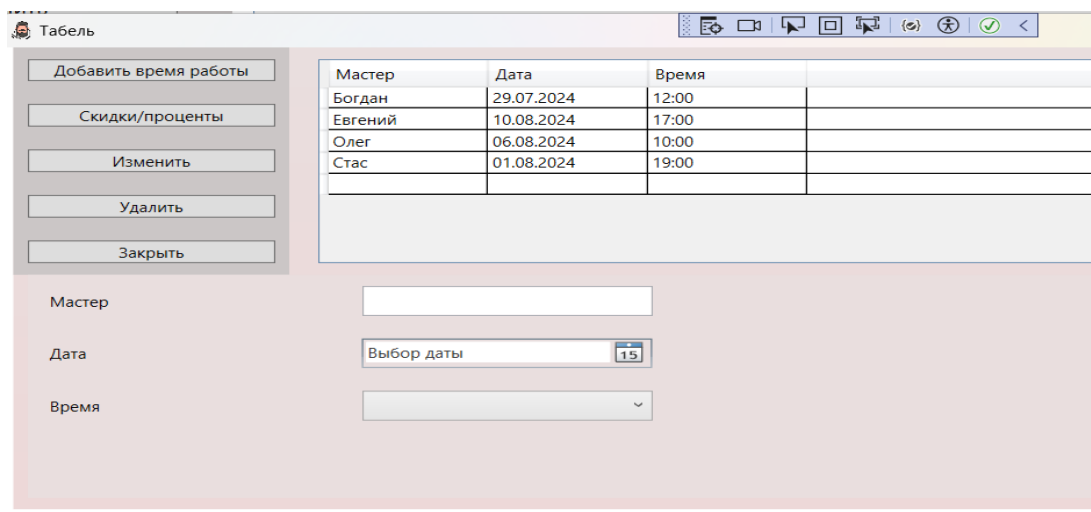
Описание

Время исполнения

Рисунок 14


1. текстовое поле для ввода наименование скидки/процента;
2. текстовое поле для ввода размера скидки/процента;
3. кнопка «Добавить» добавляет новую услугу;
4. кнопка «Удалить» удаляет выбранную услугу/строчку;
5. кнопка «Изменить» изменяет выбранную услугу;
6. кнопка «Закрыть» закрывает окно;

Макет окна, предназначенного для отображения списка рабочего времени мастеров, представлен на рисунке 15.



Мастер	Дата	Время
Богдан	29.07.2024	12:00
Евгений	10.08.2024	17:00
Олег	06.08.2024	10:00
Стас	01.08.2024	19:00

Мастер

Дата 

Время

Рисунок – 15

1. Строка для ввода имени мастера;
2. Строка для ввода даты работы мастера;
3. Строка для ввода времени работы мастера;
4. Кнопка «Добавить время работы» добавление новой строчки с расписанием мастера;
5. Кнопка «Скидки/проценты» вызов окна – справочник процентов и скидок;
6. Кнопка «Изменить» для изменения выбранной строки;
7. Кнопка «Заккрыть» для закрытия таблицы после работы с ней;

Макет окна для отображения отчета по выбранным параметрам представлен на рисунке 16.

The screenshot shows a software window titled "Отчет по записям". The left sidebar contains the following elements from top to bottom:

- Label: "Начальная дата:" followed by a date picker labeled "Выбор даты" with a calendar icon.
- Label: "Конечная дата:" followed by a date picker labeled "Выбор даты" with a calendar icon.
- Label: "Дата:" followed by a text input field.
- Label: "Время:" followed by a text input field.
- Label: "Клиент:" followed by a text input field.
- Label: "Услуга:" followed by a dropdown menu.
- Label: "Мастер:" followed by a dropdown menu.
- Label: "Материал:" followed by a dropdown menu.
- Button: "Заккрыть" (Note the typo).
- Button: "Печать".

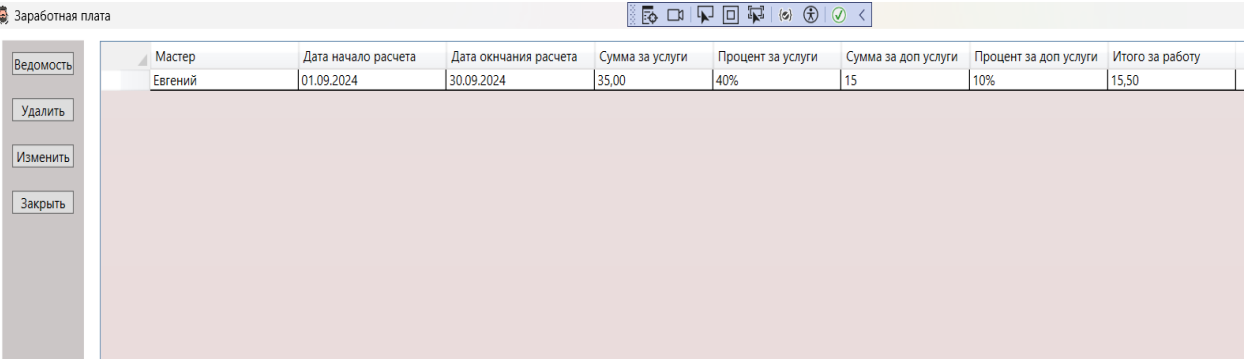
The main area of the window displays a table with the following headers: "Дата", "Время", "Клиент", "Услуга", "Мастер", "Стоимость услуги", "Доп услуги", "Стоимость доп услуги", and "Стоимость". The table body is currently empty.

Рисунок 16

Поля для определения параметров:

1. Поле для выбора даты начальной;
2. Поле для выбора даты конечной;
3. Текстовое поле для ввода дата;
4. Поле для ввода времени;
5. Поле для ввода для клиента;
6. Поле с выбором наименования услуги;
7. Поле для ввода мастера;
8. Поле с выбором дополнительных материалов;
9. Окна для отображения результатов.
10. Кнопка «Заккрыть» для закрытия окна после работы с ним;
11. Кнопка «Печать» для вывода на печать в форму Excel;

Макет окна для отображения начисления по заработной плате мастерам за период представлен на рисунке 17

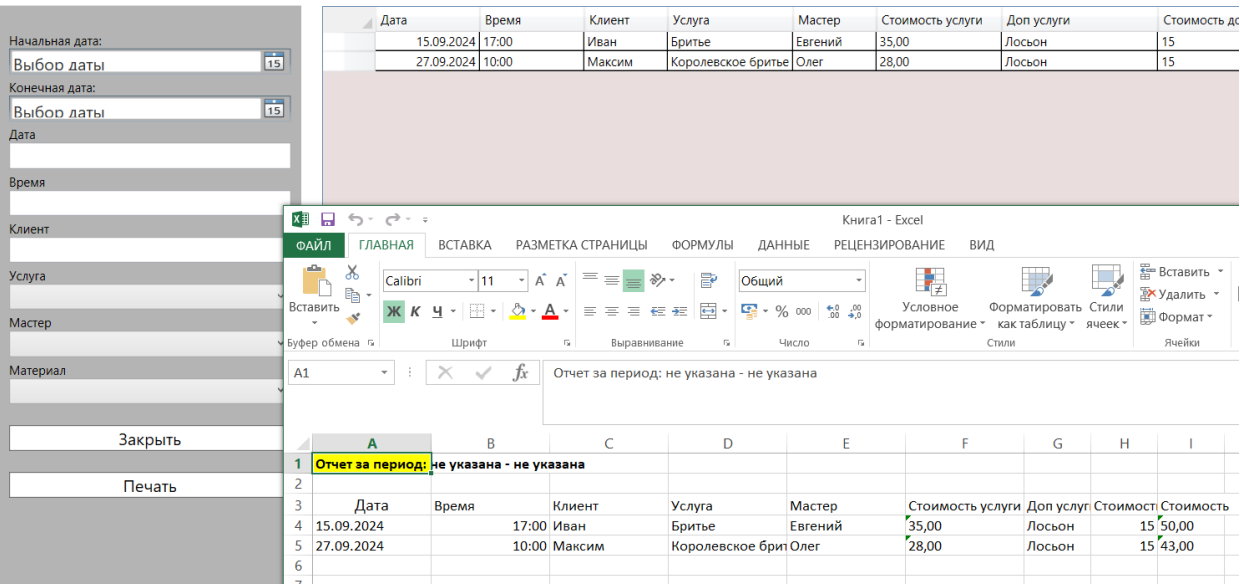


Мастер	Дата начало расчета	Дата окончания расчета	Сумма за услуги	Процент за услуги	Сумма за доп услуги	Процент за доп услуги	Итого за работу
Евгений	01.09.2024	30.09.2024	35,00	40%	15	10%	15,50

Рисунок 17

Поля для определения параметров:

- 1. Кнопка для вызова окна «Ведомость начисления»;
- 2. Кнопка «Удалить» для удаления строчки;
- 3. Кнопка «Изменить» для изменения данных в строчке;
- 4. Кнопка «Заккрыть» для закрытия окна после работы с ним;



Дата	Время	Клиент	Услуга	Мастер	Стоимость услуги	Доп услуги	Стоимость доп
15.09.2024	17:00	Иван	Бритье	Евгений	35,00	Лосьон	15
27.09.2024	10:00	Максим	Королевское бритье	Олег	28,00	Лосьон	15

Рисунок 18

4 Разработка алгоритмов и их описание

4.1 Алгоритм получение списка свободного времени

Проверка условий:

Метод проверяет, выбраны ли значения в дата и мастер. Если одно из них не выбрано, время отключается.

Создание контекста базы данных:

Используется подключение для создания контекста базы данных, что гарантирует его корректное закрытие после использования.

Загрузка записей из базы данных:

Асинхронно загружаются записи из базы данных

Фильтрация записей:

Фильтруются записи, у которых дата больше или равна выбранной дате, и мастер совпадает с выбранным значением в поле мастер.

Обновление источника данных:

Из отфильтрованных записей извлекаются все времена и устанавливаются в качестве источника данных для поля время.

Время для выбора включается.

Данный алгоритм представлен на чертеже ГУИР 2033113.002.

4.2 Алгоритм получение отчета по услуге

Проверка на null:

Если объект таблицы равен null, метод возвращает сообщение об ошибке.

Фильтрация по текстовым полям:

Проверяет, содержат ли поле наименование услуги текст, и если да, то сравнивает их с соответствующими полями объекта таблицы,

Если текст не совпадает, метод возвращает ложь.

Фильтрация по выбранным индексам:

Проверяет, выбраны ли элементы в поле наименование услуги, и если да, то сравнивает их с соответствующими полями объекта таблицы.

Если значения не совпадают, метод возвращает ложь.

Возвращение результата:

Если все проверки пройдены, метод возвращает истину. В таблице фильтруются данные по наименованию услуги.

Данный алгоритм представлен на чертеже ГУИР 2033113.003.

4.3 Алгоритм получение списка свободных дат

Обновление фильтрованных записей:

Вызывает метод (подпроцесс – 4.1), который обновляет записи на основе выбранных значений в поле дата и в поле мастер.

Проверка выбранного элемента:

Проверяет, выбран ли элемент в поле мастер. Если нет, метод завершает выполнение.

Создание контекста базы данных:

Создает новый контекст базы данных с использованием подключения, что гарантирует его корректное закрытие после использования.

Загрузка записей из базы данных:

Асинхронно загружаются записи из базы данных

Фильтрация записей по выбранному мастеру:

Извлекает выбранного мастера из поля мастер.

Фильтрует записи, чтобы оставить только те, где мастер совпадает с выбранным значением и запись зарегистрирована (IsRegistered == true).

Получение уникальных дат:

Из отфильтрованных записей извлекает уникальные даты работы мастера.

Обновление источника данных:

Устанавливает уникальные даты в качестве источника данных для поля дата. Данный алгоритм представлен на чертеже ГУИР 2033113.004.

4.4 Алгоритм получения отчета по мастеру

Проверка на null:

Если объект таблицы равен null, метод возвращает сообщение об ошибке.

Фильтрация по текстовым полям:

Проверяет, содержат ли поле имя мастера текст, и если да, то сравнивает их с соответствующими полями объекта таблицы,

Если текст не совпадает, метод возвращает ложь.

Фильтрация по выбранным индексам:

Проверяет, выбраны ли элементы в поле мастер, и если да, то сравнивает их с соответствующими полями объекта таблицы.

Если значения не совпадают, метод возвращает ложь.

Возвращение результата:

Если все проверки пройдены, метод возвращает истину. В таблице фильтруются данные по имени мастера

Данный алгоритм представлен на рисунке 17.

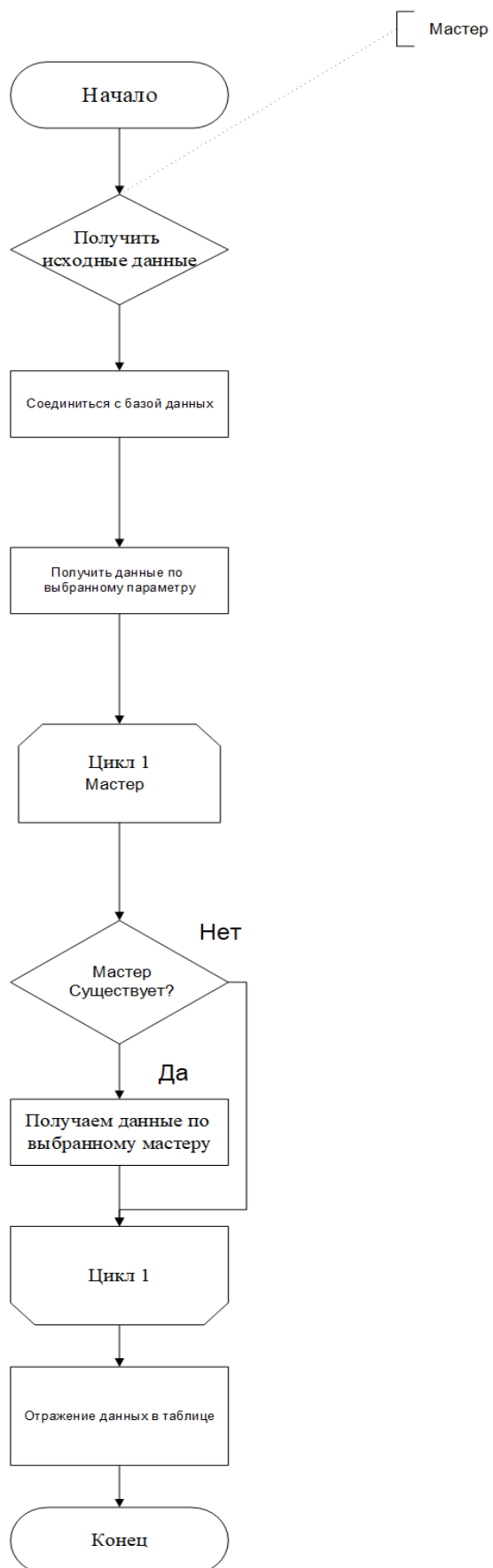


Рисунок 17 - схемы программы

4.5 Алгоритм получение начисленной суммы по мастеру

Проверка выбранного элемента:

Проверяет, выбран ли элемент в поле мастер. Если нет, метод завершает выполнение.

Загрузка записей из базы данных:

Асинхронно загружает записи из базы данных

Получение диапазонов дат из журнала заработная плата:

Загружает записи из журнала и проверяет, пересекаются ли диапазоны дат с выбранными датами. Если пересечение найдено и мастер совпадает, выводится сообщение и метод завершает выполнение.

Фильтрация записей по мастеру и датам:

Фильтрует записи, чтобы оставить только те, где дата находится в выбранном диапазоне, и мастер совпадает с выбранным значением.

Подсчет общей стоимости:

Если отфильтрованные записи найдены, подсчитывает общую стоимость за основные услуги и устанавливает её в поле сумма за услуги, а также подсчитывает стоимость за дополнительные услуги и устанавливает в поле сумма за дополнительные услуги. Если записей нет, устанавливает значение “0” и выводит сообщение.

Данный алгоритм представлен на чертеже ГУИР 2033113.005.

5 Тестирование полученного программного продукта

Тестирование программного обеспечения является очень важным и в то же время трудоемким видом деятельности. Тестирование – это прежде всего оценка промежуточных продуктов, созданных в процессе разработки программного обеспечения. Это гораздо более широкое поле деятельности, нежели просто проверка некоторых частей или программной системы в целом с целью определения степени ее соответствия техническим требованиям.

В данной работе была выбрана методология функционального тестирования.

Функциональное тестирование – это тестирование программного обеспечения в целях проверки реализуемости функциональных требований, то есть способности программного обеспечения в определенных условиях решать задачи, нужные пользователям. Функциональные требования определяют, что именно делает программное обеспечение, какие задачи оно решает.

Функциональное тестирование предполагает составление плана тестирования, который строится на основе вариантов использования и включает в себя описание тестов и их результаты. Для проверки соответствия программного продукта предъявленным требованиям были протестированы следующие модули:

- тестирование запуска приложения;
- тестирование модуля Авторизация и регистрация;
- тестирование модуля Журнал записи клиентов;
- тестирование модуля Просмотр справочника услуги;
- тестирование модуля Просмотр расписания мастеров;
- тестирование модуля Просмотр справочника скидки/проценты;
- тестирование модуля Оформление записи;
- тестирование модуля Составление отчета/поиска;
- тестирование модуля Просмотр журнала заработная плата;
- тестирование модуля Заполнение ведомости начисления заработной платы;

Таблица 13 – Функциональные тест-кейсы программного продукта

Дата тестирования: 01.08.2024	Тестировала: Матюшенко Н.П..	ОС: Windows 11	Приложение: Программное обеспечение записи клиентов в мужской салон стрижки и бритья	
Идентификатор	Модуль	Описание теста	Ожидаемый результат	Статус («выполнено успешно», «выполнено с ошибкой»)
T01	Приложение не запущено	1. Запустить в папке с приложением файл Proekt_Barber.exe	1. Отображается главная страница приложения	Выполнено успешно
T02	Авторизация и регистрация в системе	1. Нажать кнопку «Зарегистрироваться»;	1. Отображается страница регистрации;	Выполнено успешно
		2. Ввести значение «Логин»	2. В поле «Логин» отображается введенный Логин;	
		3. Ввести значение «Пароль»;	3. В поле «Пароль» отображается скрытый введенный пароль;	
		4. Ввести значение «Подтверждение пароля»;	4. В поле «Подтверждение пароля» отображается скрытый введенный подтвержденный пароль;	
		5. Ввести значение «E-mail»;	5. В поле «E-mail» отображается введенный E-mail;	
		6. Нажать кнопку «Зарегистрироваться»;	7. Отображается главная страница, в шапке появились кнопки «Зарегистрироваться» и «Войти»;	
		Авторизация	7. Отображается окно входа в систему;	
		7. Нажать кнопку «Войти»;	8. В поле «Логин» отображается введенный Логин;	
		8. Ввести значение «Пароль»;	9. В поле «Пароль» отображается скрытый введенный пароль;	
		9. Нажать кнопку «Войти»;	10. Отображается главная страница	

Продолжение таблицы 13 - Функциональные тест-кейсы
программного продукта

Идентификатор	Модуль	Описание теста	Ожидаемый результат	Статус («выполнено успешно», «выполнено с ошибкой»)
T03	Главное окно приложения/ Журнал записи клиентов	1. Провести визуальный осмотр содержимого окна приложения;	1. Все элементы окна отображаются корректно, грамматические ошибки отсутствуют;	Выполнено успешно
		2. Изменить размер окна приложения;	2. Элементы окна отображаются корректно;	
		3. Нажать кнопку «Услуги»;	3. Отображение каталога автозапчастей;	
		4. Нажать кнопку «Отчет»;	4. Отображение услуг и цен;	
		5. Нажать кнопку «Управление»;	5. Отображение контактов;	
		6. Нажать кнопку «Расчет заработной платы»	6. Отображается ведомость начислений	
T04	Просмотр справочника услуги	1. В главном окне нажать кнопку «Услуги»;	1. Отображается окно «Справочник услуг»;	Выполнено успешно
		2. Внести в тестовые поля «Наименование услуги»	2. Отображаются наименование услуги;	
		3. Внести в тестовые поля «Стоимость услуги»	3. Отображаются стоимость услуги;	
		4. Внести в тестовые поля «Описание услуги»	4. Отображаются описание услуги;	
		5. Внести в тестовые поля «Время исполнения»	5. Отображается время исполнения;	
		6. Нажать кнопку «Добавить»	6. Отображается запись о новой услуге;	
		7. Выделить запись в таблице	7. Подсвечивается строка;	
		8. Внести изменения в текстовые поля	8. Отображаются данные в строке;	
		9. Нажать на кнопку «Изменить»	9. Вносятся изменения;	
		10. Выделить запись в таблице	10. Подсвечивается строка;	
		11. Нажать кнопку «Удалить»	11. Удаляется запись;	
		12. Нажать кнопку «Закрыть»	12. Окно закрывается и отображается главное окно.	

Продолжение таблицы 13 - Функциональные тест-кейсы
программного продукта

Идентификатор	Модуль	Описание теста	Ожидаемый результат	Статус («выполнено успешно», «выполнено с ошибкой»)
T05	Просмотр расписания мастеров	1. В главном окне нажать кнопку «Управление»;	1.Отображается окно «Расписание мастеров»;	Выполнено успешно
		2. Внести в тестовые поля «Имя мастера»	2. Отображаются имя мастера;	
		3. Внести в тестовые поля «Дата работы»	3. Отображаются дата работы;	
		4. Внести в тестовые поля «Время работы»	4.Отображаются время работы;	
		5. Нажать кнопку «Добавить»	7.Отображается запись о свободном времени для записи;	
		6. Выделить запись в таблице	7. Подсвечивается строка;	
		7. Внести изменения в текстовые поля	8. Отображаются данные строки в текстовом поле для редактирования;	
		8. Нажать на кнопку «Изменить»	9. Вносятся изменения;	
		9. Выделить запись в таблице	10. Подсвечивается строка;	
		10. Нажать кнопку «Удалить»	11.Удаляется запись;	
		11. Нажать кнопку «Закрыть»	12. Окно закрывается и отображается главное окно.	
T06	Просмотр справочника скидки/проценты	1. В главном окне нажать кнопку «Управление», нажать кнопку «Скидки/проценты»;	1.Отображается окно «Справочник скидки/проценты»;	Выполнено успешно
		2. Внести в тестовые поля «Наименование скидки/процента»	2. Отображаются наименование скидки/процента;	
		3. Внести в тестовые поля «Размер скидки/процента»	3. Отображаются размер скидки;	
		4. Нажать кнопку «Добавить»	4.Отображается запись о свободном времени для записи;	
		5. Выделить запись в таблице	5. Подсвечивается строка;	
		6. Внести изменения в текстовые поля	6. Отображаются данные строки в текстовом поле для редактирования;	
		7. Нажать на кнопку «Изменить»	7. Вносятся изменения;	

Продолжение таблицы 13 - Функциональные тест-кейсы
программного продукта

Идентификатор	Модуль	Описание теста	Ожидаемый результат	Статус («выполнено успешно», «выполнено с ошибкой»)
		8. Ввести значение по необходимости «Скидка»;	8. В поле «Скидка» отображается введенная скидка;	
		9. Нажать кнопку «Расчет»;	9. Отображается страница окончательный расчет по выбранным услугам	
		10. Нажать кнопку «Записать».	10. Отображается строчка записи в общем журнале записи клиентов.	
T07	Оформление записи	1. На странице главного окна заполнить форму	1.Отображается страница «Оформление заказа»;	Выполнено успешно
		2. Ввести значение «Имя»;	2. В поле «Имя»;	
		3. Ввести значение «Дата записи»;	3. В поле «Дата записи» отображается введенное дата записи;	
		4. Ввести значение «Мастер»;	4. В поле «Мастер» отображается введенный мастер;	
		5. Ввести значение «Наименование услуги»;	5. В поле «Наименование услуги» отображается введенная услуга;	
		6. Ввести значение по необходимости «Дополнительные услуги»;	6. В поле «Дополнительная услуга» отображается дополнительная услуга;	
		7.Ввести значение «Дата рождение»;	7. В поле «Дата рождения» отображается введенная дата рождения;	
		8. Ввести значение по необходимости «Скидка»;	8. В поле «Скидка» отображается введенная скидка;	
		9. Нажать кнопку «Расчет»;	9. Отображается страница окончательный расчет по выбранным услугам	
		10. Нажать кнопку «Записать».	10. Отображается строчка записи в общем журнале записи клиентов.	

Продолжение таблицы 13 - Функциональные тест-кейсы
программного продукта

Идентификатор	Модуль	Описание теста	Ожидаемый результат	Статус («выполнено успешно», «выполнено с ошибкой»)
T08	Составление отчета/поиска	1. На странице главного окна нажать кнопку «Отчет»	1. Отображается страница «Отчет/Поиск»;	Выполнено успешно
		2. Ввести значение «Начальная дата»;	2. Отображается записи в диапазоне с «Начальная дата»;	
		3. Ввести значение «Конечная дата»;	3. Отображается записи в диапазоне с «Конечная дата»;	
		4. Ввести значение «Дата»;	4. Отображается строчка(и) с данными по значению «Дата»;	
		5. Ввести значение «Время»;	5. Отображается строчка(и) с данными по значению «Время»;	
		6. Ввести значение «Клиент»;	6. Отображается строчка(и) с данными по значению «Клиент»;	
		7. Ввести значение «Услуга»;	7. Отображается строчка(и) с данными по значению «Услуга»;	
		8. Ввести значение «Мастер»;	8. Отображается строчка(и) с данными по значению «Мастер»;	
		9. Ввести значение «Материал»;	9. Отображается строчка(и) с данными по значению «Материалы»;	
		10. Нажать кнопку «Закрыть»;	10. Закрывается окно «Отчеты/поиск»;	
		11. Нажать кнопку «Печать».	11. Отображается форма Excel.	

Продолжение таблицы 13 - Функциональные тест-кейсы
программного продукта

Идентификатор	Модуль	Описание теста	Ожидаемый результат	Статус («выполнено успешно», «выполнено с ошибкой»)
T09	Просмотр журнала заработная плата	1. В главном окне нажать кнопку «Заработная плата»	1. Отображение окна «Заработная плата»	Выполнено успешно
		2. Провести визуальный осмотр содержимого окна приложения;	2. Все элементы окна отображаются корректно, грамматические ошибки отсутствуют;	
		3. Изменить размер окна приложения;	3. Элементы окна отображаются корректно;	
		4. Нажать кнопку «Ведомость»;	4. Отображается окно «Ведомость начисления заработной платы»;	
		5. Выделить строчку. Нажать кнопку «Удалить»;	5. Выделенная строчка удаляется;	
		6. Нажать кнопку «Закрыть»;	6. Журнал заработная плата закрывается и отображается главное окно.	

Продолжение таблицы 13 - Функциональные тест-кейсы
программного продукта

Идентификатор	Модуль	Описание теста	Ожидаемый результат	Статус («выполнено успешно», «выполнено с ошибкой»)
Т10	Заполнение ведомости начисления заработной платы	1. В главном окне нажать кнопку «Заработная плата»	1. Отображение окна «Заработная плата»	Выполнено успешно
		2. В журнале «Заработная плата» нажать кнопку «Ведомость»	2. Отображается окно «Заработная плата»;	
		3. Ввести значение с «Дата»;	3. Отображается значение с «Дата»;	
		4. Ввести значение по «Дата»;	4. Отображается значение по «Дата»;	
		5. Ввести значение «Мастер»	5. Отображается значение «Мастер»;	
		6. Ввести значение «Процент за услуги»;	6. Отображается значение «Процент за услуги»;	
		7. Ввести значение «Процент за дополнительные услуги»;	7. Отображается значение «Процент за дополнительные услуги»;	
		8. Нажать кнопку «Расчет»;	8. Отображение в поле «Итого начислено»	
		9. Нажать кнопку «Записать».	9. Отображается строка в журнале «Заработная плата».	

Заключение

В данном дипломном проекте было создано программное средство, которое служит для автоматизации процесса записи клиентов в салон красоты. Цель разработки: минимизация временных затрат и повышение точности в учете клиентов.

Программа предоставляет пользователю возможность добавлять новые услуги и их описание. Разработанное программное средство позволяет пользователю отслеживать работу мастеров в определенные дни. Проводить аналитику востребованности услуг и их стоимость.

Приложение реализовано посредством языка программирования C# (среда Visual Studio 2022 Community) с использованием следующего инструментария: Lucidchart, Microsoft SQL Server Management Studio 2018.

В результате выполнения данной работы разработано программное средство, отвечающее поставленным задачам. Реализован следующий функционал:

- Авторизация/регистрация,
- авторизация пользователя,
- добавление и удаление клиента,
- выбор даты и времени,
- выбор услуги и стоимости,
- выбор дополнительных услуг и стоимости,
- формирование отчета по работе мастеров и вывод данных в табличную форму Excel,
- поиск по заданным признакам и вывод данных в табличную форму Excel,
- отслеживание статуса заявки,
- расчет заработной платы мастера за заданный период,

Несмотря на выполнение поставленных задач, в настоящее время намечены работы, которые помогут расширить функциональные возможности программного средства с добавлением новых функций «Барбершоп Чик-Чик».

Список использованных источников

1. 1С: Управление небольшой фирмой для Беларуси [Электронный ресурс]. - Режим доступа: <https://v8.1c.ru/>. – Дата доступа: 10.07.2024.
2. YCLIENS - Сотни задач. Одна экосистема [Электронный ресурс]. - Режим доступа: <https://www.yclients.com/>. – Дата доступа: 10.07.2024.
3. Главная - Microsoft Learn [Электронный ресурс] – Режим доступа: <https://learn.microsoft.com/>. – Дата доступа 30.07.2024.
4. Главная - Stack Overflow [Электронный ресурс]. – Режим доступа: <https://stackoverflow.com/>. – Дата доступа 30.07.2024.
5. Главная - Microsoft Learn [Электронный ресурс]. – Режим доступа: <https://learn.microsoft.com/>. – Дата доступа 30.07.2024.

Приложение А

(обязательное)

Листинг кода

Основной код основного окна

```
public partial class MainWindow : Window
{
    public MainWindow()
    {
        InitializeComponent();
        Person person = new Person();
        dGrid.ItemsSource = App.Db.Persons.ToList();
        dGrid.DataContext = person;
        stackpanelReg = new StackPanel();

        textBoxUsl.ItemsSource = App.Db.NameServices.ToList();
        Material.ItemsSource = App.Db.Materials.ToList();
        textBoxMaster.ItemsSource = App.Db.MastSchedules.Select(m => m.Master).Distinct().ToList(); //comboBox - заполнение данными
        (список мастеров без повторов)
        textBoxTime.ItemsSource = App.Db.MastSchedules.ToList();
        textBoxDis.ItemsSource = App.Db.DisInfo.ToList();
        textBoxDate.ItemsSource = App.Db.MastSchedules.Select(m => m.Date).Distinct().ToList();
    }
    private void dGrid_Loaded(object sender, RoutedEventArgs e)
    {
        SqlConnection sqlConnection = new SqlConnection(ConfigurationManager.ConnectionStrings["DefaultConnection"].ConnectionString);
        sqlConnection.Open();
    }
    Person selectedPr = null;

    private void dGrid_SelectionChanged(object sender, SelectionChangedEventArgs e)
    {
        if (e.RemovedItems.Count > 0) return;

        selectedPr = (Person)dGrid.SelectedItem;
        if (selectedPr == null) return;
        textBoxName.Text = selectedPr.Name;
        textBoxDate.Text = selectedPr.Date;
        textBoxTime.Text = selectedPr?.MastSchedule?.Time ?? "";
        textBoxUsl.Text = selectedPr.NameService.Usl;
        textBoxMaster.Text = selectedPr?.MastSchedule?.Master ?? "";
        Material.Text = selectedPr.Material.Name;
        Price1.Text = Price1.Text.ToString();
        Price2.Text = Price2.Text.ToString();
        Price.Text = Price.Text.ToString();

        App.Db.Persons.AddOrUpdate(selectedPr);
        App.Db.SaveChanges();
        dGrid.ItemsSource = null;
        dGrid.ItemsSource = App.Db.Persons.ToList();
    }

    private void CalculateTotalPrice()
    {
        if (decimal.TryParse(Price1.Text, out decimal pr1) && !string.IsNullOrEmpty(Price2.Text))
        {
            if (decimal.TryParse(Price2.Text, out decimal pr2))
            {
                Price.Text = (pr1 + pr2).ToString();
            }
            else if (decimal.TryParse(Price1.Text, out decimal price1))
            {
                Price.Text = price1.ToString();
            }
        }
    }
}
```

```

//Вызов метода при расчете скидки при вводе даты записи
private void Discount()
{
    DateTime? birthday = textBoxAge.SelectedDate;
    string visitDateStr = textBoxDate.Text; // Получаем строку даты

    if (DateTime.TryParse(visitDateStr, out DateTime visitDate)) // Преобразуем строку в DateTime
    {
        if (birthday.HasValue)
        {
            DateTime birthdayThisYear = new DateTime(visitDate.Year, birthday.Value.Month, birthday.Value.Day);
            DateTime twoDaysBefore = birthdayThisYear.AddDays(-2);
            DateTime twoDaysAfter = birthdayThisYear.AddDays(2);

            if (visitDate >= twoDaysBefore && visitDate <= twoDaysAfter)
            {
                MessageBox.Show("День рождения!");
            }
        }
    }
}

//Метод расчета скидки
private void textBoxDis_SelectionChanged(object sender, SelectionChangedEventArgs e)
{
    if (textBoxDis.SelectedItem is DiscountInfo dis)
    {
        Persent.Text = dis.Persent.ToString("#0%");
    }
}

//Выбор услуги и применении скидки в случае день рождения
private void Usl_SelectionChanged(object sender, SelectionChangedEventArgs e)
{
    if (textBoxUsl.SelectedItem is NameService ns)
    {
        Price1.Text = ns.Price.ToString("#0.00");

        if (textBoxDis.SelectedItem is DiscountInfo dis)
        {
            Price1.Text = (ns.Price - (ns.Price * dis.Persent)).ToString("#0.00");
        }
    }
}

//Вызов метода при расчете скидки при вводе даты рождения
private void textBoxAge_TextChanged(object sender, SelectionChangedEventArgs e)
{
    Discount();
}

private void Price_TextChanged(object sender, TextChangedEventArgs e)
{
    CalculateTotalPrice();
}

private void btnWag_Click_1(object sender, RoutedEventArgs e)
{
    Wage wage = new Wage();
    wage.Show();
}

//Выбор материал из таблицы Material
private void Material_SelectionChanged(object sender, SelectionChangedEventArgs e)
{
    if (Material.SelectedItem is Material m)
    {

```



```

        Price2.Text = m?.Price.ToString();
    }
}
public async Task<List<MastSchedule>> LoadRecordsFromDatabaseAsync(MyContext context)
{
    var records = await context.MastSchedules.ToListAsync();
    return records;
}

private async void UpdateFilteredRecords()
{
    if (textBoxDate.SelectedItem != null && textBoxMaster.SelectedItem != null)
    {
        using (var context = new MyContext())
        {
            var personRecords = await LoadRecordsFromDatabaseAsync(context);
            string dateString = textBoxDate.Text;

            string dateFormat = "dd.MM.yyyy";

            if (DateTime.TryParseExact(dateString, dateFormat, CultureInfo.InvariantCulture,
                DateTimeStyles.None, out DateTime selectedDate))
            {
                var filteredRecords = personRecords
                    .Where(p => DateTime.Parse(p.Date) >= selectedDate &&
                        p.Master.Equals(textBoxMaster.SelectedItem.ToString(), StringComparison.OrdinalIgnoreCase))
                    .ToList();

                var allMasterTimes = filteredRecords.Select(s => s.Time).ToList();

                textBoxTime.ItemsSource = allMasterTimes;
                textBoxTime.IsEnabled = true;
            }
        }
    }
    else
    {
        textBoxTime.IsEnabled = false;
    }
}

private async void textBoxMaster_SelectionChanged(object sender, SelectionChangedEventArgs e)
{
    UpdateFilteredRecords();

    if (textBoxMaster.SelectedItem != null)
    {
        using (var context = new MyContext())
        {
            var personRecords = await LoadRecordsFromDatabaseAsync(context);

            var selectedMaster = textBoxMaster.SelectedItem.ToString();

            // Фильтруем записи для выбранного мастера
            var recordsForSelectedMaster = personRecords
                .Where(p => p.Master.Equals(selectedMaster, StringComparison.OrdinalIgnoreCase) && p.IsRegistered == true)
                .ToList();

            // Получаем уникальные даты работы для выбранного мастера
            var uniqueDates = recordsForSelectedMaster
                .Select(p => p.Date)
                .Distinct()
                .ToList();

            // Устанавливаем источник данных для textBoxDate
            textBoxDate.ItemsSource = uniqueDates;
        }
    }
}

```

```

    }
    //Расчет стоимости при посещении
    private void ButtonPrice_Click(object sender, RoutedEventArgs e)
    {
        CalculateTotalPrice();
    }

    //Вызов справочника расписания и скидки
    private void btnSchedule_Click_1(object sender, RoutedEventArgs e)
    {
        Schedule schedule = new Schedule();

        schedule.ShowDialog();
    }

    //Вызов отчета
    private void btnReport_Click(object sender, RoutedEventArgs e)
    {
        Report report = new Report();
        report.Show();
    }

    //Удаление записи
    private void btnDelete_Click(object sender, RoutedEventArgs e)
    {
        if (selectedPr == null) return;
        if (System.Windows.MessageBox.Show("Вы уверены?", "Удалить запись?", MessageBoxButton.YesNo) == MessageBoxResult.No)
            return;

        // Сохраняем ссылку на MastSchedule перед удалением записи
        var mastSchedule = selectedPr.MastSchedule;

        // Удаляем запись из таблицы Person
        App.Db.Persons.Remove(selectedPr);
        App.Db.SaveChanges();

        // Обновляем источник данных для DataGrid
        dGrid.ItemsSource = null;
        dGrid.ItemsSource = App.Db.Persons.Local.ToBindingList();

        // Обновляем таблицу MastSchedule
        if (mastSchedule != null)
        {
            // Проверяем, актуальна ли дата
            DateTime scheduleDate;
            if (DateTime.TryParse(mastSchedule.Date, out scheduleDate) && scheduleDate > DateTime.Now)
            {
                mastSchedule.IsRegistered = true; // Устанавливаем IsRegistered в false
            }
            else
            {
                App.Db.MastSchedules.Remove(mastSchedule); // Удаляем запись, если дата не актуальна
            }

            App.Db.SaveChanges();
        }

        MessageBox.Show("Запись успешно удалена!");
    }

    //Вызов справочника услуг
    private void btnService_Click(object sender, RoutedEventArgs e)
    {
        Service service = new Service();
        service.ShowDialog();
    }

    //Изменение записи
    private void btnEdit_Click(object sender, RoutedEventArgs e)
    {
        if (selectedPr != null)

```

```

    {
        selectedPr.Name = textBoxName.Text;
        selectedPr.Date = textBoxDate.Text;
        selectedPr.Time = textBoxTime.Text;
        selectedPr.NameService = (NameService)textBoxUsl.SelectedItem;
        selectedPr.MastSchedule = App.Db.MastSchedules.FirstOrDefault(m => m.Master == textBoxMaster.Text);
        selectedPr.Material = (Material)Material.SelectedItem;
        selectedPr.Price1 = Price1.Text;
        selectedPr.Price2 = Price2.Text;
        selectedPr.Price = Price.Text;
        App.Db.Persons.AddOrUpdate(selectedPr);
        App.Db.SaveChanges();
        dGrid.SelectedItem = selectedPr;
        dGrid.ItemsSource = null;
        dGrid.ItemsSource = App.Db.Persons.Local.ToBindingList();
    }
}
//Кнопка очистить
public void ButtonClear_Click(object sender, RoutedEventArgs e)
{
    if (textBoxMaster.SelectedIndex == -1)
        textBoxMaster.ItemsSource = App.Db.MastSchedules.Select(m => m.Master).Distinct().ToList();
    if (textBoxDate.SelectedIndex == -1)
        textBoxDate.ItemsSource = App.Db.MastSchedules.Select(m => m.Date).Distinct().ToList();
    textBoxUsl.ItemsSource = null;
    textBoxUsl.ItemsSource = App.Db.NameServices.ToList();
    textBoxDis.ItemsSource = null;
    textBoxDis.ItemsSource = App.Db.DisInfo.ToList();
    textBoxName.Text = "";
    textBoxAge.Text = "";
    textBoxDate.Text = "";
    textBoxTime.Text = "";
    textBoxUsl.Text = " ";
    textBoxDis.Text = "";
    Price1.Text = " ";
    Price2.Text = " ";
    textBoxMaster.Text = " ";
    Price.Text = "";
    Material.Text = "";
    Persent.Text = "";
}
private async void textBoxDate_SelectionChanged(object sender, SelectionChangedEventArgs e)
{
    UpdateFilteredRecords();
    if (textBoxDate.SelectedItem != null)
    {
        using (var context = new MyContext())
        {
            {
                var personRecords = await LoadRecordsFromDatabaseAsync(context);
                var selectedDate = textBoxDate.SelectedItem.ToString();
                // Фильтруем записи для выбранной даты
                var recordsForSelectedDate = personRecords
                    .Where(p => p.Date.Equals(selectedDate, StringComparison.OrdinalIgnoreCase))
                    .ToList();
                // Получаем уникальных мастеров для выбранной даты
                var uniqueMasters = recordsForSelectedDate
                    .Select(p => p.Master)
                    .Distinct()
                    .ToList();
                // Устанавливаем источник данных для textBoxMaster
                textBoxMaster.ItemsSource = uniqueMasters;
            }
        }
    }
}
//запись клиента
private void btnAdd_Click(object sender, RoutedEventArgs e)
{
    dGrid.ItemsSource = null;
    dGrid.ItemsSource = App.Db.Persons.Local.ToBindingList();
    var mastSchedule = App.Db.MastSchedules.FirstOrDefault(m => m.Master == textBoxMaster.Text);

```

```

if (mastSchedule != null)
{
    mastSchedule.IsRegistered = false;
}
dGrid.ItemsSource = null;
dGrid.ItemsSource = App.Db.Persons.Local.ToBindingList();
App.Db.Persons.Add(new Person
{
    Name = textBoxName.Text,
    Date = textBoxDate.Text,
    Time = textBoxTime.Text,
    NameService = (NameService)textBoxUsl.SelectedItem,
    MastSchedule = App.Db.MastSchedules.FirstOrDefault(m => m.Master == textBoxMaster.Text),
    Material = (Material)Material.SelectedItem,
    Price1 = Price1.Text,
    Price2 = Price2.Text,
    Price = Price.Text,

});
MessageBox.Show("Пациент успешно зарегистрирован!");
App.Db.SaveChanges();
}
}
}

```

Приложение Б

(обязательное)

Листинг кода

Код окна - Справочник услуг

```
public partial class Service : Window
{
    public Service()
    {
        InitializeComponent();
        NameService nameService = new NameService();
        serviceGrid.ItemsSource = App.Db.NameServices.ToList();
        stackpanelService.DataContext = nameService;
        serviceGrid.DataContext = nameService;
        stackpanelService = new StackPanel();
    }
    private void Grid_Loaded(object sender, RoutedEventArgs e)
    {
        SqlConnection sqlConnection = new SqlConnection(ConfigurationManager.ConnectionStrings["DefaultConnection"].ConnectionString);
        sqlConnection.Open();
    }
    public void btnAdd_Click(object sender, RoutedEventArgs e)
    {
        var newService = new NameService
        {
            Usl = textBox1.Text,
            Price = decimal.Parse(textBox2.Text),
            Description = textBox3.Text,
            Timecomletion = textBox4.Text
        };
        App.Db.NameServices.AddOrUpdate(newService);
        App.Db.SaveChanges();
        serviceGrid.ItemsSource = null;
        serviceGrid.ItemsSource = App.Db.NameServices.Local.ToBindingList();
    }
    public void btnDelete_Click(object sender, RoutedEventArgs e)
    {
        if (selectedNs == null) return;
        if (MessageBox.Show("Вы уверены?", "Удалить запись?", MessageBoxButton.YesNo) == MessageBoxResult.No)
            return;
        App.Db.NameServices.Remove(selectedNs);
        App.Db.SaveChanges();
        serviceGrid.ItemsSource = null;
        serviceGrid.ItemsSource = App.Db.NameServices.Local.ToBindingList();
        serviceGrid.UpdateLayout(); // Добавлено обновление макета
        // Обновляем только измененный элемент
        serviceGrid.Items.Refresh();
    }
    private void btnEdit_Click(object sender, RoutedEventArgs e)
    {
        if (selectedNs != null)
        {
            selectedNs.Usl = textBox1.Text;
            selectedNs.Price = decimal.Parse(textBox2.Text);
            selectedNs.Description = textBox3.Text;
            selectedNs.Timecomletion = textBox4.Text;

            App.Db.NameServices.AddOrUpdate(selectedNs);
            App.Db.SaveChanges();
            serviceGrid.SelectedItem = selectedNs;
            serviceGrid.ItemsSource = null;
            serviceGrid.ItemsSource = App.Db.NameServices.Local.ToBindingList();
        }
    }
    private void btnClose_Click(object sender, RoutedEventArgs e)
    {
        Close();
    }
}
```

```

    }
    NameService selectedNs = null;
    private void serviceGrid_SelectionChanged(object sender, SelectionChangedEventArgs e)
    {
        if (e.RemovedItems.Count > 0) return;

        selectedNs = (NameService)serviceGrid.SelectedItem;
        if (selectedNs == null) return;
        textBox1.Text = selectedNs.Usl;
        textBox2.Text = selectedNs.Price.ToString();
        textBox3.Text = selectedNs.Description;
        textBox4.Text = selectedNs.Timecompletion;
        App.Db.NameServices.AddOrUpdate(selectedNs);
        App.Db.SaveChanges();
        serviceGrid.ItemsSource = App.Db.NameServices.ToList();
        // Обновляем только измененный элемент
        serviceGrid.Items.Refresh();
    }
}
}

```

Приложение В

(обязательное)

Листинг кода

Основной код – Справочник скидки/проценты

```
public partial class Dis : Window
{
    public Dis()
    {
        InitializeComponent();
        DiscountInfo discountInfo = new DiscountInfo();
        disGrid.ItemsSource = App.Db.DisInfo.ToList();
        disGrid.DataContext = discountInfo;
        stackpanelDis = new StackPanel();
    }
    DiscountInfo selectedDs = null;
    private void disGrid_SelectionChanged(object sender, SelectionChangedEventArgs e)
    {
        if (e.RemovedItems.Count > 0) return;
        selectedDs = (DiscountInfo)disGrid.SelectedItem;
        if (selectedDs == null) return;
        textBox1.Text = selectedDs.NameDis;
        textBox2.Text = selectedDs.Persent.ToString();
        App.Db.DisInfo.AddOrUpdate(selectedDs);
        App.Db.SaveChanges();
        disGrid.ItemsSource = App.Db.DisInfo.ToList();
    }
    private void AddButton_Click(object sender, RoutedEventArgs e)
    {
        var newDisInfo = new DiscountInfo
        {
            NameDis = textBox1.Text,
            Persent = decimal.Parse(textBox2.Text),
        };
        App.Db.DisInfo.AddOrUpdate(newDisInfo);
        App.Db.SaveChanges();
        // Обновляем DataGrid
        disGrid.ItemsSource = null;
        disGrid.ItemsSource = App.Db.DisInfo.Local.ToBindingList();
        disGrid.UpdateLayout(); // Добавлено обновление макета
        // Обновляем только измененный элемент
        disGrid.Items.Refresh();
    }
    private void EditButton_Click_1(object sender, RoutedEventArgs e)
    {
        if (selectedDs != null)
        {
            selectedDs.NameDis = textBox1.Text;
            selectedDs.Persent = decimal.Parse(textBox2.Text);
            App.Db.DisInfo.AddOrUpdate(selectedDs);
            App.Db.SaveChanges();
            disGrid.SelectedItem = selectedDs;
            disGrid.ItemsSource = null;
            disGrid.ItemsSource = App.Db.DisInfo.Local.ToBindingList();
        }
    }
    private void DelButton_Click_2(object sender, RoutedEventArgs e)
    {
        if (selectedDs == null) return;
        if (System.Windows.MessageBox.Show("Вы уверены?", "Удалить запись?", MessageBoxButton.YesNo) == MessageBoxResult.No)
            return;
        App.Db.DisInfo.Remove(selectedDs);
        App.Db.SaveChanges();
        disGrid.ItemsSource = null;
        disGrid.ItemsSource = App.Db.DisInfo.Local.ToBindingList();
        disGrid.UpdateLayout(); // Добавлено обновление макета
        // Обновляем только измененный элемент
        disGrid.Items.Refresh();
    }
    private void ClosButton_Click_3(object sender, RoutedEventArgs e)
    {
        Close();
    }
}
```

Приложение Г

(обязательное) Листинг кода

Основной код – Расписание

```
public partial class Dis : Window
{
    public Dis()
    {
        InitializeComponent();
        DiscountInfo discountInfo = new DiscountInfo();
        disGrid.ItemsSource = App.Db.DisInfo.ToList();
        disGrid.DataContext = discountInfo;
        stackpanelDis = new StackPanel();
    }
    DiscountInfo selectedDs = null;
    private void disGrid_SelectionChanged(object sender, SelectionChangedEventArgs e)
    {
        if (e.RemovedItems.Count > 0) return;
        selectedDs = (DiscountInfo)disGrid.SelectedItem;
        if (selectedDs == null) return;
        textBox1.Text = selectedDs.NameDis;
        textBox2.Text = selectedDs.Persent.ToString();
        App.Db.DisInfo.AddOrUpdate(selectedDs);
        App.Db.SaveChanges();
        disGrid.ItemsSource = App.Db.DisInfo.ToList();
    }
    private void AddButton_Click(object sender, RoutedEventArgs e)
    { var newDisInfo = new DiscountInfo
        {
            NameDis = textBox1.Text,
            Persent = decimal.Parse(textBox2.Text),
        };
        App.Db.DisInfo.AddOrUpdate(newDisInfo);
        App.Db.SaveChanges();
        // Обновляем DataGrid
        disGrid.ItemsSource = null;
        disGrid.ItemsSource = App.Db.DisInfo.Local.ToBindingList();
        disGrid.UpdateLayout(); // Добавлено обновление макета
        // Обновляем только измененный элемент
        disGrid.Items.Refresh();
    }
    private void EditButton_Click_1(object sender, RoutedEventArgs e)
    {
        if (selectedDs != null)
        {
            selectedDs.NameDis = textBox1.Text;
            selectedDs.Persent = decimal.Parse(textBox2.Text);
            App.Db.DisInfo.AddOrUpdate(selectedDs);
            App.Db.SaveChanges();
            disGrid.SelectedItem = selectedDs;
            disGrid.ItemsSource = null;
            disGrid.ItemsSource = App.Db.DisInfo.Local.ToBindingList();
        }
    }
    private void DelButton_Click_2(object sender, RoutedEventArgs e)
    {
        if (selectedDs == null) return;
        if (System.Windows.MessageBox.Show("Вы уверены?", "Удалить запись?", MessageBoxButton.YesNo) == MessageBoxResult.No)
            return;
        App.Db.DisInfo.Remove(selectedDs);
        App.Db.SaveChanges();
        disGrid.ItemsSource = null;
        disGrid.ItemsSource = App.Db.DisInfo.Local.ToBindingList();
        disGrid.UpdateLayout(); // Добавлено обновление макета
        // Обновляем только измененный элемент
        disGrid.Items.Refresh();
    }
    private void ClosButton_Click_3(object sender, RoutedEventArgs e){
        Close();
    }
}
```


Приложение Д

(обязательное) Листинг кода

Основной код – Отчет

```
public partial class Report : Window
{
    public Report()
    {
        InitializeComponent();
        Loaded+= OnLoaded;
    }
    private void OnLoaded(object sender, RoutedEventArgs e)
    {
        ns.Items.Add(new NameService());
        foreach (NameService nameService in App.Db.NameServices) ns.Items.Add(nameService);
        mat.Items.Add(new Material());
        foreach (Material material in App.Db.Materials) mat.Items.Add(material);
        ms.Items.Add(new MastSchedule());
        foreach (MastSchedule mastSchedule in App.Db.MastSchedules) ms.Items.Add(mastSchedule);
        FiltrationContext();
    }
    private void DateFilterChanged(object sender, SelectionChangedEventArgs e) => FiltrationContext();
    private void textFilterChanged(object sender, TextChangedEventArgs e) => FiltrationContext();
    private void comboFilterChanges(object sender, SelectionChangedEventArgs e) => FiltrationContext();
    private void Close_Click(object sender, RoutedEventArgs e) => Close();
    private void FiltrationContext()
    {
        dGrid.ItemsSource = null;
        dGrid.ItemsSource = App.Db.Persons.Where(Filter).ToList();
    }
    private NameService _ns => (NameService)ns.SelectedItem;
    private Material _mat => (Material)this.mat.SelectedItem;
    private MastSchedule _ms => (MastSchedule)ms.SelectedItem;
    private bool Filter(Person p)
    {
        if (p == null) return false;

        DateTime? startDate = startDatePicker.SelectedDate;
        DateTime? endDate = endDatePicker.SelectedDate;

        if (startDate.HasValue && endDate.HasValue)
        {
            DateTime personDate;
            if (!DateTime.TryParse(p.Date, out personDate) || personDate < startDate.Value || personDate > endDate.Value)
            {
                return false;
            }
        }

        return (!(string.IsNullOrEmpty(dt.Text) && p.Date != null && !p.Date.Trim().Contains(dt.Text.Trim())) ||
            (!string.IsNullOrEmpty(tm.Text) && p.Time != null && !p.Time.Trim().Contains(tm.Text.Trim())) ||
            (!string.IsNullOrEmpty(person.Text) && p.Name != null && !p.Name.Trim().Contains(person.Text.Trim())) ||
            (_ns != null && ns.SelectedIndex > 0 && p.NameService?.Id != _ns.Id) ||
            (_mat != null && mat.SelectedIndex > 0 && p.Material?.Id != _mat.Id) ||
            (_ms != null && ms.SelectedIndex > 0 && p.MastSchedule?.Id != _ms.Id));
    }
    private void Button_Click(object sender, RoutedEventArgs e)
    {
        Close();
    }

    private void ButtonExp_Click(object sender, RoutedEventArgs e)
    {
        dynamic excelApp = Activator.CreateInstance(Type.GetTypeFromProgID("Excel.Application"));
        var workbook = excelApp.Workbooks.Add();
        var sheet = workbook.Sheets[1];
        // Добавляем заголовок с диапазоном дат
        string startDate = startDatePicker.SelectedDate.HasValue ? startDatePicker.SelectedDate.Value.ToString("dd.MM.yyyy") : "не указана";
```

```

string endDate = endDatePicker.SelectedDate.HasValue ? endDatePicker.SelectedDate.Value.ToString("dd.MM.yyyy") : "не указана";
sheet.Cells[1, 1].Value2 = $"Отчет за период: {startDate} - {endDate}";
sheet.Cells[1, 1].Font.Bold = true;
sheet.Cells[1, 1].Interior.Color = System.Drawing.Color.Yellow;
// Заполняем данными из DataGrid
for (int i = 0; i < dGrid.Columns.Count; i++)
{
    sheet.Cells[3, i + 1].Value2 = dGrid.Columns[i].Header;
    for (int j = 0; j < dGrid.Items.Count; j++)
    {
        TextBlock cellContent = dGrid.Columns[i].GetCellContent(dGrid.Items[j]) as TextBlock;
        sheet.Cells[j + 4, i + 1].Value2 = cellContent?.Text ?? "-";
    }
}
// Настройка ширины столбцов
for (int i = 1; i <= 6; i++)
{
    sheet.Columns[i].ColumnWidth = 15;
}
// Центрирование текста в первой строке
sheet.Cells[3, 1].HorizontalAlignment = XlHAlign.xlHAlignCenter;
sheet.Cells[3, 1].Font.Size = 12;
excelApp.Visible = true;
}
}

```

Приложение Е

(обязательное) Листинг кода

Основной код – Ведомость начисления заработной платы

```
public partial class Zarplata : Window
{
    public async Task<List<Person>> LoadPersonRecordsFromDatabaseAsync(MyContext context)
    {
        var records = await context.Persons.ToListAsync();
        return records;
    }
    public Zarplata()
    {
        InitializeComponent();
        Master.ItemsSource = App.Db.Persons.Select(p => p.MastSchedule.Master).Distinct().ToList(); //comboBox - заполнение данными
        (список мастеров без повторений)
        Procent1.ItemsSource = App.Db.DisInfo.ToList();
        Procent2.ItemsSource = App.Db.DisInfo.ToList();
    }
    private void Date1_TextChanged(object sender, SelectionChangedEventArgs e)
    {
    }
    private void Date2_TextChanged(object sender, SelectionChangedEventArgs e)
    {
    }
    private void ButtonC1_Click(object sender, RoutedEventArgs e)
    {
        if (Master.SelectedIndex >= -1)
        {
            Master.ItemsSource = App.Db.Persons.Select(m => m.MastSchedule.Master).Distinct().ToList();

            Date1.Text = "";
            Date2.Text = "";
            Master.Text = "";
            textBox1.Text = "";
            textBox2.Text = "";
            textBox3.Text = "";
            textBox4.Text = "";
            textBox5.Text = "";
            Procent1.Text = "";
            Procent2.Text = "";
            Proc1.Text = "";
            Proc2.Text = "";
        }
    }
    private void ButtonRas_Click(object sender, RoutedEventArgs e)
    {
        if (decimal.TryParse(textBox2.Text, out decimal pr1) && !string.IsNullOrEmpty(textBox4.Text))
        {
            if (decimal.TryParse(textBox4.Text, out decimal pr2))
            {
                textBox5.Text = (pr1 + pr2).ToString();
            }
            else if (decimal.TryParse(textBox2.Text, out decimal price1))
            {
                textBox5.Text = price1.ToString();
            }
        }
    }
    private void ButtonOt_Click(object sender, RoutedEventArgs e)
    {
        App.Db.Cals.AddOrUpdate(new Cal
        {
            Date1 = Date1.Text,
            Date2 = Date2.Text,
            Master = Master.Text,
            Procent1 = (DiscountInfo)Procent1.SelectedItem,
            Summa1 = textBox1.Text,
            Procent2 = (DiscountInfo)Procent2.SelectedItem,
```

```

        Summa2 = textBox3.Text,
        Total = textBox5.Text
    });

    MessageBox.Show("Документ записан");
    App.Db.SaveChanges();
    Close();
}
private void Procent1_SelectionChanged(object sender, SelectionChangedEventArgs e)
{
    if (Procent1.SelectedItem is DiscountInfo dis)
    {
        Proc1.Text = dis.Persent.ToString("#0%");

        if (textBox1.Text != null)
        {
            textBox2.Text = (decimal.Parse(textBox1.Text) * dis.Persent).ToString("#0.00");
        }
    }
}
private void Procent2_SelectionChanged(object sender, SelectionChangedEventArgs e)
{
    if (Procent2.SelectedItem is DiscountInfo dis)
    {
        Proc2.Text = dis.Persent.ToString("#0%");

        if (textBox3.Text != null)
        {
            textBox4.Text = (decimal.Parse(textBox3.Text) * dis.Persent).ToString("#0.00");
        }
    }
}
private async void Master_SelectionChanged(object sender, SelectionChangedEventArgs e)
{
    if (string.IsNullOrEmpty(Date1.Text) || string.IsNullOrEmpty(Date2.Text))
    {
        // Даты не введены
        MessageBox.Show("Введите нужный период.");
        return;
    }

    if (Master.SelectedItem != null)
    {
        using (var context = new MyContext())
        {
            var personRecords = await LoadPersonRecordsFromDatabaseAsync(context);

            // Преобразуем строки в объекты DateTime с проверкой на успешное преобразование
            if (DateTime.TryParseExact(Date1.Text, "dd.MM.yyyy", System.Globalization.CultureInfo.InvariantCulture,
                System.Globalization.DateTimeStyles.None, out DateTime date1) &&
                DateTime.TryParseExact(Date2.Text, "dd.MM.yyyy", System.Globalization.CultureInfo.InvariantCulture,
                System.Globalization.DateTimeStyles.None, out DateTime date2))
            {
                // Получение диапазонов дат из таблицы Cal
                var calRecords = context.Cals.ToList();

                foreach (var calRecord in calRecords)
                {
                    DateTime rangeStart = DateTime.Parse(calRecord.Date1.ToString());
                    DateTime rangeEnd = DateTime.Parse(calRecord.Date2.ToString());

                    // Проверка, пересекаются ли диапазоны дат
                    if (date1 <= rangeEnd && date2 >= rangeStart && calRecord.Master == Master.Text.ToString())
                    {
                        MessageBox.Show("Целевая дата находится в указанном диапазоне");
                        return;
                    }
                }
            }

            // Фильтруем записи по мастеру и датам
            var filteredRecords = personRecords
                .Where(p => DateTime.Parse(p.Date) >= date1 && DateTime.Parse(p.Date) <= date2 &&
                    p.MastSchedule.Master.Equals(Master.SelectedItem.ToString(), StringComparison.OrdinalIgnoreCase)).ToList();

            if (filteredRecords.Any())
            {

```

```

        decimal totalCost = 0;
        foreach (var record in filteredRecords)
        {
            if (decimal.TryParse(record.Price1, out decimal price))
            {
                totalCost += price;
            }
        }
        textBox1.Text = totalCost.ToString();
    }
    else
    {
        textBox1.Text = "0";
        MessageBox.Show("За указанный период у мастера нет данных.");
    }
}
}
}
    if (Master.SelectedItem != null)
{
    using (var context = new MyContext())
    {
        var personRecords = await LoadPersonRecordsFromDatabaseAsync(context);

        string dateString1 = Date1.Text;
        string dateString2 = Date2.Text;
        string dateFormat = "dd.MM.yyyy";

        if (DateTime.TryParseExact(dateString1, dateFormat, System.Globalization.CultureInfo.InvariantCulture,
            System.Globalization.DateTimeStyles.None, out DateTime date1) &&
            DateTime.TryParseExact(dateString2, dateFormat, System.Globalization.CultureInfo.InvariantCulture,
            System.Globalization.DateTimeStyles.None, out DateTime date2))
        {
            // Фильтруем записи по мастеру и датам
            var filteredRecords = personRecords
                .Where(p => DateTime.Parse(p.Date) >= date1 && DateTime.Parse(p.Date) <= date2 &&
                p.MastSchedule.Master.Equals(Master.SelectedItem.ToString(), StringComparison.OrdinalIgnoreCase)).ToList();

            if (filteredRecords.Any())
            {
                decimal totalCost = 0;
                foreach (var record in filteredRecords)
                {
                    if (decimal.TryParse(record.Price2, out decimal price))
                    {
                        totalCost += price;
                    }
                }
                textBox3.Text = totalCost.ToString();
            }
        }
    }
}
}
}

```

