Частное учреждение образования
«Колледж бизнеса и права»

ПРОГРАММНОЕ СРЕДСТВО ДЛЯ АВТОМАТИЗАЦИИ НАЧИСЛЕНИЯ ОПЛАТЫ ТРУДА ВОДИТЕЛЯМ ТРАНСПОРТНОЙ КОМПАНИИ

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту по дисциплине
«Базы данных и системы управления базами данных»

КП Т.895017.401

Руководитель проекта (В. Ю. Купцова)
Учащийся (А. А. Петровский)

2021

Содержание

Введение

Целью курсового проекта «Программа для автоматизации начисления оплаты труда водителям транспортной компании» является разработка программного средства «AutoPay.exe», автоматизирующего процесс расчёта суммы оплаты труда водителя, в зависимости от его стажа, оклада, вычетов штрафа и премий. Данная программа служит для решения следующих задач: ведение базы данных, содержащей информацию о водителях, рабочих днях, детей водителей, чеках, премиях и вычетов штрафов; осуществление автоматических расчётов зарплаты водителя; формирование чека в word; просмотра информации о водителе и его детях; просмотр чеков.

Для достижения цели курсового проекта нужно решить следующие задачи:

– выполнить объектно-ориентированный анализ и проектирование системы, результатом которой будет модель системы;

– определить вычислительную систему, необходимую для создания программного средства;

– по модели выполнить проектирование задачи;

– разработать программное средство;

– описать созданное программное средство;

– выбрать методику испытаний;

– описать процесс тестирования применения;

– описать применение программы.

Решение поставленных задач отражено в отчёте, который состоит из шести разделов и содержит необходимую и достаточную информацию по использованию данного программного средства.

В первом разделе «Объектно-ориентированный анализ и проектирование системы» рассматривается сущность и актуальность поставленной задачи, описание существующих аналогов, проектирование модели, отображающей функциональную структуру объекта.

Второй раздел «Вычислительная система» имеет описание вычислительной системы, а именно: технические характеристики персонального компьютера (ПК), требования, которые будут предъявляться к персональному компьютеру, описание операционной системы, языка реализации и языка моделирование.

Третий раздел «Проектирование задачи» включает: требования к программному средству, концептуальный прототип, логическую и физическую структуры данных в контексте среды разработки, структуру и описание функций пользователя в рамках среды разрабатываемого программного средства, функции и элементы управления, проектирование справочной системы программного средства.

Четвёртый раздел «Описание программного средства» отражает общие сведения о программе, функциональное назначение, структуру входных и выходных данных.

В пятом разделе «Методика испытаний» рассматриваются требования к техническим средствам для проведения испытания, требования к характеристикам программы применительно к условиям эксплуатации, требования к информационной и программной совместимости. Также описывается порядок проведения испытаний: функциональное, полное тестирование.

Шестой раздел «Применение» содержит информацию, необходимую в процессе эксплуатации программного средства: его назначение, условия применения, а также справочная система.

В заключении описывается выполнение поставленной задачи, степень соответствия проектных решений задания, причины несоответствия, если таковые имеются.

Графическая часть представлена пятью диаграммами: вариантов использования, классов, последовательности, деятельности и компонентов.

1 Объектно-ориентированный анализ и проектирование системы

1.1 Сущность задачи

Приложение «AutoPay» предназначено для автоматизации расчёта зарплаты водителей, в зависимости от текущего тарифа и длительности отработанного времени, премиях и вычетах. Приложение позволит уменьшить временные затраты на расчёт зарплаты водителей. Приложение заберёт на себя часть обязанностей менеджеров компании, автоматизирует расчёт зарплаты за определённый отчётный период. Обеспечивает автоматическую печать талона и чека в word.

Из всех задач, которые будет решать разрабатываемое программное средство, можно выделить ряд основных:

– ведение базы данных, содержащей информацию о водителях, детях водителей, рабочих днях, чеках, вычетах, премиях;

– расчёт зарплаты за определённый отчётный период;

– расчёт вычета за нарушения политики компании;

– расчёта премии в зависимости от переработанных часов или часов в выходные дни;

– обеспечение организации интерфейса приложения средствами создания меню, кнопочных форм, панелей инструментов.

Данное программное средство направлено на упрощение работы менеджера. Такого рода программы упрощают работу, связанную с расчётом зарплаты, учётом отработанных часов, расчётом вычета, расчётом премии. Разрабатываемая программа будет автоматически рассчитывать зарплату водителя с учётом проведённого времени, действующего тарифа, стажа, количества отработанных часов.

Разрабатываемое программное средство будет упрощать работу менеджера, так как большой штат сотрудников сложно отслеживать. С помощью данной программы, менеджер будет выполнять только контролирующую функцию.

Аналога данного программного средства на предприятии нет.

1.2 Проектирование модели

На основании проведенного анализа предметной области и выявленного круга задач, необходимо построить модель, которая будет отображать функциональную структуру объектов программного средства, производимые ими действия и связи между этими действиями.

В качестве инструмента для проектирования модели программного средства будет выбран унифицированный язык моделирования. Это язык для специфицирования, визуализации, конструирования и документирования программных средств[4].

В рамках языка Unified Modeling Language (UML) все представления о модели сложной системы фиксируются в виде специальных графических конструкций – диаграмм. В терминах языка UML определены следующие виды диаграмм: диаграмма вариантов использования, диаграмма классов, диаграммы поведения (диаграмма деятельности), диаграммы взаимодействия (диаграмма последовательности), диаграммы реализации (диаграмма компонентов), диаграмма «Сущность-связь»[4].

Перечень этих диаграмм представляет собой неотъемлемую часть графической нотации языка UML, сам процесс объектно-ориентированного программирования (ООП) неразрывно связан с процессом построения этих диаграмм. Совокупность построенных таким образом диаграмм содержит всю информацию, необходимую для реализации проекта сложной системы.

Главной целью проектирования моделей является отображение функциональной структуры объекта, то есть производимые ими действия и связи между этими действиями.

Наиболее распространенным средством моделирования данных являются диаграммы «Сущность-связь», которые предназначены для графического представления моделей данных разрабатываемой программной системы и предлагают некоторый набор стандартных обозначений для определения данных и отношений 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 или абстрактных объектов, каждый из которых обладает одинаковыми свойствами и характеристиками. В этом случае каждый рассматриваемый объект может являться экземпляром одной и только одной сущности, должен иметь уникальное имя или идентификатор, а также отличаться от других экземпляров данной сущности. Примерами сущностей могут быть: сотрудник, клиент, товар и так далее.

Связь определяется как отношение или некоторая ассоциация между отдельными сущностями. Примерами связей могут являться родственные отношения типа «отец-сын» или производственные отношения типа «начальник-подчиненный». Другой тип связей задается отношениями «иметь в собственности» или «обладать свойством».

Графическая модель данных 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[3].

Определим сущности для данного программного средства и построим диаграмму «Сущность-связь». Исследовав предметную область можно выделить следующие сущности, относящиеся к данному дипломному проекту: «Ребёнок», «Водитель», «Рабочий день», «Чек оплаты», «Премия», «Вычет».

Диаграмма «Сущность-связь» представлена на рисунке 1.1

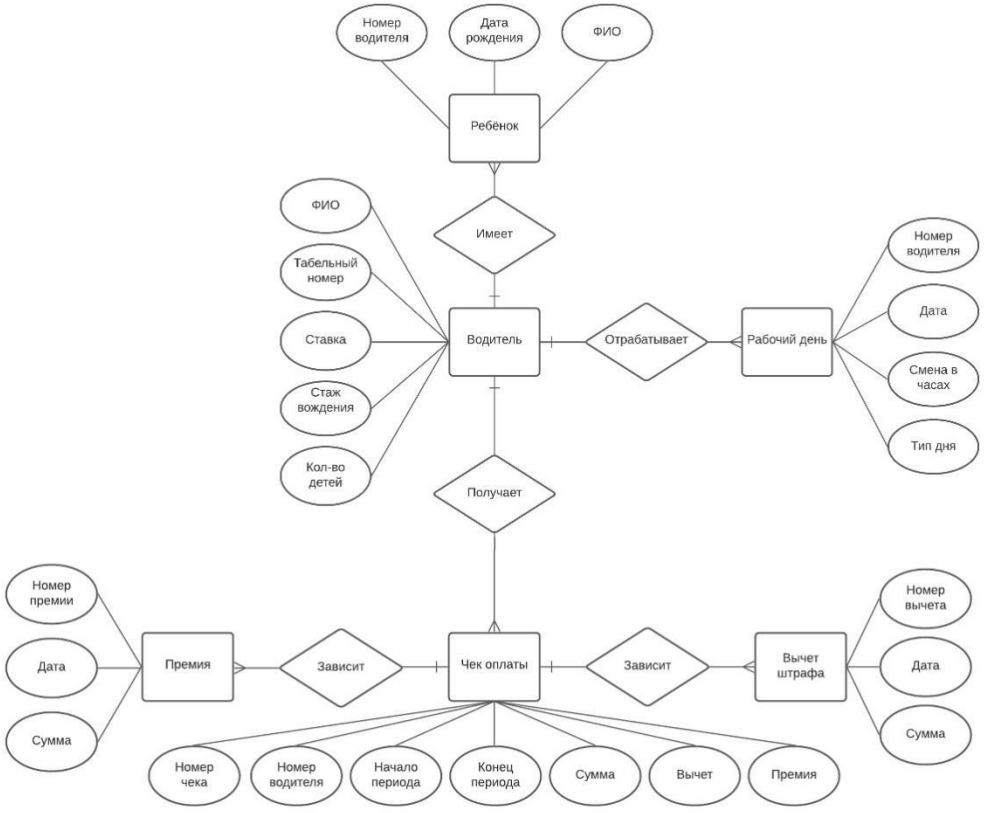

Рисунок 1.1 - Диаграмма «Сущность-связь»

Для сущности «Ребёнок» атрибутами будут являться:
- номер водителя;
- ФИО;
- дата рождения.

Для сущности «Водитель» атрибутами будут являться:
- табельный номер;
- ФИО;
- ставка;
- стаж вождения;
- количество детей.

Для сущности «Рабочий день» атрибутами будут являться:
- Номер водителя;
- дата;
- количество отработанных часов;
- тип дня.

Для сущности «Чек оплаты» атрибутами будут являться:
- номер чека;
- номер водителя;
- начало периода;
- конец периода;
- вычет;
- премия;
- сумма.

Для сущности «Вычет» атрибутами будут являться:
- номер вычета;
- дата;
- сумма.

Для сущности «Премия» атрибутами будут являться:
- номер премии;
- дата;
- сумма.

Диаграмма – граф специального вида, состоящий из вершин в форме геометрических фигур, которые связаны между собой рёбрами или дугами.

Суть диаграммы вариантов использования состоит в следующем: проектируемая система представляется в виде множества сущностей или актёров, взаимодействующих с системой с помощью, так называемых, вариантов использования.

Варианты использования описывают не только взаимодействия между пользователями и сущностью, но также реакции сущности на получение отдельных сообщений от пользователей и восприятие этих сообщений за пределами сущности. Варианты использования могут включать в себя описание особенностей способов реализации сервиса и различных исключительных ситуаций, таких как корректная обработка ошибок системы. Множество вариантов использования в целом должно определять все возможные стороны ожидаемого поведения системы.

В данной проектируемой системе в качестве актёров выступают клиент и оператор, который служит источником воздействия на моделируемую систему.

К основным функциям разрабатываемого программного средства относятся:
- осуществление ведения базы данных (БД);
- осуществление расчёта зарплаты для водителя;
- осуществление формирования печатного чека.

Диаграмма вариантов использования, отражающая варианты использования приложения для пользователя системы представлена в графической части на листе 1.

Диаграмма классов служит для представления статической структуры модели системы в терминологии классов объектно-ориентированного программирования. Диаграмма классов

может отражать, в частности, различные взаимосвязи между отдельными сущностями предметной области, такими как объекты и подсистемы, а также описывает их внутреннюю структуру и типы отношений. На данной диаграмме не указывается информация о временных аспектах функционирования системы. С этой точки зрения диаграмма классов является дальнейшим развитием концептуальной модели проектируемой системы.

Диаграмма классов представляет собой некоторый граф, вершинами которого являются элементы типа «классификатор», которые связаны различными типами структурных отношений. Следует заметить, что диаграмма классов может также содержать интерфейсы, пакеты, отношения и даже отдельные экземпляры, такие как объекты и связи. Когда говорят о данной диаграмме, имеют в виду статическую структурную модель проектируемой системы. Поэтому диаграмму классов принято считать графическим представлением таких структурных взаимосвязей логической модели системы, которые не зависят или инвариантны от времени.

Диаграмма классов представлена в графической части на листе 2.

При моделировании поведения проектируемой или анализируемой системы возникает необходимость детализировать особенности алгоритмической и логической реализации выполняемых системой операций. Для моделирования процесса выполнения операций в языке UML используются так называемые диаграммы деятельности. Каждое состояние на диаграмме деятельности соответствует выполнению некоторой элементарной операции, переход в следующее состояние срабатывает только при завершении этой операции. Графически диаграмма деятельности представляется в форме графа, вершинами которого являются состояния действия, а дугами – переходы от одного состояния действия к другому.

Основная цель использования диаграмм деятельности – визуализация особенностей реализации операций классов, когда необходимо представить алгоритмы их выполнения. Диаграмма деятельности для функции «Оформление чека» представлена в графической части на листе 3.

Временной аспект поведения имеет существенное значение при моделировании синхронных процессов, описывающих взаимодействия объектов. Именно для этой цели и используются диаграммы последовательности, в которых ключевым моментом является динамика взаимодействия объектов во времени. При этом диаграмма последовательности имеет как бы два измерения: одно – слева направо в виде вертикальных линий, каждая из которых изображает линию жизни отдельного объекта, участвующего во взаимодействии; второе – вертикальная временная ось, направленная сверху вниз, на которой начальному моменту времени соответствует самая верхняя часть диаграммы. Диаграмма последовательности для функции «Оформление чека на покупку» представлена в графической части на листе 4.

Диаграмма компонентов описывает объекты реального мира – компоненты программного обеспечения. Эта диаграмма позволяет определить архитектуру разрабатываемой системы, установив зависимости между программными компонентами. Диаграмма компонентов представлена в графической части на листе 5.

2 Вычислительная система

2.1 Требования к аппаратным и операционным ресурсам

Основными минимальными требованиями, выдвигаемыми к аппаратному обеспечению ПК, являются:

- процессор Intel Core i5 – 8250U и выше;
- оперативная память 4048 Мбайт и более;
- свободное место на диске 512 Мбайт;
- интегрированная видеокарта на 512 Мбайт и более;
- монитор;
- мышь, клавиатура;
- принтер.

Компьютер должен работать под управлением операционной системы, начиная с Windows 7 и выше. Наиболее удобной операционной системой для проведения испытаний является Windows 10, так как она ориентированна на максимальное использование всех возможностей ПК, сетевых ресурсов и обеспечение комфортных условий работы.

2.2 Инструменты разработки

Ниже описаны инструменты разработки, которые будут использоваться для написания программного средства.

Персональный компьютер со следующей аппаратной конфигурацией:

- процессор Intel Core i5 – 8250U;
- оперативная память 4048 Мбайт;
- видеокарта mx250(2 Гбайт GDDR4);
- клавиатура и мышь.

Программное обеспечение:

- операционная система Windows 10;
- программная среда разработки Microsoft Visual Studio 2019;
- система управления базами данных Microsoft structured query language (SQL) Server 2019;
- программа Microsoft (MS) Word;
- программа для создания справочной системы Doctor Explain (Dr.Explain);
- язык программирования C#;
- технология WPF(Windows Presentation Foundation).

Операционная система – это набор управляющих программ, предназначенных для управления ресурсами вычислительной системы как единого комплекса, другими словами операционная система – это набор программного обеспечения, который обеспечивает работу компьютера. Основными функциями операционной системы являются:

– управление файловой системой (запись, изменение, копирование файлов, контроль доступа);

– управление выполнением программ (распределение процессорного времени, загрузка программ с диска в оперативную память, перехват потенциально опасных действий);

– управление памятью (кэширование, распределение, контроль сохранности данных);

– диалог с пользователем (чтение команд с клавиатуры, с мыши, вывод информации на экран, на принтер).

При разработки программного средства использовалась операционная система Windows 10, так как на данный момент эта операционная система является самой распространённой

операционной системой. В Windows 10 были исправлены практически все недостатки предыдущих операционных систем. Аппаратные требования Windows 10 скромнее, она способна работать даже на маломощных компьютерах и ещё добавлено множество функций, существенно облегчающих работу за компьютерам[5].

Microsoft Visual Studio 2019 – средство разработки программного обеспечения, разрабатываемое корпорацией Microsoft и включающее язык программирования и среду разработки. В то же время Visual Basic 2019 сочетает в себе процедуры и элементы объектно-ориентированных и компонентно-ориентированных языков программирования. Среда разработки Visual Basic 2019 включает инструменты для визуального конструирования пользовательского интерфейса[6].

Microsoft SQL Server — система управления реляционными базами данных (СУБД), разработанная корпорацией Microsoft. Основной используемый язык запросов — Transact-SQL, создан совместно Microsoft и Sybase. Transact-SQL является реализацией стандарта ANSI/ISO по структурированному языку запросов (SQL) с расширениями. Используется для работы с базами данных размером от персональных до крупных баз, данных масштаба предприятия; конкурирует с другими СУБД в этом сегменте рынка[9].

Doctor Explain (Dr. Explain) – это приложение для создания файлов справки (help-файлов), справочных систем, online руководств пользователя, пособий и технической документации к программному обеспечению и техническим системам

Встроенная технология анализа структуры пользовательского интерфейса позволяет документировать экраны программных приложений практически в автоматическом режиме.

Уникальность заключается в инновационном подходе к созданию пользовательской документации, который значительно ускоряет этот трудоемкий процесс по сравнению с другими инструментами.

Программа способна анализировать пользовательский интерфейс приложений и создавать скриншоты (копии экранов) окон, автоматически расставляя на них пояснительные выноски для элементов интерфейса.

Процесс практически полностью автоматизирован, что позволяет достаточно быстро аннотировать экраны приложений и веб-сайтов для иллюстрирования пользовательской документации на программном обеспечении (ПО)[10].

Microsoft Word – Текстовый процессор, предназначенный для создания, просмотра, редактирования и форматирования текстов статей, деловых бумаг, а также иных документов, с локальным применением простейших форм таблично-матричных алгоритмов. Выпускается корпорацией Microsoft в составе пакета Microsoft Office. Первая версия была написана Ричардом Броди для IBM PC, использующих DOS, в 1983 году. Позднее выпускались версии для Apple Macintosh, SCO UNIX и Microsoft Windows. Текущей версией является Microsoft Office Word 2019 для Windows и macOS, а также веб-версия Word Online, не требующая установки программы на компьютер.

C# – объектно-ориентированный язык программирования. Разработан в 1998–2001 годах группой инженеров компании Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумота[7] как язык разработки приложений для платформы Microsoft .NET Framework. Впоследствии был стандартизирован как ECMA-334 и ISO/IEC 23270.

Технология WPF (Windows Presentation Foundation) является часть экосистемы платформы .NET и представляет собой подсистему для построения графических интерфейсов.

Если при создании традиционных приложений на основе WinForms за отрисовку элементов управления и графики отвечали такие части ОС Windows, как User32 и GDI+, то приложения WPF основаны на DirectX. В этом состоит ключевая особенность рендеринга графики в WPF: используя WPF, значительная часть работы по отрисовке графики, как простейших кнопочек, так и сложных 3D-моделей, ложиться на графический процессор на видеокарте, что также позволяет воспользоваться аппаратным ускорением графики.

3 Проектирование задачи

3.1 Требования к приложению

Разрабатываемое приложение должно иметь понятный и удобный в использовании интерфейс, чтобы взаимодействие между программой и пользователем было максимально упрощено. Для того, чтобы программа выглядела более привлекательно нужно воспользоваться сторонним графическим пользовательским интерфейсом Material Design Theme. Для обучения пользователей необходимо разработать справочную систему, в которой должны быть раскрыты все аспекты работы с программой, возможные трудности, возникшие во время работы и пути их решения.

Кроме того, при разработке форм необходимо соблюдать определённые требования: формы взаимодействия с клиентом не должны содержать лишней информации. Клиент не должен иметь возможности покинуть окно или выйти из приложения. При конструировании форм в необходимых случаях нужно предусмотреть возможность защиты данных от изменения, установить ограничения на некорректный ввод данных. Приложение должно иметь страничную структуру, то есть все изменения интерфейса должны происходить с использованием одного окна.

3.2 Концептуальный прототип

Концептуальный прототип представляет собой описание внешнего пользовательского интерфейса – системы меню, диалоговых окон и элементов управления.

Основной интерфейс программного приложения будут представлять формы. Все формы будут содержать стандартные пользовательские элементы управления.

При запуске приложения должно открываться окно со справочной информацией, возможен переход между окнами водителя, рабочего дня, расчёта отчётного периода. Реализовать возможность добавления водителей, удаления водителе. Окно с отметки рабочего дня водителя требующее id каждого водителя. Окно расчёта зарплаты должно иметь возможность: расчёта зарплаты каждого водителя с учётом отработанного времени.

Окно справки должно содержать TextBox содержащий справочную информацию о работе приложения.

Окно водителя состоит из ListView содержащий список водителей.

Окно отметки рабочего дня содержит форму для записи данных водителя для отметки его за текущий день.

Окно расчёта содержит Calendar с выбором диапазона расчётного периода для расчёта зарплаты водителя.

3.3 Организация данных

Реляционная модель основана на математическом понятии отношения, представлением которого является таблица. В реляционной модели отношения используются для хранения информации об объектах, представленных в базе данных. Отношение имеет вид двухмерной таблицы, в которой строки соответствуют записям, а столбцы – атрибутам. Каждая запись должна однозначно характеризоваться в таблице. Для этого используют первичные и

вторичные ключи. Достоинством реляционной модели является простота и удобство физической реализации.

Реляционная модель базы данных подразумевает нормализацию всех таблиц данных. Нормализация – это формальный метод анализа отношений на основе их первичного ключа и функциональных зависимостей, существующих между их атрибутами.

База данных соответствует реляционной модели данных, где каждый выделенный в ходе проектировании сущности соответствует таблица.

Структура базы данных разрабатываемого программного средства включает 5 таблиц.

Таблица «Ребёнок» хранит информацию о типах устройств, структура приведена в таблице 3.1.

Таблица 3.1 – Структура таблицы «Ребёнок»

Имя поля	Тип данных	Размер, байт	Описание
driver	int	4	Идентификатор оператора
FIO	nchar	30	ФИО ребёнка
birthday	datetime	8	День рождения ребёнка

В таблице «Водитель» хранится информация о клиентах, структура приведена в таблице 3.2.

Таблица 3.2 – Структура таблицы «Водитель»

Имя поля	Тип данных	Размер, байт	Описание
id	int	4	Идентификатор водителя
FIO	nchar	30	ФИО водителя
rate	float	8	Ставка водителя
experience	int	4	Опыт вождения
children	int	4	Идентификатор ребёнка

Таблица «Рабочий день» хранит информацию о покупках, структура приведена в таблице 3.3

Таблица 3.3 – Структура таблицы «Рабочий день»

Имя поля	Тип данных	Размер, байт	Описание
driver	int	4	Идентификатор рабочего дня
datee	datetime	8	Дата
shiftt	int	4	Длительность смены
daytype	nchar	10	Тип дня

В таблице «Премия» представлена информация о сотрудниках, структура приведена в таблице 3.4.

Таблица 3.4 – Структура таблицы «Премия»

Имя поля	Тип данных	Размер,байт	Описание
id	int	4	Идентификатор премии
datee	datetime	8	Дата
summa	decimal	8	Сумма премии

Таблица «Вычет» хранит данные о товарах, структура приведена в таблице 3.5.

Таблица 3.5 – Структура таблицы «Вычет»

Имя поля	Тип данных	Размер, байт	Описание
id	int	4	Идентификатор вычета
datee	datetime	8	Дата
summa	decimal	8	Сумма вычета

Таблица «Чек» хранит данные о товарах, структура приведена в таблице 3.6.

Таблица 3.6 – Структура таблицы «Чек»

Имя поля	Тип данных	Размер, байт	Описание
id	int	4	Идентификатор чека
driver	int	4	Идентификатор водителя
starte	datetime	8	Начало периода
ende	datetime	8	Конец периода
summa	decimal	8	Сумма зарплаты
decrease	int	4	Вычет
bonus	int	4	Премия

Физическая структура базы данных представлена схемой данных на рисунке 3.1.

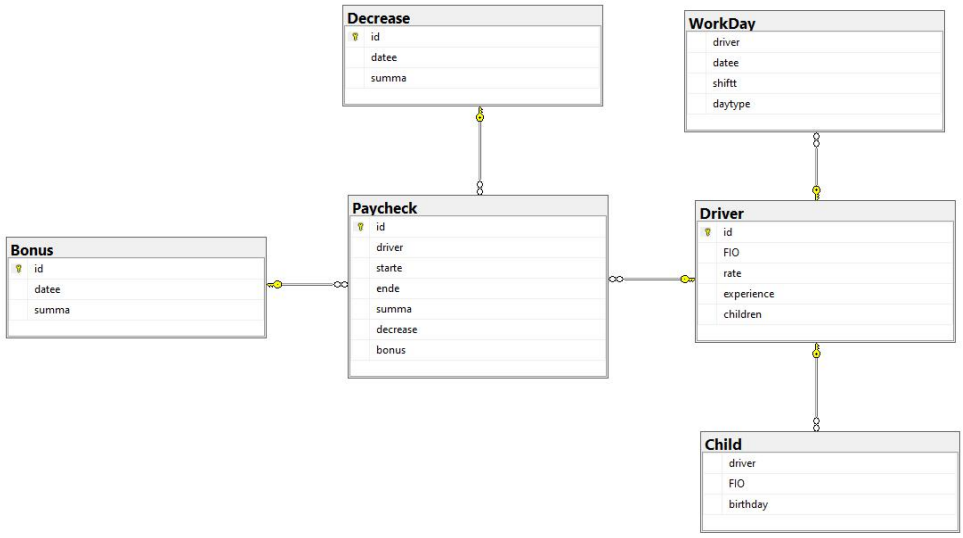

Рисунок 3.1 - Схема данных

3.4 Функции: логическая и физическая организация

Обязательными функциями программы являются добавление водителя и его детей в базу данных, расчёт зарплаты водителя, отметка рабочего дня водителя.

Функции, которые выполняет программа – печать чека о зарплате водителя, занесение водителя и его детей в базу данных, запись времени работы водителя, расчёт зарплаты водителя в зависимости от выбранного расчётного периода.

Для успешного выполнения вышеперечисленных функций программу необходимо запустить на персональном компьютере (ПК).

Функция «Расчёты зарплаты» закреплена за элементом управления типа Button. При нажатии на кнопку «Создать чек» вызывается метод CuclButton класса CulcPage. Функция требует ввод штрафа и премии водителя, выбор водителя. Код функции представлен ниже:

```
private void CuclButton(object sender, RoutedEventArgs e)
    {
      DateTime StartDate;
      DateTime EndDate;

      if (DateStart.SelectedDate == null)
      {
        AlarmMessage.Content = "Введите начальную дату!";
        return;
      }

      if (DateEnd.SelectedDate == null)
      {
        AlarmMessage.Content = "Введите конечную дату!";
        return;
      }

      StartDate = DateStart.SelectedDate.Value;
      EndDate = DateEnd.SelectedDate.Value;

      if(StartDate >= EndDate)
```

```
            {
                AlarmMessage.Content = "Введите верную дату!";
                return;
            }

            AlarmMessage.Content = "";

            if(showBonus.Text == "")
            {
                showBonus.Text = "0";
            }

            if(showDecrease.Text == "")
            {
                showDecrease.Text = "0";
            }

            DataTable summ = SQLbase.Select($"select sum(shiftt) from WorkDay where driver = {id} and datee > '{StartDate}' and datee < '{EndDate}' ");

            SQLbase.Insert($"insert into Bonus(datee, summa) values (GETDATE(), {showBonus.Text})");
            SQLbase.Insert($"insert into Decrease(datee,summa) values (GETDATE(), {showDecrease.Text})");
            DataTable table = SQLbase.Select("SELECT MAX(id) FROM Bonus");
            DataTable table1 = SQLbase.Select("SELECT MAX(id) FROM Decrease");

            showSummaBefore.Text = (Int32.Parse(summ.Rows[0][0].ToString()) * 15).ToString();
            showSumma.Text = (Int32.Parse(showSummaBefore.Text) - Int32.Parse(showDecrease.Text) + Int32.Parse(showBonus.Text)).ToString();

            SQLbase.Insert($"insert into Paycheck(driver, starte, ende, summa, decrease, bonus) values ({id}, '{StartDate}', '{EndDate}', {showSumma.Text}, {table.Rows[0][0]}, {table1.Rows[0][0]})");

            name = showNameStr.Text;
            fDate = $"{StartDate.Day}:{StartDate.Month}:{StartDate.Year}";
            sDate = $"{EndDate.Day}:{EndDate.Month}:{EndDate.Year}";
            summa = showSumma.Text;
            b = showBonus.Text;
            d = showDecrease.Text;

            isCanWord = true;
        }
```

Функция «Добавление водителя» закреплена за методом Button_Add класса AddPage. Функция требует ввод имени, фамилии, отчества, опыта вождения и стажа . Код функции представлен ниже:

```
    private void Button_Add(object sender, RoutedEventArgs e)
{
    string name = formName.Text.Trim();
    string surname = formSurnameName.Text.Trim();
```

```
string secondname = formSecondName.Text.Trim();
string experience = formExperience.Text.Trim();
string rate = formRate.Text.Trim();

bool isGood = true;

if(name == "")
{
   isGood = false;

   formName.ToolTip = "Заполните поле!";
   formName.Foreground = Brushes.Red;
}
else
{
   formName.ToolTip = "";
   formName.Foreground = Brushes.Black;
}

if (surname == "")
{
   isGood = false;

   formSurnameName.ToolTip = "Заполните поле!";
   formSurnameName.Foreground = Brushes.Red;
}
else
{
   formSurnameName.ToolTip = "";
   formSurnameName.Foreground = Brushes.Black;
}

if (secondname == "")
{
   isGood = false;

   formSecondName.ToolTip = "Заполните поле!";
   formSecondName.Foreground = Brushes.Red;
}
else
{
   formSecondName.ToolTip = "";
   formSecondName.Foreground = Brushes.Black;
}

if (experience == "")
{
   isGood = false;

   formExperience.ToolTip = "Заполните поле!";
   formExperience.Foreground = Brushes.Red;
}
else
```

```
{
    formExperience.ToolTip = "";
    formExperience.Foreground = Brushes.Black;
}

if (rate == "")
{
    isGood = false;

    formRate.ToolTip = "Заполните поле!";
    formRate.Foreground = Brushes.Red;
}
else
{
    formRate.ToolTip = "";
    formRate.Foreground = Brushes.Black;
}

foreach (Char x in name)
{
    if (char.IsDigit(x))
    {
        formName.ToolTip = "Требуется символьное значение!";
        formName.Foreground = Brushes.Red;
        isGood = false;
    }
    else
    {
        formName.ToolTip = "";
        formName.Foreground = Brushes.Black;
    }
}

foreach (Char x in surname)
{
    if (char.IsDigit(x))
    {
        formSurnameName.ToolTip = "Требуется символьное значение!";
        formSurnameName.Foreground = Brushes.Red;
        isGood = false;
    }
    else
    {
        formSurnameName.ToolTip = "";
        formSurnameName.Foreground = Brushes.Black;
    }
}

foreach (Char x in secondname)
{
    if (char.IsDigit(x))
    {
        formSecondName.ToolTip = "Требуется символьное значение!";
```

```
            formSecondName.Foreground = Brushes.Red;
            isGood = false;
        }
        else
        {
            formSecondName.ToolTip = "";
            formSecondName.Foreground = Brushes.Black;
        }
    }

    foreach (Char x in experience)
    {
        if (!char.IsDigit(x))
        {
            formExperience.ToolTip = "Требуется числовое значение!";
            formExperience.Foreground = Brushes.Red;
            isGood = false;
        }
        else
        {
            formExperience.ToolTip = "";
            formExperience.Foreground = Brushes.Black;
        }
    }

    foreach (Char x in rate)
    {
        if (!char.IsDigit(x))
        {
            formRate.ToolTip = "Требуется числовое значение!";
            formRate.Foreground = Brushes.Red;
            isGood = false;
        }
        else
        {
            formRate.ToolTip = "";
            formRate.Foreground = Brushes.Black;
        }
    }



    if (isGood)
    {
        SQLbase.Insert($"INSERT INTO Driver VALUES ('{name} {surname} {secondname}',
{rate}, {experience}, 0)");

        MessageBox.Show("Водитель добавлен.");

        formName.Text = "";
        formSecondName.Text = "";
        formSurnameName.Text = "";
        formExperience.Text = "";
```

```
        formRate.Text = "";
    }
```

Функция «Отметить водителя» закреплена за методом ButtonAddMark класса Mark. Требуется выбрать выбрать водителя, ввести тип дня и длительность смены. Код функции представлен ниже:

```
    private void ButtonAddMark(object sender, RoutedEventArgs e)
{
    string countHours = formTime.Text.Trim();
    int i = listDrivers.SelectedIndex;
    bool isGood = true;

    if (i == -1)
    {
        AlarmText.Visibility = Visibility.Visible;
        isGood = false;
    }
    else
    {
        AlarmText.Visibility = Visibility.Hidden;
    }

    if (countHours == "")
    {
        isGood = false;

        formTime.ToolTip = "Заполните поле!";
        formTime.Foreground = Brushes.Red;
    }
    else
    {
        formTime.ToolTip = "";
        formTime.Foreground = Brushes.Black;
    }

    foreach (Char x in countHours)
    {
        if (!char.IsDigit(x))
        {
            formTime.ToolTip = "Требуется числовое значение!";
            formTime.Foreground = Brushes.Red;
            isGood = false;
        }
        else
        {
            formTime.ToolTip = "";
            formTime.Foreground = Brushes.Black;
        }
    }
    if (!isGood){ return; }
```

```
        DataTable tableDriver = SQLbase.Select($"SELECT * FROM Driver");
        DataTable tableMarks = SQLbase.Select($"select * from WorkDay where
CONVERT(date,GETDATE()) like datee and driver = {tableDriver.Rows[i][0]}");
        if(tableMarks.Rows.Count > 0)
        {
            MessageBox.Show("Ошибка: водитель уже отмечен!");
            return;
        }

        if (day == TypeOfDays.working)
        {
            SQLbase.Insert($"INSERT INTO WorkDay(driver, datee, shiftt, daytype) values
({tableDriver.Rows[i][0]}, CONVERT(date,GETDATE()), {countHours}, 'Будний')");
        }
        else if(day == TypeOfDays.weekday)
        {
            SQLbase.Insert($"INSERT INTO WorkDay(driver, datee, shiftt, daytype) values
({tableDriver.Rows[i][0]}, CONVERT(date,GETDATE()), {countHours}, 'Выходной')");
        }
        else if(day == TypeOfDays.holiday)
        {
            SQLbase.Insert($"INSERT INTO WorkDay(driver, datee, shiftt, daytype) values
({tableDriver.Rows[i][0]}, CONVERT(date,GETDATE()), {countHours}, 'Праздник')");
        }

        MessageBox.Show("Водитель был отмечен!");
    }
```

Текст программы представлен в приложении А.

3.5 Проектирование справочной системы приложения

Для работы с приложением начинающего пользователя необходимо обеспечить качественной справочной системой, в которой должны быть приведены методы и приемы работы с приложением, включающие данные о том, что произойдет после нажатия на определенную кнопку или при выборе пункта меню.

Справочная система представляет собой приложение «AutoPay.chm», поставляемое вместе с программой.

Справочная система необходима для ознакомления с программой. В ней должна присутствовать информация, которая может пригодиться пользователю: о правилах пользования приложением и о его возможностях.

Система справки данного программного средства будет содержать следующие разделы:

- «Страница добавления водителей»;
- «Страница добавления детей»;
- «Страница отметки рабочего времени водителя»;
- «Страница расчёта зарплаты»;
- «Страница просмотра списка водителей».

Справочная система будет создана с помощью программного средства Dr.Explain.

4 Описание программного средства

4.1 Общие сведения

Приложение «AutoPay.exe» предназначено для автоматизации расчёта зарплаты водителя в зависимости от отработанного времени и льгот. Приложение позволит уменьшить временные затраты на контроль и расчёт зарплаты водителям. Приложение заберёт на себя часть обязанностей менеджера, автоматизирует расчёт зарплаты за определённый отчётный период. Обеспечивает автоматическую печать чека в word.

Данное программное средство было разработано при использовании персонального компьютера со следующей конфигурацией:

- процессор Intel Core i5 – 8250U;
- оперативная память 8048 Мбайт;
- твёрдотельный накопитель на 256 Гбайт;
- видеокарта mx250 на 2048 ГМбайт;

Программное средство создано в среде разработки Visual Studio 2019 на языке программирования C# в операционной системе Windows 10. Программное средство может работать в средах операционных систем семейства Microsoft Windows начиная с Windows 7. Программа не требовательна к системным ресурсам, также проста в использовании и не требует специальных навыков при работе. Для работы данного программного средства требуется предварительная установка и настройка следующих программных продуктов:

- система управления базами данных Microsoft SQL Server 2019;
- программная платформа Microsoft .Net Framework 4.7.2.

Инсталляция программного средства не требуется, достаточно только скопировать готовый проект на ПК и запустить. Название программного средства – AutoPay.exe.

4.2 Функциональное назначение

Программное средство было разработано с целью облегчения работы сотрудников предприятия путем автоматизации часто выполняемых операций, визуализации основных процессов.

Основными задачами приложения являются автоматизированный ввод данных о водителях, редактирование уже имеющейся информации,

Целью программного средства является снижение бумажной работы, уменьшение количества ошибок при составлении отчетов, быстрое использование данных.

Программа использует стандартные элементы управления, такие как кнопки, меню, списки, поля ввода, что обеспечивает единство интерфейса системы и программного средства, а так же элемент TreeView для создания древовидного каталога товаров.

В работе программного средства предусмотрены некоторые ситуации, которые должны предупреждать пользователя, чтобы он выполнял все необходимые требования по эксплуатации программы. Для этого существуют сообщения системы, например, если менеджер не выполнил заполнение всех полей для добавления водителя, то в этом случае выводится на экран сообщение о том, что администратор не заполнил все поля.

Таким образом, программа может применяться в реальных условиях, представляя собой достаточно удобный помощник.

4.3 Входные данные

Входными являются данные о водителях, детях, чеках, рабочих дня, премиях, вычетах которые необходимы для взаимодействия с программой

Входными данными при добавлении водителя являются:

- ФИО;
- ставка;
- стаж вождения;
- кол-во детей;
- выбранное место.

Входными данными при добавлении детей являются:

- номер водителя;
- дата рождения;
- ФИО.

Входными данными при добавлении рабочего дня являются:

- номер водителя;
- смена в часах.

Входными данными при добавлении чека являются:

- дата начала периода;
- дата конца периода.

4.4 Выходные данные

Выходные данные – это сформированная зарплата водителя с учётом отработанного времени и выбранного отчётного периода.

Выходными данными программы является чек с основной информацией о зарплате сотрудника.

5 Методика испытаний

5.1 Технические требования

Минимальными требованиями для оптимальной работы программного средства является персональный компьютер со следующими характеристиками:

- процессор Intel Core i5 – 8250U и выше;
- оперативная память 4024 Мбайт и более;
- свободное место на диске 512 Мбайт;
- интегрированная видеокарта на 512 Мбайт и более;

Компьютер должен работать под управлением операционной системы, начиная с Windows 7 и выше. Наиболее удобной операционной системой для проведения испытаний является Windows 10, так как она ориентированна на максимальное использование всех возможностей ПК, сетевых ресурсов и обеспечение комфортных условий работы.

5.2 Функциональное тестирование

Функциональное тестирование – это тестирование ПО в целях проверки реализуемости функциональных требований, то есть способности ПО в определенных условиях решать задачи, нужные пользователем. Функциональные требования определяют, что именно делает ПО, какие задачи оно решает. При функциональном тестировании осуществляется проверка каждого пункта меню, каждой операции с целью проверки выполнения всех функций, определенных на этапе проектирования задачи. Функциональное тестирование должно гарантировать работу всех элементов управления в автономном режиме.

Тестирование было реализовано при помощи тест-кейсов, которые представлены в таблицах 5.1-5.5. Тестирование проводилось на уровне разработчика программного средства. Допускаются погрешность временных показателей связанных с процессом тестирования.

Данное тестирование проводится для выявления неполадок и недочетов программы на этапе ее сдачи в эксплуатацию.

Функциональное тестирование предполагает проверку выполнения всех определенных на этапе проектирования функций.

Таблица 5.1 – Тест-кейс функции добавление водителя

№	Функция	Шаги выполнения	Результат
1	Добавление водителя	1. Запустить программу 2. Перейти в пункт «Водители», представленный на рисунке 5.1 3. Перейти в пункт добавления водителей, представленный на рисунке 5.2 4. Заполнить поля 5. Нажать на кнопку «Добавить»	Водитель добавлен в базу данных, результат представлен рисунке 5.3

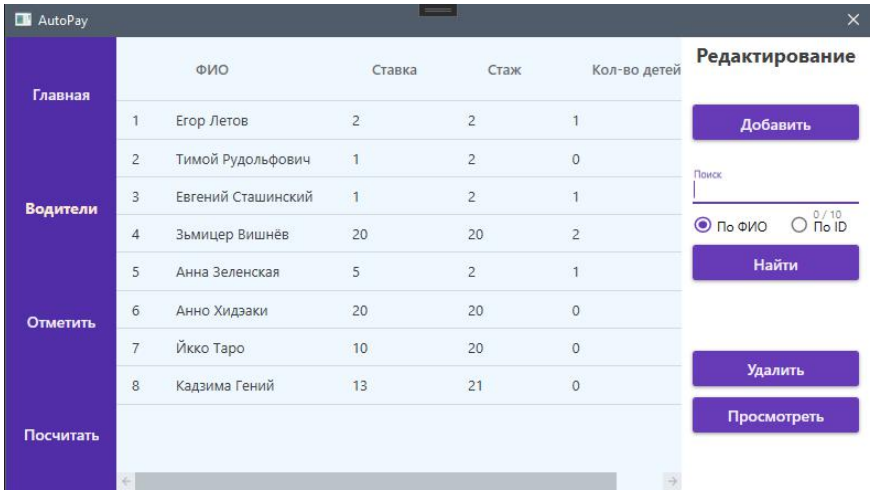

Рисунок 5.1 – Список водителей

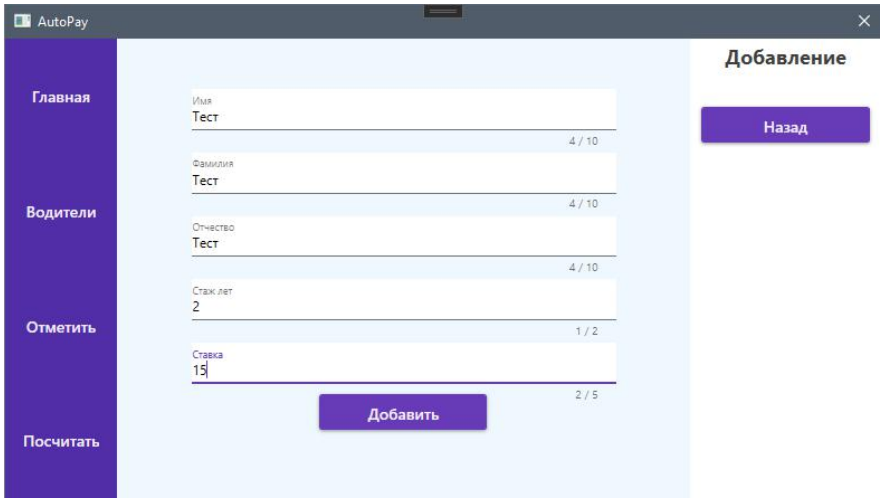

Рисунок 5.2 – Страница добавления водителя

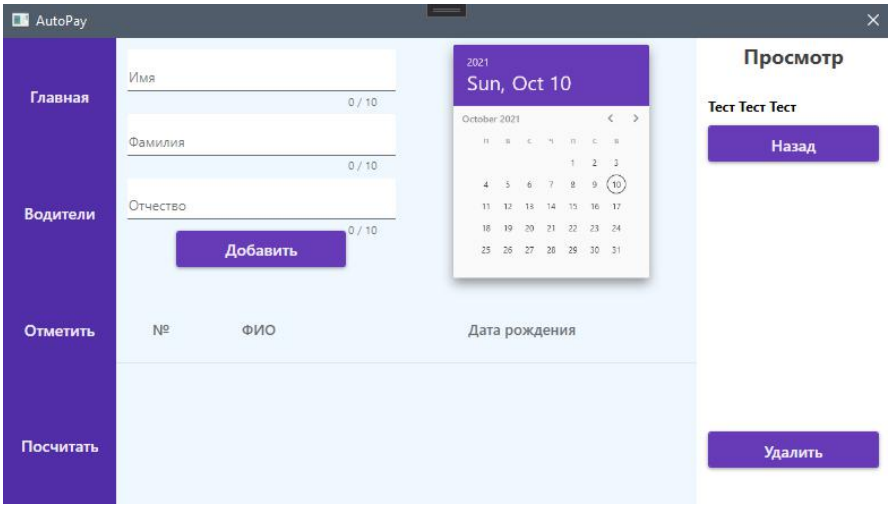

Рисунок 5.3 – Страница просмотра водителя

Таблица 5.2 – Тест-кейс функции добавления детей

№	Функция	Шаги выполнения	Результат
2	Добавление детей водителю	1. Запустить программу 2. Перейти в пункт «Водители», представленный на рисунке 5.1 3. Перейти в пункт просмотра водителей, представленный на рисунке 5.3 4. Заполнить поля 5. Нажать на кнопку «Добавить»	Ребёнок добавлен в бд, результат представлен на рисунке 5.4

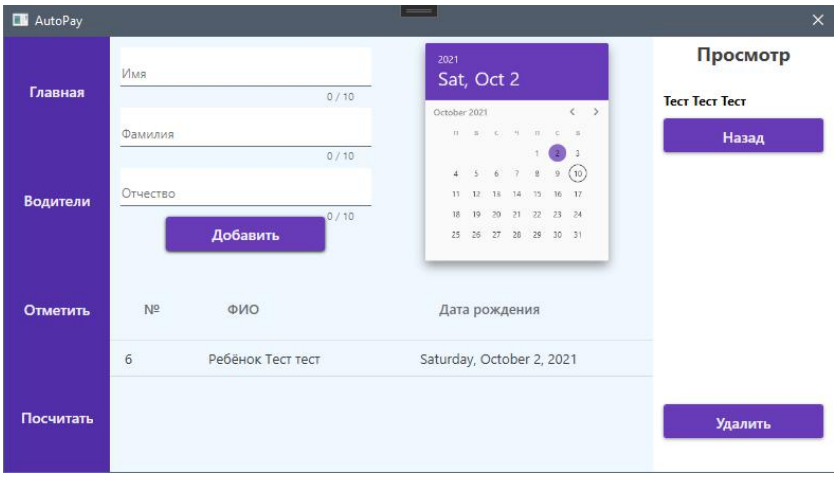

Рисунок 5.4 – Просмотр детей водителя

Таблица 5.3 – Тест-кейс функции удаления водителя

№	Функция	Шаги выполнения	Результат
2	Удаление водителя	1. Запустить программу 2. Перейти в пункт «Водители», представленный на рисунке 5.1 3. Выбрать водителя из списка 4. Нажать на кнопку «Удалить»	Водитель удалён из базы данных, результат представлен на рисунке 5.5

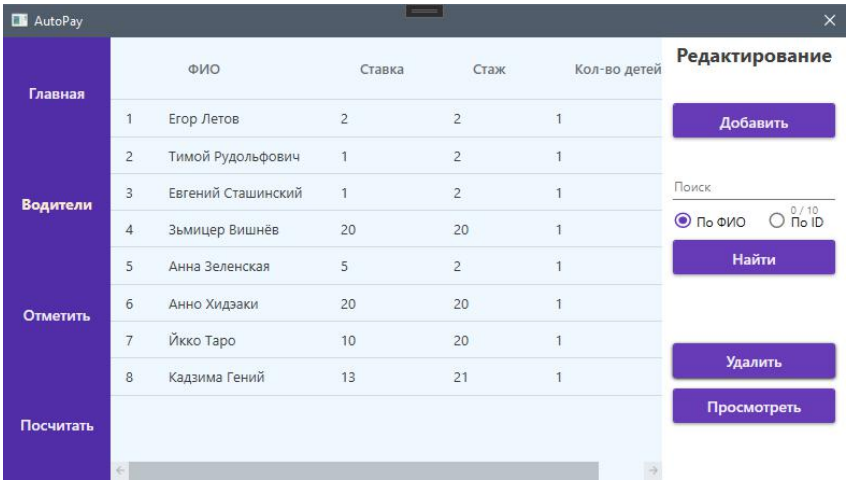

Рисунок 5.5 – Просмотр удаления водителя

Таблица 5.4 – Тест-кейс функции отметки рабочего дня водителя

№	Функция	Шаги выполнения	Результат
2	Отметка рабочего дня водителя	1. Запустить программу 2. Перейти в пункт «Отметить», представленный на рисунке 5.6 3. Выбрать водителя из списка 4. Заполнить поля 5. Нажать на кнопку «Отметить»	Водитель был отмечен на текущую дату и добавлен в базу данных, результат представлен на рисунке 5.7

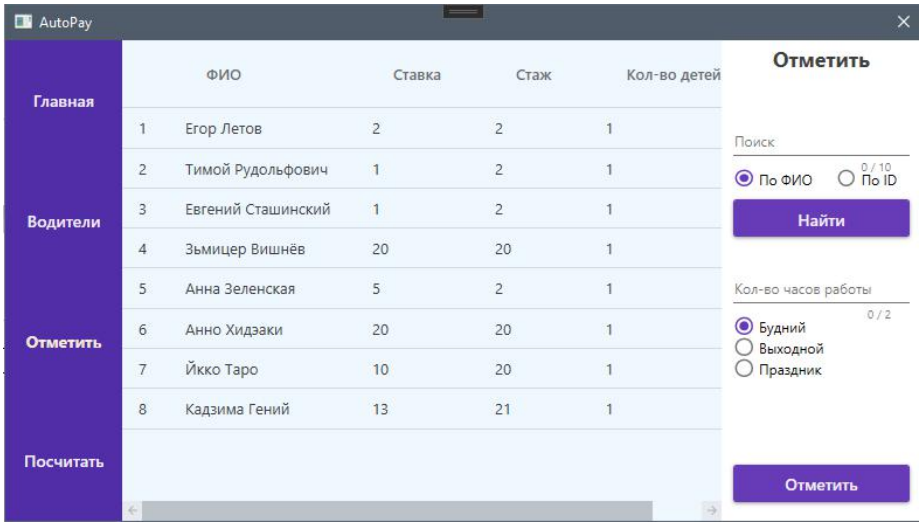

Рисунок 5.6 – Страница отметки водителей

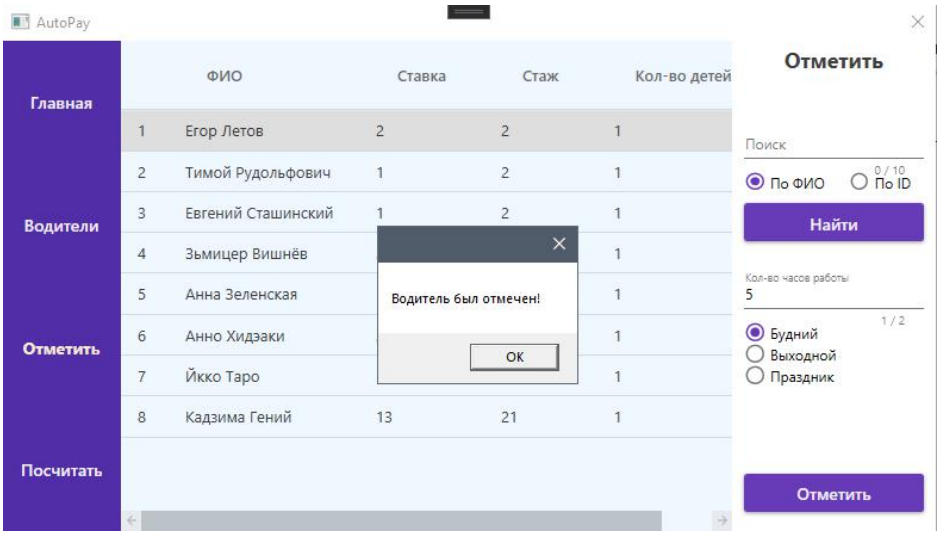

Рисунок 5.7 – Подтверждение отметки рабочего дня водителя

Таблица 5.5 – Тест-кейс функции расчёта зарплаты водителя

№	Функция	Шаги выполнения	Результат
2	Расчёт зарплаты водителя	1. Запустить программу 2. Перейти в пункт «Просчитать», представленный на рисунке 5.8 3. Выбрать водителя из списка 4. Нажать на кнопку «Посчитать» 5. Выбрать период и заполнить требуемые поля представленные на рисунке 5.9 6. Нажать на кнопку «Получить чек»	Зарплата водителя расчитана и занесена в базу данных, результат представлен на рисунке 5.10

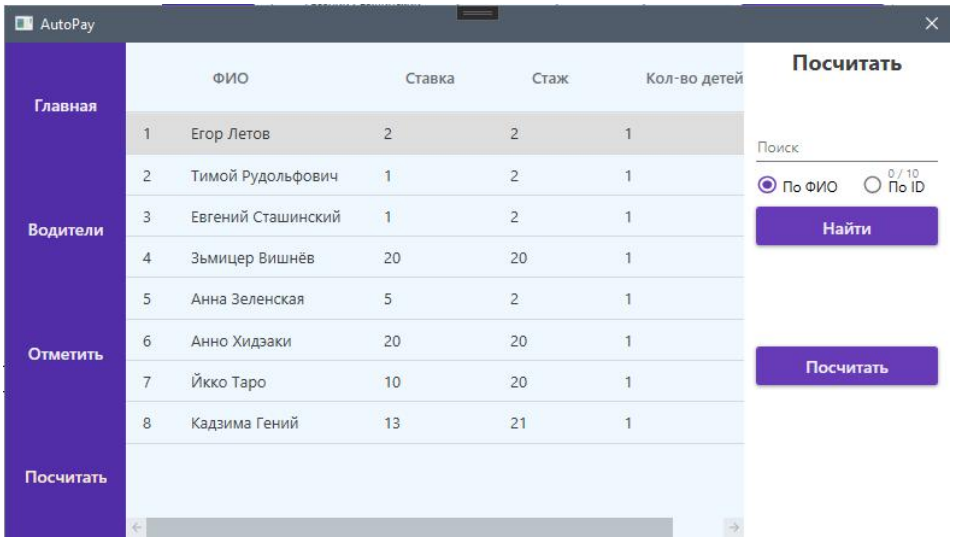

Рисунок 5.8 – Подтверждение удаления

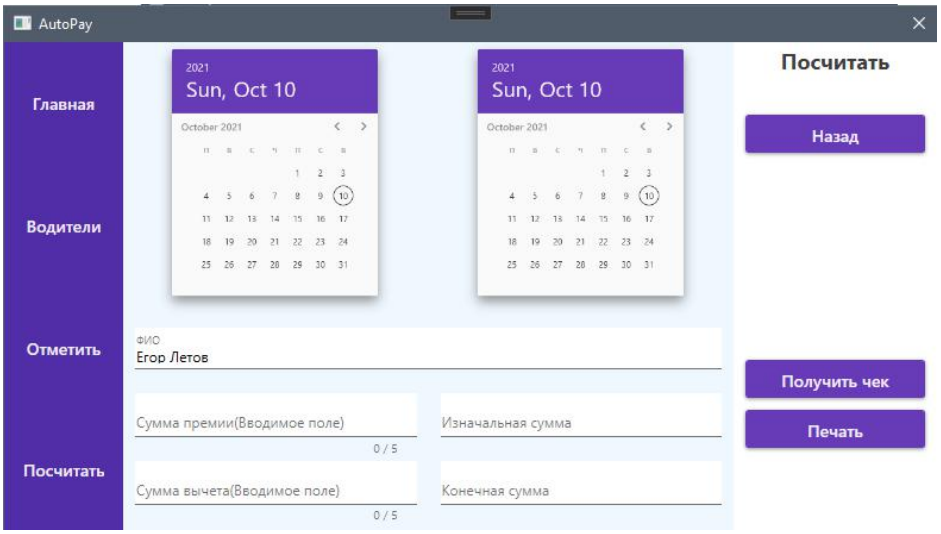

Рисунок 5.9 – Страница расчёта зарплаты водителя

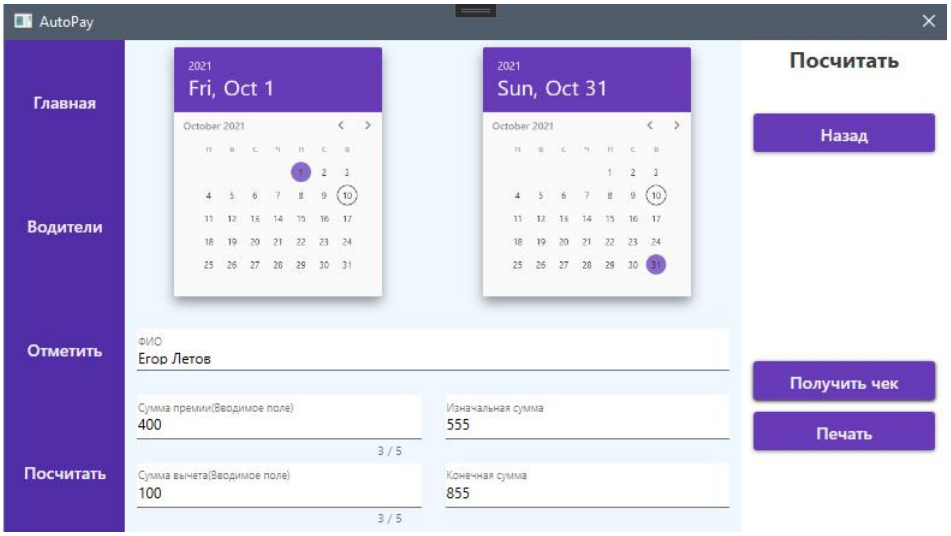

Рисунок 5.10 – Результат вычесления зарплаты водителя

6 Применение

6.1 Назначение программы

Программа для автоматизации расчёта зарплаты водителей в зависимости от отработанного времени.

Основной целью программного средства является уменьшение участия работников в процессе расчёта зарплаты водителей. Уменьшение временных затрат при составлении отчетов, быстрое использовать данных, хранящиеся в системе для учёта зарплаты за отчётный период.

База данных проекта содержит необходимые таблицы для реализации поставленных задач.

Предоставлена возможность добавления новых водителей и их удаление. Добавление детей и их удаление, отметка водителей и расчёт зарплаты с возможность печати отчёта.

6.2 Условия применения

Для применения данного программного средства необходимы следующие технические требования:

- процессор Intel Core i5 – 8250U;
- оперативная память 8048 Мбайт;
- твёрдотельный накопитель на 256 Гбайт;
- видеокарта mx250 на 2024 ГМбайт;
- операционная система Windows 7 и выше;
- Framework v4.7.5;
- рекомендуется монитор типа VGA или с лучшей разрешающей способностью;
- клавиатура;
- мышь;
- принтер.

6.3 Справочная система

Справочная система программного средства представляет собой отдельный файл «Help.chm» с описанием основных функций программы в формате *.chm. В справочной системе даны ответы на типичные вопросы, возникающие при работе с приложением, что, несомненно, должно помочь при освоении программного средства.

Система справки данного программного средства будет содержать следующие разделы:

- «Страница водителей»;
- «Страница отметок рабочего дня»;
- «Страница расчёта».

Структура справочной системы представлена на рисунке 6.1.

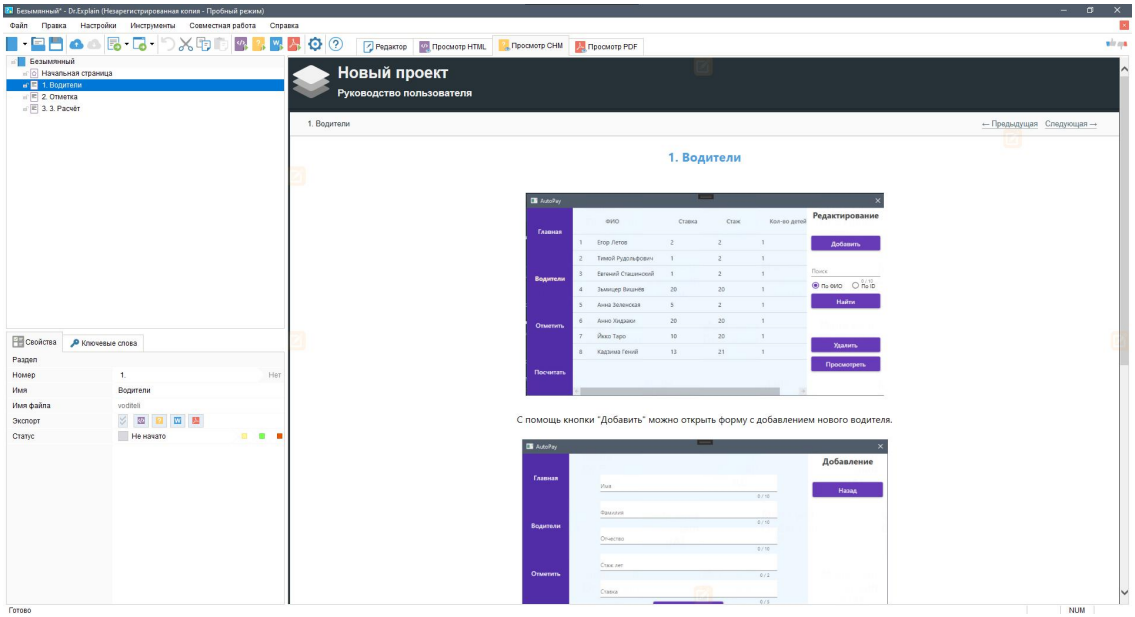

Рисунок 6.1 - Структура спарвочной системы

Заключение

В рамках курсового проектирования на тему «программа для автоматизации начисления оплатыв труда водителям транспортной компании» было разработано программное средство, автоматизирующее расчёт зарплаты водителей транспортной компании.

Для достижения цели были решены следующие задачи:

– выполнен объектно-ориентированный анализ и проектирование системы, результатом которой будет модель системы;

– определена вычислительная система, необходимая для создания программного средства;

– по модели выполнено проектирование задачи;

– разработано программное средство;

– описано созданное программное средство;

– выбрана методика испытаний;

– описан процесс тестирования применения;

– приведены примеры области.

Программа реализована в полном объеме и в соответствии с заданными требованиями, полностью отлажена и протестирована. Поставленные задачи выполнены.
Разработка имеет интуитивно понятный интерфейс, позволяющий с минимальным знанием компьютера использовать данное программное средство.

В процессе разработки данного программного средства были применены и закреплены знания по уже изученному материалу, были отработаны навыки владения методами надёжного программирования и эффективности разработки программного обеспечения.

Разработанная база позволяет получить всю необходимую информацию о прибыли за отчётный период, осуществить смену тарифа, добавление и удаление операторов, сформировать талон и чек содержащие необходимую клиенту информацию.

При разработке приложения наибольшее внимание уделялось максимальному созданию лаконичного интерфейса.

Проект был разработан в среде Microsoft Visual Studio 2019 на языке C# при разработке базы данных Microsoft SQL Server 2019.

Список используемых источников

1. Багласова, Т.Г. Методические указания по оформлению курсовых и дипломных проектов / Т.Г. Багласова, К.О. Якимович. – Минск: КБП, 2017

2. Общие требования к тестовым документам: ГОСТ 2.105-95. – Введ. 01.01.1996. – Минск: Межгос. совет по стандартизации, метрологии и сертификации, 1995. – 84 с.

3. Программа и методика испытаний. Требования к содержанию, оформлению и контролю качества: ГОСТ 19.301-2000. – Введ. 01.09.2001. – Минск: Межгос. совет по стандартизации, метрологии и сертификации, 2000. – 14 с.

4. Текст программы. Требования к содержанию, оформлению и контролю качества: ГОСТ 19.401-2000. – Введ. 01.09.2001. – Минск: Межгос. совет по стандартизации, метрологии и сертификации, 2000. – 16 с.

5. Блох, Джошуа. Java: эффективное программирование / Джошуа Блох – 3-е изд.: Пер. с англ. – СПб.: ООО «Диалектика», 2019. – 464 с.: ил. – Парал. тит. англ.

6. Волк, В.К. Базы данных. Проектирование, программирование, управления и администрирование / В.К. Волк. – Санкт-Петербург: Лань, 2010. – 244 с. : ил.

7. Гуськова, О.И. Объектно ориентированное программирование в Java: учебное пособие / О. И. Гуськова. – Москва: МПГУ, 2018. – 240 с.

8. Илюшечкин, В.М. Основы использования и проектирования баз данных: учебник СПО / В.М. Илюшечкин. – М.: Издательство Юрайт, 2019. – 213 с. – Серия: Профессиональное образование.

9. Коузен, К. Современный Java: рецепты программирования / пер. с анг. А. А. Слинкина. – М.: ДМК Пресс, 2018. – 274 с.: ил.

10. Новиков, Б.А. Основы технологий баз данных: учебное пособие / Б. А. Новиков, Е. А. Горшкова, Н. Г. Графеева; под ред. Е. В. Рогова. – 2-е изд. – М.: ДМК Пресс, 2020. – 582 с.

11. Шилдт, Герберт. Java. Полное руководство / Герберт Шилдт. – 10-е изд. : Пер. с англ. – СПб. ООО «Альфа-книга»; 2018. – 1488 с.: ил. – Парал. тит. англ.

12. IntelliJ IDEA [Электронный ресурс]. – jetbrains.com, 2021. – Режим доступа: https://www.jetbrains.com/ru-ru/idea/. – Дата доступа: 22.04.2021.

13. Microsoft Windows 10 – Полный обзор [Электронный ресурс]. – настройкапк.рф, 2021. – Режим доступа: https://настройкапк.рф/Полезная_информация/Microsoft_Windows_10/. – Дата доступа: 31.03.2021.

14. UMLet – Free UML Tool for Fast UML Diagrams. [Electronic resource]. – Mode of access: https://www.umlet.com/, 2021. – Date of access: 27.05.2021.

15. MySql Workbench – https://ru.wikipedia.org/wiki/MySQL_Workbench

16. Михалевич В.Ю. Методические указания к курсовому проектированию / В.Ю. Михалевич. – Минск: КБП, 2020

Приложение А
(обязательное)
Текст программы

Создание базы данных

```
CREATE DATABASE AutoPay

USE AutoPay
GO

DROP TABLE Paycheck;
DROP TABLE Bonus;
DROP TABLE Decrease;
DROP TABLE WorkDay;
DROP TABLE Child;
DROP TABLE Driver;

CREATE TABLE Driver(
	id int identity(1,1),
	FIO nchar(40) not null,
	rate float not null,
	experience int not null,
	children int not null,
	CONSTRAINT Prim_ID_Driver PRIMARY KEY (id),
)
GO

CREATE TABLE Child(
	id int identity(1,1),
	driver int not null,
	FIO nchar(40) not null,
	birthday date not null,
	CONSTRAINT Fore_Child_Driver FOREIGN KEY (driver) REFERENCES Driver([id])
)
GO

CREATE TABLE WorkDay(
	driver int not null,
	datee date not null,
	shiftt int not null,
	daytype nchar(10) not null,
	CONSTRAINT Fore_WorkDay_Driver FOREIGN KEY (driver) REFERENCES Driver([id])
)
GO

CREATE TABLE Bonus
(
id int primary key identity(1,1) not null,
datee date not null,
summa decimal not null
)
GO

CREATE TABLE Decrease
(
id int primary key identity(1,1) not null,
datee date not null,
summa decimal not null
)
GO

CREATE TABLE Paycheck
(
```

```
id int primary key identity(1,1) not null,
driver int not null,
starte date not null,
ende date not null,
summa decimal not null,
decrease int not null,
bonus int not null,
CONSTRAINT Fore_Paycheck_Driver FOREIGN KEY (driver) REFERENCES Driver([id]),
CONSTRAINT Fore_Paycheck_Decrease FOREIGN KEY (decrease) REFERENCES decrease([id]),
CONSTRAINT Fore_Paycheck_Bonus FOREIGN KEY (bonus) REFERENCES bonus([id])
)
GO

INSERT INTO Driver(FIO, rate, experience, children)
values
('Егор Летов', 2, 2, 1),
('Тимой Рудольфович', 1, 2, 0),
('Евгений Сташинский', 1, 2, 1),
('Зьмицер Вишнёв', 20, 20, 2),
('Анна Зеленская', 5, 2, 1),
('Анно Хидэаки', 20, 20, 0),
('Йкко Таро', 10, 20, 0),
('Кадзима Гений', 13, 21, 0)
GO

INSERT INTO Child(driver, FIO, birthday)
values
(1, 'Петровский Андрей', '17-07-2003'),
(3, 'Скребец Татьяна', '09-03-2003'),
(4, 'Алфимов Арсэн', '29-03-2003'),
(5, 'Лях Артём', '10-04-2003'),
(4, 'Сидоренко Дарья', '20-08-2003')
GO

INSERT INTO WorkDay(driver, datee, shiftt, daytype)
values
(1, '07-10-2021', 8, 'Выходной'),
(1, '06-10-2021', 8, 'Выходной'),
(1, '05-10-2021', 8, 'Выходной'),
(1, '30-10-2021', 8, 'Выходной'),
(3, '06-10-2021', 2, 'Выходной'),
(4, '07-10-2021', 3, 'Выходной')
GO

insert into Bonus(datee, summa)
values
(GETDATE(), 200)
go

insert into Decrease(datee,summa)
values
(GETDATE(), 100)
go

insert into Paycheck(driver, starte, ende, summa, decrease, bonus) values (1, '01-10-2021',
'10-10-2021', 200, 1, 1)
GO

create trigger trig_1
on Driver
instead of delete
as
if exists (select * from Deleted Driver join Child on Driver.id = Child.driver)
begin
delete from Child where driver in (select driver from deleted)
begin
```

```
delete from Driver where id in (select id from deleted)
end
end
GO

SET DATEFORMAT DMY;
GO



delete Driver where id = 6
select count(*) from Child where driver = 10

select * from Driver
select * from Child
select * from WorkDay

select getDate()

SELECT GETDATE ( )

select * from WorkDay where CONVERT(date,GETDATE()) like datee and driver = 2

SELECT * FROM Driver where FIO like '%Erop%'

SELECT MAX(id) FROM Bonus

select sum(shiftt) from WorkDay where driver = 1 and datee > '01-10-2021' and datee < '10-10-
2021'
public partial class Mark : Page
    {
        TypeOfDays day;
        TypeOfSearch tSearch;

        public Mark()
        {
            InitializeComponent();
            showInfo();
        }

        private enum TypeOfDays
        {
            working,
            weekday,
            holiday
        }

        private enum TypeOfSearch
        {
            name,
            id
        }

        private void showInfo()
        {
            DataTable table = SQLbase.Select($"SELECT * FROM Driver");
            listDrivers.ItemsSource = table.DefaultView;
        }

        private void showInfo(int id)
        {
            DataTable table = SQLbase.Select($"SELECT * FROM Driver where id = {id}");
            listDrivers.ItemsSource = table.DefaultView;
        }

        private void showInfo(string name)
```

```
        {
            DataTable table = SQLbase.Select($"SELECT * FROM Driver where FIO like
'%{name}%'");
            listDrivers.ItemsSource = table.DefaultView;
        }

        private void Button_Main(object sender, RoutedEventArgs e)
        {
            NavigationService.Navigate(new Uri("/MainPage.xaml", UriKind.Relative));
        }

        private void Button_Drivers(object sender, RoutedEventArgs e)
        {
            NavigationService.Navigate(new Uri("/Drivers.xaml", UriKind.Relative));
        }

        private void Button_Caclulate(object sender, RoutedEventArgs e)
        {
            NavigationService.Navigate(new Uri("/PayMent.xaml", UriKind.Relative));
        }

        private void RadioButton_Checked(object sender, RoutedEventArgs e)
        {
            RadioButton pressed = (RadioButton)sender;
            string s = pressed.Content.ToString();

            if (s == "Будний")
            {
                day = TypeOfDays.working;
                return;
            }
            if (s == "Выходной")
            {
                day = TypeOfDays.weekday;
                return;
            }
            if (s == "Праздник")
            {
                day = TypeOfDays.holiday;
                return;
            }
        }

        private void ButtonAddMark(object sender, RoutedEventArgs e)
        {
            string countHours = formTime.Text.Trim();
            int i = listDrivers.SelectedIndex;
            bool isGood = true;

            if (i == -1)
            {
                AlarmText.Visibility = Visibility.Visible;
                isGood = false;
            }
            else
            {
                AlarmText.Visibility = Visibility.Hidden;
            }

            if (countHours == "")
            {
                isGood = false;

                formTime.ToolTip = "Заполните поле!";
                formTime.Foreground = Brushes.Red;
            }
```

```
            else
            {
                formTime.ToolTip = "";
                formTime.Foreground = Brushes.Black;
            }

            foreach (Char x in countHours)
            {
                if (!char.IsDigit(x))
                {
                    formTime.ToolTip = "Требуется числовое значение!";
                    formTime.Foreground = Brushes.Red;
                    isGood = false;
                }
                else
                {
                    formTime.ToolTip = "";
                    formTime.Foreground = Brushes.Black;
                }
            }

            if (!isGood){ return; }


            DataTable tableDriver = SQLbase.Select($"SELECT * FROM Driver");

            DataTable tableMarks = SQLbase.Select($"select * from WorkDay where
CONVERT(date,GETDATE()) like datee and driver = {tableDriver.Rows[i][0]}");

            if(tableMarks.Rows.Count > 0)
            {
                MessageBox.Show("Ошибка: водитель уже отмечен!");
                return;
            }

            if (day == TypeOfDays.working)
            {
                SQLbase.Insert($"INSERT INTO WorkDay(driver, datee, shiftt, daytype) values
({tableDriver.Rows[i][0]}, CONVERT(date,GETDATE()), {countHours}, 'Будний')");
            }
            else if(day == TypeOfDays.weekday)
            {
                SQLbase.Insert($"INSERT INTO WorkDay(driver, datee, shiftt, daytype) values
({tableDriver.Rows[i][0]}, CONVERT(date,GETDATE()), {countHours}, 'Выходной')");
            }
            else if(day == TypeOfDays.holiday)
            {
                SQLbase.Insert($"INSERT INTO WorkDay(driver, datee, shiftt, daytype) values
({tableDriver.Rows[i][0]}, CONVERT(date,GETDATE()), {countHours}, 'Праздник')");
            }

            MessageBox.Show("Водитель был отмечен!");
        }

        private void RadioButton1_Checked(object sender, RoutedEventArgs e)
        {
            RadioButton pressed = (RadioButton)sender;
            string s = pressed.Content.ToString();

            if (s == "По ФИО")
            {
                tSearch = TypeOfSearch.name;
                return;
            }
            if (s == "По ID")
            {
```

```
            tSearch = TypeOfSearch.id;
            return;
        }
    }

    private void showBonus_PreviewTextInput(object sender, TextCompositionEventArgs e)
    {
        e.Handled = "0123456789 ,".IndexOf(e.Text) < 0;
    }

    private void Button_Find(object sender, RoutedEventArgs e)
    {
        string input = formSearch.Text;
        bool isGood = true;

        if (input.Trim() == "")
        {
            formSearch.ToolTip = "Заполните поле!";
            formSearch.Foreground = Brushes.Red;
            isGood = false;
        }
        else
        {
            formSearch.ToolTip = "";
            formSearch.Foreground = Brushes.Black; ;
        }

        if (tSearch == TypeOfSearch.name)
        {
            foreach (Char x in input)
            {
                if (char.IsDigit(x))
                {
                    formSearch.ToolTip = "Требуется символьное значение!";
                    formSearch.Foreground = Brushes.Red;
                    isGood = false;
                }
                else
                {
                    formSearch.ToolTip = "";
                    formSearch.Foreground = Brushes.Black;

                }
            }

            if (isGood)
            {
                showInfo(input);
            }
        }
        else if (tSearch == TypeOfSearch.id)
        {
            foreach (Char x in input)
            {
                if (!char.IsDigit(x))
                {
                    formSearch.ToolTip = "Требуется числовое значение!";
                    formSearch.Foreground = Brushes.Red;
                    isGood = false;
                }
                else
```

Приложение Б
(Справочное)
Выходные документы

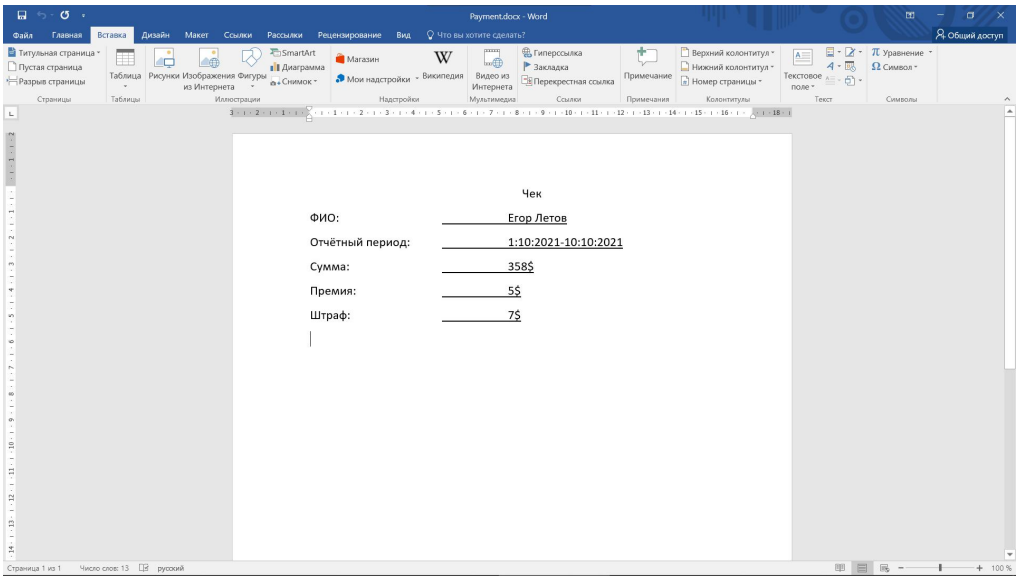

Рисунок Б.1 – Выходные данные чека водителя

КП Т.895017.401 ГЧ

Подп. и

Инв.№дубл.Инв

Взам.инв.№В

Подп. и

Инв.№подл.И

					КП Т.895017.401 ГЧ
					ПРОГРАММНОЕ СРЕДСТВО ДЛЯ АВТОМАТИЗАЦИИ НАЧИСЛЕНИЯ ОПЛАТЫ ТРУДА ВОДИТЕЛЯМ ТРАНСПОРТНОЙ КОМПАНИИ
Изм	Лист	№ докум.№	Подпись	Дата	
Разраб.		Петровский А.А.			
Провер.		Купцова В.Ю.			
Н. Контр.					
Реценз.					
Т. Контр.					
Утверд.					

Лит.	Масса	Масштаб
У		

Лист 1	Листов 5

Диаграмма использования

КБП

КП Т.895017.401 ГЧ

Подп. и | Инв.№дубл.Инв | Взам.инв.№В | Подп. и | Инв.№подл.И

Изм	Лист	№ докум.№	Подпись	Дата
Разраб.		Петровский А.А.		
Провер.		Купцова В.Ю.		
Н. Контр.				
Реценз.				
Т. Контр.				
Утверд.				

КП Т.895017.401 ГЧ

ПРОГРАММНОЕ СРЕДСТВО ДЛЯ АВТОМАТИЗАЦИИ НАЧИСЛЕНИЯ ОПЛАТЫ ТРУДА ВОДИТЕЛЯМ ТРАНСПОРТНОЙ КОМПАНИИ

Лит.	Масса	Масштаб
У		

Лист 2 | Листов 5

Диаграмма классов

КБП

КП Т.895017.401 ГЧ

Подп. и

Инв.№дубл.Инв

Взам.инв.№В

Подп. и

Инв.№подл.И

Изм	Лист	№ докум.№	Подпись	Дата
Разраб.		Петровский А.А.		
Провер.		Купцова В.Ю.		
Н. Контр.				
Реценз.				
Т. Контр.				
Утверд.				

КП Т.895017.401 ГЧ

ПРОГРАММНОЕ СРЕДСТВО ДЛЯ АВТОМАТИЗАЦИИ НАЧИСЛЕНИЯ ОПЛАТЫ ТРУДА ВОДИТЕЛЯМ ТРАНСПОРТНОЙ КОМПАНИИ

Лит.	Масса	Масштаб
У		

Лист 3	Листов 5

Диаграмма деятельности

КБП

КП Т.895017.401 ГЧ

Подп. и

Инв.№дубл.Инв

Взам.инв.№В

Подп. и

Инв.№подл.И

Изм	Лист	№ докум.№	Подпись	Дата
Разраб.		Петровский А.А.		
Провер.		Купцова В.Ю.		
Н. Контр.				
Реценз.				
Т. Контр.				
Утверд.				

КП Т.895017.401 ГЧ

ПРОГРАММНОЕ СРЕДСТВО ДЛЯ АВТОМАТИЗАЦИИ НАЧИСЛЕНИЯ ОПЛАТЫ ТРУДА ВОДИТЕЛЯМ ТРАНСПОРТНОЙ КОМПАНИИ

Лит.	Масса	Масштаб
У		

Лист 4	Листов 5

Диаграмма последовательности

КБП

КП Т.895017.401 ГЧ

Подп. и

Инв.№дубл.Инв

Взам.инв.№В

Подп. и

Инв.№подл.И

Изм	Лист	№ докум.№	Подпись	Дата
Разраб.		Петровский А.А.		
Провер.		Купцова В.Ю.		
Н. Контр.				
Реценз.				
Т. Контр.				
Утверд.				

КП Т.895017.401 ГЧ

ПРОГРАММНОЕ СРЕДСТВО ДЛЯ АВТОМАТИЗАЦИИ НАЧИСЛЕНИЯ ОПЛАТЫ ТРУДА ВОДИТЕЛЯМ ТРАНСПОРТНОЙ КОМПАНИИ

Лит.	Масса	Масштаб
У		

Лист 5	Листов 5

Диаграмма компонентов

КБП

Этикетка
для курсовых проектов

Курсовой проект

Тема Программное средство «Эмулятор парковки»

КП Т.895017.401

Разработан _____
Утвержден_____
Разработчик: _____Петровский А.А._____
Руководитель:_____Купцова В.Ю._____
Технические средства ____процессор Intel Core i5-8250U; оперативная память 4048 Мбайт; твёрдотельный накопитель 256 Гбайт_____

Программные средства: __Visual studio 2019; Microsoft SQL Server Management Studio 18.8; .NET Framework 4.7.2___

Состав документа:
Пояснительная записка – файл ПЗ.docx
Программные документы – папка с проектом AutoPay
Графическая часть – файл ПЗ.docx

Удостоверяющий лист
электронного документа – курсовой проект

Тема КП Программное средство «Эмулятор парковки»

Обозначение КП _____КП Т.895017.401_____

Разработчик __Петровский А.А.___ Руководитель ___Купцова В.Ю._____
(Ф.И.О.) (Ф.И.О.)
(Ф.И.О.)

Подписи лиц, ответственных за разработку электронного документа

Состав ЭД	Разработчик	Руководитель
Пояснительная записка и графическая часть (на бумажном носителе формата А4), файл ПЗ.docx		
Папка с проектом AutoPay		
Установочный пакет: setup.exe		
Тип носителя: оптический диск		