Шаблон отчета по производствнной практике

ФИО студента: Гнедич Даниил Денисович

Группа 28 - Б

**Содержание**

Введение 3

Глава 1. Аналитический раздел 6

1.1. Описание индивидуального задания на учебную практику 6

1.2. Исследование предметной области 7

Глава 2. Технические аспекты разработки программы 11

2.1. Логическая модель программы 11

2.2. Выбор языка разработки 17

2.3. Разработка программы 18

Глава 3. Тестирование программы 22

3.1. Выбор методов тестирования 22

3.2. Тестирование приложения 23

3.3. Создание пользовательской документации 25

Заключение 26

Список использованных источников 28

ВВЕДЕНИЕ

Автомобиль является неотъемлемой частью жизни миллионов людей по всему миру. Однако, владение и эксплуатация автомобиля сопряжены с регулярными расходами на техническое обслуживание и ремонт. Для успешного управления финансами, автовладельцам необходим четкий, наглядный и удобный инструмент для оценки предстоящих затрат.

Разработка программного модуля "Калькулятор стоимости ремонта автомобиля" позволяет решить эту проблему. Данный инструмент дает возможность автовладельцам и сотрудникам автосервисов быстро и точно рассчитать стоимость необходимых ремонтных работ, исходя из марки, модели, года выпуска автомобиля, а также характера и объема предстоящих работ.

Калькулятор стоимости ремонта автомобиля предоставляет возможность выбрать марку и модель автомобиля, а также перечень необходимых работ. Например, это может быть ремонт двигателя, замена выхлопной системы, ремонт подвески или другие типы работ. После выбора требуемых параметров, калькулятор производит расчет и выводит ориентировочную стоимость ремонта.

Ключевым преимуществом калькулятора является его универсальность. Он поддерживает широкий спектр марок и моделей автомобилей, что позволяет использовать его для оценки затрат на ремонт практически любого транспортного средства. Благодаря этому, как владельцы автомобилей, так и специалисты автосервисов могут воспользоваться данным инструментом для планирования ремонтных работ и составления бюджета.

Это делает данный инструмент востребованным как для автовладельцев, так и для специалистов автосервисов. Автовладельцы могут заранее планировать бюджет на ремонт, а сотрудники СТО - эффективно управлять затратами и предоставлять клиентам прозрачные и обоснованные расценки.

Однако важно понимать, что калькулятор стоимости ремонта авто предоставляет только ориентировочные данные и не заменяет профессиональную консультацию автомеханика. Фактическая стоимость ремонта может отличаться в зависимости от конкретных обстоятельств, таких как местоположение автосервиса, уровень сложности работ, состояние автомобиля и другие факторы. Поэтому рекомендуется использовать оценку стоимости ремонта как отправную точку для планирования и бюджетирования, но всегда обращаться к специалистам для получения точной информации и консультации.

Всегда стоит обращаться к квалифицированным специалистам для получения точной информации и рекомендаций по ремонту автомобиля.

В процессе подготовки учебной практики был проведен анализ научных статей, публикаций в специализированных журналах и интернет-ресурсах, посвященных вопросам технического обслуживания и ремонта автомобилей. Были изучены современные методики оценки стоимости запчастей, трудоемкости работ, а также факторы, влияющие на итоговую сумму ремонта.

Актуальность данной курсовой работы заключается в разработке программы, которое позволит автовладельцам быстро и удобно рассчитывать стоимость ремонта своего автомобиля, а также выбирать наиболее подходящий автосервис на основе сравнения цен, отзывов и рейтингов.

С развитием технологий и распространением мобильных устройств, потребность в подобных приложениях возрастает. Калькулятор стоимости ремонта автомобиля поможет сэкономить время и деньги автовладельцев, предоставляя им актуальную информацию о стоимости услуг автосервисов и запчастей.

Кроме того, данное приложение может стать полезным инструментом для автосервисов, позволяя им привлекать новых клиентов и повышать свою конкурентоспособность на рынке услуг по ремонту автомобилей.

Целью курсовой работы является разработка программы, а также закрепление практических навыков программирования для Windows в среде разработки Visual Studio.

Для достижения этой цели были поставлены следующие задачи:

1. Изучить существующие решения на рынке.
2. Определить функционал и требования к приложению.
3. Разработать архитектуру и дизайн приложения.
4. Реализовать, протестировать и отладить работу приложения.

Структура курсовой работы обусловлена предметом, целью и задачами исследования. Работа состоит из введения, двух глав, заключения, списка литературы и приложений.

ГЛАВА 1. АНАЛИТИЧЕСКИЙ РАЗДЕЛ.

1.1. **Описание индивидуального задания на учебную практику.**

Название программы: «калькулятор стоимости ремонта автомобиля»

Цель: разработать программный модуль "Калькулятор стоимости ремонта авто" для автосервиса, который позволит клиентам предварительно рассчитать стоимость ремонта их автомобиля.

Задачи:

1. Провести анализ предметной области и выявить основные требования к функциональности калькулятора.
2. Разработать техническое задание на создание программного модуля.
3. Спроектировать пользовательский интерфейс калькулятора, обеспечивающий удобство и простоту использования.
4. Реализовать алгоритмы расчета стоимости ремонта в зависимости от марки, модели, года выпуска автомобиля, а также характера и объема необходимых ремонтных работ.
5. Протестировать работоспособность и точность расчетов программного модуля.
6. Оформить техническую документацию, включающую руководство пользователя.

Содержание задания:

1. Изучение предметной области и требований к программному модулю.
2. Разработка технического задания на создание "Калькулятора стоимости ремонта авто".
3. Проектирование пользовательского интерфейса и архитектуры программного модуля.
4. Реализация алгоритмов расчета стоимости ремонта.
5. Тестирование и отладка программного модуля.
6. Оформление технической документации.

Ожидаемый результат: Функционирующий программный модуль "Калькулятор стоимости ремонта автомобиля" с понятным пользовательским интерфейсом и полной технической документацией.

1.2. **Исследование предметной области.**

Автомобильный ремонт является важной и востребованной услугой, которая необходима владельцам транспортных средств для поддержания их в исправном состоянии. Однако процесс определения стоимости ремонта зачастую бывает затруднительным для клиентов, так как он зависит от множества факторов, таких как тип повреждений, стоимость запчастей и трудозатраты. Для решения этой проблемы разработка программного модуля "Калькулятор стоимости ремонта автомобиля" является актуальной задачей.

Проведенный анализ рынка показал, что на сегодняшний день существует ряд онлайн-калькуляторов и программных приложений, позволяющих рассчитать примерную стоимость ремонта автомобиля. Наиболее популярными примерами являются:

1. Калькулятор ремонта автомобиля на сайте euroauto.ru

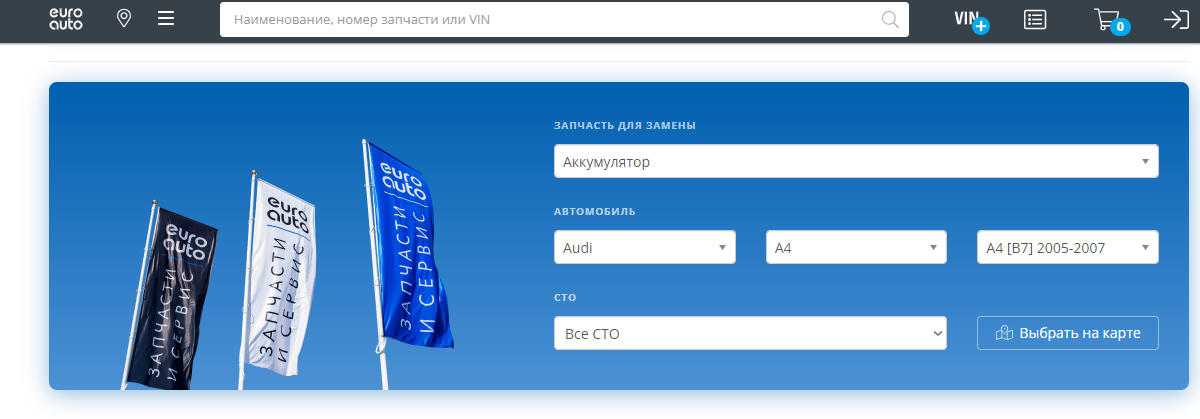


Рисунок 1.2.1. – сайт «euroauto.ru»

1. Калькулятор ремонта на сайте kuzov-russia.ru

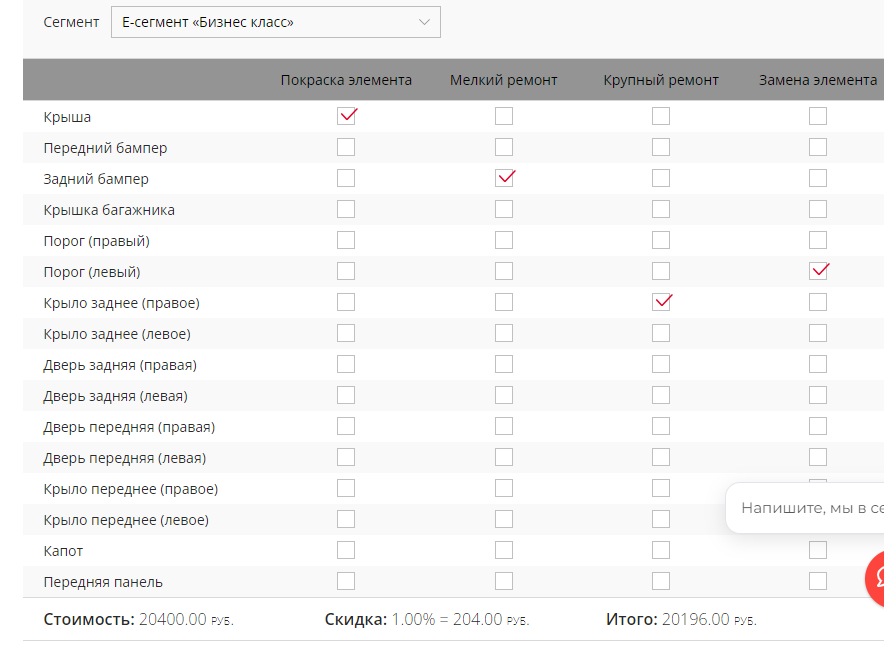


Рисунок 1.2.2. – сайт «kuzov-russia.ru»

1. Калькулятор ремонта автомобиля на сайте ddcar.ru

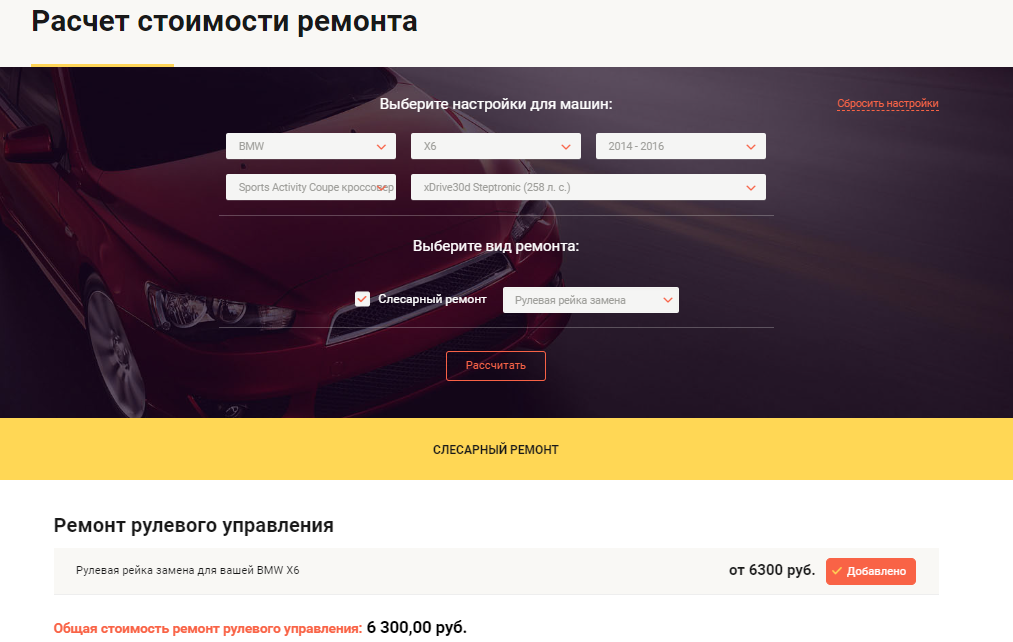


Рисунок 1.2.3. – сайт «ddcar.ru»

Данные решения предоставляют пользователям возможность ввести основную информацию об автомобиле, выбрать характер повреждений, после чего получить примерную оценку стоимости ремонта. Однако большинство из них имеют ограниченный функционал, не учитывают индивидуальные особенности автомобилей, а также не предоставляют детализированную смету расходов.

Основными пользователями программного модуля "Калькулятор стоимости ремонта автомобиля" будут:

1. Владельцы автомобилей, нуждающиеся в оперативной оценке стоимости ремонта.
2. Автосервисы и ремонтные мастерские, заинтересованные в предоставлении клиентам удобного инструмента для расчета стоимости услуг.
3. Автострахователи, использующие калькулятор для оценки ущерба при страховых случаях.

Основными потребностями пользователей являются:

1. Возможность ввода подробной информации об автомобиле и характере повреждений;
2. Получение детальной сметы с разбивкой затрат на запчасти, материалы и трудозатраты;
3. Учет специфики ремонта конкретной марки и модели автомобиля;
4. Интеграция с базами данных запчастей и актуальными прайс-листами;
5. Удобный и интуитивно понятный интерфейс.
6. Функциональные требования

Исходя из потребностей целевой аудитории, основными функциональными требованиями к программному модулю "Калькулятор стоимости ремонта автомобиля" являются:

1. Сбор информации об автомобиле: марка, модель, год выпуска, тип кузова.
2. Ввод данных о характере и локализации повреждений: тип повреждений, затронутые детали.
3. Расчет стоимости ремонта на основе введенных данных с учетом цен на запчасти и трудозатрат.
4. Формирование детальной сметы расходов с разбивкой по категориям.
5. Возможность сохранения рассчитанной сметы в формате PDF.
6. Интеграция с базами данных, содержащими информацию о ценах на запчасти и расценках на ремонтные работы.
7. Кроссплатформенность и возможность использования на различных устройствах.
8. Интуитивно понятный и удобный графический интерфейс пользователя.

Разработка программного модуля "Калькулятор стоимости ремонта автомобиля" является актуальной задачей, которая позволит владельцам транспортных средств, автосервисам и страховым компаниям быстро и точно оценивать затраты на ремонт. Реализация данного модуля должна учитывать потребности целевой аудитории и включать в себя широкий функционал, интеграцию с базами данных, а также удобный интерфейс. Внедрение такого программного решения будет способствовать повышению прозрачности и эффективности процесса ремонта автомобилей.

**ГЛАВА 2. ТЕХНИЧЕСКИЕ АСПЕКТЫ РАЗРАБОТКИ ПРОГРАММЫ.**

2.1. **Логическая модель программы.**

Чтобы нам было проще с проектированием функционала приложения необходимо создать диаграмму, которая будет показывать функции приложения. Ниже приведена диаграмма для нашего калькулятора.

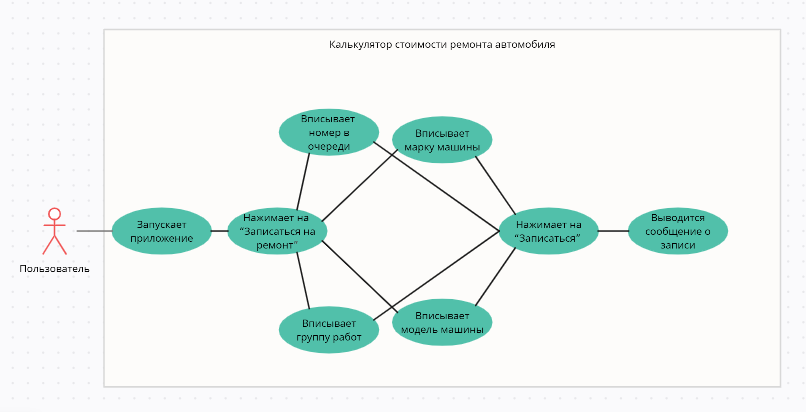


Рисунок 2.1.1. – «UML диаграмма»

Разработка калькулятора стоимости ремонта автомобиля включает несколько технических аспектов, которые могут варьироваться в зависимости от конкретных требований и платформы. Вот некоторые общие технические аспекты, которые могут быть учтены при разработке калькулятора стоимости ремонта авто:

1. Ввод данных: Данный компонент предназначен для сбора информации о состоянии автомобиля и требуемых ремонтных работах. Входные данные могут включать марку и модель автомобиля, год выпуска, пробег, типы неисправностей или ремонтных работ, необходимых для выполнения.
2. Расчет стоимости работ: Этот компонент отвечает за расчет стоимости ремонтных работ на основе введенных данных. Он может использовать базу данных со списком типов работ и соответствующих цен для выполнения расчетов. Кроме того, компонент может применять логику для учета различных факторов, таких как сложность работ, стоимость запасных частей и прочее.
3. Расчет стоимости запасных частей: В случае замены или ремонта деталей автомобиля, этот компонент отвечает за расчет стоимости необходимых запасных частей. Он может использовать базу данных с информацией о запасных частях и их ценах, а также учет факторов, таких как бренд, модель и состояние деталей.
4. Расчет общей стоимости: Данный компонент объединяет результаты расчета стоимости работ и стоимости запасных частей для получения общей стоимости ремонта автомобиля. Он также может учитывать налоги, сборы или дополнительные расходы, связанные с ремонтными работами.
5. Представление результатов: Компонент представления отвечает за отображение результатов расчета пользователю. Это может быть в виде простого текстового вывода или графического интерфейса, который показывает разбивку расчетов по категориям или типам работ. Результаты могут также быть сохранены или отправлены пользователю по электронной почте.

При выборе платформы и технологий для разработки калькулятора стоимости ремонта автомобиля следует учитывать несколько факторов, таких как целевая аудитория, функциональные требования, доступные ресурсы и опыт команды разработчиков. Вот несколько платформ и технологий, которые можно рассмотреть:

1. Веб-приложение:

Веб-приложение является одним из наиболее популярных вариантов для разработки калькулятора стоимости ремонта автомобиля. При выборе технологий для веб-приложения вы можете использовать следующие компоненты:

* Фронтенд: для создания пользовательского интерфейса вы можете использовать HTML для разметки содержимого, CSS для стилей и JavaScript для интерактивности. Фреймворки, такие как React, Angular или Vue.js, могут значительно упростить разработку фронтенда, предоставляя компонентную архитектуру, управление состоянием и другие полезные функции.
* Бэкенд: для обработки бизнес-логики и взаимодействия с базой данных можно использовать различные языки программирования и фреймворки. Например, вы можете выбрать Python с использованием фреймворков Django или Flask, Java с использованием Spring или Kotlin с использованием фреймворка Ktor. Они предоставляют инструменты для создания API, обработки запросов, валидации данных и других задач бэкенда.
* База данных: для хранения информации о работах, запасных частях и других данных вы можете использовать реляционные базы данных, такие как MySQL, PostgreSQL или SQLite. В зависимости от требований, вы также можете рассмотреть использование NoSQL-решений, таких как MongoDB или Firebase Realtime Database.
* Дополнительные инструменты: Вам может потребоваться использование других инструментов, таких как системы контроля версий (например, Git), пакетные менеджеры (например, npm или yarn), среды разработки (например, Visual Studio Code или IntelliJ IDEA) и различные библиотеки и расширения в соответствии с вашими потребностями.

1. Мобильное приложение:

Если вы хотите предоставить пользователю возможность использования калькулятора на мобильных устройствах, вы можете разработать мобильное приложение для платформ Android и iOS. Вот некоторые компоненты, которые могут использоваться при разработке мобильного приложения:

* Android: для разработки под Android вы можете использовать Java или Kotlin вместе с Android SDK. Вы также можете воспользоваться фреймворками, такими как Android Architecture Components, для упрощения разработки приложения, управления жизненным циклом и хранения данных.
* iOS: для разработки под iOS вы можете использовать Swift или Objective-C вместе с iOS SDK. Как и в случае с Android, вы можете использовать фреймворки, такие как SwiftUI или UIKit, для создания пользовательского интерфейса, обработки событий и других задач разработки под iOS.
* Кроссплатформенная разработка: если вам необходимо разработать мобильное приложение одновременно для Android и iOS, вы можете рассмотреть использование фреймворков кроссплатформенной разработки, таких как React Native или Flutter. Эти фреймворки позволяют разрабатывать приложения, используя общий код, который может быть компилирован для обеих платформ.

1. Нативное приложение для, например, Windows, macOS или Linux:

* Windows: для разработки нативных приложений для Windows вы можете использовать язык программирования C# в сочетании с платформой .NET Framework или Universal Windows Platform (UWP). Эти инструменты предоставляют средства для разработки графического интерфейса, обработки событий и взаимодействия с операционной системой Windows.
* macOS: для разработки нативных приложений для macOS вы можете использовать языки программирования Swift или Objective-C вместе с фреймворками Cocoa или SwiftUI. Они предоставляют инструменты для создания пользовательского интерфейса, работы с файловой системой и других функций macOS.
* Linux: для разработки нативных приложений для Linux вы можете использовать языки программирования, такие как C++ или Python, в сочетании с фреймворками, такими как Qt или GTK+. Они предоставляют возможности для создания графического интерфейса, взаимодействия с системой и разработки приложений под Linux.

1. Гибридное приложение:

Гибридное приложение сочетает преимущества веб-разработки с доступом к нативным функциям устройства. Вот некоторые фреймворки, которые можно использовать для разработки гибридных приложений:

* React Native: это фреймворк, основанный на JavaScript, который позволяет разрабатывать гибридные приложения, используя React для создания пользовательского интерфейса и доступа к нативным API устройства.
* Flutter: это фреймворк, разработанный компанией Google, который позволяет создавать кроссплатформенные приложения с использованием языка программирования Dart. Flutter использует свой собственный движок рендеринга, что позволяет достичь высокой производительности и схожести с нативным интерфейсом.

Гибридные приложения обычно имеют один общий кодбазис, который может быть скомпилирован для разных платформ, что упрощает поддержку и развертывание на разных устройствах.

1. Другие факторы:

Помимо выбора платформы и технологий, следует учитывать дополнительные факторы, такие как:

* Интеграция с другими системами: если вы планируете интегрировать калькулятор стоимости ремонта автомобиля с другими системами, например, системами управления ремонтом или базами данных запасных частей, вам может потребоваться выбрать технологии, которые легко интегрируются с существующими системами.
* Безопасность: Разработка калькулятора стоимости ремонта автомобиля может потребовать реализации мер безопасности, таких как защита данных пользователей, обеспечение безопасного взаимодействия с API и защита от взлома приложения. Важно выбрать технологии, которые обеспечивают подходящие механизмы безопасности.
* Масштабируемость: если вы планируете разрабатывать калькулятор стоимости ремонта автомобиля с возможностью масштабирования в будущем.

Калькулятор стоимости ремонта авто может иметь различную функциональность в зависимости от требований проекта и потребностей пользователей. Вот некоторые основные функции, которые можно включить в калькулятор стоимости ремонта авто:

1. Выбор типа ремонта: Калькулятор позволяет выбрать тип ремонта, такой как кузовной ремонт, механический ремонт, электрический ремонт и т.д.
2. Ввод деталей автомобиля: Пользователь должен ввести необходимые данные об автомобиле, такие как марка, модель, год выпуска, VIN-номер и текущий пробег, чтобы калькулятор мог учесть особенности конкретного автомобиля.
3. Оценка стоимости запчастей: Калькулятор может оценить стоимость необходимых запчастей на основе введенных данных о ремонте и модели автомобиля. Это может включать стоимость новых оригинальных запчастей или использованных запчастей.
4. Расчет стоимости работы: Калькулятор может учитывать стандартные нормы времени на выполнение определенных видов ремонта и использовать их для расчета стоимости работы. Расценки на работу могут различаться в зависимости от региона и конкретного автосервиса.
5. Учет дополнительных расходов: кроме стоимости запчастей и работы, калькулятор может учитывать дополнительные расходы, такие как стоимость масла, расходных материалов, налогов или сборов, а также стоимость услуг диагностики и тестирования.
6. Вывод общей стоимости: По завершении расчетов калькулятор предоставляет пользователю общую стоимость ремонта, которая включает стоимость запчастей, работы и дополнительные расходы. Он также может отобразить разбивку стоимости по каждому виду работ или запчасти.
7. Сравнение стоимости: Некоторые калькуляторы могут предоставлять возможность сравнить стоимость ремонта в разных автосервисах или у различных поставщиков запчастей. Это позволяет выбрать наиболее выгодное предложение.
8. Дополнительные функции: В некоторых калькуляторах могут быть включены дополнительные функции, такие как указание рекомендаций по ремонту, предупреждение о потенциальных проблемах, связанных с автомобилем, или информация о стандартных ценах на запчасти и работы.

Важно отметить, что калькулятор стоимости ремонта автомобиля может предоставлять только ориентировочную стоимость, и фактическая стоимость ремонта может различаться в зависимости от многих факторов, таких как регион, сложность ремонта, состояние автомобиля и выбранный автосервис.

2.2. **Выбор языка разработки.**

В ходе выполнения курсовой работы было выявлено, что разработка программы будет удобнее через язык программирования C#. При выборе языка разработки для Windows-приложения у вас есть несколько вариантов. Вот некоторые из самых популярных языков программирования для разработки Windows-приложений:

1. C#: C# является ведущим языком для разработки Windows-приложений благодаря тесной интеграции с платформой .NET от Microsoft. Это объектно-ориентированный, безопасный и управляемый язык, который предоставляет мощные средства для создания широкого спектра приложений, от настольных до мобильных и веб-приложений. Кроме того, C# активно развивается, регулярно получая новые возможности, что делает его привлекательным выбором для современной Windows-разработки.
2. Visual Basic .NET (VB.NET): VB.NET является эволюцией классического Visual Basic и остается популярным языком для разработки Windows-приложений, особенно для более простых и быстрых (RAD) проектов. VB.NET сохраняет знакомый синтаксис Visual Basic, что облегчает переход для разработчиков, привыкших к этому языку. Он также предоставляет широкие возможности для интеграции с Windows API и .NET Framework, делая его отличным выбором для создания традиционных Windows-приложений.
3. C++: C++ является мощным и низкоуровневым языком, который часто используется для разработки высокопроизводительных и системных Windows-приложений. Это особенно актуально для игр, драйверов устройств и приложений, требующих прямого доступа к аппаратному обеспечению. Благодаря своей гибкости и производительности, C++ остается важным инструментом в арсенале Windows-разработчиков, особенно для создания критичных к производительности программ.
4. Java: хотя Java изначально создавалась как кроссплатформенный язык, она также широко используется для разработки Windows-приложений, особенно корпоративных. Ключевые преимущества Java для Windows-разработки включают в себя надежность, безопасность, большой набор библиотек и фреймворков, а также возможность использовать одну и ту же кодовую базу для создания приложений, работающих на различных операционных системах.
5. Python: Python становится все более популярным языком для разработки Windows-приложений благодаря своей простоте, высокой выразительности и наличию большого количества библиотек и инструментов. Python широко используется для создания настольных утилит, аналитических и научных инструментов, скриптов автоматизации и даже некоторых типов корпоративных приложений. Его легкость в освоении и мощные возможности делают Python привлекательным выбором для Windows-разработчиков, особенно для быстрого прототипирования и построения MVP (минимально жизнеспособных продуктов).

Важно учитывать также экосистему, инструменты и ресурсы, связанные с каждым языком, а также поддержку и сообщество разработчиков, чтобы выбрать наиболее подходящий язык для ваших нужд.

2.3. **Разработка программы.**

Начнём с того, что какие элементы будут находится на экране:

1. DataGridView с базой данных.
2. Лист клиентов ListBox.
3. Три кнопки «Найти», «Записаться на ремонт», «Подвести итоги».
4. Поле для поиска textBox.

Теперь рассмотрим каждый элемент по подробнее и по порядку.

Первым элементов для реализации стал DataGridView с базой данных.

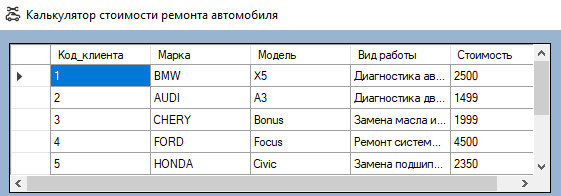


Рисунок 2.3.1. – «Пример элемента DataGridView».

DataGridView — это очень мощный и гибкий элемент управления, предоставляемый Windows Forms в .NET Framework. Его основная функция - отображение данных в табличном формате, при этом он может быть связан с различными источниками данных, такими как массивы, коллекции объектов, базы данных и т.д. DataGridView позволяет пользователям редактировать данные непосредственно в таблице, поддерживая стандартные операции редактирования: добавление, изменение и удаление строк.

Разработчики могут легко настраивать внешний вид DataGridView, включая цвета, шрифты, размеры столбцов, формат отображения данных и многое другое. Кроме того, DataGridView поддерживает встроенные возможности для сортировки данных по одному или нескольким столбцам, а также фильтрацию данных на основе произвольных критериев.

DataGridView предоставляет богатый набор событий, позволяющих разработчикам реагировать на действия пользователя, такие как выделение ячейки, изменение данных, прокрутка и т.д. Он также позволяет использовать пользовательские элементы управления в качестве шаблонов ячеек, давая разработчикам возможность создавать сложные и интерактивные интерфейсы.

Таким образом, DataGridView является мощным и гибким инструментом для отображения и управления табличными данными в Windows-приложениях, предоставляя богатый набор функций для создания интуитивных и функциональных пользовательских интерфейсов.

Вторым элементом для реализации стал лист клиентов, который был реализован через ListBox.

В ListBox будут помещаться клиенты, которые успешно были записаны на ремонт.

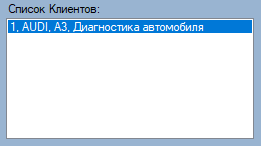


Рисунок 2.3.2. – «Пример элемента ListBox».

ListBox — это элемент управления Windows Forms, который позволяет пользователям выбирать один или несколько элементов из списка. Он широко используется в пользовательских интерфейсах для отображения и взаимодействия с коллекциями данных.

Основные возможности ListBox включают в себя: отображение элементов в вертикальном списке, поддержку множественного выбора (если это необходимо), возможность прокрутки списка при большом количестве элементов, события для отслеживания выбранных элементов и изменений в списке. Разработчики могут легко привязывать ListBox к различным источникам данных, таким как массивы, списки или коллекции объектов, что упрощает процесс наполнения списка.

Далее идет реализация кнопок «Найти», «Записаться на ремонт», «Подвести итоги», которые были реализованы через элемент Button.

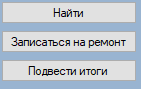


Рисунок 2.3.3. – «Пример элемента Button».

Button - это один из базовых элементов управления в Windows Forms, который позволяет пользователям выполнять различные действия, нажимая на него. Будучи простым в использовании, Button предоставляет разработчикам широкий набор возможностей для создания интерактивных и функциональных пользовательских интерфейсов.

Основные характеристики Button включают в себя возможность отображения текста, изображения или их комбинации, настройку размеров и положения кнопки на форме, применение различных стилей оформления, таких как цвета, шрифты и темы. Button также поддерживает ряд стандартных состояний, например, обычное, наведение курсора, нажатие, отключено, которые позволяют визуально отображать текущее состояние кнопки.

Последним элементом в Form1 будет поле для поиска, которое было реализовано через textBox.



Рисунок 2.3.4. – «Пример элемента textbox».

TextBox - это широко используемый элемент управления в Windows Forms, который позволяет пользователям вводить и редактировать текстовую информацию. Он играет ключевую роль в создании интерактивных и функциональных пользовательских интерфейсов, предоставляя разработчикам широкие возможности по управлению текстовыми данными.

Основные возможности TextBox включают в себя: отображение и редактирование однострочного или многострочного текста, поддержку различных текстовых форматов, таких как обычный текст, пароли, денежные значения, даты и другие, возможность задания начального текста, ограничений на длину вводимых данных и многое другое. TextBox также предоставляет события, которые позволяют отслеживать изменения в тексте, нажатие клавиш, фокусировку и другие взаимодействия пользователя.

**ГЛАВА 3. ТЕСТИРОВАНИЕ ПРОГРАММЫ.**

3.1. **Выбор методов тестирования.**

После разработки приложения нужно убедится в его работоспособности, и провести тестирование для обнаружения неисправностей. Для этого был представлен список существующих методов тестирования:

Метод черного ящика (Black Box Testing): Этот метод тестирования основан на проверке входных и выходных данных без знания внутренней реализации приложения. Тестировщик рассматривает приложение как "черный ящик" и проверяет его функциональность, интерфейс и поведение на основе ожидаемых результатов.

Метод белого ящика (White Box Testing): В этом методе тестирования тестировщик имеет полное знание о внутренней структуре и коде приложения. Тестировщик может напрямую проверять состояние переменных, ветвления кода и структуру приложения для обеспечения полного покрытия тестами.

Метод модульного тестирования (Unit Testing): Этот метод тестирования сосредоточен на отдельных модулях или компонентах приложения. Тестировщик создает и запускает тесты для проверки правильности работы отдельных функций, классов или методов в изоляции от других компонентов.

Метод интеграционного тестирования (Integration Testing): В этом методе тестирования проверяется взаимодействие между разными компонентами приложения. Тестировщик проверяет, что различные модули работают корректно вместе, обмениваясь данными и взаимодействуя друг с другом.

Автоматизированное тестирование (Automated Testing): В этом методе тестирования используются средства автоматизации для создания и выполнения тестовых сценариев. Можно использовать фреймворки для автоматического тестирования, такие как Espresso, UI Automator или Appium, для создания скриптов и автоматизации выполнения тестов на устройствах.

Это лишь некоторые из методов тестирования. Выбор конкретных методов зависит от требований, характеристик и целей вашего приложения. Комбинирование различных методов тестирования может помочь обеспечить более полное и надежное тестирование приложения перед его выпуском.

3.2. **Тестирование приложения.**

Для данного модуля был выбран метод ручного тестирования, так как он позволяет проверять внутреннюю структуру и логику программы, тестируя различные компоненты и их взаимодействие.

Преимущества ручного тестирования для данной программы:

1. Глубокое понимание внутренней реализации:

* Ручное тестирование позволяет детально изучить внутреннюю структуру и логику модуля, чего нельзя добиться при автоматизированном тестировании.
* Это дает возможность выявить скрытые ошибки и неожиданные взаимодействия между компонентами модуля.

1. Гибкость и адаптивность:

* При ручном тестировании тестировщик может быстро реагировать на изменения в требованиях или реализации модуля.
* Он может легко адаптировать тестовые сценарии и случаи использования к новым условиям.

1. Комплексная проверка:

* Ручное тестирование позволяет проверять сложные взаимодействия между различными компонентами модуля.
* Тестировщик может моделировать реальные сценарии использования и оценивать целостность работы системы.

1. Улучшение понимания предметной области:

* Во время ручного тестирования тестировщик глубже погружается в предметную область и лучше понимает бизнес-логику.
* Это знание можно использовать для улучшения дизайна и функциональности модуля.

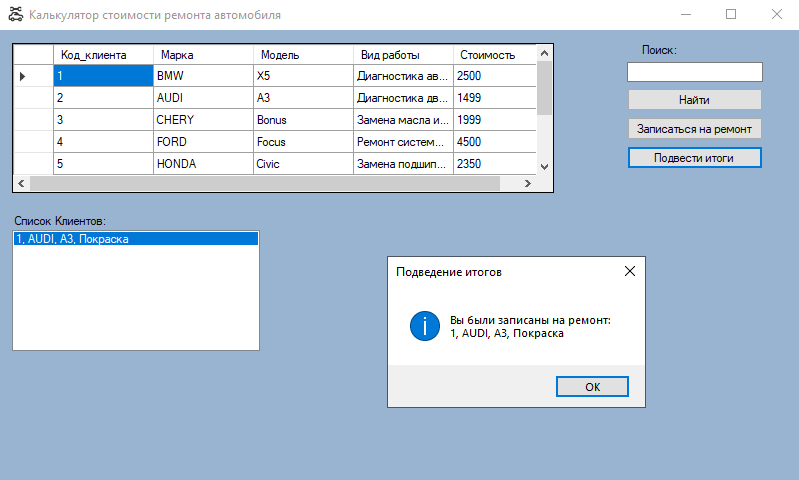
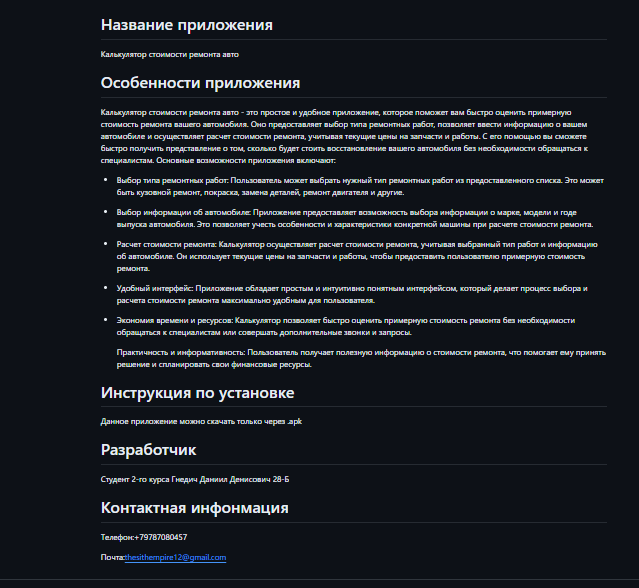


Рисунок 3.2.1. – «Итог ручного тестирования».

Итог тестирования: приложение работает без ошибок, не выдает ошибок и не "вылетает", все компоненты функционируют корректно.

3.3. **Создание пользовательской документации.**



3.3.1. – «Пользовательская документация».

Исходя из итогов тестирования можно сказать, что приложение работает стабильно и без выявленных ошибок.

**ЗАКЛЮЧЕНИЕ**

Калькулятор стоимости ремонта автомобиля является ценным инструментом для автовладельцев и автосервисов. Он позволяет быстро и удобно оценить предстоящие расходы на ремонт, что помогает правильно спланировать бюджет и принять взвешенные решения. Ключевыми преимуществами этого инструмента являются его универсальность, простота использования, а также возможность настройки различных параметров для получения наиболее точной оценки.

Для автовладельцев калькулятор помогает снизить тревогу относительно стоимости ремонта, а для автосервисов он способствует установлению доверительных отношений с клиентами и упрощает процесс планирования и оценки затрат. В целом, использование калькулятора стоимости ремонта авто является неоценимым подспорьем как для частных автовладельцев, так и для профессиональных автосервисов.

Кроме своей основной функции оценки затрат на ремонт, этот инструмент также имеет ряд дополнительных преимуществ. Он позволяет автовладельцам сравнивать стоимость ремонта в разных автосервисах, что помогает найти наиболее выгодное предложение. Это особенно полезно, когда требуется замена дорогостоящих деталей или проведение сложных ремонтных работ.

Для автосервисов калькулятор является незаменимым помощником в управлении бизнесом. Он помогает вести точный учет расходов на запчасти, материалы и трудозатраты, что позволяет формировать более обоснованные цены на услуги. Кроме того, сервисы могут использовать калькулятор для создания прайс-листов и коммерческих предложений, что повышает прозрачность и доверие клиентов.

Важно отметить, что многие современные калькуляторы имеют возможность интеграции с учетными системами автосервисов, упрощая процессы ведения документации и бухгалтерского учета.

Это позволяет оптимизировать работу сотрудников, сокращая время на подготовку сметных расчетов и счетов для клиентов.

В целом, использование калькулятора стоимости ремонта автомобиля представляет собой эффективный способ управления расходами, как для частных владельцев, так и для профессиональных автосервисов. Он помогает принимать взвешенные решения, повышает прозрачность ценообразования и способствует построению доверительных отношений между клиентами и автосервисами.

В целом, приложение калькулятора стоимости ремонта автомобиля является полезным инструментом, который помогает автовладельцам планировать и оценивать затраты на обслуживание и ремонт своих автомобилей. Однако, необходимо учитывать его приближенный характер и принимать во внимание индивидуальные особенности каждого случая. Однако оно не заменяет профессиональную оценку и советы специалистов.

В ходе курсовой работы были достигнуты следующие задачи:

1. Изучить существующие решения на рынке.
2. Определить функционал и требования к приложению.
3. Разработать архитектуру и дизайн приложения.
4. Реализовать, протестировать и отладить работу приложения.

**СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ**

1. Петров И.И. «Проектирование и разработка мобильных приложений.» – М.: Издательский дом «Питер», 2018.
2. Луценко С.П., Смирнов И.В. «Мобильная разработка на Android.» – СПб.: БХВ-Петербург, 2019.
3. Хантер Р. «Мобильная разработка для начинающих.» – М.: ДМК Пресс, 2017.
4. Сергеев В.В., Пантелеев В.В. «Мобильная разработка под iOS на языке Swift.» – М.: ДМК Пресс, 2019.
5. Кузнецов А.А., Попов А.В. «Мобильная разработка на платформе Android.» – М.: ДМК Пресс, 2018.
6. Смирнов А.А. «Программирование мобильных приложений.» – М.: Бином, 2019.
7. Красильников Ю.Н. «Разработка мобильных приложений с использованием Xamarin.Forms.» – М.: Издательство Московского университета, 2020.
8. Белоусов Д.М. «Программирование мобильных приложений на платформе iOS.» – М.: Издательство «Лори», 2018.
9. Попов А.А. «Разработка мобильных приложений на платформе Android.» – М.: БХВ-Петербург, 2020.