Министерство образования Московской области

Государственное образовательное учреждение высшего образования Московской области

«Государственный гуманитарно-технологический университет»

Ликино-Дулевский политехнический колледж – филиал ГГТУ

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

Дипломный проект

Разработка приложения по автоматизации процесса работы с потребителями для АО «НПП «Респиратор».

Выполнил:

Маркин Илья Алексеевич \_\_\_\_\_\_\_\_\_\_\_

студент группы ИСП.18.2А\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

по специальности

09.02.07 Информационные системы и программирование

очной формы обучения

Руководитель:

Кузьмина Елена Евгеньевна \_\_\_\_\_\_\_\_\_

Оценка:

\_\_\_\_\_\_\_\_\_/\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ликино-Дулево

2022 год

Оглавление

[Введение 3](#_Toc106021366)

[1. Разработка системного проекта 6](#_Toc106021367)

[1.1 Назначение разработки 6](#_Toc106021368)

[1.2 Требования к функциональным характеристикам 6](#_Toc106021369)

[1.2.1 Состав выполняемых функций 6](#_Toc106021370)

[1.3 Требование к надежности и безопасности 7](#_Toc106021371)

[1.4 Требования к составу и параметрам технических средств 7](#_Toc106021372)

[1.5 Требования к информационной и программной совместимости 8](#_Toc106021373)

[2. Разработка технического проекта 9](#_Toc106021374)

[2.1 Обоснование выбора CASE-средств 9](#_Toc106021375)

[2.2 Проектирование модели данных 10](#_Toc106021376)

[2.3 Детальное проектирование программного приложения 12](#_Toc106021377)

[2.4 Функциональная схема программного приложения 15](#_Toc106021378)

[3. Реализация 16](#_Toc106021379)

[3.1 Обоснование выбора средств разработки 16](#_Toc106021380)

[3.2 Руководства программиста 17](#_Toc106021381)

[4. Тестирование и отладка 38](#_Toc106021382)

[4.1 Виды тестирования 38](#_Toc106021383)

[4.2 Отладка программного приложения 40](#_Toc106021384)

[5. Расчет базовой себестоимости разрабатываемого продукта 42](#_Toc106021385)

[5.1 Исходные данные 42](#_Toc106021386)

[5.2 Расчёты затрат на выполнение программы 42](#_Toc106021387)

[5.3 Расчет отчислений на социальное страхование и обеспечение 43](#_Toc106021388)

[5.4 Расчет базовой себестоимости компьютерного продукта 43](#_Toc106021389)

[Заключение 46](#_Toc106021390)

[Список литературы 48](#_Toc106021391)

[Приложения 50](#_Toc106021392)

# Введение

Отношения между заказчиком или потребителем, организациями, предоставляющими продукцию и услуги, никогда еще не были настолько ориентированы на потребителей и не рассматривались как наиболее важные для достижения успеха организации. Профессионалы в сфере предоставления услуг, которые или непосредственно работают с потребителем, такие как продавцы, операторы центров телефонного обслуживания или менеджеры по обслуживанию клиентов, или управляют этим процессом, такие как менеджеры операционных центров или руководители, ответственные за обслуживание потребителей - никогда еще не играли столь важную роль в работе организаций.

Ведение претензионной работы на предприятии или в учреждении, организованное и отрегулированное соответствующим положением, способствует не только оперативному и обоснованному рассмотрению и направлению требований о выполнении, предусмотренных договором или законом обязательств, но и грамотной финансовой политике организации в целом.

Рекламация – это обращение клиента, связанное с неудовлетворенностью приобретенным товаром или услугой. Для компании рекламация клиента – это независимый взгляд на ее продукцию и работу ее сотрудников. Бывает трех типов: обоснованная (ошибка персонала, брак товара, сбой системы и пр.), необоснованная (правила использования товара не изучены, неправильные действия потребителя и пр.), провокационная — используется для получения бонусов, скидок, выплат заявителем.

Отработка рекламаций – процесс, включающий в себя:

* Прием жалобы (определение типа жалобы)
* Анализ жалобы (определение важности и срочности)
* Предоставление первичного ответа клиенту
* Анализ ситуации (регистрация заявления, назначение срока ответа)
* Формулировка способа разрешения конфликта
* Внесение коррективов в товар или предоставляемую услугу

Актуальность и значимость рассматриваемой проблемы работы с потребителями заключается в том, что ведение претензионной работы на предприятии является одним из важнейших залогов производства качественного товара и повышения рейтинга доверия предприятия.

Автоматизация этого процесса позволит предприятию избегать потерю времени на отслеживание выполнения работ по исправлению несоответствующего продукта и повысит эффективность распределения этих работ.

Цель исследования – проанализировать бизнес-процессы, происходящие в организации работы с потребителями, собрать материал теоретической и практической направленности для формирования банка данных, обработки и хранения поступающей информации, разработать приложение, которое автоматизирует рабочие процессы сотрудника розничной торговле.

Задачи:

1. Изучение литературы, относящейся к предметной области;
2. Формирование требований к разрабатываемому программному продукту;
3. Разработка программного продукта;
4. Расчет базовой стоимости программного продукта;
5. Разработка полной и понятной документации руководства пользователя и программиста.

Методы:

* Текстовый процессор Microsoft Word – ведение отчетности
* Visual Studio 2022 – создание интерфейса приложения и его логики
* СУБД Microsoft SQL Server и утилита SQL Server Management Studio – создание, заполнение и администрирование базы данных
* Microsoft PowerPoint – проектирование интерфейса приложения
* Microsoft Excel - ведение расчетов экономической части.
* Microsoft Visio – уточнение бизнес-процессов проекта и составления диаграммы действий, а также проектирование базы данных

Результатом работы является рабочее приложение по работе с потребителем.

Предлагаемая работа состоит из введения, пяти разделов, заключения и приложения. Во введении обосновывается актуальность темы, формулируются цель, задачи и практическая значимость работы.

Первый раздел посвящен описанию разработки, составу выполняемых функций, а также техническим и программным требованиям.

Второй раздел является разработкой технического проекта в Case-средствах. Описаны программы для разработки, созданы схемы в diagrams.net и Visio, определены бизнес-процессы. Разработан детальный проект программного обеспечения, описывающий алгоритм работы программы, а также визуальный интерфейс.

Третий раздел описывает этапы реализации программного продукта. Показаны таблицы базы данных, написаны руководство программиста и руководство пользователя, описывающие программу, методы обращения к программному продукту.

Четвертый раздел посвящен тестированию и отладке программного продукта. Проведено интеграционное тестирование, показаны ошибки, которые возникали при разработке приложения.

Пятый раздел описывает расчет базовой стоимости продукта. Произведены расчет затрат на выполнение задачи, электроэнергии, прочих затрат, расчет отчислений в пенсионный фонд.

В заключении описываются основные результаты выпускной квалификационной работы.

Приложение содержит пустой бланк рекламации, пример отчета, а также сценарии тестирования приложения.

Общий объем работы 52 страницы. Список литературы содержит 17 источников.

# Разработка системного проекта

## Назначение разработки

Автоматизированная информационная система «Работа с потребителем» предназначена для управления заявками на несоответствие от потребителей и сбора статистики по задачам. Пользователем программы выступает специалист по работе с рекламациями. Работа с рекламациями ведется согласно регламенту по работе с претензиями клиентов. При регистрации рекламации указывается дата регистрации, инициатор (должность, организация, ФИО), содержание обращения, устанавливается важность, определяются сроки для устранения несоответствия. Далее рекламация разбивается на задачи. При составлении задачи необходимо указать название, которое описывает действия для решения проблемы, сроки выполнения задачи, а также назначить исполнителя (отдел, ФИО ответственного). После завершения всех задач рекламация считается завершенной, формируется результирующий отчет по устранению несоответствий в рекламации для отправки поставщику. В отчете указывается номер заявки, ее название, дата регистрации и закрытия заявки, заметки к рекламации (при наличии), установленная важность, инициатор, перечислены все задачи (изложение задачи, ее исполнитель и дата назначения), общее количество затрат на устранение данного несоответствия.

## Требования к функциональным характеристикам

### Состав выполняемых функций

Автоматизированная информационная система должная обеспечивать выполнение следующих функций:

* Регистрация рекламации при обращении клиента;
* Назначение задач для решения несоответствия;
* Расчет итоговых затрат на устранение несоответствия;
* Формирование результирующего отчета по устранению несоответствий в рекламации для отправки поставщику;
* Визуальное отображение состояния заявок;
* Отображение статистики задач по исполнителям;
* Архивация закрытых претензий;
* Отображение затрат на устранение несоответствий за период.

## Требование к надежности и безопасности

Разрабатываемое программное обеспечение должно стабильно функционировать при бесперебойной работе компьютера пользователя.

* При возникновении ошибок или сбоев в работе компьютера восстановление стабильной работы программы должно производиться после полной перезагрузки операционной системы, запуска исполняемого файла. При условии, если пользователь до сбоя работы сохранил внесенные данные, в таком случае данные сохранятся в базе данных.
* Для защиты информации на компьютере пользователя должны быть предусмотрены необходимые меры: пароль на вход в компьютер, антивирусные программы, отсутствие на компьютере подозрительных программ, полученных с неофициальных источников.
* При передаче экземпляра ПО, использовать безопасные методы передачи информации.

## Требования к составу и параметрам технических средств

Минимальные системные требования для работы программного продукта должны быть следующими: частота процессора – 3 Ггц, объем оперативной памяти 4 Гб, объем свободного дискового пространства 500 Мб, разрешение монитора 1366х768. Для печати заказа необходим принтер.

|  |  |
| --- | --- |
| **Конфигурация компьютера** | |
| Процессор | AMD Ryzen 5 2400G 3.6 ГГц |
| Память | 15 гб |
| Разрешение экрана | 1920x1080 |
| Устройства ввода | Клавиатура, мышь |
| Дисковое пространство | 1 гб |

## Требования к информационной и программной совместимости

Для корректной работы программы необходимо: ОС Windows 10 64х, установленный дистрибутив .Net Framework 4.8.

MS Word 2019 – для ведения хода разработки и составления отчета для подведения итога работоспособности программы.

MS SQL[[1]](#footnote-1) Server Management Studio 18 - утилита для конфигурирования, управления и администрирования всех компонентов Microsoft SQL Server. Утилита включает скриптовый редактор и графическую программу, которая работает с объектами и настройками сервера.

MS SQL Server 2019 – система управления реляционными базами данных (РСУБД). Основной используемый язык запросов – Transact-SQL. Используется для работы с базами данных размером от персональных до крупных баз данных.

Visual Studio 2022 – это стартовая площадка для написания, отладки и сборки кода, а также последующей публикации приложений, представляет собой многофункциональную программу, которую можно использовать для различных аспектов разработки программного обеспечения.

ADO.NET – предоставляет собой технологию работы с данными, которая основана на платформе .NET Framework. Эта технология представляет набор классов, через которые можно отправлять запросы к базам данных, устанавливать подключения, получать ответ от базы данных и др.

ADO.NET Entity Framework – это набор технологий в ADO.NET, которые поддерживают разработку программных приложений, ориентированных на данные. Предоставляет возможность взаимодействия с объектами как посредством LINQ[[2]](#footnote-2) в виде LINQ to Entities, так и с использованием Entity SQL.

# Разработка технического проекта

## Обоснование выбора CASE-средств

CASE средства (Computer – Aided Software Engineering) – это набор инструментов и методов программной инженерии, который позволяет автоматизировать процесс разработки информационной системы и программного обеспечения. Разработка и создание информационных систем управления предприятием связаны с выделением бизнес-процессов, их анализом, определением взаимосвязи элементов процессов, оптимизации их инфраструктуры и т.д. Основной целью применения CASE средств является сокращение времени и затрат на разработку информационных систем, и повышение их качества.

UML (унифицированный язык моделирования) — язык графического описания для объектного моделирования в области разработки программного обеспечения, для моделирования бизнес-процессов, системного проектирования и отображения организационных структур. UML является языком широкого профиля, это — открытый стандарт, использующий графические обозначения для создания абстрактной модели системы, называемой UML-моделью.

Для данного проекта выбрано бесплатное, кроссплатформенное графическое приложение с открытым кодом diagrams.net – профессиональным инструментом построения диаграмм. С его помощью можно создавать такие диаграммы, как блок-схемы, схемы страниц, UML-диаграммы, организационные схемы и сетевые диаграммы.

Данное ПО имеет следующие преимущества:

* Бесплатный онлайн доступ
* Связь с такими сервисами как Google диск, Github, Dropbox и OneDrive.

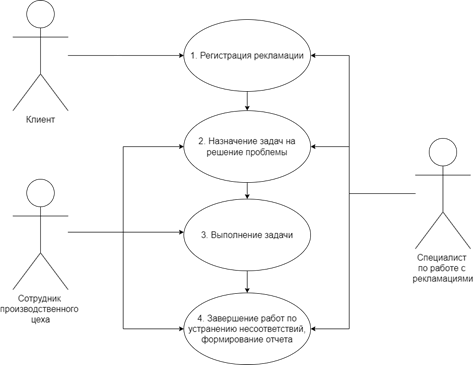
Диаграмму можно сохранить как растровое или векторное изображение, как XML-файл или HTML-файл

## Проектирование модели данных

Диаграмма прецедентов – диаграмма, отражающая отношения между акторами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне.

Актор – множество логически связанных ролей, исполняемых при взаимодействии с прецедентами. Актором может быть человек или другая система, подсистема или класс, которые представляют нечто вне сущности.

Прецедент – спецификация последовательностей действий, описывает некоторый целостный фрагмент поведения системы, не вдаваясь при этом в особенности внутренней структуры субъекта.



1. «Диаграмма прецедентов»

Далее описанные бизнес-процессы подлежат кодировке для удобства дальнейшего применения.

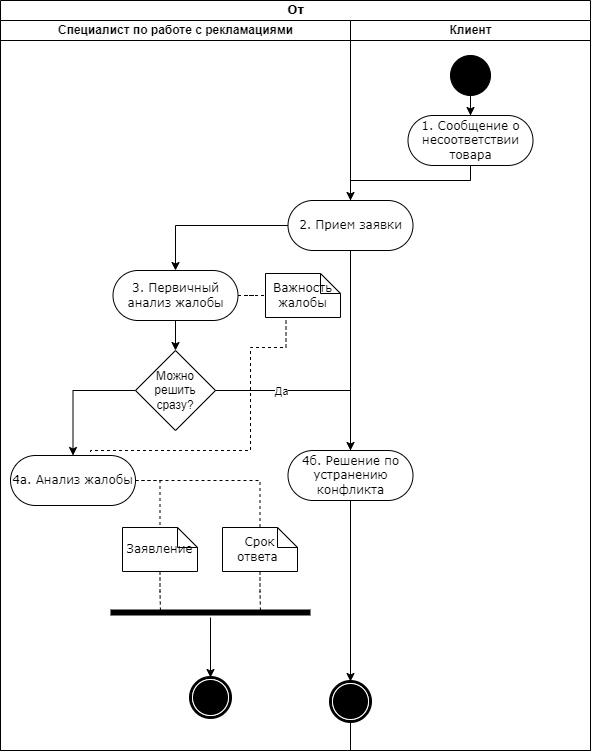
Таблица №1 «Определение бизнес-процессов»

|  |  |  |
| --- | --- | --- |
| Номер бизнес-процесса | Код бизнес-процесса | Наименование бизнес-процесса |
| 1. | Регистр\_рекл | Регистрация рекламации |
| 2. | Назнач\_задач | Назначение задач на решение проблемы |
| 3. | Выполн\_задач | Выполнение задачи |
| 4. | Форм\_отчет | Завершение работ по устранению несоответствий, формирование отчета |

Описание бизнес-процесса — это пошаговое описание действий работников при выполнении той или иной операции, включая ответственность, порядок принятия решений, порядок взаимодействия с другими сотрудниками

Выбран бизнес-процесс 1.Регистр\_рекл и сформулирован словесный алгоритм:

1. После обнаружения несоответствия в приобретенной продукции (поломка, брак и т.п.), клиент приходит в офис специалиста по работе с рекламациями и сообщает о несоответствии товара.
2. После выслушивания сообщения, специалист задает клиенту уточняющие вопросы и выясняет детали несоответствия товара.
3. Далее специалист проводит первичный анализ жалобы и выясняет ее важности.
4. При возможности решить проблему на месте, специалист устраняет проблему, в случае невозможности найти решение, специалист анализирует жалобу, составляет заявление и определяет сроки ответа



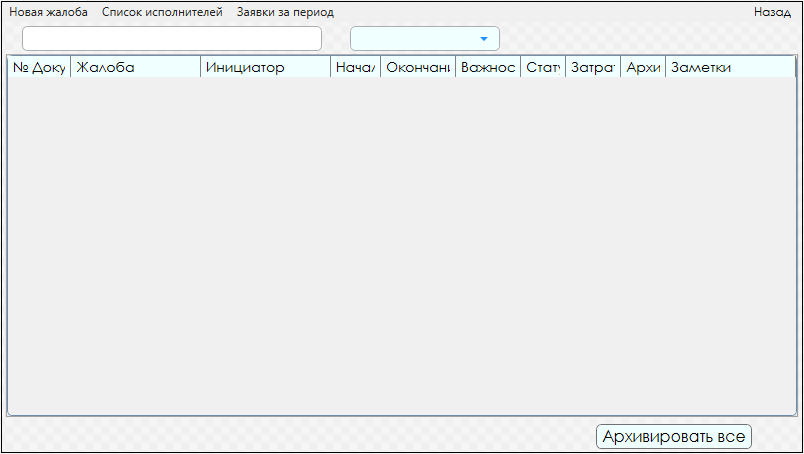
1. «Диаграмма действий»

Таблица №2 «Операция регистрация рекламации»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Диаграмма и номер операции на диаграмме | Операция | Как часто | Исполнитель | Входящие документы | Исходящие документы |
| 1. | 2. | 3. | 4. | 5. | 6. |
| 1. Регистр \_Рекл | 1. Сообщение о несоответствии товара | При обнаружении несоответствия | Клиент | Нет | Нет |
| 1. Регистр \_Рекл | 2. Прием заявки | При сообщении клиента | Специалист по работе с рекламациями  Клиент | Нет | Нет |
| 1. Регистр \_Рекл | 3. Первичный анализ жалобы | После принятия заявки | Специалист по работе с рекламациями | Нет | Важность жалобы |
| 1. Регистр \_Рекл | 4а. Анализ жалобы | При возможности найти решение сразу | Специалист по работе с рекламациями | Важность жалобы | Заявление  Срок ответа |
| 1. Регистр \_Рекл | 4б. Решение по устранению конфликта | При отсутствии моментального решения | Специалист по работе с рекламациями  Клиент | Нет | Нет |

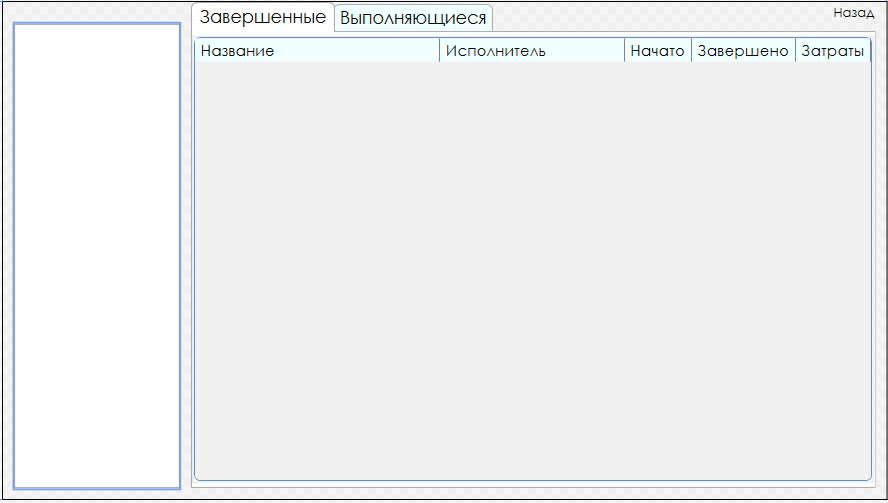
## Детальное проектирование программного приложения

Главная страница, отображает рекламации, с помощью нее осуществляется переход на другие страницы, а также на окно рекламации и окно добавления новой рекламации.

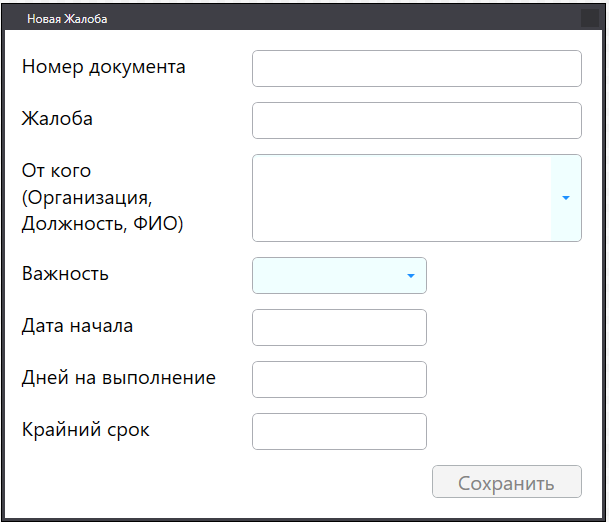


1. «Конструктор основной страницы»

На странице статистики по исполнителям при выборе исполнителя отображаются задачи, которые они выполнили или выполняют.

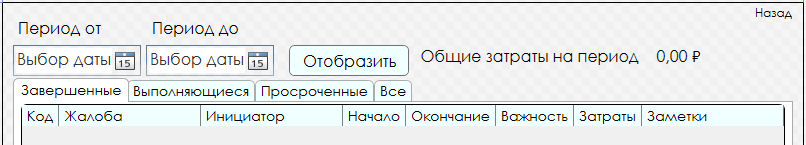


1. «Конструктор страницы «Статистика Исполнителей»»



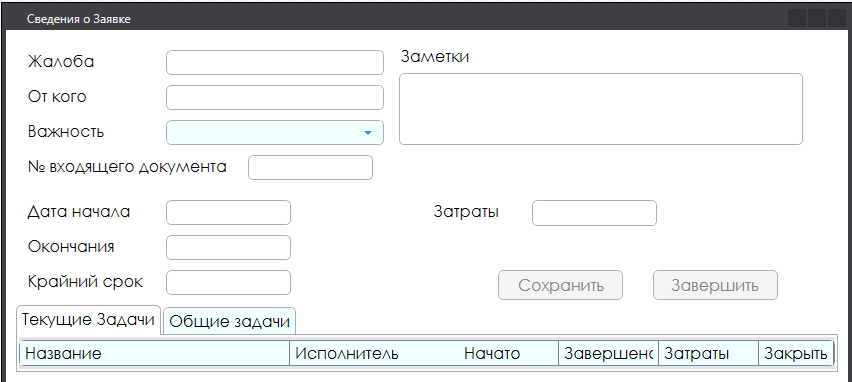
1. «Конструктор окна добавления новой рекламации»

На странице статистики за период отображаются рекламации за указанный период.

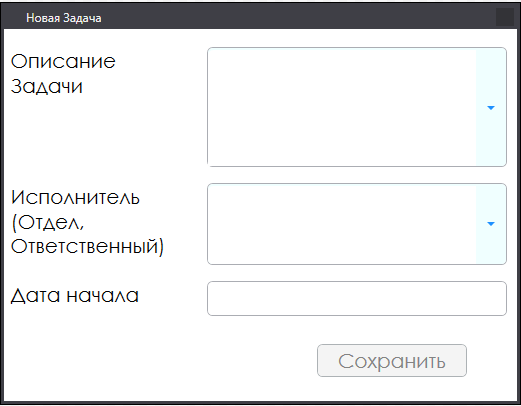


1. «Конструктор страницы «Статистика за период»»

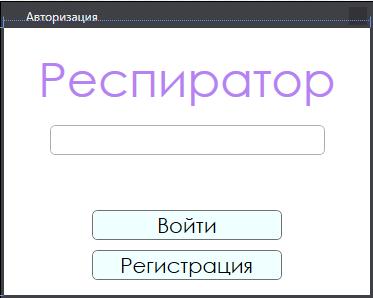
На окне информации о рекламации отображается вся информация о рекламации, добавляются задачи для устранения несоответствий.



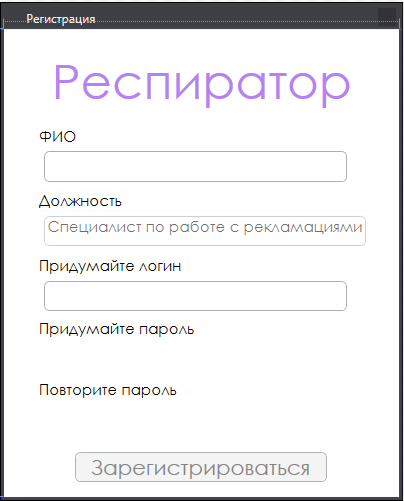
1. «Конструктор страницы информации о рекламации»



1. «Конструктор окна добавления задачи»



1. «Конструктор окна входа»



1. «Конструктор окна регистрации»

## Функциональная схема программного приложения



1. «Функциональная схема приложения»

# Реализация

## Обоснование выбора средств разработки

Для написания логики и интерфейса приложения использовались язык программирования C#, расширяемый язык разметки XAML, среда разработки Visual Studio 2022.

Объектно-ориентированный язык программирования C#. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, переменные, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML[[3]](#footnote-3).

Расширяемый язык разметки для приложений XAML. Создавать элементы можно без применения XAML, обходясь только кодом на C#, но с помощью XAML это становится проще, появляется разграничение логики приложения и графического отображения.

Интегрированная среда разработки программного обеспечения Microsoft Visual Studio 2022 Community. Данная среда разработки обладает следующими преимуществами: встроенный контроль за выполнением многопоточного кода, бесплатность, рефакторинг и реструктуризация исходных текстов программ.

Для создания базы данных использовались РСУБД SQL Server Express 2019, и среда управления инфраструктурой SQL - SQL Server Management Studio.

Система управления реляционными базами данных SQL Server. Данная РСУБД обладает следующими преимуществами: высокоскоростной доступ к данным, обеспечиваемый надежной клиент-серверной архитектурой СУБД; безопасность хранения информации в БД - благодаря возможности шифрования данных и резервного копирования; удобство подключения к соединению ADO.NET.

SQL Server Management Studio 18 - интегрированная среда для управления любой инфраструктурой SQL. Используется для доступа, настройки, администрирования и разработки всех компонентов SQL Server, Базы данных SQL Azure и Azure Synapse Analytics, а также управления ими

## Руководства программиста

|  |  |
| --- | --- |
| **Минимальные требования** | |
| Процессор | Intel Core i3 3240 3,4 ГГц |
| Память | От 4 гб |
| Разрешение экрана | От 1366х768 |
| Устройства ввода | Клавиатура, мышь |
| Дисковое пространство | От 1 Гб |
| **Оптимальные требования** | |
| Процессор | AMD Ryzen 5 2400G 3.6 ГГц |
| Память | От 8 гб |
| Разрешение экрана | 1920x1080 |
| Устройства ввода | Клавиатура, мышь |
| Дисковое пространство | От 1-4 гб |

**Требования к информационной и программной совместимости**

Для корректной работы программы необходимо: **ОС Windows 10 64х**, установленный дистрибутив **.Net Framework 4.8.**

**MS Word 2019** – для ведения хода разработки и составления отчета для подведения итога работоспособности программы.

**MS SQL Server Management Studio 18** - утилита из Microsoft SQL Server 2005 и более поздних версий для конфигурирования, управления и администрирования всех компонентов Microsoft SQL Server. Утилита включает скриптовый редактор и графическую программу, которая работает с объектами и настройками сервера.

**MS SQL Server 2019** - система управления реляционными базами данных (РСУБД), 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.

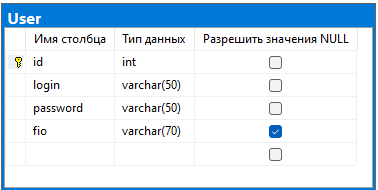
**Visual Studio 2022** – продукт включающая интегрированную среду разработки программного обеспечения и ряд других инструментальных инструментов. Нужна для реализации основного кода программы PROORG

**ADO.NET** - технология, предоставляющая доступ и управление данными, хранящимся в базе данных или других источниках.

**ADO.NET Entity Framework** - объектно-ориентированная технология доступа к данным. Предоставляет возможность взаимодействия с объектами как посредством LINQ в виде LINQ to Entities, так и с использованием Entity SQL. Для облегчения построения web-решений используется как ADO.NET Data Services, так и связка из Windows Communication Foundation и Windows Presentation Foundation, позволяющая строить многоуровневые приложения.

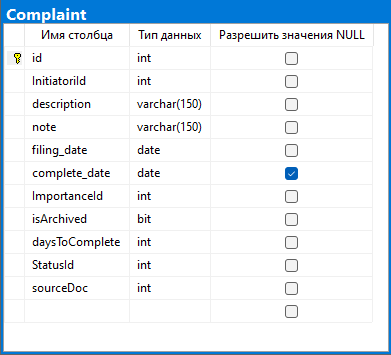
**Проектирование и разработка баз данных**

Таблица пользователей программы:



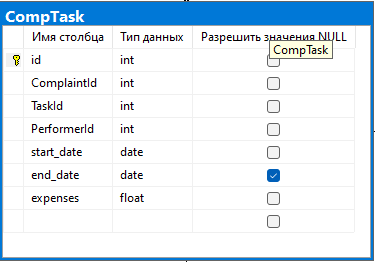
1. «Пользователь»

Таблица заявок на несоответствие:



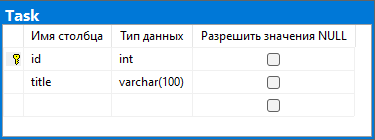
1. «Рекламация»

Таблица назначенных задач на выполнение рекламации:

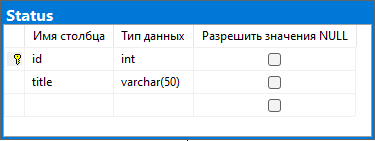


1. «Задачи на Рекламацию»

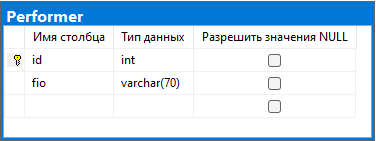
Созданы Справочные таблицы:



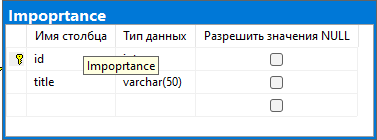
1. «Задачи»



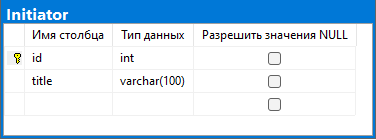
1. «Статус»



1. «Исполнитель»



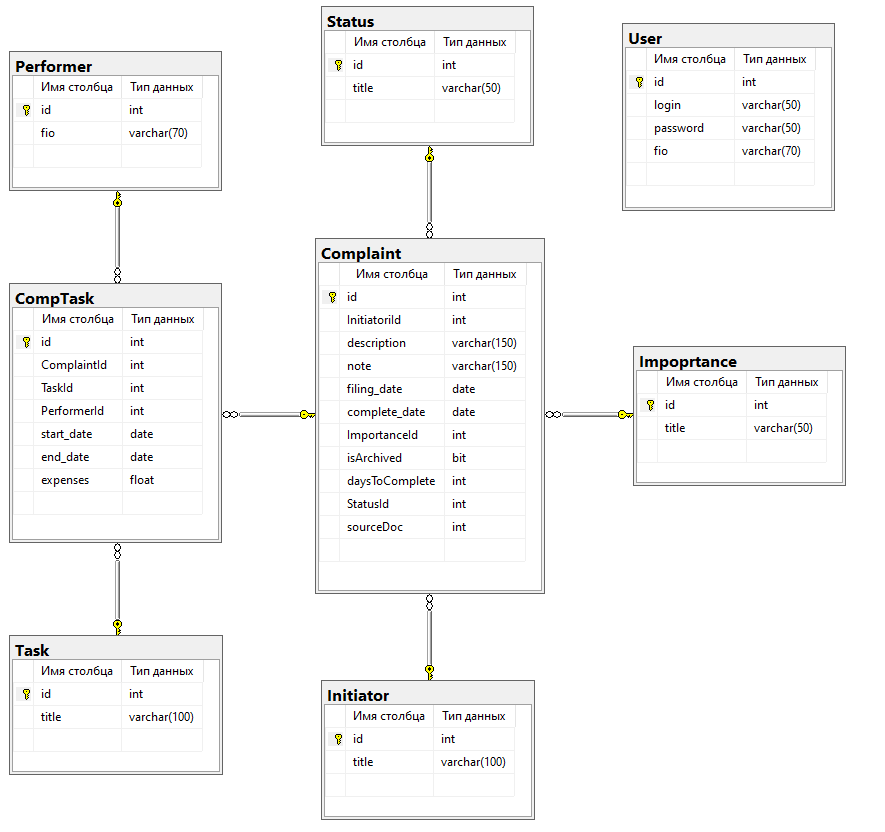
1. «Важность»



1. «Инициатор»

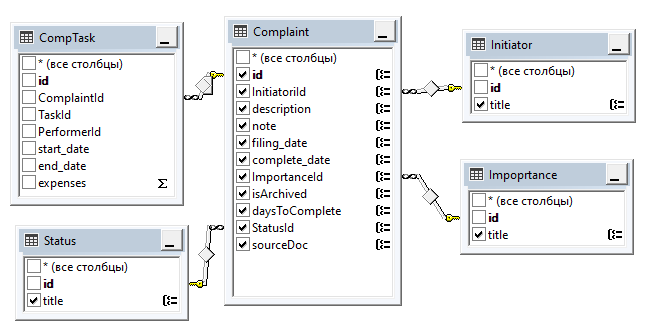
**Структура базы данных**

Диаграмма базы данных - включает в себя описания содержания, структуры и ограничений целостности, используемые для создания и поддержки базы данных. Для работы использована база данных SQL SERVER Management studio 18.



1. «Схема БД»

Представление ComplaintView, представляет таблицу рекламаций, где вместе с вторичными ключами выводятся связанные с ними названия:



1. «Представление ComplaintView»

Код представления ComplaintView:

SELECT dbo.Complaint.id AS ComplaintId, dbo.Complaint.description, dbo.Complaint.note, dbo.Complaint.filing\_date, dbo.Complaint.complete\_date, dbo.Complaint.daysToComplete, dbo.Initiator.title,

dbo.Impoprtance.title AS ImportanceTitle, dbo.Status.title AS StatusTitle, dbo.Complaint.InitiatoriId, dbo.Complaint.ImportanceId, dbo.Complaint.StatusId, dbo.Complaint.isArchived, ISNULL(SUM(dbo.CompTask.expenses), 0)

AS expenses, dbo.Complaint.sourceDoc

FROM dbo.Complaint INNER JOIN

dbo.Impoprtance ON dbo.Complaint.ImportanceId = dbo.Impoprtance.id INNER JOIN

dbo.Initiator ON dbo.Complaint.InitiatoriId = dbo.Initiator.id INNER JOIN

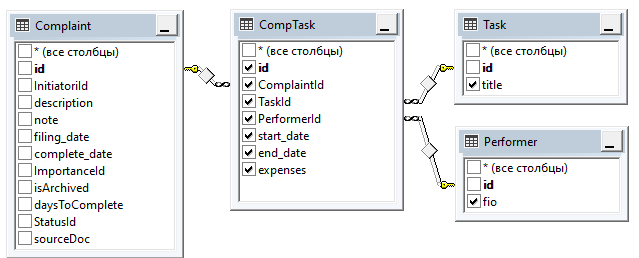
dbo.Status ON dbo.Complaint.StatusId = dbo.Status.id LEFT OUTER JOIN

dbo.CompTask ON dbo.Complaint.id = dbo.CompTask.ComplaintId

GROUP BY dbo.Complaint.description, dbo.Complaint.note, dbo.Complaint.filing\_date, dbo.Complaint.complete\_date, dbo.Complaint.daysToComplete, dbo.Initiator.title, dbo.Impoprtance.title, dbo.Status.title, dbo.Complaint.InitiatoriId,

dbo.Complaint.ImportanceId, dbo.Complaint.StatusId, dbo.Complaint.isArchived, dbo.Complaint.id, dbo.Complaint.sourceDoc

Представление TaskView представляет таблицу задач, где вместе с вторичными ключами выводятся связанные с ними названия



1. «Представление TaskView»

Код представления TaskView:

SELECT dbo.CompTask.id, dbo.CompTask.ComplaintId, dbo.CompTask.TaskId, dbo.CompTask.PerformerId, dbo.Task.title, dbo.Performer.fio, dbo.CompTask.start\_date, dbo.CompTask.end\_date, dbo.CompTask.expenses

FROM dbo.Complaint INNER JOIN

dbo.CompTask ON dbo.Complaint.id = dbo.CompTask.ComplaintId INNER JOIN

dbo.Task ON dbo.CompTask.TaskId = dbo.Task.id INNER JOIN

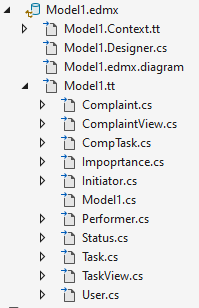
dbo.Performer ON dbo.CompTask.PerformerId = dbo.Performer.id

**Подключение к базе данных**

Таблица №3 «Подключение к базе данных»

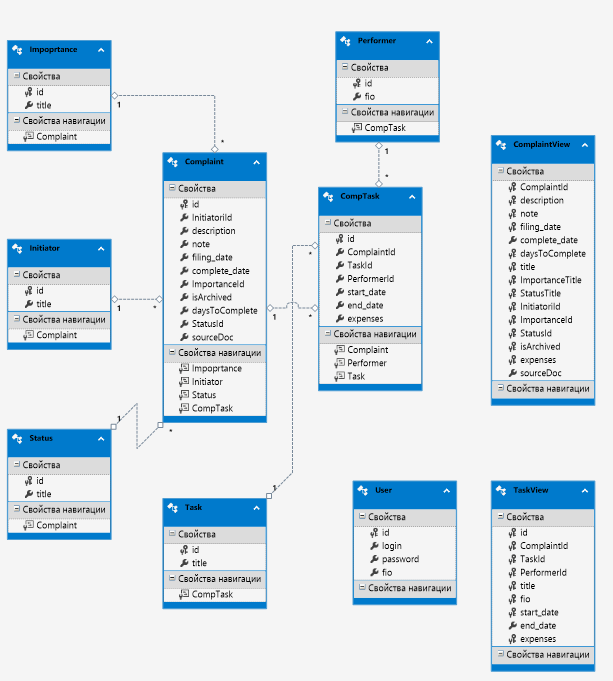
|  |  |
| --- | --- |
| В Visual Studio открыть источник данных, нажать на «Конструктор EF из базы данных» -> «Далее» | На окне выбора подключения к БД, нажать на «Создать подключение» |
| Добавить подключение | Выбрать таблицы |

Итог добавления



1. «Модель»

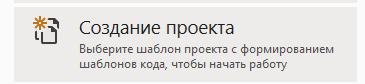
Модель данных – абстрактное, самодостаточное, логическое определение объектов, операторов и прочих элементов, в совокупности составляющих абстрактную машину доступа к данным, с которой взаимодействует пользователь.:



1. «Модель данных»

**Создание проекта**

Запустить Visual Studio и нажать на **Создание проекта**



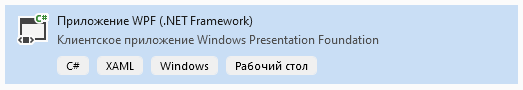
1. «Создание проекта»

Выбрать язык – **C#**, ОС – **Window**, Тип проекта – **Все типы проектов**.



1. «Создание проекта»

В списке найти **Приложение WPF (.NET Framework)**

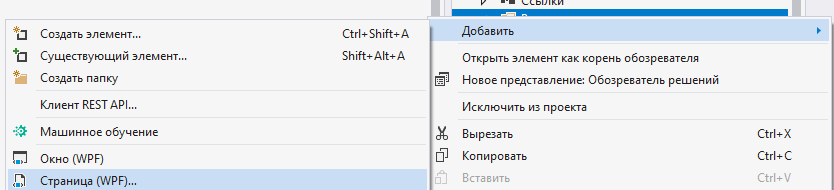


1. «Создание проекта»

Ввести имя проекта, указать путь и платформу - **.NET Framework 4.8** Нажать **Создать**.

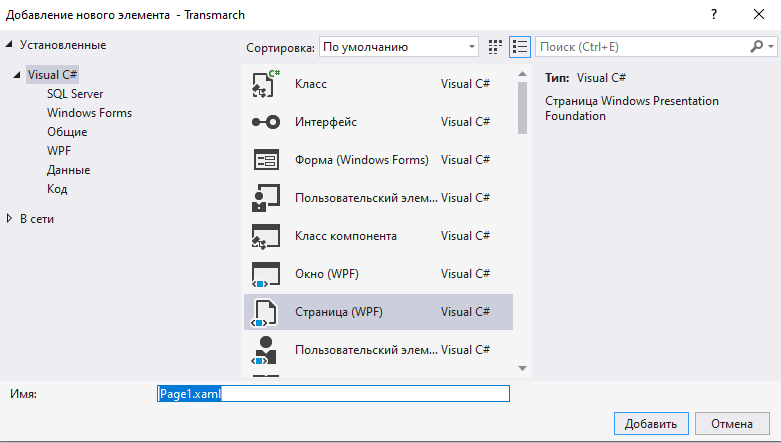
**Страницы**

Для создания страницы нужно в **Обозревателе решений** нажать ПКМ по названию проекта, затем **Добавить** и **Создать элемент.**



1. «Создание страницы»

В появившимся окне, выбрать **Страница (WPF)** и задать ей имя.

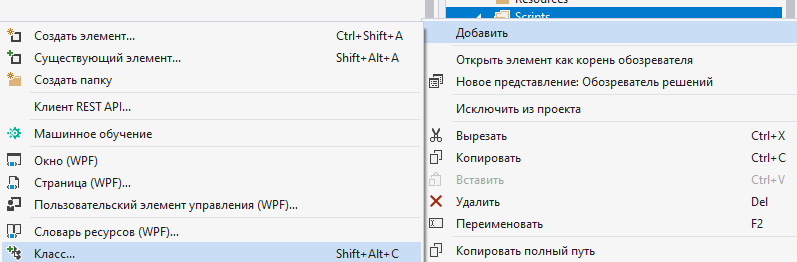


1. «Создание страницы»

**Классы**

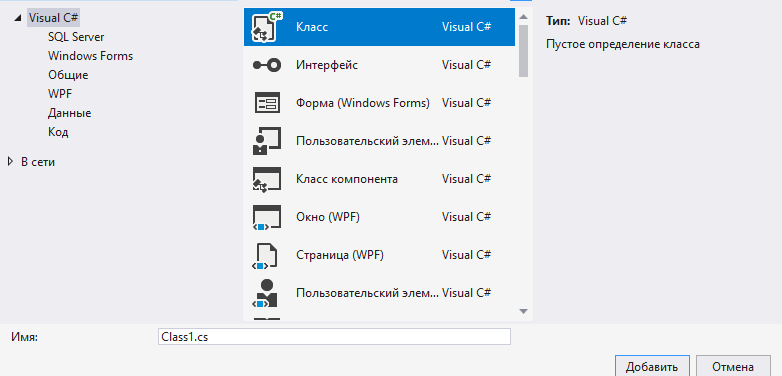
Так же в программе нужно реализовать, классы для осуществления переходов и работы с базой данных.

Для создания класса нужно в **Обозревателе решений** нажать ПКМ по названию проекта, затем **Добавить** и **Создать элемент.**

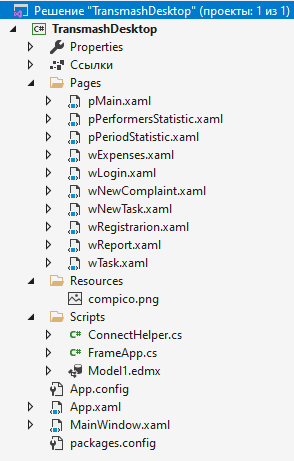


1. «Создание класса»

В появившимся окне, выбрать **Класс** и задать ему имя.



1. «Создание класса»



1. «Структура проекта»

**Стили в программе:**

Во всех элементах управления края скруглены аналогично следующему:

<Style TargetType="TabItem">

<Style.Resources>

<Style TargetType="{x:Type Border}">

<Setter Property="CornerRadius" Value="5,5,0,0" />

</Style>

</Style.Resources>

</Style>

Стиль и триггеры для ячейки статуса:

<Style x:Key="StatusCellStyle" BasedOn="{StaticResource Centering}" TargetType="DataGridCell">

<Setter Property="Background" Value="#FFFFFF6B" />

<Style.Resources>

<Style TargetType="{x:Type Border}">

<Setter Property="CornerRadius" Value="5" />

</Style>

</Style.Resources>

<Style.Triggers>

<DataTrigger Binding="{Binding StatusId}" Value="3">

<Setter Property="Background" Value="LightGreen" />

</DataTrigger>

<DataTrigger Binding="{Binding StatusId}" Value="1">

<Setter Property="Background" Value="LightCoral" />

</DataTrigger>

<Trigger Property="IsSelected" Value="True">

<Setter Property="Foreground" Value="#1C0772" />

</Trigger>

</Style.Triggers>

</Style>

Переопределены следующие элементы управления:

**PasswordBox:**

<Style TargetType="{x:Type ComboBox}">

<Setter Property="MinHeight" Value="25" />

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="{x:Type ComboBox}">

<Border

x:Name="DropDownBorder"

Background="Azure"

BorderBrush="#FFABADB3"

BorderThickness="1"

CornerRadius="5">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition />

<ColumnDefinition MaxWidth="30" />

</Grid.ColumnDefinitions>

<TextBox

Name="PART\_EditableTextBox"

Height="{TemplateBinding Height}"

Padding="5,0,0,0"

Style="{StaticResource ComboBoxTextBoxStyle}" />

<ToggleButton

Grid.Column="1"

Height="{TemplateBinding Height}"

Margin="0"

ClickMode="Press"

Focusable="False"

IsChecked="{Binding Path=IsDropDownOpen, Mode=TwoWay, RelativeSource={RelativeSource TemplatedParent}}"

Style="{StaticResource ComboBoxButtonStyle}">

<Path

Grid.Column="1"

HorizontalAlignment="Center"

VerticalAlignment="Center"

Data="M 0 0 L 4 4 L 8 0 Z"

Fill="DodgerBlue" />

</ToggleButton>

<ContentPresenter

Name="ContentSite"

Margin="5,0,0,0"

HorizontalAlignment="Left"

VerticalAlignment="Center"

Content="{TemplateBinding SelectionBoxItem}"

ContentTemplate="{TemplateBinding SelectionBoxItemTemplate}"

ContentTemplateSelector="{TemplateBinding ItemTemplateSelector}" />

<Popup

Name="Popup"

AllowsTransparency="True"

Focusable="False"

IsOpen="{TemplateBinding IsDropDownOpen}"

Placement="Bottom"

PopupAnimation="Slide">

<Grid

Name="DropDown"

MinWidth="{TemplateBinding ActualWidth}"

MaxHeight="{TemplateBinding MaxDropDownHeight}"

SnapsToDevicePixels="True">

<Border

x:Name="templateRoot"

Background="White"

BorderBrush="#FFABB4B4"

BorderThickness="{TemplateBinding BorderThickness}"

CornerRadius="5"

SnapsToDevicePixels="True" />

<ScrollViewer Margin="4,6,4,6" SnapsToDevicePixels="True">

<StackPanel IsItemsHost="True" KeyboardNavigation.DirectionalNavigation="Contained" />

</ScrollViewer>

</Grid>

</Popup>

</Grid>

</Border>

<ControlTemplate.Triggers>

<Trigger Property="IsEditable" Value="True">

<Setter TargetName="PART\_EditableTextBox" Property="Visibility" Value="Visible"/>

<Setter TargetName="PART\_EditableTextBox" Property="TextWrapping" Value="Wrap"/>

<Setter TargetName="PART\_EditableTextBox" Property="Margin" Value="0,2,0,0"/>

</Trigger>

<Trigger Property="IsEditable" Value="False">

<Setter TargetName="PART\_EditableTextBox" Property="Visibility" Value="Collapsed"/>

</Trigger>

</ControlTemplate.Triggers>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

**ComboBox:**

<Style TargetType="{x:Type ComboBox}">

<Setter Property="MinHeight" Value="25" />

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="{x:Type ComboBox}">

<Border

x:Name="DropDownBorder"

Background="Azure"

BorderBrush="#FFABADB3"

BorderThickness="1"

CornerRadius="5">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition />

<ColumnDefinition MaxWidth="30" />

</Grid.ColumnDefinitions>

<TextBox

Name="PART\_EditableTextBox"

Height="{TemplateBinding Height}"

Padding="5,0,0,0"

Style="{StaticResource ComboBoxTextBoxStyle}" />

<ToggleButton

Grid.Column="1"

Height="{TemplateBinding Height}"

Margin="0"

ClickMode="Press"

Focusable="False"

IsChecked="{Binding Path=IsDropDownOpen, Mode=TwoWay, RelativeSource={RelativeSource TemplatedParent}}"

Style="{StaticResource ComboBoxButtonStyle}">

<Path

Grid.Column="1"

HorizontalAlignment="Center"

VerticalAlignment="Center"

Data="M 0 0 L 4 4 L 8 0 Z"

Fill="DodgerBlue" />

</ToggleButton>

<ContentPresenter

Name="ContentSite"

Margin="5,0,0,0"

HorizontalAlignment="Left"

VerticalAlignment="Center"

Content="{TemplateBinding SelectionBoxItem}"

ContentTemplate="{TemplateBinding SelectionBoxItemTemplate}"

ContentTemplateSelector="{TemplateBinding ItemTemplateSelector}" />

<Popup

Name="Popup"

AllowsTransparency="True"

Focusable="False"

IsOpen="{TemplateBinding IsDropDownOpen}"

Placement="Bottom"

PopupAnimation="Slide">

<Grid

Name="DropDown"

MinWidth="{TemplateBinding ActualWidth}"

MaxHeight="{TemplateBinding MaxDropDownHeight}"

SnapsToDevicePixels="True">

<Border

x:Name="templateRoot"

Background="White"

BorderBrush="#FFABB4B4"

BorderThickness="{TemplateBinding BorderThickness}"

CornerRadius="5"

SnapsToDevicePixels="True" />

<ScrollViewer Margin="4,6,4,6" SnapsToDevicePixels="True">

<StackPanel IsItemsHost="True" KeyboardNavigation.DirectionalNavigation="Contained" />

</ScrollViewer>

</Grid>

</Popup>

</Grid>

</Border>

<ControlTemplate.Triggers>

<Trigger Property="IsEditable" Value="True">

<Setter TargetName="PART\_EditableTextBox" Property="Visibility" Value="Visible"/>

<Setter TargetName="PART\_EditableTextBox" Property="TextWrapping" Value="Wrap"/>

<Setter TargetName="PART\_EditableTextBox" Property="Margin" Value="0,2,0,0"/>

</Trigger>

<Trigger Property="IsEditable" Value="False">

<Setter TargetName="PART\_EditableTextBox" Property="Visibility" Value="Collapsed"/>

</Trigger>

</ControlTemplate.Triggers>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

**Функции программы**

* FiltSearch/DatePeriod – Заполнение таблиц с учетом всех введенных параметров например: поиск и фильтрация.

private void FiltSearch()

{

btnRemove.Visibility = Visibility.Collapsed;

List<ComplaintView> complaintView = new List<ComplaintView>();

switch (cbFilter.SelectedIndex)

{

case 0:

complaintView = ConnectHelper.entObj.ComplaintView.

Where(c => c.ComplaintId != 1 && !c.isArchived && c.description.Contains(tbSearch.Text)).

OrderBy(c => c.StatusId).ToList();

break;

case 1:

complaintView = ConnectHelper.entObj.ComplaintView.

Where(c => c.ComplaintId != 1 && !c.isArchived && c.description.Contains(tbSearch.Text) && c.StatusId != 3).

OrderBy(c => c.StatusId).ToList();

break;

case 2:

complaintView = ConnectHelper.entObj.ComplaintView.

Where(c => c.ComplaintId != 1 && !c.isArchived && c.description.Contains(tbSearch.Text) && c.StatusId == 3).

OrderBy(c => c.StatusId).ToList();

btnRemove.Visibility = Visibility.Visible;

break;

case 3:

complaintView = ConnectHelper.entObj.ComplaintView.

Where(c => c.ComplaintId != 1 && c.isArchived && c.description.Contains(tbSearch.Text)).

OrderBy(c => c.StatusId).ToList();

break;

}

if (complaintView.Count(c => c.isArchived == false && c.StatusId == 3) < 1)

dgtcArchive.Visibility = Visibility.Collapsed;

else dgtcArchive.Visibility = Visibility.Visible;

tblComplaint.ItemsSource = complaintView;

tbCount.Text = $"{complaintView.Count} из {ConnectHelper.entObj.Complaint.Count()}";

}

* Update – Загрузка и обновление данных на страницу.

public void Update()

{

List<ComplaintView> complaintView = ConnectHelper.entObj.ComplaintView.

Where(c => c.ComplaintId != 1 && !c.isArchived && !c.isArchived).

OrderBy(c => c.StatusId).ToList();

tblComplaint.ItemsSource = complaintView;

tbSearch.Text = "";

cbFilter.SelectedIndex = 0;

tbCount.Text = $"{complaintView.Count} из {ConnectHelper.entObj.Complaint.Count()}";

btnRemove.Visibility = Visibility.Collapsed;

if (complaintView.Count(c => c.isArchived == false && c.StatusId == 3) < 1)

dgtcArchive.Visibility = Visibility.Collapsed;

else dgtcArchive.Visibility = Visibility.Visible;

};

* CanSave – Проверка заполненных полей при добавлении новых данных.

public bool CanSave()

{

bool t = false;

if (tbNote.Text != comp.note)

t = t || true;

if (cbImport.SelectedItem != null)

if (cbImport.SelectedIndex + 1 != comp.ImportanceId)

t = t || true;

return t;

}

* btnPrint\_Click– Выводит отчет на печать.

private void btnPrint\_Click(object sender, RoutedEventArgs e)

{

double w = spReport.Width;

double h = spReport.Height;

double ww = Width;

double hw = Height;

const double pixelsInCentimeter = 96 / 2.54;

PrintDialog printDialog = new PrintDialog();

spReport.Height = 30 \* pixelsInCentimeter;

spReport.Width = 21 \* pixelsInCentimeter;

spHelp.Margin = new Thickness

(3 \* pixelsInCentimeter,

2 \* pixelsInCentimeter,

1.5 \* pixelsInCentimeter,

2 \* pixelsInCentimeter);

Height = 30 \* pixelsInCentimeter + 10;

Width = 21 \* pixelsInCentimeter + 10;

if (printDialog.ShowDialog() == true)

printDialog.PrintVisual(spReport,

"Результирующий отчет по жалобе №" + comp.ComplaintId);

spHelp.Margin = new Thickness(30, 0, 30, 0);

spReport.Height = h;

spReport.Width = w;

Height = hw;

Width = ww;

}

* btnSave\_Click– Сохраняет изменения или новый объект.

private void btnSave\_Click(object sender, RoutedEventArgs e)

{

Performer per = new Performer();

if (cbPerf.SelectedItem != null)

per = cbPerf.SelectedItem as Performer;

else { try {

if (MessageBox.Show($"Введенный вами исполнитель будет добавлен в БД\nХотите ли вы сохранить исполнителя: {cbPerf.Text} ?", "Сохранение Исполнителя", MessageBoxButton.YesNo, MessageBoxImage.Question) == MessageBoxResult.Yes)

{

per = new Performer { fio = cbPerf.Text };

ConnectHelper.entObj.Performer.Add(per);

ConnectHelper.entObj.SaveChanges();

}

else {cbPerf.Focus(); return;}

}

catch (Exception ex)

{MessageBox.Show($"Произошла непредвиденная ошибка в сохранении, проверьте данные и повторите попытку\nВызвано исключением:\n{ex}",

"Ошибка сохранения", MessageBoxButton.OK, MessageBoxImage.Error); return;}

}

Task tas = new Task();

if (cbName.SelectedItem != null)

tas = cbName.SelectedItem as Task;

else try

{

tas = new Task { title = cbName.Text };

ConnectHelper.entObj.Task.Add(tas);

ConnectHelper.entObj.SaveChanges();

}

catch (Exception ex)

{MessageBox.Show($"Произошла непредвиденная ошибка в сохранении, проверьте данные и повторите попытку\nВызвано исключением:\n{ex}",

"Ошибка сохранения", MessageBoxButton.OK, MessageBoxImage.Error); return;}

try

{

CompTask compTask = new CompTask

{

ComplaintId = (int)comp.ComplaintId,

TaskId = tas.id,

PerformerId = per.id,

start\_date = DateTime.Today

};

ConnectHelper.entObj.CompTask.Add(compTask);

ConnectHelper.entObj.SaveChanges();

MessageBox.Show("Задача добавлена!", "Успешное сохранение", MessageBoxButton.OK, MessageBoxImage.Information);

Close();

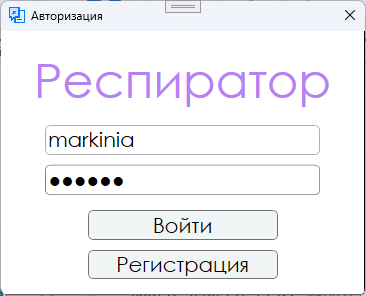
}

catch (Exception ex)

{MessageBox.Show($"Пожалуйста проверьте данные на правильность\nВызвано исключением:\n{ex}", "Ошибка сохранения", MessageBoxButton.OK, MessageBoxImage.Error);} } }

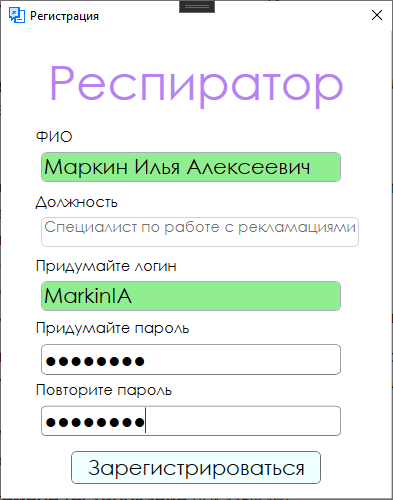
**Руководство пользователя**

Программа запускается с помощью исполняющего файла RespiratorDesktop.exe. При входе, запускается окно авторизации, для входа нужно ввести логин и пароль, либо перейти на окно регистрации, зарегистрироваться, после чего ввести данные в полях авторизации.



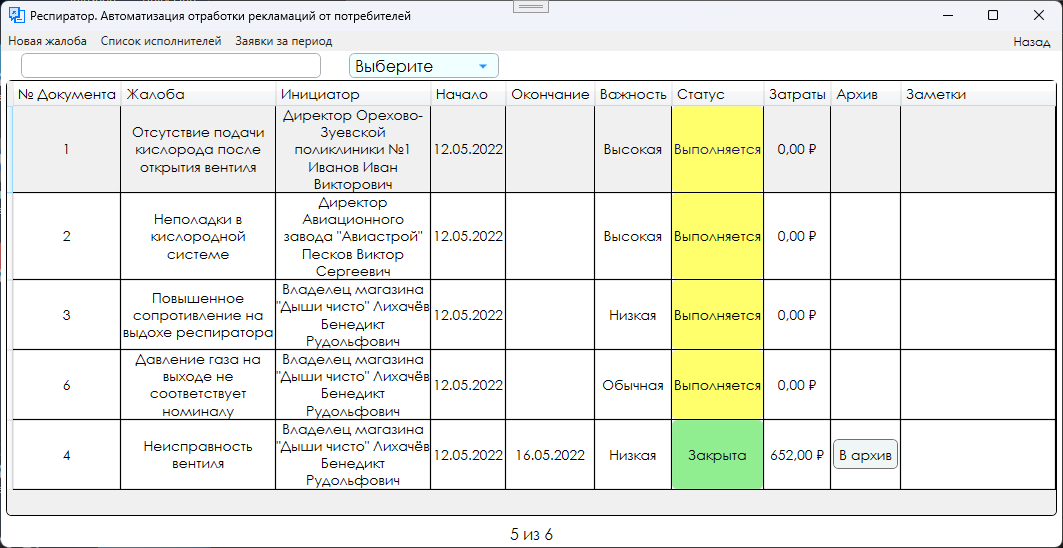
1. «Окно авторизации»

На окне регистрации необходимо ввести свое ФИО, придумать логин и пароль, после чего нажать на кнопку «Зарегистрироваться».



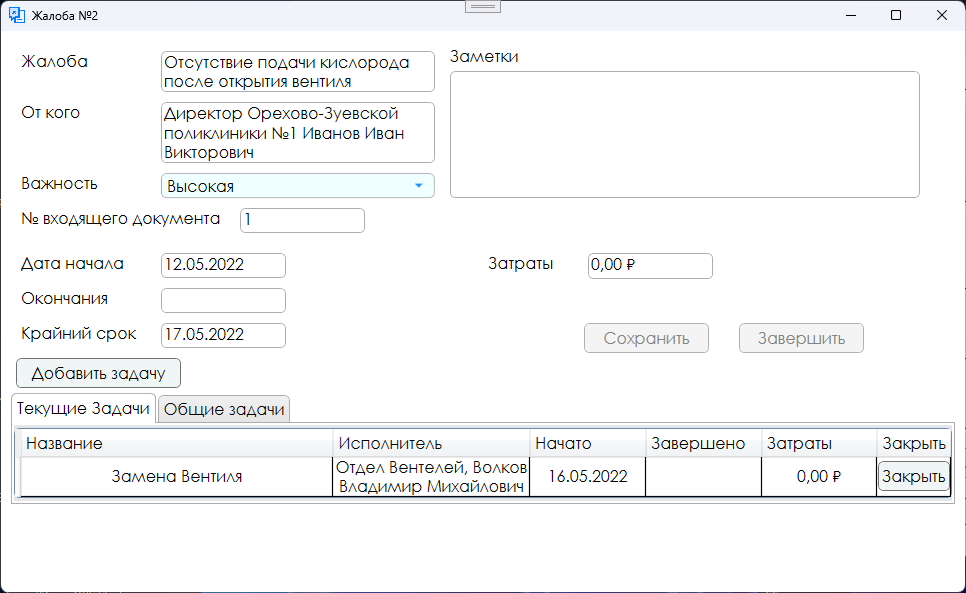
1. «Окно регистрации»

После входа отображается таблица со всеми рекламациями, также в текстовом поле сверху можно найти рекламации по названию, а в поле со списком выбрать статус отображаемых заявок. Чтобы просмотреть сведения о заявке нужно сделать двойной щелчок мышью по заявке. Чтобы зарегистрировать новую рекламацию необходимо кликнуть по пункту меню «Новая заявка». Для перехода к статистике задач по исполнителям надо кликнуть по пункту меню «Список исполнителей». Для просмотра статистики заявок за определенный период необходимо кликнуть по пункту меню «Заявки за период». Для архивирования претензии нужно нажать кнопку в архив у необходимой закрытой претензии.



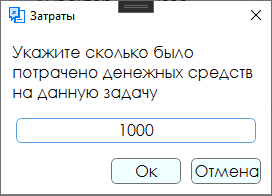
1. «Страница с рекламациями»

В сведениях о заявке указаны все данные о ней, также здесь находятся задачи, заданные для текущей заявки и общие задачи, которые при нажатии по кнопке «Добавить», добавляются к текущей заявке. По кнопке «Добавить задачу» Задача добавляется к текущей задаче или к общим, в зависимости от активной вкладки. По клику «Закрыть», появляется новое окно, в котором необходимо указать затраты по данной задаче, после чего задача считается завершенной. При отсутствии или при выполнении всех задач заявку можно завершить. После чего кнопка «Завершить» меняется на кнопку, «Отчет» которая выводит отчет по заявке.



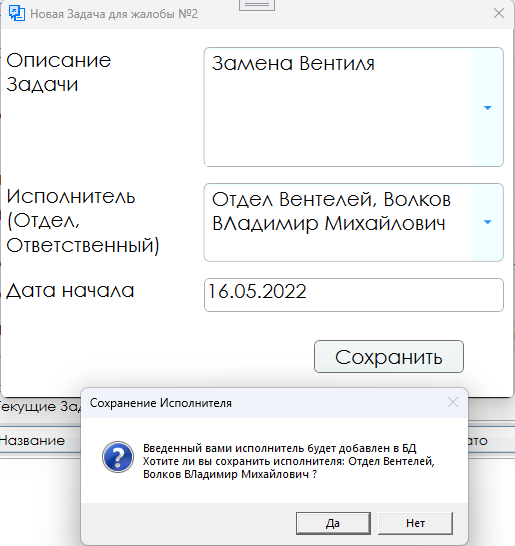
1. «Окно Сведений и задач по рекламации»

Это окно появляется по закрытию задачи, здесь необходимо указать сколько потрачено на выполнение данной задачи.



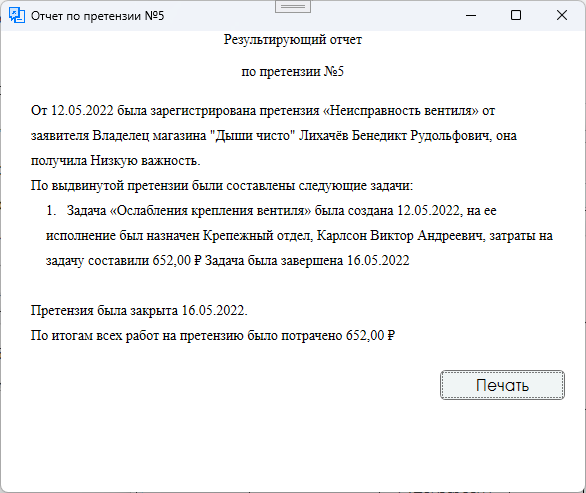
1. «Окно указания затрат»

На этой странице можно добавить задачу для текущей либо для общей задачи, для этого нужно указать описание задачи и назначить исполнителя, если текущего исполнителя нет, то создается новый.



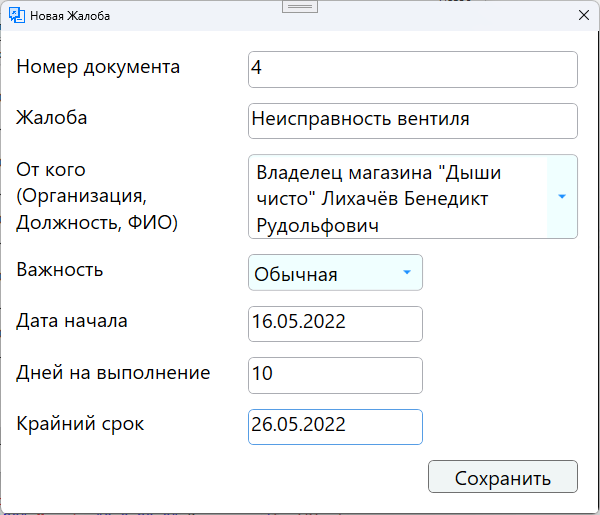
1. «Окно Добавления задачи»

Страница отчета со всеми данными о рекламации и назначенных на ее исправление задач, появляется после клика по кнопке «отчет» на окне сведений о заявке. Распечатать отчет можно нажав по кнопке «Печать».



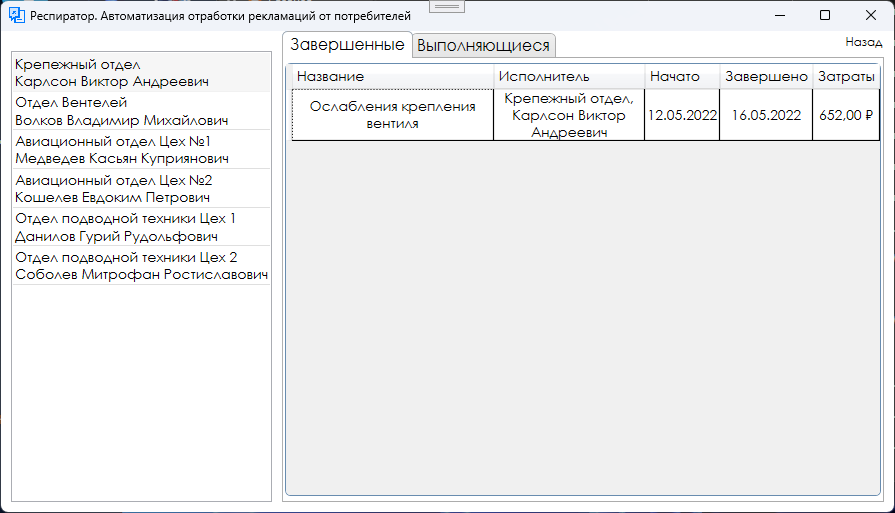
1. «Окно отчета по претензии»

При переходе на окно новой заявки, необходимо, указать № рекламации, которую подал клиент, описать претензию, указать сколько дней дается на выполнение (по умолчанию – 7) и при необходимости повысить или понизить важность. После нажатия кнопки сохранить, сохраняется клиент (если был введен новый, а не выбран существующий) и введенная претензия добавляется.



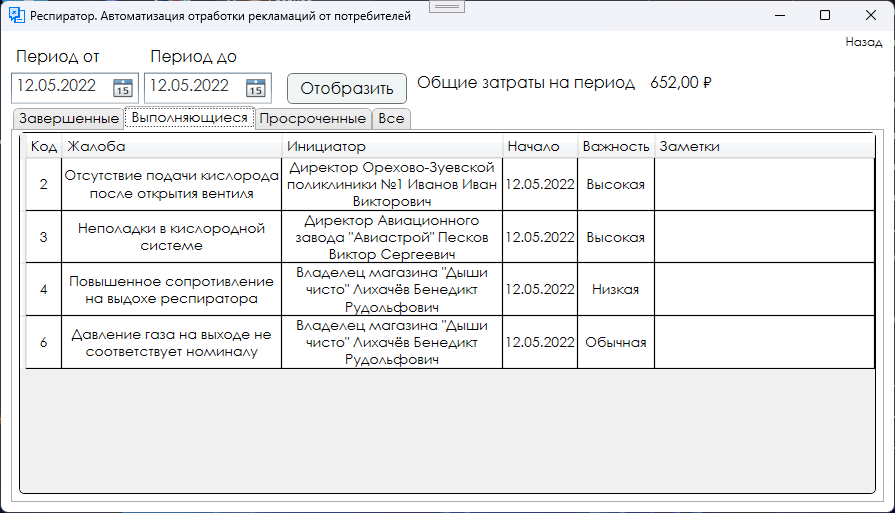
1. «Окно Регистрации рекламации»

На окне «Список исполнителей» в левой части отображаются Исполнители, при клике по нему в области справа в соответствии с состоянием появляется таблица с задачами, которые выполняет или выполнял исполнитель.



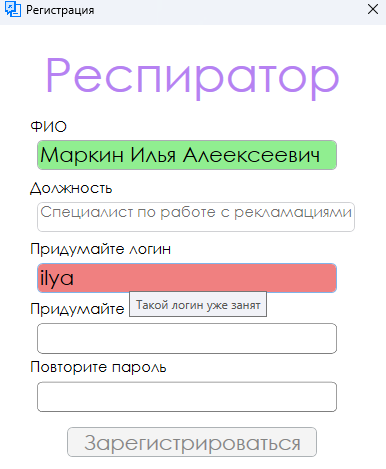
1. «Страница Статистики по исполнителям»

На странице «Заявки за период» необходимо задать промежуток времени и нажать кнопку отобразить, после чего внизу будут выведены таблицы, с заявками, которые были начаты в заданный период, также справа от кнопки отобразить высвечивается затраты по выполненным заявка за указанный период. Также, как и на главной странице, если нажать дважды по рекламации, то откроется окно сведений о ней.



1. «Страница Статистики за период»

В программе имеются сообщения для пользователя



1. «Индикаторы и подсказки неверности ввода»

Таблица №4 «Сообщения пользователю»

|  |  |
| --- | --- |
| Сообщения об успешном действии | |
|  |  |
|  |  |
| Сообщения об ошибке: | |
|  |  |
| Сообщения с подтверждением действий | |
|  |  |
|  | |

# Тестирование и отладка

## Виды тестирования

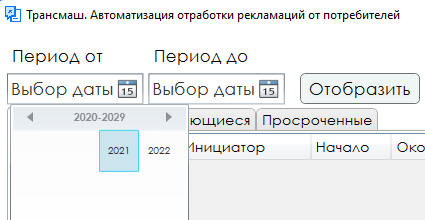
Тестирование программного обеспечения — процесс исследования, испытания программного продукта, имеющий своей целью проверку соответствия между реальным поведением программы и её ожидаемым поведением на конечном наборе тестов, выбранных определённым образом. Тестирование программного обеспечения также может обеспечить объективный, независимый взгляд на программное обеспечение, чтобы позволить бизнесу оценить и понять риски внедрения программного обеспечения.

Существуют две стратегии тестирования:

1. Черный ящик – стратегия (метод) тестирования функционального поведения объекта (программы, системы) с точки зрения внешнего мира, при котором не используется знание о внутреннем устройстве тестируемого объекта.

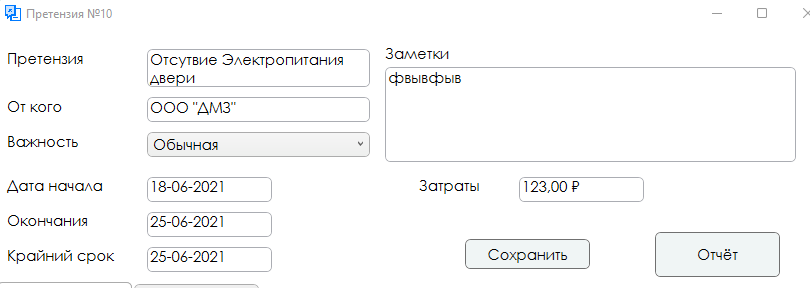
Примеры ошибок:

1.1) При выборе даты в полях периода, нельзя выбрать даты, не включенные в период с 1 января 2021 до 2 января 2022:



1. «Выбор даты периода»

1.2) Программа позволяет внести изменения в уже закрытую претензию:



1. «Кнопка сохранения доступна у закрытой заявки»

1.3) Разработаны тестовые сценарии: [см. Приложение 3](#_Приложение_3).

1. Белый ящик - метод тестирования программного обеспечения, который предполагает, что внутренняя структура/устройство/реализация системы известны тестировщику. Традиционно тестирование белого ящика выполняется на уровне модулей, однако оно используется для тестирования интеграции систем и системного тестирования, тестирования внутри устройства и путей между устройствами. Этот метод тестирования не может выявить невыполненные части спецификации, отсутствие требований или создание не того приложения.

Интеграционное тестирование – одна из фаз тестирования программного обеспечения, при которой отдельные программные модули объединяются и тестируются в группе. Обычно интеграционное тестирование проводится после модульного тестирования и предшествует системному тестированию

Пример:

2.1) При переходе на окно информации о рекламации открывается именно выбранная рекламация.

Переход и передача объекта:

private void tblComplaint\_MouseDoubleClick(object sender, MouseButtonEventArgs e)

{

if (!(tblComplaint.SelectedItem is ComplaintView))

return;

wTask.comp = tblComplaint.SelectedItem as ComplaintView;

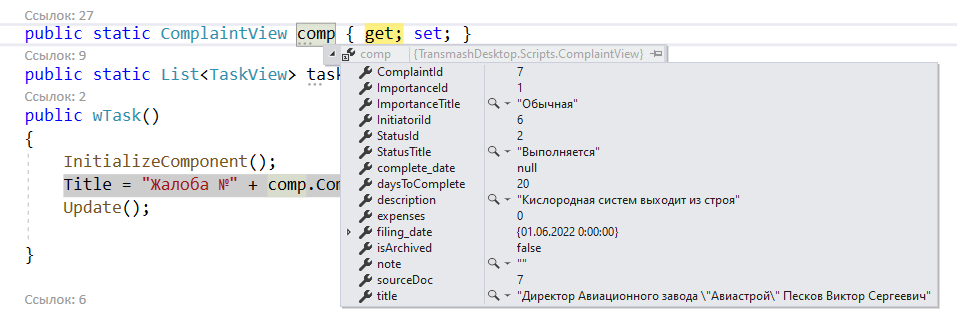
wTask window = new wTask();

window.ShowDialog();

FiltSearch();

}

Переданный объект:



1. «Переданная жалоба на новую страницу»

2.2) При переходе на страницу добавления новой задачи в зависимости от активной вкладки открывается добавление задачи в шаблон либо для текущей рекламации

Переход и передача объекта в зависимости от активной вкладки:

private void btnNewTask\_Click(object sender, RoutedEventArgs e)

{

if (tcTask.SelectedIndex == 1)

wNewTask.comp = ConnectHelper.entObj.ComplaintView.First(c => c.ComplaintId == 1);

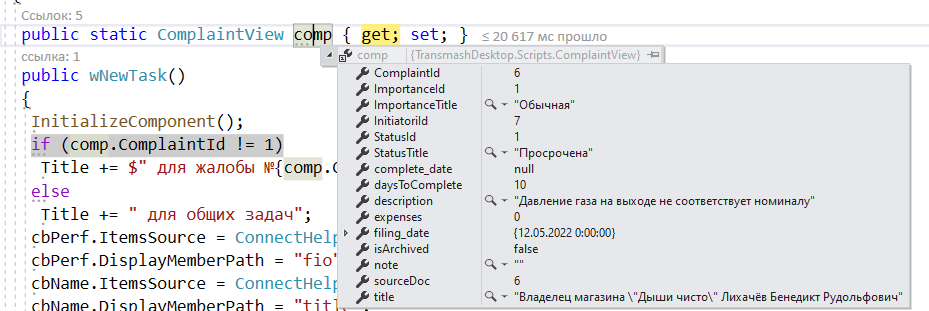
else wNewTask.comp = comp;

new wNewTask().ShowDialog();

Update();

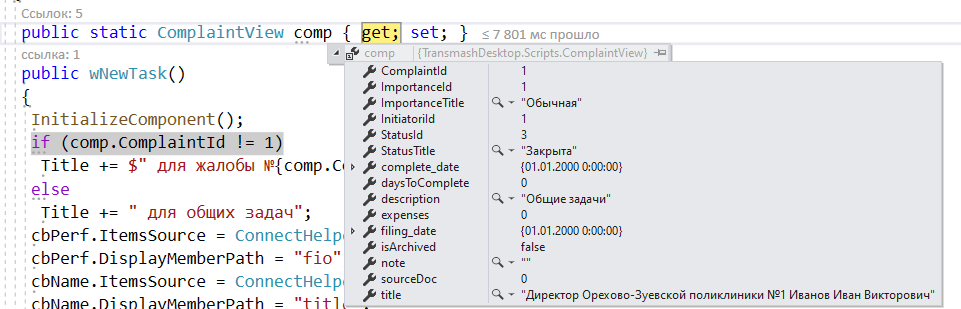
}

Передана текущая рекламация:



1. «Переданная жалоба на новую страницу»

Передано добавление в шаблон:



1. «Переданная жалоба на новую страницу»

## Отладка программного приложения

Отладка — этап разработки компьютерной программы, на котором обнаруживают, локализуют и устраняют ошибки.

Чтобы понять, где возникла ошибка, приходится:

* узнавать текущие значения переменных;
* выяснять, по какому пути выполнялась программа.

Термин отладка может иметь разные значения, но в первую очередь он означает устранение ошибок в коде. Делается это по-разному. Например, отладка может выполняться путем проверки кода на наличие опечаток или с помощью анализатора кода. Код можно отлаживать с помощью профилировщика производительности. Кроме того, отладка может производиться посредством отладчика.

Отладчик — это узкоспециализированное средство разработки, которое присоединяется к работающему приложению и позволяет проверять код.

Отладчик представляет из себя программный инструмент, позволяющий программисту наблюдать за выполнением исследуемой программы, останавливать и перезапускать её, прогонять в замедленном темпе, изменять значения в памяти и даже, в некоторых случаях, возвращать назад по времени.

Отладчик — важнейший инструмент для поиска и устранения ошибок в приложениях. Однако большое значение имеет контекст. Важно использовать все средства, имеющиеся в распоряжении, чтобы быстро устранять ошибки. Зачастую лучшим "средством" являются правильные методики написания кода. Зная, когда лучше использовать отладчик, а когда — другие средства, можно более эффективно использовать отладчик.

Таблица №5 «Ошибки в программе»

|  |  |  |  |
| --- | --- | --- | --- |
| **Название Ошибки** | **Скрин ошибки** | **Метод исправления** | **Результат** |
| Цвет текста колонки статуса «Выполняется» при выделении строки становится белым |  | Добавление триггера, который заменяет цвет текста при выделении на фиолетовый |  |
| При открытии окна добавления общей задачи, название остается как для добавления задачи на претензию |  | Исправить Title при добавлении общих задач |  |
| Текст кнопки «В архив» очень близок к границам кнопки |  | У кнопки добавить свойство Padding |  |

# Расчет базовой себестоимости разрабатываемого продукта

## Исходные данные

Затраты времени и занятость специалистов соответствуют трудоемкости и сложности этапов создания приложения представлены в таблице №6.

Таблица №6 «Исходные данные»

|  |  |  |  |
| --- | --- | --- | --- |
| **Этапы разработки программного продукта** | **Наименее возможная величина затрат, дни** | **Наиболее вероятная**  **величина затрат, дни** | **Наиболее возможная величина затрат, дни** |
| Анализ поставленной задачи | 1 | 2 | 2 |
| Разработка структуры базы данных | 1 | 1 | 1 |
| Разработка интерфейса | 5 | 10 | 15 |
| Программирование | 12 | 15 | 18 |
| Тестирование и отладка | 3 | 5 | 5 |
| Итого | 22 | 33 | 41 |

## Расчёты затрат на выполнение программы

Создание компьютерной программы связанно со значительными затратами времени и соответственно большим удельным весом в себестоимости таких затратах, как оплата труда.

В работе по проектированию и разработке программы участвовал 1 специалист - разработчик.

Расчет оплаты специалистов производиться исходя из дневной тарифной ставки.

Тарифная ставка — в трудовом праве фиксированный размер оплаты: труда работника за выполнение нормы труда определённой сложности за единицу времени.

Фонд оплаты труда- суммарные денежные средства, израсходованные в течение определённого периода времени на заработную плату, включая: стимулирующие оплаты и дополнительную оплату.

Расчёт чистых затрат на выполнение программы рассчитывается из затрат на заработную плату разработчику, умноженную на наиболее возможное количество дней разработки и суммирование с прочими затратами на разработку, такие как электричество.

Затраты на заработную плату разработчику – 25 000/21\*41= 48 809 руб.

## Расчет отчислений на социальное страхование и обеспечение

Обычный размер ставки - для наемного работника - составляет 30 %. Пенсионный фонд Российской Федерации - 22 %

* Фонд социального страхования - 2,9 %
* Фонды обязательного медицинского страхования - 5,1 %

Итого: 30 %

Таблица №7 «Расчет отчислений на социальное страхование и обеспечение»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Должность работника | Пенсионный фонд РФ – ПФР (22%), руб. | Фонд социального страхования РФ – ФССРФ (2.9%), руб. | ФОМС (5.1%), руб. | Итого отчислений на социальное страхование и обеспечение (ОСО), руб. |
| 1 | Разработчик | 10 738 | 1 415 | 2 489 | 14 642 |

Формулы расчета для таблицы №7:

ПФР = ОТ \* 22% = 48 809 руб. \* 22% = 10 738‬ руб.

ФСС РФ = ОТ \* 2,9% = 48 809 руб. \* 2,9% = 1 415 руб.

ФОМС = ОТ \* 5,1% = 48 809 руб. \* 5,1% = 2 489 руб.

ОСО=ПФР+ФССРФ+ФОМС= 10 738‬ руб. + 1 415 руб. + 2 489 руб. = 14 642 руб.

## Расчет базовой себестоимости компьютерного продукта

Себестоимость — это текущие затраты организации на производство и реализацию продукции, выраженные в денежной форме.

Себестоимость продукции является качественным показателем, так как она характеризует уровень использования всех ресурсов, находящихся в распоряжении организации.

Косвенные расходы — это затраты, связанные с производством и реализацией продукции (работ, услуг), которые нельзя напрямую отнести к изготовлению продукции (работ, услуг) и можно учесть в расходах в том периоде, когда они понесены.

К косвенным расходам можно отнести, например, затраты на услуги связи, аренду офиса и т.д.

Самое главное отличие прямых расходов от косвенных расходов в том, что сумма косвенных расходов в полном объеме относится к расходам текущего отчетного (налогового) периода, а прямых — к расходам текущего периода по мере реализации товаров, работ, то есть с учетом остатков незавершенного производства.

Исключением являются случаи, когда деятельность организации связана с оказанием услуг. Данные налогоплательщики вправе относить сумму прямых расходов отчетного (налогового) периода в полном объеме на уменьшение доходов от производства и реализации данного отчетного (налогового) периода без распределения на остатки незавершенного производства.

Относить ли расходы к прямым или косвенным расходам компания определяет отдельно для каждого производственного цикла.

Если те или иные ресурсы согласно технологическим регламентам не включены в производственный цикл, не являются его неотъемлемой частью, то затраты на них можно учитывать в составе косвенных расходов.

**Перечень косвенных расходов**

Косвенные расходы — это расходы, которые нельзя прямо отнести на конкретные виды продукции (работ, услуг).

К косвенным расходам относятся:

1. административно – управленческие расходы;
2. расходы на отопление и освещение помещений;
3. расходы на страхование;
4. расходы на содержание общехозяйственного персонала;
5. амортизационные отчисления и расходы на ремонт основных средств управленческого и общехозяйственного назначения;
6. арендная плата за помещения общехозяйственного назначения;
7. расходы по оплате информационных, аудиторских, консультационных и т. п. услуг;
8. расходы, связанные со сбытом продукции:
9. другие аналогичные по назначению управленческие расходы.

Таблица №8 «Расчёт базовой себестоимости»

|  |  |  |
| --- | --- | --- |
| **№** | **Статьи затрат** | **Сумма в рублях** |
| 1 | Фонд оплаты труда (ФОТ) | 63 451 |
| 2 | Косвенные расходы (КР) 10% от ФОТ | 6 345 |
| 3 | Расходы на продажу (РП) (ФОТ + КР) \* 10% | ‬6 980 |
| 4 | Полная себестоимость (ПС) | 76 776 |

Формулы расчета для таблицы №8:

ФОТ=ОТ+ОСО= 48 809 руб. + 14 642 ‬руб. = 63 451 ‬‬руб.

В данной организации косвенные расходы составляют 10% от ФОТ.

КР=ФОТ\*10%= 63 451‬‬ руб. \*10% = 6 345 руб.

РП=(ФОТ+ КР)\*10% = (63 451‬‬ руб. + 6 345 руб.) \* 10% = ‬6 980‬ руб.

ПС=ФОТ+КР+РП= 63 451‬‬ руб. + 6 345 руб. + 6 980 руб.= 76 776 руб.

Выгода = стоимость аналогичного программного продукта на рынке – себестоимость.

Аналоги данного программного продукта включая базовую версию 1С: Предприятия 8.3 начинаются с 115 164 рублей, в зависимости от цены часа работы специалиста при доработки нужной конфигурации.

Итоговая выгода с разработки = 115 164 руб. – 76 776 руб.

Выгода = 38 388 рублей.

По итогам экономического расчёта полная себестоимость составила 76 776 руб. и можно сделать вывод, что продукт полностью окупает затраты на его разработку, а также предоставляет выгоду от разработки в размере 38 388 рублей.

# Заключение

В ходе проведенного исследования разработан проектный модуль для информационной системы «работа с потребителями» для АО «НПП «Респиратор»

Проведён анализ материально-технической базы. Собран материал для выполнения выпускной квалификационной работы в соответствии с заданием на дипломное проектирование.

Изучена документация и структурная схема деятельности предприятия, проведено ознакомление с АРМ специалиста и инструктаж по охране труда и технике безопасности, проанализирован программно-аппаратный комплекс, а также были выполнены порученные производственные задания.

Совместно с заказчиком сформулирована постановка задачи, в которой указаны функциональные и нефункциональные требования к приложению.

Проведен анализ имеющихся на рынке программного обеспечения готовых приложений в соответствии с автоматизированным рабочим местом специалиста данной предметной области.

Согласно постановке задачи, спроектирована предметная область и разработана база данных. Разработана схема базы данных. Произведена нормализация данных.

В соответствии с постановкой задачи разработано приложение по автоматизации процесса работы с потребителями, со следующими функциональными характеристиками:

* Регистрация рекламации при обращении клиента;
* Назначение задач для решения несоответствия;
* Расчет итоговых затрат на устранение несоответствия;
* Формирование результирующего отчета по устранению несоответствий в рекламации для отправки поставщику;
* Визуальное отображение состояния заявок;
* Отображение статистики задач по исполнителям;
* Архивация закрытых претензий;
* Отображение затрат на устранение несоответствий за период.

В техническом проекте отражены требования к составу технических средств, информационно-программной совместимости и требования, к надёжности, которые должны быть соблюдены во время разработки, внедрении и эксплуатации приложения.

В соответствии с техническим заданием разработан алгоритм работы приложения, спроектированы экранные формы в Microsoft Visual Studio. Также построена диаграмма прецедентов и диаграмма действий.

Для разработки приложения использовались следующие средства:

* Visual Studio 2022
* Microsoft SQL Server 2019

В процессе разработки создано приложение, которое полностью протестировано на наличие критических ошибок, и отлажено для дальнейшего удобства в работе. Для отладки приложения использовались встроенные программные средства языка C#, WPF, а для тестирования и испытания разработаны тестовые примеры и наборы данных.

Разработана техническая документации в соответствии с ГОСТом. Составленные такие документы как:

1. Руководство программиста, где описывается вся техническая часть программы вместе с её кодом
2. Руководство пользователя, где описано как программа поведет себя на различные действия пользователей.

Для защиты приложения создана форма авторизации пользователей. Приложение разработано в установленные сроки и удовлетворяет потребностям заказчика.

Программный модуль может быть установлен на любую операционную систему, начиная с Windows 10 x64. В проекте используется язык запросов LINQ.

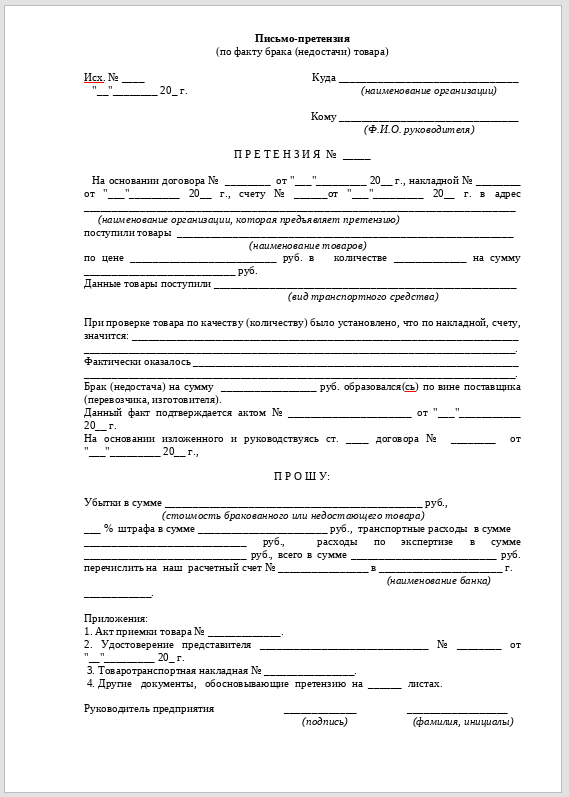
# Список литературы

1. ГОСТ 19.504-79 ЕСПД. Руководство программиста. Требования к содержанию и оформлению
2. ГОСТ 19.201-78 Единая система программной документации. Техническое задание. Требования к содержанию и оформлению.
3. ГОСТ 19.106-78 Единая система программной документации Требования к программным документам, выполненным печатным способом.
4. ГОСТ Р 56922-2016 Системная и программная инженерия. Тестирование программного обеспечения.
5. ISO/IEC TR 19759:2005
6. C# 7.0. Справочник. Полное описание языка.: Пер. с англ. – СпБ.: ООО “Альфакнига”, 2018. – 1024 с. : ил. – Парал. тит. англ.
7. Программирование на C# для начинающих. Основные сведения / Алексей Васильев. – Москва : Эксмо, 2018. – 592 с. – (Российский компьютерный бестселлер).
8. Язык программирования C# 7 и платформы .NET и .NET Core, 8-е изд. : Пер. с англ. – СПб. : ООО “Диалектика”, 2018 – 1328 с. : ил. – Парал. тит. англ.
9. Кудрявцев, К. Я. Методы оптимизации : учеб. пособие для вузов / К. Я. Кудрявцев, А. М. Прудников. — 2-е изд. — М. : Издательство Юрайт, 2019. — 140 с.
10. Трофимов, В. В. Основы алгоритмизации и программирования : учебник для СПО / В. В. Трофимов, Т. А. Павловская ; под ред. В. В. Трофимова. — М. : Издательство Юрайт, 2019. — 137 с.
11. Кудрина, Е. В. Основы алгоритмизации и программирования на языке c# : учеб. пособие для СПО / Е. В. Кудрина, М. В. Огнева. — М. : Издательство Юрайт, 2019. — 322 с.
12. Гордеев, С. И. Организация баз данных в 2 ч. Часть 2 : учебник для вузов / С. И. Гордеев, В. Н. Волошина. — 2-е изд., испр. и доп. — М. : Издательство Юрайт, 2019. — 501 с.
13. Мясникова, Н.А. Алгоритмы и структуры данных : учебное пособие / Мясникова Н.А. — Москва : КноРус, 2021. — 185 с.
14. Перлова, О.Н. Проектирование и разработка информационных систем: Учебник / О.Н. Перлова. - М.: Академия, 2018. - 272 c.
15. Федорова Г.Н. Разработка модулей программного обеспечения для компьютерных систем: учебник для студентов СПО /Г.Н. Федорова. 2-е изд., стер. - М.: Издательский центр «Академия»,2018. - 384с.
16. Грибов, В.Д. Экономика организации (предприятия): учебник /Грибов В.Д., Грузинов В.П., Кузьменко В.А. — Москва: КноРус, 2016. — 407 с. — (СПО). — ISBN 978-5-406-05026-2.
17. Растова, Ю.И. Экономика организации: учебное пособие / Растова Ю.И., Масино Н.Н., 2. Фирсова С.А., Шматко А.Д. — Москва: КноРус, 2016. — 200 с. — (СПО). — ISBN 978-5-406-04890-0.

# Приложения

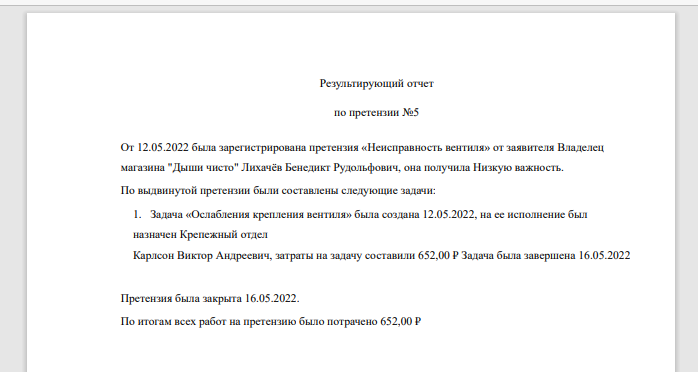
#### Приложение 1

Пустой бланк рекламации:



1. «Пример пустого бланка рекламации»

#### Приложение 2



1. «Вывод на печать»

#### Приложение 3

**Аннотация теста**

|  |  |
| --- | --- |
| **Название проекта** | RespiratiorDesktop |
| **Рабочая версия** | 4.0 |
| **Имя тестирующего** | Маркин Илья |
| **Дата(ы) теста** | 10.05.22 |

**Тестовый пример #1:**

|  |  |
| --- | --- |
| **Тестовый пример #** | Тест\_ввода\_данных: тестовый случай проверки данных программой. |
| **Приоритет тестирования** | низкий |
| **Заголовок/название теста** | Добавление рекламации |
| **Краткое изложение теста** | Кнопка сохранить должна стать активной. |
| **Этапы теста** | Запустить приложение  Войти (логин: markia пароль: 111111)  Нажать на панель меню Новая жалоба  Ввести номер документа (2), изложить жалобу (Неполадки в кислородной системе), указать инициатора (Директор Авиационного завода "Авиастрой" Песков Виктор Сергеевич) |
| **Тестовые данные** | логин: markia  пароль: 111111  Инициатор: Директор Авиационного завода "Авиастрой" Песков Виктор Сергеевич  Жалоба: Неполадки в кислородной системе  Номер документа: 2 |
| **Ожидаемый результат** | Кнопка «Сохранить» станет активной |
| **Фактический результат** | Кнопка «Сохранить» станет активной |
| **Статус** | Удачный |
| **Предварительное условие** | Авторизация пользователя. |
| **Постусловие** | Стабильное |

**Тестовый пример #2:**

|  |  |
| --- | --- |
| **Тестовый пример #** | Тест\_изменения\_данных: тестовый случай проверки изменения программой. |
| **Приоритет тестирования** | низкий |
| **Заголовок/название теста** | Ввод букв в поле количества |
| **Краткое изложение теста** | Программа позволяет изменить важность проекта после завершения |
| **Этапы теста** | Запустить приложение  Войти (логин: markia пароль: 111111)  Двойным щелчком нажать по жалобе «Неисправность вентиля»  В комбобоксе «Важность» изменит низкую на высокую |
| **Тестовые данные** | логин: markia  пароль: 111111  Жалоба: Неисправность вентиля  Важность: Низкая/Высокая |
| **Ожидаемый результат** | Комбобокс недоступен |
| **Фактический результат** | Важность меняется визуально, но изменения не фиксируются в БД. |
| **Статус** | Неудачный |
| **Предварительное условие** | Авторизация пользователя. |
| **Постусловие** | Стабильное |

**Тестовый пример #3:**

|  |  |
| --- | --- |
| **Тестовый пример #** | Тест\_печати\_отчета: тестовый случай печати отчета |
| **Приоритет тестирования** | средний |
| **Заголовок/название теста** | Вывод заказа |
| **Краткое изложение теста** | Программа выводит заказ |
| **Этапы теста** | Запустить приложение  Войти (логин: markia пароль: 111111)  Двойным щелчком нажать по жалобе «Неисправность вентиля»  Нажать на кнопку «Отчет»  Нажать на кнопку «Печать» |
| **Тестовые данные** | логин: markia  пароль: 111111  Жалоба: Неисправность вентиля |
| **Ожидаемый результат** | Отчет печатается |
| **Фактический результат** | Отчет печатается |
| **Статус** | Удачный |
| **Предварительное условие** | Авторизация пользователя |
| **Постусловие** | Стабильное |

1. SQL - structured query language (язык структурированных запросов) [↑](#footnote-ref-1)
2. LINQ (Language-Integrated Query) - язык запросов к источнику данных/ [↑](#footnote-ref-2)
3. XML - eXtensible Markup Language (расширяемый язык разметки). [↑](#footnote-ref-3)