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](#_Toc105410795)

[1. Разработка системного проекта 6](#_Toc105410796)

[1.1 Назначение разработки 6](#_Toc105410797)

[1.2 Требования к функциональным характеристикам 6](#_Toc105410798)

[1.2.1 Состав выполняемых функций 6](#_Toc105410799)

[1.3 Требование к надежности и безопасности 7](#_Toc105410800)

[1.4 Требования к составу и параметрам технических средств 7](#_Toc105410801)

[1.5 Требования к информационной и программной совместимости. 8](#_Toc105410802)

[2. Разработка технического проекта 9](#_Toc105410803)

[2.1 Обоснование выбора CASE-средств 9](#_Toc105410804)

[2.2 Проектирование модели данных 10](#_Toc105410805)

[2.3 Детальное проектирование программного приложения 12](#_Toc105410806)

[2.4 Функциональная схема программного приложения 14](#_Toc105410807)

[3. Реализация 15](#_Toc105410808)

[3.1 Обоснование выбора средств разработки 15](#_Toc105410809)

[3.2 Руководства программиста 16](#_Toc105410810)

[3.3 Руководство пользователя 33](#_Toc105410811)

[4. Тестирование и отладка 40](#_Toc105410812)

[4.1 Виды тестирования 40](#_Toc105410813)

[4.2 Отладка программного приложения 41](#_Toc105410814)

[5. Расчет базовой себестоимости разрабатываемого продукта 43](#_Toc105410815)

[5.1 Исходные данные 43](#_Toc105410816)

[5.2 Расчёты затрат на выполнение программы 43](#_Toc105410817)

[5.3 Расчет отчислений на социальное страхование и обеспечение 44](#_Toc105410818)

[5.4 Расчет базовой себестоимости компьютерного продукта 44](#_Toc105410819)

[Заключение 47](#_Toc105410820)

[Список литературы 47](#_Toc105410821)

[Приложения 47](#_Toc105410822)

# Введение

Наиболее важным показателем деятельности любого производственного предприятия является объем производства и продажи готовой продукции. Так как условия производственных возможностей ограничены и спрос является относительно неограниченным, то объем производства продукции выступает на первом месте. Но по мере того, как происходит насыщение рынка и усиливается конкуренция, определяющим звеном при разработке производственной программы здесь является возможный объем продаж, а не наоборот, когда производство определяет объем. Предприятию следует производить такие товары и в таком количестве, которое оно в силах реально реализовать Задачей каждого предприятия является продажа своей продукции потребителю.

В процессе продажи происходит выявление полезности и качества продукции. Кругооборот средств завершается продажей продукции. Предприятие возмещает свои издержки, которые связаны связанные с выпуском и продажей продукции, и реализует чистый доход, который создан в производстве, он частично переходит в государственный бюджет и частично остается с предприятием в форме прибыли. Для организаций область материального производства является основным элементом предпринимательской деятельности, и одним из основных этапов в достижении цели такой деятельности является систематическая прибыль. Бухгалтерский учет должен обеспечивать систематический контроль над производством, отгрузкой и продажей продукции, связанными с ней расходами и финансовыми результатами, состоянием расчетов с покупателями и клиентами. Основная часть задолженности включает в себя обязательства по расчетам с покупателями и заказчиками за отгруженные товары, готовую продукцию, выполненные работы и оказанные услуги.

Актуальность и значимость рассматриваемой проблемы учета производства и продажи готовой продукции заключается в том, что выпуск готовой продукции, выполнение работ или предоставление услуг является результатом деятельности любого производственного предприятия или организации. В результате стоимость готовой продукции, работ, услуг идет от сферы производства до сферы обращения. Учет готовой продукции требует больших затрат трудоемкости несмотря на такое многообразие существующих хозяйственных операций. Одной из главных задач бухгалтерского учета готовой продукции является правильная организация учета, которая могла бы позволить своевременно получить информацию о ходе поступления товаров, о том, как выполняются договорные обязательства между поставщиками и получателями продукции, о состоянии товарных запасов, а также как осуществляется ход отгрузки ценностей и происходит контролем над их сохранностью.

Цель исследования – проанализировать бизнес-процессы, происходящие в организации розничной торговли, 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;

3. Расчет базовой стоимости программного продукта;

4. Разработка полной и понятной документации руководства пользователя и программиста.

Методы:

1. Платформа 1C:Предприятие 8.3 учебная версия – для создания конфигурации приложения;

2. Microsoft Visio 2016 – для проектирования предметной области.

3. Excel для поведение расчетов экономической части.

Результатом работы является рабочее приложение для по учету готовой продукции на предприятия.

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

Первый раздел посвящена 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 BPWin 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. Описана библиотека регрессивных тестов, 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, расчет амортизации продукта.

В заключении описываются основные результаты выпускной квалификационной работы.

Приложение содержит …..листинги основных программных модулей, используемых в разработанной системе, а также диаграммы и схемы…(НАПИСАТЬ СВОЕ).

Общий объем работы 59 страницы. Список литературы содержит 15 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. Пользователями 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 (ФИО и название организации), содержание обращения, устанавливается важность. Далее рекламация разбивается на задачи. При составлении задачи необходимо указать название, которое описывает действия для решения проблемы, сроки выполнения задачи, а также назначить исполнителя. 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 (название задачи, даты начала и завершения задачи, и затраты), в конце указывается общее количество затрат на устранение данного несоответствия.

## Требования к функциональным характеристикам

### Состав выполняемых функций

Автоматизированная информационная система должная обеспечивать выполнение следующих функций:

* Регистрацию рекламации при обращении клиента;
* Назначение задач для решения несоответствия;
* Расчет затрат на решение 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 нормально функционировать при бесперебойной работе компьютера пользователя.

* При возникновении ошибок или сбоев в работе компьютера восстановление нормальной 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 – 3,1 Ггц, объем оперативной памяти 4 Гб, объем свободного дискового пространства 500 Мб, разрешение монитора 1366х768. Для печати отчета необходим принтер.

**Конфигурация Компьютера**

|  |  |
| --- | --- |
| Процессор | Intel Core i3 3240 3,1 ГГц |
| Память | От 4 гб |
| Разрешение экрана | От 1366х768 |
| Устройства ввода | Клавиатура, мышь |
| Дисковое пространство | От 1 Гб |

## Требования к информационной и программной совместимости

Программа должна работать на операционной системе Windows 10. Все формируемые отчеты должны иметь возможность печати, а также экспорта в Microsoft Word.

Для работы с базой данных и приложением, на рабочем компьютере должно быть установлено следующее программное обеспечение:

* База данных – Microsoft SQL Server 2019.
* Утилита для подключения и управления БД - SQL Server Management Studio 18.
* Программа для связи с БД – Visual Studio от 2019 или выше.

# Разработка технического проекта

## Обоснование выбора CASE-средств

CASE средства (Computer - Aided Software Engineering) – это инструмент, 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. 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.

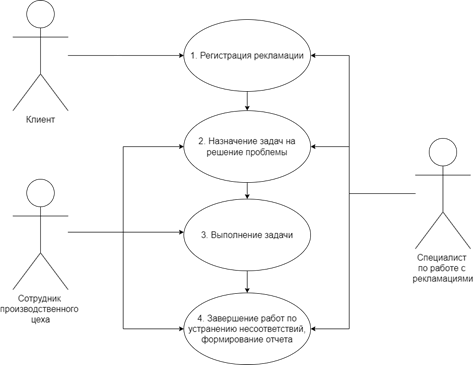
Диаграмму можно сохранить как растровое или векторное изображение, как XML-файл или HTML-файл

## Проектирование модели данных

Диаграмма прецедентов – диаграмма, отражающая отношения между акторами и прецедентами

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.

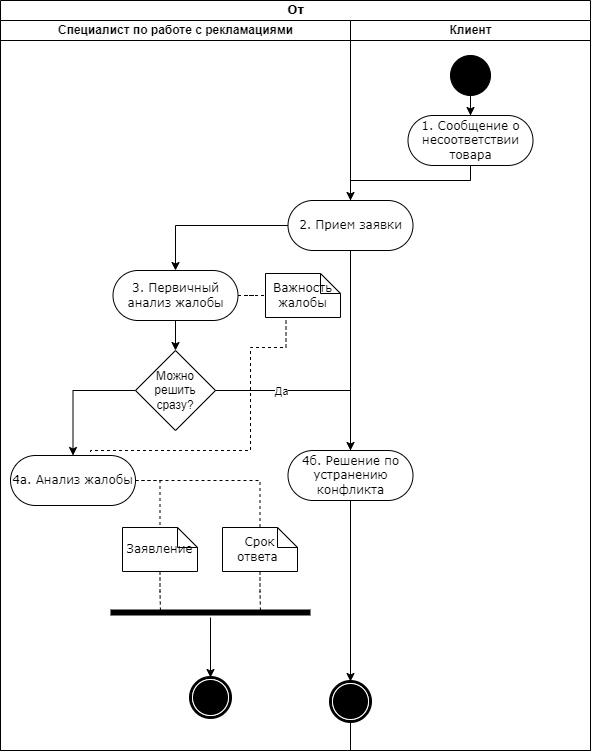
Таблица№1 «Определение бизнес-процессов»

|  |  |  |
| --- | --- | --- |
| 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



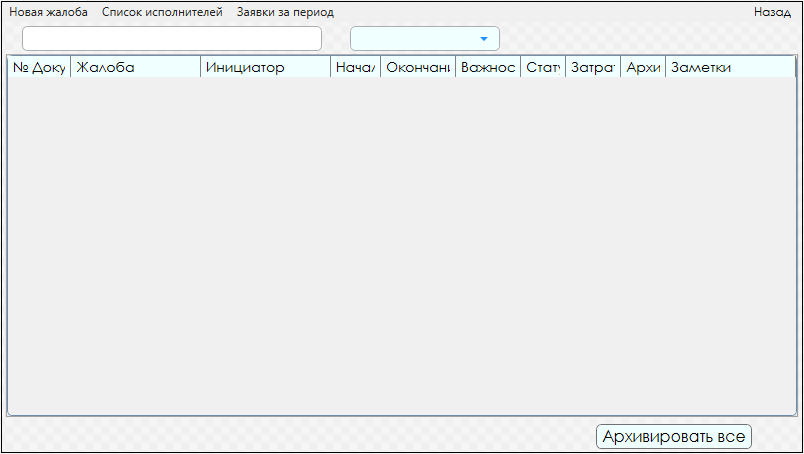
1. «Диаграмма действий»

Таблица 2 «Операция регистрация рекламации»

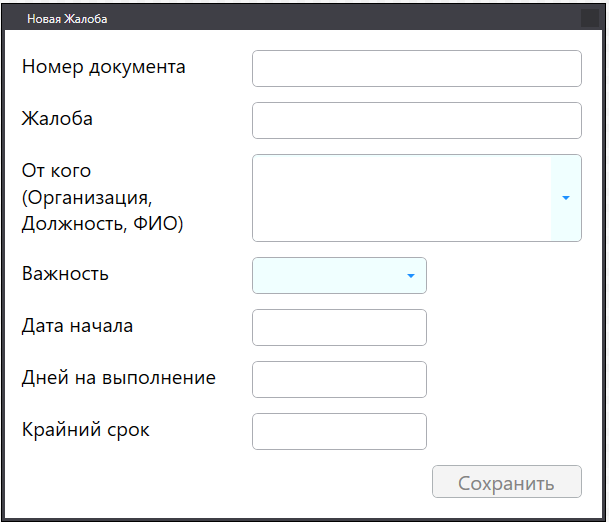
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Диаграмма и номер операции на диаграмме | Операция | Как часто | Исполнитель | Входящие документы | Исходящие документы |
| 1. | 2. | 3. | 4. | 5. | 6. |
| 1. Регистр \_Рекл | 1. Сообщение о несоответствии товара | При обнаружении несоответствия | 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.

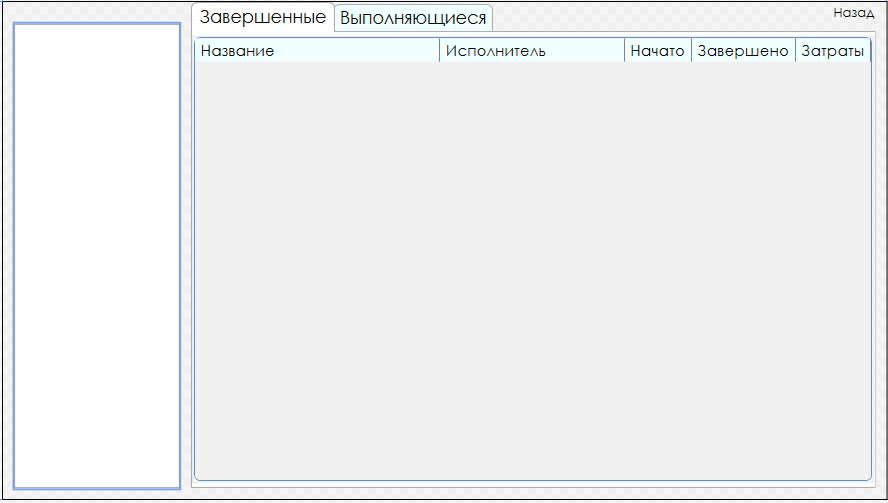


1. «Конструктор основной страницы»



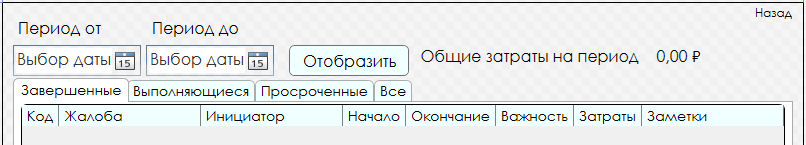
1. «Конструктор окна добавления новой рекламации»

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



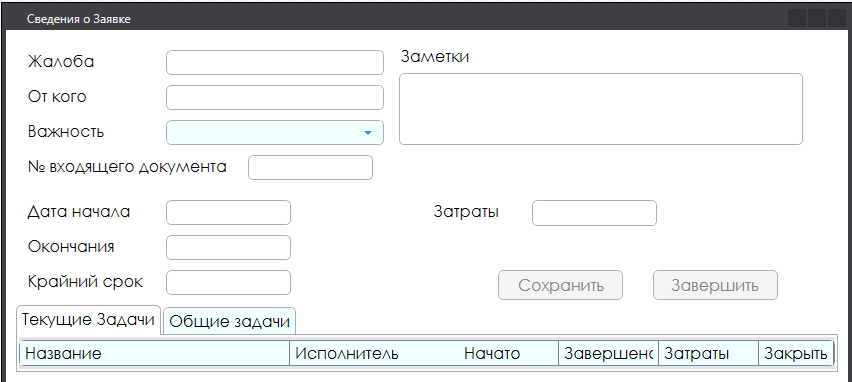
1. «Конструктор страницы «Статистика Исполнителей»»

На странице статистики за период отображаются рекламации за указанный период.

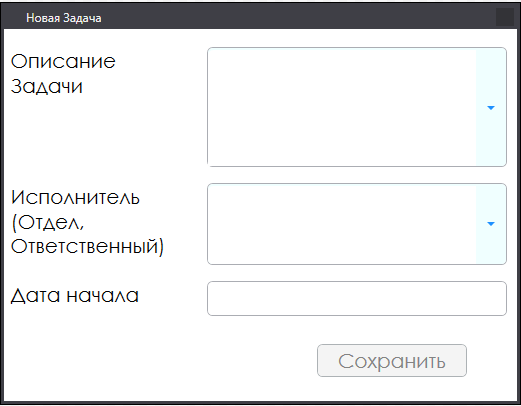


1. «Конструктор страницы «Статистика за период»»

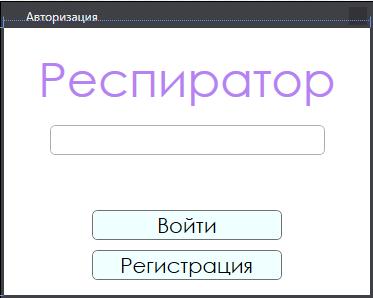
На окне информации о рекламации отображается вся информация о рекламации, добавляются задачи для устранения несоответствий.



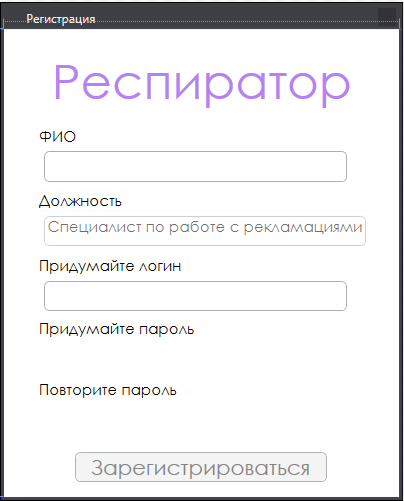
1. «Конструктор страницы информации о рекламации»



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 2019.

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.

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 2019 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 2018, 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 - 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 i5 3,2 ГГц |
| Память | От 8 гб |
| Разрешение экрана | От 1920x1080 |
| Устройства ввода | Клавиатура, мышь |
| Дисковое пространство | От 1 Гб |
| **Оптимальные требования** | |
| Процессор | Intel Xeon 2.60 ГГц |
| Память | От 16 гб |
| Разрешение экрана | От 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 4.7.2.**

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

**MS SQL Server Management Studio 2018** - 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.

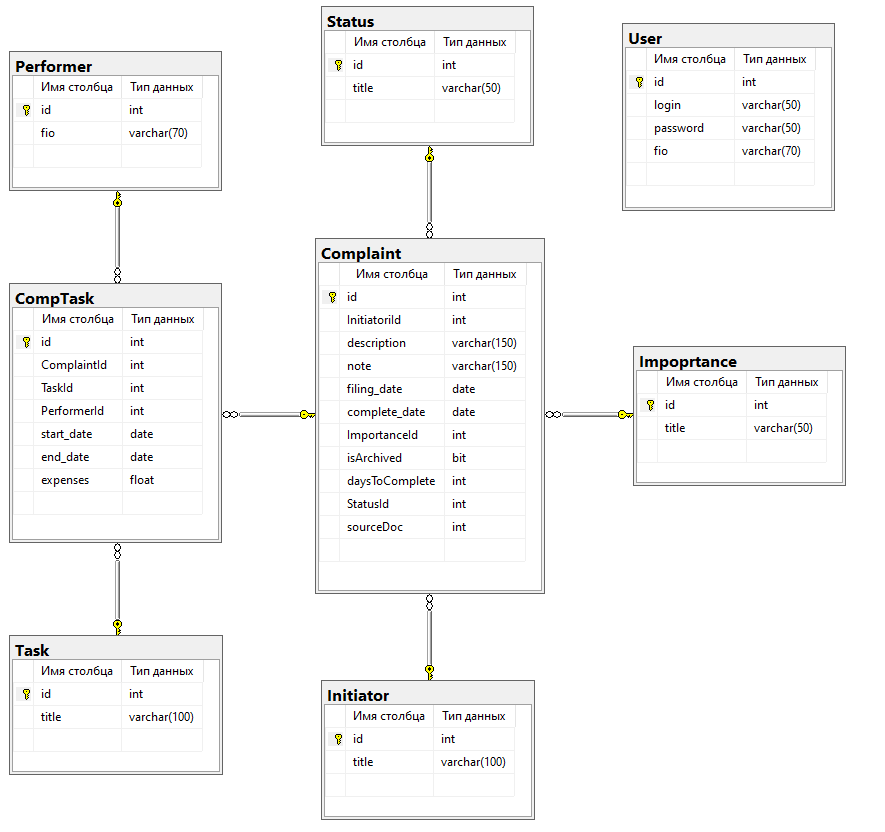
**MS SQL Server** - система управления реляционными базами данных (РСУБД), разработанная корпорацией Microsoft. Основной используемый язык запросов — Transact-SQL, создан совместно Microsoft и Sybase. Transact-SQL является реализацией стандарта ANSI/ISO по структурированному языку запросов (SQL) с расширениями. 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.

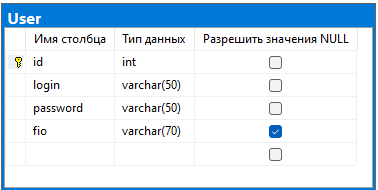
**ADO.NET Entity Framework** - объектно-ориентированная технология доступа к данным. Предоставляет возможность взаимодействия с объектами как посредством LINQ в виде LINQ to Entities, так и с использованием Entity SQL. Для облегчения построения web-решений используется как ADO.NET Data Services, так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

**Проектирование и разработка баз данных**



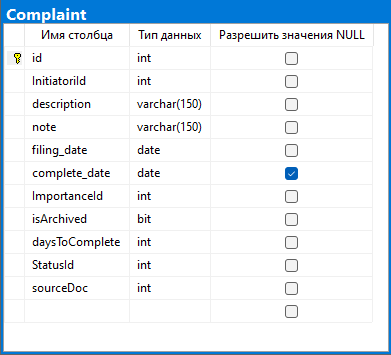
1. «Схема БД»

Таблица пользователей программы:



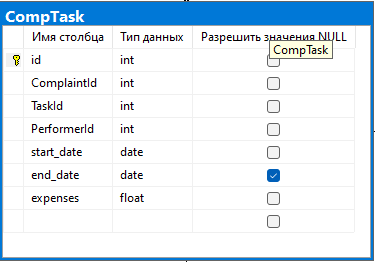
1. «Пользователь»

Таблица заявок на несоответствие:



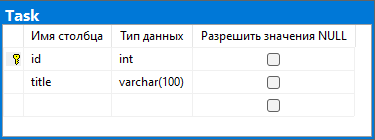
1. «Рекламация»

Таблица назначенных задач на выполнение рекламации:

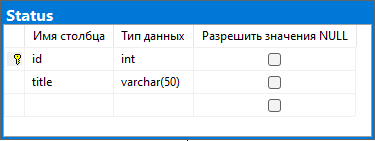


1. «Задачи на Рекламацию»

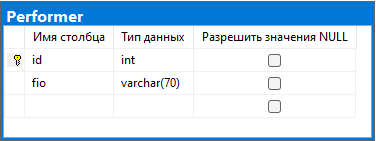
Созданы Справочные таблицы:



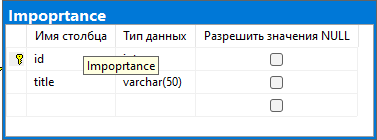
1. «Задачи»



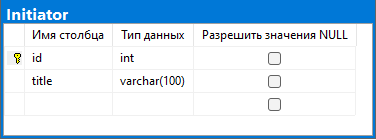
1. «Статус»



1. «Исполнитель»

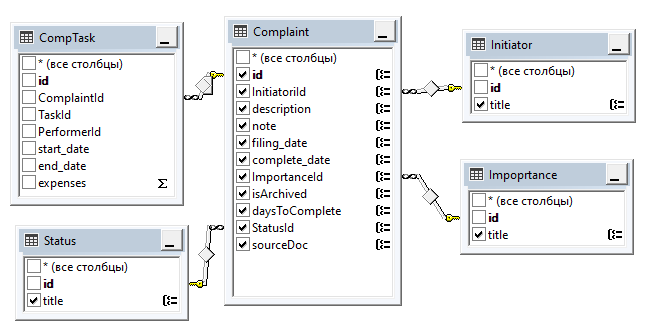


1. «Важность»



1. «Инициатор»

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

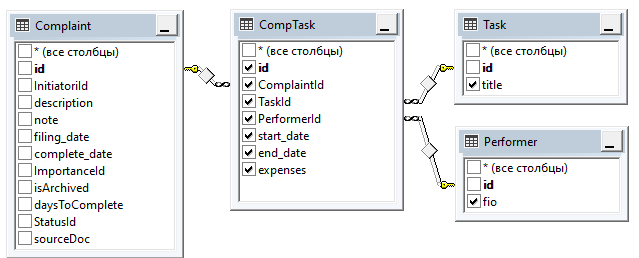
dbo.Status ON dbo.Complaint.StatusId = dbo.Status.id LEFT OUTER JOIN

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

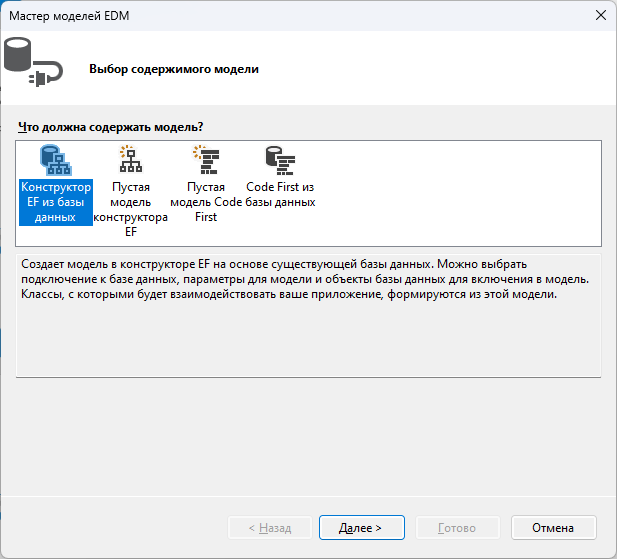
dbo.CompTask ON dbo.Complaint.id = dbo.CompTask.ComplaintId INNER JOIN

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

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

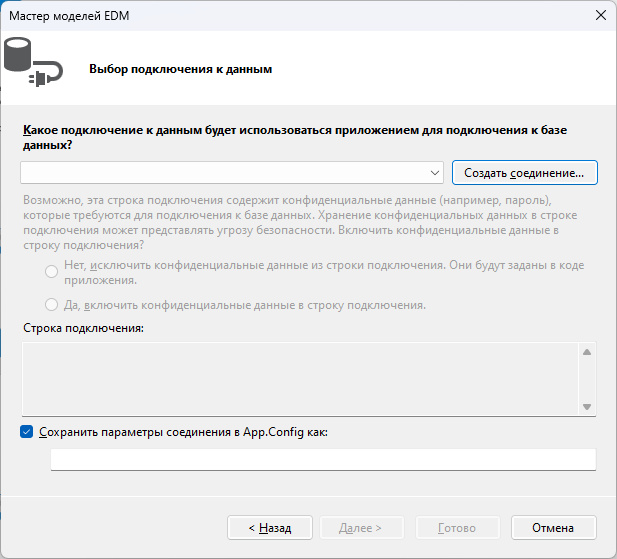
**Подключение к базе данных**

В Visual Studio открыть источник данных, нажать на «Конструктор EF из базы данных» -> «Далее»



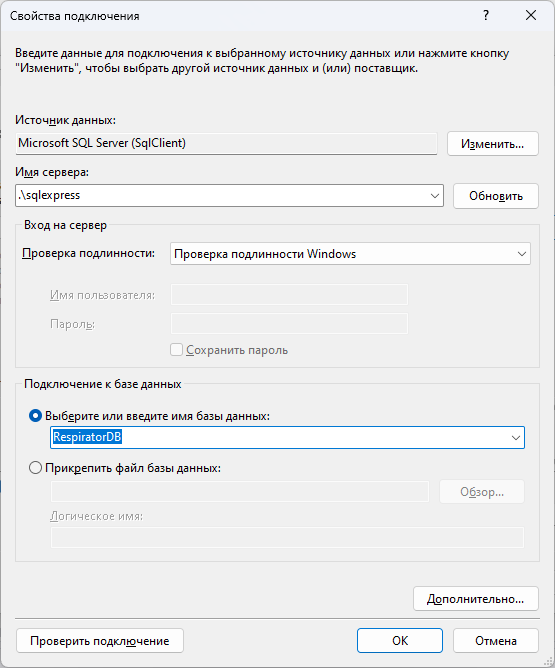
1. «Содержимое модели»

На окне выбора подключения к БД, нажать на «Создать подключение»



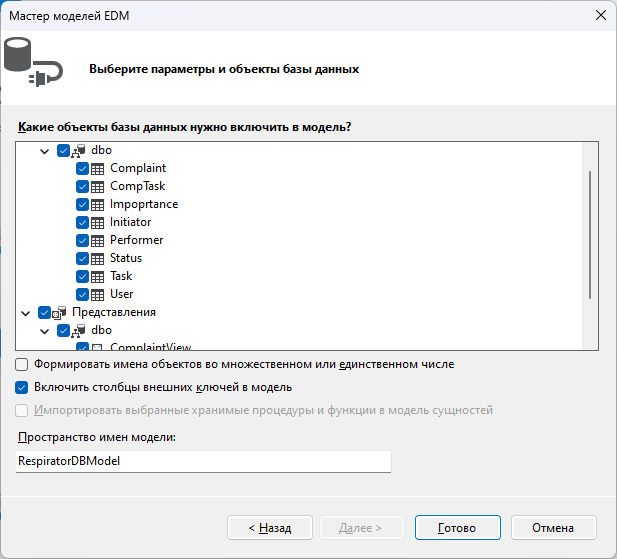
1. «Выбор подключения»

Добавить подключение



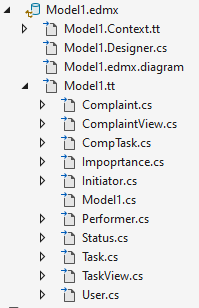
1. «Выбор базы»

Выбрать таблицы



1. «Завершение»

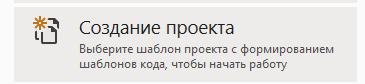
Итог добавления



1. «Модель»

**Создание проекта**

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



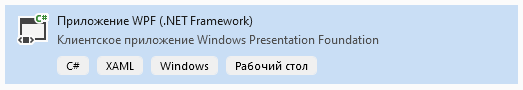
1. «Создание проекта»

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



1. «Создание проекта»

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

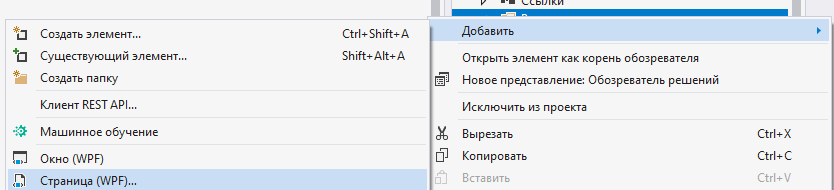


1. «Создание проекта»

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

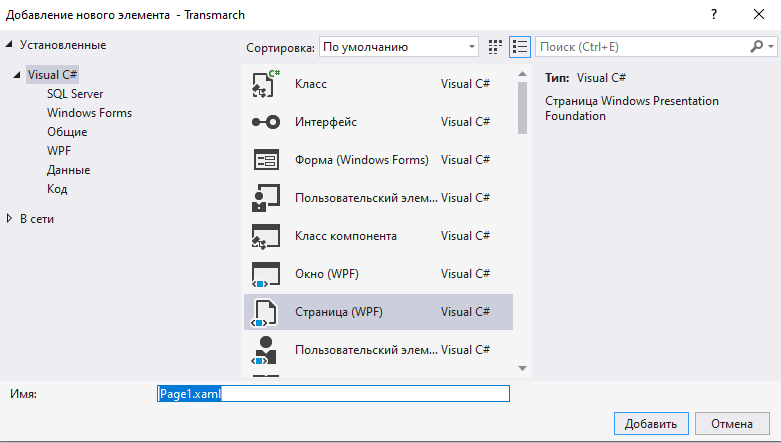
**Страницы**

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.

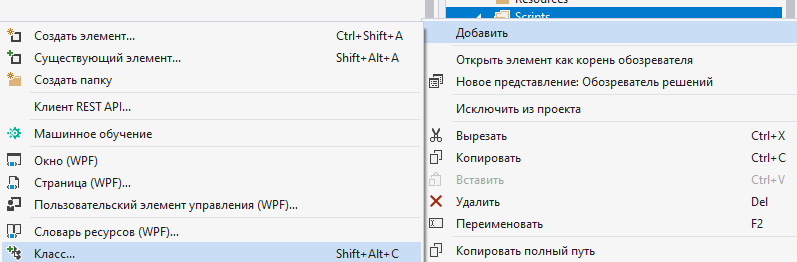


1. «Создание страницы»

**Классы**

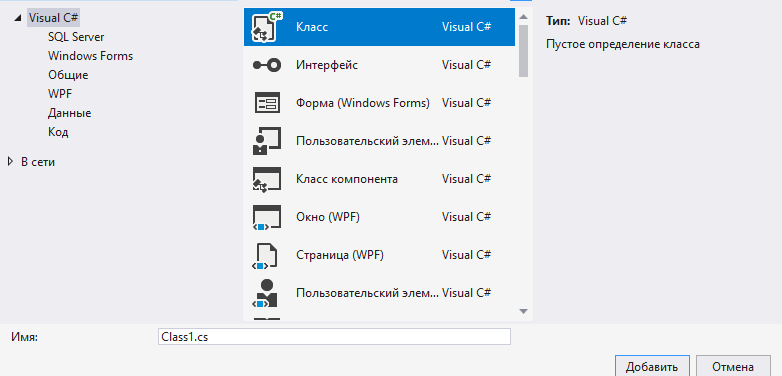
Так же в программе нужно реализовать, классы для осуществления переходов и работы с базой данных.

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



1. «Создание класса»

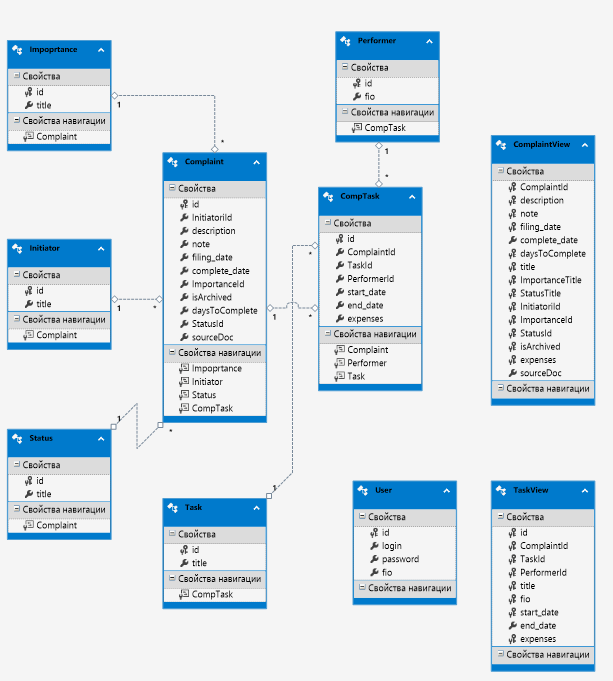
В появившимся окне, выбрать **Класс** и задать ему имя.



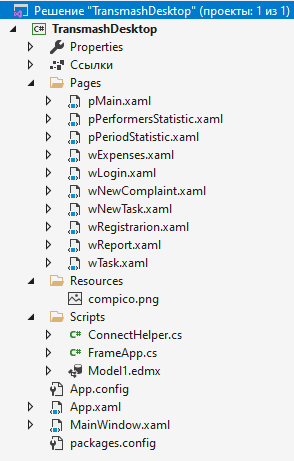
1. «Создание класса»

**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 2017.



1. «Диаграмма»



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>

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)

{

Complaint complaint = ConnectHelper.entObj.Complaint.First(c => c.id == comp.ComplaintId);

complaint.note = tbNote.Text;

complaint.ImportanceId = cbImport.SelectedIndex + 1;

ConnectHelper.entObj.SaveChanges();

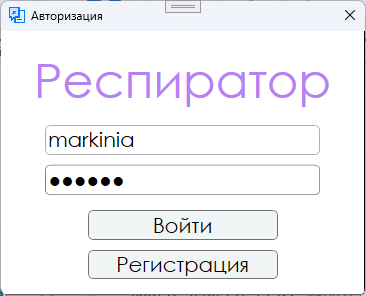
MessageBox.Show("Претензия обновлена!", "Успешное сохранение", MessageBoxButton.OK, MessageBoxImage.Information);

Update();

}

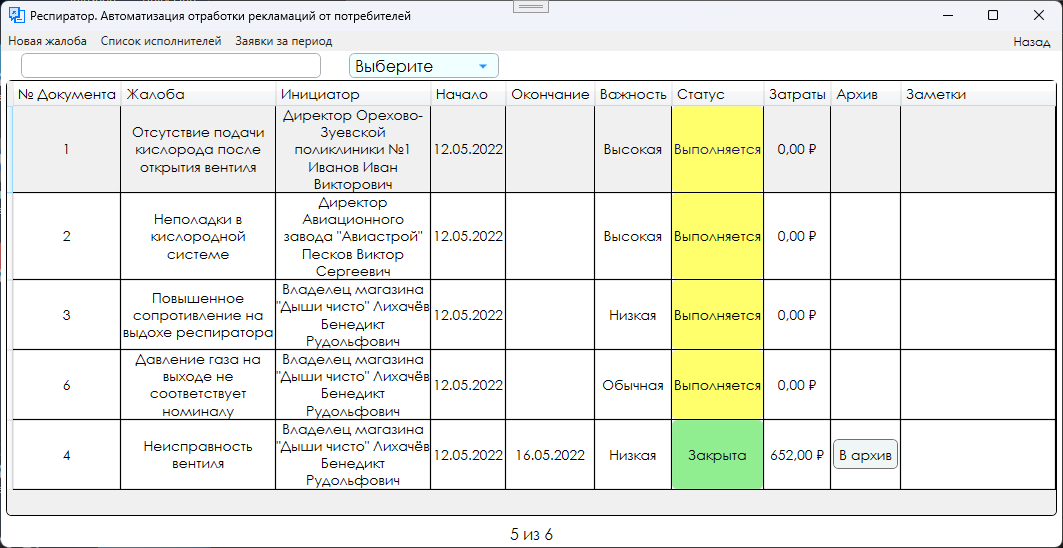
## Руководство пользователя

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, придумать логин и пароль, после чего ввести данные в полях авторизации.



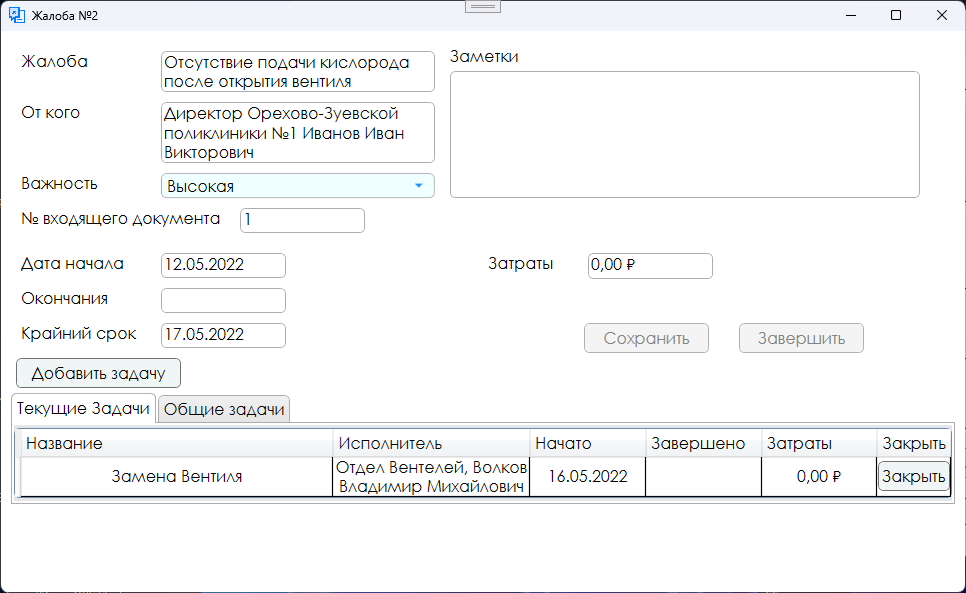
1. «Окно авторизации»

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.



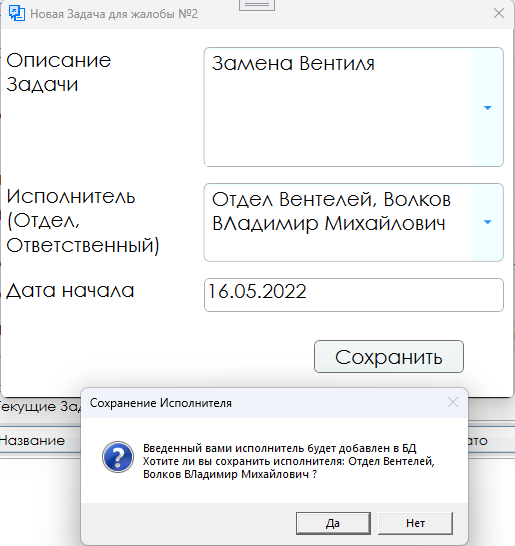
1. «Страница с рекламациями»

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.



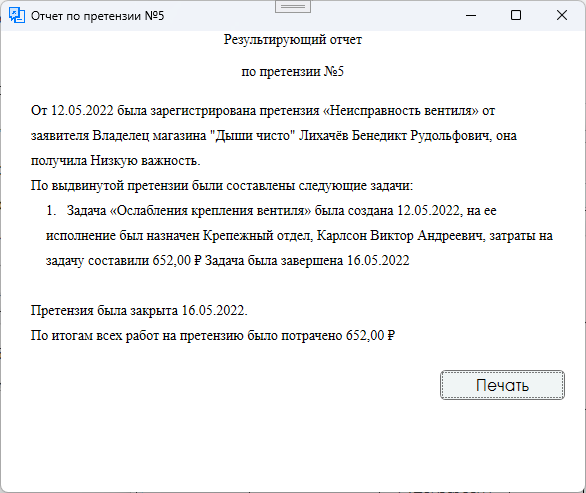
1. «Окно Сведений и задач по рекламации»

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.



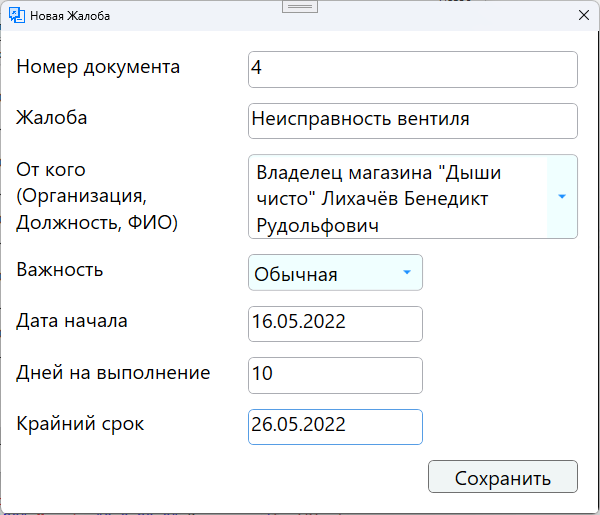
1. «Окно Добавления задачи»

Страница отчета со всеми данными о задачи, появляется после клика по кнопке «отчет» на окне сведений о заявке. Распечатать отчет можно нажав по кнопке «Печать».



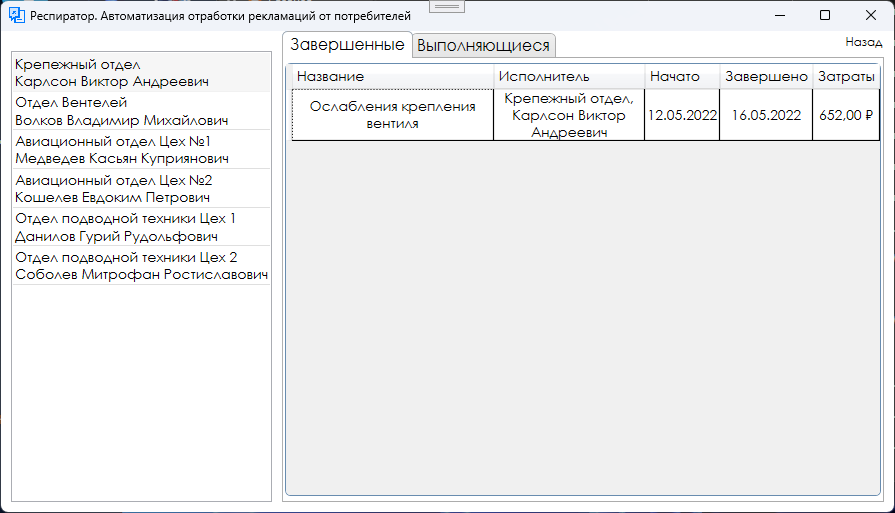
1. «Окно отчета по претензии»

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.



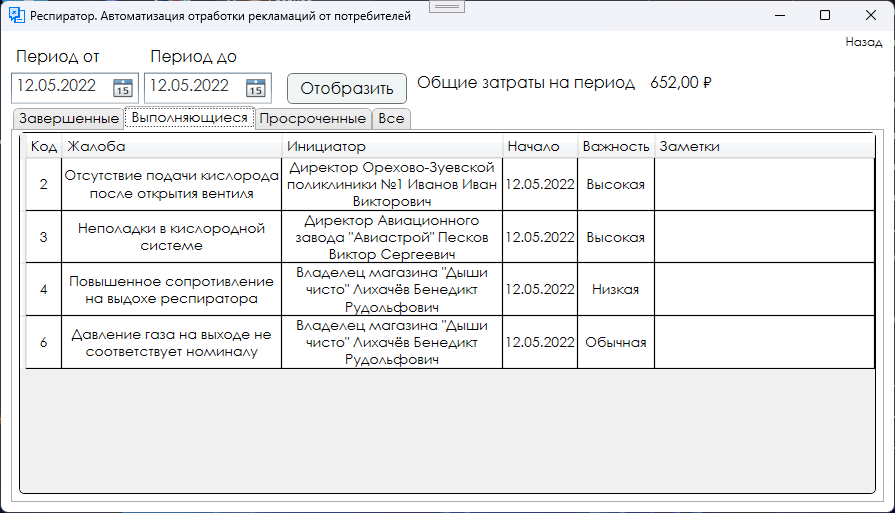
1. «Окно Регистрации рекламации»

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.



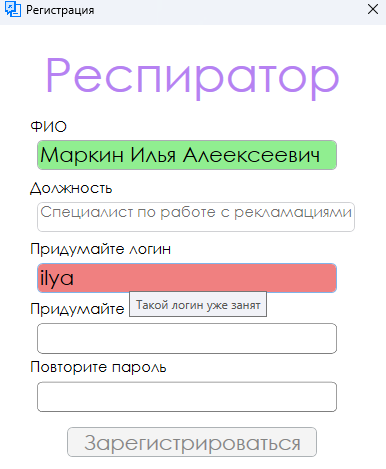
1. «Страница Статистики по исполнителям»

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.



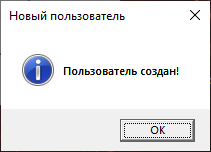
1. «Страница Статистики за период»

В программе имеются сообщения для пользователя

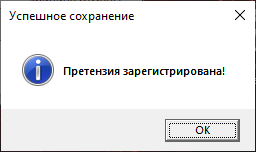


1. «Индикаторы и подсказки неверности ввода»

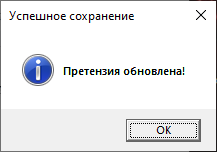
Сообщения об успешном действии:



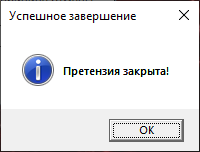
1. «Успешная регистрация»



1. «Уведомление о успешном сохранении»

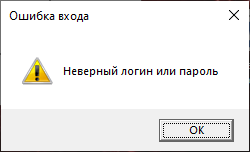


1. «Уведомление об успешном обновлении»

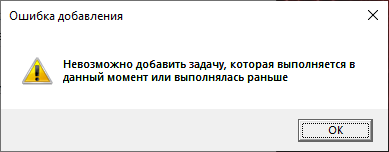


1. «Уведомление об успешном завершении претензии»

Сообщения об ошибке:

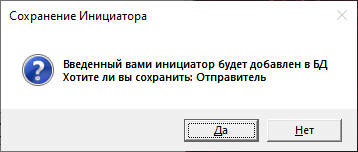


1. «Ошибка при вводе пароля»

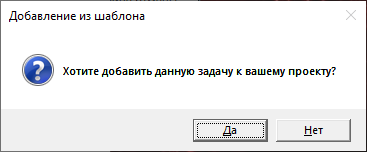


1. «Ошибка при повторном добавлении задачи»

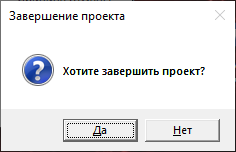
Сообщения с подтверждением действий:



1. «Подтверждение о добавлении нового инициатора»



1. «Подтверждение о добавлении задачи»



1. «Подтверждение о завершении претензии»

# Тестирование и отладка

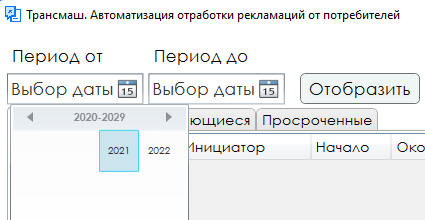
## Виды тестирования

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.

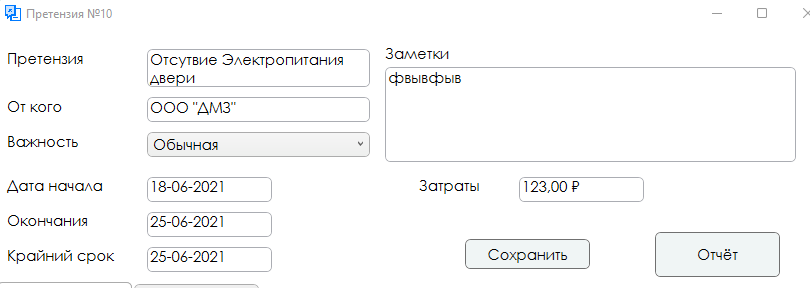
Примеры ошибок:

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:

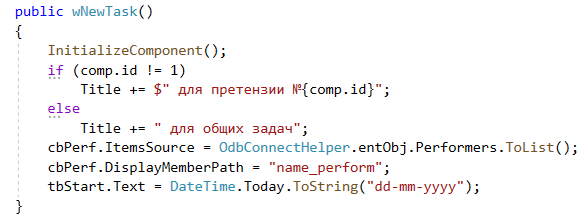


1. «Кнопка сохранения доступна у закрытой заявки»
2. Белый ящик - метод тестирования программного обеспечения, который предполагает, что внутренняя структура/устройство/реализация системы известны тестировщику.

Пример:

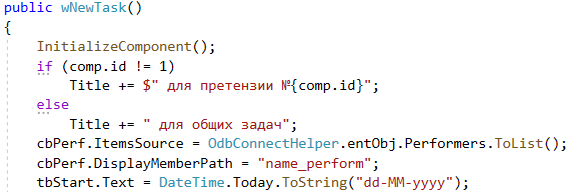
2.1) Вывод в поле «дата начала задачи» минут вместо месяца

До



1. «Вывод минут в поле даты»

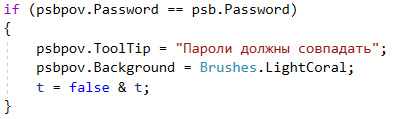
После



1. «Вывод месяца в поле даты»

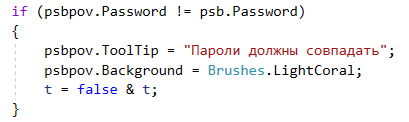
2.2) При правильном повторении пароля программа не позволяла зарегистрироваться пользователю.

До



1. «Выбор даты периода»

После



1. «Выбор даты периода»

## Отладка программного приложения

Термин отладка может иметь разные значения, но в первую очередь он означает устранение ошибок в коде.

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

Таблица №3 «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 №4.

Таблица №4 «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 метаданных | 1 | 1 | 1 |
| Разработка интерфейса | 5 | 15 | 18 |
| Программирование | 15 | 18 | 20 |
| Тестирование и отладка | 3 | 5 | 5 |
| Итого | 25 | 41 | 46 |

## 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 – 26 000/21\*46= 56 950 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 №5 «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 | 12 529 | 1 651,55 | 2 904,45 | 17 085 |

Формулы расчета для таблицы №5:

ПФР = ОТ \* 22% = 56 950 руб \* 22% = 12 529‬ руб.

ФСС РФ = ОТ \* 2,9% = 56 950 руб \* 2,9% = 1 651,55 руб.

ФОМС = ОТ \* 5,1% = 56 950 руб. \* 5,1% = 2 904,45 руб.

ОСО = ПФР + ФССРФ + ФОМС = 12 529‬ руб + 1 651,55 руб + = 2 904,45 руб. = 17 085 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 №6 «Расчёт базовой себестоимости»

|  |  |  |
| --- | --- | --- |
| **№** | **Статьи затрат** | **Сумма в рублях** |
| 1 | Фонд оплаты труда (ФОТ) | 74 035 |
| 2 | Косвенные расходы (КР) 10% от ФОТ | 7 403,5 |
| 3 | Расходы на продажу (РП) (ФОТ + КР) \* 10% | 8 143,85 |
| 4 | Полная себестоимость (ПС) | 89 582,35 |

Формулы расчета для таблицы №6:

ФОТ=ОТ+ОСО= 56 950 руб + 17 085 ‬руб = 74 035 ‬‬руб.

В данной организации косвенные расходы составляют 10% от ФОТ.

КР=ФОТ\*10%= 74 035‬‬ руб \*10% = 7 403,5 руб.

РП=(ФОТ+ Р)\*10% = (74 035‬‬ руб. + 7 403,5 руб.) \* 10% = ‬8 143,85‬ руб.

ПС=ФОТ+КР+РП= 74 035‬‬ руб. + 7 403,5 руб + 8 143,85 = 89 582,35 руб.

Выгода = стоимость аналогичного программного продукта на рынке – себестоимость.

Аналоги данного программного продукта включая базовую версию 1С: Предприятия 8.3 начинаются с 163 700 рублей, в зависимости от цены часа работы специалиста при доработки нужной конфигурации.

Итоговая выгода с разработки = 163 700 руб. – 89 582,35 руб.

Выгода = 74 117,65 рублей.

По итогам экономического расчёта полная себестоимость составила 89 582,35 руб. и можно сделать вывод, что продукт полностью окупает затраты на его разработку, а также предоставляет выгоду от разработки в размере 74 117,65 рублей.

# Заключение

В ходе проведенного исследования разработан проектный модуль для информационной системы «Заказ в снабжении расходных материалов для производства» 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 рабочего места для бригадиров. Со следующими функциональными характеристиками:

• Формирование заказа на производственный участок

• Создание, удаление и применение шаблонов бригадирами участков

• Добавление и удаление товаров в заказ

• Печать заказов по производственным участкам, общий и итоговый

• Возможность просматривать историю заказов за прошедший год

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 2019

• Microsoft SQL Server 18

• MS Office 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.

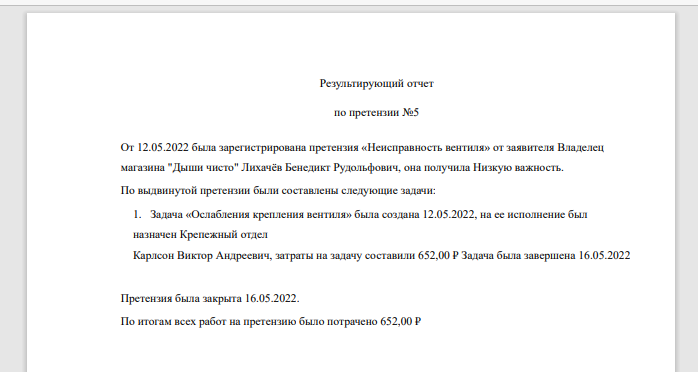
Для защиты приложения создана форма авторизации пользователей. Приложение разработано в установленные сроки и удовлетворяет потребностям заказчика.

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

# Список литературы

# Приложения

Приложение 1



1. «Вывод на печать»

Приложение 2

**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 (1), изложить жалобу (Неполадки в кислородной системе) |
| **Тестовые данные** | логин: markia  пароль: 111111  материал: Канцелярские ножи  кол-во: 12 |
| **Ожидаемый результат** | Кнопка «подтвердить» станет активной |
| **Фактический результат** | Кнопка «подтвердить» станет активной |
| **Статус** | Удачный |
| **Предварительное условие** | Авторизация под аккаунтом начальника цеха, ввод данных |
| **Постусловие** | Стабильное |

**Тестовый пример #2:**

|  |  |
| --- | --- |
| **Тестовый пример #** | Тест\_ввода\_данных2: тестовый случай проверки данных программой. |
| **Приоритет тестирования** | низкий |
| **Заголовок/название теста** | Ввод букв в поле количества |
| **Краткое изложение теста** | Не позволение ввода некорректных данных |
| **Этапы теста** | Запустить приложение  Войти под начальником цеха (логин: Vlad пароль: 1234)  Нажать на кнопку добавления  Использовать двойной щелчок чтобы открыть окно добавления  Ввести кол-во материала |
| **Тестовые данные** | логин: Vlad  пароль: 1234  материал: Канцелярские ножи  кол-во: укпукп |
| **Ожидаемый результат** | Поле не позволяет ввести некорректные данные |
| **Фактический результат** | Поле не позволяет ввести некорректные данные |
| **Статус** | Удачный |
| **Предварительное условие** | Авторизация под аккаунтом начальника цеха, ввод данных |
| **Постусловие** | Стабильное |

**Тестовый пример #3:**

|  |  |
| --- | --- |
| **Тестовый пример #** | Тест\_вывода\_заказа: тестовый случай вывода данных о заказе по участку |
| **Приоритет тестирования** | средний |
| **Заголовок/название теста** | Вывод заказа |
| **Программа** | Программа выводит заказ |
| **Этапы теста** | Запустить приложение  Войти под начальником отдела снабжения (логин: Vova пароль: 123456)  Выбрать заказ с цеха №2  Нажать кнопку вывода в Excel  Сохранить файл на рабочий стол |
| **Тестовые данные** | логин: Vova  пароль: 123456  Заказ с цеха №2 |
| **Ожидаемый результат** | Вывод заказа |
| **Фактический результат** | Вывод заказа |
| **Статус** | Удачный |
| **Предварительное условие** | Авторизация под аккаунтом начальника отдела снабжения |
| **Постусловие** | Стабильное |