|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Министерство образования и науки Пермского края  государственное бюджетное профессиональное образовательное учреждение  «Пермский химико-технологический техникум»   |  |  | | --- | --- | |  | Специальность 09.02.07 «Информационные системы и программирование» |   **Отчет по учебной практике УП.01**  **ПМ.01 «**Разработка модулей программного обеспечения для компьютерных систем»    Сроки прохождения практики 05 ноября –11 ноября 2024 года (рассредоточено)  Место прохождения практики ГБПОУ «ПХТТ»   |  |  | | --- | --- | |  | Выполнил студент гр. ИС-22-11 / \_\_\_\_\_\_ / Р.В. Глухов/  группа подпись… расшифровка подписи  Руководитель практики  ГБПОУ «ПХТТ» \_\_\_\_\_\_\_\_\_\_\_\_ / Е.Г. Рыкалова /  подпись расшифровка подписи |   2024 |

**СОДЕРЖАНИЕ ОТЧЕТА ПО УЧЕБНОЙ ПРАКТИКЕ**

|  |  |
| --- | --- |
| Характеристика | **Стр 3** |
| Программа практики | **Стр 5** |
| 1. Отчет о выполненных работах | **Стр 6** |
| * 1. Задание 1 | **Стр 6** |
| * 1. Задание 2 | **Стр 34** |
| Список литературы | **Стр 63** |
| Дневник прохождения практики | **Стр 64** |

# **ХАРАКТЕРИСТИКА**

Обучающийся (аяся) Глухов Руслан Владимирович Группы ИС-22-11

специальности СПО 09.02.07 «Информационные системы и программирование» проходил(а) учебную практику по профессиональному модулю **ПМ.01 «**Разработка модулей программного обеспечения для компьютерных систем» в период с «05» ноября 2024 г. по «11» ноября 2024 г (рассредоточено) на предприятии / в организации ГБПОУ «ПХТТ»

За время работы проявил(а) себя как ответственный/безответственный, исполнительный/неисполнительный, коммуникабельный/замкнутый, доброжелательный/наглый сотрудник.

В ходе выполнения всех видов работ обучающийся(аяся) показал(ла) сформированность следующих компетенций:

|  |  |
| --- | --- |
| **Наименование компетенции** | **Уровень (оптимальный/средний/ допустимый уровень)** |
| OK 0l. Выбирать способы решения задач профессиональной деятельности, применительно к различным контекстам. | оптимальный/средний/  допустимый / |
| ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности. | оптимальный/средний/  допустимый / |
| ОК 03. Планировать и реализовывать собственное профессиональное и личностное развитие, предпринимательскую деятельность в профессиональной сфере, использовать знания по финансовой грамотности в различных жизненных ситуациях. | оптимальный/средний/  допустимый / |
| ОК 04. Эффективно взаимодействовать и работать в коллективе и команде. | оптимальный/средний/  допустимый / |
| ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста. | оптимальный/средний/  допустимый / |
| ОК 06. Проявлять гражданско-патриотическую позицию, демонстрировать осознанное поведение на основе традиционных общечеловеческих ценностей, в том числе с учетом гармонизации межнациональных и межрелигиозных отношений, применять стандарты антикоррупционного поведения. | оптимальный/средний/  допустимый / |
| ОК 07. Содействовать сохранению окружающей среды, ресурсосбережению, применять знания об изменении климата, принципы бережливого производства, эффективно действовать в чрезвычайных ситуациях. | оптимальный/средний/  допустимый / |
| ОК 08. Использовать средства физической культуры для сохранения и укрепления здоровья в процессе профессиональной деятельности и поддержания необходимого уровня физической подготовленности. | оптимальный/средний/  допустимый / |
| ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках. | оптимальный/средний/  допустимый / |
| ПК 1.1. Формировать алгоритмы разработки программных модулей в соответствии с техническим заданием. | оптимальный/средний/  допустимый / |
| ПК 1.2. Разрабатывать программные модули в соответствии с техническим заданием. | оптимальный/средний/  допустимый / |
| ПК 1.3. Выполнять отладку программных модулей с использованием специализированных программных средств. | оптимальный/средний/  допустимый / |
| ПК 1.4. Выполнять тестирование программных модулей. | оптимальный/средний/  допустимый |
| ПК 1.5. Осуществлять рефакторинг и оптимизацию программного кода. | оптимальный/средний/  допустимый |
|  |  |

К работе относился\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Цели и задачи практики достигнуты / достигнуты не в полном объёме.

Руководитель практики от ГБПОУ «ПХТТ»

преподаватель, Рыкалова Елена Геннадьевна

|  |  |  |
| --- | --- | --- |
| Дата 11.11.2024 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | / Е.Г. Рыкалова /  Расшифровка подписи |

# **ПРОГРАММА УЧЕБНОЙ ПРАКТИКИ**

**по модулю ПМ.01 «**Разработка модулей программного обеспечения для компьютерных систем»

**по специальности 09.02.07 «Информационные системы и программирование»**

|  |  |  |
| --- | --- | --- |
| **№ п/п** | **Наименование вида работ** | **Количество часов** |
|  | Инструктаж по технике безопасности и пожарной безопасности.  Разработка библиотек. Отладка и тестирование | 8 |
|  | Разработка базы данных и импорт данных в базу данных | 4 |
|  | Разработка форм с загрузкой из базы данных. Установка фильтрации, добавление, изменение данных | 6 |
|  | Разработка форм учетом статусов в конкретных таблицах. Создание графиков | 6 |
|  | Разработка форм с проверкой полей с использованием регулярных выражений. Задание размеров изображений и проверка его веса | 4 |
|  | Разработка модульных тестов по работе с библиотекой | 4 |
|  | Разработка форм с загрузкой данных из json | 2 |
|  | Оформление отчетной документации. Зачет. | 2 |
|  | Инструктаж по технике безопасности и пожарной безопасности.  Разработка библиотек. Отладка и тестирование | 8 |

Руководитель практики от ГБПОУ «ПХТТ»

преподаватель, Рыкалова Елена Геннадьевна

|  |  |  |
| --- | --- | --- |
| Дата 11.11.2024 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | / Е.Г. Рыкалова /  Расшифровка подписи |

# **ОТЧЕТ О ВЫПОЛНЕННЫХ РАБОТАХ**

# **ЗАДАНИЕ 1**

Проанализировать предметную область, составить краткую спецификацию разрабатываемого модуля выделить входные и выходные данные; Краткую спецификацию можно рассмотреть в спецификации 1.

Сформировать основной алгоритм решения производства в виде блок-схемы в соответствии с описанием предметной области. Блок-схему можно рассмотреть на рисунке 1.

Детализировать в виде алгоритма одну из функций (расчета количества заявок, количества доставленной продукции). Детализированный алгоритм можно рассмотреть на рисунке 2.

Разработка объектов базы данных в соответствии с результатами анализа предметной области (ER можно рассмотреть на рисунке 4, с логической диаграммой можно ознакомиться на рисунке 3)

Подготовка к данных к импорту из файлов .xls или .xlsx , приведение к 3 нормальной форме согласно описанию предметной области

Формирование алгоритмов разработки программных модулей в соответствии с описанием предметной области

Разработка программных модулей в соответствии с техническим заданием.

Выполнение отладки программных модулей с использованием специализированных программных средств. Точка остановы 1 и точка остановы 2 иллюстрируют проделанную отладку.

Реализация базы данных в конкретной системе управления базами данных

Разработать систему согласно предметной области.

Протестировать систему в 3 условиях (нормальных, экстремальных и исключительных). Тест-кейсы 1 описывают все проведенные тесты.

Разработать руководство оператора. С выполненной документацией можно ознакомится в руководстве оператора 1.

**Описание предметной области**

Производственная компания «Мастер пол» выпускает напольные покрытия. Компания занимается производством и реализует свою продукцию через партнеров, которые доставляют продукцию компании до конечных потребителей.

С целью оптимизации деятельности компании разрабатывается система, в которой заказчик выделил несколько подсистем:

* продукция;
* склад и материалы;
* производство;
* сотрудники;
* партнеры.

В данной области определены следующие основные составляющие:

1. Партнеры: тип, наименование компании, юридический адрес, ИНН, ФИО директора, контактные данные (телефон, email), логотип, рейтинг, места продаж, история реализации продукции для последующего расчета скидок.

Партнеры покупают продукцию со скидками, которые зависят от общего объема их продаж. Скидки мотивируют партнеров к большему объему продаж. Партнеры реализуют товары в розничных и оптовых магазинах, интернетмагазинах, другим компаниям по продаже товаров и оказанию услуг.

2. Менеджеры: поиск и регистрация партнеров, решение об изменении рейтинга партнера, ведение истории изменений рейтинга, прием заявок от партнеров, формирование для них предложения на основании статистики истории продаж, полученной от аналитиков компании.

3. Заявка: если партнер ранее производил закупку продукции и предоставил данные о ее реализации, то на основании этих данных и текущих объемах готовой продукции менеджер формирует предложение. Если работа ведется с новым партнером, то менеджер предоставляет каталог готовой продукции и принимает заявку. Затем менеджер должен указать стоимость и дату производства каждой указанной в заявке единицы продукции. После согласования списка продукции с количеством, стоимостью и сроком изготовления заявка считается созданной.

Партнер может отменить заявку до момента внесения предоплаты. Если предоплата не поступает в течение трех дней, то заявка отменяется автоматически, о чем обязательно уведомляется партнер. После поступления предоплаты продукция, указанная в заявке и отсутствующая на складе в нужном количестве, запускается в производство. По готовности всей продукции менеджер должен предложить организовать доставку. В момент получения продукции (путем доставки или самовывоза) партнер производит полную оплату. После проверки количества и качества продукции партнером заявка считается выполненной.

4. Сотрудники: ФИО, дата рождения, паспортные данные, банковские реквизиты, наличие семьи, состояние здоровья.

5. Кадры: допуск к работе с конкретным оборудованием. Мастер производства занимается технологией производства, контролем качества готовой продукции, заказом материалов у поставщиков.

6. Доступ: двери открываются при использовании карты сотрудника, система турникетов передает данные о перемещении сотрудников в приложение для формирования аналитиками статистики по сотрудникам.

7. Материалы (сырье): тип, наименование, поставщик, количество в упаковке, единица измерения, описание, изображение, стоимость, количество на складе, минимальное допустимое количество, история изменений количества материалов на складе.

8. Склад: регистрация поступлений материалов, резервирование материалов для производства, отпуск и списание материалов со склада в производство, поступление и выбытие готовой продукции, получение информации о текущих остатках и движении товарно-материальных ценностей по складу.

9. Поставщики: тип, наименование, ИНН, история поставок материалов. Менеджеры контролируют список поставщиков, аналитики формируют информацию о качестве поставляемых материалов для принятия решения о прекращении работы с определенными компаниями.

10. Продукция: артикул, тип, наименование, описание, изображение, минимальная стоимость для партнера, размер упаковки (длина, ширина, высота), вес без упаковки, вес с упаковкой, сертификат качества (скан документа), номер стандарта, история изменений минимальной стоимости для партнера, время изготовления, себестоимость, номер цеха, количество человек на производстве, необходимые материалы для производства

**Техническое задание**

Функциональные требования:

1. Возможность просмотра каталога продукции поставщиками с возможностью поиска и фильтрации по типу продукции
2. Возможность добавления продажи поставщику (с расчетом скидки от количества продаж
3. Возможность добавления поставки материалов на слад
4. Возможность добавления заявки поставщику
5. Отмена заявки
6. Формирование производства на основе нужд поставщика (на основе заявок).
7. Организация записи входа сотрудников в систему
8. Организация просмотра входов в системы сотрудников для аналитика с возможностью поиска
9. Расчет статистики работы отдела аналитиков:

- Количество выполненных заявок;

- Просмотр истории цен с возможностью поиска по поставщику;

- Количество произведенной продукции.

**Руководство по стилю**

Все экранные формы пользовательского интерфейса должны иметь заголовок. Кроме того, на главной форме должен быть установлен логотип (представлен в ресурсах). Логотип не искажать: не менять изображение, пропорции изображения, цвет.

Для приложения должна быть установлена иконка (дана в ресурсах), если это реализуемо в рамках платформы.

Использовать шрифт Segoe UI.

В качестве основного фона используется белый цвет, дополнительного фона используется цвет #F4E8D3. Для акцентирования внимания пользователя на целевом действии интерфейса используется цвет #67BA80.

|  |  |  |
| --- | --- | --- |
| Основной фон | Дополнительный фон | Акцентирование внимания |
| #FFFFFF | #F4E8D3 | #67BA80 |

Спецификация 1

|  |  |  |
| --- | --- | --- |
| Функция | Входные данные | Выходные данные |
| Добавление заявки поставщику | Поставщик, список продукции, ее количество | Созданная заявка,  Данные по стоимости, сообщение об успешном  добавление данных в БД |
| Формирование производства по заявкам | Заявка, Список продукции и необходимых для нее материалов, данные о наличии материалов на складе | Наличие совпадающих  записей в БД, измененный статус заявки |
| Запись входа сотрудников в систему | Сотрудник, время входа | Наличие совпадающих  записей в БД, журнал записей о входе сотрудников |
| Расчет количества выполненных заявок | Список заявок, статус заявок | |  | | --- | |  | | Наличие совпадающих  записей в БД, общее количество выполненных заявок | | |

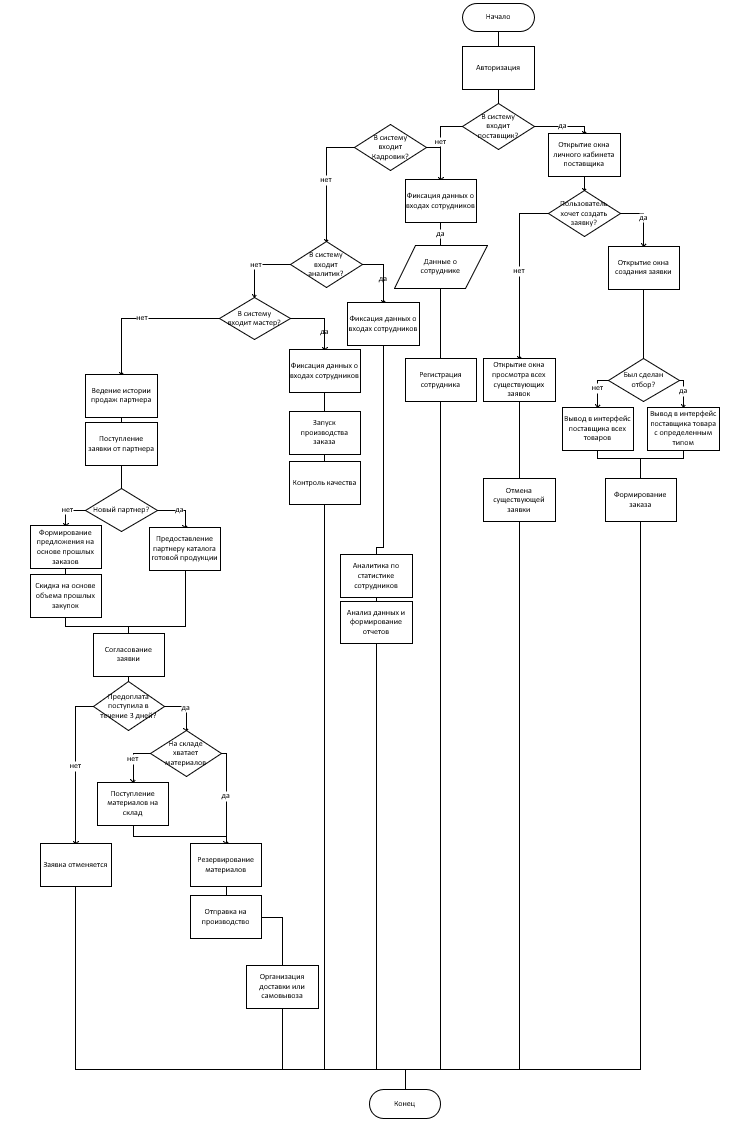


Рисунок 1. Основной алгоритм решения производства в виде блок-схемы

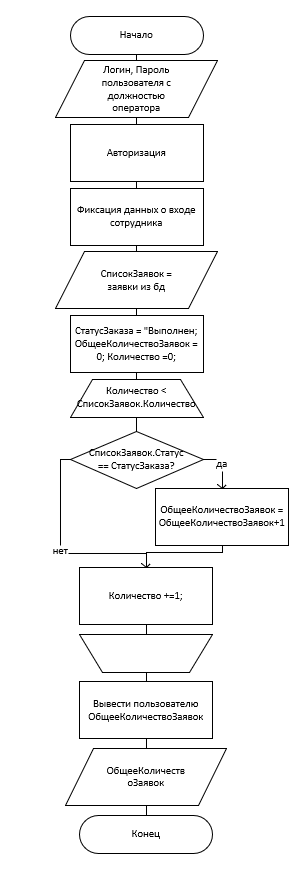


Рисунок 2.. Детализированная функция расчета количества заявок

Разработка объектов базы данных в соответствии с результатами анализа предметной области

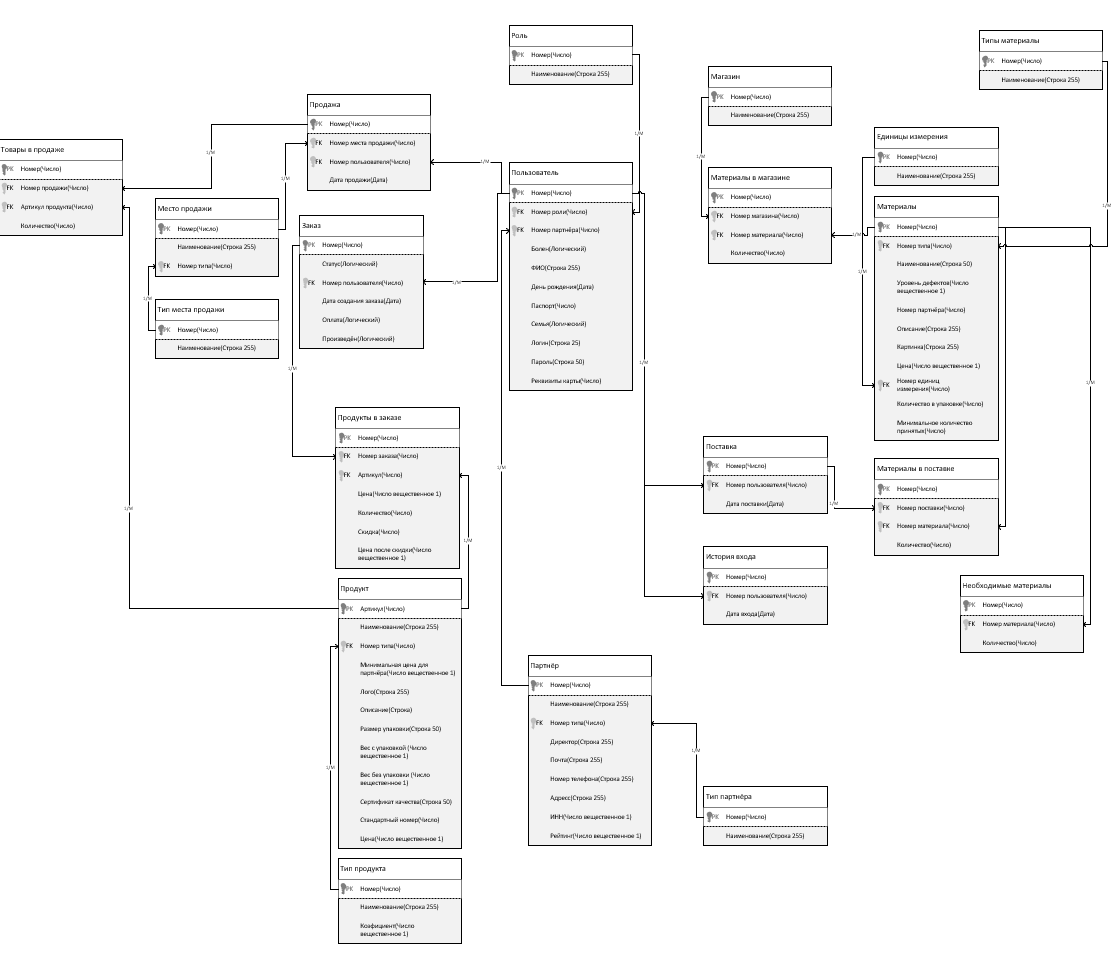
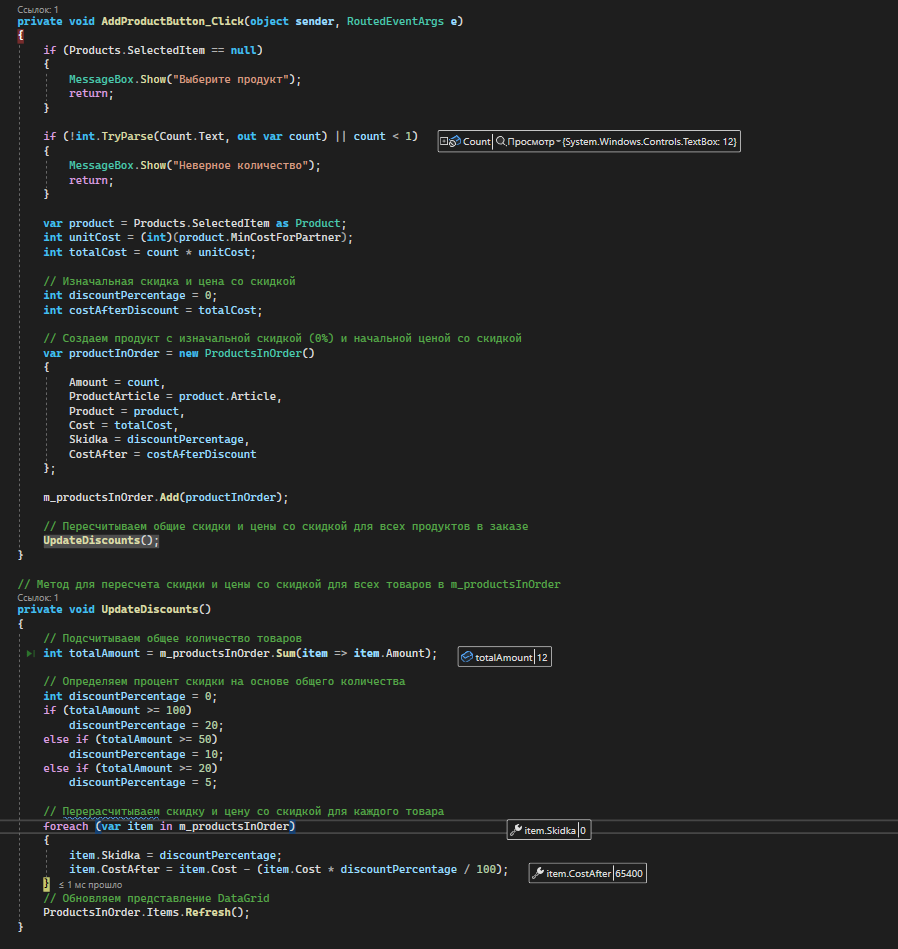


Рисунок 3. Логическая диаграмма

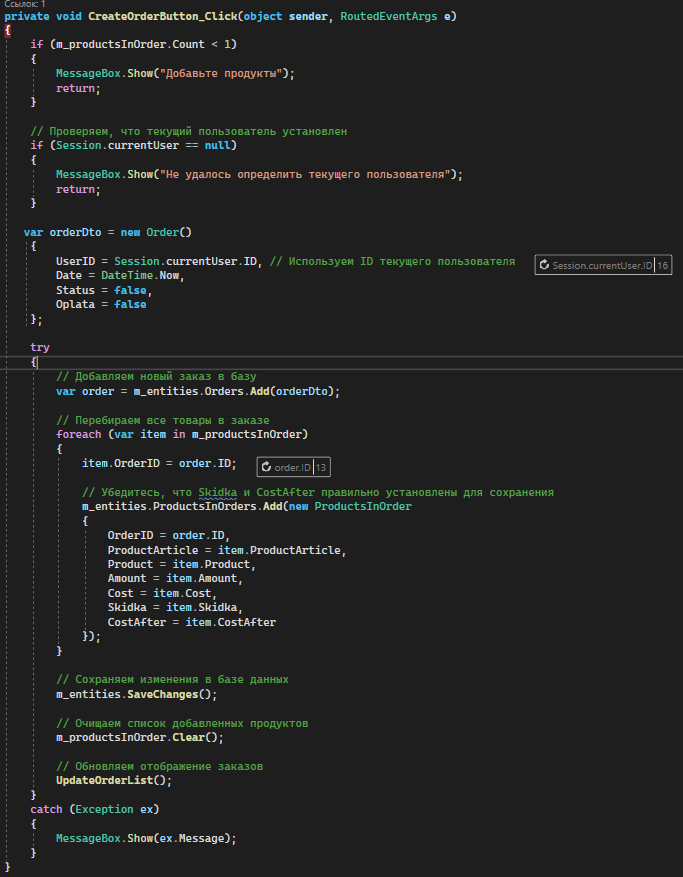


Рисунок 4. ER диаграмма



Точка остановы 1. Подсчет суммы для оплаты заказа, с скидкой

В данном коде сначала из базы достается цена товара, ищется цена по артикулу товара, после чего цена складывается и записывается в totalCost. После считается сколько всего товара в заказе и записывается в переменную totalAmount, если totalAmount больше 20 то скидка 5% если больше 50, то скидка составляет 10 % если же больше 100 то 20%, скидка записывается в переменную discountPercentege, после считается финальная цена для оплаты – переменная costAfter и выводится для пользователя финальная цена и сумма скидки.



Точка остановы 2. Запись заявки в базу данных, добавление списка товаров в базу данных

Этот код создает новый заказ, проверяя, что в заказе есть товары и что текущий пользователь авторизован. Затем создается объект заказа с данными о пользователе и текущей дате. Далее в цикле для каждого товара из списка заказов добавляется запись в таблицу с продуктами заказа. После сохранения всех данных в базу, список продуктов очищается, и обновляется отображение заказов. Если возникает ошибка, она выводится в сообщении. В целом, код организует процесс добавления заказа с товарами в базу данных и очистку данных после успешного выполнения операции.

## Общая информация о тестировании

|  |  |
| --- | --- |
| Название проекта | MasterPol |
| Номер версии | 1.0 |
| Имя тестера | Гладких Ксения |
| Даты тестирования | 08.11.2024 |

Тест-кейсы 1

## Test case #1:

|  |  |
| --- | --- |
| Test Case # | 1 |
| Приоритет теста | Высокий |
| Название тестирования/Имя | Авторизация поставщика |
| Резюме испытания | Тестирование правильности авторизации поставщика в системе с использованием правильных учетных данных. |
| Шаги тестирования | 1. Ввести правильный логин и пароль поставщика. 2. Нажать кнопку «Войти». |
| Данные тестирования | Логин: baza Пароль: baza |
| Ожидаемый результат | Открывается окно «Личный кабинет» поставщика, пользователь успешно авторизуется. |
| Фактический результат | Открывается окно «Личный кабинет» поставщика, пользователь успешно авторизуется. |
| Предпосылки | Учетная запись поставщика существует в бд |
| Постусловия | Пользователь находится в своем личном кабинете, готов к дальнейшей работе с программой. |
| Статус (Pass/Fail) | Pass |
| Комментарии |  |

## Test case #2:

|  |  |
| --- | --- |
| Test Case # | 2 |
| Приоритет тестирования | Высокий |
| Название тестирования/Имя | Авторизация с неверным паролем |
| Резюме испытания | Тестирование реакции системы при вводе неверного пароля для авторизации сотрудника |
| Шаги тестирования | 1. Ввести правильный логин и неверный пароль. 2. Нажать кнопку «Войти». |
| Данные тестирования | Логин: manager Пароль: 1231243 |
| Ожидаемый результат | Появляется сообщение об ошибке: «Неверный пароль». Пользователь не авторизуется. |
| Фактический результат | Появляется сообщение об ошибке: «Неверный пароль». Пользователь остался в том-же окне авторизации. |
| Предпосылки | Учетная запись сотрудника существует в бд |
| Постусловия | Пользователь остается на экране авторизации. |
| Статус (Pass/Fail) | Pass |
| Комментарии |  |

## Test case #3:

|  |  |
| --- | --- |
| Test Case # | 3 |
| Приоритет тестирования | Низкий |
| Название тестирования/Имя | Авторизация с длинными строками |
| Резюме испытания | Тестирование реакции системы при вводе очень длинных значений для логина и пароля. |
| Шаги тестирования | 1. Ввести в поле «Логин» строку длиной 50 символов. 2. Ввести в поле «Пароль» строку длиной 50 символов. 3. Нажать кнопку «Войти». |
| Данные тестирования | Логин: строка из 50 символов Пароль строка из 50 символов |
| Ожидаемый результат | Появляется сообщение об ошибке или система должна игнорировать слишком длинные строки и не позволять авторизацию. |
| Фактический результат | Выходит, сообщение «Пользователь не найден» |
| Предпосылки | Окно авторизации доступно для взаимодействия |
| Постусловия | Пользователь остается на экране авторизации |
| Статус (Pass/Fail) | Fail |
| Комментарии |  |

Руководство оператора

# **Основная часть**

## **Назначение программы:**

### Функциональное назначение программы

Программа предназначена для автоматизации процессов взаимодействия с поставщиками, менеджерами, мастерами и аналитиками, которые участвуют в системе производства и продажи продукции компании. Включает функции создания и управления заявками, отслеживания статуса заказов, работы с историей цен и статистикой.

### Эксплуатационное назначение программы

Программа используется для учета заявок на продукцию, учета поставок и материалов, организации производства и продажи продукции, а также для анализа данных о поставках и продажах. Программа должна быть доступна для сотрудников, работающих с заказами, производственными процессами, а также для аналитиков, собирающих статистику и историю цен.

### Функции программы:

* Создание и управление заявками от поставщиков.
* Оформление и отслеживание платежей.
* Управление складом и поставками материалов.
* Отслеживание производственных заказов.
* Анализ статистики и истории цен.

## **Условия выполнения программы:**

### Минимальный состав аппаратных средств

Процессор с архитектурой x86-64 (Intel с поддержкой Intel 64, AMD с поддержкой AMD64).

Желательно использование многопроцессорных или многоядерных машин.

### Минимальный состав программных средств

Операционная система Windows 10 или выше.

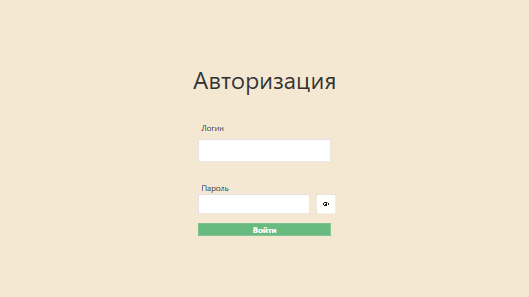
### Требования к персоналу (пользователю)

Пользователь должен иметь базовые навыки работы с компьютером, а также понимание функционала программы для эффективной работы с заявками и другими данными.

## **Выполнение программы:**

### Загрузка и запуск программы

Начало работы с программой начинается с запуска программы. Открывается окно авторизации. Ознакомьтесь с рисунком 1.



Кнопка скрытия/открытия пароля

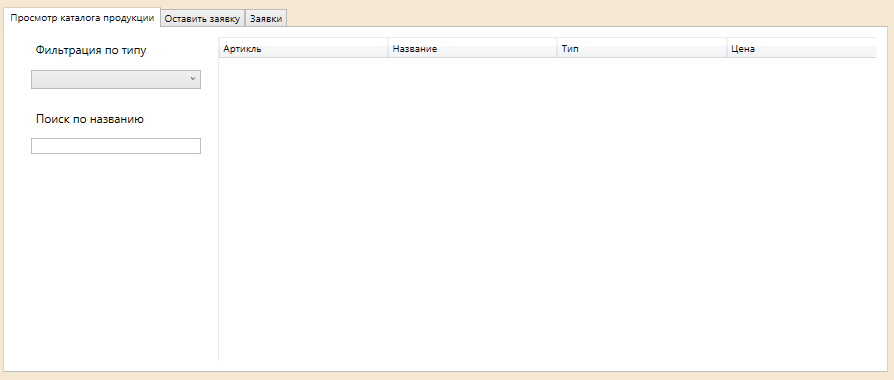
Поля ввода данных пользователя

Рисунок 1. Окно авторизации

### Выполнение программы

#### Работа поставщика

В открывшейся форме для начала работы необходимо ввести логин и пароль необходимого поставщика так же возможно скрытие и открытие пароля, при нажатии на кнопку глазика. Если ввелся верный логин и пароль тогда пользователь перейдёт на страницу «Личный кабинет». Далее необходимо нажать кнопку «Просмотр каталога продукции», после чего откроется окно создания заявки.

В окне «Просмотр каталога продукции» можно сделать отбор по типу продукции, если выбрать из выпадающего списка один из типов (выпадающий список находится в левом верхнем углу) при выборе, автоматически проведётся фильтрация заявок, так же можно провести фильтрацию по наименованию заявки, для этого нужно ввести либо полное, либо приблизительное название продукции, и при каждом изменении поля будет изменяться таблица с заявками. Ознакомьтесь с рисунком 2.  


Перемещение между страницами, нужно нажать по ним

Поле, в котором надо ввести букву или полное название

Выбор параметра, по которому будет фильтроваться таблица

Рисунок 2.Просмотр по типу

Чтобы добавить товар в заявку необходимо перейти на вкладку «Оставить заявку», выбрать товар, который мы хотим добавить в заявку, написать количество и нажатием на кнопку «Добавить» создать заявку. Если пользователю необходимо заказать более одной штуки продукции необходимо нажать кнопку еще пару раз(предварительно изменив данные если это ему нужно). Повторите действие добавление товара в заявку и с другими товарами при необходимости.

Если все необходимое уже во в правой табличке нажмите «Оставить заявку» для дальнейшей работы сотрудника. Ознакомьтесь с рисунком 3.

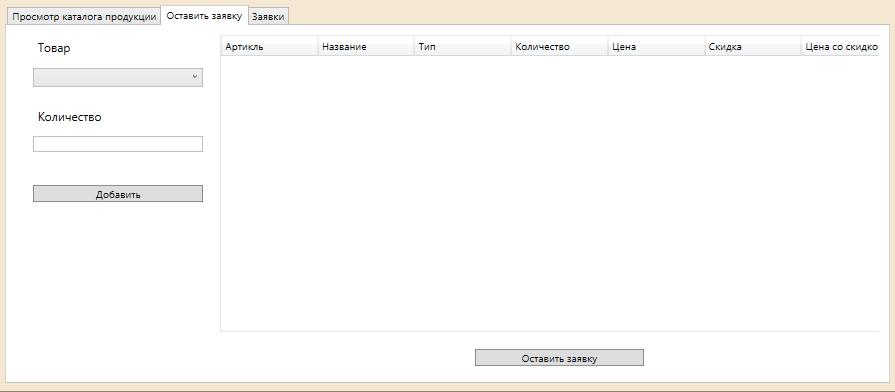


Рисунок 3. Оставить заявку

Если по какой-то причине вы захотели отменить заявку в окне «Личный кабинет» необходимо нажать на вкладку «Заявки», после этого откроется вкладка где будут все заявки пользователя, и если данная заявка не была принята, произведена и всё кроме «Новая заявка» можно при выделении её и нажатии на кнопку «Удалить заявку» удалить заявку.

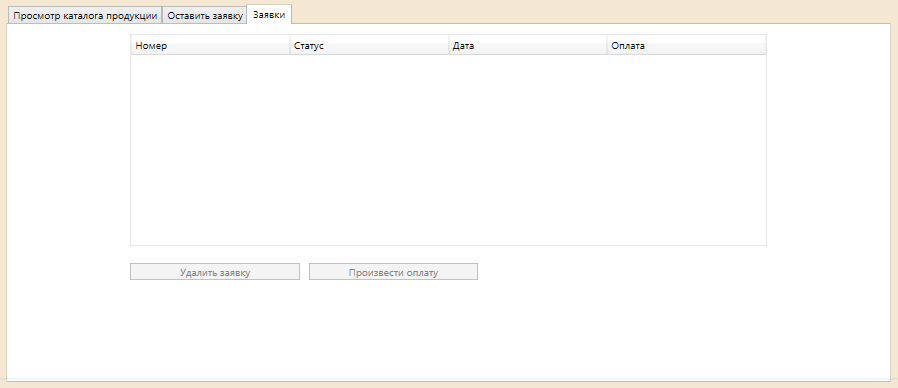
Когда продукция будет произведена необходимо будет оплатить заявку. Для этого на вкладке «Заявки» где выберите заявку которую хотите оплатить, для этого выделите её нажав по ней «Левой кнопкой мыши», нажмите на кнопку «Произвести оплату». После чего будет проведена оплата по этой заявке. Ознакомьтесь с рисунком 4  


Рисунок 4. Заявки

#### Работа менеджера

В форме авторизации для начала работы необходимо ввести логин и пароль необходимого сотрудника с должностью «Менеджер». Если ввелся верный логин и пароль тогда откроется окно «Кабинет менеджера».

В окне «Кабинет менеджера» отображаются заявки.

Для поставки материала в окне «Кабинет менеджера» нажмите на вкладку «Поставка материалов», после этого откроется окно «Поставка материала». В нем необходимо выбрать склад, на котором храниться материалы, далее нажать на редактировать. Смотрите *рисунок 5*.

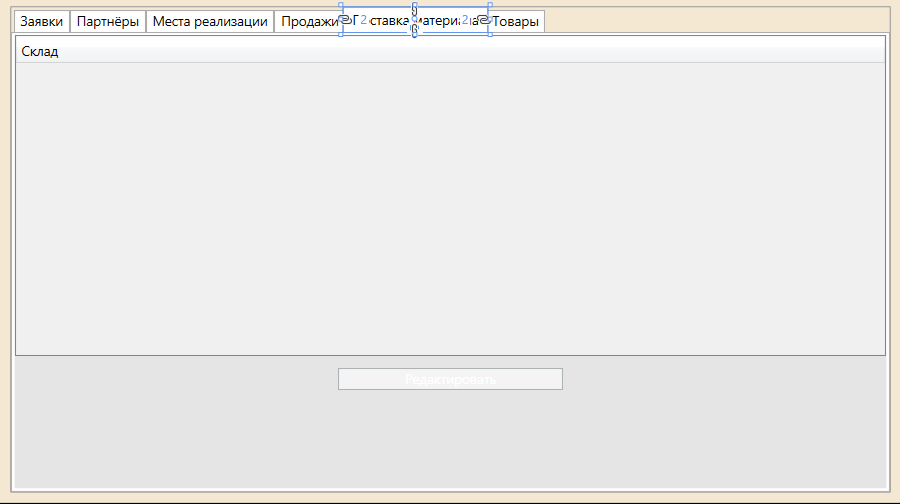


Рисунок 5.Поставка материала

Откроется окно «добавления материалов», из выпадающего списка выбрать материал, и вписать количество поступивших материалов, после этого нажать «Добавить». Смотреть *рисунок 6*

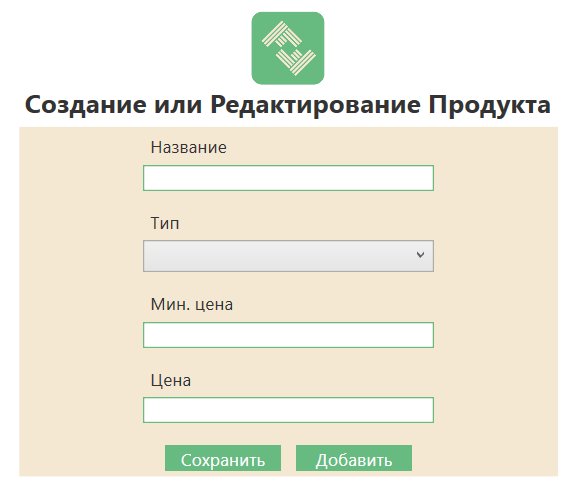
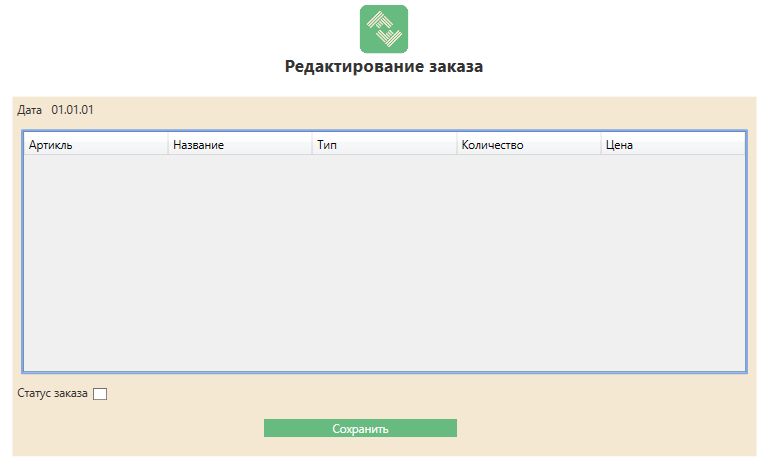


Рисунок 6. Добавления продукта

Чтоб отправить заявку изменить статус заявки нужно нажать по заявке со статусом «false», после этого нажать кнопку «Редактировать», поставить галочку на «Статус заказа». После этого статус заказа смениться на «True». Но для этого надо предварительно чтобы произвели продукт и оплатили продукт. Смотреть *рисунок 7*



Вставить галочку. Нажав на поле квадратика

Рисунок 7. Окно редактирования заказа

Работа мастера

В форме авторизации для начала работы необходимо ввести логин и пароль необходимого сотрудника с должностью «Мастер». Если ввелся верный логин и пароль тогда откроется окно «Кабинет мастера».

В окне «Кабинет мастера» отображаются заявки не произведённые заявки. Для производства продукции необходимо выбрать из таблицы заявку, после нажать на кнопку «Произвести товар». Когда это будет сделано статус производство измениться на true и пропадет из таблицы. Смотреть *рисунок 8*

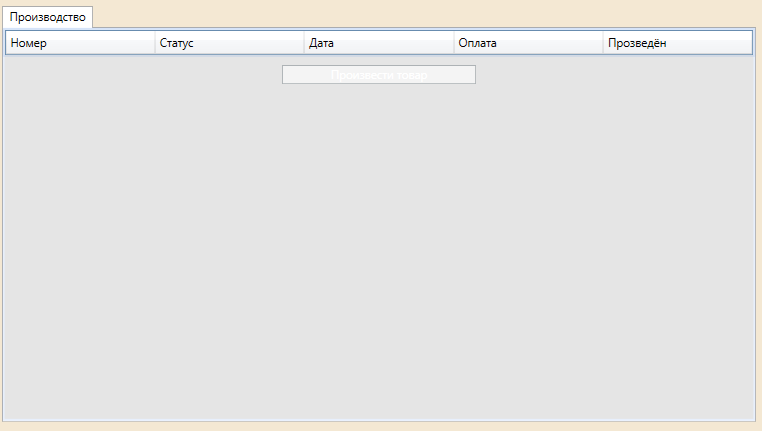


Рисунок 8. Производство

#### Работа аналитика

В форме авторизации для начала работы необходимо ввести логин и пароль необходимого сотрудника с должностью «Аналитик». Если ввелся верный логин и пароль тогда откроется окно «Аналитика работы отдела».

Для просмотра истории входов сотрудников необходимо нажать кнопку «поиск», если нужно сделать отбор по сотруднику, то перед нажатием на кнопку необходимо в поле ввода ввести ФИО искомого.

Для того чтоб узнать сколько продуктов вверху окна нажмите на вкладку «Аналитика произведённой продукции», в ней будет надпись с количеством сделанных товаров и таблица с этими заявками. Смотреть рисунок 9

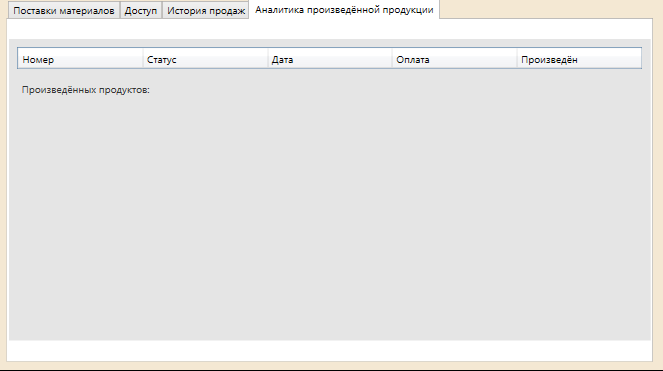
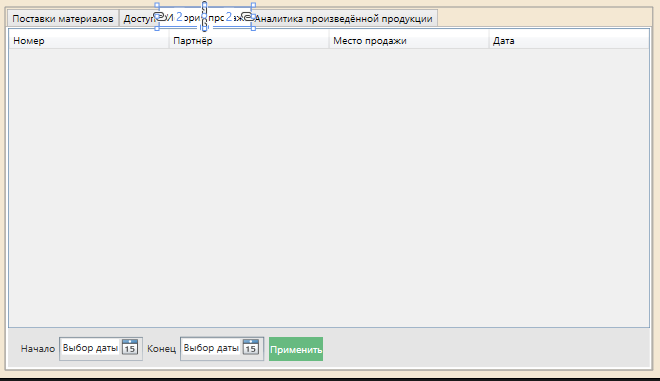


Рисунок 9. Аналитика произведённой продукции

Для просмотра истории цен перейдите на вкладку «История цен», если нужно сделать отбор по дате, то необходимо в поле ввода даты ввести дату и в «начало», и в «конец», далее нажать, после чего произведётся фильтрация и будут отображены операции в период с начала до конца. Смотреть рисунок 10



Выбор даты для сортировки.

Рисунок 10. история продаж

### 

### Завершение работы программы

Завершение работы программы осуществляется путем закрытия окна программы.

В разделе «Выполнение программы» должна быть указана последовательность действий оператора, обеспечивающих загрузку, запуск, выполнение и завершение программы, приведено описание функций, формата и возможных вариантов команд, с помощью которых оператор осуществляет загрузки и управляет выполнением программы, а также ответы программы на эти команды.

## **Сообщения оператору:**

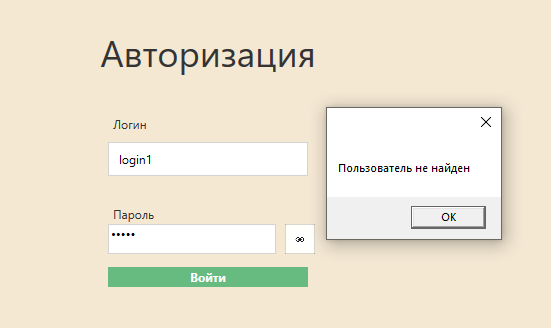
Не правильно введённый логин. Смотреть *рисунок 11*  


Рисунок 11. Пользователь не найден

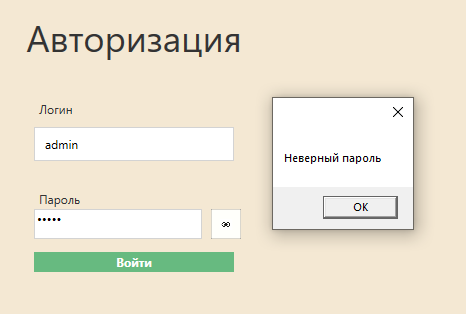
Не правильно введённый пароль. Смотреть *рисунок 12*  


Рисунок 12. Неверный пароль

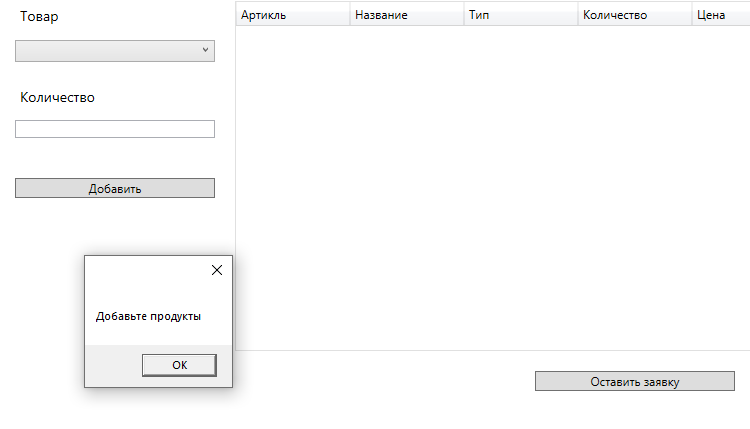
Когда не заполнили таблицу добавления товаров. Смотреть *рисунок 13*  


Рисунок 13. Добавить продукты

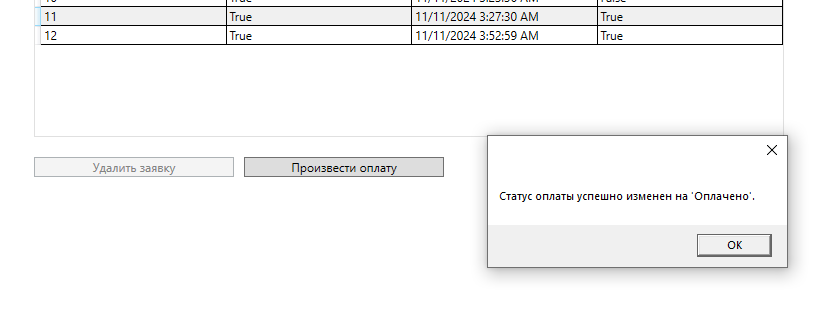
Когда выбрали правильно заявку и оплатили её. Смотреть *рисунок 14*  


Рисунок 14. статус оплаты успешно изменён на "Оплачено"

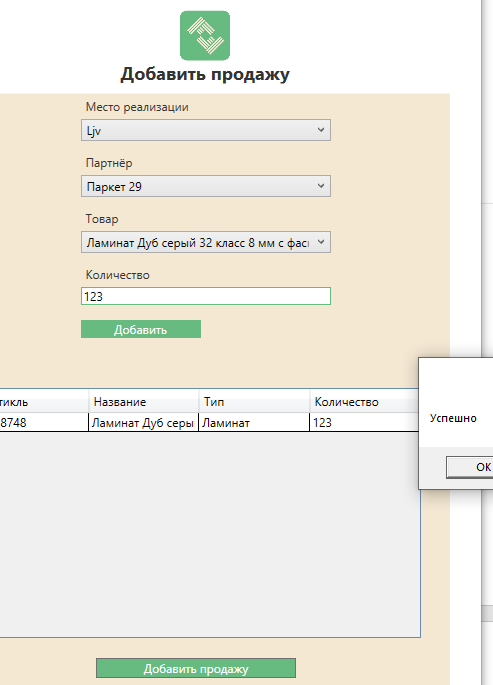
При добавлении продажи  


Рисунок 15. Успешно

# **ЗАДАНИЕ 2**

Проанализировать предметную область, составить краткую спецификацию разрабатываемого модуля выделить входные и выходные данные; Сформированную спецификацию можно рассмотреть в спецификации 2.

Сформировать основной алгоритм решения производства в виде блок-схемы в соответствии с описанием предметной области. С сформированной блок-схемой можно ознакомиться на рисунке 27.

Детализировать в виде алгоритма одну из функций (расчета количества заявок, среднего времени ремонта). Рассмотреть детализированную функцию можно на рисунке 28.

Разработка объектов базы данных в соответствии с результатами анализа предметной области (ER изображена на рисунке 729 , логическая диаграмма на рисунке 830).

Подготовка данных к импорту из файлов .xls или .xlsx, приведение к 3 нормальной форме согласно описанию предметной области

Формирование алгоритмов разработки программных модулей в соответствии с описанием предметной области

Разработка программных модулей в соответствии с техническим заданием

Выполнение отладки программных модулей с использованием специализированных программных средств. С выполнением отладки можно ознакомиться просмотрев и прочитав картинки и описания точки остановы 3 и точки остановы 4.

Реализация базы данных в конкретной системе управления базами данных

Разработать систему согласно предметной области.

Протестировать систему в 3 условиях (нормальных, экстремальных и исключительных). Протестированная система зафиксирована в тест-кейсах.

Разработать системного программиста. Ознакомиться с ним можно прочитав руководство оператора 2.

**Описание предметной области**

Предметная область "Разработка программного модуля для учета заявок на ремонт оргтехники" относится к области обслуживания и ремонта технического оборудования в офисных условиях.

Оргтехника, такая как принтеры, копировальные аппараты, сканеры и другое офисное оборудование, требует регулярного обслуживания и ремонта, чтобы оставаться в рабочем состоянии. В офисах часто возникают проблемы с оргтехникой, которые нужно оперативно решать.

Задача разработки такого программного модуля состоит в создании удобного и эффективного инструмента для учета, отслеживания и управления заявками на ремонт оргтехники, что помогает минимизировать время простоя оборудования и обеспечивает бесперебойную работу в офисных условиях.

Программный модуль для учета заявок на ремонт оргтехники предназначен для автоматизации процесса приема и обработки заявок на ремонт от сотрудников офиса или других пользователей. Он позволяет упростить и ускорить процесс решения проблем с оргтехникой.

Основные функции и возможности модуля включают:

1. Заявка на ремонт: это информация, предоставленная клиентом о неисправности оргтехники, которая требует ремонта. Клиент оставляет новую заявку. Заявка может содержать данные о виде оргтехники, модели, описании проблемы, личную информацию (ФИО клиента и номер телефона). Клиент может отредактировать свою заявку.

2. Регистрация заявки: этот процесс включает приём и регистрацию заявки оператором сервисного центра в системе учёта. Важными аспектами регистрации являются присвоение уникального идентификатора заявке, сохранение информации о заявке.

3. Обработка заявки: процесс, включающий анализ заявки, определение её приоритетности и назначение исполнителя (мастера) для задачи. В процессе обработки может потребоваться дополнительная информация или уточнение деталей проблемы у клиента.

4. Исполнение заявки: фактическое выполнение ремонта оборудования. На этом этапе назначенный мастер ремонтирует оргтехнику, вносит необходимые изменения или заменяет неисправные запчасти. Важно отметить, что на этом этапе могут возникать необходимость заказа запчастей или координации работ с другими специалистами.

5. Отчётность и информирование: важной составляющей учёта заявок на ремонт является фиксация и отчёт о выполненной работе. После завершения ремонта мастер должен предоставить отчёт о проделанной работе, включая информацию о затраченных ресурсах (время, запчастях) и оказанной помощи.

6. Мониторинг и анализ: этот этап предполагает контроль и анализ процесса учёта заявок на ремонт. Важно отслеживать и анализировать время обработки заявок, качество выполненных работ, расходы и прочие параметры, которые могут помочь в оптимизации и улучшении процесса

**Техническое задание**

1. Общие сведения

1.1. Наименование проекта: Разработка программного модуля для учета заявок на ремонт оргтехники.

1.2. Заказчик: ООО "Полюс".

1.3. Исполнитель: Сервисный центр "IT-Сan".

2. Функциональные требования

2.1. Возможность добавления заявок в базу данных с указанием следующих параметров:

- Номер заявки;

- Дата добавления;

- Вид оргтехники;

- Модель;

- Описание проблемы;

- ФИО клиента;

- Номер телефона;

- Статус заявки (новая заявка, в процессе ремонта, завершена).

2.2. Возможность редактирования заявок:

- Изменение этапа выполнения (готова к выдаче, в процессе ремонта, ожидание запчастей);

- Изменение описания проблемы;

- Изменение ответственного за выполнение работ.

2.3. Возможность отслеживания статуса заявки:

- Отображение списка заявок;

- Получение уведомлений о смене статуса заявки;

- Поиск заявки по номеру или по параметрам.

2.4. Возможность назначения ответственных за выполнение работ:

- Добавление мастера к заявке;

- Отслеживание состояния работы и получение уведомлений о ее завершении;

- Мастер может добавлять комментарии на форме заявки и фиксировать информацию о заказанных запчастях и материалах.

2.5. Расчет статистики работы отдела обслуживания:

- Количество выполненных заявок;

- Среднее время выполнения заявки;

- Статистика по типам неисправностей.

3. Нефункциональные требования

3.1. Кроссплатформенность:

- Поддержка работы на ОС семейства Windows.

3.2. Безопасность:

- Логин и пароль для доступа к приложению;

- Доступ к данным должен быть ограничен в зависимости от роли пользователя.

3.3. Удобство использования:

- Простой и интуитивный интерфейс;

- Информативные уведомления и подсказки.

4. Требования к реализации

4.1. Язык программирования: на усмотрение разработчика

4.2. СУБД: на усмотрение разработчика

5. Требования к документации

5.1. Техническое задание на разработку программного модуля.

5.2. Руководство системному программисту.

Спецификация 2

|  |  |  |
| --- | --- | --- |
| Функция | Входные данные | Выходные данные |
| Редактирование заявки | ID заявки, новые значения полей (описание проблемы, модель) | Наличие совпадающих  записей в БД, измененная заявка, подтверждение успешного редактирования заявки |
| Поиск и фильтрация заявок | Критерии фильтрации (например, по проблеме, номеру) | Наличие совпадающих  записей в БД, обновленная таблица с заявками |
| Авторизация сотрудника | Логин, пароль | Наличие совпадающих  записей в БД, доступ к определенным окнам приложения, в зависимости от должности |
| Добавление заявки | Тип оргтехники, модель, проблема, ID клиента, номер телефона | |  | | --- | |  | | Сообщение об успешном  добавление данных в БД | |   , созданная заявка |



Рисунок 27.Основной алгоритм решения производства в виде блок-схемы

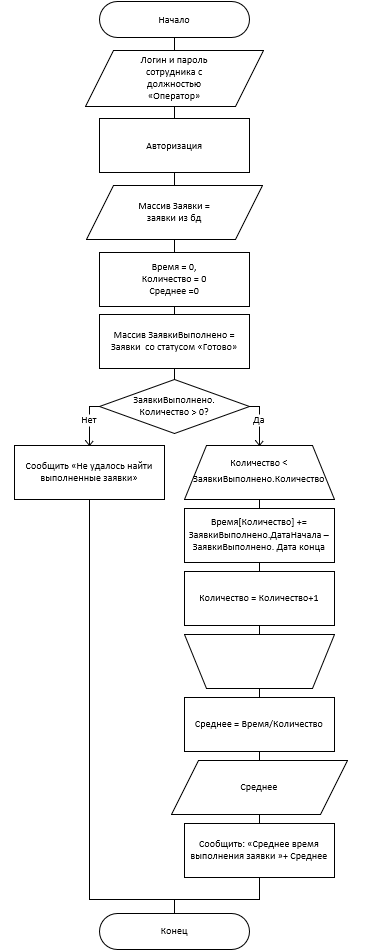


Рисунок 28.Детализированная функция расчета среднего времени затрачиваемого

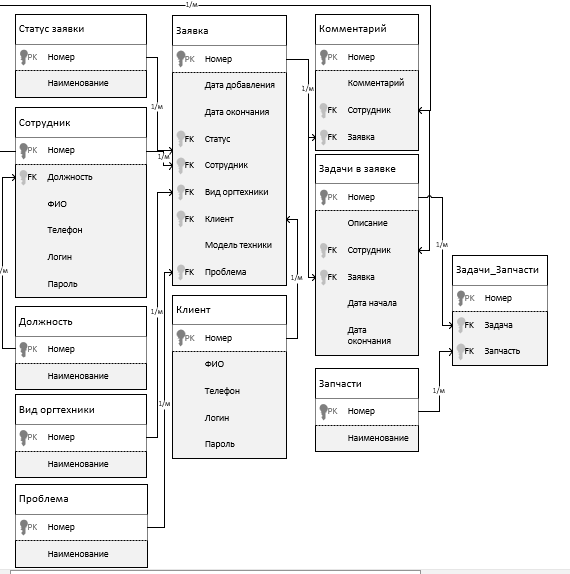
****

Рисунок 2929. ER диаграмма

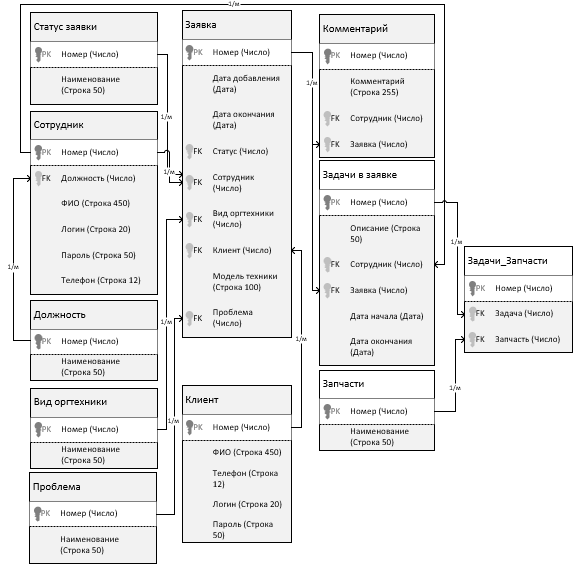
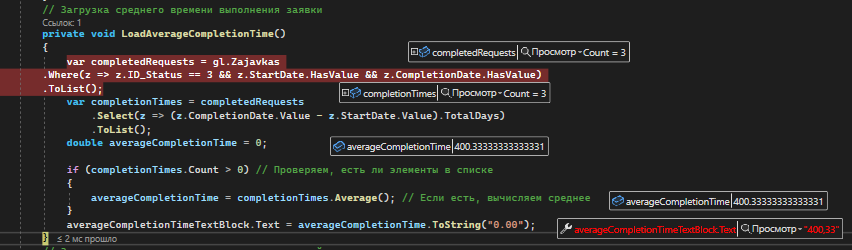
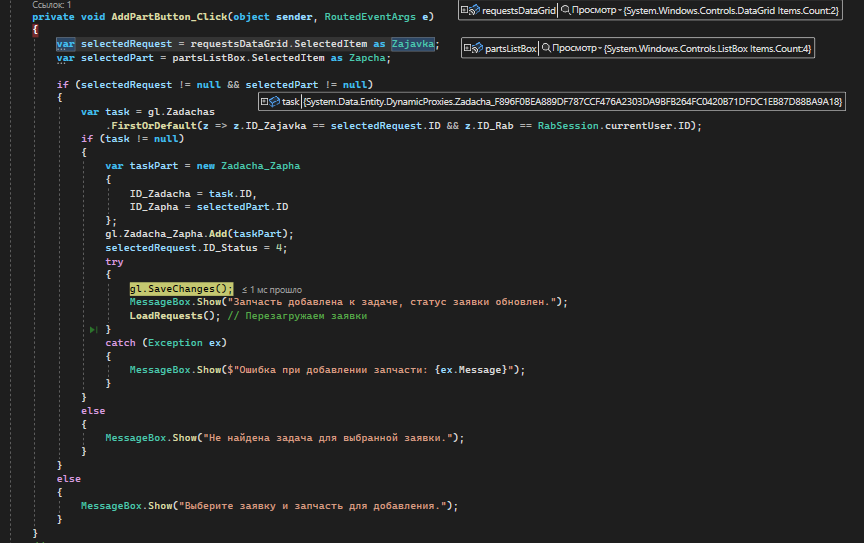


Рисунок 3030. Логическая диаграмма



Точка остановы 3.Счет среднего времени выполнения заявки

В completedRequests записываются данные из базы данных о заявках со статусом «Готова», для каждого элемента из списка completedRequests вычисляется разница между датой завершения и датой начала, Value извлекает значение даты. TotalDays — это свойство, которое возвращает общее количество дней, прошедших между датами. Если в completionTimes более одного элемента - вычисляем среднее количество времени. Average() — это метод, который вычисляет среднее значение для коллекции чисел. После в отдельное поле averageCompletionTimeTextBlock мы выводим среднее время в виде 0,00.



Точка остановы 4.Добавление запчастей к задаче

RequestsDataGrid — это таблица (DataGrid), в которой отображаются заявки. Мы пытаемся получить выбранную заявку как объект типа Zajavka. PartsListBox — это список (ListBox), в котором отображаются запчасти. Мы пытаемся получить выбранную запчасть как объект типа Zapcha. Если selectedRequest или selectedPart не выбран, то дальше код не выполняется, и выводится сообщение «Выберите заявку и запчасть для добавления», далее ищется задача идентичная выбранной и текущему пользователю. Если задача найдена, то выполняются следующие шаги. Если задача не найдена, выводится сообщение «Не найдена задача для выбранной заявки». В taskPart записываются данные о выполняемой задаче и выбранной детали. Далее данные taskPart записываются в таблицу Zadacha\_Zapha в базе данных. Изменения сохраняются, выходит сообщение «Запчасть добавлена к задаче, статус заявки обновлен». Если вышла непредвиденная ошибка при записи данных в БД пользователю выйдет сообщение «Ошибка при добавлении запчасти:» с дальнейшим описанием ошибки.

## Общая информация о тестировании

|  |  |
| --- | --- |
| **Название проекта** | Разработка программного модуля для учета заявок на ремонт оргтехники |
| **Номер версии** | 1.0 |
| **Имя тестера** | Гладких Ксения |
| **Даты тестирования** | 10.11.2024 |

Тест-кейс 1

## Test case #1:

|  |  |
| --- | --- |
| **Test Case #** | 1 |
| **Приоритет теста** | Высокий |
| **Название тестирования/Имя** | Создание новой заявки |
| **Резюме испытания** | Тестирование успешного добавления новой заявки на ремонт оргтехники |
| **Шаги тестирования** | 1. Нажать кнопку «Я клиент» 2. Авторизоваться за клиента введя логин и пароль 3. Нажать кнопку «Добавить заявку» 4. Ввести информацию о технике: вид, модель, телефон, описание проблемы 5. Нажать кнопку «Сохранить заявку» |
| **Данные тестирования** | логин login1, пароль pass1, вид Компьютер, модель DEXP Aquilon O286, телефон 89219567841, описание Перестал работать |
| **Ожидаемый результат** | Заявка успешно добавиться со статусом «Новая» |
| **Фактический результат** | Заявка добавилась, её статус «Новая» |
| **Предпосылки** | Программа запущена, клиент существует в базе данных |
| **Постусловия** | Заявка сохранена и ее видно в личном кабинете клиента, вышло сообщение об успешном создании заявки |
| **Статус (Pass/Fail)** | Pass |
| **Комментарии** |  |

## Test case #2:

|  |  |
| --- | --- |
| **Test Case #** | 2 |
| **Приоритет тестирования (Малый/Средний/высокий)** | Средний |
| **Название тестирования/Имя** | Назначение мастера к заявке |
| **Резюме испытания** | Проверка назначения мастера к заявке |
| **Шаги тестирования** | 1. Нажать кнопку «Я сотрудник» 2. Авторизоваться за менеджера введя логин и пароль 3. Кликнуть по заявке, выделив ее 4. Нажать кнопку «Отправить в ремонт» 5. В новом окне выбрать из выпадающего списка мастера |
| **Данные тестирования** | Логин login1, пароль pass1, мастер: Ильин Александр Андреевич |
| **Ожидаемый результат** | Мастер успешно назначен, заявка переведена в статус «В процессе ремонта |
| **Фактический результат** | Мастер успешно назначен, заявка переведена в статус «В процессе ремонта |
| **Предпосылки** | Заявка уже создана, приложение запущено, мастер существует в базе данных |
| **Постусловия** | Мастер назначен, заявка обновлена |
| **Статус (Pass/Fail)** | Pass |
| **Комментарии** |  |

## Test case #3:

|  |  |
| --- | --- |
| **Test Case #** | 3 |
| **Приоритет тестирования (Малый/Средний/высокий)** | Высокий |
| **Название тестирования/Имя** | Создание заявки с максимально длинными значениями |
| **Резюме испытания** | Тест реакции системы при вводе максимально возможного количества символов в полях. |
| **Шаги тестирования** | 1. Нажать кнопку «Я клиент» 2. Авторизоваться за клиента введя логин и пароль 3. Нажать кнопку «Добавить заявку» 4. Ввести информацию о технике: вид, модель, телефон, описание проблемы   Нажать кнопку «Сохранить заявку» |
| **Данные тестирования** | логин login1, пароль pass1, вид Компьютер, модель DEXP Aquilon O286, телефон 89219567841111111111111111, описание Перестал работать |
| **Ожидаемый результат** | Выйдет сообщение «В поле телефон должно быть вписано 11 цифр» |
| **Фактический результат** | Заявка сохраняется, но система ограничивает длину текста в поле |
| **Предпосылки** | Программа запущена, клиент существует в базе данных |
| **Постусловия** | Заявка добавлена, интерфейс не зависает. |
| **Статус (Pass/Fail)** | Fail |
| **Комментарии** |  |

Руководство программиста

# **Основная часть**

## **Общие сведения о программе:**

### Назначение программы:

Программа предназначена для оптимизации и автоматизации обработки заявок на ремонт офисного оборудования. Она позволяет эффективно управлять заявками, отслеживать их и составлять отчёты, улучшая процесс ремонта и сокращая время простоя оборудования.

### Функции, выполняемые программой:

Программа позволяет:

* Добавлять, просматривать и обновлять запросы на ремонт.
* Распределять запросов между техническими специалистами.
* Мониторинг состояния ремонта.
* Добавление деталей, используемых во время ремонта.
* Создание отчетов и статистических данных о ремонте и работе технического персонала.

### Сведения о технических и программных средствах.

#### Объем оперативной памяти.

Не менее 4 ГБ оперативной памяти.

#### Требования к составу периферийных устройств:

* Монитор с минимальным разрешением 1280x720 пикселей.
* Клавиатура и мышь для взаимодействия.

#### Требования к программному обеспечению:

* Windows 10 или выше.
* Visual Studio (для редактирования кода и отладки).
* SQL Manager с совместимой конфигурацией базы данных.

## **Характеристика программы.**

### Структура программы.

Программа состоит из: начального окна авторизации, двух папок Pages и Windows. В папке Pages располагаются окна: Авторизация (AuthPage), Личный кабинет клиента (ClientPage), Личный кабинет менеджера (ManagerPage), Личный кабинет мастера (MasterPage), Личный кабинет оператора (OperatorPage). В папке Window окна: Добавление заявки (AddOrderWindow) и Изменение заявки (EditOrderWindow). Добавление мастера к заказу (AssingOrderwindow).

В программе существует класс для хранения информации Session о авторизующийся личности: Сохраняются все данные учётной записи на время сессии. Также в программу подключена база данных из SQL Manager.

Программное реализованы переходы между страницами посредством нажимания соответствующих кнопок. Приложение постоянно работает с базой данных:

* проверка логина и пароля вводимых пользователем на соответствие с записями в базе данных, для доступа к определенной функциональности приложения,
* добавление новых записей в базу данных о заявках, задачах,
* постоянное изменение статуса заявки в базе данных
* и многое другое.

## **Настройка программы.**

Настройка технических средств.

* Убедитесь, что на компьютере установлена операционная система Windows 10 или выше.
* Убедитесь, что на компьютере установлено Visual Studio и SQL Manager с необходимой для работы базой данных.
* Убедитесь, что на компьютере имеется монитор с разрешением не менее 1280x720 пикселей.
* Убедитесь, что на компьютере имеется клавиатура и мышь для взаимодействия с программой.

Настройка функций программы:

* Убедитесь, что в базе данных имеются минимум 11 таблиц, у каждой таблицы должна быть как минимум 1 связь с другой таблицей.
* Убедитесь, что в приложение путь подключения к базе данных является корректным.

Настройка передачи данных:

* Убедитесь, в базе данных есть записи о сотрудниках, клиентах, должностях, типах оргтехники, проблемах и другие.

После выполнения этих шагов программа будет готова к использованию.

## **Проверка программы.**

Способом проверки станет ручное тестирование.

### Загрузка и запуск программы.

Для начала работы необходимо запустить приложение

### Выполнение программы.

В начале работы с приложением откроется окно «авторизация». Пользователь должен ввести логин и пароль существующего в базе данных (далее БД) с ролью клиента, для примера пусть он введет login9 pass9 соответственно, после чего пользователь нажмет кнопку «Войти». Далее откроется окно личного кабинета клиента, в ней отображаются существующие на данный момент в БД заявки от этого клиента. Далее пользователь нажмет на кнопку «Добавить заявку», откроется новое окно добавления заявки, в которой необходимо заполнить модель и вид оргтехники, описание проблемы, и оставить свой номер телефона. После того как все поля были заполнены пользователь возвращается в личный кабинет клиента, где он может сделать отбор по номеру заявки или по проблеме, которую он указал ранее. Если заказ еще не в статусе «Готов», можно изменить его содержимое, нажав на необходимую заявку, выделив ее, а после на кнопку «Редактировать». Откроется окно изменения заявки, в которой можно изменить модель или проблему указанной оргтехники. Закончив проверять функциональность клиента, пользователь нажмет кнопку со стрелочкой в левом верхнем углу, чтоб перейти к авторизации или на крестик чтоб закрыть приложение.

Далее пользователь должен ввести логин и пароль существующего в базе данных БД сотрудника, для примера пусть он введет login1 pass1 соответственно, в БД этот сотрудник имеет должность менеджера. После нажатия кнопки «Войти» откроется окно личного кабинета менеджера. В окне менеджера отображаются все заявки, в том числе недавно созданная. Менеджер может записать себя как работника, оформившего заявку, нажав сначала на заявку, а после на соответствующую кнопку. Также менеджер может отправить выделенную заявку со статусом «Новая» в ремонт, нажав на кнопку «Отправить в ремонт». После этого откроется окно назначения мастера, в котором необходимо из выпадающего списка выбрать мастера и нажать кнопку подтвердить. Статус выполненной заявки измениться на «В процессе ремонта» и выйдет соответствующее сообщение. После проверки работоспособности функционала менеджера необходимо нажать кнопку со стрелочкой в верхнем левом углу. Чтобы вернуться к авторизации.

Далее пользователь должен ввести логин и пароль существующего в базе данных БД сотрудника, для примера пусть он введет login2 pass2 соответственно, в БД этот сотрудник имеет должность мастера. После нажатия кнопки «Войти» откроется окно личного кабинета мастера. В окне мастера отображаются все заявки со статусом «В процессе ремонта» или «Ожидание запчастей», в том числе и недавно отправленная. Мастер при ремонте может найти неисправную деталь, чтоб ее заказать необходимо из листа с запчастями выбрать нужную и нажать «Добавить», тогда статус заявки измениться на «Ожидание запчастей». Также мастер может оставить комментарии в специальном поле. Когда ремонт закончен мастер нажмет на заявку, а после на большую кнопку «Отремонтировано», тогда статус заявки измениться на «Готово». После проверки необходимо нажать кнопку со стрелочкой в верхнем левом углу. Чтобы вернуться к авторизации.

Пользователь должен ввести логин и пароль существующего в базе данных БД сотрудника, для примера пусть он введет login4 pass4 соответственно, в БД этот сотрудник имеет должность оператора. После нажатия кнопки «Войти» откроется окно личного кабинета оператора. В окне оператора отображаются: количество выполненных заявок, среднее время выполнения заявки и статистика по типам неисправностей (таблица, в которой перечислены все проблемы (неисправности) и количество заявок, где выбрана именно эта неисправность). После проверки работоспособности функционала мастера необходимо нажать кнопку со стрелочкой в верхнем левом углу. Чтобы вернуться к авторизации.

## **Сообщения.**

Все сообщения выходят при ручном тестировании.

* 1. В любом из окон авторизации: если не заполнен логин выйдет сообщение «Введите логин». Если пароль введен неправильно выйдет сообщение «Неверный пароль». Если таблица пользователей пуста выйдет сообщение «Пользователь не найден».
  2. В окне личного кабинета клиента: если клиент не был записан в класс Session или произошла какая-то ошибка, выйдет сообщение «Клиент не найден». Если была нажата кнопка «Редактировать», но не выбрана заявка, выйдет сообщение «Выберите заявку для редактирования». Если была нажата кнопка «Редактировать», и выбрана заявка со статусом «Готова», выйдет сообщение «Эту заявку нельзя редактировать, так как её статус не позволяет изменений».
  3. В окне личного кабинета мастера: если была выбрана заявка и запчасть для нее, а после нажата кнопка «Добавить» выйдет сообщение «Запчасть добавлена к заявке, статус обновлен». Если что-либо из них не выбрано выйдет сообщение «Выберите заявку и запчасть для добавления». Если выйдет непредвиденная ошибка выйдет сообщение «Ошибка при добавлении запчасти:» и тест этой ошибки.
  4. В окне личного кабинета мастера: если была выбрана заявка, а после нажата кнопка «Отремонтировано» выйдет сообщение «Заявка отмечена как отремонтированная». Если заявка не выбрана выйдет сообщение «Выберите заявку для завершения ремонта». Если выйдет непредвиденная ошибка выйдет сообщение «Ошибка при завершении ремонта:» и тест этой ошибки.

Ссылка на github: <https://github.com/kerlondes/zadacha2.git>  
Код Аналитика  
private string \_searchText = ""; // Храним текст для поиска

private readonly BDEntities m\_entities;

private ICollectionView \_authHistoryView; // Представление данных для фильтрации AuthHistory

public AnalystPage()

{

InitializeComponent();

m\_entities = BDEntities.GetInstance();

}

private void Page\_Loaded(object sender, RoutedEventArgs e)

{

Supplies.ItemsSource = m\_entities.MaterialsInStores.ToList();

// Загружаем все заказы и устанавливаем их в DataGrid

Orders.ItemsSource = m\_entities.Orders.ToList();

var producedOrders = m\_entities.Orders

.Where(order => order.Proizweden == true)

.ToList();

// Считаем общее количество произведённых продуктов по этим заказам

int totalProducedAmount = producedOrders

.SelectMany(order => order.ProductsInOrders) // Получаем все продукты, связанные с каждым заказом

.Sum(product => product.Amount); // Суммируем количество продуктов в каждом заказе

// Отображаем общее количество произведённых продуктов в Label

ProducedLabel.Content = $"Произведённых продуктов: {totalProducedAmount}";

// Инициализация данных для AuthHistory с именем пользователя вместо ID

var authHistoryData = m\_entities.AuthHistories

.Select(auth => new

{

UserName = m\_entities.Users

.Where(user => user.ID == auth.UserID)

.Select(user => user.FIO)

.FirstOrDefault(),

Date = auth.Date

})

.ToList();

// Установка источника данных и представления для AuthHistory

AuthHistory.ItemsSource = authHistoryData;

\_authHistoryView = CollectionViewSource.GetDefaultView(authHistoryData);

Sales.ItemsSource = m\_entities.Sales.ToList();

}

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

{

var start = StartDate.SelectedDate;

var end = EndDate.SelectedDate;

var result = m\_entities.Sales

.Where(sale => start != null ? sale.Date >= start : end != null ? sale.Date <= end : true)

.ToList();

Sales.ItemsSource = result;

}

private void NameFind\_TextChanged(object sender, TextChangedEventArgs e)

{

// Получаем введенный текст для поиска

\_searchText = NameFind.Text?.ToLower();

// Обновляем фильтр для представления данных в AuthHistory

\_authHistoryView.Filter = item =>

{

var authItem = item as dynamic; // Поскольку это анонимный тип, используем dynamic

return string.IsNullOrEmpty(\_searchText) ||

authItem.UserName?.ToLower().Contains(\_searchText) == true;

};

}  
Код авторизации  
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using Учебаня\_практика.Properties;

using static System.Collections.Specialized.BitVector32;

namespace Учебаня\_практика.Pages

{

/// <summary>

/// Логика взаимодействия для AuthPage.xaml

/// </summary>

public partial class AuthPage : Page

{

public AuthPage()

{

InitializeComponent();

m\_mainWindow = Application.Current.MainWindow as MainWindow;

}

private BDEntities m\_entities = BDEntities.GetInstance();

private readonly MainWindow m\_mainWindow;

private void Page\_Loaded(object sender, RoutedEventArgs e)

{

//MessageBox.Show(m\_mainWindow.mainFrame.CanGoBack.ToString());

}

private void TogglePasswordVisibility(object sender, RoutedEventArgs e)

{

if (Password.Visibility == Visibility.Visible)

{

// Если видим PasswordBox, показываем TextBox с текстом

PasswordTextBox.Text = Password.Password;

Password.Visibility = Visibility.Collapsed;

PasswordTextBox.Visibility = Visibility.Visible;

EyeIcon.Source = new System.Windows.Media.Imaging.BitmapImage(new System.Uri("/icon/glas.jpg", System.UriKind.Relative)); // Открытый глаз

}

else

{

// Если видим TextBox, переключаем обратно на PasswordBox с символами

Password.Password = PasswordTextBox.Text;

PasswordTextBox.Visibility = Visibility.Collapsed;

Password.Visibility = Visibility.Visible;

EyeIcon.Source = new System.Windows.Media.Imaging.BitmapImage(new System.Uri("/icon/neglas.jpg", System.UriKind.Relative)); // Закрытый глаз

}

}

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

{

if (string.IsNullOrEmpty(Login.Text))

{

MessageBox.Show("Введите логин");

return;

}

var user = m\_entities.Users

.Where(a => a.Login == Login.Text)

.FirstOrDefault();

if (user == null)

{

MessageBox.Show("Пользователь не найден");

return;

}

if (user.Password != Password.Password)

{

MessageBox.Show("Неверный пароль");

return;

}

Session.currentUser = user;

m\_entities.AuthHistories.Add(new AuthHistory() { UserID = user.ID, Date = DateTime.Now });

switch (user.Role.Name)

{

case "Менеджер":

m\_mainWindow.NavigateToManagerPage();

break;

case "Мастер":

m\_mainWindow.NavigateToMasterPage();

break;

case "Кадры":

m\_mainWindow.NavigateToStaffPage();

break;

case "Партнёр":

m\_mainWindow.NavigateToPartnerPage();

break;

case "Аналитик":

m\_mainWindow.NavigateToAnalystPage();

break;

}

}

}

}  
Код навигации  
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using Учебаня\_практика.Pages;

namespace Учебаня\_практика

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

mainFrame.Navigate(new AuthPage());

}

public void NavigateToManagerPage()

{

mainFrame.Navigate(new ManagerPage());

}

public void NavigateToAnalystPage()

{

mainFrame.Navigate(new AnalystPage());

}

public void NavigateToMasterPage()

{

mainFrame.Navigate(new MasterPage());

}

public void NavigateToPartnerPage()

{

mainFrame.Navigate(new PartnerPage());

}

public void NavigateToStaffPage()

{

mainFrame.Navigate(new StaffPage());

}

}

}  
Код Менеджера  
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using Учебаня\_практика.Properties;

using Учебаня\_практика.windows;

namespace Учебаня\_практика.Pages

{

/// <summary>

/// Логика взаимодействия для ManagerPage.xaml

/// </summary>

public partial class ManagerPage : Page

{

private BDEntities m\_entities;

public ManagerPage()

{

InitializeComponent();

m\_entities = BDEntities.GetInstance();

}

private void Page\_Loaded(object sender, RoutedEventArgs e)

{

Orders.ItemsSource = m\_entities.Orders.ToList();

Partners.ItemsSource = m\_entities.Partners.ToList();

SalePlaces.ItemsSource = m\_entities.SalePlaces.ToList();

SalePlaceType.ItemsSource = m\_entities.SalePlaceTypes.ToList();

Stores.ItemsSource = m\_entities.Stores.ToList();

Sales.ItemsSource = m\_entities.Sales.ToList();

Products.ItemsSource = m\_entities.Products.ToList();

}

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

{

if (string.IsNullOrEmpty(SalePlaceName.Text))

{

MessageBox.Show("Заполните название");

return;

}

if (SalePlaceType.SelectedItem == null)

{

MessageBox.Show("Укажите тип");

return;

}

var place = new SalePlace() { Name = SalePlaceName.Text, TypeID = (SalePlaceType.SelectedItem as SalePlaceType).ID };

m\_entities.SalePlaces.Add(place);

m\_entities.SaveChanges();

MessageBox.Show("Успешно");

SalePlaces.ItemsSource = m\_entities.SalePlaces.ToList();

}

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

{

var window = new CreateOrEditPartnerWindow();

window.Closed += UpdatePartners;

window.Show();

}

private void UpdatePartners(object sender, EventArgs e)

{

Partners.ItemsSource = m\_entities.Partners.ToList();

}

private void UpdateOrders(object sender, EventArgs e)

{

Orders.ItemsSource = m\_entities.Orders.ToList();

}

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

{

var window = new CreateOrEditPartnerWindow(Partners.SelectedItem as Partner);

window.Closed += UpdatePartners;

window.Show();

}

private void Partners\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

EditPartnerButton.IsEnabled = Partners.SelectedItem != null;

}

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

{

var window = new EditOrderWindow(Orders.SelectedItem as Order);

window.Closed += UpdateOrders;

window.Show();

}

private void Orders\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

var order = Orders.SelectedItem as Order;

EditOrderButton.IsEnabled = Orders.SelectedItem != null && order.Oplata && order.Proizweden && !order.Status;

}

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

{

var window = new EditStoreWindow(Stores.SelectedItem as Store);

window.Show();

}

private void Stores\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

EditStoreButton.IsEnabled = Stores.SelectedItem != null;

}

private void Products\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

EditProductButton.IsEnabled = Products.SelectedItem != null;

}

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

{

var window = new CreateOrEditProductWindow();

window.Closed += UpdateProducts;

window.Show();

}

private void UpdateProducts(object sender, EventArgs e)

{

Products.ItemsSource = m\_entities.Products.ToList();

}

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

{

var window = new CreateOrEditProductWindow(Products.SelectedItem as Product);

window.Closed += UpdateProducts;

window.Show();

}

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

{

var window = new CreateSaleWindow();

window.Closed += UpdateSales;

window.Show();

}

private void UpdateSales(object sender, EventArgs e)

{

Sales.ItemsSource = m\_entities.Sales.ToList();

}

}

}