Министерство образования и науки Республики Башкортостан

Государственное бюджетное профессиональное образовательное учреждение

Уфимский колледж статистики, информатики и

вычислительной техники

|  |  |
| --- | --- |
|  | УТВЕРЖДАЮ  Заместитель директора  по учебной работе  З.З. Курмашева  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г. |

Автоматизация учета гарантийного оборудования в сервисном центре

Пояснительная записка к дипломному проекту

|  |  |
| --- | --- |
| Рецензент  К.М. Айс  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г. | Руководитель  Р.С. Зиганшина  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г. |
|  | Выпускник гр. 16П-2  Н.С. Батурин  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г. |

2020

Министерство образования и науки Республики Башкортостан

Государственное бюджетное профессиональное образовательное учреждение

Уфимский колледж статистики, информатики и вычислительной техники

|  |  |
| --- | --- |
|  | УТВЕРЖДАЮ  Заместитель директора  по учебной работе  З.З. Курмашева  «\_\_» \_\_\_\_\_\_\_\_\_\_ 2020 год |

ЗАДАНИЕ

на дипломный проект студенту дневного отделения, группы 16П-2, специальности 09.02.03 Программирование в компьютерных системах.

Фамилия, имя, отчество: Батурин Никита Сергеевич

Тема дипломного проекта: «Автоматизация учета гарантийного оборудования в сервисном центре».

Текст задания:

при выполнении дипломного проекта должны быть решены следующие задачи:

1. разработана структура программы;
2. реализованы функции создания и изменения статусов заказов;
3. реализовано создание оформленных актов и отчетов;
4. построена справочная система.

В результате выполнения дипломного проекта должны быть представлены:

1. пояснительная записка, состоящая из следующих разделов:

Введение

1 Постановка задачи

2 Экспериментальный раздел

3 Экономический раздел

Заключение

Приложения

Список сокращений

Список использованных источников

1. электронный носитель, содержащий разработанный программный продукт;
2. презентация дипломного проекта в электронном виде.

Список рекомендуемых источников:

1. ГОСТ 19.701-90 ЕСПД. Схемы алгоритмов, программ, данных и систем. Обозначения условные и правила выполнения. [Текст]. – Введ. 1992–01–01 -М.: Изд-во стандартов, 1992.- 24 с.
2. Семакин, И.Г. Основы алгоритмизации и программирования [Текст]: учеб. для студ. учреждений СПО / И.Г. Семакин, А.П. Шестаков. - М.: Издательский центр "Академия", 2017.- 304 с.
3. [Агальцов В. П.](http://znanium.com/catalog.php?item=booksearch&code=%D0%B1%D0%B0%D0%B7%D1%8B+%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85&page=1#none) Базы данных. В 2-х кн. Кн. 2. Распределенные и удаленные базы данных: учеб. / В.П. Агальцов. - М.: ИД ФОРУМ: НИЦ Инфра-М, 2013. - 272 с.: ил. - <http://znanium.com/bookread2.php?book=372740>.
4. Википедия [Электронный ресурс] // Свободная энциклопедия. – Режим доступа: <http://ru.wikipedia.org/wiki/>, свободный.

Задание к выполнению получил «30» марта 2020г.

Студент Батурин Никита Сергеевич

Срок окончания «08» июня 2020 г.

Руководитель дипломного проекта Р.С. Зиганшина

Задание рассмотрено на заседании цикловой комиссии информатики   
«25» марта 2020 г. Протокол №5.

Председатель цикловой комиссии информатики О.В. Фатхулова

Министерство образования и науки Республики Башкортостан

Государственное бюджетное профессиональное образовательное учреждение

Уфимский колледж статистики, информатики и

вычислительной техники

ЗАКЛЮЧЕНИЕ

на дипломный проект

Дипломник

Группа

Специальность 09.02.03 Программирование в компьютерных системах

Тема

Объем дипломного проекта:

количество листов пояснительной записки

количество листов графической части

Заключение о степени соответствия заданию на дипломный проект

Характеристика качеств, проявленных выпускником при работе над дипломным проектом: самостоятельность, дисциплинированность, умение планировать работу и пользоваться литературным материалом и т.д.

Положительные стороны

Недостатки

Характеристика общетехнической и специальной подготовки выпускника

Заключение и предлагаемая оценка за дипломный проект

Руководитель

Должность преподаватель

Место работы ГБПОУ Уфимский колледж статистики, информатики и вычислительной техники

« » 2020г.

Подпись

Министерство образования и науки Республики Башкортостан

Государственное бюджетное профессиональное образовательное учреждение

Уфимский колледж статистики, информатики и

вычислительной техники

РЕЦЕНЗИЯ

на выпускную квалификационную работу

Дипломник

Группа

Специальность 09.02.03 Программирование в компьютерных системах

Тема

Объем дипломного проекта:

количество листов пояснительной записки

количество листов графической части

Заключение о степени соответствия заданию на дипломный проект

Характеристика выполнения каждого раздела дипломного проекта

Перечень положительных качеств дипломного проекта, возможность его использования на производстве

Недостатки

Оценка качества выполнения графической части дипломного проекта

Оценка качества выполнения пояснительной записки дипломного проекта

Оценка общеобразовательной и технической подготовки выпускника

Отзыв о дипломном проекте в целом, предлагаемая оценка

Рецензент

Должность

Место работы

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г.

Подпись

Изм.

Лист

№ докум.

Подпись

Дата

Лист

2

40.Б-1034-2020 09.02.03 ДП-ПЗ

Разраб.

Батурин Н.С.

Провер.

Зишаншина Р.С.

Реценз.

*Айс К.М.*

Н. Контр.

*Каримова Р.Ф.*

Утверд.

*Курмашева З.З*

Автоматизация учета гарантийного оборудования в сервисном центре

Лит.

Листов

70

УКСИВТ 16П-2

АННОТАЦИЯ

Пояснительная записка к дипломному проекту содержит постановку и программу решения задачи «Автоматизация учета гарантийного оборудования в сервисном центре».

Программа service\_center.exe написана на языке C# в среде программирования Microsoft Visual Studio 2019 с использованием технологий Windows Forms и сервера баз данных MySQL Server, предназначена для работы в операционной системе Microsoft Windows 10 и отлажена на данных контрольного примера.

СОДЕРЖАНИЕ

|  |  |
| --- | --- |
|  | лист |
| Введение | 4 |
| 1 Постановка задачи | 6 |
| 1.1 Описание предметной области | 6 |
| 1.2 Функциональная модель предметной области | 8 |
| 1.3 Описание входной информации | 9 |
| 1.4 Описание выходной информации | 9 |
| 1.5 Общие требования к программному продукту | 11 |
| 1.6 Описание структуры базы данных | 12 |
| 1.7 Контрольный пример | 15 |
| 2 Экспериментальный раздел | 18 |
| 2.1 Описание программы | 18 |
| 2.2 Протокол тестирование программного продукта | 21 |
| 2.3 Руководство пользователя | 33 |
| 3 Экономический раздел | 38 |
| 3.1 Расчёт затрат на создание программного продукта | 38 |
| 3.2 Расчет цены предложения и минимального количества копий тиражирования | 41 |
| Заключение | 45 |
| Приложение А. Исходный код программы | 46 |
| Список использованных источников | 69 |

# 

# ВВЕДЕНИЕ

В современном мире информация рассматривается, как один из основных компонентов развития общества, а информационные системы и технологии используются в производственной, управленческой, финансовой и многих других профессиональных отраслях.

Отыскание рациональных решений в любой сфере требует больших финансовых и временных ресурсов. А обработка массивного объема информации невозможна без привлечения специальных технологических средств.

Внедрение информационных технологий, современных средств переработки и передачи информации в различные сферы деятельности послужило началом информатизации, которая является реакцией общества на потребность в существенном увеличении производительности труда в различных сферах деятельности человека, а также ее автоматизация и учета, что позволяет облегчить работу ручного труда.

Для успешного развития бизнеса необходимо решить проблемы учета приема и выполнения заявок клиентов.

Актуальностью темы заключается в экономической целесообразности использования денежных ресурсов фирмы заказчика, а также в последующей поддержки программного продукта для исправления непредвиденных неисправностей, модернизации функциональности и реализации новых требований информационной системы.

Цель дипломного проекта состоит в разработке автоматизированной системы, которая облегчит работу гарантийного сервисного центра и упростит учет проведенных ремонтов.

Для достижения поставленной цели необходимо решить следующие задачи:

* изучить структуру организации, выявление его основных задач и функций;
* провести анализ информационных процессов, протекающих в организации;
* выявить функциональные задачи информационной системы;
* выполнить проектирование базы данных, содержащую сведения о заказах;
* разработать и протестировать приложения для автоматизации учета гарантийного оборудования в сервисном центре, которое позволит принимать заказы, формировать акты и создавать отчет.

# 1 Постановка задачи

# Описание предметной области

Существует магазин розничной торговли, специализирующийся на продаже компьютерной, цифровой и бытовой техники, магазин имеет гарантийный сервисный центр, для него требуется разработать программный продукт «Автоматизация учета гарантийного оборудования в сервисном центре».

В сервисный центр по ремонту гарантийного оборудования от клиентов поступают заказы на неисправность устройства различных типов, производителей и моделей, купленные у вышеуказанного магазина розничной торговли.

Прием оборудования в ремонт осуществляет приемщик – сотрудник сервисного центра. Он регистрирует новые заказы. Для этого с клиента требуется: чек покупки неисправного оборудования и гарантийных талон, на котором должен продолжаться период гарантийного обслуживания; ФИО и мобильный телефон клиента.

С помощью чека и указанных в нем артикула или названия товара идентифицируется гарантийный период у оборудования, с чека берется дата покупки и подтверждается продолжающаяся гарантия.

Созданный заказ будет иметь следующие поля:

* код заказа;
* дата покупки по чеку;
* артикул товара;
* код клиента;
* серийный номер;
* дата приема;
* дата выдачи (будет пустым до момента выдачи);
* код статуса.

После регистрации заказа печатается акт приема, который выдается клиенту и является документом, подтверждающим факт приема изделия сервисным центром на диагностику и ремонт.

Мастер сервисного центра принимает заказ, производит диагностику изделия, выявляет имеющиеся неисправности и целесообразность дальнейшего ремонта. При возможности осуществляется ремонт изделия. Мастер проводит необходимые ремонтные работы для устранения неисправности, и вносит в изменение состояния отметки о текущем статусе заказа:

* проводится диагностика;
* диагностика проведена, ремонт невозможен;
* диагностика проведена, ремонт возможен;
* диагностика проведена, не гарантийный случай;
* ремонт проведен успешно.

При успешном завершении ремонта клиент ставится в известность об окончании ремонта. В случае не гарантийной неисправности или невозможности ремонта, клиент также ставится в известность. Далее клиенту возвращается изделие, и в заказе фиксируется дата выдачи. Заказу присваивается статус «изделие выдано клиенту».

Программа должна предусматривать разные режимы работы в зависимости от авторизированного пользователя. С данным приложением будут работать следующие группы пользователей:

* приемщик;
* мастер.

Приемщик должен иметь возможность решать следующие задачи:

* создавать новый заказ;
* оформлять акты приема и выдачи по выбранному заказу;
* формировать отчет из отфильтрованного списка заказов.

Мастер должен иметь возможность менять отметку статуса у выбранного заказа.

# Функциональная модель предметной области

AS-IS – модель «как есть», модель существующего состояния организации. Данная модель позволяет систематизировать протекающие в данный момент процессы, а также используемые информационные объекты. На основе этого выявляются узкие места в организации и взаимодействии бизнес-процессов, определяется необходимость тех или иных изменения в существующей структуре.

Функциональная модель представлена на рисунке 1.2.1.

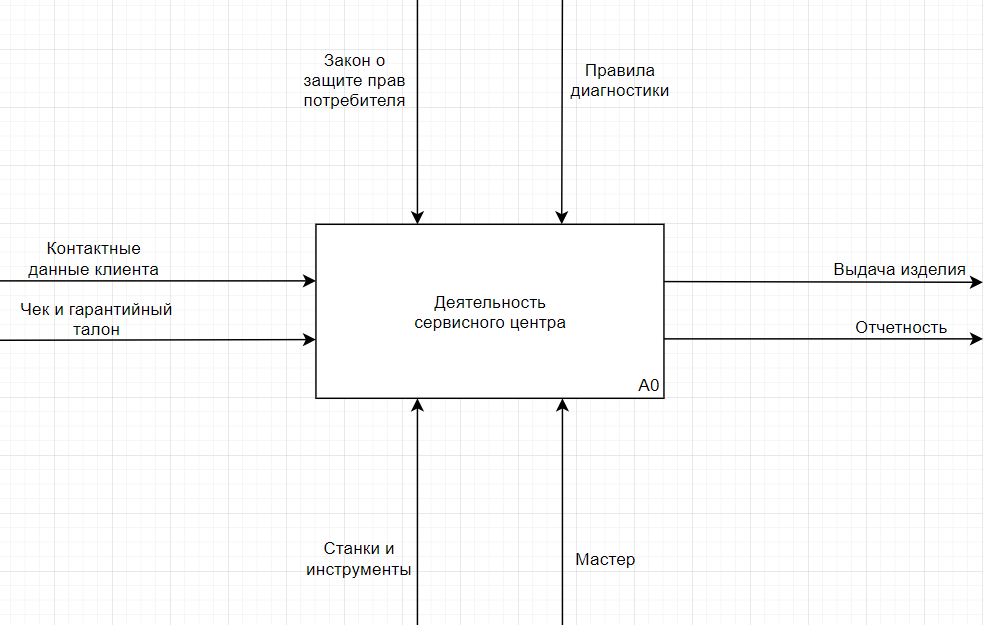


Рисунок 1.2.1 – Функциональная модель

Декомпозиция функциональной модели представлена на рисунке 1.2.2.

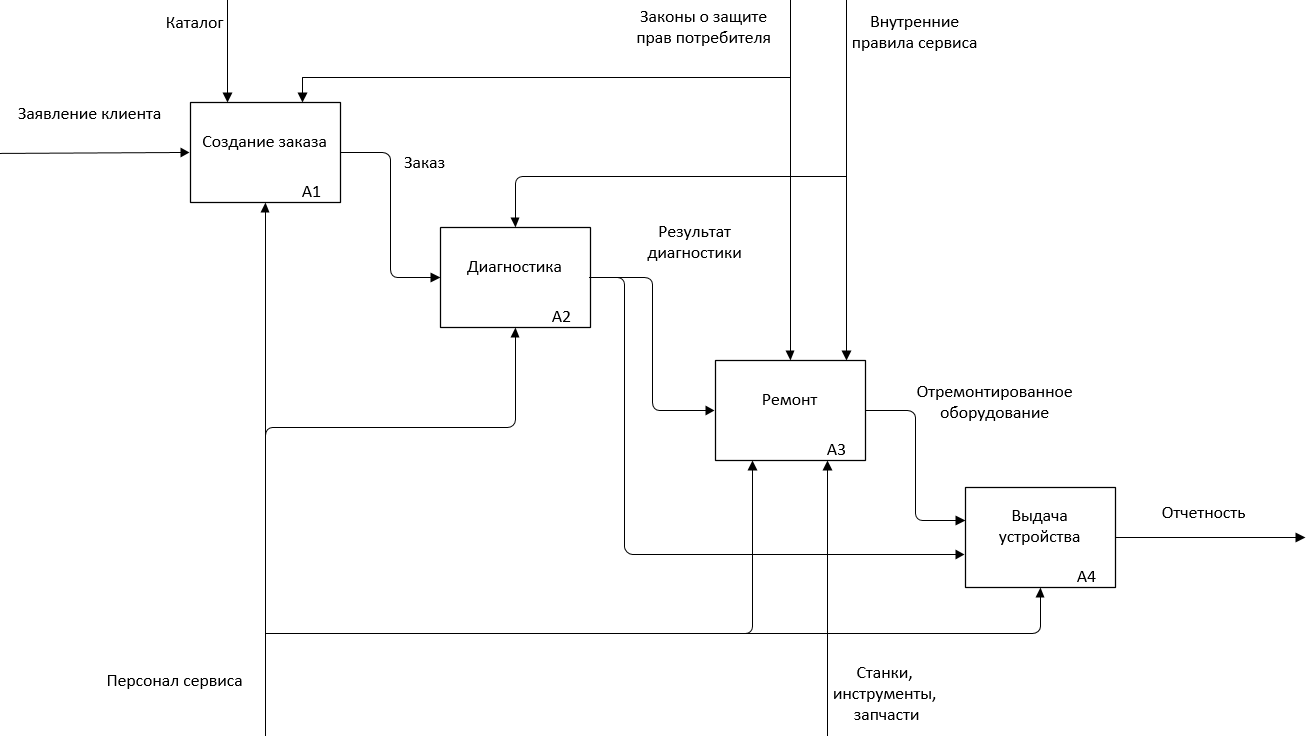


Рисунок 1.2.2 – Декомпозиция функциональной модели

# 1.3 Описание входной информации

Входными данными для формирования заказа будет являться следующая информация:

* чек покупки – содержит артикул, название товара и дату покупки;
* информация о клиенте – содержит личные данные клиента;
* каталог товаров – список товаров с указанием срока гарантии.

Входные данные представлены в таблице 1.3.1.

Таблица 1.3.1 – Описание входных документов

|  |  |  |
| --- | --- | --- |
| Наименование документа (шифр) | Периодичность поступления документа | Откуда поступает документ |
| Чек покупки | При поступлении | От клиента |
| Информация о клиенте | При поступлении | От клиента |
| Каталог товаров | При поступлении | От приемщика |

# 1.4 Описание выходной информации

Выходными данными будут являться следующие документы:

* акт приема – сформированный документ о приеме устройства;
* акт выдачи – сформированный документ о выдаче устройства;
* отчет – сформированный по списку заказов отчет.

Выходные данные представлены в таблице 1.4.1.

Таблица 1.4.1 – Описание выходных документов

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Наименование документа  (шифр) | Периодичность выдачи документа | Кол-во экз. | Куда передаются | Поля сортировки | Поля группировки | Итоги |
| Акт приема | При приеме | 1 | Клиенту | – | – | – |
| Акт выдачи | При выдачи | 1 | Клиенту | – | – | – |
| Отчет | При необходимости | 1 | Приемщику | Код заказа | Статус, дата | – |

Шаблоны выходных документов представлены на рисунках 1.4.1 – 1.4.3.

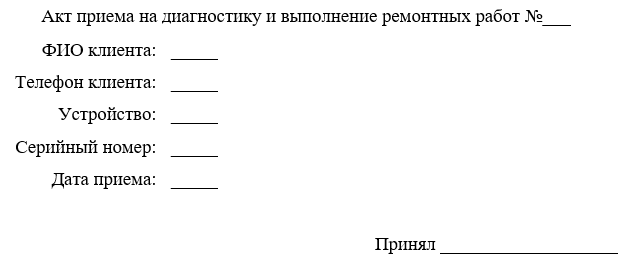


Рисунок 1.4.1– Шаблон акта приема

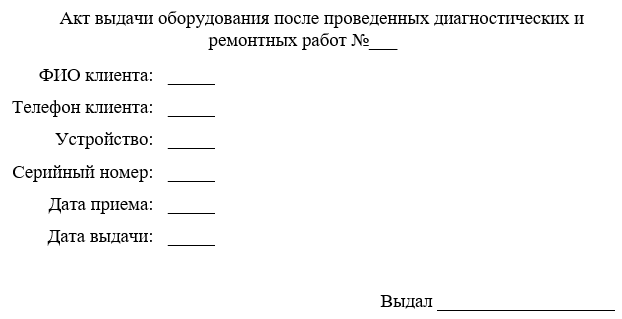


Рисунок 1.4.2 – Шаблон акта выдачи

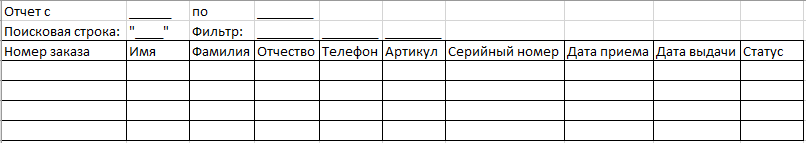


Рисунок 1.4.3 – Шаблон отчета

# 1.5 Общие требования к программному продукту

Общее наименование информационной системы «Service\_center». Для работы с ним приемщик и мастер должны обладать базовыми навыками работы с ПК.

Минимальные системные требования для компьютера, на котором планируется использование программы:

* + - * объем ОЗУ: 4 ГБ и выше;
* процессор: двухъядерные с тактовой частотой 3 ГГц и лучше;
* место на жестком диске: 4 ГБ и больше;
  + - * ОС: Windows 10 64 bit;
      * монитор;
      * принтер;
      * манипулятор типа «мышь»;
      * манипулятор типа «клавиатура»;
      * драйверы: .Net Framework 2.0 – 4.5, Visual C++ 2005–2019 (x86, x64).

Техническими и программными средствами необходимыми для ввода продукта в эксплуатацию является удаленный сервер с СУБД MySQL Server. Сервер должен находиться на стабильном хостинге обладающим следующими характеристиками:

* + - * объем памяти не менее 10Гб;
      * поддержка СУБД MySQL Server.

В результате работы приложения будут формироваться текстовые документы и электронные таблицы, необходимо наличие текстового редактора Microsoft Word и редактора электронных таблиц Microsoft Excel.

# 1.6 Описание структуры базы данных

На основании входных документов создаются таблицы базы данных, описание которых приведено ниже в таблице 1.6.1.

Для создания базы данных будет использована СУБД MySQL.

Таблица 1.6.1 – Описание структуры базы данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Описание поля | Тип данных | Размер поля | Тип ключа[[1]](#footnote-1) |
| 1 | 2 | 3 | 4 | 5 |
| Users(Пользователи) | | | | |
| user\_id | Код пользователя | INT | 4 | PK |
| login | Логин пользователя | VARCHAR | 76 |  |
| name | Имя пользователя | VARCHAR | 301 |  |

Продолжение таблицы 1.6.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| surname | Фамилия пользователя | VARCHAR | 301 |  |
| patronymic | Отчество пользователя | VARCHAR | 301 |  |
| password | Пароль пользователя | VARCHAR | 301 |  |
| role | Роль пользователя | VARCHAR | 136 |  |
| Catalog(Список товаров) | | | | |
| article | Артикул | INT | 4 | PK |
| title | Название | VARCHAR | 766 |  |
| warranty\_period | Срок гарантии в месяцах | INT | 4 |  |
| Clients(Список клиентов) | | | | |
| client\_id | Код клиента | INT | 4 | PK |
| name | Имя клиента | VARCHAR | 301 |  |
| surname | Фамилия клиента | VARCHAR | 301 |  |
| patronymic | Отчество клиента | VARCHAR | 301 |  |
| phone | Телефон клиента | VARCHAR | 136 |  |
| Statuses(Справочник статусов) | | | | |
| status\_id | Код статуса | INT | 4 | PK |
| name | Название статуса | VARCHAR | 301 |  |

Продолжение таблицы 1.6.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Orders(Журнал заказов) | | | | |
| order\_id | Код заказа | INT | 4 | PK |
| date\_check | Дата чека | DATE | 3 |  |
| article | Артикул | INT | 4 | FK |
| client\_id | Код клиента | INT | 4 | FK |
| serial\_number | Серийных номер | VARCHAR | 301 |  |
| date\_in | Дата приема | DATE | 3 |  |
| date\_out | Дата выдачи | DATE | 3 |  |
| status\_id | Код статуса | INT | 4 | FK |

На рисунке 1.6.1 представлена схема отношений базы данных.

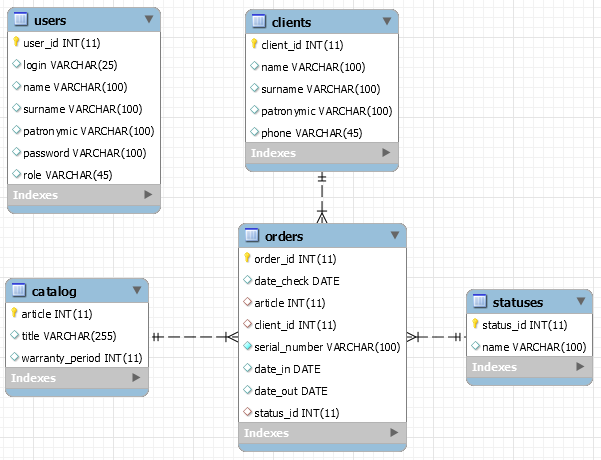


Рисунок 1.6.1 – Схема отношений базы данных

# 1.7 Контрольный пример

Контрольный пример является ручным подсчётом задачи. Он представляет собой вариант задачи с известными результатами и используется для проверки правильности решения на ПЭВМ.

По составленной программе обрабатываются исходные данные контрольного примера. Полученные результаты сравниваются с известными результатами контрольного примера. При несовпадении результатов производится поиск, исправление ошибок, и снова производится выполнение программы. И так до тех пор, пока не будет получен правильный результат.

Входные данные контрольного примера представлены в таблицах 1.7.1 – 1.7.5.

Выходные данные контрольного примера представлены в таблицах 1.7.6 – 1.7.8.

Таблица 1.7.1 – Входные данные для таблицы список пользователей

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Код пользователя | Логин | Имя | Фамилия | Отчество | Пароль |
| 1 | user\_r1 | Антон | Богов | Михайлович | 46h4m1 |
| 2 | user\_m1 | Борис | Гагарин | Семенов | 64n5u9 |

Таблица 1.7.2 – Входные данные для таблицы каталог товаров

|  |  |  |
| --- | --- | --- |
| Артикул | Название | Срок гарантии |
| 1381298 | Видеокарта PCI-E MSI GeForce RTX 2060 Super VENTUS OC 8192MB 256bit GDDR6 | 36 |

Таблица 1.7.3 – Входные данные для таблицы список клиентов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Код клиента | Имя | Фамилия | Отчество | Телефон |
| 1 | Василий | Укупов | Леонидович | 89245622365 |

Таблица 1.7.4 – Входные данные для таблицы справочник статусов

|  |  |
| --- | --- |
| Код статуса | Название |
| 1 | проводится диагностика |
| 2 | диагностика проведена, ремонт невозможен |
| 3 | диагностика проведена, ремонт возможен |
| 4 | диагностика проведена, не гарантийный случай |
| 5 | ремонт проведен успешно |
| 6 | изделие выдано клиенту |

Таблица 1.7.5 – Входные данные для таблицы журнал заказов

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Код заказа | Дата чека | Артикул | Код клиента | Серийный номер | Дата приема | Дата выдачи | Код статуса |
| 1 | 11.01.2020 | 1381298 | 1 | 345-st-562 | 02.03.2020 | 08.03.2020 | 6 |

Таблица 1.7.6 – Выходные данные акт приема

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Код заказа | ФИО клиента | Телефон клиента | Устройство | Серийный номер | Дата приема | Принял |
| 1 | Укупов Василий Леонидович | 89245622365 | Видеокарта PCI-E MSI GeForce RTX 2060 Super VENTUS OC | 345-st-562 | 02.03.2020 | Богов Антон Михайлович |

Таблица 1.7.7 – Выходные данные акт выдачи

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Код заказа | ФИО клиента | Телефон клиента | Устройство | Серийный номер | Дата приема | Дата выдачи | Выдал |
| 1 | Укупов Василий Леонидович | 89245622365 | Видеокарта PCI-E MSI GeForce RTX 2060 Super VENTUS OC | 345-st-562 | 02.03.2020 | 08.03.2020 | Богов Антон Михайлович |

Таблица 1.7.8 – Выходные данные отчет за 01.03.2020 по 31.03.2020

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Номер заказа | Имя | Фамилия | Отчество | Телефон | Артикул | Серийный номер | Дата приема | Дата выдачи | Статус |
| 1 | Василий | Укупов | Леонидович | 89245622365 | 1381298 | 345-st-562 | 02.03.2020 | 02.03.2020 | Изделие выдано клиенту |

2 Экспериментальный раздел

2.1 Описание программы

Программа имеет модульную структуру, она было разработана в среде разработки Visual Studio 2019 с использованием технологии Windows Forms и СУБД MySQL Server. Содержит программные модули: Login, MasterForm, ReceiverForm, CreateOrder, OrderList, PrintForm, ConnectionDB. Описание модулей приведено в таблице 2.1.1.

Таблица 2.1.1 – Описание модулей

|  |  |
| --- | --- |
| Модуль | Назначение |
| 1 | 2 |
| Модуль Login – Форма авторизации, главный модуль | |
| LoginForm\_Load | Загрузка формы |
| LoginButton\_Click | Нажатие кнопки входа |
| Модуль MasterForm – Форма мастера для изменения статусов заказов | |
| MasterForm\_FormClosing | Выход из авторизации перед закрытием формы |
| OrderListDGV\_SelectionChanged | Выбор другой строки |
| SearchOrderTextBox\_TextChanged | Поиск одновременно с изменением |
| ConfirmStatusButton\_Click | Подтверждение изменения статуса |
| Модуль ReceiverForm – Форма приемщика для оформления нового заказа и печати актов | |

Продолжение таблицы 2.1.1

|  |  |
| --- | --- |
| ReceiverForm\_FormClosing | Выход из авторизации перед закрытием формы |
| SideBarButton\_Click | Меню для открытия MDI форм |
| Модуль CreateOrder – Форма для оформления заказа | |
| SearchTextBox\_TextChanged | Поиск одновременно с изменением |
| CatalogDGV\_SelectionChanged | Выбор другой строки |
| InfoBoxes\_TextChanged | Событие изменения данных в полях |
| CheckFillFields | Первичная и вторичная валидации проверяющая данных в полях |
| SpendButton\_Click | Проведение заказа и открытие формы формирования актов |
| Модуль OrderList – Форма для отображения списка заказов | |
| SearchOrderTextBox\_TextChanged | Поиск одновременно с изменением |
| PrintActButton\_Click | Кнопка для открытия окна формирования актов |
| PrintReportButton\_Click | Кнопка для формирования отчета |
| Модуль PrintForm – Форма для печати актов приема и выдачи | |
| Replace | Заменяет элемент шаблона в документе |

Продолжение таблицы 2.1.1

|  |  |
| --- | --- |
| ActReceiveButton\_Click | Кнопка для печати акта о приеме |
| ActIssueButton\_Click | Кнопка для печати акта о выдаче |
| BackToFormButton\_Click | Кнопка для возвращения в прошлую форму |
| Модуль ConnectionDB – Класс для подключения к базе данных и реализации этого подключения | |
| CheckLogin | Метод для верификации логина и пароля пользователя |
| DisplayCatalog | Метод отображает каталог с заданной строкой поиска |
| CompareDublicatesSerialNumbers | Метод для проверки отсутствия дубликатов серийных номеров у одно устройства |
| InsertingClients | Метод для вставки клиента |
| InsertingOrder | Метод для проведения заказа |
| DisplayOrders | Метод отображает журнал заказов с заданной строкой поиска |
| ReportData | Метод для получения данных для актов |
| TitleOrder | Метод для получения названия ремонтируемого устройства |
| StatusList | Метод для получения списка статусов |
| UpdateStatus | Метод обновляет статус у заданного заказа |

Модульная схема программы представлена на рисунке 2.1.1.

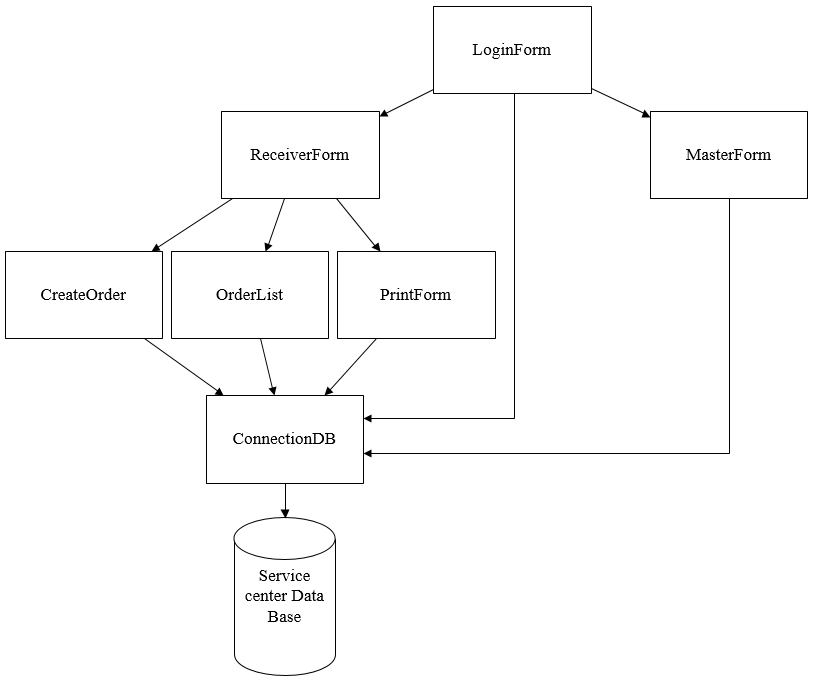


Рисунок 2.1.1 – Модульная схема приложения

2.2 Протокол тестирование программного продукта

В ходе тестирования программного продукта на корректных и некорректных данных не было обнаружено ошибок, которые влияли бы на работу самого программного продукта и всей системы.

Данный программный продукт удовлетворяет всем предъявленным требованиям, имеет комфортный интерфейс и интуитивно понятный функционал, исключает появления системных ошибок.

В таблицах 2.2.1 – 2.2.10 представлена общая информация о тестировании приложения.

Таблица 2.2.1 – Протокол тестирования авторизации на некорректных данных

|  |  |
| --- | --- |
| Общая информация о тестировании | |
| Название проекта | Service\_center |
| Номер версии | 1.0 |
| Имя тестера | Батурин Никита Сергеевич |
| Даты тестирования | 10.05.2020 |
| Описание информационных полей для тестирования | |
| Наименование | Авторизация |
| Test Case # | TC\_F1\_1 |
| Приоритет тестирования (Малый/Средний/высокий) | Высокий |
| Название тестирования | Проверка авторизации в модуле LoginForm с неправильными логином и паролем |
| Резюме испытания | Необходимо добиться корректного поведения программы при вводе неправильных данных |
| Шаги тестирования | Введены неверные логин и пароль  Нажата кнопка «войти» |
| Данные тестирования | Логин: qwer  Пароль: qwer |
| Ожидаемый результат | Программа должна оповестить о неверных логине или пароле |
| Фактический результат | В результате тестирования было получено оповещение о том, что были введены неверные логин или пароль (см. рисунок 2.2.1) |
| Предпосылки | Запустить программу |
| Постусловия | Работоспособность системы не нарушена |

Продолжение таблицы 2.2.1

|  |  |
| --- | --- |
| Статус (Pass/Fail) | Pass |

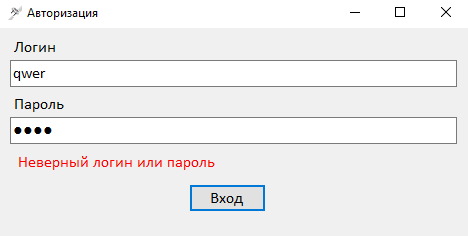


Рисунок 2.2.1 – Сообщение о некорректных данных аккаунта

Таблица 2.2.2 – Протокол тестирования авторизации на корректных данных

|  |  |
| --- | --- |
| Общая информация о тестировании | |
| Название проекта | Service\_center |
| Номер версии | 1.0 |
| Имя тестера | Батурин Никита Сергеевич |
| Даты тестирования | 10.05.2020 |
| Описание информационных полей для тестирования | |
| Наименование | Описание |
| Test Case # | TC\_F1\_2 |
| Приоритет тестирования (Малый/Средний/высокий) | Высокий |
| Название тестирования | Проверка авторизации в модуле LoginForm с правильными логином и паролем |
| Резюме испытания | Необходимо добиться корректного поведения программы при вводе правильных данных |
| Шаги тестирования | Введены верные логин и пароль  Нажата кнопка «войти» |
| Данные тестирования | Логин: user\_r  Пароль: 123 |
| Ожидаемый результат | Программа должна подключиться к системе |
| Фактический результат | В результате программа успешно подключилась к системе и отобразилось окно оформления заказа (см. рисунок 2.2.2) |
| Предпосылки | Запустить программу |
| Постусловия | Работоспособность системы не нарушена |
| Статус (Pass/Fail) | Pass |

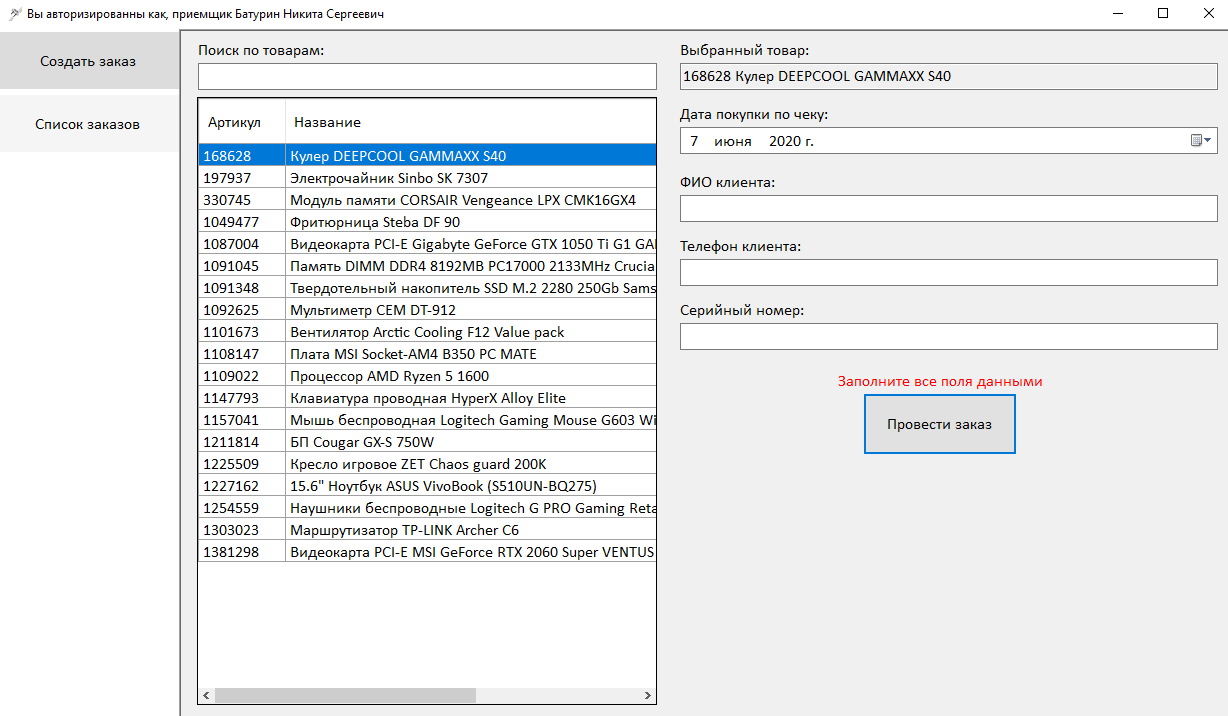


Рисунок 2.2.2 – Форма оформления заказа

Таблица 2.2.3 – Протокол тестирования проведения заказа с отсутствием данных

|  |  |
| --- | --- |
| Общая информация о тестировании | |
| Название проекта | Service\_center |
| Номер версии | 1.0 |
| Имя тестера | Батурин Никита Сергеевич |
| Даты тестирования | 10.05.2020 |
| Описание информационных полей для тестирования | |
| Наименование | Описание |
| Test Case # | TC\_F1\_3 |
| Приоритет тестирования (Малый/Средний/высокий) | Высокий |
| Название тестирования | Проверка оформления заказа |
| Резюме испытания | Необходимо добиться корректного поведения программы при проведении заказа с пустыми полями |
| Шаги тестирования | Совершен вход в систему  Нажата кнопка «Провести заказ» |
| Данные тестирования | Логин: user\_r  Пароль: 123 |
| Ожидаемый результат | Отображении окна уведомляющего об некорректности данных |
| Фактический результат | После нажатия кнопки «Провести заказ» отобразилось окно уведомление (см. рисунок 2.2.3) |
| Предпосылки | Запустить программу и авторизоваться |
| Постусловия | Работоспособность системы не нарушена |
| Статус (Pass/Fail) | Pass |

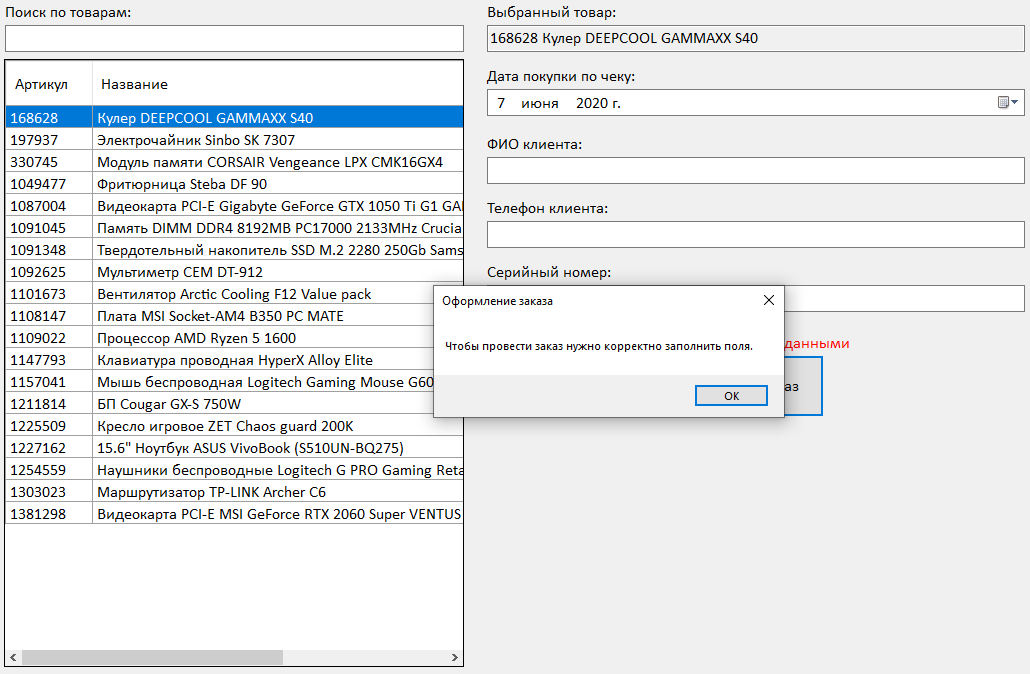


Рисунок 2.2.3 – Невозможность провести заказ с незаполненными полями

Таблица 2.2.4 – Протокол тестирования проведения заказа на некорректных данных

|  |  |
| --- | --- |
| Общая информация о тестировании | |
| Название проекта | Service\_center |
| Номер версии | 1.0 |
| Имя тестера | Батурин Никита Сергеевич |
| Даты тестирования | 10.05.2020 |
| Описание информационных полей для тестирования | |
| Наименование | Описание |
| Test Case # | TC\_F1\_4 |
| Приоритет тестирования (Малый/Средний/высокий) | Высокий |
| Название тестирования | Проверка оформления заказа |
| Резюме испытания | Необходимо добиться корректного поведения программы при проведении заказа с некорректными данными |

Продолжение таблицы 2.2.4

|  |  |
| --- | --- |
| Шаги тестирования | Совершен вход в систему  Поля заполнены некорректными данными  Нажата кнопка «Провести заказ» |
| Данные тестирования | Логин: user\_r  Пароль: 123 |
| Ожидаемый результат | Должно высветится уведомление о некорректности введенных данных |
| Фактический результат | После нажатия кнопки «Провести заказ» отобразилось уведомление о некорректности введенных данных (см. рисуноки 2.2.4 – 2.2.6) |
| Предпосылки | Запустить программу и авторизоваться |
| Постусловия | Работоспособность системы не нарушена |
| Статус (Pass/Fail) | Pass |

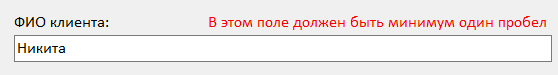


Рисунок 2.2.4 – Уведомление о некорректности ФИО клиента

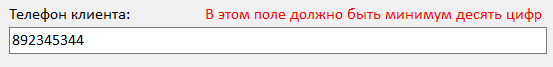


Рисунок 2.2.5 – Уведомление о некорректности телефона клиента



Рисунок 2.2.6 – Уведомление о некорректности серийного номера

Таблица 2.2.5 – Протокол тестирования проведения заказа на корректных данных

|  |  |
| --- | --- |
| Общая информация о тестировании | |
| Название проекта | Service\_center |
| Номер версии | 1.0 |
| Имя тестера | Батурин Никита Сергеевич |
| Даты тестирования | 10.05.2020 |
| Описание информационных полей для тестирования | |
| Наименование | Описание |
| Test Case # | TC\_F1\_5 |
| Приоритет тестирования (Малый/Средний/высокий) | Высокий |
| Название тестирования | Проверка оформления заказа |
| Резюме испытания | Необходимо добиться корректного поведения программы при проведении заказа с корректными данными |
| Шаги тестирования | Совершен вход в систему  Поля заполнены корректными данными  Нажата кнопка «Провести заказ» |
| Данные тестирования | Логин: user\_r  Пароль: 123 |
| Ожидаемый результат | Должно отобразится окно об успешном проведении заказа |
| Фактический результат | После нажатия кнопки «Провести заказ» отобразился окно об успешном проведении заказа (см. рисуноки 2.2.7) |
| Предпосылки | Запустить программу, авторизоваться и заполнить все поля |

Продолжение таблицы 2.2.5

|  |  |
| --- | --- |
| Постусловия | Работоспособность системы не нарушена |
| Статус (Pass/Fail) | Pass |

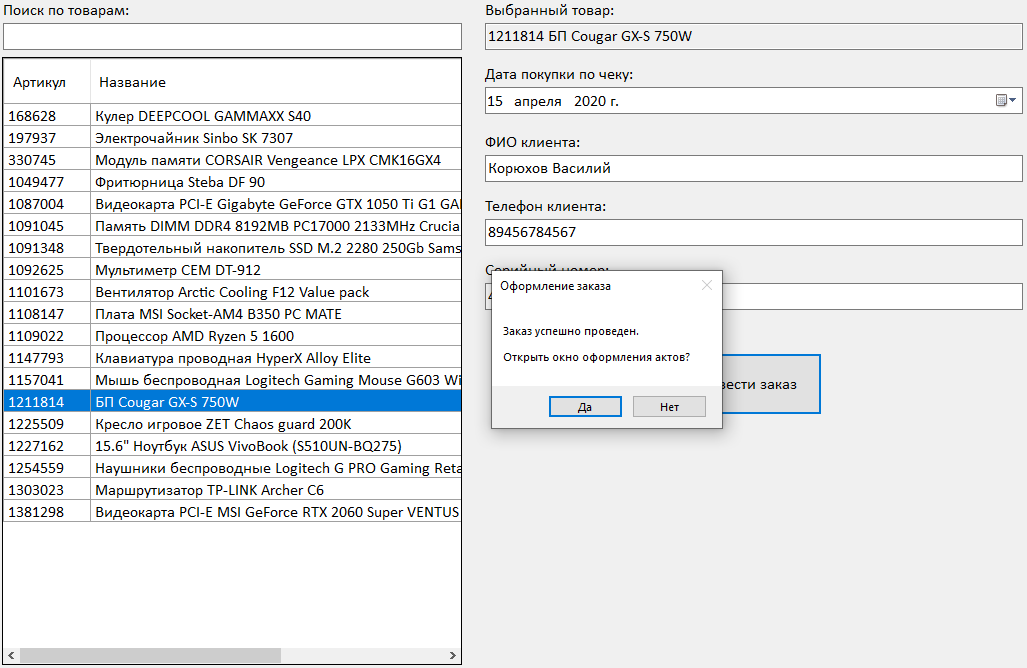


Рисунок 2.2.7 – Отображение окна об успешно проведении заказа

Таблица 2.2.6 – Протокол тестирования взятия заказа мастером на диагностику и ремонт и дальнейшего изменения статуса заказа

|  |  |
| --- | --- |
| Общая информация о тестировании | |
| Название проекта | Service\_center |
| Номер версии | 1.0 |
| Имя тестера | Батурин Никита Сергеевич |
| Даты тестирования | 10.05.2020 |
| Описание информационных полей для тестирования | |
| Наименование | Описание |
| Test Case # | TC\_F1\_6 |
| Приоритет тестирования (Малый/Средний/высокий) | Высокий |
| Название тестирования | Проверка обновления статуса заказа |

Продолжение таблицы 2.2.6

|  |  |
| --- | --- |
| Резюме испытания | Необходимо добиться корректного поведения программы при изменении статуса заказа |
| Шаги тестирования | Совершен вход в систему  Мастером был выбран заказ и проведена диагностика и ремонт  В выпадающем списке статусов для заданного заказа был выбран «ремонт проведен успешно»  Нажата кнопка «Подтвердить статус» |
| Данные тестирования | Логин: user\_m  Пароль: 123 |
| Ожидаемый результат | Обновления статуса заказа на «ремонт проведен успешно» |
| Фактический результат | После нажатия кнопки «Подтвердить статус» статус изменился на «ремонт проведен успешно» (см. рисунок 2.2.8) |
| Предпосылки | Запустить программу, авторизоваться и указать новый статус |
| Постусловия | Работоспособность системы не нарушена |
| Статус (Pass/Fail) | Pass |

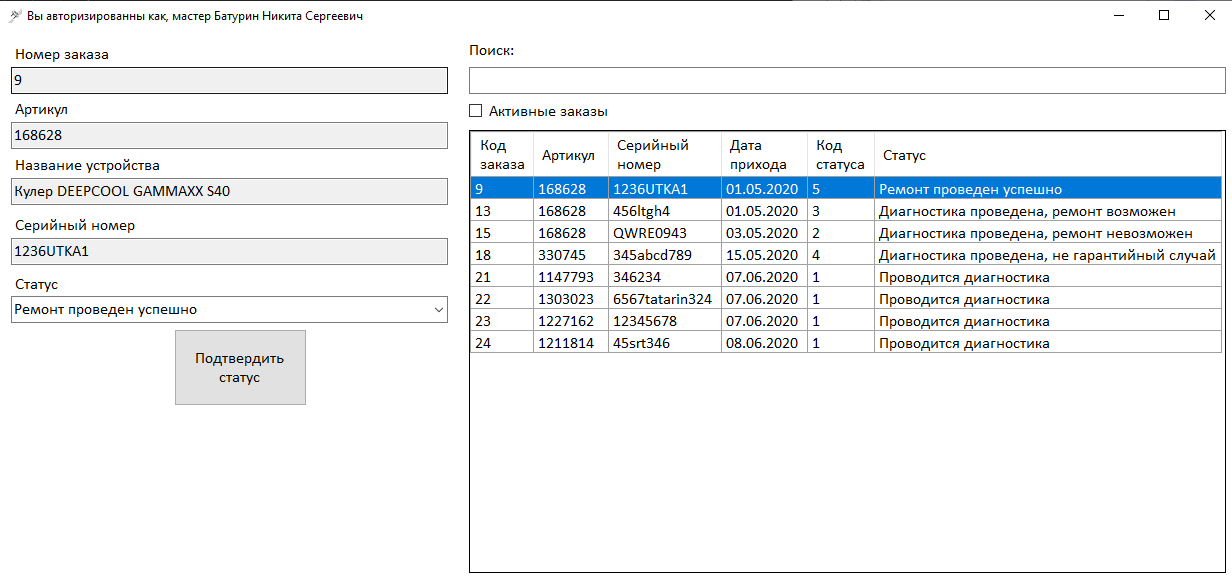


Рисунок 2.2.8 – Успешное обновление статуса заказа

Таблица 2.2.7 – Протокол тестирования печати акта о выдаче

|  |  |
| --- | --- |
| Общая информация о тестировании | |
| Название проекта | Service\_center |
| Номер версии | 1.0 |
| Имя тестера | Батурин Никита Сергеевич |
| Даты тестирования | 10.05.2020 |
| Описание информационных полей для тестирования | |
| Наименование | Описание |
| Test Case # | TC\_F1\_7 |
| Приоритет тестирования (Малый/Средний/высокий) | Высокий |
| Название тестирования | Проверка печати акта о выдаче |
| Резюме испытания | Необходимо добиться корректного поведения программы на печать акта о выдаче и обновления статуса на «изделие выдано клиенту» |
| Шаги тестирования | Совершен вход в систему  Клиент пришел забрать устройство  Печатается акт о выдаче |

Продолжение таблицы 2.2.7

|  |  |
| --- | --- |
| Данные тестирования | Логин: user\_r  Пароль: 123 |
| Ожидаемый результат | Обновления статуса заказа на «изделие выдано клиенту» и печать акта о выдаче |
| Фактический результат | После нажатия кнопки «Подтвердить выдачу» статус изменился на «изделие выдано клиенту» и был оформлен акт о выдаче (см. рисунок 2.2.9) |
| Предпосылки | Запустить программу, авторизоваться и оформить акт о выдаче |
| Постусловия | Работоспособность системы не нарушена |
| Статус (Pass/Fail) | Pass |

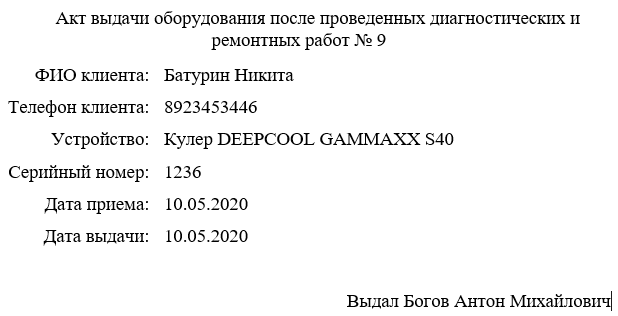


Рисунок 2.2.9 – Сформированный акт о выдаче

2.3 Руководство пользователя

Руководство пользователя относится к эксплуатационной документации. Основная цель руководства пользователя заключается в обеспечении пользователя необходимой информацией для самостоятельной работы с программой или автоматизированной системой.

Для использования программного решения пользователю необходимо иметь персональный компьютер и постоянное подключение к базе данных через интернет.

Пользователями программы являются приемщики и мастера, для каждого пользователя существует роль, которая определяет, какой функционал предоставить пользователю. Приемщики оформляют заказы от клиентов путем заполнения всех необходимых данных в соответствующие поля, далее проводят заказ, которому, в свою очередь задается статус «проводится диагностика». Дальнейшая судьба заказа передается в руки мастеру, он производит диагностику устройства и по ее итогам указывает один из статусов: «диагностика проведена, ремонт невозможен», «диагностика проведена, ремонт возможен» или «диагностика проведена, не гарантийный случай». При заключении невозможности или не гарантийного случая устройство возвращается клиенту, иначе мастером производиться ремонт и записывается статус «ремонт проведен успешно». О чем узнает приемщик и связывается с клиентов для возвращения. При выдаче аппаратуры приемщиком, у заказа указывается статус «изделие выдано клиенту». На этом работа с данным заказом закончена.

Руководство пользователя для приемщика.

При запуске программы открывается окно авторизации, где нужно ввести личные логин и пароль. Окно авторизации представлено на рисунке 2.3.1.

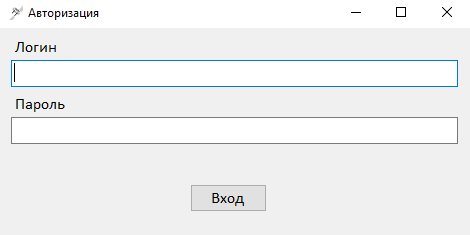


Рисунок 2.3.1 – Окно авторизации

После успешной авторизации для приемщика открывается главное окно, в котором по умолчанию открыта вкладка «Создать заказ», окно представлено на рисунке 2.3.2.

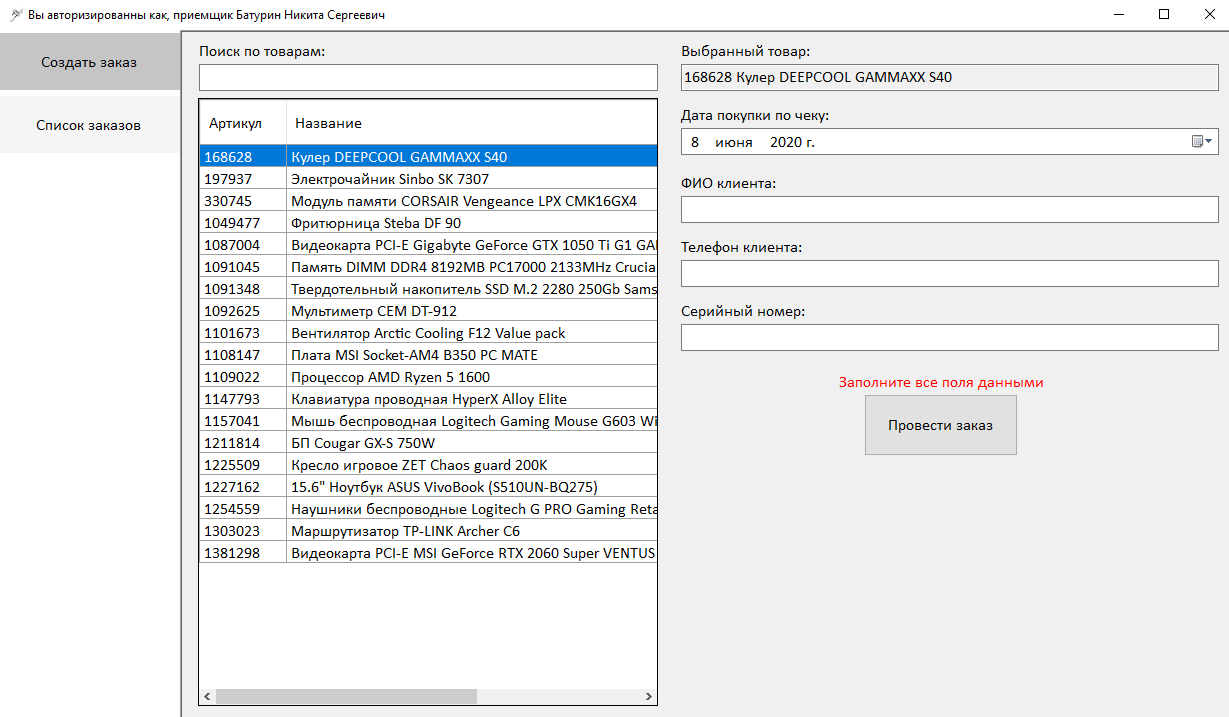


Рисунок 2.3.2 – Главное окно с открытой вкладкой «Создать заказ»

На этом окне заполняются данные по заявлению клиента о гарантийном обслуживании. Сначала, при помощи поисковой строки ищется товар в каталоге. Нужный товар выбирается и доступен для чтения в поле «Выбранный товар», далее с чека покупки заносится дата покупки, для выяснения продолжает ли свое действия гарантия. Если гарантия еще активна, приемщик вносит контактные данные клиента и серийный номер устройства.

При корректном вводе данных и нажатии на кнопку «Провести заказа», будет автоматически сформирован акт о приеме устройства на диагностику и ремонт. Акт представлен на рисунке 2.3.3.

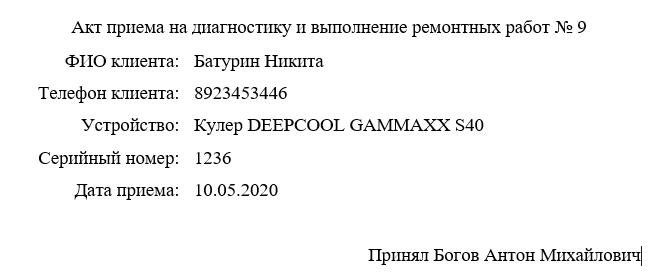


Рисунок 2.3.3 – Акт приема на диагностику и ремонт

После этого устройство и заказ переходят к мастеру, его часть программного решения будет рассмотрена далее, вне зависимости от результата диагностики и ремонта итог один – устройство возвращается к владельцу. Когда клиент возвращается для выдачи ему аппаратуры, также автоматически формируется акт выдачи. Акт выдачи представлен на рисунке 2.3.4.

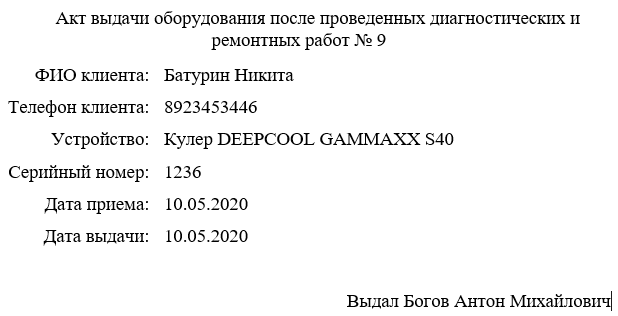


Рисунок 2.3.4 – Акт выдачи оборудования после проделанной работы

Руководство пользователя для мастера.

При запуске программы открывается окно авторизации, где нужно ввести личные логин и пароль. Окно авторизации представлено на рисунке 2.3.1.

После успешной авторизации для мастера открывается окно выбора заказа и возможностью изменить статус заказа. Окно представлено на рисунке 2.3.5.

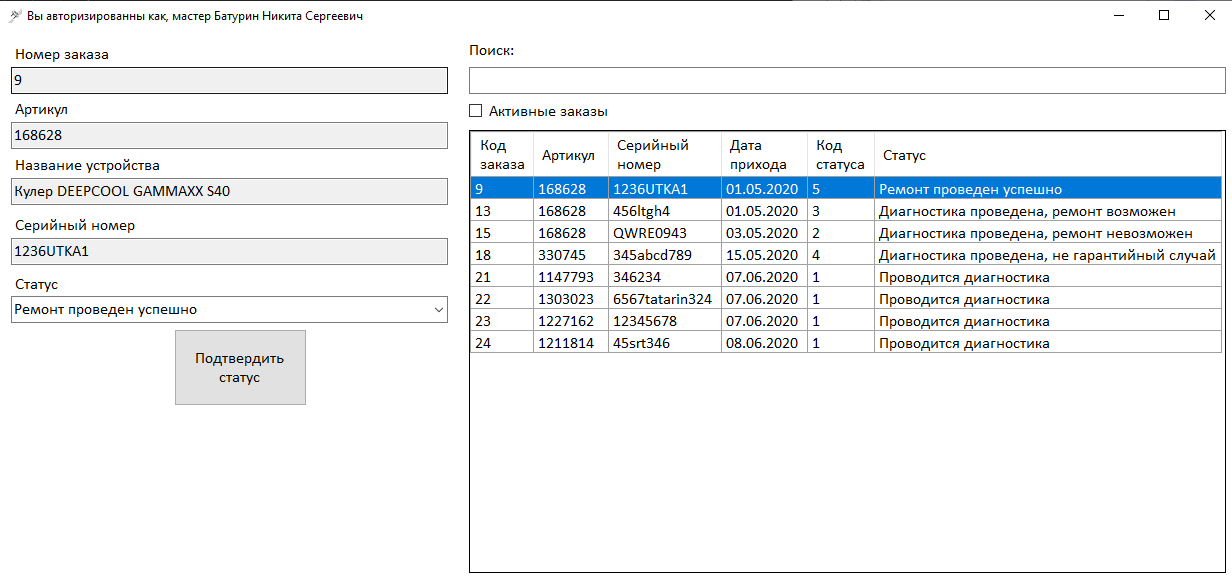


Рисунок 2.3.5 – Окно мастера

Работа мастера заключается в том, чтобы найти заказ в списке для этого есть поисковая строка, далее нужно реальности произвести диагностику, по результату которой, указывается соответствующий статус, далее если итог диагностики означает возможность ремонта, мастер ремонтирует устройство. По завершению ремонта, мастер указывает статус о завершенности ремонта.

Далее заказ переходит снова к приемщику для дальнейшего возвращения оборудования клиенту.

При необходимости приемщик может сформировать отчет из списка заказов. Окно списка заказов представлено на рисунке 2.3.6.

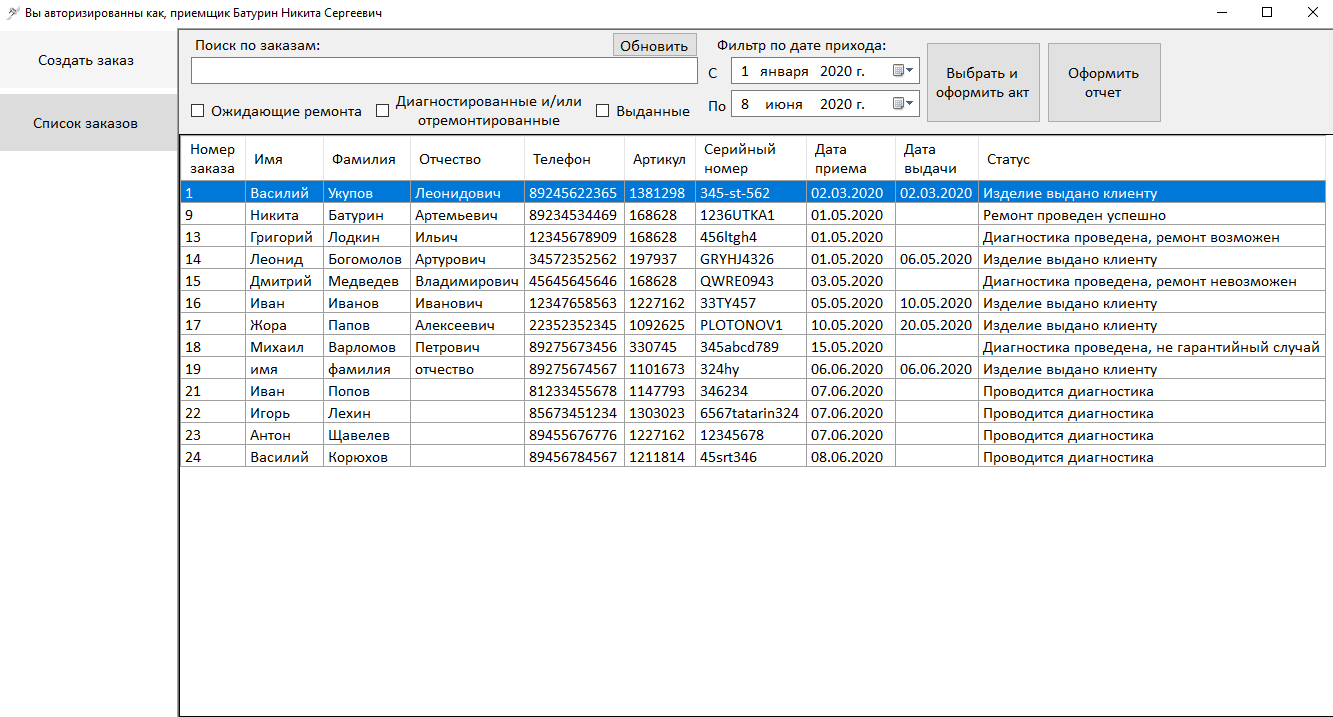


Рисунок 2.3.6 – Окно списка заказов

Сформированный отчет о выданных заказах за 01.05.2020 по 31.05.2020 представлен на рисунке 2.3.7.

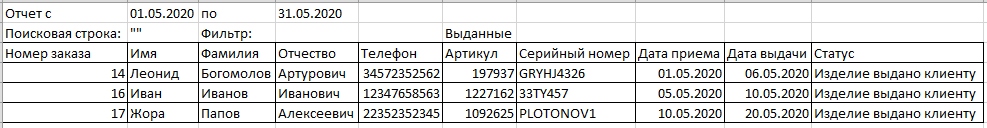


Рисунок 2.3.7 - Отчет о выданных заказах за 01.05.2020 по 31.05.2020

3 Экономический раздел

3.1 Расчёт затрат на создание программного продукта

Расчет себестоимости машинного часа эксплуатации вычислительной и оргтехники (ВиОТ):

, (3.1.1)

где См.ч.– себестоимость машинного часа;

Зi– годовые затраты, связанные с эксплуатацией и обслуживанием ВиОТ;

Fп– годовой полезный фонд времени работы единицы оборудования;

Kг– коэффициент готовности.

, (3.1.2)

где Fн= 1 954ч. – номинальный годовой фонд рабочего времени в РБ по 40-часовой рабочей недели в 2020 г.;

– коэффициент, учитывающий время, затраченное на ремонт, настройку, обслуживание ВиОТ.

(ч.)

Для расчета годовых затрат, необходимо определить балансовую стоимость условного комплекта, необходимого для создания программного продукта. Состав условного комплекта перечислен в таблице 3.1.1.

, (3.1.3)

где – цена единицы условного комплекта;

– количество единиц условного комплекта;

– дополнительные расходы на доставку, установку, первоначальную наладку.

, (3.1.4)

Таблица 3.1.1 – Состав условного комплекта

|  |  |  |  |
| --- | --- | --- | --- |
| № | Наименование | Кол-во, шт. | Цена, руб. |
| 1 | Ноутбук | 1 | 55 290 |

Продолжение таблицы 3.1.1

|  |  |  |  |
| --- | --- | --- | --- |
| № | Наименование | Кол-во, шт. | Цена, руб. |
| 2 | Монитор | 2 | 16 490 |
| 3 | Клавиатура | 1 | 790 |
| 4 | Мышь |  | 790 |
|  | Итого: | | 89 850 |

(руб.).

(руб.).

Затраты на материалы:

, (3.1.5)

(руб.)

Основная и дополнительная заработная плата персонала, обслуживающего ВиОТ:

, (3.1.6)

где – оклад i-го работника, обслуживающего ВиТО;

– коэффициент дополнительной заработной платы;

– районный коэффициент;

= 20.

(руб.)

Начисления на заработную плату , подробный список перечислен в таблице 3.1.2.

, (3.1.7)

(руб.).

Таблица 3.1.2 – Состав единого страхового налога на 2019г.

|  |  |  |
| --- | --- | --- |
| № | Статья | % |
| 1 | Пенсионный фонд | 22 |
| 2 | Фонд социального страхования | 2,9 |
| 3 | Федеральный фонд обязательного медицинского страхования | 5,1 |
| 4 | Страховой налог от несчастного случая на производстве для служащих | 0,2 |
|  | Итого: | 30,2 |

Амортизационные отчисления ВиОТ:

, (3.1.8)

где – норма амортизационных отчислений.

(руб.).

Износ программных продуктов.

Условный комплект необходимых программных продуктов представлен в таблице 3.1.3:

Таблица 3.1.3 – Условный комплект программных продуктов

|  |  |
| --- | --- |
| Наименование | Цена (руб.) |
| MS Windows 10 Pro | 20 230‬ |
| MS Office 2019 | 10 000 |
| MS Visual Studio 2019 Professional | 25 000 |
| Итого: | 55 230 |

Амортизационные отчисления программных продуктов:

, (3.1.9)

где – суммарная стоимость программных продуктов;

– норма амортизационных отчислений.

(руб.)

Расходы на содержание и эксплуатацию оборудования.

Расходы на содержание и эксплуатацию оборудования состоят из:

* затраты на ремонт и специальное обслуживание

, (3.1.10)

* затраты на электрическую энергию

, (3.1.11)

где – суммарная мощность;

– коэффициент готовности;

– стоимость кВт/ч.

(руб.).

(руб.).

Расходы на содержание и эксплуатацию оборудования:

, (3.1.12)

(руб.).

Прочие расходы.

Так как используется один тип техники и количество обслуживающего персонала не превышает трех человек, следовательно:

, (3.1.13)

(руб.).

Суммарные годовые затраты:

, (3.1.14)

(руб.).

Себестоимость машинного часа, из формулы (3.1.1), составляет:

(руб.).

3.2 Расчет цены предложения и минимального количества копий тиражирования

Фонд оплаты труда за время работы над программным продуктом – ФОТ:

, (3.2.1)

где – оклад сотрудника, работающего над продуктом;

– общее время работы над программным продуктом;

– коэффициент дополнительной заработной платы;

– районный коэффициент.

(руб.).

Начисления на ФОТ:

, (3.2.2)

(руб.).

Затраты, связанные с эксплуатацией и обслуживанием ВиОТ:

, (3.2.3)

где – машинное время работы над программным продуктом;

– число рабочих часов в месяце;

– число условных комплектов;

– коэффициент использования оборудования.

(руб.).

Затраты на специальные программные продукты – (руб.), представлены в таблице 3.2.1.

Таблица 3.2.1 – Специальные программные продукты

|  |  |
| --- | --- |
| Наименование | Цена |
| Visual Studio 2017 Professional | 25 000 |
| Итого: | 25 000 |

Затраты на хозяйственные операции и нужды – (руб.), представлены в таблице 3.2.2.

Таблица 3.2.2 – Затраты на хозяйственные нужды

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование | Цена за единицу, (руб.) | Кол-во | Всего |
| Бумага | 235 | 4 | 940 |
| Ручка | 25 | 2 | 50 |
| Блокнот | 360 | 1 | 360 |
| Карандаш | 25 | 3 | 75 |
| Итого: | | | 1 425 |

Накладные расходы:

, (3.2.4)

(руб.)

Полные затраты на разработку программного продукта:

, (3.2.5)

(руб.)

Расчет установочной прибыли:

, (3.2.6)

где - установочная рентабельность.

(руб.).

Расчет величины налога на добавленную стоимость (НДС):

, (3.2.7)

(руб.).

Цена предложения разрабатываемого программного продукта:

, (3.2.8)

(руб.).

Расчет минимального количества копий, необходимых для реализации, для получения установочной прибыли:

, (3.2.9)

где – цена на рынке одной копии программного продукта;

– затраты на тиражирование одной копии программного продукта.

, (3.2.10)

где – цена носителя программного продукта;

– время копирования.

(руб.).

шт.

ЗАКЛЮЧЕНИЕ

В результате выполнения дипломного проекта все поставленные цели и задачи были выполнены. Было разработано программное решения для использования его в бизнес среде, выполнено физическое проектирование базы данных, спроектированы экранные формы ввода-вывода и формирования актов.

Разработанное программное обеспечение располагают функциональным решением для выполнения таких задач, как:

– авторизация в системе;

– создание заказов на гарантийное обслуживание;

– формирование актов приема и выдачи;

– просмотр заказов;

– формирование отчета по списку заказов с заданными фильтрами;

– редактирование статуса заказа.

Результатом работы стал программный продукт для автоматизации учета гарантийного оборудования в сервисном центре. Программа разработана на языке программирования C# с использованием технологий Windows Forms и MDI контейнеры, также для работы используется система управления базами данных MySQL.

Приложение А.

Исходный код программы

Класс LoginForm.cs:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using service\_center.Classes;

using service\_center.Forms;

namespace service\_center

{

public partial class LoginForm : Form

{

//экземпляр для подключения к бд

ConnectionDB cn;

public LoginForm()

{

InitializeComponent();

}

private void LoginForm\_Load(object sender, EventArgs e)

{

LoginTextBox.Text = "";

PasswordTextBox.Text = "";

cn = new ConnectionDB();

}

private void LoginButton\_Click(object sender, EventArgs e)

{

//кнопка входа - авторизирует пользователя

string[] userData;

if (cn.CheckLogin(LoginTextBox.Text, PasswordTextBox.Text, out userData))

{

ExceptionLabel.Visible = false;

LoginTextBox.Text = "";

PasswordTextBox.Text = "";

this.Hide();

if (userData[3] == "r")//роль приемщика

{

ReceiverForm receiverForm = new ReceiverForm(userData[0], userData[1], userData[2]);

receiverForm.Show();

}

else if (userData[3] == "m")//роль мастера

Продолжение приложения А

{

MasterForm masterForm = new MasterForm(userData[0], userData[1], userData[2]);

masterForm.Show();

}

else if (userData[3] == "a")//роль админа

{

ReceiverForm receiverForm = new ReceiverForm(userData[0], userData[1], userData[2]);

receiverForm.Show();

MasterForm masterForm = new MasterForm(userData[0], userData[1], userData[2]);

masterForm.Show();

}

else

{

}

}

else

{

ExceptionLabel.Visible = true;

}

}

}

}

Класс ReceiverForm.cs:

using service\_center.Classes;

using service\_center.Forms.FormsForReceiver;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace service\_center.Forms

{

public partial class ReceiverForm : Form

{

//экземпляр для подключения к бд

ConnectionDB cn;

//экземпляры форм

CreateOrder createOrder;

OrderList orderList;

PrintForm printForm;

public ReceiverForm(string name, string surname, string patronymic)

{

InitializeComponent();

this.Text = $"Вы авторизированны как, приемщик {surname} {name} {patronymic}";

cn = new ConnectionDB();

Продолжение приложения А

//инициализация и настройка вложеных форм

createOrder = new CreateOrder();

createOrder.MdiParent = this;

createOrder.Dock = DockStyle.Fill;

createOrder.Text = $"{surname} {name} {patronymic}";

orderList = new OrderList();

orderList.MdiParent = this;

orderList.Dock = DockStyle.Fill;

orderList.Text = $"{surname} {name} {patronymic}";

printForm = new PrintForm("", "");

printForm.MdiParent = this;

printForm.Dock = DockStyle.Fill;

printForm.Text = $"{surname} {name} {patronymic}";

createOrder.Show();

}

private void ReceiverForm\_FormClosing(object sender, FormClosingEventArgs e)

{

//открытия форма авторизации при выходе

Application.OpenForms["LoginForm"].Show();

}

private void SideBarButton\_Click(object sender, EventArgs e)

{

//кнопки на боковой панели - переключают вложенные формы

var b = (Button)sender;

CreateOrderButton.BackColor = Color.WhiteSmoke;

OrderListButton.BackColor = Color.WhiteSmoke;

PrintFormButton.BackColor = Color.WhiteSmoke;

createOrder.Hide();

orderList.Hide();

printForm.Hide();

b.BackColor = Color.Gainsboro;

if ((string)b.Tag == "0")//форма создания заказов

{

createOrder.Show();

}

else if ((string)b.Tag == "1")//форма списка заказов

{

orderList.Show();

}

else if ((string)b.Tag == "2")//форма оформления актов

{

printForm.Show();

}

else

{

Продолжение приложения А

}

}

}

}

Класс MasterForm.cs:

using service\_center.Classes;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace service\_center.Forms

{

public partial class MasterForm : Form

{

//экземпляр для подключения к бд

ConnectionDB cn;

public MasterForm(string name, string surname, string patronymic)

{

InitializeComponent();

this.Text = $"Вы авторизированны как, мастер {surname} {name} {patronymic}";

cn = new ConnectionDB();

//заполнение и оформление списка заказов

OrderListDGV.DataSource = cn.DisplayOrdersForMaster("", false);

OrderListDGV.AutoResizeColumns(DataGridViewAutoSizeColumnsMode.ColumnHeader);

OrderListDGV.Columns[0].HeaderText = "Код заказа";

OrderListDGV.Columns[1].HeaderText = "Артикул";

OrderListDGV.Columns[2].HeaderText = "Серийный номер";

OrderListDGV.Columns[3].HeaderText = "Дата прихода";

OrderListDGV.Columns[4].HeaderText = "Код статуса";

OrderListDGV.Columns[5].HeaderText = "Статус";

StatusComboBox.Items.Clear();

StatusComboBox.Items.AddRange(cn.StatusListForMaster());

}

private void MasterForm\_FormClosing(object sender, FormClosingEventArgs e)

{

//открытия форма авторизации при выходе

Application.OpenForms["LoginForm"].Show();

}

private void OrderListDGV\_SelectionChanged(object sender, EventArgs e)

{

//изменение выбора заказа и вставка его данных

var selectedRows = OrderListDGV.SelectedRows;

Продолжение приложения А

StatusComboBox.Enabled = true;

if (selectedRows.Count > 0)

{

OrderIdTextBox.Text = selectedRows[0].Cells[0].Value.ToString();

ArticleTextBox.Text = selectedRows[0].Cells[1].Value.ToString();

TitleTextBox.Text = cn.TitleOrder(Convert.ToInt32(selectedRows[0].Cells[0].Value));

SerialNumberTextBox.Text = selectedRows[0].Cells[2].Value.ToString();

StatusComboBox.SelectedIndex = Convert.ToInt32(selectedRows[0].Cells[4].Value) - 1;

}

else

{

OrderIdTextBox.Text = "";

ArticleTextBox.Text = "";

TitleTextBox.Text = "";

SerialNumberTextBox.Text = "";

StatusComboBox.Enabled = false;

}

}

private void SearchAndFilter\_Changed(object sender, EventArgs e)

{

//обновление списка заказов по результату поиска

OrderListDGV.DataSource = cn.DisplayOrdersForMaster(SearchOrderTextBox.Text, ActiveOrdersCheckBox.Checked);

}

private void ConfirmStatusButton\_Click(object sender, EventArgs e)

{

//кнопка изменения статуса

if (OrderIdTextBox.Text != "")

{

int orderId = Convert.ToInt32(OrderIdTextBox.Text),

statusId = StatusComboBox.SelectedIndex + 1,

statusIdOld = Convert.ToInt32(OrderListDGV.SelectedRows[0].Cells[4].Value);

if (statusId != statusIdOld)//обновляет статус если он не такой же

{

cn.UpdateStatus(orderId, statusId);

OrderListDGV.DataSource = cn.DisplayOrdersForMaster(SearchOrderTextBox.Text, ActiveOrdersCheckBox.Checked);

MessageBox.Show("Статус успешно изменен.", "Изменение статуса");

}

else

{

MessageBox.Show("Нельзя изменить статус на такой же.", "Изменение статуса");

}

}

else

{

MessageBox.Show("Выберите заказ, чтобы изменить статус.", "Изменение статуса");

}

}

}

}

Продолжение приложения А

Класс CreateOrder.cs:

using service\_center.Classes;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace service\_center.Forms.FormsForReceiver

{

public partial class CreateOrder : Form

{

//экземпляр для подключения к бд

ConnectionDB cn;

public CreateOrder()

{

InitializeComponent();

cn = new ConnectionDB();

//заполнение и оформление каталога товаров

CatalogDGV.DataSource = cn.DisplayCatalog(SearchTextBox.Text);

CatalogDGV.AutoResizeColumns(DataGridViewAutoSizeColumnsMode.DisplayedCells);

CatalogDGV.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.AllCells;

CatalogDGV.ScrollBars = ScrollBars.Both;

CatalogDGV.AllowUserToResizeColumns = true;

CatalogDGV.Columns[0].HeaderText = "Артикул";

CatalogDGV.Columns[1].HeaderText = "Название";

CatalogDGV.Columns[2].HeaderText = "Период гарантии в месяцах";

}

private void SearchTextBox\_TextChanged(object sender, EventArgs e)

{

//обновление каталога по результату поиска

CatalogDGV.DataSource = cn.DisplayCatalog(SearchTextBox.Text);

}

private void CatalogDGV\_SelectionChanged(object sender, EventArgs e)

{

//изменение выбора товара и вставка его данных

var selectedRows = CatalogDGV.SelectedRows;

if (selectedRows.Count > 0)

{

ProductInfoTextBox.Text = selectedRows[0].Cells[0].Value + " " + selectedRows[0].Cells[1].Value;

CheckFillFields();

}

else

{

ProductInfoTextBox.Text = "";

Продолжение приложения А

CheckFillFields();

}

}

private void InfoBoxes\_TextChanged(object sender, EventArgs e)

{

//изменение полей данных о заказе

CheckFillFields();

}

/// <summary>

/// Проверка заполненности и корректности полей данных.

/// </summary>

/// <returns>Возвращает true если валидация прошла успешно, иначе false.</returns>

private bool CheckFillFields()

{

//проверка заполненности и корректности полей данных

ExceptionNameLabel.Visible = false;

ExceptionPhoneLabel.Visible = false;

ExceptionSerialLabel.Visible = false;

ExceptionEmptyLabel.Visible = false;

ExceptionWarrantyLable.Visible = false;

//проверка полей: товар, ФИО, телефон, серийный номер

if (ProductInfoTextBox.Text == "" || NameClientTextBox.Text == "" || PhoneClientTextBox.Text == "" || SerialNumberTextBox.Text == "")

{

ExceptionEmptyLabel.Visible = true;

return false;

}

//проверка продолжительности гарантии

int warrantyPeriod = Convert.ToInt32(CatalogDGV.SelectedRows[0].Cells[2].Value.ToString());

if (warrantyPeriod / 12 \* 365 < DateTime.Today.Subtract(DateCheckDTP.Value).Days)

{

ExceptionWarrantyLable.Visible = true;

return false;

}

//вторичная валидация поля ФИО

if (!NameClientTextBox.Text.Contains(" "))

{

ExceptionNameLabel.Visible = true;

return false;

}

//вторичная валидация поля телефон

if (PhoneClientTextBox.Text.Length < 10)

{

ExceptionPhoneLabel.Visible = true;

return false;

}

//вторичная валидация поля серийный номер

int article = Convert.ToInt32(CatalogDGV.SelectedRows[0].Cells[0].Value);

if (cn.CompareDublicatesSerialNumbers(SerialNumberTextBox.Text, article))

{

ExceptionSerialLabel.Visible = true;

return false;

Продолжение приложения А

}

//все правильно

return true;

}

private void SpendButton\_Click(object sender, EventArgs e)

{

//кнопка проведения заказа

//проверка заполненности и корректности полей данных

if (!CheckFillFields())

{

MessageBox.Show("Чтобы провести заказ нужно корректно заполнить поля.", "Оформление заказа");

return;

}

//заполнение данных

int article = Convert.ToInt32(CatalogDGV.SelectedRows[0].Cells[0].Value);

string name = NameClientTextBox.Text.Split(' ')[1],

surname = NameClientTextBox.Text.Split(' ')[0],

patronymic = "",

phone = PhoneClientTextBox.Text;

if (NameClientTextBox.Text.Split(' ').Length > 2)

{

patronymic = NameClientTextBox.Text.Split(' ')[2];

}

//вставка новых данных

int clientId = cn.InsertingClients(name, surname, patronymic, phone),

orderId = cn.InsertingOrder(DateCheckDTP.Value, article, clientId, SerialNumberTextBox.Text);

if (MessageBox.Show("Заказ успешно проведен.\n\nОткрыть окно оформления актов?", "Оформление заказа", MessageBoxButtons.YesNo) == DialogResult.Yes)

{

//открытие формы оформления актов

PrintForm printForm = new PrintForm(this.Text, orderId.ToString());

printForm.MdiParent = this.MdiParent;

printForm.Dock = DockStyle.Fill;

printForm.Show();

}

NameClientTextBox.Text = "";

PhoneClientTextBox.Text = "";

SerialNumberTextBox.Text = "";

DateCheckDTP.Value = DateTime.Today;

}

}

}

Класс OrderList.cs:

using service\_center.Classes;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

Продолжение приложения А

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using ExcelApp = Microsoft.Office.Interop.Excel;

namespace service\_center.Forms.FormsForReceiver

{

public partial class OrderList : Form

{

//экземпляр для подключения к бд

ConnectionDB cn;

public OrderList()

{

InitializeComponent();

cn = new ConnectionDB();

DateInEndDTP.Value = DateTime.Today;

//заполнение и оформление списка заказов

OrderListDGV.DataSource = cn.DisplayOrders("", false, false, false, DateInStartDTP.Value, DateInEndDTP.Value);

OrderListDGV.Columns[0].HeaderText = "Номер заказа";

OrderListDGV.Columns[1].HeaderText = "Имя";

OrderListDGV.Columns[2].HeaderText = "Фамилия";

OrderListDGV.Columns[3].HeaderText = "Отчество";

OrderListDGV.Columns[4].HeaderText = "Телефон";

OrderListDGV.Columns[5].HeaderText = "Артикул";

OrderListDGV.Columns[6].HeaderText = "Серийный номер";

OrderListDGV.Columns[7].HeaderText = "Дата приема";

OrderListDGV.Columns[8].HeaderText = "Дата выдачи";

OrderListDGV.Columns[9].HeaderText = "Статус";

}

private void SearchAndFilters\_Changed(object sender, EventArgs e)

{

//изменение в поисковой строке или переключения фильтров - обновляет список заказов

OrderListDGV.DataSource = cn.DisplayOrders(SearchOrderTextBox.Text, DiagnostedCheckBox.Checked, IssuedCheckBox.Checked,

WaitRepairCheckBox.Checked, DateInStartDTP.Value, DateInEndDTP.Value);

}

private void PrintActButton\_Click(object sender, EventArgs e)

{

//кнопка открывает окно оформления актов

if (OrderListDGV.SelectedRows.Count > 0)

{

int orderId = Convert.ToInt32(OrderListDGV.SelectedRows[0].Cells[0].Value);

PrintForm printForm = new PrintForm(this.Text, orderId.ToString());

printForm.MdiParent = this.MdiParent;

printForm.AccessibleName = orderId.ToString();

printForm.Dock = DockStyle.Fill;

printForm.Text = this.Text;

printForm.Show();

Продолжение приложения А

}

else

{

MessageBox.Show("Выберите заказ.", "Оформление актов");

}

}

private void PrintReportButton\_Click(object sender, EventArgs e)

{

//кнопка оформление отчета

if (saveFileDialog.ShowDialog() == DialogResult.OK)

{

DataTable dt = (DataTable)OrderListDGV.DataSource;

var excelapp = new ExcelApp.Application();

excelapp.Visible = false;

try

{

//отчет будет сохранен как Excel файл

excelapp.SheetsInNewWorkbook = 1;

ExcelApp.Workbook workbook = excelapp.Workbooks.Add(Type.Missing);

excelapp.DisplayAlerts = false;

ExcelApp.Worksheet sheet = (ExcelApp.Worksheet)excelapp.Worksheets[1];

sheet.Name = $"Отчет";

//заполнение вводной информацией

sheet.Cells[1, 1] = $"Отчет с";

sheet.Cells[1, 2] = DateInStartDTP.Value.ToShortDateString();

sheet.Cells[1, 3] = "по";

sheet.Cells[1, 4] = DateInEndDTP.Value.ToShortDateString();

sheet.Cells[2, 1] = "Поисковая строка:";

sheet.Cells[2, 2] = $"\"{SearchOrderTextBox.Text}\"";

sheet.Cells[2, 3] = "Фильтр:";

sheet.Cells[2, 4] = WaitRepairCheckBox.Checked ? WaitRepairCheckBox.Text : "";

sheet.Cells[2, 5] = DiagnostedCheckBox.Checked ? "Диагностированные" : "";

sheet.Cells[2, 6] = IssuedCheckBox.Checked ? IssuedCheckBox.Text : "";

sheet.Cells[3, 1] = "Номер заказа";

sheet.Cells[3, 2] = "Имя";

sheet.Cells[3, 3] = "Фамилия";

sheet.Cells[3, 4] = "Отчество";

sheet.Cells[3, 5] = "Телефон";

sheet.Cells[3, 6] = "Артикул";

sheet.Cells[3, 7] = "Серийный номер";

sheet.Cells[3, 8] = "Дата приема";

sheet.Cells[3, 9] = "Дата выдачи";

sheet.Cells[3, 10] = "Статус";

//заполнение данными

for(int i = 0; i < dt.Rows.Count; i++)

{

for(int j = 0; j < dt.Columns.Count; j++)

{

sheet.Cells[i + 4, j + 1] = dt.Rows[i][j];

}

}

//марафет

Продолжение приложения А

sheet.Range[sheet.Cells[1,1], sheet.Cells[1, 10]].EntireColumn.AutoFit();

var DataRange = (ExcelApp.Range)sheet.Range[sheet.Cells[3, 1], sheet.Cells[dt.Rows.Count + 3, 10]];

DataRange.Borders.get\_Item(ExcelApp.XlBordersIndex.xlEdgeBottom).LineStyle = ExcelApp.XlLineStyle.xlContinuous;

DataRange.Borders.get\_Item(ExcelApp.XlBordersIndex.xlEdgeRight).LineStyle = ExcelApp.XlLineStyle.xlContinuous;

DataRange.Borders.get\_Item(ExcelApp.XlBordersIndex.xlInsideHorizontal).LineStyle = ExcelApp.XlLineStyle.xlContinuous;

DataRange.Borders.get\_Item(ExcelApp.XlBordersIndex.xlInsideVertical).LineStyle = ExcelApp.XlLineStyle.xlContinuous;

DataRange.Borders.get\_Item(ExcelApp.XlBordersIndex.xlEdgeTop).LineStyle = ExcelApp.XlLineStyle.xlContinuous;

workbook.SaveAs(saveFileDialog.FileName);

workbook.Close();

MessageBox.Show("Отчет успешно создан.", "Формирование отчета");

}

catch (Exception ex)

{

MessageBox.Show("Отчет не удалось сохранить.", "Ошибка сохранения");

}

finally

{

excelapp.Quit();

}

}

}

}

}

Класс PrintForm.cs:

using service\_center.Classes;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using WordApp = Microsoft.Office.Interop.Word;

namespace service\_center.Forms.FormsForReceiver

{

public partial class PrintForm : Form

{

//экземпляр для подключения к бд

ConnectionDB cn;

//данные о имени приемщика и номер заказа

string nameReceiver = "";

string orderIdString = "";

Продолжение приложения А

public PrintForm(string nameReceiver, string orderIdString)

{

InitializeComponent();

cn = new ConnectionDB();

//заполнение виртуальных форм

if (orderIdString != "")

{

int orderId = Convert.ToInt32(orderIdString);

string[] orderData = cn.ActData(orderId);

ActReceiveRTB.Text = $"Акт приема на диагностику и выполнение ремонтных работ № {orderId}\n" +

$"ФИО клиента: {orderData[0]}\n" +

$"Телефон клиента: {orderData[1]}\n" +

$"Устройство: {orderData[2]}\n" +

$"Серийный номер: {orderData[3]}\n" +

$"Дата приема: {orderData[4]}\n" +

"\n" +

$"Принял {nameReceiver}";

ActIssueRTB.Text = $"Акт выдачи оборудования после проведенных диагностических и ремонтных работ № {orderId}\n" +

$"ФИО клиента: {orderData[0]}\n" +

$"Телефон клиента: {orderData[1]}\n" +

$"Устройство: {orderData[2]}\n" +

$"Серийный номер: {orderData[3]}\n" +

$"Дата приема: {orderData[4]}\n" +

$"Дата выдачи: {(orderData?[5] != null && orderData?[5] == "" ? DateTime.Today.ToString("dd.MM.yyyy") + " (Сегодня)" : orderData?[5])}\n" +

"\n" +

$"Выдал {nameReceiver}";

}

else

{

ActReceiveRTB.Text = "Заказ не выбран, перейдите во вкладку \"Список заков\" и выберите заказ.";

ActReceiveButton.Enabled = false;

ActIssueButton.Enabled = false;

}

this.nameReceiver = nameReceiver;

this.orderIdString = orderIdString;

}

/// <summary>

/// Метод для замены данных в документе.

/// </summary>

/// <param name="stub">Строка, которую заменит заданный текст.</param>

/// <param name="text">Строка, на которую замениться заданый шаблон.</param>

/// <param name="worddoc">Документ, в котором производиться замена.</param>

private void Replace(string stub, string text, WordApp.Document worddoc)

{

//метод для замены данных в документе

Продолжение приложения А

var range = worddoc.Content;

range.Find.ClearFormatting();

range.Find.Execute(FindText: stub, ReplaceWith: text);

}

private void ActReceiveButton\_Click(object sender, EventArgs e)

{

//формирование акта приема

if (orderIdString != "")

{

if (saveFileDialog.ShowDialog() == DialogResult.OK)

{

int orderId = Convert.ToInt32(orderIdString);

string[] orderData = cn.ActData(orderId);

var wordapp = new WordApp.Application();

wordapp.Visible = false;

try

{

var worddoc = wordapp.Documents.Open($@"{Application.StartupPath}\sample1.docx");

Replace("\_OrderId\_", orderId.ToString(), worddoc);

Replace("\_NameClient\_", orderData[0], worddoc);

Replace("\_Phone\_", orderData[1], worddoc);

Replace("\_Title\_", orderData[2], worddoc);

Replace("\_Serial\_", orderData[3], worddoc);

Replace("\_DateIn\_", orderData[4], worddoc);

Replace("\_NameReceiver\_", nameReceiver, worddoc);

worddoc.SaveAs(saveFileDialog.FileName);

worddoc.Close();

MessageBox.Show("Акт приема успешно создан.", "Печать актов");

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

finally

{

wordapp.Quit();

}

}

}

else

{

MessageBox.Show("Не выбран заказ.", "Печать актов");

}

}

private void ActIssueButton\_Click(object sender, EventArgs e)

{

//формирование акта выдачи

if (orderIdString != "")

{

Продолжение приложения А

if (saveFileDialog.ShowDialog() == DialogResult.OK)

{

int orderId = Convert.ToInt32(orderIdString);

//обновление данных о заказе

string[] orderData = cn.ActData(orderId);

if(orderData[5] == null || orderData[5] == "")

{

cn.UpdateStatus(orderId, 6);

cn.UpdateOrderSetDateOut(orderId);

orderData = cn.ActData(orderId);

}

var wordapp = new WordApp.Application();

wordapp.Visible = false;

try

{

var worddoc = wordapp.Documents.Open($@"{Application.StartupPath}\sample2.docx");

Replace("\_OrderId\_", orderId.ToString(), worddoc);

Replace("\_NameClient\_", orderData[0], worddoc);

Replace("\_Phone\_", orderData[1], worddoc);

Replace("\_Title\_", orderData[2], worddoc);

Replace("\_Serial\_", orderData[3], worddoc);

Replace("\_DateIn\_", orderData[4], worddoc);

Replace("\_DateOut\_", orderData?[5], worddoc);

Replace("\_NameReceiver\_", nameReceiver, worddoc);

worddoc.SaveAs(saveFileDialog.FileName);

worddoc.Close();

MessageBox.Show("Акт выдачи успешно создан.", "Печать актов");

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

finally

{

wordapp.Quit();

}

}

}

else

{

MessageBox.Show("Не выбран заказ.", "Печать актов");

}

}

private void BackToFormButton\_Click(object sender, EventArgs e)

{

//возвращается назад по форме

this.Hide();

}

}

}

Продолжение приложения А

Класс ConnectionDB.cs:

using System;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using MySql.Data.MySqlClient;

namespace service\_center.Classes

{

/// <summary>

/// Класс для подключения к базе данных и реализацией всех запросов.

/// </summary>

class ConnectionDB

{

//экземпляр для подключения к бд

private MySqlConnection cn;

public ConnectionDB()

{

//инициализация подключения к бд и тестирование подключения

cn = new MySqlConnection("port=3306;server=localhost;userid=root;password=123654;database=service\_center");

try

{

cn.Open();

}

catch(Exception ex)

{

MessageBox.Show("Не удается подключиться к удаленному серверу.", "Ошибка подключения");

}

finally

{

cn.Close();

}

}

/// <summary>

/// Метод для проверки данных для авторизации.

/// </summary>

/// <param name="login">Логин</param>

/// <param name="password">Пароль</param>

/// <param name="userData">Выходной параметр, возвращает данные пользователя</param>

/// <returns>Возвращает булевое значение успешности авторизации</returns>

public bool CheckLogin(string login, string password, out string[] userData)

{

bool statusLogin = false;

userData = new[] { "", "", "", "" };

try

Продолжение приложения А

{

cn.Open();

MySqlCommand checkQuery = new MySqlCommand($"select \* from users where login='{login}' and password='{password}';", cn);

MySqlDataReader userReader = checkQuery.ExecuteReader();

statusLogin = userReader.HasRows;

if (statusLogin)

{

userReader.Read();

userData = new[] { userReader.GetString("name"), userReader.GetString("surname"), userReader.GetString("patronymic"), userReader.GetString("role") };

}

}

catch (Exception ex)

{

MessageBox.Show("Не удается подключиться к удаленному серверу.", "Ошибка авторизации");

}

finally

{

cn.Close();

}

return statusLogin;

}

/// <summary>

/// Метод отображет каталог.

/// </summary>

/// <param name="searchString">Поисковая строка</param>

/// <returns>Возвращает таблицу по результатам поисковой строки в каталоге</returns>

public DataTable DisplayCatalog(string searchString)

{

DataTable dt = new DataTable();

try

{

cn.Open();

string query = $"select \* from catalog where article like '%{searchString}%' or title like '%{searchString}%'";

MySqlDataAdapter mySqlDataAdapter = new MySqlDataAdapter(query, cn);

mySqlDataAdapter.Fill(dt);

}

catch (Exception ex)

{

MessageBox.Show("Не удается подключиться к удаленному серверу.", "Ошибка отображения каталога");

}

finally

{

cn.Close();

}

return dt;

Продолжение приложения А

}

/// <summary>

/// Метод проверяет отсутствие дубликатов серийного номера одного и того же товара.

/// </summary>

/// <param name="serial">Искомый серийный номер</param>

/// <param name="article">Артикул товара</param>

/// <returns>Возвращает true если дубликат существует, иначе false</returns>

public bool CompareDublicatesSerialNumbers(string serial, int article)

{

bool IsDublicate = false;

try

{

cn.Open();

MySqlCommand mySqlCommand = new MySqlCommand($"select \* from orders where serial\_number = '{serial}' and article = '{article}'", cn);

MySqlDataReader mySQLDataReader = mySqlCommand.ExecuteReader();

IsDublicate = mySQLDataReader.HasRows;

}

catch (Exception ex)

{

MessageBox.Show("Не удается подключиться к удаленному серверу.", "Ошибка серийных номеров");

}

finally

{

cn.Close();

}

return IsDublicate;

}

/// <summary>

/// Метод вставляет запись клиента с указанными данными.

/// </summary>

/// <param name="name">Имя клиента</param>

/// <param name="surname">Фамилия клиента</param>

/// <param name="patronymic">Отчество клиента</param>

/// <param name="phone">Телефон клиента</param>

/// <returns>Возвращает код вставленной записи</returns>

public int InsertingClients(string name, string surname, string patronymic, string phone)

{

int clientCode = -1;

try

{

cn.Open();

MySqlCommand mySqlCommand = new MySqlCommand($"insert into clients values (null, '{name}', '{surname}','{patronymic}','{phone}')", cn);

mySqlCommand.ExecuteNonQuery();

//получет код

mySqlCommand = new MySqlCommand("select max(client\_id) from clients", cn);

MySqlDataReader mySQLDataReader = mySqlCommand.ExecuteReader();

mySQLDataReader.Read();

Продолжение приложения А

clientCode = mySQLDataReader.GetInt32(0);

}

catch (Exception ex)

{

MessageBox.Show("Не удается подключиться к удаленному серверу.", "Ошибка вставки клиентов");

}

finally

{

cn.Close();

}

return clientCode;

}

/// <summary>

/// Метод вставляет новый заказ.

/// </summary>

/// <param name="checkDate">Дата чека</param>

/// <param name="article">Артикул товара</param>

/// <param name="clientId">Код клиента</param>

/// <param name="serial">Серийный номер</param>

/// <returns>Возвращает код заказа</returns>

public int InsertingOrder(DateTime checkDate, int article, int clientId, string serial)

{

string dateIn = DateTime.Today.ToString("yyyy-MM-dd"),

dateCheck = checkDate.ToString("yyyy-MM-dd");

int orderCode = -1;

try

{

cn.Open();

MySqlCommand mySqlCommand = new MySqlCommand($"insert into orders values (null, '{dateCheck}', '{article}','{clientId}','{serial}', '{dateIn}', null, 1)", cn);

mySqlCommand.ExecuteNonQuery();

//получет код

mySqlCommand = new MySqlCommand("select max(order\_id) from orders", cn);

MySqlDataReader mySQLDataReader = mySqlCommand.ExecuteReader();

mySQLDataReader.Read();

orderCode = mySQLDataReader.GetInt32(0);

}

catch (Exception ex)

{

MessageBox.Show("Не удается подключиться к удаленному серверу.", "Ошибка вставки заказа");

}

finally

{

cn.Close();

}

return orderCode;

}

/// <summary>

/// Метод отображает список заказов.

Продолжение приложения А

/// </summary>

/// <param name="searchString">Поисковая строка</param>

/// <param name="IsDiagnosted">Фильтр диагностированного</param>

/// <param name="IsIssued">Фильтр выданного</param>

/// <param name="IsWaitRepair">Фильтр ждущего ремонта</param>

/// <param name="DateInStart">Фильтр даты начала периода</param>

/// <param name="DateInEnd">Фильтр даты конца периода</param>

/// <returns>Возвращает результирующию таблицу по поисковой строке и фильтрам</returns>

public DataTable DisplayOrders(string searchString, bool IsDiagnosted, bool IsIssued, bool IsWaitRepair, DateTime DateInStart, DateTime DateInEnd)

{

DataTable dt = new DataTable();

try

{

cn.Open();

//компоновка фильтров

string StatusFilter = "";

if (IsDiagnosted)//диагностированного

{

StatusFilter = "2,4,5";

}

if (IsIssued)//выданного

{

StatusFilter += StatusFilter == "" ? "6" : ",6";

}

if (IsWaitRepair)//ждущего ремонта

{

StatusFilter += StatusFilter == "" ? "1,3" : ",1,3";

}

if (IsDiagnosted || IsIssued || IsWaitRepair)

{

string temporaryStatusFilter = StatusFilter;

StatusFilter = $" and orders.status\_id in ({StatusFilter})";

}

string query = $"select order\_id, clients.name, surname, patronymic, phone, article, serial\_number, date\_in, date\_out, statuses.name " +

$"from orders, clients, statuses where orders.client\_id=clients.client\_id and orders.status\_id=statuses.status\_id " +

$"and ((order\_id like '%{searchString}%' or article like '%{searchString}%' or serial\_number like '%{searchString}%'){StatusFilter}) " +

$"and date\_in >= '{DateInStart.ToString("yyyy-MM-dd")}' and date\_in <= '{DateInEnd.ToString("yyyy-MM-dd")}' " +

$"order by order\_id";

MySqlDataAdapter mySqlDataAdapter = new MySqlDataAdapter(query, cn);

mySqlDataAdapter.Fill(dt);

}

catch (Exception ex)

{

MessageBox.Show("Не удается подключиться к удаленному серверу." , "Ошибка отображения заказов");

}

finally

{

Продолжение приложения А

cn.Close();

}

return dt;

}

/// <summary>

/// Метод для получения данных для форирования актов.

/// </summary>

/// <param name="orderId">Код заказа</param>

/// <returns>Возвращает набор данных о заказе</returns>

public string[] ActData(int orderId)

{

string[] data = new[] { "", "", "", "", "", "" };

try

{

cn.Open();

string query = $"SELECT name, surname, patronymic, phone, title, serial\_number, date\_in, date\_out FROM orders, catalog, clients " +

$"WHERE orders.article = catalog.article AND orders.client\_id = clients.client\_id AND orders.order\_id = {orderId}";

MySqlCommand mySqlCommand = new MySqlCommand(query, cn);

MySqlDataReader mySQLDataReader = mySqlCommand.ExecuteReader();

mySQLDataReader.Read();

//заполнение набора данных

data[0] = mySQLDataReader.GetString(1) + " " + mySQLDataReader.GetString(0) + " " + mySQLDataReader.GetString(2); // full name

data[1] = mySQLDataReader.GetString(3); // phone

data[2] = mySQLDataReader.GetString(4); // title

data[3] = mySQLDataReader.GetString(5); // serial\_number

data[4] = mySQLDataReader.GetDateTime(6).ToShortDateString(); // date\_in

if (mySQLDataReader.GetValue(7) != null && mySQLDataReader.GetValue(7).ToString() != "")

data[5] = mySQLDataReader.GetDateTime(7).ToShortDateString(); // date\_out

else

data[5] = "";

}

catch (Exception ex)

{

MessageBox.Show("Не удается подключиться к удаленному серверу.", "Ошибка отчета заказа");

}

finally

{

cn.Close();

}

return data;

}

/// <summary>

/// Метод получает название товара.

/// </summary>

/// <param name="orderId">Код заказа</param>

/// <returns>Возвращает строку названия товара</returns>

Продолжение приложения А

public string TitleOrder(int orderId)

{

string title = "";

try

{

cn.Open();

string query = $"SELECT title FROM orders, catalog WHERE orders.article = catalog.article AND orders.order\_id = '{orderId}'";

MySqlCommand mySqlCommand = new MySqlCommand(query, cn);

MySqlDataReader mySQLDataReader = mySqlCommand.ExecuteReader();

mySQLDataReader.Read();

title = mySQLDataReader.GetString(0);

}

catch (Exception ex)

{

MessageBox.Show("Не удается подключиться к удаленному серверу.", "Ошибка получения названия устройства");

}

finally

{

cn.Close();

}

return title;

}

/// <summary>

/// метод для получения списка статусов для мастера.

/// </summary>

/// <returns>Возвращает массив из статусов</returns>

public string[] StatusListForMaster()

{

string[] list = new[] { "", "", "", "", "" };

try

{

cn.Open();

string query = $"SELECT \* FROM statuses where status\_id <> 6";

MySqlCommand mySqlCommand = new MySqlCommand(query, cn);

MySqlDataReader mySQLDataReader = mySqlCommand.ExecuteReader();

int i = 0;

while (mySQLDataReader.Read())

{

list[i] = mySQLDataReader.GetString(1);

i++;

}

mySQLDataReader.Close();

}

catch (Exception ex)

{

MessageBox.Show("Не удается подключиться к удаленному серверу.", "Ошибка списка статусов");

}

finally

Продолжение приложения А

{

cn.Close();

}

return list;

}

/// <summary>

/// Метод для обновления статуса у заказа.

/// </summary>

/// <param name="orderId">Код заказа</param>

/// <param name="statusId">Код нового статуса</param>

public void UpdateStatus(int orderId, int statusId)

{

try

{

cn.Open();

string query = $"update orders set status\_id = '{statusId}' where order\_id = '{orderId}'";

MySqlCommand mySqlCommand = new MySqlCommand(query, cn);

mySqlCommand.ExecuteNonQuery();

}

catch (Exception ex)

{

MessageBox.Show("Не удается подключиться к удаленному серверу.", "Ошибка обновления статуса");

}

finally

{

cn.Close();

}

}

/// <summary>

/// Метод для отображения заказов для мастера.

/// </summary>

/// <param name="searchString">Поисковая строка</param>

/// <param name="IsActiveOrders">Фильтр активных заказов</param>

/// <returns>Возвращает результирующую таблицу по поисковой строке и фильтра</returns>

public DataTable DisplayOrdersForMaster(string searchString, bool IsActiveOrders)

{

DataTable dt = new DataTable();

try

{

cn.Open();

string StatusFilter = "";

if (IsActiveOrders)//Фильтр активных заказов

{

StatusFilter = " and orders.status\_id in (1,3)";

}

string query = $"select order\_id, article, serial\_number, date\_in, orders.status\_id, statuses.name from orders, statuses " +

$"where orders.status\_id = statuses.status\_id and orders.status\_id <> 6{StatusFilter} and" +

Продолжение приложения А

$"(order\_id like '%{searchString}%' or article like '%{searchString}%' or serial\_number like '%{searchString}%') " +

$"order by order\_id";

MySqlDataAdapter mySqlDataAdapter = new MySqlDataAdapter(query, cn);

mySqlDataAdapter.Fill(dt);

}

catch (Exception ex)

{

MessageBox.Show("Не удается подключиться к удаленному серверу.\n" + ex, "Ошибка отображения заказов");

}

finally

{

cn.Close();

}

return dt;

}

/// <summary>

/// Метод задает дату выдачи заказа.

/// </summary>

/// <param name="OrderId">Код заказа</param>

public void UpdateOrderSetDateOut(int OrderId)

{

string dateOut = DateTime.Today.ToString("yyyy-MM-dd");

try

{

cn.Open();

string query = $"update orders set date\_out = '{dateOut}' where order\_id = '{OrderId}'";

MySqlCommand mySqlCommand = new MySqlCommand(query, cn);

mySqlCommand.ExecuteNonQuery();

}

catch (Exception ex)

{

MessageBox.Show("Не удается подключиться к удаленному серверу.", "Ошибка вставки заказа");

}

finally

{

cn.Close();

}

}

}

}

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Кормен, Т. П. Алгоритмы: построение и анализ [Текст] / Т. Кормен. — М.: МЦНМО, 2013. — 969 с.
2. Скиена, С. Алгоритмы. Руководство по разработке [Текст] / С. Скиена. — 2-е изд.— СПб.: БХВ-Петербург, 2013. — 720 с.
3. Система стандартов безопасности труда [Текст] / Под ред. В. А. Фомичева. – М.: Изд-во стандартов, 2016. – 102 с.
4. Мартин, Р. К. Чистый код: создание, анализ и рефакторинг. Библиотека программиста [Текст] / Р. Мартин. — СПб.: Питер, 2015. — 464 с.
5. Фаулер, М. Рефакторинг: улучшение существующего кода [Текст] / М. Фаулер. – СПб.: Символ-Плюс, 2013. – 432 с.
6. Глушаков, С. В. База данных [Текст] / С. В. Глушаков. – Харьков: Фолио. – М.: ООО «Издательство ACT», 2012. – 232 с.
7. Илюшечкин, В. М. Основы использования и проектирования баз данных: учебник для вузов [Текст] / В. М. Илюшечкин. – М.:Юрайт, 2011. – 2013 с.
8. Иванова, Г. С. Технология программирования [Текст] / Г. С. Иванова. – М.: КноРус, 2011. – 336 с.
9. Давыдова, Н. А. Программирование [Электронный ресурс] / Н. А. Давыдова, Е. В. Боровская. - М.: БИНОМ. Лаборатория знаний, 2012. - 238 с.
10. Гагарина, Л. Г. Технология разработки программного обеспечения [Текст] / Л. Г. Гагарина, Е. В. Кокорева, Б. Д. Виснадул. – М.: ИД. «Форум»: ИНФРА-М, 2011. – 400 с.
11. Бужан, В. В. Объектно-ориентированное программирование: учебно-методическое пособие для студентов очной и заочной форм обучения направления подготовки 230100.62 [Текст] / В. В. Бужан. – М.: Краснодар: ИМСИТ, 2013. – 52 с.
12. Бабаш, А. В. Информационная безопасность. Лабораторный практикум[Текст] / А. В. Бабаш. - М.: КНОРУС, 2013.-135 с.
13. Башлы, П. Н. Информационная безопасность и защита информации [Электронный ресурс]: / П. Н. Башлы, А. В. Бабаш, Е. К. Баранова. - М.: РИОР, 2013. - 222 с.
14. Баранова, Е. К. Основы информатики и защиты информации [Электронный ресурс]: / Е. К. Баранова. - М.: РИОР : ИНФРА-М, 2013. - 183 с.
15. Баранова, Е. К. Информационная безопасность и защита информации [Текст]: учеб. пособие.- 2-е изд./ Баранова, Е. К, Бабаш А. В. - М.: РИОР: ИНФРА-М, 2014.-256 с.
16. Карпова, И. П. Базы данных. Учебное пособие. Курс лекций и материалы для практических занятий [Текст] / И. П. Карпова. – М.: СПб.: Питер, 2013. — 240 с.
17. Российская государственная библиотека [Электронный ресурс]/ Центр информ. РГБ; ред. Власенко Т.В.; Web-мастер Козлова Н.В. Электрон. дан. –М.: Рос. гос. б-ка, 1997. – Режим доступа http://www.rsl.ru, свободный. Загл. с экрана – Яз. рус., англ.
18. Википедия [Электронный ресурс] – Режим доступа https://ru.wikipedia.org, свободный. Загл. с экрана – Яз. рус., англ.
19. Электронно-библиотечная система Znanium.com/ разработка Научно-издательского центра ИНФРА-М. – Режим доступа https://znanium.com, свободный. Загл. С экрана – Яз. рус., англ.
20. Рихтер, Дж. CLR via C# программирование на платформе Microsoft .Net Framework 4.5 на языке C# [Текст] / Дж. Рихтер. – 4-е издание – СПб.: Питер, 2016. – 896 с.: ил. – (Серия «Мастер-класс»).

1. PK-первичный ключ

   FK-внешний ключ [↑](#footnote-ref-1)