

Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ОРЕНБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет математики и информационных технологий

Кафедра информатики

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Направление подготовки 09.03.02 Информационные системы и технологии

**Разработка информационной системы учета заявок по ремонту
компьютерной техники**

Пояснительная записка

ОГУ 09.03.02. 1022. 202 ПЗ

Заведующий кафедрой
канд. техн. наук, доцент

 14.06.2022
подпись дата

М.А. Токарева

Руководитель
канд. пед. наук, доцент

 14.06.2022
подпись дата

М.И. Глотова

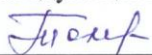
Студент

 14.06.2022
подпись дата

А.А. Мещеринов

Оренбург 2022

Утверждаю
заведующий кафедрой информатики


М.А. Токарева
подпись

« 20 » октября 2021 г.

ЗАДАНИЕ

на выполнение выпускной квалификационной работы

студенту Мещеринову Антону Александровичу

по направлению подготовки 09.03.02 Информационные системы и технологии

1 Тема ВКР: Разработка информационной системы учета заявок по ремонту компьютерной техники

2 Срок сдачи студентом работы «14» июня 2022 г.

3 Цель и задачи работы

Цель: разработка информационной системы учёта заявок по ремонту компьютерной техники

Задачи:

- проанализировать деятельность работы сервисного центра по ремонту компьютерной техники и выявить информационные потоки;
- провести сравнительный анализ аналогов проектируемой информационной системы;
- обосновать выбор инструментальных средств и СУБД для разработки информационной системы;
- провести структурное (функциональное) и объектно-ориентированное моделирование информационной системы;
- спроектировать базу данных, разработать алгоритм и приложение информационной системы;
- протестировать систему и разработать руководства системного программиста, администратора и пользователя системы.

4 Исходные данные к ВКР: результаты преддипломной практики.

5 Перечень вопросов, подлежащих разработке:

- 5.1 анализ информационных процессов предметной области;
- 5.2 обзор аналогов проектируемой системы;
- 5.3 выбор инструментальных средств разработки проектируемой системы;
- 5.4 проектирование концепций информационной системы;
- 5.5 разработка базы данных и компонентов проектируемой системы;
- 5.6 тестирование информационной системы;
- 5.7 разработка руководства системного программиста, администратора системы и пользователя системы.

6 Перечень графического материала:

- 6.1 постановка задачи;
- 6.2 диаграмма декомпозиции работы сервисного центра и проектируемой ИС;
- 6.3 диаграмма прецедентов проектируемой ИС;
- 6.4 диаграмма классов проектируемой ИС;
- 6.5 диаграмма компонентов проектируемой ИС;
- 6.6 примеры работы приложения.

Дата выдачи и получения задания

Руководитель «19» октября 2021 г.

Студент «19» октября 2021 г.


подпись

подпись

М.И Глотова

А.А. Мещеринов

Аннотация

Выпускная квалификационная работа посвящена разработке информационной системы учета заявок по ремонту компьютерной техники.





В данной работе для проектирования информационной системы были использованы средства как структурно-функционального, так и объектно-ориентированного моделирования.

В первом разделе пояснительной записки был проведен анализ предметной области, исследование информационных процессов сервисного центра по ремонту компьютерной техники, также представлен обзор аналогов систем учета заявок по ремонту компьютерной техники.

Во второй главе было проведено объектно-ориентированное моделирование информационной системы, спроектирована база данных, разработана и протестирована информационная система учета заявок по ремонту компьютерной техники.

В третьей главе было разработано руководство для системного программиста, администратора системы, а также руководство пользователя.

Пояснительная записка содержит 80 страниц, включая 1 приложение, 73 рисунка, 7 таблиц, список из 30 используемых источников.

					ОГУ 09.03.02. 1022. 202 ПЗ						
Изм	Лист	№ документа	Подп.	Дата	Разработка информационной системы учета заявок по ремонту компьютерной техники.	Литер			Лист	Листов	
Разраб		Мещеринов А.А.		14.06		В	К	Р	3	80	
Проб		Глотова М.И.		14.06							
Н. контр.		Мучкаева Е.А.		14.06		ФМИТ, 18ИСТ(ба)ОП					
Зав. каф.		Токарева М.А.		14.06.							

Summary

The final qualifying work is devoted to the development of an information system for accounting applications for the repair of computer equipment.

In this paper, both structural-functional and object-oriented modeling tools were used to design the information system.

In the first section of the explanatory note, an analysis of the subject area, a study of the information processes of the service center for the repair of computer equipment was carried out, and an overview of analogs of accounting systems for applications for the repair of computer equipment was also presented.

In the second chapter, an object-oriented modeling of the information system was carried out, a database was designed, an information system for accounting applications for computer equipment repair was developed and tested.

In the third chapter, a manual was developed for the system programmer, the system administrator, as well as a user manual.

The explanatory note contains 80 pages, including 1 appendix, 73 figures, 7 tables, a list of 30 sources used.

Содержание

Введение.....	6
1 Исследование процесса работы сервисного центра	7
1.1 Анализ предметной области	7
1.2 Обзор аналогов проектируемой системы	9
1.3 Исследование информационных потоков ИС.....	12
1.4 Формальная постановка задач для ИС.....	20
2 Разработка информационной системы учета заявок по ремонту компьютерной техники	22
2.1 Объектно-ориентированное моделирование ИС	22
2.2 Проектирование базы данных	28
2.3 Разработка алгоритма и приложения	36
2.4 Тестирование информационной системы	49
3 Эксплуатационно-технологический раздел.....	56
3.1 Руководство для системного администратора.....	56
3.2 Руководство для администратора системы	57
3.3 Руководство для пользователя.....	59
Заключение	61
Список используемых источников	62
Приложение А (обязательное) Фрагмент листинга программы.....	64

Введение

В современном мире почти у каждого человека есть персональный компьютер, ноутбук или другой вид компьютерной техники. Каждый человек сталкивался с поломкой компьютерной техники и если иногда пользователь может сам починить мелкую неисправность, то в большинстве случаев, для починки требуются специалисты. Решая проблему потребности в ремонте компьютерной техники и, создаются множество сервисных центров для ремонта.

Для сервисных центров основная деятельность заключается в приёме устройств нуждающиеся в ремонте, от юридических или физических лиц, а также возможной диагностики, модернизации, обслуживания по гарантии и других действиях, которые невозможно провести без специалиста.

Предприятия, которые оказывают услуги по ремонту и обслуживанию компьютерной техники, ставят одной из важных задач учет информации о заявках на ремонт и состоянии устройства. Поэтому для качественного выполнения производственно-технического процесса, необходимо грамотно вести учет всех необходимых для этого элементов.

На данный момент существует множество готовых информационных систем для сервисных центров, но многие из них стоят немало денег и требуют настройку под само предприятие. Функционал таких систем очень большой, что иногда тоже является минусом, потому что компания переплачивает за функционал, которым не будет пользоваться. В данном случае можно разработать простую информационную систему, которая будет выполнять все необходимые функции и обойдется не так дорого, как уже имеющиеся аналоги.

Объектом исследования является деятельность сервисного центра по ремонту компьютерной техники.

Предметом исследования является автоматизация учета заявок по ремонту компьютерной техники.

Целью выпускной квалификационной работы (ВКР) является разработка информационной системы учёта заявок по ремонту компьютерной техники.

Для достижения поставленной цели необходимо решить следующие задачи:

- 1) проанализировать деятельность работы сервисного центра по ремонту компьютерной техники и выявить информационные потоки;
- 2) провести сравнительный анализ аналогов проектируемой информационной системы;
- 3) обосновать выбор инструментальных средств и СУБД для разработки информационной системы;
- 4) провести структурное (функциональное) и объектно-ориентированное моделирование информационной системы;
- 5) спроектировать базу данных, разработать алгоритм и приложение информационной системы;
- 6) протестировать систему и разработать руководства системного программиста, администратора и пользователя системы.

1 Исследование процесса работы сервисного центра

1.1 Анализ предметной области

Технологическое развитие с каждым годом набирает все большие обороты и сейчас многие вещи, которые раньше казались невыполнимыми, встречаются на каждом шагу и давно вошли в жизнь каждого человека. Такой процесс произошел и с персональными компьютерами (ПК), когда-то ПК занимал целую комнату и имел огромный вес и сложные элементы управления, только специалисты могли справиться с его работой. Применялись же такие устройства в узких научных целях и для простых пользователей, оставались недостижимы. Однако время идет и технологии не стоят на месте, поэтому на данный момент почти у каждого человека в России есть персональный компьютер, ноутбук или другой вид компьютерной техники. Развитие рынка компьютерной техники в нашей стране достигло очень больших высот и теперь число ПК уже превышает число жителей. Например, в период с 2010 по 2020 год число персональных компьютеров изменилось с 63 до 125 на 100 человек. Так же стоит заметить, что почти все компании по производству ПК или ноутбуков, часто выпускают обновленную продукцию, которая имеет свежие характеристики и является более привлекательной для покупателя. В данном случае модель годичной или двухгодичной давности становится дешевле, на неё появляются скидки и пользователи, которые не могли её себе позволить, теперь приобретают заветный товар.

К сожалению, в России, производится не так много комплектующих для сборки компьютерной техники, и большая часть закупается из-за рубежа. Самую большую часть поставок осуществляют страны азиатского региона, а также Соединенные Штаты Америки. Можно выделить основные страны, которые поставляют комплектующие для России к ним относятся: Китай, США, Тайвань, Таиланд, Германия, Венгрия, Япония, Ирландия, Мексика. На период 2021 года самым большим поставщиком компьютеров стала Американская компания «Apple». На долю этой компании пришлось 17,1% всех поставок, так же ближайших конкурентов были следующие показатели – 16,5% у Lenovo, Asus 9,1%, HP 15% [1]. Более подробный рост поставок представлен на рисунке 1.1. Общее же число поставок персональных компьютеров в 2021 году составило 9,9 миллионов, а ноутбуков 1,3 миллиона, в то время как на конец 2020 года эти показатели были 7, 75 и 1,14 миллиона.

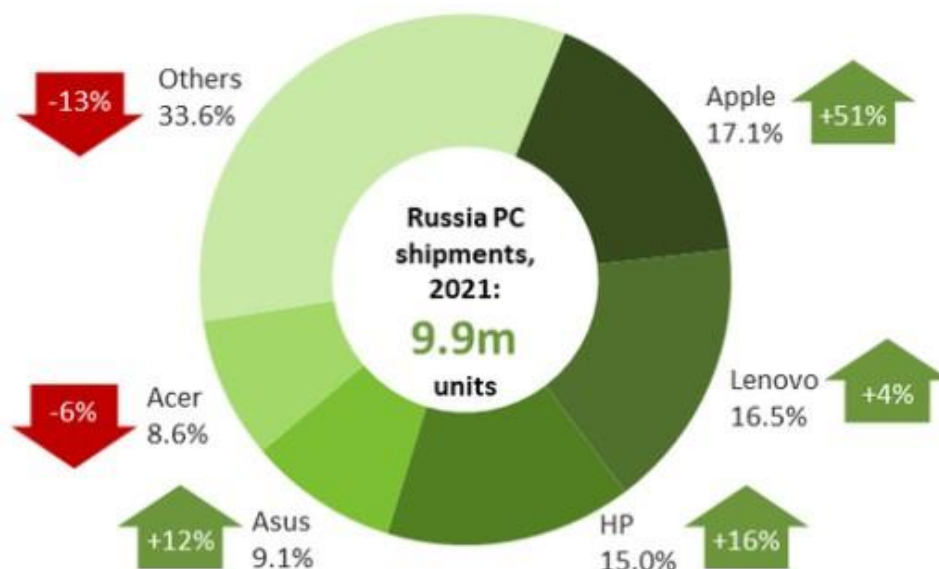


Рисунок 1.1 – Доля поставок ПК на российский рынок за 2021год

В первом квартале 2022 года в связи с санкционными событиями происходящим в России доля всех поставок ПК и комплектующих успела сократиться на 4,3% и будет продолжать падать. Этому так же могут послужить резких спрос, произошедший в период пандемии 2020 – 2021 год. Теперь же показатели будут падать. В связи с дефицитом комплектующих и ПК цены на них сильно поднялись и появился большой спрос на покупку уже поддержанной компьютерной техники, на момент марта 2022 года спрос уже вырос на 35% по сравнению с февралём 2022 года, где составлял 16%. По данным сайта «Авито» на март 2022 года количество объявлений о продаже поддержанных ноутбуков увеличились на 28%, а персональных компьютеров на 27% и данная динамика будет увеличиваться.

Таким образом проведя анализ рынка компьютерной техники в России и беря в учет события происходящие в данный период, можно сделать простой вывод: цены на компьютерную технику и комплектующие будут расти и многие покупают уже поддержанную технику, а значит риск поломки у неё гораздо выше чем у новой, в связи с чем спрос на ремонт будет еще больше. Пользователю будет намного дешевле починить неисправность чем приобрести новый девайс. Естественно, спрос на различные сервисы по ремонту компьютерной техники будет расти.

Сервисный центр по ремонту компьютерной техники осуществляет прием заявок на ремонт и оборудование, которое клиент приносит для ремонта. Так же сам сервисный центр осуществляет закупку необходимых деталей для ремонта [2-3].

Полная информационная система сервисного центра по ремонту ПК имеет множество отделов (элементов функционирования) (рисунок 1.2):

- 1) Финансовый отдел. Служит для учета дохода при выполнении заявки и расходы на закупку необходимых деталей;
- 2) Директор компании. Найма новых сотрудников и регулирования всех остальных отделов;

- 3) Производственный отдел. Осуществляет приём заявок, диагностику и ремонт оборудования, и составление отчета по выполненной работе;
- 4) Отдел закупок. Составление списков необходимых деталей и закупку их у поставщиков;
- 5) Юридический отдел. Составление все нормативных документов и проверка документов гарантии, а также решение возникших проблем при формировании документации.

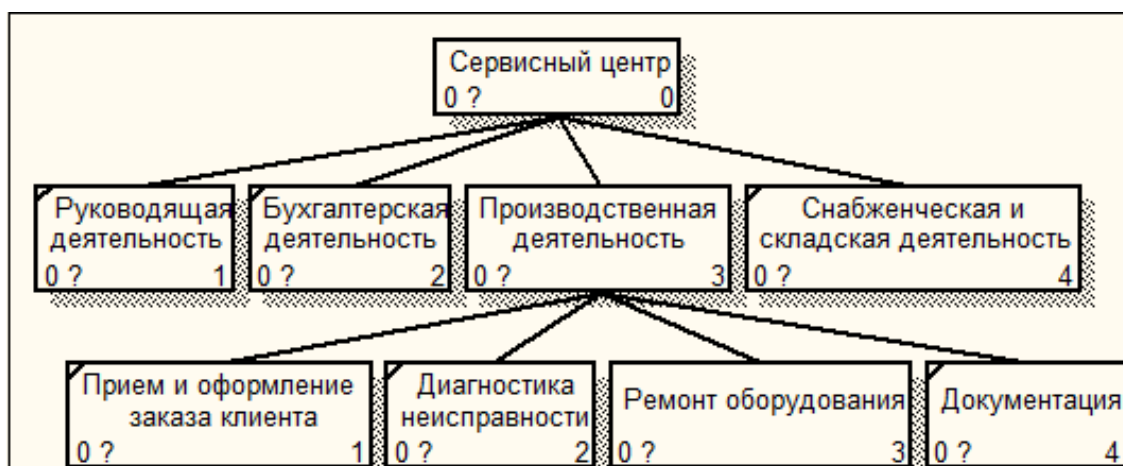


Рисунок 1.2 – Диаграмма распределения видов деятельности сервисного центра по ремонту компьютерной техники

Так как разрабатываемая нами информационная система будет использована для производственного отдела его деятельность разбивается на четыре подуровня для подробного понимания процессов в производственной деятельности сервисного центра [4].

Формирование заявки на ремонт компьютерной техники, осуществляется следующим образом. Клиент обращается в компанию с какой-либо неисправностью, описывает проблему. После передачи устройства производится диагностика неисправности, инженер определяет сколько времени и какие детали необходимы для ремонта. Отправляется запрос на склад с уточнением о наличии необходимых деталей. После положительного ответа формируется документ, в котором указывается проблема, результат диагностики, какие детали необходимы для ремонта и стоимость работы. Данный документ отправляется по почте или другими способами клиенту и после согласования происходит ремонт оборудования.

1.2 Обзор аналогов проектируемой системы

Для решения задачи учета заявок на ремонт компьютерной техники, существуют уже готовые системы. Они имеют весь необходимый функционал и обслуживаются компаниями, которые их разработали. Все они имеют свои плюсы и минусы, различные функционал и способы работы с ними. Для примера проведём

анализ нескольких систем, а также разберемся, почему разработка системы в задаче ВКР является актуальной.

Первой системой рассмотрим «1С: Предприятие». Разработанная система для сервисного центра имеет комплексное решение многих задач, а именно осуществляется автоматизация деятельности организации в следующих разделах: финансовый учет, управленческий учет, кадровый учет и т.д. Так же система имеет разделение на разделы и компоненты управления для каждого из них, иными словами директор при входе в систему сможет вести деятельность по управлению всей системы, бухгалтер иметь доступ к отчетам и финансовым операциям, а менеджер и инженер вести деятельность по учету заявок, ремонту и т.д. Система созданная на платформе «1С: Предприятие», так же имеет свойства изменять конфигурацию и подстраиваться под индивидуальные особенности того или иного сервисного центра, то есть является очень гибкой в плане разработке [5]. На рисунке 1.3 представлено окно приема на ремонт системы, разработанной на платформе «1С: Предприятие», для сервисного центра.

The screenshot shows the 'Ремонт и обслуживание' (Repair and Maintenance) window in the '1C: Enterprise' software. The window has a top bar with navigation icons and a title bar. Below the title bar, there are buttons for 'Создать' (Create), 'Создать на основании' (Create based on), 'Отбор' (Selection), and 'Настройка: Мастер' (Settings: Master). A search bar is also present.

The main area contains a table with columns: 'Дата' (Date), 'Тип документа' (Document type), 'Текущий этап' (Current stage), 'Номенклатура' (Nomenclature), 'Принадлежность т...' (Ownership), and 'Серийный номер' (Serial number). The table lists several repair orders, including 'Принем в ремонт' (Accept for repair), 'Выезд мастера' (Master's visit), 'Задание на работу' (Work order), 'Заказ-наряд' (Order-work order), 'Поступление в кассу' (Receipt in cash), and 'Возврат из ремонта' (Return from repair).

Below the table, there is a detailed form for a specific repair order. The form includes fields for 'Номер' (Number), 'Тип ремонта' (Type of repair), 'Организация' (Organization), 'Номер гарантии' (Warranty number), 'Срок действия' (Term of validity), and 'Склад' (Warehouse). It also has a section for 'Ремонт' (Repair) with sub-sections for 'Результат' (Result), 'Контрагент' (Counterparty), 'Доставка' (Delivery), 'Предоплата' (Prepayment), 'Обсуждения' (Discussions), and 'Дополнительно' (Additional). The 'Результат' section shows 'Не работает' (Not working).

Рисунок 1.3 – Система для управления сервисным центром на базе «1С: Предприятие»

Естественно, система, разработанная на платформе «1С: Предприятие» имеет множество плюсов, что подчеркивается так же тем, что многие крупные компании используют её для своих целей. Но к минусам можно отнести высокая стоимость покупки системы, а также затраты на специалистов для её изменения или создания с полного нуля. В среднем покупка лицензии на 5 рабочих мест стоит 21000 рублей.

Следующим аналогом проектируемой системы можно привести «Gincore». Данная программа для учета в сервисном центре так же, как и описанный ранее «1С: Предприятие» имеет большой функционал, а именно: учет ремонта, продаж,

остатки на складе, бухгалтерский учет, логистика, формирование документов и многое другое. Программу ненужно устанавливать так как доступ к ней устанавливается через браузер с доступом в интернет, то есть является веб-приложением. Программа имеет пробный период для тестирования, но как правило данную систему выбирают крупные фирмы, для которых важен большой функционал и возможность отслеживать каждый шаг. Меню программы представлено на рисунке 1.4.

The screenshot shows the Gincore web application interface. On the left is a sidebar menu with options: Главная, Заказы, Клиенты, Бухгалтерия, Склады, Логистика, Товары, Категории, and Еще. The main area displays a table of orders with columns: № заказа, Приемщик, Менеджер, Статус, Устройство, Стоимость, Оплачено, Клиент, and Сроки. The table contains 15 rows of order data with various statuses like 'В процессе ремонта', 'Ожидает запчасти', 'Готов', 'Выдан', and 'На диагностике'.

№ заказа	Приемщик	Менеджер	Статус	Устройство	Стоимость	Оплачено	Клиент	Сроки
435	Сергей		В процессе ремонта	Ноутбук Acer Extensa...	0	0	sdf 79647584712	Не срочно
431	Владимир Смирнов	Менеджер	Ожидает запчасти	4 шт.	1000	1000	BB 74994233443	Не срочно
430	Sophie Walker	Sophie Walker	Готов	3 шт.	1000	1000	Киселевич Богдан Вас...	Не срочно
428	Sophie Walker		Принят в ремонт	3 шт.	3698	3333	Киселевич Богдан Вас...	Не срочно
427	Sophie Walker	Sophie Walker	Выдан без ремонта	3 шт.	9999	90	B 380631786290	Не срочно
426	Sophie Walker	Sophie Walker	В процессе ремонта	3 шт.	9797	9797	Киселевич Богдан Вас...	Срочно
424	Сергей		Выдан	Ноутбук Acer Extensa...	1223	1223	79463546111	Не срочно
371	Владимир Смирнов	Менеджер	Выдан без ремонта	Sony VAIO Pro SVP132...	2000	2000	Пивцакин Никита	Срочно
387	Владимир Смирнов	Менеджер	Ожидает запчасти	Apple MacBook Air 13...	2000	999	Квартальный Данила	Срочно
403	Владимир Смирнов	Менеджер	Выдан	Apple MacBook Pro 13...	2000	2121	Да Коста Стефан	Не срочно
419	Владимир Смирнов	Менеджер	Клиент отказался	Apple MacBook Pro 15...	2000	3124	Да Коста Стефан	Не срочно
372	Владимир Смирнов	Менеджер	Готов	Sony VAIO Pro SVP112...	2000	0	Ожиганов Игорь	Не срочно
388	Владимир Смирнов	Менеджер	Ожидает запчасти	Sony Vaio Z Flip (VJ...	2000	0	Да Коста Стефан	Не срочно
404	Владимир Смирнов	Менеджер	Принят в ремонт	Apple MacBook Pro 13...	1000	2192	Да Коста Стефан	Не срочно
420	Владимир Смирнов	Менеджер	На диагностике	Apple MacBook Pro 15...	2000	3195	Да Коста Стефан	Не срочно

Рисунок 1.4 – Система для управления сервисным центром на базе веб-приложения «Gincore»

Весь богатый функционал программы является и её минусом, для компаний которым не нужно так много действий в системе. Из-за многообразия функционала в нем не всегда легко разобраться и старт такой системы в новую компанию будет сложным испытанием. При покупке лицензии покупатель может выбрать тарифный план, цена минимального ровняется 1156 рублей за месяц подписки, цена максимального 5783 рублей за месяц подписки. К сожалению, только максимальный тарифный план имеет возможность настройки функционала под требования клиента, то есть при других тарифах будет лишний функционал. Так же не всегда месячная подписка является подходящим вариантом для покупки [6-8].

Анализируя аналоги разрабатываемой информационной системы, можно сделать выводы, что основным минусом является излишний функционал системы и высокая стоимость программного продукта. Система, которая будет разработана в ходе выполнения цели ВКР будет иметь только необходимый функционал, для учета заявок по ремонту компьютерной техники и так же её разработка не требует больших затрат.

1.3 Исследование информационных потоков ИС

1.3.1 Сравнение и выбор CASE-средства. Для исследования информационных потоков, необходимо использовать средства автоматизации разработки программ (CASE-средства). Так как разработка и создание информационных систем напрямую связаны с выявлением всех бизнес-процессов предприятия и дальнейшим их анализом, определением взаимосвязи элементов процессов, необходимо очень грамотно подойти к выбору средств разработки. На данный момент можно выделить два самых востребованных CASE-средства для структурного моделирования и анализа процессов, происходящих в предприятии, этими программами являются All Fusion Process Modeler 7 и Ramus. Необходимо выделить основные плюсы каждой программы и сделать сравнение, чтобы выбрать подходящий для решения поставленной цели.

Первым рассмотрим All Fusion Process Modeler (BPwin). Данный программный продукт является достаточно распространённым решением для моделирования, позволяющий проводить анализ, улучшение бизнес-процессов и документирование. Благодаря данной программе можно моделировать различные процессы, определять их ресурсы и порядок взаимодействия. Так же выделим основные плюсы данного программного продукта:

- 1) пользовательский интерфейс является интуитивно понятным;
- 2) автоматизированные процессы проектирования;
- 3) использование стандарта FIPS;
- 4) предварительное тестирование;
- 5) несколько стандартов моделирования;
- 6) документация к каждой модели;
- 7) генератор отчетов [10-11].

Теперь для сравнения рассмотрим особенности и плюсы программы Ramus. Данный программный продукт так же служит для моделирования бизнес-процессов и дальнейшего их анализа, программа позволяет и визуально демонстрировать все процессы и задачи. Основные плюсы:

- 1) имеет редактор графических файлов;
- 2) возможность создавать отчеты и документы;
- 3) поддержка популярных методологий DFD и IDEF0;
- 4) кроссплатформенность.

Проведя небольшой анализ двух программ для моделирования бизнес-процессов, можно сделать вывод, что All Fusion Process Modeler, является более оптимальным решением. Так как имеет больше плюсов и менее сложный интерфейс по сравнению с Ramus.

1.3.2 Разработка функциональной модели IDEF0. Перейдем к разработке диаграммы IDEF0 в выбранной среде – All Fusion Process Modeler. Первым пунктом необходимо разработать контекстную диаграмму. Изначальный блок будет иметь название «Сервисный центр по ремонту компьютерной техники». К данному блоку будет входить 3 стрелки. Данная диаграмма строится для моделирования процессов в сервисном центре, поэтому первой стрелкой выделим «Заявку клиента», а именно

данные о клиенте. Следующей немаловажной стрелкой является «Оборудование клиента» именно по нему проводится диагностика и последующий ремонт, и последняя стрелка — это «Закупленные детали» для работы сервисного центра [12].

Теперь перейдем к этапу формирования стрелок управления контекстной диаграммы. Для управления сервисным центром служат некоторые нормативные документы, а также некоторые виды законов для регулирования прав потребителя, на возврат или ремонт по гарантии:

1) Закон РФ от 07.02.1992 N 2300-1 (ред. от 11.06.2021) О защите прав потребителей;

2) Постановлению Правительства РФ от 31.12.2020 N 2463;

3) Закон РФ от 07.02.1992 № 2300-1 [13].

Так же для управления сервисным центром осуществляются документы управления, установленные руководством.

В качестве механизмов в данной диаграмме будут служить отделы, выполняющие различные виды работы. А именно:

1) Финансовый отдел. Служит для осуществления бухгалтерской деятельности, а именно расчет дохода за определенный период, расход на покупку комплектующих, зарплата сотрудникам и составление документации для отчетов в руководящий отдел, а именно директору;

2) Руководящий отдел. Деятельность по регулированию работы, анализ всей деятельности, найма новых сотрудников осуществляется данным разделом. В него входит директор компании и администраторы;

3) Производственный отдел. Осуществляет работу и главное функционирование сервисного центра, а именно приём заявок, ремонт, диагностику оборудования и различную документацию к другим отделам;

4) Отдел закупок. Данный отдел осуществляет закупку необходимых деталей для ремонта, а также правильную логистику их доставки. Принимает отчеты об остатках деталей от производственного отдела и документ о закупке от руководящего раздела;

5) Юридический отдел. Осуществляет проверку документации, гарантийных документов, различных договоров [14].

Все пункты для создания контекстной диаграммы были разработаны и можно перейти к процессу её построения в выбранной среде All Fusion Process Modeler. Контекстная диаграмма функционирования сервисного центра по ремонту компьютерной техники представлена на рисунке 1.5.

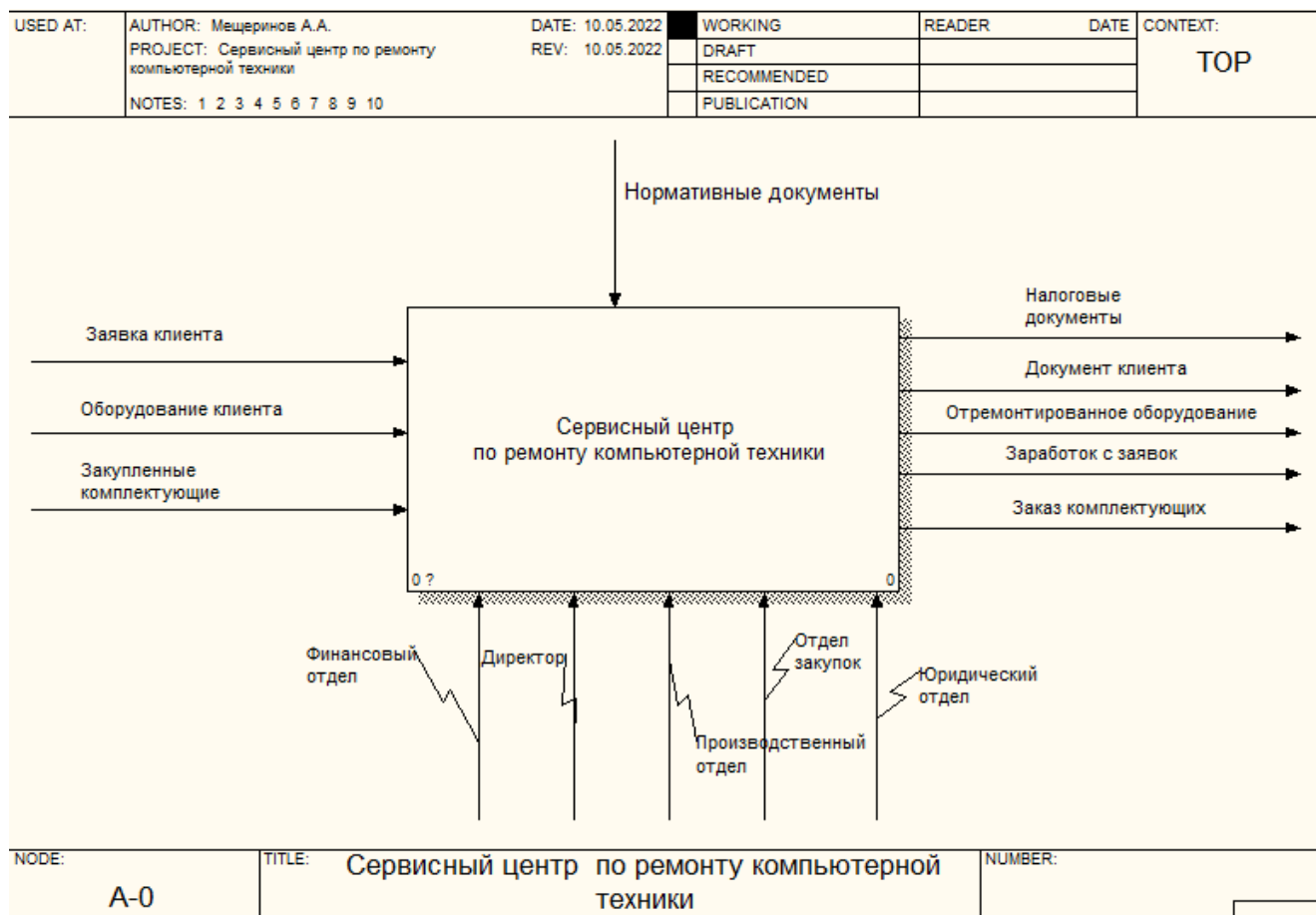


Рисунок 1.5 – Контекстная диаграмма работы сервисного центра

Теперь необходимо разработать декомпозицию контекстной диаграммы. А именно разделим целый блок на составляющие его части, на 4 функциональных блоков. Данные блоки будут демонстрировать взаимосвязи отделов сервисного центра и имеют следующие наименования:

- 1) Руководящая деятельность;
- 2) Бухгалтерская деятельность;
- 3) Производственная деятельность;
- 4) Складская и снабженческая деятельность.

Для функционального блока «Руководящая деятельность» входными стрелками будут являться отчеты остальных блоков, а именно: отчет закупок, отчет работы, отчет финансов.

Функциональный блок «Бухгалтерская деятельность» служит для учета заработка с заявок и формирование налоговых документов, они изображены выходными стрелками на диаграмме.

Для функционального блока «Производственная деятельность» входными стрелками будут «оборудование клиента» и «заявка клиента». На выходе же будут отремонтированное оборудование и гарантийный документ, а также при необходимости запрос на необходимые детали к блоку снабженческая и складская деятельность. И отправка отчета к блоку «Руководящая деятельность».

Последним функциональным блоком будет «снабжение и складская деятельность». Входной стрелкой будет запрос на необходимые детали от блока «Производственная деятельность», выходные же стрелки «необходимые детали» и «заказ деталей». На рисунке 1.6 представлена декомпозиция первого уровня диаграммы IDEF0.

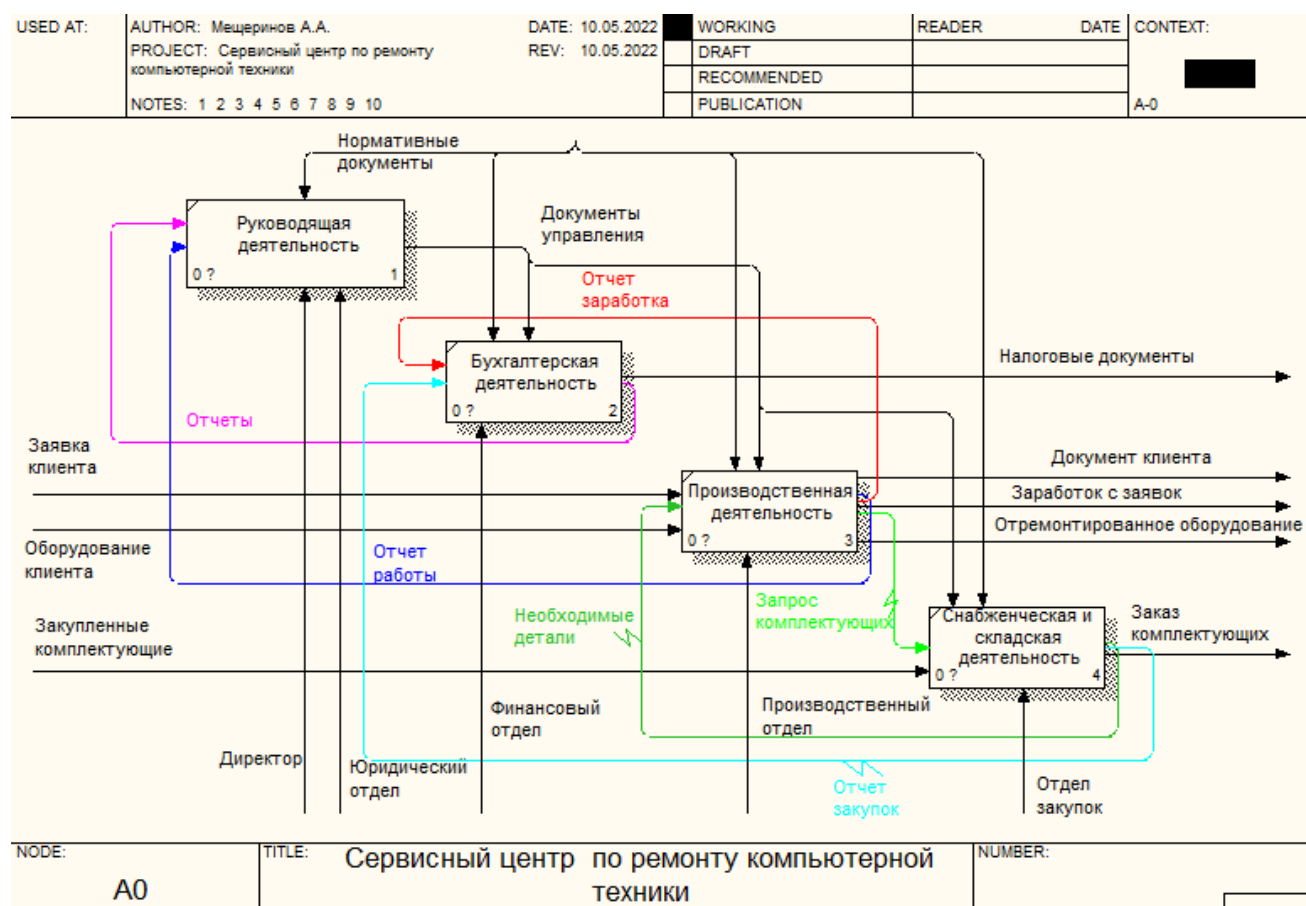


Рисунок 1.6 – Декомпозиция первого уровня

На данной диаграмме в качестве механизмов, будут те же что мы указывали ранее. Каждый из механизмов относится к своему функциональному блоку, соотношение видно по одинаковому наименованию. Для всех четырёх блоков в качестве механизмов управления будут являться «Нормативные документы», а также от блока «Руководящая деятельность» будут исходить механизмы управления, а именно «Документы управления». На данном этапе моделирование декомпозиции первого уровня можно считать завершенной. Разработка информационной системы будет осуществляться для производственной деятельности, в которой и осуществляется учет заявок клиента. На рисунке 1.7 изображена контекстная диаграмма проектируемой системы учета заявок. Входными данными в ней будут «информация о клиенте», «оборудование клиента» и «комплектующие для ремонта». Механизмами или пользователями системы будут «сотрудник» и «администратор», элемент управления «нормативные документы». А данные на выходе отремонтированное оборудование, отчет заявок, отчет остатков на складе, документ клиента.

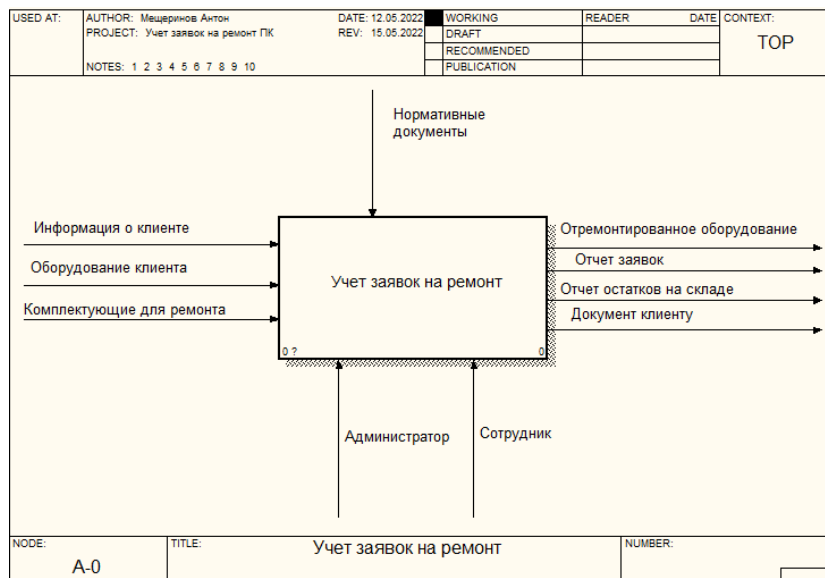


Рисунок 1.7 – Контекстная диаграмма учета заявок на ремонт

Построим диаграмму декомпозиции, описывающую процесс формирования заявки, в разрабатываемой системе (рисунок 1.8). Первым функциональным блоком будет «регистрация заявки» в который входит информация о клиенте, далее данные клиента переходят в блок «Диагностика», в который в свою очередь входит «Оборудование клиента». После диагностики оборудования необходимо внести в заявку комплектующие которые необходимы для ремонта и для этого есть блок «Комплектующие для на складе» и в него входят комплектующие для ремонта при их внесении после заказа, механизмом для этого блока служит администратор, а для всех остальных сотрудник. После добавления комплектующих для ремонта осуществляется «ремонт оборудования» и выходными данными будут документ клиенту и отремонтированное оборудование.

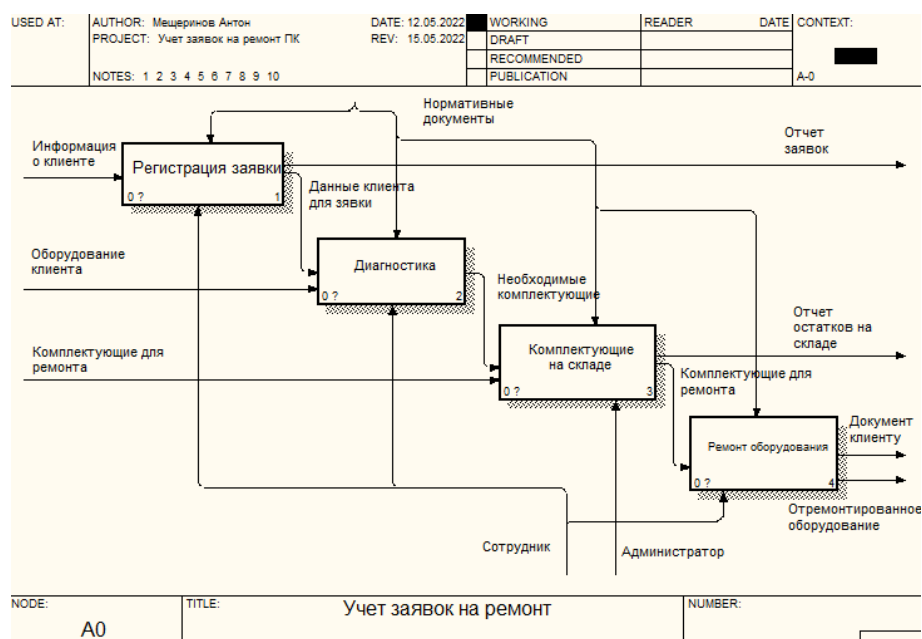


Рисунок 1.8 – Декомпозиция первого уровня «Учет заявок на ремонт»

На диаграмме декомпозиции первого уровня были изображены 4 функциональных блока, проведем декомпозицию второго уровня для каждого из них. Первым блоком рассмотрим «Регистрация заявки». Декомпозиция данного блока простая, она имеет два функциональных блока, а именно внесение данных о клиенте и после данного процесса внесение данных о статусе заявки (рисунок 1.9) [15].

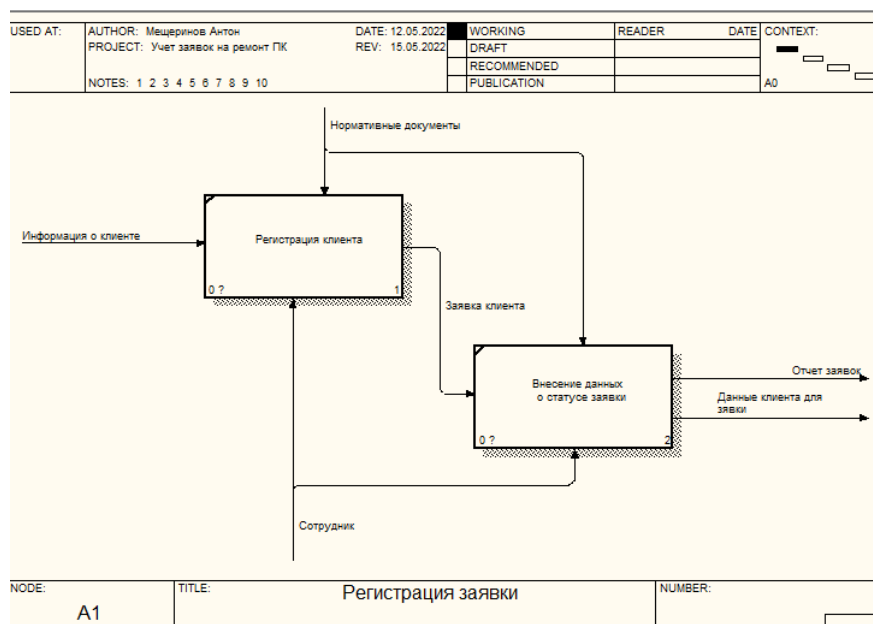


Рисунок 1.9 – Декомпозиция второго уровня блока «Регистрация заявки»

Следующий блок будет «Диагностика», в данном блоке вносится девайс клиента и данные о неисправности со слов клиента, далее идет подробная диагностика инженером и просмотр необходимых комплектующих для ремонта (рисунок 1.10).

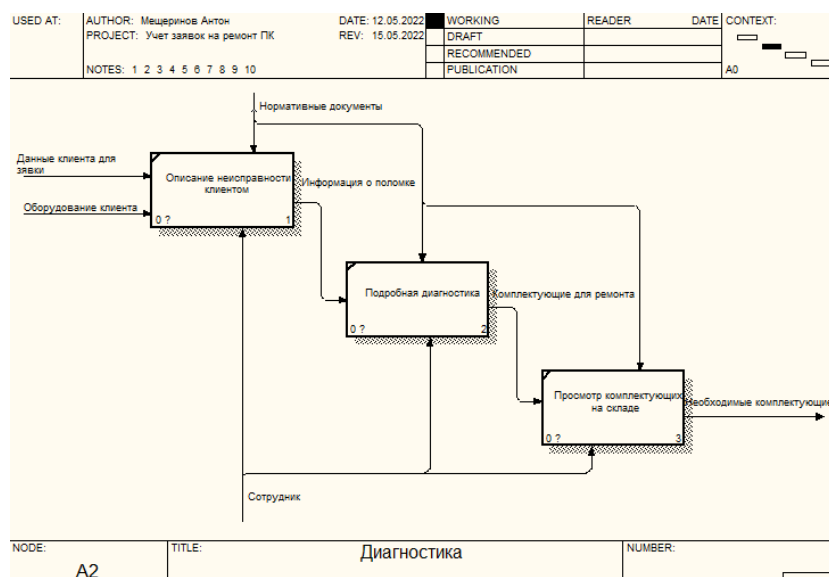


Рисунок 1.10 – Декомпозиция блока «Диагностика»

Предпоследним блоком рассмотрим «Комплектующие на складе». Для регулирования данного блока служит механизм «администратор». Администратор вносит новые комплектующие на склад и выводит отчет по ним, сотрудник же может проверить нужные комплектующие и вносить их для заявки и ремонта оборудования (рисунок 1.11).

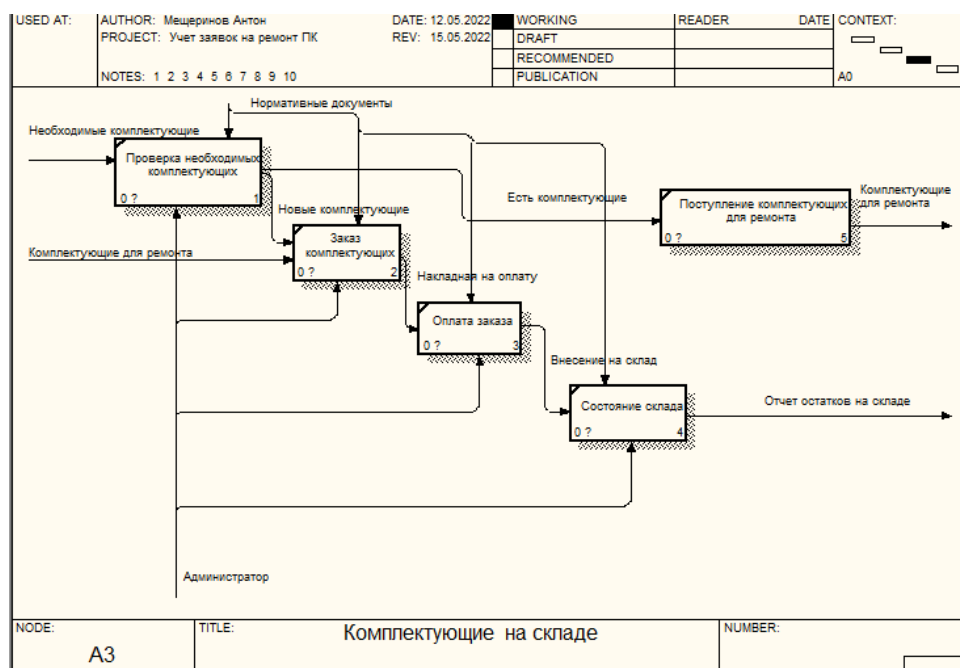


Рисунок 1.11 – Декомпозиция блока «Комплектующие на складе»

1.3.3 Разработка диаграммы потоков данных. Для создания диаграммы DFD воспользуемся All Fusion Process Modeler. Как и с диаграммой IDEF0 для разработки диаграммы потоков данных первым действием создадим контекстную диаграмму. На рисунке 1.12 представлена контекстная диаграмма потоков данных учета заявок по ремонту компьютерной техники [16].

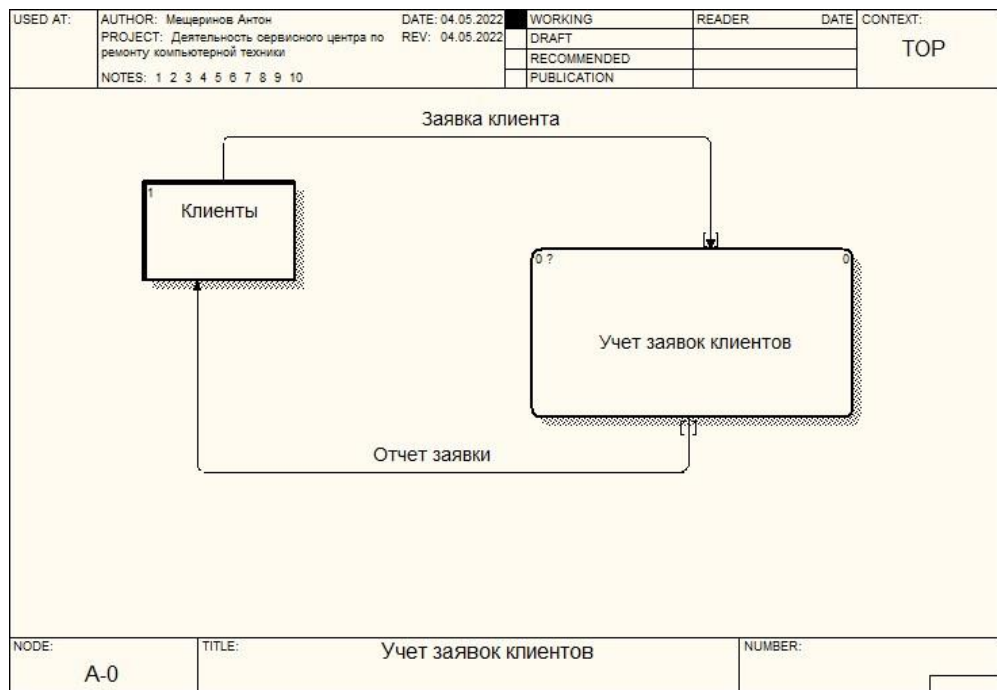


Рисунок 1.12 – Контекстная диаграмма DFD

Теперь необходимо провести декомпозицию данной диаграммы, для более подробного анализа потоков данных (рисунок 1.13).

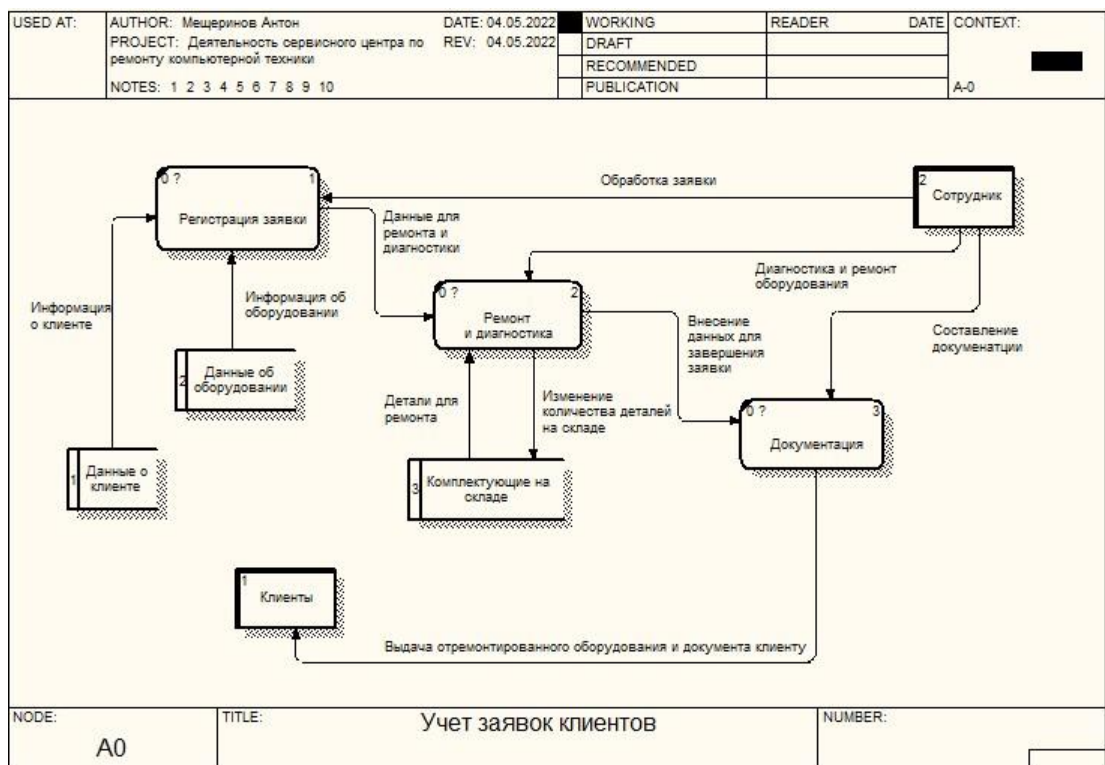


Рисунок 1.13 – Декомпозиция контекстной диаграммы DFD

Главным блоком будет «Сотрудник», который проводит основные операции для работы с заявками и данными о клиентах, оборудовании и комплектующих.

После выполнения всех этапов функционирования сервисного центра клиенту выдается документ о ремонте и его оборудовании. На данном этапе моделирование информационных потоков можно считать завершённым.

1.4 Формальная постановка задач для ИС

1.4.1 Введение. Настоящее техническое задание распространяется на разработку информационной системы учёта заявок по ремонту компьютерной техники.

Объектом исследования является учет заявок на ремонт. Проектируемая система разделяется на два уровня доступа «Администратор» и «Сотрудник». Для администратора будет доступ для внесения новых категорий комплектующих, внесение новых комплектующих для ремонта и пополнение уже имеющихся на складе, внесение новых сотрудников. При входе же сотрудника он сможет внести клиента, девайс клиента и сформировать заявку. Так же в системе будет реализован поиск по таблицам «Заявки» и «Комплектующие», а также формирование отчетов в Excel по данным таблиц. После формирования заявки будет построен график, показывающий динамику изменения дохода с заявок на период. Так же по заявке формируется чек клиенту с оплатой. Данный функционал является простым, но выполняет все необходимые элементы для поставленной задачи, и разработка такой системы не требует больших денежных затрат. Из этого следует, что минусы, обнаруженные в ходе анализа аналогичных систем, исчезают при разработке данной системы.

1.4.2 Назначение. Данная версия информационной системы предназначена для учета заявок по ремонту компьютерной техники.

1.4.3 Требования к программному обеспечению.

1.4.4.1 Требования к функциональным характеристикам. Информационная система учета заявок на ремонт компьютерной техники, должна иметь проверки на внесение деталей на ремонт, а именно сообщать об ошибке если деталей не хватает, не проводить заявку на прошлую дату или будущую. В функции информационной системы так же должны входить:

1) учет деталей для осуществления ремонта компьютерной техники, а именно их добавление, редактирование, удаление;

2) учет и добавление новых категорий для комплектующих, в связи с возможным расширением производственной деятельности сервисного центра;

3) учет клиентов, которые обратились с заявкой на ремонт, а также привязка к ним их девайса;

4) учет сотрудников предприятия, а также добавление, удаление и редактирование их;

5) разделение уровней доступа входа в систему (администратор и сотрудник);

6) учет заявок на ремонт;

7) формирование заявки для ремонта с учетом затраченных деталей и вида работы, расчет полной стоимости, предоставленной к оплате;

- 8) формирование графика дохода за весь период работы;
- 9) формирование и экспортирование в Excel данных по заявкам;
- 10) формирование чека для выдачи клиенту.

1.4.4.2 Требования к надежности системы. Необходимо разработать диалоговые окна для сообщения об ошибках, а именно:

- 1) ошибка неверного логина или пароля при авторизации;
- 2) ошибки на внесение пустых значений;
- 3) ошибка при добавлении одинаковых данных;
- 4) ошибка при внесении некорректных данных;
- 5) ошибка при выборе неправильной даты.

1.4.5 Требования к документации. Данная система так же должна иметь справочную информацию, о разработчике и подсказки пользователю. Сопровождающая информация должна содержать:

- 1) руководство пользователя системы;
- 2) руководство администратор системы;
- 3) руководство системного программиста.

При разработке информационной системы необходимо соблюсти все вышеперечисленные требования.

2 Разработка информационной системы учета заявок по ремонту компьютерной техники

2.1 Объектно-ориентированное моделирование ИС

2.1.1 Основные понятия. Объектно-ориентированный подход к моделированию информационных систем в настоящее время считается одним из самых эффективных. Данную популярность можно объяснить тем, что среди всех способов проектирования ИС именно объектно-ориентированный подход (ООП) оперирует абстракциями реальных операций и объектов из окружающего мира. Целью такого подхода является выделение объектов, составляющих организацию, и описание взаимодействия между этими объектами.

Объектно-ориентированное моделирование программного обеспечения связана с применением объектно-ориентированных технологий. Обычно эти объектно-ориентированные методологии поддерживаются инструментальными программными средствами, но и без таких средств они полезны, так как позволяют хорошо понять различные аспекты и свойства разрабатываемой программной системы, что в последующем существенно облегчает ее реализацию, тестирование, сопровождение, разработку новых версий и более существенную модификацию.

Так же стоит отметить, что одной из ключевых особенностей объектно-ориентированного подхода к моделированию, является унифицированность процесса разработки ИС. Разработка в данном случае происходит по средствам языка UML – унифицированный язык моделирования. Плюс данного подхода заключается в распределении и формированию обязанностей на этапе проектирования информационной системы [17].

2.1.2 Обоснование выбора продукта описания и анализа бизнес-процессов. Прежде чем начать создавать основные компоненты программы необходимо описать взаимодействие этих компонентов до написания кода, а также разработать классы, которые будут взаимодействовать в программе. Для решения процесса проектирования структуры программы до реализации её в среде разработки была реализована методология объектно-ориентированного анализа и проектирования.

В качестве программы в которой будет происходить проектирование модели системы была выбрана Rational Rose. Данный программный продукт относится к семейству объектно-ориентированных CASE-средств, главной целью которых является автоматизация проектирования ПО. В основе данного программного продукта лежит спецификация UML, которая определяет динамические и статические аспекты архитектуры системы. Основные плюсы Rational Rose:

- 1) интуитивно понятный графический интерфейс пользователя;
- 2) есть средство автоматической генерации кодов программы;
- 3) наличие средства контроля проектов;
- 4) репозиторий представляет собой базу данных проекта;
- 5) генерация документов на основе информации из репозитория;
- 6) для просмотра проекта реализовано средство просмотра – браузер;

- 7) интеграция моделей;
- 8) интеграция с другими программными продуктами, например с Microsoft Visual Studio;
- 9) открытая архитектура.

Данный список охватывает далеко не все плюсы Rational Rose, но и на основе него и небольшого анализа становится понятно, что для решения поставленной задачи, а именно разработать объектно-ориентированные модели будущей системы, данный программный продукт является лучшим [18].

2.1.3 Построение диаграммы прецедентов. Первым пунктом перейдем к созданию диаграммы прецедентов (диаграмма вариантов использования). Данная диаграмма отражает отношение между актерами и прецедентами. Такой подход позволяет описать систему на концептуальном уровне. Диаграмма прецедентов проектируемой системы представлена на рисунке 2.1.

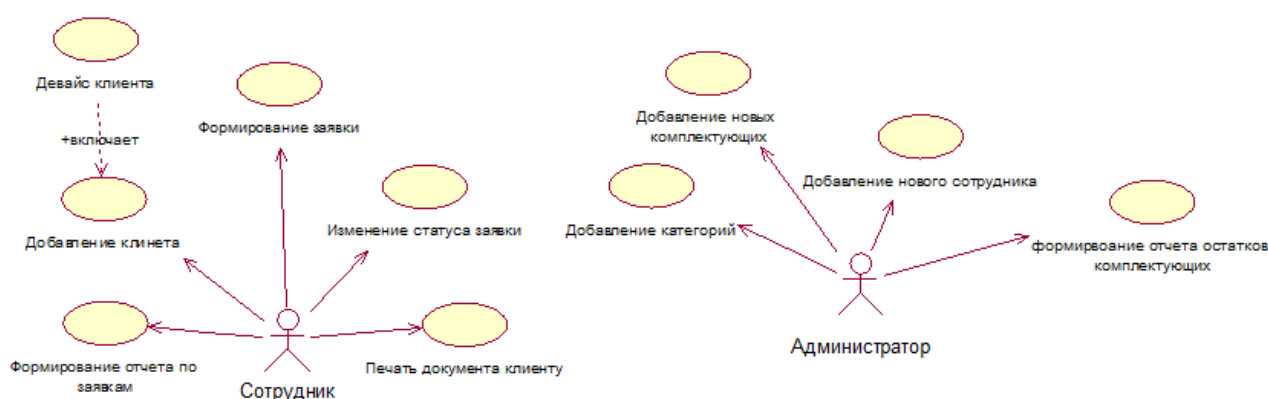


Рисунок 2.1 – Диаграмма прецедентов

На данной диаграмме имеется 2 актера: сотрудник и администратор. Сотрудник данной информационной системы сможет производить следующие действия:

- 1) Добавление клиента и его девайса;
- 2) Формирование заявки;
- 3) Изменение статуса заявки;
- 4) Печать документа клиенту;
- 5) Формирование отчета по заявкам.

Для администратора системой предусмотрены другие функции, а именно добавление категорий, новых комплектующих, нового сотрудника и формирование отчета по остаткам на складе. После построения диаграммы прецедентов, перейдем к описанию диаграмме взаимодействия [19].

2.1.4 Построение диаграммы взаимодействия. Простыми словами диаграмма взаимодействия служит для изображения взаимодействий множества объектов и взаимодействие между ними по средствам сообщений, которыми они обмениваются. Основным объектом в проектируемой системе будет сотрудник, поэтому большую часть взаимодействия будет связана с ним. На рисунке 2.2 изображена диаграмма взаимодействия.



Рисунок 2.2 – Диаграмма взаимодействия

Для данной диаграммы ключевым актером будет являться сотрудник, так как чаще всего работать в системе будет именно он. Так же в системе имеется взаимодействие с самой ИС, администратором и клиентом. Сотрудник вносит в систему необходимые данные в системе же они группируются по таблицам и фильтруются по средствам поиска по системе. Администратор может добавлять необходимые детали если таковых нет в наличии и в конечном итоге, сотрудник формирует документ для выдачи клиенту

2.1.5 Построение диаграммы активностей. Построение данной диаграммы заключается в описании динамических аспектов поведения системы. По существу, эта диаграмма представляет собой блок-схему, которая наглядно показывает, как поток управления переходит от одной деятельности к другой. На рисунке 2.3 представлена диаграмма деятельности информационной системы учета заявок по ремонту компьютерной техники.

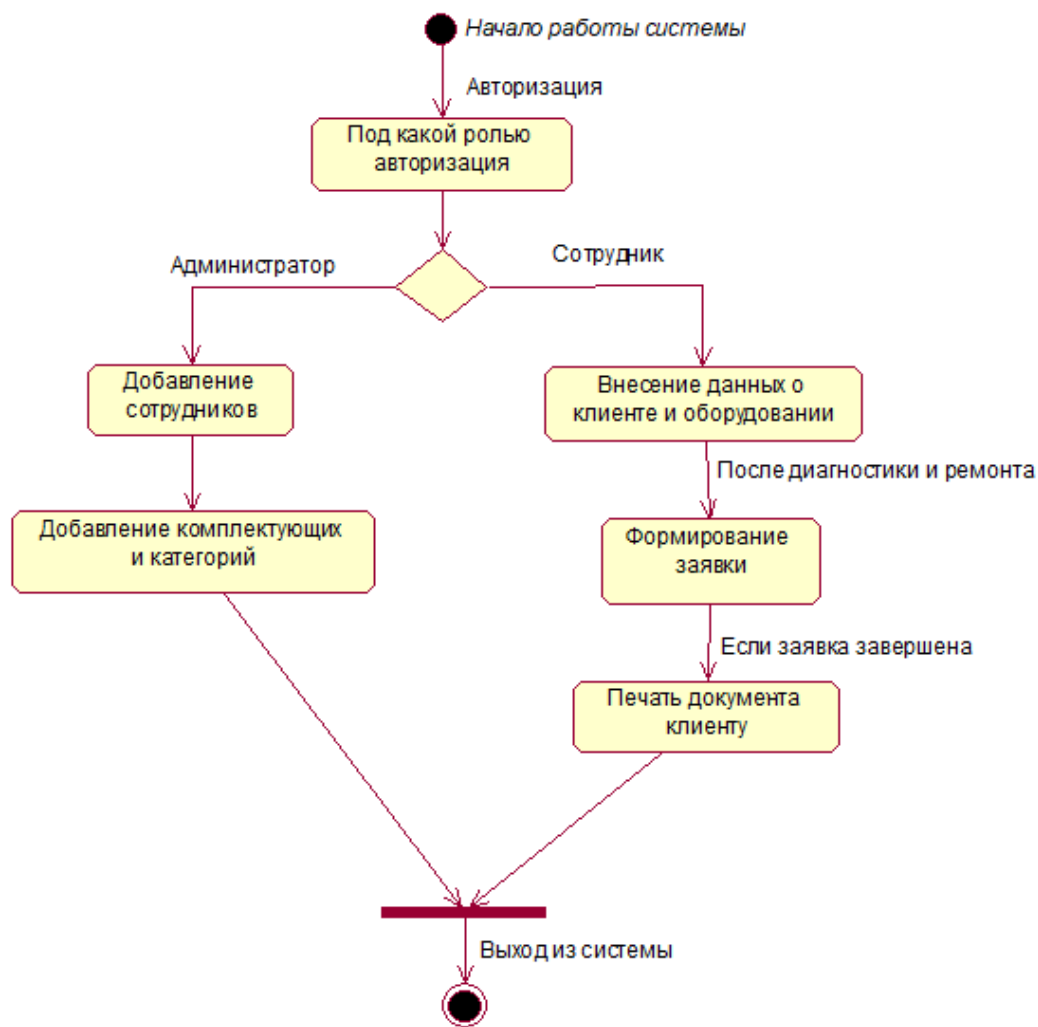


Рисунок 2.3 – Диаграмма деятельности информационной системы

Поскольку в проектируемой системе разделение будет происходить на две роли, а именно сотрудник и администратор, то данная диаграмма так же имеет 2 разветвления и описывает процессы которые выполняет каждая роль.

2.1.6 Построение диаграммы классов. Диаграмма классов является важной частью проектирования информационной системы, так как именно данный вид диаграммы показывает общую структуру иерархии классов будущей системы, а также их атрибуты, методы и взаимодействие друг с другом по средствам интерфейса или взаимодействий. На рисунке 2.4 представлена диаграмма классов информационной системы учета заявок по ремонту компьютерной техники.

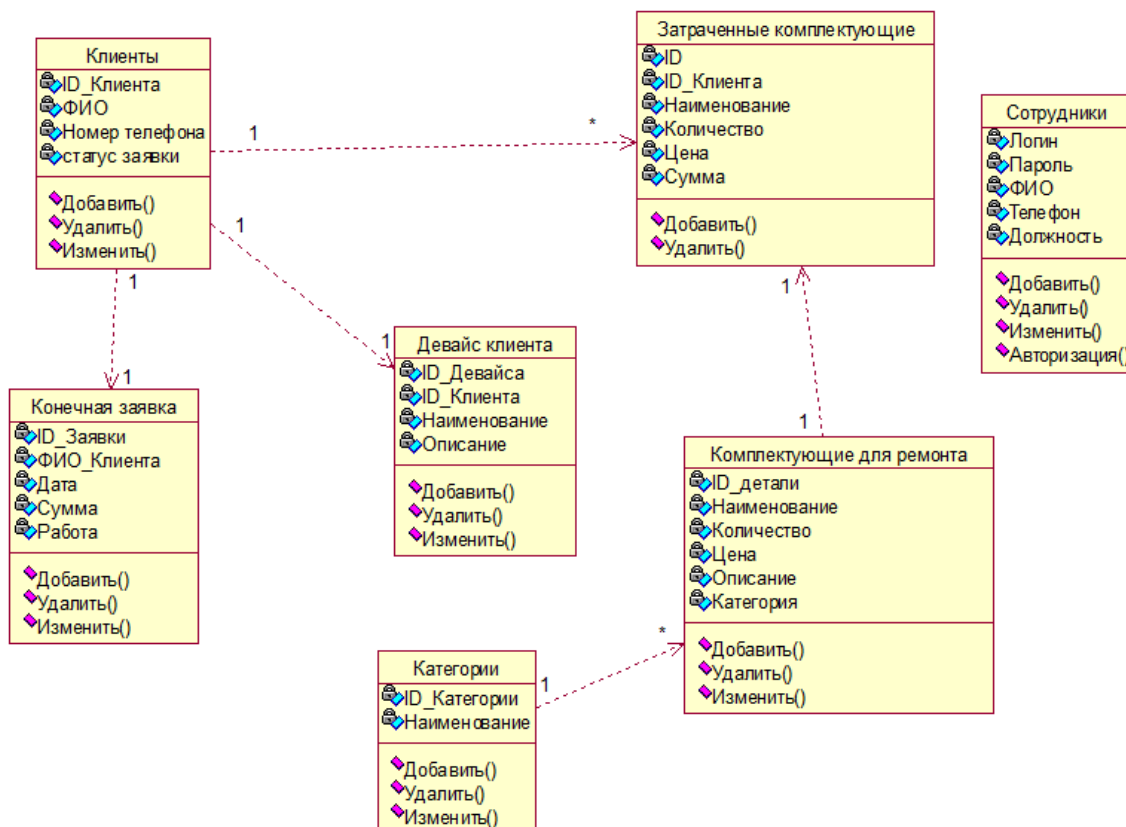


Рисунок 2.4 – Диаграмма классов

На данной диаграмме продемонстрированы 7 классов будущей системы. Основная часть взаимосвязей между классами будет один к одному, но также есть связи один ко многим. Все таблицы классов связаны друг с другом, кроме таблицы «сотрудники» так как она служит для входа в систему.

2.1.7 Построение диаграммы компонентов. Создание диаграммы компонентов является важным шагом, так как именно эта диаграмма описывает особенности физического представления системы, а также будущую архитектуру разрабатываемой системы. Диаграмма компонентов обеспечивает согласованный переход от логического представления к конкретной реализации проекта в виде программного кода, а также отображает общие зависимости между компонентами. На рисунке 2.5 представлена диаграмма компонентов [20-21].

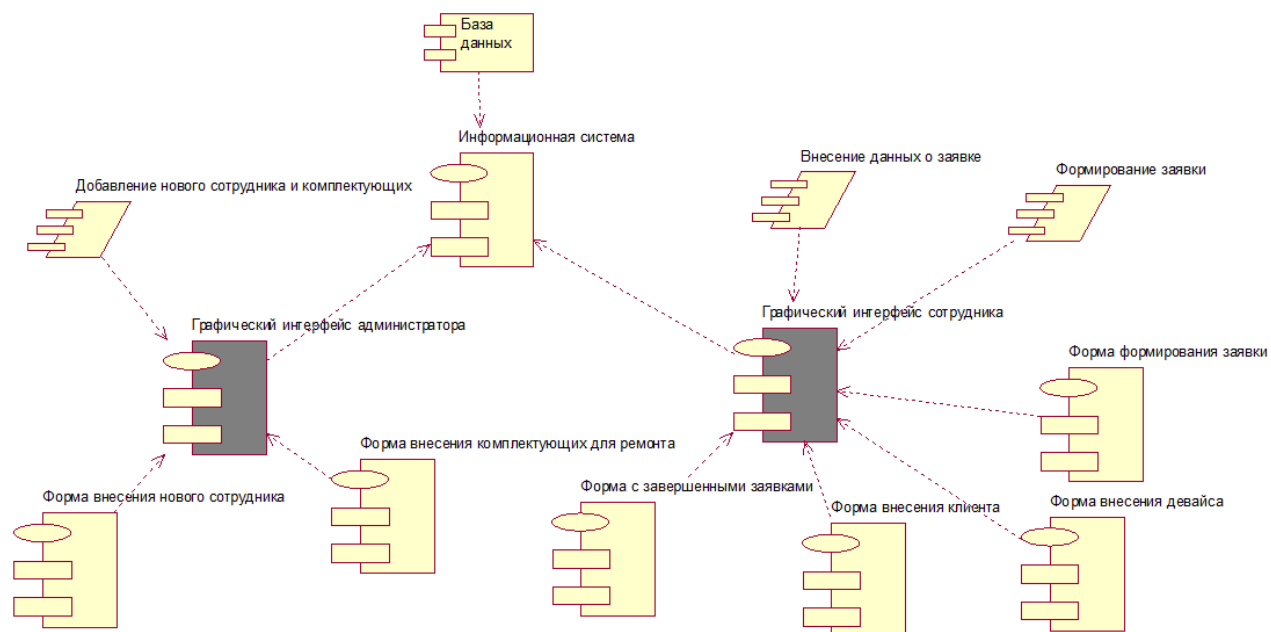


Рисунок 2.5 – Диаграмма компонентов информационной системы учета заявок по ремонту компьютерной техники

В данной диаграмме компонентов информационной системы есть 6 форм:

- 1) Форма внесения нового сотрудника;
- 2) Форма внесения комплектующий для ремонта;
- 3) Форма внесения клиента;
- 4) Форма внесения девайса;
- 5) Форма формирования заявки;
- 6) Форма с конечными заявками.

Сотрудник, использующий данную информационную систему, всегда имеет доступ к необходимой для него информации по средствам вывода её из базы данных на созданный для него графический интерфейс. Для администратора же графический интерфейс будет слегка другим и иметь дополнительные возможности.

2.1.8 Построение диаграммы размещения. Последняя диаграмма будет являться диаграммой размещения. Суть данной диаграмм в демонстрации взаимосвязи между программными и аппаратными компонентами системы. Диаграмма размещения показывает топологию системы и распределение компонентов системы по ее узлам, а также соединения - маршруты передачи информации между аппаратными узлами. На рисунке 2.6 представлена диаграмма размещения информационной системы.

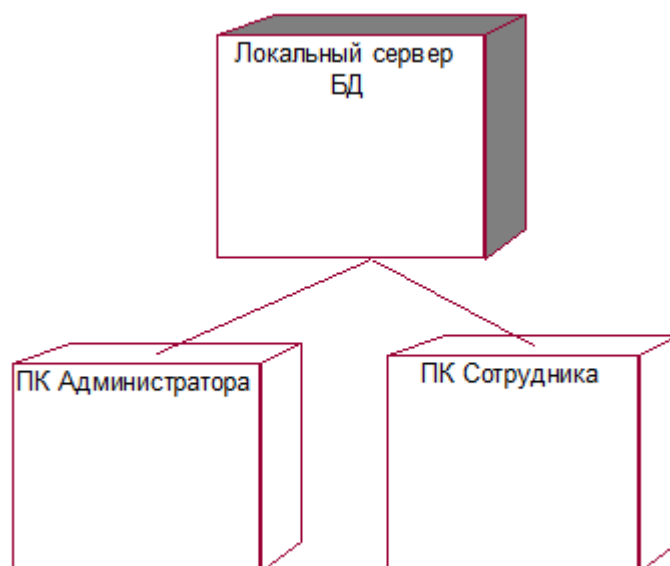


Рисунок 2.6 – Диаграмма размещения

База данных хранится на каком-либо локальном сервере и представляет доступ по средствам подключения к нему ПК пользователей. Например, база данных может храниться на главном компьютере и по локальной сети подключаться к другим ПК сотрудников.

2.2 Проектирование базы данных

2.2.1 Основные понятия и выбор СУБД. Перед созданием информационной системы необходимо разработать базу данных с которой будет происходить работа. База данных (БД) – это совокупность данных, включающая в себя ряд определенных правил, принципов хранения, описания и методы управления данными. База данных управляется специальной системой управления базой данных (СУБД).

На данный момент существует множество типов баз данных, самые популярные из них:

- 1) Реляционные. Данные в таких БД хранятся в виде таблиц со строками и столбцами, данный вид БД обеспечивает наиболее эффективный доступ к систематизированным данным;
- 2) Хранилище данных. В основном такой тип БД создается и используется для быстрого анализа данных и выполнения запросов;
- 3) Объектно-ориентированные. Данный тип БД использует принцип хорошо знакомого ООП и данные хранятся в виде объекта;
- 4) Иерархические. В данном типе БД данные хранятся в виде древовидной структуры.

Существует еще около десятка различных видов БД, но мы остановимся на реляционных, так как большинство современных СУБД построены именно на данной модели данных [22].

В реляционных базах данных, для осуществления обработки данных, контроля данных, обработки данных и т.д. используется язык программирования SQL.

СУБД (система управления базами данных) так же по-другому можно назвать программное обеспечение базы данных, которое используется для обслуживания, создания, редактирования записей БД. Разработка БД в системе управления базами данных, позволяет упростить процесс создания и записи файлов, вводимых, редактирование, обновление и отчетность. СУБД так же служит для хранения данных, осуществляет резервное копирование, осуществляет безопасность данных, что является очень важным фактором в современном мире. Почти у всех современных СУБД имеется многофункциональный графический интерфейс, при помощи которого создание БД упрощается и повышает эффективность.

В качестве СУБД для создания реляционной базы данных была выбрана система управления MSSQL Server. Естественно, существует множество других систем для управления и создания баз данных, но именно данный выбор является оптимальным по ряду причин [23]. Разберем основные из них:

- 1) масштабируемость. Возможность работы как на персональном компьютере, так и на мощной мультипроцессорной технике;
- 2) продуманный функционал и настройка системы;
- 3) максимальный размер базы данных 16 ТБ, а максимальный размер таблицы 532 ГБ;
- 4) высокая производительность. Каждый год компания Microsoft проводит транзакционные тесты и тесты на производительность, которыми подтверждает высокие значения производительности SQL Server;
- 5) поддержка постоянной памяти (PMEM). Данная технология позволяет ускорять запросы на 30%;
- 6) гибкость в выборе платформы и языка программирования [24].

Исходя из данных плюсов можно сделать вывод, что выбранная СУБД будет сполна удовлетворять всем требованиям при разработке БД и дальнейшей разработке информационной системы.

2.2.2 Проектирование базы данных в MSSQL Server. Процесс администрирования и создания основных элементов базы данных будет происходить в среде разработки Visual Studio 2019. Плюсом выбора данной среды является еще то, что в ней происходит разработка и самой ИС, а это значит, что при работе с БД нет необходимости покидать среду разработки. Проект базы данных Visual Studio.NET предназначен для разработки, тестирования и выполнения сценариев и запросов SQL, а также для управления ими. На рисунке 2.7 представлена структура базы данных [25].

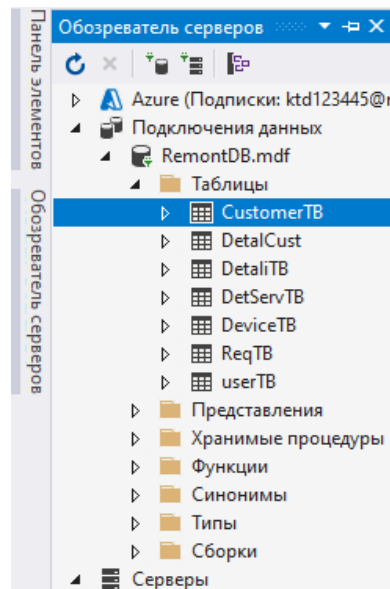


Рисунок 2.7 – Структура базы данных в Visual Studio

Теперь построим диаграмму физической модели базы данных, на которой показаны связи в таблицах (рисунок 2.8).

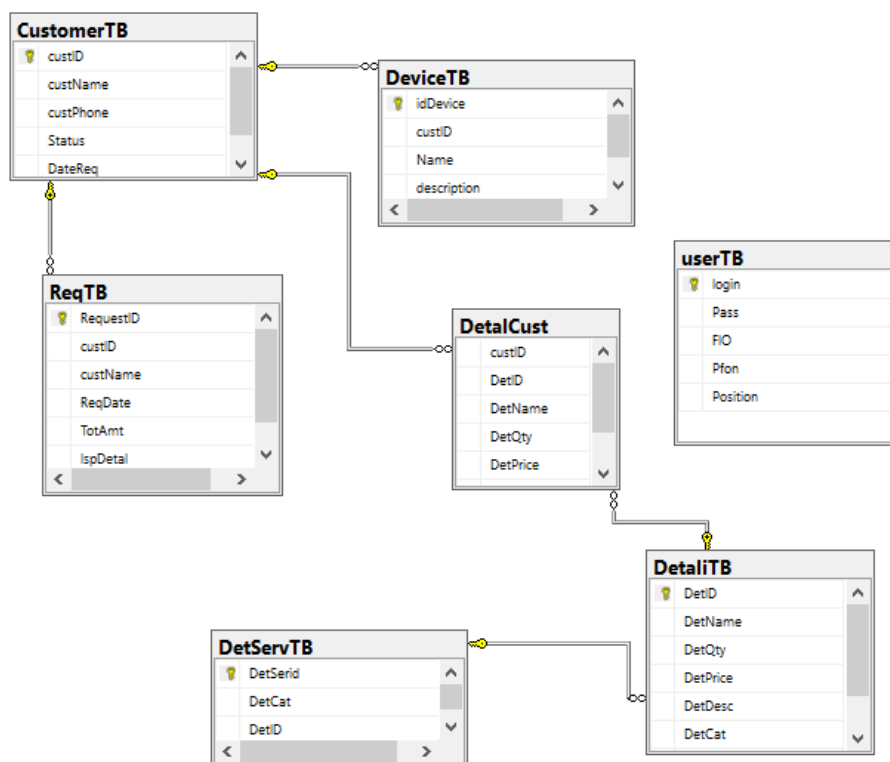


Рисунок 2.8 –Диаграмма базы данных

Разберем внутреннюю структуру каждой таблицы созданной базы данных. Таким образом все таблицы базы данных представлены в таблицах 2.1-2.7.

Таблица 2.1 – Таблица «Категории»

Имя поля	Ключ	Тип	Длина	Возможность использования NULL	Дополнительно
DetSerid	primary key	int	10	нет	IDENTITY(1,1)
DetCat		nvarchar	50	нет	

Таблица 2.2 – Таблица «Сотрудники»

Имя поля	Ключ	Тип	Длина	Возможность использования NULL	Дополнительно
login	primary key	nvarchar	50	нет	IDENTITY(1,1)
Pass		nvarchar	50	нет	
Pfon		int	11		
Position		nvarchar	50		

Таблица 2.3 – Таблица «Девайс»

Имя поля	Ключ	Тип	Длина	Возможность использования NULL	Дополнительно
idDevice	primary key	int	10	нет	IDENTITY(1,1)
custID	foreign key	int	10	нет	
Name		nvarchar	50	нет	
description		nvarchar	150	нет	

Таблица 2.4 – Таблица «Завершенные заявки»

Имя поля	Ключ	Тип	Длина	Возможность использования NULL	Дополнительно
RequestID	primary key	int	10	нет	IDENTITY(1,1)
custID	foreign key	int	10	нет	
CustName		nvarchar	50	нет	
ReqDate		datetime		нет	
TotAmt		int		нет	
Work		nvarchar	150	нет	

Таблица 2.5 – Таблица «Клиенты»

Имя поля	Ключ	Тип	Длина	Возможность использования NULL	Дополнительно
custID	primary key	int	10	нет	IDENTITY(1,1)
custName		nvarchar	50	нет	
custPhone		nvarchar	50	нет	
Status		nvarchar	50	нет	

Таблица 2.6 – Таблица «Затраченные детали»

Имя поля	Ключ	Тип	Длина	Возможность использования NULL	Дополнительно
custID	foreign key	int	10	нет	
DetID	foreign key	int	10	нет	
DetName		nvarchar		нет	
DetQty		int	50	нет	
DetPrice		int	50	нет	
Summ		int	50	нет	
id	primary key		10	нет	IDENTITY(1,1)

Таблица 2.7 – Таблица «Комплектующие для ремонта»

Имя поля	Ключ	Тип	Длина	Возможность использования NULL	Дополнительно
DetID	primary key	int	10	нет	IDENTITY(1,1)
DetName		nvarchar	50	нет	
DetQty		int	50	нет	
DetPrice		int	50	нет	
DetDesc		nvarchar	50	нет	
DetCat	foreign key	nvarchar	50	нет	

2.2.3 SQL-запросы. SQL-запрос – это специальное строковое обращение к базе данных, которое с помощью определенных SQL команд выполняет манипуляции с базой данных. Основные SQL команды: CREATE TABLE, SELECT, INSERT, DELETE, UPDATE, DROP TABLE и другие.

При создании таблиц в базе данных используют команду CREATE TABLE. Создадим описанные выше 7 таблиц. SQL-запросы создания таблиц представлены на рисунках 2.9 – 2.15.

```
CREATE TABLE [dbo].[userTB] (
    [login] NVARCHAR (50) NOT NULL,
    [Pass] NVARCHAR (50) NOT NULL,
    [FIO] NVARCHAR (50) NOT NULL,
    [Pfon] VARCHAR (50) NOT NULL,
    [Position] NVARCHAR (50) NOT NULL,
    CONSTRAINT [PK_Table] PRIMARY KEY CLUSTERED ([login] ASC)
);
```

Рисунок 2.9 – SQL-запрос создания таблицы «Сотрудники»

```
CREATE TABLE [dbo].[DetServTB] (
    [DetSerId] INT NOT NULL,
    [DetCat] NCHAR (50) NOT NULL,
    PRIMARY KEY CLUSTERED ([DetSerId] ASC)
);
```

Рисунок 2.10 – SQL-запрос создания таблицы «Категории»

```
CREATE TABLE [dbo].[CustomerTB] (
    [custID] INT IDENTITY (1, 1) NOT NULL,
    [custName] NVARCHAR (50) NOT NULL,
    [custPhone] NVARCHAR (50) NOT NULL,
    [Status] NVARCHAR (50) NULL,
    PRIMARY KEY CLUSTERED ([custID] ASC)
);
```

Рисунок 2.11 – SQL-запрос создания таблицы «Клиенты»

```
CREATE TABLE [dbo].[DetalCust] (
    [custID] INT NOT NULL,
    [DetID] INT NOT NULL,
    [DetName] NVARCHAR (50) NOT NULL,
    [DetQty] INT NOT NULL,
    [DetPrice] INT NOT NULL,
    [Summ] INT NOT NULL,
    [id] INT IDENTITY (1, 1) NOT NULL
);
```

Рисунок 2.12 – SQL-запрос создания таблицы «Затраченные детали»

```
CREATE TABLE [dbo].[DeviceTB] (
    [IdDevice] INT NOT NULL,
    [custID] INT NOT NULL,
    [Name] NVARCHAR (50) NOT NULL,
    [description] NVARCHAR (MAX) NOT NULL,
    PRIMARY KEY CLUSTERED ([IdDevice] ASC)
);
```

Рисунок 2.13 – SQL-запрос создания таблицы «Девайс клиента»

```
CREATE TABLE [dbo].[ReqTB] (
    [RequestID] INT IDENTITY (1, 1) NOT NULL,
    [CustID] INT NOT NULL,
    [CustName] NVARCHAR (50) NOT NULL,
    [ReqDate] NVARCHAR (50) NOT NULL,
    [TotAmt] INT NOT NULL,
    [Work] NVARCHAR (150) NULL,
    PRIMARY KEY CLUSTERED ([RequestID] ASC)
);
```

Рисунок 2.14 – SQL-запрос создания таблицы «Завершенные заявки»

```
CREATE TABLE [dbo].[DetailTB] (
    [DetID] INT NOT NULL,
    [DetName] NVARCHAR (50) NOT NULL,
    [DetQty] INT NOT NULL,
    [DetPrice] INT NOT NULL,
    [DetDesc] NVARCHAR (50) NOT NULL,
    [DetCat] NVARCHAR (50) NOT NULL,
    PRIMARY KEY CLUSTERED ([DetID] ASC)
);
```

Рисунок 2.15 – Создание таблицы «Комплектующие»

После создания базы данных необходимо создать SQL-запросы для работы с ней, а именно изменять данные и осуществлять поиск по необходимым значениям. Так же для сделать вывод данных для просмотра пользователю. Первым пунктом создадим запрос на вывод данных из БД, для этого воспользуемся оператором запроса SELECT. Команда для вывода полной информации о комплектующих представлена на рисунке 2.16.

```

Con.Open();
string Myquery = "Select * from DetaliTB";
SqlDataAdapter da = new SqlDataAdapter(Myquery, Con);
SqlCommandBuilder builder = new SqlCommandBuilder(da);
var ds = new DataSet();
da.Fill(ds);
DetaliGV.DataSource = ds.Tables[0];
DetaliGV.Columns[0].HeaderText = "ID";
DetaliGV.Columns[1].HeaderText = "Наименование";
DetaliGV.Columns[2].HeaderText = "Количество";
DetaliGV.Columns[3].HeaderText = "Цена";
DetaliGV.Columns[4].HeaderText = "Описание";
DetaliGV.Columns[5].HeaderText = "Категория";

```

Рисунок 2.16 – SQL-запрос вывода всех данных о комплектующих

Как видно из рисунка 2.16 данные выводятся в среду DataGridView, для которой назначены наименования столбцов идентичные таким же как в БД. На примере данной таблицы разберемся как происходит процесс добавления новой записи в БД по средствам оператора добавления данных INSERT (рисунок 2.17). Для данного запроса необходимо указать таблицу, в которую происходит запрос и затем перечислить поля для заполнения по такому же порядку как в таблице. В качестве данных используется текст, находящийся в поле textbox. В данном запросе можно заметить префикс «N» перед добавлением данных с буквенным значением, это необходимо для передачи значения Unicode и тем самым символы на русском языке добавятся корректно.

```

SqlCommand cmd = new SqlCommand("insert into DetaliTB values('" + ID.Text + "','" +
    "N'" + Dname.Text + "','" + Dqty.Text + "','" + Dprice.Text + "','" + description.Text + "','" + Categor.Text + "','')", Con);
cmd.ExecuteNonQuery();

```

Рисунок 2.17 – SQL-запрос добавления данных в таблицу комплектующие

Таблица с комплектующими создана для администрирования системы для этого необходимо создать запрос для изменения данных о количестве комплектующих или других их значений. Для изменения данных воспользуемся оператором изменения значений в таблице UPDATE (рисунок 2.18). В данном запросе мы, как и ранее указываем наименование таблицы и так же для каждого поля таблицы значения textbox, где вводим информацию для изменения. В данном запросе используется ключевое слово WHERE, которое служит для указания условия, что изменение данных происходит именно для этой записи [26-27].

```

SqlCommand cmd = new SqlCommand("update DetaliTB set DetName=N'" + Dname.Text + "'" +
    ",DetQty='" + Dqty.Text + "',DetPrice='" + Dprice.Text + "',DetDesc=N'" + description.Text + "'" +
    ",DetCat=N'" + Categor.SelectedValue.ToString() + "' where DetID='" + ID.Text + "'", Con);
cmd.ExecuteNonQuery();

```

Рисунок 2.18 – SQL-запрос изменения данных в таблице комплектующие

Необходимо реализовать возможность удаления записи из БД, для данной задачи потребуется обратиться к оператору удаления записи DELETE. Для

данного запроса мы указываем таблицу, а также значение ID для удаляемой записи (рисунок 2.19).

```
string del = "delete from DetaliTB where DetID =" + ID.Text + "';";
```

Рисунок 2.19 – SQL-запрос удаления данных в таблице комплектующие

При большом количестве данных нужно реализовать быстрый поиск по необходимым значениям. Для этого воспользуемся оператором запроса SELECT и теперь для вывода значений в поле таблицы укажем оператор LIKE, который используется в предложении WHERE для поиска указанного шаблона в столбце. Значение же данного оператора будем брать из вводимых данных в поле поиска (рисунок 2.20).

```
SqlDataAdapter da = new SqlDataAdapter("SELECT custID, custName, custPhone, Status" +  
    " FROM CustomerTB WHERE custName LIKE N'" + textBox1.Text + "%' OR" +  
    " custPhone LIKE N'" + textBox1.Text + "%'OR"+  
    " Status LIKE N'" + textBox1.Text + "%'" , Con);
```

Рисунок 2.20 – SQL-запрос на поиск данных

2.3 Разработка алгоритма и приложения

2.3.1 Выбор языка программирования и среды разработки. Для создания информационной системы необходимо выбрать инструментальные средства для её разработки. Итогом данного раздела будет разработанная программа с графическим интерфейсом для работы с ней через персональный компьютер.

Для создания информационной системы первым делом необходимо выбрать язык программирования, на котором будет написан весь функционал и дизайн программы. Решая эту проблему, был выбран язык программирования C# — это очень мощный язык программирования, который в свою очередь является простым для понимания и объектно-ориентированным, что в свою очередь отвечает требованиям современных способов разработки и позволяет разработчикам создавать многофункциональные приложения. К преимуществам данного языка можно так же отнести:

- 1) относится к языкам компилируемого типа;
- 2) объединяет в себе лучшие идеи таких популярных языков программирования как Java, C++, Visual Basic и т.д;
- 3) является типизированным, что значительно облегчает поиск ошибок в коде;
- 4) большое количество библиотек, для решения множества задач;
- 5) большое разнообразие синтаксических конструкций и возможность работы с .NET, что позволяет создавать положения на данном языке быстрее других;

б) надежность и масштабируемость языка.

Перечислены далеко не все плюсы языка C#, но исходя даже из этих пунктов можно сделать вывод, что данный язык программирования является лучшим для создания информационной системы [28].

В качестве среды разработки идеально подойдет интегрированная среда разработки Microsoft Visual Studio 2019. Данная среда разработки спроектирована таким образом, что процессы написания кода, отладка и компиляция, сделаны очень простыми для разработчика. Но это не означает, что в Visual Studio мало функционала и она имеет ограниченный спектр применения, скорее наоборот для выбранного языка программирования C# данная среда является лучшей. Функциональная же структура среды включает в себя:

- 1) отладку кода;
- 2) дизайнер классов;
- 3) редактор исходного кода, в котором присутствует множество дополнительных функций, например, как IntelliSense (технология автодополнения кода);
- 4) редактор форм, созданный для быстрого создания графических интерфейсов;
- 5) возможность проводить тесты благодаря встроенному менеджеру тестов;
- 6) возможность находить ошибки в коде Code Analyzer;
- 7) отображение затрат ресурсов при работе приложения/сервиса в виде статистики и графиков.

Особенностью среды разработки Visual Studio является платформа Windows Forms для создания интерфейса пользователя, включающий в себя функции для разработки приложений, включая элементы управления, графику, привязку данных и ввод пользователя.

2.3.2 Разработка пользовательского интерфейса приложения. Каждое приложение или программа, установленная на компьютере, телефоне или другом девайсе, должна иметь возможность общаться с пользователем, и чтобы сам пользователь мог передавать ей необходимую информацию. Для решения данной задачи разрабатываются специальные пользовательские графические интерфейсы. Пользовательский интерфейс – это все компоненты программы, которые помогают пользователю взаимодействовать с ней, по средствам голоса, нажатия клавиш или ввода данных через командную строку.

Для разработки информационной системы по выбранной предметной области необходимо учитывать специфику предприятия и тонкости работы проектируемой программы. Наша информационная система должна осуществлять учет заявок, поэтому основными компонентами будут поля ввода данных и кнопки для добавления этих данных в БД и в дальнейшем их редактирование или удаление.

Для входа в систему необходимо создать специальное окно авторизации и проверку вводимых данных, а также реализовать меню, по которому пользователь системы сможет ориентироваться по разделам программы для выполнения необходимой задачи.

Начнем разработку приложения с создания загрузочного окна системы, данное окно необходимо для загрузки программы и всех компонентов, чтобы при дальнейшей работе приложение не зависало. На форме загрузки разместим

название приложения, индикатор загрузки и полосу загрузки при достижении значения 100% загрузка завершится (рисунок 2.21).

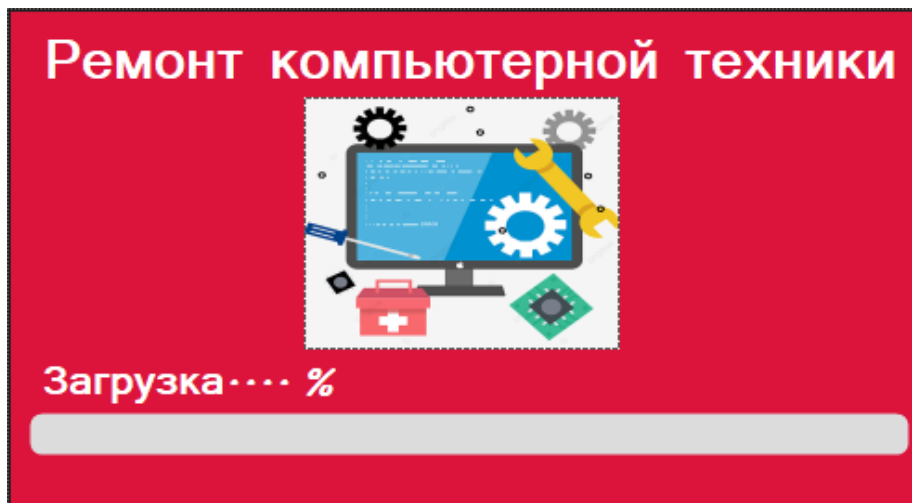


Рисунок 2.21 – Форма загрузки приложения

Для загрузки необходимо при запуске приложения запустить таймер, а также объявить переменную, к которой при каждом тике таймера прибавлять 2 и когда счётчик дойдет до 100 завершить загрузку и перейти к следующей форме, код представлен на рисунке 2.22.

```
int startpoint = 0;

ссылка: 1
private void timer1_Tick(object sender, EventArgs e)
{
    startpoint += 2;
    progress.Value = startpoint;
    label3.Text = startpoint + "%";
    if(progress.Value==100)
    {
        progress.Value = 0;
        timer1.Stop();
        Form1 mainForm = new Form1();
        this.Hide();
        mainForm.Show();
    }
}

ссылка: 1
private void loading_Load(object sender, EventArgs e)
{
    timer1.Start();
}
```

Рисунок 2.22 – Код загрузки приложения

Следующим этапом необходимо создать форму для авторизации. На данной форме будет размещены поля для ввода логина и пароля, а также роли под которой

пользователь входит в систему, после ввода данных нужно нажать на кнопку для входа (рисунок 2.23).

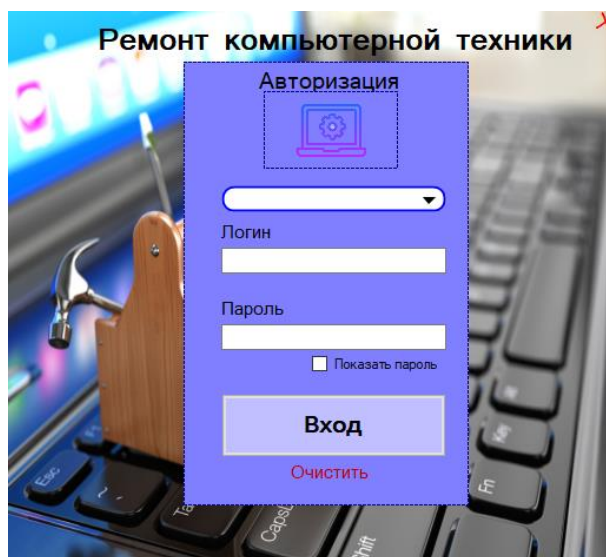


Рисунок 2.23 – Форма авторизации в систему

При входе в систему необходимо проверить поля на корректные данные и являются ли логин с паролём правильными для входящей роли, код кнопки «Вход» представлен на рисунке 2.24 Логин и пароль для администратора системы задан в коде самой программы, а для сотрудника находится в базе данных и может быть изменен администратором.

```
private void button1_Click(object sender, EventArgs e)
{
    if (access.Text == "" || Login.Text == "" || Password.Text == "")
    {
        MessageBox.Show(
            "Заполните все поля ",
            "Сообщение",
            MessageBoxButtons.OK,
            MessageBoxIcon.Information);
    }
    if (access.Text == "Сотрудник")
    {
        Data.Lvl += 2;
        Con.Open();
        SqlDataAdapter sda = new SqlDataAdapter("select Count(*) from userTB where login = " +
            Login.Text + " and Passw = " + Password.Text + " and Position= N" + access.Text + "", Con);
        DataTable dt = new DataTable();
        sda.Fill(dt);
        if (dt.Rows[0][0].ToString() == "1")
        {
            MainForm mainform = new MainForm();
            mainform.Show();
            this.Hide();
        }
        else
        {
            MessageBox.Show("Ошибка ввода логина или пароля ");
            Data.Lvl = null;
            Con.Close();
        }
    }
    if (access.Text == "Администратор")
    {
        if (Login.Text == "Sport23" && Password.Text == "124589")
        {
            Data.Lvl += 1;
            MainForm mainform = new MainForm();
            mainform.Show();
            this.Hide();
        }
        else
        {
            MessageBox.Show("Ошибка ввода логина или пароля ");
            Data.Lvl = null;
        }
    }
}
```

Рисунок 2.24 – Код кнопки «Вход» для авторизации

После успешной авторизации пользователь перейдет к форме «Меню». На данной форме необходимо разместить все разделы системы (рисунок 2.25).

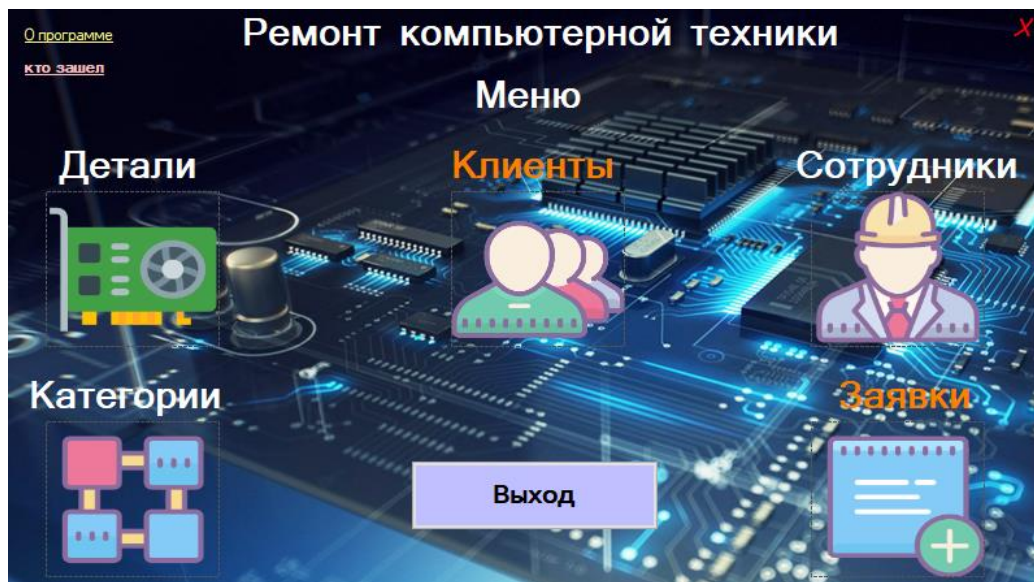


Рисунок 2.25 – Форма главного меню приложения

В верхнем левом углу главного меню имеются надписи «О программе» для краткой информации данной системы и текстовый индикатор, который изменится в зависимости того кто зашел в систему, а именно на «Вы вошли как сотрудник» и «Вы вошли как администратор» так же при входе как сотрудник будут недоступны некоторые разделы и функции системы. В самой программе была объявлена глобальная переменная, к которой при входе администратора прибавляется единица, а при входе сотрудника двойка в дальнейшем в программе задаются условия некоторых функций исходя из значения этой переменной. На форме главного меню не так много функционала, как на остальных, основными элементами являются картинки под надписями, при нажатии на которые происходит переход в соответствующий раздел, на рисунке 2.26 представлен код при нажатии на кнопку «Сотрудники»

```
private void pictureBox2_Click(object sender, EventArgs e)
{
    if (Data.LvL == "2")
    {
        MessageBox.Show("У вас нет доступа к этому разделу");
    }
    else
    {
        Users users = new Users();
        users.Show();
        this.Hide();
    }
}
```

Рисунок 2.26 – Код при нажатии на картинку «Сотрудники»

Как видно из рисунка 2.26 если переменная «LvL» равна 2, то есть в системе был авторизован сотрудник, переход в форму «Сотрудники» не будет осуществлен и появится соответствующее окно об ошибке.

Перейдём теперь к созданию основных форм для работы нашей информационной системы, а именно создадим форму с клиентами (рисунок 2.27). На данной форме происходит вывод всех клиентов из базы данных в поле «DataGridView», добавление новых клиентов, удаление клиента и изменение данных. Так же строка поиска по данным и три блока для отображения информации о уже сформированной заявки клиента и индикатор внесённого девайса клиента.

Рисунок 2.27 – Форма для внесения клиентов

Разберем код кнопок для управления данной формы, а именно кнопку «Добавить» при нажатии на которую происходит добавление новой записи в БД по заполненным и данная запись отображается в таблице, код данной процедуры изображён на рисунке 2.28. Основная задача при нажатии проверить заполненные поля и внести запись в БД по средствам SQL команды.

```

private void button1_Click(object sender, EventArgs e)
{
    if (CustName.Text == null || CustName.Text == "")
        MessageBox.Show(
            "Заполните поле ФИО",
            "Сообщение",
            MessageBoxButtons.OK,
            MessageBoxIcon.Information);
    else if (CustPhone.Text == null || CustPhone.Text == "")
        MessageBox.Show(
            "Заполните поле номер телефона ",
            "Сообщение",
            MessageBoxButtons.OK,
            MessageBoxIcon.Information);
    else if (Statys.Text == null || Statys.Text == "")
        MessageBox.Show(
            "Укажите статус заявки",
            "Сообщение",
            MessageBoxButtons.OK,
            MessageBoxIcon.Information);
    else
    {
        try
        {
            Con.Open();
            SqlCommand cmd = new SqlCommand("insert into CustomerTB values (N'" + CustName.Text + "','" +
                "" + CustPhone.Text + "','" + Statys.Text + "')", Con);
            cmd.ExecuteNonQuery();
            MessageBox.Show("Клиент добавлен");
            Con.Close();
            populate();
        }
        catch (SqlException)
        {
            MessageBox.Show("Ошибка");
            Con.Close();
        }
    }
}

```

Рисунок 2.28 – Код кнопки «Добавить» на форме «Клиенты»

На данной форме нужно так же реализовать удаление клиента выбрав его в таблице и нажав на соответствующую кнопку, так же будут удалены записи, относящиеся к клиенту, код кнопки представлен на рисунке 2.29.

```

private void button3_Click(object sender, EventArgs e)
{
    if (CustID.Text == "")
    {
        MessageBox.Show("Выберите ID клиента");
    }
    else
    {
        Con.Open();
        string del = "delete from CustomerTB where custID='" + CustID.Text + "'";
        SqlCommand cmd = new SqlCommand(del, Con);
        cmd.ExecuteNonQuery();
        MessageBox.Show("Запись удалена ");
        Con.Close();
        DeleteDevice();
        DeleteZaivca();
        populate();
    }
}

```

Рисунок 2.29 – Код кнопки «Удалить» на форме «Клиенты»

Если данные были внесены неверные и для изменения статуса заявки необходимо реализовать кнопку «Изменить», код для этой кнопки представлен на рисунке 2.30.


```

private void button2_Click(object sender, EventArgs e)
{
    if (CustName.Text == null || CustName.Text == "")
    {
        MessageBox.Show(
            "Заполните поле ФИО",
            "Сообщение",
            MessageBoxButtons.OK,
            MessageBoxIcon.Information);
    }
    else if (CustPhone.Text == null || CustPhone.Text == "")
    {
        MessageBox.Show(
            "Заполните поле номер телефона ",
            "Сообщение",
            MessageBoxButtons.OK,
            MessageBoxIcon.Information);
    }
    else if (Statys.Text == null || Statys.Text == "")
    {
        MessageBox.Show(
            "Укажите статус заявки",
            "Сообщение",
            MessageBoxButtons.OK,
            MessageBoxIcon.Information);
    }
    else
    {
        try
        {
            Con.Open();
            SqlCommand cmd = new SqlCommand("update CustomerTB set custName = " +
                "N'" + CustName.Text + "', custPhone=N'" + CustPhone.Text + "', Status=" +
                "N'" + Statys.Text + "' where custID=" + CustID.Text + "'", Con);
            cmd.ExecuteNonQuery();
            MessageBox.Show("Данные изменены");
            Con.Close();
            populate();
        }
        catch (SqlException)
        {
            MessageBox.Show("Ошибка");
            Con.Close();
        }
    }
}

```

Рисунок 2.30 – Код кнопки «Изменить» на форме «Клиенты»

При нажатии на клиента в таблице необходимо автоматически заполнять поля вводы имеющимися данными, так же при помощи запроса проверять внесён ли девайс клиента и если он есть изменить текст иконки девайса на зелёный, а если нет, то на красный, так же выводить данные в созданные блоки с номером заявки, суммы для оплаты и даты формирования конечной заявки, если заявка имеется, программный код данного функционала представлен на рисунке 2.31.

```

private void CostGV_CellContentClick(object sender, DataGridViewCellEventArgs e)
{
    CustID.Text = CostGV.CurrentRow.Cells[0].Value.ToString();
    CustName.Text = CostGV.CurrentRow.Cells[1].Value.ToString();
    CustPhone.Text = CostGV.CurrentRow.Cells[2].Value.ToString();
    Statys.Text = CostGV.CurrentRow.Cells[3].Value.ToString();
    Con.Open();

    SqlDataAdapter sda4 = new SqlDataAdapter("select Count(*) from DeviceTB where IdKlient = " +
        "N'" + CustID.Text + "'", Con);
    DataTable dt4 = new DataTable();
    sda4.Fill(dt4);
    if (dt4.Rows[0][0].ToString() == "1")
    {
        pictureBox1.BackColor = Color.GreenYellow;
        DevaisEst.Text = "Девайс внесён";
    }
    else
    {
        pictureBox1.BackColor = Color.Red;
        DevaisEst.Text = "Внести девайс";
    }
}

SqlDataAdapter sda = new SqlDataAdapter("select Min(RequestID) from ReqTB where CustID= " + CustID.Text + "'", Con);
DataTable dt = new DataTable();
sda.Fill(dt);
//RequestID
ReqLab.Text = dt.Rows[0][0].ToString();

SqlDataAdapter sda1 = new SqlDataAdapter("select Sum(TotAmt) from ReqTB where CustID= " + CustID.Text + "'", Con);
DataTable dt1 = new DataTable();
sda1.Fill(dt1);
Summa.Text = dt1.Rows[0][0].ToString();

SqlDataAdapter sda2 = new SqlDataAdapter("select max(ReqDate) from ReqTB where CustID= " + CustID.Text + "'", Con);
DataTable dt2 = new DataTable();
sda2.Fill(dt2);
DataLab.Text = dt2.Rows[0][0].ToString();
Con.Close();
}

```

Рисунок 2.31 – Код при нажатии за запись в таблице

После внесения данных о клиенте нужно добавить девайс, который будет проходить для ремонта, для этого создадим событие при нажатии на иконку девайса на форме «Клиенты» появится новая форма для внесения девайса клиента (рисунок 2.32)

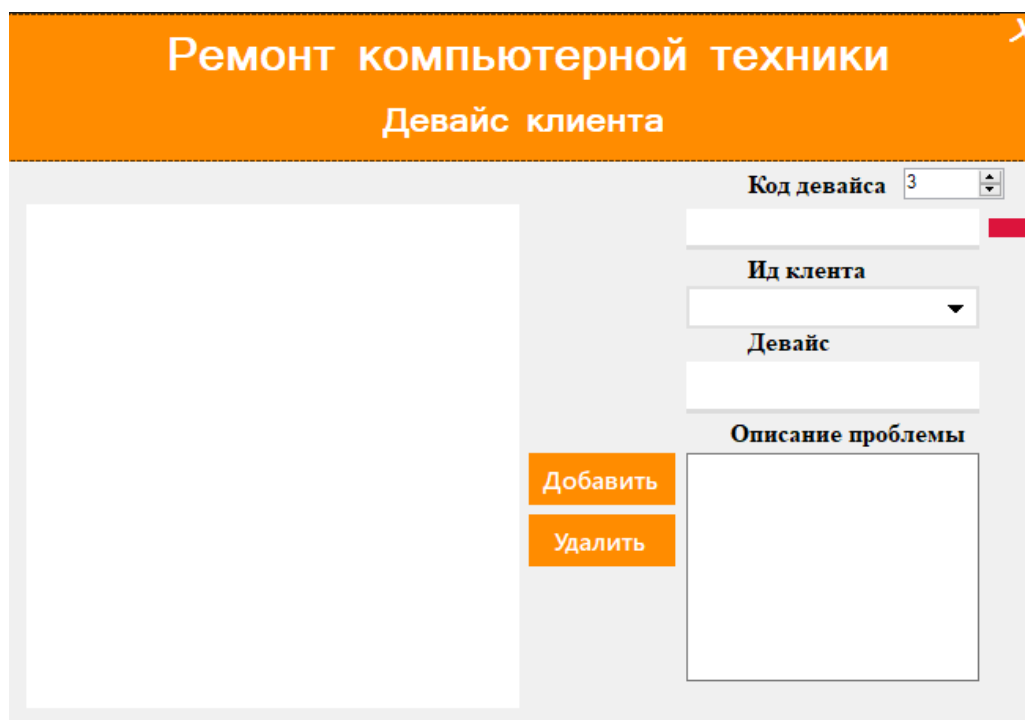


Рисунок 2.32 – Создание формы «Девайс клиента»

На данной форме можно сгенерировать код для девайса, выбрать из выпадающего списка ид клиента и привязать именно к нему внесённый девайс. Функционал кнопок и других действий на данной форме не отличается от описанной ранее, поэтому рассмотрение кода не имеет смысл, поменяются только SQL запросы. После внесения клиента и его девайса сотрудник проведет диагностику и определит какую деталь необходимо заменить или провести ремонт, для этого нужно создать отдельную форму на которой будет внесены детали для ремонта и по ним сформируется сумма для оплаты (рисунок 33). На форме будут отображаться таблицы с клиентами и фильтры по новым заявкам и категориям комплектующих, также комплектующие находящиеся на складе в данный момент, выбрав деталь из таблицы и внося количество деталей в соответствующее поле, деталь будет добавлена в нижнюю таблицу и сформируется сумма, при внесении не той детали её можно убрать. После заполнения всех полей заявку можно добавить в таблицу к завершённым заявкам.

Рисунок 2.33 – Создание формы «Заявки»

При добавлении детали в таблицу необходимо проверить поле с количеством на корректные данные, а также имеется ли такое количество деталей, далее произвести необходимый расчет суммы деталей и конечной суммы по таблице, код данной операции представлен на рисунке 2.33.

```
private void button1_Click(object sender, EventArgs e)
{
    if (flag == 0)
        MessageBox.Show("Выберите деталь");
    else if (qtTb.Text == "")
        MessageBox.Show("Введите количество ");
    else if (qtTb.Text == "0")
        MessageBox.Show("Никаких нулей ");
    else if (Convert.ToInt32(qtTb.Text) < 0)
        MessageBox.Show("Ошибка записи");

    else if (Convert.ToInt32(qtTb.Text) > stock)
        MessageBox.Show("недостаточно деталей");
    else
    {
        num = Convert.ToInt32(DetalGV.CurrentRow.Cells[0].Value.ToString());
        qty = Convert.ToInt32(qtTb.Text);
        totprice = qty * uprice;
        Con.Open();
        SqlCommand cmd = new SqlCommand("insert into DetalCust values" +
            "(" + ClientID.Text + "," +
            "" + num + ",N" + detal + "," + qtTb.Text + "," + uprice + "," + totprice + ")", Con);
        cmd.ExecuteNonQuery();
        Con.Close();
        MessageBox.Show("Запись добавлена ");

        ReqGV.DataSource = table;
        flag = 0;
        updateDetali();
        qtTb.Text = "";
        showCustDetal();
    }
}
```

Рисунок 2.33 – Код кнопки «Добавить деталь»

После заполнения всех необходимых полей нужно добавить заявку к таблице БД, для этого необходимо проверить поля на корректные данные и не внесена ли

уже такая заявка в БД и если всё прошло успешно, то заявка добавится, код данной операции представлен на рисунке 2.34.

```
private void button2_Click(object sender, EventArgs e)
{
    if (ClientID.Text == "" || clientName.Text == "" || TotAm.Text == "")
    {
        MessageBox.Show("Не все поля заполнены");
    }
    else if (Work.Text == "" || Work.Text == null)
    {
        MessageBox.Show("Пожалуйста заполните поле Описание работы ");
    }

    else if (dateReq.Value < DateTime.Today || dateReq.Value > DateTime.Today)
    {
        MessageBox.Show("Ошибка в дате");
        dateReq.Value = DateTime.Today;
    }
    else
    {
        Con.Open();
        SqlDataAdapter sda = new SqlDataAdapter("select Count(*) from ReqTB where CustID = " +
            ClientID.Text + " ", Con);
        DataTable dt = new DataTable();
        sda.Fill(dt);
        if (dt.Rows[0][0].ToString() == "1")
        {
            MessageBox.Show("У этого клиента уже есть заявка");
            Con.Close();
        }
        else
        {
            try
            {
                SqlCommand cmd = new SqlCommand("insert into ReqTB values" +
                    "(" + ClientID.Text + "," +
                    ClientName.Text + "," + dateReq.Text + "," + TotAm.Text + "," + Work.Text + ")", Con);
                cmd.ExecuteNonQuery();
                MessageBox.Show("Заявка добавлена ");
                Con.Close();
                clientName.Text = "";
                ClientID.Text = "";
                TotAm.Text = "";
                Work.Text = "";
                int rowsCount = ReqGV.Rows.Count;
                for (int i = 0; i < rowsCount; i++)
                {
                    ReqGV.Rows.Remove(ReqGV.Rows[i]);
                }
            }
            catch (SqlException)
            {
                MessageBox.Show("Заявка с таким ID уже есть");
                Con.Close();
            }
        }
    }
}
```

Рисунок 2.34 – Код кнопки «Добавить»

После добавления заявки нужно создать форму для отображения уже завершённых заявок, так же реализовать экспорт данных в среду Excel и при двойном клике печатать документ для выдачи клиенту еще строить график по доходу всех заявок.

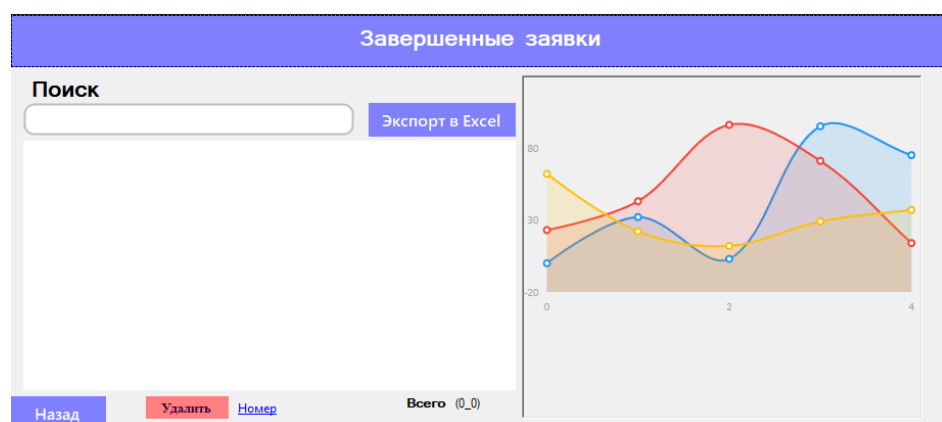


Рисунок 2.35 – Форма с завершёнными заявками

И так на данной форме из новых функций появится кнопка «Экспорт в Excel», а также печать документа при двойном клике на запись в таблице, код для этих двух операций представлен на рисунках 2.36 и 2.37.

```
private void Excel_Click(object sender, EventArgs e)
{
    int kol = ViewGV.Rows.Count;
    if (kol != 0)
    {
        Microsoft.Office.Interop.Excel.Application ExcelApp =
            new Microsoft.Office.Interop.Excel.Application();
        Microsoft.Office.Interop.Excel.Workbook ExcelWorkBook;
        Microsoft.Office.Interop.Excel.Worksheet ExcelWorkSheet;
        //Книга.
        ExcelWorkBook = ExcelApp.Workbooks.Add(System.Reflection.Missing.Value);
        //Таблица.
        ExcelWorkSheet = (Microsoft.Office.Interop.Excel.Worksheet)ExcelWorkBook.Worksheets.get_Item(1);
        for (int i = 0; i < ViewGV.ColumnCount; i++)
        {
            ExcelApp.Cells[1, i + 1] = Convert.ToString(ViewGV.Columns[i].HeaderText);
        }
        for (int i = 0; i < ViewGV.Rows.Count; i++)
        {
            for (int j = 0; j < ViewGV.ColumnCount; j++)
            {
                ExcelApp.Cells[i + 2, j + 1] = Convert.ToString(ViewGV.Rows[i].Cells[j].Value);
            }
        }
        //Вызываем приложение Excel.
        ExcelApp.Visible = true;
        ExcelApp.UserControl = true;
    }
    else
    {
        MessageBox.Show("Для импорта данных из таблицы в Excel для начало заполните таблицу данными!", "Импорт данных из таблицы в Excel");
    }
}
```

Рисунок 2.36 – Код кнопки «Экспорт в Excel»

```
private void printDocument1_PrintPage(object sender, System.Drawing.Printing.PrintPageEventArgs e)
{
    e.Graphics.DrawString("Отчет заявки: " + ViewGV.CurrentRow.Cells[0].Value.ToString(), new Font("Century", 25, FontStyle.Bold), Brushes.Red, new Point(230));
    e.Graphics.DrawString("Номер клиента: " + ViewGV.CurrentRow.Cells[1].Value.ToString(), new Font("Century", 20, FontStyle.Regular), Brushes.Black, new Point(80, 85));
    e.Graphics.DrawString("ФИО клиента: " + ViewGV.CurrentRow.Cells[2].Value.ToString(), new Font("Century", 20, FontStyle.Regular), Brushes.Black, new Point(80, 115));
    e.Graphics.DrawString("Дата обращения: " + ViewGV.CurrentRow.Cells[3].Value.ToString(), new Font("Century", 20, FontStyle.Regular), Brushes.Black, new Point(80, 150));
    e.Graphics.DrawString("К оплате: " + ViewGV.CurrentRow.Cells[4].Value.ToString(), new Font("Century", 20, FontStyle.Regular), Brushes.Blue, new Point(80, 195));
    e.Graphics.DrawString("Дата выдачи: " + this.Date, new Font("Century", 20, FontStyle.Regular), Brushes.Green, new Point(180, 240));
    e.Graphics.DrawString("Продоланная работа: " + ViewGV.CurrentRow.Cells[5].Value.ToString(), new Font("Century", 20, FontStyle.Regular), Brushes.Black, new Point(10, 280));
    e.Graphics.DrawString("подпись", new Font("Century", 20, FontStyle.Regular), Brushes.DarkBlue, new Point(80, 320));
    e.Graphics.DrawString("подпись", new Font("Century", 20, FontStyle.Regular), Brushes.Black, new Point(130, 360));
}
```

Рисунок 2.37 – Код печати документа

Для администрирования системы необходимо добавить еще форму для внесения новых категорий комплектующих или услуг в систему, для возможного расширения деятельности сервисного центра. Данная форма не будет обладать новым функционалом, поэтому просто рассмотрим её на рисунке 2.38.

Ремонт компьютерной техники

Комплектующие и услуги

ID категории

Наименование

[Очистить](#)

Добавить Изменить

Удалить

Меню

Рисунок 2.38 – Форма для добавления категории и услуги

После внесения категории нужно добавить комплектующие или услуги для этой категории, для этого создадим форму с деталями (рисунок 2.39).

Ремонт компьютерной техники

Детали

ID детали

Наименование

Количество

Цена

Описание

Добавить Изменить

Удалить

Главная

Экспорт в Excel

Поиск

Сброс

Рисунок 2.39 – Форма для деталей для ремонта

И последней формой необходимо создать возможность добавления новых сотрудников для авторизации в системе (рисунок 2.40).

Ремонт компьютерной техники

Сотрудники

Логин

Пароль

ФИО

Телефон

Должность

[Очистить](#)

Добавить Изменить

Удалить Главная

Рисунок 2.40 – Форма добавления сотрудников

Так же стоит отметить, что для всех полей ввода данных необходимо поставить ограничение на количество вводимых символов и запретить вводить некорректные данные в некоторые поля, например в поле «ФИО» нельзя внести цифры.

На данном этапе разработку информационной системы можно считать завершённой и теперь необходимо перейти к следующему пункту, а именно протестировать её.

2.4 Тестирование информационной системы

Тестом можно назвать набор входных данных и набор ожидаемых результатов, набор условий, разработанных для проверки определенного пути выполнения программы.

Тестирование – это процесс, требующий планирования и выполнения ряда предварительных процедур, основной из которых является составление набора тестовых примеров.

Тестирование программного обеспечения — процесс исследования, испытания программного продукта, имеющий своей целью проверку соответствия между реальным поведением программы и её ожидаемым поведением на конечном наборе тестов, выбранных определённым образом (ISO/IEC TR 19759:2005).

На рисунке 2.41 изображен процесс загрузки приложения.

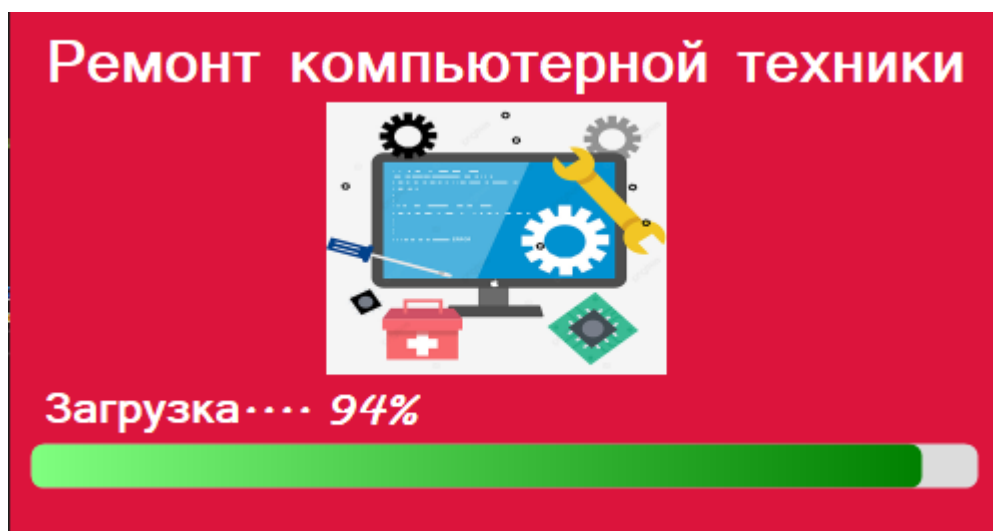


Рисунок 2.41 – Загрузка приложения

После загрузки приложения открывается форма авторизации, проверим её на корректные данные и войдём в систему под ролью «Сотрудник», внесем правильный логин до допустим ошибку в пароле, программа не сможет осуществить вход и выдаст соответствующую ошибку (рисунок 2.42).

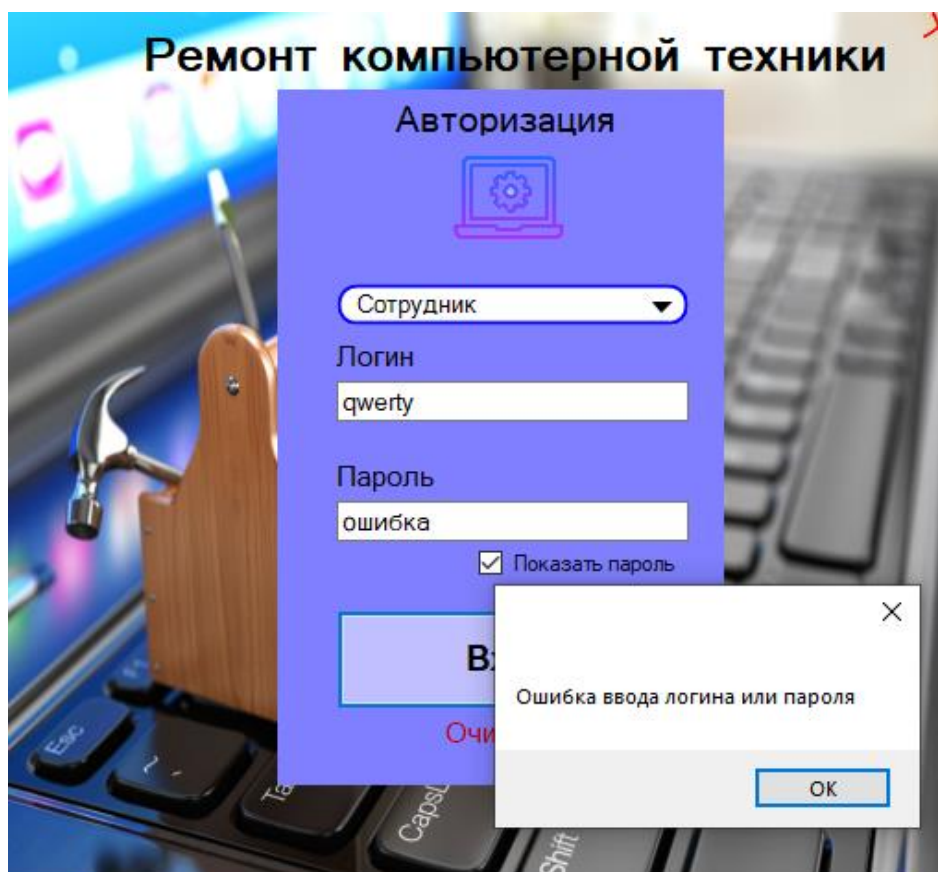


Рисунок 2.42 – Ошибка при авторизации

После ввода правильных данных для авторизации будет произведен вход в систему и отобразится форма главного меню, как описывалось в разработке главного меню, сотруднику не должны быть доступны некоторые разделы, нажмем на картинку сотрудники и появится окно об ошибке (рисунок 2.43).

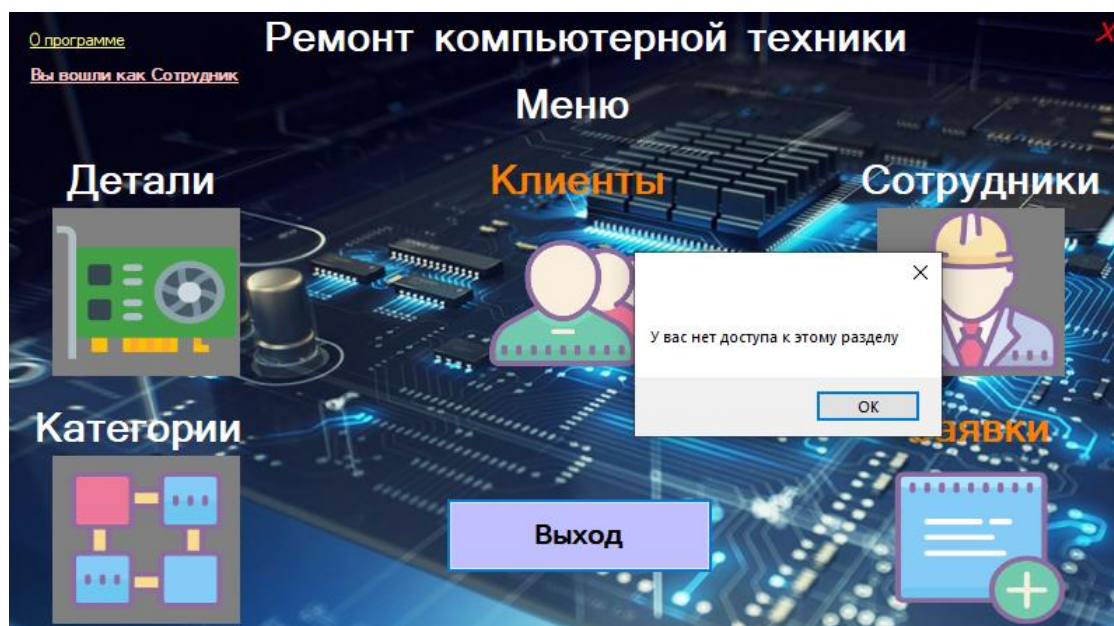


Рисунок 2.43 – Ошибка доступа к разделу

Перейдём в раздел «Клиенты», при нажатии на строку в таблице автоматически заполнятся поля для редактирования информации, а также данные об уже завершенной заявке (рисунок 2.44).

ID	ФИО	Телефон	Статус заявки
3	Вася Васечкин	+7(932)123-23-23	Новая заявка
4	Петя Петечкин	+7(922)675-10-81	Завершена и оплачена
5	Макс	+7(932)867-11-05	Завершена и оплачена
6	Сергей Александрович	+7(922)567-32-79	Новая заявка
7	Александр Невский	+7(922)678-12-54	Завершена и оплачена
99	Александр Александрович	+7(922)756-02-09	Завершена и оплачена

Рисунок 2.44 – Корректное отображение данных формы «Клиенты»

Внесём нового клиента для проверки корректности запроса к БД и отображения данных, поле «ID клиента» не заполняется, так как автоматически генерируется в базе данных, заполним поле «ФИО» текстом Сергей Фроликов, номер телефона 9224516703, программа автоматически разобьет телефон на группы и статус заявки укажем «Новая заявка». Так же воспользуемся поиском внесем в него начало имени «Сер» (рисунок 2.45).

ID	ФИО	Телефон	Статус заявки
6	Сергей Александрович	+7(922)567-32-79	Новая заявка
102	Сергей Фроликов	+7(922)451-67-03	Новая заявка

Рисунок 2.45 – Добавление нового клиента

Так же иконка для внесения девайса стала красной и появилась надпись внести девайс, нажмем на неё и добавим клиенту девайс выбрав его ID из списка, для привязки девайса именно к клиенту (рисунок 2.46).

Код девайса	Ид клиента	Девайс	Описание
329	4	Асег(ноутбук)	Не
461	7	Процессор	Пр
715	5	Асег(ноутбук)	Со
950	102	Блок питания	На
6251	99	Ноутбук(НР)	кла

Рисунок 2.46 – Добавление девайса клиента

Протестируем форму для формирования конечной заявки, а именно выберем нашего нового клиента и заполнив все необходимые поля добавим заявку (рисунок 2.47).

The screenshot shows a web application titled "Ремонт компьютерной техники Заявки" (Repair of computer equipment Requests). It features several sections:

- Клиенты (Clients):** A table with columns "ФИО" (Full Name) and "Статус заявки" (Request Status). It lists three clients: "Вася Васечкин" (New request), "Сергей Алексан..." (New request), and "Сергей Фроликов" (New request). A checkbox "Только новые заявки" (Only new requests) is checked.
- Комплектующие (Components):** A table with columns "Наименование" (Name), "Количество" (Quantity), "Цена" (Price), "Описание" (Description), and "Категория" (Category). It lists "Thermaltake TR2..." (9 units, 4000 price) and "Куллер" (Cooler) (9 units, 700 price).
- Form Fields:** Includes "ID клиента" (Client ID) with value "102", "ФИО клиента" (Client Name) with value "Сергей Фроликов", and a date field "31 мая 2022 г.". There are buttons "Добавить" (Add), "Заявки" (Requests), and "Главная" (Home).
- Количество (Quantity):** A text input field with a "Добавить деталь" (Add detail) button.
- Summary Table:** A table with columns "Номер" (Number), "Деталь" (Detail), "Количество" (Quantity), "Цена" (Price), and "Сумма" (Sum). It shows "6" (Cooler) with a quantity of 1, price of 700, and sum of 700. A "Сумма 700" (Sum 700) label and a "Убрать" (Remove) button are also present.
- Modal Dialog:** A small dialog box titled "Заявка добавлена" (Request added) with an "OK" button.

Рисунок 2.47 – Добавление заявки в завершенные

Но данная форма имеет множество проверок на внесение данных, а именно сообщение о внесении некорректной даты, сообщение о пустых полях, если у клиента уже есть сформированная заявка, то программа так же подскажет про это. Так же при добавлении комплектующих нельзя добавить ноль штук, добавить не выбрав комплектующее, напоминание об внесении количества и, если введенное количество превышает доступный на складе так же будет соответствующее сообщение (рисунок 2.48).

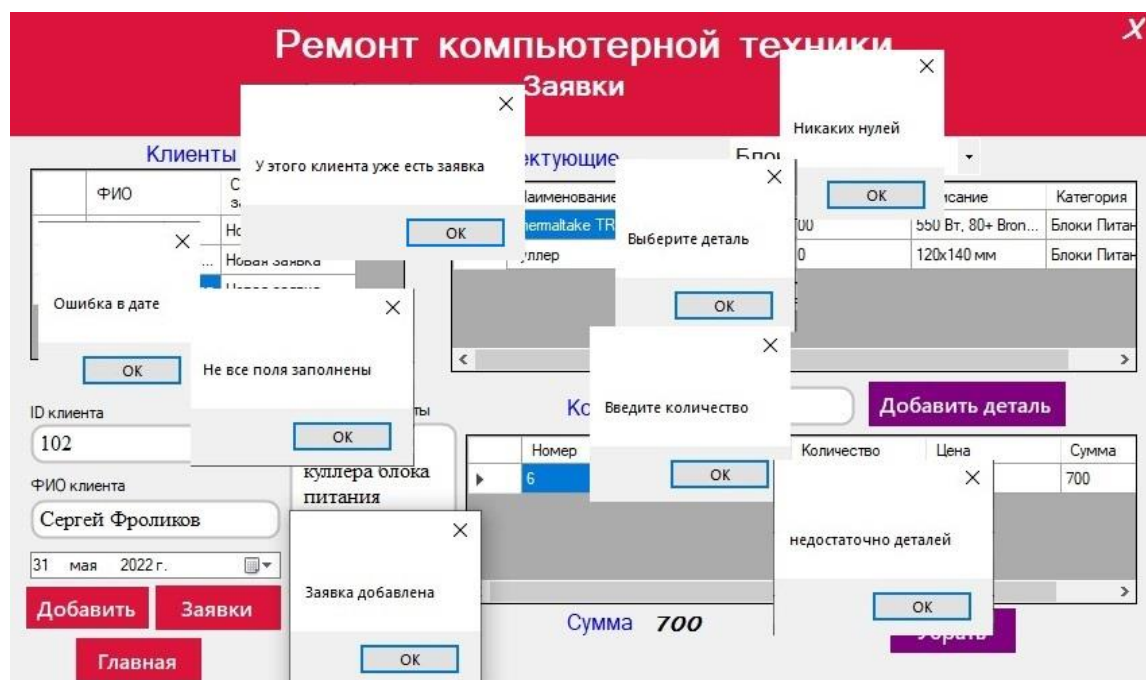


Рисунок 2.48 – Обработка ошибок формы «заявки»

Заявка после обработки добавляется в новую таблицу БД и отображается на форме с завершенными заявками, посмотрим появилась ли там наша заявка на имя Сергея Фроликова (рисунок 2.49).

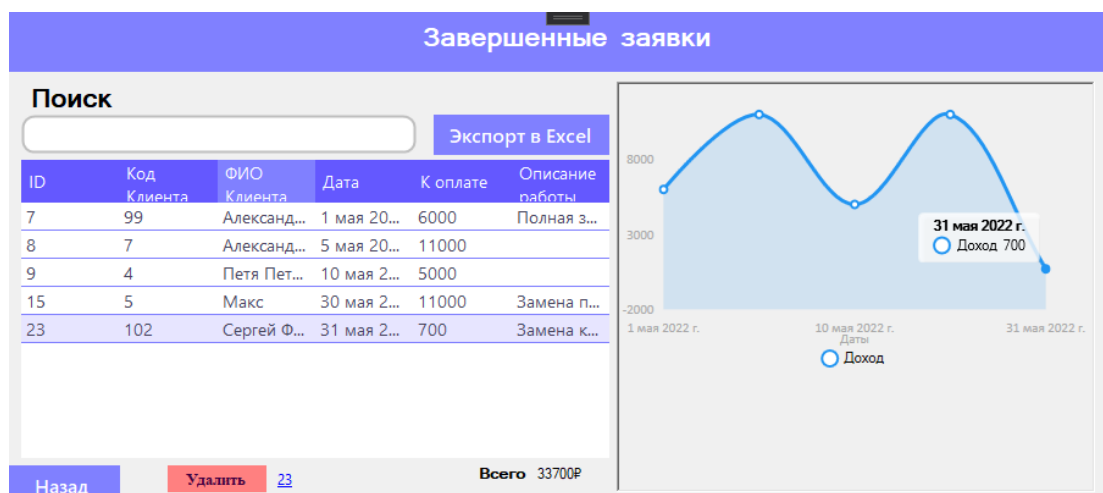


Рисунок 2.49 – Отображение данных формы с завершенными заявками.

Как видно из рисунка 2.49, заявка добавилась и даже стала отображаться на графике, теперь проверим печатается ли документ клиенту при двойном клике на запись в таблице рисунок 2.50.

Отчет заявки: 23

Номер клиента: 102
 ФИО клиента: Сергей Фроликов
 Дата обращения: 31 мая 2022 г.
 К оплате: 700
 Дата выдачи: 03.06.2022 23:30:17
 Прodelанная работа: Замена куллера блока питания

подпись

Рисунок 2.50 – Печать документа клиенту

Проверим так же нашу кнопку на экспорт, при нажатии на кнопку «Экспорт в Excel» вся таблица корректно экспортируется (рисунок 2.51).

ID	Код Клиента	ФИО Клиента	Дата	К оплате	Описание работы
7	99	Александр Алек	1 мая 2022	6000	Полная замена видеокарты
8	7	Александр Невс	5 мая 2022	11000	
9	4	Петя Петечкин	10 мая 2022	5000	
15	5	Макс	30 мая 2022	11000	Замена процессора
23	102	Сергей Фролико	31 мая 2022	700	Замена куллера блока питания

Рисунок 2.51 – Экспорт данных в Excel

3 Эксплуатационно-технологический раздел

3.1 Руководство для системного администратора

3.1.1 Назначение и условия использования системы. Данная версия информационной системы предназначена для учета заявок по ремонту компьютерной техники. Система упрощает процесс учета клиента, его девайса для ремонта так же осуществляет учет комплектующих и формирование конечной заявки с учетом стоимости работы и комплектующих.

3.1.2 Минимальные и рекомендуемые требования системы. Для работы с системой используется следующее серверное ПО:

- система управления базами данных SQL Server 2019;
- процессор Intel(R) Core (TM)+ / AMD Athlon X4+;
- 2 ГБ оперативной памяти;
- Интернет-канал со скоростью 100 Мбит/с.

Системные требования для программного обеспечения:

- процессор Intel Core i3 4130+ / AMD Athlon II+
- 512 МБ оперативной памяти;
- Операционная система Windows 7 и выше;
- 100 МБ дискового пространства;
- Интернет-канал со скоростью 10 Мбит/с.

Осуществлять работу с информационной системой может любой желающий, который имеет данное ПО. Для запуска программы необходимо открыть файл «Remont», который имеет расширение «exe».

3.1.3 Входные и выходные данные системы. Входными данными ИС являются данные клиента и его девайса. Выводными же являются отчеты и документ клиенту о проделанной работе.

3.1.4 Основные возможности ИС. В системе учета заявок на ремонт компьютерной техники помимо описанных выше выходных данных присутствуют следующие функции:

- 1) поиск по букве/слогу/фразе;
- 2) добавления новых сотрудников;
- 3) расширение услуг и комплектующий в виде добавления новых категорий;
- 4) построение графика по доходу.

3.1.5 Данные для авторизации. Для входа в систему под ролью администратора необходимо ввести логин и пароль привязанные к самой программе, а именно логин: Sport23 и пароль: 124589.

При входе под данным логином можно будет добавлять новых пользователей системы, а также удалить записи из БД.

3.2 Руководство для администратора системы

Роль администратора информационной системы учета заявок по ремонту компьютерной техники, является очень важной, так как внесение новых сотрудников, пополнение комплектующих для ремонта и добавление новых категорий для них, осуществляется администратором системы. Для входа администратора необходимо при загрузке формы авторизации выбрать данную роль и внести логин и пароль. На рисунке 3.1 показана авторизация администратора системы.

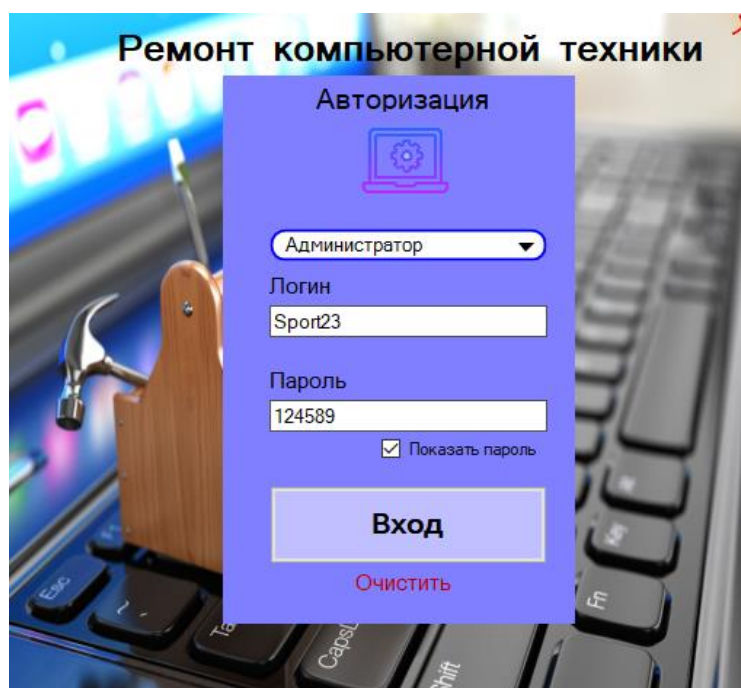


Рисунок 3.1 – Авторизация под ролью «Администратор»

После входа администратора в систему откроется форма главного меню на которой будут доступны все разделы системы (рисунок 3.2).

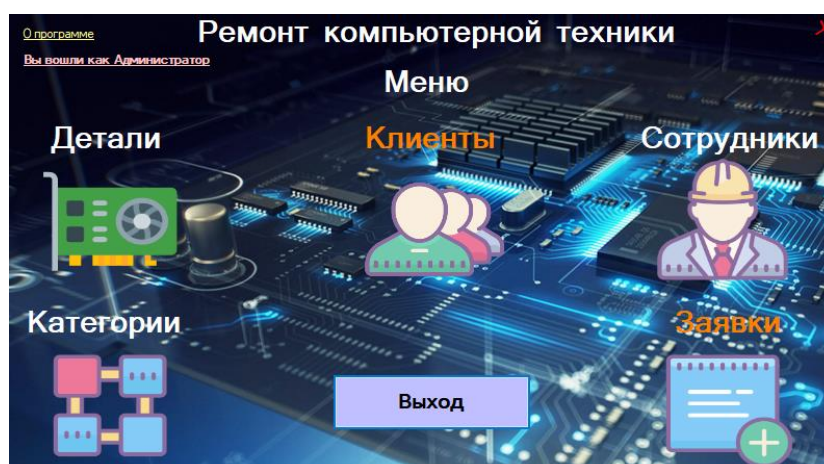


Рисунок 3.2 – Окно главного меню администратора

Как говорилось ранее для администратора будут доступны все разделы системы, но основными будут разделы «Категории», «Детали», «Сотрудники». Перейдем в раздел с категориями и добавим новую под названием «Термопасты». Теперь данная категория отобразится при добавлении новых комплектующих, добавим термопасту «GELID GC-Extreme», она является лучшей для мощных процессоров и видеокарт, в описании мы это укажем (рисунок 3.3-3.4).

Ремонт компьютерной техники
Комплектующие и услуги

ID категории: 9
Наименование: Термопасты
Очистить
Добавить Изменить
Удалить
Меню

ID	Наименование
1	Процессоры
	Мониторы
	Материнская плата
	Другое
	Блоки Питания
	Корпус
7	Видеокарты
8	КомМышка

добавлен
OK

Рисунок 3.3 – Внесение новой категории

Ремонт компьютерной техники
Детали

ID детали: 7
Наименование: Gelid gc-Extreme
Количество: 5
Цена: 3000
Термопасты
Добавить Изменить
Удалить
Главная

Описание: Подходит для мощных видеокарт и процессоров

Экспорт в Excel Термопасты Поиск Сброс

ID	Наименование	Количество	Цена	Описание	Ка
1	Thermaltake TR2...	8	4000	550 Вт, 80+ Bron...	Блс
3	Samsung	11	5000	1920x1080	Мо
4	Intel Core i9-1290...	11	11000	Интел	Пр
	AMD Radeon RX...	14	1000	AMD	Вид
	Куллер	12	700	120x140 мм	Блс

Деталь добавлена
OK

Рисунок 3.4 – Добавление новой комплектующей для ремонта

Теперь перейдем к добавлению нового сотрудника, в данной форме необходимо указать логин и пароль сотрудника, а также личные данные (рисунок 3.5).

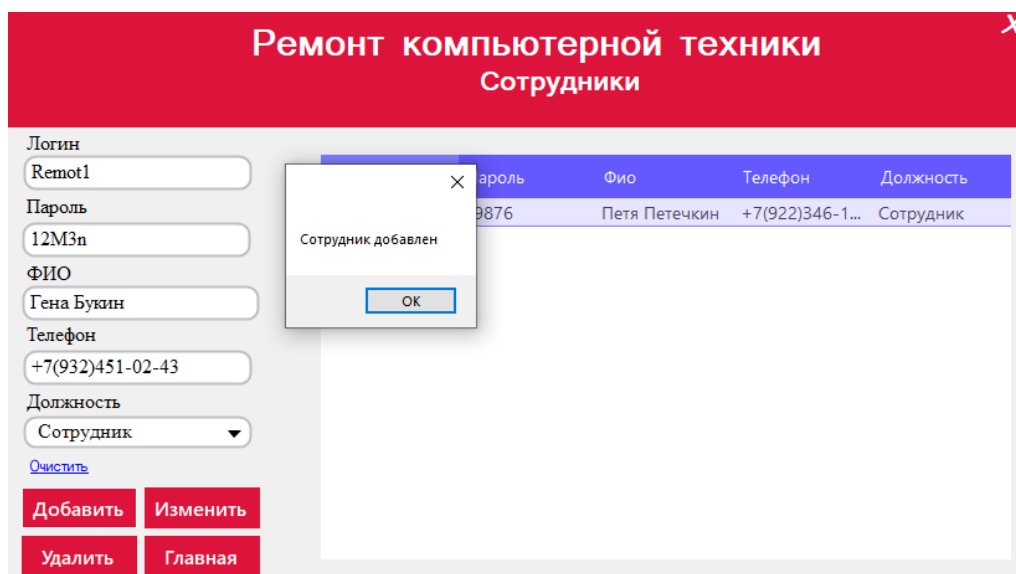


Рисунок 3.5 – Добавление нового сотрудника

3.3 Руководство для пользователя

Для использования данной информационной системы необходимо получить логин и пароль для авторизации. Так как регистрацию пользователей осуществляет администратор, то выдачу данных для авторизации выполнит так же он. В предыдущем этапе был зарегистрирован новый пользователь, авторизуемся системе под его данными (рисунок 3.6).

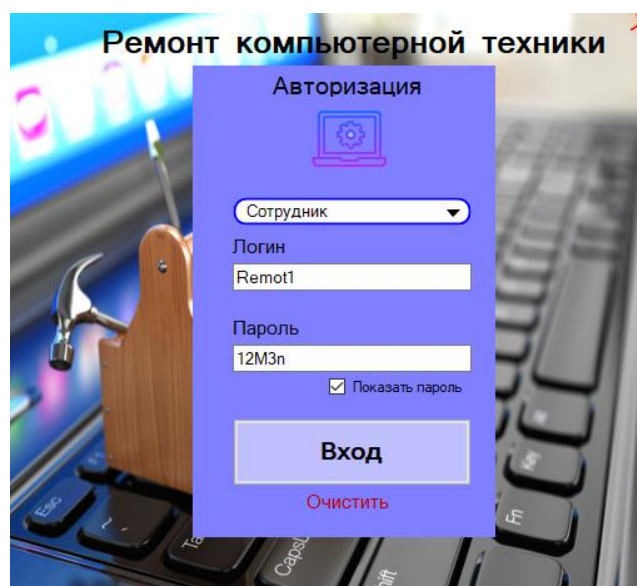


Рисунок 3.6 – Вход в систему нового сотрудника.

После входа в систему сотрудник может выбрать один из двух разделов и выполнить действия в соответствии с разделом (рисунок 3.7). Например, в разделе

«Клиенты» можно добавить нового клиента и его девайс для ремонта, а в разделе с заявками сформировать заявку с учетом комплектующих для ремонта.

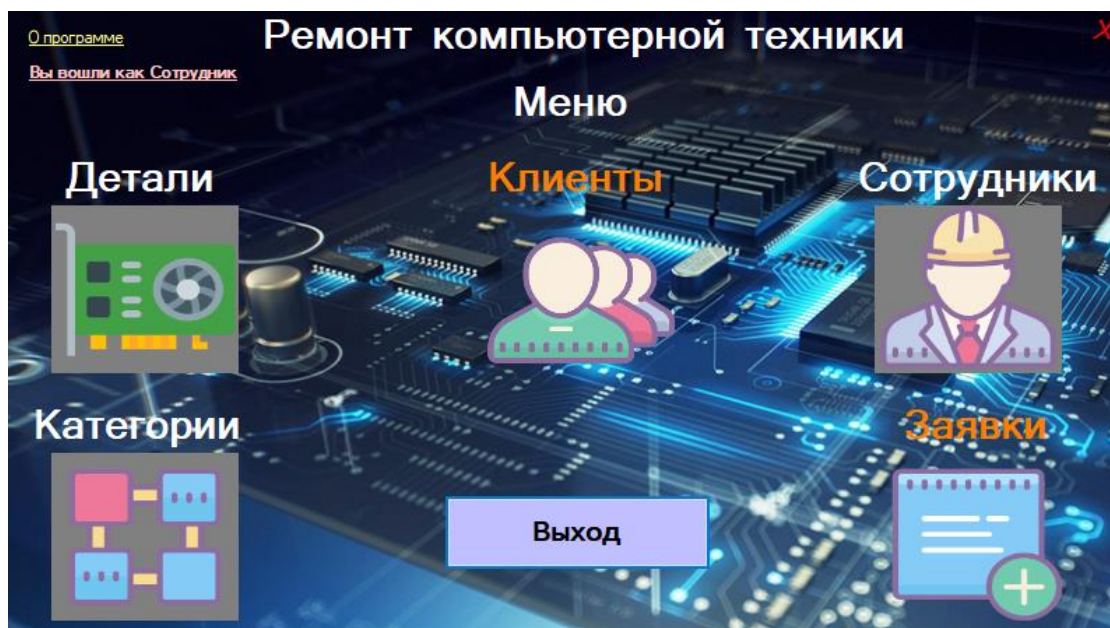


Рисунок 3.7 – Меню программы

В зависимости от выбранного раздела пользователь системы, то есть сотрудник, может вносить клиентов и устанавливать статус заявки в разделе «Клиенты» и обрабатывать уже имеющиеся заявки в разделе «Заявки». Так же формировать отчет по завершенным заявкам и печатать отчет по конкретному клиенту для выдачи с отремонтированным оборудованием.

Заключение

В ходе выполнения выпускной квалификационной работы была достигнута основная цель: разработка информационной системы учета заявок по ремонту компьютерной техники. Для достижения данной цели последовательно были выполнены все поставленные задачи.

В рамках решения первой задачи была проанализирована деятельность сервисного центра по ремонту компьютерной техники, а также построены диаграммы бизнес-процессов (IDEF0) и диаграммы потоков данных (DFD) в выбранной программе All Fusion Process Modeler. Для диаграмм была произведена декомпозиция, что позволило более подробно понять деятельность сервисного центра, для проектирования информационной системы.

В рамках решения второй задачи был произведен сравнительный анализ аналогов проектируемой информационной системы, а именно «1С: Предприятие» и «Gincore». В ходе анализа были выявлены основные минусы данных программ, а именно высокая цена и излишний функционал, данные минусы были учтены при разработке информационной системы.

В рамках решения третьей задачи был произведен выбор инструментальных средств и СУБД для разработки информационной системы. В качестве СУБД был выбран MS SQL Server. Разработка же самой информационной системы происходила в Visual Studio на языке программирования C#.

В рамках решения четвертой задачи было проведено структурное (функциональное) и объектно-ориентированное моделирование информационной системы в выбранной программе Rational Rose. В процессе моделирования были построены следующие диаграммы: диаграмма прецедентов, диаграмму взаимодействия, диаграмму активностей, диаграмму классов, диаграмму компонентов информационной системы, диаграмму размещения. Что позволило определить основные компоненты, графические модули и алгоритм разрабатываемой информационной системы.

В рамках решения пятой задачи была спроектирована база данных в выбранной СУБД. База данных имеет 7 таблиц и связи один ко многим. После создания базы данных были разработаны основные компоненты информационной системы, описан код основного функционала и графические модули.

В рамках решения шестой задачи был проведен ряд тестов для сравнения ожидаемых результатов работы программы. Тесты показали, что программа работает так, как и было задумано при разработке. Так же были описаны руководства пользования для системного программиста, администратора и пользователя системы.

Разработанная информационная система апробирована на XLIV Международной студенческой научной конференции в работе секции «Информационные системы и технологии», проводимой в апреле 2022 г, а также зарегистрирована в УФЭР ОГУ, свидетельство о регистрации № 3318 от 6 июня 2022 г.

Все задачи были рассмотрены и проработаны, что в конечном итоге позволило выполнить главную цель ВКР.

Список используемых источников

- 1 Синдеев, Ю.Г. Персональный компьютер: ремонт, обслуживание. - Ростов на Дону: Феникс, 2011. – 235 с.
- 2 Ярыгина, Е.Н. Анализ и диагностика финансово-хозяйственной деятельности предприятия: Учебное пособие. Е.Н.Ярыгина – Петропавловск-Камчатский: КамчатГТУ, 2006. – 93 с.
- 3 Кустова, Т.Н. Анализ и диагностика финансово-хозяйственной деятельности предприятия: Учебное пособие. Кустова Т.Н. - Рыбинск: РГАТА, 2013. - 200 с.
- 4 Гринберг, А.С. Информационный менеджмент. - М.: Юнити, 2003. - 415с.
- 5 Радченко, М. Г. 1С: Предприятие 8.2. Практическое пособие разработчика. Примеры и типовые приемы / М. Г. Радченко, Е. Ю. Хрусталева. -М.: ООО «1С-Паблишинг», 2009. – 874 с: ил.
- 6 Гиляровской, Л.Т. Экономический анализ: Учебник для вузов/Под ред. Л.Т. Гиляровской. – М.: ЮНИТИ-ДАНА, 2011. –527с.
- 7 Мякшин, В.Н. Экономическое обоснование проектов программных средств: Методические указания к дипломному проектированию. Мякшин В.Н.– Архангельск: Изд-во С(А)ФУ, 2010. – 60 с.
- 8 Леснова, Л. А. ИТ-инфраструктура современного предприятия: проблемные точки / Л. А. Леснова. – 8-е изд. – Петропавловск-Камчатский: Мир связи, 2010. – 8-12 с.
- 9 Александров, Д.В. Инструментальные средства информационного менеджмента. CASE-технологии и распределенные информационные системы / Д.В. Александров. - М.: Финансы и статистика, 2011. – 945 с.
- 10 Дубейковский, В.И. Эффективное моделирование с СА ERwin Process Modeler (BPwin; AllFusion Process Modeler) / В.И. Дубейковский. - М.: Диалог-Мифи, 2009. - 970 с.
- 11 Маклаков, С. В. Моделирование бизнес-процессов с AllFusion Process Modeler / С.В. Маклаков. - М.: Диалог-Мифи, 2004. – 240 с.
- 12 Черемных, С.В., Семенов И.О., Ручкин В.С. Моделирование и анализ систем. IDEF - технологии. - М.: «Финансы и кредит», 2011. – 240 с.
- 13 Закон РФ от 07.02.1992 N 2300-1 (ред. от 01.05.2017) «О защите прав потребителей»
- 14 Ансофф, И. М Новая корпоративная стратегия. В 2 частях. Часть 1. Методология управления: учеб. пособие / И. М. Ансофф, П. В. Доиль. – Москва: Финансы и статистика, 2006. – 416 с.
- 15 Маглинец, Ю.А. Анализ требований к автоматизированным информационным системам [Электронный ресурс]: учебное пособие / Маглинец Ю.А. – Электрон. текстовые данные. – Москва, Саратов: Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2020. – 191 с.
- 16 Коваленко, В. В. Проектирование информационных систем, Издательство: Форум, 2011 – 100с.
- 17 Буч Г., Рамбо Д., Джекобсон А. Язык UML. Руководство пользователя: Пер. с англ. Слинкин А.А. – Учебное пособие – М.: ДМК Пресс, 2000. - 432 с.

- 18 Боггс У., Боггс М. UML и Rational Rose – М.: Лори, 2001. – 582 с.
- 19 Леоненков, А. В. Объектно-ориентированный анализ и проектирование с использованием UML и IBM Rational Rose [Текст]: учебное пособие / А. В. Леоненков. – Москва: Интернет-Ун-т Информ. Технологий: БИНОМ. Лаборатория знаний, 2013. – 320 с.
- 20 Ларман, Крэг Применение UML 2.0 и шаблонов проектирования. Введение в объектно-ориентированный анализ, проектирование и итеративную разработку / Крэг Ларман. - М.: Вильямс, 2013. – 736 с.
- 21 Леоненков, А. В. Объектно-ориентированный анализ и проектирование с использованием UML и IBM Rational Rose / А.В. Леоненков. - Москва: Гостехиздат, 2006. - 320 с.
- 22 Бен-Ган, Ицик Microsoft SQL Server 2012. Основы T-SQL [Электронный ресурс]: учебное пособие / Ицик Бен-Ган. – М.: Эксмо, 2012. – 167 с. Режим доступа: <https://www.labirint.ru/books/471025>
- 23 Грабер, Мартин SQL для простых смертных / Мартин Грабер. - М.: ЛОРИ, 2014. - 378 с. Режим доступа: <https://clck.ru/q73yR>
- 24 Маркин, А.В. СУБД «РЕД БАЗА ДАННЫХ». ОСНОВЫ SQL. [Электронный ресурс]: учебное пособие / Маркин, А.В. – Электрон. текстовые данные. – Москва: Евразийский открытый институт, 2022. – 370 с. – Режим доступа: <https://www.iprbookshop.ru/119617.html>
- 25 Биллиг, В.А. Основы объектного программирования на С# (С# 3.0, Visual Studio 2008) [Электронный ресурс]: учебное пособие / Биллиг, В.А. – Москва: Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2021. – 409 с. – Режим доступа: <https://www.iprbookshop.ru/102029.html>
- 26 Токмаков, Г.П. Базы данных: модели и структуры данных, язык SQL, программирование баз данных. [Электронный ресурс]: учебное пособие / Токмаков Г.П. – Ульяновск: Ульяновский государственный технический университет, 2021. – 362 с. – Режим доступа: <https://www.iprbookshop.ru/121263.html>
- 27 Бурков, А.В. Проектирование информационных систем в Microsoft SQL Server 2008 и Visual Studio 2008. [Электронный ресурс]: / Бурков А.В. – Москва, Саратов: Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2020. – 310 с. – Режим доступа: <https://www.iprbookshop.ru/89466.html>
- 28 Павловская, Т.А. Программирование на языке высокого уровня С#: [Электронный ресурс]: / Павловская Т.А. – Москва: Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2021. – 245 с. – Режим доступа: <https://www.iprbookshop.ru/102051.html>
- 29 Баканов А.С. Проектирование пользовательского интерфейса: эргономический подход [Электронный ресурс]: / Баканов А.С., Обознов А.А. – Москва: Издательство «Институт психологии РАН», 2019. – 184 с. – Режим доступа: <https://www.iprbookshop.ru/88367.html>
- 30 Гайнанова Р.Ш. Разработка приложений в Visual C# для работы с базой данных MS SQL SERVER 2012: [Электронный ресурс]: / Гайнанова Р.Ш. – Казань: Казанский национальный исследовательский технологический университет, 2019. – 84 с. – Режим доступа: <https://www.iprbookshop.ru/109589.html>

Приложение А (обязательное)

Фрагмент листинга программы

customers.cs

```
using System;
using System.Data;
using System.Windows.Forms;
using System.Data.SqlClient;
using System.Text.RegularExpressions;
using System.Drawing;
```

```
namespace Diplom
```

```
{
    public partial class customers : Form
    {
        public customers()
        {
            InitializeComponent();
            textBox1.ShortcutsEnabled = false;
            textBox1.MaxLength = 30;
            CustID.ShortcutsEnabled = false;
            CustID.MaxLength = 3;
            CustName.ShortcutsEnabled = false;
            CustName.MaxLength = 30;
            CustPhone.MaxLength = 10;
            CustPhone.ShortcutsEnabled = false;
        }
    }
}
```

```
SqlConnection Con = new SqlConnection(@"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFile-
name=C:\RemDB\RemontDB.mdf;Integrated Security=True;Connect Timeout=30");
```

```
private void label3_Click(object sender, EventArgs e)
{
    Application.Exit();
}
void populate()
{
    try
    {
        Con.Open();
        string Myquery = "Select * from CustomerTB";
        SqlDataAdapter da = new SqlDataAdapter(Myquery, Con);
        SqlCommandBuilder builder = new SqlCommandBuilder(da);
        var ds = new DataSet();
        da.Fill(ds);
        CostGV.DataSource = ds.Tables[0];
        CostGV.Columns[0].HeaderText = "ID";
        CostGV.Columns[1].HeaderText = "ФИО";
        CostGV.Columns[2].HeaderText = "Телефон";
        CostGV.Columns[3].HeaderText = "Статус заявки";
        Con.Close();
    }

    catch
    {
    }
}
```

```

    }

}

private void button1_Click(object sender, EventArgs e)
{
    if (CustName.Text == null || CustName.Text == "")
        MessageBox.Show(
            "Заполните поле ФИО",
            "Сообщение",
            MessageBoxButtons.OK,
            MessageBoxIcon.Information);
    else if (CustPhone.Text == null || CustPhone.Text == "")
        MessageBox.Show(
            "Заполните поле номер телефона ",
            "Сообщение",
            MessageBoxButtons.OK,
            MessageBoxIcon.Information);
    else if (Statys.Text == null || Statys.Text == "")
        MessageBox.Show(
            "Укажите статус заявки",
            "Сообщение",
            MessageBoxButtons.OK,
            MessageBoxIcon.Information);
    else
    {
        try
        {
            Con.Open();
            SqlCommand cmd = new SqlCommand("insert into CustomerTB values (N'" + CustName.Text + "'," +
                """" + CustPhone.Text + "',N'" + Statys.Text + "')", Con);
            cmd.ExecuteNonQuery();
            MessageBox.Show("Клиент добавлен");
            Con.Close();
            populate();

        }

        catch (SqlException)
        {
            MessageBox.Show("Ошибка");
            Con.Close();
        }
    }
}

private void customers_Load(object sender, EventArgs e)
{
    populate();
    if (Data.LvL == "2")
    {
        button3.Visible = false;
    }
    else
    {
        button3.Visible = true;
    }
}

```

```

    }
}

private void button3_Click(object sender, EventArgs e)
{
    if (CustID.Text == "")
    {
        MessageBox.Show("Выберите ID клиента");
    }
    else
    {
        Con.Open();
        string del = "delete from CustomerTB where custID='" + CustID.Text + "';";
        SqlCommand cmd = new SqlCommand(del, Con);
        cmd.ExecuteNonQuery();
        MessageBox.Show("Запись удалена ");
        Con.Close();
        DeleteDevice();
        DeleteZaivca();
        populate();
    }
}

private void DeleteZaivca()
{
    Con.Open();
    string del = "delete from ReqTB where CustID='" + CustID.Text + "';";
    SqlCommand cmd = new SqlCommand(del, Con);
    cmd.ExecuteNonQuery();
    Con.Close();
}

private void DeleteDevice()
{
    Con.Open();
    string del = "delete from DeviceTB where IdKlient='" + CustID.Text + "';";
    SqlCommand cmd = new SqlCommand(del, Con);
    cmd.ExecuteNonQuery();
    Con.Close();
}

private void CostGV_CellContentClick(object sender, DataGridViewCellEventArgs e)
{
    CustID.Text = CostGV.CurrentRow.Cells[0].Value.ToString();
    CustName.Text = CostGV.CurrentRow.Cells[1].Value.ToString();
    CustPhone.Text = CostGV.CurrentRow.Cells[2].Value.ToString();
    Statys.Text = CostGV.CurrentRow.Cells[3].Value.ToString();
    Con.Open();

    SqlDataAdapter sda4 = new SqlDataAdapter("select Count(*) from DeviceTB where IdKlient = " +
        "" + CustID.Text + "" ", Con);
    DataTable dt4 = new DataTable();
    sda4.Fill(dt4);
    if (dt4.Rows[0][0].ToString() == "1")
    {
        pictureBox1.BackColor = Color.GreenYellow;
        DevaisEst.Text = "Девайс внесён";
    }
    else
    {
        pictureBox1.BackColor = Color.Red;
        DevaisEst.Text = "Внести девайс";
    }
}

```

```

    }

    SqlDataAdapter sda = new SqlDataAdapter("select Min(RequestID) from ReqTB where CustID= " + CustID.Text +
    + "", Con);
    DataTable dt = new DataTable();
    sda.Fill(dt);
    //RequestID
    ReqLab.Text = dt.Rows[0][0].ToString();

    SqlDataAdapter sda1 = new SqlDataAdapter("select Sum(TotAmt) from ReqTB where CustID= " + CustID.Text +
    "", Con);
    DataTable dt1 = new DataTable();
    sda1.Fill(dt1);
    Summa.Text = dt1.Rows[0][0].ToString();

    SqlDataAdapter sda2 = new SqlDataAdapter("select max(ReqDate) from ReqTB where CustID= " + CustID.Text +
    "", Con);
    DataTable dt2 = new DataTable();
    sda2.Fill(dt2);
    DataLab.Text = dt2.Rows[0][0].ToString();
    Con.Close();
}

private void button2_Click(object sender, EventArgs e)
{
    if (CustName.Text == null || CustName.Text == "")
        MessageBox.Show(
            "Заполните поле ФИО",
            "Сообщение",
            MessageBoxButtons.OK,
            MessageBoxIcon.Information);
    else if (CustPhone.Text == null || CustPhone.Text == "")
        MessageBox.Show(
            "Заполните поле номер телефона ",
            "Сообщение",
            MessageBoxButtons.OK,
            MessageBoxIcon.Information);
    else if (Statys.Text == null || Statys.Text == "")
        MessageBox.Show(
            "Укажите статус заявки",
            "Сообщение",
            MessageBoxButtons.OK,
            MessageBoxIcon.Information);
    else
    {
        try
        {
            Con.Open();
            SqlCommand cmd = new SqlCommand("update CustomerTB set custName =" +
            "N'" + CustName.Text + "',custPhone=N'" + CustPhone.Text + "', Status=" +
            "N'" + Statys.Text + "'where custID='" + CustID.Text + "'", Con);
            cmd.ExecuteNonQuery();
            MessageBox.Show("Данные изменены");
            Con.Close();
            populate();
        }
        catch (SqlException)
        {
            MessageBox.Show("Ошибка");
            Con.Close();
        }
    }
}

```

```

    }

}

private void button4_Click(object sender, EventArgs e)
{
    MainForm mainForm = new MainForm();
    mainForm.Show();
    this.Hide();
}

private void ReqLab_Click(object sender, EventArgs e)
{
    View Vi = new View();
    Vi.Show();
}

private void CustName_KeyPress(object sender, KeyPressEventArgs e)
{
    char l = e.KeyChar;
    if ((l < 'A' || l > 'Я') && l != '\b' && l != ' ')
    {
        e.Handled = true;
    }
}

private void CustPhone_KeyPress(object sender, KeyPressEventArgs e)
{
    char ch = e.KeyChar;
    if (!Char.IsDigit(ch) && ch != 8)
    {
        e.Handled = true;
    }
}

private void CustID_KeyPress(object sender, KeyPressEventArgs e)
{
    char ch = e.KeyChar;
    if (!Char.IsDigit(ch) && ch != 8)
    {
        e.Handled = true;
    }
    CustID.ReadOnly = true;
}

private void CustPhone_Leave(object sender, EventArgs e)
{
    var input = CustPhone.Text;
    var output = Regex.Replace(input, @"(?:\+7|8)?(?:\()?(?(\d{3})(?:\))?(?(\d{3})(?:-)?(\d{2})(?:-)?(\d{2}))", "+7($1)$2-$3-$4");
    CustPhone.Text = output;
}

private void textBox1_TextChanged(object sender, EventArgs e)
{
    Con.Open();
    SqlDataAdapter da = new SqlDataAdapter("SELECT custID, custName, custPhone, Status" +

```

```

        " FROM CustomerTB WHERE custName LIKE N'" + textBox1.Text + "' OR" +
        " custPhone LIKE N'" + textBox1.Text + "' OR" +
        " Status LIKE N'" + textBox1.Text + "' , Con);
SqlCommandBuilder cb = new SqlCommandBuilder(da);
DataSet ds = new DataSet();
da.Fill(ds, "CustomerTB");

CostGV.DataSource = ds.Tables[0];

Con.Close();
}

private void label7_Click(object sender, EventArgs e)
{

}

private void pictureBox1_Click(object sender, EventArgs e)
{
    device dev = new device();
    dev.Show();
}
}
}

Form1.cs

using System;
using System.Data;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace Diplom
{
    public partial class Form1 : Form
    {

        public Form1()
        {
            InitializeComponent();
            Password.ShortcutsEnabled = false;
            Login.MaxLength = 10;

            Login.ShortcutsEnabled = false;
            Password.MaxLength = 10;
        }

        SqlConnection Con = new SqlConnection(@"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFile-
name=C:\RemDB\RemontDB.mdf;Integrated Security=True;Connect Timeout=30");

        private void checkBox1_CheckedChanged(object sender, EventArgs e)
        {
            if (checkBox1.Checked == true)
                Password.UseSystemPasswordChar = false;
            else
                Password.UseSystemPasswordChar = true;
        }
    }
}

```

```

private void label3_Click(object sender, EventArgs e)
{
    Login.Text = "";
    Password.Text = "";
}

private void label6_Click(object sender, EventArgs e)
{
    Application.Exit();
}

private void button1_Click(object sender, EventArgs e)
{
    if (access.Text == "" || Login.Text == "" || Password.Text == "")
        MessageBox.Show(
            "Заполните все поля ",
            "Сообщение",
            MessageBoxButtons.OK,
            MessageBoxIcon.Information);
    if (access.Text == "Сотрудник")
    {
        Data.LvL += 2;
        Con.Open();
        SqlDataAdapter sda = new SqlDataAdapter("select Count(*) from userTB where login = " +
            "" + Login.Text + " and Pass= " + Password.Text + "and Position= N" + access.Text + "", Con);
        DataTable dt = new DataTable();
        sda.Fill(dt);
        if (dt.Rows[0][0].ToString() == "1")
        {
            MainForm mainForm = new MainForm();
            mainForm.Show();
            this.Hide();
        }
        else
        {
            MessageBox.Show("Ошибка ввода логина или пароля ");
            Data.LvL = null;
        }
        Con.Close();
    }
    if (access.Text == "Администратор" )
    {
        if(Login.Text == "Sport23" && Password.Text == "124589")
        {
            Data.LvL += 1;

            MainForm mainForm = new MainForm();
            mainForm.Show();
            this.Hide();
        }
        else
        {
            MessageBox.Show("Ошибка ввода логина или пароля ");
            Data.LvL = null;
        }
    }
}

```



```

    }

    private void Form1_KeyDown(object sender, KeyEventArgs e)
    {
        if (e.KeyValue == (char)Keys.Enter)
        {
            button1_Click(button1, null);
        }
    }
}

```

Request.cs

```

using System;
using System.Data;
using System.Windows.Forms;
using System.Data.SqlClient;
namespace Diplom
{
    public partial class Request : Form
    {
        public Request()
        {
            InitializeComponent();

            Work.ShortcutsEnabled = false;
            Work.MaxLength = 60;
            ClientID.ShortcutsEnabled = false;
            clientName.ShortcutsEnabled = false;
            qtTb.ShortcutsEnabled = false;
            qtTb.MaxLength = 2;
            textBox1.Visible = false;

        }
        DataTable table = new DataTable();
        SqlConnection Con = new SqlConnection(@"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFile-
name=C:\RemDB\RemontDB.mdf;Integrated Security=True;Connect Timeout=30");

        private void label3_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }
        void populate()
        {
            try
            {
                Con.Open();
                string Myquery = "Select * from CustomerTB";
                SqlDataAdapter da = new SqlDataAdapter(Myquery, Con);
                SqlCommandBuilder builder = new SqlCommandBuilder(da);
                var ds = new DataSet();
                da.Fill(ds);
                clientyGV.DataSource = ds.Tables[0];
                clientyGV.Columns[0].HeaderText = "ID";
                clientyGV.Columns[0].Visible = false;
                clientyGV.Columns[1].HeaderText = "ФИО";
            }
            catch { }
        }
    }
}

```

```

        clientyGV.Columns[2].HeaderText = "Телефон";
        clientyGV.Columns[2].Visible = false;
        clientyGV.Columns[3].HeaderText = "Статус заявки";

        Con.Close();

    }

    catch
    {

    }

}

void fillcategory()
{
    string query = "select * from DetServTB ";
    SqlCommand cmd = new SqlCommand(query, Con);
    SqlDataReader rdr;
    try
    {
        Con.Open();
        DataTable dt = new DataTable();
        dt.Columns.Add("DetSerName", typeof(string));
        rdr = cmd.ExecuteReader();
        dt.Load(rdr);
        // Categor.ValueMember = "DetSerName";
        // Categor.DataSource = dt;
        searchCat.ValueMember = "DetSerName";
        searchCat.DataSource = dt;

        Con.Close();

    }
    catch
    {

    }
}

void populateDetali()
{
    try
    {
        Con.Open();
        string Myquery = "Select * from DetaliTB";
        SqlDataAdapter da = new SqlDataAdapter(Myquery, Con);
        SqlCommandBuilder builder = new SqlCommandBuilder(da);
        var ds = new DataSet();
        da.Fill(ds);
        DetaliGV.DataSource = ds.Tables[0];
        DetaliGV.Columns[0].HeaderText = "ID";
        DetaliGV.Columns[0].Visible = false;
        DetaliGV.Columns[1].HeaderText = "Наименование";
        DetaliGV.Columns[2].HeaderText = "Количество";
        DetaliGV.Columns[3].HeaderText = "Цена";
        DetaliGV.Columns[4].HeaderText = "Описание";
        DetaliGV.Columns[5].HeaderText = "Категория";

        Con.Close();
    }
    catch
    {

    }
}

```

```

    }

    catch
    {

    }

}

private void Request_Load(object sender, EventArgs e)
{
    dateReq.Value = DateTime.Today;

    clientName.ReadOnly = true;
    populate();
    populateDetali();
    fillcategory();

}

private void showCustDetal ()
{
    try
    {
        Con.Open();
        string Myquery = "Select * from DetalCust where IDcust = " + ClientID.Text + "";
        SqlDataAdapter da = new SqlDataAdapter(Myquery, Con);
        SqlCommandBuilder builder = new SqlCommandBuilder(da);
        var ds = new DataSet();
        da.Fill(ds);
        ReqGV.DataSource = ds.Tables[0];
        ReqGV.Columns[0].HeaderText = "ID";
        ReqGV.Columns[0].Visible = false;
        ReqGV.Columns[1].HeaderText = "Номер детали";
        ReqGV.Columns[2].HeaderText = "Наименование";
        ReqGV.Columns[3].HeaderText = "Количество";
        ReqGV.Columns[4].HeaderText = "Цена";
        ReqGV.Columns[5].HeaderText = "Сумма";
        ReqGV.Columns[6].HeaderText = "id";
        ReqGV.Columns[6].Visible = false;

        Con.Close();
        TotBalans();

    }

    catch
    {

    }

}

```

```

int num = 0;
int uprice, totprice, qty;
string detal;
int flag = 0;
//int id = 0;
int stock = 0;
void updateDetali()
{

    int id = Convert.ToInt32( DetaliGV.CurrentRow.Cells[0].Value.ToString());
    int newqty = stock - Convert.ToInt32(qtTb.Text);
    if (newqty < 0)
        MessageBox.Show("невозможно");

    else
    {
        Con.Open();
        string query = "update DetaliTB set DetQty = " + newqty + " where DetID= " + id + "; ";
        SqlCommand cmd = new SqlCommand(query, Con);
        cmd.ExecuteNonQuery();
        Con.Close();
        populateDetali();
    }
}

```

```

private void DetaliGV_CellContentClick(object sender, DataGridViewCellEventArgs e)
{
    num = Convert.ToInt32(DetaliGV.CurrentRow.Cells[0].Value.ToString());
    detal = DetaliGV.CurrentRow.Cells[1].Value.ToString();
    // qty = Convert.ToInt32 (qtTb.Text);
    stock = Convert.ToInt32(DetaliGV.CurrentRow.Cells[2].Value.ToString());
    uprice = Convert.ToInt32(DetaliGV.CurrentRow.Cells[3].Value.ToString());
    //totprice = qty * uprice;
    flag = 1;
}

```

```

private void TotBalans()
{

    double balans = 0;
    foreach (DataGridViewRow row in ReqGV.Rows)
    {
        double incom;
        double.TryParse((row.Cells[5].Value ?? "0").ToString().Replace(".", ","), out incom);
        balans += incom;
    }
    TotAm.Text = balans.ToString();

}

```

```

int sum = 0;
private void button1_Click(object sender, EventArgs e)
{
    if (flag == 0)
        MessageBox.Show("Выберите деталь");
    else if (qtTb.Text == "")
        MessageBox.Show("Введите количество ");
    else if (qtTb.Text == "0")

```

```

        MessageBox.Show("Никаких нулей ");
    else if (Convert.ToInt32(qtTb.Text) < 0)
        MessageBox.Show("Ошибка записи");

    else if (Convert.ToInt32(qtTb.Text) > stock)
        MessageBox.Show("недостаточно деталей");
    else
    {

        num = Convert.ToInt32(DetaliGV.CurrentRow.Cells[0].Value.ToString());
        qty = Convert.ToInt32(qtTb.Text);
        totprice = qty * uprice;
        Con.Open();
        SqlCommand cmd = new SqlCommand("insert into DetalCust values" +
            "(" + ClientID.Text + "," +
            num + ",N" + detal + "," + qtTb.Text + "," + uprice + "," + totprice + ")", Con);
        cmd.ExecuteNonQuery();
        Con.Close();
        MessageBox.Show("Запись добавлена ");

        ReqGV.DataSource = table;
        flag = 0;
        updateDetali();
        qtTb.Text = "";
        showCustDetal();
    }
}

int BackDetal, FL, nazat, idnazat;

private void ReqGV_CellContentClick(object sender, DataGridViewCellEventArgs e)
{
    BackDetal = Convert.ToInt32(ReqGV.CurrentRow.Cells[0].Value.ToString());
    textBox1.Text = ReqGV.CurrentRow.Cells[6].Value.ToString();
    idnazat = Convert.ToInt32(ReqGV.CurrentRow.Cells[1].Value.ToString());
    nazat = Convert.ToInt32(ReqGV.CurrentRow.Cells[3].Value.ToString());
    // qty = Convert.ToInt32 (qtTb.Text);
    //BackStock = Convert.ToInt32(ReqGV.CurrentRow.Cells[2].Value.ToString());

    FL = 1;
}

private void button6_Click(object sender, EventArgs e)
{
    if (textBox1.Text == "")
    {
        MessageBox.Show("Выберите Деталь");
    }
    else if (idnazat == num)
    {
        Con.Open();
        string del = "delete from DetalCust where id =" + textBox1.Text + ",";
        SqlCommand cmd = new SqlCommand(del, Con);
        cmd.ExecuteNonQuery();
        MessageBox.Show("Запись удалена ");

        Con.Close();
        showCustDetal();
    }
}

```

```

        BakDetali();

        textBox1.Text = "";
    }
    else
    {
        MessageBox.Show("Пожалуйста выберите одинаковые детали из верхней и нижней таблицы");
    }
}

void BakDetali()
{
    // int id = Convert.ToInt32(ReqGV.CurrentRow.Cells[1].Value.ToString());

    int newqty = stock + nazat;
    if (newqty < 0)
        MessageBox.Show("невозможно");

    else
    {
        Con.Open();
        string query = "update DetaliTB set DetQty = " + newqty + " where DetID= " + idnazat + "; ";
        SqlCommand cmd = new SqlCommand(query, Con);
        cmd.ExecuteNonQuery();
        Con.Close();
        populateDetali();
    }
}

private void button2_Click(object sender, EventArgs e)
{
    if ( ClientID.Text=="" || clientName.Text=="" || TotAm.Text=="")
    {
        MessageBox.Show("Не все поля заполнены");
    }
    else if(Work.Text == "" || Work.Text == null)
    {
        MessageBox.Show("Пожалуйста заполните поле Описание работы ");
    }

    else if (dateReq.Value < DateTime.Today || dateReq.Value > DateTime.Today)
    {
        MessageBox.Show("Ошибка в дате");
        dateReq.Value = DateTime.Today;
    }
    else
    {
        Con.Open();
        SqlDataAdapter sda = new SqlDataAdapter("select Count(*) from ReqTB where CustID = " +
            "" + ClientID.Text + " ", Con);
        DataTable dt = new DataTable();
        sda.Fill(dt);
        if (dt.Rows[0][0].ToString() == "1")
        {
            MessageBox.Show("У этого клиента уже есть заявка");
            Con.Close();
        }
        else
    }
}

```

```

    {
        try
        {
            SqlCommand cmd = new SqlCommand("insert into ReqTB values" +
                "(" + ClientID.Text + "," +
                "N" + clientName.Text + ",N" + dateReq.Text + "," + TotAm.Text + ",N" + Work.Text + ")", Con);
            cmd.ExecuteNonQuery();
            MessageBox.Show("Заявка добавлена ");
            Con.Close();
            clientName.Text = "";
            ClientID.Text = "";
            TotAm.Text = "";
            Work.Text = "";
            int rowsCount = ReqGV.Rows.Count;
            for (int i = 0; i < rowsCount; i++)
            {
                ReqGV.Rows.Remove(ReqGV.Rows[0]);
            }
        }
        catch (SQLException)
        {
            MessageBox.Show("Ошибка подключения");
            Con.Close();
        }
    }
}

```

```

private void button3_Click(object sender, EventArgs e)
{

```

```

    View view = new View();
    view.Show();
    this.Hide();

```

```

}

```

```

private void label6_Click(object sender, EventArgs e)
{

```

```

    MainForm mainForm = new MainForm();
    mainForm.Show();
    this.Hide();

```

```

}

```

```

private void button4_Click(object sender, EventArgs e)
{

```

```

    MainForm mainForm = new MainForm();
    mainForm.Show();
    this.Hide();

```

```

}

```

```

private void clientyGV_CellContentDoubleClick(object sender, DataGridViewCellEventArgs e)
{

```

```

    if (clientyGV.CurrentCell != null)

```



```

{

    device dev = new device();

    dev.Show();
    // shDEv.Show();
    // shDEv.ShName = clientyGV.Rows[clientyGV.CurrentCell.RowIndex].Cells[1].Value.ToString();
    //shDEv.ID = clientyGV.Rows[clientyGV.CurrentCell.RowIndex].Cells[0].Value.ToString();

    // shDEv.DeviceName.Text = clientyGV.Rows[clientyGV.CurrentCell.RowIndex].Cells[1].Value.ToString();

}
else
{
    MessageBox.Show("не тыкай !");
}

}

private void qtTb_KeyPress(object sender, KeyPressEventArgs e)
{
    char ch = e.KeyChar;
    if (!Char.IsDigit(ch) && ch != 8)
    {
        e.Handled = true;
    }
}

private void ClientID_KeyPress(object sender, KeyPressEventArgs e)
{
    ClientID.ReadOnly = true;
}

private void clientName_KeyPress(object sender, KeyPressEventArgs e)
{
    clientName.ReadOnly = true;
}

private void checkBox1_CheckedChanged(object sender, EventArgs e)
{
    if (checkBox1.Checked == true)
    {
        Con.Open();
        string Myquery = "Select * from CustomerTB where Status = N'Новая заявка' ";
        SqlDataAdapter da = new SqlDataAdapter(Myquery, Con);
        SqlCommandBuilder builder = new SqlCommandBuilder(da);
        var ds = new DataSet();
        da.Fill(ds);
        clientyGV.DataSource = ds.Tables[0];

        Con.Close();
    }
    else
    {
        populate();
    }
}
}

```

```

private void clientyGV_CellContentClick(object sender, DataGridViewCellEventArgs e)
{
    ClientID.Text = clientyGV.CurrentRow.Cells[0].Value.ToString();
    clientName.Text = clientyGV.CurrentRow.Cells[1].Value.ToString();
    showCustDetal();
    TotBalans();
}

private void searchCat_SelectionChangeCommitted(object sender, EventArgs e)
{
    try
    {
        Con.Open();
        string Myquery = "Select * from DetaliTB where DetCat = N" + searchCat.SelectedValue.ToString() + " ";
        SqlDataAdapter da = new SqlDataAdapter(Myquery, Con);
        SqlCommandBuilder builder = new SqlCommandBuilder(da);
        var ds = new DataSet();
        da.Fill(ds);
        DetaliGV.DataSource = ds.Tables[0];

        Con.Close();
    }

    catch
    {
    }
}
}

```

Users.cs

```

using System;
using System.Data;
using System.Windows.Forms;
using System.Data.SqlClient;
using System.Text.RegularExpressions;

namespace Diplom
{
    public partial class Users : Form
    {
        public Users()
        {
            InitializeComponent();

            Login.ShortcutsEnabled = false;
            Login.MaxLength = 7;
            Password.ShortcutsEnabled = false;
            Password.MaxLength = 7;
            FIO.ShortcutsEnabled = false;
            Phone.ShortcutsEnabled = false;
        }

        SqlConnection Con = new SqlConnection(@"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFile-
name=C:\RemDB\RemontDB.mdf;Integrated Security=True;Connect Timeout=30");
    }
}

```

```

private void label3_Click(object sender, EventArgs e)
{
    Application.Exit();
}
void populate()
{
    try
    {
        Con.Open();
        string Myquery = "Select * from userTB";
        SqlDataAdapter da = new SqlDataAdapter(Myquery, Con);
        SqlCommandBuilder builder = new SqlCommandBuilder(da);
        var ds = new DataSet();
        da.Fill(ds);
        UserGV.DataSource = ds.Tables[0];
        UserGV.Columns[0].HeaderText = "Логин";
        UserGV.Columns[1].HeaderText = "Пароль";
        UserGV.Columns[2].HeaderText = "Фино";
        UserGV.Columns[3].HeaderText = "Телефон";
        UserGV.Columns[4].HeaderText = "Должность";
        Con.Close();

    }

    catch
    {

    }

}

private void button1_Click(object sender, EventArgs e)
{
    if (Login.Text.Length < 5)
        MessageBox.Show(
            "Данные в поле логин должны быть равны 5-7 символов ",
            "Сообщение",
            MessageBoxButtons.OK,
            MessageBoxIcon.Information);
    else if (Password.Text.Length < 5)
        MessageBox.Show(
            "Данные в поле пароль должны быть равны 5-7 символов ",
            "Сообщение",
            MessageBoxButtons.OK,
            MessageBoxIcon.Information);
    else if (Login.Text == null || Login.Text == "")
        MessageBox.Show(
            "Заполните поле логин",
            "Сообщение",
            MessageBoxButtons.OK,
            MessageBoxIcon.Information);
    else if (Password.Text == null || Password.Text == "")
        MessageBox.Show(
            "Заполните поле пароль",
            "Сообщение",
            MessageBoxButtons.OK,
            MessageBoxIcon.Information);
    else if (FIO.Text == null || FIO.Text == "")
        MessageBox.Show(

```