МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН
Некоммерческое акционерное общество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ ИМЕНИ ГУМАРБЕКА ДАУКЕЕВА»
Кафедра «IT - Инжиниринг»

«ДОПУЩЕН К ЗАЩИТЕ»
Зав. кафедрой PhD, доцент Досжанова А.А
_____ «____» _____2020 г.

ДИПЛОМНЫЙ ПРОЕКТ

На тему: Разработка информационной системы работы с клиентами автосервиса
Специальность 5В060200 Информатика
Выполнил Рыскулбеков Дастан Данияруулы Группа Инф-16-2
(Ф.И.О.)
Научный руководитель: к.т.н., Тусупова Бэлла Блялевна
(ученая степень, звание, Ф.И.О.)

Консультанты:
по экономической части: к.э.н., профессор Габелашвили Кахабер Ревазович
(учѐная степень, звание, Ф.И.О.)
_____ «____» _____2020 г.
(подпись)
по безопасности жизнедеятельности: ассистент Тыщенко Елена Михайловна
(учѐная степень, звание, Ф.И.О.)
_____ «____» _____2020 г.
(подпись)
по применению вычислительной техники:
ст. преп. Майкотов Мухит Нурдаулетович
(учѐная степень, звание, Ф.И.О.)
_____ «____» _____2020 г.
(подпись)
Нормоконтролер: ст.преп. Абсатарова Б.Р
(учѐная степень, звание, Ф.И.О.)
_____ «____» _____2020 г.
(подпись)
Рецензент:_____
(учѐная степень, звание, Ф.И.О.)
_____ «____» _____2020 г.
(подпись)

Алматы 2020

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН
Некоммерческое акционерное общество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ ИМЕНИ ГУМАРБЕКА ДАУКЕЕВА»

Институт систем управления и информационных технологий»

Кафедра «IT - Инжиниринг»

Специальность 5В060200-Информатика

ЗАДАНИЕ
на выполнение дипломного проекта

Студенту Рыскулбекову Дастану Даниярулы
(Ф.И.О.)

Тема проекта “Разработка информационной системы работы с клиентами автосервиса”

Утверждена приказом по университету №_____ от «__» _____ 201__ г.
Срок сдачи законченного проекта «_____»_____201___г.

Исходные данные к проекту (требуемые параметры результатов исследования (проектирования) и исходные данные объекта):
В данной работе нужно разработать информационную систему автосервиса, необходимую для работы с клиентами.

Перечень вопросов, подлежащих разработке в дипломном проекте, или краткое содержание дипломного проекта: анализ существующих автоматизированных информационных систем, проектирование информационной системы, разработка программного обеспечения системы.

Перечень графического материала (с точным указанием обязательных чертежей): физическая модель базы данных стр.23, схема базы данных стр.23, диаграмма последовательности работы базы данных стр.24, схема помещения, где будет использоваться информационная система стр.67.

Основная рекомендуемая литература:

1 Вендров А.М. CASE–технологии. Современные методы и средства проектирования информационных систем. – М.: Финансы и статистика, 2000.

2 Грекул В.И., Денищенко Г.Н., Коровкина Н.Л. Проектирование информационных систем. – М.: Интернет–университет информационных технологий – ИНТУИТ.ру, 2005.

3 Гарсиа–Молина Г., Ульман Дж., Уидом Дж. Системы баз данных. Полный курс. – М.: Вильямс, 2003. – 1088 с.

Консультации по проекту с указанием относящихся к ним разделов проекта

Раздел	Консультант	Сроки	Подпись
Нормоконтроль	Абсатарова Б.Р.	18.05.2020	
Программное обеспечение	Майкотов М.Н.	14.05.2020	
Подготовка экономической части дипломного проекта	Габелашвили К.Р.	24.04.2020	
Подготовка части БЖД	Тыщенко Е.М.	24.04.2020	

График
подготовки дипломного проекта:

Наименование разделов, перечень разрабатываемых вопросов	Сроки представления научному руководителю	Примечание
Обзор литературы и анализ современного состояния вопроса	13.01.2020-13.02.2020	
Проектирование информационной системы	16.01.2020-22.02.2020	
Разработка программного обеспечения	20.01.2020-21.05.2020	
Экономическая часть	02.03.2020-24.04.2020	
Охрана труда и БЖД	02.03.2020-24.04.2020	

Дата выдачи задания «13» января 2020 г.

Заведующий кафедрой _____ А.А. Досжанова

Научный руководитель работы _____ Б.Б. Тусупова

Задание принял к исполнению студент _____ Д.Д. Рыскулбеков

Аңдатпа

Автосервис үшін ақпараттық жүйені әзірлеу - қойылған тапсырманың басты шешімі. Ақпараттық жүйе автосервистің жұмыс істеу барысын жеңілдете алады. Жобада тапсырыстарға, тұтынушымен кері байланысқа, сақтауға, импортқа және экспортқа үлкен көңіл бөлінген. Тапсырыстардың импорты MySQL базасы арқылы жүзеге асады және бұл жобада үлкен рөл ойнайды. Қосымшадағы барлық процесс деректер базасын қолдану арқылы жүзеге асады. Қосымша тапсырыс жасауға және оны деректер базасында сақтауға мүмкіндік береді. Кері байланыс тұтынушы мен менеджер бір-бірімен сөйлесе алатын форма арқылы жүзеге асады.

Аннотация

Разработка информационной системы для автосервиса – это главное решение поставленной задачи. Информационная система позволяет упростить процесс работы автосервиса. Огромное внимание в проекте уделено заказам, обратной связи с клиентом, хранению, импорту и экспорту. Импорт заказов ведется с помощью базы данных MySQL и играет важную роль в этом проекте. Весь процесс в приложении протекает с использованием базы данных. Приложение дает возможность совершить заказ и хранить его в базе данных. Обратная связь осуществляется через форму, в которой клиент и менеджер могут общаться друг с другом.

The Summary

The main solution for the given task is the development of an information system for the car service. The information system allows you to simplify the workflow of car service. Orders, customer feedback, storing, importing and exporting data are the main focus in this project. Order import gets done by MySQL database and plays an important part in the project. Entire process is conducted using database. Application allows you to make an order and store it in database. Customer feedback executes in a form, using which manager and customer can communicate with each other.

Содержание

Введение

Главным элементом организации работы любого предприятия, является разработка собственной информационной системы. Информационная система значительно упрощает рабочий процесс предприятия, а также повышает эффективность сотрудников при работе с клиентами.

Внедрение информационных систем в деятельность предприятий также сопровождается сокращением производственных затрат, что является выгодным вариантом для многих компаний.

Основной целью дипломного проекта является автоматизация работы автосервиса для упрощения взаимодействия между клиентом и менеджером.

Основными задачами дипломного проекта являются:

- создать средство обмена сообщениями между клиентом и менеджером;
- осуществить возможность редактирования данных внутри приложения;
- осуществить выдачу электронного чека для текущего заказа;

Данная информационная система повышает эффективность производства, ускоряет рабочий процесс и облегчает работу автосервиса.

Основным объектом экономической части выступает информационная система, которая предоставляет решение задач, возникающих в экономических системах, и конечная цель функционирования является эффективное управление экономической системой.

Пояснительная записка дипломного проекта состоит из пяти разделов.

В первом разделе был анализ предметной области. В этом разделе необходимо рассмотреть существующие системы и найти их проблемы, что в дальнейшем решить в программной реализации.

Второй раздел посвящен выбранному языку программирования. В проекте был выбран язык C# с подключенной MySQL базой данных.

В третьем разделе была разработана и описана программная реализация. Главной задачей раздела было описание работы программы в деталях, показав работу кода и интерфейс программы.

В четвертом разделе необходимо обосновать эффективность внедряемой программы на предприятие.

Пятый раздел посвящен вопросам охраны труда и безопасности жизнедеятельности.

1 Анализ предметной области

1.1 Анализ существующих систем и их проблемы

CRM является программой, которая обеспечивает возможность бизнесу взаимодействовать с клиентами. Программа дает возможность вашим сотрудникам эффективнее и быстрее произвести продажу. Рассмотрим положительные стороны CRM на определенном примере.

Преимущества интеграции CRM-систем [1]:

- история работы с каждым клиентом;
- контакты;
- данные о клиенте;
- файлы и документы;
- записи звонков и письма;
- история покупок и платежей;
- примечания менеджеров и история задач.

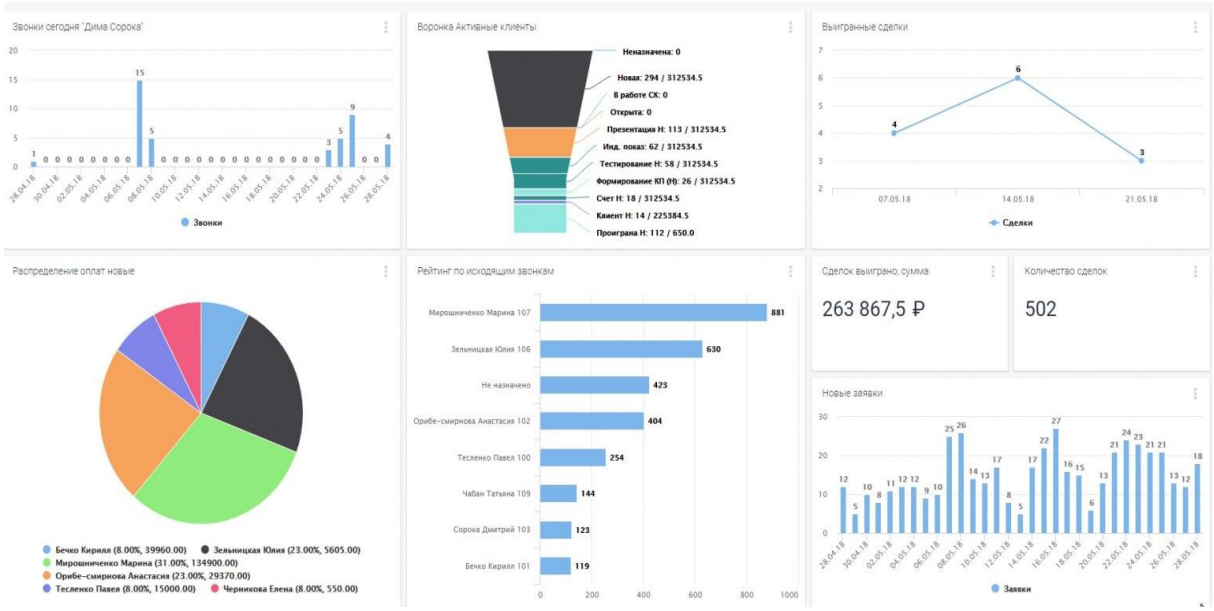


Рисунок 1.1 - Интерфейс SaleSap

Недостатки CRM.

На самом деле внедрение такого ПО предполагает много сложностей. Стоит рассмотреть такие моменты.

Высокая цена.

Достаточно приемлемая программа стоит денег. Цена использования облачных CRM-программ начинается от 3000 тенге за пользователя в месяц. Одним словом, чем больше штат, тем больше затрат. Есть необходимость доплатить за интеграцию программ. Хотя у множества CRM-программ есть бесплатные версии, но нужно смотреть правде в глаза понимать, что функционал будет меньше, и нет смысла от такой интеграции.

Сопротивление сотрудников.

Главной проблемой является обучаемость сотрудников. Чтобы научить сотрудников, есть необходимость произвести лишние затраты и время на обучения использования.

Саботаж со стороны менеджеров.

Большинство менеджеров привыкло бездельничать на рабочем месте и производить холодные звонки своих клиентов. Менеджеры могут просто отказаться от программы или сознательно портить работу. Проблема в том, что программа позволяет следить за персоналом, а им зачем это?

Маленький эффект от интеграции CRM-системы.

Иметь в наличие CRM-системы на предприятии не дает гарантию больших продаж, а также потока клиентов, так как:

- CRM не превратит плохого работника в хорошего;
- CRM не покажет верные отчеты, если в программу будут внесены не все данные;
- CRM не напомнит о встрече с клиентом, если не будут поставлены такие задачи и т.д.

В итоге, таким инструментом нужно уметь пользоваться, а, иначе, не стоит рассчитывать на рост прибыли. Необходимо изначально понимать, зачем внедряете CRM, а также постоянное совершенствование своего бизнес-процесса.

SaleSapCRM:

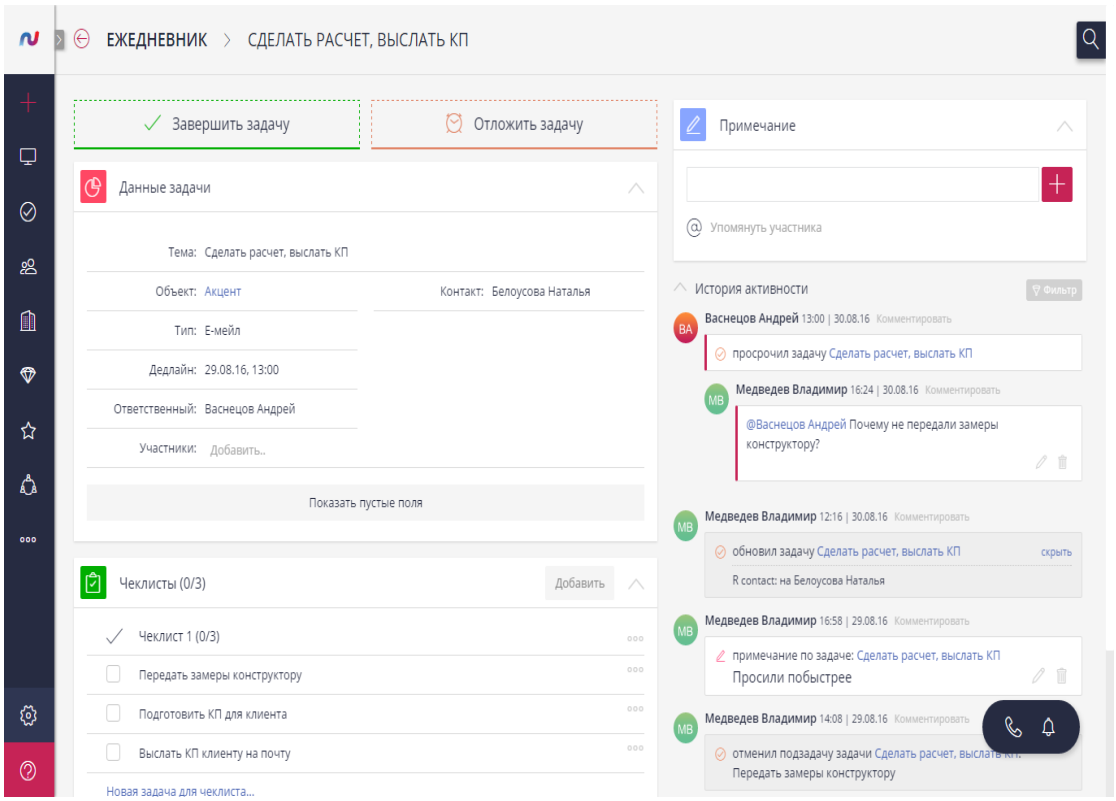


Рисунок 1.2- Ежедневник SaleSap

SalesapSRM начала работу в 2015 году и с этого же приступила к интеграции своей разработанной CRM-системы для контроля продаж. На июль 2017 года, около 3000 предприятий малого и среднего бизнеса из самых разных отраслей пользуются данным облачным сервисом.

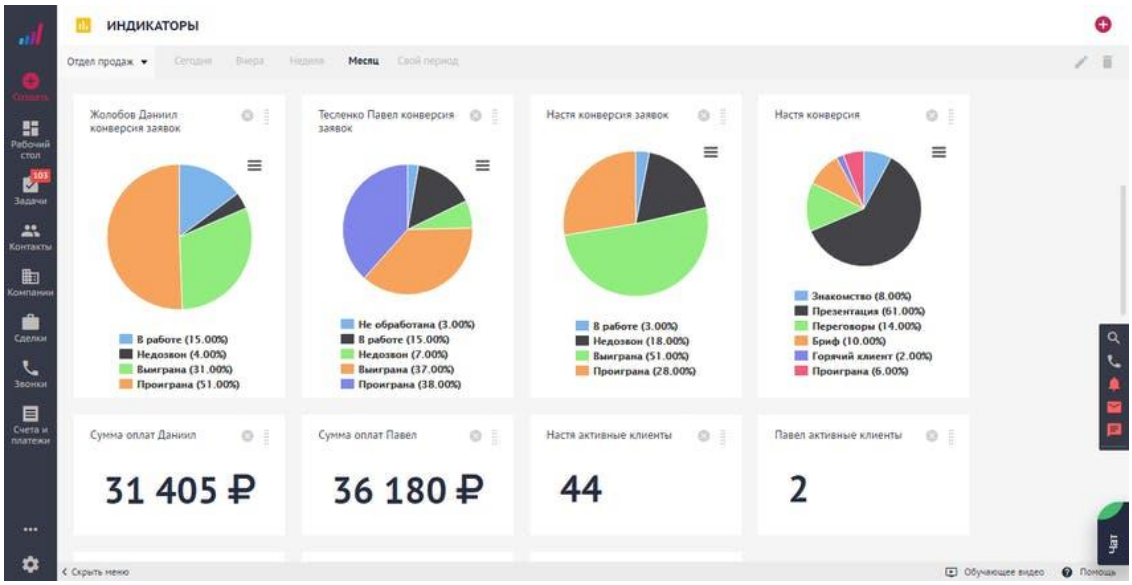


Рисунок 1.3 - Индикаторы SaleSap

Как известно, CRM-системы могут быть компьютерными и облачными. В первом случае, есть необходимость установки на компьютер, при этом все данные хранятся на сервере клиента, а, во втором случае, облачные CRM-системы нет необходимости в установке, так как все данные, а также само программное обеспечение находятся и хранятся на сервере разработчика.

Интерфейс представлен на рисунке 1.4.

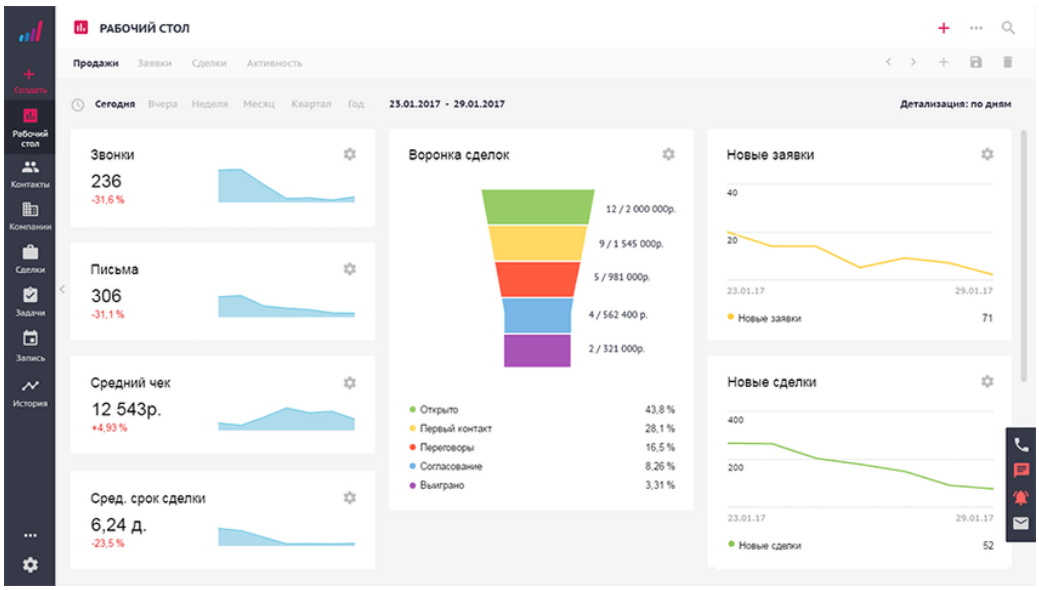


Рисунок 1.4 - Рабочий стол SaleSap

Возможности.

Возможность облачной системы SalesapCRM имеет в себе все основные параметры, которыми должна владеть CPM-система для эффективного контроля бизнеса:

- контроль клиентской базой;
- контроль продажами;
- контроль задачами;
- автоматизация бизнес-процессов.

Редактирование пользователя.

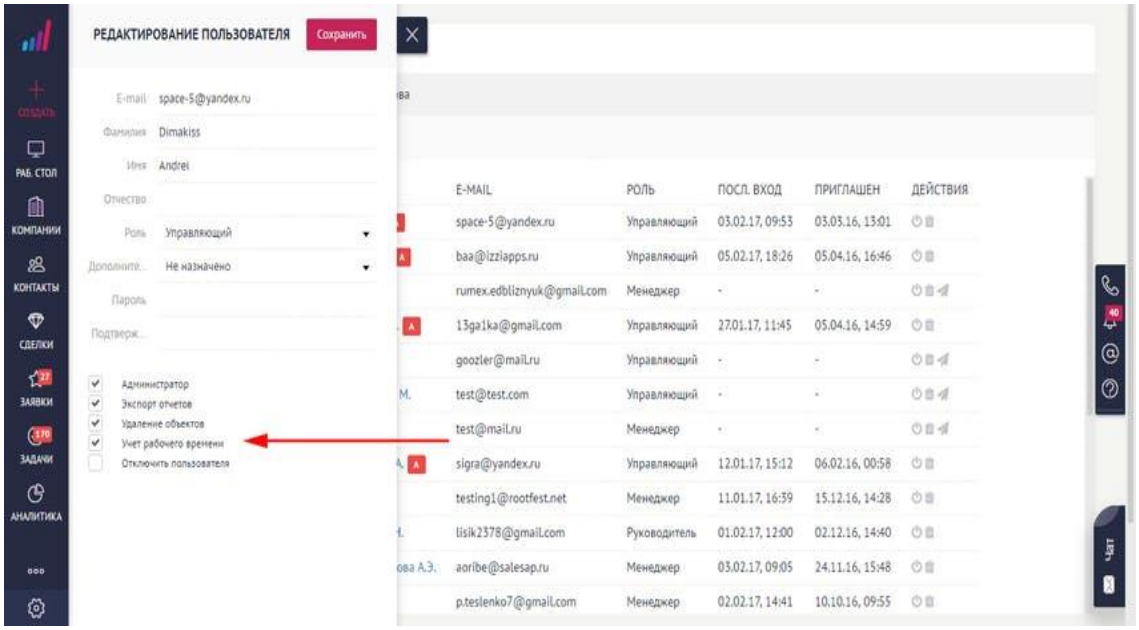


Рисунок 1.5 - Редактирование пользователя

Стоит отметить, что на основе универсальной CRM-системы для бизнеса произведен целый ряд SalesapCRM для разных отраслей, среди которых можно выбрать ту, которая подходит для требуемого бизнеса.

Так, в ассортименте представлены CRM-решения для:

- отдела продаж;
- малого бизнеса;
- агентства недвижимости;
- предприятий сферы услуг;
- парикмахерских и салонов красоты;
- автосервиса;
- фитнес-клуба;
- юридической фирмы;
- интеграция с другими системами.

Важным достоинством CRM-системы для продаж являются многочисленные интеграции, в частности:

- с IP-телефонией;
- с сервисом смс-рассылок;
- с сайтом;
- с электронной почтой;
- по API;
- с Телеграм.

На сайте имеется SalesapCRM 4 тарифных плана, среди которых есть бесплатный. Так как SalesapCRM является облачным, то необходимо ежемесячно производить оплату.

Бесплатный.

Имеется возможность подключить только три пользователя, объем хранилища - всего 200 Мб, можете импортировать данные до 500 шт, очень маленький функционал.

Старт.

За данный тариф необходимо производить ежемесячную оплату в 4000 тенге, если оплатите на полгода вперед, то можете получить скидку. Есть имеете возможность подключить только пять пользователя, есть автоматизация бизнес-процессов, а также размер хранилища до 1 Гб. Можете подключить один адрес электронной почты. Начиная с этого тарифа, есть интеграция смс-рассылок.

Компания.

С этого тарифа уже необходимо платить за каждого пользователя по 2000 тенге. В этом тарифе больше функций, а также возможностей для своих пользователей. К примеру, хранилища на каждого пользователя до 500 Мб, также можно подключить каждому пользователю по 2 электронных ящика.

Корпорация.

Тариф, открывающий перед пользователями больше возможностей, и стоит в месяц 2500 тенге за одного пользователя. Он позволяет пользоваться всеми имеющимися функциями без ограничений и подключать любое количество пользователей.

Выводы.

Несмотря на наличие базового (бесплатного) варианта программы, за 15 дней работы в полной версии CPM-системы настолько привыкаешь к ее достоинствам, что понимаешь – ограниченный функционал не даст бизнесу развиваться полноценно.

Однако для того, чтобы платные версии «работали» на полную мощность и позволяли получить максимальную отдачу от их внедрения, стоит особое внимание уделить настройкам и тщательному изучению всех функций и возможностей SalesapCRM. Идеально походит, если нужно заточенная под нишу CRM.

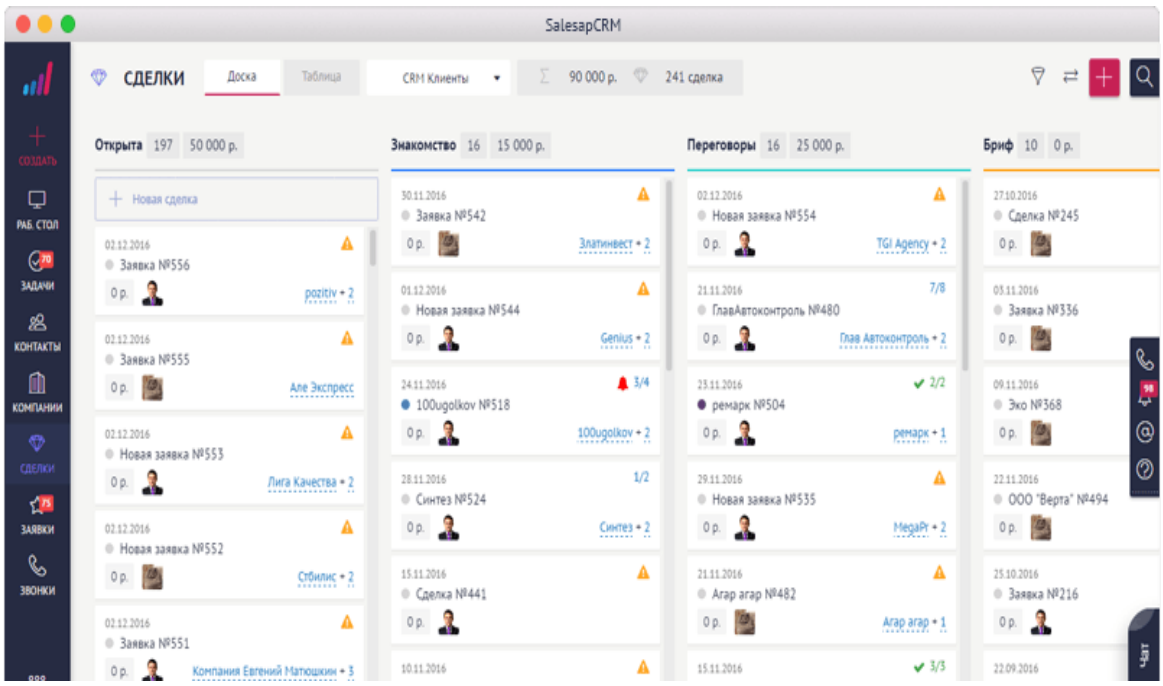


Рисунок 1.6 - Сделки SaleSap

Программа позволит по максимуму автоматизировать все бизнес-процессы, наладить грамотное обслуживание клиентов (что увеличит их лояльность), повысит уровень взаимодействия сотрудников компании и качество их работы. Итогом станет рост числа постоянных клиентов и, соответственно, прибыли.

1.2 Решение проблем

В суровые реалии современных дней, предприятиям трудно вести бизнес без использования своей информационной системы. Информационная система является организационно-упорядоченной взаимосвязанной совокупностью средств, а также методов информационных технологий, которые используется для хранения, обработки и выдачи информации для достижения поставленной цели. В данный момент информационная система используется на персональном компьютере в качестве основного технического средства переработки информации. В больших предприятиях вместе с персональным компьютером. Мэйнфрейм или суперЭВМ также входит в состав технической базы информационной системы [2].

Информационные системы дают возможность сбора, хранения, обработку, поиск, а также выдачу информации, которая необходима в процессе принятия решений задач из любой области. ИС дают возможность анализировать проблемы и производить новые продукты. Хотя техническое производство информационной системы ничего не значить, если не будет учтена роль человека, которому предназначена информация в системе.

Можно выделить следующие положительные стороны от внедрения информационных систем в организации:

- улучшение качества;
- увеличение эффективности;
- улучшение процесса принятия решения;
- улучшение коммуникации;
- улучшение использования знаний.

ИС дает возможность улучшить товары или услуги, при этом повышая качество, уменьшая издержки или добавляя желательные свойства. ИС в основном используется для более быстрого обслуживания, а также для быстрого поиска эксперта, который имеет опыт и знания, необходимые в определенном случае.

ИС также увеличивает эффективность и производительность. К примеру, в организации ИС может управлять оборудованием и мгновенно извещать оператора при возникновении неполадок. В результате есть возможность получить продукцию более высокого качества с меньшим количеством брака.

ИС дает возможность использовать своевременную и надежную информацию, которая позволяет модернизировать процесс принятия решений. Собранная информация о продажах оптовым поставщиком, дает возможность своевременно обнаружить спад в продаже отдельных товаров, давая возможность выяснить причины и принять меры.

ИС модернизирует коммуникации. Находясь в командировках, дома или просто в другом здании, персонал может использовать компьютеры для входа в сеть своего предприятия, чтобы отправлять и получать сообщения, просматривать файлы данных предприятия, исследовать проблемы, готовить презентации.

ИС модернизирует использование знаний. К примеру, консалтинговые фирмы обеспечивают возможностью своим клиентам планировать налоги, которые использует экспертные системы по налогообложению, где сконцентрированы знания лучших экспертов фирмы.

Абсолютно любому предприятию желательно иметь несколько локальных ИС для разных предназначений, которые могут взаимодействовать между собой и, которые могут поддерживать контрольные решения на всех уровнях.

Между локальными ИС происходят связи разного характера и назначения. В первом случае, определенные системы могут иметь связь с огромным количеством работающих в предприятии систем, а также иметь выход во внешнюю среду, во втором случае, могут иметь связь только с одной или несколькими родственными.

В итоге, использование информационной технологии на современной компании во многом показывает его дальнейшее развитие. Информационные продукты улучшают и рационализируют систему управления оперативной деятельности предприятия, обеспечивают возможностью управления

взаимоотношения с заказчиками и поставщиками, управлять процессом продаж. В результате их внедрения на предприятии происходит сокращение операционных издержек, получение дополнительных доходов вследствие увеличения оборота и роста инвестиционной привлекательности компании [3].

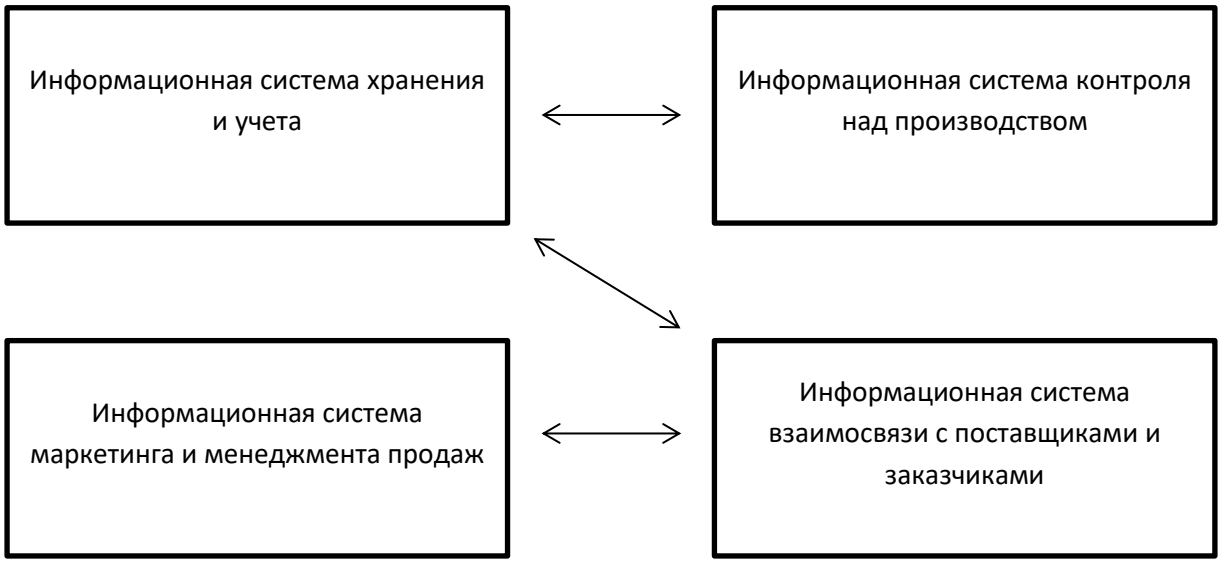


Рисунок 1.7 - Управленческие решения на разных уровнях

1.3 Постановка задачи дипломного проекта

В проекте была поставлена задача разработать информационную систему контроля заказов с выдачей электронного чека и поддержкой обратной связи с клиентом через средство обмена сообщениями.

Приложение должно упрощать деятельность предприятия и полностью ее автоматизировать, а также ускорить весь процесс принятия решений.

Данное приложение имеет богатый функционал, которая будет очень полезна для предприятия и будет использована в полном объеме.

Перед разработкой информационной системы были поставлены следующие задачи:

- сбор исходных материалов;
- эскизный проект;
- проектирование информационной системы;
- технический проект;
- разработка информационной системы;
- рабочий проект;
- внедрение.

На основании проведенного анализа существующих информационных систем, можно сделать следующие выводы:

Внедрение информационных систем для облегчения и улучшения рабочих процессов очень актуально и применяется многими предприятиями на сегодняшний день. Информационная система может не только повысить эффективность при работе с клиентами, но и значительно снизить потенциальные расходы компании. Однако, многие системы, использующиеся автосервисами для облегчения рабочих процессов, порой не приносят ожидаемого результата. Высокая стоимость, ограниченность функций а также плохая репутация таких систем являются этому причиной. Другим фактором неэффективности таких систем является сложный и непонятный интерфейс, что является невыгодным вариантом для автосервисов из-за необходимости обучения персонала. На примере рассмотрения CRM-системы SaleSap, можно сделать вывод, что многие автоматизированные информационные системы создаются для использования в различных сферах, чему следует ограниченность функциональности в более узких сферах деятельности, таких как автосервис.

2 Описание и обоснование выбора языка программирования и СУБД

2.1 Проектирование информационной системы

На рисунке 2.1 показана диаграмма возможностей для пользователя.

Через информационную систему можно использовать следующие функции пользователя:

- импорт заказов (необходимо заполнить ячейки в самой информационной системе);
- экспорт из БД (имеется возможность экспорта загруженных работ в базу данных);
- просмотр заказов (просмотр заказов в самой программе);
- чат (есть возможность вести беседу с клиентом);
- регистрация пользователя (имеется возможность добавлять новых пользователей).

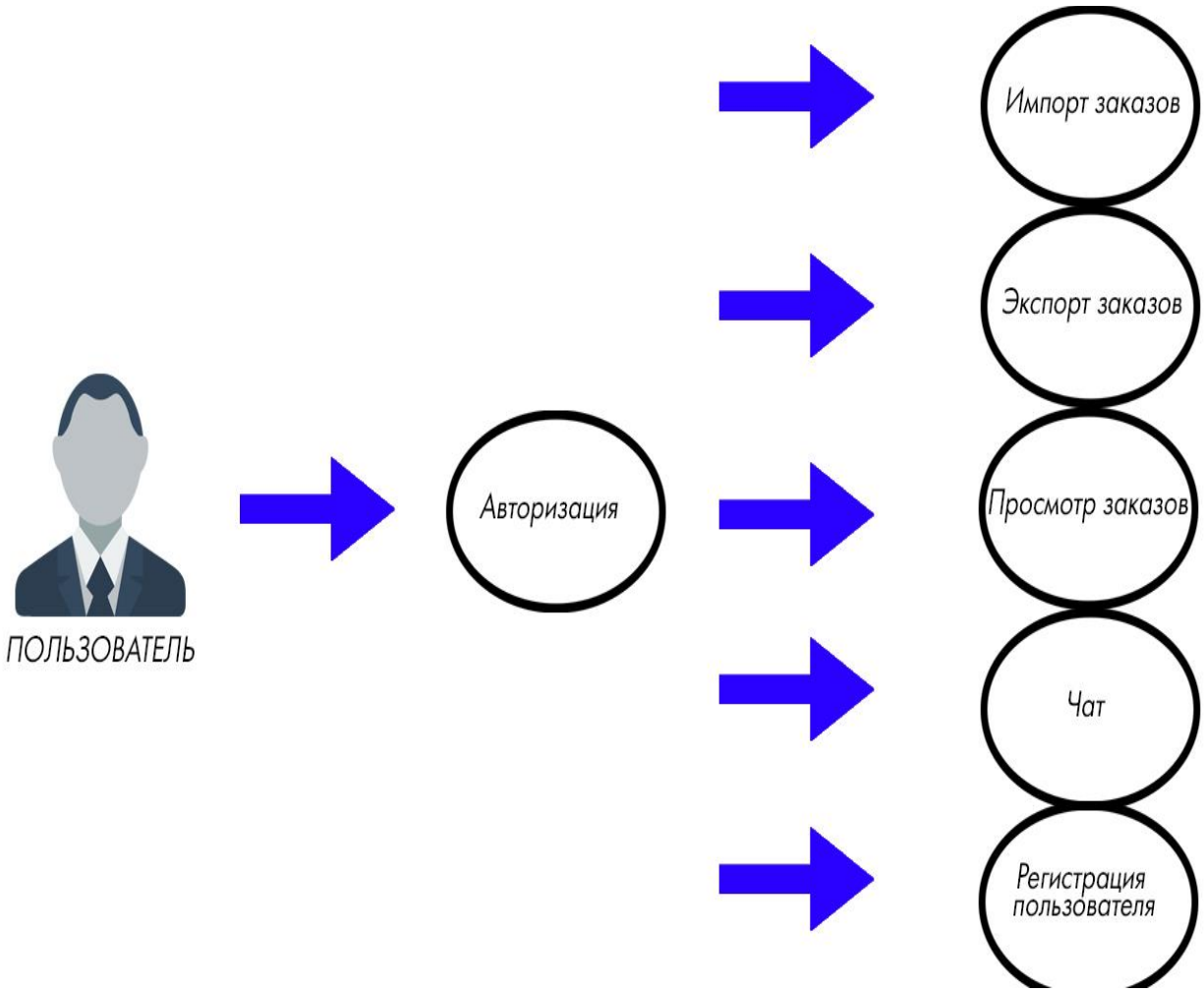


Рисунок 2.1 – Диаграмма возможностей для пользователя

На рисунке 2.2 и рисунке 2.3 представлена физическая модель и схема базы данных. Рисунок содержит детали физической модели базы данных.

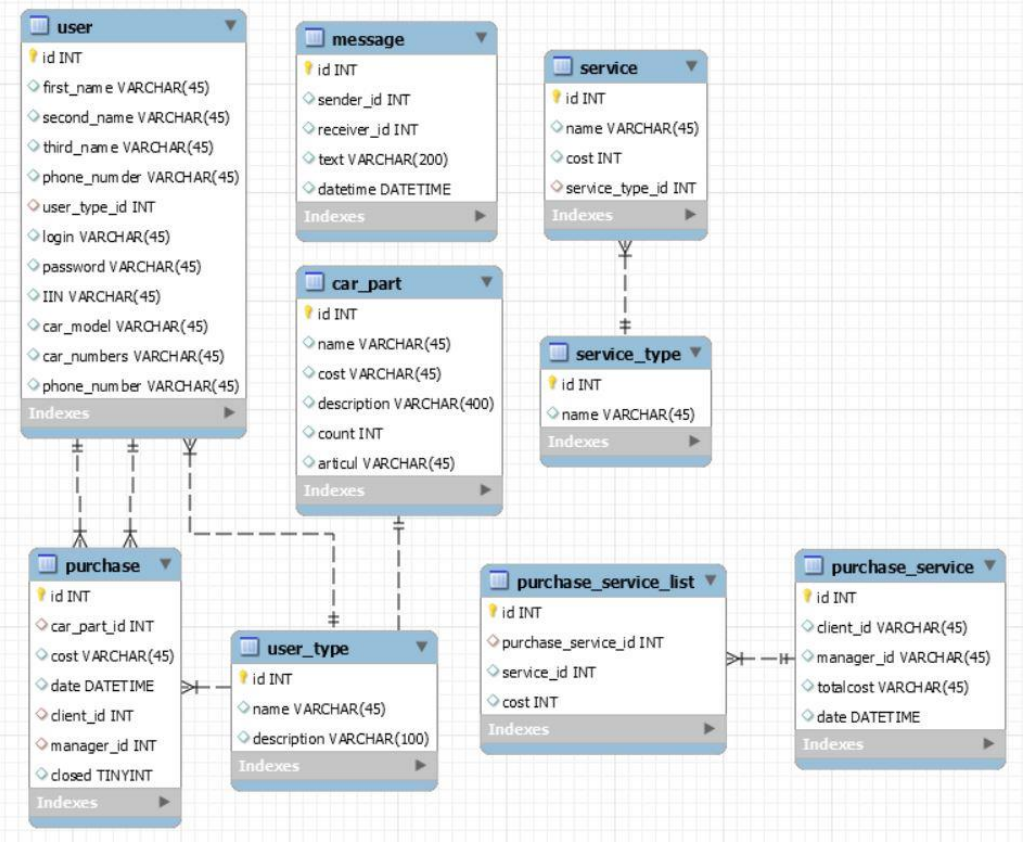


Рисунок 2.2 – Физическая модель базы данных

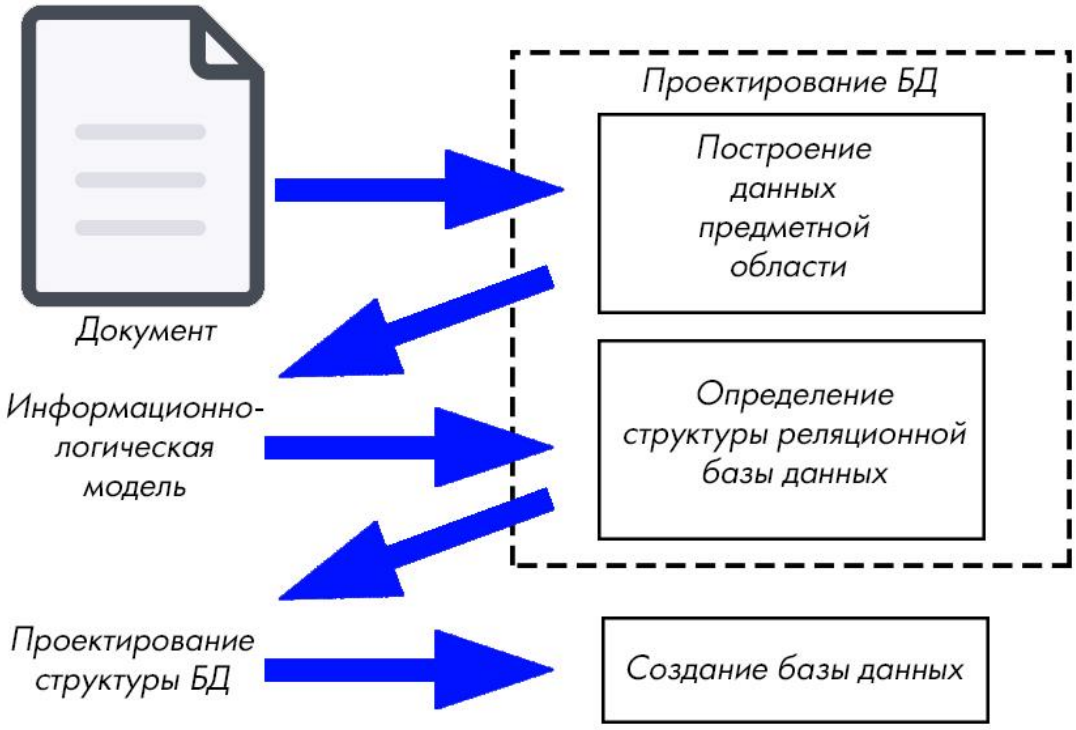


Рисунок 2.3 – Схема базы данных

На рисунке 2.4 представлена диаграмма последовательности работы информационной системы.

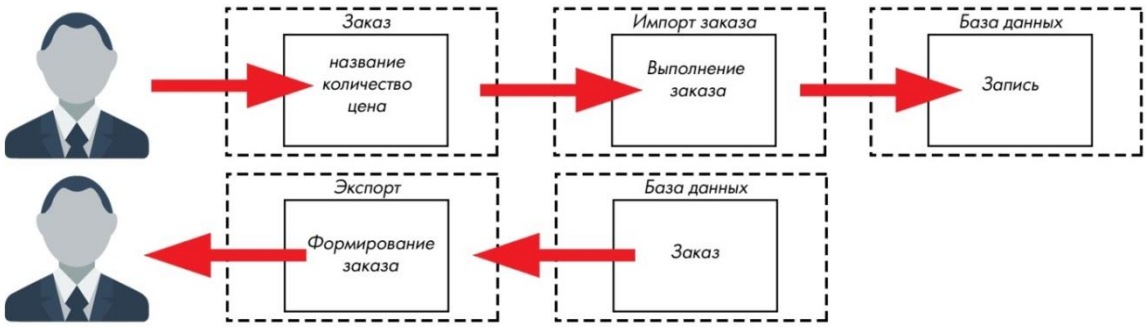


Рисунок 2.4 – Диаграмма последовательности

2.2 Анализ современных языков программирования

Среда разработки - это программная система, которую разработчики используют для создания программного обеспечения.

- текстовый редактор;
- компилятор и / или интерпретатор;
- средства автоматизации сборки;
- отладчик.

В некоторых случаях оно содержит инструменты для реализации в системах управления версиями и различные инструменты для облегчения разработки графического интерфейса пользователя. Значительная часть современных сред разработки включает браузер классов, инспектор объектов и диаграмму иерархии классов, которая используется в разработке программного обеспечения методом объектно-ориентированного программирования.

Встроенные среды разработки предназначены для повышения эффективности разработчиков, поэтому связаны с компонентами простых интерфейсов. Программисту это даст возможность осуществлять меньше действий для переключения разных режимов, в отличие от дискретных сред разработки.

Обычно среда разработки приспособлена на один язык программирования, предоставляя набор функций.

Есть универсальные IDE с поддержкой нескольких языков:

- Eclipse;
- ActiveState Komodo;
- NetBeans последние версии;
- Microsoft Visual Studio.

IDE обычно представляет собой единственную программу, в которой происходит вся разработка. Среда разработки также содержит множество функций для разработки, изменения, компиляции, развертывания и отладки

программного обеспечения. Основная цель среды разработки - абстрагировать конфигурацию, необходимую для объединения утилит командной строки в одном модуле, что позволяет сократить время на изучение языка и повысить производительность разработчика.

Интегрированная среда разработки представляет собой набор программных инструментов. Он поддерживает абсолютно все этапы производства программного обеспечения. Начиная с написания исходного кода программы до ее компиляции и отладки, которое налаживает простое и быстрое взаимодействие с другими инструментами.

Другими словами, внедренные среды разработки не относятся к числу средств отладки. Отладка является лишь одной из свойств интегрированных сред, которые являются фундаментом любой визуальной среды разработки или RAD-среды.

Первый этап написания программы строится следующим образом:

- исходный код набирается, используя любой текстовый редактор;
- после завершения набора текста, работа с текстовым редактором прекращается и запускается кросс компилятор;
- как правило, вновь написанная программа имеет синтаксические ошибки, и компилятор сообщает о них на консоль оператора;
- оператор должен найти и исправить выявленные ошибки.

Цикл такого типа может повторятся множество раз. Если программа имеет огромный объем, она также будет долго редактироваться или улучшаться. Даже этот первый этап может потребовать много усилий и времени. Далее начинается этап отладки программы, в котором к редактору с компилятором будут добавлены эмулятор или симулятор.

Работа в интегрированной среде дает программисту:

- возможность использования встроенный текстовый редактор;
- использование автоматической диагностики, которая выявляет ошибки при компиляции;
- возможность работы над несколькими проектами одновременно;
- возможность минимизировать количество перекомпиляций;
- возможность импорта программы в имеющиеся средства отладки, а также возможность работы с ними без выхода из оболочки;
- возможность, подключения к оболочке практически любых программных средств.

Python.

Является популярным объектно-ориентированным языком, известный благодаря простоте своей структуры. Не требует углубленных знаний от разработчика и подходит для начинающих программистов.

Данный язык имеет огромное количество успешных стартапов.

Проекты, разработанные на языке Python:

- YouTube;
- Instagram;
- Pinteres;

- SurveyMonkey.

Полезные ресурсы:

Java.

Один из самых популярных языков объектно-ориентированного программирования. В основном используется в больших проектах. Не является простым по структуре, но поддается осваиванию. Часто используется в корпоративных приложениях. Операционная система Android является одним из главных примеров использования языка Java.

C / C++.

Считаются сложными по своей структуре и являются основой многих низкоуровневых систем. Язык очень востребован и многофункционален по сравнению с другими языками ООП. Существует множество программ написанных на этих языках.

JavaScript.

Является простым по своей структуре и широко используется в сфере веб-программирования. Легкость использования и быстрота позволяют программистам эффективно решать проблемы. На данный момент занимает первое место в рейтинге популярных языков веб-программирования.

PHP.

Такой же язык, как JavaScript, но используется исключительно для создания сайтов. Освоение не требует больших навыков и можно быстро научиться разрабатывать на этом языке.

C#.

Это универсальный язык программирования на уровне с Java, но у него есть несколько преимуществ, как и огромный минус. Язык довольно сложен для изучения и требует от разработчика отличных навыков, но все это более чем компенсируется гибкостью функционала и позволяет создавать разные программы. От небольших приложений до огромных веб-сервисов. Основными возможностями языка являются полная поддержка классов и объектно-ориентированного программирования, ясное представление базовых типов и поддержка автоматической генерации XML-документации. Сейчас язык активно используется для разработки игр [4].

2.3 Выбор языка программирования C#

Универсальность языка программирования C# является причиной его использования в данном проекте. Среда разработки Visual Studio содержит в себе множество инструментов для написания кода на языке C#.

C# является объектно-ориентированным языком, который способен использовать компонентно-ориентированное программирование.

Разработанный компанией Microsoft, C# является популярным современным языком программирования. C# в основном используется для работы с программами на ПК, разработки объемных веб-сервисов или мобильных приложений.

Microsoft начала разрабатывать язык в 1998 году. В 2001 году произошел релиз первой версии. Во главе разработки стоял Андерс Хейлсберг. C# часто обновляется, а нынешние доработки, исправление багов и увеличение директив происходят практически на постоянной основе.

В результате, язык получился достаточно гибким. Благодаря чему, на нем могут писать все, что угодно, начиная с мелких веб-приложений до крупных игр. Это стало доступным из-за удобного синтаксиса, огромному количеству фреймворков и библиотек.

По своей натуре, C# является инструментом, достойным внимания. Microsoft и по сей день уделяет огромное внимание поддержке языка, а потому разработчики регулярно получают обновления и дополнения, с помощью которых исправляются выявленные ошибки и баги.

Инструментарий C# позволяет решать широкий круг задач, язык действительно очень мощный и универсальный. Его используют для разработки:

- приложений для WEB;
- различных игровых программ;
- приложений для платформ Андроид или iOS;
- программ для Windows.

Из-за огромного набора инструментов, список возможностей не имеет ограничений. Однако все это может быть реализовано с помощью других языков, но большинство из них разработаны для особых задач. C# имеет все решения для различных задач, которые помогают программисту эффективно использовать все функции и сократить время разработки.

Другой особенностью является автоматическая очистка памяти от ненужных объектов, которые не используются программой.

Этот инструмент дает возможность выявить и обработать ошибки в исходном коде.

В C# есть общая система работы с типами, начинающаяся от примитивов и заканчивающаяся комплексными, пользовательскими наборами. Объединенный набор операций используется для обработки и хранения значений типизации. Также имеется возможность использования ссылочных типов пользователя, что позволяет динамически выделять память под объект или хранить упрощенную структуру в сети.

Для работы приложений на C# необходимо установить и настроить платформу NET Framework. Платформа распространяется абсолютно бесплатно и применяется крайне широко, поэтому проблем не должно возникнуть с пользовательскими устройствами. Платформа интегрирована в установочный пакет Windows.

Текст программы переводится в промежуточный язык IL, после работы компилятора, который понимает CLI. IL и все необходимые ресурсы, которые включают строки и рисунки в формате BMP, которые сохраняются на жесткий диск в виде dll или exe. Из таких файлов формируется сборка

приложения, включающая в себя описание с полной информацией обо всех важных параметрах работы.

При выполнении программы CLR обращается к сборке и производит действия в зависимости от полученных сведений. Если код написан правильно и проходит проверку безопасности системы, производится компиляция из IL в инструкции в машинные команды. Среда CLR попутно выполняет еще много побочных функций:

- удаление «программного» мусора;
- работа с исключениями;
- распределение ресурсов;
- контроль типизации;
- управление версиям;
- типизация;
- управление версиями.

В итоге код C# считается контролируемым, т.е. он компилируется в двоичный вид на пользовательском устройстве с учетом особенностей установленной системы [5].

2.4 Особенность использования СУБД

ИС заставляет использовать базу данных. Это связано непосредственно с хранением всех заказов предприятия. В Visual Studio можно использовать локальную базу данных. Можно без проблем использовать синтаксис языка MySQL.

MySQL – гибкая субд, которая выражается поддержкой большого количества типов таблиц:

- пользователи могут выбрать как таблицы типа MyISAM;
- имеет возможность текстового поиска, так и таблицы InnoDB;
- имеет возможность транзакции на уровне отдельных записей.

СУБД MySQL кладется с особенным типом таблиц EXAMPLE, которая показывает принципы производства новых типов таблиц. В СУБД MySQL перманентно появляются новые типы таблиц.

MySQL имеет API и коннекторы для многих языков [6]:

- Delphi;
- C/C++/C#;
- JAVA;
- PHP;
- Python.

Данный список является не полным.

На основании рассмотренных выше инструментов разработки можно сделать следующие выводы:

Простой интерфейс среды разработки Visual Studio позволяет значительно улучшить и ускорить процесс написания кода. Язык программирования C# является очень удобным инструментом для создания комплексных программ и, благодаря встроенным библиотекам, дает возможность улучшить функциональность приложений. Работа информационной системы автосервиса невозможна без наличия системы хранения данных о заказах, клиентах и т.д. Такой системой в данном проекте является субд MySQL. Совместное использование субд и программы в процессе разработки позволяет разработчикам создать полноценные информационные системы с различными функциями и возможностями, при этом сохраняя необходимую информацию в базе данных.

3 Разработка информационной системы

3.1 Программная реализация

Программа, как и любой другой проект на C# начинается с функции Main. Функция Main находится в главном классе программы, Program.

```
static void Main()
{

    Application.EnableVisualStyles();
    Application.SetCompatibleTextRenderingDefault(false);
    Application.Run(new Login());
}
```

Рисунок 3.1 – Функция Main

Здесь включаются визуальные стили, задаются значения по умолчанию во всем приложении для свойств UseCompitableTextRendering, определенного в некоторых элементах управления и запускается приложение с начальной формой авторизации – Login.

Для упрощения работы, отдельные функции, выполняющие обработку информации, были размещены в классе Program.

FormatToGrind превращает список словарей, представляющий собой список объектов, в матрицу.

```
public static string[,] FormatToGrid(List<Dictionary<string, string>> objects)
{

    string[] headers = objects[0].Keys.ToArray();
    string[,] grid = new string[objects.Count + 1, headers.Length];
    for (int i = 0; i < headers.Length; i++)
    {
        grid[0, i] = headers[i];
    }
    for (int i = 0; i < objects.Count; i++)
    {
        for (int j = 0; j < headers.Length; j++)
        {
            grid[i + 1, j] = objects[i][headers[j]];
        }
    }
    return grid;
}
```

Рисунок 3.2 – FormatToGrid

ToJsonArray превращает список словарей, представляющий собой список объектов, объект JsonArray.

```
public static string ToJsonArray(List<Dictionary<string, string>> dic)
{
    string s = "[";
    for (int i = 0; i < dic.Count; i++)
    {
        s += ToJson(dic[i]);
        s += ",";
    }
    s = s.Remove(s.Length - 1);
    s += "]";
    return s;
}
```

Рисунок 3.3 – ToJsonArray

FromJsonArray превращает строку являющуюся объектом в формате JSON, в список словарей, представляющий собой список объектов.

```
public static List<Dictionary<string, string>> FromJsonArray(string json)
{
    List<Dictionary<string, string>> diclist = new List<Dictionary<string, string>>();
    json = json.Replace("[", "").Replace("]", "");
    var objects = json.Replace("{", "").Split('}');

    foreach (var obj in objects)
    {
        if (obj != "")
        {
            var val = FromJson(obj);
            diclist.Add(val);
        }

    }
    return diclist;
}
```

Рисунок 3.4 – FromJsonArray

ToJson превращает словарь, представляющий собой объект, в строку в формате Json. Ввиду несовершенности данного метода, в дальнейшем использовался класс JObject, имеющий метод FromObject.

```
public static string ToJson(Dictionary<string, string> dic)
{
    string s = "{";
    foreach (var kvp in dic)
    {
        if (kvp.Value[0] != '[')
        {
            s += "\"" + kvp.Key + "\":" + "\"" + kvp.Value + "\"";
        }
        else
        {
            s += "\"" + kvp.Key + "\":" + kvp.Value;
        }

        s += ",";
    }
    s = s.Remove(s.Length - 1);
    s += "}";
    return s;
}
```

Рисунок 3.5 – Превращение из словаря в строку

ToJson превращает строку в формате JSON, представляющую собой объект, в список словарей. Ввиду несовершенности данного метода, в дальнейшем использовался класс JObject, имеющий метод Parse.

```
public static Dictionary<string, string> FromJson(string json)
{
    Dictionary<string, string> dic = new Dictionary<string, string>();
    var objFields = json.Split(',');
    foreach (var o in objFields)
    {
        if (o != "")
        {
            var o_ = o.Split(':');
            dic.Add(o_[0].Replace("\"", ""), o_[1].Replace("\"", ""));
        }

    }

    return dic;
}
```

Рисунок 3.6 – Превращение из строки в словарь

GetColumnIndex возвращает индекс колонки таблицы DataGridView, с данным HeaderText.

```
public static int GetColumnIndex(string headerName,DataGridView dgv)
{
    int m = dgv.ColumnCount;
    int ret = -1;
    for (int i = 0; i < m; i++)
    {
        if (dgv.Columns[i].HeaderText == headerName)
        {
            ret = i;
        }
    }
    return ret;
}
```

Рисунок 3.7 – Возвращение индекса колонки

ToQueryString принимает словарь, представляющий из себя объект и возвращает в себя строку query для помещения в строку запроса.

```
public static string ToQueryString(Dictionary<string,string> dic)
{
    string s = "";
    foreach (KeyValuePair<string,string> kvp in dic)
    {
        s += kvp.Key;
        s += "=";
        s += kvp.Value;
        s += "&";
    }
    if (s.Length > 0)
        s = s.Substring(0, s.Length - 1);
    return s;
}
```

Рисунок 3.8 – ToQueryString

GetRequest посылает get запрос по указанному адресу с добавлением строки query.

```
public static string GetRequest(string url, Dictionary<string, string> query)
{
    query.Add("login", Login.login);
    query.Add("password", Login.password);
    query.Add("user_type", Login.user_type.ToString());
    string queryString = ToQueryString(query);


    url += "?" + queryString;
    var webClient = new WebClient();
    webClient.Encoding = Encoding.UTF8;
    return webClient.DownloadString(url);

}
```

Рисунок 3.9 – GetRequest

SelectTableRequest посылает запрос на получение значений таблицы с использованием строки query для условий запроса. Возвращает строку, представляющую из себя массив в формате JSON.

```
public static string SelectTableRequest(Table table,Dictionary<string, string> query = null)
{
    if (query == null)
        query = new Dictionary<string, string>();

    var keysToUpdate = new List<string>();

    foreach (var entry in query)
    {

        keysToUpdate.Add(entry.Key);

    }

    foreach (string keyToUpdate in keysToUpdate)
    {
        string value = query[keyToUpdate];

        string newKey ="option_" + keyToUpdate;

        // increment the key until arriving at one that doesn't already exist

        query.Remove(keyToUpdate);
        query.Add(newKey, value);
    }


    query.Add("login", Login.login);
    query.Add("password", Login.password);
    query.Add("user_type", Login.user_type.ToString());
    query.Add("table",table.ToString());
    string queryString = ToQueryString(query);

    string url = domain+selectTableAdress;
    url += "?" + queryString;

    var webClient = new WebClient();
    webClient.Encoding = Encoding.UTF8;

    return webClient.DownloadString(url);

}
```

Рисунок 3.10 – SelectTableRequest

FormatDataGridView изменяет визуальные компоненты таблицы под определенный стандарт.

```
public static void FormatDataGridView(DataGridView dgv)
{
    dgv.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;
    dgv.AutoSizeRowsMode = DataGridViewAutoSizeRowsMode.DisplayedCells;
    dgv.DefaultCellStyle = new DataGridViewCellStyle(dgv.DefaultCellStyle) { WrapMode = DataGridViewTriState.True };
}
```

Рисунок 3.11 – FormatDataGridView

GetHeaderTexts возвращает список заголовков столбцов таблицы.

```
public static List<string> GetHeaderTexts(DataGridView dgv)
{
    List<string> headerTexts = new List<string>();

    for (int i = 0; i < dgv.ColumnCount; i++)
    {
        headerTexts.Add(dgv.Columns[i].HeaderText);
    }
    return headerTexts;
}
```

Рисунок 3.12 – GetHeaderTexts

CreateTextBoxedColumn создает колонку полей ввода TextBox и строковых подсказок Label, соответствующих названию TextBox и возвращает словарь, в котором ключ string соответствует строковой подсказке и “названию” TextBox, а значение это ссылка на сам TextBox.

Это позволяет динамически работать с полями ввода TextBox.

```
public static Dictionary<string, TextBox> CreateTextBoxesColumn(List<string> labels, Control parent,int space,int horizontalPadding)
{
    Dictionary<string, TextBox> outDictionary = new Dictionary<string, TextBox>();
    Point point = new Point(horizontalPadding, 5);
    for (int i = 0; i < labels.Count; i++)
    {
        var s = labels[i];

        Label lab = new Label();

        lab.Text = s;

        lab.Location = point;

        lab.Parent = parent;


        int height = lab.GetPreferredSize(Size.Empty).Height + space;
        int width = parent.Width - horizontalPadding * 3;
        point.Y += height;


        TextBox textB = new TextBox();

        textB.Location = point;

        Size buttonSize = lab.GetPreferredSize(Size.Empty);

        textB.Size = new Size(width, height);

        textB.Parent = parent;


        point.Y += Convert.ToInt32(height * 1.5f);

        outDictionary.Add(s, textB);


    }
    return outDictionary;
}
```

Рисунок 3.13 – CreateTextBoxedColumn

DictionaryToList позволяет превратить словарь в список путем удаления ключей. Индексирование в таком случае задается по порядку добавления элемента в словарь.

```
public static List<T> DictionaryToList<T>(Dictionary<string,T> dic)
{
    List<T> newList = new List<T>();
    foreach(var kvp in dic)
    {
        newList.Add(kvp.Value);
    }
    return newList;
}
```

Рисунок 3.14 – DictionaryToList

TableBaseObjectsToGrid метод был задуман для превращения списка объектов, наследующих от TableBase в матрицу строк для таблицы. В дальнейшем от этого метода пришлось отказаться в пользу более простого решения.

```
public static string[,] TableBaseObjectsToGrid(List<Tables.TableBase> list)
{
    string[,] grid = new string[list.First().GetType().GetProperties().Length,list.Count];
    for (int i = 0; i < list.Count; i++)
    {
        Type type = list[i].GetType();
        var properties = type.GetProperties();
        for (int j = 0; j < properties.Length; j++)
        {
            if (properties[j].GetType() == typeof(Tables.TableBase))
            {
                grid[i, j] = properties[j].GetValue(list[i]).ToString();
            }
        }
    }
    return grid;
}
```

Рисунок 3.15 – TableBaseObjectsToGrid

FromJArrayToDictionary был задуман для превращения объекта JArray, представляющего собой массив объектов JSON, в словарь, в котором ключем являлся id, а значением являлся сам объект. В дальнейшем от этого метода пришлось отказаться в пользу более простого решения.

```
public static Dictionary<string,T> FromJArrayToDictionary<T>(JArray array)
{
    Dictionary<string, T> dic = new Dictionary<string, T>();
    foreach(var j in array)
    {
        dic.Add(j.ToObject<Tables.TableBase>().id, j.ToObject<T>());
    }
    return dic;
}
```

Рисунок 3.16 – FromJArrayToDictionary

InsertIntoDataGridView принимает в себя JArray(список JObject) и DataGridView(объект таблицы) и заносит значения JArray в таблицу. Заголовки берутся из названий полей объекта, а значения в таблице это значения соответствующих полей в объектах JObject.

```
public static void InsertIntoDataGridView<T>(JArray jArray, DataGridView dgv)
{

    if (jArray.Count>0)
    {


        if (dgv.ColumnCount < jArray.Children<JObject>().ToArray()[0].Properties().Count())
            foreach (var j in jArray.Children<JObject>().ToArray()[0].Properties())
            {
                dgv.ColumnCount++;
                dgv.Columns[dgv.ColumnCount - 1].HeaderText = j.Name;
            }
        for (int i = 0; i < jArray.Count; i++)
        {
            List<object> row = new List<object>();
            var fields = typeof(T).GetFields(System.Reflection.BindingFlags.Public | System.Reflection.BindingFlags.Instance);
            for (int j = 0; j < fields.Length; j++)
            {
                var obj = jArray[i].ToObject<T>();
                var val = obj.GetType().GetField(fields[j].Name).GetValue(obj);
                if (val != null)
                    row.Add(val.ToString());
                else
                    row.Add(null);


            }
            dgv.Rows.Add(row.ToArray());
        }
    }
}
```

Рисунок 3.17 – InsertIntoDataGridView

DeleteFromTableRequest посылает запрос на удаление строки таблицы по строке query.

```
public static string DeleteFromTableRequest(Table table, string postString, string queryString = "")
{
    string url = domain + deleteTableAdress;

    var query = new Dictionary<string, string>();
    query.Add("table", table.ToString());
    query.Add("login", Login.login);
    query.Add("password", Login.password);
    query.Add("user_type", Login.user_type.ToString());

    queryString += ToQueryString(query);

    var httpWebRequest = (HttpWebRequest)WebRequest.Create(url + "?" + queryString);
    httpWebRequest.ContentType = "application/json";
    httpWebRequest.Method = "POST";
    using (var streamWriter = new StreamWriter(httpWebRequest.GetRequestStream()))
    {
        streamWriter.Write(postString);
        streamWriter.Flush();
        streamWriter.Close();
    }
    string result = "";
    var httpResponse = (HttpWebResponse)httpWebRequest.GetResponse();
    using (var streamReader = new StreamReader(httpResponse.GetResponseStream()))
    {
        result = streamReader.ReadToEnd();
    }
    return result;
}
```

Рисунок 3.18 – DeleteFromTableRequest

UpdateTableRequest посылает запрос на обновление значений элементов таблицы.

```
public static string UpdateTableRequest(Table table, string postString, string queryString = "")
{
    string url = domain + updateTableAdress;

    var query = new Dictionary<string, string>();
    query.Add("table", table.ToString());
    query.Add("login", Login.login);
    query.Add("password", Login.password);
    query.Add("user_type", Login.user_type.ToString());

    queryString += ToQueryString(query);

    var httpWebRequest = (HttpWebRequest)WebRequest.Create(url + "?" + queryString);
    httpWebRequest.ContentType = "application/json";
    httpWebRequest.Method = "POST";
    using (var streamWriter = new StreamWriter(httpWebRequest.GetRequestStream()))
    {
        streamWriter.Write(postString);
        streamWriter.Flush();
        streamWriter.Close();
    }
    string result = "";
    var httpResponse = (HttpWebResponse)httpWebRequest.GetResponse();
    using (var streamReader = new StreamReader(httpResponse.GetResponseStream()))
    {
        result = streamReader.ReadToEnd();
    }
    return result;
}
```

Рисунок 3.19 – UpdateTableRequest

InsertIntoTableRequest посылает запрос на добавление строк в таблицу на сервер.

```
public static string InsertIntoTableRequest(Table table,string postString, string queryString="")
{
    //Сделать обработку ошибки при пустом заказе
    string url = domain + insertTableAdress;

    var query = new Dictionary<string, string>();
    query.Add("table", table.ToString());
    query.Add("login", Login.login);
    query.Add("password", Login.password);
    query.Add("user_type", Login.user_type.ToString());

    queryString += ToQueryString(query);



    var httpWebRequest = (HttpWebRequest)WebRequest.Create(url + "?" + queryString);
    httpWebRequest.ContentType = "application/json";
    httpWebRequest.Method = "POST";
    using (var streamWriter = new StreamWriter(httpWebRequest.GetRequestStream()))
    {
        streamWriter.Write(postString);
        streamWriter.Flush();
        streamWriter.Close();
    }
    string result = "";
    var httpResponse = (HttpWebResponse)httpWebRequest.GetResponse();
    using (var streamReader = new StreamReader(httpResponse.GetResponseStream()))
    {
        result = streamReader.ReadToEnd();
    }
    return result;
}
```

Рисунок 3.20 – InsertIntoTableRequest

Table это enum элемент, означающий названия таблиц в базе данных. Значительно упрощает работу со множеством строк, использующихся в разных местах и уменьшает вероятность ошибки.

```
public enum Table
{
    car_part,
    message,
    purchase,
    purchase_service,
    purchase_service_list,
    service,
    service_type,
    user,
    user_type
}
```

Рисунок 3.21 – Table

Поля строк web адресов для запроса:

```
public static readonly string domain = "http://localhost/";
public static readonly string selectTableAdress = "api/table/select";
public static readonly string insertTableAdress = "api/table/insert";
public static readonly string updateTableAdress = "api/table/update";
public static readonly string deleteTableAdress = "api/table/delete";
```

Рисунок 3.22 – Поля строк web адресов для запроса

Domain отвечает за главный домен, на который отправляются запросы.
Класс Login отвечает за авторизацию и аутентификацию пользователя.
Поля класса Login:

```
public static readonly int user_type = 2;
private static readonly HttpClient client = new HttpClient();
public static string login;
public static string password;
public static int clientID;
```

Рисунок 3.23 – Класс Login

User_type это тип пользователя для программы, указывается единожды в программе. HttpClient client это вебклиент, отправляющий запросы на веб адрес. Login это поле логина, password это поле пароля. ClientID это id клиента в базе данных.

LoginPassword это query представление login и password.

```
ссылок: 3
public static string LoginPassword { get { return "login=" + login + "&" + "password=" + password; } }
```

Рисунок 3.24 – LoginPassword

SendRequest - это метод, посылающий запрос проверки действительности пароля и логина на такой тип пользователя на сервер. При получении утвердительного ответа, разрашается доступ и открывается главная форма и закрывается форма авторизации.

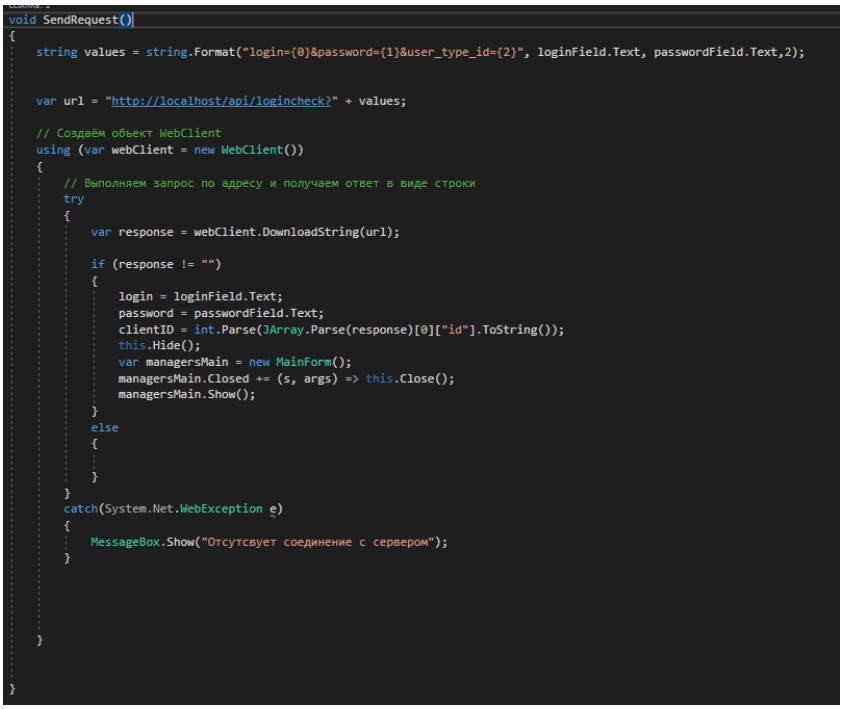


Рисунок 3.25 – SendRequest

При нажатии на кнопку “Войти” срабатывает метод:

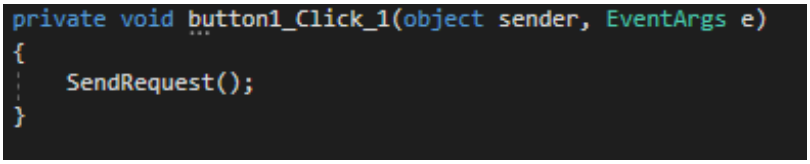


Рисунок 3.26 – Вызов метода

MainForm является главной формой программы, на которой содержатся все элементы.

MainForm имеет следующие поля:

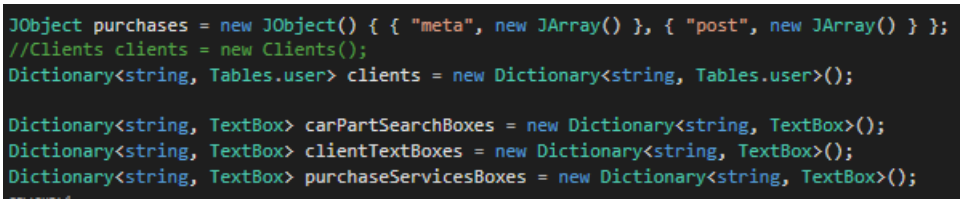


Рисунок 3.27 – MainForm

purchases это список покупок, имеющий два представления - meta, являющееся списком метаданных покупки для визуализации пользователю и post, являющееся списком данных для отправки на сервер.

clients это словарь клиентов, в котором ключом является id клиента.

carPartSearchBoxes это словарь полей ввода для использования их при поисковых запросах в таблице автомобильных запчастей.

clientTextBoxes это словарь полей ввода для использования их при отображении данных клиентов.

purchaseServiceBoxes это словарь полей ввода для использования их при поисковых запросах в таблицу использования сервисов.

При загрузке формы обновляются клиенты и таблицы и устанавливаются значения ввода в поля пользователя по умолчанию.

```
private void MainForm_Load(object sender, EventArgs e)
{

    UpdateClients();
    UpdateTables();

    SetActiveTextBoxes(false);
}
```

Рисунок 3.28 – Загрузка формы

В методе UpdateClients обновляется список клиентов. Для этого посылается соответствующий запрос на сервер, откуда приходит обновленный список клиентов, преобразующийся в нужный тип данных. При этом обновляются значения полей панели клиента.

```
void UpdateClients()
{
    string s1 = Program.SelectTableRequest(Program.Table.user, new Dictionary<string, string>() { { "user_type_id", "3" } });
    JArray clientsArray = JArray.Parse(s1);

    clients.Clear();
    foreach (var cl in clientsArray)
    {
        Tables.user user = cl.ToObject<Tables.user>();
        clients.Add(user.id, user);
    }

    JObject client = JObject.Parse(JsonConvert.SerializeObject(new Clients.Client()));
    List<string> clientFields = new List<string>();
    foreach (var k in client)
    {
        clientFields.Add(k.Key);
    }
    clientTextBoxes = Program.CreateTextBoxesColumn(clientFields, clientSidePanel, 10, 10);

    foreach (var kvp in clientTextBoxes)
    {
        kvp.Value.BorderStyle = BorderStyle.None;
        kvp.Value.ReadOnly = true;
        //kvp.Value.Multiline = true;
        kvp.Value.Margin = new Padding() { Top = 5, Bottom = 5, Left = 5, Right = 5 };
        kvp.Value.BackColor = SystemColors.ControlLightLight;
        // kvp.Value.BackColor = System.Drawing.Color.Aqua;
    }

    clientComboBox.Items.Clear();

    foreach (var u in clients)
    {
        string s = u.Value.id + ":" + u.Value.first_name + " " + u.Value.second_name + " " + u.Value.third_name;
        clientComboBox.Items.Add(s);
        clientComboBox.Text = s;

    }
}
```

Рисунок 3.29 – UpdateClients

UpdateTables метод выполняет несколько функций:

- очищает таблицы;
- вставляет в таблицы новые, полученные из запроса на сервер, данные;
- форматирует таблицы под указанный стандарт.

```
void UpdateTables()
{
    purchaseServicesDGV.Rows.Clear();

    carPartsDGV.Rows.Clear();

    serviceListDGV.Rows.Clear();

    EnableServicesEditing(true);

    Program.InsertIntoDataGridView<Tables.car_part>(JArray.Parse(Program.SelectTableRequest(Program.Table.car_part)), carPartsDGV);

    Program.InsertIntoDataGridView<Tables.purchase_service>(JArray.Parse(Program.SelectTableRequest(Program.Table.purchase_service)), purchaseServicesDGV);


    Program.InsertIntoDataGridView<Tables.service>(JArray.Parse(Program.SelectTableRequest(Program.Table.service)), serviceListDGV);

    Program.InsertIntoDataGridView<Tables.service>(JArray.Parse(Program.SelectTableRequest(Program.Table.service)), selectedServicesDGV);
    selectedServicesDGV.Rows.Clear();


    Program.FormatDataGridView(carPartsDGV);
    Program.FormatDataGridView(serviceListDGV);
    Program.FormatDataGridView(purchaseServicesDGV);
    Program.FormatDataGridView(selectedServicesDGV);



    carPartSearchBoxes = Program.CreateTextBoxesColumn(Program.GetHeaderTexts(carPartsDGV), carPartSearchPanel, 10, 10);


    purchaseServicesBoxes = Program.CreateTextBoxesColumn(Program.GetHeaderTexts(purchaseServicesDGV), purchaseServicesSearchPanel, 10, 10);



}
```

Рисунок 3.30 – UpdateTables

При нажатии на кнопку добавления услуги во внутренний список услуг и услуга копируется из таблицы каталога услуг в таблицу выбранных услуг.

```
private void button1_Click(object sender, EventArgs e)
{



    var purchase = new JObject();
    purchase["service_id"] = serviceListDGV.Rows[serviceListDGV.CurrentCell.RowIndex].Cells[0].Value.ToString();
    purchase["cost"] = serviceListDGV.Rows[serviceListDGV.CurrentCell.RowIndex].Cells[Program.GetColumnIndex("cost", serviceListDGV)].Value.ToString();

    var purchaseMeta = new JObject();
    purchaseMeta["name"] = serviceListDGV.Rows[serviceListDGV.CurrentCell.RowIndex].Cells[Program.GetColumnIndex("name", serviceListDGV)].Value.ToString();
    purchaseMeta["cost"] = serviceListDGV.Rows[serviceListDGV.CurrentCell.RowIndex].Cells[Program.GetColumnIndex("cost", serviceListDGV)].Value.ToString();

    purchases["post"].Value<JArray>().Add(purchase);
    purchases["meta"].Value<JArray>().Add(purchaseMeta);

    List<string> copy = new List<string>();
    for (int i = 0; i < serviceListDGV.ColumnCount; i++)
    {
        copy.Add(serviceListDGV.Rows[serviceListDGV.CurrentCell.RowIndex].Cells[i].Value.ToString());
    }
    selectedServicesDGV.Rows.Add(copy.ToArray());
}
```

Рисунок 3.31 – Копирование из одной таблицы в другую

При нажатии на кнопку оформления заказа посылается запрос на сервер, в котором в таблицу заказы помещается новый заказ, а наименования каждого отдельного товара/услуги помещаются во вложенную таблицу.

```
private void button3_Click(object sender, EventArgs e)
{
    //Сделать обработку ошибки при пустом заказе

    JObject j = new JObject();
    j["client_id"] = clientComboBox.SelectedItem.ToString().Split(':')[0];
    j["manager_id"] = Login.clientID.ToString();
    j["purchase_service_list"] = purchases["post"].Value<JArray>();
    j["table"] = Program.Table.purchase_service.ToString();
    //j["totalcost"]

    string s = j.ToString();

    Program.InsertIntoTableRequest(Program.Table.purchase_service, s);
    EnableServicesEditing(false);

}
```

Рисунок 3.32 – Отправление запроса на сервер

EnableServiceEditing метод отключает или включает возможность редактировать таблицу услуг.

```
void EnableServicesEditing(bool active)
{
    selectedServicesDGV.ReadOnly = !active;
    selectedServicesDGV.AllowUserToDeleteRows = active;
    purchaseServicesBtn.Enabled = active;
    addServiceBtn.Enabled = active;
    printServicesBtn.Enabled = !active;
}
```

Рисунок 3.33 – EnableServiceEditing

При нажатии на кнопку печати заказа создается pdf документ заказа и открывается.

```
ссылка: 1
private void button4_Click(object sender, EventArgs e)
{
    OpenDoc();
    System.Diagnostics.Process.Start(@"test.pdf");
}
```

Рисунок 3.34 – Работа кнопки

При нажатии на кнопку очистки очищается таблица выбранных заказов и возвращается возможность редактирования услуг.

```
private void button2_Click(object sender, EventArgs e)
{
    selectedServicesDGV.Rows.Clear();

    foreach (var j in purchases.Children<JArray>())
    {
        j.Clear();
    }
    EnableServicesEditing(true);
}
```

Рисунок 3.35 – Работа кнопки

Метод OpenDoc создает документ pdf заказа и заносит в него данные из выбранных заказов, исользуя метаданные.

```
void OpenDoc()
{
    FileInfo file = new FileInfo(@"test.pdf");
    file.Directory.Create();
    try
    {
        PdfDocument pdfDoc = new PdfDocument(new PdfWriter(@"test.pdf"));
        Document doc = new Document(pdfDoc);



        float width = (pdfDoc.GetNumberOfPages() > 0) ? pdfDoc.GetFirstPage().GetPageSize().GetWidth() : pdfDoc.GetDefaultPageSize().GetWidth();


        Thread.CurrentThread.CurrentCulture = CultureInfo.GetCultureInfo("en-GB");
        PdfFont font = PdfFontFactory.CreateFont(@"title.otf", "Cp1251", true);


        align


        title.SetFont(font);
        title.SetFontSize(20);

        title.SetHorizontalAlignment(iText.Layout.Properties.HorizontalAlignment.CENTER);
        doc.Add(title);


        Paragraph date = new Paragraph(DateTime.Now.ToLongDateString());
        date.SetFont(font);
        date.SetFontSize(12);
        date.SetHorizontalAlignment(iText.Layout.Properties.HorizontalAlignment.CENTER);
        doc.Add(date);

        //foreach two tables
        int allCost = 0;

        Table table = new Table(UnitValue.CreatePercentArray(1)).UseAllAvailableWidth();


        Table purchaseTable = new Table(UnitValue.CreatePercentArray(2)).UseAllAvailableWidth();
        var p1 = new Paragraph("Название");
        var p2 = new Paragraph("Стоимость");
        p1.SetFont(font);
        p2.SetFont(font);
        purchaseTable.AddHeaderCell(p1);
        purchaseTable.AddHeaderCell(p2);

        int fullCost = 0;

        for (int k = 0; k < purchases["meta"].Value<JArray>().Count; k++)
        {
            Paragraph name = new Paragraph(purchases["meta"].Value<JArray>()[k]["name"].ToString());
            name.SetFont(font);
            Paragraph cost = new Paragraph(purchases["meta"].Value<JArray>()[k]["cost"].ToString());
            cost.SetFont(font);

            purchaseTable.AddCell(name);
            purchaseTable.AddCell(cost);

            fullCost += purchases["meta"].Value<JArray>()[k]["cost"].Value<int>();
        }

        table.AddCell(purchaseTable);

        doc.Add(table);

        Table fullCostTable = new Table(UnitValue.CreatePercentArray(2)).UseAllAvailableWidth();

        var fullCostP = new Paragraph("Полная стоимость");

        fullCostP.SetFont(font);

        fullCostTable.AddCell(fullCostP);
        fullCostTable.AddCell(fullCost.ToString());
        doc.Add(fullCostTable);

        var d = new Paragraph();

        List<TabStop> tabStops = new List<TabStop>();

        // Create a TabStop at the middle of the page
        tabStops.Add(new TabStop(width / 2, iText.Layout.Properties.TabAlignment.CENTER, line));

        var clientSignature = new Paragraph();
        var managerSignature = new Paragraph();

        clientSignature.SetFont(font);
        managerSignature.SetFont(font);

        clientSignature.Add("Подпись клиента").Add(new Tab());
        managerSignature.Add("Подпись менеджера").Add(new Tab());

        clientSignature.AddTabStops(tabStops);
        managerSignature.AddTabStops(tabStops);

        d.Add(clientSignature);
        d.Add(managerSignature);
        doc.ShowTextAligned(d, pdfDoc.GetFirstPage().GetPageSize().GetWidth() / 2, pdfDoc.GetFirstPage().GetPageSize().GetBottom() + 20, 1, TextAlignment.CENTER, VerticalAlignment.BOTTOM, 0);
        //doc.Add(d);




        doc.Close();
    }
    catch (System.IO.IOException)
    {
        MessageBox.Show("Пожалуйста, закройте приложение, использующее файл test.pdf");
    }
```

Рисунок 3.36 – Метод OpenDoc

AddTablesToParagraph добавляет линии справа и слева от параграфа документа.

```
private static Paragraph AddTabsToParagraph(iText.Kernel.Pdf.Canvas.Draw.ILineDrawer line, float width, string text)
{
    List<TabStop> tabStops = new List<TabStop>();

    // Create a TabStop at the middle of the page
    tabStops.Add(new TabStop(width / 2, iText.Layout.Properties.TabAlignment.CENTER, line));

    // Create a TabStop at the end of the page
    tabStops.Add(new TabStop(width, iText.Layout.Properties.TabAlignment.LEFT, line));

    Paragraph p = new Paragraph().AddTabStops(tabStops);
    p
        .Add(new Tab())
        .Add(text)
        .Add(new Tab());
    return p;
}
```

Рисунок 3.37 – AddTablesToParagraph

MyLine класс был создан для возможности создания линий для параграфов.

```
private class MyLine : ILineDrawer
{
    private float lineWidth = 1;
    private float offset = 2.02f;
    private iText.Kernel.Colors.Color color = ColorConstants.BLACK;

    ссылка: 1
    public void Draw(PdfCanvas canvas, iText.Kernel.Geom.Rectangle drawArea)
    {
        float coordY = drawArea.GetY() + lineWidth / 2 + offset;
        canvas
            .SaveState()
            .SetStrokeColor(color)
            .SetLineWidth(lineWidth)
            .MoveTo(drawArea.GetX(), coordY)
            .LineTo(drawArea.GetX() + drawArea.GetWidth(), coordY)
            .Stroke()
            .RestoreState();
    }

    ссылка: 1
    public float GetLineWidth()
    {
        return lineWidth;
    }

    ссылка: 1
    public void SetLineWidth(float lineWidth)
    {
        this.lineWidth = lineWidth;
    }

    ссылка: 1
    public iText.Kernel.Colors.Color GetColor()
    {
        return color;
    }

    ссылка: 1
    public void SetColor(iText.Kernel.Colors.Color color)
    {
        this.color = color;
    }

    ссылок: 0
    public float GetOffset()
    {
        return offset;
    }

    ссылок: 0
    public void SetOffset(float offset)
    {
        this.offset = offset;
    }
}
```

Рисунок 3.38 – MyLine

При нажатии на кнопку изменения запчастей срабатывает метод:

```
ссылка: 1
private void changeCarPartBtn_Click(object sender, EventArgs e)
{
    EnableCarPartsChangins(true);
}
```

Рисунок 3.39 – Работа кнопки

При нажатии на кнопку сохранения изменений запчастей срабатывает метод:

```
private void saveChangingsCarPartsBtn_Click(object sender, EventArgs e)
{
    EnableCarPartsChangins(false);

    JArray carParts = new JArray();
    //
    var headers = new List<string>();
    for (int i = 0; i < carPartsDGV.ColumnCount; i++)
    {
        headers.Add(carPartsDGV.Columns[i].HeaderText);
    }
    for (int i = 0; i < carPartsDGV.RowCount; i++)
    {
        JObject carPart = new JObject();
        bool rowIsEmpty = true;
        for (int j = 0; j < headers.Count; j++)
        {
            var val = carPartsDGV.Rows[i].Cells[j].Value;
            if (val != null)
            {
                carPart[headers[j]] = val.ToString();
                rowIsEmpty = false;
            }
            else
            {
                carPart[headers[j]] = null;
            }
        }
        if (!rowIsEmpty)
            carParts.Add(carPart);
    }
    Program.UpdateTableRequest(Program.Table.car_part, carParts.ToString());

    UpdateTables();
}
```

Рисунок 3.40 – Работа кнопки

При этом посылается запрос на обновление таблицы с данными измененной таблицы.

Метод EnableCarPartsChangins позволяет включать и отключать возможность редактирования запчастей.

```
void EnableCarPartsChangins(bool active)
{
    saveChangingsCarPartsBtn.Enabled = active;
    changeCarPartBtn.Enabled = !active;
    carPartsDGV.ReadOnly = !active;

}
```

Рисунок 3.41 – Метод EnableCarPartsChangins

При нажатии на кнопку изменения клиента срабатывает метод:

```
private void button8_Click(object sender, EventArgs e)
{
    SetActiveTextBoxes(true);
    button8.Enabled = false;

    button10.Enabled = false;

    button9.Enabled = true;

    clientComboBox.Enabled = false;

    button1.Enabled = true;
}
```

Рисунок 3.42 – Работа кнопки

При нажатии на кнопку сохранения изменения пользователя срабатывает метод:

```
private void button9_Click(object sender, EventArgs e)
{
    SetActiveTextBoxes(false);

    button9.Enabled = false;

    button10.Enabled = true;

    button8.Enabled = true;

    button1.Enabled = false;

    Clients.Client client = new Clients.Client();
    client.user_type_id = "3";
    client.id = clientComboBox.Text.Split(':')[0];
    client.first_name = firstNameBox.Text;
    client.second_name = secondNameBox.Text;
    client.third_name = thirdNameBox.Text;
    client.IIN = iinBox.Text;
    client.car_model = carModelBox.Text;
    client.car_numbers = carNumbersBox.Text;
    client.login = loginBox.Text;
    client.password = passwordBox.Text;
    client.phone_number = phoneNumBox.Text;
    //update user request
    Program.UpdateTableRequest(Program.Table.user, JsonConvert.SerializeObject(client));
    //update users combobox request
    UpdateClients();
    clientComboBox.Enabled = true;
}
```

Рисунок 3.43 – Работа кнопки

При этом на сервер посылается запрос на обновление данных и обновляются клиенты.

Метод SetActiveTextBoxes позволяет активировать и деактивировать возможность изменения полей пользователя.

```
ссылок: 6
void SetActiveTextBoxes(bool active)
{
    firstNameBox.Enabled = active;
    secondNameBox.Enabled = active;
    thirdNameBox.Enabled = active;
    passwordBox.Enabled = active;
    loginBox.Enabled = active;
    iinBox.Enabled = active;
    carModelBox.Enabled = active;
    carNumbersBox.Enabled = active;
    phoneNumBox.Enabled = active;
}
//new user
```

Рисунок 3.44 – Метод SetActiveTextBoxes

При нажатии на кнопку нового пользователя срабатывает метод:

```
private void button10_Click(object sender, EventArgs e)
{
    SetActiveTextBoxes(true);
    ClearUserDataTextBoxes();
    button10.Enabled = false;
    button1.Enabled = true;
    button8.Enabled = false;
    button9.Enabled = false;
    button6.Enabled = true;

    clientComboBox.Enabled = false;
}
```

Рисунок 3.45 – Работа кнопки

При нажатии на кнопку регистрации клиента срабатывает метод:

```
private void button6_Click(object sender, EventArgs e)
{
    SetActiveTextBoxes(false);

    button6.Enabled = false;
    button10.Enabled = true;
    button8.Enabled = true;
    button1.Enabled = false;
    Clients.Client client = new Clients.Client();
    client.user_type_id = "3";
    client.first_name = firstNameBox.Text;
    client.second_name = secondNameBox.Text;
    client.third_name = thirdNameBox.Text;
    client.IIN = iinBox.Text;
    client.login = loginBox.Text;
    client.password = passwordBox.Text;
    client.car_model = carModelBox.Text;
    client.car_numbers = carNumbersBox.Text;
    client.phone_number = phoneNumBox.Text;
    //insert user request
    Program.InsertIntoTableRequest(Program.Table.user, JsonConvert.SerializeObject(client).ToString());
    //update users combobox request
    UpdateClients();
    clientComboBox.Enabled = true;
}
```

Рисунок 3.46 – Работа кнопки

При этом на сервер посылается запрос на добавление новой строки в таблицу пользователи.

При нажатии на кнопку очистить срабатывает метод:

```
private void button7_Click(object sender, EventArgs e)
{
    ClearUserDataTextBoxes();
}
void ClearUserDataTextBoxes()
{
    firstNameBox.Clear();
    secondNameBox.Clear();
    thirdNameBox.Clear();
    carModelBox.Clear();
    iinBox.Clear();
}
```

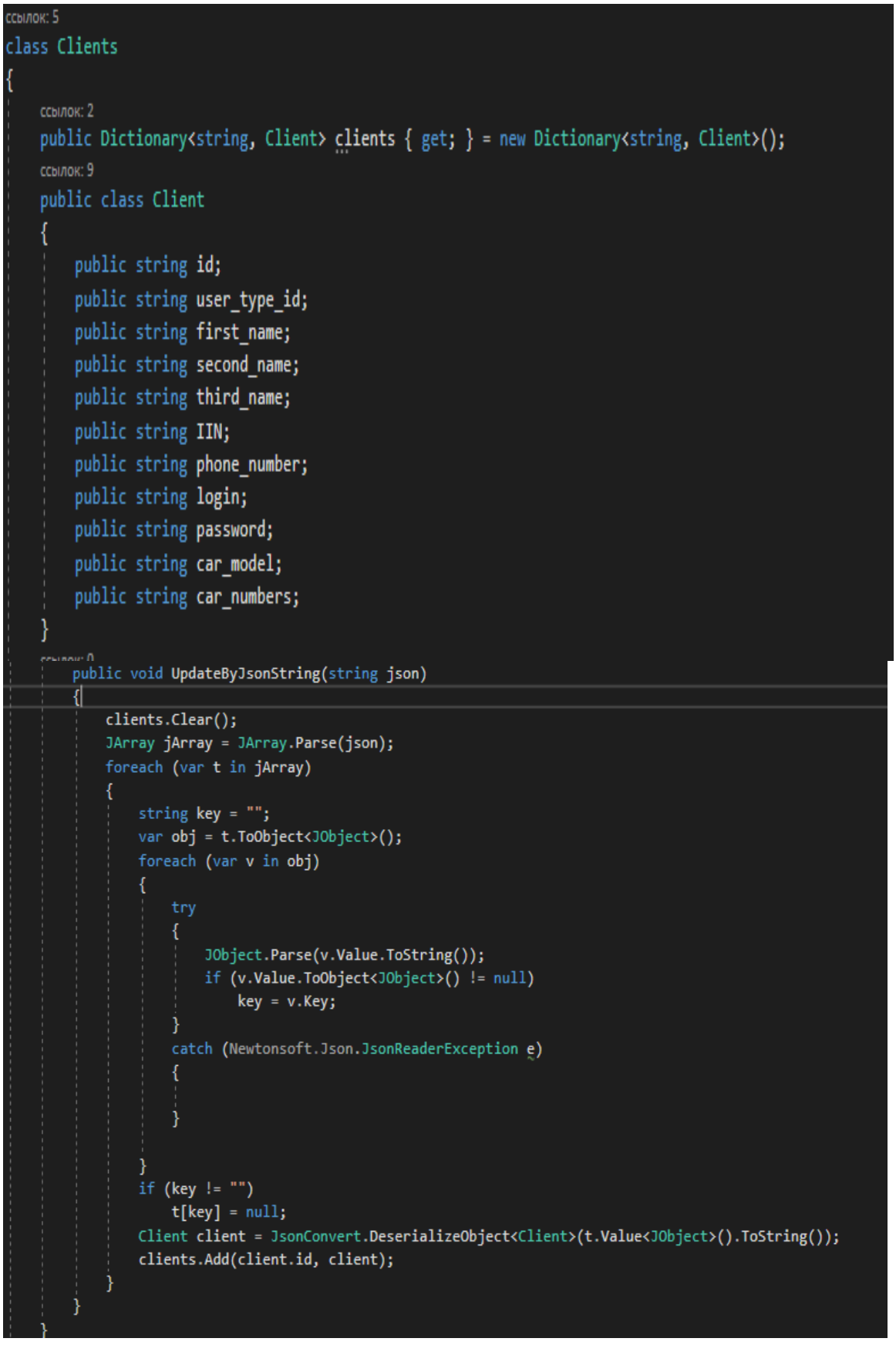
Рисунок 3.47 – Работа кнопки

При нажатии на кнопку отмены срабатывает метод:

```
ссылка: 1
private void button1_Click_1(object sender, EventArgs e)
{
    SetActiveTextBoxes(false);
    button10.Enabled = true;
    button8.Enabled = true;
    button6.Enabled = false;
    button9.Enabled = false;
    clientComboBox.Enabled = true;
}
```

Рисунок 3.48 – Работа кнопки

Для упрощения работы с данными клиентов был создан класс:



```
ссылок: 5
class Clients
{
    ссылок: 2
    public Dictionary<string, Client> clients { get; } = new Dictionary<string, Client>();
    ссылок: 9
    public class Client
    {
        public string id;
        public string user_type_id;
        public string first_name;
        public string second_name;
        public string third_name;
        public string IIN;
        public string phone_number;
        public string login;
        public string password;
        public string car_model;
        public string car_numbers;
    }
    ссылок: 0
    public void UpdateByJsonString(string json)
    {
        clients.Clear();
        JArray jArray = JArray.Parse(json);
        foreach (var t in jArray)
        {
            string key = "";
            var obj = t.ToObject<JObject>();
            foreach (var v in obj)
            {
                try
                {
                    JObject.Parse(v.Value.ToString());
                    if (v.Value.ToObject<JObject>() != null)
                        key = v.Key;
                }
                catch (Newtonsoft.Json.JsonReaderException e)
                {

                }

            }
            if (key != "")
                t[key] = null;
            Client client = JsonConvert.DeserializeObject<Client>(t.Value<JObject>().ToString());
            clients.Add(client.id, client);
        }
    }
}
```

Рисунок 3.49 – Класс Clients

От него пришлось частично отказаться в пользу лучшей структурированности.

При нажатии на кнопку удаления запчасти срабатывает метод:

```
private void deleteCarPartBtn_Click(object sender, EventArgs e)
{
    Int32 selectedRowCount = carPartsDGV.Rows.GetRowCount(DataGridViewElementStates.Selected);
    List<int> ids = new List<int>();
    if (selectedRowCount > 0)
    {

        for (int i = 0; i < selectedRowCount; i++)
        {
            int index = carPartsDGV.SelectedRows[i].Index;

            string val = carPartsDGV.Rows[index].Cells[Program.GetColumnIndex("id", carPartsDGV)].Value.ToString();
            int id = 0;
            if (int.TryParse(val, out id))
            {
                ids.Add(id);
            }
        }
    }
    JArray data = new JArray();
    foreach (int i in ids)
    {
        JObject jObject = new JObject();
        jObject["id"] = i;
        data.Add(jObject);
    }
    Program.DeleteFromTableRequest(Program.Table.car_part, data.ToString());
}
```

Рисунок 3.50 – Работа кнопки

При этом посылается запрос на сервер на удаление столбца с данным id.

При нажатии на кнопку поиска по запчастям срабатывает метод:

```
private void carPartsSearchBtn_Click(object sender, EventArgs e)
{
    carPartsDGV.Rows.Clear();
    Dictionary<string, string> query = new Dictionary<string, string>();
    foreach (var kvp in carPartSearchBoxes)
        if (!string.IsNullOrWhiteSpace(kvp.Value.Text))
            query.Add(kvp.Key, kvp.Value.Text);
    Program.InsertIntoDataGridView<Tables.service>(JArray.Parse(Program.SelectTableRequest(Program.Table.car_part, query)), carParts
}
```

Рисунок 3.51 – Работа кнопки

При этом:

- таблица запчастей очищается.
- берутся данные для запроса из полей ввода для поиска.
- посылается запрос на выборку из таблицы запчастей с данными условиями.
- данные вставляются в таблицу.

При изменении выбранного пользователя из выпадающего списка пользователей срабатывает метод:

```
ссылка: 1
private void clientComboBox_SelectedIndexChanged(object sender, EventArgs e)
{
    var id = clientComboBox.Text.Split(':')[0];

    firstNameBox.Text = clients[id].first_name;

    secondNameBox.Text = clients[id].second_name;

    thirdNameBox.Text = clients[id].third_name;

    iinBox.Text = clients[id].IIN;

    carModelBox.Text = clients[id].car_model;

    carNumbersBox.Text = clients[id].car_numbers;

    loginBox.Text = clients[id].login;

    passwordBox.Text = clients[id].password;

    phoneNumBox.Text = clients[id].phone_number;

    JObject client = JObject.Parse(JsonConvert.SerializeObject(clients[id]));
    foreach (var kvp in client)
    {
        clientTextBoxes[kvp.Key].Text = kvp.Value.ToString();
    }
}
```

Рисунок 3.52 – Вызов метода

При нажатии на кнопку поиска по каталогу услуг срабатывает метод:

```
private void purchaseServicesSearchBtn_Click(object sender, EventArgs e)
{
    purchaseServicesDGV.Rows.Clear();
    Dictionary<string, string> query = new Dictionary<string, string>();
    foreach (var kvp in purchaseServicesBoxes)
        if (!string.IsNullOrWhiteSpace(kvp.Value.Text))
            query.Add(kvp.Key, kvp.Value.Text);
    Program.InsertIntoDataGridView<Tables.service>(JArray.Parse(Program.SelectTableRequest(Program.Table.purchase_service, query)),
}
```

Рисунок 3.53 – Работа кнопки

При этом:

- таблица услуг очищается.
- берутся данные для запроса из полей ввода для поиска.
- посылается запрос на выборку из таблицы запчастей с данными условиями.
- данные вставляются в таблицу.

При нажатии на кнопку открытия чата срабатывает метод:

```
private void button2_Click_1(object sender, EventArgs e)
{
    Messages messages = new Messages();
    messages.other_id = Int32.Parse(clientComboBox.Text.Split(':')[0].ToString());
    messages.Show();
}
```

Рисунок 3.54 – Работа кнопки

Класс Messages был создан для обмена сообщениями между пользователем и менеджером.

Messages имеет поле:

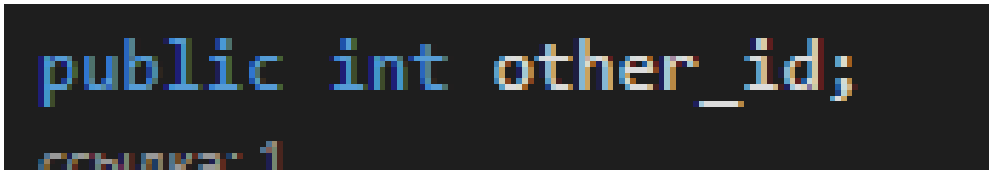


Рисунок 3.55 – Messages

Поле other_id это id выбранного собеседника.

При нажатии на кнопку отправки сообщения срабатывает метод:

```
private void messageButton_Click(object sender, EventArgs e)//sendingMessage
{
    if (textBox1.Text != "")
    {
        M_Message m_Message = new M_Message();

        m_Message.receiver_id = other_id.ToString();
        m_Message.sender_id = Login.clientID.ToString();
        m_Message.text = textBox1.Text;
        m_Message.datetime = DateTime.Now;//.ToString("yyyy-MM-dd HH:mm:ss");
        JObject j = JObject.FromObject(m_Message);
        j["datetime"] = j["datetime"].Value<DateTime>().ToString("yyyy-MM-dd HH:mm:ss");
        Program.InsertIntoTableRequest(Program.Table.message, j.ToString());


        GetMessages(new object());
    }
```

Рисунок 3.56 – Работа кнопки

При этом на сервер посылается запрос на добавление в таблицу messages новой строки.

Для получения сообщений активируется метод GetMessages:

```
public void GetMessages(object obj)
{



    var senderMessages = Program.SelectTableRequest(Program.Table.message, new Dictionary<string, string>() { { "sender_id", other_
    var receiverMessages = Program.SelectTableRequest(Program.Table.message, new Dictionary<string, string>() { { "receiver_id", ot
    //lol = lol.Replace("\"", "'");

    if (!string.IsNullOrWhiteSpace(senderMessages) || !string.IsNullOrWhiteSpace(receiverMessages))
    {
        JArray sMessages = new JArray();
        if (!string.IsNullOrWhiteSpace(senderMessages))
            sMessages = JArray.Parse(senderMessages);//getting arrays of messages

        JArray rMessages = new JArray();
        if (!string.IsNullOrWhiteSpace(receiverMessages))
            rMessages = JArray.Parse(receiverMessages);

        var list = new List<M_Message>();
        foreach (var mes in sMessages)
        {
            list.Add(mes.ToObject<M_Message>());
        }
        foreach (var mes in rMessages)
        {
            list.Add(mes.ToObject<M_Message>());
        }
    if (list.Count > 0)
    {
        var orderedList = list.OrderBy(x => x.datetime).ToList();//sorting by datetime



        messagePanel.Controls.Clear();

        for (int i = 0; i < orderedList.Count; i++)//adding labels to messagePanel
        {
            Label lbl = new Label();
            lbl.Text = " " + orderedList[i].text + " ";
            if (int.Parse(orderedList[i].sender_id) == other_id)
            {

                lbl.Width = lbl.PreferredWidth + 2;

                lbl.Height = lbl.PreferredHeight + 10;
                lbl.TextAlign = ContentAlignment.MiddleRight;




                lbl.BackColor = Color.LightGray;
                lbl.Location = new Point(messagePanel.Width - lbl.Width - 10, messagePanel.Controls.Count * (lbl.PreferredHeight

            }
            else
            {

                lbl.Width = lbl.PreferredWidth + 2;

                lbl.Height = lbl.PreferredHeight + 10;
                lbl.TextAlign = ContentAlignment.MiddleLeft;
                lbl.BackColor = Color.LightBlue;
                lbl.Location = new Point(10, messagePanel.Controls.Count * (lbl.PreferredHeight + 15) + 5);


            }

            messagePanel.Controls.Add(lbl);
        }
```

Рисунок 3.57 –Активация метода GetMessages

При этом посылается запрос на сервер на получение данных из таблицы messages, а полученные данные сортируются по отправителю и получателю и по дате отправления. Затем эти данные визуально выводятся в панель сообщений.

Класс M_Message был создан для упрощения работы с сообщениями.

```
private class M_Message//message class for json Objects
{
    public string id;
    public string sender_id;
    public string receiver_id;
    public string text;
    public DateTime datetime;
}
```

Рисунок 3.58 – Класс M_Message

Класс CreateRoundRectRgn был создан для декорации. Он позволяет округлять края элементов.

```
[DllImport("Gdi32.dll", EntryPoint = "CreateRoundRectRgn")]//for rounded corners
ссылок: 0
private static extern IntPtr CreateRoundRectRgn
(
    int nLeftRect,     // x-coordinate of upper-left corner
    int nTopRect,      // y-coordinate of upper-left corner
    int nRightRect,    // x-coordinate of lower-right corner
    int nBottomRect,   // y-coordinate of lower-right corner
    int nWidthEllipse, // width of ellipse
    int nHeightEllipse // height of ellipse
);
```

Рисунок 3.59 – Класс CreateRoundRectRgn

3.2 Системные требования

Рекомендуемые системные требования:
- операционная система: Windows 7/8/10 - 64-bit;
- процессор (CPU): Intel Celeron (Desktop);Л
- оперативная память (RAM): 1 ГБ (или больше);
- свободное место на жёстком диске: ~300 MB.

Рекомендуемые системные требования:
- операционная система: Windows 7/8/10 - 64-bit;
- процессор (CPU): Intel Core i3 (Desktop);
- оперативная память (RAM): 2 ГБ (или больше);
- свободное место на жёстком диске: ~300 MB.

3.3 Интерфейс

Для начала работы с приложением, необходимо авторизоваться.

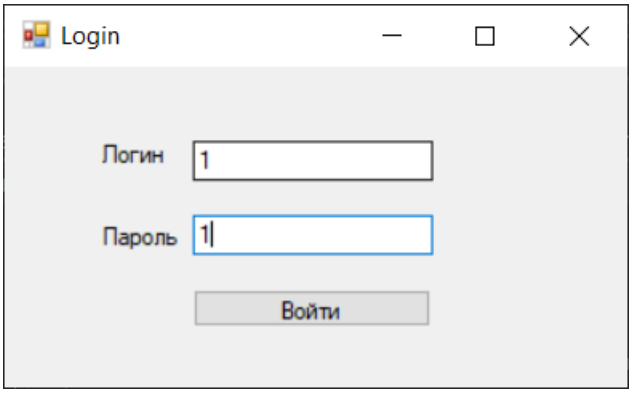


Рисунок 3.60 – Форма авторизации

После авторизации откроется главная форма, в которой есть возможность начать работу с приложением. На данном рисунке можно увидеть архив заказов

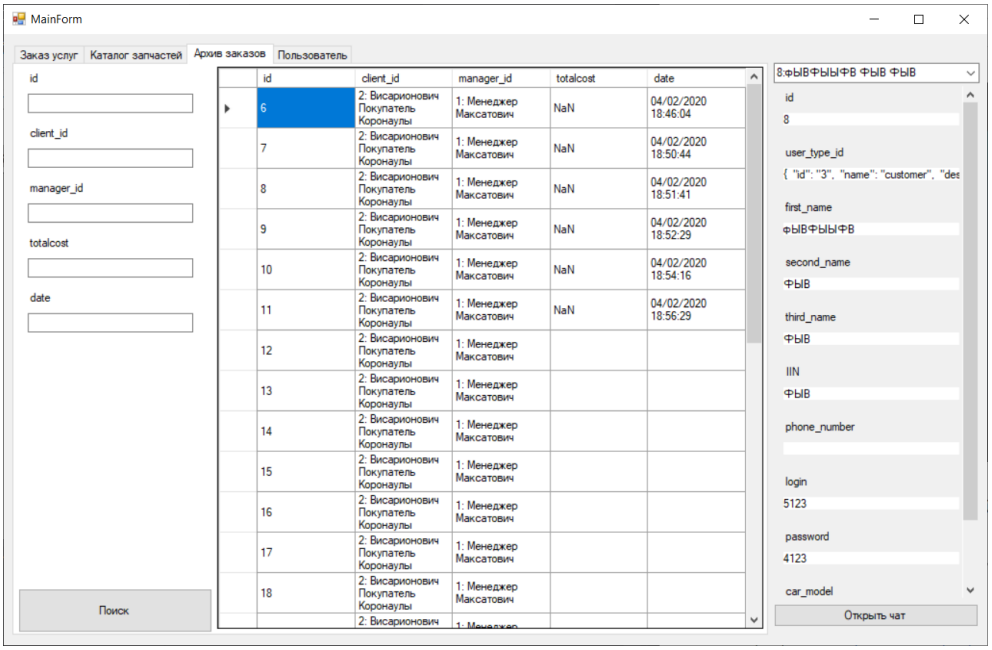


Рисунок 3.61 – Главная форма

Клиент имеет возможность заказать услуги у предприятия через приложение. Через кнопку добавить, есть возможность выбрать услуги.

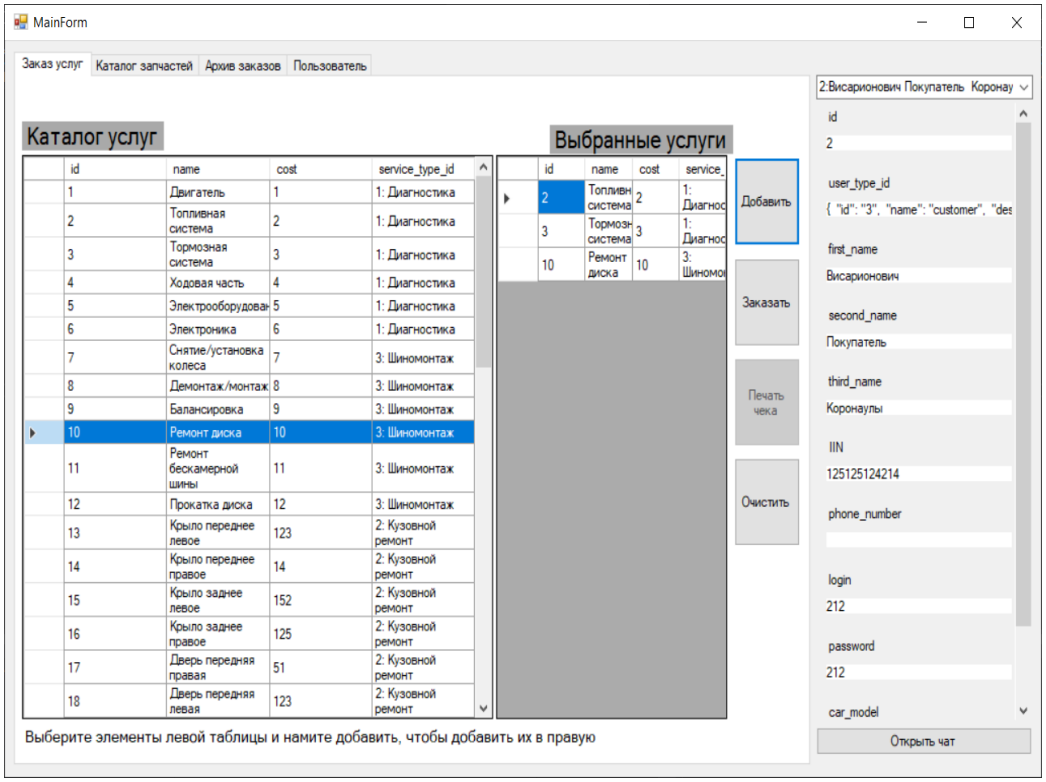


Рисунок 3.62 – Выбор услуг

Безусловно, у клиента есть возможность удаления услуг, которые у него нету необходимости.

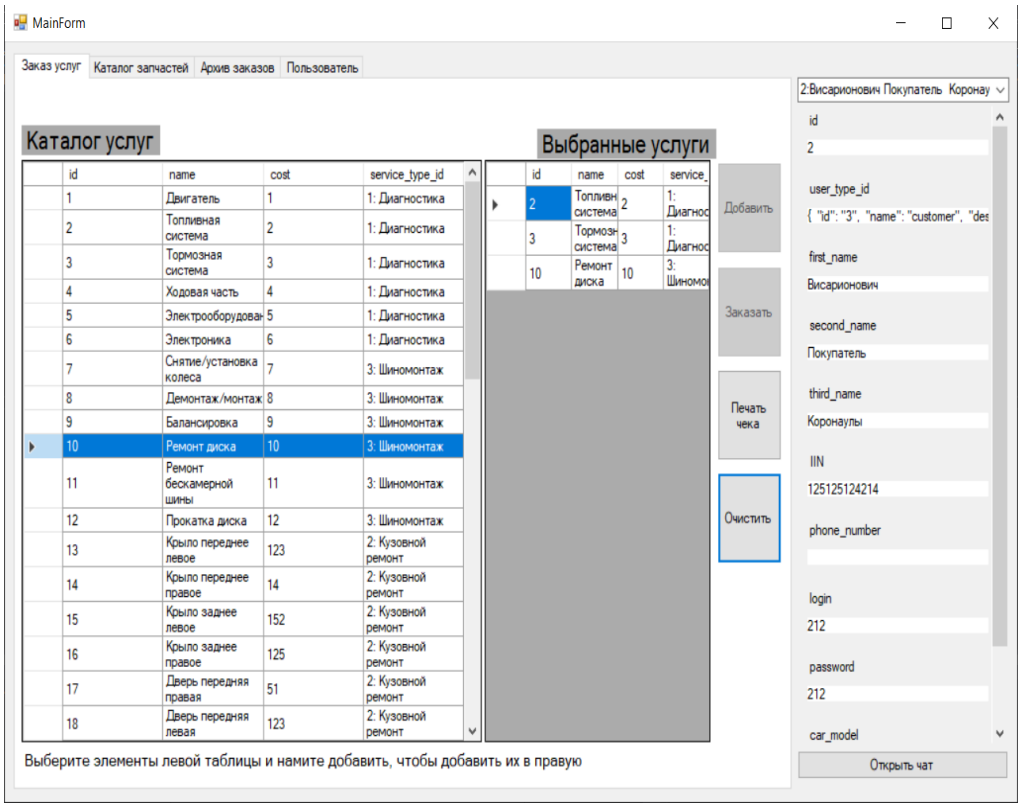


Рисунок 3.63 – Удаление услуг

После выбора услуг, есть возможность напечатать себе чек в pdf формате.

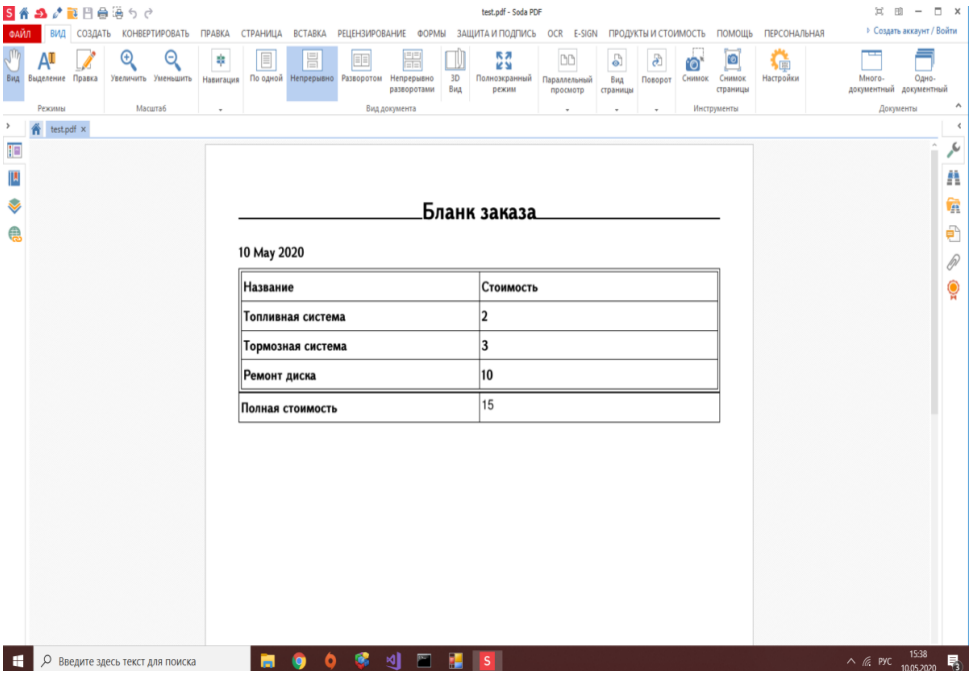


Рисунок 3.64 – Электронный чек

Есть возможность пополнения каталога.

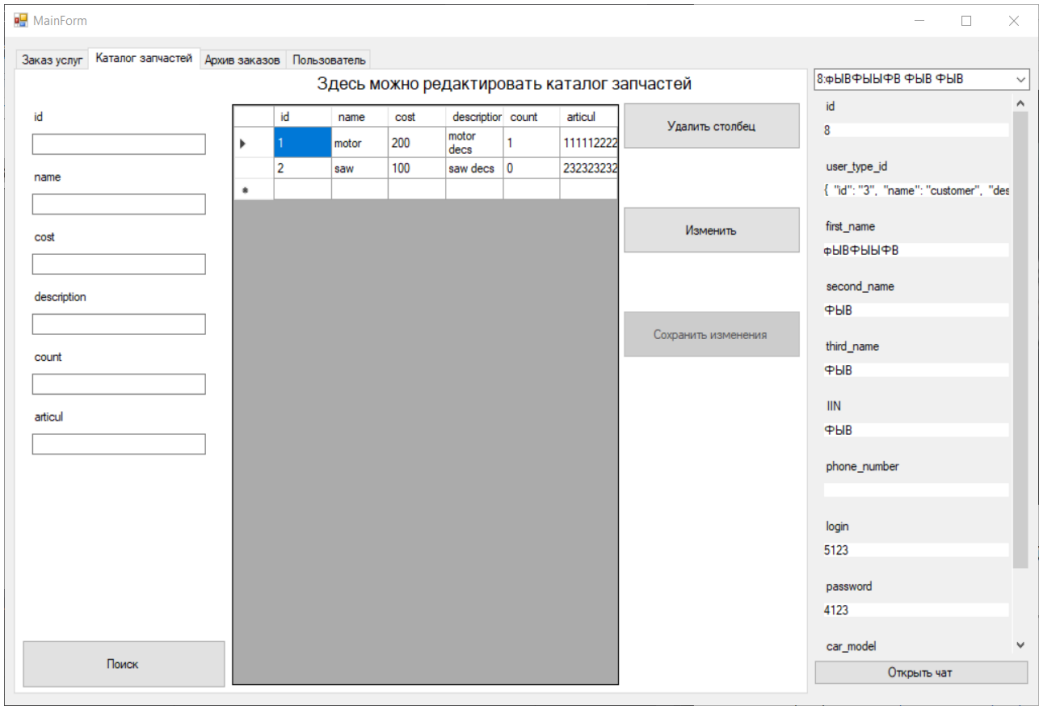


Рисунок 3.65 – Каталог запчастей

Также без проблем можно добавить нового пользователя

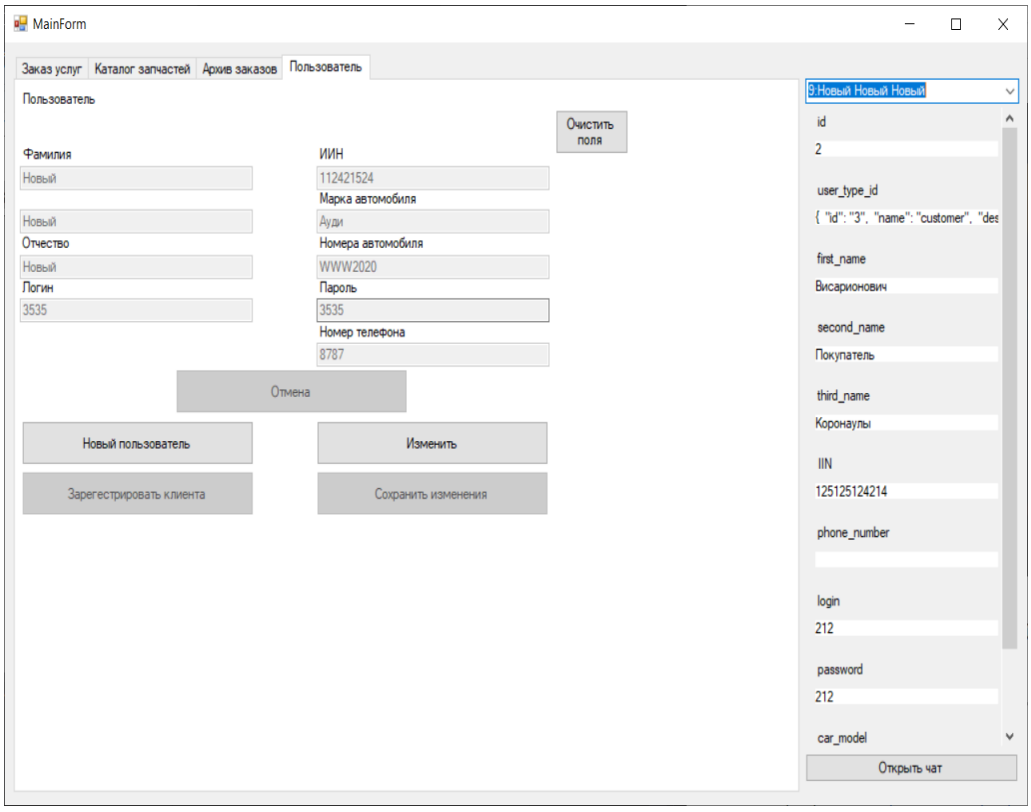


Рисунок 3.66 – Добавление нового пользователя

Приложение поддерживает чат между клиентом и менеджером.

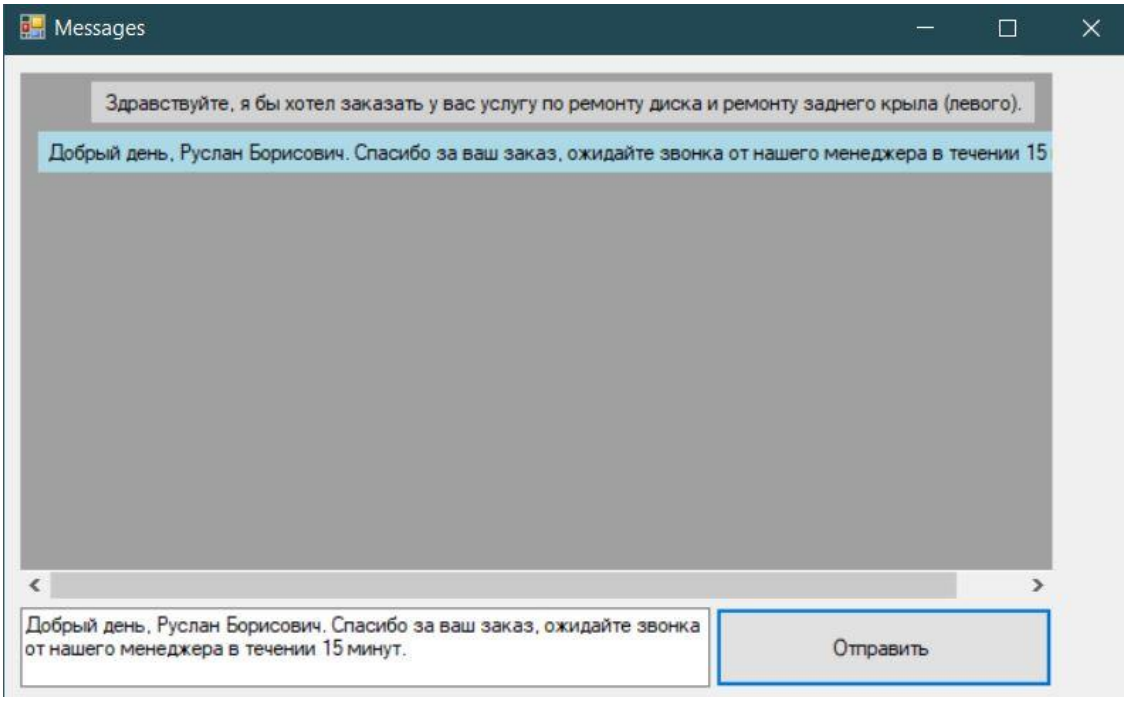


Рисунок 3.67 – Чат

Исходя из программной реализации информационной системы, можно сделать вывод, что важным компонентом для управления и защиты базы данных является сервер. Сервер является неотъемлемой частью данной информационной системы, так как он служит для осуществления взаимодействия менеджера с клиентами автосервиса, используя встроенный чат. Возможность управления данными о клиенте, заказах в приложении менеджера, ускоряет его работу при решении различных проблем. Также, функция автоматической генерации электронного чека значительно упрощает процесс оформления заказов менеджером.

Таким образом, разработанная информационная система улучшает эффективность менеджеров при работе с клиентами автосервиса.

4 Обоснование эффективности внедрения проекта

4.1 Технико-экономическое обоснование дипломных работ, связанных с разработкой программного продукта (ПП)

Технико-экономическое обоснование разработки должно содержать:
- определение трудоемкости разработки ПП;
- расчет затрат на разработку ПП;
- определение возможной цены разработанного ПП;
- оценку социально - экономических результатов функционирования ПП.

4.2 Трудоемкость разработки ПП

Трудоемкость – это экономический показатель, характеризующий показатель затраты рабочего времени на производство единицы продукции или на выполнение конкретной технологической операции. Создание ПП занимает примерно 112 часов для одной продукции.

Таблица 4.1 - Распределение работ по этапам и видам и оценка их трудоемкости

Этап разработки ПП	Вид работы на данном этапе	Трудоемкость разработки ПП, чел.× ч.
Техническое задание	Перед началом разработки нужно получить техническое задание и составить план по разработке: интерфейс, функции и прочее.	12
Разработка	Данный этап подразумевает создание самого ПП. Здесь происходит реализация всех идей из этапа планирования.	60
Тестирование	После создания ПП, нужно обязательно его протестировать для выявления ошибок в работе или его улучшения.	12
ИТОГО трудоемкость выполнения дипломной работы		84

4.3 Расчет затрат на разработку ПП

Определение затрат на разработку ПП производится путем составления соответствующей сметы, которая включает следующие статьи:

1) Материальные затраты.
2) Затраты на оплату труда.
3) Социальный налог.
4) Амортизация основных фондов.
5) Прочие затраты.

Когда происходит процесс создания ПП, требуется обеспечить персонал материальными ресурсами для комфортной работы.

Таблица 4.2 - Затраты на материальные ресурсы

Наименование материального ресурса	Количество израсходованного материала	Цена за единицу, тг	Сумма, тг
Visual Studio 2019	1	600000	600000
Acer Aspire	1	200000	200000
Windows 10 Pro	1	75000	75000
ИТОГО затраты на материальные ресурсы			875000

Общая сумма затрат на материальные ресурсы (ЗМ) определяется по формуле:

$$З_с = 600000 + 200000 + 75000 = 875000$$

Формула 4.1

Таблица 4.3 - Затраты на электроэнергию

Наимен. оборудов.	Паспортная мощность, кВт	Коэф. использ. мощности	Время работы оборудования для разработки ПП, ч	Цена элек/энер, тг/кВт×ч	Кол-во	Сумма, тг
Acer Aspire	0,55	0,9	84	19,17	1	1180,87
Лампочка	0,08	0,9			3	515,29
Принтер	0,31	0,9			1	665,58
ИТОГО затраты на электроэнергию						2361,74

Общая сумма затрат на электроэнергию (ЗЭ) рассчитывается по формуле:

$$З_э = (0{,}55 + 0{,}08 + 0{,}31) * 84 * 19{,}17 = 1513{,}66$$

При разработке ПП требуется сотрудники имеющие знание в данной сфере и оплата их труда.

Затраты на оплату труда рассчитываются по форме, приведенной в таблице 4.4.

Таблица 4.4 - Затраты на оплату труда

Категория работника	Количество	Квалификация	Трудоемкость разработки ПП, чел.×ч	Часовая ставка, тг/ч	Сумма, тг
Разработчик	1	Бакалавриат	84	892,86	75000
ИТОГО затраты на оплату труда					75000

Таблица 4.5 – Расходы на социальные налоги

СО(Социальные отчисления)	3,5	(ЗП - ОПВ)*3,5%	2 362,5
ВОСМСЮ (Отчисления на ВОСМСЮ)	2,0	ЗП*2%	1 500,00
СН (Социальный налог)	9,5	(ЗП-ОПВ-ВОСМС)*9,5%-СО	3 907,5
Всего уплаченные налоги			7 770,0

$$НР = ЗП \cdot \frac{Н_{нр}}{100} = 75\,000 \cdot 0{,}7 = 52\,500 \text{ тенге} \qquad (4.3)$$

Амортизационные отчисления — это денежные средства, предназначенные для возмещения износа предметов, относящихся к основным средствам предприятия

Таблица 4.6 - Амортизация основных фондов (ОФ)

Наименование оборудования и ПО	Стоимость оборудования и ПО, тг	Годовая норма амортизации, %	Время работы оборудов-я и ПО для разработки ПП, ч	Сумма, тг
Acer Aspire	200000	14,29	16	2381,67
Windows 10 Pro	75000	33,33	16	2083,125
Visual Studio	600000	33,33	16	16665
ИТОГО амортизация основных фондов				21192,795

Общая сумма амортизационных отчислений определяется по формуле:

Применение основных фондов колеблется от 3 до 10 лет. Оборудование используется в течение 7 лет, а программное обеспечение – 3 года. Применяя формулу, заполним Таблицу 1.5 для отображения амортизации основных фондов.

$$Н_{А1}= 100/7 = 14,29\%.$$

$$Н_{А3}= 100/3 = 33,33\%.$$

$$З_{ам} = \frac{200000 * 0,1429 * 16}{1 * 12 * 16} = 2381,67 \text{ тг}$$

$$З_{ам} = \frac{75000 * 0,3333 * 16}{1 * 12 * 16} = 2083,125 \text{ тг}$$

$$З_{ам} = \frac{600000 * 0,3333 * 16}{1 * 12 * 16} = 16665 \text{ тг}$$

Также стоит учитывать прочие расходы:

- аренда помещения - 75000;
- коммунальные услуги – 5492,94

Таблица 4.7 - Смета затрат на разработку ПП

Статьи затрат	Сумма, тг
1. Материальные затраты, в том числе:	
- материалы	875 000
- электроэнергия	2 361,74
2. Затраты на оплату труда.	75 000
3. Отчисления на социальные нужды.	7 770,0
4. Амортизация основных фондов.	21192,795
5. Прочие затраты.	80 492,94
ИТОГО по смете	1 061 817,475

4.4 Определение договорной цены ПП

Величина возможной (договорной) цены ПП должна устанавливаться с учетом эффективности, качества и сроков ее выполнения на уровне, отвечающем экономическим интересам заказчика (потребителя) и исполнителя.

Договорная цена (ЦД) для прикладных ПП рассчитывается по формуле:

$$Ц_д = З_{нир} * \left(1 + \frac{P}{100}\right) = 1\,061\,817,185 * \left(1 + \frac{30}{100}\right) = 1380362,72 \quad (4.4)$$

$З_{нир}$ - затраты на разработку ПП (из таблицы 4.7), тг;

Р - средний уровень рентабельности ПП. %

Далее определяется цена реализации с учетом налога на добавленную стоимость (НДС), ставка НДС устанавливается законодательно Налоговым Кодексом РК. На 2013 год ставка НДС установлена в размере 12%.

Цена реализации с учетом НДС рассчитывается по формуле:

$$Ц_р = Ц_д + Ц_д * ндс = 1380362{,}72 + 1380362{,}72 * 0{,}12 = 1546006{,}25 \quad (4.5)$$

4.5 Расчет результатов от создания и использования ИС

Информационная система для управления проектированием предназначена в главную очередь для контроля работы проектной группы, а также упрощает процесс принятия решений по управлению, способна повысить эффективность и скорость работы сотрудников, за счет уменьшения времени.

Для оценки экономии от использования разрабатываемой ИС необходимо сравнить эксплуатационные расходы с ее применением и без.

Статьи затрат при применении ИС включают в себя:

- заработная плата специалиста, осуществляющего поддержку и сопровождение системы;
- износ оборудования;
- накладные расходы.

Данная система не предполагает расхода каких-либо материалов.

В расходные материалы входит бумага, картридж.

Пачка бумаги стоит – 1500 тг.

Картридж – 23000.

В месяц уходит 5 пачек бумаги и один картридж.

$$Р = \big((1500 \cdot 5) + (23000)\big) \cdot 12 = 366\,000 \text{ тенге}$$

В распоряжении автомастерской находятся один персональный компьютер (стоимость одного ~ 400 000 тенге), принтер (120 000 тенге) и прочая компьютерная периферия (~50 000 тенге); итого затрат на оборудование – 570 000 тенге.

Износ оборудования рассчитывается исходя из 25 % амортизационных отчислений за год.

$$А = (400\,000 + 120\,000 + 50000) \cdot 0{,}25 = 142\,500 \text{ тенге}$$

Данной работой занимаются 2 сотрудника, нанятые для контроля бумажного оборота методических трудов и прочих документов.

Каждый сотрудник получает – 125 000 тенге

ЗП = 3 000 000 тенге

ОПВ = 3 000 000 ·0,1=300 000

СО = (3 000 000 – 300 000)· 0,035=94 500

ВОСМС = 3 000 000 · 0,02 = 60 000
СН = (3 000 000 - 300 000 – 60 000) · 0,095 – 94 500 = 156 300
ОТ = СО + СН + ВОСМС = 75 600 + 125 600 + 48 000 = 310 800 тенге
Общие накладные расходы составят:

$$НР = ЗП \cdot \frac{Н_{нр}}{100} = 3\,000\,000 \cdot 0{,}7 = 2\,100\,000 \text{ тенге} \quad (4.6)$$

Статьи затрат без применения ИС включают в себя:
– заработная плата сотрудников;
– износ используемого ими оборудования;
– расход материалов (к примеру, канцелярия – бумага, картридж и т.д.);
– накладные расходы.

Таблица 4.8 – Годовые эксплуатационные затраты

Статьи	Без применения ИС	С применением ИС
Годовая заработная плата	3 000 000	900 000
Социальные отчисления и налоги	310 800	93 240
Расходуемые материалы	366 000	0
Амортизационные отчисления	142 500	21192
Накладные расходы	2 100 000	630 000
Всего	5 919 300	1 644 432

Ожидаемая условно-годовая экономия определяется по формуле:

$$Э_{уг} = С_1 - С_2 + \sum Э_i \quad (4.7)$$

где $Э_{уг}$- величина экономии, тенге;
$С_1$ и $С_2$ – показатели текущих затрат по базовому и внедряемому вариантам, тенге:
$\Sigma Э_i$ – ожидаемый дополнительный эффект от различных факторов, тенге.
$Э_{уг} = С_1 - С_2$= 4 274 868 тенге

4.6 Расчет основных показателей экономической эффективности

Так как разработанная информационная система несет более социальный эффект, чем экономический, целесообразно оценивать его эффективность за счет экономии в сравнении с предыдущим периодом работы без использования ИС.

Величина ожидаемого годового экономического эффекта от внедрения ИС рассчитывается по формуле:

$$Э_г = Э_{уг} - К \cdot E_Н \quad (4.7)$$

где Эг - ожидаемый годовой экономический эффект, тенге:

Эуг — ожидаемая условно-годовая экономия, тенге;

К — капитальные вложения, тенге;

$E_Н$ - нормативный коэффициент экономической эффективности капитальных вложений.

Нормативный коэффициент экономической эффективности капитальных вложений определяется по формуле:

$$E_Н = \frac{1}{T_Н} \quad (4.7)$$

где Тн — нормативный срок окупаемости капитальных вложений, лет.

Нормативный срок окупаемости капитальных вложений. принимается исходя из срока морального старения - технических средств и проектных решений ИС (Тн=1,2,3...n), для программных продуктов срок окупаемости принимаем равным 4 года. 1546006

$$E_Н = \frac{1}{4} = 0{,}25$$

$$Э_г = 4\,274\,868 - 1\,546\,006 \cdot 0{,}25 = 3\,888\,366 \text{ тенге}$$

Расчетный коэффициент экономической эффективности капитальных вложений составляет:

$$E_р = \frac{Э_{уг}}{К}$$

где Ер - расчетный коэффициент экономической эффективности капитальных вложений;

Эуг — ожидаемая условно-годовая экономия, тенге;

К — капитальные вложения на создание системы, тенге.

$$E_р = \frac{4\,274\,868}{1\,546\,006} = 2{,}77$$

Расчетный срок окупаемости капитальных вложений составляет:

$$T_P = \frac{1}{E_P} \quad (4.8)$$

где Ер - коэффициент экономической эффективности капитальных вложений.

$$T_P = \frac{1}{2{,}77} = 0{,}4 \text{ года} \approx 4{,}8 \text{ месяцев}$$

Таблица 4.9 – Показатели сравнительной экономической эффективности от внедрения программного продукта

Наименование показателей	Значение
Условная годовая экономия затрат, тенге	4 274 868
Коэффициент экономической эффективности капитальных вложений (Ер)	2,77
Срок окупаемости капитальных вложений (Тр)	0,4

Проведенные расчеты доказали эффективность внедрения информационной системы как с экономической, так и с технической стороны. В ходе расчетов была найдена общая сумма затрат на разработку программного продукта, равная 1 061 817,475 тг. В ходе расчета этой суммы учитывались показатели материальных затрат, затрат на оплату труда, отчисления на социальные нужды, амортизационные отчисления и прочие затраты. Для вычисления договорной цены, равная 1380362,72 тг, использовалась рентабельность разработчика, принятая за 30%.

Для нахождения экономической эффективности программного продукта было приведено сравнение с деятельностью предприятия без использования информационной системы. В результате сравнения, наблюдается сокращение следующих затрат: заработная плата сотрудников на 2 100 000 тг, социальные отчисления и налоги на 217 560 тг., расходуемые материалы на 366 000 тг., амортизационные отчисления на 121 308 тг., накладные расходы на 1 470 000 тг. Ожидаемая условно-годовая экономия составила 4 274 868 тг. Отсюда можно сделать вывод, что внедрение информационной системы позволяет предприятию сэкономить 72% от общей суммы годовых эксплуатационных затрат.

Найденное значение коэффициента экономической эффективности капитальных вложений позволило рассчитать срок окупаемости капитальных вложений, который в итоге равняется 0.4 годам. Этот показатель доказывает что продукт окупится за 4 месяца после эксплуатации.

В целом можно считать данную работу экономически целесообразной и актуальной.

5 Безопасность жизнедеятельности

5.1 Порядок проектирования рабочего места оператора

1) Выбор производственной среды (помещения). Определение его класса по опасности поражения электротоком и категорию по взрыво- и пожароопасности. В соответствие с этим, выбор требуемых технических средств (огнетушителей, аптечек и т.п.). Анализ необходимости принятия дополнительных мер по обеспечению параметров микроклимата, освещения, шума и вибрации.

2) Расчет количества рабочих мест операторов в помещении, проектирование их размещения. Разработка рабочих мест (выбор столов, стульев, перегородок между рабочими местами и т.п.).

3) Выбор оборудования (типов мониторов, определения количества принтеров, необходимости использования сканеров, источников бесперебойного питания и т.п.). При этом возможна корректировка п.2.

4) Проектирование схемы подключения оборудования с учетом электробезопасности.

5.2 Анализ и проектирование производственной среды

Необходимо привести характеристику помещения для размещения ИС, определить его класс по опасности поражения электротоком и категорию по взрыво и пожароопасности.

Помещение в автомастерской является взрывоопасным помещением. В основном опасность представляют наличие аппаратов, автомобилей и расходных материалов, которые могут к этому привести.

Помещения особо опасные (большая часть производственных помещений, в том числе все цехи машиностроительных заводов, испытательные станции, гальванические цехи, мастерские) характеризуются наличием одного из следующих трех условий, создающих особую опасность:

- особой сырости, когда относительная влажность воздуха близка к 100% (стены, пол и предметы, находящиеся в помещении, покрыты влагой); такие помещения называются особо сырыми;
- химически активной или органической среды, т. е. помещения, в которых постоянно или в течение длительного времени содержатся агрессивные пары, газы, жидкости, образующие отложения или плесень, действующие разрушающе на изоляцию и токоведущие части электрооборудования; такие помещения называются помещениями с химически активной или органической средой;
- одновременного наличия двух и более условий, свойственных помещениям с повышенной опасностью.

Таблица 5.1 - Категории помещений по взрывопожарной опасности

Категория	Степень	Вещества, используемые	Опасные

	опасности	(выделяющиеся) в процессе производства*	(вредные) факторы
1	2	3	4
Б	Взрывопожаро-опасная	А)ЛВЖ Твсп > 28°С Б) ГП с Сн<65г/м^3	-«-

ЛВЖ – легко воспламеняющиеся жидкости

ТВ – твердые вещества

ГП – горючие пыли

Отсюда следует, что помещение, где будет размещено ИС, необходимо установить огнетушители.

Ручные углекислотные огнетушители (например ОУ-5), устанавливаются в помещениях с вычислительным оборудованием из расчета один огнетушитель на 40-50м^2 площади, но не менее двух в помещении.

Необходимо определить количество огнетушителей, выбрать их марку и место размещения.

Помещения с ПК должны быть оснащены и аптечкой первой помощи.

Необходимо выбрать место ее размещения.

При выполнении работ с ПК температура воздуха должна быть 21-24°С при относительной влажности 40-60% и скорости его движения не более 0,1м/с. Уровень шума не должен превышать 50дБА. Помещения для эксплуатации ПК должны иметь естественное и искусственное освещение. Освещенность на поверхности стола в зоне размещения рабочего документа должна быть 300-500лк. Освещение не должно создавать бликов на поверхности экрана. Освещенность поверхности экрана не должна быть более 300 лк. Коэффициент естественной освещенности должен быть не ниже 1,2-1,5%.

5.3 Проектирование рабочих мест

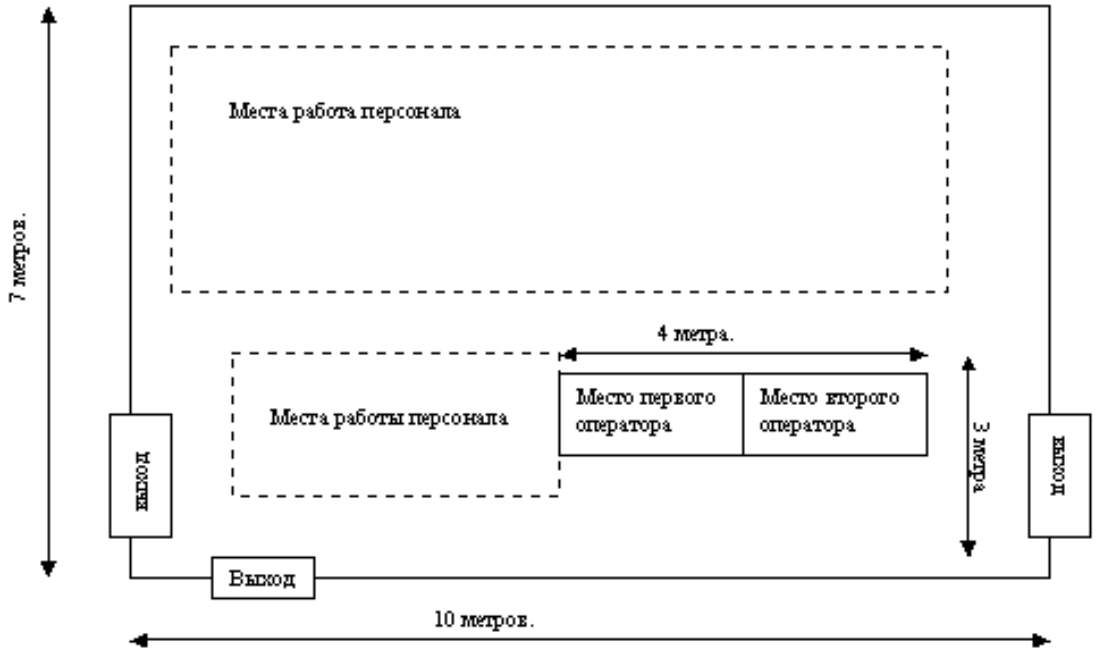


Рисунок 5.1 – Схема рабочего места

Характеристики помещения:
- площадь на одного рабочего – 6 м. в кв.;
- объем на одного рабочего – 20 м. в кубе.

5.4 Выбор оборудования

Необходимо определить:

1) требуемые характеристики системного блока;
2) тип монитора;
3) необходимость использования дополнительного оборудования (принтер, сканер, плоттер и т.п.);

Для работы с ИС достаточно иметь ПК средней комплектации. Достаточно использовать материнскую с сокетом под Intel Core i3, 4 гб оперативной памяти и 500 гб жесткого диска. Для удобной работы с ИС будет использован Full HD монитор с яркостью 200 кд/м. в кв, контрастностью 1000:1 и без мерцаний. Для ИС не требуется дополнительное оборудование, так как все выполняется внутри и исключает возможность возвращения к бумажной работе.

5.5 Проектирование схемы подключения оборудования

Помещения, где размещаются рабочие места с ПК, для защиты от косвенного прикосновения должны быть оборудованы защитным заземлением (занулением) в соответствии с техническими требованиями по эксплуатации.

Как правило, для питания ПК применяется система *TN* (глухозаземленная нейтраль трансформатора). Поэтому, в соответствие с ПУЭ, все открытые проводящие части должны быть присоединены к глухозаземленной нейтрали источника питания, то есть используется защитное зануление. В качестве дополнительной меры защиты и для защиты от статического электричества применяется дополнительное заземление корпуса ПК, подводимое к каждому рабочему месту.

Необходимо спроектировать схему подключения. В качестве варианта возможно подключение нулевого защитного проводника через третий контакт евророзетки и дополнительного заземления проводником от земляного контакта на распределительной коробке розеток к винту, крепящему встроенный источник питания в системном блоке.

На рисунке 5.2 показаны рекомендуемые и не рекомендуемые (с точки зрения электробезопасности) варианты компоновки рабочего места.

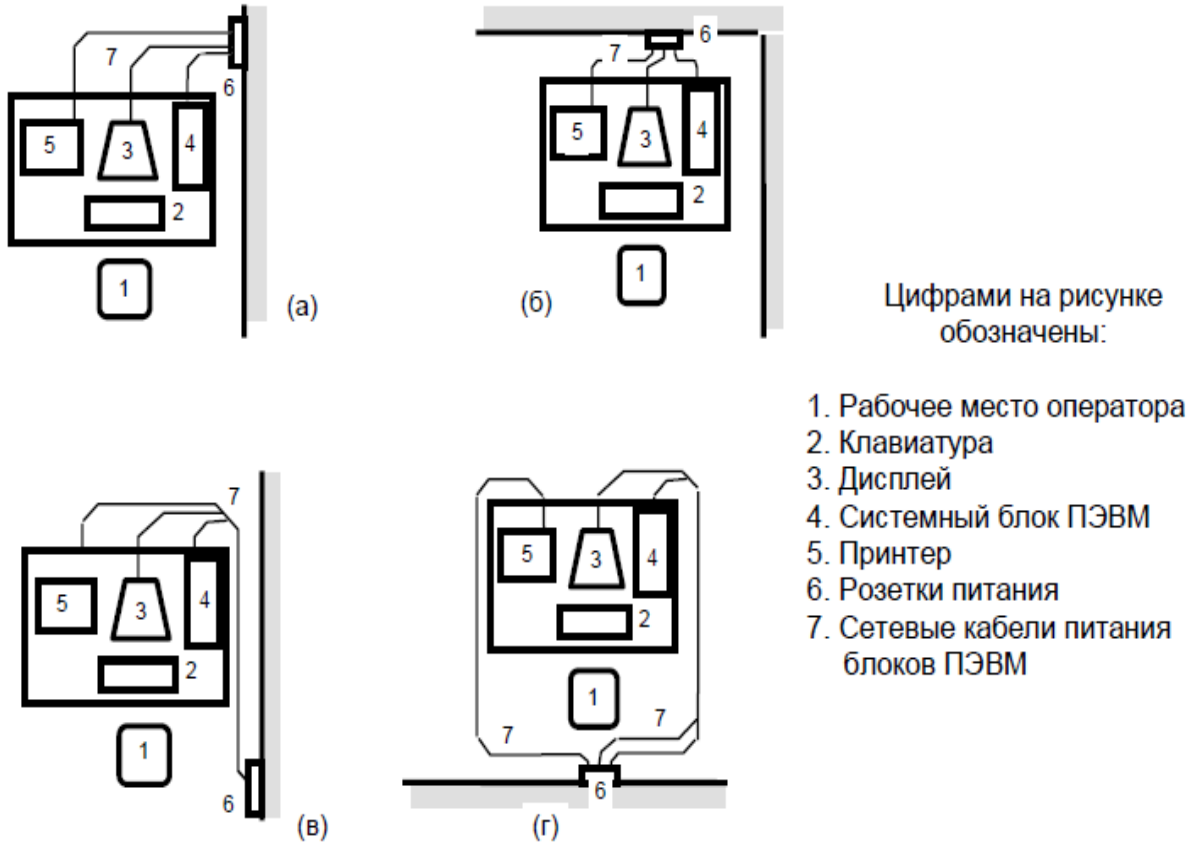


Рисунок 5.2 – Компоновка рабочего места

Наиболее оптимальной следует признать планировку, когда полностью разделены зона местонахождения пользователя ПК и зона, где расположены кабели электропитания технических средств рабочего места, включая розетки сетевого электропитания (рисунок 5.2а, 5.2б). Менее оптимальной является планировка, представленная на рисунке 5.2в., когда рядом с пользователем расположены сетевые кабели электропитания рабочего места. Данную планировку нежелательно использовать, если на рабочем месте установлено большое количество технических средств со значительным энергопотреблением. Крайне нежелательной является планировка на рисунке 5.2г.

В соответствии с набором оборудования и марки сетевого фильтра или источника бесперебойного питания, необходимо определить необходимое число розеток.

Средством защиты от прямого прикосновения является недоступность токоведущих частей. Все токоведущие части закрыты корпусом. Необходимо выбрать такой тип корпуса системного блока ПК, который снабжен блокировкой, недопускающей его снятие при включенном оборудовании. Например, существуют корпуса, предусматривающие их запирание на навесной замок небольшого размера. Возможно предусмотреть пломбирование корпуса путем установки мастичной пломбы на крепежных винтах или бумажной пломбы, наклеиваемой одновременно на съемные и несъемные элементы.

На основании проведенного анализа производственной среды можно сделать следующие выводы.

Наличие взрывоопасных веществ внутри автомастерской, присваивает ей категорию Б в таблице взрывопожарной опасности. Помещения с ПК должны быть оснащены огнетушителем и аптечкой на случай возможного возгорания. Для оптимальной работы, менеджер должен соблюдать технику безопасности при работе с компьютером, а также соблюдать меры пожарной безопасности.

Также в ходе исследования было установлено, что важным условием для безопасной работы с компьютером является расположение кабелей электропитания рядом с техническими средствами.

Заключение

В ходе выполнения дипломного проекта мною были освоены основные принципы создания комплексных информационных систем, позволяющих автоматизировать работу предприятий. Использование информационных систем происходит в целях повышения эффективности производства и упрощения рабочего процесса организации.

На сегодняшний день существует множество многофункциональных систем которые используются автосервисами. Однако слишком часто они не приносят ожидаемого результата. Главной причиной такого явления является высокая стоимость, ограниченность надстроек, непонятный интерфейс и неприспособленность к работе в узких сферах деятельности.

Разработка информационной системы проходила в среде Visual Studio на языке C#. Данная среда разработки по своей сущности является очень удобным инструментом при создании комплексных приложений и систем. Гибкость языка C# позволяет программистам раскрывать весь свой потенциал и благодаря богатой библиотеке разрабатывать приложения любой сложности.

Для хранения информации о заказах, клиенте и т.д. использовалась субд MySQL. Связь с базой данных осуществляется через сервер, который является обязательным компонентом информационной системы. Средство обмена сообщениями, встроенное в информационную систему является одной из главных особенностей ИС для удобной работы менеджера. Используя чат, менеджер может ответить на все возникшие вопросы клиента и получить необходимую информацию для оформления заказа. При оформлении заказа в информационной системе менеджеру предоставляется возможность распечатать электронный чек с информацией о выбранных услугах и общей стоимости заказа.

При расчете экономических показателей была доказана технико-экономическая эффективность внедрения информационной системы в предприятие. Ожидаемая условно-годовая экономия составила 4 274 868 тг., что говорит об экономии в размере 72% от общей суммы годовых эксплуатационных затрат. Также был найден срок окупаемости программного продукта, который равен 0.4 годам.

В ходе исследования производственной среды, автомастерской была присвоена категория взрывопожарной опасности. Также было установлено, что помещения имеющие ПК должны быть оснащены огнетушителями и аптечкой, на случай внезапного возгорания. Важным условием для оптимальной работы менеджера на компьютере является соблюдение техники безопасности и мер пожарной безопасности. Также была выбрана наиболее оптимальная планировка рабочего места. Как оказалось, отдаленность кабелей питания от пользователя ПК и их близкое расположение к техническим средствам являются самым безопасным вариантом планировки при работе с компьютером.

Список литературы

1 Вендров А.М. CASE–технологии. Современные методы и средства проектирования информационных систем. – М.: Финансы и статистика, 2000.

2 Коннолли Т., Бегг К. Базы данных. Проектирование, реализация и сопровождение. Теория и практика = Database Systems: A Practical Approach to Design, Implementation, and Management. – 3–е изд. – М.: Вильямс, 2003. – 1436 с.

3 Грекул В.И., Денищенко Г.Н., Коровкина Н.Л. Проектирование информационных систем. – М.: Интернет–университет информационных технологий – ИНТУИТ.ру, 2005.

4 Соммервилл Иан. Инженерия программного обеспечения / Пер. с англ. – 6–е издание. – М.: Вильямс, 2002. – 624 с.

5 Биллиг В.А. Основы программирования на C#. Издательства: Бином. Лаборатория знаний, Интернет–университет информационных технологий, 2009. 488 с.

6 Гарсиа–Молина Г., Ульман Дж., Уидом Дж. Системы баз данных. Полный курс. – М.: Вильямс, 2003. – 1088 с.

7 Мирзоян Н.В. Оценка стоимости недвижимости. – М.: Московская финансово–промышленная академия, 2005. – 199 с.

8 Смирнова Г.Н. Проектирование экономических информационных систем: Учеб. Для вузов / Г.Н. Смирнова, А.А. Сорокин, Ю.Ф. Тельнов; Под ред. Ю.Ф. Тельнова. –М.: Финансы и статистика, 2002. – 512 с.: ил.

9 Лори Ульрих Фуллер, Кен Кук, Джон Кауфельд. Microsoft Office Access 2007 для «чайников»: Пер. с англ. – М.: «Диалектика», 2007. – 384 стр. с ил.

10 Драпиковский А. И., Иванова И. Б., Игнатенко Н. С., Исаев Н. Б., Лукашова И. В., Мокроусов Н. В., Романенко Л. В. – изд. 2–ое – Б.: «Ега–Басма», 2007. – 480 с.

11 Вальвачев А.Н., Сурков К.А., Сурков Д.А., Четырько Ю.М. Программирование на языке Delphi. Учебное пособие. – 2005.

12 Асаул А. Н., Иванов С. Н., Старовойтов М. К. Экономика недвижимости: учебник для вузов. – 3–е изд., исправл. – СПб.: АНО «ИПЭВ», 2009. – 304 с.

13 Гражданский кодекс Республики Казахстан от 27 декабря 1994 года (с изменениями и дополнениями по состоянию на 24.11.2015 г.) – [электронный ресурс]: www.online.zakon.kz

14 Гагарина Л.Г., Кокорева Е.В., Виснадул Б.Д. Технология разработки программного обеспечения. – М.: ИД «ФОРУМ»; ИНФРА–М, 2008. – С. 400.

15 Закон Республики Казахстан об оценочной деятельности в Республике Казахстан № 109–II от 30 ноября 2000 года (с изменениями и дополнениями по состоянию на 04.12.2015 г.) – [электронный ресурс]: www.online.zakon.kz

Приложение А
Техническое задание

1 Общие сведения

1.1 Наименование системы

Полное наименование системы:
Информационная система для работы с клиентами автосервиса

1.2 Сроки начала и окончания работ

Дата начала: 18.02.2020
Дата окончания: 06.05.2020

2 Назначение и цели создания системы

2.1 Назначение системы

Информационная система предназначена для автоматизации работы автосервиса при обслуживании клиентов.

3 Рекомендации к разработке программы

Информационная система может быть разработана на языке C# с использованием СУБД Microsoft SQL Server.

4 Требование к внешнему виду системы

Обязательным условием является простота и понятливость интерфейса, не яркая, приятная цветовая гамма.

4 Технические требования

Специальных технических требований для ПК для работы приложения нет, кроме как Windows 7 или. Подойдёт любой ПК, на котором может работать Windows.

5 Экономические требования

- возможная (договорная) цена продукта составила 1 380 362 тг;
- стоимость разработки продукта составила 1 061 817 тг.

Приложение Б

(листинг программы)

Форма Program.cs

```
using Newtonsoft.Json.Linq;
using System;
using System.Collections.Generic;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Net;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace DastanDiplom

{
   static class Program
   {
      /// <summary>
      /// Главная точка входа для приложения.
      /// </summary>
      public static readonly string domain = "http://localhost/";
      public static readonly string selectTableAdress = "api/table/select";
      public static readonly string insertTableAdress = "api/table/insert";
      public static readonly string updateTableAdress = "api/table/update";
      public static readonly string deleteTableAdress = "api/table/delete";
      public static event Action onPurchaseChange;
      static Dictionary<string, Form> formDic = new Dictionary<string, Form>();
      [STAThread]
      static void Main()
      {
         Application.EnableVisualStyles();
         Application.SetCompatibleTextRenderingDefault(false);
         Application.Run(new Login());
      }
      public static void onPurchaseChangeInvoke()
      {
         onPurchaseChange.Invoke();
      }
      public static void EnableWindow<T>(bool active) where T : Form, new()
      {
         Form form = null;
         if (!formDic.ContainsKey(typeof(T).ToString()))
         {
            form = new T();
            formDic.Add(typeof(T).ToString(), form);
         }
         else
         {
            form = formDic[typeof(T).ToString()];
         }
         if (active)
```

```
    {
        form.Show();
        form.Focus();
    }
    else
    {
        form.Hide();
    }
}
public static string[,] FormatToGrid(List<Dictionary<string, string>> objects)
{
    string[] headers = objects[0].Keys.ToArray();
    string[,] grid = new string[objects.Count + 1, headers.Length];
    for (int i = 0; i < headers.Length; i++)
    {
        grid[0, i] = headers[i];
    }
    for (int i = 0; i < objects.Count; i++)
    {
        for (int j = 0; j < headers.Length; j++)
        {
            grid[i + 1, j] = objects[i][headers[j]];
        }
    }
    return grid;
}
public static string ToJsonArray(List<Dictionary<string, string>> dic)
{
    string s = "[";
    for (int i = 0; i < dic.Count; i++)
    {
        s += ToJson(dic[i]);
        s += ",";
    }
    s = s.Remove(s.Length - 1);
    s += "]";
    return s;
}
public static List<Dictionary<string, string>> FromJsonArray(string json)
{
    List<Dictionary<string, string>> diclist = new List<Dictionary<string, string>>();
    json = json.Replace("[", "").Replace("]", "");
    var objects = json.Replace("{", "").Split('}');
    foreach (var obj in objects)
    {
        if (obj != "")
        {
            var val = FromJson(obj)
            diclist.Add(val);
        }
    }
```

```
    return diclist;
}
public static string ToJson(Dictionary<string, string> dic)
{
    string s = "{";
    foreach (var kvp in dic)
    {
        if (kvp.Value[0] != '[')
        {
            s += "\"" + kvp.Key + "\":" + "\"" + kvp.Value + "\"";
        }
        else
        {
            s += "\"" + kvp.Key + "\":" + kvp.Value;
        }
        s += ",";
    }
    s = s.Remove(s.Length - 1);
    s += "}";
    return s;
}
public static Dictionary<string, string> FromJson(string json)
{
    Dictionary<string, string> dic = new Dictionary<string, string>();
    var objFields = json.Split(',');
    foreach (var o in objFields)
    {
        if (o != "")
        {
            var o_ = o.Split(':');
            dic.Add(o_[0].Replace("\"", ""), o_[1].Replace("\"", ""));
        }
    }
    return dic;
}
public static int GetColumnIndex(string headerName,DataGridView dgv)
{
    int m = dgv.ColumnCount;
    int ret = -1;
    for (int i = 0; i < m; i++)
    {
        if (dgv.Columns[i].HeaderText == headerName)
        {
            ret = i;
        }
    }
    return ret; }
public static string ToQueryString(Dictionary<string,string> dic)
{
    string s = "";
    foreach (KeyValuePair<string,string> kvp in dic)
```

```
    {
        s += kvp.Key;
        s += "=";
        s += kvp.Value;
        s += "&";
    }
    if (s.Length > 0)
        s = s.Substring(0, s.Length - 1);
    return s;
}
public static string GetRequest(string url, Dictionary<string, string> query)
{
    query.Add("login", Login.login);
    query.Add("password", Login.password);
    query.Add("user_type", Login.user_type.ToString());
    string queryString = ToQueryString(query);
    url += "?" + queryString;
    var webClient = new WebClient();
    webClient.Encoding = Encoding.UTF8;
    return webClient.DownloadString(url);
}
public static string SelectTableRequest(Table table,Dictionary<string, string> query = null)
{
    if (query == null)
        query = new Dictionary<string, string>();
    var keysToUpdate = new List<string>();
    foreach (var entry in query)
    {
        keysToUpdate.Add(entry.Key);
    }
    foreach (string keyToUpdate in keysToUpdate)
    {
        string value = query[keyToUpdate];
        string newKey ="option_" + keyToUpdate;
        // increment the key until arriving at one that doesn't already exist
        query.Remove(keyToUpdate);
        query.Add(newKey, value);
    }
    query.Add("login", Login.login);
    query.Add("password", Login.password);
    query.Add("user_type", Login.user_type.ToString());
    query.Add("table",table.ToString());
    string queryString = ToQueryString(query);
    string url = domain+selectTableAdress;
    url += "?" + queryString;
    var webClient = new WebClient();
    webClient.Encoding = Encoding.UTF8;
    return webClient.DownloadString(url);
}
public static void FormatDataGridView(DataGridView dgv)
{
```

```
            dgv.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;
            dgv.AutoSizeRowsMode = DataGridViewAutoSizeRowsMode.DisplayedCells;
            dgv.DefaultCellStyle = new DataGridViewCellStyle(dgv.DefaultCellStyle) { WrapMode
= DataGridViewTriState.True };
        }
        public static List<string> GetHeaderTexts(DataGridView dgv)
        {
            List<string> headerTexts = new List<string>();
            for (int i = 0; i < dgv.ColumnCount; i++)
            {
                headerTexts.Add(dgv.Columns[i].HeaderText);
            }
            return headerTexts;
        }
        public static Dictionary<string, TextBox> CreateTextBoxesColumn(List<string> labels,
Control parent,int space,int horizontalPadding)
        {
            Dictionary<string, TextBox> outDictionary = new Dictionary<string, TextBox>();
            Point point = new Point(horizontalPadding, 5);
            for (int i = 0; i < labels.Count; i++)
            {
                var s = labels[i];
                Label lab = new Label();
                lab.Text = s;
                lab.Location = point;
                lab.Parent = parent;
                int height = lab.GetPreferredSize(Size.Empty).Height + space;
                int width = parent.Width - horizontalPadding * 3;
                point.Y += height;
                TextBox textB = new TextBox();
                textB.Location = point;
                Size buttonSize = lab.GetPreferredSize(Size.Empty);
                textB.Size = new Size(width, height);
                textB.Parent = parent;
                point.Y += Convert.ToInt32(height * 1.5f);
                outDictionary.Add(s, textB);
            }
            return outDictionary;
        }
        public static void InsertIntoDataGridView(JArray jArray, DataGridView dgv)
        {
            if (jArray.Count>0)
            {
            if (dgv.ColumnCount < jArray.Children<JObject>().ToArray()[0].Properties().Count())
                foreach (var j in jArray.Children<JObject>().ToArray()[0].Properties())
                {
                    dgv.ColumnCount++;
                    dgv.Columns[dgv.ColumnCount - 1].HeaderText = j.Name;
                }
            for (int i = 0; i < jArray.Count; i++)
            {
```

```
                List<object> row = new List<object>();
                for (int j = 0; j < dgv.ColumnCount; j++)
                {
                    var kek = jArray[i].Value<string>(dgv.Columns[j].HeaderText);
                    if (kek != null)
                    {
                        var num = 0;
                        if (int.TryParse(kek, out num))
                        {
                            row.Add(num);
                        }
                        else
                        {
                            row.Add(kek);
                        }
                    }
                    else
                    {
                        row.Add(null);
                    }
                }
                dgv.Rows.Add(row.ToArray());
            }
        }
    }
    public static string DeleteFromTableRequest(Table table, string postString, string
queryString = "")
    {
        string url = domain + deleteTableAdress;
        var query = new Dictionary<string, string>();
        query.Add("table", table.ToString());
        query.Add("login", Login.login);
        query.Add("password", Login.password);
        query.Add("user_type", Login.user_type.ToString());
        queryString += ToQueryString(query);
        var httpWebRequest = (HttpWebRequest)WebRequest.Create(url + "?" + queryString);
        httpWebRequest.ContentType = "application/json";
        httpWebRequest.Method = "POST";
        using (var streamWriter = new StreamWriter(httpWebRequest.GetRequestStream()))
        {
            streamWriter.Write(postString);
            streamWriter.Flush();
            streamWriter.Close();
        }
        string result = "";
        var httpResponse = (HttpWebResponse)httpWebRequest.GetResponse();
        using (var streamReader = new StreamReader(httpResponse.GetResponseStream()))
        {
            result = streamReader.ReadToEnd();
        }
        return result;
```

```
        }
        public static string UpdateTableRequest(Table table, string postString, string queryString =
"")
        {
            string url = domain + updateTableAdress;
            var query = new Dictionary<string, string>();
            query.Add("table", table.ToString());
            query.Add("login", Login.login);
            query.Add("password", Login.password);
            query.Add("user_type", Login.user_type.ToString());
            queryString += ToQueryString(query);
            var httpWebRequest = (HttpWebRequest)WebRequest.Create(url + "?" + queryString);
            httpWebRequest.ContentType = "application/json";
            httpWebRequest.Method = "POST";
            using (var streamWriter = new StreamWriter(httpWebRequest.GetRequestStream()))
            {
                streamWriter.Write(postString);
                streamWriter.Flush();
                streamWriter.Close();
            }
            string result = "";
            var httpResponse = (HttpWebResponse)httpWebRequest.GetResponse();
            using (var streamReader = new StreamReader(httpResponse.GetResponseStream()))
            {
                result = streamReader.ReadToEnd();
            }
            return result;
        }
        public static string InsertIntoTableRequest(Table table,string postString, string
queryString="")
        {
            //Сделать обработку ошибки при пустом заказе
            string url = domain + insertTableAdress;
            var query = new Dictionary<string, string>();
            query.Add("table", table.ToString());
            query.Add("login", Login.login);
            query.Add("password", Login.password);
            query.Add("user_type", Login.user_type.ToString());
            queryString += ToQueryString(query);
            var httpWebRequest = (HttpWebRequest)WebRequest.Create(url + "?" + queryString);
            httpWebRequest.ContentType = "application/json";
            httpWebRequest.Method = "POST";
            using (var streamWriter = new StreamWriter(httpWebRequest.GetRequestStream()))
            {
                streamWriter.Write(postString);
                streamWriter.Flush();
                streamWriter.Close();
            }
            string result = "";
            var httpResponse = (HttpWebResponse)httpWebRequest.GetResponse();
            using (var streamReader = new StreamReader(httpResponse.GetResponseStream()))
```

```
            {
                result = streamReader.ReadToEnd();
            }
            return result;
        }
        public enum Table
        {
            car_part,
            message,
            purchase,
            purchase_service,
            purchase_service_list,
            service,
            service_type,
            user,
            user_type
        }
    }
}
```

Класс Login.cs

```
public partial class Login : Form
    {
        public static readonly int user_type = 3;
        public static string login;
        public static string password;
        public static int clientID;
        public Login()
        {
            InitializeComponent();
        }
        public static string LoginPassword { get { return "login=" + login + "&" + "password=" +
password; } }
        void SendRequest()
        {
            string values = string.Format("login={0}&password={1}&user_type_id={2}",
loginField.Text, passwordField.Text,user_type);
            var url = "http://localhost/api/logincheck?" + values;
            // Создаём объект WebClient
            using (var webClient = new WebClient())
            {
                // Выполняем запрос по адресу и получаем ответ в виде строки
                try
                {
                    var response = webClient.DownloadString(url);

                    if (response != "")
                    {
                        login = loginField.Text;
                        password = passwordField.Text;
                        clientID = int.Parse(JArray.Parse(response)[0]["id"].ToString());
```

```
                this.Hide();
```

```
                var managersMain = new MainForm();
                managersMain.Closed += (s, args) => this.Close();
                managersMain.Show();
            }
            else
            {
            }
        }
        catch(System.Net.WebException e)
        {
            MessageBox.Show("Отсутсвует соединение с сервером");
        }
    }
}
private void button1_Click_1(object sender, EventArgs e)
{
    SendRequest();
}
}
```

Форма MainForm.cs

```
public partial class MainForm : Form
  {
    JObject purchases = new JObject() { {"meta",new JArray() },{"post",new JArray() } };
    Clients clients = new Clients();
    Dictionary<string, TextBox> carPartSearchBoxes = new Dictionary<string, TextBox>();
    Dictionary<string, TextBox> clientTextBoxes = new Dictionary<string, TextBox>();
    Dictionary<string, TextBox> purchaseServicesBoxes = new Dictionary<string, TextBox>();
    public MainForm()
    {
        InitializeComponent();
    }
```

Загрузка формы

```
    private void MainForm_Load(object sender, EventArgs e)
    {
        UpdateTables();
        UpdateClients();
        SetActiveTextBoxes(false);
    }
    void UpdateClients()
    {
        clients.UpdateByJsonString(Program.SelectTableRequest(Program.Table.user, new
Dictionary<string, string>() { { "login",Login.login },{"password",Login.password },{
"user_type_id", "3" } }));
        foreach (var c in clients.clients)
            clientTextBox.Text = c.Value.id + ":" + c.Value.first_name + " " +
c.Value.second_name + " " + c.Value.third_name;
        clientComboBox_SelectedIndexChanged();
    }
    void UpdateTables()
```

```
        {
            purchaseServicesDGV.Rows.Clear();
```

```
            carPartsDGV.Rows.Clear();
            serviceListDGV.Rows.Clear();
Program.InsertIntoDataGridView(JArray.Parse(Program.SelectTableRequest(Program.Table.car
_part)),carPartsDGV);
Program.InsertIntoDataGridView(JArray.Parse(Program.SelectTableRequest(Program.Table.pur
chase_service)), purchaseServicesDGV);
Program.InsertIntoDataGridView(JArray.Parse(Program.SelectTableRequest(Program.Table.ser
vice)), serviceListDGV);
            Program.FormatDataGridView(carPartsDGV);
            Program.FormatDataGridView(serviceListDGV);
            Program.FormatDataGridView(purchaseServicesDGV);
            carPartSearchBoxes =
Program.CreateTextBoxesColumn(Program.GetHeaderTexts(carPartsDGV),
carPartSearchPanel, 10, 10);
            purchaseServicesBoxes =
Program.CreateTextBoxesColumn(Program.GetHeaderTexts(purchaseServicesDGV),
purchaseServicesSearchPanel, 10, 10);
            JObject client = JObject.Parse(JsonConvert.SerializeObject(new Clients.Client()));
            List<string> clientFields = new List<string>();
            foreach (var k in client)
            {
                clientFields.Add(k.Key);
            }
            clientTextBoxes = Program.CreateTextBoxesColumn(clientFields, clientSidePanel, 10,
10);
            foreach(var kvp in clientTextBoxes)
            {
                kvp.Value.BorderStyle = BorderStyle.None;
                kvp.Value.ReadOnly = true;
                //kvp.Value.Multiline = true;
                kvp.Value.Margin = new Padding() {Top=5,Bottom=5,Left=5,Right=5 };
                kvp.Value.BackColor = SystemColors.ControlLightLight;
               // kvp.Value.BackColor = System.Drawing.Color.Aqua;
            }
        }
        private void button4_Click(object sender, EventArgs e)
        {
            OpenDoc();
            System.Diagnostics.Process.Start(@"test.pdf");
        }
        //Очистить
        private void button2_Click(object sender, EventArgs e)
        {
            foreach (var j in purchases.Children<JArray>())
            {
                j.Clear();
            }
        }
        void OpenDoc()
```

```
    {
        FileInfo file = new FileInfo(@"test.pdf");
```

```
        file.Directory.Create();
        try
        {
            PdfDocument pdfDoc = new PdfDocument(new PdfWriter(@"test.pdf"));
            Document doc = new Document(pdfDoc);
            float width = (pdfDoc.GetNumberOfPages() > 0) ?
pdfDoc.GetFirstPage().GetPageSize().GetWidth() : pdfDoc.GetDefaultPageSize().GetWidth();
            Thread.CurrentThread.CurrentCulture = CultureInfo.GetCultureInfo("en-GB");
            PdfFont font = PdfFontFactory.CreateFont(@"title.otf", "Cp1251", true);
            #region align
            SolidLine line = new SolidLine();
            Paragraph title = AddTabsToParagraph(line, width - doc.GetLeftMargin() * 2, "Бланк
заказа");
            // Draw a custom line to fill both sides, as it is described in iText5 example
            MyLine customLine = new MyLine();
            #endregion align
            title.SetFont(font);
            title.SetFontSize(20);
           title.SetHorizontalAlignment(iText.Layout.Properties.HorizontalAlignment.CENTER);
            doc.Add(title);
            Paragraph date = new Paragraph(DateTime.Now.ToLongDateString());
            date.SetFont(font);
            date.SetFontSize(12);
            ate.SetHorizontalAlignment(iText.Layout.Properties.HorizontalAlignment.CENTER);
            doc.Add(date);
            //foreach two tables
            int allCost = 0;
            Table table = new Table(UnitValue.CreatePercentArray(1)).UseAllAvailableWidth();
            Table purchaseTable = new
Table(UnitValue.CreatePercentArray(2)).UseAllAvailableWidth();
            var p1 = new Paragraph("Название");
            var p2 = new Paragraph("Стоимость");
            p1.SetFont(font);
            p2.SetFont(font);
            purchaseTable.AddHeaderCell(p1);
            purchaseTable.AddHeaderCell(p2);
            int fullCost = 0;
            for (int k = 0; k < purchases.Count; k++)
            {
                Paragraph name = new
Paragraph(purchases["meta"].Value<JArray>()[k]["name"].ToString());
                name.SetFont(font);
                Paragraph cost = new
Paragraph(purchases["meta"].Value<JArray>()[k]["cost"].ToString());
                cost.SetFont(font);
                purchaseTable.AddCell(name);
                purchaseTable.AddCell(cost);
                fullCost += purchases["meta"].Value<JArray>()[k]["cost"].Value<int>();
            }
```

```
            table.AddCell(purchaseTable);
            doc.Add(table);
```

Продолжение приложения Б

```
            Table fullCostTable = new
Table(UnitValue.CreatePercentArray(2)).UseAllAvailableWidth();
            var fullCostP = new Paragraph("Полная стоимость");
            fullCostP.SetFont(font);
            fullCostTable.AddCell(fullCostP);
            fullCostTable.AddCell(fullCost.ToString());
            doc.Add(fullCostTable);
            var d = new Paragraph();
            List<TabStop> tabStops = new List<TabStop>();
            // Create a TabStop at the middle of the page
            tabStops.Add(new TabStop(width / 2,
iText.Layout.Properties.TabAlignment.CENTER, line));
            var clientSignature = new Paragraph();
            var managerSignature = new Paragraph();
            clientSignature.SetFont(font);
            managerSignature.SetFont(font);
            clientSignature.Add("Подпись клиента").Add(new Tab());
            managerSignature.Add("Подпись менеджера").Add(new Tab());
            clientSignature.AddTabStops(tabStops);
            managerSignature.AddTabStops(tabStops);
            d.Add(clientSignature);
            d.Add(managerSignature);
            doc.ShowTextAligned(d, pdfDoc.GetFirstPage().GetPageSize().GetWidth() / 2,
pdfDoc.GetFirstPage().GetPageSize().GetBottom() + 20, 1, TextAlignment.CENTER,
VerticalAlignment.BOTTOM, 0);
            //doc.Add(d);
            doc.Close();
        }
        catch (System.IO.IOException)
        {
            MessageBox.Show("Пожалуйста, закройте приложение, использующее файл
test.pdf");
        }
    }
    private static Paragraph AddTabsToParagraph(iText.Kernel.Pdf.Canvas.Draw.ILineDrawer
line, float width, string text)
    {
        List<TabStop> tabStops = new List<TabStop>();
        // Create a TabStop at the middle of the page
        tabStops.Add(new TabStop(width / 2, iText.Layout.Properties.TabAlignment.CENTER,
line));
        // Create a TabStop at the end of the page
        tabStops.Add(new TabStop(width, iText.Layout.Properties.TabAlignment.LEFT, line));

        Paragraph p = new Paragraph().AddTabStops(tabStops);
        p
            .Add(new Tab())
            .Add(text)
            .Add(new Tab());
```

```
    return p;
}
```

Продолжение приложения Б

```
private class MyLine : ILineDrawer
{
    private float lineWidth = 1;
    private float offset = 2.02f;
    private iText.Kernel.Colors.Color color = ColorConstants.BLACK;

    public void Draw(PdfCanvas canvas, iText.Kernel.Geom.Rectangle drawArea)
    {
        float coordY = drawArea.GetY() + lineWidth / 2 + offset;
        canvas
            .SaveState()
            .SetStrokeColor(color)
            .SetLineWidth(lineWidth)
            .MoveTo(drawArea.GetX(), coordY)
            .LineTo(drawArea.GetX() + drawArea.GetWidth(), coordY)
            .Stroke()
            .RestoreState();
    }
    public float GetLineWidth()
    {
        return lineWidth;
    }
    public void SetLineWidth(float lineWidth)
    {
        this.lineWidth = lineWidth;
    }
    public iText.Kernel.Colors.Color GetColor()
    {
        return color;
    }
    public void SetColor(iText.Kernel.Colors.Color color)
    {
        this.color = color;
    }
    public float GetOffset()
    {
        return offset;
    }
    public void SetOffset(float offset)
    {
        this.offset = offset;
    }
}
private void changeCarPartBtn_Click(object sender, EventArgs e)
{
    EnableCarPartsChangins(true);
}
private void saveChangingsCarPartsBtn_Click(object sender, EventArgs e)
{
```

```
    EnableCarPartsChangins(false);
```

Продолжение приложения Б

```
    JArray carParts = new JArray();
    //
    var headers = new List<string>();
    for(int i = 0; i < carPartsDGV.ColumnCount; i++)
    {
        headers.Add(carPartsDGV.Columns[i].HeaderText);
    }
    for(int i = 0; i < carPartsDGV.RowCount; i++)
    {
        JObject carPart = new JObject();
        bool rowIsEmpty = true;
        for (int j = 0; j < headers.Count; j++)
        {
            var val = carPartsDGV.Rows[i].Cells[j].Value;
            if (val != null)
            {
                carPart[headers[j]] = val.ToString();
                rowIsEmpty = false;
            }
            else
            {
                carPart[headers[j]] = null;
            }
        }
        if (!rowIsEmpty)
            carParts.Add(carPart);
    }
    Program.UpdateTableRequest(Program.Table.car_part, carParts.ToString());
    UpdateTables();
}
void EnableCarPartsChangins(bool active)
{
    carPartsDGV.ReadOnly = !active;
}
//enable editing
private void button8_Click(object sender, EventArgs e)
{
    SetActiveTextBoxes(true);
    button8.Enabled = false;
    button9.Enabled = true;
    button1.Enabled = true;
}
//save changes
private void button9_Click(object sender, EventArgs e)
{
    SetActiveTextBoxes(false);
    button9.Enabled = false;
    button8.Enabled = true;
    button1.Enabled = false;
```

```
    Clients.Client client = new Clients.Client();
    client.user_type_id = "3";
```

```
    client.id = clientTextBox.Text.Split(':')[0];
    client.first_name = firstNameBox.Text;
    client.second_name = secondNameBox.Text;
    client.third_name = thirdNameBox.Text;
    client.IIN = iinBox.Text;
    client.car_model = carModelBox.Text;
    client.car_numbers = carNumbersBox.Text;
    client.login = loginBox.Text;
    client.password = passwordBox.Text;
    client.phone_number = phoneNumBox.Text;
    //update user request
    Program.UpdateTableRequest(Program.Table.user, JsonConvert.SerializeObject(client));
    //update users combobox request
    UpdateClients();
}
void SetActiveTextBoxes(bool active)
{
    firstNameBox.Enabled = active;
    secondNameBox.Enabled = active;
    thirdNameBox.Enabled = active;
    passwordBox.Enabled = active;
    loginBox.Enabled = active;
    iinBox.Enabled = active;
    carModelBox.Enabled = active;
    carNumbersBox.Enabled = active;
    phoneNumBox.Enabled = active;
}
//new user
private void button10_Click(object sender, EventArgs e)
{
    SetActiveTextBoxes(true);
    ClearUserDataTextBoxes();
    button1.Enabled = true;
    button8.Enabled = false;
    button9.Enabled = false;
}
        //register client
private void button6_Click(object sender, EventArgs e)
{
    SetActiveTextBoxes(false);
    button8.Enabled = true;
    button1.Enabled = false;
    Clients.Client client = new Clients.Client();
    client.user_type_id = "3";
    client.first_name = firstNameBox.Text;
    client.second_name = secondNameBox.Text ;
    client.third_name = thirdNameBox.Text;
    client.IIN =iinBox.Text;
    client.login = loginBox.Text;
```

```
            client.password = passwordBox.Text;
            client.car_model = carModelBox.Text;
```

```
            client.car_numbers = carNumbersBox.Text;
            client.phone_number = phoneNumBox.Text;
            //insert user request
Program.InsertIntoTableRequest(Program.Table.user,JsonConvert.SerializeObject(client).ToStri
ng());
            //update users combobox request
            UpdateClients();
        }
        private void button7_Click(object sender, EventArgs e)
        {
            ClearUserDataTextBoxes();
        }
        void ClearUserDataTextBoxes()
        {
            firstNameBox.Clear();
            secondNameBox.Clear();
            thirdNameBox.Clear();
            carModelBox.Clear();
            iinBox.Clear();
        }
        //cancel
        private void button1_Click_1(object sender, EventArgs e)
        {
            SetActiveTextBoxes(false);
            button8.Enabled = true;
            button9.Enabled = false;
        }
        class Clients
        {
            public Dictionary<string, Client> clients { get; } = new Dictionary<string, Client>();
            public class Client
            {
                public string id;
                public string user_type_id;
                public string first_name;
                public string second_name;
                public string third_name;
                public string IIN;
                public string phone_number;
                public string login;
                public string password;
                public string car_model;
                public string car_numbers;
            }
            public void UpdateByJsonString(string json)
            {
                clients.Clear();
                JArray jArray = JArray.Parse(json);
                foreach(var t in jArray)
```

```
            {
```

Продолжение приложения Б

```
Client client = JsonConvert.DeserializeObject<Client>(t.Value<JObject>().ToString());
                clients.Add(client.id,client);
            }
        }
    }

    private void deleteCarPartBtn_Click(object sender, EventArgs e)
    {
        Int32 selectedRowCount =
carPartsDGV.Rows.GetRowCount(DataGridViewElementStates.Selected);
        List<int> ids = new List<int>();
        if (selectedRowCount > 0)
        {
            for (int i = 0; i < selectedRowCount; i++)
            {
                int index = carPartsDGV.SelectedRows[i].Index;
                string val = carPartsDGV.Rows[index].Cells[Program.GetColumnIndex("id",
carPartsDGV)].Value.ToString();
                int id = 0;
                if (int.TryParse(val,out id))
                {
                    ids.Add(id);
                }
            }
        }
        JArray data = new JArray();
        foreach(int i in ids)
        {
            JObject jObject = new JObject();
            jObject["id"] = i;
            data.Add(jObject);
        }
        Program.DeleteFromTableRequest(Program.Table.car_part, data.ToString());
    }
    private void carPartsSearchBtn_Click(object sender, EventArgs e)     {
        carPartsDGV.Rows.Clear();
        Dictionary<string, string> query = new Dictionary<string, string>();
        foreach (var kvp in carPartSearchBoxes)
            if (!string.IsNullOrWhiteSpace(kvp.Value.Text))
                query.Add(kvp.Key, kvp.Value.Text);
Program.InsertIntoDataGridView(JArray.Parse(Program.SelectTableRequest(Program.Table.car
_part, query)), carPartsDGV);
    }
    private void clientComboBox_SelectedIndexChanged()
    {
        var id = clientTextBox.Text.Split(':')[0];
        firstNameBox.Text = clients.clients[id].first_name;
        secondNameBox.Text = clients.clients[id].second_name;
        thirdNameBox.Text = clients.clients[id].third_name;
```

```
            iinBox.Text = clients.clients[id].IIN;
            carModelBox.Text = clients.clients[id].car_model;
```

```
            carNumbersBox.Text = clients.clients[id].car_numbers;
            loginBox.Text = clients.clients[id].login;
            passwordBox.Text = clients.clients[id].password;
            phoneNumBox.Text = clients.clients[id].phone_number;
            JObject client = JObject.Parse(JsonConvert.SerializeObject(clients.clients[id]));
            foreach(var kvp in client)
            {
               clientTextBoxes[kvp.Key].Text = kvp.Value.ToString();
            }
        }
        private void purchaseServicesSearchBtn_Click(object sender, EventArgs e)
        {
            purchaseServicesDGV.Rows.Clear();
            Dictionary<string, string> query = new Dictionary<string, string>() { { "client_id",
Login.clientID.ToString() } };
            foreach (var kvp in purchaseServicesBoxes)
               if (!string.IsNullOrWhiteSpace(kvp.Value.Text))
                  query.Add(kvp.Key, kvp.Value.Text);
Program.InsertIntoDataGridView(JArray.Parse(Program.SelectTableRequest(Program.Table.pur
chase_service, query)), purchaseServicesDGV);
        }
        private void button2_Click_1(object sender, EventArgs e)
        {
            Messages messages = new Messages();
            messages.other_id = 1;
            messages.Show();
        }
    }
```

Метод GetMessages

```
public void GetMessages()
        {
            var senderMessages = Program.SelectTableRequest(Program.Table.message, new
Dictionary<string, string>() { { "sender_id", other_id.ToString() }
});//,{"manager_id",Login.clientID.ToString() } });//request + getting response
            var receiverMessages = Program.SelectTableRequest(Program.Table.message, new
Dictionary<string, string>() { { "receiver_id", other_id.ToString() } });
            //lol = lol.Replace("\"", "'");
            if
(!string.IsNullOrWhiteSpace(senderMessages)||!string.IsNullOrWhiteSpace(receiverMessages))
            {
            JArray sMessages = new JArray();
            if (!string.IsNullOrWhiteSpace(senderMessages))
            sMessages = JArray.Parse(senderMessages);//getting arrays of messages

            JArray rMessages = new JArray();
            if (!string.IsNullOrWhiteSpace(receiverMessages))
            rMessages = JArray.Parse(receiverMessages);
```

```
        var list = new List<M_Message>();
        foreach(var mes in sMessages)
           {
        list.Add(mes.ToObject<M_Message>());
           }
        foreach (var mes in rMessages)
           {
        list.Add(mes.ToObject<M_Message>());
           }

        if (list.Count > 0)
           {
        var orderedList = list.OrderBy(x => x.datetime).ToList();//sorting by datetime
        messagePanel.Controls.Clear();

        for (int i = 0; i < orderedList.Count; i++)//adding labels to messagePanel
        {
        Label lbl = new Label();
        lbl.Text = " " + orderedList[i].text + " ";
        if (int.Parse(orderedList[i].sender_id) == Login.clientID)
        {
        lbl.Width = lbl.PreferredWidth + 2;

        lbl.Height = lbl.PreferredHeight + 10;
        lbl.TextAlign = ContentAlignment.MiddleLeft;
        lbl.BackColor = Color.LightBlue;
        lbl.Location = new Point(10, messagePanel.Controls.Count * (lbl.PreferredHeight + 15)
+ 5);
                }
        else
                {
        lbl.Width = lbl.PreferredWidth + 2;

        lbl.Height = lbl.PreferredHeight + 10;
        lbl.TextAlign = ContentAlignment.MiddleRight;

        lbl.BackColor = Color.LightGray;
        lbl.Location = new Point(messagePanel.Width - lbl.Width - 10,
messagePanel.Controls.Count * (lbl.PreferredHeight + 15) + 5);
                }
         messagePanel.Controls.Add(lbl);
             }
           }
         }
      }
```

Утверждаю
Директор ТОО «Капитал Энерго Инвест»
Титова С.А
«20» мая 2020г.

АКТ ВНЕДРЕНИЯ

Настоящий акт составлен о том, что результаты выпускной работы студента НАО АУЭС гр. ИНФ-16-2 очной формы обучения Рыскулбекова Д.Д. на тему «*Разработка информационной системы работы с клиентами автосервиса*» внедрены в ТОО «Капитал Энерго Инвест» и используются в компании для представительства в интернете. Использование результата выпускной работы Рыскулбекова Д.Д. автоматизирует работу автосервиса, а также упрощает работу менеджера с клиентами.

Директор ТОО «Капитал Энерго Инвест»
Титова С.А «подпись, печать»