

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН
Некоммерческое акционерное общество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ»
Кафедра IT-инжиниринг

ДОПУЩЕН К ЗАЩИТЕ

Заведующий кафедрой

PhD, доцент

_____ Т.С. Картбаев

« ____ » _____ 2019 г.

ДИПЛОМНЫЙ ПРОЕКТ

На тему: Разработка базы данных для метрологического центра

Специальность: 5В070400 – «Вычислительная техника и программное обеспечение»

Выполнила: Беленкова А.С. Группа: ВТ-15-2

Научный руководитель: к.ф.-м.н., доцент Аманбаев А.А.

Консультанты:

по экономической части: к.э.н., профессор _____ Ж.Г. Аренбаева
« 13 » _____ 2019 г.

по безопасности
жизнедеятельности: д.т.н., ст. преп. _____ Ш.Ш. Бекбасаров
« 14 » _____ 2019 г.

по применению
вычислительной техники: ст. преп. _____ М.Н. Майкотов
« 14 » _____ 2019 г.

Нормоконтролер: ст. преп. _____ А.А. Айтказина
« 15 » _____ 2019 г.

Рецензент: д.т.н., профессор _____ Р.К. Ускенбаева
« ____ » _____ 2019 г.

Алматы 2019

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

Институт систем управления и информационных технологий

Кафедра IT-инжиниринг

Специальность 5В070400 – «Вычислительная техника и программное обеспечение»

ЗАДАНИЕ

на выполнение дипломного проекта

Студентке Беленковой Анне Сергеевне

Тема проекта: Разработка базы данных для метрологического центра

Утверждена приказом по университету № 124 от «26» октября 2018 г.

Срок сдачи законченного проекта «24» мая 2019 г.

Исходные данные к проекту (требуемые параметры результатов исследования (проектирования) и исходные данные объекта): Руководство системы менеджмента качества на предприятии; международные стандарты ИСО-9001, данные преддипломной практики.

Перечень вопросов, подлежащих разработке в дипломном проекте, или краткое содержание дипломного проекта:

- аналитическая часть;
- проектная часть;
- экспериментальная часть;
- экономическая часть;
- безопасность жизнедеятельности;
- приложение А. Техническое задание;
- приложение Б. Листинг программы;
- приложение В. Акт внедрения.

Перечень графического материала (с точным указанием обязательных чертежей): представлены 11 таблиц, 28 иллюстрации.

Основная рекомендуемая литература:

1 Преснякова, Г.В. Проектирование интегрированных реляционных баз данных: Учебное пособие / Г.В. Преснякова. - М.: КДУ, 2007. - 224 с.

2 Стружкин, Н.П. Базы данных: проектирование: Учебник для академического бакалавриата / Н.П. Стружкин, В.В. Годин. - Люберцы: Юрайт, 2016. - 477 с.

3 Стружкин, Н.П. Базы данных: проектирование, практикум: Учебное пособие для академического бакалавриата / Н.П. Стружкин, В.В. Годин. - Люберцы: Юрайт, 2016. - 291 с.

4 Шпак, Ю.А. Проектирование баз данных. Просто как дважды два / Ю.А. Шпак. - М.: Эксмо, 2007. - 304 с.

Консультации по проекту с указанием относящихся к ним разделов проекта





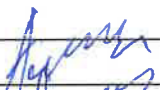
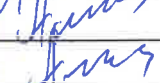

Раздел	Консультант	Сроки	Подпись
Экономическая часть	Аренбаева Ж.Г.	04.03.2019г - 14.05.2019г	
Безопасность жизнедеятельности	Бекбасаров Ш.Ш.	04.03.2019г - 13.05.2019г	
Программное обеспечение	Майкотов М.Н.	04.04.2019г - 14.05.2019г	
Нормоконтролер	Айтказина А.А.	02.04.2019г - 15.05.2019г	

ГРАФИК
подготовки дипломного проекта

Наименование разделов, перечень разрабатываемых вопросов	Сроки представления научному руководителю	Примечание
Аналитическая часть	05.11.2018г - 22.12.2018г	
Проектная часть	07.01.2019г - 30.01.2019г	
Экспериментальная часть	04.02.2019г - 13.04.2019г	

Дата выдачи задания «14» января 2019 г.

Заведующий кафедрой _____ Т.С. Картбаев

Научный руководитель проекта  _____ А.А. Аманбаев

Задание приняла к исполнению студентка  _____ А.С. Беленкова

Аннотация

«Метрология орталығының деректер базасын құру» атты бітіру жобасында зерттеу объектісі - бағдарламалау технологиялары көмегімен метрологиялық орталықтардың тиімді жұмыс істеуі; жоба заманауи технологияларды қолдана отырып және метрология орталығының жұмысын оңайлатумен ыңғайлы пайдаланушы интерфейсі бар деректер базасын құруға бағытталған; бітіру жобасының практикалық құндылығы - бағдарламалық қамтамасыз етудің жалпы деректер базасына жаңа деректерді енгізудің оңайлатылған әдісімен метрологиялық зертханалар мен орталықтар жылдамдығын арттыру.

Дипломдық жобаның келесі міндеттері қойылған: пайдаланушылардың ыңғайлы интерфейсі бар, жаңа технологияларды қолданып, метрология орталығының қызметін жеңілдету, бір бағдарламада барлық деректерді жинау үшін деректер базасын құру. Жұмыстың теориялық негізі жиналған әдебиеттерді және түрлі көздерден алынған ақпараттарды талдау арқылы ұйымдастырылды.

Дипломдық жобаның практикалық маңызы - бағдарламалық қамтамасыз етудің жалпы базасына жаңа деректерді енгізудің оңайлатылған әдісімен метрология зертханалары мен орталықтарының жылдамдығын арттыру.

Аннотация

В дипломном проекте «Разработка базы данных для метрологического центра» объектом исследования является эффективное функционирование работы метрологических центров с помощью технологий программирования; цели проекта направлены на разработку базы данных с комфортным интерфейсом для пользователей, используя новейшие технологии и упрощение работы метрологического центра; практическая ценность дипломной работы состоит в увеличении скорости работы метрологических лабораторий и центров, с помощью упрощенного способа ввода новых данных в общую базу программного продукта.

Ставятся следующие цели дипломного проекта: разработать базу данных с комфортным интерфейсом для пользователей, используя новейшие технологии и упростить работу метрологического центра, собрав все данные в одной программе. Теоретическая основа работы была организована методом анализа собранной литературы и информации с различных источников.

Практическая ценность дипломного проекта - увеличение скорости работы метрологических лабораторий и центров, с помощью упрощенного способа ввода новых данных в общую базу программного продукта.

Annotation

In the graduation project " Database development for metrology center " the object of research is the effective functioning of the metrological centers with the help of programming technologies; the project aims to develop a database with a comfortable user interface, using the latest technology and simplifying the work of the metrology center; the practical value of the graduation project is to increase the speed of the metrological laboratories and centers, using a simplified method for entering new data into the overall database of the software product.

The following objectives of the graduation project are set: to develop a database with a comfortable interface for users, using the latest technologies and to simplify the work of the metrology center, collecting all the data in one program. The theoretical basis of the work was organized by analyzing the collected literature and information from various sources.

The practical value of the graduation project is an increase in the speed of the metrology laboratories and centers, using a simplified method of entering new data into the general base of the software product.

Содержание

Введение	8
1 Метрологическая база данных как программный продукт	10
1.1 Цели и задачи метрологических баз данных	11
1.2 Программный продукт и его пользователи	12
1.3 Сравнительный анализ программного продукта с рынком разработок	12
1.4 Обзор и выбор подходящих технологий и средств разработки	16
2 Проектирование программного продукта	24
2.1 Функционирование программного продукта	24
2.2 Структурирование программного продукта	25
2.3 Разработка модели базы данных	30
2.4 Интерфейс и способы привязки	41
3 Разработка программного продукта	43
4 Экономическая часть	50
4.1 Трудоемкость разработки ПП	50
4.2 Расчет затрат на разработку ПП	51
4.3 Расчет затрат на электроэнергию	53
4.4 Расчет затрат на оплату труда	54
4.5 Расчет затрат по социальному налогу	55
4.6 Амортизация основных фондов и прочие затраты	55
4.7 Смета расходов на разработку ПО	57
4.8 Определение возможной (договорной) цены ПО	58
5 Охрана труда и безопасность жизнедеятельности	60
5.1 Расчёт освещения в метрологических лабораториях для проведения калибровки и поверки средств измерений	63
Заключение	68
Список литературы	69
Приложение А. Техническое задание	71
Приложение Б. Листинг программы	75
Приложение В. Акт внедрения	83

Введение

В предпринимательской либо индивидуальной области деятельности зачастую необходимо проводить работу с информацией с различных источников информации, любой из которых сопряжен с конкретным типом деятельности. С целью координации, данных нужны конкретные познания и организационные умения.

Данные в таких сферах деятельности систематизируются с помощью создания программного обеспечения, направленного, непосредственно, на раскрытие и конструирование правильного пути обработки данных в нужной сфере рабочего процесса. Вся информация хранится и обрабатывается в базах, данных. База данных, в данном случае, это хранилище, доступ к которому получают пользователи извне с помощью интерфейса созданного приложения. Базы данных помогают сохранить в безопасности все данные, внесенные ранее, с помощью ролей и всевозможных видов доступов, что полностью подходит для работы всех видов предприятий, включая государственные учреждения.

Что касается сферы метрологии, стандартизации и сертификации, то только некоторые программные продукты могут предложить качественную работу базы данных, в которую, в свою очередь, заносятся данные о проведении работ или учёту средств измерений. В большинстве случаев, это приложение, которое раскрывает только одну из областей работы метрологических центров и лабораторий. Это катастрофично, так как данные, которые введены в одну базу данных могут не согласовываться с иными документами или базами, которые ведут родственные или зависимые работы и учёты работ сотрудников.

Актуальность темы разработки программного продукта основывается на нехватке качественных средств обработки метрологической информации, включая все работы по поверке, калибровке и ремонту средств измерений. Данная проблема решается путём совмещения всех данных в строгом порядке сортировки и поиска данных с возможностью их всевозможного объединения.

Главная «метрологическая» цель создания программного продукта - содействие обеспечению единства и требуемой точности измерений средствами информационных технологий.

Главная цель дипломного проекта – это полнофункциональная разработка программного обеспечения, которое будет упрощать работу метрологических центров и лабораторий. Данная цель подразумевает достижение и двух косвенных целей:

- разработать базу данных - сервер для метрологического центра;
- написание приложения - клиента для доступа к хранящимся на сервере файлам работников метрологического центра.

Согласно всем целям дипломного проекта можно выделить основные задачи для выполнения:

- физически расположить удалённую базу данных, управляемую посредством СУБД, на сервере;

– написать приложение, располагающееся на рабочих станциях пользователей сети, а также осуществить доступ к данной базе данных при помощи SQL-запросов.

По завершению стадии разработки и тестирования программного продукта, должно получиться полнофункциональное приложение с доступным для каждого пользователя интерфейсом, которое поможет систематизировать работу метрологических центров и повысить качество работы сотрудников в сфере метрологии, стандартизации и сертификации.

1 Метрологическая база данных как программный продукт

Что мы понимаем под «программным продуктом»? Это полностью и всесторонне законченное программное обеспечение, которое облегчает и автоматизирует рабочий процесс пользователей. К автоматизации рабочего процесса, непосредственно, можно отнести внесение новых данных и их обработку в кратчайшие сроки.

В сфере метрологии, стандартизации и сертификации недостаточно программных продуктов, которые качественно обрабатывают всю информацию и хранят большое количество информации в необходимом виде и формате. Данная метрологическая база характеризуется как программа, которая будет помогать структурировать всю документацию согласно ГОСТ РК сотрудникам метрологических центров в работах по поверке, калибровке, ремонту оборудования и учёту средств измерений.

База данных разрабатывается с доступным для каждого пользователя интерфейсом, с помощью которого любой сотрудник может без особого труда найти всю информацию, поменять её при необходимости, преобразовать в нужный формат документации и отправить на подпись руководителю. Таким образом, решается всеобщепризнанная проблема потери данных о каждой проведенной работе в метрологических центрах лабораториях.

В данном случае, метрологическая база данных объединяет в себе сразу несколько баз данных, в которых хранились, в корне разные, виды данных и их форматы. Программа может использоваться в любой сфере, связанной с метрологическими измерениями, приборами и средствами измерений. Данная база будет использоваться для нескольких городов и стран одновременно, храня данные на общем сервере, то есть, базу данных можно назвать распределенной.

В общепризнанном формате, основа информации о метрологических свойствах средств измерений смогут применяться: при проведении расчетов погрешности замеров, погрешности средств измерений и замерных каналов (в этом числе ИИС, АСУТП, операторских и иных систем) для условий эксплуатации, а кроме того при подборе либо сопоставлении средств измерений согласно точности с целью условий эксплуатации. Базы данных, включающие промышленные (в том числе метрологические) свойства средств измерений, могут использоваться при выборе или сопоставлении согласно классу точности (основной погрешности) средств измерений; при этом калькуляция погрешности измерений для условий эксплуатации никак не выполняется. БД, включающие значения основной погрешности конкретных экземпляров средств измерений, имеют все шансы применяться при установлении и исправлении межповерочного промежутка, при моделировании метрологической надежности средств измерений, в задачах оптимизации правильности замеров и уменьшения погрешности посредством внесения поправок в итоги замеров.

1.1 Цели и задачи метрологических баз данных

Если поставить общие цели и задачи к программным продуктам, которые обрабатывают и хранят всю информацию о метрологических центрах (лабораториях) и их работе, то база данных обязана гарантировать:

– полноту данных о метрологических данных средств измерений и других сведений (производитель, список источников данных, госномер согласно реестру общегосударственной регистрации типа средств измерений и т.п.) в соответствии с направлением и сферой применения базы данных:

а) возможность стремительно и легко отыскать требуемые сведения о СИ;

б) возможность просмотра, исправления и дополнения БД;

в) возможность выведения в распечатку нужной инфы;

г) компактность;

д) работу с базой данных в интерактивном порядке;

– при конструировании интерактивных дисплеев следует нацеливаться в неподготовленного юзера. Следовательно, при разработке концепции управления базой данных рационально руководствоваться следующим:

а) действия системы согласно взаимоотношению, к юзеру обязано являться подобным, для того чтобы юзер был должен работать точно предписанным способом;

б) поведение системы обязано являться очевидным юзеру;

в) для любого вероятного затруднения в операции юзера обязана предусматриваться подсказка;

г) количество альтернатив операций юзера при разветвлении разговора обязано быть урезано количеством разделов, любая версия и её результаты обязаны быть отмеченными;

д) для применения системы никак не должны требоваться специализированные умения;

е) должны предусматриваться вероятные отличительные ошибки юзера, в том числе рассеянность и неустойчивость внимательности; в данной взаимосвязи рекомендовано учитывать при появлении погрешностей надлежащие предупреждения юзеру;

ж) СУБД обязана иметь интегрированную концепцию контролирования корректности вводимых данных.

Для достижения данных целей необходимо поставить задачи по совершенствованию и упрощению работы метрологических баз данных, согласно требованиям стандартов и, непосредственно, желанию пользователей (сотрудников) метрологических центров и лабораторий. Часть данных целей реализованы с помощью поставленных задач по систематизированию и структурированию данных в программном продукте.

1.2 Программный продукт и его пользователи

Если начинать писать любое приложение, то стоит задаться вопросом «Для кого создается это приложение?» или «Кто будет работать с данным программным продуктом?».

Пользователи баз данных имеют все шансы быть ненамеренными пользователями, обращающимися к базе данных в той или иной необходимой ситуации, а также могут быть и постоянными пользователями, работающими с информацией каждый день.

Метрологическая база данных будет иметь два вида пользователей:

- сотрудники метрологического центра, которые будут вносить новые данные обрабатывать имеющуюся информацию в базе данных;
- администратор баз данных, который будет иметь полное право на изменение, чтение, сохранение новых данных, а также на удаление или добавление новых составных частей базы данных.

Техподдержкой данного приложения будет заниматься пользователь, так называемый, администратор метрологической базы данных. Данный пользователь будет иметь права: на добавление новых составных частей метрологической базы данных, на добавление новых пользователей и присвоение им новых прав и ролей, а также будет иметь полный доступ ко всем видам деятельности в базе данных.

Следующий вид пользователя - это сотрудники метрологического центра и лаборатории. К сотрудникам метрологического центра относятся: начальник метрологического центра, ведущий инженер метрологической службы города Алматы, инженер метрологической службы города Алматы и других городов Казахстана.

Начальник метрологического центра вид имеет право доступа ко всем сведениям всех городов Казахстана, на добавление новых данных в базу, на составление отчётности по информации, на проверку данных сторонних пользователей - сотрудников метрологического центра.

Сотрудники метрологического центра - это пользователи, которые имеют право на внесение новых данных, на изменение данных, удаление данных только в своем городском подразделении метрологической службы.

1.3 Сравнительный анализ программного продукта с рынком разработок

На сегодняшний день в любой период времени разрабатывается огромное число программных обеспечений, любой из которых можно считать уникальным. Все без исключения программные продукты, занимающие конкретную нишу в области предоставляемого перечня возможностей, имеют схожие элементы, таким образом и то что-то собственное, то что-то неповторимое.

Именно по этой причине у 1 программных продуктов может быть весьма значительная публика признательных пользователей, а прочие продукты, невзирая на их сходство, имеют все шансы таким образом и остаться безуспешной попыткой решения определенных вопросов.

В ходе проектирования каждой концепции в приоритете обязан быть пользователь, который в конечном счете и станет применять микропрограммный продукт.

В данном подразделе будет идти речь о сравнении разработанного продукта «Метрологическая база данных» с другими продуктами, разработанными на данный момент для тех же целей и задач работы метрологических служб.

1.3.1 Программное обеспечение «Метролог» на базе Microsoft Access

Программный продукт «Метролог» нужен для учета средств измерений (СИ), присутствующих в подразделениях фирмы: ведения журналов СИ в электронном виде, развития и осуществлении контроля выполнения графиков периодических поверок и калибровок средств измерения, и указателей. «Метролог» предоставляет возможность хранить информация о паспортных сведениях и нахождении СИ, отслеживать ситуацию различия записей в бизнес-журнале, реализовывать их быстрый подбор в соответствии с выбранными нюансами и получать в печать документа конкретной формы: графики поверки/калибровки/аттестации/ТО, перечни СИ и указателей обновления, длительное обслуживание, промышленное обслуживание, передача и прочие операции. «Метролог» является многопользовательской компьютерной концепцией с утверждённым допуском законных в ней пользователей с помощью районную компьютерную сеть и предоставляет возможность обеспечивать одновременную работу отдельных сотен пользователей.

Группа пользователей	Права
Admins (встроенная)	Любые действия
Операторы	Разрешена работа с данными, но нет возможности менять настройки программы и исправлять журнал операций
Гости	Только просмотр данных

Рисунок 1.3.1 – Таблица групп пользователей и их права

Возможности программного продукта:

- централизация хранения сведений о СИ, а помимо прочего отчетных, ссылочных и нормативных сведений;
- разграничение прав допуска юзеров к данным о СИ;
- разграничение прав допуска к отчетности;

– метрологический учёт СИ: ведение паспортов, присоединение удостоверения СИ согласно стандартам, формирование «главных» средств измерений;

– добавление в поля документа спецсимволов (надстрочные и подстрочные), создание документа с разделами: единые сведения, сведения согласно поверкам/калибровкам, исправлениям и промышленному сервису, подсчет состояния, подсчет ценных металлов в составе, учёт данных согласно поверкам, калибровкам, аттестациям, исправлениям, продолжительном сохранении, промышленном сервисе, списанию, подсчет цены поверки средств измерений.

«Метролог» - это добавок, интеллигентное в области Microsoft Access, прибывающего в состав Microsoft Office Professional. Подобным способом так как, точно также, равно как и набор Microsoft Access, добавок горазд работать приблизительно управления Microsoft Windows 98/2000/XP/7/8. Разделение объективен пользователей осуществляется с использованием встроенной концепции безопасности Microsoft Access. С целью данного имеется 3 категории юзеров, один с каковых считается интегрированным.

В случае потребности, вмонтированными средствами Microsoft Access просто может быть присоединена новейшая категория юзеров с другим комплектом прав либо изменены полномочия с целью имеющейся категории. Кроме того, может быть прибавлено различное число юзеров в различную с встречающихся компаний, посмотрите пункт "Пользователи".

1.3.2 Автоматизированная система расчета погрешности измерений (АСРПИ)

При разработке БД АСРПИ применялась теория словаря сведений, в согласовании с каковой конструктор БД в основе условий юзера является лексикон изображений сведений, к который и направляется сокет. Лексикон изображений сведений предполагает собою отображения полей сведений. Такого рода аспект дает возможность реализовывать понимание почти различных информативных строений в логичном степени. Отбор и развитие сведений выполняется согласно полям и группам полей.

В программном комплексе АСРПИ в должностную БД размещены значимости коэффициентов, применяемых присутствие расчете грех посредственных смыслов с целью промежутков периода усреднения, одинаковых 8 - 24 ч. и 1 мес.

Пользователю предоставляется вероятность в интерактивном порядке подобрать с список пункт БД, с коим некто планирует функционировать. Далее юзер обладает вероятность подобрать с список предмет с целью деятельность: «Деятельность с отображениями», «Деятельность с массивами». Присутствие подборе «Деятельность с отображениями» юзер обладает вероятность подобрать с список порядок деятельность: «Показ и корректирование

имеющегося отображения», «Формирование новейшего отображения», «Устранение имеющегося отображения».

При подборе порядка «Показ и корректирование имеющегося отображения» юзеру в интерактивном порядке предоставляется вероятность просмотра изображений массивов БД, исправления изображений в подобранных им фонах, вставки новейших полей и вытаскивания имеющихся полей.

При подборе порядка «Формирование новейшего отображения» юзер в интерактивном порядке внедряет: абсолютное название массива, число полей в акте, представление любого степь. Число полей в акте точно также числу реквизитов, требуемых с целью отображения массива.

При подборе порядка «Устранение имеющегося отображения» юзеру предоставляется вероятность убрать каждое представление, при этом присутствие данном удаляется подходящий скопление в БД.

Если предметом с целью деятельность подобрана «Деятельность с массивами», в таком случае юзеру предоставляется вероятность просмотра подобранного массива и его исправления согласно полям. Присутствие формирования новейшего отображения образовывается бесполезный скопление, степь коего поочередно наполняются.

Ключом присутствие отыскивании требуемого средства измерения предназначаются первоначальные 3 реквизита: код массива, вид средства измерения, группа правильности.

Шифр массиву присваивается программно в отсутствии тот или иной-или роли юзера.

Если у средства измерения нет класса правильности, в таком случае указывается его главная допустимая неточность, или данное область никак не берется. В отображении каждой категории средств измерений данный атрибут обязательно имеется.

Четвертым реквизитом в отображении компаний считается «Вспомогательная сведения». В случае если в ходе отыскивания средств измерений согласно первоначальным 3 реквизитам и согласно пределам замеров станет обнаружено ряд видов средств измерений, в дисплее монитора вводятся все без исключения обнаруженные виды (модификации) средств измерений с добавочными данными (к примеру, определение градуированной свойства с целью второстепенных устройств, трудящихся с указателями температуры противодействия либо термopарами) и юзеру предоставляется вероятность подобрать с их необходимый вид (форма) средства измерения.

Предусмотрено, то что у 1-го вида средств измерений имеют все шансы быть границы замеров с различными единицами замера.

Для второстепенных устройств и определенных частей замерных каналов данный атрибут в отображении массива способен быть в отсутствии.

Логические массивы базы данных на вещественном уровне презентованы отдельными комплектами сведений. В структуре базы данных вступают 2 закономерных вида комплектов информации - комплекты информации о

средствах измерений и о стандартных схемах замеров. Число комплектов сведений никак не ограничивается. Согласно желанию юзера список данных о СИ способен быть дополненным различным числом реквизитов в отсутствие значительной реорганизации БД.

БД программного ансамбля АСРПИ заключается с 3-х областей: БД о медикаментах замеров, БД о стандартных схемах замеров, должностная БД.

1.4 Обзор и выбор подходящих технологий и средств разработки

В данном подразделе будет проведён анализ описательного и сравнительного характера всех технологий, использованных в дипломном проекте. Будут кратко описаны основные сильные стороны каждого выбранного средства разработки, а также приведены доводы о необходимости использовать именно этот вид того или иного вида технологий.

1.4.1 Объектно-ориентированный язык программирования С#

Главная концепция С# — многофункциональность. Существует возможность осуществить все без исключения в одном стиле и для какой угодно платформы, однако с огромной частью вероятности это станет функционировать слабее, нежели созданное намеренно под определенную платформу.

Преимущества С#:

- в последние дни многое улучшается. Таким образом равно как С# был основан позднее, нежели Java и прочие стили, то было необходимо весьма многое подвергнуть доработке. Кроме того, это могло иметь отношение к популяризации и бесплатности - было обещано раскрыть первоначальный шифр, а приборы (Visual Studio, Xamarin) стали безвозмездными с целью индивидуальных персон и маленьких фирм;

- поддержка майкрософтом. В различии с Java, которой далеко не направился в выгоду переходение во владение Oracle, С# хорошо раскрывается из-за действий Microsoft;

- средний предел вхождения. Синтаксис схожий в С, С++ либо Java упрощает трансформация с целью иных разработчиков программного обеспечения. С целью начинающих данное кроме того единственный с наиболее многообещающих стилей с целью исследования;

- Xamarin. Вследствие приобретения Xamarin в С# сейчас возможно записывать около Android и iOS. Данное, безусловно, огромное преимущество, таким образом равно как их личная подвижная ОС (Windows Phone) никак не приобрела огромной известности;

- добавлено многофункциональное кодирование (на машинном (F#)).

- много синтаксического сахара. Синтаксический сахарный песок - данное подобные системы, какие сформированы с целью облегчения сочинения

и представления программный код (в особенности в случае если данное шифрованное разработчика программного обеспечения) и никак не представляют значимости присутствие компиляции;

Недостатки C#:

- ориентированность, в основном, только лишь в .NET (в Windows платформу);

- бесплатность только лишь с целью маленьких фирмы, обучающихся и разработчиков программного обеспечения-одиночек. С целью крупных установок приобретение лицензий получится дорого. По этой причине в случае если у вас имеется собственная компания, в таком случае будет необходимо расщедриться;

- сохранили диспетчер go to.

Данный язык применяет объектно-направленный подход к программированию в абсолютно всем. Это обозначает, то что тебе необходимо станет характеризовать теоретические системы в базе настоящей сфере, а затем осуществлять меж ними связь. Этот аспект пользуется огромной известностью, вследствие того то что дает возможность никак не удерживать в уме все сведения, а функционировать согласно принципу темного ящика: отдал входные сведения.

1.4.2 LINQ (Language Integrated Query) - язык интегрированных запросов

LINQ — набор расширений стиля, поддерживающий образование запросов сведений с контролированием видов.

LINQ дает 2 типа запросов: местные (с целью местных коллекций) и интерпретируемые (с целью далёких ключей сведений).

Функции стиля, продемонстрированные в основном с целью LINQ, обширно используются в иных участках (you for lambdas).

Выражения запросов мало понятны и слишком применяются. Зачастую обычный требование способа одним словом и легче.

Неизбежные несоответствия среди несоответствием среди интернет – провайдером и импедансом все без исключения ещё содержатся, то что считается целесообразным, однако его следует осознавать.

В SQL постоянно отыщутся определенные предмета, однако никак не в LINQ.

Не осознавая, то что совершается, просто составить весьма малоэффективный шифр.

Трудно сделать запись генпоставщика LINQ. Данное способен являться неминуемо, однако более советов с Microsoft станет оценено.

Это новейший метод мышления о доступе к сведениям с целью многих создателей и понадобится период с целью представления перколяции.

Не однозначно LINQ, однако взаимосвязанный с ним - метод раскрытия способов выявления в C# мало гранулирован.

Некоторые операторы "отсутствуют", в особенности эквиваленты OrderBy с целью предметов, непохожих с упорядочения - к примеру, отбор компонента с наибольшим смыслом качества.

Базовые считанные единицы сведений в LINQ - очередности.

Плюсы:

- декларативный аспект упрощает представление запросов и наиболее малогабаритный;
- расширяемость и деревья формулировок дают возможность в главном координировать требования некоторых ключей;
- даже внутрипроцессные требования имеют все шансы являться выполнены методами, превосходными с LINQ to Objects - к примеру, Синхронный LINQ, и моя личная состав Push LINQ. Весьма гибкий;
- целесообразно с целью in-process запросов, в каком месте легче в целом осознать;
- отлично, для того чтобы исключить SQL в строчках и т.д.

Широкий спектр операторов, предоставляемых согласно умолчанию, и прочие имеют все шансы являться просто добавлены с целью LINQ to Objects.

Отложенное выполнение и потоковая предоставление слабо исследованы (однако улучшаются).

Отладка способна быть весьма непростой сиз-за отсроченного выполнения и потоковой передачи.

В определенных определенных вариантах LINQ способен являться существенно медлительнее, нежели рукодельный шифр. Нежели правильнее вам осознаете внутреннюю службу, этим правильнее вам сумеете данное прогнозировать. (И, безусловно, в случае если эффективность значима с целью вам, у вам обязаны являться надлежащие исследования).

Я замечаю данное правильнее в целом присутствие труде с запросами в ходе. Их просто прогнозировать, осознать и увеличить. Вспомогательные технологические процессы, подобные равно как LINQ to XML и Parallel LINQ, восхитительны. LINQ to Objects возможно применять почти в каждом участке.

LINQ to SQL и т.д. на самом деле неплохи там, в каком месте они уместны, однако их сложнее осознать и иметь необходимость в огромном опыте. Они кроме того применимы только лишь в конкретных сферах вашего программного листинга.

1.4.3 Microsoft Visual Studio

Microsoft Visual Studio - данная программная среда по разработке дополнений в интересах ОС Windows, равно как консольных, таким образом и с графичным дизайном.

В набор вступают последующие ключевые элементы:

- Visual Basic.NET - с целью исследования дополнений в VisualBasic;
- Visual C++ - в классическом стиле C++;

- Visual C# - в стиле C# (Microsoft);
- Visual F# - в F# (Microsoft Developer Division).

Функциональная состав сферы содержит в себе:

- редактор начального программный код, что содержит большое число добавочных функций, равно как автодополнение IntelliSense, рефракторинг программный код и т. д.;
- отладчик программный код;
- редактор конфигураций, рассчитанный с целью упрощённого конструирования графичных интерфейсов;
- веб-техред;
- дизайнер классов;
- дизайнерсхем двор сведений.

Visual Studio кроме того дает возможность формировать и подсоединять посторонние добавления (плагины) с целью расширения функциональности почти в любом степени, в том числе дополнение помощи концепций контролирования версий начального программный код (Subversion и VisualSourceSafe), дополнение новейших комплектов приборов (с целью редактирования и зрительного проектирования программный код в наглядно-направленных стилях программирования либо приборов с целью других нюансов хода исследования программного предоставления).

Коммерческие версии в режиме возрастания стоимости: Visual Studio Professional, Visual Studio Премиум и Visual Studio Ultimate.

Интегрированная среда исследования (IntegratedDevelopmentEnvironment - IDE) Visual Studio дает несколько высокоуровневых многофункциональных способностей, какие уходят из-за граница базисного управления кодом.

Ниже перечислены ключевые достоинства IDE-сферы Visual Studio.

Встроенный интернет-компьютер. С целью сервиса интернет-дополнения ASP.NET нужен Интернет-компьютер, что станет ждать интернет-требования и подвергать обработке надлежащие странички. Присутствие в Visual Studio встроенного Интернет-сервера дает возможность включать интернет-веб-сайт непосредственно с сферы проектирования, а кроме того увеличивает защищенность, за исключением возможность извлечения допуска к тестовому Web-веб-сайту с тот или иной-нибудь наружного ПК, так как пробный компьютер способен осуществлять объединения только лишь с местного ПК.

Поддержка большого количества стилей присутствие исследованию. Visual Studio дает возможность записывать шифр в собственном стиле либо различных иных ценимых стилях, применяя все без исключения период единственный и этот ведь сокет (IDE). Наиболее этого, Visual Studio кроме того ещё дает возможность формировать Интернет-странички в различных стилях, однако размещать их все без исключения в один и в таком случае ведь Интернет-дополнение. Одним-единственным лимитированием считается в таком случае, то что в любой Интернет-страничке возможно применять только лишь тот или иной-в таком случае единственный речь (несомненно, то что в

ином случае трудностей присутствие компиляции существовало б попросту никак не исключить).

Меньше программный код с целью сочинения. С целью формирования многих дополнений необходимо немалое число обычного типичного программный код, и Интернет-странички ASP. NET этому никак не редкий случай. К примеру, дополнение Интернет-компонента управления, включение обработчиков происшествий и корректирование форматирования потребует конструкции в разметке странички строя элементов. В Visual Studio подобные элемента формируются автоматом.

Интуитивный образ кодировки. Согласно умолчанию Visual Studio форматирует шифр согласно грани его ввода, автоматом вклинивая требуемые отступы и используя цветное шифрование с целью отделения компонентов вида объяснений. Подобные небольшие различия совершают шифр наиболее комфортным с целью чтения и меньше подвластным просчетам. Используемые Visual Studio автоматом характеристики форматирования возможно в том числе и адаптировать, то что весьма комфортно в вариантах, если создатель любит иной образ размещения скобок (к примеру, образ K&R, присутствие коем раскрывающая знак располагается в этой ведь строчке, то что и сообщение, который возлюбленная предшествует).

Более значительная темп исследования. Многочисленные с многофункциональных способностей Visual Studio ориентированы в в таком случае, для того чтобы оказать помощь создателю выполнять собственную службу равно как возможно стремительнее. Комфортные функции, наподобие функции IntelliSense (что может захватывать погрешности и представлять верные виды), функции отыскивания и смены (что дает возможность искать основные фразы равно как в 1 файле, таким образом и в абсолютно всем плане) и функции механического прибавления и вытаскивания объяснений (что способен на время утаивать конструкции программный код), дают возможность создателю функционировать стремительно и результативно.

Возможности отладки. Представляемые в Visual Studio приборы отладки считаются лучшим орудием с целью наблюдения таинственных погрешностей и диагностирования необычного действия. Создатель способен осуществлять собственный шифр согласно строчке из-за один раз, определять умственные места прерывания, присутствие стремлении удерживая их с целью применения в перспективе, и в каждое период смотреть нынешнюю данные с памяти.

Visual Studio кроме того обладает и множеством иных функций: право управления планом; интегрированная функция управления начальным кодом; вероятность рефакторизации программный код; сильная форма расширяемости. Наиболее этого, в случае применения Visual Studio 2008 Team System создатель приобретает наращенные способности с целью модульного испытания, коллективной деятельность и управления версиями программный код (то что существенно более этого, то что предполагается в наиболее обычных приборах наподобие Visual SourceSafe).

В качестве нехватки возможно выделить неосуществимость отладчика (Microsoft Visual Studio Debugger) проследивать в коде порядка ядра. Настройка в Windows в порядке ядра в всеобщем случае производится при применении WinDbg, KD либо SoftICE.

1.4.4 TPL (Task Parallel Library) - библиотека для создания многопоточных приложений

Библиотека параллельных задач (TPL) предполагает собою комплект раскрытых видов и API-интерфейсов в местах фамилий System.Threading и System.Threading.Tasks. Задача TPL — увеличение производительности работы создателей из-за результат упрощения операции прибавления параллелизма в дополнения. TPL подвижно масштабирует уровень параллелизма с целью более результативного применения абсолютно всех общедоступных процессоров. Помимо этого, в библиотеке синхронных вопросов исполняется секционирование деятельность, составление плана струй в пуле ThreadPool, помощь отмены, руководство капиталу и производятся прочие низкоуровневые проблемы. Применяя библиотеку синхронных вопросов, возможно увеличить эффективность программный код, сосредоточившись в труде, с целью каковой специализирована проект.

Начиная с .NET Framework 4 библиотека параллельных задач считается преимущественным методом формирования многопоточного и синхронного программный код. Но никак не каждый шифр подойдет с целью параллелизации; к примеру, в случае если оборот из-за любую итерацию осуществляет незначительный размер трудов либо производится с целью незначительного количества итераций, сиз-за добавочной перегрузки, какую параллелизация проявляет в концепцию, шифр способен осуществляться медлительнее. Помимо этого, параллелизация, равно как и каждой трехпоточный шифр, усложняет осуществление проекты. Несмотря на то книгохранилище синхронных вопросов упрощает многопоточные сценарии, рекомендовано обладать базисное представление определений струй, к примеру, блокировки, взаимоблокировки и капиталом гонки, для того чтобы результативно применять библиотеку синхронных вопросов.

1.4.5 Microsoft SQL Server - система управления реляционными базами данных (РСУБД)

Microsoft SQL Server представляет собой СУБД, обеспечивающую создание ин-формационных систем с архитектурой “клиент-сервер”, в которой он играет роль сервера баз данных. SQL Server поддерживают: тиражи-рование данных, параллельную обработку, создание и обработку больших баз дан-ных, отличается простотой управления и ис-пользования.

Эта СУБД обеспечивает высокую степень защиты данных, как от случайных потерь, так и от несанкционированного доступа, обладает

развитыми средствами обработки данных и хорошим быстродействием. SQL-Server предназначен для хранения большого объема данных.

SQL Server добавляет к сетевым компонентам специальные сервисы, та-кие как OLE DB (Object Linking and Embedding-Database - связывание и внедрение объектов базы данных) и ODBC (Open Database Connectivity - совместимость от-крытых баз данных). С их помощью обеспечивается совместимость различных кли-ентских приложений при работе с сервером.

Пользователь компьютера-клиента с помощью сетевых средств своей операционной системы может устанавливать связь с компьютером-сервером, где установлен SQL Server. На компьютерах-клиентах размещаются локаль-ные базы данных, работа с которыми ведется с помощью персональных СУБД (Access, Visual FoxPro) или языков программирования (Visual Basic, Delphi, C++ Builder, Visual C++). С их помощью через ODBC осу-ществляется доступ к базам дан-ных, размещенным на сервере.

В SQL Server используется понятие роли(role). Каждому пользователю может быть назначено произ-вольное число ролей. Например, пользователю может быть назначена роль Администратора.

SQL Server имеет следующие варианты поставки:

- Enterprise (предприятий) работает под управлением операционной системы Windows NT Server Enterprise Edition 4.0;
- Standard (стандартный) может работать под управлением Windows NT Server Enterprise Edition 4.0;
- Desktop(настольный) работает под управлением Windows NT Workstation 4.0 и Windows 9x.

Является реляционной (Реляционная база данных представляет собой набор взаимосвязанных двухмерных таблиц (отношений)), полнофункциональной. Используется для создания распределенных баз данных коллективного использования. Реализует многопользовательский удаленный режим типа Клиент- Сервер доступа к данным. Имеет развитые средства реляционного доступа к данным (Transact-SQL, PL/SQL), администрирования, защиты и восстановления базы данных.

1.4.6 SQL - универсальный язык для работы с базами данных

Невзирая на присутствие диалектов и отличий в синтаксисе, в большинстве своём скрипты SQL-запросов, включающие DDL и DML, имеют все шансы являться довольно просто вынесены с одной СУБД в иную. Имеются концепции, создатели каких первоначально разбирались в использование согласно наименьшей грани некоторых СУБД. Безусловно, то что присутствие использовании определенных специфичных с целью осуществлении способностей такого рода переносимости достичь ранее весьма сложно.

Наличие стереотипов и комплекта исследований с целью раскрытия сопоставимости и соотношения определенной осуществлении SQL общепризнанному эталону только лишь содействует «стабилизации» стиля.

Хотя, нужно сосредоточить интерес, то что непосредственно согласно себя образец зонами слишком формализован и разнесен в объемах.

С поддержкой SQL разработчик программного обеспечения показывает только лишь в таком случае, которые сведения необходимо достать либо изменить. В таком случае, каковым способом данное совершить, принимать решение СУБД напрямую присутствию обрабатыванию SQL-запроса. Но никак не нужно мыслить, то что данное целиком многофункциональный правило - разработчик программного обеспечения показывает комплект сведений с целью подборки либо изменения, но ему присутствию данном целесообразно продемонстрировать, равно как СУБД станет рассматривать документ его запроса. Нежели труднее построен требование, этим более некто позволяет альтернатив сочинения, разных согласно быстроты исполнения, однако схожих согласно окончательному комплекту сведений.

Создатели реляционной модификации сведений страж города кодд, Кристофер Дейт и их приверженцы свидетельствуют в в таком случае, то что SQL никак не считается подлинно реляционным стилем. В частности, они свидетельствуют в последующие трудности SQL: повторяющиеся строчки, неопределённые значимости, очевидное обозначение режима хищник по левую сторону вправо, колонки в отсутствии фамилии и дублирующиеся фамилии хищник, недостаток помощи качества «=», применение указателей, значительная избыточность

В опубликованном Кристофером Дейтом и Хью Дарвенем 3-ем Манифесте они объясняют основы СУБД последующего поколения и дают речь Tutorial D, что считается поистине реляционным.

Хотя SQL и думал равно как способ деятельность окончательного юзера, в окончании точек некто начал до такой степени трудным, то что преобразовался в механизм разработчика программного обеспечения.

Несмотря в присутствии интернационального эталона ANSI SQL-92, многочисленные фирмы, специализирующиеся исследованием СУБД, записывают переменные в речь SQL, используемый в разрабатываемой СУБД, этим наиболее отходя с эталона. Подобным способом, возникают специфичные с целью, любой определенной СУБД диалекты стиля SQL.

Ранее диалекты SQL основной массы СУБД никак не давали метода манипуляции древовидными текстурами. Определенные поставщики СУБД предложили собственные ответы. В наше время период в ANSI стандартизована рекурсивная система WITH с диалекта SQL DB2. В MS SQL Server рекурсивные требования возникли только в версии MS SQL Server 2005.

2 Проектирование программного продукта

Данная глава представляет собой описание проектирования базы данных. Также раскроются понятия структура ПО и функциональной структуры приложения. Важно отметить, почему мы выбираем именно этот тип базы данных и каким образом этот выбор сможет повлиять на дальнейшее развитие приложения. В связи с этим необходимо предусмотреть все преимущества и недостатки выбранного типа базы данных, чтобы в дальнейшем учесть их при проектировании

Также описание изучения предметной области: какие таблицы и сущности должны присутствовать в базе данных для корректной работы приложения.

База данных должна содержать в себе четкие, упорядоченные таблицы, данные в которые разделены по группам и связаны между собой.

2.1 Функционирование программного продукта

Структура программного обеспечения включает в себя любое программное обеспечение, которое было использовано при создании определенного приложения или пакета приложений.

Все программное обеспечение, используемое в создании ПО, можно разделить на 3 группы:

- системное ПО: программное обеспечение общего пользования. Сюда относятся операционные системы и их оболочки, различные специфические драйверы и утилиты, системы, которые предназначены для технического обслуживания машины разработчика;

- прикладное ПО: сюда относятся различные прикладные программы и пакеты этих программ. Например, это программы для работы с текстовой информацией, её оформлением, программы для просмотра веб-страниц или веб-браузеры и т. д.;

- инструментальное ПО: также называют системами программирования. Это инструменты, которые разработчик приложения использует для просмотра исходного кода программы, его редактирования и дальнейшего запуска. В настоящее время такие системы чаще всего называют интегрированными средами разработки, поскольку они включают в себя большой перечень инструментов для работы с различными языками программирования и СУБД.

Также существует 4 группа ПО, которую редко упоминают, — это базовое ПО. Базовым является то ПО, которое встроено в аппаратное обеспечение компьютера. Чаще всего к данной категории относят базовую систему ввода-вывода (BIOS).

Исходя из схемы на рисунке 2.1.1, видно, что разработка приложения происходила под управлением операционной системы Windows 10 Pro Edition.

Для оформления отчета по проделанной работе был использован пакет программ Microsoft Office 2013 Pro и веб-приложение draw.io, которое использовалось для построения структурных и других схем.

Разработка программного продукта велась в программах Microsoft Visual Studio 2017 и СУБД Microsoft SQL Server 2016.

Далее приведена схема структуры разрабатываемого приложения:

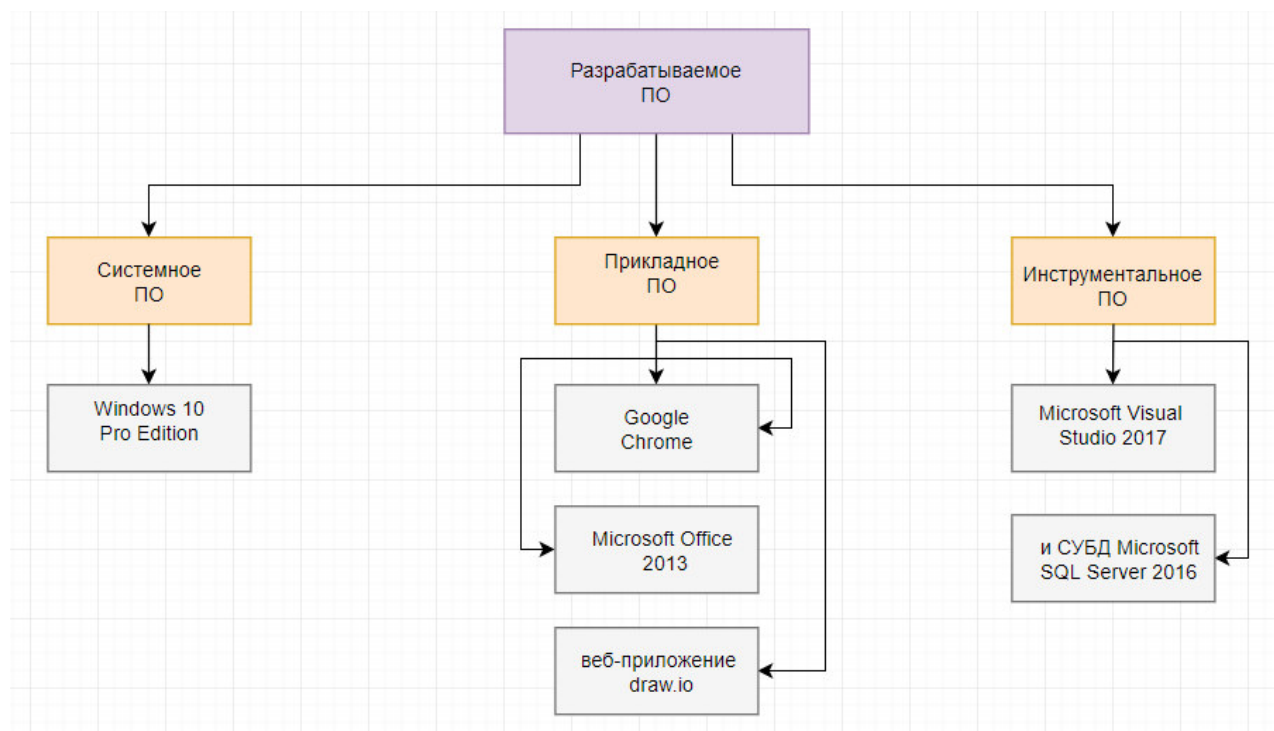


Рисунок 2.1.1 – Структура ПО

Такие продукты, как интегрированные среды разработки, позволяют сделать разработку программного обеспечения более простой и наглядной. Также они влияют на скорость разработки программного обеспечения и более простую реализацию каких-либо сложных моментов.

2.2 Структурирование программного продукта

Функциональная структура веб-приложения предназначена для того, чтобы наглядно показать какие функции может предоставить приложение. Глядя на такую структуру, сразу становится ясным стоит ли использовать это приложение и приведет ли оно к желаемому результату.

Приведенная далее структурная схема показывает, какие конкретно функции будут предоставляться разработанным программным продуктом. У любой структурной схемы веб-приложения корневым элементом схемы является либо главная страница, либо форма входа в аккаунт пользователя.

В данном случае главным корневым элементом схемы является форма входа, которая предлагает пользователю ввести свой логин и пароль для входа

в систему. В том случае, если сотрудник является новым человеком для системы, то эта же форма предлагает ему перейти в форму регистрации. Таким образом, человек знает, куда направится в том случае, если вход в систему ему выполнить нужно, но он не имеет необходимых данных для этого.

Регистрация предоставляется с целью уникальной идентификации каждого сотрудника системы. Каждый из сотрудников имеет собственный аккаунт, за которым сохраняется не только его логин и пароль, но и вся информация о поставленных задачах и беседах пользователя. Таким образом, данные пользователя, которые он вводит в систему, защищены от посторонних глаз и большое количество пользователей может полноценно пользоваться данной системой.

Далее начинается процесс, который открывает пользователю к аккаунту. Этот процесс состоит из трех частей: идентификации, аутентификации и авторизации.

Процесс идентификации представляет собой сопоставление распознавания пользователя по его идентификатору. В данном случае идентификатором пользователя является его номер телефона, который в принципе является уникальным набором цифр для каждого человека. Получая от пользователя в поле какое-либо значение, программа сравнивает, есть ли в системе такой зарегистрированный пользователь. Если наличие пользователя подтверждается, то программа переходит к процессу аутентификации.

Процесс аутентификации представляет собой процедуру проверки подлинности пользователя. Другими словами, пользователю необходимо доказать, что он именно тот человек, которому принадлежит идентификатор. В данном приложении в качестве проверяемого значения был использован пароль. Программа сравнивает пароль, введенный пользователем и пароль, хранящийся в базе данных, и в случае совпадения осуществляет процедура авторизации.

Авторизация – это предоставление доступа к какому-либо ресурсу, в данном случае к аккаунту пользователя. Этот процесс осуществляется после идентификации и аутентификации, которые предшествуют его выполнению.

Далее, пользователь, попадая в систему может выбрать сферу деятельности для работы: учёт средств измерений в СП/СЕ в ОДС, графики и планы, учёт работ, отчёты.

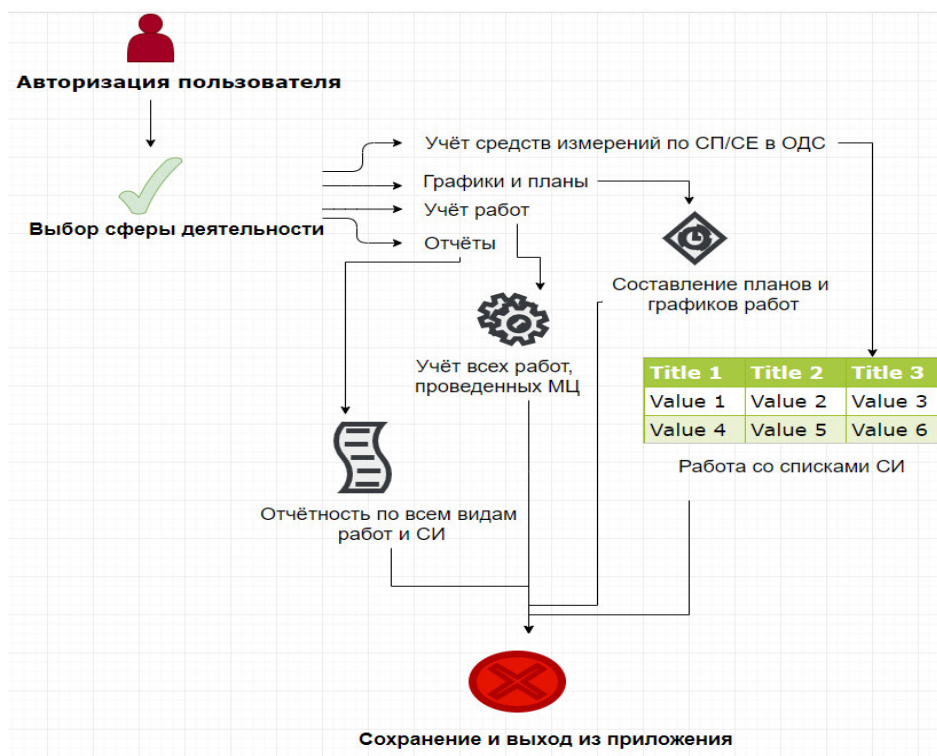


Рисунок 2.2.1 – Функциональная структура приложения

Направляясь на форму со списками учёта средств измерений, пользователь может заняться заполнением списков учёта средств измерений. Там предусмотрен ряд функциональных возможностей:

- пользователь может добавлять новые данные в таблицу данных;
- пользователь может удалить данные: для этого ему нужно на иконку минуса рядом с текстом поля табличного пространства. Данная функция служит для удаления неправильно описанных данных, либо для информации, которая больше неактуальна;
- пользователь может просмотреть информацию по спискам средств измерений за определенную дату: для этого необходимо выбрать данные, списка, который интересует, и нажать на кнопку «Показать».

В том случае, если пользователь решит начать свою работу с чата, то он перейдет на страницу с чатом. Там ему представится свой выбор действий: он либо сам напишет сообщение какому-либо из своих коллег, либо начнет читать сообщения, пришедшие ему, в том случае, если таковые имеются. Список коллег расположен слева от основного окна с сообщениями. Внизу располагается текстовое поле для ввода сообщения и кнопка для отправки сообщения.

Таким образом, функциональная структура приложения, представленная в виде схемы, поясняет все возможности, которые пользователь имеет в системе. Подобные структуры наглядно показывают все возможности разрабатываемого продукта.

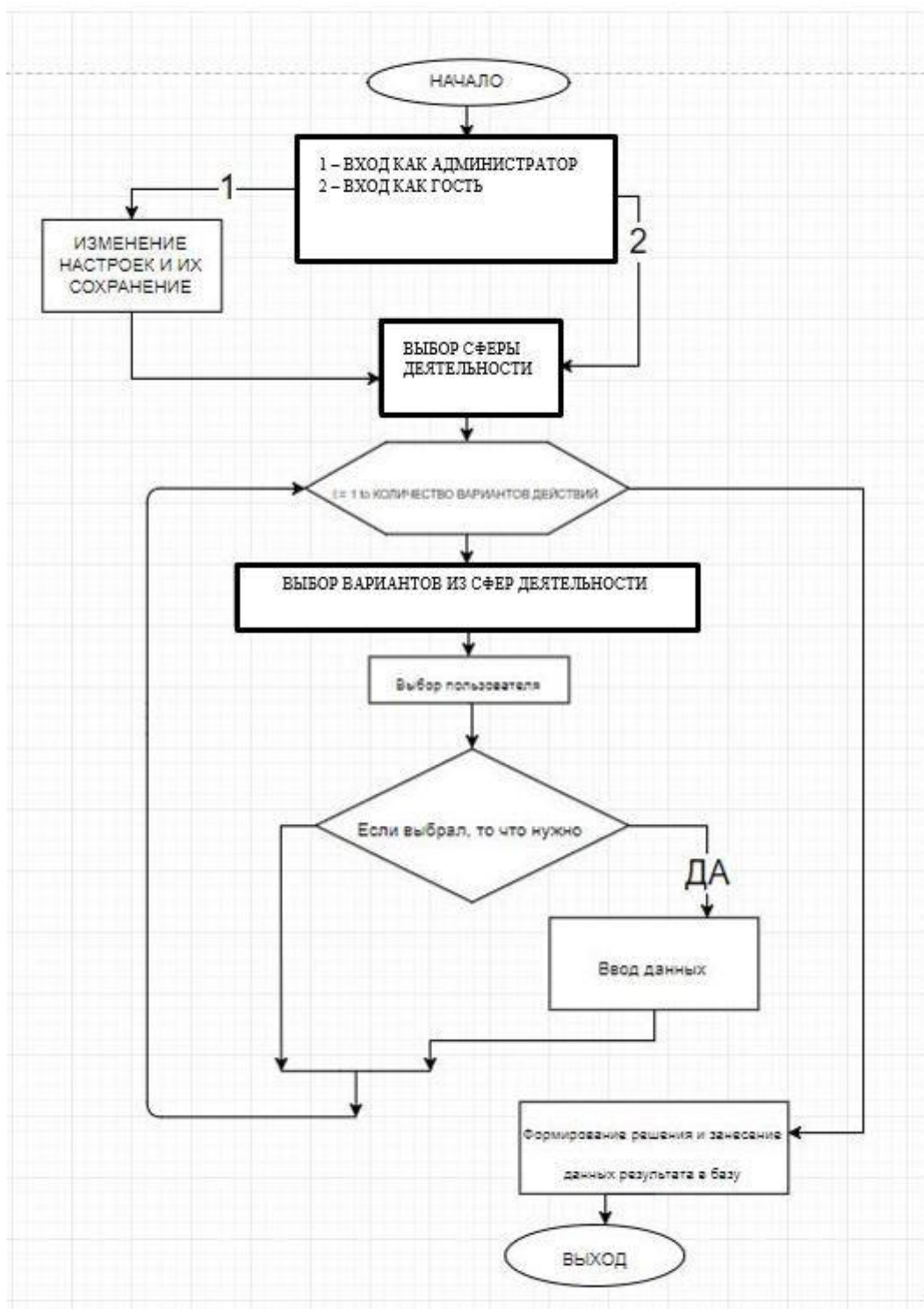


Рисунок 2.2.2 – Общая блок-схема ПО

Блок-схема – это понимание метода в графичной фигуре. Все без исключения указания и воздействия презентованы геометрическими фигурами (блоками). Изнутри любой формы проставляется все сведения о этих поступках, какие необходимо осуществить. Взаимосвязи представлены в варианте простых направлений со стрелками (присутствие потребности). С целью дизайна блок-схем алгоритмов существует ГОСТ 19.701-90. Он

показывает последовательность и принципы формирования их в графической фигуре, а вдобавок ключевые способы постановления. В данной главе приведены ключевые компоненты блок-схем, какие применяются при постановлении вопросов, к примеру, согласно информатике. А сейчас проанализируем принципы концепции.

Из ключевых законов формирования блок-схемы возможно отметить таковые характерные черты, какие обязаны быть у каждой блок-схемы:

- обязательно обязано находиться 2 блока – «Основание» и «Окончание». К тому же в отдельном экземпляре;
- от первоначального блока вплоть до окончательного обязаны быть проложены направления взаимосвязи;
- из абсолютно всех конструкций, помимо окончательного, обязаны выступать направления струи;
- обязательно обязана находиться нумерование абсолютно всех конструкций: поверх книзу, по левую сторону вправо;
- порядковый госномер необходимо проставлять в левом верхнем углу, совершая несоответствие начертания;
- все конструкции обязаны являться объединены товарищ с ином направлениями. Непосредственно они обязаны устанавливать очередность, с каковой производятся воздействия;
- если течение перемещается внизу-наверх либо с правой стороны влево (иными текстами, в противоположном режиме), в таком случае непременно рисуются стрелки;
- линии разделяются в происходящие и вступающие. Присутствие данном необходимо выделить, то что один направление считается с целью 1-го блока сходящей, а с целью иного поступающей;
- от первоначального блока в схеме направление струи только лишь вылезает, таким образом равно как некто считается наиболее первоначальным. А смотри у окончательного блока существует только лишь доступ;
- чтобы легче существовало прочитывать блок-схемы, вступающие направления представляются поверх, а идущие внизу;
- допускается присутствие разрывов в направлениях струи. Непременно они помечаются особыми соединителями.

Разработка блок-схем производится на стадиях проектирования и документирования, в соответствии с последовательностью модификации исследования, которая в настоящее время практически никак не используется, т.к. сопутствуется крупными рисками, сопряженными с погрешностями в стадиях проектирования.

На рисунке 2.2.2 изображена общая блок-схема ПО. Она наглядно описывает основные действия работы программы, начиная со входа со стороны администратора и пользователя, и, заканчивая, работой базы данных и внесением в неё результатов вычислительных действий ПО.

2.3 Разработка модели базы данных

В большинстве разрабатываемых приложений должна существовать база данных, в которой будет храниться информация, необходимая для корректной работы приложения и для его взаимодействия с пользователем.

Предполагается, что для базы данных всегда должен быть отведен достаточно большой объем памяти компьютера (или сервера). Поскольку одновременно база данных должна иметь доступ к огромнейшему количеству информации.

Проектирование базы данных — это всегда достаточно сложный процесс, который требует не только внимательности разработчика, но и проявления его умственных способностей. От того насколько точно будет продумана логика приложения, а затем и перенесена в таблицы базы данных, зависит насколько просто потом будет разработчику получать данные из этой таблицы.

При разработке предлагаемого приложения возникла необходимость создать свою базу данных, поскольку пользователю необходимо хранить данные, загружаемые в приложение и после его выхода из аккаунта, и отключения браузера. База данных предназначена для того, чтобы хранить в ней все данные пользователя, его переписку с другими пользователями, задачи пользователя вместе с их статусами, которые определяют степень выполнения задачи в данный момент.

Сначала может показаться, что нужно разрабатывать сложную базу данных, с большим количеством таблиц и связей в ней. Но на самом деле при правильном подходе к построению таблиц и продумывании для чего нам нужны те или данные, выясняется, что данные можно хранить в более структурированном виде.

Именно этот структурированный вид и логическая продуманность базы позволяет ускорить процесс выполнения запросов в базу данных. А также иметь доступ к конкретному сегменту СУБД, а не искать нужную запись в хаотичном потоке информации, как это было бы, если бы данные хранились в обычном файле.

Также необходимо хранить в базе данных лишь те значения, которые действительно нужны для работы с приложением. Лишние значения, которые в дальнейшем в базе данных не используются, лишь нагружают ее и не дают в полной мере ощутить преимущества скорости работы современных СУБД. К тому же все СУБД обладают функцией, позволяющей нам в любой момент времени добавить нужные поля в любую из таблиц базы данных. Так что данное свойство позволяет разработчику не создавать поля в структуру таблицы впрок.

Также свойство транзакционности при пользовании базами данных позволяет избежать утечки данных из базы данных. Все запросы, отправляемые в базу данных, либо будут выполнены, либо нет, но данные при этом не исчезнут никуда, как это могло бы произойти, если бы для хранения данных приложения использовались обычные текстовые файлы.

В современном проектировании баз данных существует две основные технологии, согласно которым проектируется большинство актуальных на сегодняшний день баз данных: SQL и NoSQL, реляционные и нереляционные базы данных.

Различия этих технологий заключаются в том, какой принцип используется в каждой технологии для проектирования базы данных, какие типы информации поддерживают, как хранят информацию и разница в функционале, предоставляемом базами данных.

Реляционные базы отличаются тем, что хранят в себе описание каких-либо конкретных объектов. Это могут сведения о человеке, сотруднике, описание содержимого товарной корзины в интернет-магазине и т. д. Все эти данные хранятся в таблицах, сгруппированных по типу объектов. Формат таблиц для хранения объектов обычно задается на этапе проектирования базы данных.

Нереляционные базы данных работают по иному принципу. То, что в реляционных базах данных разбито на взаимосвязи нескольких таблиц, в нереляционной базе данных те же самые значения могут храниться в виде целостной самостоятельной сущности.

Каждая из множества систем управления базами данных имеет собственное внутренне устройство, которое влияет на методы работы с ней. Например, нереляционные базы данных лучше поддаются процессу масштабирования, в то время как реляционные базы данных больше подходят для хранения уже структурированной информации.

Конечный выбор той или другой системы управления зависит от того, какие особенности имеет проект, для которого, собственно, и проектируется хранилище данных. На данный момент в большинстве крупных проектов одновременно используют оба типа систем управления базами данных.

Для приложения, разрабатываемого в рамках дипломного проекта, будет достаточно использования только реляционной базы данных, поскольку все данные, которыми оперируют компоненты приложения могут быть преобразованы в несколько взаимосвязанных таблиц. В данном случае структура базы данных составляется на этапе проектирования базы данных и мало подвержена дальнейшим изменениям. Поскольку приложения предоставляет конкретный неизменяющийся со временем функционал, то нет необходимости вольно обращаться с потоком предоставляемой информации и все время подстраивать его под определенные моменты.

Система управления базами данных — это совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных.

СУБД — это комплекс программ, позволяющих создать базу данных и манипулировать данными (вставлять, обновлять, удалять и выбирать). Система обеспечивает безопасность, надёжность хранения и целостность данных, а также предоставляет средства для администрирования БД.

Для проектирования базы данных в дипломном проекте была выбрана СУБД SQL Server. Эта СУБД обеспечивает высокую степень защиты данных, как от случайных потерь, так и от несанкционированного доступа, обладает развитыми средствами обработки данных и хорошим быстродействием. SQL-Server предназначен для хранения большого объема данных.

SQL Server добавляет к сетевым компонентам специальные сервисы, та-кие как OLE DB (Object Linking and Embedding-Database - связывание и внедрение объектов базы данных) и ODBC (Open Database Connectivity - совместимость от-крытых баз данных). С их помощью обеспечивается совместимость различных клиентских приложений при работе с сервером.

В процессе выбора реляционной СУБД для работы с проектом уделялось внимание лицензии, которая предоставлялась разработчиками. PostgreSQL имеет свободную лицензию, что позволяет использовать данную СУБД для любых необходимых целей, а также позволяет видоизменять функционал приложения опираясь на свои нужды. К тому же из всех известных СУБД PostgreSQL является самой развитой системой управления. Также свободная лицензия – это большое преимущество перед другими, платными, СУБД, потому что стоимость лицензии других популярных СУБД может достигать до нескольких сотен тысяч долларов, что просто неприемлемо для небольших компаний.

Данная СУБД поддерживает большую часть официального стандарта SQL.

По мимо этого здесь реализован ряд важных современных функций для работы с базами данных: внешние ключи, сложные запросы, изменяемые представления, триггеры и т. п.

По истине уникальными функциями в работе с PostgreSQL является возможность создавать собственные элементы для работы с данными. Например, можно создать собственную функцию, оператор, агрегатные функции, методы индексирования, и даже собственные типы данных.

PostgreSQL имеет клиент-серверную архитектуру. Таким образом, есть главный серверный процесс, именуемый postgres. Он принимает запросы от клиента и оперирует с файлами базы данных. Этот серверный процесс может работать как локально на клиентской машине, так и находясь на сервере. Одновременно серверный процесс может принимать подключения сразу от нескольких клиентов и работать с ними. Он все время находится в ожидании подключения клиентов.

В качестве клиента используется специальное клиентское приложение, которое имеет возможность выполнять запросы в базе данных. Клиентские приложения очень разнообразны: различные текстовые утилиты, графические приложения, либо другой специализированный инструмент для работы с базой данных.

Использование подобной СУБД является мощным заделом к будущему развитию программного продукта. Гораздо проще сразу создать хранилище данных с использованием современного инструмента, который в будущем

сможет удовлетворить все необходимые потребности, нежели переводить проект с одной СУБД на другую.

Перевод проекта на новую СУБД процесс сложный и дорогой. Может возникнуть проблема утечки информации или конфликта типов данных, что усложнит этот процесс. К тому же СУБД PostgreSQL активно поддерживается, дорабатывается, совершенствуется и обновляется по сегодняшний день.

После выбора СУБД можно переходить к анализу предметной области и проектированию схемы базы данных.

Анализ предметной области представляет собой один из наиболее важных этапов в разработке приложения. Этот этап позволяет выявить все данные, которые потребуются для корректного взаимодействия пользователя с приложением, а также учтут какие данные пользователь захочет хранить в базе данных.

Для того, чтобы не забыть добавить данные из какой-либо области приложения, необходимо поэтапно переходить к каждому функционалу приложения от начала взаимодействия с пользователем и анализировать, какие данные окажутся востребованными для корректной реализации данного функционала.

Первое взаимодействие с системой: вход в приложение.

Для того, чтобы пользователь мог осуществить вход в систему ему необходимо иметь собственный идентификатор и пароль. В качестве идентификатора может выступать какой-либо уникальный номер, имя пользователя, или, например, номер телефона. В данном случае мы будем использовать номер телефона, потому что он уникален для каждого пользователя, а также потому, что при помощи вывода этого поля рядом с именем пользователя можно предоставить номер пользователя, который будет использован для срочной связи служащих посредством мобильного телефона.

Пароль – какой-либо набор символов, придуманный пользователем. Единственное особое условие этого поля состоит в том, что пароль пользователя хранится в базе данных в зашифрованном виде, и недоступен для просмотра системному администратору команды [12].

В том случае, если пользователь не зарегистрирован, то от формы входа он направится к форме регистрации. Здесь уже будут использоваться два ранее рассмотренных свойства: номер телефона, который служит идентификатором пользователя и пароль от аккаунта пользователя. Также здесь необходимо будет добавить поля имени и фамилии пользователя, для удобства поиска необходимого сотрудника, его должность.

В итоге подошел момент к созданию первой таблицы базы данных с названием «встроенные_средства». Эта таблица будет хранить в себе все данные относительно каждого средства измерения, относящегося к категории «Встроенных средств измерений» в полях, названия которых были перечислены выше: номер, наименование цеха, наименование и тип аппаратуры, наименование и тип СИ, заводской номер, пределы измерений, статус, мат.ответственные люди и т.д.

Запрос на создание таблицы в базе данных выглядит следующим образом:

```
CREATE TABLE ВСТРОЕННЫЕ_СРЕДСТВА
(
    NUM_VS bigint NOT NULL,
    СЕХ_NAME_VS varchar(100),
    TYPE_APPARATURE_VS varchar(250),
    TYPE_BLOK_VS varchar(200),
    TYPE_SI_VS varchar(150),
    ZAVOD_NUM_VS varchar(50),
    PREDELI_VS varchar(250),
    SOSTOYANIE_VS varchar(100),
    INV_NUM_VS VARCHAR(1000),
    FIO_VS varchar (50),
    MESTO_VS varchar (100)
);

alter table ВСТРОЕННЫЕ_СРЕДСТВА add primary key ([№])

SP_RENAME 'ВСТРОЕННЫЕ_СРЕДСТВА.NUM_VS', '№'
SP_RENAME 'ВСТРОЕННЫЕ_СРЕДСТВА.СЕХ_NAME_VS', 'Наименование цеха'
SP_RENAME 'ВСТРОЕННЫЕ_СРЕДСТВА.TYPE_APPARATURE_VS', 'Наименование, тип аппаратуры'
SP_RENAME 'ВСТРОЕННЫЕ_СРЕДСТВА.TYPE_BLOK_VS', 'Наименование, тип блока, №'
SP_RENAME 'ВСТРОЕННЫЕ_СРЕДСТВА.TYPE_SI_VS', 'Наименование, тип СИ'
SP_RENAME 'ВСТРОЕННЫЕ_СРЕДСТВА.ZAVOD_NUM_VS', 'Заводской номер или присвоенный номер'
SP_RENAME 'ВСТРОЕННЫЕ_СРЕДСТВА.PREDELI_VS', 'Пределы измерений'

SP_RENAME 'ВСТРОЕННЫЕ_СРЕДСТВА.SOSTOYANIE_VS', 'Статус'
SP_RENAME 'ВСТРОЕННЫЕ_СРЕДСТВА.INV_NUM_VS', 'Номер О.С. или номер забаланса'
SP_RENAME 'ВСТРОЕННЫЕ_СРЕДСТВА.FIO_VS', 'Ф.И.О. мат. ответственного лица'
SP_RENAME 'ВСТРОЕННЫЕ_СРЕДСТВА.MESTO_VS', 'Место проведения поверки'
```

Рисунок 2.3.1 – Запрос для создания таблицы «встроенные_средства»

Результат запроса на все данные из таблицы, которая уже заполнена тестовыми значениями выглядит следующим образом.

№	Наименование цеха	Наименование, тип аппаратуры	Наименовани...	Наименование, тип СИ	Заводской номер или приса...	Пределы измерений	Статус	Номер О.С. или номер забаланса	Ф.И.О. мат. ответственного лица	Место проведения поверки
1	МЦК-1 ЦБСКТ	Система измерения длительности измерения	-/-	NGN 9ESS	3203803	3600 с, 600 с, 200 с, 100 с, 1 с	поверка	130013864445	Агапова Г.В.	РДТ "Амгальтелком"
2	МЦК-1 ЦБСКТ	Система временного учета соединений ГУТС	-/-	NGN CS 2000	13303646	3600 с, 600 с, 200 с, 100 с, 1 с	поверка	130013866208	Агапова Г.В.	РДТ "Амгальтелком"
3	RRRR	Первичный эталонный генератор Time Ceium	-/-	Time Ceium	0519005117	5, 10 МГц	поверка	130014687109	Пантошин А.А.	ЮФ РПТ "КалМетр"
4	ОКСТ и УП	Первичный эталонный источник Time Ceium	-/-	Time Ceium 4400	1702010892	5, 10 МГц	поверка	130014726206	Пантошин А.А.	ЮФ РПТ "КалМетр"

Рисунок 2.3.2 – Данные таблицы «ВСТРОЕННЫЕ_СРЕДСТВА»

После входа в систему пользователь может выбрать каким инструментом ему воспользоваться: чатом или планировщиком.

Для того, чтобы в базе данных могли храниться все сообщения из всех переписок пользователей, необходимо, чтобы каждое сообщение хранило о себе следующие данные: отправитель сообщения, получатель сообщения, сам текст сообщения, идентификатор сообщения, а также время отправки сообщения. Далее рассматривается каждое из полей и его пригодность.

Поле, которое хранит в себе идентификатор отправителя сообщения хранится для того, чтобы у получателя корректно отображалось от кого пришло данное сообщение, а также при распределении сообщений между чатами будет сразу понятно, чату с каким пользователем принадлежит данное сообщение.

Поле, хранящее в себе идентификатор получателя сообщения, предназначено для того, чтобы сообщение пришло именно к конкретному получателю в конкретный чат с пользователем, а не какому-либо случайному пользователю, для которого это сообщение вообще не предназначалось.

Поле с текстом сообщения хранит в себе само сообщение, которое пересылается от пользователя к пользователю.

Идентификатор сообщения необходим для того, чтобы каждое сообщение было уникальным, и разработчик при необходимости мог с легкостью добраться для каждого из сообщений. Это поле пригодится для дальнейшего совершенствования программы, когда появится возможность пересылать сообщения между пользователями или вовсе удалять их.

Время отправки сообщения фиксируется для пользователя. Например, если пользователю пришло сообщение, которое он не прочитал сразу, то возможно к тому моменту, когда он его прочитает оно уже потеряет актуальность и пользователь будет знать об этом. Время отправки сообщения – это поле, которое на самом деле несет в себе много пользы для использования чата.

В итоге будет создана вторая таблица базы данных с названием «User_Activity_Login». Эта таблица будет хранить в себе все данные относительно каждого сообщения в полях, названия которых были перечислены выше: отправитель сообщения, получатель сообщения, текст сообщения, идентификатор сообщения, время отправки сообщения.

Запрос на создание таблицы в базе данных выглядит следующим образом:

```

:CREATE TABLE [dbo].[User_Activity_Log](
    [UAL_User_Id] [varchar](75) NOT NULL,
    [UAL_Employee_No] [varchar](75) NOT NULL,
    [UAL_Timestamp] [datetime] NOT NULL,
    [UAL_Function_Performed] [nvarchar](100) NOT NULL,
    [UAL_Other_Information] [nvarchar](100) NULL,
    [UAL_IP_Address] [nvarchar](25) NOT NULL,
    PRIMARY KEY CLUSTERED
    (
        [UAL_User_Id] ASC,
        [UAL_Timestamp] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO

:SET ANSI_PADDING OFF

:CREATE TABLE [dbo].[Login](
    [LON_User_Name] [varchar](75) NULL,
    [LON_Login_Name] [varchar](75) NULL,
    [LON_Employee_No] [varchar](20) NULL,
    [LON_Login_Password] [varchar](20) NULL,
    [LON_Type] [nvarchar](20) NULL,
    [LON_Status] [varchar](20) NULL
) ON [PRIMARY]

GO

:SET ANSI_PADDING OFF

```

Рисунок 2.3.3 – Запрос для создания таблицы логинов для авторизации пользователей

Далее осталось создание нескольких хранимых процедур, которые будут хранить в себе результаты работы с базой данных, составлять отчётность по определенным спискам и формировать справочный материал из существующих данных.

Результат запроса на все данные из таблицы, которая уже заполнена тестовыми значениями выглядит следующим образом:

	UAL_User_Id	UAL_Employee_No	UAL_Timestamp	UAL_Function_Performed	UAL_Other_Information	UAL_IP_Address
1			2019-05-05 16:32:33.480	Переход в списки		192.168.1.140
2			2019-05-05 16:33:39.180	Переход в графики и планы		192.168.1.140
3			2019-05-05 16:46:50.177	Переход в списки		192.168.1.140
4			2019-05-05 16:47:01.580	Переход в отчеты		192.168.1.140
5			2019-05-05 16:48:00.120	Переход в отчеты		192.168.1.140
6			2019-05-05 16:51:26.490	Переход в отчеты		192.168.1.140
7			2019-05-05 16:53:18.313	Переход в отчеты		192.168.1.140
8			2019-05-05 16:54:54.300	Переход в отчеты		192.168.1.140
9			2019-05-06 11:08:37.073	Переход в списки		127.0.0.1
10			2019-05-06 11:09:20.070	Переход в графики и планы		127.0.0.1
11			2019-05-06 11:09:26.177	Переход в отчеты		127.0.0.1
12	admin	0	2018-08-22 12:39:48.213	Login		::1
13	admin	0	2018-08-22 12:39:53.430	Login		::1
14	admin	0	2018-08-22 14:23:56.370	Login		::1
15	admin	0	2018-08-22 14:24:01.537	Login		::1
16	admin	1	2018-08-22 14:25:31.103	Login		::1

Рисунок 2.3.4 – Данные таблицы логинов для авторизации пользователей

Каждая информация, добавляемая в хранилище базы данных, имеет своего пользователя, который создал эту задачу. Таким образом, можно выбирать только те данные из базы, которые принадлежат конкретному пользователю и отображать ему только его задачи. Это поможет избежать появления в списке данных пользователя тех данных, которые ему не принадлежат.

Далее, каждые данные представляют собой какое-либо слово, набор слов или предложение. Это является самым основным полем в таблице, которое несет в себе главную информацию во всем сервисе.

В итоге будет созданы три хранимые процедуры базы данных с названием «SearchAllTables» для поиска по всем таблицам, «sp_Отчет» для составления отчётности по данным и т.д.

Создание таких таблиц позволяет наглядно продемонстрировать не только саму структуру базы данных, но и описывает все данные, которые в дальнейшем должны использоваться в приложении. Каждая таблица на схеме выделена в отдельный прямоугольник, в котором указывается название таблицы и все ее поля. Ключевые поля в таблицах выделяются специальным символом слева от названия поля.

Запросы на создание хранимой процедуры «SearchAllTables» в базе данных выглядит следующим образом:

```

CREATE PROCEDURE [dbo].[SearchAllTables]
(
    @SearchStr nvarchar(100)
)
AS
BEGIN
    CREATE TABLE #Results (ColumnName nvarchar(370), ColumnValue nvarchar(3630))

    SET NOCOUNT ON

    DECLARE @TableName nvarchar(256), @ColumnName nvarchar(128), @SearchStr2 nvarchar(110)
    SET @TableName = ''
    SET @SearchStr2 = QUOTENAME('%' + @SearchStr + '%', '')

    WHILE @TableName IS NOT NULL
    BEGIN
        SET @ColumnName = ''
        SET @TableName =
        (
            SELECT MIN(QUOTENAME(TABLE_SCHEMA) + '.' + QUOTENAME(TABLE_NAME))
            FROM INFORMATION_SCHEMA.TABLES
            WHERE TABLE_TYPE = 'BASE TABLE'
            AND QUOTENAME(TABLE_SCHEMA) + '.' + QUOTENAME(TABLE_NAME) > @TableName
            AND OBJECTPROPERTY(
                OBJECT_ID(
                    QUOTENAME(TABLE_SCHEMA) + '.' + QUOTENAME(TABLE_NAME)
                ), 'IsMSShipped'
            ) = 0
        )

        WHILE (@TableName IS NOT NULL) AND (@ColumnName IS NOT NULL)
        BEGIN
            SET @ColumnName =
            (
                SELECT MIN(QUOTENAME(COLUMN_NAME))
                FROM INFORMATION_SCHEMA.COLUMNS
                WHERE TABLE_SCHEMA = PARSENAME(@TableName, 2)
                AND TABLE_NAME = PARSENAME(@TableName, 1)
                AND DATA_TYPE IN ('char', 'varchar', 'nchar', 'nvarchar')
                AND QUOTENAME(COLUMN_NAME) > @ColumnName
            )

            IF @ColumnName IS NOT NULL
            BEGIN
                INSERT INTO #Results
                EXEC
                (
                    'SELECT ''' + @TableName + '.' + @ColumnName + ''', LEFT(' + @ColumnName + ', 3630)
                    FROM ' + @TableName + ' (NOLOCK) ' +
                    ' WHERE ' + @ColumnName + ' LIKE ' + @SearchStr2
                )
            END
        END
    END

    SELECT ColumnName, ColumnValue FROM #Results
END

```

Рисунок 2.3.5 – Запрос для создания хранимой процедуры «SearchAllTables»

Запросы на создание хранимой процедуры «sp_Отчет» в базе данных выглядит следующим образом:

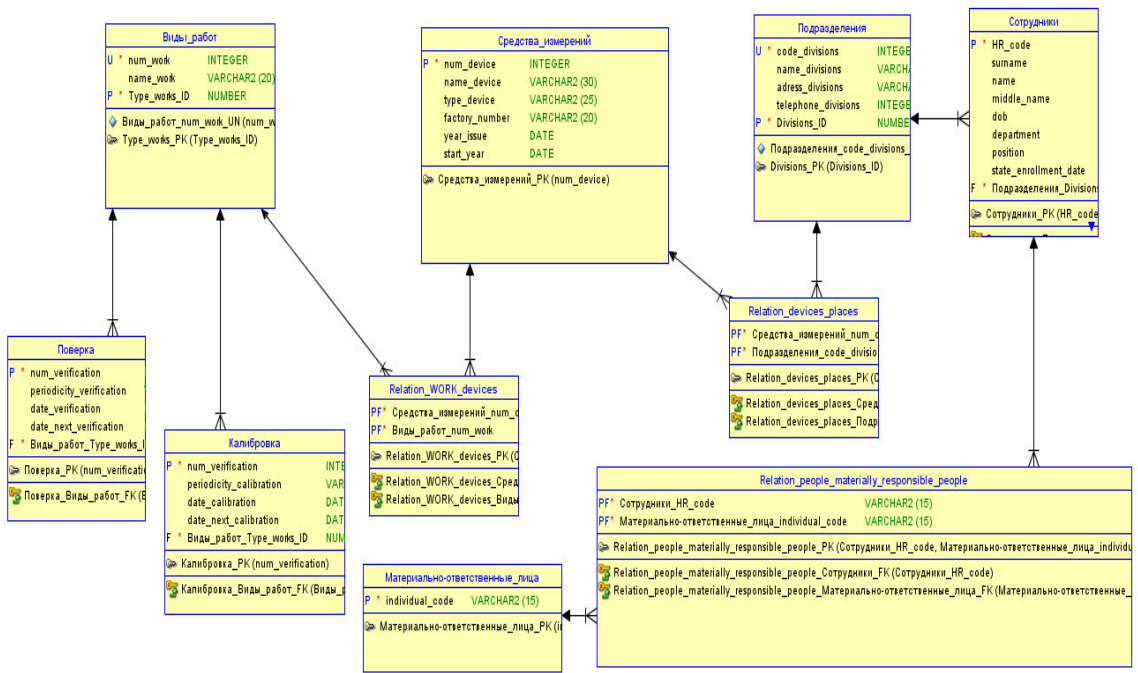


Рисунок 2.3.7 – Структура базы данных в виде физической модели базы данных

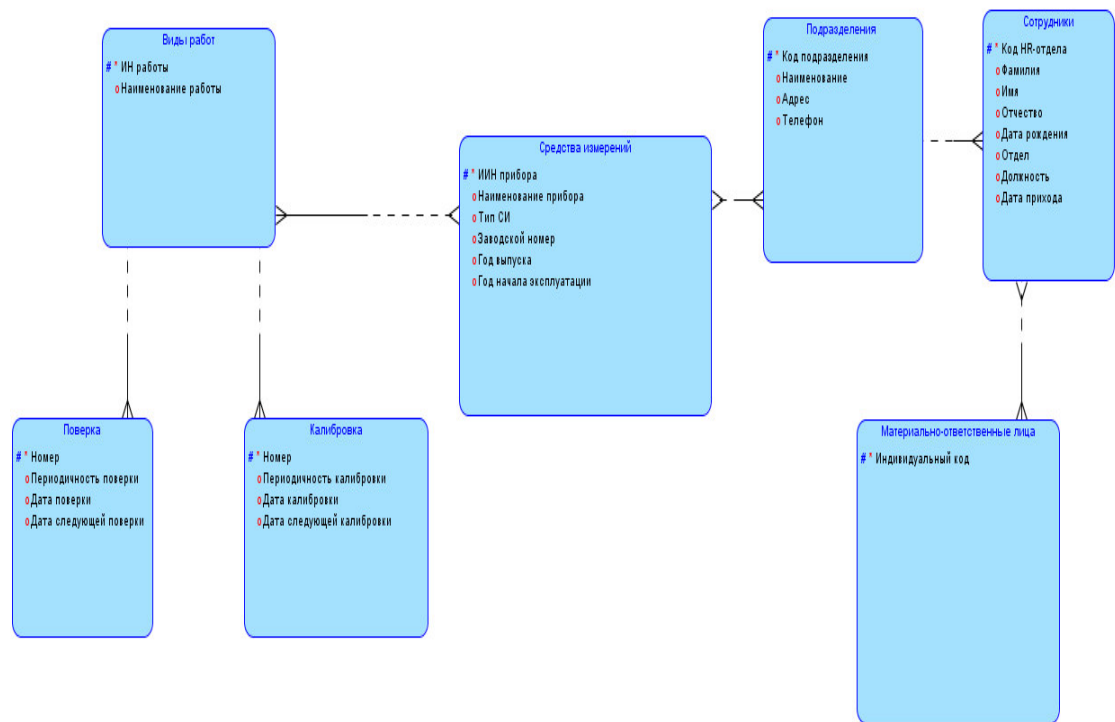


Рисунок 2.3.8 – Структура базы данных в виде логической модели базы данных

2.4 Интерфейс и способы привязки данных

Разработчикам приложений баз данных при создании кода в Visual Studio приходится постоянно обращаться к SQL-серверам типа Oracle или MS SQL Server. Просмотр содержания таблиц, создание индексов и запросов - совершать все эти рутинные манипуляции, крайне желательно не переключаясь в какие-либо сторонние программы. Создатели предусмотрели этот момент и включили в состав Visual Studio «Обозреватель серверов»:

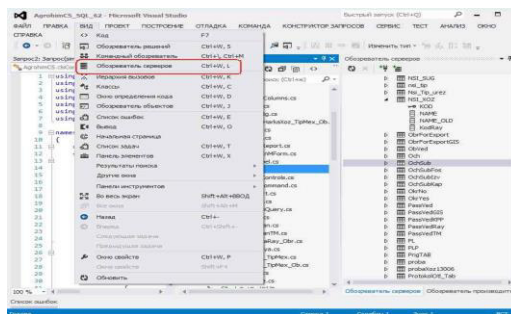


Рисунок 2.4.1 – Обозреватель серверов в среде разработки Visual Studio

Но прежде чем начать с ним работать, необходимо будет выполнить кое-какие предварительные действия - создать подключение. Реализовать такую процедуру попробуем следующим образом - по меню «Сервис - Подключиться к базе данных»:

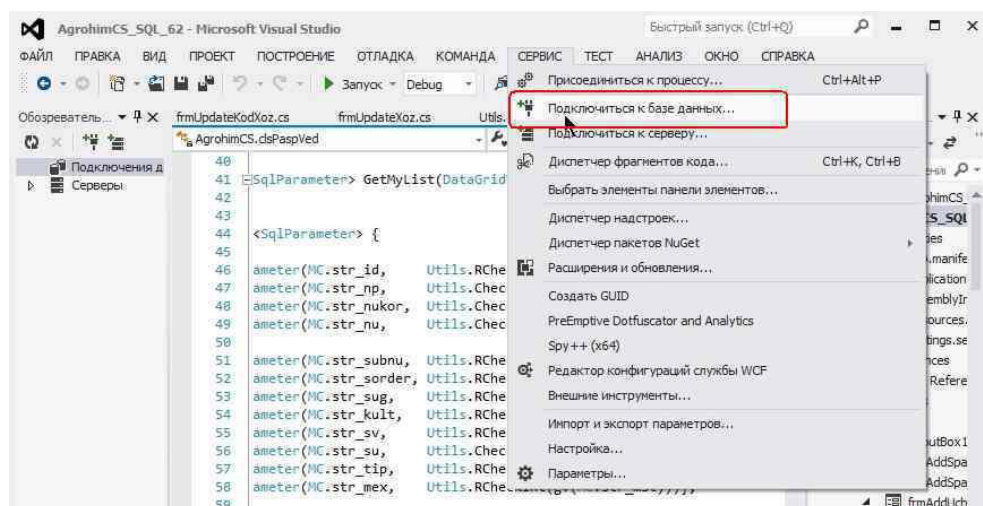


Рисунок 2.4.2 – Подключение к SQL серверу в среде разработки Visual Studio

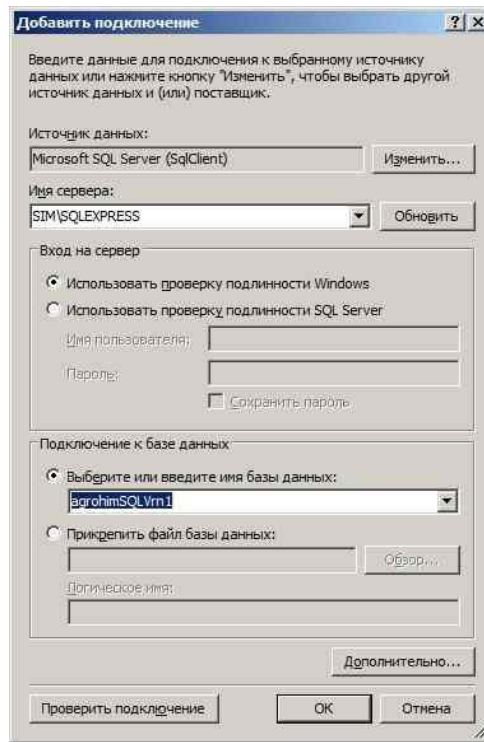


Рисунок 2.4.3 – Добавление данных подключения к SQL серверу в среде разработки Visual Studio.

3 Разработка программного продукта

Для создания графического проекта нам потребуется среда разработки Visual Studio. Поскольку наиболее распространенная пока версия Visual Studio 2013, то для данного руководства я буду использовать бесплатную версию данной среды Visual Studio Community 2013.

После установки среды и всех ее компонентов, запустим Visual Studio и создадим проект графического приложения. Для этого в меню выберем пункт File (Файл) и в подменю выберем New - Project (Создать-Проект). После этого перед нами откроется диалоговое окно создания нового проекта:

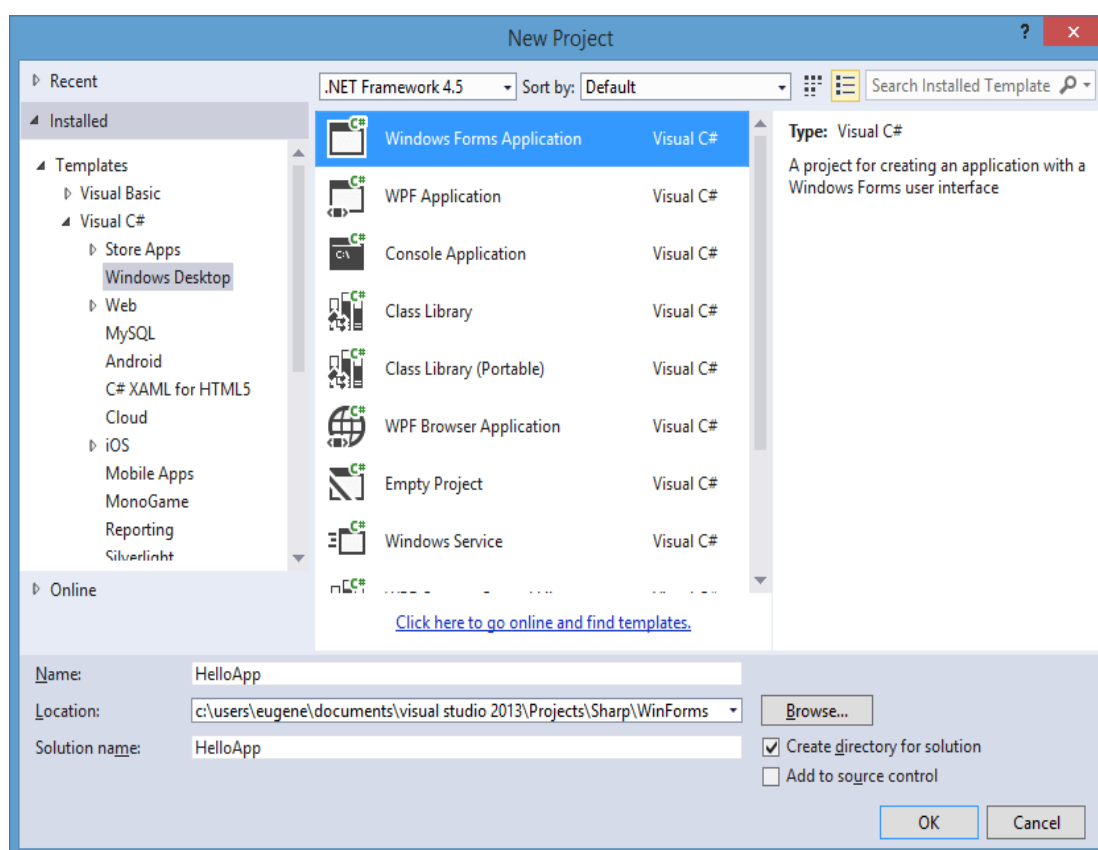


Рисунок 3.1 – Создание нового проекта Windows Forms C#

В левой колонке выберем Windows Desktop, а в центральной части среди типов проектов - тип Windows Forms Application и дадим ему какое-нибудь имя в поле внизу. Например, назовем его HelloApp. После этого нажимаем ОК.

После этого Visual Studio откроет наш проект с созданными по умолчанию файлами:

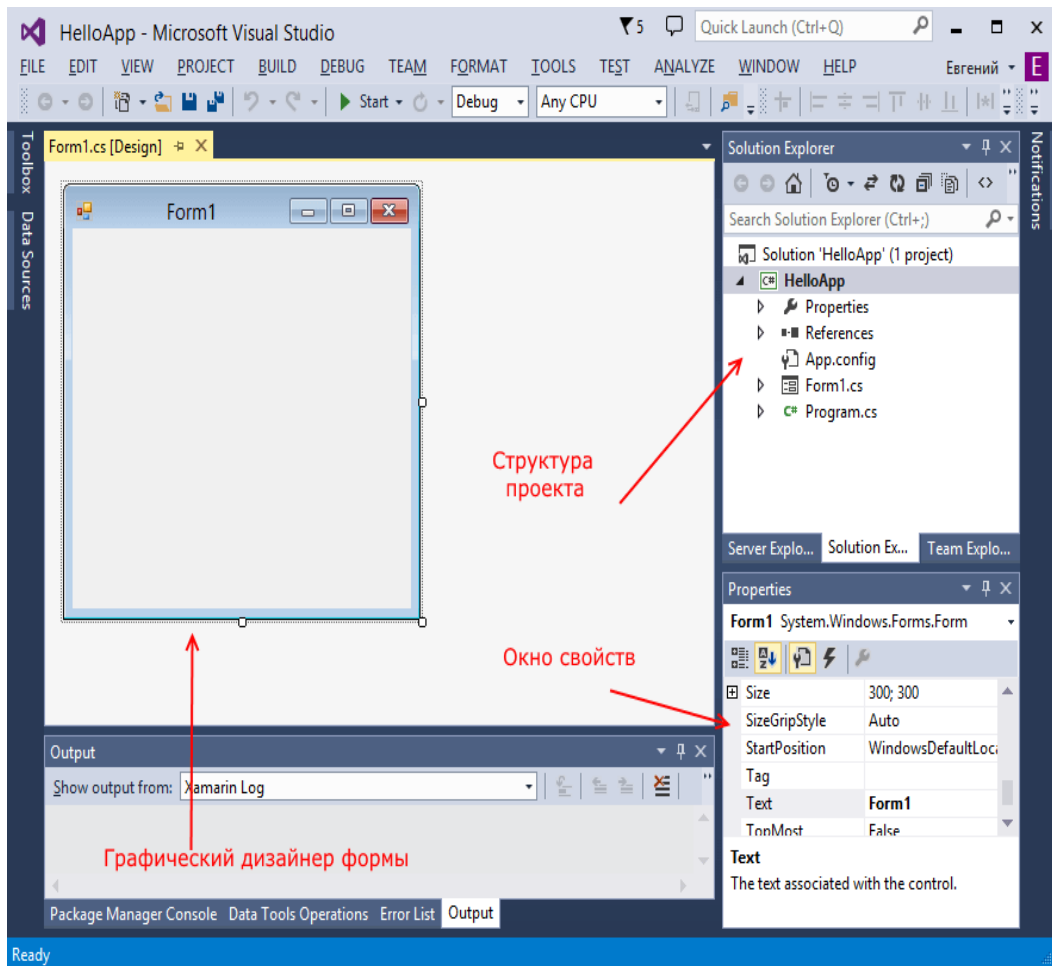


Рисунок 3.2 – Описание созданного проекта Windows Forms C#

Большую часть пространства Visual Studio занимает графический дизайнер, который содержит форму будущего приложения. Пока она пуста и имеет только заголовок Form1. Справа находится окно файлов решения/проекта - Solution Explorer (Обозреватель решений). Там и находятся все связанные с нашим приложением файлы, в том числе файлы формы Form1.cs.

Чтобы запустить приложение в режиме отладки, нажмем на клавишу F5 или на зеленую стрелочку на панели Visual Studio. После этого запустится наша форма с одинокой кнопкой. И если мы нажмем на кнопку на форме, то нам будет отображено сообщение с приветствием.

После запуска приложения студия компилирует его в файл с расширением exe. Найти данный файл можно, зайдя в папку проекта и далее в каталог bin/Debug или bin/Release.

Вся работа выполнялась на сервере Windows Server 2016 года, где были установлены среда программирования и разработки Microsoft Visual Studio 2017 и среда обработки данных баз данных SQL Server Management Studio 2016 года.

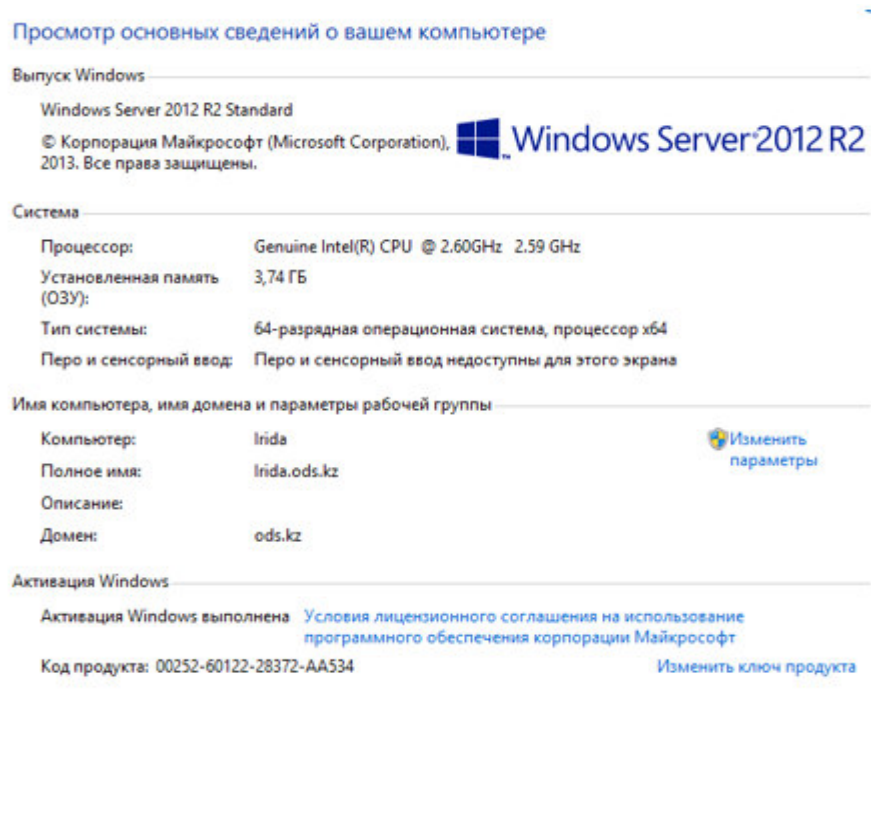


Рисунок 3.3 – Характеристика сервера, на котором была создана база

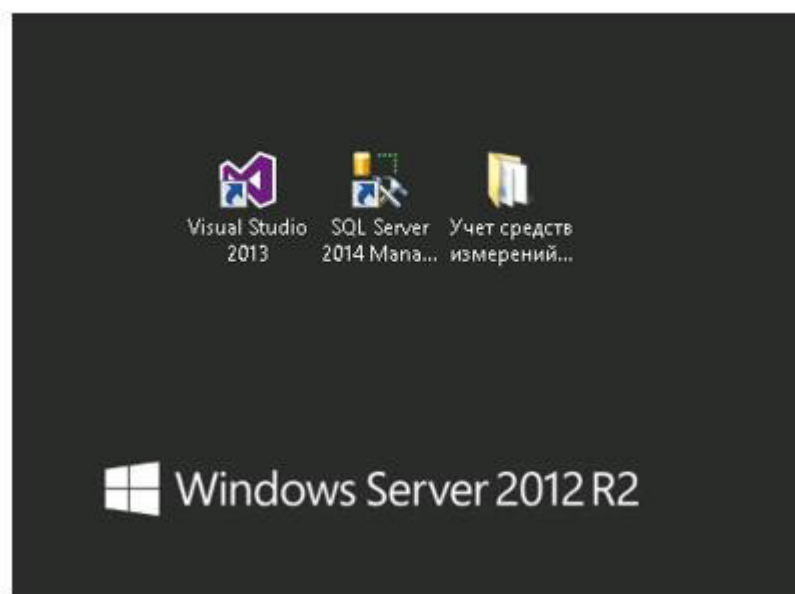


Рисунок 3.4 – Необходимые компоненты для создания и разработки базы

Проект состоит из нескольких частей и множества компонентов, таких как:

– «BD» – папка, где хранится весь написанный код интерфейса программы (формы авторизации пользователей, выбора списков и т.п.);

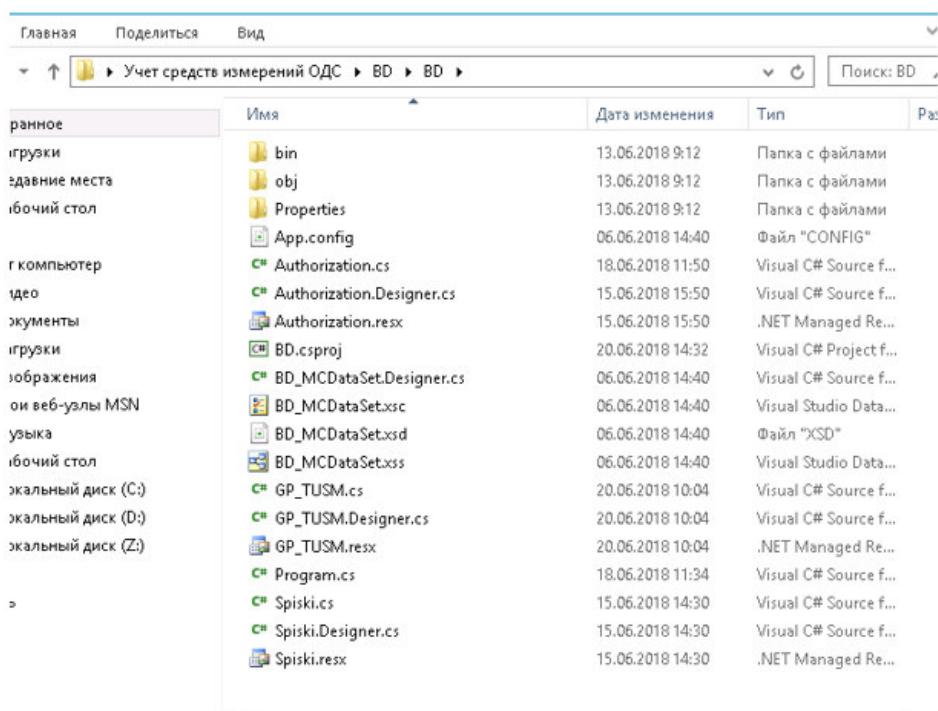


Рисунок 3.5 – Содержимое папки «BD»

– «bd_mc.sql» – файл запроса базы данных SQL, в котором прописана структура базы данных (таблицы, авторизация пользователей);

– 4 файла Excel – это файлы для выгрузки шаблонов информации по спискам базы «Учет средств измерений ОДС».

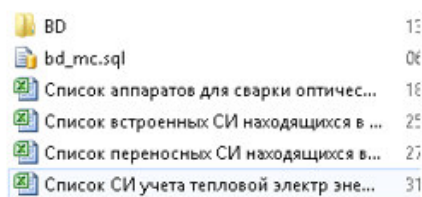


Рисунок 3.6 – Файлы, необходимые для полноценной работы проекта

При запуске программы нас встречает окно «Авторизация пользователей».

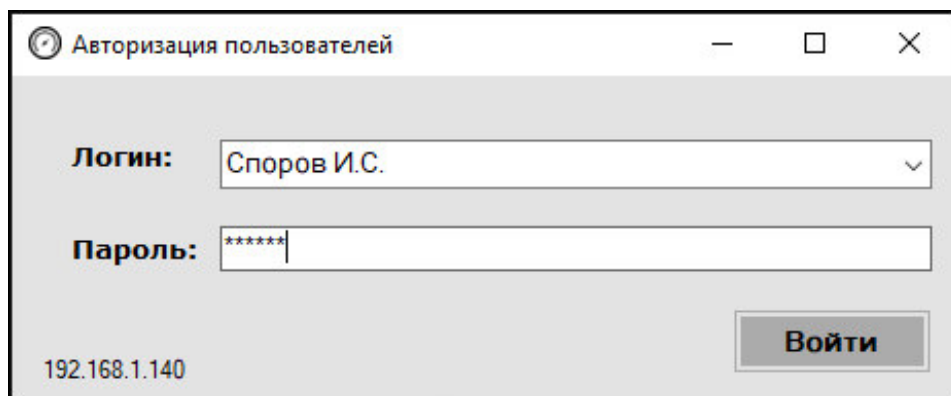


Рисунок 3.7 – Окно авторизации пользователя

Далее, после авторизации, выходит окно выбора определенного подразделения, в котором будут обрабатываться данные списков. Списки у каждого филиала индивидуальные.

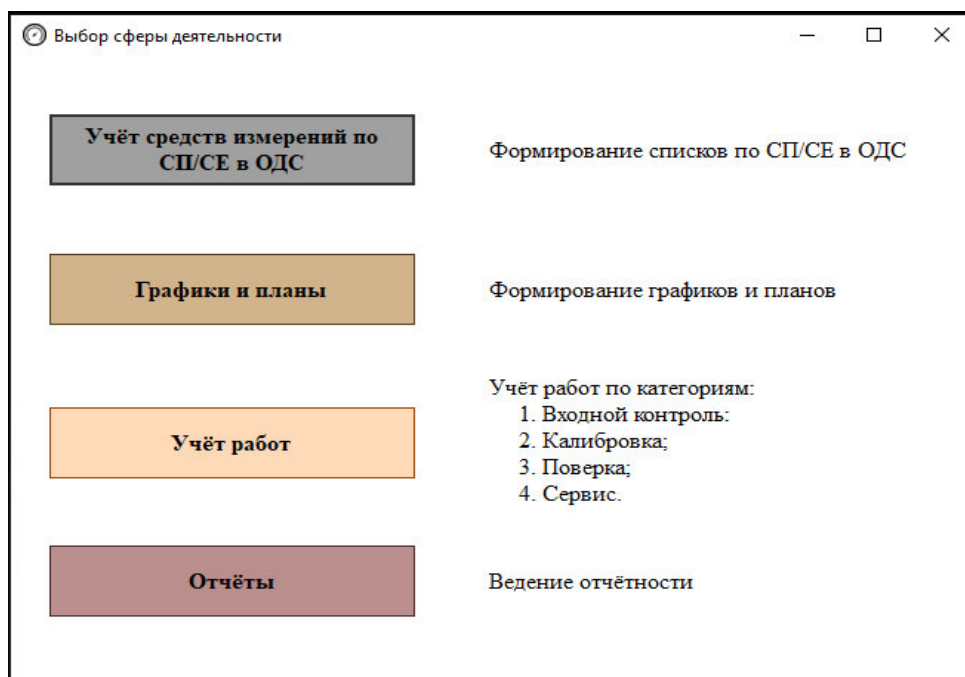


Рисунок 3.7 – Окно выбора необходимого филиала для работы

После выбора филиала открывается окно работы со списками. В нем мы можем выгрузить и загрузить данные из Excel при помощи шаблонных файлов списков, отобразить отчет по измененным действиям в HTML, а также сохранять и редактировать данные прямо в таблице окна.

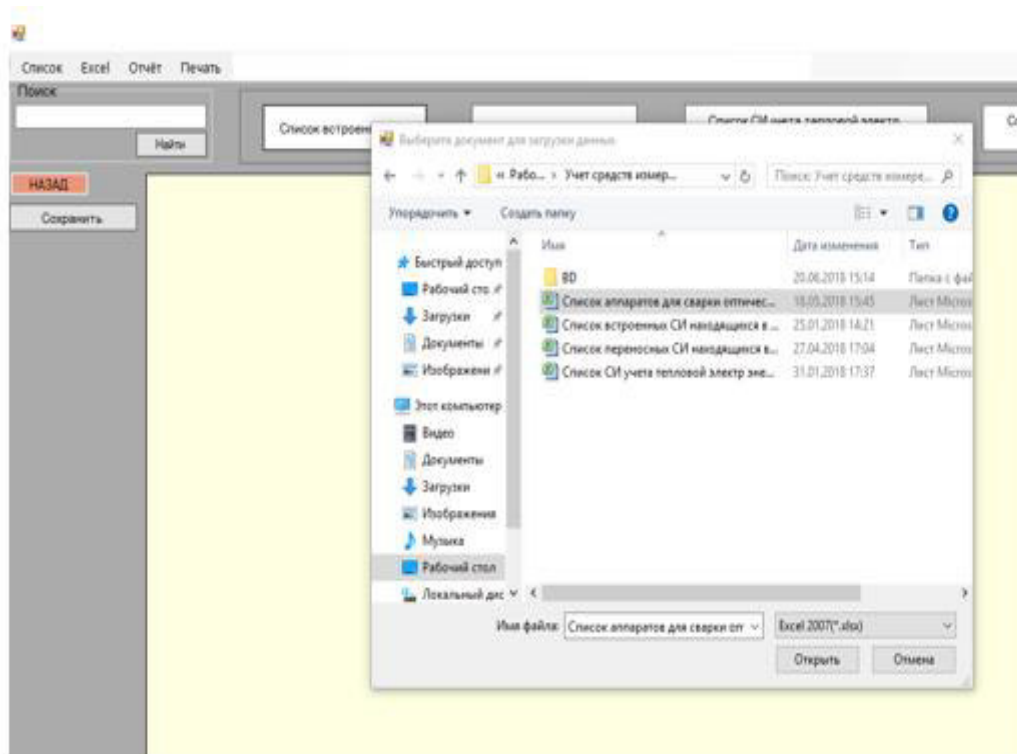


Рисунок 3.8 – Открытие файла Excel для последующей выгрузки в окно работы со списками

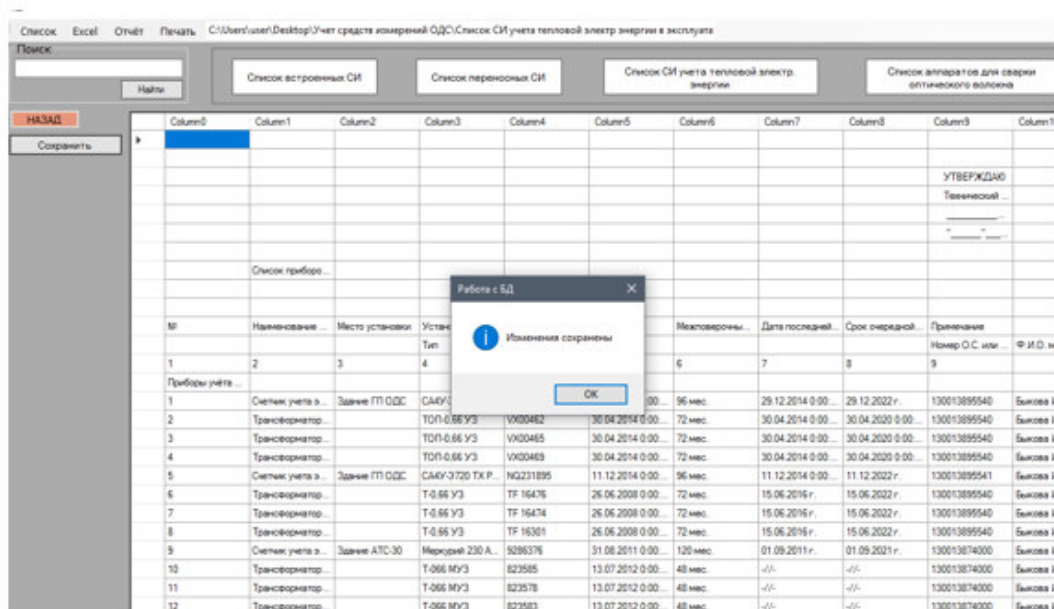


Рисунок 3.9 – Сообщение о том, что все наши отредактированные данные успешно сохранены в базу данных

Полный код программы представлен не будет, так как был использован стандартный спектр функций языка С#. Самый интересный и новый для меня момент был в написании загрузки/выгрузки в Excel.

```
OpenFileDialog ofd = new OpenFileDialog();
ofd.DefaultExt = "*.xls;*.xlsx";
ofd.Filter = "Excel 2007(*.xlsx)|*.xlsx|Excel 2003(*.xls)|*.xls";
ofd.Title = "Выберите документ для загрузки данных";

if (ofd.ShowDialog() == DialogResult.OK)
{
    toolStripTextBox1.Text = ofd.FileName;

    System.IO.FileStream stream =
        System.IO.File.Open(ofd.FileName, System.IO.FileMode.Open,
System.IO.FileAccess.Read);

    Excel.IExcelDataReader IEDR;

    int fileformat = ofd.SafeFileName.IndexOf(".xlsx");

    if (fileformat > -1)
    {
        //2007 format *.xlsx
        IEDR = Excel.ExcelReaderFactory.CreateOpenXmlReader(stream);
    }
    else
    {
        //97-2003 format *.xls
        IEDR = Excel.ExcelReaderFactory.CreateBinaryReader(stream);
    }

    //Если данное значение установлено в true
    //то первая строка используется в качестве
    //заголовков для колонок
    IEDR.IsFirstRowAsColumnNames = true;

    DataSet ds = IEDR.AsDataSet();

    //Устанавливаем в качестве источника данных dataset
    //с указанием номера таблицы. Номер таблицы указывает
    //на соответствующий лист в файле нумерация листов
    //начинается с нуля.
    mainDataGrid.DataSource = ds.Tables[0];
    IEDR.Close();
}
else
{
    MessageBox.Show("Вы не выбрали файл для открытия",
"Загрузка данных...", MessageBoxButtons.OK, MessageBoxIcon.Error);
}
}
```

Рисунок 3.9 – Код выгрузки из базы данных в MS Excel.

4 Экономическая часть

Технико-экономическое обоснование дипломной работы, связанное с разработкой программного продукта (ПП).

4.1 Трудоемкость разработки ПП

Для того, чтобы подсчитать трудоемкость разработки программного продукта «База данных метеорологического центра» должен быть выделен следующий перечень работ: поиск информации, составление технического задания, разработка разных версий приложения, работы по завершению и т.д.

Чтобы рассчитать время на создание программного продукта, определяем время, затраченное на каждый этап создания программного продукта. Все этапы работ и виды показаны в таблице 1, с учётом их трудоёмкости разработки.

Таблица 1 - Распределение работ по этапам и видам и оценка их трудоемкости

Этап разработки ПП	Вид работы на данном этапе	Трудоемкость разработки ПП, чел.× ч.
Этап изучения	Получение и поиск информации о предметной области ПП	24
Этап разработки ТЗ (технического задания)	Составление технического задания	86
Подготовительный этап	-утверждение эскиза проекта; -разработка первоначальной версии приложения	86
Технический проект	Разработка технических версий проекта для определения лучшей версии конфигурации в физико-логическом плане устройств	130
Интерфейсный этап	Рабочая версия проекта с разработанным интерфейсом	130
Этап внедрения	- тестирование и внедрение программного продукта; - написание инструкций использования ПП для пользователей.	363
Итого трудоемкость выполнения дипломной работы		819

Продолжительность рабочего дня равна 8 часам. То есть, если разделить общую трудоемкость разработки в 819 часов на человека на 8 часов рабочего дня, то получим, что для всей разработки необходимо 102 рабочих дня.

4.2 Расчет затрат на разработку ПП

Определение затрат необходимых для разработки программного обеспечения производится на основе имеющейся сметы, которая включает следующие элементы:

- материальные затраты;
- затраты на оплату труда;
- социальный налог;
- амортизация основных фондов;
- прочие затраты.

Материальные затраты делятся на основные и вспомогательные затраты на материалы, энергию и другие затраты необходимые для разработки ПО. Расчет материальных затрат происходит по форме, предоставленной в таблице 2.

Таблица 2 – Затраты на материальные ресурсы

Наименование материала	Марка	Единица измерения	Кол-во	Цена за ед. в тенге	Сумма в тенге
Бумага для офиса	«Снежинка»	Упаковка	5	1110,02	5550,10
Органайзер	«Letts of London»	Штук	1	3250,00	3250,00
Ручки	«Cello Jetta Steel»	Штук	3	85,00	255,00
Компьютерная USB мышь и клавиатура	«Genius» KB-900X	Штук	1	12100,00	12100,00
Итого:					21155,10

Для разработки программного обеспечения будет использоваться персональный стационарный компьютер с системным блоком Neo Game (Cі5-8400 2,8Ghz/8GB/1TB/GTX1050 2GB/DVD-RW/V3XB), мощности которого достаточно для выполнения поставленных задач. Так как ПК содержит установленную операционную систему Windows 10 Pro x64 и Windows Server 2012 Pro x64 и программное обеспечение необходимое для разработки ПО

(Microsoft Visual Studio, MS SQL Server, MS Office), нет нужды производить дополнительные затраты на новую ОС и ПО.

Расчет затрат на необходимое оборудование и программное обеспечение производится по форме, приведенной в таблице 3.

Таблица 3 – Расчет затрат на оборудование и ПО, необходимое для проекта

Наименование материала	Марка	Ед. измерения	Кол-во	Цена за ед. в тенге	Сумма в тенге
Системный блок	Neo Game (Ci5-8400 2,8Ghz/8GB/1TB/GT X1050 2GB/DVD-RW/V3XB)	Штук	1	264090,11	264090,11
Принтер	HP LaserJet 1022	Штук	1	67389,00	67389,00
Монитор	Philips 276E9QSB, Black-Silver	Штук	1	90917,00	90917,00
Модем	TP-Link TD-W8968	Штук	1	10500,00	10500,00
ОС	Windows 10	Штук	1	-	-
Сервер	Windows Server 2012	Штук	1	-	-
Среда разработки	Microsoft Visual Studio 2017 Professional	Штук	1	-	-
Среда обработки базы данных	Microsoft SQL Server 2014 Express	Штук	1	-	-
Офисные приложения	Microsoft Office 2016 Pro	Штук	1	-	-
Итого:					432 896,11

4.3 Расчет затрат на электроэнергию

Для разработки данного программного продукта используется электрооборудование. Значит, необходимо рассчитать затраты на электроэнергию.

Общая сумма затрат на электроэнергию ($Z_э$) рассчитывается по формуле:

$$Z_э = \sum_{i=1}^n M_i \cdot K_i \cdot T_i \cdot Ц, \quad (4.3.1)$$

где M_i - паспортная мощность i -го электрооборудования, кВт;
 K_i - коэффициент использования мощности i -го электрооборудования (принимается $K_i=0.7, 0.9$);
 T_i - время работы i -го оборудования за весь период разработки ПП ч;
 $Ц$ - цена электроэнергии, тг/кВт×ч (1кВт/ч = 23,85 тг);
 i - вид электрооборудования;
 n - количество электрооборудования.

Все затраты на электроэнергию по электроприборам приведены в таблице 4.

Таблица 4 – Затраты на электроэнергию

Наименование приборов	Паспортная мощность, кВт	Коэффициент мощности	Время работы оборудования, ч	Цена ЭЭ тг/кВтч	Сумма, тг.
Системный блок	0,4	0,7	327	23,85	2183,71
Модем	0,9	0,8	327	23,85	5615,24
Монитор	0,7	0,6	327	23,85	3275,56
Принтер	0,4	0,9	14	23,85	120,20
Освещение	0,3	0,7	197	23,85	986,67
				Итого:	12181,38

Могут понадобиться дополнительные расходы на электроэнергию в течении всех этапов разработки. Расчёт оплаты данных доп. расходов подсчитываются на основе повышенного показателя в объеме 5% от расходов на электроэнергию:

$$Z_{доп} = 0,05 \cdot Z_э, \quad (4.3.2)$$

Определим затраты на дополнительные потребности согласно формуле (4.3.2):

$$Z_{\text{доп}} = 0,05 \cdot 12181,38 = 609,07 \text{ (тенге)}$$

Итак, мы можем сделать вывод, что полная сумма затрат на электроэнергию будет составлять 12790,45 тенге.

4.4 Расчет затрат на оплату труда

Для разработки программного продукта необходимо два работника:

- руководитель проекта – управление рабочим временем, корректировка рабочих процессов, координация, изучение предметной области;
- разработчик – разработка ПО, тестирование и сопровождение.

Все расходы на оплату труда сотрудников можно рассчитать по формуле:

$$Z_{\text{тр}} = \sum ЧС_i * T_i \quad (4.4.1)$$

где $ЧС_i$ - часовая ставка i -го работника, тг;

T_i - трудоемкость разработки модели, чел.×ч; i - категория работника;

n - количество работников, занятых разработкой ПП.

Часовую ставку сотрудника можно вычислить по следующей формуле:

$$ЧС_i = \frac{ЗП_i}{ФРВ_i} \quad (4.4.2)$$

где $ЗП_i$ - месячная заработная плата i -го работника, тг;

$ФРВ_i$ - месячный фонд рабочего времени i -го работника, час.

Месячная заработная плата руководителя равняется 270 000 тенге и месячная заработная плата разработчика равняется 210 000 тенге. Рассчитаем часовую ставку каждого работника согласно формуле (4.4.2):

$$ЧС_{\text{руководитель}} = \frac{270\,000}{22 * 8} = 978,26 \text{ тг/ч}$$

$$ЧС_{\text{разработчик}} = \frac{210\,000}{22 * 8} = 760,86 \text{ тг/ч}$$

Часовая ставка руководителя составляет 978,26 (тг/ч), трудоемкость разработки равняется 100 часам. Часовая ставка разработчика составляет 760,86 (тг/ч), трудоемкость разработки равняется 818 часам. Согласно формуле (4.4.1) можно рассчитать сумму расходов на заработную плату работников:

$$Z_{\text{тр}} = 978,26 * 100 + 760,86 * 819 = 97826 + 623144,34 = 720970,34 \text{ тенге}$$

Все показатели и вычисления по оплате труда показаны в таблице 5.

Таблица 5 – Расчет заработной платы

Категория работника	Квалификация	Трудоемкость разработки ПП, час.	Часовая ставка, тг/ч	Сумма, тг.
Руководитель	Проектный руководитель	100	978,26	97826,00
Разработчик	Программист	819	760,86	623144,34
Итого:				720970,34

4.5 Расчет затрат по социальному налогу

Согласно Налоговому кодексу Республики Казахстан социальный налог составляет 9,5% от фонда оплаты труда. Социальный налог можно рассчитать по следующей формуле:

$$C_H = (\text{ФОТ} - \text{ПО}) \cdot 0,095 \quad (4.5.1)$$

где ПО - отчисления в пенсионный фонд (10% от ФОТ).

$$\text{ПО} = 720970,34 \cdot 0,1 = 72097,03 \text{ тенге}$$

$$C_H = (720970,34 - 72097,03) \cdot 0,095 = 61642,97 \text{ тенге}$$

Результаты расчетов представлены в таблице 6.

Таблица 6 – Начисление социального налога

Категория работника	Количество человек	Заработная плата, тг	Пенсионные отчисления, тг	Социальный налог, тг
Руководитель	1	97826,00	9782,60	8364,13
Разработчик	1	623144,34	62314,43	53278,84
Итого:				61642,97

4.6 Амортизация основных фондов и прочие затраты

Под амортизацией основных фондов подразумевается сумма амортизационных отчислений от стоимости оборудования и программного обеспечения (ПО), используемых при разработке программных продуктов.

Общая сумма амортизационных отчислений определяется по формуле:

$$Z_{\text{амор}} = \sum_{i=1}^n \frac{\Phi_i \cdot N_{Ai} \cdot T_{\text{НИР}i}}{100 \cdot T_{\text{ЭФ}i}} \quad (4.6.1)$$

где Φ_i - стоимость i -го ОФ, тг;
 N_{Ai} - годовая норма амортизации i -го ОФ, % (по налоговому кодексу РК);

$T_{\text{НИР}i}$ - время работы i -го ОФ за весь период разработки ПП, ч;
 $T_{\text{ЭФ}i}$ - эффективный фонд времени работы i -го ОФ за год, ч/год;
 i - вид ОФ;
 n - количество ОФ.

Годовой (эффективный) фонд времени работы оборудования – время работы оборудования в течение года, выраженное в часах:

$$\Phi_{\text{н}} = ((D_{\text{год}} - D_{\text{празд}} - D_{\text{вых}}) \cdot \text{Ч}_{\text{раб.дня}} - D_{\text{пв}} \cdot \Delta\text{Ч}_{\text{раб.дня}}) \cdot C \quad (4.6.2)$$

где $D_{\text{год}}$ – 365 дней в году;
 $D_{\text{празд}}$ – количество праздничных дней в году (14 дней);
 $D_{\text{вых}}$ – количество выходных дней в году – 106 дней (53 субботы и 53 воскресенья);
 $\text{Ч}_{\text{раб.дня}}$ - продолжительность рабочего дня;
 $D_{\text{пв}}$ – число предвыходных и предпраздничных дне с сокращенной продолжительностью смены (10 дней);
 $\Delta\text{Ч}_{\text{раб.дня}}$ – время на которое сокращается номинальная продолжительность рабочего дня (2 часа);
 C – число рабочих смен в сутках.

$$\Phi_{\text{н}} = ((365 - 14 - 106) \cdot 8 - 10 \cdot 2) \cdot 1 = 1940 \text{ ч/год}$$

Таблица 7 – Амортизация ОФ

Наименование оборудования и ПО	Стоимость оборудования и ПО, тг	Годовая норма амортизации, %	Эффективный фонд времени работы оборудования и ПО, ч/год	Время работы оборудования и ПО для разработки ПП, ч	Сумма, тг
Системный блок	264090,11	25	1940	327	11128,54
Принтер	67389,00	25	1940	14	121,58

Продолжение таблицы 7

Наименование оборудования и ПО	Стоимость оборудования и ПО, тг	Годовая норма амортизации, %	Эффективный фонд времени работы оборудования и ПО, ч/год	Время работы оборудования и ПО для разработки и ПП, ч	Сумма, тг
Монитор	90917,00	20	1940	327	3064,93
Модем	10500,00	20	1940	327	353,96
				Итого:	14669,01

4.7 Смета расходов на разработку ПО

На основе всех представленных расчетов необходимо оформить смету расходов на разработку ПО согласно форме, которая приведена в таблице 8.

Таблица 8 – Смета затрат на разработку ПО

Статьи затрат	Сумма, тг	В %
Затраты на оборудование	432 896,11	34,2
Затраты на материальные ресурсы	21155,10	1,7
Затраты на оплату труда	720970,34	57
Социальные налоги	61642,97	4,9
Затраты на электроэнергию	12181,38	1
Амортизация основных фондов	14669,01	1,2
Итого по смете:	1263514,91	100 %

На рисунке 1 продемонстрирована диаграмма сметы затрат на разработку программного обеспечения.

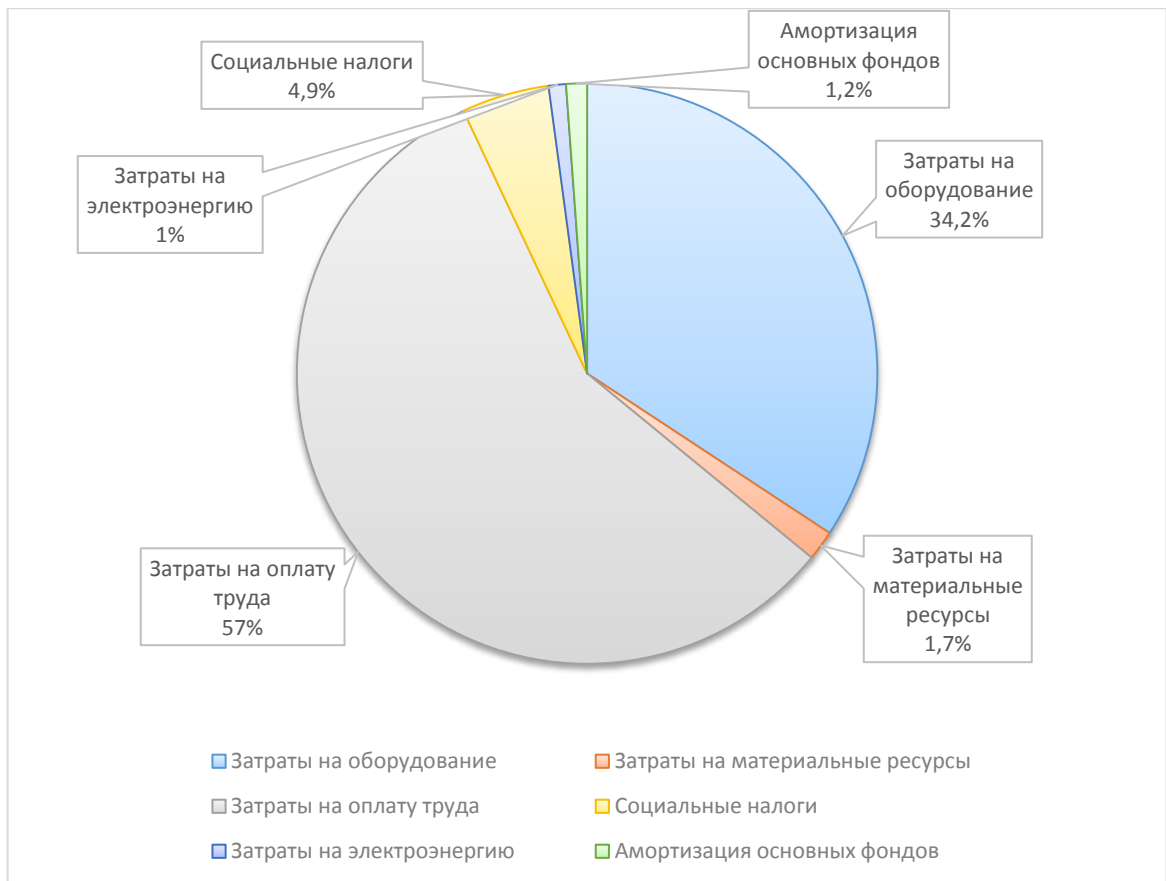


Рисунок 4.7.1 – Диаграмма сметы затрат на разработку программного обеспечения.

4.8 Определение возможной (договорной) цены ПО

Стоимость программного обеспечения определяется на основе качества разработанного продукта, сроков его разработки и производительности продукта. Стоимость C_d для программного обеспечения можно рассчитать по следующей формуле:

$$C_d = Z_{\text{нир}} \cdot \left(1 + \frac{P}{100}\right), \quad (4.8.1)$$

где $Z_{\text{нир}}$ – затраты на разработку программного обеспечения, тг;

P – средний уровень рентабельности ПО, (%). Данный параметр принят равным 25%.

$$\begin{aligned} C_d &= 1263514,91 \cdot \left(1 + \frac{25}{100}\right) = 1263514,91 + (1263514,91 \cdot 0,25) \\ &= 1263514,91 + 315878,73 = 1579393,64 \text{ тенге} \end{aligned}$$

Исходя из данных расчетов, прибыль от разработки равна:

$$1263514,91 \cdot 0,25 = 315878,73 \text{ тенге}$$

Далее необходимо определить стоимость реализации с учетом НДС, ставка НДС устанавливается законодательством РК. На 2019 года ставка НДС составляет 12%. Стоимость реализации учитывая НДС можно рассчитать по следующей формуле:

$$C_p = C_d + C_d \cdot \text{НДС}, \quad (4.8.2)$$

$$C_p = 1579393,64 + 1579393,64 \cdot 0,12 = 1768920,88 \text{ тенге}$$

Прибыль от разработки программного продукта равна 315878,73 тенге.
Себестоимость программного продукта равна 1263514,91 тенге.
Стоимость реализации с учетом НДС равна 1768920,88 тенге.

5 Охрана труда и безопасность жизнедеятельности

Дипломная работа на тему «Разработка базы данных для метрологического центра» была разработана для метрологических центров и лабораторий, которые имеют как центры в офисных помещениях, так и в зданиях заводского типа со специализированными лабораториями. Следуя из этого, мы можем выделить основной фактор, который может повлиять на работу сотрудников метрологических центров с данным программным обеспечением – это освещение.

Так как, требования и нормы к освещению, как к производственным факторам, в лабораторных условиях и помещении под офис будут в корне разными, то следует рассмотреть каждый случай отдельно и рассчитать все параметры, исходя из конкретной местности работы сотрудника.

Метрологические центры и лаборатории занимаются, в первую очередь, капремонтом и наладкой средств измерений, и калибровкой приборов. Для данных помещений имеется несколько собственных условий:

- здание с целью выполнения калибровки и ремонтных работ средств измерений обязаны отвечать функционирующим строительным и санитарным нормам, правилам и условиям защищенности работы и защиты окружающей среды. Они обязаны оставаться сухими, чистыми и отделенными, с производственными зонами. Никак не допускается проводить через них парогазопровода и фановые трубы;

- починка средств измерительной техники организуется в специально назначенных комнатах либо в специально оснащенных трудовых зонах;

- объемы дверных проемов и уровень порожка должны выходить с критериев беспрепятственного передвижения оборудования, существующего в метрологической сфере. При надобности, здания оборудуются лифтами и подъемниками, внутренней телефонной взаимосвязью, охранной сигнализацией. Стены комнат и потолок рекомендуется красить стойкой краской ярких тонов, позволяющей мокрую протирку. Полы комнат обязаны быть укрыты линолеумом (линолеум, пластмассой) либо глиняной плиткой в связи с характером проводимых работ. Разрешается древесное напыление. Перекрытия и потолки комнат обязаны содержать гидроизоляцию;

- здание с целью проведения калибровки средств измерительной техники обязаны значиться в специализированных зданиях либо отделенных комнатах совместных строений отдаленно с высоковольтных линий электропередач, контактной электросети (электротранспорта), источников пульсации, гула, радиопомех (электросварочного и частотного электрооборудования) и с объектов, образуют значительные магнитные либо гальванические поля (реформаторских подстанций, конструкций индуктивного нагрева и т.п.). Допускаемая степень преград устанавливается в нормативных бумагах в надлежащие технологии калибровки (поверки);

– здание для выполнения градуировочных и исправительных работ уместно располагать окнами на север, северо-западное направление. ant. восток и северо-восток;

Искусственное освещение комнат обязано оставаться комбинированным (совместное и местное) и растерянным (однородным). Список источников освещения обязаны быть заключены в арматуру с непрозрачным либо снежным стеклом. Использование раскрытых ламп никак не разрешается. Свет комнат на уровне трудовых зон ($H = 0,8$ м) присутствие лампах накаливания обязана являться никак не меньше 150 лк, а присутствие люминесцентных лампах - никак не меньше триста лк. Показатель природной освещенности необходимо осуществлять одинаковым 1,5. С целью периодического развлечений трудящихся обязаны учитываться зоны с непосредственным миром присутствие коэффициенте природного освещения никак не меньше 0,5. Данные здания оборудуются подобно помещений в целях развлечений в согласовании с условиями строй общепризнанных мерок и законов согласно проектированию добавочных строений и комнат индустриальных компаний. С целью этого имеют все шансы быть использованы коридоры, холлы, вестибюли и прочие здания с непосредственным светом. Свет добавочных комнат обязан отвечать санитарно-эпидемиологическим законам СП 2.2.1.1312-03. В комнатах должен учитываться аварийный свет;

– отопление, вентилирование и кондиционирование воздуха комнат обязаны отвечать санитарно-эпидемиологическим правилам СП 2.2.1.1312-03. Рекомендуются использовать водяное либо воздушное отопление и кондиционирование, совмещенные с приточной вентиляцией. Нагревательные оборудование и приборы обязаны являться гладкими, подходящими в целях очищения от пыли и загрязнений. При надобности нагревательные батареи оборудуются тепловыми экранами.

Помещения для выполнения калибровки и ремонтных работ средств замеров, обычно, обязаны снабжаться приточно-дымовытяжной вентиляцией с обогревом в прохладный период и с 3–6-кратным обменом воздуха в 1 ч.;

– рабочие зоны по калибровке, где нормировано преобразование температуры охватывающей среды в единицу времени, обязаны размещаться в термостатированной комнате, оснащенной конструкциями и приборами для регулировки температуры и влаги. В случае недоступности в метрологической работе термостатированной комнаты эти трудовые зоны обязаны снабжаться приборами с целью регулировки условий калибровки (воздушные либо маслоподкачивающие термостаты, калориферы, кондиционеры и др.) в соответствии с условиями нормативной документации в методологии калибровки (поверки).

Работы, объединенные с подготовкой средств измерений к поверке, калибровке либо наладке (расконсервация, чистка, дозаправка и т.п.), сопровождаемые засорением атмосферы либо огнеопасными выделениями, обязаны проводиться в единичных отделенных комнатах. Трудовые зоны в данных комнатах обязаны оставаться оборудованы вытяжными шкафами,

местными отсосами и иными приборами с целью вытаскивания вредоносных либо огнеопасных жидкостей, паров и газов, а освещение обязано обладать взрывобезопасное осуществление.

Допустимые общепризнанные мерки температуры в производственных комнатах составляют:

- основные здания – (двадцатый...двадцать пятый) °С;
- вспомогательные здания – (пятнадцати...двадцати шести) °С.;
- относ. влажность в всех комнатах – (тридцать...семьдесят пять) %.

Допустимые степени голосового давления в трудовых участках обязаны отвечать условиям ГОСТ 12.1.003. В трудовых зонах для калибровки эти уровни никак не должны быть выше шестьдесят дБ.

– в комнатах с целью выполнения градировочных и исправительных работ параметры окружающей среды (жар, относительная влага, атмосферное давление), а еще иные обстоятельства (скорость изменения температуры, степень пульсации и т.п.), оказывающие большое влияние на метрологические и рабочие свойства средств измерений, обязаны отвечать условиям инженерных описаний, руководств по эксплуатации и нормативных бумаг в методологии их калибровки (поверки). В данных комнатах обязаны являться определены ресурсы замеров температуры, давления и влаги, а кроме того ресурсы пожарной сигнализации;

– силовая сеть питания станочного оборудования, вентиляционной системы, освещения, подъемников, лифтов и т.п. обязана выполняться сквозь основной распределительный щит и оставаться изолированной от сети питания трудовых зон в целях калибровки и ремонтных работ средств измерений. Сетевые цепи обязаны быть исполнены с помощью экранированных кабелей с заземлением экранов напрямую в распределительном щите. Электропитание средств измерений в трудовых зонах исполняется посредством розетки, количество и месторасположение каковых формируются непосредственно для любой трудовой зоны с учетом количества оборудования и специфичности производимых работ;

– зоны калибровки и ремонтных работ средств измерений геометрических величин уместно размещать на главном этаже. Оптико-механические рабочие эталоны (измерительные механизмы, многоцелевые микроскопы, отвесные и горизонтальные оптиметры и др.) обязаны монтироваться в фундаментах либо стабильных массивных стойках-подставках, исключаящих давление пульсации в корректность замеров. Данные аппараты обязаны иметь застекленные футляры либо чехлы с уплотненного использованного материала.

Помещения для калибровки водосчетчиков, нефтесчетчиков, маслосчетчиков, бензо и газосчетчиков обязаны содержать ровный паркет, уложенный глиняной плиткой, с уклоном в сторону маслосливных трапов. Стены обязаны оставаться выложены глиняной плиткой в высоту вплоть до 2 м. Схема электрического освещения обязана быть заключена в газовые трубы, а список источников света – в воздухо непроницаемую арматуру. Выключатели

освещения, за исключением комнат с целью калибровки (поверки) водосчетчиков, обязаны утверждаться извне здания либо быть сделаны в взрывобезопасном представлении.

Помещения обязаны содержать приточно-вытяжную вентиляцию с 3-сложным обменом воздуха в 1 ч. Подобные здания снабжают охраннопожарными средствами долговременного воздействия;

– пространство комнат должна находиться с расчетом на численности работников зон, достаточных для выполнения калибровочных и ремонтных работ, а кроме того беспрепятственного прохода к устройствам и оборудованию с учетом условий эксплуатационной документации на них и условиями данного стандарта. При нынешнем минимально разрешенная площадь 1-го работника зоны обязана составлять никак не меньше 6 м².

Площадь запасных комнат (аккумуляторная, агрегатная, вентиляционная, машинные мастерские, склад приема и выдачи устройств, учебный класс, складские здания, здания с целью мытья, очистки, хим и электрических покрытий, окрашивание, сушки, тестирований, консервации, гардеробные, умывальные и т. п.) и построек (состав горючих и трансмиссионных использованных материалов, боксы с целью мобильных денег и др.) обуславливается в соответствии с санитарно-эпидемиологическим законом ПРЕДПРИЯТИЕ 2.2.1.1312-03. Растрачиваемые использованные материалы (теплоносители, гидромеханические воды, в т.ч. топкие воды, спирты, очистные воды и т.п.) сохраняются в намеренно оснащенной комнатке либо отдельных шкафах, оснащенных дымовытяжной вентиляцией;

– здания, где выполняют калибровку, регулирование и восстановление средств измерений с ртутным наполнением, обязаны соответствовать условиям санитарных правил СП 4607-88.

5.1 Расчёт освещения в метрологических лабораториях для проведения калибровки и поверки средств измерений

Метрологические лаборатории для поверки имеют естественное и освещение искусственного происхождения в соответствии с СанПиН 2.2.2/2.41340-03.

Окна в комнатах, где поверяется аппарат, предпочтительно нацелены на север и север-восток. Оконные проемы оборудованы контролируемые установками вида: штора, занавесок, наружных козырьков и др.

Искусственное освещение в метрологической лаборатории для поверки осуществляется системой равномерного освещения.

Согласно СанПиН 23-05-95, в комнатах для IV ряда зрительных работ нужно предоставить освещенность 200лк. С целью искусственного освещения используют светильники с люминесцентными лампами вида ЛБ40 (ГОСТ 6825-91 «Лампы люминесцентные цилиндрические для совместного освещения»), со световым потоком $F=3200$ лк. Коэффициент распределения светового потока для люминесцентных ламп равен $Z=1,1$.

Проведем расчет искусственного и естественного освещения.

Расчет естественного освещения заключается в определении площади световых проемов для помещения.

Помещение имеет следующие размеры:

- А-длина - 5м
- В-ширина - 6м
- h- высота - 2,5м

В таблице 5.1 представлены нормированные значения естественного освещения по СН-П23.05-95.

Таблица 5.1 - Нормированные значения естественного освещения

Наименование	Обозначение	Значение
Коэффициент светопропускания	φ_0	1
Коэффициент отражения света от внутренней поверхности	S	30м ²
Коэффициент световой характеристики	h_0	0,7
Коэффициент запаса	K_3	18

Расчет площади окон производится по следующей формуле:

$$S_o = \frac{S \cdot e_H \cdot h_0 \cdot K_3}{\varphi_0 \cdot r_1 \cdot 100} \quad (5.1)$$

$$S_o = \frac{30 \cdot 1,5 \cdot 1,5 \cdot 18}{1 \cdot 0,7 \cdot 100} = 17,35 \text{ м}^2$$

Площадь окна равна:

$$S_{\text{окна}} = B \cdot h \quad (5.2)$$

Окно имеет следующие размеры:

- В-ширина - 2,5м
- h-высота - 1,5м.

$$S_{\text{окна}} = 2,5 \cdot 1,5 = 3,75 \text{ м}^2$$

Количество окон в метрологической лаборатории для поверки:

$$K = \frac{S_o}{S_{\text{окна}}} \quad (5.3)$$

$$K = \frac{17,35}{3,75} = 4,62 \text{ шт.}$$

Значит в лаборатории должно быть 4 окна.

Расчет искусственного освещения.

В метрологической лаборатории имеются 2 рабочих места с двумя светильниками по 4 лампы в каждом, т.е. имеется 8 ламп. Чтобы выяснить, соответствует ли данное количество светильников и ламп для данной лаборатории проведем проверку соответствия минимальной заданной освещенности при имеющихся 2 светильниках.

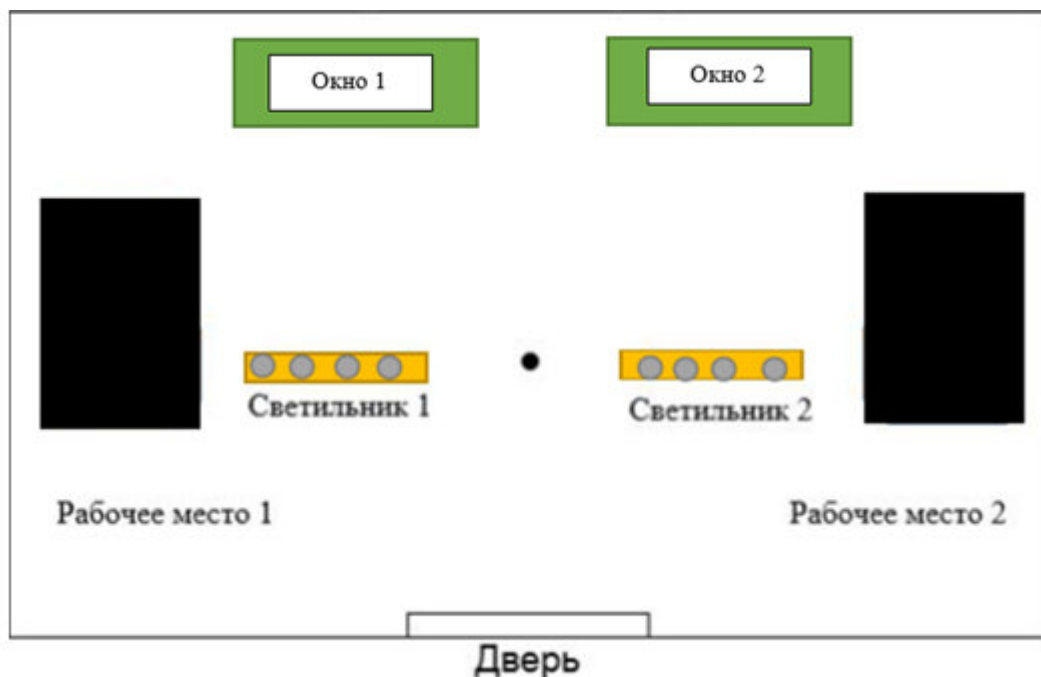


Рисунок 5.1 – Схема помещения лаборатории

$$E = (N \cdot n \cdot \Phi_{\text{л}} \cdot \eta) / (k_z \cdot S_{\text{ос}} \cdot Z) \quad (5.4)$$

Где $S_{\text{ос}}$ – площадь помещения (30 м²);

k_z – коэффициент запаса;

N – количество светильников;

Z – коэффициент неравномерности освещения, $Z = 1,1$;

n – количество ламп в светильнике;

$\Phi_{\text{л}}$ – световой поток выбранной лампы, $\Phi_{\text{л}} = 3750$ лм

η – коэффициент использования, $\eta = 65\%$

$$E = (2 \cdot 2 \cdot 3750 \cdot 0,65) / (1,3 \cdot 30 \cdot 1,1) \approx 217 \text{ лк}$$

При 2 светильниках освещенность примерно равна 217 лк, что не соответствует условиям труда. В следствие чего возникает необходимость, увеличить количество ламп для обеспечения приемлемой освещенности.

Таблица 2 - Исходные данные для расчета количества ламп

Наименование исходных данных	Обозначение	Значение
Площадь лаборатории	S	30м ²
Разряд зрительной работы	-	IV
Норма освещенности	E	450 лк
Величина светового потока	F	3200лк
Коэффициент использования	K _{исп.}	0,65
Коэффициент распространения светового потока	Z	1,1
Коэффициент запаса	K _{зап.}	1,5

Количество ламп рассчитывается по формуле:

$$n = \frac{E \cdot S \cdot K_{\text{зап}} \cdot Z}{K_{\text{исп}} \cdot F} \quad (5.5)$$

где E - нормируемая освещенность,

K_{зап.} - коэффициент запаса,

S - площадь помещения,

Z - коэффициент минимальной освещенности,

K_{исп.} - коэффициент использования светового потока, n - число ламп.

Индекс помещения вычисляется по формуле:

$$i = \frac{A \cdot B}{h \cdot (A + B)} \quad (5.6)$$

где A - длина помещения,

B - ширина помещения,

h - высота светильников над расчетной поверхностью.

$$i = \frac{5 \cdot 6}{2,4 \cdot (5 + 6)} = 1,136$$

При i = 1,136;

K_{исп.} = 0,65

r_{ст} - коэффициент отражения стен 30%,

r_п - коэффициент отражения потолка 50%.

$$n = \frac{450 \cdot 30 \cdot 1,5 \cdot 1,1}{0,65 \cdot 3200} = 10 \text{ шт.}$$

n = 10 ламп

Количество ламп - 10 штук в 2 светильниках (в каждом светильнике по 5 ламп).

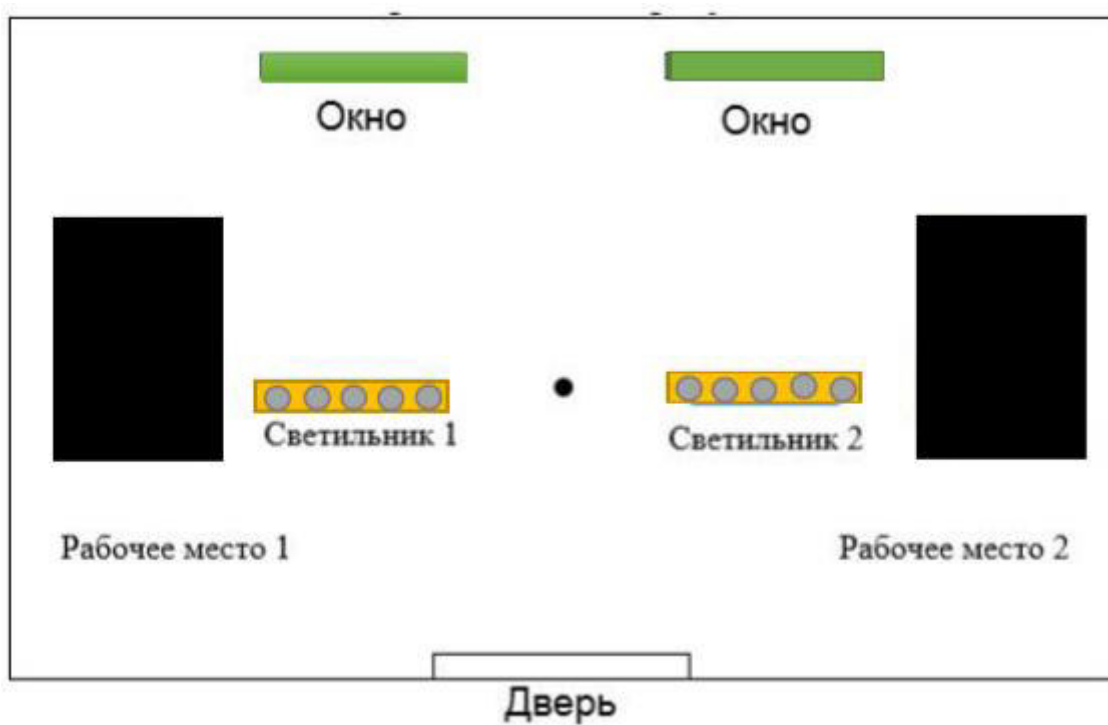


Рисунок 5.2 – Схема помещения лаборатории, согласно нормам.

Заключение

В рамках данного дипломного проекта разработана база данных для метрологического центра. Была проанализирована предметная область и на базе полученных знаний выделены сущности, для каждой из которых были спроектированы и созданы таблицы, а также первичный интерфейс.

В ходе работы были реализованы простые и многотабличные запросы, изучена проблема параллелизма выполняемых запросов, а также обучение работе с различными дампами базы данных.

Для полноты разработки дипломного проекта и её описания, был решён ряд вопросов, основанных на первоначальных целях и задачах дипломного проекта:

- проведено изучение предметной области на примере метрологических центров и служб города Алматы;

- приведены и выполнены всевозможные требования к программному продукту;

- произведен сбор и анализ информации по существующим технологиям в данной предметной области, а также сделан выбор надежных технологий для разработки и проектирования базы данных;

- с помощью объектно-ориентированного языка программирования С# был разработан пользовательский интерфейс, который полностью удовлетворяет всем нормам и требованиям работы для сотрудников метрологических служб;

- произведено проектирование базы данных, которая содержит необходимое количество таблиц и процедур для полнофункционального режима работы программного продукта;

- функциональной основой сервера является: авторизация пользователей, выбор сфер деятельности, работа со списками учёта средств измерений и т.п.;

- произведено экономическое обоснование целесообразности разрабатываемого продукта, в ходе которого можно сделать следующие выводы: цена реализации окупает все затраты на разработку программного продукта; прибыль от реализации программного продукта приблизительно равна 316 000 тенге, что тоже является неплохим доходом при разработках данного типа;

- была рассмотрена ситуация по охране труда и безопасности жизнедеятельности, в ходе которой были проделаны расчёты по улучшению освещения помещений, предназначенных для лабораторных работ метрологических центров.

Таким образом, была проведена разработка программного обеспечения, которое качественно и без ошибок выполняет весь функционал предложенных действий пользователя.

Список литературы

- 1 Мюллер, Р.Д. Проектирование баз данных и UML / Р.Д. Мюллер. - М.:Лори, 2013. - 420 с.
- 2 Пирогов, В. Информационные системы и базы данных: организация и проектирование: Учебное пособие / В. Пирогов. - СПб.: ВHV, 2009. - 528 с.
- 3 Пирогов, В.Ю. Информационные системы и базы данных: организация и проектирование: Учебное пособие / В.Ю. Пирогов. - СПб.: БХВ-Петербург, 2009. - 528 с.
- 4 Преснякова, Г.В. Проектирование интегрированных реляционных баз, данных: Учебное пособие / Г.В. Преснякова. - М.: КДУ , 2007. - 224 с.
- 5 Стружкин, Н.П. Базы данных: проектирование: Учебник для академического бакалавриата / Н.П. Стружкин, В.В. Годин. - Люберцы: Юрайт, 2016. - 477 с.
- 6 Стружкин, Н.П. Базы данных: проектирование. практикум: Учебное пособие для академического бакалавриата / Н.П. Стружкин, В.В. Годин. - Люберцы: Юрайт, 2016. - 291 с.
- 7 Шпак, Ю.А. Проектирование баз данных. Просто как дважды два / Ю.А. Шпак. - М.: Эксмо, 2007. - 304 с.
- 8 Эмблер, С. Рефакторинг баз данных: эволюционное проектирование / С. Эмблер, П. Садаладж. - М.: Вильямс, 2007. - 672 с.
- 9 Аллен, Г. Тейлор SQL для чайников / Аллен Г. Тейлор. - М.: Диалектика, Вильямс, 2015. - 416 с.
- 10 Бен, Форта SQL за 10 минут / Форта Бен. - М.: Диалектика / Вильямс, 2015. - 120 с.
- 11 Бьюли, Алан Изучаем SQL / Алан Бьюли. - М.: Символ-плюс, 2014. - 0 с.
- 12 Грабер, Мартин SQL для простых смертных / Мартин Грабер. - М.: ЛОРИ, 2014. - 378 с.
- 13 Гудсон, Джон Практическое руководство по доступу к данным (+ DVD-ROM) / Джон Гудсон , Роб Стюард. - М.: БХВ-Петербург, 2013. - 304 с.
- 14 Дейт, К. Дж. SQL и реляционная теория. Как грамотно писать код на SQL / К.Дж. Дейт. - М.: Символ-плюс, 2016. - 0 с.
- 15 Джеймс, Р. Грофф SQL. Полное руководство / Джеймс Р. Грофф, Пол Н. Вайнберг, Эндрю Дж. Оппель. - М.: Вильямс, 2014. - 960 с.
- 16 Абрамян, Михаил Visual C# на примерах (+ CD-ROM) / Михаил Абрамян. - М.: БХВ-Петербург, 2012. - 496 с.
- 17 Агуров, Павел C#. Сборник рецептов (+CD-ROM) / Павел Агуров. - М.: БХВ-Петербург, 2007. - 432 с.
- 18 Бишоп, Дж. C# в кратком изложении / Дж. Бишоп, Н. Хорспул. - М.: Бином. Лаборатория знаний, 2011. - 472 с.
- 19 Васильев, Алексей C#. Объектно-ориентированное программирование / Алексей Васильев. - М.: Питер, 2012. - 320 с.

20 Зиборов, В. В. Visual C# 2012 на примерах / В.В. Зиборов. - М.: БХВ-Петербург, 2013. - 480 с.

21 Зиборов, Виктор Visual C# 2010 на примерах / Виктор Зиборов. - М.: "БХВ-Петербург", 2011. - 432 с.

22 Касаткин, А. И. Профессиональное программирование на языке си. Управление ресурсами / А.И. Касаткин. - М.: Высшая школа, 1992. - 432 с.

23 Бекишева А.И. Методические указания к выполнению экономической части дипломной работы для бакалавров специальности 5В0703 - Информационные системы – Алматы: АУЭС; 2013. –24 с.

Приложение А (обязательное)

Техническое задание

Наименование дипломного проекта «Разработка базы данных для метрологического центра».

Целью разработки ПО является полнофункциональная разработка программного обеспечения, которое будет упрощать работу метрологических центров и лабораторий.

Идеология программного обеспечения: данный программный продукт, создается для более простого и быстрого структурирования информационных данных пользователя.

Используемая технология создания ПО - визуальное программирование.

Выбор модели ПО: структурная модель. Позволяет оформлять отдельные блоки программы (повторяющиеся и неповторяющиеся) в процедуры и функции, которые затем могут использоваться в других частях программы. Изменения в коде функций и процедур не влекут за собой изменение других частей кода программы.

Выбор метода разработки структуры ПП: нисходящий метод. Желательно применить нисходящий метод разработки структуры ПП т.к. данный метод наиболее выгоден при использовании в данном ПП. Сначала необходимо построить модульную структуру программы, далее поочередно формируются модули программы, начиная с верхнего уровня (головного), переходя к модулю только в том случае, если уже запрограммирован модуль, который к нему обращается. После того, как все модули программы запрограммированы, производится их поочередное тестирование и отладка в таком же (нисходящем) порядке. При таком порядке разработки программы вся необходимая глобальная информация формируется своевременно, т.е. ликвидируется весьма неприятный источник просчетов при программировании модулей.

Выбор языка программирования: С#. Так как этот язык программирования удобен и прост для создания приложений с графическим интерфейсом. С# имеет широкий набор необходимых функций, которые значительно облегчают создание программы. А также интерфейс на данном языке программирования будет доступен для понимания среднестатистическому человеку.

Выбор базы данных: база данных, созданная с помощью непроцедурного языка SQL. SQL отлично подходит для проектирования и разработки баз данных для данного вида программного обеспечения, разрабатываемого для дипломного проекта. Он ориентирован на операции с данными, представленными в виде логически взаимосвязанных совокупностей таблиц.

Предполагаемая аудитория (тип, возраст и т.д.): люди, возрастной

Продолжение приложения А

категории 18+ с техническим образованием, а также любого пола, образования, мат. положения и т.д.

Общий объем ПП, Мб: программный продукт будет весить около 77 Мб

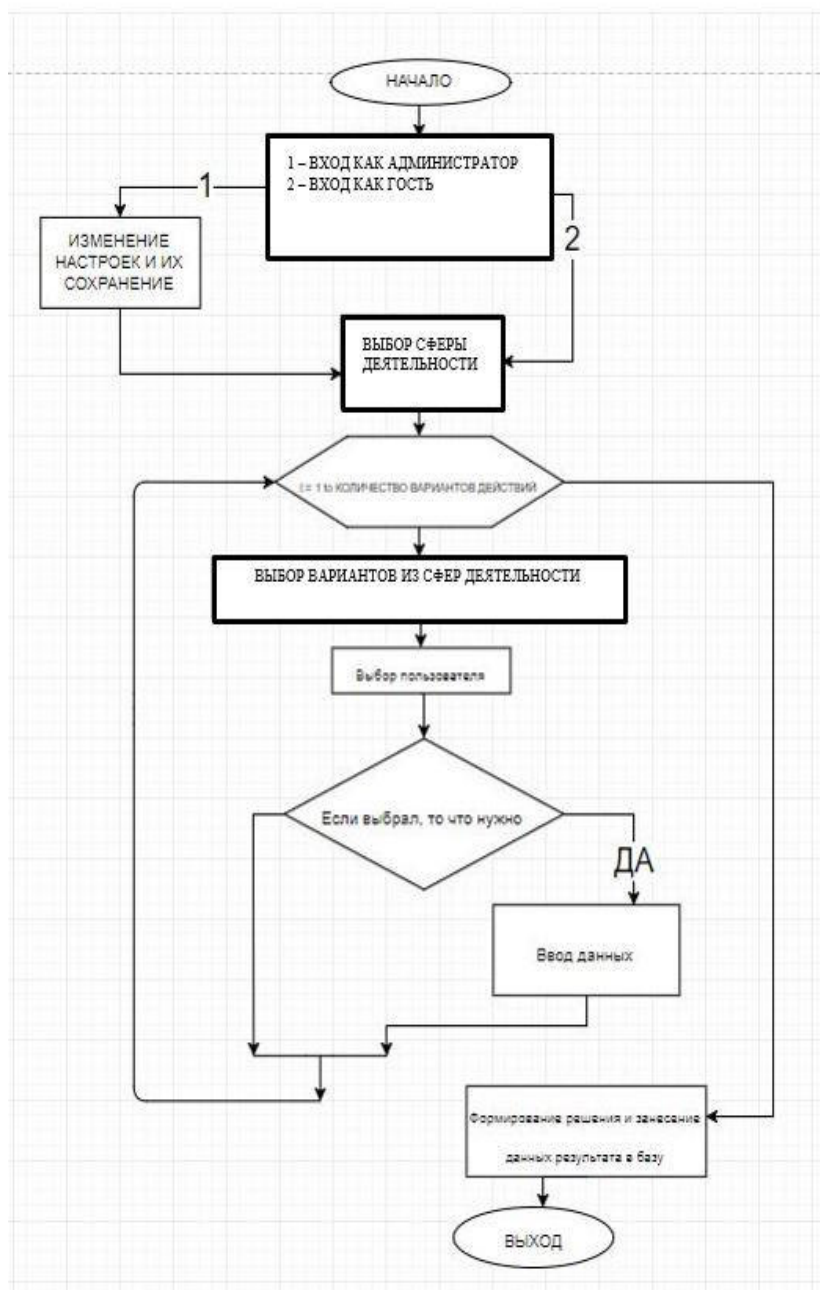


Рисунок А - Общая блок-схема ПО

Объем, состав текстовой и графической информации, звуковой информации, файлов базы данных и т.д.: в программе будут стандартные файлы Microsoft Visual Studio C# - проекта (не более 1МБ), EXE-файл примерно 1 МБ.

Продолжение приложения А

Файл данных пользователей и их результатов (примерно 200 КБ), один файл справки (не более 3 МБ), различные картинки для улучшения дизайна программы (не более 2МБ).

Семантика ПП: с помощью данной программы, возможно, будет легко и быстро создать новую базу возможных операций над данными, отредактировать старые форматы данных и т.д.

Технические требования:

- основной диапазон разрешения мониторов, на которых будет просматриваться ПП: 1920x1080 пикселей;

- минимальное разрешение монитора, в котором будет просматриваться ПП: 800x600 пикселей;

- операционные системы, на которых работает программное обеспечение: семейство поколений Windows, начиная с Windows 7;

- минимальные требования ПК: частота процессора 1.0 ГГц, 512 Мб ОЗУ, 32 Мб видео памяти, USB-порт (3.0), колонки;

- рекомендуемая конфигурация компьютера: частота процессора 2.7 ГГц, 4 Гб ОЗУ, 64 Мб видео памяти, USB – порты (3.0, 2.0), колонки.

Обеспечение защищенности программного продукта: в основе защиты ПП лежат методы шифрования данных в файлах путем создания двоичных файлов вопросов (в формате bin).

Надежность программного обеспечения: должна находиться на высоком уровне. Ни при каких обстоятельствах ПП не должен вызывать сбои. Работа программы не вызовет конфликт с другими программными продуктами, системами и платформами Windows.

Разработка инструкции помощи пользования: необходимо внедрение HELP-файла, который будет содержать необходимую и понятную информацию для работы пользователя с программой. Можно будет ознакомиться с назначением этой программы, изучить правила пользования программой. Файл Help будет содержать наглядные картинки, для понимания программы.

Тестирование и отладка программы будут проводиться в параллельной разработке пробной версии. Изначально данный ПП будет работать в виде пробной версии (в течении 7 дней). Активировать данный ПП можно будет специальным ключом, которой можно будет приобрести у разработчиков.

После завершения создания ПП необходимо будет протестировать его на наличие вредоносного кода с помощью известных программ Kaspersky, NOD32 и т.д. При наличии вредоносного кода в программе необходимо будет обезвредить его. Наличие вредоносного кода не будет умышлено, так как создание данной программы не подразумевает создание вредоносных функций (уничтожение данных, уничтожение реестра и т.д.).

В данном ПП необходимо будет применить стиль Windows 7. ПП будет

Продолжение приложения А

содержать нейтральные цвета (не будет неприятных большинству пользователей ярко-зелёных, ярко-красных цветов). Программа не будет содержать резких анимационных эффектов, способных сильно утомить пользователя. Эстетическая сторона ПП должна быть выбрана таким образом, чтобы она вызвала у потребителя положительную реакцию и интерес.

Приложение Б (обязательное)

Листинг программы

General.cs

```
private void
вСписокКонтроляРаботоспособностиИндикаторовToolStripMenuItem_Click(object sender,
EventArgs e)
{
    DialogResult dialogResult = MessageBox.Show("Вы действительно хотите загрузить
данные из файла EXCEL?", "", MessageBoxButtons.YesNo);
    if (dialogResult == DialogResult.Yes)
    {
        OpenFileDialog ofd = new OpenFileDialog();
        ofd.DefaultExt = "*.xls;*.xlsx";
        ofd.Filter = "Excel 2007(*.xlsx)|*.xlsx|Excel 2003(*.xls)|*.xls";
        ofd.Title = "Выберите документ для загрузки данных";
        if (ofd.ShowDialog() == DialogResult.OK)
        {
            System.IO.FileStream stream =
                System.IO.File.Open(ofd.FileName, System.IO.FileMode.Open,
System.IO.FileAccess.Read);
            Excel.IExcelDataReader IEDR;
            int fileformat = ofd.SafeFileName.IndexOf(".xlsx");
            if (fileformat > -1)
            {
                //2007 format *.xlsx
                IEDR = Excel.ExcelReaderFactory.CreateOpenXmlReader(stream);
            }
            else
            {
                //97-2003 format *.xls
                IEDR = Excel.ExcelReaderFactory.CreateBinaryReader(stream);
            }

            IEDR.IsFirstRowAsColumnNames = false;
            DataSet ds = IEDR.AsDataSet();

            if (dgINDAAlmaty.Visible == true) { dgINDAAlmaty.DataSource = ds.Tables[0]; }
            if (dgINDAktobe.Visible == true) { dgINDAktobe.DataSource = ds.Tables[0]; }
            if (dgINDAstana.Visible == true) { dgINDAstana.DataSource = ds.Tables[0]; }
            if (dgINDTUSM1.Visible == true) { dgINDTUSM1.DataSource = ds.Tables[0]; }
            if (dgINDTUSM2.Visible == true) { dgINDTUSM2.DataSource = ds.Tables[0]; }
            if (dgINDTUSM6.Visible == true) { dgINDTUSM6.DataSource = ds.Tables[0]; }
            if (dgINDTUSM8.Visible == true) { dgINDTUSM8.DataSource = ds.Tables[0]; }
            if (dgINDTUSM10.Visible == true) { dgINDTUSM10.DataSource = ds.Tables[0]; }
            if (dgINDTUSM11.Visible == true) { dgINDTUSM11.DataSource = ds.Tables[0]; }
```

Продолжение приложения Б

```
        if (dgINDTUSM13.Visible == true) { dgINDTUSM13.DataSource = ds.Tables[0]; }
        if (dgINDTUSM14.Visible == true) { dgINDTUSM14.DataSource = ds.Tables[0]; }
        IEDR.Close();
    }
    else
    {
        MessageBox.Show("Вы не выбрали файл для открытия",
            "Загрузка данных...", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
else if (dialogResult == DialogResult.No) { MessageBox.Show("Действие отменено!"); }
}
private void dataGridView_poverka_KeyDown(object sender,
System.Windows.Forms.KeyEventArgs e) {
    if (e.Control && e.KeyCode == Keys.C)
        if (((DataGridView)sender).SelectedCells.Count > 0)
        {
            copySelectedRowsToClipboard((DataGridView)sender);
        }
}
private void dgVSAلمات_KeyDown(object sender, System.Windows.Forms.KeyEventArgs
e){
    if (e.Control && e.KeyCode == Keys.C)
        if (((DataGridView)sender).SelectedCells.Count > 0)
        {
            copySelectedRowsToClipboard((DataGridView)sender);
        }
}
```

Spravochnik.cs

```
public partial class Spravochnik : Form {
    private const string conStr = "Data Source=DESKTOP-TV4BJSK\\MEINE;Initial
Catalog=bd_mc;Integrated Security=True";
    private SqlConnection connection = new SqlConnection(conStr);
    public Spravochnik(){
        InitializeComponent();
        sprGrid.RowHeadersVisible = false; }
    private void radioButton_lica_CheckedChanged(object sender, EventArgs e) {
        connection.Open();
        SqlDataAdapter da = new SqlDataAdapter("exec [dbo].[sp_МатЛица]", connection);
        SqlCommandBuilder cb = new SqlCommandBuilder(da);
```

Продолжение приложения Б

```
DataSet ds = new DataSet();
da.Fill(ds, "bd_mc");
sprGrid.DataSource = ds.Tables[0];
connection.Close(); }

private void radioButton_uchet_CheckedChanged(object sender, EventArgs e)
{ connection.Open();
  SqlDataAdapter da = new SqlDataAdapter("exec [dbo].[sp_УчетРабот]", connection);
  SqlCommandBuilder cb = new SqlCommandBuilder(da);
  DataSet ds = new DataSet();
  da.Fill(ds, "bd_mc");
  sprGrid.DataSource = ds.Tables[0];
  connection.Close();
}

private void radioButton_ci_CheckedChanged(object sender, EventArgs e)
{ connection.Open();
  SqlDataAdapter da = new SqlDataAdapter("exec [dbo].[sp_СредстваИзмерения]",
connection);
  SqlCommandBuilder cb = new SqlCommandBuilder(da);
  DataSet ds = new DataSet();
  da.Fill(ds, "bd_mc");
  sprGrid.DataSource = ds.Tables[0];
  connection.Close();}

private void radioButton_name_CheckedChanged(object sender, EventArgs e)
{ connection.Open();
  SqlDataAdapter da = new SqlDataAdapter("exec [dbo].[sp_СредстваИзмеренияНаименования]", connection);
  SqlCommandBuilder cb = new SqlCommandBuilder(da);
  DataSet ds = new DataSet();
  da.Fill(ds, "bd_mc");
```

Продолжение приложения Б

```
sprGrid.DataSource = ds.Tables[0];
connection.Close();
}
private void radioButton_type_CheckedChanged(object sender, EventArgs e)
{
    connection.Open();
    SqlDataAdapter da = new SqlDataAdapter("exec [dbo].[sp_СредстваИзмеренияТипы]",
connection);
    SqlCommandBuilder cb = new SqlCommandBuilder(da);
    DataSet ds = new DataSet();
    da.Fill(ds, "bd_mc");
    sprGrid.DataSource = ds.Tables[0];
    connection.Close();
}
private void radioButton_num_CheckedChanged(object sender, EventArgs e)
{
    connection.Open();
    SqlDataAdapter da = new SqlDataAdapter("exec [dbo].[sp_СредстваИзмеренияЗав]",
connection);
    SqlCommandBuilder cb = new SqlCommandBuilder(da);
    DataSet ds = new DataSet();
    da.Fill(ds, "bd_mc");

    sprGrid.DataSource = ds.Tables[0];
    connection.Close();
}
```

script_bd.sql

```
SET ANSI_NULLS ON
```

```
GO
```

```
SET QUOTED_IDENTIFIER ON
```

```
GO
```

```
CREATE PROCEDURE [dbo].[SearchAllTables]
```

```
(
```

```
    @SearchStr nvarchar(100)
```

Продолжение приложения Б

```
)
AS
BEGIN
    CREATE TABLE #Results (ColumnName nvarchar(370), ColumnValue nvarchar(3630))

    SET NOCOUNT ON

    DECLARE @TableName nvarchar(256), @ColumnName nvarchar(128), @SearchStr2
nvarchar(110)
    SET @TableName = "
    SET @SearchStr2 = QUOTENAME('%' + @SearchStr + '%','')

    WHILE @TableName IS NOT NULL
    BEGIN
        SET @ColumnName = "
        SET @TableName =
        (
            SELECT MIN(QUOTENAME(TABLE_SCHEMA) + '.' +
QUOTENAME(TABLE_NAME))
            FROM INFORMATION_SCHEMA.TABLES
            WHERE TABLE_TYPE = 'BASE TABLE'
            AND QUOTENAME(TABLE_SCHEMA) + '.' +
QUOTENAME(TABLE_NAME) > @TableName
            AND OBJECTPROPERTY(
                OBJECT_ID(
                    QUOTENAME(TABLE_SCHEMA) + '.' +
QUOTENAME(TABLE_NAME)
                ), 'IsMSShipped'
            ) = 0
        )

        WHILE (@TableName IS NOT NULL) AND (@ColumnName IS NOT NULL)
        BEGIN
            SET @ColumnName =
            (
                SELECT MIN(QUOTENAME(COLUMN_NAME))
                FROM INFORMATION_SCHEMA.COLUMNS
                WHERE TABLE_SCHEMA =
PARSENAME(@TableName, 2)
                AND TABLE_NAME =
PARSENAME(@TableName, 1)
                AND DATA_TYPE IN ('char', 'varchar', 'nchar', 'nvarchar')
                AND QUOTENAME(COLUMN_NAME) >
@ColumnName
            )
        END
    END
```

Продолжение приложения Б

```
IF @ColumnName IS NOT NULL
    BEGIN
        INSERT INTO #Results
        EXEC
        (
            'SELECT ''' + @TableName + '.' + @ColumnName + ''',
LEFT(' + @ColumnName + ', 3630)
            FROM ' + @TableName + ' (NOLOCK) ' +
            ' WHERE ' + @ColumnName + ' LIKE ' + @SearchStr2
        )
    END
END
END

SELECT ColumnName, ColumnValue FROM #Results
END
GO
/***** Object: StoredProcedure [dbo].[sp_Отчет]  Script Date: 15.05.2019 3:00:29 *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[sp_Отчет]
AS
    select [Наименование,тип СИ],[Статус],[Ф.И.О. мат. ответственного лица],[Место
проведения поверки] from ВСТРОЕННЫЕ_СРЕДСТВА
        union
        select [Наименование СИ],[Состояние СИ],[Ф.И.О. мат. ответственного лица],[Место
проведения поверки (калибровки)] from ПЕРЕНОСНЫЕ_СРЕДСТВА
        union
        select [Наименование СИ],[Состояние СИ],[Ф.И.О. мат. ответственного лица],[Место
приписки] from СВАРКА_ОПТИКИ
        union
        select [Наименование,тип СИ],[Статус],[Ф.И.О. мат. ответственного лица],[Место
проведения поверки] from ВСТРОЕННЫЕ_СРЕДСТВА_АСТАНА
        union
        select [Наименование СИ],[Состояние СИ],[Ф.И.О. мат. ответственного лица],[Место
проведения поверки (калибровки)] from ПЕРЕНОСНЫЕ_СРЕДСТВА_АСТАНА
        union
        select [Наименование СИ],[Состояние СИ],[Ф.И.О. мат. ответственного лица],[Место
приписки] from СВАРКА_ОПТИКИ_АСТАНА
        union
        select [Наименование,тип СИ],[Статус],[Ф.И.О. мат. ответственного лица],[Место
проведения поверки] from ВСТРОЕННЫЕ_СРЕДСТВА_АКТОБЕ
        union
```


Продолжение приложения Б

```
select [Наименование СИ],[Состояние СИ],[Ф.И.О. мат. ответственного лица],[Место
проведения поверки (калибровки)] from ПЕРЕНОСНЫЕ_СРЕДСТВА_АКТОБЕ
union
select [Наименование СИ],[Состояние СИ],[Ф.И.О. мат. ответственного лица],[Место
приписки] from СВАРКА_ОПТИКИ_АКТОБЕ
union
select [Наименование,тип СИ],[Статус],[Ф.И.О. мат. ответственного лица],[Место
проведения поверки] from ВСТРОЕННЫЕ_СРЕДСТВА_ТУСМ1
union
select [Наименование СИ],[Состояние СИ],[Ф.И.О. мат. ответственного лица],[Место
проведения поверки (калибровки)] from ПЕРЕНОСНЫЕ_СРЕДСТВА_ТУСМ1
union
select [Наименование СИ],[Состояние СИ],[Ф.И.О. мат. ответственного лица],[Место
приписки] from СВАРКА_ОПТИКИ_ТУСМ1
union
select [Наименование,тип СИ],[Статус],[Ф.И.О. мат. ответственного лица],[Место
проведения поверки] from ВСТРОЕННЫЕ_СРЕДСТВА_ТУСМ2
union
select [Наименование СИ],[Состояние СИ],[Ф.И.О. мат. ответственного лица],[Место
проведения поверки (калибровки)] from ПЕРЕНОСНЫЕ_СРЕДСТВА_ТУСМ2
union
select [Наименование СИ],[Состояние СИ],[Ф.И.О. мат. ответственного лица],[Место
приписки] from СВАРКА_ОПТИКИ_ТУСМ2
union
select [Наименование,тип СИ],[Статус],[Ф.И.О. мат. ответственного лица],[Место
проведения поверки] from ВСТРОЕННЫЕ_СРЕДСТВА_ТУСМ6
СИ],[Ф.И.О. мат. ответственного лица],[Место приписки] from СВАРКА_ОПТИКИ_ТУСМ6
union
select [Наименование,тип СИ],[Статус],[Ф.И.О. мат. ответственного лица],[Место
проведения поверки] from ВСТРОЕННЫЕ_СРЕДСТВА_ТУСМ8
union
select [Наименование СИ],[Состояние СИ],[Ф.И.О. мат. ответственного лица],[Место
проведения поверки (калибровки)] from ПЕРЕНОСНЫЕ_СРЕДСТВА_ТУСМ8
union
select [Наименование СИ],[Состояние СИ],[Ф.И.О. мат. ответственного лица],[Место
приписки] from СВАРКА_ОПТИКИ_ТУСМ8
union
select [Наименование,тип СИ],[Статус],[Ф.И.О. мат. ответственного лица],[Место
проведения поверки] from ВСТРОЕННЫЕ_СРЕДСТВА_ТУСМ10
union
select [Наименование СИ],[Состояние СИ],[Ф.И.О. мат. ответственного лица],[Место
проведения поверки (калибровки)] from ПЕРЕНОСНЫЕ_СРЕДСТВА_ТУСМ10
union
select [Наименование СИ],[Состояние СИ],[Ф.И.О. мат. ответственного лица],[Место
приписки] from СВАРКА_ОПТИКИ_ТУСМ10
```

Продолжение приложения Б

```
union
select [Наименование,тип СИ],[Статус],[Ф.И.О. мат. ответственного лица],[Место
проведения поверки] from ВСТРОЕННЫЕ_СРЕДСТВА_ТУСМ11
union
select [Наименование СИ],[Состояние СИ],[Ф.И.О. мат. ответственного лица],[Место
проведения поверки (калибровки)] from ПЕРЕНОСНЫЕ_СРЕДСТВА_ТУСМ11
union
select [Наименование СИ],[Состояние СИ],[Ф.И.О. мат. ответственного лица],[Место
приписки] from СВАРКА_ОПТИКИ_ТУСМ11
union
select [Наименование,тип СИ],[Статус],[Ф.И.О. мат. ответственного лица],[Место
проведения поверки] from ВСТРОЕННЫЕ_СРЕДСТВА_ТУСМ13
union
select [Наименование СИ],[Состояние СИ],[Ф.И.О. мат. ответственного лица],[Место
проведения поверки (калибровки)] from ПЕРЕНОСНЫЕ_СРЕДСТВА_ТУСМ13
union
select [Наименование СИ],[Состояние СИ],[Ф.И.О. мат. ответственного лица],[Место
проведения поверки (калибровки)] from ПЕРЕНОСНЫЕ_СРЕДСТВА_ТУСМ14
union
select [Наименование СИ],[Состояние СИ],[Ф.И.О. мат. ответственного лица],[Место
приписки] from СВАРКА_ОПТИКИ_ТУСМ14
GO
/***** Object: StoredProcedure [dbo].[sp_СредстваИзмеренияЗав]   Script Date: 15.05.2019
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[sp_СредстваИзмеренияЗав]
AS
select [Заводской номер или присвоенный номер] from ВСТРОЕННЫЕ_СРЕДСТВА
union
select [Заводской номер] from ПЕРЕНОСНЫЕ_СРЕДСТВА
union
select [Заводской номер] from СВАРКА_ОПТИКИ
union
select [Заводской номер] from УЧЕТ_ЭНЕРГИИ
union
select [Заводской номер или присвоенный номер] from
ВСТРОЕННЫЕ_СРЕДСТВА_АСТАНА
union
select [Заводской номер] from ПЕРЕНОСНЫЕ_СРЕДСТВА_АСТАНА
union
select [Заводской номер] from СВАРКА_ОПТИКИ_АСТАНА
union
select [Заводской номер] from УЧЕТ_ЭНЕРГИИ_АСТАНА
```

Приложение В (обязательное)

Акт внедрения

ТОО "Оптиктелеком СтройСервис" Республика Казахстан, 050010, г. Алматы, ул. Кармысова, д. 76А/2 тел.: (727) 266 40 02, 266 40 03, факс: (727) 250 73 27 A-5@optictelcom.kz / www.optictelcom.kz		
г.Алматы		Утверждаю Генеральный директор ТОО «Оптиктелеком СтройСервис» Жигулин О.А. «17» мая 2019г.
АКТ ВНЕДРЕНИЯ		
<p>Настоящий Акт составлен о том, что результат выпускного дипломного проекта студентки Некоммерческого акционерного общества «Алматинского университета энергетики и связи» группы ВТ-15-2 очной формы обучения Беленковой А.С. на тему «Разработка базы данных для метрологического центра» внедрены в метрологическую службу ТОО «Оптиктелеком СтройСервис» и используется в компании для оптимизации работы метрологической службы.</p> <p>Использование выпускного дипломного проекта Беленковой А.С. обеспечивает быстрый ввод и учёт средств измерений, находящихся в эксплуатации, а также работ, проведенных метрологической службой.</p>		
Начальник метрологической службы		Панаилидис Д.Г.