**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**Рязанский государственный радиотехнический университет**

**Факультет вычислительной техники К ЗАЩИТЕ**

**Кафедра ЭВМ** Заведующий кафедрой ЭВМ

**Направление 02.03.03 «Математическое** д.т.н., профессор

**обеспечение и администрирование** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Б.В.Костров

**информационных систем»** «\_\_\_»\_\_\_\_\_\_\_\_\_\_ 2018 г.

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к выпускной квалификационной работе бакалавра на тему

**«Автоматизация системы метрологического контроля»**

**Студент**  (Краснобаев К. А.)

**Руководитель работы**  (Бабаев С. И.)

**“\_\_\_\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2018 г.**

**АННОТАЦИЯ**

Дипломная работа посвящена разработке клиентского приложения для учета средств измерений и контроля метрологических процедур.

Работа состоит из пояснительной записки, состоящей из 7 частей, и приложений к ней. Объем пояснительной записки составляет 60 страниц, в который входит 50 рисунков.

Во введении рассматривается роль информационных систем в настоящее время. Также в работе приводится постановка задачи, осуществляется анализ проблемы и аналогов системы, проводится обоснование выбора средств реализации: Visual Studio 2015 и СУБД MS SQL Server 2014.

В специальных частях описывается проектирование базы данных и разработка клиентского приложения для работы с ней. Также пояснительная записка содержит программную документацию и тестирование разработанного программного обеспечения.

Приложения к пояснительной записке содержат сценарий создания объектов базы данных, код хранимых процедур на языке SQL, а также исходный код разработанного клиентского приложения на языке С#.

**ANNOTATION**

The thesis is devoted to the development of a client application for accounting of measuring instruments and control of metrological procedures.

The thesis work consists of an explanatory note, consisting of 7 parts, and annexes to it. The volume of the explanatory note is 60 pages, which includes 50 images.

The introduction examines the role of information systems at the present time. The thesis work also contains a statement of the problem, an analysis of the problem and analogues of the system, an argumentation of the choice of implementation tools: Visual Studio 2015 and MS SQL Server 2014.

The special parts describe database design and development of the client application to work with it. Also explanatory note contains software documentation and testing of the developed software.

The annexes to the explanatory note contain the script for creating database objects, the code of stored procedures in SQL, as well as the source code of the developed client application in C#.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 8](#_Toc517039921)

[1 Технико-экономическое обоснование 10](#_Toc517039922)

[1.1 Общий анализ проблемы 10](#_Toc517039923)

[1.2 Назначение разрабатываемого программного обеспечения 10](#_Toc517039924)

[1.3 Анализ аналогичных решений 11](#_Toc517039925)

[1.4 Обоснование проведения разработки программного обеспечения 12](#_Toc517039926)

[2 Постановка задачи 13](#_Toc517039927)

[3 Выбор технологий и средств проектирования 14](#_Toc517039928)

[3.1 Выбор методологии разработки информационной системы 14](#_Toc517039929)

[3.2 Выбор СУБД 15](#_Toc517039930)

[3.3 Выбор инструментов программирования 16](#_Toc517039931)

[4 Разработка серверной части информационной системы 18](#_Toc517039932)

[4.1 Инфологическое проектирование БД 18](#_Toc517039933)

[4.1.1 Выявление сущностей и связей 18](#_Toc517039934)

[4.1.2 Построение ER-диаграммы 18](#_Toc517039935)

[4.2 Даталогическое проектирование БД 21](#_Toc517039936)

[4.2.1 Переход от ER-диаграммы к предварительным отношениям 21](#_Toc517039937)

[4.2.2 Заполнение предварительных отношений атрибутами 24](#_Toc517039938)

[4.2.3 Проверка предварительных отношений на соответствие нормальным формам 25](#_Toc517039939)

[4.2.4 Построение схемы данных 31](#_Toc517039940)

[4.2.5 Задание частных ограничений целостности данных 32](#_Toc517039941)

[5 Разработка клиентской части информационной системы 35](#_Toc517039942)

[5.1 Организация взаимодействия приложения с базой данных 35](#_Toc517039943)

[5.2 Разработка интерфейса пользователя 36](#_Toc517039944)

[5.2.1 Проектирование прототипа интерфейса 36](#_Toc517039945)

[5.2.2 Реализация интерфейса 38](#_Toc517039946)

[5.2.3 Разработка отчетов 44](#_Toc517039947)

[5.3 Разработка установочного пакета 46](#_Toc517039948)

[6 Разработка программной документации 49](#_Toc517039949)

[6.1 Руководство программиста 49](#_Toc517039950)

[6.1.1 Общие сведения и структура программы 49](#_Toc517039951)

[6.1.2 Установка и настройка программы 49](#_Toc517039952)

[6.2 Руководство оператора (пользователя) 50](#_Toc517039953)

[6.2.1 Назначение и условия применения программы 50](#_Toc517039954)

[6.2.2 Работа с программой 50](#_Toc517039955)

[6.2.3 Сообщения оператору 53](#_Toc517039956)

[7 Программа и методика испытаний 55](#_Toc517039957)

[7.1 Методы испытаний 55](#_Toc517039958)

[7.2 Результаты испытаний 57](#_Toc517039959)

[ЗАКЛЮЧЕНИЕ 59](#_Toc517039960)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 60](#_Toc517039961)

[ПРИЛОЖЕНИЕ А 1](#_Toc517039962)

[ПРИЛОЖЕНИЕ Б 1](#_Toc517039963)

[ПРИЛОЖЕНИЕ В 1](#_Toc517039964)

# ВВЕДЕНИЕ

В современном мире практически во всех сферах деятельности человека интенсивно используются информационные технологии. Под давлением активно развивающихся технологий происходит внедрение информационных систем в те области, где они не используются или слабо развиты. Такое внедрение позволяет уменьшить время на обработку данных, сократить финансовые затраты и увеличить производительность труда.

Под информационной системой понимают предназначенную для хранения, поиска и обработки информации совокупность средств, методов, а также технических и человеческих ресурсов.

Основным назначением любой информационной системы является удовлетворение определенных информационных потребностей в рамках конкретной предметной области.

Благодаря повсеместному использованию компьютеров стало возможным создание автоматизированных информационных систем, позволяющих переложить основную деятельность по работе с информацией на технические средства.

Под автоматизированной информационной системой принято понимать совокупность информационных массивов, технических, программных и языковых средств, предназначенных для сбора, хранения, поиска, обработки и выдачи данных по запросам пользователей.

Целью данной работы является разработка информационной системы метрологического контроля. Автоматизация учета средств измерений и контроля метрологических процедур представляет актуальную проблему для многих предприятий и организаций в различных областях, таких как промышленность, медицина или наука.

Для достижения поставленной цели определены следующие задачи:

* автоматизировать учет средств измерений;
* автоматизировать ведение журналов метрологических процедур (поверок, калибровок) и ремонта;
* автоматизировать формирование отчетности и документации;
* автоматизировать контроль над своевременным прохождением поверок средствами измерений.

# Технико-экономическое обоснование

## Общий анализ проблемы

Создание и внедрение автоматизированной информационной системы, как и ее поддержка, является серьезной и актуальной проблемой для большинства предприятий и организаций независимо от сферы деятельности.

Конкретные задачи, решение которых обеспечивает информационная система, зависят от той прикладной области, для которой эта система предназначена. Для внедрения системы в определенную область деятельности проводится большой объем работы по аналитике, моделированию и анализу бизнес-процессов, протекающих в конкретной организации.

Однако, независимо от специфики предметной области, существуют два основных аспекта, которые являются общими для всех информационных систем.

Во-первых, в основе любой системы лежит среда хранения и доступа к данным. Такая среда представляет собой единую базу данных, доступ к которой осуществляется с помощью системы управления базами данных (СУБД), позволяющей систематизировать и организовывать данные для их хранения и обработки.

Во-вторых, ориентация любой информационной системы на конечного пользователя требует наличия клиентского приложения, обладающего простым и удобным интерфейсом. Такое приложение должно предоставлять пользователям, в том числе с базовыми навыками работы с компьютером, необходимый функционал, но в то же время исключать возможность выполнения каких-либо лишних действий.

## Назначение разрабатываемого программного обеспечения

Разрабатываемая в рамках данной работы информационная система предназначена для автоматизации учета средств измерений и контроля метрологических процедур. Не имея привязки к конкретному предприятию, система является гибкой и может быть адаптирована к любой организации.

Наличие единой базы данных в данной сфере обеспечит возможность хранения всей необходимой информации в электронном виде, снижая бумажный документооборот. Клиентское приложение даст пользователям возможность доступа к журналам метрологических процедур, информации о средствах измерений, содержащейся в базе данных, а также обеспечит удобное формирование отчетности и документации.

## Анализ аналогичных решений

На данный момент на рынке существует несколько вариантов программных решений для данной предметной области. Рассмотрим две системы, которые являются схожими по назначению с разрабатываемой.

*«Метролог СИ 2.3»* от компании *TOP\_RIDER* [3]*.*Основные возможности данного программного обеспечения: сбор и обработка данных о наличии, состоянии и движении средств измерений;получение статистической информации;составление планов и графиков, а также списков различной структуры, бирок, счётов, накладных, используя фильтры или группированный поиск;ведение истории изменения средств измерений, включая эксплуатации и ремонты; быстрый доступ к электронным документам средств измерений.

*«Метролог»* от фирмы *VISComp* [4]*.* Основными возможностями данной системы являются: централизация хранения данных о средствах измерений, а также отчетной, справочной и нормативной информации; метрологический учёт средств измерений; планирование метрологического контроля; формирование отчетной информации.

Рассмотренные выше системы являются платными и имеют бесплатные демо-версии.

## Обоснование проведения разработки программного обеспечения

Рассмотрев аналогичные программные решения, можно сделать вывод, что они не подходят для решения поставленных задач, так как перегружены избыточными функциями и являются дорогостоящими.

Коробочный программный продукт не может быть использован ввиду уникальности предметной области.

Таким образом, создание автоматизированной системы метрологического контроля, позволяющей вести учет средств измерений, журналы метрологических процедур, а также формировать отчетность, является актуальной задачей.

# Постановка задачи

Целью данной работы является разработка базы данных и клиентского приложения для ведения метрологического контроля. Для достижения первой поставленной цели необходимо:

* провести инфологическое проектирование базы данных;
* провести даталогическое проектирование базы данных;
* разработать основные объекты структуры базы данных;
* разработать сценарий развертывания базы данных.

Для достижения второй поставленной цели необходимо:

* проанализировать требования к клиентской части информационной системы;
* разработать прототип интерфейса и дерево форм;
* разработать и реализовать интерфейс пользователя на языке программирования высокого уровня;
* сформировать программную документацию;
* разработать установочный пакет.

# Выбор технологий и средств проектирования

## Выбор методологии разработки информационной системы

Методология разработки ПО **–** набор методов и критериев оценки, которые используются для постановки задачи, планирования, контроля и в конечном итоге – для достижения поставленной цели проекта [5].

Можно выделить следующие основные методологии разработки ПО:

* ГОСТ (каскадная модель);
* Rational Unified Process (RUP) ;
* Agile.

Для выбора подходящей методологии необходимо сравнить подходы по основным критериям.

Результаты анализа критериев представлены в таблице 1.

Таблица 1 – Подбор методологии по критериям

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Критерий | Результат анализа | ГОСТ | RUP | Agile |
| Масштаб системы | Небольшой | + | - | + |
| Сроки проекта | Средние (1 год) | + | + | + |
| Полнота и определенность требований к системе в начале проектирования | Высокая | + | + | + |
| Вероятность изменения требований к проекту и технологии работ | Низкая | +- | + | + |

Продолжение таблицы 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Необходимость промежуточных рабочих версий продукта | Отсутствует | + | + | + |
| Необходимость сопровождения системы | Присутствует | + | +- | - |

Разрабатываемая система имеет небольшие сроки и масштабы, что делает невыгодным в экономическом плане использование методологии RUP.

В начале проектирования имеются вполне определенные требования, которые не меняются в ходе разработки. Необходимость в промежуточных версиях продукта отсутствует.

При выборе между Agile и каскадной моделью, основополагающим критерием становится необходимость в полном комплекте технической и пользовательской документации.

Таким образом, учитывая, что система может использоваться не только в частных, но и в государственных предприятиях, наиболее подходящей является каскадная модель.

## Выбор СУБД

Система управления базами данных (СУБД) – специальное программное обеспечение, предназначенное для создания баз данных и работы с ними [6]. Главной функцией СУБД является манипулирование данными, находящимися в памяти (как оперативной, так и внешней). Любая СУБД отвечает за резервное копирование и восстановление информации после сбоев и обеспечивает поддержку языков баз данных.

В настоящее время на рынке существует множество СУБД. Среди основных можно выделить такие как MySQL, Oracle Database, PostgreSQL, Microsoft SQL Server.

Для выполнения поставленных задач была выбрана Microsoft SQL Server. Выбор данной СУБД обусловлен тем, что она обладает такими необходимыми для СУБД качествами, как надежность, стабильность, производительность и масштабируемость.

Высокая надежность обеспечивается в первую очередь за счёт отказоустойчивой кластеризации. Отказоустойчивая кластеризация обеспечивает защиту не только базы данных, но и сервера. Данная функция позволяет предотвратить любую потерю данных, что является одним из важнейших аспектов.

Следующее достоинство Microsoft SQL Server – это ее возможности по масштабированию и высокая производительность. Увеличенный до 8 КБ размер страниц способствует быстрому извлечению данных, позволяет использовать строки и столбцы большего размера, что открывает возможность эффективного хранения сложных, подробных данных. Также Microsoft SQL Server имеет функции партиционирования [7]. Эта технология позволяет разбивать большие таблицы на несколько элементов, которые привязаны к разным файл-группам. В результате данные физически размещаются на нескольких жестких дисках, и таким образом операции чтения/записи распараллеливаются. Данная технология совершенно прозрачна для пользовательского приложения, и ее работа осуществляется автоматически средствами самой СУБД.

Особое внимание в Microsoft SQL Server уделено вопросам безопасности. В СУБД реализована поддержка современных криптоалгоритмов.

## Выбор инструментов программирования

В качестве среды разработки была выбрана интегрированная среда разработки программного обеспечения MS Visual Studio, поскольку имеется множество инструментов взаимодействия между ней и используемой СУБД MS SQL SERVER. В качестве языка программирования был выбран язык C#.

Microsoft Visual Studio – линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework и Silverlight.

# Разработка серверной части информационной системы

## Инфологическое проектирование БД

### Выявление сущностей и связей

Инфологическое проектирование основано на ER-методе.

Суть ER-метода: построение диаграммы, отображающей в графической форме основные объекты предметной области, связи между ними и определение характеристик этих связей [8].

В данной предметной области можно выделить следующие сущности:

* Прибор;
* Сотрудник;
* Подразделение;
* Марка;
* Организация.

Между сущностями установлены следующие связи:

* Прибор имеет Марку;
* Прибор находится в Подразделении;
* Прибор закреплен за Сотрудником;
* Прибор поверяется Организацией;
* Прибор калибруется Организацией;
* Прибор ремонтируется Организацией.

### Построение ER-диаграммы

Диаграммы должны включать все выделенные сущности и связи между ними. На данном этапе должны быть определены характеристики связи: степень связи и класс принадлежности сущности к связи. Общая ER-диаграмма представлена на рисунке 7.

1. Прибор имеет Марку (Рисунок 1).

Степень связи:

* прибор может иметь только одну марку;
* может быть несколько приборов одной марки.

Класс принадлежности:

* прибор обязательно имеет марку;
* марка не обязательно принадлежит прибору.

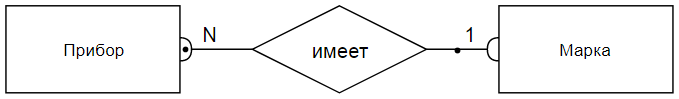


Рисунок 1 – ER-диаграмма «Прибор имеет Марку»

1. Прибор закреплен за Сотрудником (Рисунок 2).

Степень связи:

* прибор закрепляется за одним сотрудником;
* за одним сотрудником может быть закреплено несколько приборов.

Класс принадлежности:

* прибор обязательно закрепляется за сотрудником;
* сотрудник может не иметь закрепленных за ним приборов.

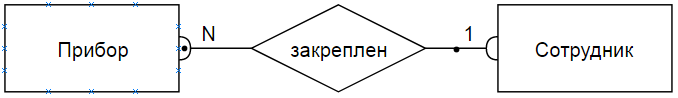


Рисунок 2 – ER-диаграмма «Прибор закреплен за Сотрудником»

1. Прибор находится в Подразделении (Рисунок 3).

Степень связи:

* прибор находится в одном подразделении;
* в одном подразделении может находиться несколько приборов.

Класс принадлежности:

* прибор обязательно находится в подразделении;
* в подразделении может не быть приборов.

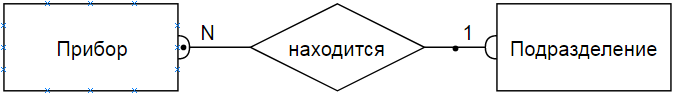


Рисунок 3 – ER-диаграмма «Прибор находится в Подразделении»

1. Прибор поверяется Организацией (Рисунок 4).

Степень связи:

* прибор может поверяться во многих организациях;
* организация может поверять много приборов.

Класс принадлежности:

* прибор не обязательно поверяется организацией;
* организация не обязательно поверяет прибор.

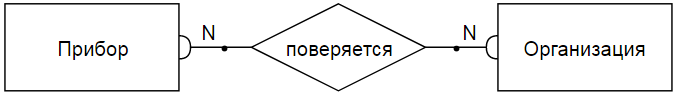


Рисунок 4 – ER-диаграмма «Прибор поверяется Организацией»

1. Прибор калибруется Организацией (Рисунок 5).

Степень связи:

* прибор может калиброваться во многих организациях;
* организация может калибровать много приборов.

Класс принадлежности:

* прибор не обязательно калибруется организацией;
* организация не обязательно калибрует прибор.

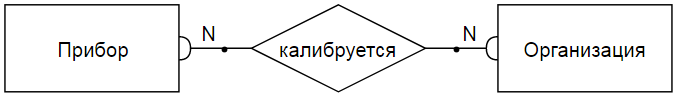


Рисунок 5 – ER-диаграмма «Прибор калибруется Организацией»

1. Прибор ремонтируется Организацией (Рисунок 6).

Степень связи:

* прибор может ремонтироваться во многих организациях;
* организация может ремонтировать много приборов.

Класс принадлежности:

* прибор не обязательно ремонтируется организацией;
* организация не обязательно ремонтирует прибор.

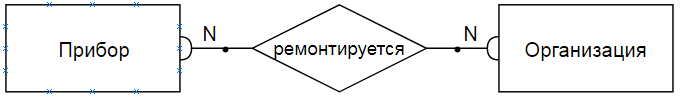


Рисунок 7 – ER-диаграмма «Прибор ремонтируется Организацией»

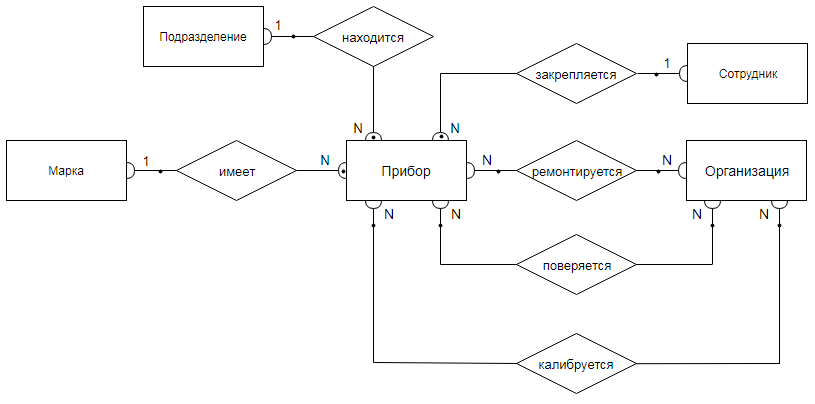


Рисунок 7 – Общая ER-диаграмма

## Даталогическое проектирование БД

### Переход от ER-диаграммы к предварительным отношениям

Даталогическое проектирование является проектированием логической структуры БД, что означает определение всех информационных единиц и связей между ними, задание их имен и типов [9].

Для реляционной БД проектирование логической структуры заключается в том, чтобы разбить всю информацию по таблицам, определив состав полей для каждой из этих таблиц.

Правила формирования отношений по ER-диаграммам.

Правило 1:

Если степень бинарной связи 1:1 и класс принадлежности обеих сущностей к связи обязательный, то формируют одно отношение. Ключом может быть ключ любой сущности.

Правило 2:

Если связь 1:1 и класс принадлежности одной сущности обязательный, а другой необязательный, то формируется два отношения, по одному для каждой сущности, где ключом является ключ соответствующей сущности. Кроме этого ключ сущности с необязательным классом принадлежности добавляется в качестве атрибута в отношение для сущности с обязательным классом принадлежности.

Правило 3:

Если степень связи 1:1 и класс принадлежности обеих сущностей к связи необязательный, то формируют три отношения: два для сущностей и одно для связи. В первых двух отношениях ключом является ключ соответствующей сущности. Отношение для связи должно содержать ключи обеих сущностей. В этом отношении ключом может быть ключ любой сущности.

Правило 4:

Если степень бинарной связи 1:N и класс принадлежности N-связной сущности обязательный, то формируется два отношения, по одному для каждой сущности. Ключ отношения - это ключ соответствующей сущности. Кроме этого в отношение для N-связной сущности добавляется в качестве атрибута ключ односвязной сущности. Класс принадлежности односвязной сущности на результат не влияет.

Правило 5:

Если степень бинарной связи 1:N, а класс принадлежности N-связной сущности необязательный, то формируют три отношения: по одному для каждой сущности и одно для связи. Ключом в первых двух отношениях является ключ соответствующей сущности. Отношение для связи должно содержать ключи обеих сущностей. Ключ этого отношения – это ключ N-связной сущности. Класс принадлежности односвязной сущности на результат не влияет.

Правило 6:

Если степень бинарной связи N:M, то формируется три отношения: два отношения для сущностей, где ключом является ключ соответствующей сущности и одно отношение для связи, где первичный ключ состоит из ключей обеих сущностей. Класс принадлежности сущности к связи на результат не влияет.

Правило 7:

При наличии тернарной связи необходимо использовать 4 отношения: три для каждой сущности, где ключом является ключ соответствующей сущности и одно для связи, где ключ содержит ключи всех сущностей. При наличии n-ой связи строится n+1 отношение.

Прибор имеет Марку. По правилу 4 получаем отношения:

* Прибор (ЗаводскойНомер, МаркаПрибора);
* Марка (МаркаПрибора).

Прибор закреплен за Сотрудником. По правилу 4 получаем отношения:

* Прибор (ЗаводскойНомер, МаркаПрибора, КодСотрудника);
* Сотрудник (КодСотрудника).

Прибор находится в Подразделении. По правилу 4 получаем отношения:

* Прибор (ЗаводскойНомер, МаркаПрибора, КодСотрудника, МестоУстановки);
* Подразделение (НазваниеПодразделения).

Прибор поверяется Организацией. По правилу 6 получаем отношения:

* Прибор (ЗаводскойНомер, МаркаПрибора, КодСотрудника, МестоУстановки);
* Организация (НазваниеОрганизации);
* Поверка (ЗаводскойНомер, НазваниеОрганизации, НомерПротокола).

Прибор калибруется Организацией. По правилу 6 получаем отношения:

* Прибор (ЗаводскойНомер, МаркаПрибора, КодСотрудника, МестоУстановки);
* Организация (НазваниеОрганизации);
* Калибровка (ЗаводскойНомер, НазваниеОрганизации, НомерПротокола).

Прибор ремонтируется Организацией. По правилу 6 получаем отношения:

* Прибор (ЗаводскойНомер, МаркаПрибора, КодСотрудника, МестоУстановки);
* Организация (НазваниеОрганизации);
* Ремонт (ЗаводскойНомер, НазваниеОрганизации, НомерПротокола).

### Заполнение предварительных отношений атрибутами

После распределения атрибутов получаем следующие отношения:

* Прибор (ЗаводскойНомер, МаркаПрибора, КодСотрудника, МестоУстановки, ДатаВыпуска, ДатаВводаВЭксплуатацию, Состояние);
* Организация (НазваниеОрганизации, НомерЛицензии);
* Сотрудник (КодСотрудника, ФИО, Должность);
* Марка (МаркаПрибора, Тип, Производитель, ПериодичностьПоверки, СрокЭксплуатации);
* Поверка (НазваниеОрганизации, ЗаводскойНомер, НомерПротокола, Дата, ФИОПоверяющего, Результат);
* Калибровка (НазваниеОрг, ЗаводскойНомер, НомерПротокола, Дата, ФИОКалибрующего, Результат);
* Подразделение (НазваниеПодразделения, Описание);
* Ремонт (НазваниеОрг, ЗаводскойНомер, НомерПротокола, ДатаНач, ДатаКон, ФИОРемонтирующего).

### Проверка предварительных отношений на соответствие нормальным формам

Для отношения Прибор (Рисунок 8):

* Отношение находится в 2НФ, так как каждый неключевой атрибут функционально полно зависит от первичного ключа;
* Отношение находится в 3НФ, так как в нём нет транзитивных зависимостей не ключевых атрибутов от любого потенциального ключа, в частности первичного;
* Отношение находится в БКНФ, так как детерминантом всех функциональных зависимостей является первичный ключ.

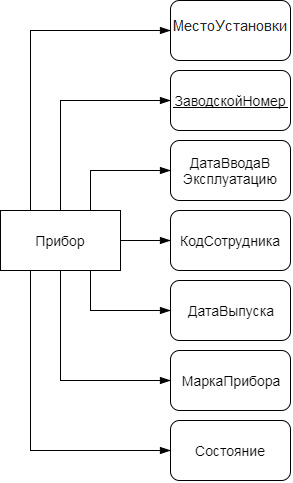


Рисунок 8 – Отношение Прибор

Для отношения Организация (Рисунок 9):

* Отношение находится в 2НФ, так как каждый неключевой атрибут функционально полно зависит от первичного ключа;
* Отношение находится в 3НФ, так как в нём нет транзитивных зависимостей не ключевых атрибутов от любого потенциального ключа, в частности первичного;
* Отношение находится в БКНФ, так как детерминантом всех функциональных зависимостей является первичный ключ.



Рисунок 9 – Отношение Организация

Для отношения Сотрудник (Рисунок 10):

* Отношение находится в 2НФ, так как каждый неключевой атрибут функционально полно зависит от первичного ключа;
* Отношение находится в 3НФ, так как в нём нет транзитивных зависимостей не ключевых атрибутов от любого потенциального ключа, в частности первичного;
* Отношение находится в БКНФ, так как детерминантом всех функциональных зависимостей является первичный ключ.

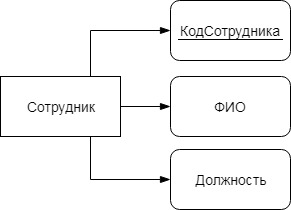


Рисунок 10 – Отношение Сотрудник

Для отношения Марка (Рисунок 11):

* Отношение находится в 2НФ, так как каждый неключевой атрибут функционально полно зависит от первичного ключа;
* Отношение находится в 3НФ, так как в нём нет транзитивных зависимостей не ключевых атрибутов от любого потенциального ключа, в частности первичного;
* Отношение находится в БКНФ, так как детерминантом всех функциональных зависимостей является первичный ключ.

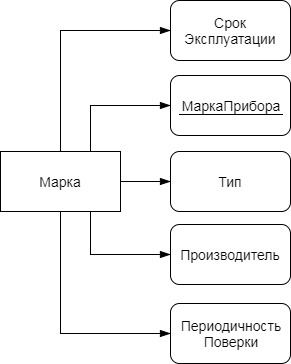


Рисунок 11 – Отношение Марка

Для отношения Подразделение (Рисунок 12):

* Отношение находится в 2НФ, так как каждый неключевой атрибут функционально полно зависит от первичного ключа;
* Отношение находится в 3НФ, так как в нём нет транзитивных зависимостей не ключевых атрибутов от любого потенциального ключа, в частности первичного;
* Отношение находится в БКНФ, так как детерминантом всех функциональных зависимостей является первичный ключ.



Рисунок 12 – Отношение Подразделение

Для отношения Поверка (Рисунок 13):

* Отношение находится в 2НФ, так как каждый неключевой атрибут функционально полно зависит от первичного ключа;
* Отношение находится в 3НФ, так как в нём нет транзитивных зависимостей не ключевых атрибутов от любого потенциального ключа, в частности первичного;
* Отношение находится в БКНФ, так как детерминантом всех функциональных зависимостей является первичный ключ.

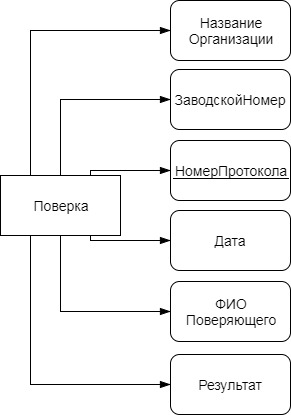


Рисунок 13 – Отношение Поверка

Для отношения Калибровка (Рисунок 14):

* Отношение находится в 2НФ, так как каждый неключевой атрибут функционально полно зависит от первичного ключа;
* Отношение находится в 3НФ, так как в нём нет транзитивных зависимостей не ключевых атрибутов от любого потенциального ключа, в частности первичного;
* Отношение находится в БКНФ, так как детерминантом всех функциональных зависимостей является первичный ключ.

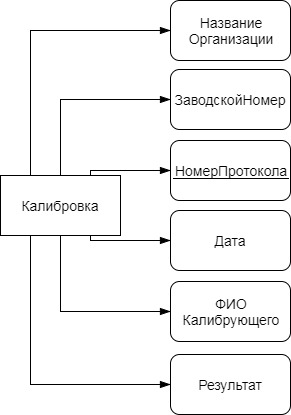


Рисунок 14 – Отношение Калибровка

Для отношения Ремонт (Рисунок 15):

* Отношение находится в 2НФ, так как каждый неключевой атрибут функционально полно зависит от первичного ключа;
* Отношение находится в 3НФ, так как в нём нет транзитивных зависимостей не ключевых атрибутов от любого потенциального ключа, в частности первичного;
* Отношение находится в БКНФ, так как детерминантом всех функциональных зависимостей является первичный ключ.

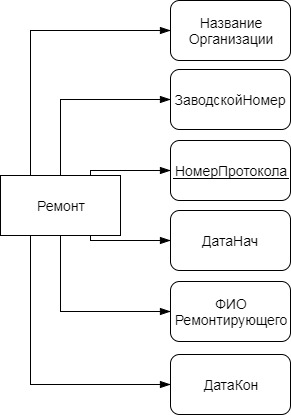


Рисунок 15 – Отношение Ремонт

### Построение схемы данных

Схема данных представлена на рисунке 16.

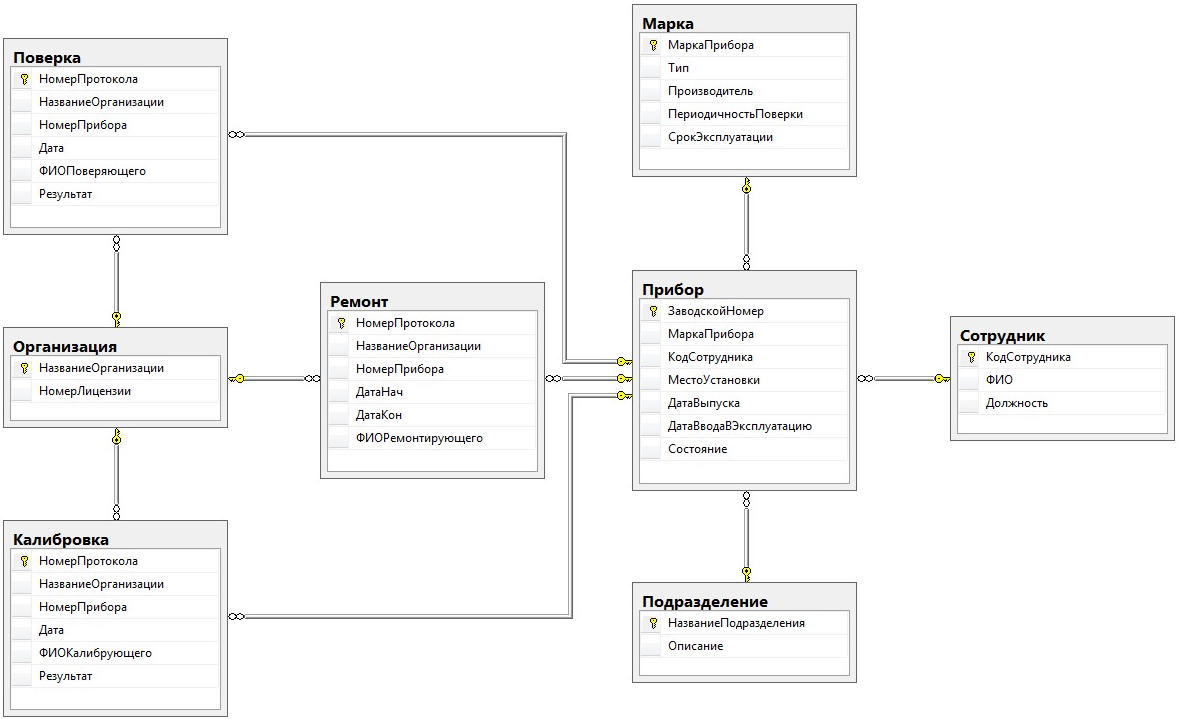


Рисунок 16 – Схема данных

### Задание частных ограничений целостности данных

Отношение Марка содержит следующие ограничения:

* Primary key (МаркаПрибора);
* Not null (Тип, ПериодичностьПоверки, СрокЭксплуатации).

Отношение Сотрудник содержит следующие ограничения:

* Primary key (КодСотрудника);
* Not null (ФИО);

Отношение Подразделение содержит следующие ограничения:

* Primary key (НазваниеПодразделения).

Отношение Прибор содержит следующие ограничения:

* Primary key (ЗаводскойНомер);
* Not null (ДатаВводаВЭксплуатацию, КодСотрудника, МестоУстановки, МаркаПрибора);
* Установка текущей даты как значения по умолчанию для столбца ДатаВводаВЭксплуатацию: default getdate();
* Установка значения по умолчанию для столбца Состояние: default 'Не используется';
* Foreign key (МаркаПрибора ссылается на Марка.МаркаПрибора);
* Foreign key (КодСотрудника ссылается на Сотрудник.КодСотрудника);
* Foreign key (МестоУстановки ссылается на Подразделение.НазваниеПодразделения).

Отношение Организация содержит следующие ограничения:

* Primary key (НазваниеОрганизации).

Отношение Поверка содержит следующие ограничения:

* Primary key (НомерПротокола);
* Not null (НазваниеОрганизации, НомерПрибора, Дата, Результат);
* Установка значения по умолчанию для столбца Результат: default 'Годен';
* Foreign key (НазваниеОрганизации ссылается на Организация.НазваниеОрганизации);
* Foreign key (НомерПрибора ссылается на Прибор.ЗаводскойНомер).

Отношение Калибровка содержит следующие ограничения:

* Primary key (НомерПротокола);
* Not null (НазваниеОрганизации, НомерПрибора, Дата);
* Foreign key (НазваниеОрганизации ссылается на Организация.НазваниеОрганизации);
* Foreign key (НомерПрибора ссылается на Прибор.ЗаводскойНомер).

Отношение Ремонт содержит следующие ограничения:

* Primary key (НомерПротокола);
* Not null (НазваниеОрганизации, НомерПрибора);
* Foreign key (НазваниеОрганизации ссылается на Организация.НазваниеОрганизации);
* Foreign key (НомерПрибора ссылается на Прибор.ЗаводскойНомер).

Сценарий создания объектов базы данных и текст хранимых процедур приведены в приложениях А и Б соответственно.

# Разработка клиентской части информационной системы

## Организация взаимодействия приложения с базой данных

Взаимодействие с базой данных осуществляется в отсоединенном режиме с помощью программной платформы .NET Framework и технологии ADO.NET [10].

.NET Framework — программная платформа, выпущенная компанией Microsoft в 2002 году. Основой платформы является общеязыковая среда исполнения Common Language Runtime (CLR), которая подходит для разных языков программирования. Хотя .NET является патентованной технологией корпорации Microsoft и официально рассчитана на работу под операционными системами семейства Microsoft Windows, существуют независимые проекты (прежде всего это Mono и Portable.NET), позволяющие запускать программы .NET на некоторых других операционных системах. В настоящее время .NET Framework получает развитие в виде .NET Core, изначально предполагающей кросплатформенную разработку и эксплуатацию.

ADO.NET - технология, предоставляющая доступ к данным для приложений, основанных на Microsoft .NET. Является не развитием более ранней технологии ADO, а самостоятельной технологией, частью фреймворка .NET. В отличие от классической ADO, которая была в основном предназначена для тесно связанных клиент-серверных систем, ADO.NET больше нацелена на автономную работу с помощью объектов DataSet. Эти типы представляют локальные копии любого количества взаимосвязанных таблиц данных, каждая из которых содержит набор строк и столбцов. Объекты DataSet позволяют вызывающей сборке (наподобие веб-страницы или программы, выполняющейся на настольном компьютере) работать с содержимым DataSet, изменять его, не требуя подключения к источнику данных, и отправлять обратно блоки измененных данных для обработки с помощью соответствующего адаптера данных

Среди основных классов, использованных при разработке, можно выделить DataSet, DataTable, DataAdapter, SqlCommand.

DataSet – представляет собой контейнер для любого количества объектов DataTable, каждый из которых содержит коллекцию объектов DataRow и DataColumn. Объект адаптера данных конкретного поставщика данных автоматически обслуживает подключение к базе данных. Для повышения масштабируемости адаптеры данных держат подключение открытым минимально возможное время. Как только вызывающий процесс получит объект DataSet, вызывающий уровень полностью отключается от базы данных и остается с локальной копией удаленных данных.

DataAdapter осуществляет связь между БД и отсоединенными объектами. Он предоставляет методы, позволяющие передавать данные из источника данных в отсоединенные объекты различного типа для последующей автономной работы с ними, передавать отложенные изменения из отсоединенных объектов обратно в БД.

SqlCommand можно использовать для выполнения как операторов подмножества DML (SELECT, INSERT, UPDATE,DELETE), так и для иных операторов языка SQL (CREATE, ALTER, DROP и др.). Также класс SqlCommand используется для вызова хранимых процедур или получения прямого доступа к записям определенных таблиц.

## Разработка интерфейса пользователя

### Проектирование прототипа интерфейса

Прототип интерфейса пользователя разрабатывается для каждой формы, определенной в дереве форм, которое представлено на рисунке 17.

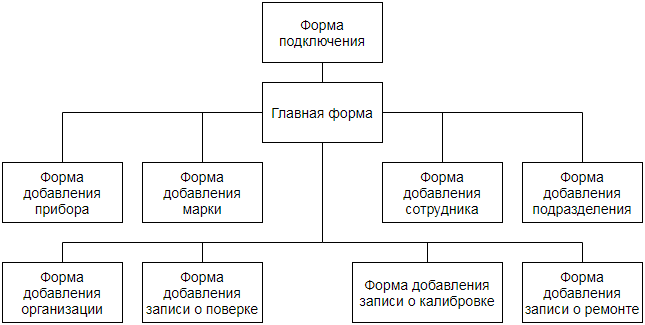


Рисунок 17 – Дерево форм

Форма подключения. Предполагает наличие полей для ввода информации (имя сервера, имя базы данных, имя пользователя базы данных, пароль), а также кнопку подключения. Прототип формы подключения представлен на рисунке 18.

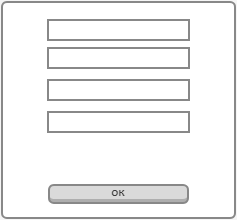


Рисунок 18 – Прототип формы подключения

Главная форма. Предполагает наличие восьми вкладок, каждой из которых соответствует таблица базы данных. Каждая вкладка имеет поле ввода информации для поиска по текущей таблице, а также кнопки добавления и удаления записей из таблицы. Прототип главной формы представлен на рисунке 19.

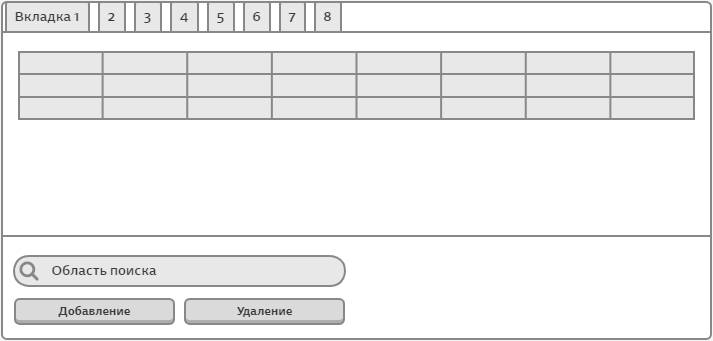


Рисунок 19 – Прототип главной формы

Форма добавления данных в таблицу. Предполагает наличие области ввода информации, а также кнопки добавления записи в таблицу. Прототип формы добавления представлен на рисунке 20.

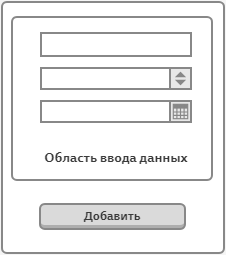


Рисунок 20 – Прототип формы добавления данных

### Реализация интерфейса

В ходе реализации интерфейса пользователя на основе прототипов, описанных в п. 5.2.1, были использованы следующие компоненты:

* TabControl. С помощью данного компонента реализуется структура вкладок главной формы. Каждой вкладке соответствует таблица базы данных;
* DataGridView. Данный компонент используется для вывода информации из базы данных на экран в виде таблицы;
* Button. Все кнопки, используемые в приложении, реализуются с помощью данного компонента;
* Label. Компонент используется для вывода всех текстовых подписей в приложении;
* Panel. С помощью данного компонента реализуется визуальное отделение области редактирования от области получения информации о приборах, которые необходимо поверить в указанный срок, на вкладке «Приборы»;
* TextBox. Все поля для ввода информации, используемые в приложении, реализуются с помощью данного компонента;
* ComboBox. Данный компонент используется для реализации всех выпадающих списков, не связанных с датой, используемых в приложении;
* CrystalReportViewer. Используется для работы с отчетами;
* DateTimePicker. Компонент реализует выпадающие списки для выбора даты.

С помощью перечисленных выше компонентов были разработаны следующие формы:

* форма подключения. Содержит четыре строки для ввода информации (имя сервера, имя базы данных, имя пользователя базы данных, пароль), а также кнопку подключения. В случае удачного соединения открывается главная форма. Форма подключения представлена на рисунке 21.

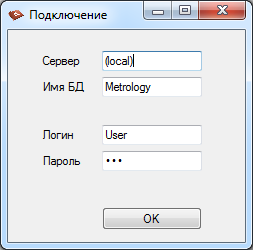


Рисунок 21 – Форма подключения

* главная форма. Имеет восемь вкладок, каждой из которых соответствует таблица базы данных. При загрузке формы, таблицы заполняются полной информацией из базы данных. Каждая вкладка имеет поле ввода информации для поиска по текущей таблице, а также кнопки добавления и удаления записей из таблицы. Для получения информации о приборах, которые необходимо поверить в указанный срок, на вкладке «Приборы» присутствует поле для ввода срока, а также кнопки для вывода таких приборов в таблицу и сброса заполненного поля. Главная форма показана на рисунке 22.

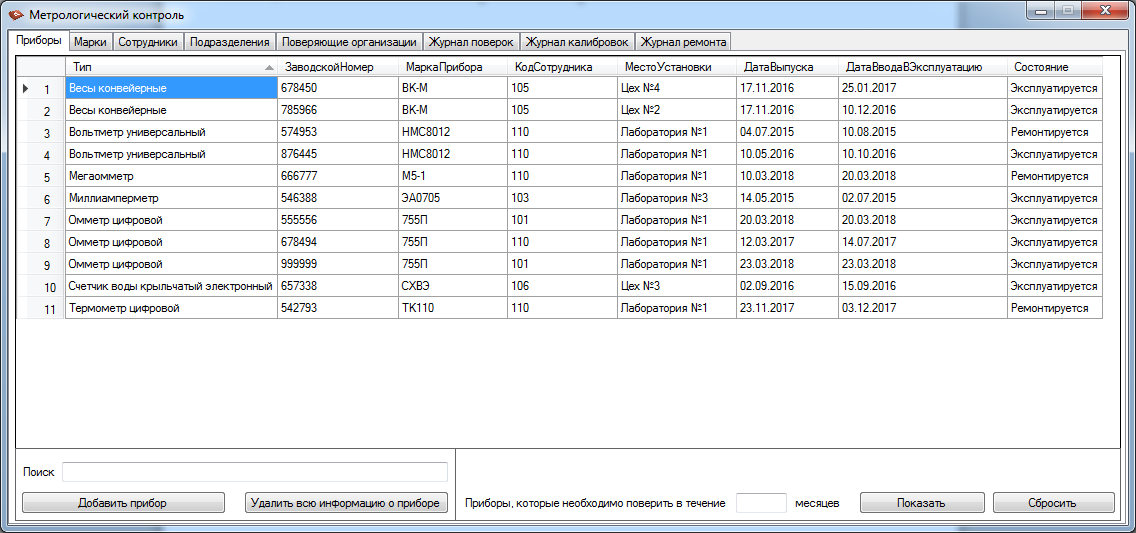


Рисунок 22 – Главная форма

* форма добавления прибора. Содержит поля для ввода заводского номера прибора, а также выпадающие списки для выбора марки прибора, подразделения, сотрудника, даты выпуска и даты ввода в эксплуатацию. Форма также имеет кнопку добавления записи в таблицу. Форма добавления прибора представлена на рисунке 23.

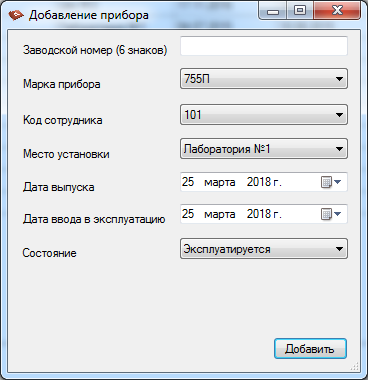


Рисунок 23 – Форма добавления прибора

* форма добавления марки. Имеет поля для ввода марки, типа, производителя, периодичности поверки и срока эксплуатации. Форма также содержит кнопку добавления записи в таблицу. Форма добавления марки представлена на рисунке 24.

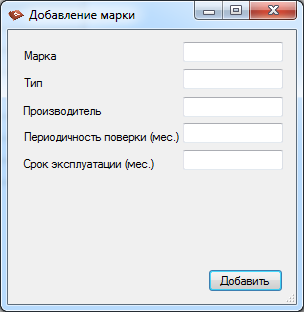


Рисунок 24 – Форма добавления марки

* форма добавления сотрудника. Имеет поля для ввода кода сотрудника, ФИО и должности. Форма также содержит кнопку добавления записи в таблицу. Форма добавления сотрудника представлена на рисунке 25.

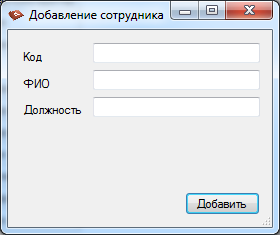


Рисунок 25 – Форма добавления сотрудника

* форма добавления подразделения. Имеет поля для ввода названия подразделения и его описания. Форма также содержит кнопку добавления записи в таблицу. Форма добавления подразделения представлена на рисунке 26.

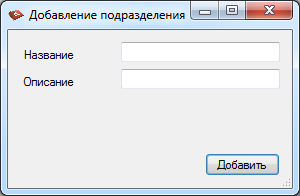


Рисунок 26 – Форма добавления подразделения

* форма добавления поверяющей организации. Имеет поля для ввода названия организации и номера лицензии. Форма также содержит кнопку добавления записи в таблицу. Форма добавления поверяющей организации представлена на рисунке 27.

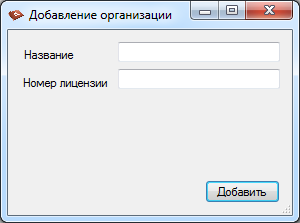


Рисунок 27 – Форма добавления организации

* форма добавления записи о поверке. Имеет поля для ввода номера протокола и ФИО поверяющего, а также выпадающие списки для выбора номера прибора, организации, результата и даты поверки. Форма также содержит кнопку добавления записи в таблицу. Форма добавления прибора представлена на рисунке 28.

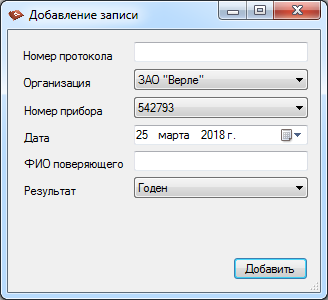


Рисунок 28 – Форма добавления записи о поверке

* форма добавления записи о калибровке. Имеет поля для ввода номера протокола, ФИО калибрующего и результата калибровки, а также выпадающие списки для выбора номера прибора, организации и даты калибровки. Форма также содержит кнопку добавления записи в таблицу. Форма добавления прибора представлена на рисунке 29.

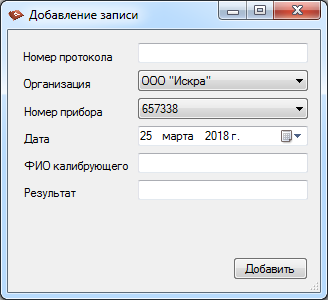


Рисунок 29 – Форма добавления записи о калибровке

* форма добавления записи о ремонте. Имеет поля для ввода номера протокола и ФИО ремонтирующего, а также выпадающие списки для выбора номера прибора, организации и дат начала и завершения ремонта. Форма также содержит кнопку добавления записи в таблицу. Форма добавления прибора представлена на рисунке 30.

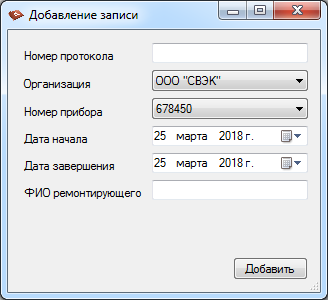


Рисунок 30 – Форма добавления записи о ремонте

### Разработка отчетов

Под отчетом понимают средство для подготовки информации из базы данных к печати на принтере. С помощью отчетов удается формировать необходимую документацию, основанную на данных, с которыми работает информационная система. Отчетом для разрабатываемой системы является журнал поверок средств измерений на текущее число.

В качестве инструмента для создания отчета был выбран Crystal Reports для Visual Studio, который является стандартным средством генерации отчетов.

На рисунке 31 представлена главная форма с областью работы с отчетом. С помощью кнопки «Таблица/отчет», расположенной на вкладке «Журнал поверок», осуществляется переключение между областями работы с таблицей и отчетом. Итоговый вид отчета, экспортированного в формате .pdf, представлен на рисунке 32.

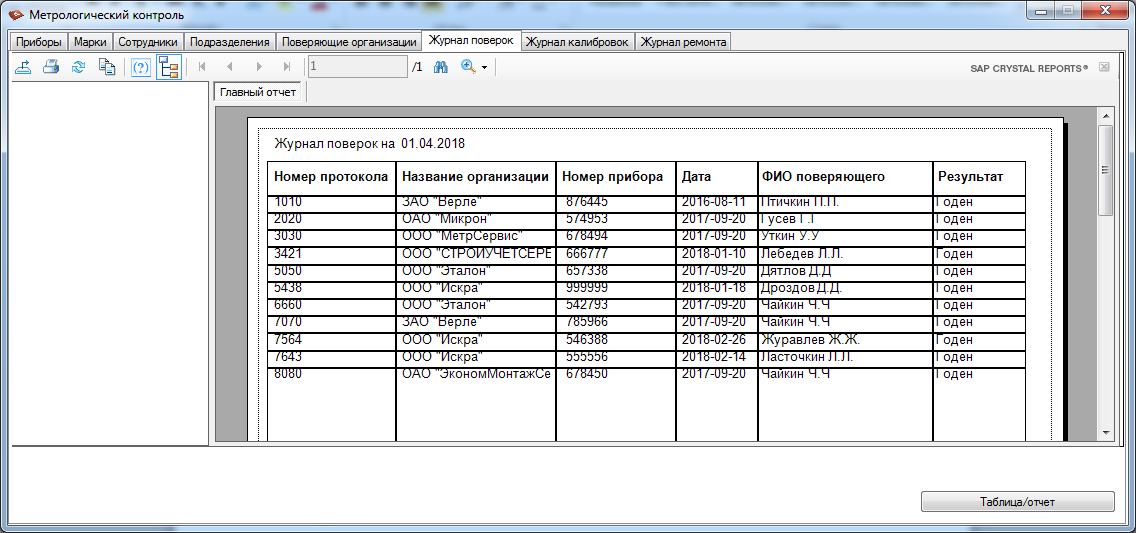


Рисунок 31 – Область работы с отчетом

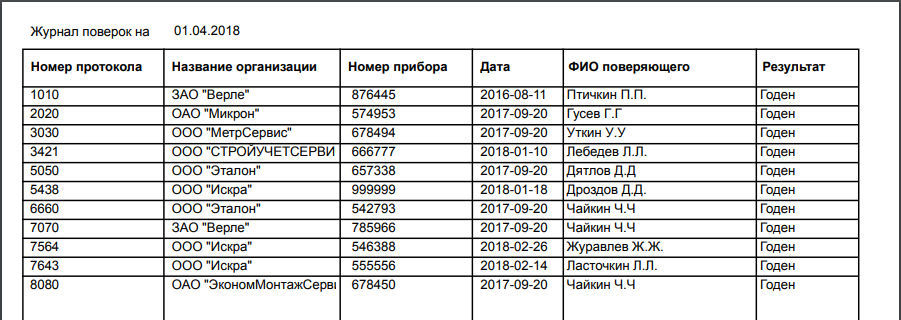


Рисунок 32 – Экспортированный отчет в формате .pdf

## Разработка установочного пакета

Установочный пакет для разработанного приложения был реализован с помощью программы Inno Setup 5.0.

Основные этапы процесса создания установочного пакета приведены на рисунках 33-37. На рисунке 33 приведена форма, появляющаяся в начале создания инсталлятора. На рисунке 34 показано окно с основной информацией о программе. На рисунке 35 представлена форма, в которой производится выбор исполняемого файла приложения, а также файлов базы данных.

Запуск разработанного инсталлятора осуществляется с помощью исполняемого файла, представленного на рисунке 36. Окно запущенного установочного пакета приведено на рисунке 37.

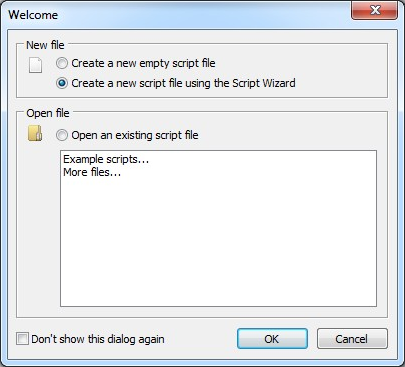


Рисунок 33 – Начало создания установочного пакета

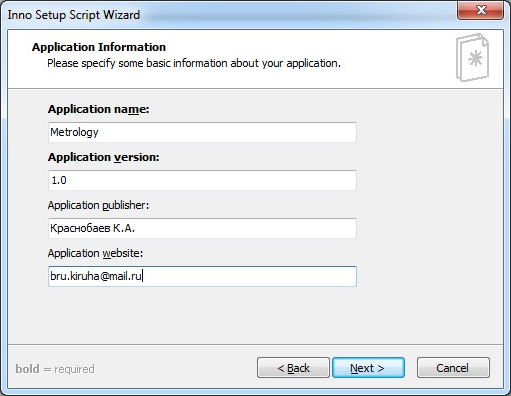


Рисунок 34 – Основная информация о программе

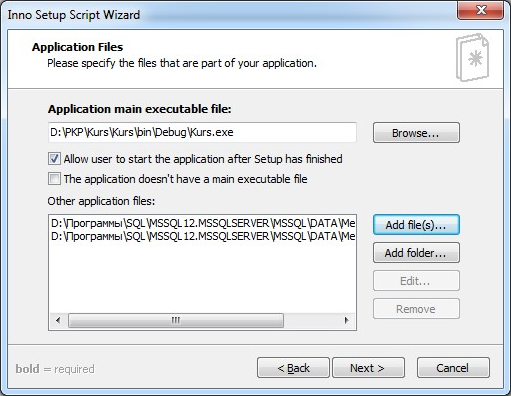


Рисунок 35 – Выбор файлов приложения



Рисунок 36 – Исполняемый файл инсталлятора

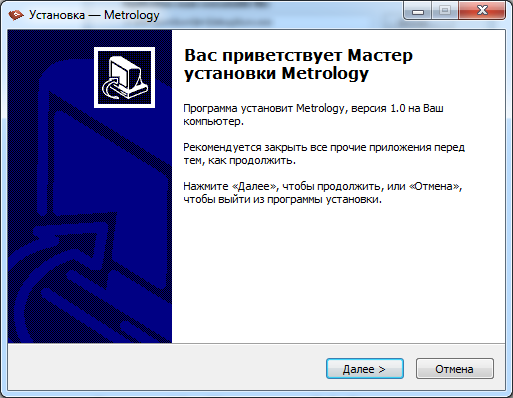


Рисунок 37 – Окно запущенного установочного пакета

# Разработка программной документации

## Руководство программиста

### Общие сведения и структура программы

Разработанная программа написана на языке высокого уровня С# в интегрированной среде разработки программного обеспечения MS Visual Studio 2015.

Приложение представлено следующими файлами:

* Program.cs (основной файл приложения);
* LoginForm.cs Form1.cs, AddSotrudnikForm.cs, AddDeviceForm.cs, AddMarkaForm.cs, AddPodrazdForm.cs, AddOrganizForm.cs, AddPoverkaForm.cs, AddKalibrovkaForm.cs, AddRemont.cs, Poverki.cs (файлы с исходными текстами модулей);
* App.config (файл конфигурации).

Исходный текст клиентской программы приведен в приложении В.

### Установка и настройка программы

Установка программы производится с помощью установочного пакета. Запуск инсталлятора осуществляется путем открытия исполняемого файла SetupMetrology.exe.

Рекомендуемые системные требования:

* Операционная система: Windows 7 и выше;
* Процессор: Intel(R) Core(TM) i5-3210M CPU @ 2.50 GHz;
* Оперативная память: 512 Мб и выше;
* Экран с разрешением 1366×768 и выше;
* Дисковое пространство: не менее 100 Мб;
* Клавиатура, мышь.

Приложение не требует дополнительной настройки.

## Руководство оператора (пользователя)

### Назначение и условия применения программы

Основным назначением разработанного приложения является предоставление пользователю возможности просматривать и изменять данные о средствах измерения, находящихся на учете предприятия, информацию о марках, сотрудниках, подразделениях. Также пользователь имеет доступ к журналам метрологического обслуживания (поверки, калибровки и ремонта).

### Работа с программой

Запуск программы осуществляется путем открытия исполняемого файла Metrology.exe.

После запуска программы на экране появляется форма подключения (рисунок 38). Пользователю необходимо ввести в соответствующие поля имя сервера, имя базы данных, имя пользователя базы данных и пароль, после чего нажать кнопку «ОК».

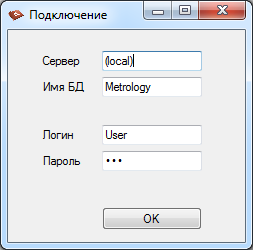


Рисунок 38 – Подключение к базе данных

После подключения пользователь попадает на главную форму программы (рисунок 39), которая имеет восемь вкладок, каждой из которых соответствует таблица базы данных. Выбор необходимой вкладки осуществляется щелчком по ней.

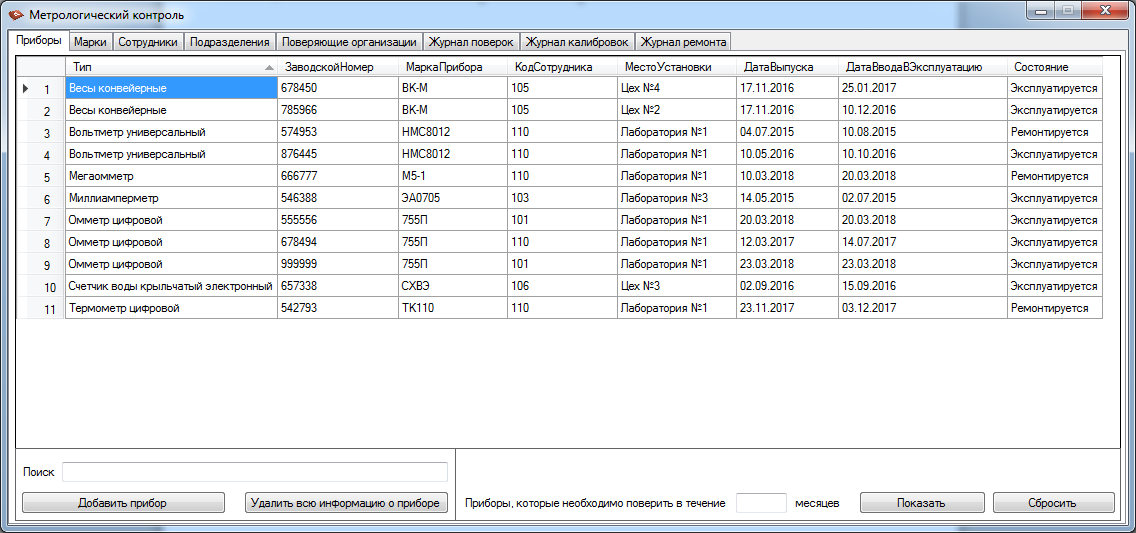


Рисунок 39 – Главная форма

Для добавления записи в базу данных пользователю требуется нажать на кнопку добавления, после чего на экране появится окно с полями для ввода информации (рисунок 40). После заполнения всех полей и выбора информации из выпадающих списков необходимо нажать кнопку «Добавить».

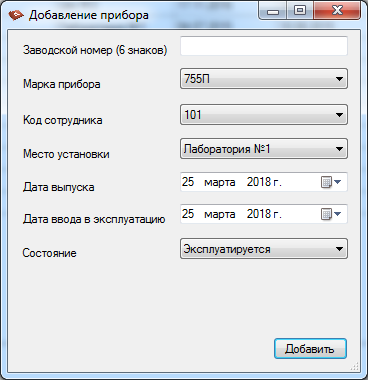


Рисунок 40 – Окно добавления информации в базу данных

Для удаления записи из базы данных пользователь должен выделить строку, которую необходимо удалить. Выделение осуществляется щелчком по номеру строки. После того как строка выделена, необходимо нажать на кнопку удаления.

Поиск по таблицам осуществляется при вводе поискового запроса в поле поиска, находящееся в нижней левой части окна.

Для получения информации о приборах, которые необходимо поверить в указанный срок, пользователю необходимо, находясь на вкладке «Приборы», ввести срок в месяцах в поле, расположенное в нижней правой части окна. После чего нажать кнопку «Показать». На экране появится требуемый список приборов. Для сброса результатов операции необходимо нажать кнопку «Сбросить».

Для работы с отчетом необходимо перейти на вкладку «Журнал поверок» и нажать кнопку «Таблица/отчет», которая предназначена для включения (отключения) отображения области работы с отчетом (рисунок 41).

Если пользователю требуется распечатать отчет, то необходимо нажать кнопку «Распечатать отчет», расположенную на панели инструментов. Если пользователю требуется сохранить отчет, то необходимо нажать кнопку «Экспортировать отчет».

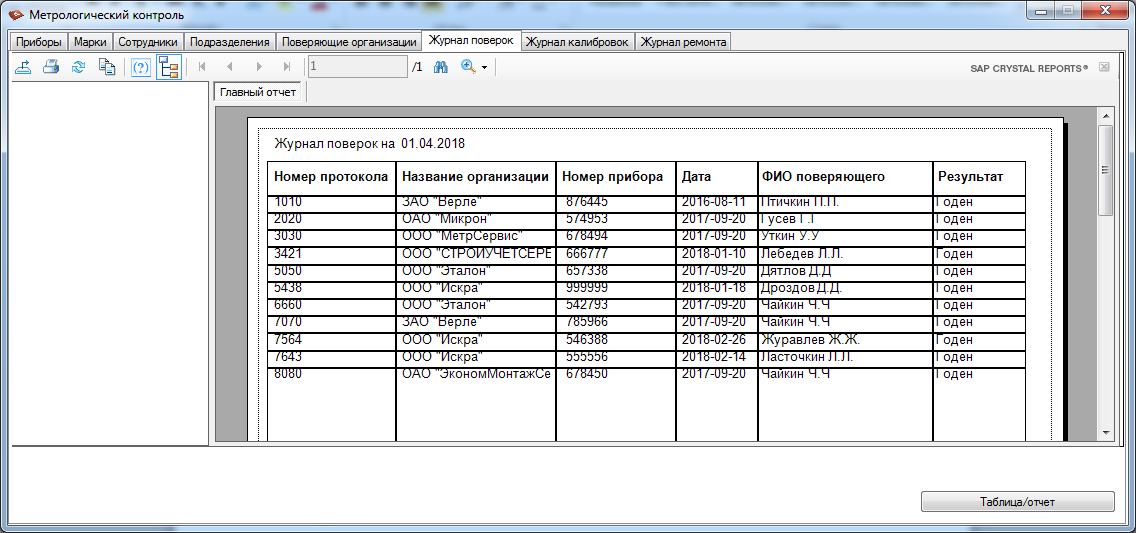


Рисунок 41 – Область работы с отчетом

### Сообщения оператору

В программе предусмотрены сообщения в следующих ситуациях: при неудачной попытке подключиться к серверу (рисунок 42); при неудачной попытке удаления записи из таблицы (рисунок 43); при некорректном заполнении полей на форме добавления, например, вводе текста в поля, предназначенные для ввода числовых значений, или установке даты начала ремонта более поздней, чем дата завершения (рисунок 44); при некорректном заполнении поля, в котором указывается срок поверки (рисунок 45).

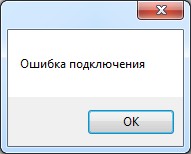


Рисунок 42 – Сообщение об ошибке подключения

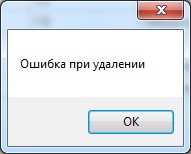


Рисунок 43 – Сообщение об ошибке при удалении

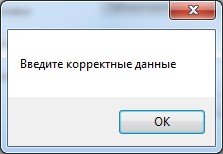


Рисунок 44 – Сообщение о вводе некорректных данных

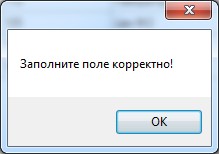


Рисунок 45 – Сообщение о некорректном заполнении поля

# Программа и методика испытаний

## Методы испытаний

Проведем проверку процедуры «Подключение» с помощью тестирования базового пути. Данная процедура на псевдокоде выглядит следующим образом:

0. ввод сервера, базы данных, логина и пароля

1, 2, 3, 4. если сервер доступен и база данных доступна и логин введен верно и пароль введен верно

4. то

5. выполнение подключения

6. иначе

6. вывод сообщения об ошибке подключения

7. конец

Потоковый граф, построенный для данной процедуры, представлен на рисунке 46.

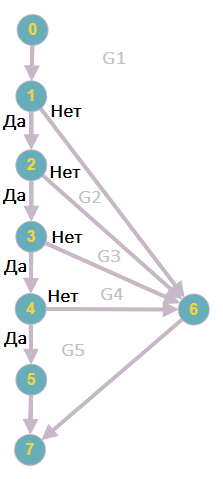


Рисунок 46 – Потоковый граф

Далее вычислим цикломатическую сложность алгоритма тремя способами:

1) (G) = E – N + 2 = 11 – 8 + 2 = 5, где E – количество дуг, N – количество вершин;

2) (G) = p + 1 = 4 + 1 = 5, где p – количество предикатных вершин;

3) (G) = количество регионов в графе = 5.

Так как по всем трем формулам значения совпали, потоковый граф был построен верно.

Далее определим множество независимых линейных путей:

1) 0 – 1 – 6 – 7;

2) 0 – 1 – 2 – 6 – 7;

3) 0 – 1 – 2 – 3 – 6 – 7;

4) 0 – 1 – 2 – 3 – 4 – 6 – 7;

5) 0 – 1 – 2 – 3 – 4 – 5 – 7.

Для каждого пути опишем исходные данные (ИД) и ожидаемый результат (ОР).

1) Тестовый вариант для пути 1

ИД: Сервер недоступен.

ОР: Сообщение об ошибке подключения.

2) Тестовый вариант для пути 2

ИД: Сервер доступен, база данных недоступна.

ОР: Сообщение об ошибке подключения.

3) Тестовый вариант для пути 3

ИД: Сервер доступен, база данных доступна, логин введен неверно.

ОР: Сообщение об ошибке подключения.

4) Тестовый вариант для пути 4

ИД: Сервер доступен, база данных доступна, логин введен верно, пароль введен неверно.

ОР: Сообщение об ошибке подключения.

5) Тестовый вариант для пути 4

ИД: Сервер доступен, база данных доступна, логин введен верно, пароль введен верно.

ОР: Успешное подключение.

## **Результаты испытаний**

В результате проверки работы приложения с тестовыми исходными данными были получены следующие результаты (рисунки 47-50).

Для тестовых вариантов с первого по четвертый:

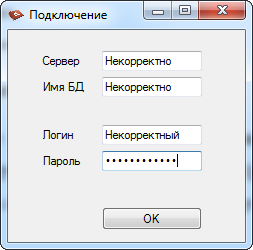


Рисунок 47 – Ввод некорректных значений

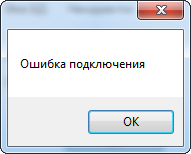


Рисунок 48 – Вывод сообщения об ошибке подключения

Для пятого тестового варианта:

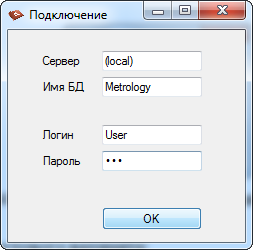


Рисунок 49 – Ввод корректных значений

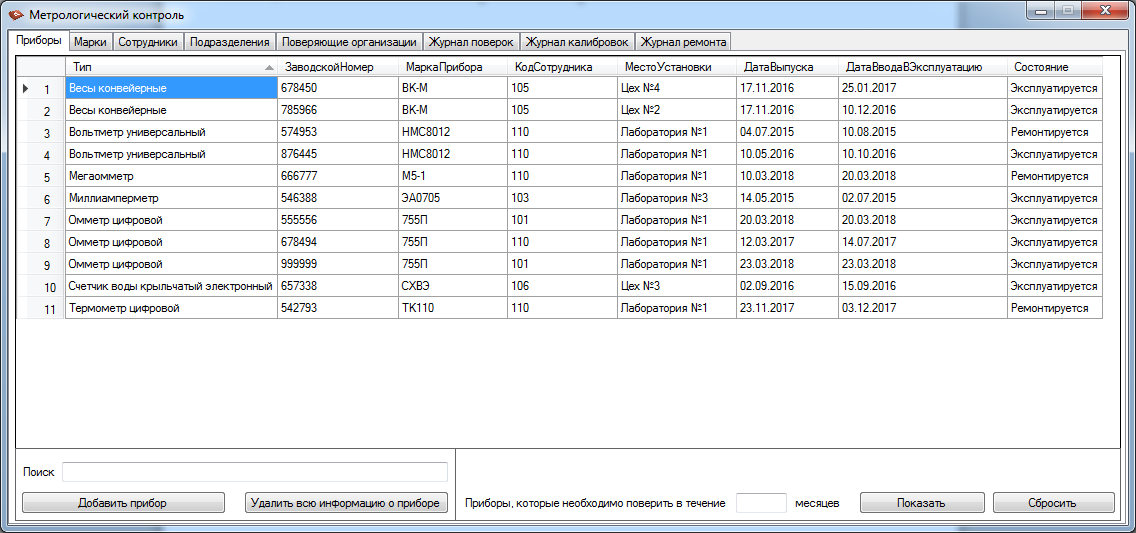


Рисунок 50 – Вывод главной формы после удачного подключения

Результаты выполнения всех тестовых вариантов полностью совпадают с ожидаемыми результатами. Аналогичным образом были протестированы все остальные процедуры, реализованные в приложении.

# ЗАКЛЮЧЕНИЕ

В соответствии с выявленными требованиями было разработано программное обеспечение, позволяющее вести учет средств измерений и контроль метрологических процедур. Приложение имеет простой и понятный интерфейс, который позволяет работать с программой пользователям с любым уровнем подготовки. При разработке программы были предусмотрены возможные ошибки, при возникновении которых на экран выводятся соответствующие сообщения.

Разработанное приложение было реализовано в среде Visual Studio 2015 на высокоуровневом языке C#. Серверная часть была выполнена под управлением СУБД MS SQL Server 2014. Выбор данных программных средств объясняется тем, что они являются продуктами компании Microsoft и имеют множество инструментов взаимодействия.

Эксплуатация разработанного приложения позволит перевести основную часть документооборота предприятия, связанного с учетом приборов и журналами учета метрологических процедур, в электронный вид; снизит временные затраты на разработку отчетности; обеспечит более удобное планирование проведения метрологических процедур. Продукт готов к использованию и может дорабатываться и улучшаться за счет добавления нового функционала с учетом требований конкретного предприятия.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

* + - 1. ГОСТ 2.105-95 ЕСКД «Общие требования к текстовым документам»;
      2. ГОСТ 7.32-2001 «Отчет о научно-исследовательской работе. Структура и правила оформления»;
      3. Top-Rider [Электронный ресурс]. – http://www.top-rider.ru;
      4. Viscomp [Электронный ресурс]. – https://viscomp.ru;
      5. Ипатова, Э. Р. Методологии и технологии системного проектирования информационных систем / Э.Р. Ипатова, Ю.В. Ипатов. - М.: Флинта, 2016;
      6. Дейт К. Дж.: Введение в системы баз данных / Дейт К. Дж.; -М.: Издательский дом «Вильямс», 2001;
      7. Малыхина, М. П. Базы данных. Основы, проектирование, использование / М.П. Малыхина. - М.: БХВ-Петербург, **2013**;
      8. Мюллер, Р. Дж. Базы данных и UML. Проектирование / Р.Дж. Мюллер. - М.: ЛОРИ, **2016;**
      9. Карпова, Т.С. Базы данных. Модели, разработка, реализация / Т.С. Карпова. - М.: СПб: Питер, **2012;**
      10. Культин, Н. Microsoft Visual C# в задачах и примерах / Н. Культин. - М.: БХВ-Петербург, 2012.

# ПРИЛОЖЕНИЕ А

Сценарий создания объектов базы данных:

USE master;

GO

CREATE DATABASE Metrology;

GO

USE Metrology;

GO

CREATE TYPE string FROM nvarchar(50);

GO

CREATE TYPE interval FROM tinyint not null;

GO

Create Table Марка

(МаркаПрибора nvarchar(20) primary key,

Тип string not null,

Производитель string,

ПериодичностьПоверки interval,

СрокЭксплуатации interval)

GO

Create Table Сотрудник

(КодСотрудника int primary key,

ФИО string not null,

Должность string)

GO

Create Table Подразделение

(НазваниеПодразделения string primary key,

Описание nvarchar(max))

GO

Create Table Прибор

(ЗаводскойНомер int primary key,

МаркаПрибора nvarchar(20) not null,

КодСотрудника int not null,

МестоУстановки string not null,

ДатаВыпуска date,

ДатаВводаВЭксплуатацию date not null default getdate(),

Состояние nvarchar(20) default 'Не используется',

constraint fk\_Прибор\_Марка foreign key (МаркаПрибора) references Марка(МаркаПрибора),

constraint fk\_Прибор\_Сотрудник foreign key (КодСотрудника) references Сотрудник(КодСотрудника),

constraint fk\_Прибор\_Подразделение foreign key (МестоУстановки) references Подразделение(НазваниеПодразделения))

GO

Create Table Организация

(НазваниеОрганизации string primary key,

НомерЛицензии int)

GO

Create Table Поверка

(НомерПротокола int primary key,

НазваниеОрганизации string not null,

НомерПрибора int not null,

Дата date not null,

ФИОПоверяющего string,

Результат nvarchar(10) not null default 'Годен',

constraint fk\_Поверка\_Организация foreign key (НазваниеОрганизации) references Организация(НазваниеОрганизации),

constraint fk\_Поверка\_Прибор foreign key (НомерПрибора) references Прибор(ЗаводскойНомер))

GO

Create Table Калибровка

(НомерПротокола int primary key,

НазваниеОрганизации string not null,

НомерПрибора int not null,

Дата date not null,

ФИОКалибрующего string,

Результат nvarchar(max),

constraint fk\_Калибровка\_Организация foreign key (НазваниеОрганизации) references Организация(НазваниеОрганизации),

constraint fk\_Калибровка\_Прибор foreign key (НомерПрибора) references Прибор(ЗаводскойНомер))

GO

Create Table Ремонт

(НомерПротокола int primary key,

НазваниеОрганизации string not null,

НомерПрибора int not null,

ДатаНач date,

ДатаКон date,

ФИОРемонтирующего string,

constraint fk\_Ремонт\_Организация foreign key (НазваниеОрганизации) references Организация(НазваниеОрганизации),

constraint fk\_Ремонт\_Прибор foreign key (НомерПрибора) references Прибор(ЗаводскойНомер))

GO

create view [dbo].[norepeat]

as

select НомерПрибора, MAX(Дата) as Дата from dbo.Поверка group by НомерПрибора

GO

# ПРИЛОЖЕНИЕ Б

Исходный текст хранимых процедур:

ALTER PROC [dbo].[P11] (@ЗаводскойНомер int, @МаркаПрибора nvarchar(20), @КодСотрудника int, @МестоУстановки string, @ДатаВыпуска date, @ДатаВводаВЭксплуатацию date, @Состояние nvarchar(20))

As

insert Metrology.dbo.Прибор

values (@ЗаводскойНомер, @МаркаПрибора, @КодСотрудника, @МестоУстановки, @ДатаВыпуска, @ДатаВводаВЭксплуатацию, @Состояние)

ALTER PROC [dbo].[P12] (@МаркаПрибора nvarchar(20),@Тип nvarchar(50),@Производитель nvarchar(50),

@ПериодичностьПоверки tinyint,@СрокЭксплуатации tinyint)

As

insert Metrology.dbo.Марка

values (@МаркаПрибора,@Тип,@Производитель,

@ПериодичностьПоверки ,@СрокЭксплуатации)

ALTER PROC [dbo].[P13] (@КодСотрудника int,@ФИО nvarchar(50),@Должность nvarchar(50))

As

insert Metrology.dbo.Сотрудник

values (@КодСотрудника,@ФИО,@Должность)

ALTER PROC [dbo].[P14] (@НазваниеПодразделения nvarchar(50),@Описание nvarchar(max))

As

insert Metrology.dbo.Подразделение

values (@НазваниеПодразделения,@Описание)

ALTER PROC [dbo].[P15] (@НазваниеОрганизации nvarchar(50),@НомерЛицензии int)

As

insert Metrology.dbo.Организация

values (@НазваниеОрганизации,@НомерЛицензии)

ALTER PROC [dbo].[P16] (@НомерПротокола int,

@НазваниеОрганизации nvarchar(50), @НомерПрибора int,

@Дата date, @ФИОПоверяющего nvarchar(50),

@Результат nvarchar(10))

As

insert Metrology.dbo.Поверка

values (@НомерПротокола, @НазваниеОрганизации, @НомерПрибора,

@Дата,@ФИОПоверяющего,@Результат)

ALTER PROC [dbo].[P17] (@НомерПротокола int,

@НазваниеОрганизации nvarchar(50),@НомерПрибора int,

@Дата date, @ФИОКалибрующего nvarchar(50),

@Результат nvarchar(50))

As

insert Metrology.dbo.Калибровка

values (@НомерПротокола, @НазваниеОрганизации, @НомерПрибора,

@Дата, @ФИОКалибрующего,@Результат)

ALTER PROC [dbo].[P18] (@НомерПротокола int,

@НазваниеОрганизации nvarchar(50), @НомерПрибора int,

@ДатаНач date, @ДатаКон date, @ФИОРемонтирующего nvarchar(50))

As

insert Metrology.dbo.Ремонт

values (@НомерПротокола, @НазваниеОрганизации, @НомерПрибора,

@ДатаНач, @ДатаКон, @ФИОРемонтирующего)

ALTER PROCEDURE [dbo].[P19] @id int

AS

delete from dbo.Поверка where dbo.Поверка.НомерПрибора=@id;

delete from dbo.Калибровка where dbo.Калибровка.НомерПрибора=@id;

delete from dbo.Ремонт where dbo.Ремонт.НомерПрибора=@id;

delete from dbo.Прибор where dbo.Прибор.ЗаводскойНомер=@id;

ALTER PROCEDURE [dbo].[P20] @id int

AS

delete from dbo.Поверка where dbo.Поверка.НомерПротокола=@id;

ALTER PROCEDURE [dbo].[P21] @id int

AS

delete from dbo.Калибровка where dbo.Калибровка.НомерПротокола=@id;

ALTER PROCEDURE [dbo].[P22] @id int

AS

delete from dbo.Ремонт where dbo.Ремонт.НомерПротокола=@id;

ALTER PROC [dbo].[P10] (@Срок int)

As

select Тип,ЗаводскойНомер, Прибор.МаркаПрибора, КодСотрудника, МестоУстановки,

ДатаВыпуска, ДатаВводаВЭксплуатацию, Состояние from (Metrology.dbo.Прибор

inner join Metrology.dbo.Марка on Прибор.МаркаПрибора = Марка.МаркаПрибора) inner join Metrology.dbo.norepeat

on Прибор.ЗаводскойНомер=norepeat.НомерПрибора

where DATEADD(MONTH,ПериодичностьПоверки,Дата)<=DATEADD(MONTH,@Срок,getdate())

# ПРИЛОЖЕНИЕ В

Исходный текст клиентской программы. Форма подключения:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.SqlClient;

namespace Kurs

{

public partial class LoginForm : Form

{

public LoginForm()

{

InitializeComponent();

}

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

{

var foo = @"Data Source=" + ServerTextBox.Text + ";Initial Catalog=" + DBTextBox.Text + ";User ID = "

+ LoginTextBox.Text + "; Password =" + PasswordTextBox.Text;

var connection = new SqlConnection(foo);

try

{

connection.Open();

ConnectToSql.ConnectionString = foo;

var mainForm = new Form1();

mainForm.Show();

Hide();

}

catch (Exception)

{

MessageBox.Show("Ошибка подключения");

}

}

}

}

Главная форма:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Kurs

{

public partial class Form1 : Form

{

private SqlDataAdapter \_dataAdapter = new SqlDataAdapter();

private void GetData(string selectCommand, BindingSource bindingSource)

{

try

{

\_dataAdapter = new SqlDataAdapter(selectCommand, ConnectToSql.ConnectionString);

var table = new DataTable();

\_dataAdapter.Fill(table);

bindingSource.DataSource = table;

}

catch (SqlException)

{

throw;

}

}

public Form1()

{

InitializeComponent();

}

public void Refreshh()

{

Form1\_Load(null, null);

}

private void Form1\_Load(object sender, EventArgs e)

{

dataGridView1.DataSource = bindingSource1;

GetData("select Тип, ЗаводскойНомер, Прибор.МаркаПрибора, КодСотрудника, МестоУстановки, ДатаВыпуска, ДатаВводаВЭксплуатацию, Состояние from Metrology.dbo.Прибор inner join Metrology.dbo.Марка on Прибор.МаркаПрибора = Марка.МаркаПрибора", bindingSource1);

dataGridView2.DataSource = bindingSource2;

GetData("select \* from Марка", bindingSource2);

dataGridView3.DataSource = bindingSource3;

GetData("select \* from Сотрудник", bindingSource3);

dataGridView4.DataSource = bindingSource4;

GetData("select \* from Подразделение", bindingSource4);

dataGridView5.DataSource = bindingSource5;

GetData("select \* from Организация", bindingSource5);

dataGridView6.DataSource = bindingSource6;

GetData("select \* from Поверка", bindingSource6);

dataGridView7.DataSource = bindingSource7;

GetData("select \* from Калибровка", bindingSource7);

dataGridView8.DataSource = bindingSource8;

GetData("select \* from Ремонт", bindingSource8);

crystalReportViewer2.RefreshReport();

}

private void search(TextBox tb, DataGridView dg, object sender, EventArgs e) //Поиск

{

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

{

dg.CurrentCell = null;

bool isVisible = false;

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

{

if (dg[j, i].Value.ToString().IndexOf(tb.Text, StringComparison.OrdinalIgnoreCase) >= 0)

{

isVisible = true;

}

}

dg.Rows[i].Visible = isVisible;

}

}

private void AddDevice\_Click(object sender, EventArgs e) // Добавление прибора

{

var form = new AddDeviceForm();

form.Owner = this;

form.ShowDialog();

textBox1.Text = "";

}

private void AddMarka\_Click(object sender, EventArgs e) // Добавление марки

{

var form = new AddMarkaForm();

form.Owner = this;

form.ShowDialog();

textBox2.Text = "";

}

private void AddSotrudnik\_Click(object sender, EventArgs e) // Добавление сотрудника

{

var form = new AddSotrudnikForm();

form.Owner = this;

form.ShowDialog();

textBox3.Text = "";

}

private void AddPodrazd\_Click(object sender, EventArgs e) // Добавление подразделения

{

var form = new AddPodrazdForm();

form.Owner = this;

form.ShowDialog();

textBox4.Text = "";

}

private void AddOrganiz\_Click(object sender, EventArgs e) // Добавление организации

{

var form = new AddOrganizForm();

form.Owner = this;

form.ShowDialog();

textBox5.Text = "";

}

private void AddPoverka\_Click(object sender, EventArgs e) // Добавление записи о поверке

{

var form = new AddPoverkaForm();

form.Owner = this;

form.ShowDialog();

textBox6.Text = "";

crystalReportViewer2.RefreshReport();

}

private void AddKalibr\_Click(object sender, EventArgs e) // Добавление записи о калибровке

{

var form = new AddKalibrovkaForm();

form.Owner = this;

form.ShowDialog();

textBox7.Text = "";

}

private void AddRemont\_Click(object sender, EventArgs e) // Добавление записи о ремонте

{

var form = new AddRemontForm();

form.Owner = this;

form.ShowDialog();

textBox8.Text = "";

}

private void Delete(string storedProcedure, DataGridView dataGrid, int x,BindingSource bs) //Удаление записей

{

try

{

var selected = int.Parse(dataGrid.SelectedRows[0].Cells[x].FormattedValue.ToString());

dataGrid.DataSource = bs;

GetData(storedProcedure + " " + selected, bs);

Refreshh();

}

catch (Exception)

{

MessageBox.Show("Ошибка при удалении");

}

}

private void DeleteButton1\_Click(object sender, EventArgs e)

{

Delete("P19", dataGridView1, 1, bindingSource1);

search(textBox1, dataGridView1, sender, e);

}

private void DeleteButton6\_Click(object sender, EventArgs e)

{

Delete("P20", dataGridView6, 0, bindingSource6);

search(textBox6, dataGridView6, sender, e);

crystalReportViewer2.RefreshReport();

}

private void DeleteButton7\_Click(object sender, EventArgs e)

{

Delete("P21", dataGridView7, 0, bindingSource7);

search(textBox7, dataGridView7, sender, e);

}

private void DeleteButton8\_Click(object sender, EventArgs e)

{

Delete("P22", dataGridView8, 0, bindingSource8);

search(textBox8, dataGridView8, sender, e);

}

private void ShowPoverka\_Click(object sender, EventArgs e) //Приборы, которые необходимо поверить в указанный срок

{

try

{

dataGridView1.DataSource = bindingSource1;

GetData("P10" + " " + showpov\_.Text, bindingSource1);

search(textBox1, dataGridView1, sender, e);

}

catch (Exception)

{

MessageBox.Show("Заполните поле корректно!");

}

}

}

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

{

if (crystalReportViewer2.Visible == false)

{

crystalReportViewer2.Visible = true;

label6.Visible = false;

textBox6.Visible = false;

DeleteButton6.Visible = false;

AddPoverka.Visible = false;

}

else

{

crystalReportViewer2.Visible = false;

label6.Visible = true;

textBox6.Visible = true;

DeleteButton6.Visible = true;

AddPoverka.Visible = true;

}

}

}

}

Форма добавления прибора:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Kurs

{

public partial class AddDeviceForm : Form

{

private SqlDataAdapter \_dataAdapter = new SqlDataAdapter();

public AddDeviceForm()

{

InitializeComponent();

}

public AddDeviceForm(string id)

{

InitializeComponent();

}

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

{

sost\_.SelectedIndex = 0;

\_dataAdapter = new SqlDataAdapter("select \* from dbo.Марка", ConnectToSql.ConnectionString);

var table = new DataTable();

\_dataAdapter.Fill(table);

marka\_.DataSource = table.Rows.OfType<DataRow>().Select(dr => dr.Field<string>("МаркаПрибора")).ToList();

\_dataAdapter = new SqlDataAdapter("select \* from dbo.Сотрудник", ConnectToSql.ConnectionString);

var table1 = new DataTable();

\_dataAdapter.Fill(table1);

kodsotr\_.DataSource = table1.Rows.OfType<DataRow>().Select(dr => dr.Field<int>("КодСотрудника")).ToList();

\_dataAdapter = new SqlDataAdapter("select \* from dbo.Подразделение", ConnectToSql.ConnectionString);

var table2 = new DataTable();

\_dataAdapter.Fill(table2);

mesto\_.DataSource = table2.Rows.OfType<DataRow>().Select(dr => dr.Field<string>("НазваниеПодразделения")).ToList();

}

private void button1\_Click(object sender, EventArgs e)

{

using (var connection = new SqlConnection(ConnectToSql.ConnectionString))

{

connection.Open();

using (var command = new SqlCommand("P11", connection))

{

try

{

command.CommandType = CommandType.StoredProcedure;

command.Parameters.Add(new SqlParameter("@ЗаводскойНомер", int.Parse(zavnomer\_.Text)));

command.Parameters.Add(new SqlParameter("@МаркаПрибора", marka\_.Text));

command.Parameters.Add(new SqlParameter("@КодСотрудника", int.Parse(kodsotr\_.Text)));

command.Parameters.Add(new SqlParameter("@МестоУстановки", mesto\_.Text));

command.Parameters.Add(new SqlParameter("@ДатаВыпуска", datevyp\_.Value));

command.Parameters.Add(new SqlParameter("@ДатаВводаВЭксплуатацию", datevvoda\_.Value));

command.Parameters.Add(new SqlParameter("@Состояние", sost\_.Text));

command.ExecuteNonQuery();

}

catch (Exception)

{

MessageBox.Show("Введите корректные данные");

}

Form1 main = this.Owner as Form1;

if (main != null)

{

main.Refreshh();

}

}

}

}

}

}

Форма добавления марки:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Kurs

{

public partial class AddMarkaForm : Form

{

private SqlDataAdapter \_dataAdapter = new SqlDataAdapter();

public AddMarkaForm()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

using (var connection = new SqlConnection(ConnectToSql.ConnectionString))

{

connection.Open();

using (var command = new SqlCommand("P12", connection))

{

try

{

command.CommandType = CommandType.StoredProcedure;

command.Parameters.Add(new SqlParameter("@МаркаПрибора",marka\_.Text));

command.Parameters.Add(new SqlParameter("@Тип", type\_.Text));

command.Parameters.Add(new SqlParameter("@Производитель", proizv\_.Text));

command.Parameters.Add(new SqlParameter("@ПериодичностьПоверки", int.Parse(period\_.Text)));

command.Parameters.Add(new SqlParameter("@СрокЭксплуатации", int.Parse(srok\_.Text)));

command.ExecuteNonQuery();

}

catch (Exception)

{

MessageBox.Show("Введите корректные данные");

}

Form1 main = this.Owner as Form1;

if (main != null)

{

main.Refreshh();

}

}

}

}

}

}

Форма добавления сотрудника:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Kurs

{

public partial class AddSotrudnikForm : Form

{

private SqlDataAdapter \_dataAdapter = new SqlDataAdapter();

public AddSotrudnikForm()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

using (var connection = new SqlConnection(ConnectToSql.ConnectionString))

{

connection.Open();

using (var command = new SqlCommand("P13", connection))

{

try

{

command.CommandType = CommandType.StoredProcedure;

command.Parameters.Add(new SqlParameter("@КодСотрудника", int.Parse(code\_.Text)));

command.Parameters.Add(new SqlParameter("@ФИО", fio\_.Text));

command.Parameters.Add(new SqlParameter("@Должность",dolzh\_.Text));

command.ExecuteNonQuery();

}

catch (Exception)

{

MessageBox.Show("Введите корректные данные");

}

Form1 main = this.Owner as Form1;

if (main != null)

{

main.Refreshh();

}

}

}

}

}

}

Форма добавления подразделения:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Kurs

{

public partial class AddPodrazdForm : Form

{

private SqlDataAdapter \_dataAdapter = new SqlDataAdapter();

public AddPodrazdForm()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

using (var connection = new SqlConnection(ConnectToSql.ConnectionString))

{

connection.Open();

using (var command = new SqlCommand("P14", connection))

{

try

{

command.CommandType = CommandType.StoredProcedure;

command.Parameters.Add(new SqlParameter("@НазваниеПодразделения",nazv\_.Text));

command.Parameters.Add(new SqlParameter("@Описание", opis\_.Text));

command.ExecuteNonQuery();

}

catch (Exception)

{

MessageBox.Show("Введите корректные данные");

}

Form1 main = this.Owner as Form1;

if (main != null)

{

main.Refreshh();

}

}

}

}

}

}

Форма добавления организации:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Kurs

{

public partial class AddOrganizForm : Form

{

private SqlDataAdapter \_dataAdapter = new SqlDataAdapter();

public AddOrganizForm()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

using (var connection = new SqlConnection(ConnectToSql.ConnectionString))

{

connection.Open();

using (var command = new SqlCommand("P15", connection))

{

try

{

command.CommandType = CommandType.StoredProcedure;

command.Parameters.Add(new SqlParameter("@НазваниеОрганизации", nazv\_.Text));

command.Parameters.Add(new SqlParameter("@НомерЛицензии", int.Parse(nomer\_.Text)));

command.ExecuteNonQuery();

}

catch (Exception)

{

MessageBox.Show("Введите корректные данные");

}

Form1 main = this.Owner as Form1;

if (main != null)

{

main.Refreshh();

}

}

}

}

}

}

Форма добавления записи о поверке:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Kurs

{

public partial class AddPoverkaForm : Form

{

private SqlDataAdapter \_dataAdapter = new SqlDataAdapter();

public AddPoverkaForm()

{

InitializeComponent();

}

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

{

res\_.SelectedIndex = 0;

\_dataAdapter = new SqlDataAdapter("select \* from dbo.Организация", ConnectToSql.ConnectionString);

var table = new DataTable();

\_dataAdapter.Fill(table);

organ\_.DataSource = table.Rows.OfType<DataRow>().Select(dr => dr.Field<string>("НазваниеОрганизации")).ToList();

\_dataAdapter = new SqlDataAdapter("select \* from dbo.Прибор", ConnectToSql.ConnectionString);

var table1 = new DataTable();

\_dataAdapter.Fill(table1);

nomerprib\_.DataSource = table1.Rows.OfType<DataRow>().Select(dr => dr.Field<int>("ЗаводскойНомер")).ToList();

}

private void button1\_Click(object sender, EventArgs e)

{

using (var connection = new SqlConnection(ConnectToSql.ConnectionString))

{

connection.Open();

using (var command = new SqlCommand("P16", connection))

{

try

{

command.CommandType = CommandType.StoredProcedure;

command.Parameters.Add(new SqlParameter("@НомерПротокола", int.Parse(nomerprot\_.Text)));

command.Parameters.Add(new SqlParameter("@НазваниеОрганизации", organ\_.Text));

command.Parameters.Add(new SqlParameter("@НомерПрибора", int.Parse(nomerprib\_.Text)));

command.Parameters.Add(new SqlParameter("@Дата", date\_.Value));

command.Parameters.Add(new SqlParameter("@ФиоПоверяющего", fio\_.Text));

command.Parameters.Add(new SqlParameter("@Результат", res\_.Text));

command.ExecuteNonQuery();

}

catch (Exception)

{

MessageBox.Show("Введите корректные данные");

}

Form1 main = this.Owner as Form1;

if (main != null)

{

main.Refreshh();

}

}

}

}

}

}

Форма добавления записей о калибровке:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Kurs

{

public partial class AddKalibrovkaForm : Form

{

private SqlDataAdapter \_dataAdapter = new SqlDataAdapter();

public AddKalibrovkaForm()

{

InitializeComponent();

}

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

{

\_dataAdapter = new SqlDataAdapter("select \* from dbo.Организация", ConnectToSql.ConnectionString);

var table = new DataTable();

\_dataAdapter.Fill(table);

organ\_.DataSource = table.Rows.OfType<DataRow>().Select(dr => dr.Field<string>("НазваниеОрганизации")).ToList();

\_dataAdapter = new SqlDataAdapter("select \* from dbo.Прибор", ConnectToSql.ConnectionString);

var table1 = new DataTable();

\_dataAdapter.Fill(table1);

nomerprib\_.DataSource = table1.Rows.OfType<DataRow>().Select(dr => dr.Field<int>("ЗаводскойНомер")).ToList();

}

private void button1\_Click(object sender, EventArgs e)

{

using (var connection = new SqlConnection(ConnectToSql.ConnectionString))

{

connection.Open();

using (var command = new SqlCommand("P17", connection))

{

try

{

command.CommandType = CommandType.StoredProcedure;

command.Parameters.Add(new SqlParameter("@НомерПротокола", int.Parse(nomerprot\_.Text)));

command.Parameters.Add(new SqlParameter("@НазваниеОрганизации", organ\_.Text));

command.Parameters.Add(new SqlParameter("@НомерПрибора", int.Parse(nomerprib\_.Text)));

command.Parameters.Add(new SqlParameter("@Дата", date\_.Value));

command.Parameters.Add(new SqlParameter("@ФиоКалибрующего", fio\_.Text));

command.Parameters.Add(new SqlParameter("@Результат", res\_.Text));

command.ExecuteNonQuery();

}

catch (Exception)

{

MessageBox.Show("Введите корректные данные");

}

Form1 main = this.Owner as Form1;

if (main != null)

{

main.Refreshh();

}

}

}

}

}

}

Форма добавления записей о ремонте:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Kurs

{

public partial class AddRemontForm : Form

{

private SqlDataAdapter \_dataAdapter = new SqlDataAdapter();

public AddRemontForm()

{

InitializeComponent();

}

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

{

\_dataAdapter = new SqlDataAdapter("select \* from dbo.Организация", ConnectToSql.ConnectionString);

var table = new DataTable();

\_dataAdapter.Fill(table);

organ\_.DataSource = table.Rows.OfType<DataRow>().Select(dr => dr.Field<string>("НазваниеОрганизации")).ToList();

\_dataAdapter = new SqlDataAdapter("select \* from dbo.Прибор", ConnectToSql.ConnectionString);

var table1 = new DataTable();

\_dataAdapter.Fill(table1);

nomerprib\_.DataSource = table1.Rows.OfType<DataRow>().Select(dr => dr.Field<int>("ЗаводскойНомер")).ToList();

}

private void button1\_Click(object sender, EventArgs e)

{

using (var connection = new SqlConnection(ConnectToSql.ConnectionString))

{

connection.Open();

using (var command = new SqlCommand("P18", connection))

{

try

{

command.CommandType = CommandType.StoredProcedure;

command.Parameters.Add(new SqlParameter("@НомерПротокола", int.Parse(nomerprot\_.Text)));

command.Parameters.Add(new SqlParameter("@НазваниеОрганизации", organ\_.Text));

command.Parameters.Add(new SqlParameter("@НомерПрибора", int.Parse(nomerprib\_.Text)));

command.Parameters.Add(new SqlParameter("@ДатаНач", daten\_.Value));

command.Parameters.Add(new SqlParameter("@ДатаКон", datek\_.Value));

command.Parameters.Add(new SqlParameter("@ФиоРемонтирующего", fio\_.Text));

command.ExecuteNonQuery();

}

catch (Exception)

{

MessageBox.Show("Введите корректные данные");

}

Form1 main = this.Owner as Form1;

if (main != null)

{

main.Refreshh();

}

}

}

}

}

}