****

**БЛАНК ЗАДАНИЯ**

Бланк задания, подписанный руководителем курсовой работы и заведующим кафедрой приведен на Рисунке 1.

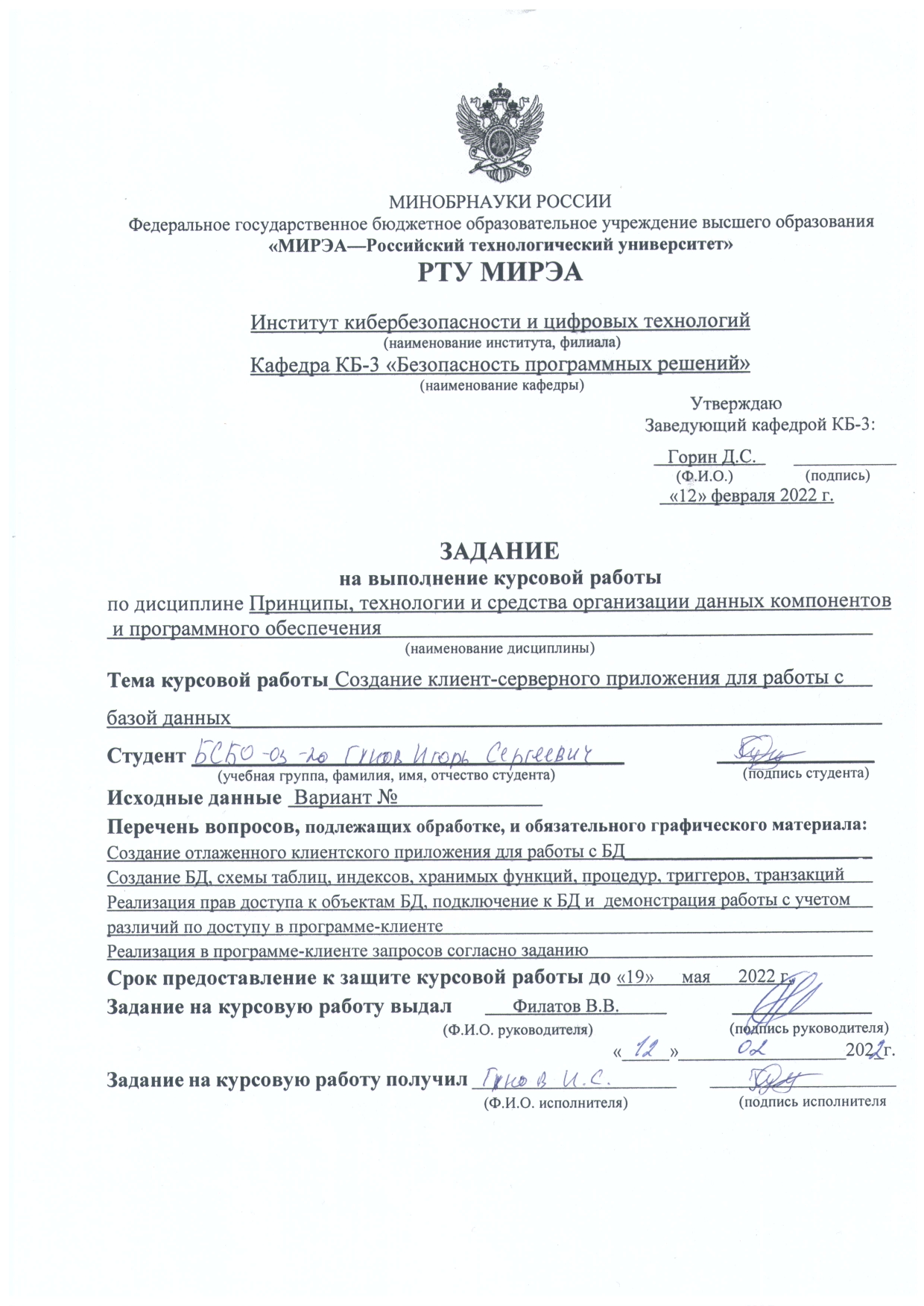


Рисунок 1 – Бланк задания

Расширенное задание заключается в разработке клиент-серверного приложения, серверная часть которого должна быть реализована на Microsoft SQL Server или PostgreSQL, представляющая собой модель предметной области в соответствии с вариантом задания. В рамках заданной предметной области необходимо реализовать заданную (по варианту) схему отношений,

т.е. выделить сущности и их атрибуты, так чтобы связи между сущностями соответствовали представленной схеме. Допускается небольшое отступление от заданной схемы. В рамках курсовой работы необходимо на стороне сервера реализовать и использовать при демонстрации приложения следующие компоненты:

1. Постоянные таблицы и связи между ними, количество таблиц и наличие связей должно соответствовать заданию, допускается увеличение числа таблиц и их полей для более адекватного представления предметной области;
2. В приложении (на стороне клиента) реализовать не менее пяти запросов (для демонстрации навыков работы), которые могут реализовывать задания из п.3.
3. Реализовать запросы по заданиям (в любых фрагментах скриптов как на стороне сервера, так и на стороне клиента):
4. Составной многотабличный запрос с CASE-выражением;
5. Многотабличный VIEW, с возможностью его обновления;
6. Запросы, содержащие подзапрос в разделах SELECT, FROM и WHERE (в каждом хотя бы по одному);
7. Коррелированные подзапросы (минимум 3 запроса).
8. Многотабличный запрос, содержащий группировку записей, агрегатные функции и параметр, используемый в разделе HAVING;
9. Запросы, содержащий предикат ANY(SOME) или ALL (для каждого предиката);
10. Создать индексы (минимум 3 штуки) для увеличения скорости выполнения запросов; Предусмотреть индексы разных типов. Индексы должны быть созданы для разных таблиц. В отчет включить план запроса, показывающий применение индекса при выполнении запроса.
11. В таблице (в соответствии с вариантом) предусмотреть поле, которое заполняется (и актуализируется) автоматически по срабатыванию триггера при добавлении, обновлении и удалении данных, иметь возможность

продемонстрировать работу триггера при работе приложения. Триггеры должны обрабатывать только те записи, которые были добавлены, изменены или удалены в ходе текущей операции (транзакции).

1. Операции добавления, удаления и обновления реализовать в виде хранимых процедур или функций с параметрами для всех таблиц;
2. Реализовать отдельную хранимую процедуру или функцию, состоящую из нескольких отдельных операций в виде единой транзакции, которая при определенных условиях может быть зафиксирована или откатана;
3. Реализовать курсор на обновления отдельных данных (вычисления значения полей выбранной таблицы);
4. Реализовать собственную скалярную и векторную функции. Функции сохранить в базе данных;
5. Распределение прав пользователей: предусмотреть не менее двух пользователей с разным набором привилегий. Каждый набор привилегий оформить в виде роли.

Вариант:43

|  |  |  |  |
| --- | --- | --- | --- |
| 43 | Метрологическая служба предприятия | Ведение БД по измерительным и поверочным средствам.  Формирование справок и графика поверок (проверок) оборудования.  Ответственные сотрудники. | 4 |

|  |  |  |  |
| --- | --- | --- | --- |
| **№ пункта задания** | **Краткое наименование (суть критерия)** | **Дополнительное требование** | **Балл** |
| 1 | Соответствие схемы БД схеме задания (п.1 задания на КР) |  |  |
| 2 | Индексирование БД (физическая и логическая  организация БД) |  |  |
| 3 | Реализация и объяснение работы запросов (п.3.) в SQL-скриптах | CASE |  |
| VIEW |  |
| Подзапрос в SELECT |  |
| Подзапрос в FROM |  |
| Подзапрос в WHERE |  |
| Коррелированный подзапрос |  |
| Агрегативные функции,  группировка записей |  |
| Предикаты ANY(SOME)  или ALL |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 4 | Объяснение работы триггеров |  |  |
| 5 | Использование хранимых процедур |  |  |
| 6 | Использование в запросах пользовательских  хранимых функций |  |  |
| 7 | Реализация и объяснение  работы курсора |  |  |
| 8 | Права, роли,  разграничение доступа, разрешение и запрет |  |  |
| 9 | Демонстрация работы клиента в виде отдельного приложения |  |  |
| 10 | Ввод, модификация, удаление данных на стороне клиента |  |  |
| 11 | Демонстрация работы запросов на стороне клиента (п.2 задания на КР) |  |  |
| 12 | Оформление КР | Пункты 1-11 в части оформления курсовой  работы |  |

ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 8](#_Toc106009370)

[1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 9](#_Toc106009371)

[2 ПРАКТИЧЕСКАЯ ЧАСТЬ 10](#_Toc106009372)

[2.1 Схема базы данных метрологической службы предприятия 10](#_Toc106009373)

[2.2 Создание пользователей и разграничение прав доступа 11](#_Toc106009374)

[2.3 Протокол настройки СУБД 12](#_Toc106009375)

[2.4 Протокол работы программы (клиента) 21](#_Toc106009376)

[2.5 Исходный код программы 28](#_Toc106009377)

[3 ЗАКЛЮЧЕНИЕ 29](#_Toc106009378)

[4 СПИСОК ИСТОЧНИКОВ 30](#_Toc106009379)

[5 ПРИЛОЖЕНИЕ 1 31](#_Toc106009380)

# ВВЕДЕНИЕ

Целью данной курсовой работы является автоматизация работы метрологической службы предприятия.

Задачами являются:

* ведение реестра измерительных средств (инструментов);
* ведение реестра организаций – держателей поверочных средств (организации);
* формирование справок (связь сотрудника и инструмента, история поверки);
* формирование графика поверки в табличном представлении, с указанием наименования организации, инструмента и даты поверки.

Объект исследования – метрологическая служба предприятия.

Предмет исследования – поверка инструментов.

Используемые методы исследования: поиск в Internet и опрос.

Информационная база исследования – тема курсовой работы.

Курсовая работа состоит из описательной части и исходного кода.

# ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Метрологическая служба - юридическое лицо, подразделение юридического лица или объединение юридических лиц, либо работник (работники) юридического лица, либо индивидуальный предприниматель, либо подведомственная организация федерального органа исполнительной власти, его подразделение или должностное лицо, выполняющие работы и (или) оказывающие услуги по обеспечению единства измерений и действующие на основании положения о метрологической службе [1]. Метрологическая служба (далее - МС) предприятия обеспечивает организационное и техническое поддержание в надлежащем состоянии измерительных инструментов, необходимых как в ходе основного технологического процесса, так и предназначенных для удовлетворения внутренних потребностей, с необходимой производительностью, точностью, экономической эффективностью, при условии соблюдения всех технических и нормативных требований [2].

Измерения – единственный способ получения объективной информации [3]. С целью подтверждения технических характеристик инструментов МС осуществляет поверку средств измерений в поверочных организациях. Различают следующие виды поверок: первичная, периодическая, внеочередная, инспекционная [4]. В текущем исследовании учитывается периодическая поверка.

За сотрудниками МС закреплены инструменты, которые они контролируют, в свою очередь инструменты закреплены за организацией или организациями, в которых можно осуществлять их поверку в соответствии с межповерочным интервалом.

С целью реализации задач исследования создано клиент-серверное приложение на языке С++ в фреймворке QT с использованием системы управления базами данных MySQL. Реализовано 3 обычные таблицы (инструменты, организации, сотрудники) и 2 таблицы связи (сотрудник и инструмент, за который он отвечает, организация и инструмент, который поверяли в этой организации и дата поверки) – связь многие ко многим.

# ПРАКТИЧЕСКАЯ ЧАСТЬ

## Схема базы данных метрологической службы предприятия

Схема базы данных метрологической службы предприятия приведена на Рисунке 2.

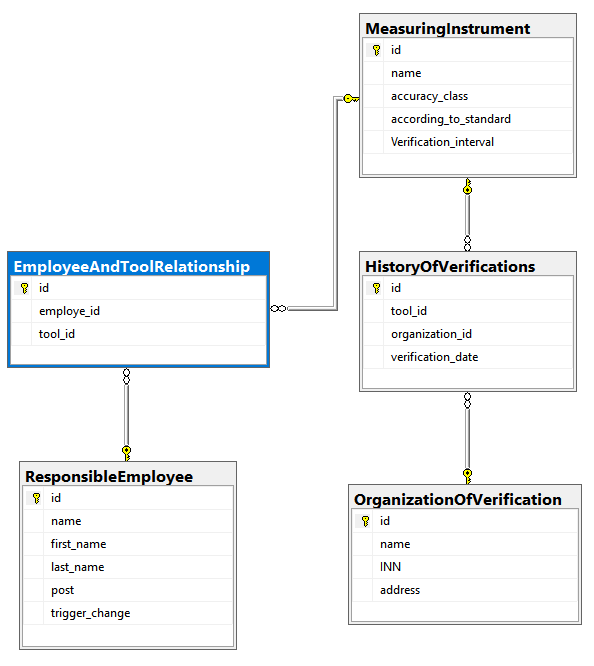


Рисунок 2 - Схема базы данных метрологической службы предприятия

## Создание пользователей и разграничение прав доступа

Пользователь с правами администратора имеет возможность добавлять, изменять, удалять записи во всех таблицах. Создание пользователя Igor с правами администратора (реализовано с помощью источника [5]) приведено на Рисунке 3.

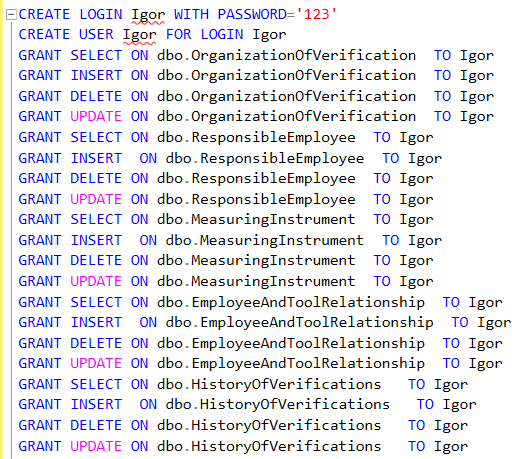


Рисунок 3 - Создание пользователя Igor с правами администратора

Пользователь с правами пользователя имеет возможность только просматривать данные. Создание пользователя Alex с правами пользователя приведено на Рисунке 4.

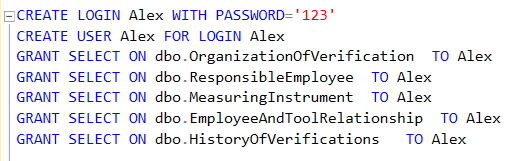


Рисунок 4 - Создание пользователя Alex с правами пользователя

## Протокол настройки СУБД

**Создание таблиц:**

* Таблица организаций (см. Рисунок 5):

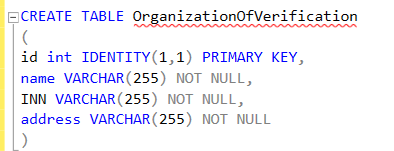


Рисунок 5 – Создание таблицы организаций

* Таблица сотрудников (см. Рисунок 6):

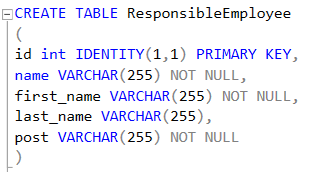


Рисунок 6 – Создание таблицы сотрудников

* Таблица инструмента (см. Рисунок 7):

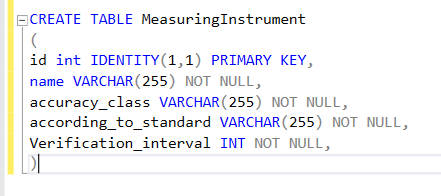


Рисунок 7 – Создание таблицы инструментов

* Таблица связи сотрудника и инструмента (см. Рисунок 8):

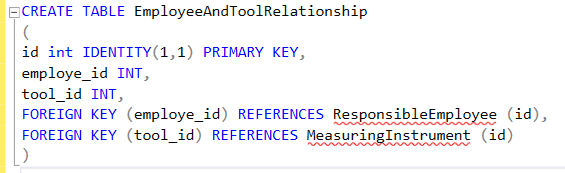


Рисунок 8 – Создание таблицы связи сотрудника и инструмента

* Таблица истории поверки (см. Рисунок 9):

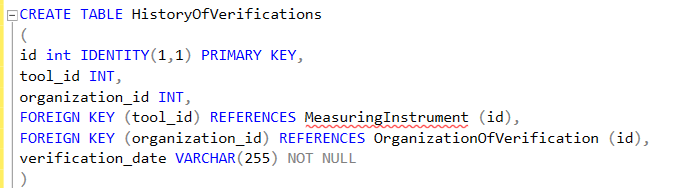
****

Рисунок 9 – Создание таблицы истории поверки

**Создание индексов:**

* Создание некластеризованного индекса и неповторяющегося некластеризованного индекса в таблице (см. Рисунок 10, 11), отображение индексов в MySQL приведено на Рисунке 12:



Рисунок 10 - Создание некластеризованного индекса в таблице

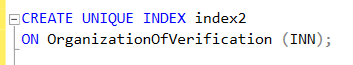


Рисунок 11- Создание неповторяющегося некластеризованного индекса в таблице

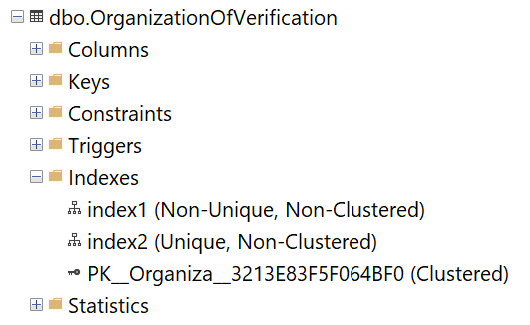


Рисунок 12 - Создание неповторяющегося некластеризованного индекса в таблице

* Пример использования (см. Рисунок 13):

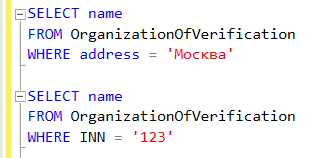
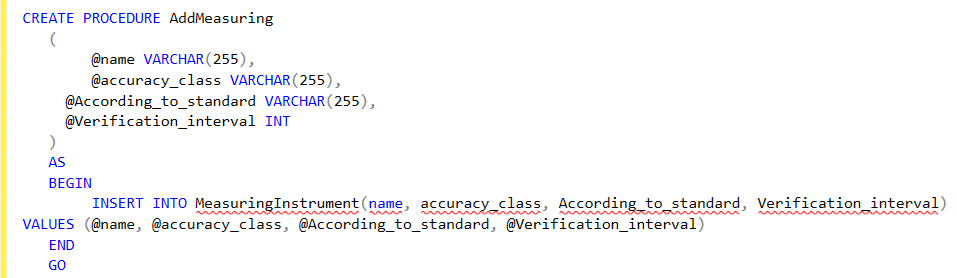
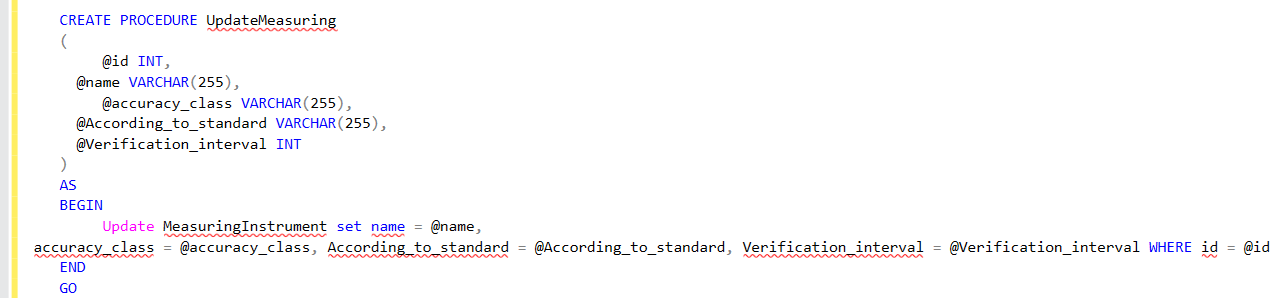


Рисунок 13 – Пример использования

**Хранимые процедуры:**

* Добавление, обновление, удаление инструмента (см. Рисунок 14):





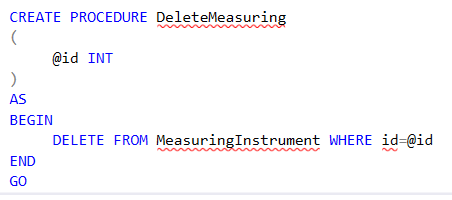
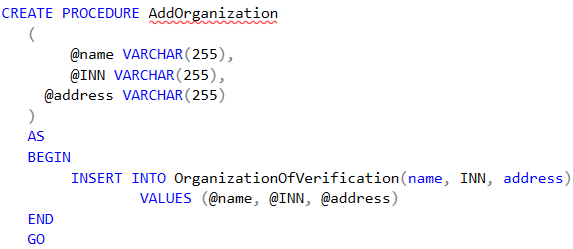
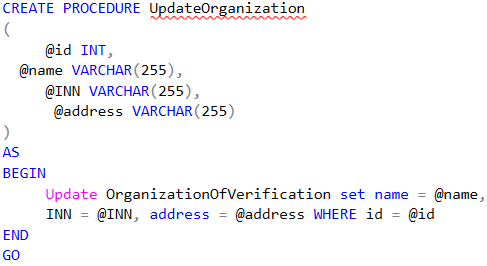


Рисунок 14 - Добавление, обновление, удаление инструмента

* Добавление, обновление, удаление организации (см. Рисунок 15):





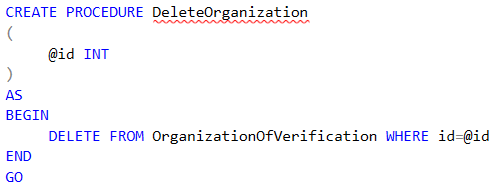
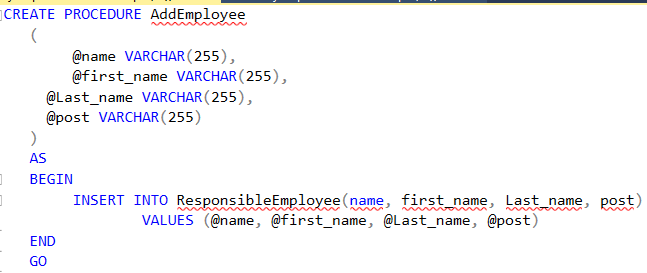
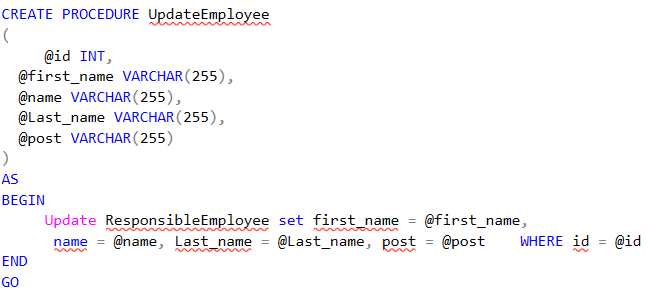


Рисунок 15 - Добавление, обновление, удаление организации

* Добавление, обновление, удаление сотрудника (см. Рисунок 16):





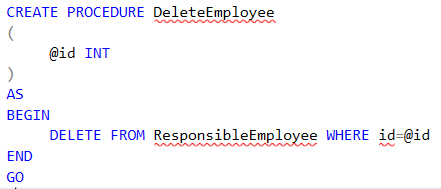
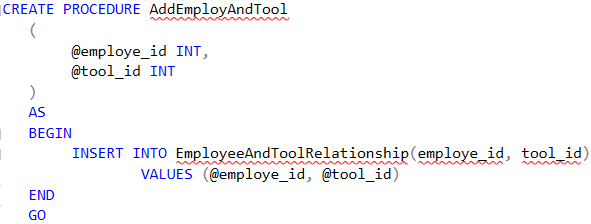
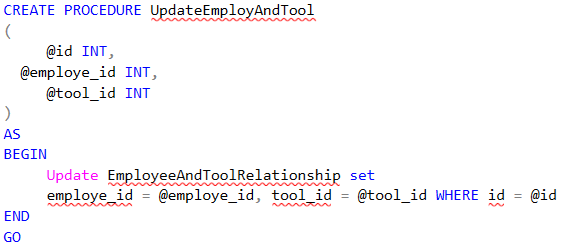


Рисунок 16 - Добавление, обновление, удаление сотрудника

* Добавление, обновление, удаление связи инструмента и сотрудника (см. Рисунок 17):





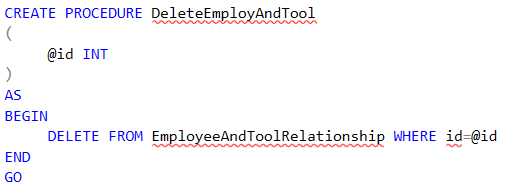
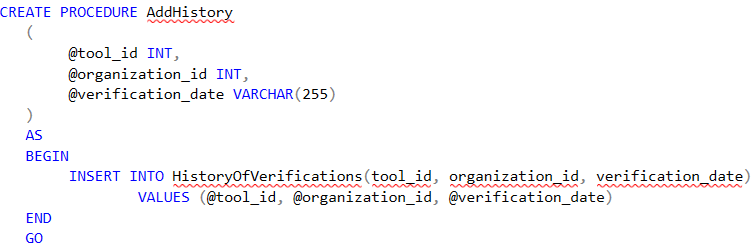
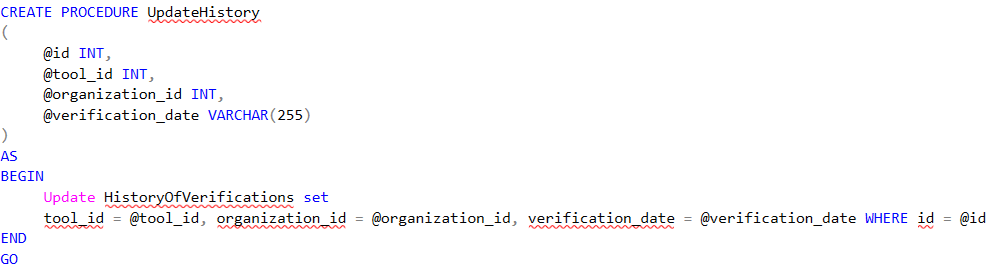


Рисунок 17 - Добавление, обновление, удаление связи инструмента и сотрудника

* Добавление, обновление, удаление истории поверки (см. Рисунок 18):





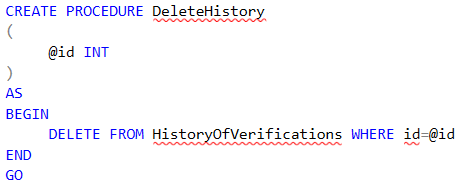


Рисунок 18 - Добавление, обновление, удаление истории поверки

* Составной многотабличный запрос с CASE-выражением (см. Рисунок 19, 21):

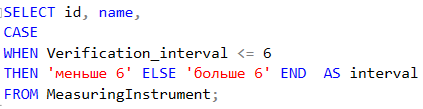


Рисунок 19 - Составной многотабличный запрос с CASE-выражением

Результат запроса приведен на Рисунок 20.



Рисунок 20 – Результат составного многотабличного запроса 1 с CASE-выражением

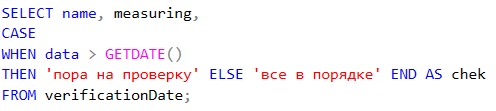


Рисунок 21 - Составной многотабличный запрос 2 с CASE-выражением

Результат запроса приведен на Рисунок 22.

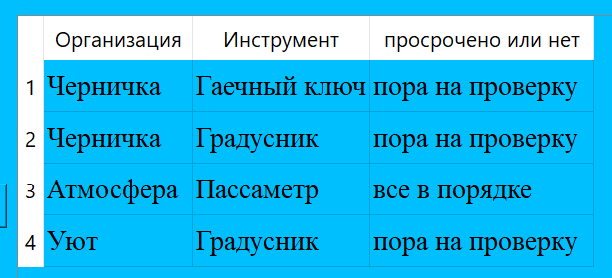


Рисунок 22 – Результат составного многотабличного запроса 2 с CASE-выражением

* Многотабличный VIEW, с возможностью его обновления (см. Рисунок 23):

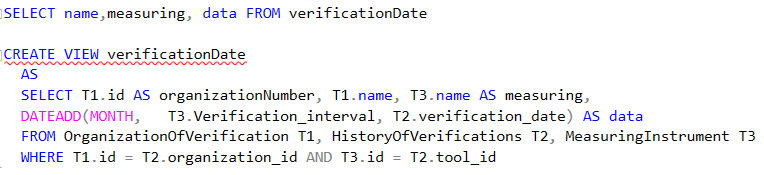


Рисунок 23 - Многотабличный VIEW, с возможностью его обновления

Результат запроса приведен на Рисунок 24



Рисунок 24 – Результат многотабличного VIEW, с возможностью его обновления

* Запрос, содержащий вложенные запросы SELECT, FROM и WHERE (см. Рисунок 25, 27):

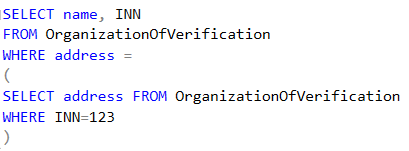


Рисунок 25 - Запрос, содержащий вложенный WHERE

Результат запроса приведен на Рисунок 26.

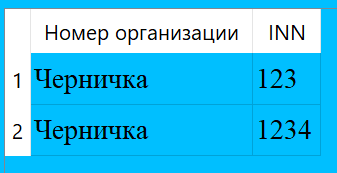


Рисунок 26 – Результат запроса 1, содержащего коррелированные и некоррелированные подзапросы в разделах SELECT, FROM и WHERE

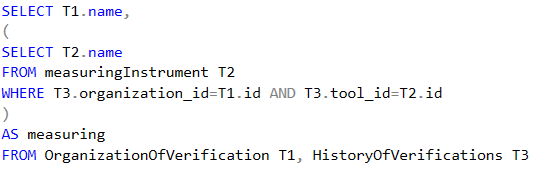


Рисунок 27 - Запрос, содержащий вложенный SELECT

Результат запроса приведен на Рисунок 28.

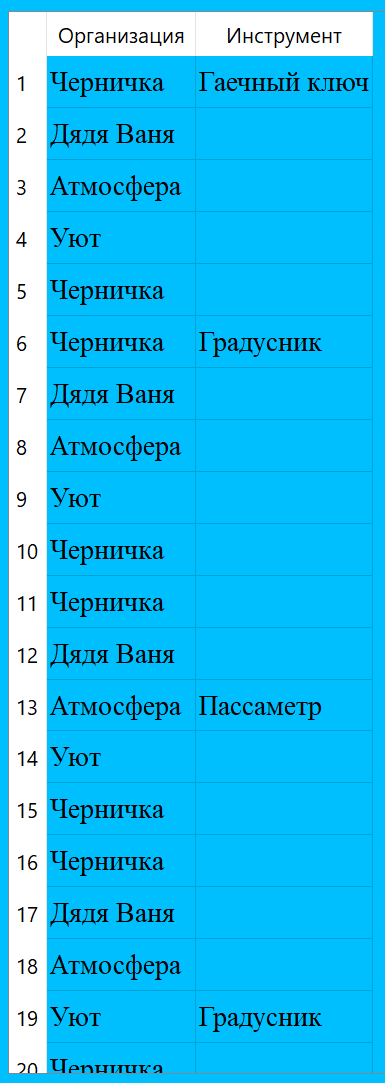


Рисунок 28 – Результат запроса 2, содержащего коррелированные и некоррелированные подзапросы в разделах SELECT, FROM и WHERE

* Многотабличный запрос, содержащий группировку записей, агрегатные функции и параметр, используемый в разделе HAVING (см. Рисунок 29):

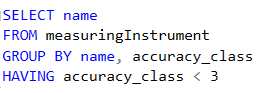


Рисунок 29 - Многотабличный запрос, содержащий группировку записей, агрегатные функции и параметр, используемый в разделе HAVING

Результат запроса приведен на Рисунок 30.

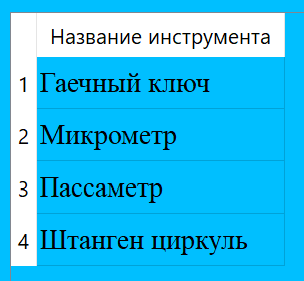


Рисунок 30 – Результат многотабличного запроса, содержащего группировку записей, агрегатные функции и параметр, используемый в разделе HAVING

* Запрос, содержащий предикат ALL (см. Рисунок 31):

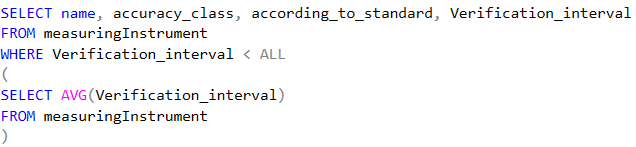


Рисунок 31 - Запрос, содержащий предикат ALL

Результат запроса приведен на Рисунок 32.



Рисунок 32 – Результат запроса, содержащего предикат ALL

* GROUP BY (см. Рисунок 33):



Рисунок 33 - GROUP BY

Результат запроса приведен на Рисунок 34.



Рисунок 34 – Результат использования группировки

* Создание триггера (см. Рисунок 35):

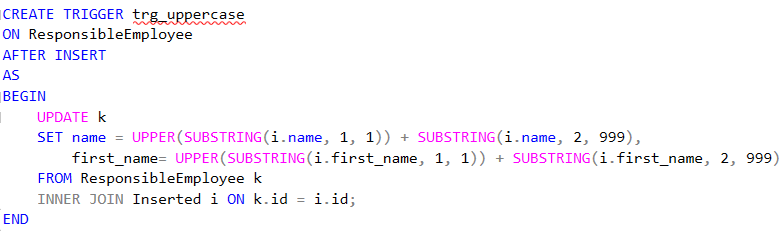


Рисунок 35 - Создание триггера

* Создание хранимой процедуры, которая может быть откатана (реализовано с помощью источника [6]) (см. Рисунок 36):

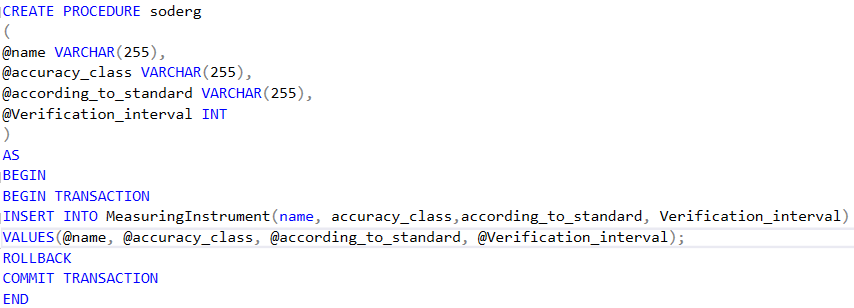


Рисунок 36 - Создание хранимой процедуры, которая может быть откатана

* Создание курсора (реализовано с помощью источника [7]) (см. Рисунок 37):



Рисунок 37 - Создание курсора

## Протокол работы программы (клиента)

Авторизация приведена на Рисунок 38.

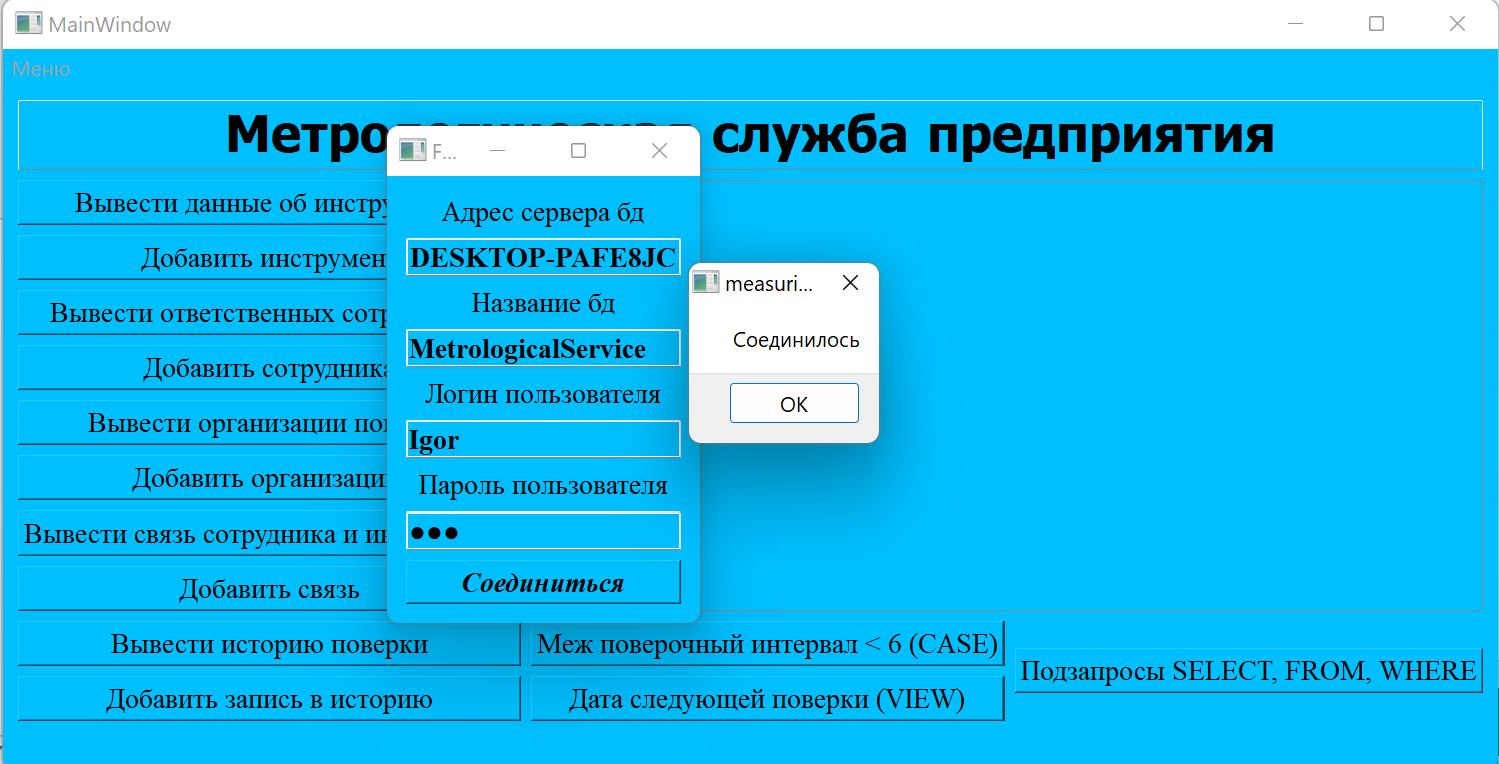


Рисунок 38 - Авторизация

Добавление инструмента приведено на Рисунок 39.

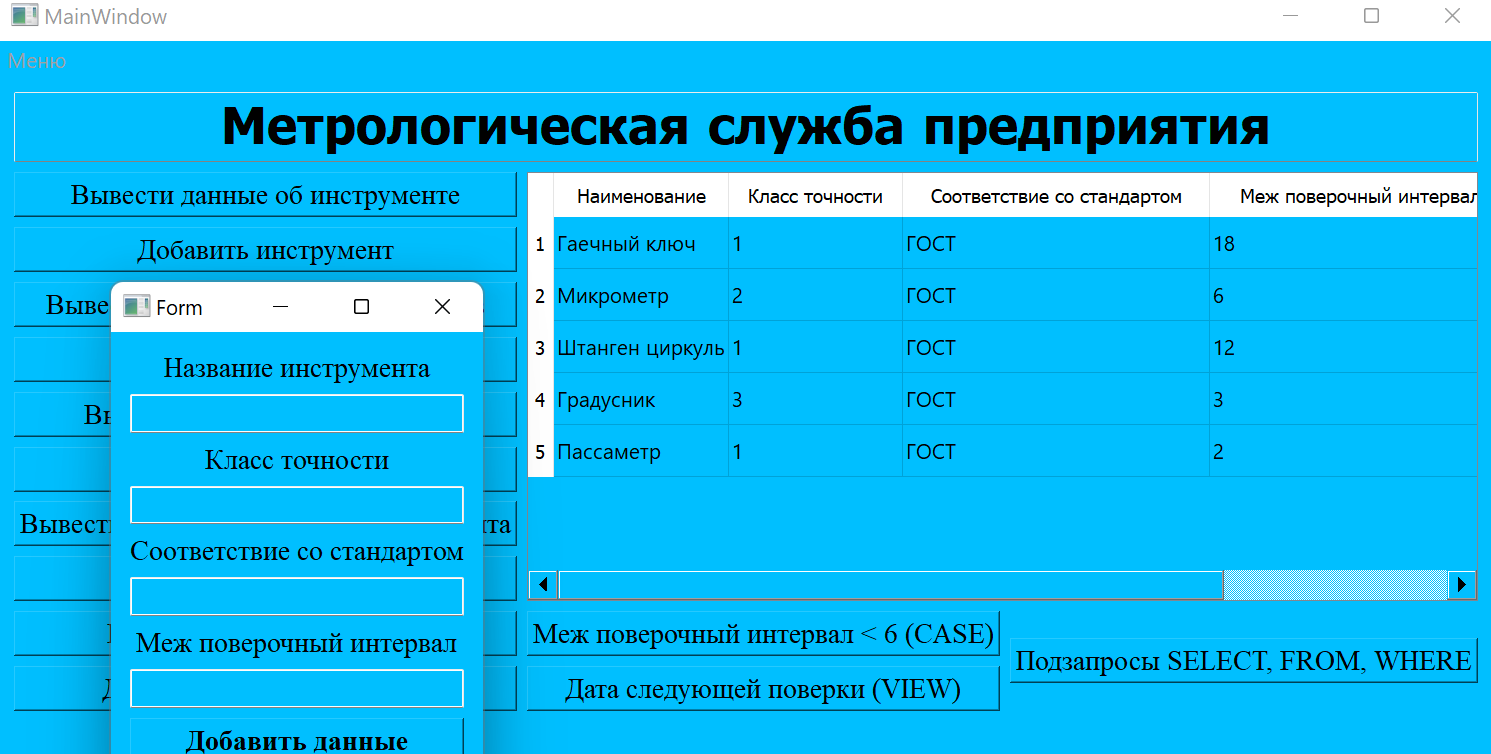


Рисунок 39 - Добавление инструмента

Изменение информации об инструменте приведено на Рисунок 40.

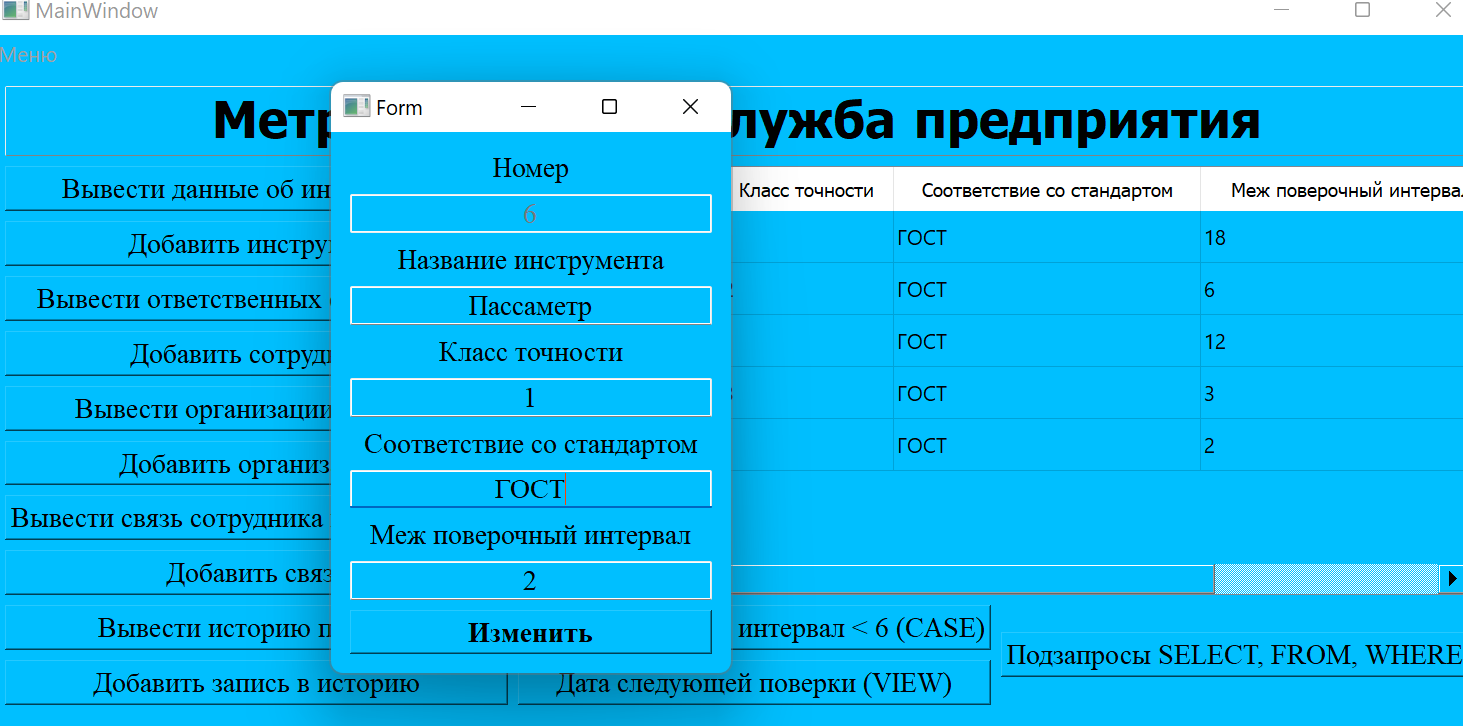


Рисунок 40 - Изменение информации об инструменте

Работа триггера и добавление сотрудника приведена на Рисунке 41 -42.

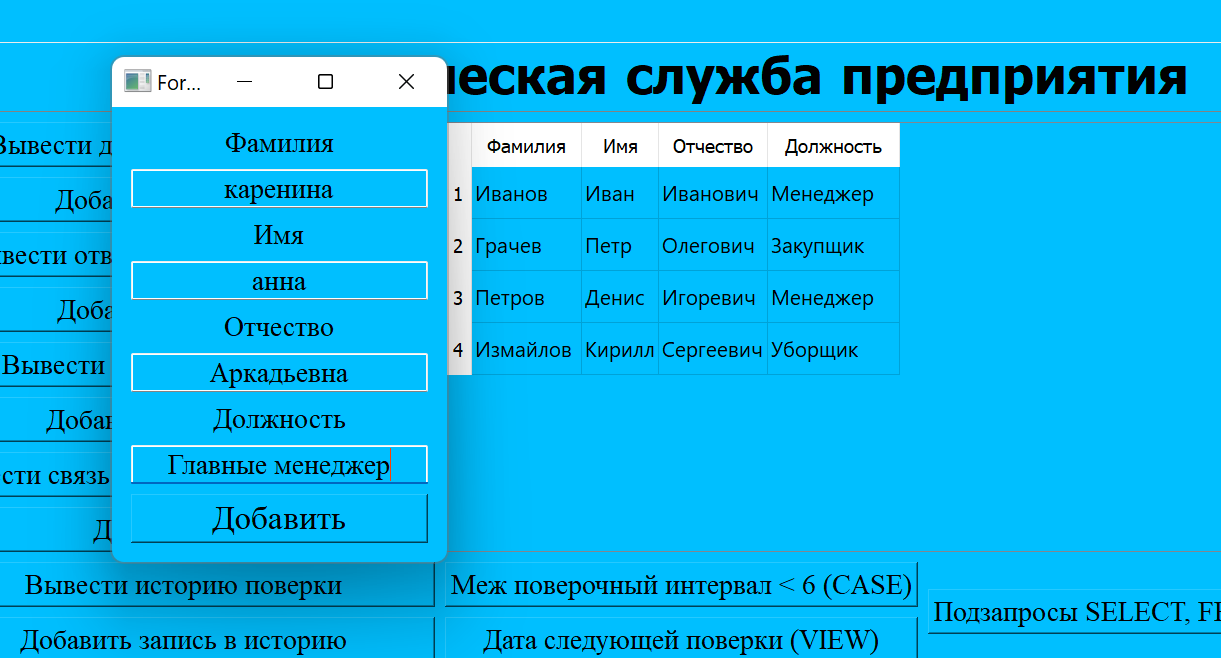


Рисунок 41 - Работа триггера и добавление сотрудника 1

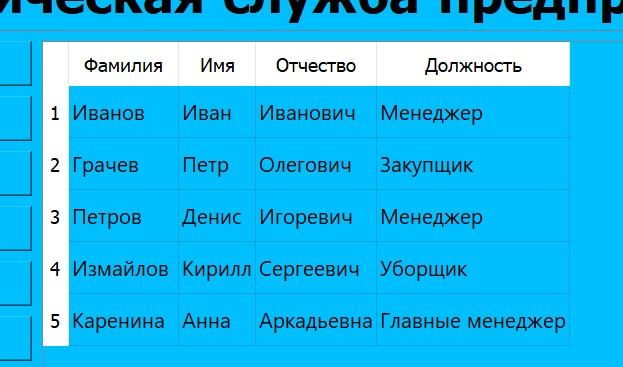


Рисунок 42 - Работа триггера и добавление сотрудника 2

Всплывающее меню, позволяющее изменять и удалять записи приведено на Рисунке 43.

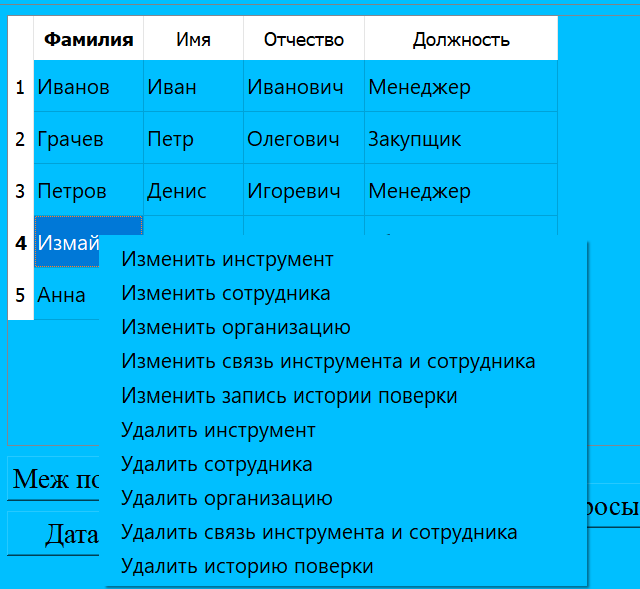


Рисунок 43 – Всплывающее меню

Изменение данных о сотруднике приведено на Рисунок 44.

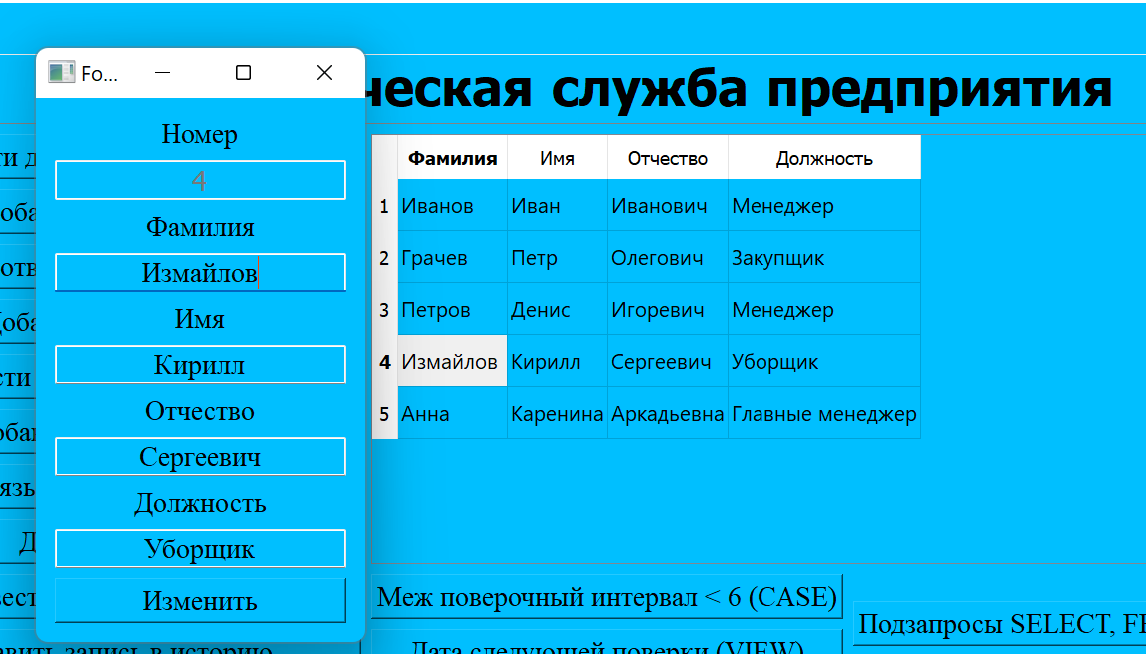


Рисунок 44 - Изменение данных о сотруднике

Добавление организации приведено на Рисунок 45.



Рисунок 45 – Добавление организации

Изменение сведений об организации приведено на Рисунок 46.



Рисунок 46 - Изменение сведений об организации

Добавление связи сотрудника и инструмента приведено на Рисунок 47.

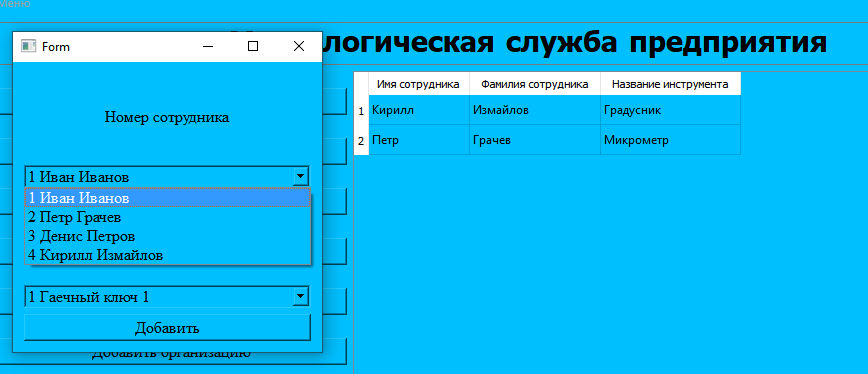


Рисунок 47 - Добавление связи сотрудника и инструмента

Изменение связи сотрудника и инструмента приведено на Рисунок 48.

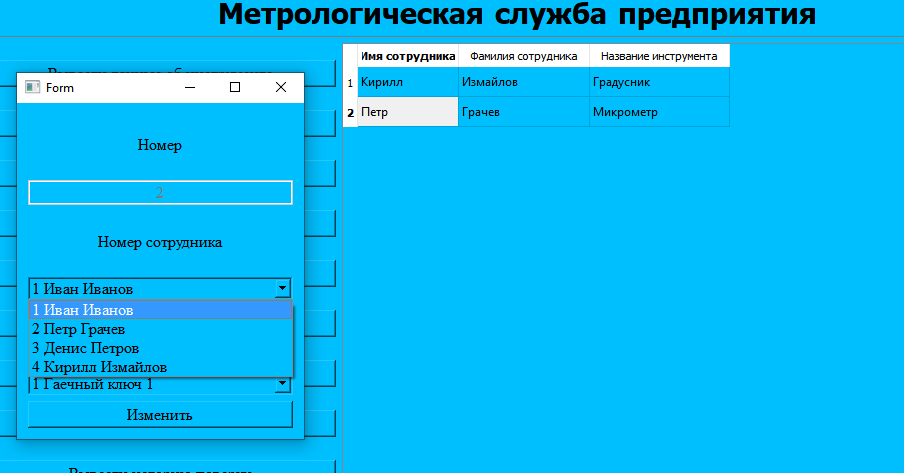


Рисунок 48 - Изменение связи сотрудника и инструмента

Добавление истории поверки приведено на Рисунок 49.

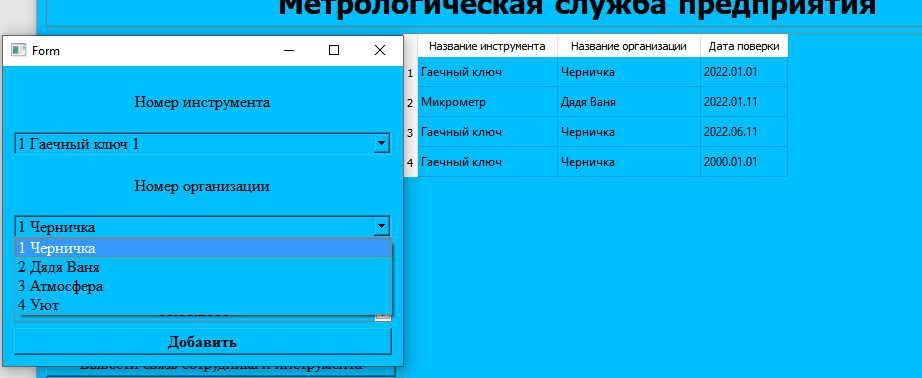


Рисунок 49 - Добавление истории поверки

Изменение истории поверки приведено на Рисунок 50.

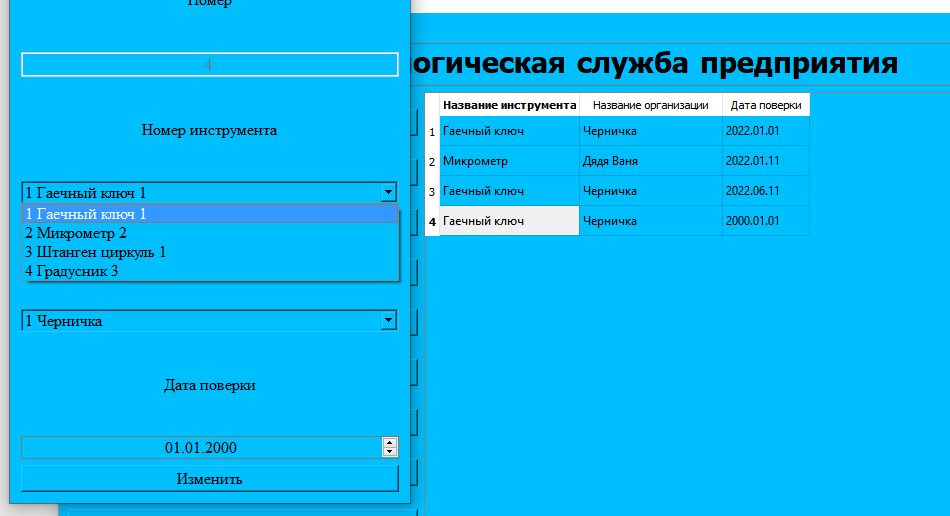


Рисунок 50 - Изменение истории поверки

## Исходный код программы

Исходный код программы приведен в Приложении А.

# ЗАКЛЮЧЕНИЕ

В рамках курсовой работы реализовано приложение, автоматизирующее работу метрологической службы предприятия, которое позволяет вести реестр используемых на предприятии инструментов, вести реестр организаций – держателей поверочных средств, формировать справки о связи сотрудника и инструмента и справку истории поверки, формировать в табличном виде график поверки.

# СПИСОК ИСТОЧНИКОВ

1. Федеральный закон от 26.06.2008 № 102-ФЗ «Об обеспечении единства измерений». — URL: http://www.consultant.ru/document/cons\_doc\_LAW\_77904/ (дата обращения 03.06.2022).
2. Метрологическая служба предприятия [Электронный ресурс]. — URL: <https://info.metrologu.ru/spravochnik/obespechenie_edinstva_izmereniy-1/metrologicheskaya_sluzhba_predpriyatiya.html> (дата обращения 03.06.2022).
3. Артемьев Б.Г., Лукашов Ю.Е. Справочное пособие для специалистов метрологических служб. –М.: ИПК Издательство стандартов, 2004.- 646 с.
4. Хамханова Д.Н. Прикладная метрология. Учебное пособие. [Электронная библиотека]. - URL: http/wwwbookree.org (дата обращения 03.06.2022).
5. Статьи о программировании [Электронный ресурс]. — URL: <https://habr.com/ru/all/> (дата обращения 03.06.2022).
6. Сайт вопросов и ответов для программистов [Электронный ресурс]. — URL: <https://ru.stackoverflow.com/> (дата обращения 03.06.2022).
7. Форум программистов и сисадминов Киберфорум [Электронный ресурс]. — URL: <https://www.cyberforum.ru/> (дата обращения 03.06.2022).

# ПРИЛОЖЕНИЕ А

**Исходный код**

**Заголовочные**

#ifndef ADDHISTORY\_H

#define ADDHISTORY\_H

#include <QWidget>

#include <QMessageBox>

#include <QSqlQuery>

#include <QTimer>

namespace **Ui** {

class **addHistory**;

}

class **addHistory** : public QWidget

{

Q\_OBJECT

public:

explicit **addHistory**(QWidget \*parent = nullptr);

~***addHistory***();

int postcombo;

int postcombo2;

private slots:

void **on\_pushButton\_clicked**();

void **on\_comboBox\_currentIndexChanged**(int index);

void **on\_comboBox\_2\_currentIndexChanged**(int index);

signals:

void **refresh\_table**();

private:

Ui::addHistory \*ui;

};

#endif // ADDHISTORY\_H

#ifndef ADDRECORD\_H

#define ADDRECORD\_H

#include <QWidget>

#include <QMessageBox>

#include <QSqlQuery>

namespace **Ui** {

class **addrecord**;

}

class **addrecord** : public QWidget

{

Q\_OBJECT

public:

explicit **addrecord**(QWidget \*parent = nullptr);

~***addrecord***();

private slots:

void **on\_pushButton\_clicked**();

signals:

void **refresh\_table**();

private:

Ui::addrecord \*ui;

};

#endif // ADDRECORD\_H

#ifndef ADDRECORD2\_H

#define ADDRECORD2\_H

#include <QWidget>

#include <QMessageBox>

#include <QSqlQuery>

namespace **Ui** {

class **addrecord2**;

}

class **addrecord2** : public QWidget

{

Q\_OBJECT

public:

explicit **addrecord2**(QWidget \*parent = nullptr);

~***addrecord2***();

private slots:

void **on\_pushButton\_clicked**();

signals:

void **refresh\_table**();

private:

Ui::addrecord2 \*ui;

};

#endif // ADDRECORD2\_H

#ifndef ADDRECORD3\_H

#define ADDRECORD3\_H

#include <QWidget>

#include <QMessageBox>

#include <QSqlQuery>

namespace **Ui** {

class **addrecord3**;

}

class **addrecord3** : public QWidget

{

Q\_OBJECT

public:

explicit **addrecord3**(QWidget \*parent = nullptr);

~***addrecord3***();

int postcombo;

int postcombo2;

private slots:

void **on\_pushButton\_clicked**();

void **on\_comboBox\_2\_currentIndexChanged**(int index);

void **on\_comboBox\_currentIndexChanged**(int index);

signals:

void **refresh\_table**();

private:

Ui::addrecord3 \*ui;

};

#endif // ADDRECORD3\_H

#ifndef FORM\_H

#define FORM\_H

#include <QWidget>

#include <QMessageBox>

#include <QSqlQuery>

namespace **Ui** {

class **Form**;

}

class **Form** : public QWidget

{

Q\_OBJECT

public:

explicit **Form**(QWidget \*parent = nullptr);

~***Form***();

private slots:

void **on\_pushButton\_clicked**();

signals:

void **refresh\_table**();

private:

Ui::Form \*ui;

};

#endif // FORM\_H

#ifndef IZMENENIE\_H

#define IZMENENIE\_H

#include <QWidget>

#include <QSqlQuery>

namespace **Ui** {

class **izmenenie**;

}

class **izmenenie** : public QWidget

{

Q\_OBJECT

public:

explicit **izmenenie**(QWidget \*parent = nullptr);

~***izmenenie***();

private slots:

void **obr\_sendId**(int);

void **on\_pushButton\_clicked**();

private:

Ui::izmenenie \*ui;

};

#endif // IZMENENIE\_H

#ifndef IZMENENIE2\_H

#define IZMENENIE2\_H

#include <QWidget>

#include <QSqlQuery>

namespace **Ui** {

class **izmenenie2**;

}

class **izmenenie2** : public QWidget

{

Q\_OBJECT

public:

explicit **izmenenie2**(QWidget \*parent = nullptr);

~***izmenenie2***();

private slots:

void **obr\_sendId**(int);

void **on\_pushButton\_clicked**();

private:

Ui::izmenenie2 \*ui;

};

#endif // IZMENENIE2\_H

#ifndef IZMENENIE3\_H

#define IZMENENIE3\_H

#include <QWidget>

#include <QSqlQuery>

namespace **Ui** {

class **izmenenie3**;

}

class **izmenenie3** : public QWidget

{

Q\_OBJECT

public:

explicit **izmenenie3**(QWidget \*parent = nullptr);

~***izmenenie3***();

private slots:

void **obr\_sendId**(int);

void **on\_pushButton\_clicked**();

private:

Ui::izmenenie3 \*ui;

};

#endif // IZMENENIE3\_H

#ifndef IZMENENIE4\_H

#define IZMENENIE4\_H

#include <QWidget>

#include <QMessageBox>

#include <QSqlQuery>

namespace **Ui** {

class **izmenenie4**;

}

class **izmenenie4** : public QWidget

{

Q\_OBJECT

public:

explicit **izmenenie4**(QWidget \*parent = nullptr);

~***izmenenie4***();

int postcombo;

int postcombo2;

private slots:

void **obr\_sendId**(int);

void **on\_pushButton\_clicked**();

void **on\_comboBox\_currentIndexChanged**(int index);

void **on\_comboBox\_2\_currentIndexChanged**(int index);

private:

Ui::izmenenie4 \*ui;

};

#endif // IZMENENIE4\_H

#ifndef LOGIN\_H

#define LOGIN\_H

#include <QWidget>

#include <QtSql>

#include <QMessageBox>

namespace **Ui** {

class **Login**;

}

class **Login** : public QWidget

{

Q\_OBJECT

public:

explicit **Login**(QWidget \*parent = nullptr);

~***Login***();

QMessageBox\* mes;

private slots:

void **on\_pushButton\_clicked**();

private:

Ui::Login \*ui;

QSqlDatabase db;

};

#endif // LOGIN\_H

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <QSqlQueryModel>

#include "login.h"

#include "ui\_login.h"

#include "addrecord.h"

#include "ui\_addrecord.h"

#include "addrecord2.h"

#include "ui\_addrecord2.h"

#include "addrecord3.h"

#include "ui\_addrecord3.h"

#include "izmenenie.h"

#include "ui\_izmenenie.h"

#include "izmenenie2.h"

#include "ui\_izmenenie2.h"

#include "izmenenie3.h"

#include "ui\_izmenenie3.h"

#include "izmenenie4.h"

#include "ui\_izmenenie4.h"

#include "form.h"

#include "ui\_form.h"

#include "addhistory.h"

#include "ui\_addhistory.h"

#include "subqueries.h"

#include "ui\_subqueries.h"

#include "updatehistory.h"

#include "ui\_updatehistory.h"

QT\_BEGIN\_NAMESPACE

namespace **Ui** { class **MainWindow**; }

QT\_END\_NAMESPACE

class **MainWindow** : public QMainWindow

{

Q\_OBJECT

public:

**MainWindow**(QWidget \*parent = nullptr);

~***MainWindow***();

Login\* logwin;

addrecord\* dobzap;

izmenenie\* izm;

izmenenie2\* izm2;

izmenenie3\* izm3;

izmenenie4\* izm4;

updatehistory\* izm5;

Form\* dobzap2;

addrecord2\* dobzap3;

addrecord3\* dobzap4;

addHistory\* dobzap5;

subqueries\* sub;

int globid;

int recId;

int fl;

private slots:

void **on\_action\_triggered**();

void **on\_pushButton\_clicked**();

void **on\_pushButton\_2\_clicked**();

void **obt\_ref\_tab**();

void **CustomMenuReq**(QPoint);

void **del\_zap**();

void **del\_zap2**();

void **del\_zap3**();

void **del\_zap4**();

void **del\_zap5**();

void **izm\_zap**();

void **izm\_zap2**();

void **izm\_zap3**();

void **izm\_zap4**();

void **izm\_zap5**();

void **on\_pushButton\_5\_clicked**();

void **on\_pushButton\_6\_clicked**();

void **on\_pushButton\_7\_clicked**();

void **on\_pushButton\_8\_clicked**();

void **on\_pushButton\_9\_clicked**();

void **on\_pushButton\_10\_clicked**();

void **on\_pushButton\_11\_clicked**();

void **on\_pushButton\_12\_clicked**();

void **obt\_ref\_tab2**();

void **on\_pushButton\_3\_clicked**();

void **obt\_ref\_tab3**();

void **on\_pushButton\_4\_clicked**();

void **obt\_ref\_tab4**();

void **on\_pushButton\_14\_clicked**();

void **on\_pushButton\_15\_clicked**();

void **obt\_ref\_tab5**();

signals:

void **sendId**(int);

private:

Ui::MainWindow \*ui;

QSqlQueryModel\* model;

};

#endif // MAINWINDOW\_H

#ifndef SUBQUERIES\_H

#define SUBQUERIES\_H

#include <QWidget>

#include <QMessageBox>

#include <QSqlQuery>

#include <QSqlQueryModel>

namespace **Ui** {

class **subqueries**;

}

class **subqueries** : public QWidget

{

Q\_OBJECT

public:

explicit **subqueries**(QWidget \*parent = nullptr);

~***subqueries***();

private slots:

void **on\_pushButton\_clicked**();

void **on\_pushButton\_2\_clicked**();

void **on\_pushButton\_3\_clicked**();

void **on\_pushButton\_4\_clicked**();

void **on\_pushButton\_5\_clicked**();

void **on\_pushButton\_6\_clicked**();

private:

Ui::subqueries \*ui;

QSqlQueryModel\* model;

};

#endif // SUBQUERIES\_H

#ifndef UPDATEHISTORY\_H

#define UPDATEHISTORY\_H

#include <QWidget>

#include <QMessageBox>

#include <QSqlQuery>

namespace **Ui** {

class **updatehistory**;

}

class **updatehistory** : public QWidget

{

Q\_OBJECT

public:

explicit **updatehistory**(QWidget \*parent = nullptr);

~***updatehistory***();

int postcombo;

int postcombo2;

private slots:

void **obr\_sendId**(int);

void **on\_pushButton\_clicked**();

signals:

void **refresh\_table**();

private:

Ui::updatehistory \*ui;

};

#endif // UPDATEHISTORY\_H

**Исходники:**

#include "addhistory.h"

#include "ui\_addhistory.h"

addHistory::**addHistory**(QWidget \*parent) :

QWidget(*parent*),

ui(new Ui::addHistory)

{

ui->setupUi(this);

QSqlQuery\*query=new QSqlQuery();

query->exec("SELECT id, name, accuracy\_class FROM MeasuringInstrument");

QSqlQuery\*query2=new QSqlQuery();

query2->exec("SELECT id, name, inn FROM OrganizationOfVerification");

while (query->next() && query2->next())

{

ui->comboBox->addItem(query->value(0).toString());

ui->comboBox\_2->addItem(query2->value(0).toString());

}

postcombo=0;

postcombo2=0;

this->setStyleSheet("background-color: DeepSkyBlue;");

}

addHistory::~***addHistory***()

{

delete ui;

}

void addHistory::**on\_pushButton\_clicked**()

{

QSqlQuery\* query = new QSqlQuery();

query->prepare("exec AddHistory ?,?,?");

query->bindValue(0, ui->comboBox->itemText(postcombo));

query->bindValue(1, ui->comboBox\_2->itemText(postcombo2));

query->bindValue(2, ui->dateEdit->date().toString("yyyy.MM.dd"));

QMessageBox\* mess = new QMessageBox();

if(!query->exec()){

mess->setText("Запрос составлен неверно или у вас нет доступа");

mess->show();

}

emit refresh\_table();

}

void addHistory::**on\_comboBox\_currentIndexChanged**(int index)

{

postcombo=index;

}

void addHistory::**on\_comboBox\_2\_currentIndexChanged**(int index)

{

postcombo2=index;

}

#include "addrecord.h"

#include "ui\_addrecord.h"

addrecord::**addrecord**(QWidget \*parent) :

QWidget(*parent*),

ui(new Ui::addrecord)

{

ui->setupUi(this);

this->setStyleSheet("background-color: DeepSkyBlue;");

}

addrecord::~***addrecord***()

{

delete ui;

}

void addrecord::**on\_pushButton\_clicked**()

{

QSqlQuery\* query = new QSqlQuery();

query->prepare("exec AddMeasuring :name,:accuracy\_class,:according\_to\_standard,:Verification\_interval");

query->bindValue(":name", ui->lineEdit->text());

query->bindValue(":accuracy\_class", ui->lineEdit\_2->text());

query->bindValue(":according\_to\_standard", ui->lineEdit\_3->text());

query->bindValue(":Verification\_interval", ui->lineEdit\_4->text());

QMessageBox\* mess = new QMessageBox();

if(!query->exec()){

mess->setText("Запрос составлен неверно или у вас нет доступа");

mess->show();

}

emit refresh\_table();

}

#include "addrecord2.h"

#include "ui\_addrecord2.h"

addrecord2::**addrecord2**(QWidget \*parent) :

QWidget(*parent*),

ui(new Ui::addrecord2)

{

ui->setupUi(this);

this->setStyleSheet("background-color: DeepSkyBlue;");

}

addrecord2::~***addrecord2***()

{

delete ui;

}

void addrecord2::**on\_pushButton\_clicked**()

{

QSqlQuery\*query=new QSqlQuery();

query->prepare("exec AddOrganization :name, :INN, :address");

query->bindValue(":INN", ui->lineEdit->text());

query->bindValue(":name", ui->lineEdit\_2->text());

query->bindValue(":address", ui->lineEdit\_3->text());

QMessageBox\*mess=new QMessageBox();

if(!query->exec())

{

mess->setText("запрос составлен неверно");

mess->show();

}

emit refresh\_table();

}

#include "addrecord3.h"

#include "ui\_addrecord3.h"

addrecord3::**addrecord3**(QWidget \*parent) :

QWidget(*parent*),

ui(new Ui::addrecord3)

{

ui->setupUi(this);

QSqlQuery\*query=new QSqlQuery();

query->exec("SELECT id, name, first\_name FROM ResponsibleEmployee");

QSqlQuery\*query2=new QSqlQuery();

query2->exec("SELECT id, name, accuracy\_class FROM MeasuringInstrument");

while (query->next() && query2->next())

{

ui->comboBox->addItem(query->value(0).toString());

ui->comboBox\_2->addItem(query2->value(0).toString());

}

postcombo=0;

postcombo2=0;

this->setStyleSheet("background-color: DeepSkyBlue;");

}

addrecord3::~***addrecord3***()

{

delete ui;

}

void addrecord3::**on\_pushButton\_clicked**()

{

QSqlQuery\* query = new QSqlQuery();

query->exec("exec AddEmployAndTool @employe\_id = " + ui->comboBox->itemText(postcombo) + ",@tool\_id = " + ui->comboBox\_2->itemText(postcombo2));

/\*QMessageBox\* mess = new QMessageBox();

if(!query->exec()){

mess->setText("Запрос составлен неверно или у вас нет доступа ");

mess->show();

}\*/

emit refresh\_table();

}

void addrecord3::**on\_comboBox\_currentIndexChanged**(int index)

{

postcombo=index;

}

void addrecord3::**on\_comboBox\_2\_currentIndexChanged**(int index)

{

postcombo2=index;

}

#include "form.h"

#include "ui\_form.h"

Form::**Form**(QWidget \*parent) :

QWidget(*parent*),

ui(new Ui::Form)

{

ui->setupUi(this);

this->setStyleSheet("background-color: DeepSkyBlue;");

}

Form::~***Form***()

{

delete ui;

}

void Form::**on\_pushButton\_clicked**()

{

QSqlQuery\*query=new QSqlQuery();

query->prepare("exec AddEmployee :first\_name, :name, :last\_name, :post");

query->bindValue(":first\_name", ui->lineEdit\_2->text());

query->bindValue(":name", ui->lineEdit->text());

query->bindValue(":last\_name", ui->lineEdit\_3->text());

query->bindValue(":post", ui->lineEdit\_4->text());

QMessageBox\*mess=new QMessageBox();

if(!query->exec())

{

mess->setText("запрос составлен неверно");

mess->show();

}

emit refresh\_table();

}

#include "izmenenie.h"

#include "ui\_izmenenie.h"

izmenenie::**izmenenie**(QWidget \*parent) :

QWidget(*parent*),

ui(new Ui::izmenenie)

{

ui->setupUi(this);

this->setStyleSheet("background-color: DeepSkyBlue;");

}

izmenenie::~***izmenenie***()

{

delete ui;

}

void izmenenie::**obr\_sendId**(int aa)

{

QSqlQuery\* query = new QSqlQuery();

query->prepare("SELECT name, accuracy\_class, according\_to\_standard, Verification\_interval FROM MeasuringInstrument WHERE id=?");

query->bindValue(0, aa);

if(query->exec())

{

query->next();

ui->lineEdit->setText(query->value(0).toString());

ui->lineEdit\_2->setText(query->value(1).toString());

ui->lineEdit\_3->setText(QString::number(aa));

ui->lineEdit\_4->setText(query->value(2).toString());

ui->lineEdit\_5->setText(query->value(3).toString());

}

}

void izmenenie::**on\_pushButton\_clicked**()

{

QSqlQuery\* query = new QSqlQuery();

query->prepare("exec UpdateMeasuring ?,?,?,?,?");

query->bindValue(0,ui->lineEdit\_3->text());

query->bindValue(1,ui->lineEdit->text());

query->bindValue(2,ui->lineEdit\_2->text());

query->bindValue(3,ui->lineEdit\_4->text());

query->bindValue(4,ui->lineEdit\_5->text());

if (query->exec()){

}

}

#include "izmenenie2.h"

#include "ui\_izmenenie2.h"

izmenenie2::**izmenenie2**(QWidget \*parent) :

QWidget(*parent*),

ui(new Ui::izmenenie2)

{

ui->setupUi(this);

this->setStyleSheet("background-color: DeepSkyBlue;");

}

izmenenie2::~***izmenenie2***()

{

delete ui;

}

void izmenenie2::**obr\_sendId**(int aa)

{

QSqlQuery\* query = new QSqlQuery();

query->prepare("SELECT first\_name, name, last\_name, post FROM ResponsibleEmployee WHERE id=?");

query->bindValue(0, aa);

if(query->exec())

{

query->next();

ui->lineEdit->setText(QString::number(aa));

ui->lineEdit\_2->setText(query->value(0).toString());

ui->lineEdit\_3->setText(query->value(1).toString());

ui->lineEdit\_4->setText(query->value(2).toString());

ui->lineEdit\_5->setText(query->value(3).toString());

}

}

void izmenenie2::**on\_pushButton\_clicked**()

{

QSqlQuery\* query = new QSqlQuery();

query->prepare("exec UpdateEmployee ?,?,?,?,?");

query->bindValue(0,ui->lineEdit->text());

query->bindValue(1,ui->lineEdit\_3->text());

query->bindValue(2,ui->lineEdit\_2->text());

query->bindValue(3,ui->lineEdit\_4->text());

query->bindValue(4,ui->lineEdit\_5->text());

if (query->exec()){

}

}

#include "izmenenie3.h"

#include "ui\_izmenenie3.h"

izmenenie3::**izmenenie3**(QWidget \*parent) :

QWidget(*parent*),

ui(new Ui::izmenenie3)

{

ui->setupUi(this);

this->setStyleSheet("background-color: DeepSkyBlue;");

}

izmenenie3::~***izmenenie3***()

{

delete ui;

}

void izmenenie3::**obr\_sendId**(int aa)

{

QSqlQuery\* query = new QSqlQuery();

query->prepare("SELECT name, INN, address FROM OrganizationOfVerification WHERE id=?");

query->bindValue(0, aa);

if(query->exec())

{

query->next();

ui->lineEdit->setText(QString::number(aa));

ui->lineEdit\_2->setText(query->value(0).toString());

ui->lineEdit\_3->setText(query->value(1).toString());

ui->lineEdit\_4->setText(query->value(2).toString());

}

}

void izmenenie3::**on\_pushButton\_clicked**()

{

QSqlQuery\* query = new QSqlQuery();

query->prepare("exec UpdateOrganization ?,?,?,?");

query->bindValue(0,ui->lineEdit->text());

query->bindValue(1,ui->lineEdit\_2->text());

query->bindValue(2,ui->lineEdit\_3->text());

query->bindValue(3,ui->lineEdit\_4->text());

}

#include "izmenenie4.h"

#include "ui\_izmenenie4.h"

izmenenie4::**izmenenie4**(QWidget \*parent) :

QWidget(*parent*),

ui(new Ui::izmenenie4)

{

ui->setupUi(this);

QSqlQuery\*query1=new QSqlQuery();

query1->exec("SELECT id, name, first\_name FROM ResponsibleEmployee");

QSqlQuery\*query2=new QSqlQuery();

query2->exec("SELECT id, name, accuracy\_class FROM MeasuringInstrument");

while (query1->next() && query2->next())

{

ui->comboBox->addItem(query1->value(0).toString());

ui->comboBox\_2->addItem(query2->value(0).toString());

}

postcombo=0;

postcombo2=0;

this->setStyleSheet("background-color: DeepSkyBlue;");

}

izmenenie4::~***izmenenie4***()

{

delete ui;

}

void izmenenie4::**obr\_sendId**(int aa)

{

QSqlQuery\* query = new QSqlQuery();

query->prepare("SELECT employe\_id, tool\_id FROM EmployeeAndToolRelationship WHERE id=?");

query->bindValue(0, aa);

if(query->exec())

{

query->next();

ui->lineEdit->setText(QString::number(aa));

ui->comboBox->itemText(query->value(0).toInt());

ui->comboBox\_2->itemText(query->value(1).toInt());

}

}

void izmenenie4::**on\_pushButton\_clicked**()

{

QSqlQuery\* query = new QSqlQuery();

query->exec("exec UpdateEmployAndTool @id="+ui->lineEdit->text()+", @employe\_id = " + ui->comboBox->itemText(postcombo) + ",@tool\_id = " + ui->comboBox\_2->itemText(postcombo2));

/\*QMessageBox\* mess = new QMessageBox();

if(!query->exec()){

mess->setText("Запрос составлен неверно или у вас нет доступа ");

mess->show();

}\*/

}

void izmenenie4::**on\_comboBox\_currentIndexChanged**(int index)

{

postcombo=index;

}

void izmenenie4::**on\_comboBox\_2\_currentIndexChanged**(int index)

{

postcombo2=index;

}

#include "login.h"

#include "ui\_login.h"

Login::**Login**(QWidget \*parent) :

QWidget(*parent*),

ui(new Ui::Login)

{

ui->setupUi(this);

this->setStyleSheet("background-color: DeepSkyBlue;");

ui->lineEdit->setText("DESKTOP-HCBND9M\SQLEXPRESS");

ui->lineEdit\_2->setText("MetrologicalService");

ui->lineEdit\_4->setEchoMode(QLineEdit::Password);

mes = new QMessageBox();

}

Login::~***Login***()

{

delete ui;

}

void Login::**on\_pushButton\_clicked**()

{

db = QSqlDatabase::addDatabase("QODBC");

db.setDatabaseName("DRIVER={SQL Server};SERVER="+ui->lineEdit->text()+";DATABASE="+ui->lineEdit\_2->text()+";");

db.setUserName(ui->lineEdit\_3->text());

db.setPassword(ui->lineEdit\_4->text());

if (db.open()){

mes->setText("Соединилось");

} else{

mes->setText("Соединение НЕ установлено");

}

mes->show();

}

#include "mainwindow.h"

#include <QApplication>

int main(int argc, char \*argv[])

{

QApplication a(*argc*, *argv*);

MainWindow w;

w.show();

return a.exec();

}

#include "mainwindow.h"

#include "ui\_mainwindow.h"

MainWindow::**MainWindow**(QWidget \*parent)

: QMainWindow(*parent*)

, ui(new Ui::MainWindow)

{

ui->setupUi(this);

this->setStyleSheet("background-color: DeepSkyBlue;");

ui->tableView->setContextMenuPolicy(Qt::CustomContextMenu);

connect(ui->tableView, SIGNAL(customContextMenuRequested(QPoint)), SLOT(CustomMenuReq(QPoint)));

fl = 0;

}

MainWindow::~***MainWindow***()

{

delete ui;

}

void MainWindow::**on\_action\_triggered**()

{

logwin = new Login();

logwin->show();

}

void MainWindow::**on\_pushButton\_clicked**()

{

fl=1;

model=new QSqlQueryModel();

model->setQuery("SELECT id, name, accuracy\_class, according\_to\_standard, Verification\_interval FROM MeasuringInstrument");

model->*setHeaderData*(0,Qt::Horizontal,"ID");

model->*setHeaderData*(1,Qt::Horizontal,"Наименование");

model->*setHeaderData*(2,Qt::Horizontal,"Класс точности");

model->*setHeaderData*(3,Qt::Horizontal,"Соответствие со стандартом");

model->*setHeaderData*(4,Qt::Horizontal,"Меж поверочный интервал");

ui->tableView->*setModel*(*model*);

ui->tableView->setColumnHidden(0, true);

ui->tableView->resizeColumnsToContents();

ui->tableView->show();

}

void MainWindow::**on\_pushButton\_2\_clicked**()

{

dobzap = new addrecord();

connect(dobzap, SIGNAL(refresh\_table()), this, SLOT(obt\_ref\_tab()));

dobzap->show();

}

void MainWindow::**obt\_ref\_tab**(){

on\_pushButton\_clicked();

}

void MainWindow::**CustomMenuReq**(QPoint pos)

{

if(fl==1) {

QModelIndex index = ui->tableView->*indexAt*(pos);

globid = ui->tableView->model()->*data*(ui->tableView->model()->*index*(index.row(), 0)).toInt();

recId = ui->tableView->model()->*data*(ui->tableView->model()->*index*(index.row(), 2)).toInt();

QMenu\* menu = new QMenu(this);

QAction\* izm = new QAction("Изменить инструмент", this);

connect(izm, SIGNAL(triggered()), this, SLOT(izm\_zap()));

QAction\* izm2 = new QAction("Изменить сотрудника", this);

connect(izm2, SIGNAL(triggered()), this, SLOT(izm\_zap2()));

QAction\* izm3 = new QAction("Изменить организацию", this);

connect(izm3, SIGNAL(triggered()), this, SLOT(izm\_zap3()));

QAction\* izm4 = new QAction("Изменить связь инструмента и сотрудника", this);

connect(izm4, SIGNAL(triggered()), this, SLOT(izm\_zap4()));

QAction\* izm5 = new QAction("Изменить запись истории поверки", this);

connect(izm5, SIGNAL(triggered()), this, SLOT(izm\_zap5()));

QAction\* ud = new QAction("Удалить инструмент", this);

connect(ud, SIGNAL(triggered()), this, SLOT(del\_zap()));

QAction\* ud2 = new QAction("Удалить сотрудника", this);

connect(ud2, SIGNAL(triggered()), this, SLOT(del\_zap2()));

QAction\* ud3 = new QAction("Удалить организацию", this);

connect(ud3, SIGNAL(triggered()), this, SLOT(del\_zap3()));

QAction\* ud4 = new QAction("Удалить связь инструмента и сотрудника", this);

connect(ud4, SIGNAL(triggered()), this, SLOT(del\_zap4()));

QAction\* ud5 = new QAction("Удалить историю поверки", this);

connect(ud5, SIGNAL(triggered()), this, SLOT(del\_zap5()));

menu->addAction(*izm*);

menu->addAction(*izm2*);

menu->addAction(*izm3*);

menu->addAction(*izm4*);

menu->addAction(*izm5*);

menu->addAction(*ud*);

menu->addAction(*ud2*);

menu->addAction(*ud3*);

menu->addAction(*ud4*);

menu->addAction(*ud5*);

menu->popup(ui->tableView->viewport()->mapToGlobal(pos));

}

}

void MainWindow::**del\_zap**(){

QSqlQuery\* query = new QSqlQuery();

query->prepare("exec DeleteMeasuring ?");

query->bindValue(0, globid);

if (query->exec()){

on\_pushButton\_clicked();

}

}

void MainWindow::**del\_zap2**(){

QSqlQuery\* query = new QSqlQuery();

query->prepare("exec DeleteEmployee ?");

query->bindValue(0, globid);

if (query->exec()){

on\_pushButton\_9\_clicked();

}

}

void MainWindow::**del\_zap3**(){

QSqlQuery\* query = new QSqlQuery();

query->prepare("exec DeleteOrganization ?");

query->bindValue(0, globid);

if (query->exec()){

on\_pushButton\_10\_clicked();

}

}

void MainWindow::**del\_zap4**(){

QSqlQuery\* query = new QSqlQuery();

query->prepare("exec DeleteEmployAndTool ?");

query->bindValue(0, globid);

if (query->exec()){

on\_pushButton\_11\_clicked();

}

}

void MainWindow::**del\_zap5**(){

QSqlQuery\* query = new QSqlQuery();

query->prepare("exec DeleteHistory ?");

query->bindValue(0, globid);

if (query->exec()){

on\_pushButton\_14\_clicked();

}

}

void MainWindow::**izm\_zap**()

{

izm = new izmenenie();

connect(this, SIGNAL(sendId(int)), izm, SLOT(obr\_sendId(int)));

emit sendId(globid);

izm->show();

disconnect(this, SIGNAL(sendId(int)), izm, SLOT(obr\_sendId(int)));

}

void MainWindow::**izm\_zap2**()

{

izm2 = new izmenenie2();

connect(this, SIGNAL(sendId(int)), izm2, SLOT(obr\_sendId(int)));

emit sendId(globid);

izm2->show();

disconnect(this, SIGNAL(sendId(int)), izm2, SLOT(obr\_sendId(int)));

}

void MainWindow::**izm\_zap3**()

{

izm3 = new izmenenie3();

connect(this, SIGNAL(sendId(int)), izm3, SLOT(obr\_sendId(int)));

emit sendId(globid);

izm3->show();

disconnect(this, SIGNAL(sendId(int)), izm3, SLOT(obr\_sendId(int)));

}

void MainWindow::**izm\_zap4**()

{

izm4 = new izmenenie4();

connect(this, SIGNAL(sendId(int)), izm4, SLOT(obr\_sendId(int)));

emit sendId(globid);

izm4->show();

disconnect(this, SIGNAL(sendId(int)), izm4, SLOT(obr\_sendId(int)));

}

void MainWindow::**izm\_zap5**()

{

izm5 = new updatehistory();

connect(this, SIGNAL(sendId(int)), izm5, SLOT(obr\_sendId(int)));

emit sendId(globid);

izm5->show();

disconnect(this, SIGNAL(sendId(int)), izm5, SLOT(obr\_sendId(int)));

}

void MainWindow::**on\_pushButton\_5\_clicked**()

{

fl=1;

model = new QSqlQueryModel();

model->setQuery("SELECT id, name,CASE WHEN Verification\_interval <= 6 THEN 'меньше 6' ELSE 'больше 6' END AS interval FROM MeasuringInstrument;");

model->*setHeaderData*(0, Qt::Horizontal,"ID");

model->*setHeaderData*(1, Qt::Horizontal,"Наименование");

model->*setHeaderData*(2, Qt::Horizontal,"Интервал поверки");

ui->tableView->*setModel*(*model*);

ui->tableView->setColumnHidden(0, true);

ui->tableView->resizeColumnsToContents();

ui->tableView->show();

}

void MainWindow::**on\_pushButton\_6\_clicked**()

{

}

void MainWindow::**on\_pushButton\_7\_clicked**()

{

sub = new subqueries();

connect(sub, SIGNAL(refresh\_table()), this, SLOT(obt\_ref\_tab()));

sub->show();

}

void MainWindow::**on\_pushButton\_8\_clicked**()

{

fl=1;

model = new QSqlQueryModel();

model->setQuery("SELECT name,measuring, data FROM verificationDate");

model->*setHeaderData*(0, Qt::Horizontal,"Название организации");

model->*setHeaderData*(1, Qt::Horizontal,"Название инструмента");

model->*setHeaderData*(2, Qt::Horizontal,"Дата поверки");

ui->tableView->*setModel*(*model*);

ui->tableView->setColumnHidden(0, false);

ui->tableView->resizeColumnsToContents();

ui->tableView->show();

}

void MainWindow::**on\_pushButton\_9\_clicked**()

{

fl=1;

model = new QSqlQueryModel();

model->setQuery("SELECT id,first\_name, name, last\_name,post FROM ResponsibleEmployee");

model->*setHeaderData*(0, Qt::Horizontal,"ID");

model->*setHeaderData*(1, Qt::Horizontal,"Фамилия");

model->*setHeaderData*(2, Qt::Horizontal,"Имя");

model->*setHeaderData*(3, Qt::Horizontal,"Отчество");

model->*setHeaderData*(4, Qt::Horizontal,"Должность");

ui->tableView->*setModel*(*model*);

ui->tableView->setColumnHidden(0, true);

ui->tableView->resizeColumnsToContents();

ui->tableView->show();

}

void MainWindow::**on\_pushButton\_10\_clicked**()

{

fl=1;

model = new QSqlQueryModel();

model->setQuery("SELECT id, name, INN, address FROM OrganizationOfVerification");

model->*setHeaderData*(0, Qt::Horizontal,"ID");

model->*setHeaderData*(1, Qt::Horizontal,"Наименование");

model->*setHeaderData*(2, Qt::Horizontal,"ИНН");

model->*setHeaderData*(3, Qt::Horizontal,"Адрес");

ui->tableView->*setModel*(*model*);

ui->tableView->setColumnHidden(0, true);

ui->tableView->resizeColumnsToContents();

ui->tableView->show();

}

void MainWindow::**on\_pushButton\_11\_clicked**()

{

fl=1;

model = new QSqlQueryModel();

model->setQuery("SELECT t1.id, t2.name, t2.first\_name, t3.name FROM EmployeeAndToolRelationship T1 JOIN ResponsibleEmployee T2 ON T1.tool\_id = T2.id JOIN MeasuringInstrument T3 ON T1.tool\_id = T3.id");

model->*setHeaderData*(0, Qt::Horizontal,"ID");

model->*setHeaderData*(1, Qt::Horizontal,"Имя сотрудника");

model->*setHeaderData*(2, Qt::Horizontal,"Фамилия сотрудника");

model->*setHeaderData*(3, Qt::Horizontal,"Название инструмента");

ui->tableView->*setModel*(*model*);

ui->tableView->setColumnHidden(0, true);

ui->tableView->resizeColumnsToContents();

ui->tableView->show();

}

void MainWindow::**on\_pushButton\_12\_clicked**()

{

dobzap2 = new Form();

connect(dobzap2, SIGNAL(refresh\_table()), this, SLOT(obt\_ref\_tab2()));

dobzap2->show();

}

void MainWindow::**obt\_ref\_tab2**(){

on\_pushButton\_9\_clicked();

}

void MainWindow::**on\_pushButton\_3\_clicked**()

{

dobzap3 = new addrecord2();

connect(dobzap3, SIGNAL(refresh\_table()), this, SLOT(obt\_ref\_tab3()));

dobzap3->show();

}

void MainWindow::**obt\_ref\_tab3**(){

on\_pushButton\_10\_clicked();

}

void MainWindow::**on\_pushButton\_4\_clicked**()

{

dobzap4 = new addrecord3();

connect(dobzap4, SIGNAL(refresh\_table()), this, SLOT(obt\_ref\_tab4()));

dobzap4->show();

}

void MainWindow::**obt\_ref\_tab4**(){

on\_pushButton\_11\_clicked();

}

void MainWindow::**on\_pushButton\_14\_clicked**()

{

fl=1;

QSqlQueryModel\* model1;

model1 = new QSqlQueryModel();

model1->setQuery("SELECT t1.id, t2.name AS measuring\_name, t3.name AS organization\_name, T1.verification\_date FROM HistoryOfVerifications T1 JOIN MeasuringInstrument T2 ON T1.tool\_id = T2.id JOIN OrganizationOfVerification T3 ON T1.organization\_id = T3.id");

model1->*setHeaderData*(1, Qt::Horizontal,"Название инструмента");

model1->*setHeaderData*(2, Qt::Horizontal,"Название организации");

model1->*setHeaderData*(3, Qt::Horizontal,"Дата поверки");

ui->tableView->*setModel*(*model1*);

ui->tableView->setColumnHidden(0, true);

ui->tableView->resizeColumnsToContents();

ui->tableView->show();

}

void MainWindow::**on\_pushButton\_15\_clicked**()

{

dobzap5 = new addHistory();

connect(dobzap5, SIGNAL(refresh\_table()), this, SLOT(obt\_ref\_tab5()));

dobzap5->show();

}

void MainWindow::**obt\_ref\_tab5**(){

on\_pushButton\_14\_clicked();

}

#include "subqueries.h"

#include "ui\_subqueries.h"

subqueries::**subqueries**(QWidget \*parent) :

QWidget(*parent*),

ui(new Ui::subqueries)

{

ui->setupUi(this);

this->setStyleSheet("background-color: DeepSkyBlue;");

}

subqueries::~***subqueries***()

{

delete ui;

}

void subqueries::**on\_pushButton\_clicked**()

{

model = new QSqlQueryModel();

model->setQuery("SELECT name, INN FROM OrganizationOfVerification WHERE address = ( SELECT address FROM OrganizationOfVerification WHERE INN=123)");

model->*setHeaderData*(0, Qt::Horizontal,"Номер организации");

model->*setHeaderData*(1, Qt::Horizontal,"INN");

ui->tableView->*setModel*(*model*);

ui->tableView->resizeColumnsToContents();

ui->tableView->show();

}

void subqueries::**on\_pushButton\_2\_clicked**()

{

model = new QSqlQueryModel();

model->setQuery("SELECT T1.name, (SELECT T2.name FROM measuringInstrument T2 WHERE T3.organization\_id=T1.id AND T3.tool\_id=T2.id) AS measuring FROM OrganizationOfVerification T1, HistoryOfVerifications T3");

model->*setHeaderData*(0, Qt::Horizontal,"Организация");

model->*setHeaderData*(1, Qt::Horizontal,"Инструмент");

ui->tableView->*setModel*(*model*);

ui->tableView->resizeColumnsToContents();

ui->tableView->show();

}

void subqueries::**on\_pushButton\_3\_clicked**()

{

model = new QSqlQueryModel();

model->setQuery("SELECT name FROM measuringInstrument GROUP BY name, accuracy\_class HAVING accuracy\_class < 3");

model->*setHeaderData*(0, Qt::Horizontal,"Название инструмента");

ui->tableView->*setModel*(*model*);

ui->tableView->resizeColumnsToContents();

ui->tableView->show();

}

void subqueries::**on\_pushButton\_4\_clicked**()

{

model = new QSqlQueryModel();

model->setQuery("SELECT name, accuracy\_class, according\_to\_standard, Verification\_interval FROM measuringInstrument WHERE Verification\_interval < ALL (SELECT AVG(Verification\_interval) FROM measuringInstrument)");

model->*setHeaderData*(0, Qt::Horizontal,"Название инструмента");

model->*setHeaderData*(1, Qt::Horizontal,"Класс точности");

model->*setHeaderData*(2, Qt::Horizontal,"Соответствие госту");

model->*setHeaderData*(3, Qt::Horizontal,"Интервал поверки");

ui->tableView->*setModel*(*model*);

ui->tableView->resizeColumnsToContents();

ui->tableView->show();

}

void subqueries::**on\_pushButton\_5\_clicked**()

{

model = new QSqlQueryModel();

model->setQuery("SELECT name, measuring, CASE WHEN data < GETDATE()THEN 'пора на проверку' ELSE 'все в порядке' END AS chek FROM verificationDate;");

model->*setHeaderData*(0, Qt::Horizontal,"Организация");

model->*setHeaderData*(1, Qt::Horizontal,"Инструмент");

model->*setHeaderData*(2, Qt::Horizontal,"просрочено или нет");

ui->tableView->*setModel*(*model*);

ui->tableView->resizeColumnsToContents();

ui->tableView->show();

}

void subqueries::**on\_pushButton\_6\_clicked**()

{

model = new QSqlQueryModel();

model->setQuery("SELECT post, COUNT(name) AS count FROM ResponsibleEmployee GROUP BY post");

model->*setHeaderData*(0, Qt::Horizontal,"Должность");

model->*setHeaderData*(1, Qt::Horizontal,"Колличество сотрудников");

ui->tableView->*setModel*(*model*);

ui->tableView->resizeColumnsToContents();

ui->tableView->show();

}

#include "updatehistory.h"

#include "ui\_updatehistory.h"

updatehistory::**updatehistory**(QWidget \*parent) :

QWidget(*parent*),

ui(new Ui::updatehistory)

{

ui->setupUi(this);

QSqlQuery\*query=new QSqlQuery();

query->exec("SELECT id, name, accuracy\_class FROM MeasuringInstrument");

QSqlQuery\*query2=new QSqlQuery();

query2->exec("SELECT id, name FROM OrganizationOfVerification");

while (query->next() && query2->next())

{

ui->comboBox->addItem(query->value(0).toString());

ui->comboBox\_2->addItem(query2->value(0).toString());

}

postcombo=0;

postcombo2=0;

this->setStyleSheet("background-color: DeepSkyBlue;");

}

updatehistory::~***updatehistory***()

{

delete ui;

}

void updatehistory::**obr\_sendId**(int aa)

{

QSqlQuery\* query = new QSqlQuery();

query->prepare("SELECT tool\_id, organization\_id, verification\_date FROM HistoryOfVerifications WHERE id=?");

query->bindValue(0, aa);

if(query->exec())

{

query->next();

ui->lineEdit->setText(QString::number(aa));

ui->comboBox->itemText(query->value(0).toInt());

ui->comboBox\_2->itemText(query->value(1).toInt());

ui->dateEdit->setDate(query->value(2).toDate());

}

}

void updatehistory::**on\_pushButton\_clicked**()

{

QSqlQuery\* query = new QSqlQuery();

query->prepare("exec UpdateHistory ?,?,?,?");

query->bindValue(0, ui->lineEdit->text());

query->bindValue(1, ui->comboBox->itemText(postcombo));

query->bindValue(2, ui->comboBox->itemText(postcombo2));

query->bindValue(3, ui->dateEdit->date().toString("yyyy.MM.dd"));

QMessageBox\* mess = new QMessageBox();

if(!query->exec()){

mess->setText("Запрос составлен неверно или у вас нет доступа");

mess->show();

}

emit refresh\_table();

}