**ВВЕДЕНИЕ**

В настоящее время работа с документами отнимает у пользователей, особенно у офисных сотрудников, очень много времени. На сегодняшний день существует немало программ, позволяющих существенно оптимизировать ручной труд при работе с документами путем автоматизации некоторых процессов. Многие из таких решений просты в применении и очень удобны. Ввиду личной заинтересованности для данного курсового проекта в качестве программы для разработки была выбрана программа управления таможенным оформлением. Помимо практического интереса, тема имеет широкие возможности для последующей модернизации проекта с применением навыков, полученных в ходе изучения курса КПиЯП.

Основными концепциями объектно-ориентированного программирования являются инкапсуляция, полиморфизм и наследование. Язык C++ предоставляет исчерпывающие возможности для реализаций этих концепций. В результате использования инкапсуляции, программа, написанные на C++, обладает повышенной защищённостью объектов от влияния на них кода других частей этой же программы. Наследование предоставляет важную возможность повторного использования кода, что может значительно уменьшить количество кода, однако требует предварительного проектирования архитектуры. Полиморфизм позволяет программе быть более гибкой. Такая система удобна в тестировании и позволяет легко заменять и модифицировать свои компоненты.

В ходе работы используется фреймворк Qt.

Qt предоставляет программисту не только удобный набор библиотек классов, но и определённую модель разработки приложений, определённый каркас их структуры. Следование принципам и правилам «хорошего стиля программирования на C++/Qt» существенно снижает частоту таких трудно отлавливаемых ошибок в приложениях, как утечки памяти (memory leaks), необработанные исключения, незакрытые файлы или неосвобождённые дескрипторы ресурсных объектов, чем нередко страдают программы, написанные «на голом C++» без использования библиотеки Qt.

Важным преимуществом Qt является хорошо продуманный, логичный и стройный набор классов, предоставляющий программисту очень высокий уровень абстракции. Благодаря этому программистам, использующим Qt, приходится писать значительно меньше кода, чем это имеет место при использовании, например, библиотеки классов MFC. Сам же код выглядит стройнее и проще, логичнее и понятнее, чем аналогичный по функциональности код MFC или код, написанный с использованием «родного» для X11 тулкита Xt. Его легче поддерживать и развивать.

Таможенное оформление — это процедура помещения товаров и транспортных средств под определенный таможенный режим и завершения действия этого режима в соответствии с требованиями и положениями Таможенного кодекса. Оно базируется на выполнении участниками внешнеэкономической деятельности требований таможенного законодательства об обязательном декларировании всех товаров и транспортных средств, перемещаемых через таможенную границу.

Декларирование — это операция, являющаяся составной частью процедуры помещения товаров и транспортных средств под определенный таможенный режим или по завершении действия такого режима. Ее суть заключается в заполнении по установленной форме сведений о товарах и транспортных средствах, перемещаемых через таможенную границу государства, товарах и транспортных средствах, уже перемещенных через таможенную границу, таможенный режим которых изменяется, а также о других товарах и транспортных средствах, которые подлежат декларированию.

1. **ПОСТАНОВКА ЗАДАЧИ**

Главной задачей данного курсового проекта является разработка удобного приложения для работы с формами документов. В рамках данного курсового проекта планируется разработать собственную базу данных и связать её с графическим интерфейсом при помощи фреймворка Qt.

Программа должна иметь удобный и простой интерфейс и давать возможность работать с несколькими документами. В ней должен быть обеспечен валидный и удобный ввод данных. Помимо этого, должны присутствовать элементарные действия над множеством документ (удаление, добавление).

1. **СТРУКТУРНОЕ ПРОЕКТИРОВАНИЕ**
   1. **Общее описание состава программного обеспечения**

Общее структуру данного программного обеспечения можно описать двумя пунктами:

1. Реагирование на действия пользователя с помощью сигналов;
2. Обработка сигнала в определенном слоте (слотах).

В данном программном обеспечении можно выделить несколько блоков, отвечающих за успешное выполнение программы:

1. Блок взаимодействия пользователя с приложением;
2. Блок синхронизации с БД;
3. Блок записи в БД;
4. Блок чтения с БД.

Структурная схема приведена в приложении А.

* 1. **Краткое описание сторонних программных компонентов**

Для разработки данного программного средства будет использоваться язык программирования C++, кроссплатформенный фреймворк Qt для разработки программного обеспечения на языке программирования C++, СУБД SQLite, менеджер работы с БД SQLite Studio и среда разработки Qt Creator.

1. **ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ**

При запуске программы мы видим окно со списком всех деклараций, хранящихся в базе данных. Общее функционирование данного ПО строится на следующем: пользователь может выбрать декларацию из списка или создать новую и затем приступить к редактированию формы.

В главном окне пользователю доступны следующие функции:

1. Список деклараций

Данная функция отображает окно со списком всех деклараций,

имеющихся в базе данных в данный момент. Список представлен в виде таблицы, реализованной в классе QTableWidget. При двойном щелчке ЛКМ на определенную строку посылается сигнал tableItemDoubleClicked который несёт с собой информацию о номере нажатой строки, далее он обрабатывается в определенном слоте класса MainWindow, где выполняется поиск записи по первичному ключу в БД и происходит отображение окна Frame с необходимой формой документа. В окне с таблицей мы можем выполнить сортировку списка по любому столбцу.

1. Обновить

Предназначение данной функции состоит в обновлении списка всех

документов в соответствии с имеющимися в данный момент данным в БД. При нажатии на данную кнопку вызывается сигнал и поступает в слот on\_action\_update\_triggered(). В данном сигнале инициализируется объект класса запроса QSqlQuery. С помощью запроса SELECT получаем все записи в БД. Далее посылаем сигнал toRemoveTable(), который косвенно поступает в слот объекта класса DeclList где происходит удаление списка деклараций. Далее с помощью методa query->next() проходимся по всем записям посылая сигнал toUpdateTable(query->record()) косвенно в объект класса DeclList, где добавляется новая строка и происходит запись из объекта класса QSqlRecord.

1. Добавить документ

С помощью этой опции пользователь может создать новый документ, при нажатии на кнопку, выполняющую данную опцию, открывается диалоговое окно DeclTypeDialog где можно выбрать один из двух типов деклараций. После выбора типа декларации открывается окно с формой Frame для заполнения только что созданного документа.

1. Удалить декларацию

С помощью этой опции пользователь может удалить любой документ из списка и базы данных. При нажатии на данную вызывается сигнал нажатия clicked() и попадает в слот on\_action\_delete\_triggered(), в котором высылается сигнал toRemoveSelectedRow(), который попадает в слот объекта класса DeclList, где мы определяем индекс выделенной строки с помощью класса QItemSelectionModel.

При открытии формы документа мы можем приступить к заполнению полей. Правильное заполнение полей гарантируется удобным заполнением с помощью вспомогательного диалогового окна для некоторых полей, которое можно открыть с помощью кнопки рядом с полем при её наличии, либо в некоторых остальных местах при нажатии клавиши ENTER нажав на поле, где необходимо вводить данные, левой кнопкой мыши.

1. **РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ**

При разработке приложения использовался механизм сигналов и слотов, где в последних обрабатывались введённые данные и далее через некоторую последовательность сигналов и слотов в классе MainWindow данные поступали в базу данных. Рассмотрим основные методы работы с базами данных.

createConnection() – метод, служащий для соединения с БД. Данный метод реализован следующим образом:

dbDecl = QSqlDatabase::addDatabase("QSQLITE", "DeclarationConnection");

dbDecl.setDatabaseName("C:\\Users\\User\\Documents\\untitled3\\decl.db3");

dbDecl.setPort(5432);

dbDecl.setHostName("localhost");

dbDecl.setUserName("postgres");

dbDecl.setPassword("root");

if(!dbDecl.open())

{

qDebug() << "Cannot open database:" << dbDecl.lastError();

return false;

}

else

{

return true;

}

slotGoodsButtonAddClicked() – слот, который принимает сигнал от кнопки добавления в окне списка товаров. Он имеет следующую реализацию:

model = new QSqlTableModel(nullptr, dbDecl);

model->*setTable*("decl");

model->*select*();

QSqlRecord rec = model->record(ui->stackedWidget->currentIndex() - 1);

int id = rec.value("id").toInt();

query = new QSqlQuery(dbDecl);

QString strF = "INSERT INTO goods (uid) "

"VALUES(%1);";

QString str = strF.arg(id);

query->exec(str);

slotToDeleteEntryFromDb(int id) – слот который принимает сигнал от кнопки удаления с передачей ключа для удаления записи с БД:

query = new QSqlQuery(dbDecl);

QString argStr = "DELETE FROM decl "

"WHERE id = %1;";

QString str = argStr.arg(id);

query->exec(str);

1. **РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ**
2. Открыть исполняемый файл

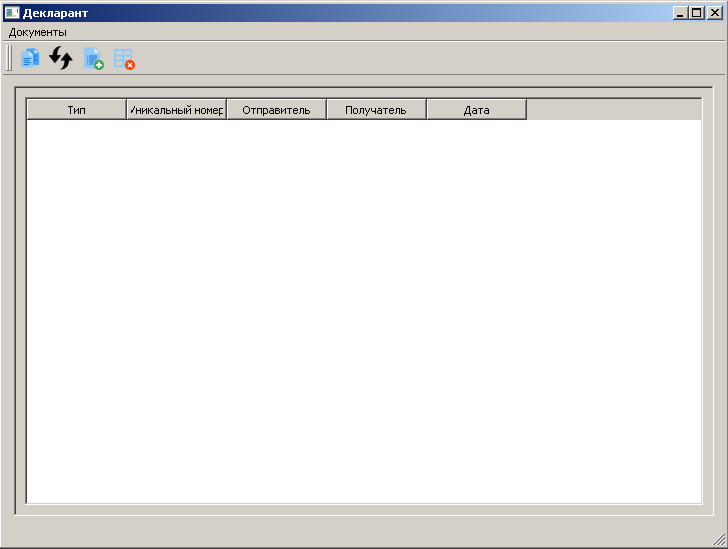
****

Рисунок 4.1

1. В открывшемся окне (рис. 4.1.) предоставляется возможность выбрать

необходимый документ в списке. Также на панели инструментов имеются основные элементы для работы с документами.

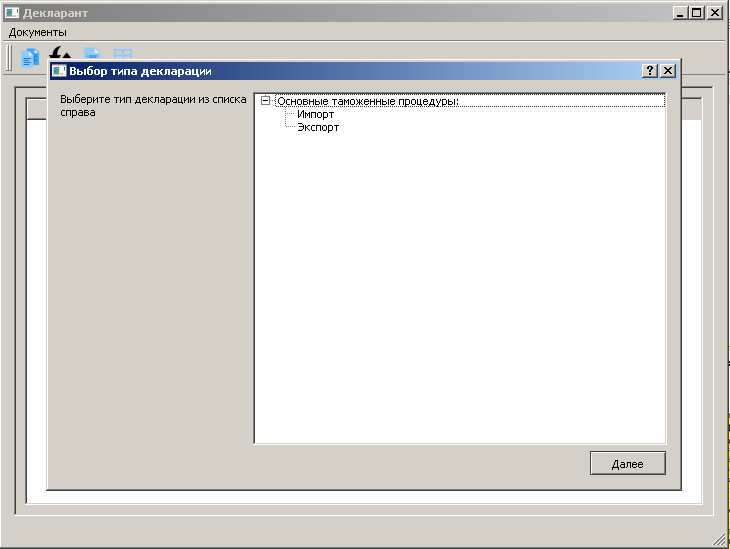
****

Рисунок 4.2

1. При выборе элемента на панели инструментов “Добавить документ”

мы переходим в окно выбора типа декларации (рис. 4.2).

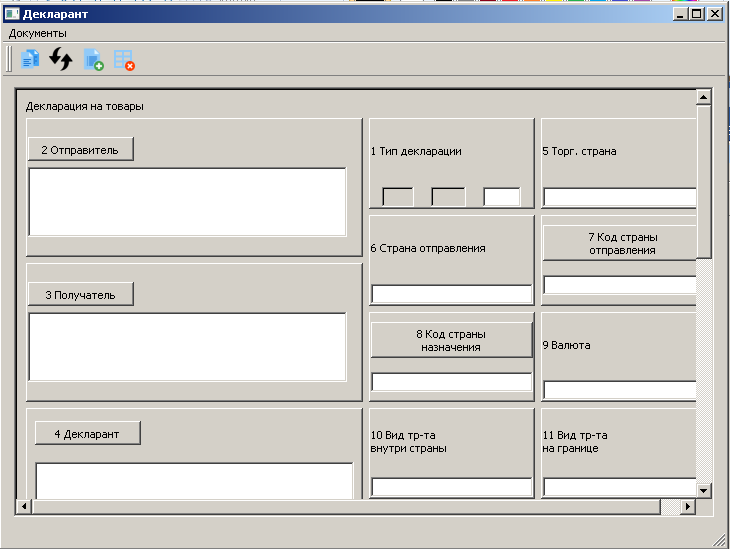
****

Рисунок 4.3

1. При выборе типа декларации мы попадаем в окно заполнения формы

декларации (рис. 4.3).

1. **ТЕСТИРОВАНИЕ**

Результаты тестирования программы представлены ниже

Таблица *6.1* – Тестирование программного средства

|  |  |  |
| --- | --- | --- |
| Тестируемая функция | Ожидаемый результат | Полученный результат |
| 1 | 2 | 3 |
| 1. Запуск программы  2. Нажатие пункта меню “Выход” | 1. Отображение необходимых компонентов  2. Закрытие программы. | 1. Отображение необходимых компонентов  2. Закрытие программы. |
| Добавить декларацию | 1. Создание новой записи в БД  2. Появление нового окна с новой формой для заполнения декларации | 1. Создание новой записи в БД  2. Появление нового окна с новой формой для заполнения декларации |
| Удалить декларацию | 1. Удалений записи с определенным первичным ключом в БД  2. Удаление записи из списка деклараций | 1. Удалений записи с определенным первичным ключом в БД  2. Удаление записи из списка деклараций |
| Обновить | 1. Отображение измененной информации в списке документов | 1. Отображение измененной информации в списке документов |

**ЗАКЛЮЧЕНИЕ**

В результате выполнения данного курсового проекта была произведена работа с базами данных, включая её создание и редактирование. Также к работе были подключены сторонние базы данных, так называемые справочники, где хранилась уже готовая информация. Далее, следуя механизму сигналов и слотов в Qt, был обеспечен правильный и удобный ввод данных пользователем и.

В ходе выполнения данного курсового проекта было изучено и успешно использовано большое количество текстовой и графической информации, в результате чего стало возможным проектирование программного продукта, его тестирование и устранение ошибок.

Системные требования:

-Операционная система Windows 98, XP, Vista,7, 8.1, 10;

-Процессор Intel Pentium III 1 GHz;

-Оперативная память 520 мб;

-Свободное место на жестком диске: 100 Мб;

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Герберт, Ш. Самоучитель C++/Ш. Герберт. Санкт-Петербург 2003г;

[2] Евгений, К. Программирование шахмат и других логических игр/К. Евгений. Санкт-Петербург 2005г;

[3] Страуструп, Б. Язык программирования С++/ Б.Страуструп; специальное издание. Пер. с англ. – Спб.: BHV, 2008. – 1098 с;

[4] MSDN – информационный сервис для разработчиков [Электронный ресурс]. – Режим доступа: https://msdn.microsoft.com/ru-ru/;

[5] Sfml-dev.org – информационный сервис для изучения библиотеки SFML [Электронный ресурс]. – Режим доступа: https://www.sfml-dev.org/index.php/.

**ПРИЛОЖЕНИЕ А**

(обязательное)

Структурная схема программы «Декларант»

**ПРИЛОЖЕНИЕ Б**

(обязательное)

UML диаграмма классов

**ПРИЛОЖЕНИЕ В**

(обязательное)

Исходный код программы

mainwindow.cpp

#include <QtWidgets>

#include <QtSql>

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include "decltypewindow.h"

#include "frame.h"

#include "decllist.h"

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

QMainWindow(parent),

ui(new Ui::MainWindow),

declMenu(nullptr),

nRecords(0)

{

ui->setupUi(this);

//MAIN WINDOW CONNECTIONS

connect(this,

SIGNAL(toUpdateTextEdit\_1()),

SLOT(slotToUpdateTextEdit\_1()));

connect(this,

SIGNAL(toUpdateTextEdit\_2()),

SLOT(slotToUpdateTextEdit\_2()));

connect(this,

SIGNAL(toUpdateTextEdit\_3()),

SLOT(slotToUpdateTextEdit\_3()));

connect(this,

SIGNAL(toUpdateList()),

SLOT(on\_action\_update\_triggered()));

// DECLARATION LIST ELEMENT ON TOOL BAR

QPixmap declListMap("C:\\Users\\User\\Documents\\untitled3\\images\\declListMap.png");

QIcon declListIcon(declListMap);

ui->mainToolBar->addAction(declListIcon,

"Список деклараций",

this,

SLOT(on\_action\_declList\_triggered()));

// UPDATE ELEMENT ON TOOL BAR

QPixmap updateMap("C:\\Users\\User\\Documents\\untitled3\\images\\updateMap.png");

QIcon updateIcon(updateMap);

ui->mainToolBar->addAction(updateIcon,

"Обновить",

this,

SLOT(on\_action\_update\_triggered()));

// ADD ELEMENT ON TOOL BAR

QPixmap addMap("C:\\Users\\User\\Documents\\untitled3\\images\\addMap.png");

QIcon addIcon(addMap);

ui->mainToolBar->addAction(addIcon,

"Добавить документ",

this,

SLOT(on\_action\_add\_triggered()));

// DELETE ELEMENT ON TOOL BAR

QPixmap deleteMap("C:\\Users\\User\\Documents\\untitled3\\images\\deleteMap.png");

QIcon deleteIcon(deleteMap);

ui->mainToolBar->addAction(deleteIcon,

"Удалить декларацию",

this,

SLOT(on\_action\_delete\_triggered()));

createConnection();

declList = new DeclList;

// LIST CONNECTIONS

connect(this,

SIGNAL(toUpdateTable(QSqlRecord)),

declList,

SLOT(slotToUpdateTable(QSqlRecord)));

connect(this,

SIGNAL(toRemoveSelectedRow()),

declList,

SLOT(slotToRemoveRow()));

connect(this,

SIGNAL(toRemoveTable()),

declList,

SLOT(slotToRemoveTable()));

connect(declList,

SIGNAL(toChangeNumRecords()),

this,

SLOT(slotToChangeNumRecords()));

connect(declList,

SIGNAL(toDeleteEntryFromDb(int)),

this,

SLOT(slotToDeleteEntryFromDb(int)));

connect(declList,

SIGNAL(toDeleteEntryFromStack(int)),

this,

SLOT(slotToDeleteEntryFromStack(int)));

connect(declList,

SIGNAL(tableItemDoubleClicked(QTableWidgetItem\*)),

this,

SLOT(tableItemDoubleClicked(QTableWidgetItem\*)));

ui->stackedWidget->addWidget(declList->getScrollArea());

ui->stackedWidget->setCurrentIndex(0);

dbSynchronization();

}

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

{

delete ui;

}

bool MainWindow::**createConnection**()

{

dbDecl = QSqlDatabase::addDatabase("QSQLITE", "DeclarationConnection");

dbDecl.setDatabaseName("C:\\Users\\User\\Documents\\untitled3\\decl.db3");

dbDecl.setPort(5432);

dbDecl.setHostName("localhost");

dbDecl.setUserName("postgres");

dbDecl.setPassword("root");

if(!dbDecl.open())

{

qDebug() << "Cannot open database:" << dbDecl.lastError();

return false;

}

else

{

return true;

}

}

void MainWindow::**dbSynchronization**()

{

query = new QSqlQuery(dbDecl);

QString str = "SELECT \* "

"FROM decl";

query->exec(str);

while(query->next())

{

++nRecords;

declContents = new Frame;

QScrollArea\* declScroll = new QScrollArea;

declContents->setFixedSize(700, 1000);

declScroll->setWidget(declContents);

ui->stackedWidget->addWidget(declScroll);

frameVec.push\_back(declContents);

frameConnections();

emit toUpdateTable(query->record());

toUpdateField\_1\_1(query->record());

toUpdateField\_1\_2(query->record());

toUpdateField\_1\_3(query->record());

toUpdateField\_5(query->record());

toUpdateField\_6(query->record());

toUpdateField\_7(query->record());

toUpdateField\_8(query->record());

toUpdateField\_9(query->record());

toUpdateField\_10(query->record());

toUpdateField\_11(query->record());

toUpdateField\_12(query->record());

toUpdateField\_13(query->record());

toUpdateField\_14(query->record());

toUpdateField\_15(query->record());

toUpdateField\_16(query->record());

toUpdateField\_18(query->record());

emit toUpdateTextEdit\_1();

emit toUpdateTextEdit\_2();

emit toUpdateTextEdit\_3();

}

}

void MainWindow::**setDataIntoDb**(const QString &text, int col)

{

model = new QSqlTableModel(nullptr, dbDecl);

model->*setTable*("decl");

model->*select*();

model->*setData*(model->*index*(ui->stackedWidget->currentIndex()-1, col), text);

model->submitAll();

}

void MainWindow::**setDataIntoDbGoods**(const QString &text, int row, int col)

{

model = new QSqlTableModel(nullptr, dbDecl);

model->*setTable*("goods");

model->*select*();

model->*setData*(model->*index*(row, col), text);

model->submitAll();

}

void MainWindow::**frameConnections**()

{

connect(declContents,

SIGNAL(toUpdateGoodsDial()),

this,

SLOT(slotToUpdateGoodsDial()),

Qt::UniqueConnection);

connect(this,

SIGNAL(toUpdateGoodsTable(QSqlRecord)),

declContents,

SLOT(slotToUpdateGoodsTable(QSqlRecord)),

Qt::UniqueConnection);

connect(declContents,

SIGNAL(deleteEntryFromDbGoods(int)),

this,

SLOT(slotDeleteEntryFromDbGoods(int)));

connect(declContents,

SIGNAL(toAskForUpdateDataSender()),

this,

SLOT(slotToGiveUpdateDataSender()),

Qt::UniqueConnection);

connect(this,

SIGNAL(toGiveUpdateDataSender(QSqlRecord)),

declContents,

SLOT(slotToUpdateTextSender(QSqlRecord)));

connect(declContents,

SIGNAL(sender\_checkBox\_stateChanged(int)),

this,

SLOT(sender\_on\_checkBox\_stateChanged(int)));

connect(declContents,

SIGNAL(sender\_textEdit\_textChanged(const QString&)),

this,

SLOT(sender\_on\_textEdit\_textChanged(const QString&)));

connect(declContents,

SIGNAL(sender\_lineEdit\_textChanged(const QString&)),

this,

SLOT(sender\_on\_lineEdit\_textChanged(const QString&)));

connect(declContents,

SIGNAL(sender\_lineEdit\_5\_textChanged(const QString&)),

this,

SLOT(sender\_on\_lineEdit\_5\_textChanged(const QString&)));

connect(declContents,

SIGNAL(sender\_lineEdit\_textChanged(const QString&)),

this,

SLOT(sender\_on\_lineEdit\_textChanged(const QString&)));

connect(declContents,

SIGNAL(sender\_lineEdit\_2\_textChanged(const QString&)),

this,

SLOT(sender\_on\_lineEdit\_2\_textChanged(const QString&)));

connect(declContents,

SIGNAL(sender\_lineEdit\_6\_textChanged(const QString&)),

this,

SLOT(sender\_on\_lineEdit\_6\_textChanged(const QString&)));

connect(declContents,

SIGNAL(sender\_lineEdit\_3\_textChanged(const QString&)),

this,

SLOT(sender\_on\_lineEdit\_3\_textChanged(const QString&)));

connect(declContents,

SIGNAL(sender\_lineEdit\_4\_textChanged(const QString&)),

this,

SLOT(sender\_on\_lineEdit\_4\_textChanged(const QString&)));

connect(declContents,

SIGNAL(sender\_ButtonFillClicked()),

this,

SLOT(sender\_slotButtonFillClicked()), Qt::UniqueConnection);

connect(declContents,

SIGNAL(receiver\_textEdit\_textChanged(const QString&)),

this,

SLOT(receiver\_on\_textEdit\_textChanged(const QString&)));

connect(declContents,

SIGNAL(receiver\_lineEdit\_textChanged(const QString&)),

this,

SLOT(receiver\_on\_lineEdit\_textChanged(const QString&)));

connect(declContents,

SIGNAL(receiver\_lineEdit\_5\_textChanged(const QString&)),

this,

SLOT(receiver\_on\_lineEdit\_5\_textChanged(const QString&)));

connect(declContents,

SIGNAL(receiver\_lineEdit\_2\_textChanged(const QString&)),

this,

SLOT(receiver\_on\_lineEdit\_2\_textChanged(const QString&)));

connect(declContents,

SIGNAL(receiver\_lineEdit\_6\_textChanged(const QString&)),

this,

SLOT(receiver\_on\_lineEdit\_6\_textChanged(const QString&)));

connect(declContents,

SIGNAL(receiver\_lineEdit\_3\_textChanged(const QString&)),

this,

SLOT(receiver\_on\_lineEdit\_3\_textChanged(const QString&)));

connect(declContents,

SIGNAL(receiver\_lineEdit\_4\_textChanged(const QString&)),

this,

SLOT(receiver\_on\_lineEdit\_4\_textChanged(const QString&)));

connect(declContents,

SIGNAL(receiver\_lineEdit\_8\_textChanged(const QString&)),

this,

SLOT(receiver\_on\_lineEdit\_8\_textChanged(const QString&)));

connect(declContents,

SIGNAL(receiver\_lineEdit\_7\_textChanged(const QString&)),

this,

SLOT(receiver\_on\_lineEdit\_7\_textChanged(const QString&)));

connect(declContents,

SIGNAL(receiver\_lineEdit\_9\_textChanged(const QString&)),

this,

SLOT(receiver\_on\_lineEdit\_9\_textChanged(const QString&)));

connect(declContents,

SIGNAL(receiver\_lineEdit\_10\_textChanged(const QString&)),

this,

SLOT(receiver\_on\_lineEdit\_10\_textChanged(const QString&)));

connect(declContents,

SIGNAL(receiver\_lineEdit\_11\_textChanged(const QString&)),

this,

SLOT(receiver\_on\_lineEdit\_11\_textChanged(const QString&)));

connect(declContents,

SIGNAL(receiver\_ButtonFillClicked()),

this,

SLOT(receiver\_slotButtonFillClicked()),

Qt::UniqueConnection);

connect(declContents,

SIGNAL(declarant\_textEdit\_textChanged(const QString&)),

this,

SLOT(declarant\_on\_textEdit\_textChanged(const QString&)));

connect(declContents,

SIGNAL(declarant\_lineEdit\_textChanged(const QString&)),

this,

SLOT(declarant\_on\_lineEdit\_textChanged(const QString&)));

connect(declContents,

SIGNAL(declarant\_lineEdit\_5\_textChanged(const QString&)),

this,

SLOT(declarant\_on\_lineEdit\_5\_textChanged(const QString&)));

connect(declContents,

SIGNAL(declarant\_lineEdit\_2\_textChanged(const QString&)),

this,

SLOT(declarant\_on\_lineEdit\_2\_textChanged(const QString&)));

connect(declContents,

SIGNAL(declarant\_lineEdit\_6\_textChanged(const QString&)),

this,

SLOT(declarant\_on\_lineEdit\_6\_textChanged(const QString&)));

connect(declContents,

SIGNAL(declarant\_lineEdit\_3\_textChanged(const QString&)),

this,

SLOT(declarant\_on\_lineEdit\_3\_textChanged(const QString&)));

connect(declContents,

SIGNAL(declarant\_lineEdit\_4\_textChanged(const QString&)),

this,

SLOT(declarant\_on\_lineEdit\_4\_textChanged(const QString&)));

connect(declContents,

SIGNAL(declarant\_lineEdit\_7\_textChanged(const QString&)),

this,

SLOT(declarant\_on\_lineEdit\_7\_textChanged(const QString&)));

connect(declContents,

SIGNAL(declarant\_lineEdit\_9\_textChanged(const QString&)),

this,

SLOT(declarant\_on\_lineEdit\_9\_textChanged(const QString&)));

connect(declContents,

SIGNAL(declarant\_lineEdit\_10\_textChanged(const QString&)),

this,

SLOT(declarant\_on\_lineEdit\_10\_textChanged(const QString&)));

connect(declContents,

SIGNAL(declarant\_lineEdit\_11\_textChanged(const QString&)),

this,

SLOT(declarant\_on\_lineEdit\_11\_textChanged(const QString&)));

connect(declContents,

SIGNAL(declarant\_ButtonFillClicked()),

this,

SLOT(declarant\_slotButtonFillClicked()),

Qt::UniqueConnection);

connect(declContents,

SIGNAL(declarant\_comboBox\_activated(const QString&)),

this,

SLOT(declarant\_comboBox\_activated(const QString&)),

Qt::UniqueConnection);

connect(declContents,

SIGNAL(goods\_textEdit\_textChanged(const QString&, int)),

this,

SLOT(goods\_on\_textEdit\_textChanged(const QString&, int)));

connect(declContents,

SIGNAL(goods\_tableWidget\_itemChanged(QTableWidgetItem\*)),

this,

SLOT(goods\_on\_tableWidget\_itemChanged(QTableWidgetItem\*)));

connect(declContents,

SIGNAL(goodsButtonAddClicked()),

this,

SLOT(slotGoodsButtonAddClicked()));

connect(declContents,

SIGNAL(field\_5\_textChanged(const QString&)),

this,

SLOT(slotField\_5\_textChanged(const QString&)));

connect(declContents,

SIGNAL(field\_6\_textChanged(const QString&)),

this,

SLOT(slotField\_6\_textChanged(const QString&)));

connect(declContents,

SIGNAL(field\_7\_textChanged(const QString&)),

this,

SLOT(slotField\_7\_textChanged(const QString&)));

connect(declContents,

SIGNAL(field\_8\_textChanged(const QString&)),

this,

SLOT(slotField\_8\_textChanged(const QString&)));

connect(declContents,

SIGNAL(field\_9\_textChanged(const QString&)),

this,

SLOT(slotField\_9\_textChanged(const QString&)));

connect(declContents,

SIGNAL(field\_10\_textChanged(const QString&)),

this,

SLOT(slotField\_10\_textChanged(const QString&)));

connect(declContents,

SIGNAL(field\_11\_textChanged(const QString&)),

this,

SLOT(slotField\_11\_textChanged(const QString&)));

connect(declContents,

SIGNAL(field\_12\_textChanged(const QString&)),

this,

SLOT(slotField\_12\_textChanged(const QString&)));

connect(declContents,

SIGNAL(field\_13\_textChanged(const QString&)),

this,

SLOT(slotField\_13\_textChanged(const QString&)));

connect(declContents,

SIGNAL(field\_14\_textChanged(const QString&)),

this,

SLOT(slotField\_14\_textChanged(const QString&)));

connect(declContents,

SIGNAL(field\_15\_textChanged(const QString&)),

this,

SLOT(slotField\_15\_textChanged(const QString&)));

connect(declContents,

SIGNAL(field\_16\_textChanged(const QString&)),

this,

SLOT(slotField\_16\_textChanged(const QString&)));

connect(declContents,

SIGNAL(field\_18\_textChanged(const QString&)),

this,

SLOT(slotField\_18\_textChanged(const QString&)));

}

MainWindow.h

#pragma once

#include <QMainWindow>

#include <QtSql>

#include <QVector>

#include "decltypewindow.h"

#include "decllist.h"

class Frame;

namespace Ui {

class MainWindow;

}

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

explicit MainWindow(QWidget \*parent = nullptr);

~MainWindow();

bool createConnection();

void dbSynchronization();

void setDataIntoDb(const QString &text, int col);

void setDataIntoDbGoods(const QString &text, int row, int col);

void frameConnections();

void toUpdateField\_1\_1(QSqlRecord rec);

void toUpdateField\_1\_2(QSqlRecord rec);

void toUpdateField\_1\_3(QSqlRecord rec);

void toUpdateField\_5(QSqlRecord rec);

void toUpdateField\_6(QSqlRecord rec);

void toUpdateField\_7(QSqlRecord rec);

void toUpdateField\_8(QSqlRecord rec);

void toUpdateField\_9(QSqlRecord rec);

void toUpdateField\_10(QSqlRecord rec);

void toUpdateField\_11(QSqlRecord rec);

void toUpdateField\_12(QSqlRecord rec);

void toUpdateField\_13(QSqlRecord rec);

void toUpdateField\_14(QSqlRecord rec);

void toUpdateField\_15(QSqlRecord rec);

void toUpdateField\_16(QSqlRecord rec);

void toUpdateField\_18(QSqlRecord rec);

signals:

void toRemoveSelectedRow();

void toChangeTextEdit1(QSqlRecord rec);

void toChangeTextEdit2(QSqlRecord rec);

void toChangeTextEdit3(QSqlRecord rec);

void toUpdateTable(QSqlRecord rec);

void toUpdateGoodsTable(QSqlRecord rec);

void toUpdateList();

void toBlockAllFields();

void toUnblockAllFields();

void toRemoveTable();

void toUpdateTextEdit\_1();

void toUpdateTextEdit\_2();

void toUpdateTextEdit\_3();

void toGiveUpdateDataSender(QSqlRecord rec);

private slots:

void on\_action\_6\_triggered();

void on\_action\_7\_triggered();

void tableItemDoubleClicked(QTableWidgetItem\* item);

void treeItemDoubleClicked(QTreeWidgetItem\* treeItem, int column);

void sender\_on\_checkBox\_stateChanged(int state);

void sender\_on\_textEdit\_textChanged(const QString &text);

void sender\_on\_lineEdit\_5\_textChanged(const QString &text);

void sender\_on\_lineEdit\_textChanged(const QString &text);

void sender\_on\_lineEdit\_2\_textChanged(const QString &text);

void sender\_on\_lineEdit\_6\_textChanged(const QString &text);

void sender\_on\_lineEdit\_3\_textChanged(const QString &text);

void sender\_on\_lineEdit\_4\_textChanged(const QString &text);

void sender\_slotButtonFillClicked();

void receiver\_on\_textEdit\_textChanged(const QString &text);

void receiver\_on\_lineEdit\_5\_textChanged(const QString &text);

void receiver\_on\_lineEdit\_textChanged(const QString &text);

void receiver\_on\_lineEdit\_2\_textChanged(const QString &text);

void receiver\_on\_lineEdit\_6\_textChanged(const QString &text);

void receiver\_on\_lineEdit\_3\_textChanged(const QString &text);

void receiver\_on\_lineEdit\_4\_textChanged(const QString &text);

void receiver\_on\_lineEdit\_8\_textChanged(const QString &text);

void receiver\_on\_lineEdit\_7\_textChanged(const QString &text);

void receiver\_on\_lineEdit\_9\_textChanged(const QString &text);

void receiver\_on\_lineEdit\_10\_textChanged(const QString &text);

void receiver\_on\_lineEdit\_11\_textChanged(const QString &text);

void receiver\_slotButtonFillClicked();

void declarant\_on\_textEdit\_textChanged(const QString &text);

void declarant\_on\_lineEdit\_5\_textChanged(const QString &text);

void declarant\_on\_lineEdit\_textChanged(const QString &text);

void declarant\_on\_lineEdit\_2\_textChanged(const QString &text);

void declarant\_on\_lineEdit\_6\_textChanged(const QString &text);

void declarant\_on\_lineEdit\_3\_textChanged(const QString &text);

void declarant\_on\_lineEdit\_4\_textChanged(const QString &text);

void declarant\_on\_lineEdit\_8\_textChanged(const QString &text);

void declarant\_on\_lineEdit\_7\_textChanged(const QString &text);

void declarant\_on\_lineEdit\_9\_textChanged(const QString &text);

void declarant\_on\_lineEdit\_10\_textChanged(const QString &text);

void declarant\_on\_lineEdit\_11\_textChanged(const QString &text);

void declarant\_slotButtonFillClicked();

void declarant\_comboBox\_activated(const QString &text);

void goods\_on\_textEdit\_textChanged(const QString &text, int row);

void goods\_on\_tableWidget\_itemChanged(QTableWidgetItem \*item);

void slotDeleteEntryFromDbGoods(int id);

void slotGoodsButtonAddClicked();

void slotField\_5\_textChanged(const QString& text);

void slotField\_6\_textChanged(const QString& text);

void slotField\_7\_textChanged(const QString& text);

void slotField\_8\_textChanged(const QString& text);

void slotField\_9\_textChanged(const QString& text);

void slotField\_10\_textChanged(const QString& text);

void slotField\_11\_textChanged(const QString& text);

void slotField\_12\_textChanged(const QString& text);

void slotField\_13\_textChanged(const QString& text);

void slotField\_14\_textChanged(const QString& text);

void slotField\_15\_textChanged(const QString& text);

void slotField\_16\_textChanged(const QString& text);

void slotField\_18\_textChanged(const QString& text);

void slotToUpdateTextEdit\_1();

void slotToUpdateTextEdit\_2();

void slotToUpdateTextEdit\_3();

void on\_action\_declList\_triggered();

void on\_action\_delete\_triggered();

void on\_action\_add\_triggered();

void on\_action\_update\_triggered();

void slotToChangeNumRecords();

void slotToDeleteEntryFromDb(int id);

void slotToDeleteEntryFromStack(int id);

void slotToGiveUpdateDataSender();

void slotToUpdateGoodsDial();

private:

Frame \*declContents;

DeclList \*declList;

Ui::MainWindow \*ui;

DeclTypeWindow \*declWindow;

QSqlDatabase dbDecl;

QSqlQuery \*query;

QMenu \*declMenu;

int nRecords;

QSqlTableModel \*model;

QVector<Frame\*> frameVec;

QAction\* fieldAction;

};