

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«МИРЭА—Российский технологический университет»**

**РТУ МИРЭА**

**Институт комплексной безопасности и специального приборостроения**

(наименование института, филиала)

Кафедра КБ-2 «Прикладные информационные технологии»

(наименование кафедры)

**Курсовая работа**

по дисциплине \_\_\_\_\_\_\_\_Базы данных и экспертные системы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(наименование дисциплины)

**Тема курсовой работы** Создание клиент-серверного приложения для работы с базой данных

**Студент группы \_\_БИСО-03-20 Рудзик Александр Романович\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(учебная группа, фамилия, имя, отчество студента) (подпись студента)

**Руководитель курсовой работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

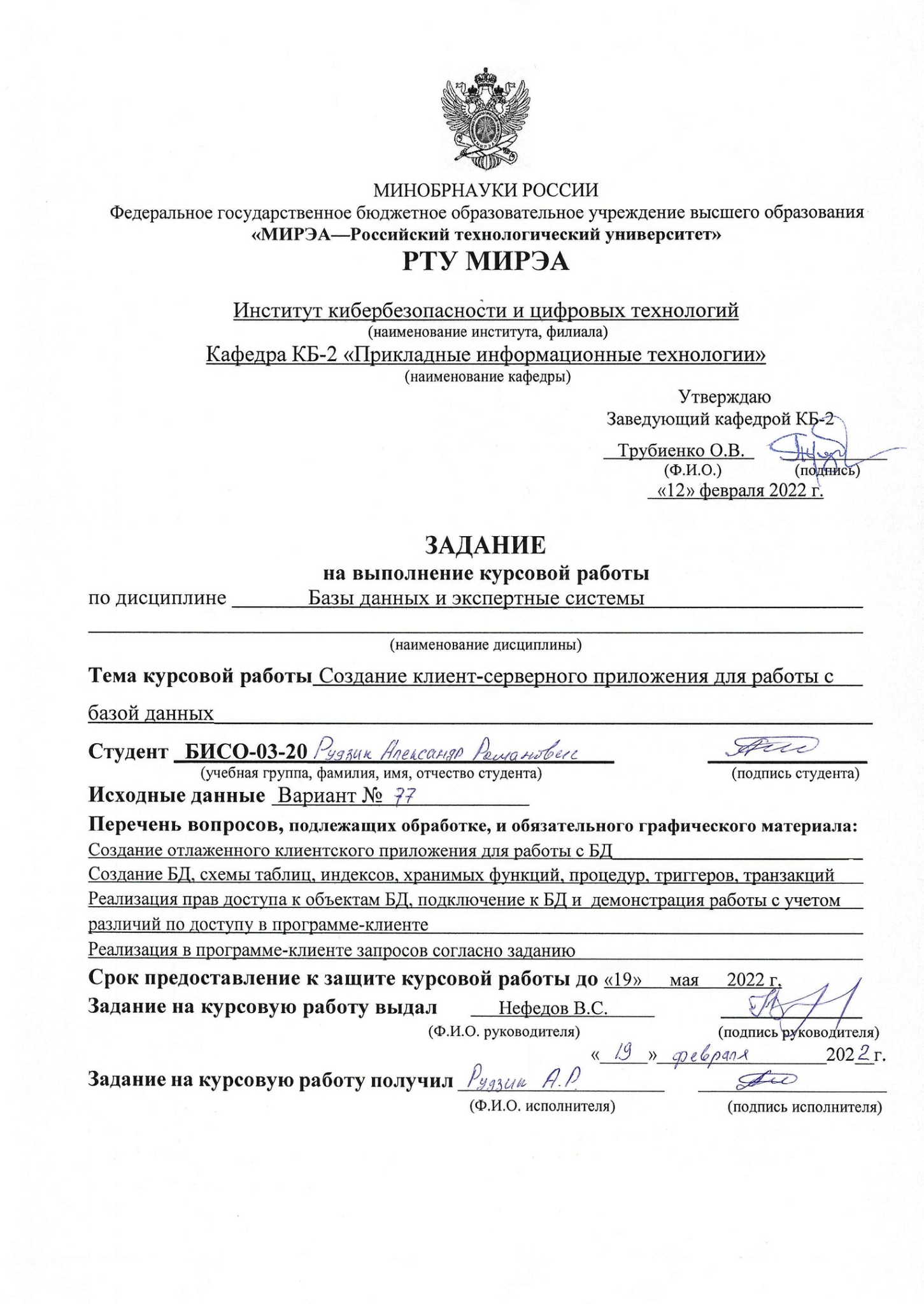
(должность, звание, ученая степень) (подпись руководителя)

**Рецензент** (при наличии)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(должность, звание, ученая степень) (подпись рецензента)

Работа представлена к защите «\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_г.

Допущен к защите «\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_г.



**Оглавление**

**Постановка задачи 4**

**Теоретическая часть 6**

**Практическая часть 9**

**Листинг программы 21**

**Демонстрация работы приложения 49**

**Заключение 55**

**Список источников 56**

**Постановка задачи**

Разработать клиент-серверное приложение, 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 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):

1. Составной многотабличный запрос с CASE-выражением;
2. Многотабличный VIEW, с возможностью его обновления;
3. Запросы, содержащие подзапрос в разделах SELECT, FROM и WHERE (в каждом хотя бы по одному);
4. Коррелированные подзапросы (минимум 3 запроса).
5. Многотабличный запрос, содержащий группировку записей, агрегатные функции и параметр, используемый в разделе HAVING;
6. Запросы, содержащий предикат ANY(SOME) или ALL (для каждого предиката);

4. 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.

5. 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).

6. Операции добавления, удаления и обновления реализовать в виде хранимых процедур или функций с параметрами для всех таблиц;

7. Реализовать отдельную хранимую процедуру или функцию, состоящую из нескольких отдельных операций в виде единой транзакции, которая при определенных условиях может быть зафиксирована или откатана;

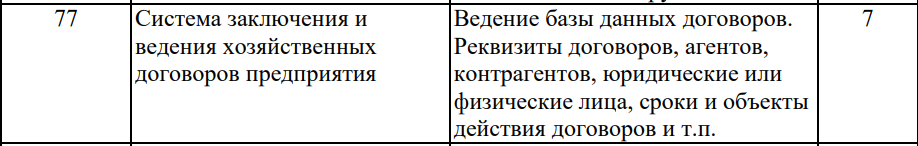
8. Реализовать курсор на обновления отдельных данных (вычисления значения полей выбранной таблицы);

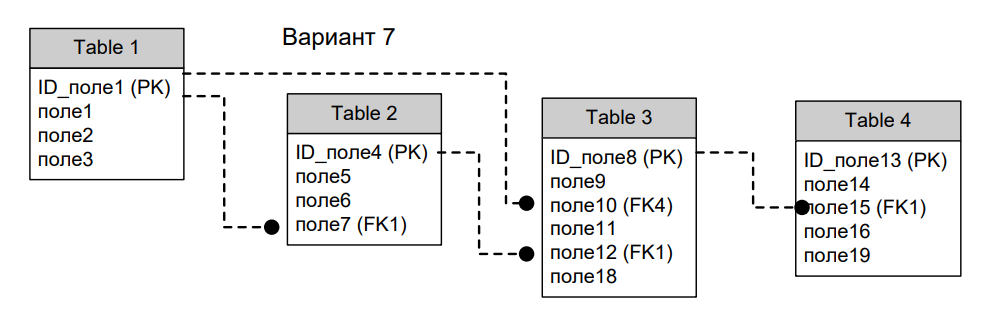
9. Реализовать собственную скалярную и векторную функции. Функции сохранить в базе данных;

10. Распределение прав пользователей: предусмотреть не менее двух пользователей с разным набором привилегий. Каждый набор привилегий оформить в виде роли.

Клиент должен обеспечивать добавление, модификацию и удаление данных по всей предметной области. Добавление, редактирование данных в таблице производить в отдельном окне.

Запрещено в качестве вводимых данных, в том числе для связи таблиц, указывать значения первичных и внешних ключей – для обеспечения ссылочной целостности пользователь должен выбирать значения из справочника, а соответствующие значения должны подставляться программно (тем или иным способом – автоматически).





**Теоретическая часть**

В основе решения многих задач лежит обработка информации. Для

облегчения обработки информации создаются информационные системы (ИС). Автоматизированными называют ИС, в которых применяют технические средства.

База данных (БД) представляет собой совокупность специальным образом организованных данных, хранимых в памяти вычислительной системы и отображающих состояние объектов, и их взаимосвязей в рассматриваемой предметной области.

Базы данных бывают централизованными, которые хранятся на одном

компьютере, и распределенными, которые хранятся на нескольких компьютерах некоторой сети.

Логическую структуру хранимых в базе данных называют моделью

представления данных. К основным моделям представления данных относятся

следующие: иерархическая, сетевая, реляционная, постреляционная, многомерная и объектно-ориентированная.

Система управления базами данных (СУБД) — это комплекс языковых и программных средств, предназначенных для создания, ведения и совместного использования БД многими пользователями. Обычно СУБД различают по используемой модели данных. Так, СУБД, основанные на использовании реляционной модели данных, называют реляционными СУБД.

Приложение представляет собой программу или комплекс программ,

обеспечивающих автоматизацию обработки информации для прикладной задачи. Приложения, разработанные в среде СУБД, часто называют приложениями СУБД, а приложения, разработанные вне СУБД, — внешними приложениями.

Сервером определенного ресурса в компьютерной сети называется компьютер или программа, управляющие этим ресурсом, клиентом — компьютер или программа, использующие этот ресурс. В качестве ресурса компьютерной сети могут выступать, к примеру, базы данных, файловые системы, службы печати, почтовые службы. Тип сервера определяется видом ресурса, которым он управляет. Например, если управляемым ресурсом является база данных, то соответствующий сервер называется сервером базы данных.

Достоинством организации информационной системы по архитектуре клиент- сервер является удачное сочетание централизованного хранения, обслуживания и коллективного доступа к общей корпоративной информации с индивидуальной работой пользователей над персональной информацией. Архитектура клиент-сервер допускает различные варианты реализации.

Важнейшим достоинством применения БД в информационных системах является обеспечение независимости данных от прикладных программ. Это дает возможность пользователям не заниматься проблемами представления данных на физическом уровне: размещения данных в памяти, методов доступа к ним ит. д.

Для работы с хранящейся в базе данных информацией СУБД предоставляет программам и пользователям следующие два типа языков:

 язык описания данных - высокоуровневый непроцедурный язык декларативного типа, предназначенный для описания логической структуры данных;

 язык манипулирования данными - совокупность конструкций, обеспечивающих выполнение основных операций по работе с данными: ввод, модификацию и выборку данных по запросам.

ЗОЕ (Знисвте Оцегу Гапеиаре) — структурированный язык запросов — сочетает в себе свойства языков обоих типов — описания и манипулирования данными.

Механизм транзакций используется в СУБД для поддержания целостности данных в базе. Транзакцией называется некоторая неделимая последовательность операций надданными БД, которая отслеживается СУБД от начала и до завершения. Если по каким-либо причинам (сбои и отказы оборудования, ошибки в программном обеспечении, включая приложение) транзакция остается незавершенной, то она отменяется.

Ведение журнала изменений в БД выполняется СУБД для обеспечения надежности хранения данных в базе при наличии аппаратных сбоев и отказов, а также ошибок в программном обеспечении.

Обеспечение целостности БД составляет необходимое условие успешного функционирования БД, особенно для случая использования БД в сетях. Целостность БД есть свойство базы данных, означающее, что в ней содержится полная, непротиворечивая и адекватно отражающая предметную область информация. Поддержание целостности БД включает проверку целостности и ее восстановление в случае обнаружения противоречий в базе данных. Целостное состояние БД описывается с помощью ограничений целостности в виде условий, которым должны удовлетворять хранимые в базе данные.

Нормальная форма — свойство отношения в реляционной модели данных, характеризующее его с точки зрения избыточности, потенциально приводящей к логически ошибочным результатам выборки или изменения данных. Нормальная форма определяется как совокупность требований, которым должно удовлетворять отношение.

Процесс преобразования отношений базы данных к виду, отвечающему нормальным формам, называется нормализацией. Нормализация предназначена для приведения структуры БД к виду, обеспечивающему минимальную логическую избыточность, и не имеет целью уменьшение или увеличение производительности работы или же уменьшение, или увеличение физического объёма базы данных. Конечной целью нормализации является уменьшение потенциальной противоречивости, хранимой в базе данных информации. Как отмечает К. Дейт, общее назначение процесса нормализации заключается в следующем:

 исключение некоторых типов избыточности;

 устранение некоторых аномалий обновления;

 разработка проекта базы данных, который является достаточно «качественным» представлением реального мира, интуитивно понятен и может служить хорошей основой для последующего расширения;

 упрощение процедуры применения необходимых ограничений целостности.

Устранение избыточности производится, как правило, за счёт декомпозиции отношений таким образом, чтобы в каждом отношении хранились только первичные факты (то есть факты, не выводимые из других хранимых фактов).

Первая нормальная форма

Переменная отношения находится в первой нормальной форме (1НФ) тогда и только тогда, когда в любом допустимом значении отношения каждый его кортеж содержит только одно значение для каждого из атрибутов.

В реляционной модели отношение всегда находится в первой нормальной форме по определению понятия отношение. Что же касается различных таблиц, то они могут не быть правильными представлениями отношений и, соответственно, могут не находиться в 1НФ.

Вторая нормальная форма

Переменная отношения находится во второй нормальной форме тогда и только тогда, когда она находится в первой нормальной форме и каждый не ключевой атрибут неприводимо (функционально полно) зависит от ее потенциального ключа.

Третья нормальная форма

Переменная отношения находится в третьей нормальной форме тогда и только тогда, когда она находится во второй нормальной форме, и отсутствуют транзитивные функциональные зависимости неключевых атрибутов от ключевых.

**Практическая часть**

1. В рамках курсовой работы для эффективного выполнения задачи была спроектирована база данных, представленная на рисунке 1.

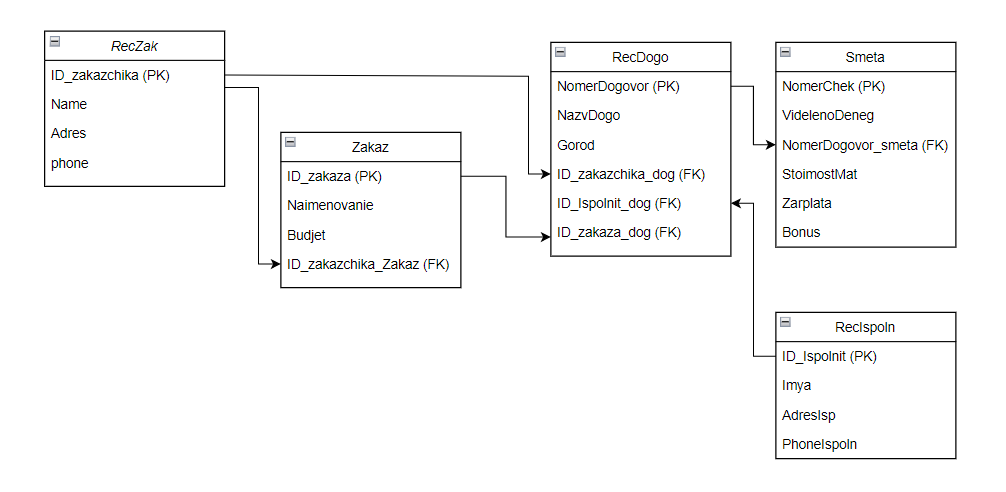


Рисунок 1 – схема базы данных

Скрипт создания базы данных:

CREATE TABLE [dbo].[RecZak]

(

[ID\_zakazchika] INT NOT NULL Primary key identity(1,1),

[Name] nvarchar(20) not null,

[Adres] varchar(20) not null,

[phone] varchar(20) not null

);

CREATE TABLE [dbo].[Zakaz]

(

[ID\_zakaza] INT NOT NULL Primary key identity(1,1),

[Naimenovanie] varchar(20) not null,

[Budjet] nvarchar(20) not null,

[ID\_zakazchika\_Zakaz] int not null

);

CREATE TABLE [dbo].[RecDogo]

(

[NomerDogovor] int NOT NULL Primary key identity(1,1),

[NazvDogo] nvarchar(20) not null,

[Gorod] nvarchar(20) not null,

[ID\_Zakazchika\_dog] int not null,

[ID\_Ispolnit\_dog] int not null,

[ID\_zakaza\_dog] int not null,

[date] SMALLDATETIME, --автобинд времени

);

CREATE TABLE [dbo].[Smeta]

(

[NomerChek] INT not null primary key identity(1,1),

[VidelenoDeneg] nvarchar(30) not null,

[NomerDogovor\_smeta] int not null,

[StoimostMat] nvarchar(20) not null,

[Zarplata] nvarchar(20) not null,

[Bonus] nvarchar(20) not null

);

CREATE TABLE [dbo].[RecIspoln]

(

[ID\_Ispolnit] INT NOT NULL Primary key identity(1,1),

[Imya] varchar(20) not null,

[AdresIsp] nvarchar(20) not null,

[PhoneIspoln] nvarchar(20) not null

);

ALTER TABLE Zakaz

ADD CONSTRAINT FK\_ID\_zakazchika\_Zakaz FOREIGN KEY (ID\_zakazchika\_Zakaz)

REFERENCES RecZak (ID\_zakazchika)

ON DELETE CASCADE

ON UPDATE CASCADE

ALTER TABLE RecDogo

ADD CONSTRAINT FK\_ID\_zakazchika\_dog FOREIGN KEY (ID\_zakazchika\_dog)

REFERENCES RecZak (ID\_zakazchika)

ON DELETE NO ACTION

ON UPDATE NO ACTION

ALTER TABLE RecDogo

ADD CONSTRAINT FK\_ID\_zakaza\_dog FOREIGN KEY (ID\_zakaza\_dog)

REFERENCES Zakaz (ID\_zakaza)

ON DELETE CASCADE

ON UPDATE CASCADE

ALTER TABLE RecDogo

ADD CONSTRAINT FK\_ID\_Ispolnit\_dog FOREIGN KEY (ID\_Ispolnit\_dog)

REFERENCES RecIspoln (ID\_Ispolnit)

ON DELETE CASCADE

ON UPDATE CASCADE

ALTER TABLE Smeta

ADD CONSTRAINT FK\_NomerDogovor\_smeta FOREIGN KEY (NomerDogovor\_smeta)

REFERENCES RecDogo (NomerDogovor)

ON DELETE CASCADE

ON UPDATE CASCADE

1. В приложении реализовать не менее пяти запросов, включая (для демонстрации навыков работы)
   1. Составной многотабличный запрос с CASE-выражением;

void prosmotrZakaz::on\_pushButton\_2\_clicked()

{

model22 = new QSqlQueryModel();

model22 -> setQuery("EXEC get\_zakaz\_by\_namezakazchik @n = " + ui->lineEdit->text());

model22->*setHeaderData*(0, Qt::Horizontal, "Имя заказчика");

model22->*setHeaderData*(1, Qt::Horizontal, "Номер телефона");

model22->*setHeaderData*(2, Qt::Horizontal, "Наименование");

model22->*setHeaderData*(3, Qt::Horizontal, "Бюджет");

model22->*setHeaderData*(4, Qt::Horizontal, "Статус");

ui->tableView->*setModel*(*model22*);

ui->tableView->resizeColumnsToContents();

ui->tableView->show();

}

* 1. Многотабличный VIEW, с возможностью его обновления;
  2. Запросы, содержащие подзапрос в разделах SELECT, FROM и WHERE (в каждом хотя бы по одному);

void prosmotrdogovor::on\_pushButton\_2\_clicked() //вывод где бонус больше вводимой суммы

{

model51 = new QSqlQueryModel();

model51 -> setQuery("EXEC where\_bonus\_more @d = " + ui->lineEdit->text());

model51->*setHeaderData*(0, Qt::Horizontal, "Номер договора");

model51->*setHeaderData*(1, Qt::Horizontal, "Имя заказчика");

model51->*setHeaderData*(2, Qt::Horizontal, "Бонус");

ui->tableView->*setModel*(*model51*);

ui->tableView->resizeColumnsToContents();

ui->tableView->show();

}

void prosmotrdogovor::on\_pushButton\_3\_clicked() //вывод договора с одинак авансом и стоим.матер. выше вводим суммы

{

model52 = new QSqlQueryModel();

model52 -> setQuery("EXEC dogovori\_s\_odinak\_avansom @o = " + ui->lineEdit\_2->text() + ", @y = " + ui->lineEdit\_3->text());

model52->*setHeaderData*(0, Qt::Horizontal, "Номер договора");

model52->*setHeaderData*(1, Qt::Horizontal, "Наименование заказа");

ui->tableView->*setModel*(*model52*);

ui->tableView->resizeColumnsToContents();

ui->tableView->show();

}

* 1. Коррелированные подзапросы (минимум 3 запроса).

void prosmotrIspolnit::on\_pushButton\_clicked()

{

model1 = new QSqlQueryModel();

model1 -> setQuery ("SELECT NazvDogo, date FROM RecDogo WHERE date = (SELECT MIN(date) FROM RecDogo)");

model1->*setHeaderData*(0, Qt::Horizontal, "Название договора");

model1->*setHeaderData*(1, Qt::Horizontal, "Дата создания");

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

ui->tableView->resizeColumnsToContents();

ui->tableView->show();

}

void prosmotrIspolnit::on\_pushButton\_2\_clicked()

{

model2 = new QSqlQueryModel();

model2 -> setQuery ("SELECT NazvDogo, date FROM RecDogo WHERE date = (SELECT MAX(date) FROM RecDogo)");

model2->*setHeaderData*(0, Qt::Horizontal, "Название договора");

model2->*setHeaderData*(1, Qt::Horizontal, "Дата создания");

ui->tableView->*setModel*(*model2*);

ui->tableView->resizeColumnsToContents();

ui->tableView->show();

}

* 1. Многотабличный запрос, содержащий группировку записей, агрегатные функции и параметр, используемый в разделе HAVING;

void prosmotrdogovor::on\_pushButton\_5\_clicked()

{

model55 = new QSqlQueryModel();

model55 -> setQuery("EXEC kolich\_zak\_v\_dog @s = " + ui->lineEdit\_4->text());

model55->*setHeaderData*(0, Qt::Horizontal, "Название договора");

model55->*setHeaderData*(1, Qt::Horizontal, "Город");

model55->*setHeaderData*(2, Qt::Horizontal, "Всего заказов");

ui->tableView->*setModel*(*model55*);

ui->tableView->resizeColumnsToContents();

ui->tableView->show();

}

* 1. Запросы, содержащий предикат ANY(SOME) или ALL;

void ProsmotrZakazchika::on\_pushButton\_2\_clicked()

{

model31 = new QSqlQueryModel();

model31 -> setQuery("EXEC zakazov\_bolsche\_odnogo @p = " + ui->lineEdit->text());

model31->*setHeaderData*(0, Qt::Horizontal, "Имя");

model31->*setHeaderData*(1, Qt::Horizontal, "Адрес");

model31->*setHeaderData*(2, Qt::Horizontal, "Телефон");

ui->tableView->*setModel*(*model31*);

ui->tableView->resizeColumnsToContents();

ui->tableView->show();

}

1. Реализовать запросы по заданиям (в любых фрагментах скриптов как на стороне сервера, так и на стороне клиента):
2. Составной многотабличный запрос с CASE-выражением;

-- Вывод заказа по имени заказчика и выведение статуса заказа

CREATE PROCEDURE get\_zakaz\_by\_namezakazchik @n nvarchar(20)

AS

SELECT

Name,

phone,

Naimenovanie,

Zakaz.Budjet,

(CASE WHEN dbo.Zakaz.Budjet > 1000000 THEN 'Гос.заказ' ELSE 'Частный заказ' END) AS [Status]

FROM RecZak JOIN Zakaz

ON RecZak.ID\_zakazchika = Zakaz.ID\_zakazchika\_Zakaz

WHERE Name = @n;

1. Многотабличный VIEW, с возможностью его обновления;

-- Вывод заказов по названию договора и городу подписания

CREATE OR ALTER VIEW zakazi\_po\_nazv\_gorod

AS

SELECT Naimenovanie, Budjet, RecDogo.NazvDogo, RecDogo.Gorod, RecDogo.ID\_Zakazchika\_dog FROM Zakaz JOIN RecDogo

ON ID\_zakaza = ID\_zakaza\_dog

1. Запросы, содержащие подзапрос в разделах SELECT, FROM и WHERE (в каждом хотя бы по одному);

-- Вывод количества договоров всего

CREATE PROCEDURE get\_kolichestvo\_dogovorov

AS

SELECT COUNT(\*) AS kolichDogovorov, (SELECT COUNT(\*) FROM Zakaz) AS kolichZakazov FROM RecDogo

EXEC get\_kolichestvo\_dogovorov

drop procedure get\_kolichestvo\_dogovorov

-- Вывод номера договора и ИД заказчика по сумме бонуса выше вводимой суммы

CREATE PROCEDURE where\_bonus\_more @d float(5)

AS

SELECT NomerDogovor, (SELECT Name FROM RecZak WHERE ID\_Zakazchika\_dog = ID\_zakazchika), Bonus FROM (SELECT NomerDogovor, ID\_Zakazchika\_dog, Bonus FROM Smeta JOIN RecDogo ON NomerDogovor = Smeta.NomerDogovor\_smeta WHERE Smeta.Bonus > @d)

AS Smeta WHERE Smeta.Bonus>@d

EXEC where\_bonus\_more 10000

drop procedure where\_bonus\_more

--JOIN RecZak ON RecDogo.ID\_Zakazchika\_dog = RecZak.ID\_zakazchika -- , RecZak.Name

-- Вывод номера договора и наименование заказа с одинаковым авансом и стоимостью материалов выше вводимой суммы

-- avans=videlenoDeneg

CREATE PROCEDURE dogovori\_s\_odinak\_avansom @o float(5), @y float(5)

AS

SELECT NomerDogovor, (SELECT Naimenovanie FROM Zakaz WHERE ID\_zakaza\_dog = ID\_zakaza) FROM RecDogo cons WHERE @o IN (SELECT VidelenoDeneg FROM Smeta where NomerDogovor = NomerDogovor\_smeta and StoimostMat > @y)

EXEC dogovori\_s\_odinak\_avansom 234000, 70000

drop procedure dogovori\_s\_odinak\_avansom

1. Коррелированные подзапросы (минимум 3 запроса).

--самый старый договор--

SELECT NazvDogo, date FROM RecDogo WHERE date = (SELECT MIN(date) FROM RecDogo)

--самый новый договор--

SELECT NazvDogo, date FROM RecDogo WHERE date = (SELECT MAX(date) FROM RecDogo)

--найти договоры без сметы--

SELECT \* FROM RecDogo r

WHERE not EXISTS(SELECT NomerDogovor\_smeta FROM Smeta WHERE NomerDogovor\_smeta = r.NomerDogovor)

1. Многотабличный запрос, содержащий группировку записей, агрегатные функции и параметр, используемый в разделе HAVING;

--количество заказов в договоре

CREATE PROCEDURE kolich\_zak\_v\_dog @s int

AS

SELECT NazvDogo, Gorod, COUNT(Zakaz.Naimenovanie) AS [Всего заказов]

FROM RecDogo JOIN Zakaz

ON RecDogo.ID\_zakaza\_dog = Zakaz.ID\_zakaza

GROUP BY NazvDogo, Gorod

HAVING COUNT(Zakaz.Naimenovanie) >= @s

ORDER BY NazvDogo, Gorod;

1. Запросы, содержащий предикат ANY(SOME) или ALL;

-- Вывод где кол-во заказов у заказчика больше @p

CREATE PROCEDURE zakazov\_bolsche\_odnogo @p int(5)

AS

SELECT Name, Adres, phone FROM RecZak WHERE @p < ALL( SELECT COUNT(\*) FROM Zakaz WHERE RecZak.ID\_Zakazchika = ID\_zakazchika\_Zakaz)

1. Создать индексы для увеличения скорости выполнения запросов;

CREATE INDEX RecZak\_Name\_Adres\_phone\_idx

ON RecZak(Name, Adres, phone DESC)

drop index RecZak\_Name\_Adres\_phone\_idx on RecZak

CREATE UNIQUE INDEX zakaz\_idx

ON Zakaz (Naimenovanie, Budjet ASC)

drop index zakaz\_idx on Zakaz

CREATE INDEX RecDogo\_klast\_idx

ON RecDogo(NomerDogovor)

drop index RecDogo\_klast\_idx on RecDogo

1. В таблице (в соответствии с вариантом) предусмотреть поле, которое заполняется 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).

--добавление даты(автобинд времени) при добавлении договора

CREATE TRIGGER update\_dogovorn\_time ON RecDogo FOR INSERT

AS

SET NOCOUNT ON

UPDATE RecDogo SET date = CURRENT\_TIMESTAMP

FROM RecDogo r join inserted i ON r.NomerDogovor = i.NomerDogovor

--обновление даты(автобинд времени) при обновлении договора

CREATE TRIGGER update\_dogovorn\_time\_1 ON RecDogo FOR UPDATE

AS

SET NOCOUNT ON

UPDATE RecDogo SET date = CURRENT\_TIMESTAMP

FROM RecDogo r join inserted i ON r.NomerDogovor = i.NomerDogovor

drop trigger update\_dogovorn\_time\_1

1. Операции добавления, удаления и обновления реализовать в виде хранимых процедур (функций) с параметрами для всех таблиц;

CREATE PROCEDURE add\_zakaz @dnz varchar(20), @dbz nvarchar(20), @zz nvarchar(20)

AS

INSERT INTO Zakaz(Naimenovanie, Budjet, ID\_zakazchika\_Zakaz) VALUES(@dnz, @dbz, (SELECT ID\_zakazchika FROM RecZak WHERE Name = @zz))

drop procedure add\_zakaz

CREATE PROCEDURE add\_ispolnit @im varchar(20), @ai nvarchar(20), @ti nvarchar(20)

AS

INSERT INTO RecIspoln(Imya, AdresIsp, PhoneIspoln) VALUES(@im, @ai, @ti)

drop procedure add\_ispolnit

CREATE PROCEDURE add\_zakazchik @zz nvarchar(20), @az varchar(20), @tz varchar(20)

AS

INSERT INTO RecZak(Name, Adres, phone) VALUES(@zz, @az, @tz)

drop procedure add\_zakazchik

CREATE PROCEDURE add\_dogovor @rdnd nvarchar(20), @rdg nvarchar(20), @zz nvarchar(20), @im varchar(20), @dnz varchar(20)

AS

INSERT INTO RecDogo(NazvDogo, Gorod, ID\_Zakazchika\_dog, ID\_Ispolnit\_dog, ID\_zakaza\_dog) VALUES(@rdnd, @rdg, (SELECT ID\_zakazchika FROM RecZak WHERE Name = @zz), (SELECT ID\_Ispolnit FROM RecIspoln WHERE Imya = @im), (SELECT ID\_zakaza FROM Zakaz WHERE Naimenovanie = @dnz))

drop procedure add\_dogovor

CREATE PROCEDURE add\_smeta @svd nvarchar(30), @rdnd nvarchar(20), @ssm nvarchar(20), @sz nvarchar(20), @sb nvarchar(20)

AS

INSERT INTO Smeta(VidelenoDeneg, NomerDogovor\_smeta, StoimostMat, Zarplata, Bonus) VALUES(@svd, (SELECT NomerDogovor FROM RecDogo WHERE NazvDogo = @rdnd), @ssm, @sz, @sb)

drop procedure add\_smeta

CREATE PROCEDURE izm\_smeta @nomchek int, @vidden nvarchar(20), @nomdog\_smeta nvarchar(20), @stmat nvarchar(20), @zarp nvarchar(20), @bon nvarchar(20)

AS

UPDATE Smeta

SET VidelenoDeneg= @vidden, NomerDogovor\_smeta = (SELECT NomerDogovor FROM RecDogo WHERE NazvDogo = @nomdog\_smeta), StoimostMat = @stmat, Zarplata= @zarp, Bonus= @bon

WHERE NomerChek = @nomchek

EXEC izm\_smeta 5, 20000, 14, 3000, 15000, 30000

drop procedure izm\_smeta

CREATE PROCEDURE izm\_zakazchik @id\_zakazchika int, @nam\_zakazchik nvarchar(20), @adr\_zakazchik varchar(20), @phone\_zakazchik varchar(20)

AS

UPDATE RecZak

SET Name= @nam\_zakazchik, Adres = @adr\_zakazchik, phone = @phone\_zakazchik

WHERE ID\_zakazchika = @id\_zakazchika

EXEC izm\_zakazchik 21, 'alexo', 'lobnya', 89789999

drop procedure izm\_zakazchik

CREATE PROCEDURE izm\_zakaz @idzakaza int, @naimen varchar(20), @dudj nvarchar(20), @idzakazchikazakaz nvarchar(20)

AS

UPDATE Zakaz

SET Naimenovanie= @naimen, Budjet = @dudj, ID\_zakazchika\_Zakaz = (SELECT ID\_zakazchika FROM RecZak WHERE Name = @idzakazchikazakaz)

WHERE ID\_zakaza = @idzakaza

EXEC izm\_zakaz 17, 'system', 354000, 21

drop procedure izm\_zakaz

CREATE PROCEDURE izm\_dogovor @nomerDogovor int, @nazvdogo nvarchar(20), @gorod nvarchar(20), @idzakazchikadogo nvarchar(20), @idispolnitdogo varchar(20), @idzakazadogo varchar(20)

AS

UPDATE RecDogo

SET NazvDogo= @nazvdogo, Gorod = @gorod, ID\_Zakazchika\_dog = (SELECT ID\_zakazchika FROM RecZak WHERE Name = @idzakazchikadogo), ID\_Ispolnit\_dog= (SELECT ID\_Ispolnit FROM RecIspoln WHERE Imya = @idispolnitdogo), ID\_zakaza\_dog= (SELECT ID\_zakaza FROM Zakaz WHERE Naimenovanie = @idzakazadogo)

WHERE NomerDogovor = @nomerDogovor

EXEC izm\_dogovor 14, 'avtotochka', 'Spetersburg', 21, 15, 17

drop procedure izm\_dogovor

CREATE PROCEDURE izm\_ispolnitel @id\_ispolnitel int, @imya varchar(20), @adr\_isp nvarchar(20), @phone\_ispolnit nvarchar(20)

AS

UPDATE RecIspoln

SET Imya= @imya, AdresIsp = @adr\_isp, PhoneIspoln = @phone\_ispolnit

WHERE ID\_Ispolnit = @id\_ispolnitel

EXEC izm\_ispolnitel 15, 'motolavina', 'lobnya', 89789999

drop procedure izm\_ispolnitel

CREATE PROCEDURE delete\_smeta @deleteNomerCheck int

AS

DELETE FROM Smeta

WHERE NomerChek = @deleteNomerCheck

CREATE PROCEDURE delete\_zakazchik @deleteidzakazchik int

AS

DELETE FROM RecZak

WHERE ID\_zakazchika = @deleteidzakazchik

CREATE PROCEDURE delete\_zakaz @deleteidzakaza int

AS

DELETE FROM Zakaz

WHERE ID\_zakaza = @deleteidzakaza

CREATE PROCEDURE delete\_dogovor @deleteNomerDogovor int

AS

DELETE FROM RecDogo

WHERE NomerDogovor = @deleteNomerDogovor

CREATE PROCEDURE delete\_ispolnitel @deleteisp int

AS

DELETE FROM RecIspoln

WHERE ID\_Ispolnit = @deleteisp

1. Реализовать отдельную хранимую процедуру(функцию), состоящую из нескольких отдельных операций в виде единой транзакции, которая при определенных условиях может быть зафиксирована или откатана;

CREATE PROCEDURE add\_zarplata @rdnd nvarchar(20), @rdg nvarchar(20), @sum float(5), @bonus nvarchar(20), @vd nvarchar(20), @smsm nvarchar(20)

AS

BEGIN

BEGIN TRANSACTION

INSERT INTO Smeta(NomerDogovor\_smeta, zarplata, Bonus, VidelenoDeneg, StoimostMat) VALUES((SELECT NomerDogovor FROM RecDogo WHERE NazvDogo = @rdnd and Gorod = @rdg), @sum, @bonus, @vd, @smsm)

IF @sum > 0 and @sum < 100000

BEGIN

COMMIT TRANSACTION

END

ELSE

BEGIN

ROLLBACK TRANSACTION

END

END;

1. Реализовать курсор на обновления отдельных данных (вычисления значения полей выбранной таблицы);

CREATE PROCEDURE curzor

@id int,

@n varchar(20)

AS

DECLARE @id2 varchar(20)

DECLARE curz CURSOR FOR

SELECT ID\_zakaza FROM Zakaz

OPEN curz

FETCH NEXT FROM curz INTO @id2

WHILE @@fetch\_status = 0

BEGIN

IF @id = @id2

UPDATE Zakaz

SET Naimenovanie = @n

WHERE ID\_zakaza = @id;

fetch next from curz into @id2

END

CLOSE curz

DEALLOCATE curz

1. Реализовать собственную скалярную и векторную функции. Функции сохранить в базе данных;

--Возвращает введенный id--

CREATE FUNCTION SK\_zakazchik(@id INT)

RETURNS int

BEGIN

declare @id\_zak int

SELECT @id\_zak = ID\_zakazchika FROM RecZak WHERE ID\_zakazchika = @id

RETURN @id\_zak

END;

drop function SK\_zakazchik

SELECT dbo.SK\_zakazchik(24)

----------------------------------------------------

--ВЕКТОРНАЯ --

--Выводит данные заказчика по id

CREATE FUNCTION TB\_zakazchik(@id int)

RETURNS @zak1 TABLE (ID\_zakazchika INT, Name nvarchar(20), Adres varchar(20), phone varchar(20))

BEGIN

INSERT @zak1

SELECT ID\_zakazchika, Name, Adres, phone FROM RecZak WHERE ID\_zakazchika = @id

RETURN

END

select \* from RecZak

SELECT \* FROM dbo.TB\_zakazchik(22)

DROP FUNCTION TB\_zakazchik

1. Распределение прав пользователей: предусмотреть не менее двух пользователей с разным набором привилегий. Каждый набор привилегий оформить в виде роли.

CREATE LOGIN user1 WITH PASSWORD = 'user1'; CREATE USER user1 for login user1

GO

CREATE ROLE selecter; ALTER ROLE selecter ADD MEMBER user1;

GRANT SELECT ON RecZak

TO selecter;

GRANT SELECT ON Zakaz

TO selecter;

GRANT SELECT ON RecDogo

TO selecter;

GRANT SELECT ON RecIspoln

TO selecter;

GRANT SELECT ON Smeta

TO selecter;

----------------

CREATE LOGIN sanches WITH PASSWORD = '1'; CREATE USER sanches for login sanches

GO

CREATE ROLE creater; ALTER ROLE creater ADD MEMBER sanches;

GRANT SELECT ON RecZak

TO creater;

GRANT SELECT ON Zakaz

TO creater;

GRANT SELECT ON RecDogo

TO creater;

GRANT SELECT ON RecIspoln

TO creater;

GRANT SELECT ON Smeta

TO creater;

GRANT CREATE TABLE, CREATE PROCEDURE

TO creater

GRANT UPDATE ON RecZak (ID\_zakazchika , Name, Adres, phone)

TO creater;

**Листинг программы**

**main.cpp**

#include "mainwindow.h"

#include <QApplication>

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

{

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

MainWindow w;

w.show();

return a.exec();

}

**mainwindow.cpp**

#include "mainwindow.h"

#include "ui\_mainwindow.h"

MainWindow::MainWindow(QWidget \*parent)

: QMainWindow(*parent*)

, ui(new Ui::MainWindow)

{

ui->setupUi(this);

setWindowTitle("Курсовая");

}

MainWindow::~*MainWindow*()

{

delete ui;

}

void MainWindow::on\_actionconnect\_to\_DB\_triggered()

{

logwin=new login();

logwin->show();

}

void MainWindow::on\_action\_2\_triggered()

{

prosIspolnit = new prosmotrIspolnit();

prosIspolnit->show();

}

void MainWindow::on\_action\_3\_triggered()

{

prosZakaz = new prosmotrZakaz();

prosZakaz->show();

}

void MainWindow::on\_action\_triggered()

{

prosZakazchik =new ProsmotrZakazchika();

prosZakazchik->show();

}

void MainWindow::on\_action\_4\_triggered()

{

addDogo = new AddDogovor();

addDogo->show();

}

void MainWindow::on\_action\_7\_triggered()

{

prosDogo = new prosmotrdogovor();

prosDogo->show();

}

void MainWindow::on\_action\_8\_triggered() //просмотр заказов по сроку договора

{

prosDogov = new prosmotrdogovora();

prosDogov->show();

}

void MainWindow::on\_action\_9\_triggered()

{

addSme = new addSmeta();

addSme->show();

}

void MainWindow::on\_action\_10\_triggered()

{

addzak = new addzakazchik();

addzak->show();

}

void MainWindow::on\_action\_11\_triggered()

{

addisp = new addispolnitel();

addisp->show();

}

void MainWindow::on\_action\_12\_triggered()

{

addzaka = new addzakaz();

addzaka->show();

}

**login.cpp**

#include "login.h"

#include "ui\_login.h"

login::login(QWidget \*parent) :

QWidget(*parent*),

ui(new Ui::login)

{

ui->setupUi(this);

setWindowTitle("Подключение к БД");

ui->lineEdit->setText("DESKTOP-4R42CK9\\SQLEXPRESS");

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

ui->lineEdit\_3->setText("sanches");

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("Успешное соединение");

}

else

{

mes->setText("Соединение провалено");

}

mes->show();

}

**adddogovor.cpp**

#include "adddogovor.h"

#include "ui\_adddogovor.h"

AddDogovor::AddDogovor(QWidget \*parent) :

QWidget(*parent*),

ui(new Ui::AddDogovor)

{

ui->setupUi(this);

QSqlQuery\* query = new QSqlQuery();

query->exec("SELECT Name FROM RecZak");

while(query->next())

{

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

}

zakcombo = 0;

QSqlQuery\* query1 = new QSqlQuery();

query1->exec("SELECT Imya FROM RecIspoln");

while(query1->next())

{

ui->comboBox\_2->addItem(query1->value(0).toString());

}

ispcombo = 0;

QSqlQuery\* query2 = new QSqlQuery();

query2->exec("SELECT Naimenovanie FROM Zakaz");

while(query2->next())

{

ui->comboBox\_3->addItem(query2->value(0).toString());

}

zakazcombo = 0;

}

AddDogovor::~*AddDogovor*()

{

delete ui;

}

void AddDogovor::on\_pushButton\_clicked()

{

QSqlQuery\* addDogo = new QSqlQuery;

addDogo->prepare("EXEC add\_dogovor @rdnd = " + ui->lineEdit->text() + ", @rdg = " + ui->lineEdit\_2->text() + ", @zz = " + ui->comboBox->itemText(zakcombo) + ", @im = " + ui->comboBox\_2->itemText(ispcombo) + ", @dnz = " + ui->comboBox\_3->itemText(zakazcombo));

QMessageBox\* mess1 = new QMessageBox();

if (!addDogo->exec())

{

mess1->setText("Неправильно заполнен договор");

mess1->show();

}

}

void AddDogovor::on\_pushButton\_2\_clicked()

{

QSqlQuery\* addZakaz = new QSqlQuery;

addZakaz->prepare("EXEC add\_zakaz @dnz = " + ui->comboBox\_3->itemText(zakazcombo) + ", @dbz = " + ui->lineEdit\_9->text() + ", @zz = " + ui->comboBox->itemText(zakcombo));

QMessageBox\* mess = new QMessageBox();

if (!addZakaz->exec())

{

mess->setText("Неправильно заполнен заказ");

mess->show();

}

}

void AddDogovor::on\_pushButton\_3\_clicked()

{

QSqlQuery\* addIsp = new QSqlQuery;

addIsp->prepare("EXEC add\_ispolnit @im = " + ui->comboBox\_2->itemText(ispcombo) + ", @ai = " + ui->lineEdit\_8->text() + ", @ti = " + ui->lineEdit\_11->text());

QMessageBox\* mess2 = new QMessageBox();

if (!addIsp->exec())

{

mess2->setText("Неправильно заполнен исполнитель");

mess2->show();

}

}

void AddDogovor::on\_pushButton\_4\_clicked()

{

QSqlQuery\* addZakazchik = new QSqlQuery;

addZakazchik->prepare("EXEC add\_zakazchik @zz = " + ui->comboBox->itemText(zakcombo) + ", @az = " + ui->lineEdit\_4->text() + ", @tz = " + ui->lineEdit\_7->text());

QMessageBox\* mess3 = new QMessageBox();

if (!addZakazchik->exec())

{

mess3->setText("Неправильно заполнен заказчик");

mess3->show();

}

}

void AddDogovor::on\_pushButton\_13\_clicked()

{

model3 = new QSqlQueryModel();

model3 -> setQuery("SELECT \* FROM RecZak");

model3->*setHeaderData*(0, Qt::Horizontal, "ID\_Заказчика");

model3->*setHeaderData*(1, Qt::Horizontal, "Имя");

model3->*setHeaderData*(2, Qt::Horizontal, "Адрес");

model3->*setHeaderData*(3, Qt::Horizontal, "Телефон");

ui->tableView->*setModel*(*model3*);

ui->tableView->resizeColumnsToContents();

ui->tableView->show();

}

void AddDogovor::on\_pushButton\_14\_clicked()

{

model1 = new QSqlQueryModel();

model1 -> setQuery ("SELECT \* FROM RecIspoln");

model1->*setHeaderData*(0, Qt::Horizontal, "ID\_Исполнителя");

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

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

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

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

ui->tableView->resizeColumnsToContents();

ui->tableView->show();

}

void AddDogovor::on\_pushButton\_15\_clicked()

{

model2 = new QSqlQueryModel();

model2 -> setQuery("SELECT ID\_zakaza, Naimenovanie, Budjet, (SELECT Name FROM RecZak WHERE ID\_zakazchika\_Zakaz = ID\_zakazchika) FROM Zakaz");

model2->*setHeaderData*(0, Qt::Horizontal, "ID\_Заказа");

model2->*setHeaderData*(1, Qt::Horizontal, "Наименование");

model2->*setHeaderData*(2, Qt::Horizontal, "Бюджет");

model2->*setHeaderData*(3, Qt::Horizontal, "Имя заказчика");

ui->tableView->*setModel*(*model2*);

ui->tableView->resizeColumnsToContents();

ui->tableView->show();

}

void AddDogovor::on\_pushButton\_16\_clicked()

{

model5 = new QSqlQueryModel();

model5 -> setQuery("SELECT NomerDogovor, NazvDogo, Gorod, (SELECT Name FROM RecZak WHERE ID\_zakazchika\_dog = ID\_zakazchika), (SELECT Imya FROM RecIspoln WHERE ID\_Ispolnit\_dog = ID\_Ispolnit), (SELECT Naimenovanie FROM Zakaz WHERE ID\_zakaza\_dog = ID\_zakaza) FROM RecDogo");

model5->*setHeaderData*(0, Qt::Horizontal, "Номер договора");

model5->*setHeaderData*(1, Qt::Horizontal, "Название договора");

model5->*setHeaderData*(2, Qt::Horizontal, "Город подписания договора");

model5->*setHeaderData*(3, Qt::Horizontal, "Имя заказчика");

model5->*setHeaderData*(4, Qt::Horizontal, "Имя исполнителя");

model5->*setHeaderData*(5, Qt::Horizontal, "Наименование заказа");

ui->tableView->*setModel*(*model5*);

ui->tableView->resizeColumnsToContents();

ui->tableView->show();

}

void AddDogovor::on\_tableView\_clicked(const QModelIndex &index)

{

int temp\_nom\_dog;

temp\_nom\_dog = ui->tableView->model()->*data*(ui->tableView->model()->*index*(index.row(),0)).toInt();

ui->lineEdit\_14->setText(QString::number(temp\_nom\_dog));

QSqlQuery\*query4 = new QSqlQuery();

query4->prepare("SELECT NazvDogo, Gorod FROM RecDogo WHERE NomerDogovor=:NomerDogovor");

query4->bindValue(":NomerDogovor", temp\_nom\_dog);

if (query4->exec())

{

query4->next();

ui->lineEdit->setText(query4->value(0).toString());

ui->lineEdit\_2->setText(query4->value(1).toString());

}

}

void AddDogovor::on\_pushButton\_11\_clicked()

{

QSqlQuery\*query4 = new QSqlQuery();

query4->prepare("izm\_dogovor @nomerDogovor = " + ui->lineEdit\_14->text() + ", @nazvdogo = " + ui->lineEdit->text() +", @gorod = " + ui->lineEdit\_2->text() + ", @idzakazchikadogo = " + ui->comboBox->itemText(zakcombo) + ", @idispolnitdogo = " + ui->comboBox\_2->itemText(ispcombo) +", @idzakazadogo = " + ui->comboBox\_3->itemText(zakazcombo));

query4->exec();

on\_pushButton\_16\_clicked();

}

void AddDogovor::on\_pushButton\_12\_clicked()

{

QSqlQuery\*query4 = new QSqlQuery();

query4->prepare("EXEC delete\_dogovor @deleteNomerDogovor = " + ui->lineEdit\_14->text());

query4->exec();

on\_pushButton\_16\_clicked();

}

void AddDogovor::on\_comboBox\_currentIndexChanged(int index)

{

zakcombo = index;

}

void AddDogovor::on\_comboBox\_2\_currentIndexChanged(int index)

{

ispcombo = index;

}

void AddDogovor::on\_comboBox\_3\_currentIndexChanged(int index)

{

zakazcombo = index;

}

**addispolnitel.cpp**

#include "addispolnitel.h"

#include "ui\_addispolnitel.h"

addispolnitel::addispolnitel(QWidget \*parent) :

QWidget(*parent*),

ui(new Ui::addispolnitel)

{

ui->setupUi(this);

}

addispolnitel::~*addispolnitel*()

{

delete ui;

}

void addispolnitel::on\_pushButton\_clicked()

{

model1 = new QSqlQueryModel();

model1 -> setQuery ("SELECT \* FROM RecIspoln");

model1->*setHeaderData*(0, Qt::Horizontal, "ID\_Исполнителя");

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

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

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

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

ui->tableView->resizeColumnsToContents();

ui->tableView->show();

}

void addispolnitel::on\_pushButton\_2\_clicked()

{

QSqlQuery\* addIsp = new QSqlQuery;

addIsp->prepare("EXEC add\_ispolnit @im = " + ui->lineEdit\_2->text() + ", @ai = " + ui->lineEdit\_3->text() + ", @ti = " + ui->lineEdit\_4->text());

QMessageBox\* mess2 = new QMessageBox();

if (!addIsp->exec())

{

mess2->setText("Неправильно заполнен исполнитель");

mess2->show();

}

}

void addispolnitel::on\_tableView\_clicked(const QModelIndex &index)

{

int temp\_nom\_isp;

temp\_nom\_isp = ui->tableView->model()->*data*(ui->tableView->model()->*index*(index.row(),0)).toInt();

ui->lineEdit->setText(QString::number(temp\_nom\_isp));

QSqlQuery\*query2 = new QSqlQuery();

query2->prepare("SELECT Imya, AdresIsp, PhoneIspoln FROM RecIspoln WHERE ID\_Ispolnit=:ID\_Ispolnit");

query2->bindValue(":ID\_Ispolnit", temp\_nom\_isp);

if (query2->exec())

{

query2->next();

ui->lineEdit\_2->setText(query2->value(0).toString());

ui->lineEdit\_3->setText(query2->value(1).toString());

ui->lineEdit\_4->setText(query2->value(2).toString());

}

}

void addispolnitel::on\_pushButton\_3\_clicked()

{

QSqlQuery\*query2 = new QSqlQuery();

query2->prepare("EXEC izm\_ispolnitel @id\_ispolnitel = " + ui->lineEdit->text() + ", @imya = " + ui->lineEdit\_2->text() + ", @adr\_isp = " + ui->lineEdit\_3->text() + ", @phone\_ispolnit = " + ui->lineEdit\_4->text());

query2->exec();

on\_pushButton\_clicked();

}

void addispolnitel::on\_pushButton\_4\_clicked()

{

QSqlQuery\*query2 = new QSqlQuery();

query2->prepare("EXEC delete\_ispolnitel @deleteisp = " + ui->lineEdit->text());

query2->exec();

on\_pushButton\_clicked();

}

**addsmeta.cpp**

#include "addsmeta.h"

#include "ui\_addsmeta.h"

addSmeta::addSmeta(QWidget \*parent) :

QWidget(*parent*),

ui(new Ui::addSmeta)

{

ui->setupUi(this);

QSqlQuery\* query9 = new QSqlQuery();

query9->exec("SELECT NazvDogo FROM RecDogo");

while(query9->next())

{

ui->comboBox->addItem(query9->value(0).toString());

}

smetcombo = 0;

}

addSmeta::~*addSmeta*()

{

delete ui;

}

void addSmeta::on\_pushButton\_clicked()

{

QSqlQuery\* addSmeta = new QSqlQuery;

addSmeta->prepare("EXEC add\_smeta @svd = " + ui->lineEdit\_2->text() + ", @rdnd = " + ui->comboBox->itemText(smetcombo) + ", @ssm = " + ui->lineEdit\_3->text() + ", @sz = " + ui->lineEdit\_4->text() + ", @sb = " + ui->lineEdit\_5->text());

QMessageBox\* mess7 = new QMessageBox();

if (!addSmeta->exec())

{

mess7->setText("Неправильно заполнена смета");

mess7->show();

}

}

void addSmeta::on\_pushButton\_2\_clicked()

{

model7 = new QSqlQueryModel();

model7 -> setQuery("SELECT NomerChek, VidelenoDeneg, (SELECT NazvDogo FROM RecDogo WHERE NomerDogovor\_smeta = NomerDogovor), StoimostMat, Zarplata, Bonus FROM Smeta");

model7->*setHeaderData*(0, Qt::Horizontal, "Номер чека");

model7->*setHeaderData*(1, Qt::Horizontal, "Аванс");

model7->*setHeaderData*(2, Qt::Horizontal, "Название договора");

model7->*setHeaderData*(3, Qt::Horizontal, "Стоимость материалов");

model7->*setHeaderData*(4, Qt::Horizontal, "Зарплата");

model7->*setHeaderData*(5, Qt::Horizontal, "Бонус");

ui->tableView->*setModel*(*model7*);

ui->tableView->resizeColumnsToContents();

ui->tableView->show();

}

void addSmeta::on\_tableView\_clicked(const QModelIndex &index)

{

int temp\_nom;

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

ui->lineEdit\_6->setText(QString::number(temp\_nom));

QSqlQuery\*query = new QSqlQuery();

query->prepare("SELECT VidelenoDeneg, StoimostMat, Zarplata, Bonus FROM Smeta WHERE NomerChek=:NomerChek");

query->bindValue(":NomerChek", temp\_nom);

if (query->exec())

{

query->next();

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

//ui->lineEdit\_7->setText(query->value(1).toString());, NomerDogovor\_smeta

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

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

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

//ui->lineEdit->setText(query->value(4).toString()); , (SELECT NazvDogo FROM RecDogo WHERE NomerDogovor\_smeta = NomerDogovor)

}

}

void addSmeta::on\_pushButton\_3\_clicked()

{

QSqlQuery\*query = new QSqlQuery();

query->prepare("EXEC izm\_smeta @nomchek = " + ui->lineEdit\_6->text() + ", @vidden = " + ui->lineEdit\_2->text() + ", @nomdog\_smeta = " + ui->comboBox->itemText(smetcombo) + ", @stmat = " + ui->lineEdit\_3->text() + ", @zarp = " + ui->lineEdit\_4->text() + ", @bon = " + ui->lineEdit\_5->text());

//ui->lineEdit\_6->text()

query->exec();

on\_pushButton\_2\_clicked();

}

void addSmeta::on\_pushButton\_4\_clicked()

{

QSqlQuery\*query = new QSqlQuery();

query->prepare("EXEC delete\_smeta @deleteNomerCheck = "+ ui->lineEdit\_6->text());

query->exec();

on\_pushButton\_2\_clicked();

}

void addSmeta::on\_comboBox\_currentIndexChanged(int index)

{

smetcombo = index;

}

**addzakaz.cpp**

#include "addzakaz.h"

#include "ui\_addzakaz.h"

addzakaz::addzakaz(QWidget \*parent) :

QWidget(*parent*),

ui(new Ui::addzakaz)

{

ui->setupUi(this);

QSqlQuery\* query = new QSqlQuery();

query->exec("SELECT Name FROM RecZak");

while(query->next())

{

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

}

zakcombo = 0;

}

addzakaz::~*addzakaz*()

{

delete ui;

}

void addzakaz::on\_pushButton\_clicked()

{

model2 = new QSqlQueryModel();

model2 -> setQuery("SELECT ID\_zakaza, Naimenovanie, Budjet, (SELECT Name FROM RecZak WHERE ID\_zakazchika\_Zakaz = ID\_zakazchika) FROM Zakaz");

model2->*setHeaderData*(0, Qt::Horizontal, "ID\_Заказа");

model2->*setHeaderData*(1, Qt::Horizontal, "Наименование");

model2->*setHeaderData*(2, Qt::Horizontal, "Бюджет");

model2->*setHeaderData*(3, Qt::Horizontal, "Имя заказчика");

ui->tableView->*setModel*(*model2*);

ui->tableView->resizeColumnsToContents();

ui->tableView->show();

}

void addzakaz::on\_pushButton\_2\_clicked()

{

QSqlQuery\* addZakaz = new QSqlQuery;

addZakaz->prepare("EXEC add\_zakaz @dnz = " + ui->lineEdit\_2->text() + ", @dbz = " + ui->lineEdit\_3->text() + ", @zz = " + ui->comboBox->itemText(zakcombo));

QMessageBox\* mess = new QMessageBox();

if (!addZakaz->exec())

{

mess->setText("Неправильно заполнен заказ");

mess->show();

}

}

void addzakaz::on\_tableView\_clicked(const QModelIndex &index)

{

int temp\_nom\_zakaz;

temp\_nom\_zakaz = ui->tableView->model()->*data*(ui->tableView->model()->*index*(index.row(),0)).toInt();

ui->lineEdit->setText(QString::number(temp\_nom\_zakaz));

QSqlQuery\*query3 = new QSqlQuery();

query3->prepare("SELECT Naimenovanie, Budjet FROM Zakaz WHERE ID\_zakaza=:ID\_zakaza");

query3->bindValue(":ID\_zakaza", temp\_nom\_zakaz);

if (query3->exec())

{

query3->next();

ui->lineEdit\_2->setText(query3->value(0).toString());

ui->lineEdit\_3->setText(query3->value(1).toString());

//ui->lineEdit\_4->setText(query3->value(2).toString()); , (SELECT Name FROM RecZak WHERE ID\_zakazchika\_Zakaz = ID\_zakazchika)

}

}

void addzakaz::on\_pushButton\_3\_clicked()

{

QSqlQuery\*query3 = new QSqlQuery();

query3->prepare("EXEC izm\_zakaz @idzakaza = " + ui->lineEdit->text() + ", @naimen = " + ui->lineEdit\_2->text() + ", @dudj = " + ui->lineEdit\_3->text() + ", @idzakazchikazakaz = " + ui->comboBox->itemText(zakcombo));

query3->exec();

on\_pushButton\_clicked();

}

void addzakaz::on\_pushButton\_4\_clicked()

{

QSqlQuery\*query3 = new QSqlQuery();

query3->prepare("EXEC delete\_zakaz @deleteidzakaza = " + ui->lineEdit->text());

query3->exec();

on\_pushButton\_clicked();

}

void addzakaz::on\_comboBox\_currentIndexChanged(int index)

{

zakcombo = index;

}

**addzakazchik.cpp**

#include "addzakazchik.h"

#include "ui\_addzakazchik.h"

addzakazchik::addzakazchik(QWidget \*parent) :

QWidget(*parent*),

ui(new Ui::addzakazchik)

{

ui->setupUi(this);

}

addzakazchik::~*addzakazchik*()

{

delete ui;

}

void addzakazchik::on\_pushButton\_clicked()

{

model3 = new QSqlQueryModel();

model3 -> setQuery("SELECT \* FROM RecZak");

model3->*setHeaderData*(0, Qt::Horizontal, "ID\_Заказчика");

model3->*setHeaderData*(1, Qt::Horizontal, "Имя");

model3->*setHeaderData*(2, Qt::Horizontal, "Адрес");

model3->*setHeaderData*(3, Qt::Horizontal, "Телефон");

ui->tableView->*setModel*(*model3*);

ui->tableView->resizeColumnsToContents();

ui->tableView->show();

}

void addzakazchik::on\_pushButton\_2\_clicked()

{

QSqlQuery\* addZakazchik = new QSqlQuery;

addZakazchik->prepare("EXEC add\_zakazchik @zz = " + ui->lineEdit\_2->text() + ", @az = " + ui->lineEdit\_3->text() + ", @tz = " + ui->lineEdit\_4->text());

QMessageBox\* mess3 = new QMessageBox();

if (!addZakazchik->exec())

{

mess3->setText("Неправильно заполнен заказчик");

mess3->show();

}

}

void addzakazchik::on\_tableView\_clicked(const QModelIndex &index)

{

int temp\_nom\_zakazchik;

temp\_nom\_zakazchik = ui->tableView->model()->*data*(ui->tableView->model()->*index*(index.row(),0)).toInt();

ui->lineEdit->setText(QString::number(temp\_nom\_zakazchik));

QSqlQuery\*query1 = new QSqlQuery();

query1->prepare("SELECT Name, Adres, phone FROM RecZak WHERE ID\_zakazchika=:ID\_zakazchika");

query1->bindValue(":ID\_zakazchika", temp\_nom\_zakazchik);

if (query1->exec())

{

query1->next();

ui->lineEdit\_2->setText(query1->value(0).toString());

ui->lineEdit\_3->setText(query1->value(1).toString());

ui->lineEdit\_4->setText(query1->value(2).toString());

}

}

void addzakazchik::on\_pushButton\_3\_clicked()

{

QSqlQuery\*query1 = new QSqlQuery();

query1->prepare("EXEC izm\_zakazchik @id\_zakazchika = " + ui->lineEdit->text() + ", @nam\_zakazchik = " + ui->lineEdit\_2->text() + ", @adr\_zakazchik = " + ui->lineEdit\_3->text() +", @phone\_zakazchik = " + ui->lineEdit\_4->text());

query1->exec();

on\_pushButton\_clicked();

}

void addzakazchik::on\_pushButton\_4\_clicked()

{

QSqlQuery\*query1 = new QSqlQuery();

query1->prepare("EXEC delete\_zakazchik @deleteidzakazchik = " + ui->lineEdit->text());

query1->exec();

on\_pushButton\_clicked();

}

**prosmotrdogovor.cpp**

#include "prosmotrdogovor.h"

#include "ui\_prosmotrdogovor.h"

prosmotrdogovor::prosmotrdogovor(QWidget \*parent) :

QWidget(*parent*),

ui(new Ui::prosmotrdogovor)

{

ui->setupUi(this);

setWindowTitle("Договоры");

}

prosmotrdogovor::~*prosmotrdogovor*()

{

delete ui;

}

void prosmotrdogovor::on\_pushButton\_clicked() //вывод всей таблицы

{

model5 = new QSqlQueryModel();

model5 -> setQuery("SELECT NomerDogovor, NazvDogo, Gorod, (SELECT Name FROM RecZak WHERE ID\_zakazchika\_dog = ID\_zakazchika), (SELECT Imya FROM RecIspoln WHERE ID\_Ispolnit\_dog = ID\_Ispolnit), (SELECT Naimenovanie FROM Zakaz WHERE ID\_zakaza\_dog = ID\_zakaza) FROM RecDogo");

model5->*setHeaderData*(0, Qt::Horizontal, "Номер договора");

model5->*setHeaderData*(1, Qt::Horizontal, "Название договора");

model5->*setHeaderData*(2, Qt::Horizontal, "Город подписания договора");

model5->*setHeaderData*(3, Qt::Horizontal, "Имя заказчика");

model5->*setHeaderData*(4, Qt::Horizontal, "Имя исполнителя");

model5->*setHeaderData*(5, Qt::Horizontal, "Наименование заказа");

ui->tableView->*setModel*(*model5*);

ui->tableView->resizeColumnsToContents();

ui->tableView->show();

}

void prosmotrdogovor::on\_pushButton\_2\_clicked() //вывод где бонус больше вводимой суммы

{

model51 = new QSqlQueryModel();

model51 -> setQuery("EXEC where\_bonus\_more @d = " + ui->lineEdit->text());

model51->*setHeaderData*(0, Qt::Horizontal, "Номер договора");

model51->*setHeaderData*(1, Qt::Horizontal, "Имя заказчика");

model51->*setHeaderData*(2, Qt::Horizontal, "Бонус");

ui->tableView->*setModel*(*model51*);

ui->tableView->resizeColumnsToContents();

ui->tableView->show();

}

void prosmotrdogovor::on\_pushButton\_3\_clicked() //вывод договора с одинак авансом и стоим.матер. выше вводим суммы

{

model52 = new QSqlQueryModel();

model52 -> setQuery("EXEC dogovori\_s\_odinak\_avansom @o = " + ui->lineEdit\_2->text() + ", @y = " + ui->lineEdit\_3->text());

model52->*setHeaderData*(0, Qt::Horizontal, "Номер договора");

model52->*setHeaderData*(1, Qt::Horizontal, "Наименование заказа");

ui->tableView->*setModel*(*model52*);

ui->tableView->resizeColumnsToContents();

ui->tableView->show();

}

void prosmotrdogovor::on\_pushButton\_4\_clicked()

{

model53 = new QSqlQueryModel();

model53 -> setQuery("EXEC get\_kolichestvo\_dogovorov");

model53->*setHeaderData*(0, Qt::Horizontal, "Количество договоров");

model53->*setHeaderData*(1, Qt::Horizontal, "Количество заказов");

ui->tableView->*setModel*(*model53*);

ui->tableView->resizeColumnsToContents();

ui->tableView->show();}

**prosmotrzakaz.cpp**

#include "prosmotrzakaz.h"

#include "ui\_prosmotrzakaz.h"

prosmotrZakaz::prosmotrZakaz(QWidget \*parent) :

QWidget(*parent*),

ui(new Ui::prosmotrZakaz)

{

ui->setupUi(this);

setWindowTitle("Заказы");

}

prosmotrZakaz::~*prosmotrZakaz*()

{

delete ui;

}

void prosmotrZakaz::on\_pushButton\_clicked()

{

model2 = new QSqlQueryModel();

model2 -> setQuery("SELECT ID\_zakaza, Naimenovanie, Budjet, (SELECT Name FROM RecZak WHERE ID\_zakazchika\_Zakaz = ID\_zakazchika) FROM Zakaz");

model2->*setHeaderData*(0, Qt::Horizontal, "ID заказа");

model2->*setHeaderData*(1, Qt::Horizontal, "Наименование");

model2->*setHeaderData*(2, Qt::Horizontal, "Бюджет");

model2->*setHeaderData*(3, Qt::Horizontal, "Имя заказчика");

ui->tableView->*setModel*(*model2*);

ui->tableView->resizeColumnsToContents();

ui->tableView->show();

}

void prosmotrZakaz::on\_pushButton\_2\_clicked()

{

model22 = new QSqlQueryModel();

model22 -> setQuery("EXEC get\_zakaz\_by\_namezakazchik @n = " + ui->lineEdit->text());

model22->*setHeaderData*(0, Qt::Horizontal, "Имя заказчика");

model22->*setHeaderData*(1, Qt::Horizontal, "Номер телефона");

model22->*setHeaderData*(2, Qt::Horizontal, "Наименование");

model22->*setHeaderData*(3, Qt::Horizontal, "Бюджет");

model22->*setHeaderData*(4, Qt::Horizontal, "Статус");

ui->tableView->*setModel*(*model22*);

ui->tableView->resizeColumnsToContents();

ui->tableView->show();

}

void prosmotrZakaz::on\_pushButton\_3\_clicked()

{

model23 = new QSqlQueryModel();

model23 -> setQuery("EXEC get\_zakaz\_by\_naimenovanie @e = " + ui->lineEdit\_2->text());

model23->*setHeaderData*(0, Qt::Horizontal, "Наименование");

model23->*setHeaderData*(1, Qt::Horizontal, "Бюджет");

ui->tableView->*setModel*(*model23*);

ui->tableView->resizeColumnsToContents();

ui->tableView->show();

}

**prosmotrzakazchika.cpp**

#include "prosmotrzakazchika.h"

#include "ui\_prosmotrzakazchika.h"

ProsmotrZakazchika::ProsmotrZakazchika(QWidget \*parent) :

QWidget(*parent*),

ui(new Ui::ProsmotrZakazchika)

{

ui->setupUi(this);

setWindowTitle("Заказчики");

}

ProsmotrZakazchika::~*ProsmotrZakazchika*()

{

delete ui;

}

void ProsmotrZakazchika::on\_pushButton\_clicked()

{

model3 = new QSqlQueryModel();

model3 -> setQuery("SELECT \* FROM RecZak");

model3->*setHeaderData*(0, Qt::Horizontal, "ID\_Заказчика");

model3->*setHeaderData*(1, Qt::Horizontal, "Имя");

model3->*setHeaderData*(2, Qt::Horizontal, "Адрес");

model3->*setHeaderData*(3, Qt::Horizontal, "Телефон");

ui->tableView->*setModel*(*model3*);

ui->tableView->resizeColumnsToContents();

ui->tableView->show();

}

void ProsmotrZakazchika::on\_pushButton\_2\_clicked()

{

model31 = new QSqlQueryModel();

model31 -> setQuery("EXEC zakazov\_bolsche\_odnogo @p = " + ui->lineEdit->text());

model31->*setHeaderData*(0, Qt::Horizontal, "Имя");

model31->*setHeaderData*(1, Qt::Horizontal, "Адрес");

model31->*setHeaderData*(2, Qt::Horizontal, "Телефон");

ui->tableView->*setModel*(*model31*);

ui->tableView->resizeColumnsToContents();

ui->tableView->show();

}

void ProsmotrZakazchika::on\_pushButton\_3\_clicked()

{

model32 = new QSqlQueryModel();

model32 -> setQuery("EXEC get\_zakazchik\_by\_namezakazchik @gzn = " + ui->lineEdit\_2->text());

model32->*setHeaderData*(0, Qt::Horizontal, "Имя");

model32->*setHeaderData*(1, Qt::Horizontal, "Адрес");

model32->*setHeaderData*(2, Qt::Horizontal, "Телефон");

ui->tableView->*setModel*(*model32*);

ui->tableView->resizeColumnsToContents();

ui->tableView->show();

}

**adddogovor.h**

#ifndef ADDDOGOVOR\_H

#define ADDDOGOVOR\_H

#include <QWidget>

#include <QSqlQuery>

#include <QMessageBox>

#include <QSqlQueryModel>

namespace Ui {

class AddDogovor;

}

class AddDogovor : public QWidget

{

Q\_OBJECT

public:

explicit AddDogovor(QWidget \*parent = nullptr);

~*AddDogovor*();

int zakcombo;

int ispcombo;

int zakazcombo;

private slots:

void on\_pushButton\_clicked();

void on\_pushButton\_2\_clicked();

void on\_pushButton\_3\_clicked();

void on\_pushButton\_4\_clicked();

void on\_pushButton\_13\_clicked();

void on\_pushButton\_14\_clicked();

void on\_pushButton\_15\_clicked();

void on\_pushButton\_16\_clicked();

void on\_tableView\_clicked(const QModelIndex &index);

void on\_pushButton\_11\_clicked();

void on\_pushButton\_12\_clicked();

void on\_comboBox\_currentIndexChanged(int index);

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

void on\_comboBox\_3\_currentIndexChanged(int index);

private:

Ui::AddDogovor \*ui;

QSqlDatabase db;

QSqlQueryModel\*model3;

QSqlQueryModel\*model1;

QSqlQueryModel\*model2;

QSqlQueryModel\*model5;

};

#endif // ADDDOGOVOR\_H

**addispolnit.h**

#ifndef ADDISPOLNITEL\_H

#define ADDISPOLNITEL\_H

#include <QWidget>

#include <QSqlQuery>

#include <QMessageBox>

#include <QSqlQueryModel>

#include <QSqlTableModel>

namespace Ui {

class addispolnitel;

}

class addispolnitel : public QWidget

{

Q\_OBJECT

public:

explicit addispolnitel(QWidget \*parent = nullptr);

~*addispolnitel*();

private slots:

void on\_pushButton\_clicked();

void on\_pushButton\_2\_clicked();

void on\_tableView\_clicked(const QModelIndex &index);

void on\_pushButton\_3\_clicked();

void on\_pushButton\_4\_clicked();

private:

Ui::addispolnitel \*ui;

QSqlQueryModel\*model1;

QSqlDatabase db;

};

#endif // ADDISPOLNITEL\_H

**addsmeta.h**

#ifndef ADDSMETA\_H

#define ADDSMETA\_H

#include <QWidget>

#include <QSqlQuery>

#include <QMessageBox>

#include <QSqlTableModel>

namespace Ui {

class addSmeta;

}

class addSmeta : public QWidget

{

Q\_OBJECT

public:

explicit addSmeta(QWidget \*parent = nullptr);

~*addSmeta*();

int smetcombo;

private slots:

void on\_pushButton\_clicked();

void on\_pushButton\_2\_clicked();

void on\_tableView\_clicked(const QModelIndex &index);

void on\_pushButton\_3\_clicked();

void on\_pushButton\_4\_clicked();

void on\_comboBox\_currentIndexChanged(int index);

private:

Ui::addSmeta \*ui;

QSqlDatabase db;

QSqlQueryModel\*model7;

};

#endif // ADDSMETA\_H

**addzakaz.h**

#ifndef ADDZAKAZ\_H

#define ADDZAKAZ\_H

#include <QWidget>

#include <QSqlQuery>

#include <QMessageBox>

#include <QSqlQueryModel>

#include <QSqlTableModel>

namespace Ui {

class addzakaz;

}

class addzakaz : public QWidget

{

Q\_OBJECT

public:

explicit addzakaz(QWidget \*parent = nullptr);

~*addzakaz*();

int zakcombo;

private slots:

void on\_pushButton\_clicked();

void on\_pushButton\_2\_clicked();

void on\_tableView\_clicked(const QModelIndex &index);

void on\_pushButton\_3\_clicked();

void on\_pushButton\_4\_clicked();

void on\_comboBox\_currentIndexChanged(int index);

private:

Ui::addzakaz \*ui;

QSqlQueryModel\*model2;

QSqlDatabase db;

};

#endif // ADDZAKAZ\_H

**addzakazchik.h**

#ifndef ADDZAKAZCHIK\_H

#define ADDZAKAZCHIK\_H

#include <QWidget>

#include <QSqlQuery>

#include <QMessageBox>

#include <QSqlQueryModel>

#include <QSqlTableModel>

namespace Ui {

class addzakazchik;

}

class addzakazchik : public QWidget

{

Q\_OBJECT

public:

explicit addzakazchik(QWidget \*parent = nullptr);

~*addzakazchik*();

private slots:

void on\_pushButton\_clicked();

void on\_pushButton\_2\_clicked();

void on\_tableView\_clicked(const QModelIndex &index);

void on\_pushButton\_3\_clicked();

void on\_pushButton\_4\_clicked();

private:

Ui::addzakazchik \*ui;

QSqlQueryModel\*model3;

QSqlDatabase db;

};

#endif // ADDZAKAZCHIK\_H

**login.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

**mainwindow.h**

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include "login.h"

#include "ui\_login.h"

#include "prosmotrzakazchika.h"

#include "ui\_mainwindow.h"

#include "prosmotrispolnit.h"

#include "ui\_prosmotrispolnit.h"

#include "prosmotrzakaz.h"

#include "ui\_prosmotrzakaz.h"

#include "adddogovor.h"

#include "ui\_adddogovor.h"

#include "prosmotrdogovor.h"

#include "ui\_prosmotrdogovor.h"

#include "prosmotrdogovora.h"

#include "ui\_prosmotrdogovora.h"

#include "addsmeta.h"

#include "ui\_addsmeta.h"

#include "addzakazchik.h"

#include "ui\_addzakazchik.h"

#include "addispolnitel.h"

#include "ui\_addispolnitel.h"

#include "addzakaz.h"

#include "ui\_addzakaz.h"

#include <QSqlTableModel>

#include <QSqlQueryModel>

QT\_BEGIN\_NAMESPACE

namespace Ui { class MainWindow; }

QT\_END\_NAMESPACE

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

MainWindow(QWidget \*parent = nullptr);

~*MainWindow*();

login\*logwin;

ProsmotrZakazchika \*prosZakazchik;

prosmotrIspolnit\*prosIspolnit;

prosmotrZakaz\*prosZakaz;

AddDogovor\*addDogo;

prosmotrdogovor\*prosDogo;

prosmotrdogovora\*prosDogov;

AddDogovor\*addZakaz;

addSmeta\*addSme;

addzakazchik\*addzak;

addispolnitel\*addisp;

addzakaz\*addzaka;

private slots:

void on\_actionconnect\_to\_DB\_triggered();

//void on\_tableView\_activated(const QModelIndex &index);

//void on\_pushButton\_clicked();

//void on\_action\_6\_triggered();

void on\_action\_2\_triggered();

void on\_action\_3\_triggered();

void on\_action\_triggered();

void on\_action\_4\_triggered();

void on\_action\_7\_triggered();

void on\_action\_8\_triggered(); //доп просмотр договора(поменять или удалить)

void on\_action\_9\_triggered();

void on\_action\_10\_triggered();

void on\_action\_11\_triggered();

void on\_action\_12\_triggered();

private:

Ui::MainWindow \*ui;

QSqlTableModel\*model;

};

#endif // MAINWINDOW\_H

**prosmotrdogovor.h**

#ifndef PROSMOTRDOGOVOR\_H

#define PROSMOTRDOGOVOR\_H

#include <QWidget>

#include <QtSql>

namespace Ui {

class prosmotrdogovor;

}

class prosmotrdogovor : public QWidget

{

Q\_OBJECT

public:

explicit prosmotrdogovor(QWidget \*parent = nullptr);

~*prosmotrdogovor*();

private slots:

void on\_pushButton\_clicked();

void on\_pushButton\_2\_clicked();

void on\_pushButton\_3\_clicked();

void on\_pushButton\_4\_clicked();

private:

Ui::prosmotrdogovor \*ui;

QSqlDatabase db;

QSqlQueryModel\*model5;

QSqlQueryModel\*model51;

QSqlQueryModel\*model52;

QSqlQueryModel\*model53;

};

#endif // PROSMOTRDOGOVOR\_H

**prosmotrdogovora.h**

#ifndef PROSMOTRDOGOVORA\_H

#define PROSMOTRDOGOVORA\_H

#include <QWidget>

#include <QtSql>

namespace Ui {

class prosmotrdogovora;

}

class prosmotrdogovora : public QWidget

{

Q\_OBJECT

public:

explicit prosmotrdogovora(QWidget \*parent = nullptr);

~*prosmotrdogovora*();

private slots:

void on\_pushButton\_clicked();

private:

Ui::prosmotrdogovora \*ui;

QSqlDatabase db;

QSqlQueryModel\*model33;

};

#endif // PROSMOTRDOGOVORA\_H

**prosmotrispolnit.h**

#ifndef PROSMOTRISPOLNIT\_H

#define PROSMOTRISPOLNIT\_H

#include <QWidget>

#include <QtSql>

namespace Ui {

class prosmotrIspolnit;

}

class prosmotrIspolnit : public QWidget

{

Q\_OBJECT

public:

explicit prosmotrIspolnit(QWidget \*parent = nullptr);

~*prosmotrIspolnit*();

private slots:

void on\_pushButton\_clicked();

private:

Ui::prosmotrIspolnit \*ui;

QSqlDatabase db;

QSqlQueryModel\*model1;

};

#endif // PROSMOTRISPOLNIT\_H

**prosmotrzakaz.h**

#ifndef PROSMOTRZAKAZ\_H

#define PROSMOTRZAKAZ\_H

#include <QWidget>

#include <QtSql>

namespace Ui {

class prosmotrZakaz;

}

class prosmotrZakaz : public QWidget

{

Q\_OBJECT

public:

explicit prosmotrZakaz(QWidget \*parent = nullptr);

~*prosmotrZakaz*();

private slots:

void on\_pushButton\_clicked();

void on\_pushButton\_2\_clicked();

void on\_pushButton\_3\_clicked();

private:

Ui::prosmotrZakaz \*ui;

QSqlDatabase db;

QSqlQueryModel\*model2;

QSqlQueryModel\*model22;

QSqlQueryModel\*model23;

};

#endif // PROSMOTRZAKAZ\_H

**prosmotrzakazchika.h**

#ifndef PROSMOTRZAKAZCHIKA\_H

#define PROSMOTRZAKAZCHIKA\_H

#include <QWidget>

#include <QtSql>

namespace Ui {

class ProsmotrZakazchika;

}

class ProsmotrZakazchika : public QWidget

{

Q\_OBJECT

public:

explicit ProsmotrZakazchika(QWidget \*parent = nullptr);

~*ProsmotrZakazchika*();

private slots:

void on\_pushButton\_clicked();

void on\_pushButton\_2\_clicked();

void on\_pushButton\_3\_clicked();

private:

Ui::ProsmotrZakazchika \*ui;

QSqlDatabase db;

QSqlQueryModel\*model3;

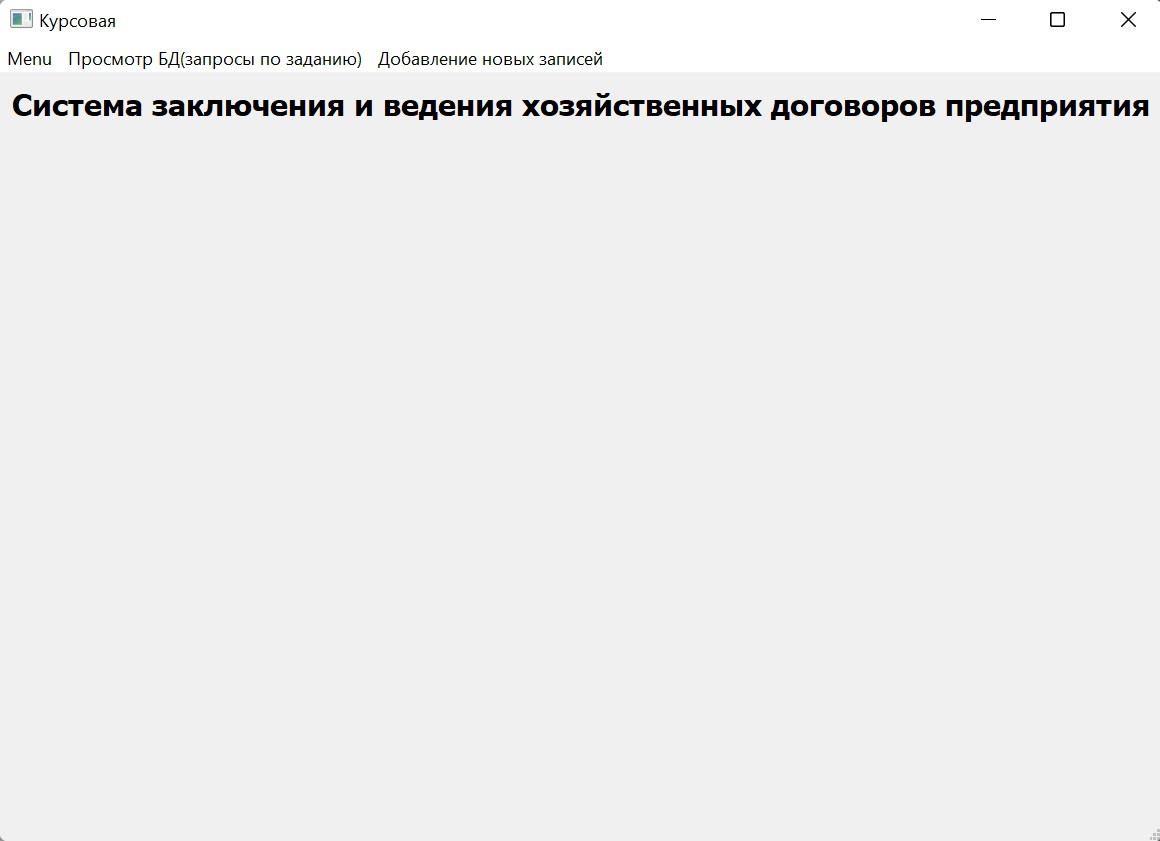
QSqlQueryModel\*model31;

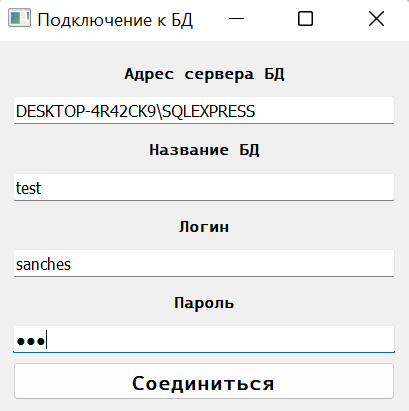
QSqlQueryModel\*model32;

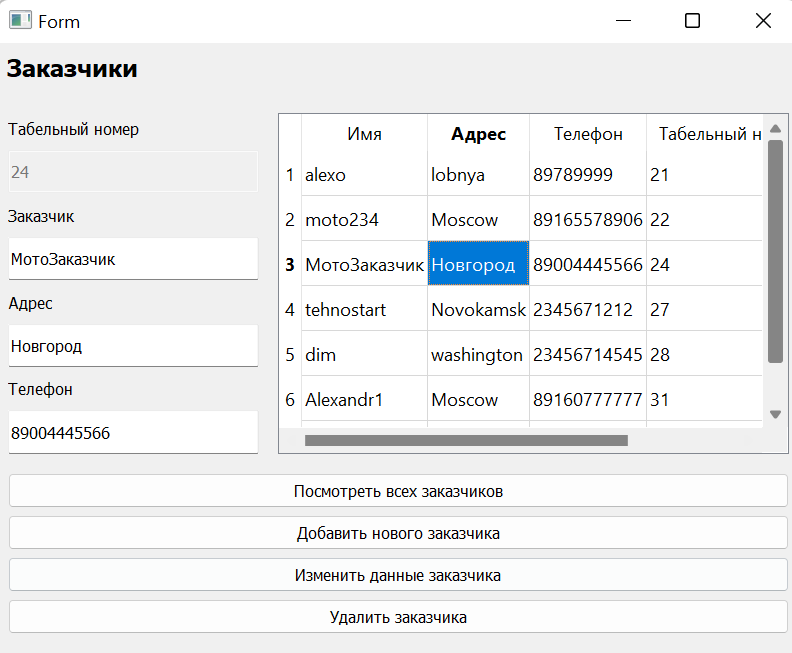
};

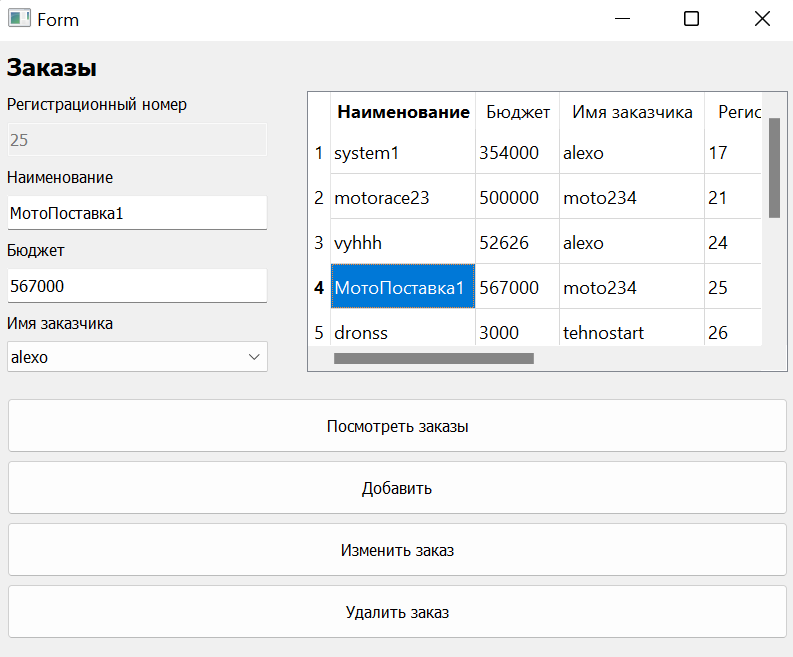
#endif // PROSMOTRZAKAZCHIKA\_H

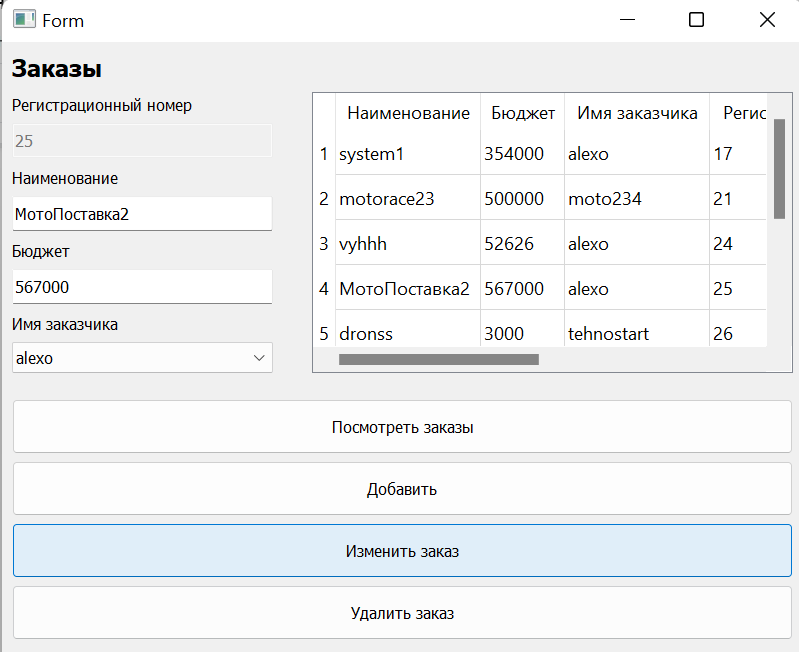
**Демонстрация работы приложения**

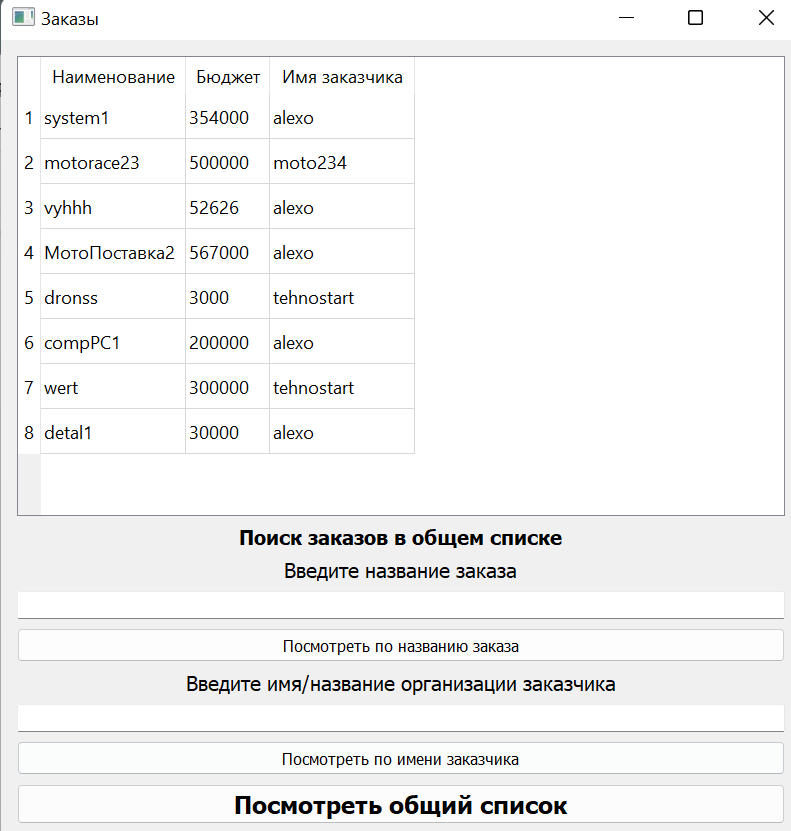


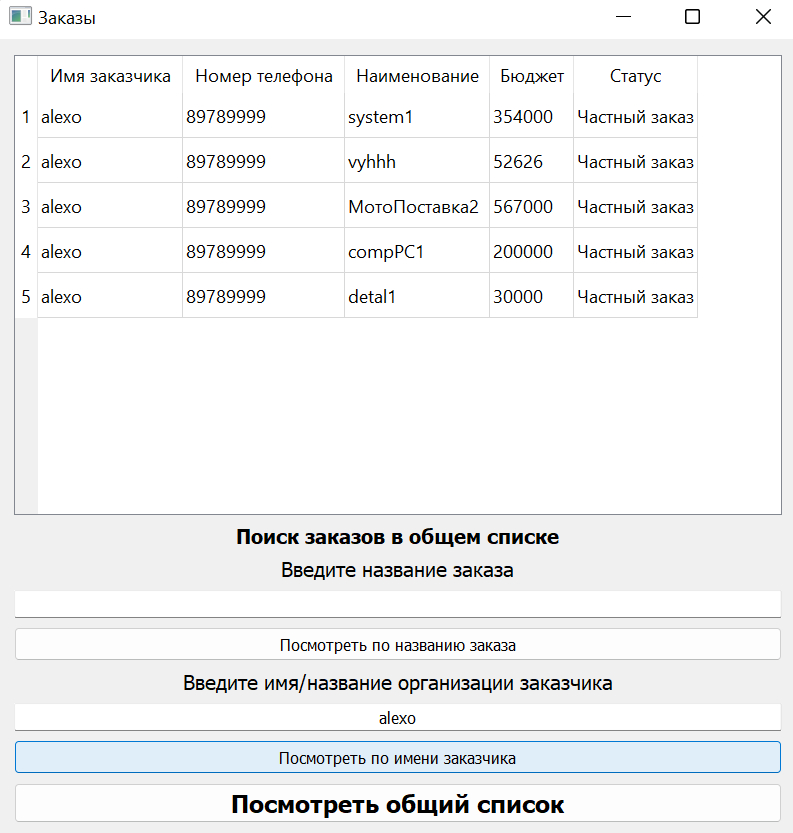


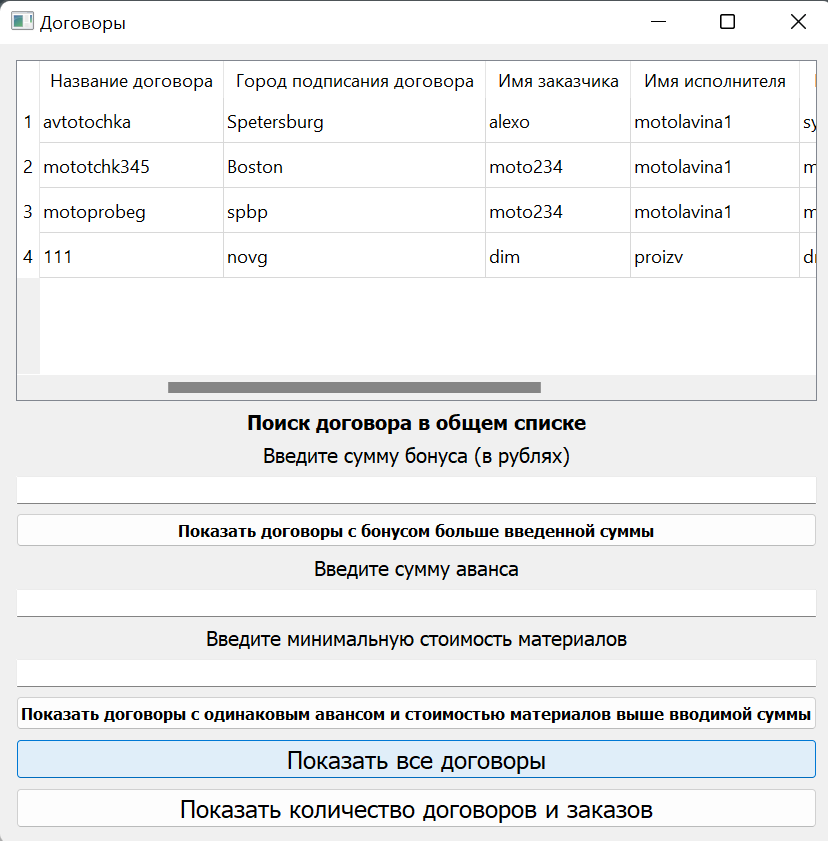


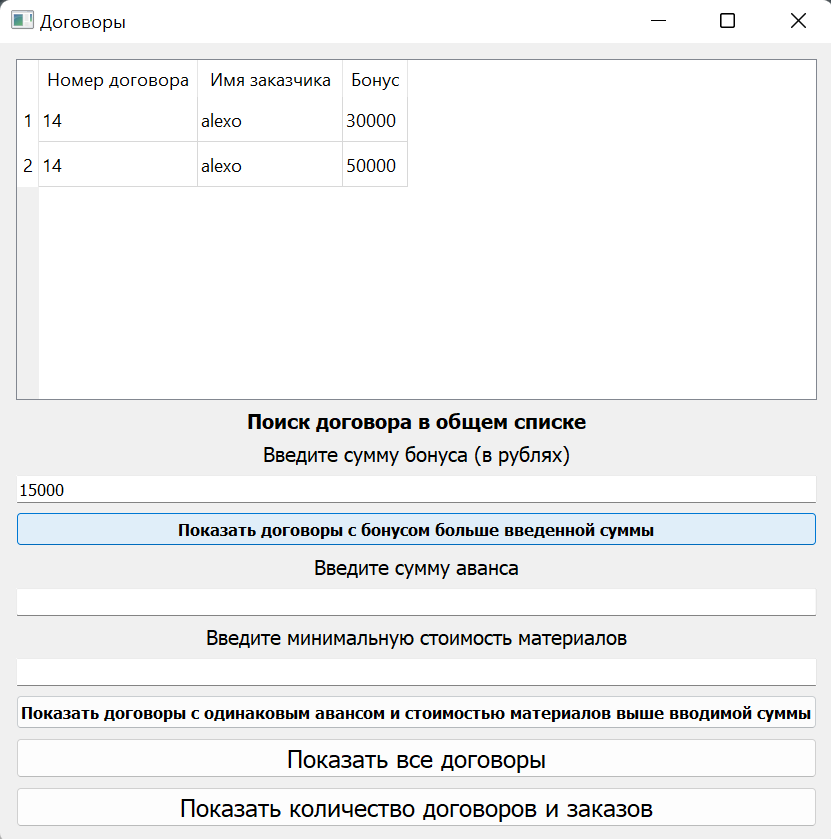


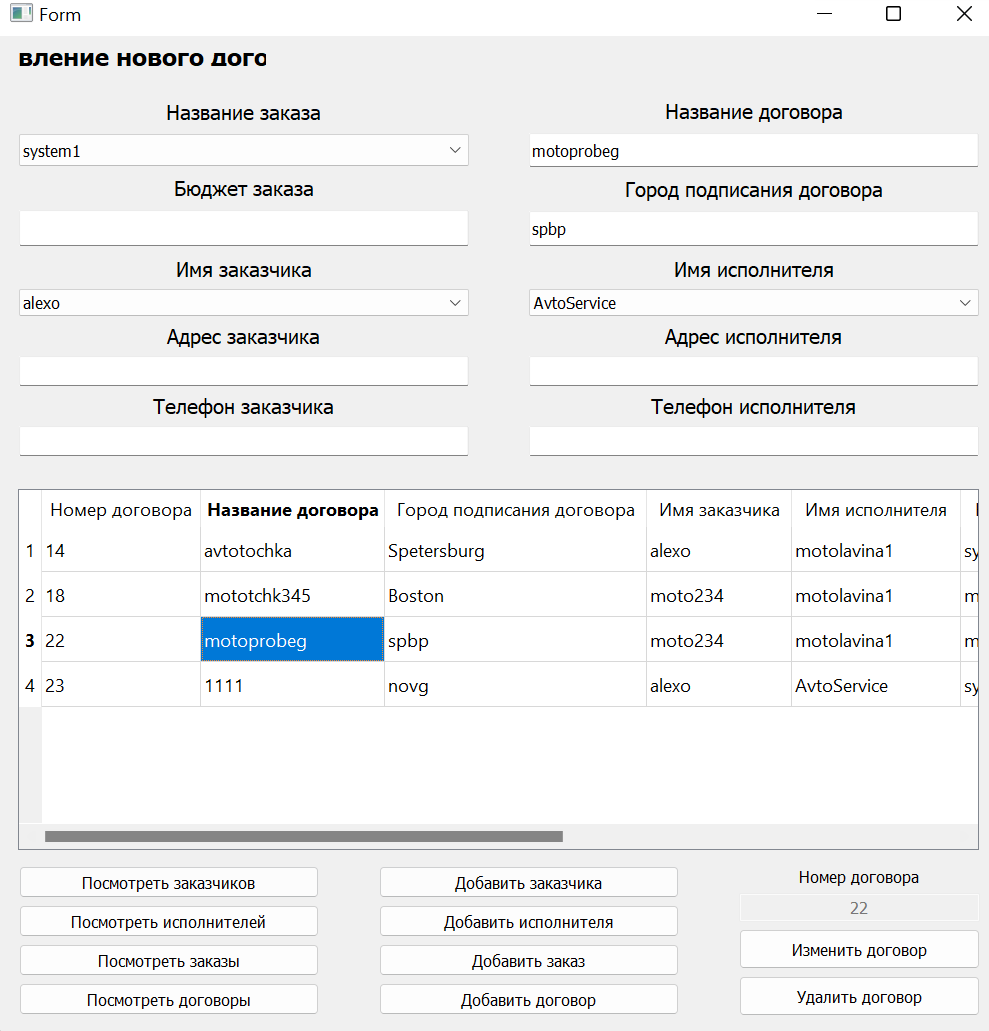


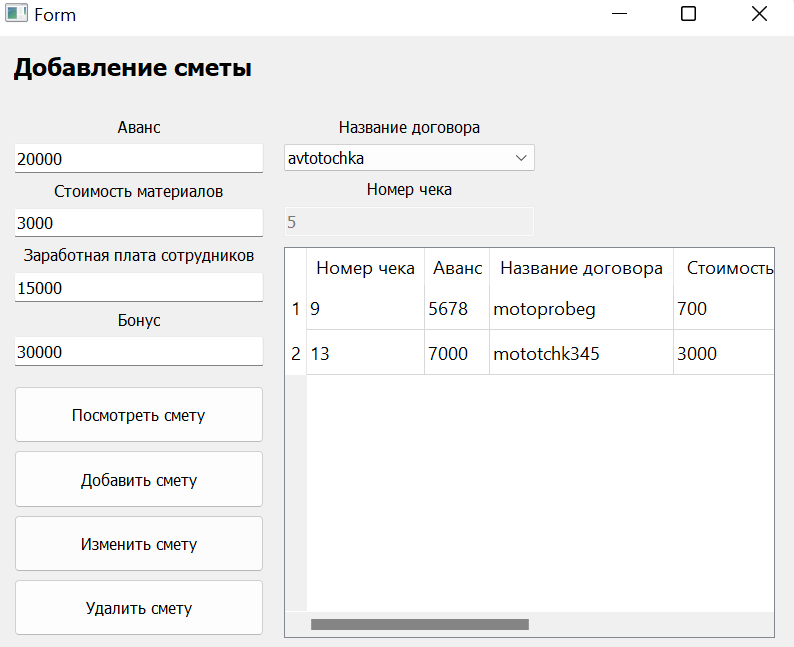
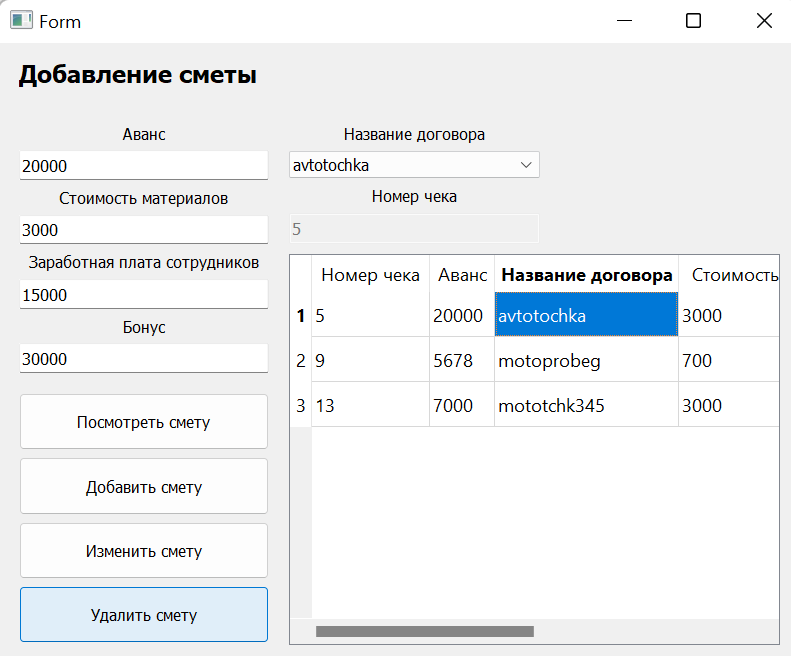












**Заключение**

Данная курсовая работа представляет собой законченное ПО в клиент-серверной архитектуре. В ПО клиентская часть выполнена в виде отдельного приложения на языке С++ (Qt). Серверная часть реализована на MS SQL и содержит необходимые компоненты для работы СУБД.

**Список источников**

1. Давыдова, Е.М. Базы данных [Электронный ресурс] : учебное пособие / Е.М. Давыдова, Н.А. Новгородова. — Электрон. дан. — М. : ТУСУР (Томский государственный университет систем управления и радиоэлектроники), 2007. — 166 с. — Режим доступа: http://e.lanbook.com/books/element.php?pl1\_id=1 1636— Загл. с экрана.

2. Муравьев, А.И. Базы данных [Электронный ресурс] : учебное пособие. — Электрон. дан. — М. : ТУСУР (Томский государственный университет систем управления и радиоэлектроники), 2006. — 137 с. — Режим доступа: http://e.lanbook.com/books/element.php?pl1\_id=11788 — Загл. с экрана.

3. Харрингтон Д. Проектирование объектно-ориентированных баз данных [Электронный ресурс] : . — Электрон. дан. — М. : ДМК Пресс, 2007. — 272 с. — Режим доступа: http://e.lanbook.com/books/element.php?pl 1\_id=1231 — Загл. с экрана.

4. Цимбал А.А., Аншина М.Л. Технологии создания распределенных систем. Для профессионалов. - СПб.: Питер, 2003.

5. Кагаловский М.Р. Энциклопедия технологий баз данных. — М.: Финансы и статистика, 2002.

6. Камер Э. Д. Сети ТСРЛР. Том 3. Разработка приложений типа клиент/сервер - Изд. Вильямс, М.-СПБ-Киев, 2002.

7. Арсеньев Б.П., Яковлев С.А. Интеграция распределенных баз данных. - СПб.: Издательство «Лань», 2001.

8. Сайт о программировании. URL: [https://metanit.com/] Время обращения март 2022 – май 2022