

МИНОБРНАУКИ РОССИИ

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«МИРЭА - Российский технологический университет»**

**РТУ МИРЭА**

Институт комплексной безопасности и специального приборостроения

Кафедра КБ-4 «Интеллектуальные системы информационной безопасности»

**ОТЧЕТ по дисциплине**

**«Клиент-серверные СУБД»**

По курсовой работе

Студент: Соловьев И.?.

Шифр учебной группы: БСБО-??-18

Руководитель: Иванова И.А.

Москва 2021 г.

**Оглавление**

# Задание на курсовую работу - 3

# Введение – 4

Анализ предметной области - 5

Структура БД - 5

- ER-Модель - 5

- Содержание БД и таблиц - 6

- Нормализация - 8

Программный продукт - 11

- Восстановление базы данных - 11

- Клиентское приложение - 12

Результат работы - 16

Заключение - 25

# **Задание на курсовую работу**

Тема: «Разработка клиент-серверного приложения»

Предметная область (ПрО): выкуп автомобилей с пробегом.

Цель работы: Закрепление теоретического материала по предмету «Клиент-серверные СУБД», а также по другим ранее изученным предметам.

Задание: Разработать клиент-серверное приложение, серверная часть которого реализована на PostgreSQL, а клиентская часть - на любом языке программирования. Приложение должно содержать модель предметной области в соответствии с вариантом.

**Введение**

База данных — это организованная структура, предназначенная

для хранения информации. Для управления базами данных необходимы СУБД - системы управления базами данных. Такие системы имеют следующие функции:

- Разграничение доступа к данным для разных пользователей

- Управление данными во внешней и оперативной памяти

- Журнализация вносимых изменений

- Резервное копирование и восстановление базы данных после сбоев

В рамках курсовой работы используется СУБД PostgreSQL, клиент написан на C#, интерфейс создан с помощью WPF, подключение к СУБД осуществляется через фреймворк Npgsql. Для воссоздания БД также может потребоваться pgAdmin (см. пункт "Восстановление базы данных").

Были поставлены следующие подзадачи:

1. Создать базу данных (далее - БД), достаточно подробно описывающую предметную область (10+ связанных заполненных таблиц)

1.1. Сделать структуру БД достаточно стабильной и минимизировать возможные аномалии без излишнего усложнения структуры (привести её к 3 нормальной форме)

2. Реализовать многопользовательский доступ к созданной БД для разграничения возможностей разных пользователей

3. Обеспечить безопасность данных пользователей (их шифрование)

4. Добавить средства автоматизации и повышения удобства пользования функционалом БД

5. Обеспечить безопасность базы данных в виде защиты от потенциальных SQL-инъекций

6. Создать интуитивный и функциональный пользовательский интерфейс

**Анализ предметной области**

Автоцентр «Автоград», который было взято в качестве «живого» примера организации, осуществляющей продажу автомобилей с пробегом, и от структуры и функций которого я отталкивался при построении даталогической модели, имеет следующую структуру подразделений (Рисунок 1).

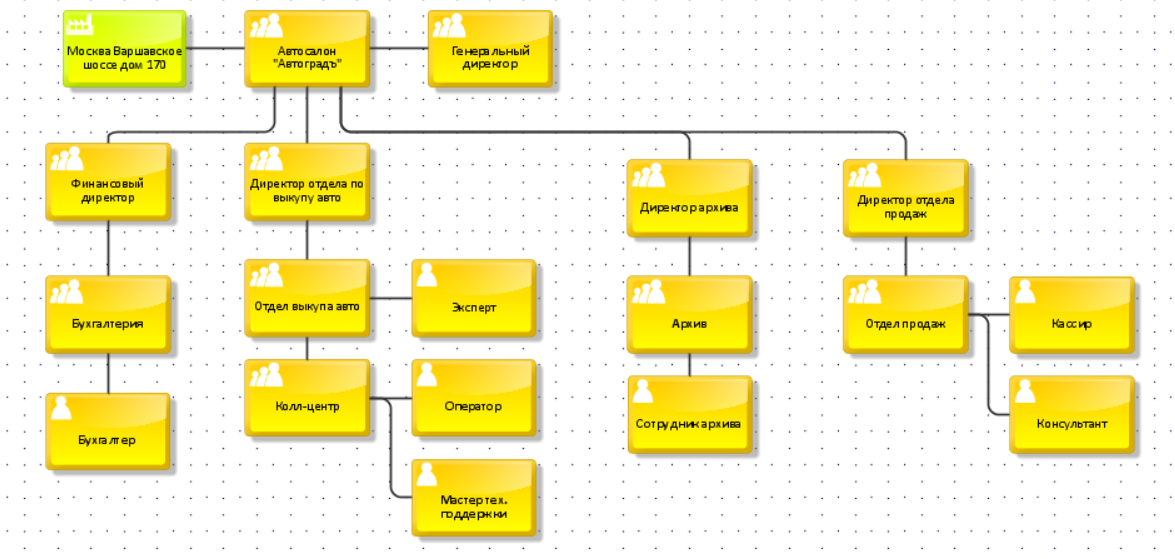


Рисунок 1 – Структура подразделений

В выкупе автомобиля с пробегом ту или иную роль играют:

1. Бухгалтерия.
2. Отдел выкупа авто.

При анализе должностной и функциональной структуры выше перечисленных подразделений (разумеется, с учётом специфики предметной области) я опирался на состав сотрудников автосалона и соответствующие должностные инструкции. Информация о составе рабочего коллектива взята из открытых источников (официального сайта).

Ниже представлена краткая концентрированная информационная «выжимка».

Отдел выкупа авто: Эксперт, колл-Центр. Колл-центр принимает заявки на выкуп авто от продавцов. С точки зрения ПрО, важны дата заявки, ФИО продавца, а также табельный номер оператора, принявшего заявку и название выкупаемой машины. Эксперт из отдела выкупа авто оценивает автомобиль, который хочет выкупить компания, устанавливает стоимость выкупа, восстановления и ремонта, а так же стоимость продажи. С точки зрения ПрО, важны лишь ФИО, а также табельный номер работника.

Бухгалтерия: Финансовый директор, Бухгалтер. Бухгалтер одобряет или отклоняет заявки Эксперта по выкупу автомобилей.

Полное и исчерпывающее понимание рассматриваемого бизнес-процесса возможно только на основании тщательного анализа потока нормативных документов, «циркулирующих» в организации.

Значимыми и основными такими документами являются:

1. Договор о покупке (Договор) – основополагающий документ покупки машины у Продавца.

Содержит информацию о сторонах договора: Эксперте и Продавце, а также о выкупаемом автомобиле.

Подписывается сторонами договора.

1. Паспорт транспортного средства (ПТС).

Документ, в котором содежатся сведения о автомобиле, его технической части и прежних владельцах.

1. Свидетельство о регистрации транспортного средства (СТС)

Документ, в котором так же содержатся сведения о автомобиле, необходим для постановки его в будущем на учет.

1. Расписка о получении денежных средств (РПДС).

Расписка, которую дает Продавец, необходима для отсутсвия конфликтов в будущем.

1. Доверенности.

Данный документ необходим для того, чтобы Эксперт мог заключать договора от имени компании.

1. Налоговая декларация

Документ, составляемый Бухгалтером, для налоговой инспекции

Исходя из вышесказанного, приблизительная обобщённая схема и хронология рассматриваемого бизнес-процесса (а также прямо или косвенно, но близко с ним связанных процессов) выглядят следующим образом.

Этап 1. Создание заявки на продажу.

На данном этапе происходит запись Оператором информации от Продавца о машине: марка, модель, ее состояние и тд. А так же информацию о самом Продавце. После заявка передается Эксперту.

Этап 2. Оценка машины.

Эксперт связывается с Продавцом, узнает все необходимые данные о машине, после чего Продавец обязан привезти машину в автосалон, где будет произведен ее осмотр Экспертом.

Эксперт проводит осмотр машины и оценивает ее. Называя стомость покупки Продавцу, а так же передает в Бухгалтерию информации о стоимости восстановления машины и предполагаемой стоимости продажи в автосалоне.

Также Эксперт может отказать Продавцу в покупке в случае, если машина окажется неликвидным товаром.

Этап 3. Покупка и оформление.

После осмотра и оценки Эксперт заключает Договор о покупке с Продавцом. После чего передает деньги Продавцу и составляется расписка о получении денег. Затем Договор о покупке подписывается обеими сторонами.

Продавец передает Эксперту все документы на машину (СТС и ПТС).

Таким образом, получается внятная целостная картина исследуемого бизнес-процесса, в котором чётко обозначены участники, их функции, а также последовательность «субакций», на которые данный процесс разбивается.

# **Структура БД**

Рисунок 2 – модель «Сущность – Связь» иследуемой области ПрО.

На рисунке 2 изображена модель «Сущность-Связь» исследуемой ПрО.

Полагаю, рисунок нуждается в комментарии. Спецификация сущностей и их атрибутов представлена в формате «Сущность: первичный ключ, *альтернативный ключ*, атрибут».

Записи ПТС: № Записи [varchar 20], № ПТС [varchar 40], Дата покупки [timestamp], Дата продажи [timestamp], Фамилия [varchar 40] Отчество [varchar 40], Имя [varchar 40].

ПТС: № ПТС[integer], Кол. владельцев [varchar 9], Компания производитель [varchar 40], Номер авто [varchar 9].

Авто: Гос номер[varchar 9], Год выпуска [varchar 12], Марка [varchar 40], Модель[varchar 40], Страна пр.[varchar 40], № Двигателя [bigint], № Колёс [bigint], № КПП [bigint], Тип топлива [varchar 10], Категория [varchar 20], Эко класс [varchar 20].

Бухгалтер: № сотрудника [bigint], Имя [varchar 40], Фамилия[varchar 40], Отчество[varchar 40].

Двигатель: № Записи[bigint], Модель [varchar 50], Мощность [varchar 30], Объем [varchar 30], Производитель[varchar 30], Тип[varchar 30].

Диски: № Записи [bigint], Производитель [varchar 50], Модель [varchar 30], Ширина [varchar 30], Тип [varchar 30].

Доверенность:№ доверенности[bigint], Дата выдачи [timestamp], № сотрудника[bigint], Срок действия [integer].

Дов-сть авто (Доверенность на продажу автомобиля): :№ записи [bigint], № автомобиля[varchar 9], Дата выдачи [timestamp], Кем выдана[bigint], Срок действия [integer].

Договор: № Договора [bigint], № доверенности[bigint], № продавца [bigint], № сотрудника. [bigint], Бухгалтер [bigint], № автомобиля [varchar 9], Дата сделки[timestamp], ПТС. [varchar 40], СТС [varchar 40].

Заявка: № Записи [bigint], № оператора [bigint], Эксперт [bigint], Номер Авто[varchar 9], Телефон [varchar 17].

Колесо: № Записи [bigint], № дисков [bigint], № шин [bigint], Диаметр[integer].

Контакты: № сотрудника [bigint], Email [varchar 40], Мобильный телефон [varchar 17], Корпоративный телефон [varchar 17].

Коорд. (Координаты продавца): № Записи [bigint], Email [varchar 40], Мобильный телефон [varchar 17], Город [varchar 100], Улица [varchar 100], Дом [integer], Квартира [integer].

КПП: № Записи [bigint], Тип коробки[varchar 40], Кол-во передач [integer].

Налог.Дек.(Налоговая декларация): № декларации [bigint], КПП [integer], Доход [integer], Налог [varchar 10], ОГРН [varchar 13], ОГРНИП [varchar 15], Период[varchar 10], Составляющий[bigint], Сумма[integer].

Оператор: № сотрудника [bigint], Имя [varchar 40], Фамилия [varchar 40], Отчество [varchar 40], ИНН [bigint].

Паспорт: Код паспорта [bigint], Номер [integer], Серия [integer], Кем выдан [varchar 100], Дата выдачи [timestamp].

Переч. Штрафов (Перечень штрафов): № в ГИБДД[bigint], Дата[timestamp], Сумма[integer], Номер авто [varcahr 9].

Продавец: № Записи[bigint], Имя [varchar 40], Фамилия [varchar 40], Отчество [varchar 40], ИНН [bigint], Паспорт [bigint].

РПДС(Расписка о получении денежных средств): № Записи [bigint], Кому выдана [bigint], Сумма [integer],

СТС: № СТС [varchar 40], Вес ТС[integer], VIN номер[integer], Номер авто[varchar 9], Номер двигателя[bigint], Номера кузова[integer], Номера шасси[integer], Цвет кузова[varchar 20].

Счет-фактура: № счета-фактуры[bigint], № сотрудника[integer], ИНН[bigint], КПП[integer], Налог[integer], Сумма документа[integer].

Шины: № Записи [bigint] , Модель[varchar 40], Сезонность[varchar 20], Производитель [varchar 40].

Эксперт: № сотрудника[bigint], Имя[varchar 40], Фамилия[varchar 40], Отчество[varchar 40], ИНН[bigint], КПП[integer], ОКПО[integer], Паспорт [bigint].

**Содержание БД и таблиц**

Вся БД представляет собой информацию об автоцентре и его клиентах.

Названия таблиц и столбцов соответствуют их содержимому, далее идёт краткое их описание и пояснение уникальности некоторых записей.

**"Двигатель"** - таблица для хранения двигателей и их характеристик (мощность, производитель, тип и т.д.).

**"КПП"** – таблица, содержащая характеристики различных коробок передач (вид и кол-во передач).

**"Шины"** – таблица, содержащая различные наименования шин (отличаются сезонностью, производителем и моделью).

**"Диски"** - таблица, содержащая различные наименования дисков.

**"Колесо"** - таблица, объединяющая в себе **"Шины"** и **"Диски"**

**"Авто"** - таблица для всех автомобилей, представленных к продаже в автосалоне. Включает себя данные из всех вышеперечисленных таблиц, дополняя их госномером и различной справочной информацией

**"ПТС"** -

**"Записи ПТС"** -

**"Переч. Штрафов"** -

**"СТС"** -

**"Бухгалтер"** -

**"Паспорт"** -

**"Продавец"** -

**"Эксперт"** -

**"Оператор"** -

**"Доверенность"** -

**"Дов-сть авто"** –

**"Заявка"** -

**"Контакты"** -

**"Координаты"** -

**"Налог.Дек."** –

**"РПДС"** -

**"Счет-фактура"** -

**"Договор"** -

**Нормализация**

**1НФ:**

В каждой из таблиц все столбцы атомарны (не подразделяются) и имеют строгую типизацию; строки в пределах каждой таблицы содержат в себе одинаковое количество столбцов и имеют независимый первичный ключ - поле ID, не связанное с внешними данными. Всё это видно в разделе "ER-Модель". Также, строки не имеют в себе дубликатов, несущих одинаковую информацию. Если есть две записи, где отличаются только ID - значит, они говорят о разных вещах. Так, например, в таблице Service\_Record могут быть записи об одинаковых по содержанию оказаниях услуг, но записи должны быть о каждом.

**2НФ:**

В данной БД для любых двух таблиц, связанных внешним ключом, имеет место отношение 1-ко-многим или 1-к-1. Рассмотрим эти связи:

Client:ID - Visit\_Record:Client\_ID - один клиент может совершить несколько визитов в разные отели сети.

Client:ID - Service\_Record:Client\_ID - одному клиенту в ходе его пребывания в отелях сети может быть оказано множество услуг.

Client:ID - Report\_Record:Client\_ID - один клиент может быть упомянут или быть автором нескольких жалоб.

Report\_Record:ID - Service\_Record\_ID - несколько (но обычно не больше одной) жалоб могут быть связаны с одним оказанием услуг. Одна жалоба не может включать в себя множество оказаний услуг сразу.

Hotel:City\_ID - City:ID - в одном городе может быть несколько отелей сети.

Room:ID - Visit\_Record:Room\_ID - один номер могут снимать разные люди.

Room:Hotel\_ID - Hotel:ID - отели могут содержать множество номеров, которые не могут принадлежать другим отелям.

Big\_Staff:City\_ID - City:ID - к офисам одного города может быть привязано несколько работников.

Hotel\_Staff:Hired\_By\_ID - Big\_Stuff:ID - один HR мог завербовать много сотрудников, даже не в своём городе привязки.

Hotel\_Staff:Hotel\_ID - Hotel:ID - много сотрудников может быть привязано к одному отелю.

Hotel\_Staff:ID - Report\_Record:Staff\_ID - один сотрудник может быть упомянут в нескольких жалобах.

Hotel\_Staff:ID - Service\_Record:Staff\_ID - один сотрудник может оказать множество услуг посетителям.

Schedule:Staff\_ID - Staff:ID - в расписании в определённое время на конкретной позиции может быть только один сотрудник, для дополнительных постов необходимо и дополнительное расписание.

Schedule:Next\_Record\_ID - Schedule:ID - за каждой записью в расписании может идти только одна следующая либо никакой, расписание не может раздваиваться.

Schedule\_List:First\_Record - Schedule:ID - любое одно расписание начинается с одной конкретной, своей записи.

Schedule\_List:Hotel\_ID - Hotel:ID - несколько расписаний может быть зафиксировано за одним конкретным отелем.

Также стоит заметить, что каждый не ключевой атрибут везде неприводимо зависит от первичного ключа и не подвергается дальнейшей декомпозиции.

**3НФ:**

Каждый не ключевой атрибут нетранзитивно зависит от первичного ключа, так как любая запись в данной БД имеет природу уникальности (см. примечание к 1НФ). В разделе "Содержание БД и таблиц" есть списки атрибутов, из которых видна нетранзитивность - эти атрибуты не выводятся друг из друга.

Следует добавить, что в ходе анализа БД я не выявил аномалий, мешающих 4НФ, но цель довести БД до 3НФ уже была достигнута.

Пример действий, направленных на достижение 2НФ и 3НФ и уменьшение повторного хранения данных - декомпозиция расписаний на Schedule\_List и Schedule, где в первой таблице хранятся название расписания (вместо того чтобы хранить его в каждой строке расписания) и ключ, указывающий на его начало. При создании таблицы Schedule было два варианта, первый - сделать там Foreign Key, указывающий на ID расписания, и второй, итоговый - сделать ссылку на следующую запись. Основное преимущество второго варианта, повлиявшее на выбор - отсутствие необходимости индексации поля с номером расписания (причём эффективность индексации была бы понижена за счёт множества неуникальных полей) и прямой доступ по ID всех записей, которые закреплены кластеризованным индексом, что теоретически должно повысить скорость работы при больших объёмах данных. При данном подходе сложнее (дольше) получить доступ к номеру расписания, имея только запись из этого самого расписания, но данная задача мне видится менее полезной и менее используемой чем получение расписаний целиком из их имени/ID.

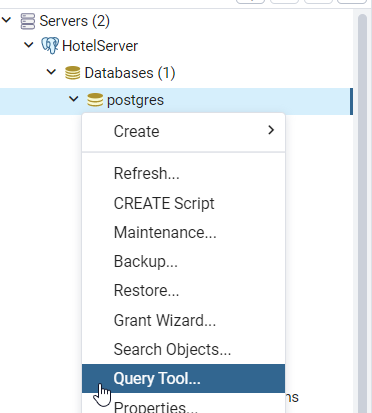
**Программный продукт**

**Восстановление базы данных**

Для просмотра кода, проверки работоспособности и дальнейшего использования описываемой программы и базы данных нужно подготовить для неё среду.

Чтобы использовать базу данных в PostgreSQL, данную СУБД нужно установить. Взять установщик можно с официального сайта для любой доступной ОС: <https://www.postgresql.org/download/> Обычно pgAdmin, желательный для удобства подготовки, идёт в комплекте, но если он не установится вместе с PostgreSQL, его можно найти здесь: <https://www.pgadmin.org/download/>

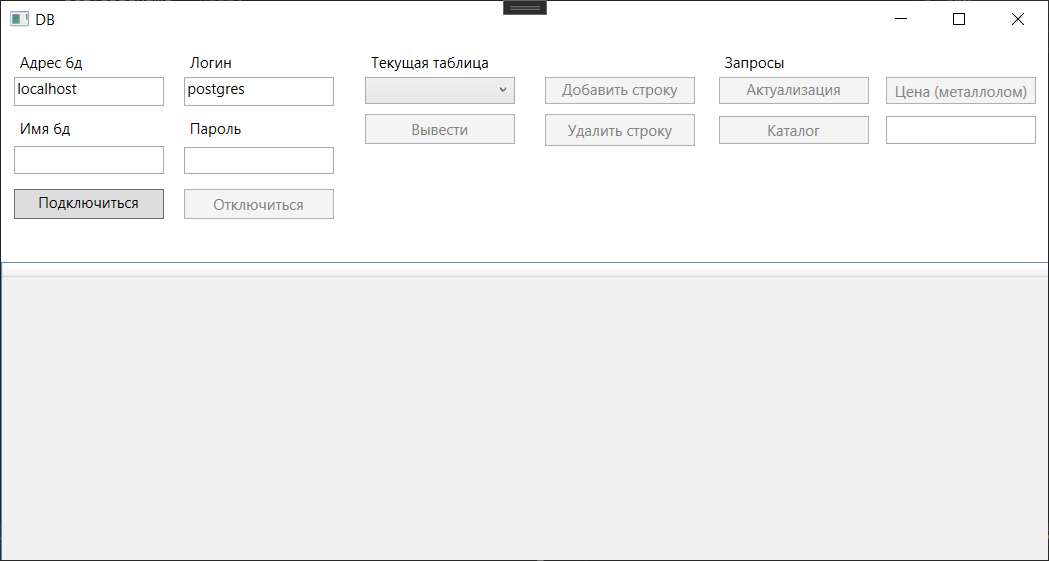
Закончив процесс установки и настройки и запустив pgAdmin, необходимо перейти в содержимое уже существуюшего сервера (либо создать новый) и там для стандартной (или опять же созданной) базы данных вызвать контекстное меню, где выбрать "Query Tool".



Далее нужно вставить содержимое файла формата .pgsql (приложен, открывается блокнотом) в поле ввода запроса и его выполнить. К важным частям его содержимого мы вернёмся позже. База данных восстановлена.

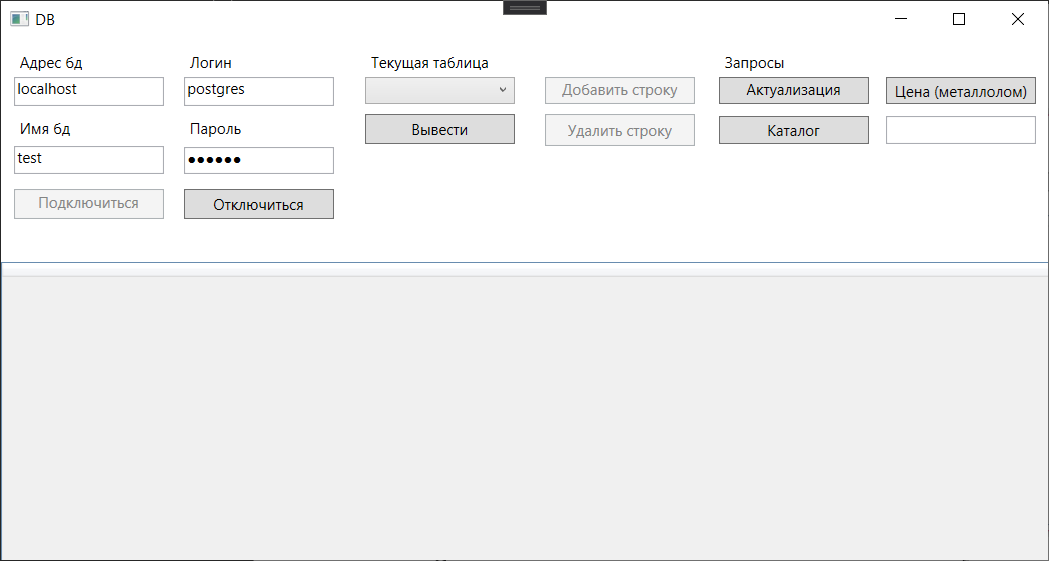
**Клиентское приложение**

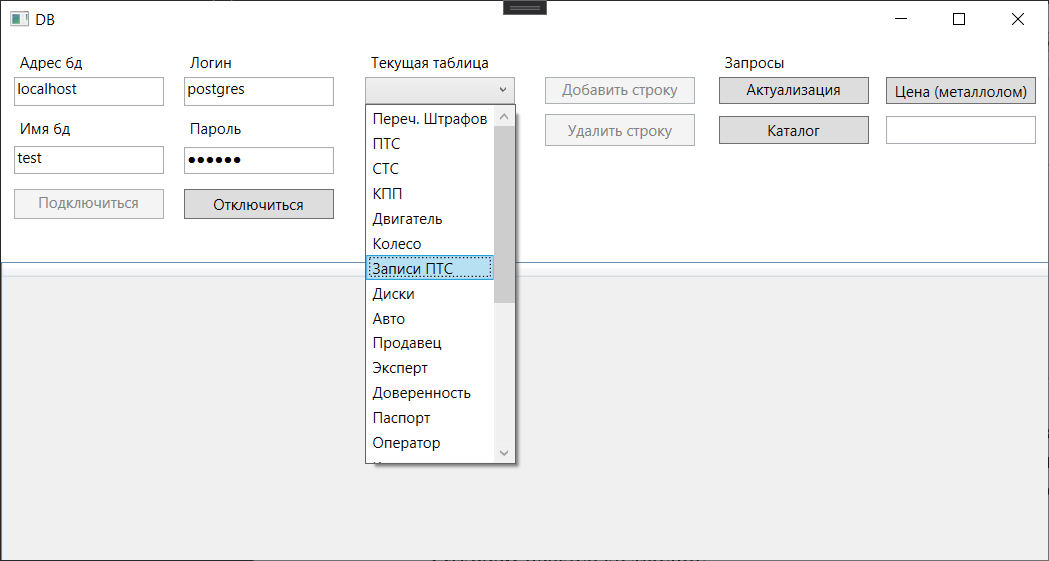
Переходим к клиентскому приложению. Проект с ним можно открыть с помощью Visual Studio - <https://visualstudio.microsoft.com/ru/downloads/> Возможно также придётся установить Npgsql через Nuget - встроенный менеджер расширений.

(Скомпилировав и) запустив приложение, можно сразу увидеть основной экран.

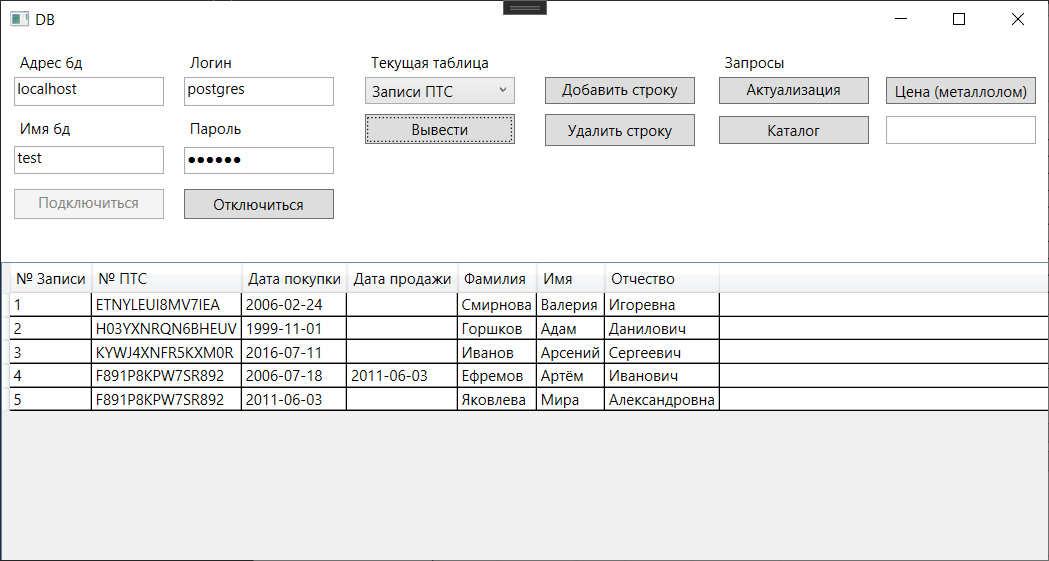
Весь функционал приложения находится в одном окне. Для подключения к БД необходимо ввести данные для входа в поля слева. Первое поле "Адрес бд" заполняется в соответствии с тем, где находится БД. Если она развёрнута на этом же устройстве, ничего менять не нужно, иначе нужно ввести IP-адрес устройства к которому нужно подключаться. Поле "Имя" заполняется в соответствии с внутренним именем БД. "Логин" и "Пароль" заполняются в соответствии с настроенной при установке PostgreSQL учётной записью. Стандартный логин - postgres. Также можно ввести логин-пароль других тестовых пользователей, созданных для данной БД - например, admin - admin с правами суперпользователя. В реальных рабочих условиях этого пользователя рекомендуется удалить.

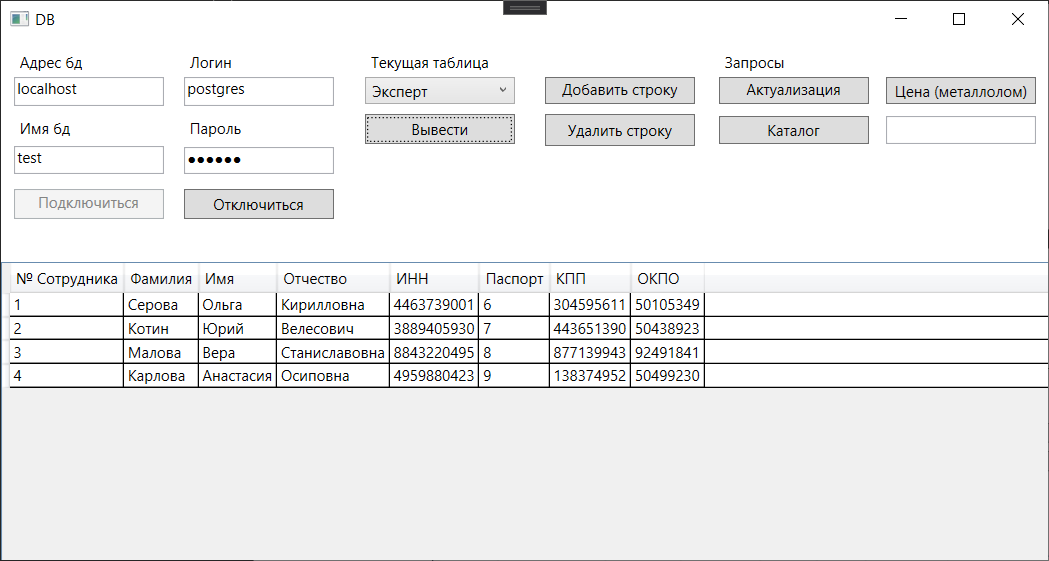
После заполнения полей, нужно нажать на кнопку "Подключиться" для входа.

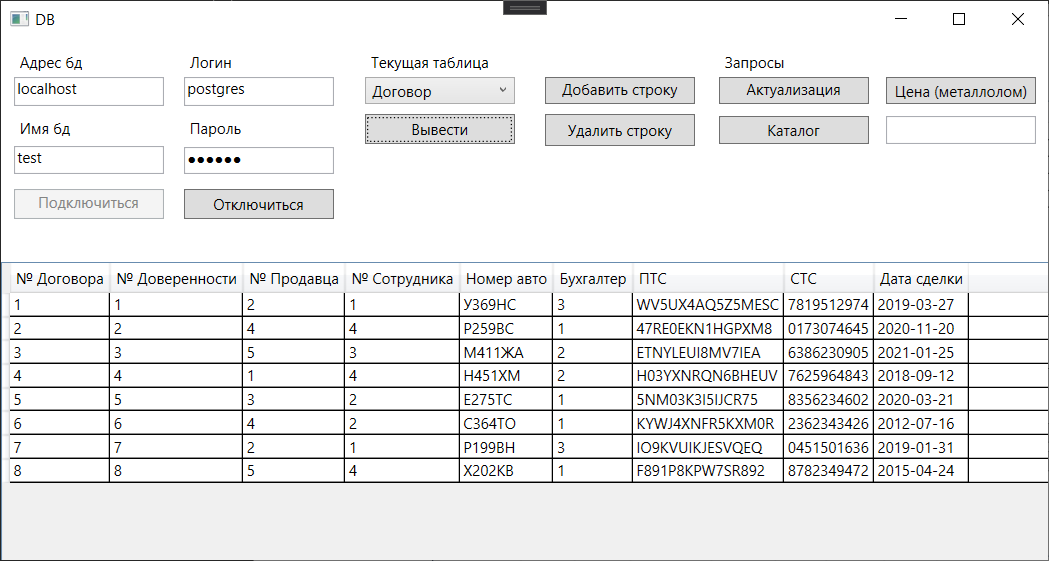


Активировались кнопки, в ComboBox подгрузились все доступные для пользователя таблицы.

Откроем несколько таблиц:







Изменение значений в таблице осуществляется посредством редактирования содержимого ячеек отображаемой таблицы, все изменения отправляются на сервер, что легко проверяется повторным запросом таблицы после её изменения. Если пользователю не хватает прав для данного действия, будет получено сообщение об ошибке.

**Результат работы**

Для демонстрации результата вернёмся к подзадачам, поставленным мной перед выполнением курсовой работы и перечисленным в пункте "Введение".

1. Создать базу данных (далее - БД), достаточно подробно описывающую предметную область (10+ связанных заполненных таблиц)

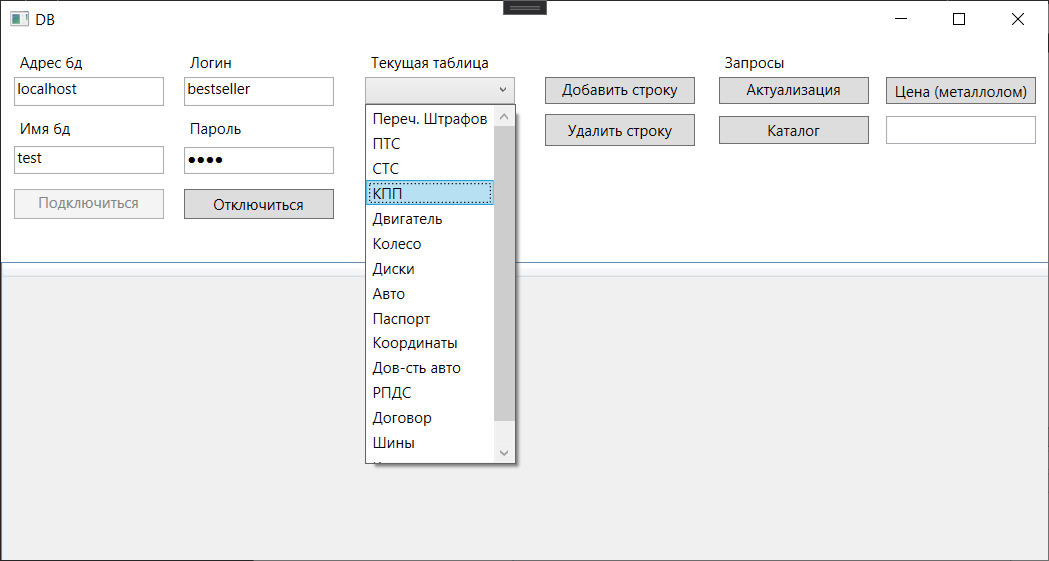
- Уже было продемонстрировано в разделе "Структура БД"

1.1. Сделать структуру БД достаточно стабильной и минимизировать возможные аномалии без излишнего усложнения структуры (привести её к 3 нормальной форме)

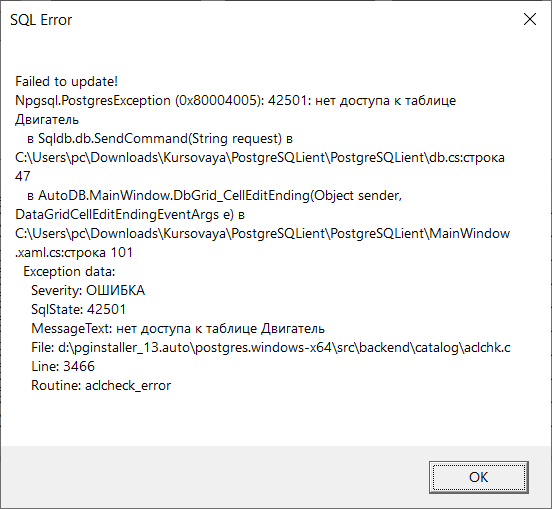
- Было продемонстрировано в пункте "Нормализация"

2. Реализовать многопользовательский доступ к созданной БД для разграничения возможностей разных пользователей

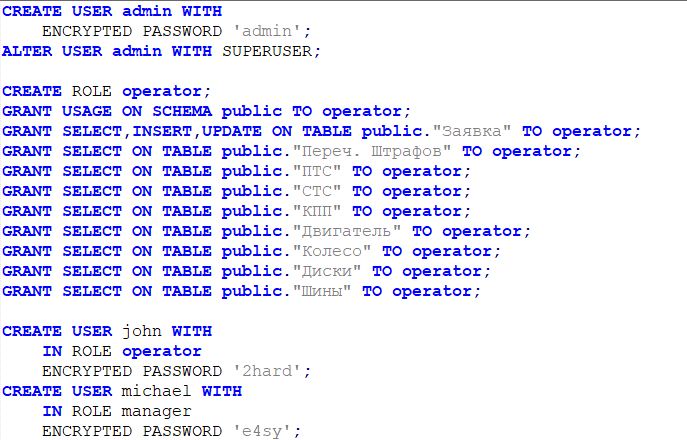
- Возможность подключаться от различных пользователей была частично продемонстрирована в пункте "Клиентское приложение", но не были напрямую продемонстрированы разграничения возможностей. Чтобы показать это разграничение, воспользуемся одним из дополнительных пользователей для входа - bestseller.



Уже видно, что не все таблицы доступны. Попробуем изменить данные в одной из таблиц, к которой у данного пользователя неполный доступ:



Всех доступных пользователей можно увидеть в файле roles.sql вместе с их разрешениями. Здесь видны роли operator, seller и accountant с конкретными разрешениями и пользователи, привязанные к данным ролям, а также пользователь admin-admin.





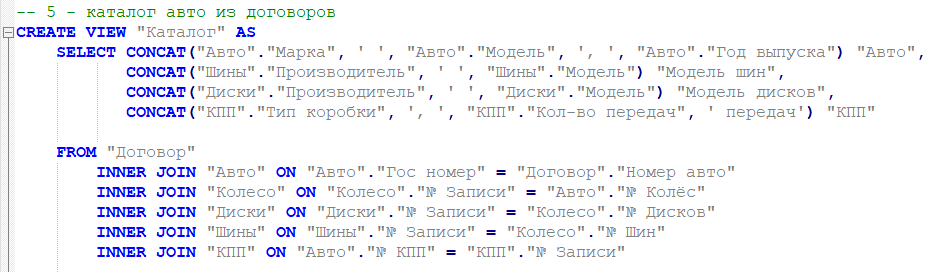
3. Обеспечить безопасность данных пользователей (их шифрование)

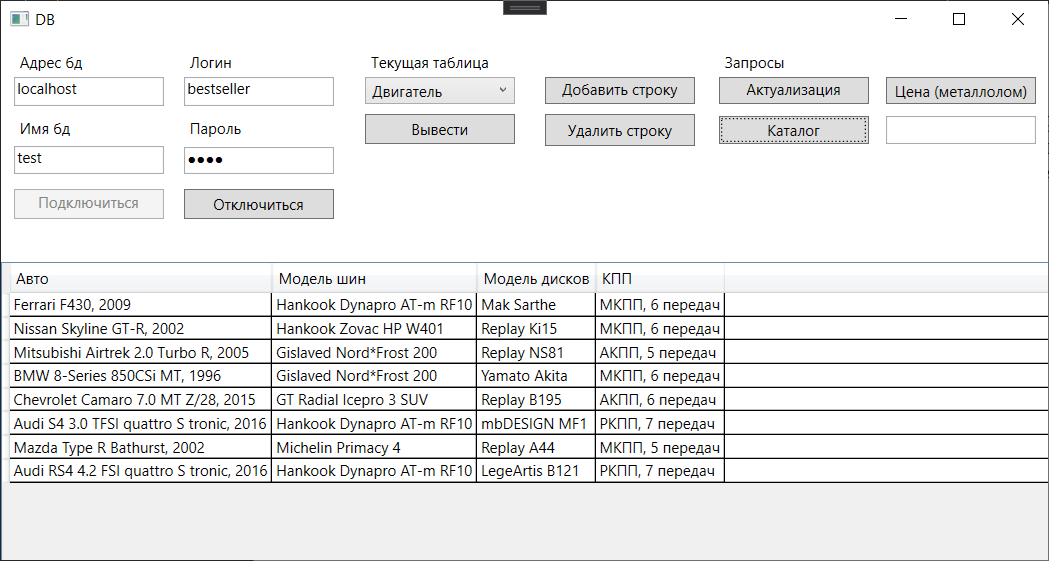
- Если обратить внимание на код выше, можно заметить, что при создании каждого пользователя пароль задаётся с ключевым словом ENCRYPTED, что сообщает СУБД напрямую, что данный пароль следует хранить зашифрованным.

4. Добавить средства автоматизации и повышения удобства пользования функционалом БД

- Были добавлены представление, хранимая процедура и функция, упрощающие работу. Обо всех по порядку:

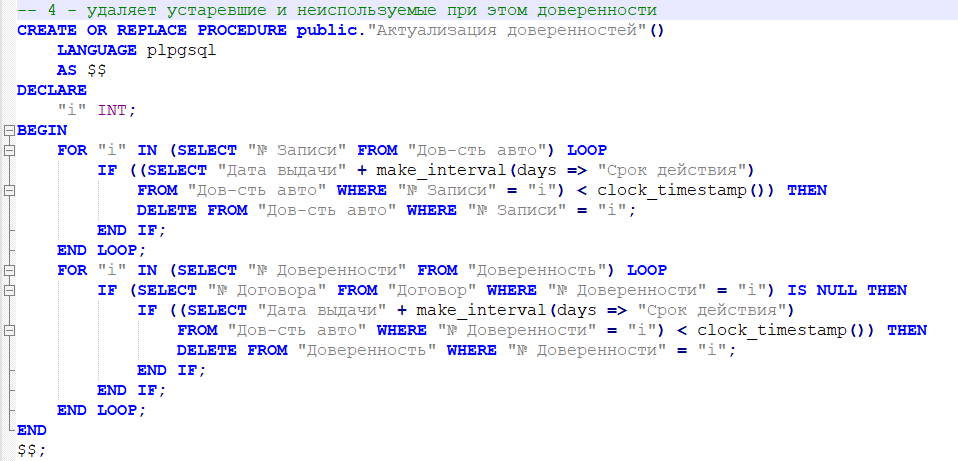
Представление “Каталог”:

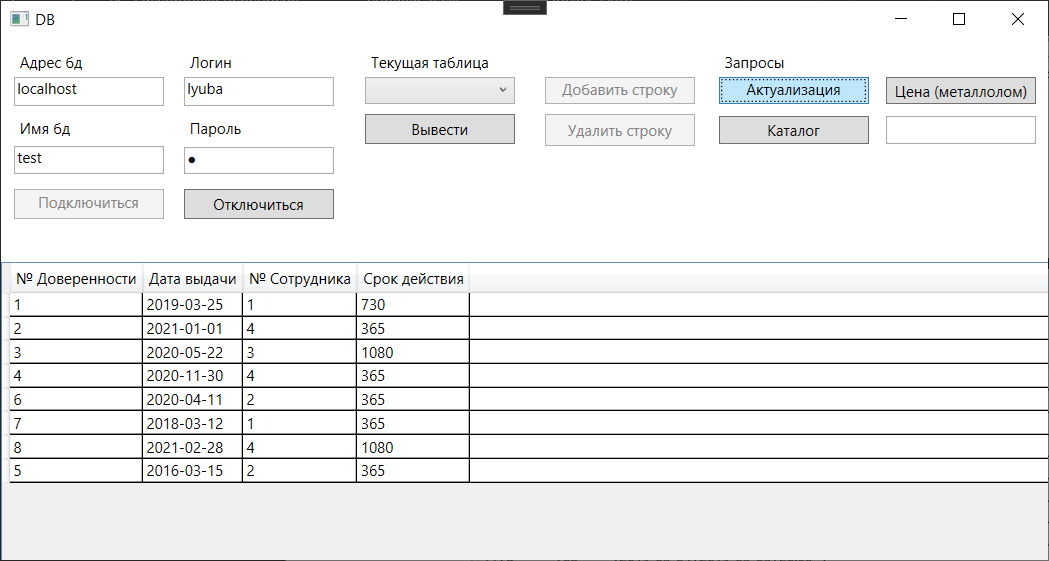




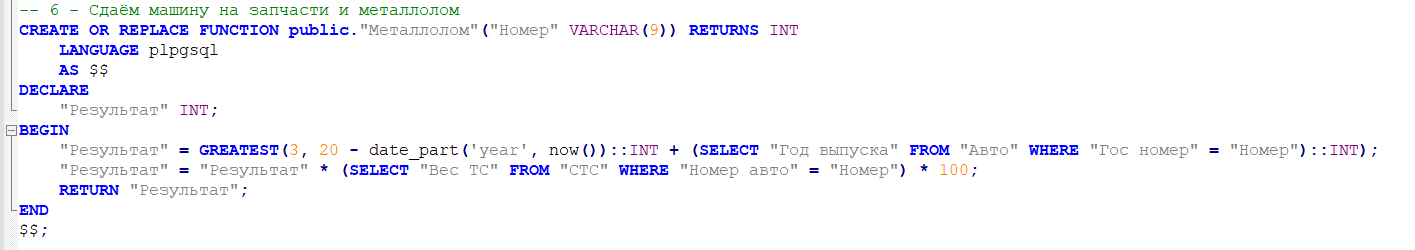
Данное представление отображает содержимое таблиц Авто, Колесо, Диски, Шины и КПП, объединяя их и приводя к удобочитаемый каталог. Доступно всегда (если у пользователя есть права).

Процедура “Актуализация доверенностей”

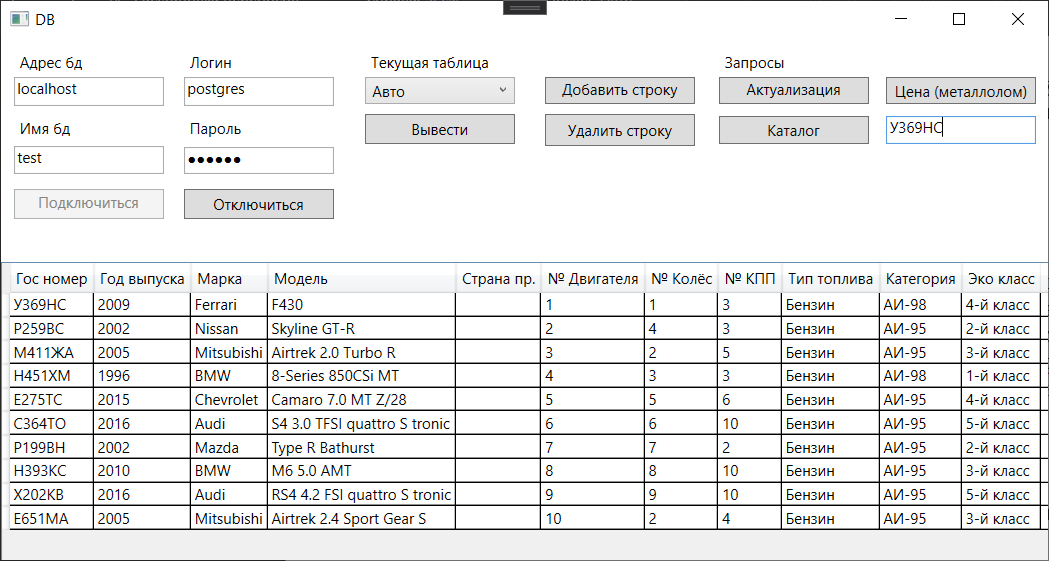


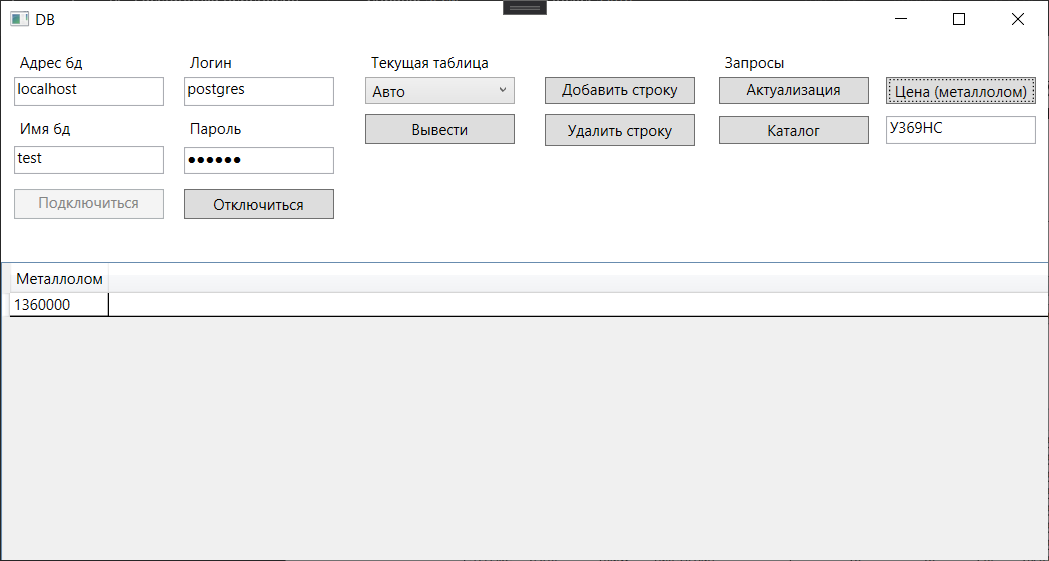
Данная процедура призвана помочь бухгалтерам и убирает неактуальные доверенности

Скалярная функция “Металлолом”



Данная функция высчитывает сумму, которую можно получить за автомобиль, если продать его на металлолом, на вход подается номер автомобиля.

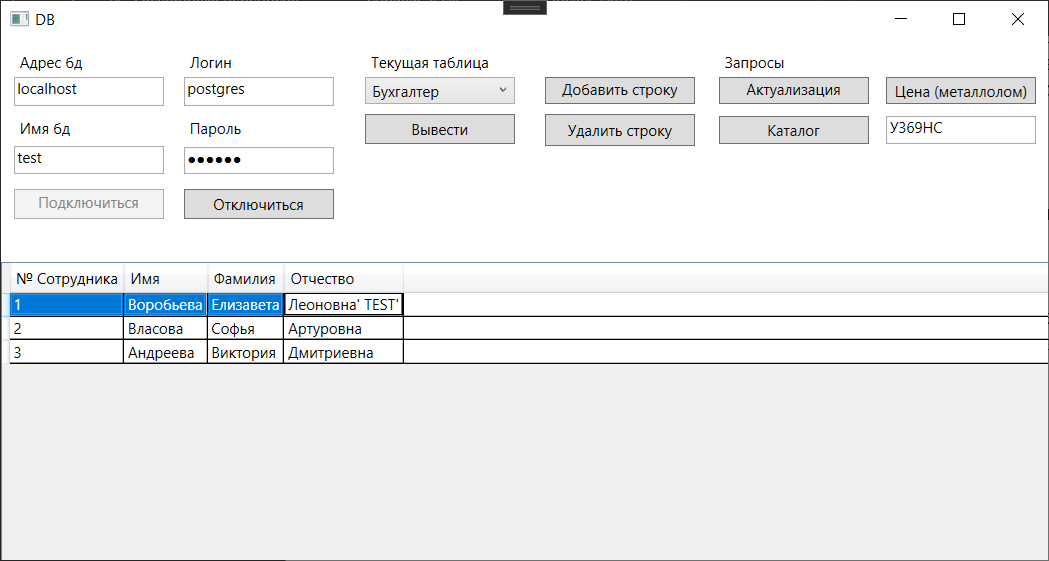


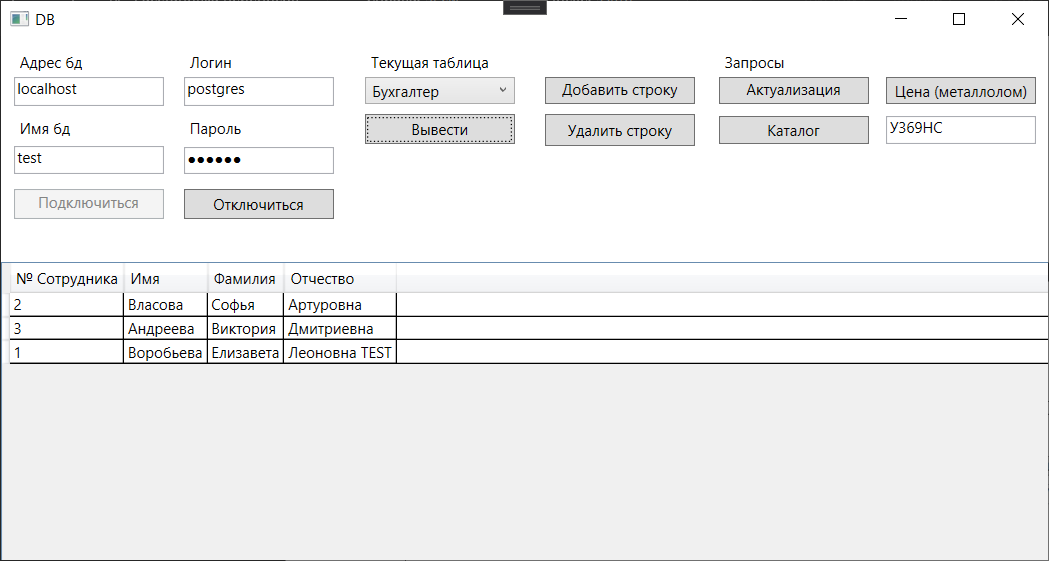


Стоит упомянуть, что в PostgreSQL любая функция/процедура является транзакцией и потому использование вышеперечисленных средств автоматизации можно рассчитывать как атомарные операции, не способные выполниться лишь наполовину.

5. Обеспечить безопасность базы данных в виде защиты от потенциальных SQL-инъекций

- Единственное место для пользовательского ввода в программе - ячейки таблицы. Когда пользователь пытается обновить значение, введённая строка проверяется на одиночные кавычки ('), которые и используются для подобных манипуляций.





6. Создать интуитивный и функциональный пользовательский интерфейс

- Весь интерфейс находится в одном окне, минималистичен и интуитивен в использовании.

**Заключение**

В ходе данной курсовой работы была разработана база данных, пригодная для использования при ведении гостиничного бизнеса, а также приложение с пользовательским интерфейсом, позволяющее с ней работать. Также, как показано в пункте "Результат работы", были выполнены все поставленные перед выполнением подзадачи.