

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ ФЕДЕРАЛЬНОЕ  
ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ  
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ «ВОРОНЕЖСКИЙ  
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

**Отчет по лабораторной работе: «Разработка приложений баз данных»**

Выполнила студентка 2 курса 71 группы  
Шаталина Надежда

Воронеж 2020

## Содержание

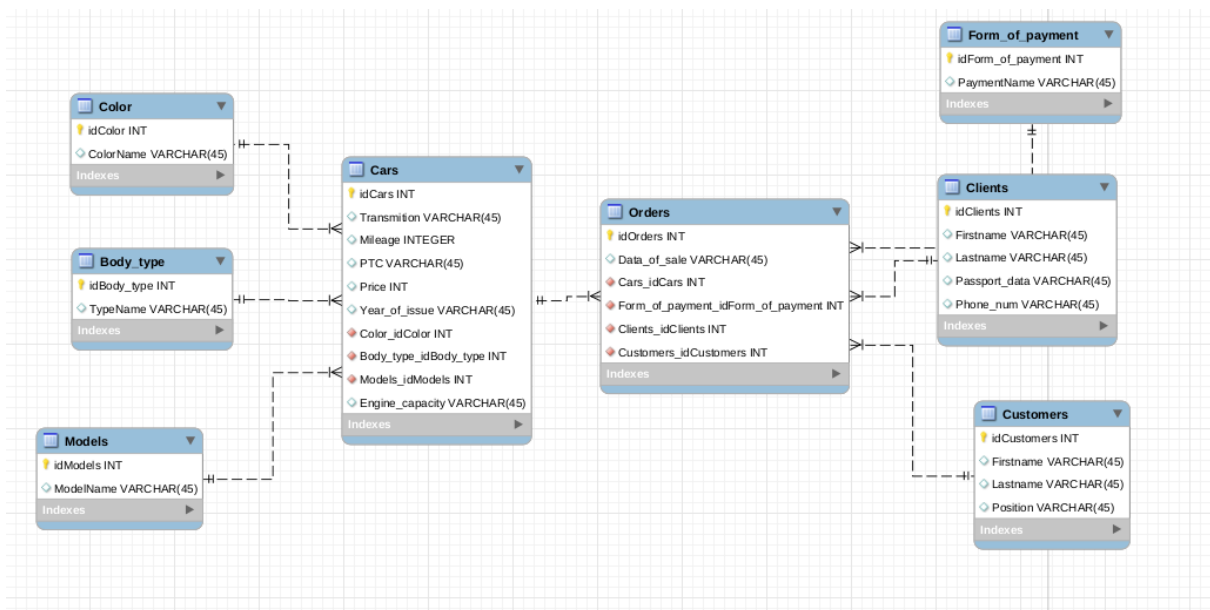
1. Постановка задачи	3
2. Построение модели базы данных	3
3. Работа на Python в Sublime Text	9
4. Вывод	16
5. Список использованных источников	16

## Постановка задачи

Разработать приложение для работы с базой данных на тему «Автомобильный салон» на языке программирования Python, используя для построения, работы и размещения базы данных «MySQL Workbench», для разработки приложения текстовый редактор Submle Text и терминал(командная строка) для запуска.

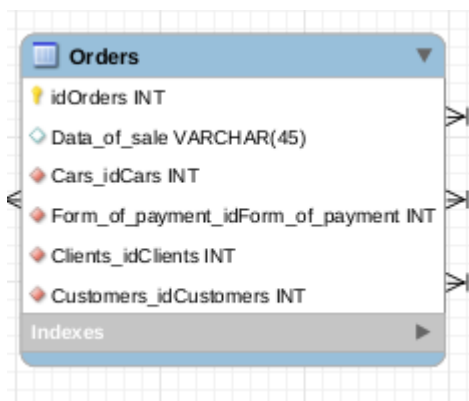
## Построение модели базы данных

Тема базы данных — Автосалон. Построение производила в «MySQL Workbench». Результат построения:

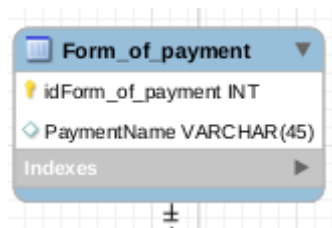


В модели можно выделить две основные сущности «Cars» и «Orders».

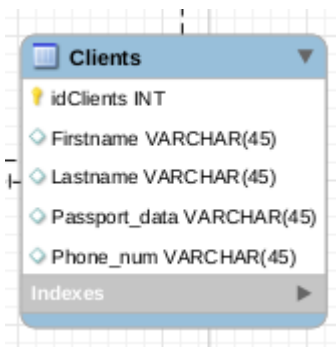
Сущность «Orders» представляет собой заказы покупателей автосалона. Она является дочерней сущностью для «Cars», «Form of payment», «Clients», «Customers» имеет поля: идентифкаторы родительских сущностей, свой идентифкатор и год продажи - «Data of sale».



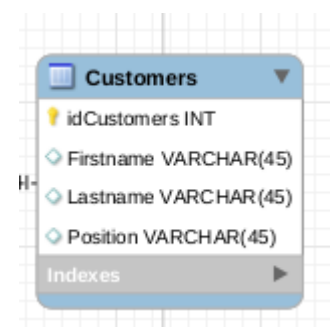
Родительская сущность «Form of Payment» показывает форму оплаты заказа. Она имеет два поля: идентификатор и название метода оплаты.



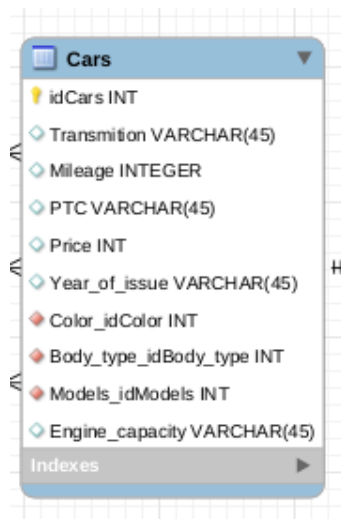
Родительская сущность «Clients» хранит информацию о клиентах автосалона. Она имеет поля: идентификатор, фамилия, имя, паспортные данные и телефон клиента.



Родительская сущность «Customers» хранит информацию о работниках автосалона. Она имеет поля: идентификатор, фамилия, имя и позицию работника.



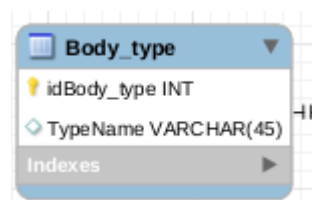
Вторая основная сущность «Cars» является дочерней для сущностей «Color», «Body Type», «Model» и предоставляет информацию о машинах в автосалоне. Эта сущность содержит идентификатор свой и родительских сущностей, также тип коробки передач, пробег, номер РТС, цена, год выпуска и объем двигателя.



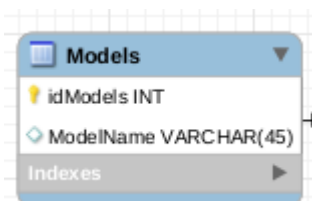
Родительская сущность «Color» представляет собой каталог цветов машины, имеет поля: идентификатор и название цвета.



Родительская сущность «Body type» хранит информацию о типах кузова машины и имеет два поля: идентификатор и название типа кузова.



Родительская сущность «Models» - каталог марок машин, представленных в автосалоне имеет два поля: идентификатор и название марки.



Далее происходит автоматическое создание скрипта и по нашей модели создается база данных. Появляются таблицы, которые я заполняю:

«Orders»:

#	idOrders	Data_of_sale	Cars_idCars	Clients_idClients	Customers_idCustomers	Form_of_payment_idForm_of_payment
1	2	2018	2	2	2	2
2	4	2020	4	4	2	2
3	14	2020	5	3	2	2
*	NULL	NULL	NULL	NULL	NULL	NULL

«Models»:

#	idModels	ModelName
1	1	BMW
2	2	VOLVO
3	3	HONDA
4	4	Lexus
*	NULL	NULL

«Form of payment»:

#	idForm_of_payment	PaymentName
1	1	Cash
2	2	Cashless
*	NULL	NULL

«Customers»:

#	idCustomers	Firstname	Lastname	Position
1	1	Козлова	Лилия	директор салона
2	2	Меренков	Алексей	менеджер
*	NULL	NULL	NULL	NULL

«Color»:

#	idColor	ColorName
1	1	White
2	2	Black
3	3	Red
4	4	Silver
5	5	Grey
6	6	Green
7	7	Blue
*	NULL	NULL

«Clients»:

#	idClients	Firstname	Lastname	Passport_data	Phone_num
1	1	Бурякова	Кристина	20156789	89515610784
2	2	Радина	Елена	20134587	89745896201
3	3	Мальцева	Анастасия	20168493	89456712308
4	4	Емельянов	Александр	20173458	89632145789
5	5	Кочкин	Владислав	20120594	89236475186
6	6	Попов	Михаил	20157984	89201336777
7	7	Серавин	Дмитрий	20194656	89316500354
8	8	Петрова	Екатерина	20175456	89234256215

«Cars»:

#	idCars	Transmission	Mileage	PTC	Price	Year_of_issue	Engine_capacity	Color_idColor	Body_type_idBody_type	Model_idModel
1	1	АКПП	120000	5465F8GJ4G5	3500000	2018	2.1	1	2	4
2	2	АКПП	240000	SD546123JR8	2780000	2017	1.9	1	2	2
3	3	МКПП	120000	GFJ99721KS8	2910000	2018	1.9	2	3	3
4	4	МКПП	357000	687687YH7S1	1470000	2016	1.8	3	4	1
5	5	АКПП	180000	F4564J6FJ866	3010000	2017	1.8	4	1	2
6	6	МКПП	96000	GF5JF6526123	2980000	2018	1.8	3	2	3
7	7	АКПП	57000	8464DH897SR	4700000	2019	2.1	4	3	1
8	8	АКПП	250000	3487FK968479	3200000	2017	1.5	3	4	3

«Body type»:

#	idBody_type	TypeName
1	1	Hatchback
2	2	Sedan
3	3	Coupe
4	4	Van
5	5	MUV/SUV
6	6	Converti...
7	7	Wagon
8	8	Jeep

## Работа на Python в Sublime Text

Прежде всего надо связать нашу базу данных и наше приложение. Это делается при помощи коннектора из библиотеки `mysql`.

```
mydb = mysql.connector.connect(
    host='localhost',
    user='root',
    passwd='9884Nadya',
    database='mydb')
mycursor = mydb.cursor()

# Commit changes
mydb.commit()
```

Далее я создаю графический интерфейс: окно, вкладки с помощью библиотеки Tkinter.

Окно:

```
window = Tk()
window.title('CarShop App')
window.geometry('700x400')
```

Вкладки:

```
#-----Interface-----
tab_control = ttk.Notebook(window)
catalog_tab = ttk.Frame(tab_control)
search_tab = ttk.Frame(tab_control)
order_tab = ttk.Frame(tab_control)
all_orders_tab = ttk.Frame(tab_control)
tab_control.add(catalog_tab, text='Каталог')
tab_control.add(search_tab, text='Поиск')
tab_control.add(order_tab, text='Заказать')
tab_control.add(all_orders_tab, text='Заказы')
```

Таким образом, у меня есть 4 вкладки: Каталог, Поиск, Заказать и Заказы.

**Вкладка Каталог:**

Данная вкладка отвечает за добавление, изменение и удаление машин в базе данных. Также мы можем просмотреть все машины находящиеся в базе.



Кнопка «Добавить»:

```
#submit car to database
def add_car():
    ...id_color = get_id_color(color_input.get())
    ...id_model = get_id_model(model_name_input.get())
    ...id_body_type = get_id_body_type(body_type_input.get())

    ...sql_command = 'INSERT INTO Cars (Transmission, Mileage, PTC, Price, Year of issue, \
    ...                                     Engine capacity, Color_id_color, Body_type_id_body_type, Model_id_model) \
    ...                                     VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s)'
    ...values = (transmission_input.get(), mileage_input.get(),
    ...          num_ptc_input.get(), price_input.get(),
    ...          year_issue_input.get(), float(engine_capacity_input.get()),
    ...          int(id_color), int(id_body_type), int(id_model))
    ...mycursor.execute(sql_command, values)

    ...#commit changes
    ...mydb.commit()
    ...messagebox.showinfo('', 'Машина добавлена!')
```

Кнопка «Просмотреть все»:

```
def watch_all():
    ...sql_command = 'SELECT * FROM Cars'
    ...mycursor.execute(sql_command)
    ...result = [i for i in mycursor]
    ...for i in result:
    ...    all_car_text.insert(INSERT, display_result(i) + '\n')
```

Для того, чтобы удалить или изменить машину необходимо ввести ее идентификатор.

Изменить можно любое одно или сразу несколько характеристик машины.

Кнопка «Изменить»:

```
def update_car():
    sql_command = 'UPDATE Cars SET %s WHERE idCars=%s'
    idCar = idCar_input_c.get()

    set_clauses = []
    if not(color_input.get() == ''):
        idColor = get_idcolor(color_input.get())
    else:
        idColor = ''

    if not(model_name_input.get() == ''):
        idModel = get_idModel(model_name_input.get())
    else:
        idModel = ''

    if not(body_type_input.get() == ''):
        idBody_type = get_idBody_type(body_type_input.get())
    else:
        idBody_type = ''

    transimition = transimition_input.get()
    mileage = mileage_input.get()
    engineCapacity = engine_capacity_input.get()
    year = year_issue_input.get()
    price = price_input.get()
```

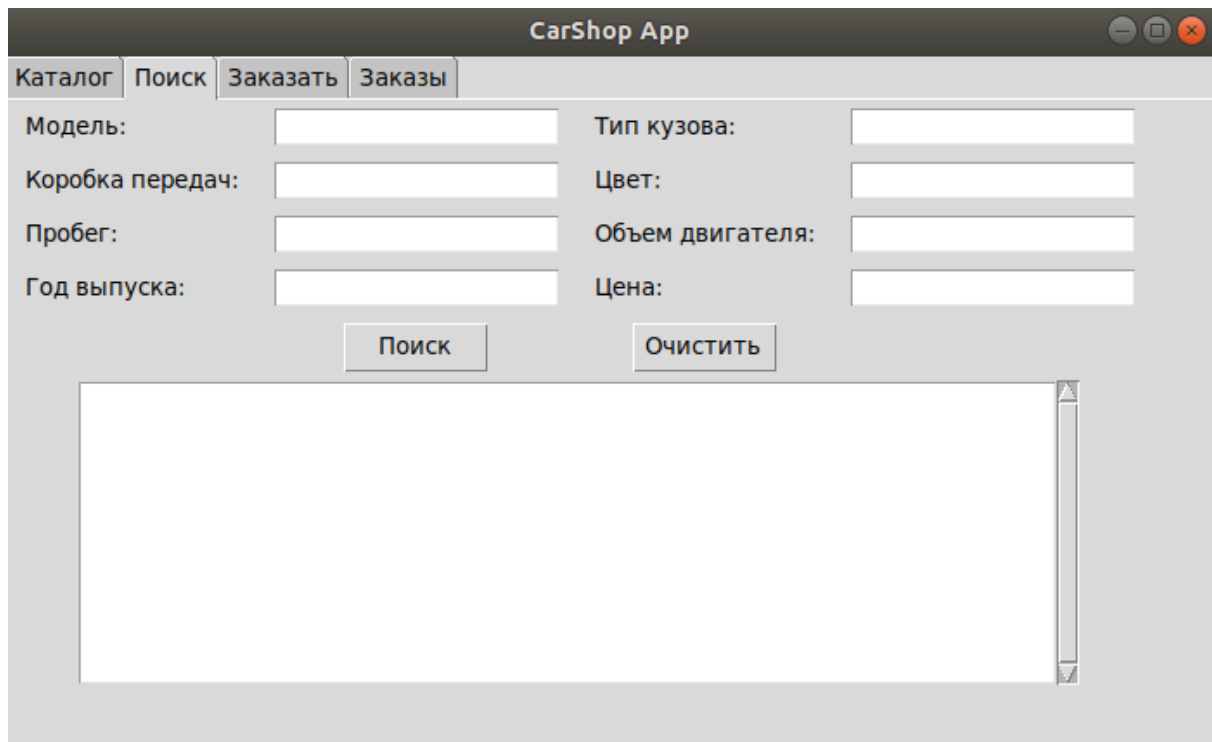
```
car_dict = create_dict(idModel, idBody_type, transimition,
                        idColor, mileage, engineCapacity,
                        year, price)
for key in car_dict.keys():
    if not(car_dict.get(key) == None or car_dict.get(key) == ''):
        set_clauses.append(key + "=" + str(car_dict.get(key)))
set_clauses = ','.join(set_clauses)
print(sql_command % (set_clauses, int(idCar)))
if idCar:
    mycursor.execute(sql_command % (set_clauses, int(idCar)))
    mydb.commit()
    messagebox.showinfo('', 'Данные успешно изменены!')
else:
    messagebox.showinfo('Ошибка!', 'Для изменения необходимо ввести номер машины!')
```

Кнопка «Удалить»:

```
def delete_car():
    sql_command = 'DELETE FROM Cars WHERE idCars=%s' % (idCar_input_c.get())
    confirmation = messagebox.askyesno(title='Подтверждение',
                                       message='Вы действительно хотите удалить машину с № %s' % (idCar_input_c.get()))
    if confirmation:
        mycursor.execute(sql_command)
        mydb.commit()
        messagebox.showinfo('', 'Данные успешно удалены!')
```

## Вкладка Поиск:

Данная вкладка производит поиск машин в базе данных. Машины можно искать по одному или нескольким признакам.



The screenshot shows the 'CarShop App' window with the 'Поиск' (Search) tab selected. The interface includes a header with four tabs: 'Каталог', 'Поиск', 'Заказать', and 'Заказы'. Below the tabs, there are eight input fields arranged in two columns. The left column contains: 'Модель:', 'Коробка передач:', 'Пробег:', and 'Год выпуска:'. The right column contains: 'Тип кузова:', 'Цвет:', 'Объем двигателя:', and 'Цена:'. Each label is followed by a text input field. Below the input fields are two buttons: 'Поиск' (Search) and 'Очистить' (Clear). At the bottom of the window is a large, empty rectangular area, likely a placeholder for search results.

## Кнопка «Поиск»:

```
# search a car
def search_car():
    sql_command = "SELECT * FROM Cars WHERE "
    where_clauses = []
    if not(color_input_s.get() == ''):
        idColor = get_idcolor(color_input_s.get())
    else:
        idColor = ''

    if not(model_name_input_s.get() == ''):
        idModel = get_idModel(model_name_input_s.get())
    else:
        idModel = ''

    if not(body_type_input_s.get() == ''):
        idBody_type = get_idBody_type(body_type_input_s.get())
    else:
        idBody_type = ''

    transmition = transmition_input_s.get()
    mileage = mileage_input_s.get()
    engineCapacity = engine_capacity_input_s.get()
    year = year_issue_input_s.get()
    price = price_input_s.get()

    car_dict = create_dict(idModel, idBody_type, transmition,
                           idColor, mileage, engineCapacity,
                           year, price)
    for key in car_dict.keys():
        if not(car_dict.get(key) == None or car_dict.get(key) == ''):
            where_clauses.append(key + "=" + "'" + car_dict.get(key) + "'")
    where_clauses = ' AND '.join(where_clauses)
```

```

...mycursor.execute(sql_command + where_clauses)
...result = [i for i in mycursor]
...if result:
...    for i in result:
...        if check_availability(i) == None:
...            txt.insert(INSERT, display_result(i) + '\n')
...        else:
...            messagebox.showinfo('Ошибка', "Машина с такими характеристиками отсутствует!")

```

### Вкладка Заказать:

Здесь производится оформление заказа. Если покупателя еще нет в базе, то он туда заносится проверка осуществляется через паспортные данные

Кнопка «Сделать заказ»:

```

def make_order():
    sql_command = "INSERT INTO Orders(Data_of_sale, Cars_idCars, \
    Clients_idClients, Customers_idCustomers, \
    Form_of_payment_idForm_of_payment) VALUES(%s, %s, %s, %s, %s)"

    customer_firstName = customer_firstName_input.get()
    customer_lastName = customer_lastName_input.get()
    passport_data = passport_data_input.get()
    phone = phone_input.get()
    seller_firstName = seller_firstName_input.get()
    seller_lastName = seller_lastName_input.get()
    seller_position = seller_position_input.get()
    payment_form = payment_form_input.get()
    idseller = idseller_input.get()

    date = datetime.date.today().year

    if not check_car(idCar_input.get()):
        idCar = None
        messagebox.showinfo('Ошибка', 'Машина уже продана')
    else:
        idCar = idCar_input.get()

    idClient = get_idClients(passport_data)
    if idClient == None:
        add_client(passport_data, customer_firstName, customer_lastName, phone)
        idClient = get_idClients(passport_data)

```

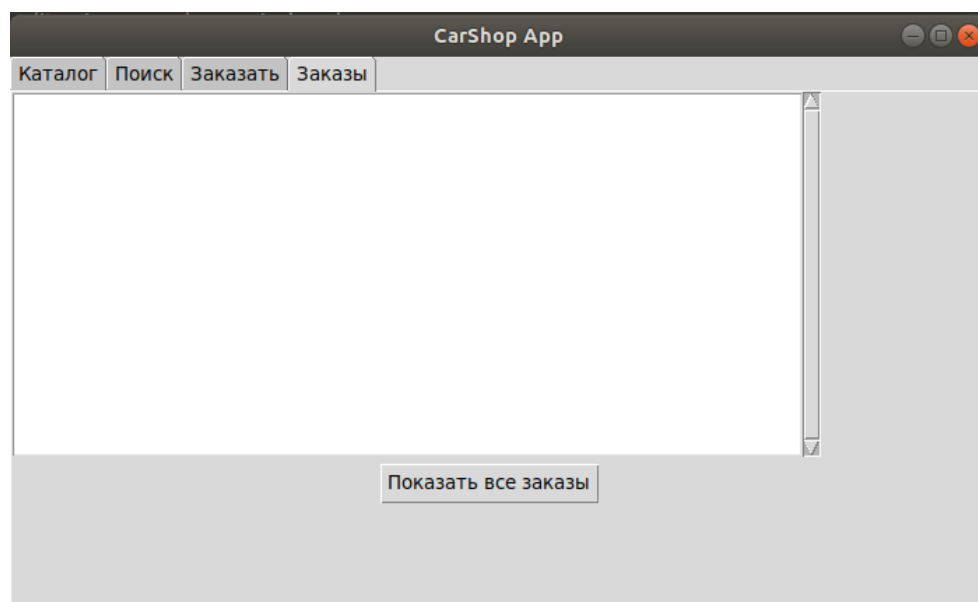
```

    idForm_of_payment = get_idform_payment(payment_form)
    values = (date, idCar, idClient, idForm_of_payment, idseller)
    val = []
    for data in values:
        if not data == None:
            val.append(data)
    if len(val) == len(values):
        mycursor.execute(sql_command, values)
        mydb.commit()
        messagebox.showinfo("", "Заказ оформлен!")
    else:
        messagebox.showinfo("Ошибка!", "Все поля должны быть заполнены!")

```

## Вкладка Заказы:

На этой вкладке можно просмотреть список сделок автосалона.



Кнопка «Посмотреть все заказы»:

```
def show_all_orders():  
    sql_command = 'SELECT * FROM Orders'  
    mycursor.execute(sql_command)  
    result = [i for i in mycursor]  
    for i in result:  
        orders_txt.insert(INSERT, modify_str(i) + '\n')
```

### **Вывод**

Результатом, проделанной работы, является UI-приложение, взаимодействующее с базой данных. С помощью приложения можно просматривать, искать, добавлять, изменять и удалять машины, которые есть в базе, также можно оформить заказ и просмотреть список всех заказов. Данное приложение предназначено для продавцов-менеджеров, потому что в нем не реализовано изменение работников, что может делать только руководитель.

### **Список использованных источников**

1. Веб-сайт: <https://metanit.com/sql/mysql/1.3.php>
2. Веб-сайт: <https://likegeeks.com/python-gui-examples-tkinter-tutorial/>