Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерного проектирования

Кафедра проектирования информационно-компьютерных систем

Дисциплина: Операционные системы, базы данных

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовой работе

на тему

**ПРОГРАММНОЕ СРЕДСТВО АВТОСАЛОНА**

Студент:

гр.­­­­­ 113802 Разумов Д.А.

Руководитель:

Василькова А. Н.

Медведев О. С.

Минск 2023

# Содержание

Введение 4

1 Анализ предметной области 5

1.1 Описание предметной области 5

1.2 Группы пользователей, их основные задачи и запросы к базе данных 6

1.3 Цели и задачи курсового проектирования 7

2 Проектирование базы данных 8

2.1 Инфологический этап проектирования 8

2.2 Логический этап проектирования 10

2.3 Обоснование выбора технологий и описание среды реализации 21

3 Программная реализация 22

3.1 Физическая структура базы данных 22

3.2 Реализация проекта базы данных 28

3.2.1 Создание и заполнение таблиц базы данных 28

3.2.2 Создание представлений 38

3.2.3 Назначение прав доступа 42

3.2.4 Создание индексов 44

3.2.5 Создание триггеров 46

3.3 Структура приложения 49

4 Тестирование 54

5 Описание применения 63

Заключение 64

Список использованных источников 65

Приложение A (обязательное) Листинг кода с комментариями 66

**ВВЕДЕНИЕ**

В этой курсовой работе будет спроектирована база данных для автосалона. В пояснительной записке будут представлены пять разделов: анализ предметной области, проектирование базы данных, программная реализация, тестирование, описание применения.

В первом разделе «Анализ предметной области» будут раскрыты следующие аспекты курсовой работы: описание предметной области, группы пользователей, их основные задачи и запросы к базе данных, цели и задачи курсового проектирования.

Во втором разделе «Проектирование базы данных» будут раскрыты следующие аспекты курсовой работы: инфологический этап проектирования, логический этап проектирования, обоснование выбора технологий и описание среды реализации.

В третьем разделе «Программная реализация» будут раскрыты следующие аспекты курсовой работы: Физическая структура базы данных, Реализация проекта базы данных, Структура приложения.

В четвертом разделе «Тестирование» будет продемонстрирована работа сайта.

В пятом разделе «Описание применения» будут представлены инструкции и рекомендации к использованию.

Перечень графического материала данной курсовой работы будет включать в себя две диаграммы: диаграмму деятельности и диаграмму вариантов использования.

# Анализ предметной области

## 1.1 Описание предметной области

В рамках данной курсовой работы рассматривается предметная область «Программное средство автосалона». В современном мире, в условиях, когда автомобиль является не роскошью, а постоянным средством передвижения, то практически каждый человек нуждается в нем. За частую, если человек не испытывает финансовые трудности, то он обращается в автосалон за покупкой нового автомобиля. Эта информация обширна и разрознена. Чтобы вести учет всех автомобилей, которые можно купить, в организации имеется потребность в структурировании данных об авто и заказах. Отсутствие такой возможности приводит к проблеме утери данных и большим временным затратам на выборку данных.

В функции автосалона входит не только продажа автомобилей, но и регистрация новых авто, обмен поддержанных и старых автомобилей на утилизацию, осуществление своих обязательств перед покупателем в виде ремонта и обслуживания автомобиля, осуществление сделок по поставке и продаже автомобилей, за частую и на международном уровне, профессиональная помощь и консультация, возможно, своим будущим покупателям.

Функционирование автосалона без автоматизации процесса достаточно трудоемкая работа. При автоматизации значительно сократиться время и осуществление основных операций автосалона. Автосалон располагает современной материальной базой, включающей вычислительную технику.

Автосалон – важная часть структуры каждого автомобильного бренда. Они представляют собой специальные магазины, офисы, дилеров, осуществляющие посреднические функции для каждого бренда. В большинстве случаев автосалоны являются импортерами каждого бренда автомобиля, в отдельной стране, а также в тех странах, которым принадлежит данный бренд автомобиля, автосалоны являются государственными учреждениями, но при этом имеют большое число посреднических фирм, эффективность деятельности которых очень высока.

Данная категория включает в себя исторически сформировавшийся экономический механизм, который осуществляет определенный комплекс услуг и отношений, содействует установлению и соблюдению баланса интересов между клиентами, автоконцернами (заводами), бизнесом и государством.

Автосалон - юридическое лицо, имеющее зарегистрированный товарный знак, фирменный знак, расчетный и иные счета в рублях и иностранной валюте в учреждениях банков. Автосалон имеет в своем составе администрацию, бухгалтерию, отдел кадров, отдел маркетинга и менеджмента, магазин.

Автомобильный бренд может быть представлен разными «фигурами» в зависимости от того, в чье имущество входит бренд. Им могут быть: отдельный человек или компания, которая является создателем данного бренда и имеет на него большую часть прав (Tesla Илона Маска), но в большинстве случаев это государственные большие автоконцерны (BMW Германия).

Клиенты – это люди, которые включены в процесс приобретения нового автомобиля по своим личным целям. В свою очередь автосалоны и автоконцерны являются организаторами процесса продажи и производства автомобилей. Клиенты, соответственно, люди разного пола и возраста, с абсолютно разным уровнем заработка, с собственным вкусом и предпочтением, для которых необходим широкой выбор как в ценовом диапазоне, так и в большом выборе моделей каждого бренда. Для каждого покупателя автомобиль имеет разную ценность, поскольку все люди различаются по признакам возраста, пола, личного заработка и уровня жизни, и у каждого эксплуатация автомобиля разная.

Ограничениями каждого автосалона являются:

* продажа авто в кредит или рассрочку людям неспособным выплатить данную сумму в срок;
* продажа авто лицам младше 18 лет;
* оформление авто на людей, не имеющих водительского удостоверения;

Главными мерками успешной покупки автомобиля из салона является:

* высококлассная подготовка, позволяющая менеджеру правильно изложить всю информацию, чтобы облегчить выбор покупателю;
* больший выбор автомобилей;
* минимальное представление, что он ожидает от каждого автомобиля.

В каждом автосалоне развернута локальная вычислительная сеть (ЛВС), которая позволяет осуществлять централизованное хранение и обработку информации. Сеть охватывает все отделы.

## 1.2 Группы пользователей, их основные задачи и запросы к базе данных

Для определения информационных потребностей необходимо знать кто будет являться пользователем базы данных, и какая информация может потребоваться.

Пользователь базы данных – это физическое лицо, которое имеет доступ к базе данных и пользуется услугами информационной системы для получения информации.

Параметрические (конечные) пользователи – это наиболее многочисленная группа лиц, для удовлетворения информационных потребностей которых и создается БД. Это специалисты в своей области деятельности (руководители подразделений предприятия, работники медицинских учреждений, читатели тематических библиотек, кассиры в сберегательных кассах и т. д.), которые обычно не имеют специальной подготовки по программированию. Они подразделяются на прямых конечных пользователей и косвенных конечных пользователей.

Прямые конечные пользователи обращаются с базой данных в диалоговом режиме. Часть из них умеет обращаться к заранее составленным приложениям и интерпретировать ответы информационной системы. Другие умеют самостоятельно разрабатывать новые приложения.

С данным программным средством работают следующие группы пользователей:

* клиенты;
* посетители;
* менеджеры;
* администраторы;

«Поcетитель» может:

* получить полный перечень автомобилей, находящихся в автосалоне;
* получить полную характеристику каждого автомобиля;
* получить все адреса и контакты всех филиалов данного автосалона;
* зарегистрироваться на сайте, чтобы стать клиентом;

«Клиент» может:

* получить полный перечень автомобилей, находящихся в автосалоне;
* получить полную характеристику каждого автомобиля;
* получить все адреса и контакты всех филиалов данного автосалона;
* просмотреть свои заказы;
* совершить заказ;
* управлять своими личными данными;

«Менеджеры» может:

* просматривать данные в таблице «Атомобили», «Модели», «Заказы»;
* вносить изменения (добавлять, редактировать, удалять) в таблицах: «Автомобили», «Модели», «Заказы»;

«Администратор» может:

* просматривать данные в таблицах: «Клиенты», «Сотрудники», «Контакты»;
* вносить изменения (добавлять, редактировать, удалять) в таблицах: «Клиенты», «Сотрудники», «Контакты»;

## 1.3 Цели и задачи курсового проектирования

Целью курсовой работы является разработка базы данных «Автосалон».

Для реализации поставленной цели необходимо решить следующие задачи:

* Изучить литературу по теме курсового проекта.
* Сделать анализ предметной области.
* Спроецировать и реализовать базу данных в среде конкретной СУБД.
* Подготовить графическую часть.

# Проектирование базы данных

## 2.1 Инфологический этап проектирования

Первым этапом и самым главным этапом в процессе проектирования и создания базы данных, является разработка инфологической модели.

Цель инфологического моделирования – обеспечение наиболее естественных для человека способов сбора и представления той информации, которую предполагается хранить в создаваемой базе данных. Основными конструктивными элементами инфологических моделей являются сущности, связи между ними и их свойства (атрибуты).

В данном курсовом проекте база данных создается для систематизации данных в автосалоне. База должна содержать данные о сотрудниках, клиентах, заказах, автомобилях и контактную информацию для связи.

В соответствии с предметной областью база данных проектируется с учётом следующих особенностей:

1. все авто оформляются в рамках заказа;
2. каждое авто содержит только одну модель, но каждая модель может содержать несколько авто.
3. каждый заказ совершается только одним клиентом, но каждый клиент может совершить несколько заказов;
4. каждый заказ совершается только одним сотрудником, но каждый сотрудник может совершить несколько заказов;
5. каждый авто может заказать несколько клиентов;
6. каждой модели авто может принадлежать несколько автомобилей;

Для базы данных «Автосалон» на основании проведенного анализа предметной области выделены следующие сущности:

**Сотрудники**. Атрибуты: ФИО; Номер телефона; Должность; Дата рождения; Логин; Пароль; Доступ; Роль; Email; Дата регистрации; Паспортные данные; Адрессные данные.

**Клиенты**. Атрибуты: ФИО; Номер телефона; Дата рождения; Логин; Пароль; Доступ; Роль; Email; Количество заказов; Дата регистрации; Паспортные данные; Адрессные данные.

**Заказы**. Атрибуты: Дата заказа; Клиент; Авто.

**Автомобили**. Атрибуты: Название; Двигатель; Год выпуска; Цена; Цвет; Комплектация; Фото; Модель; Наличие; Состояние;

**Контакты**. Атрибуты: Название; Номер телефона; Email; Время работы.

Анализ выявленных сущностей позволяет построить ЕR–диаграмму как показано на рисунке 1.



Рисунок 1 – ЕR-диаграмма предметной области

Для преобразования ЕR-диаграммы в схему базы данных приведём уточнённую ЕR-диаграмму, содержащую атрибуты сущностей как показано на рисунке 2.



Рисунок 2 – Уточненная ЕR-диаграмма предметной области

## 2.2 Логический этап проектирования

Процесс проектирования БД должен опираться на определенную модель данных (реляционная, сетевая, иерархическая), которая определяется типом предполагаемой для реализации информационной системы СУБД.

Логическое проектирование базы данных – это процесс создания модели используемой на предприятии информации на основе выбранной модели организации данных, но без учета типа целевой СУБД и других физических аспектов реализации.

На основе созданных сущностей составим реляционные отношения.

В сущности, «Заказы» первичным ключом является атрибут «Код заказа». Описание сущности «Заказы», предназначенное для хранения данных о всех совершенных заказах, представлено в таблице 1.

Таблица 1 – Описание отношения «Заказы»

|  |  |  |
| --- | --- | --- |
| Название атрибута | Заполнение | Описание |
| Код заказа | Обязательное  (ключевое поле) | Идентификационный номер заказа |
| Дата | Обязательное | Дата оформления заказа |
| Клиент | Обязательное | Клиент оформивший заказ |
| Сотрудник | Обязательное | Сотрудник |
| Автомобиль | Обязательное | Автомобиль, который купили |
| Контакты | Обязательное | Контактные данные |

В сущности, «Сотрудники» первичным ключом является атрибут «Код сотрудники». Описание сущности «Сотрудники», предназначенное для хранения данных обо всех сотрудниках, представлено в таблице 2.

Таблица 2 – Описание отношения «Сотрудники»

|  |  |  |  |
| --- | --- | --- | --- |
| Название атрибута | Заполнение | Описание | |
| Код сотрудника | Обязательное (ключевое поле) | Идентификационный номер сотрудника | |
| ФИО | Обязательное | Фамилия, имя, отчество сотрудника | |
| Номер телефона | Необязательное | Номер телефона сотрудника | |
| Дата рождения | Обязательное | Дата рождения сотрудника | |
| Должность | Обязательное | Админ или менеджер |
| Количество заказов | Обязательное | Количество заказов данного сотрудника |

Продолжжение таблицы 2

|  |  |  |
| --- | --- | --- |
| Адрес | Необязательное | Адресные данные: город, улица, дом, квартира |
| Паспорт | Необязательное | Паспортные данные: серия и номер паспорта, кем выдан |

В сущности, «Клиенты» первичным ключом является атрибут «Код клиента». Описание сущности «Клиенты», предназначенное для хранения данных обо всех клиентах, представлено в таблице 3.

Таблица 3 – Описание отношения «Клиенты»

|  |  |  |
| --- | --- | --- |
| Название атрибута | Заполнение | Описание |
| Код клиента | Обязательное (ключевое поле) | Идентификационный номер клиента |
| ФИО | Обязательное | Фамилия, имя, отчество клиента |
| Номер телефона | Необязательное | Номер телефона клиента |
| Дата рождения | Обязательное | Дата рождения клиента |
| Количество заказов | Обязательное | Количество заказов клиента |
| Адрес | Необязательное | Адресные данные: город, улица, дом, квартира |
| Паспорт | Необязательное | Паспортные данные: серия и номер паспорта, кем выдан |

В сущности, «Контакты» первичным ключом является атрибут «Код контакты». Описание сущности «Контакты», предназначенное для хранения контактных данных автосалона, представлено в таблице 4.

Таблица 4 – Описание отношения «Контакты»

|  |  |  |
| --- | --- | --- |
| Название атрибута | Заполнение | Описание |
| Код контакта | Обязательное (ключевое поле) | Идентификационный номер контакта |
| Название | Обязательное | Название |
| Номер телефона | Обязательное | Номер телефона |
| Email | Обязательное | Email контакного центра |

Продолжение таблицы 4

|  |  |  |
| --- | --- | --- |
| Время работы | Обязательное | Время работы контакного центра |
| Адрес | Обязательное | Адресные данные: город, улица, дом |

В сущности, «Автомобили» первичным ключом является атрибут «Код автомобиля». Описание сущности «Автомобили», предназначенное для хранения данных обо всех автомобилях автосалона, представлено в таблице 5.

Таблица 5 – Описание отношения «Автомобили»

|  |  |  |
| --- | --- | --- |
| Название атрибута | Заполнение | Описание |
| Код автомобиля | Обязательное (ключевое поле) | Идентификационный номер автомобиля |
| Название | Обязательное | Название автомобиля |
| Модель | Обязательное | Модель автомобиля |
| Двигатель | Обязательное | Электрический или бензиновый |
| Год выпуска | Обязательное | Год выпуска автомобиля |
| Цена | Обязательное | Цена автомобиля |
| Цвет | Обязательное | Цвет автомобиля |
| Комплектация | Обязательное | Базовая, полная или средняя |
| Фото | Обязательное | Фотография автомобиля |
| Состояние | Обязательное | Новое или БУ |
| Статус | Обязательное | Есть в наличии или нету в наличии |

Общий вид логической схемы базы данных приведен на рисунке 3.



Рисунок 3 – Схема базы данных

Нормализация отношений (таблиц) – одна из основополагающих частей теории реляционных баз данных. Нормализация имеет своей целью избавиться от избыточности в отношениях и модифицировать их структуру таким образом, чтобы процесс работы с ними не был обременён различными посторонними сложностями. При игнорировании такого подхода эффективность проектирования стремительно снижается, что вкупе с прочими подобными вольностями может привести к критическим последствиям.

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.

Приведем разработанную базу данных к 3НФ.

Первая нормальная форма (1НФ) предполагает, что каждое сохраняемое данное должно представлять скалярное значение, а отношение не должны содержать повторяющихся строк.

Нормализации подлежат все отношения.

Вынесем сложный атрибут Адрес у клиентов и сотрудников на «Город», «Улица», «Дом», «Квартира» в отдельные отношения «Адреса клиентов» и «Адреса сотрудников». Вынесем сложный атрибут Адрес у контактов на «Город», «Улица», «Дом» в отдельное отношение «Адреса контактов». Вынесем сложный атрибут Паспорт у клиентов и сотрудников на «Серия», «Номер», «Кем выдан» в отдельные отношения «Паспортные данные клиентов» и «Паспортные данные сотрудников». Разобъем поле «ФИО» на «Фамилия», «Имя», «Отчество».

Затем создадим отношение «Пользователи», которая будет связана с отношениями «Клиенты» и «Сотрудники».

После приведения к первой нормальной форме отношений «Заказы», «Клиенты», «Сотрудники», «Контакты», «Автомобили» примут вид, представленный в таблицах 6–17 соответственно.

Таблица 6 – Описание отношения «Заказы»

|  |  |  |
| --- | --- | --- |
| Название атрибуты | Заполнение | Описание |
| Код заказа | Обязательное  (ключевое поле) | Идентификационный номер заказа |
| Дата | Обязательное | Дата продажи |
| Код сотрудника | Обязательное | Идентификационный номер сотрудника |
| Код клиента | Обязательное | Идентификационный номер клиента |
| Код контакта | Обязательное | Идентификационный номер контактов |
| Код автомобиля | Обязательное | Идентификационных номер автомобиля |

Таблица 7 – Описание отношения «Контакты»

|  |  |  |
| --- | --- | --- |
| Название атрибута | Заполнение | Описание |
| Код контакта | Обязательное (ключевое поле) | Идентификационный номер |
| Название | Обязательное | Название |
| Номер телефона | Обязательное | Номер телефона |
| Email | Обязательное | Email |
| Время работ | Обязательное | Время работы контактного центра |
| Код адреса | Обязательное | Идентификационный номер адресных данных |

Таблица 8 – Описание отношения «Адреса контактов»

|  |  |  |
| --- | --- | --- |
| Название атрибута | Заполнение | Описание |
| Код адреса | Обязательное (ключевое поле) | Идентификационный номер адреса |
| Город | Обязательное | Город контактного центра |
| Улица | Обязательное | Улица контактного центра |
| Дом | Обязательное | Номер дома контактного центра |

Таблица 9 – Описание отношения «Клиенты»

|  |  |  |
| --- | --- | --- |
| Название атрибуты | Заполнение | Описание |
| Код клиента | Обязательное  (ключевое поле) | Идентификационный номер |
| Фамилия | Обязательное | Фамилия клиента |
| Имя | Обязательное | Имя клиента |
| Отчество | Необязательное | Отчество клиента |
| Дата рождения | Обязательное | Дата рождения клиента |
| Номер телефона | Необязательное | Номер телефона клиента |
| Количество заказов | Обязательное | Количество заказов клиента |
| Код адреса | Обязательное | Идентификационный номер адресных данных |
| Код паспорта | Обязательное | Идентификационный номер паспортных данных |
| Код авторизации | Обязательное | Идентификационный номер данных для авторизации |

Таблица 10 – Описание отношения «Адреса клиентов»

|  |  |  |
| --- | --- | --- |
| Название атрибута | Заполнение | Описание |
| Код адреса | Обязательное (ключевое поле) | Идентификационный номер адресных данных |

Продолжение таблицы 10

|  |  |  |
| --- | --- | --- |
| Город | Необязательное | Город клиента |
| Улица | Необязательное | Улица клиента |
| Дом | Необязательное | Номер дома клиента |
| Квартира | Необязательное | Номер квартиры клиента |

Таблица 11 – Описание отношения «Паспортные данные клиентов»

|  |  |  |
| --- | --- | --- |
| Название атрибута | Заполнение | Описание |
| Код паспорта | Обязательное (ключевое поле) | Идентификационный номер паспортных данных |
| Серия | Необязательное | Серия паспорта клиента |
| Номер | Необязательное | Номер паспорта клиента |
| Кем выдан | Необязательное | Кем выдан паспорт |

Таблица 12 – Описание отношения «Сотрудники»

|  |  |  |
| --- | --- | --- |
| Название атрибуты | Заполнение | Описание |
| Код сотрудника | Обязательное  (ключевое поле) | Идентификационный номер |
| Фамилия | Обязательное | Фамилия сотрудника |
| Имя | Обязательное | Имя сотрудника |
| Отчество | Необязательное | Отчество сотрудника |
| Дата рождения | Обязательное | Дата рождения сотрудника |
| Номер телефона | Необязательное | Номер телефона сотрудника |
| Должность | Обязательное | Админ или менеджер |
| Количество заказов | Обязательное | Количество заказов сотрудника |
| Код адреса | Обязательное | Идентификационный номер адресных данных |

Продолжение таблицы 12

|  |  |  |
| --- | --- | --- |
| Код паспорта | Обязательное | Идентификационный номер паспортных данных |
| Код авторизации | Обязательное | Идентификационный номер данных для авторизации |

Таблица 13 – Описание отношения «Адреса сотрудников»

|  |  |  |
| --- | --- | --- |
| Название атрибута | Заполнение | Описание |
| Код адреса | Обязательное (ключевое поле) | Идентификационный номер адресных данных |
| Город | Необязательное | Город сотрудника |
| Улица | Необязательное | Улица сотрудника |
| Дом | Необязательное | Номер дома сотрудника |
| Квартира | Необязательное | Номер квартиры сотрудника |

Таблица 14 – Описание отношения «Паспортные данные сотрудников»

|  |  |  |
| --- | --- | --- |
| Название атрибута | Заполнение | Описание |
| Код паспорта | Обязательное (ключевое поле) | Идентификационный номер паспортных данных |
| Серия | Необязательное | Серия паспорта сотрудника |
| Номер | Необязательное | Номер паспорта сотрудника |
| Кем выдан | Необязательное | Кем выдан паспорт |

Таблица 15 – Описание сущности «Авторизация»

|  |  |  |
| --- | --- | --- |
| Название атрибута | Заполнение | Описание |
| Код авторизации | Обязательное (ключевое поле) | Идентификационный номер данных для авторизации |
| Логин | Обязательное | Логин |
| Пароль | Обязательное | Пароль |
| Доступ | Обязательное | Доступ имеется или не имеется |

Продолжение таблицы 15

|  |  |  |
| --- | --- | --- |
| Роль | Обязательное | Клиент или сотрудник |
| Дата регистрации | Обязательное | Дата регистрации |
| Email | Обязательное | Email |

2НФ. В нашем случае составные первичные ключи отсутствуют, так имеют простой первичный ключ и поэтому они сразу находятся во второй нормальной форме.

3НФ. В отношении «Автомобили» атрибут «Модель» зависит не от первичного ключа, поэтому его следует вынести в отдельное отношение «Модели».

Таблица 16 – Описание сущности «Автомобили»

|  |  |  |
| --- | --- | --- |
| Название атрибута | Заполнение | Описание |
| Код автомобиля | Обязательное (ключевое поле) | Идентификационный номер автомобиля |
| Название | Обязательное | Название автомобиля |
| Двигатель | Обязательное | Электрический или бензиновый |
| Год выпуска | Обязательное | Год выпуска автомобиля |
| Цена | Обязательное | Стоимость автомобиля |
| Цвет | Обязательное | Цвет автомобиля |
| Комплектация | Обязательное | Базовая, средняя или полная |
| Фотография | Обязательное | Фотография автомобиля |
| Состояние | Обязательное | Новое или БУ |
| Статус | Обязательное | Есть в наличии или нету |
| Код модели | Обязательное | Идентификационный номер модели автомобиля |

Таблица 17 – Описание сущности «Модели»

|  |  |  |
| --- | --- | --- |
| Название атрибута | Заполнение | Описание |
| Код модели | Обязательное (ключевое поле) | Идентификационный номер модели автомобиля |
| Модель | Обязательное | Название модели |
| Фотография | Обязательное | Общая фотография для данной модели автомобиля |
| Количество | Обязательное | Количество машин данной модели в автосалоне |

Схема базы данных после нормализации приведена на рисунке 4.



Рисунок 4 – Схема данных после нормализации

## 2.3 Обоснование выбора технологий и описание среды реализации

В качестве среды проектирования базы данных был выбран MySQL 8.0.31 – система управления реляционными базами данных.

Разработку и поддержку MySQL осуществляет корпорация Oracle, получившая права на торговую марку вместе с поглощённой Sun Microsystems, которая ранее приобрела шведскую компанию GNU General Public License, так и под собственной коммерческой лицензией.

Улучшения MySQL 8.0.31:

1. Использование по умолчанию движка InnoDB.
2. Улучшение функций по секционированию данных. Расширенный синтаксис для разбиения больших таблиц на несколько частей, размещенных в файловых системах.
3. Новый механизм оптимизации вложенных запросов и JOIN-операций.
4. Переработана система внутренних блокировок.

Для визуального проектирования был использован продукт PHPMyAdmin. PHPMyAdmin – это унифицированный визуальный инструмент для архитекторов баз данных, разработчиков и администраторов баз данных. PHPMyAdmin предоставляет моделирование данных, разработку SQL-кода и комплексные инструменты администрирования для настройки сервера, администрирования пользователей, резервного копирования и многого другого. PHPMyAdmin доступен в Windows, Linux и Mac OS X.

Ключевые особенности включают:

* позволяет создавать модели и управлять ими, преобразовывать динамическую базу данных в модель, а также создавать и редактировать таблицы, и вставлять данные;
* преобразовывать диаграммы ER в операторы SQL и отправлять их на сервер SQL, создавать модели из целевой базы данных или даже из импортированных файлов SQL;
* позволяет создавать, управлять и настраивать подключения и параметры подключения к серверам баз данных MySQL, также позволяет выполнять SQL-запросы к этим соединениям с помощью встроенного редактора;
* позволяет создавать, управлять и организовывать подключения к базе данных, визуально выбирать таблицы и столбцы, легко перемещаться между схемами, выбирать таблицы и поля, создавать новые или отбрасывать их;
* значительно упрощает управление пользователями: просмотр учетных записей все пользователей сервера, добавление и удаление пользователей, предоставлять и отзывать привилегии, изменять глобальные разрешения и разрешения базы данных, изменять пароли, проводить аудит;

Все эти функции и интуитивно понятный интерфейс делают PHPMyAdmin незаменимым инструментом для MySQL на локальном рабочем столе.

# Программная реализация

## 3.1 Физическая структура базы данных

Физическая структура базы данных определяет тип и свойства данных, которые будут записаны в память компьютера.

На стадии физического проектирования происходит непосредственная реализация базы данных. Были созданы следующие таблицы: работодатели, вакансии, пол, должности, агент, образование, соискатели, сделки. Определены первичные и внешние ключи, типы данных полей таблиц.

Типы данных разделяют на числовые, дробные, символьные, даты и времени. Основные типы данных:

1. Числовые – TINYINT, SMALLINT, MEDIUMINT, INT, BIGINT;
2. Дробные – FLOAT, DOUBLE, DECIMAL;
3. Символьные – CHAR, VARCHAR, TINYTEXT, TEXT и т.д.;
4. Даты и времени – DATE, DATETIME, TIMESTAMP, TIME, YEAR.

В базе данных были использованы: INT, VARCHAR, DATE, DOUBLE:

* INT – хранит числа в диапазоне от -2147483648 до 2147483647;
* VARCHAR – может хранить не более 255 символов;
* DATE – дата в формате ГГГГ-ММ-ДД;
* TEXT – может хранить не более 65 535 символов.

Таблица «Заказы» хранит информацию о заказах. Структура приведена в таблице 18.

Таблица 18 – Структура таблицы «Заказы».

|  |  |  |  |
| --- | --- | --- | --- |
| Название поля | Тип | Ключ | Описание |
| id | Int | Первичный | Идентификационный номер заказа |
| date | Timestamp |  | Дата оформления заказа |
| id\_client | Int | Внешний | Идентификационный номер клиента |
| id\_employee | Int | Внешний | Идентификационный номер сотрудника |
| id\_auto | Int | Внешний | Идентификационный номер автомобиля |
| id\_contact | Int | Внешний | Идентификационный номер контактов |

Таблица «Контакты» хранит информацию о контактах. Структура приведена в таблице 19.

Таблица 19 – Структура таблицы «Контакты».

|  |  |  |  |
| --- | --- | --- | --- |
| Название поля | Тип | Ключ | Описание |
| id | Int | Первичный | Идентификационный номер заказа |
| name | Varchar(60) |  | Название контактного центра |
| phone | Varchar(20) |  | Номер телефона |
| email | Varchar(60) |  | Email |
| work\_time | Text |  | Время работы |
| id\_address | Int | Внешний | Идентификационный номер адресных данных |

Таблица «Адреса контактов» хранит информацию об адресах контактов. Структура приведена в таблице 20.

Таблица 20 – Структура таблицы «Адреса контактов».

|  |  |  |  |
| --- | --- | --- | --- |
| Название поля | Тип | Ключ | Описание |
| id | Int | Первичный | Идентификационный номер адреса |
| city | Varchar(30) |  | Город |
| street | Varchar(30) |  | Улица |
| house | Varchar(30) |  | Дом |

Таблица «Сотрудники» хранит информацию о сотрудниках автосалона. Структура приведена в таблице 21.

Таблица 21 – Структура таблицы «Сотрудники».

|  |  |  |  |
| --- | --- | --- | --- |
| Название поля | Тип | Ключ | Описание |
| id | Int | Первичный | Идентификационный номер сотрудника |
| last\_name | Varchar(60) |  | Фамилия сотрудника |
| first\_name | Varchar(60) |  | Имя сотрудника |
| surname | Varchar(60) |  | Отчество сотрудника |

Продолжение таблицы 21

|  |  |  |  |
| --- | --- | --- | --- |
| phone | Varchar(20) |  | Номер телефона сотрудника |
| date\_birth | Date |  | Дата рождения сотрудника |
| job | Varchar(255) |  | Должность |
| count\_orders | Int |  | Количество заказов |
| id\_address | Int | Внешний | Идентификационный номер адреса |
| id\_auth | Int | Внешний | Идентификационный номер авторизации |
| id\_passport | Int | Внешний | Идентификационный номер паспорта |

Таблица «Адреса сотрудников» хранит информацию об адресных данных сотрудников автосалона. Структура приведена в таблице 22.

Таблица 22 – Структура таблицы «Адреса сотрудников».

|  |  |  |  |
| --- | --- | --- | --- |
| Название поля | Тип | Ключ | Описание |
| id | Int | Первичный | Идентификационный номер сотрудника |
| city | Varchar(60) |  | Город сотрудника |
| street | Varchar(60) |  | Улица сотрудника |
| house | Varchar(5) |  | Номер дома сотрудника |
| apartment | Int |  | Номер квартиры сотрудника |

Таблица «Паспортные данные сотрудников» хранит информацию об паспортных данных сотрудников. Структура приведена в таблице 23.

Таблица 23 – Структура таблицы «Паспортные данные сотрудников».

|  |  |  |  |
| --- | --- | --- | --- |
| Название поля | Тип | Ключ | Описание |
| id | Int | Первичный | Идентификационный номер сотрудника |
| series | Varchar(5) |  | Серия паспорта сотрудника |
| number | Varchar(10) |  | Номер паспорта сотрудника |

Таблица «Клиенты» хранит информацию о клиентах автосалона. Структура приведена в таблице 24.

Таблица 24 – Структура таблицы «Клиента».

|  |  |  |  |
| --- | --- | --- | --- |
| Название поля | Тип | Ключ | Описание |
| id | Int | Первичный | Идентификационный номер клиента |
| last\_name | Varchar(60) |  | Фамилия клиента |
| first\_name | Varchar(60) |  | Имя клиента |
| surname | Varchar(60) |  | Отчество клиента |
| phone | Varchat(30) |  | Номер телефона клиента |
| date\_birth | Date |  | Дата рождения клиента |
| id\_address | Int | Внешний | Идентификационный номер адреса |
| id\_auth | Int | Внешний | Идентификационный номер авторизации |
| id\_passport | Int | Внешний | Идентификационный номер паспорта |

Таблица «Адреса клиентов» хранит информацию об адресных данных клиентов автосалона. Структура приведена в таблице 25.

Таблица 25 – Структура таблицы «Адреса клиентов».

|  |  |  |  |
| --- | --- | --- | --- |
| Название поля | Тип | Ключ | Описание |
| id | Int | Первичный | Идентификационный номер клиента |
| city | Varchar(60) |  | Город клиента |
| street | Varchar(50) |  | Улица клиента |
| house | Varchar(5) |  | Номер дома клиента |
| apartment | Varchar(5) |  | Номер квартиры клиента |

Таблица «Паспортные данные клиентов» хранит информацию об паспортных данных клиентов автосалона. Структура приведена в таблице 26.

Таблица 26 – Структура таблицы «Паспортные данные клиентов».

|  |  |  |  |
| --- | --- | --- | --- |
| Название поля | Тип | Ключ | Описание |
| id | Int | Первичный | Идентификационный номер клиента |
| series | Varchar(5) |  | Серия паспорта клиента |
| number | Varchar(10) |  | Номер паспорта клиента |
| issued\_by | Varchar(60) |  | Кем выдан клиента |

Таблица «Авторизации» хранит информацию обо всех пользователях автосалона. Структура приведена в таблице 27.

Таблица 27 – Структура таблицы «Авторизация».

|  |  |  |  |
| --- | --- | --- | --- |
| Название поля | Тип | Ключ | Описание |
| id | Int | Первичный | Идентификационный номер пользователя |
| login | Varchar(45) |  | Логин пользователя |
| password | Varchar(255) |  | Пароль пользователя |
| access | Tinyint |  | Доступ имеется или не имеется |
| role | Tinyint |  | Клиент или сотрудник |
| date\_regist | Timestampt |  | Дата регистрации пользователя |
| email | Varchar(60) |  | Email пользователя |

Таблица «Автомобили» хранит информацию обо всех автомобилях в автосалоне. Структура приведена в таблице 28.

Таблица 28 – Структура таблицы «Автомобили».

|  |  |  |  |
| --- | --- | --- | --- |
| Название поля | Тип | Ключ | Описание |
| id | Int | Первичный | Идентификационный номер автомобиля |
| name | Varchar(30) |  | Название автомобиля |
| engine | Varchar(30) |  | Двигатель автомобиля |

Продолжение таблицы 28

|  |  |  |  |
| --- | --- | --- | --- |
| year | Int |  | Год выпуска автомобиля |
| price | Varchar(12) |  | Стоимость автомобиля |
| color | Varchar(12) |  | Цвет автомобиля |
| complexion | Varchar(20) |  | Комплектация базовая средняя или полная |
| img | Varchar(255) |  | Фотография автомобиля |
| status | Tinyint |  | Есть в наличии или нет |
| state | Varchar(12) |  | Состояние |
| id\_model | Int | Внешний | Идентификационный номер модели автомобиля |

Таблица «Модели» хранит информацию обо всех моделях автомобилей автосалона. Структура приведена в таблице 29.

Таблица 29 – Структура таблицы «Модели».

|  |  |  |  |
| --- | --- | --- | --- |
| Название поля | Тип | Ключ | Описание |
| id | Int | Первичный | Идентификационный номер |
| model | Varchar(30) |  | Название модели |
| main\_foto | Varchar(255) |  | Главное фото модели |
| counts | Int |  | Количество автомобилей данной модели |

Структура базы данных показана на рисунке 5.



Рисунок 5 – Физическая структура базы данных

## Реализация проекта базы данных

### 3.2.1 Создание и заполнение таблиц базы данных

Для создания таблиц используется оператор CREATE TABLE. Таблицы создаются по следующим правилам:

* Имя таблицы указывается после ключевого слова CREATE TABLE (если имя состоит из нескольких слов, то его следует заключить в одинарные кавычки).
* Далее в круглых скобках следуют имена столбцов таблицы (полей), после которых указывается тип данных, которому будет принадлежать поле.
* Не обязательно: затем указывается может ли поле содержать пустые значения (NULL — может быть пустым или NOT NULL — обязательно для заполнения).
* Одно из полей назначается первичным ключом (Primary key).
* Поля отделяются запятыми.

Общая команда создания таблицы представлена на рисунке 6:

CREATE TABLE название\_таблицы

(название\_столбца1 тип\_данных атрибуты\_столбца1,

название\_столбца2 тип\_данных атрибуты\_столбца2,

................................................

название\_столбцаN тип\_данных атрибуты\_столбцаN,

атрибуты\_таблицы);

Рисунок 6 – Общая команда создания таблицы

Создание таблиц пресдтавлено на рисунках 7–18.

Создание таблицы «Заказы» представлено на рисунке 7.

[CREATE](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/create-table.html) [TABLE](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/create-table.html) `bmv\_dealership`.`orders` (

`id` INT [NOT](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_not) NULL AUTO\_INCREMENT ,

`date` TIMESTAMP [NOT](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_not) NULL,

`id\_client` INT NULL,

`id\_employee` INT NULL,

`id\_auto` INT NULL,

`id\_contact` INT NULL,

PRIMARY KEY (`id`)

Рисунок 7 – Создание таблицы «Заказы»

Создание таблицы «Контакты» пресдавлено на рисунке 8.

[CREATE](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/create-table.html) [TABLE](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/create-table.html) `bmv\_dealership`.`contacts` (

`id` INT [NOT](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_not) NULL AUTO\_INCREMENT ,

`name` VARCHAR(60) [NOT](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_not) NULL ,

`phone` VARCHAR(20) [NOT](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_not) NULL ,

`work\_time` TEXT [NOT](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_not) NULL ,

`email` VARCHAR(60) [NOT](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_not) NULL ,

`id\_address` INT [NOT](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_not) NULL ,  PRIMARY KEY (`id`));

Рисунок 8 – Создание таблицы «Контакты»

Создание таблицы «Адреса контактов» представлено на рисунке 9.

[CREATE](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/create-table.html) [TABLE](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/create-table.html) `bmv\_dealership`.`contacts\_address` (

`id` INT [NOT](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_not) NULL AUTO\_INCREMENT , `

city` VARCHAR(30) [NOT](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_not) NULL ,

`street` VARCHAR(30) [NOT](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_not) NULL ,

`house` VARCHAR(30) [NOT](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_not) NULL , PRIMARY KEY (`id`));

Рисунок 9 – Создание таблицы «Адреса контактов»

Создание таблицы «Автомобили» представлено на рисунке 10.

[CREATE](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/create-table.html) [TABLE](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/create-table.html) `bmv\_dealership`.`auto` (

`id` INT [NOT](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_not) NULL AUTO\_INCREMENT ,

`name` VARCHAR(30) [NOT](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_not) NULL ,

`engine` VARCHAR(30) [NOT](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_not) NULL ,

`year` INT [NOT](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_not) NULL ,

`price` VARCHAR(12) [NOT](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_not) NULL ,

`color` VARCHAR(20) [NOT](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_not) NULL ,

`complexion` VARCHAR(20) [NOT](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_not) NULL ,

`img` VARCHAR(255) [NOT](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_not) NULL ,

`status` TINYINT [NOT](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_not) NULL ,

`state` VARCHAR(12) [NOT](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_not) NULL ,

`id\_model` INT [NOT](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_not) NULL ,

PRIMARY KEY (`id`));

Рисунок 10 – Создание таблицы «Автомобили»

Создание таблицы «Модели» представлено на рисунке 11.

[CREATE](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/create-table.html) [TABLE](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/create-table.html) `bmv\_dealership`.`models` (

`id` INT [NOT](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_not) NULL AUTO\_INCREMENT ,

`model` VARCHAR(30) [NOT](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_not) NULL ,

`main\_foto` VARCHAR(255) [NOT](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_not) NULL ,

`counts` INT [NOT](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_not) NULL [DEFAULT](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/miscellaneous-functions.html%23function_default) '0' ,

PRIMARY KEY (`id`));

Рисунок 11 – Создание таблицы «Автомобили»

Создание таблицы «Авторизация» представлено на рисунке 12.

[CREATE](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/create-table.html) [TABLE](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/create-table.html) `bmv\_dealership`.`authorization` (

`id` INT [NOT](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_not) NULL AUTO\_INCREMENT ,

`login` VARCHAR(45) [NOT](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_not) NULL ,

`password` VARCHAR(255) [NOT](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_not) NULL ,

`access` TINYINT [NOT](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_not) NULL ,

`role` TINYINT [NOT](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_not) NULL ,

`date\_regist` TIMESTAMP [NOT](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_not) NULL ,

`email` VARCHAR(60) [NOT](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_not) NULL ,

PRIMARY KEY (`id`));

Рисунок 12 – Создание таблицы «Авторизация»

Создание таблицы «Клиенты» представлено на рисунке 13.

[CREATE](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/create-table.html) [TABLE](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/create-table.html) `bmv\_dealership`.`clients` (

`id` INT [NOT](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_not) NULL AUTO\_INCREMENT ,

`last\_name` VARCHAR(60) [NOT](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_not) NULL ,

`first\_name` VARCHAR(60) [NOT](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_not) NULL ,

`surname` VARCHAR(60) NULL ,

`date\_birth` DATE [NOT](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_not) NULL ,

`phone` VARCHAR(30) NULL ,

`count\_orders` INT [NOT](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_not) NULL [DEFAULT](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/miscellaneous-functions.html%23function_default) '0' ,

`id\_address` INT [NOT](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_not) NULL ,

`id\_auth` INT [NOT](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_not) NULL ,

`id\_passport` INT [NOT](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_not) NULL ,

PRIMARY KEY (`id`));

Рисунок 13 – Создание таблицы «Клиенты»

Создание таблицы «Адреса клиентов» представлено на рисунке 14.

[CREATE](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/create-table.html) [TABLE](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/create-table.html) `bmv\_dealership`.`clients\_address` (

`id` INT [NOT](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_not) NULL AUTO\_INCREMENT ,

`city` VARCHAR(60) NULL ,

`street` VARCHAR(50) NULL ,

`house` VARCHAR(5) NULL ,

`apartment` VARCHAR(5) NULL ,

PRIMARY KEY (`id`));

Рисунок 14 – Создание таблицы «Адреса клиентов»

Создание таблицы «Паспортные данные клиентов» представлено на рисунке 15.

[CREATE](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/create-table.html) [TABLE](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/create-table.html) `bmv\_dealership`.`clients\_address` (

`id` INT [NOT](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_not) NULL AUTO\_INCREMENT ,

`series` VARCHAR(5) NULL ,

`number` VARCHAR(10) NULL ,

`issued\_by` VARCHAR(60) NULL ,

PRIMARY KEY (`id`));

Рисунок 15 – Создание таблицы «Паспортные данные клиентов»

Создание таблицы «Сотрудники» представлено на рисунке 16.

[CREATE](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/create-table.html) [TABLE](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/create-table.html) `bmv\_dealership`.`employees` (

`id` INT [NOT](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_not) NULL AUTO\_INCREMENT ,

`last\_name` VARCHAR(60) [NOT](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_not) NULL ,

`first\_name` VARCHAR(60) [NOT](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_not) NULL ,

`surname` VARCHAR(60) NULL ,

`date\_birth` DATE NULL ,

`job` VARCHAR(255) [NOT](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_not) NULL ,

`id\_address` INT [NOT](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_not) NULL ,

`id\_auth` INT [NOT](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_not) NULL ,

`id\_passport` INT [NOT](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_not) NULL ,  PRIMARY KEY (`id`));

Рисунок 16 – Создание таблицы «Сотрудники»

Создание таблицы «Адреса сотрудников» представлено на рисунке 17.

[CREATE](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/create-table.html) [TABLE](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/create-table.html) `bmv\_dealership`.`employees\_address` (

`id` INT [NOT](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_not) NULL AUTO\_INCREMENT ,

`city` VARCHAR(60) NULL ,

`street` VARCHAR(60) NULL ,

`house` VARCHAR(5) NULL ,

`apartment` INT NULL ,

PRIMARY KEY (`id`));

Рисунок 17 – Создание таблицы «Адреса сотрудников»

Создание таблицы «Паспортные данные сотрудников» представлено на рисунке 18.

[CREATE](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/create-table.html) [TABLE](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/create-table.html) `bmv\_dealership`.`employees\_passports` (

`id` INT [NOT](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_not) NULL AUTO\_INCREMENT ,

`series` VARCHAR(5) NULL ,

`number` VARCHAR(10) NULL ,

`issued\_by` VARCHAR(60) NULL ,

PRIMARY KEY (`id`));

Рисунок 18 – Создание таблицы «Паспортные данные сотрудников»

Создание связей для таблиц представлено на рисуках 19–23.

Создание связей для таблицы «Заказы» представлено на риунке 19.

[ALTER](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/alter-table.html) [TABLE](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/alter-table.html) `orders` ADD FOREIGN KEY (`id\_auto`) REFERENCES `auto`(`id`) ON [DELETE](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/delete.html) [SET](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/set.html) NULL ON [UPDATE](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/update.html) [SET](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/set.html) NULL;

[ALTER](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/alter-table.html) [TABLE](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/alter-table.html) `orders` ADD FOREIGN KEY (`id\_client`) REFERENCES `clients`(`id`) ON [DELETE](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/delete.html) [SET](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/set.html) NULL ON [UPDATE](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/update.html) [SET](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/set.html) NULL;

[ALTER](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/alter-table.html) [TABLE](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/alter-table.html) `orders` ADD FOREIGN KEY (`id\_contact`) REFERENCES `contacts`(`id`) ON [DELETE](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/delete.html) [SET](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/set.html) NULL ON [UPDATE](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/update.html) [SET](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/set.html) NULL;

[ALTER](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/alter-table.html) [TABLE](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/alter-table.html) `orders` ADD FOREIGN KEY (`id\_employee`) REFERENCES `employees`(`id`) ON [DELETE](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/delete.html) [SET](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/set.html) NULL ON [UPDATE](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/update.html) [SET](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/set.html) NULL;

Рисунок 19 – Создание связей для таблицы «Заказы»

Создание связи для таблицы «Контакты» представлено на рисунке 20.

[ALTER](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/alter-table.html) [TABLE](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/alter-table.html) `contacts` ADD FOREIGN KEY (`id\_address`) REFERENCES `contacts\_address`(`id`) ON [DELETE](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/delete.html) CASCADE ON [UPDATE](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/update.html) CASCADE;

Рисунок 20 – Создание связей для таблицы «Контакты»

Создание связи для таблицы «Автомобили» представлено на рисунке 21.

[ALTER](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/alter-table.html) [TABLE](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/alter-table.html) `auto` ADD FOREIGN KEY (`id\_model`) REFERENCES `models`(`id`) ON [DELETE](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/delete.html) CASCADE ON [UPDATE](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/update.html) CASCADE;

Рисунок 21 – Создание связей для таблицы «Автомобили»

Создание связей для таблицы «Клиенты» представлено на рисунке 22.

[ALTER](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/alter-table.html) [TABLE](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/alter-table.html) `clientss` ADD FOREIGN KEY (`id\_address`) REFERENCES `clients\_address`(`id`) ON [DELETE](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/delete.html) CASCADE ON [UPDATE](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/update.html) CASCADE;

[ALTER](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/alter-table.html) [TABLE](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/alter-table.html) `clientss` ADD FOREIGN KEY (`id\_auth`) REFERENCES `authorization`(`id`) ON [DELETE](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/delete.html) CASCADE ON [UPDATE](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/update.html) CASCADE;

[ALTER](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/alter-table.html) [TABLE](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/alter-table.html) `clientss` ADD FOREIGN KEY (`id\_passport`) REFERENCES `clients\_passport`(`id`) ON [DELETE](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/delete.html) CASCADE ON [UPDATE](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/update.html) CASCADE;

Рисунок 22 – Создание связей для таблицы «Автомобили»

Создание связей для таблицы «Сотрудники» представлено на рисунке 23.

[ALTER](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/alter-table.html) [TABLE](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/alter-table.html) `employees` ADD FOREIGN KEY (`id\_address`) REFERENCES `employees\_address`(`id`) ON [DELETE](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/delete.html) CASCADE ON [UPDATE](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/update.html) CASCADE;

[ALTER](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/alter-table.html) [TABLE](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/alter-table.html) `employeess` ADD FOREIGN KEY (`id\_auth`) REFERENCES `authorization`(`id`) ON [DELETE](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/delete.html) CASCADE ON [UPDATE](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/update.html) CASCADE;

[ALTER](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/alter-table.html) [TABLE](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/alter-table.html) `employeess` ADD FOREIGN KEY (`id\_passport`) REFERENCES `employees\_passport`(`id`) ON [DELETE](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/delete.html) CASCADE ON [UPDATE](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/update.html) CASCADE;

Рисунок 23 – Создание связей для таблицы «Сотрудники»

Заполнение таблиц происходит с помощью оператора INSERT языка SQL. Общая команда для заполнения таблиц представлена на рисунке 24:

INSERT INTO название\_таблицы (название\_столбца\_1, название\_столбца\_2, название\_столбца\_3, …)

VALUES (знаение1, значение2, значение3, …);

Рисунок 24 – Общая команда для заполнения таблиц

Примеры частичного заполнения таблиц показаны на рисунках 25–36.

Заполнения таблицы «Адреса контактов» представлено на рисунке 25.

[INSERT](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/insert.html) INTO `contacts\_address` (`id`, `city`, `street`, `house`)

[VALUES](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/miscellaneous-functions.html%23function_values) (NULL, 'Минск', 'Машерова', '76'),

(NULL, 'Минск', 'Долгиновский тракт', '51'),

(NULL, 'Минск', 'Партизанский проспект', '91')

Рисунок 25 – Заполнение таблицы «Адреса контактов»

Заполнения таблицы «Контакты» представлено на рисунке 26.

[INSERT](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/insert.html) INTO `contacts` (`id`, `name`, `phone`, `email`, `work\_time`, `id\_address`)

[VALUES](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/miscellaneous-functions.html%23function_values) (NULL, 'Отдел продаж автомобиля', '+375294572912', 'info@autoidea.by', 'Пн-Вс: 8:00 - 20:00', '3'), (NULL, 'Отдел сервисного обслуживания', '+37529500081', 'service@autoidea.by', 'Пн-Вс: 8:00 - 20:00', '5')

Рисунок 26 – Заполнение таблицы «Контакты»

Заполнения таблицы «Модели» представлено на рисунке 27.

[INSERT](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/insert.html) INTO `models` (`id`, `model`, `main\_foto`, `counts`)

[VALUES](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/miscellaneous-functions.html%23function_values) (NULL, 'X', '1681385693\_X.jpg', '0'),

(NULL, ' M', '1681385756\_M.jpg', '0')

Рисунок 27 – Заполнение таблицы «Модели»

Заполнения таблицы «Автомобили» представлено на рисунке 28.

[INSERT](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/insert.html) INTO `auto` (`id`, `name`, `engine`, `year`, `price`, `color`, `complexion`, `img`, `status`, `state`, `id\_model`)

[VALUES](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/miscellaneous-functions.html%23function_values)

(NULL, 'X', 'Бензиновый', '2021', '110000', 'Красный', 'Базвая',  '1681642646\_iX.webp', '1', 'Новое', '65'),

(NULL, 'X M60', 'Бензиновый', '2023', '160000', 'Тифани', 'Базвая', '1681642715\_iX\_M60.webp', '1', 'Новое', '65')

Рисунок 28 – Заполнение таблицы «Автомобили»

Заполнения таблицы «Авторизации» представлено на рисунке 29.

[INSERT](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/insert.html) INTO `authorization` (`id`, `login`, `password`, `access`, `role`, `date\_regist`, `email`)

[VALUES](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/miscellaneous-functions.html%23function_values)

(NULL, 'andrei', '$2y$10$QtGuxF0YC14vvM4w1LEWDOoDoS.Wup7uF5e4BNy3fCRf7E3Oqg7bi', '1', '0', [CURRENT\_TIMESTAMP](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/date-and-time-functions.html%23function_current_timestamp), 'andrei@mail.ru'),

(NULL, 'sava', '$2y$10$7.Cu7pQPylz86clTjUxKie/sWMS66u95d35ehgLJq16YmFhvAhtFS', '1', '1', [CURRENT\_TIMESTAMP](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/date-and-time-functions.html%23function_current_timestamp), 'sava@mail.ru')

Рисунок 29 – Заполнение таблицы «Авторизации»

Заполнения таблицы «Заказы» представлено на рисунке 30.

[INSERT](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/insert.html) INTO `orders` (`id`, `date`, `id\_client`, `id\_employee`, `id\_auto`, `id\_contact`)

[VALUES](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/miscellaneous-functions.html%23function_values) (NULL, [CURRENT\_TIMESTAMP](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/date-and-time-functions.html%23function_current_timestamp), '87', '11', '131', '20'),

(NULL, [CURRENT\_TIMESTAMP](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/date-and-time-functions.html%23function_current_timestamp), '88', '10', '107', '20')

### Рисунок 30 – Заполнение таблицы «Заказы»

Заполнения таблицы «Клиенты» представлено на рисунке 31.

### [INSERT](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/insert.html) INTO `clients` (`id`, `last\_name`, `first\_name`, `surname`, `date\_birth`, `phone`, `count\_orders`, `id\_address`, `id\_auth`, `id\_passport`)

### [VALUES](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/miscellaneous-functions.html%23function_values) (NULL, 'Шемерей', 'Андрей', 'Алексеевич', '2003-01-01', '80295454546', '0', '53', '137', '49'),

### (NULL, 'Рощупкин', 'Артур', 'Максимович', '2003-09-04', '80295671321', '0', '54', '139', '51')

### Рисунок 31 – Заполнение таблицы «Клиенты»

Заполнения таблицы «Адреса клиентов» представлено на рисунке 32.

[INSERT](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/insert.html) INTO `clients\_address` (`id`, `city`, `street`, `house`, `apartment`)

[VALUES](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/miscellaneous-functions.html%23function_values) (NULL, 'Ивацевичи', 'Набережная', '21', '21'),

(NULL, 'Жлобин', 'Багратиона', '23', '1')

### Рисунок 32 – Заполнение таблицы «Адреса клиентов»

Заполнение таблицы «Паспортные данные клиентов» представлено на рисунке 33.

[INSERT](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/insert.html) INTO `clients\_passport` (`id`, `series`, `number`, `issued\_by`)

[VALUES](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/miscellaneous-functions.html%23function_values) (NULL, 'MP', '2020201', 'Ивацевичским РОВД'),

(NULL, 'MK', '6567678', 'Жлобинским РОВ')

Рисунок 33 – Заполнение таблицы «Паспортные данные клиентов»

Заполнение таблицы «Сотрудники» представлено на рисунке 34.

[INSERT](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/insert.html) INTO `employees` (`id`, `last\_name`, `first\_name`, `surname`, `phone`, `date\_birth`, `job`, `id\_address`, `id\_auth`, `id\_passport`)

[VALUES](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/miscellaneous-functions.html%23function_values) (NULL, 'Чернова', 'Яна', 'Викторовна', '80338541948', '2005-05-04' 'Менеджер', '23', '142', '32'),

(NULL, 'Титов', 'Игорь', 'Владимирович', '80294443434', '2004-01-31', 'Админ', '21', '140', '29')

Рисунок 34 – Заполнение таблицы «Сотрудники»

Заполнение таблицы «Адреса сотрудников» показано на рисунке 35.

[INSERT](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/insert.html) INTO `employees\_address` (`id`, `city`, `street`, `house`, `apartment`)

[VALUES](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/miscellaneous-functions.html%23function_values) (NULL, 'Калинковичи', 'Спортивная', '34', '31'),

(NULL, 'Барановичи', 'Строителей', '76', '75')

Рисунок 35 – Заполнение таблицы «Адреса сотрудников»

Заполнение таблицы «Паспортные данные сотрудников» представлено на рисунке 36.

[INSERT](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/insert.html) INTO `employees\_passport` (`id`, `series`, `number`, `issued\_by`)

[VALUES](http://127.0.0.1/openserver/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/miscellaneous-functions.html%23function_values) (NULL, 'MO', '3892044', 'Калинковичским РОВД'),

(NULL, 'MC', '2030405', 'Солигорским РОВД')

Рисунок 36 – Заполнение таблицы «Паспортные данные сотрудников»

### 3.2.2 Создание представлений

Представление (VIEW) – объект базы данных, являющийся результатом выполнения запроса к базе данных, определенного с помощью оператора SELECT, в момент обращения к представлению.

Представления иногда называют «виртуальными таблицами». Такое название связано с тем, что представление доступно для пользователя как таблица, но само оно не содержит данных, а извлекает их из таблиц в момент обращения к нему. Если данные изменены в базовой таблице, то пользователь получит актуальные данные при обращении к представлению, использующему данную таблицу; кэширования результатов выборки из таблицы при работе представлений не производится. При этом, механизм кэширования запросов работает на уровне запросов пользователя безотносительно к тому, обращается ли пользователь к таблицам или представлениям.

Представления могут основываться как на таблицах, так и на других представлениях, т.е. могут быть вложенными (до 32 уровней вложенности).

Общая команда создания представления представлена на рисунке 37:

CREATE VIEW название\_представления

AS SELECT столбцы\_таблиц

FROM таблицы

WHERE условие;

Рисунок 37 – Общая команда создания представления

В данной работе были созданы следующие представления:

1. Представления с запросом на выборку клиентов.
2. Представления с запросом на выборку сотрудников.
3. Представление с запросом на выборку заказов.
4. Представление с запросом на выборку автомобилей.
5. Представления с запросом о сотрудниках менеджерах.

В первом представлении демонстрируем полную информацию о каждом клиенте: ФИО клиента, дата рождения, номер телефона, фотография, город, улица, номер дома, номер квартиры, серия и номер паспорта, кем выдан паспорт, логин, роль, email, дата регистрации.

Создание представления обо всех клиентах представлено на рисунке 38.

CREATE VIEW clientsView AS

SELECT clients.id, last\_name, first\_name, surname, date\_birth, phone, city, street, house, apartment, number, series, issued\_by, login, access, role, email, date\_regist

FROM clients INNER JOIN clients\_passport ON clients.id\_passport = clients\_passport.id

INNER JOIN clients\_address ON clients.id\_address = clients\_address.id

INNER JOIN authorization ON clients.id\_auth = authorization.id;

Рисунок 38 – Создания представления «Клиенты»

Результат выполнения представлен на рисунке 39.

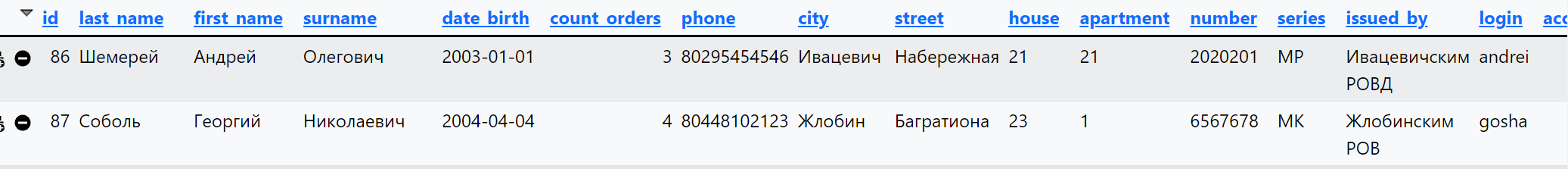


Рисунок 39 – Результат создания представления «clientsview»

Во втором представлении храниться полная информация о каждом сотруднике: ФИО сотрудника, дата рождения, номер телефона, фотография, должность, город, улица, номер дома, номер квартиры, серия и номер паспорта, кем выдан паспорт, логин, роль, email, дата регистрации.

Создание представления обо всех сотрудниках представлено на рисунке 40.

CREATE VIEW

employeesView AS

SELECT last\_name, first\_name, surname, date\_birth, phone, job, city, street, house, apartment, number, series, issued\_by, issued\_when, validity, login, access, role, email, date\_regist

FROM employees INNER JOIN employees\_passport ON employees.id\_passport = employees\_passport.id

INNER JOIN employees\_address ON employees.id\_address = employees\_address.id

INNER JOIN authorization ON employees.id\_auth = authorization.id;

Рисунок 40 – Создания представления «Сотрудники»

Результат выполнения представлен на рисунке 41.

Изображение выглядит как текст, Шрифт, линия, число

Автоматически созданное описание

Рисунок 41 – Результат создания представления «employeesview»

В третьем представлении храниться полная информация о каждом заказе: дата оформления заказа, стоимость, название автомобиля, модель автомобиля, ФИО клиента, номер телефона и email клиента.

Создание представления обо всей информации каждого заказа представлено на рисунке 42.

CREATE VIEW ordersView AS

SELECT orders.id, orders.date, price, name, model, clients.last\_name AS client\_last\_name,

CREATE VIEW ordersView AS

SELECT orders.id,

orders.id\_client AS id\_client,

orders.date, price, name, model,

clients.last\_name AS client\_last\_name,

clients.first\_name AS client\_first\_name,

clients.surname AS client\_surname,

clients.phone AS phone,

authorization.email

FROM orders INNER JOIN clients ON orders.id\_client = clients.id

INNER JOIN auto ON orders.id\_auto = auto.id

INNER JOIN authorization ON authorization.id = clients.id\_auth

INNER JOIN models ON auto.id\_model = models.id ;

Рисунок 42 – Создания представления «Заказы»

Результат выполнения представлен на рисунке 43.



Рисунок 43 – Результат создания представления «ordersview»

В четвертом представлении храниться полная информация о каждом автомобиле: модель, название, двигатель, год выпуска, стоимость, цвет, комплектация, фото, наличие и состояние.

Создание представления обо всей информации каждого автмобиля представлено на рисунке 44.

CREATE VIEW autosview AS

SELECT auto.id, model, name, `engine`, `year`, price, color, complexion, img, `status`, state

FROM auto INNER JOIN models ON auto.id\_model = models.id;

Рисунок 44 – Создание представления «Автомобилей»

Результат выполнения представлен на рисунке 45.

Изображение выглядит как текст, Шрифт, число, линия

Автоматически созданное описание

Рисунок 45 – Результат создания представления «autosview»

В пятом представлении соберем информацию обо всех менеджерах автосалона, для оформления заказов. Представление содержит: ФИО, номер телефона и его количество заказов.

Создание представления обо всех менеджерах автосалона представлено на рисунке 46.

CREATE VIEW managersview

AS SELECT employees.id, last\_name, first\_name, surname, phone, email, count\_orders

FROM employees INNER JOIN authorization ON employees.id\_auth = authorization.id

WHERE job = 'Менеджер';

Рисунок 46 – Создание представления «Менеджеры»

Результат выполнения представлен на рисунке 47.

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Рисунок 47 – Результат создания представления «managersview»

### 3.2.3 Назначение прав доступа

Рано или поздно, в любой многопользовательской базе данных возникает вопрос о разграничении прав на различные объекты, которые хранятся в этой базе. Причем разновидности этих прав могут быть самыми разными.

Вариант, когда пользователю необходимо дать право на изменение только заданного атрибута, у некоторых объектов, лишь на какие-то допустимые значения, может выглядеть вполне заурядным.

Поэтому, необходимо иметь достаточно гибкую систему прав, позволяющую легко раздавать пользователям и группам самые разнообразные права на любые объекты.

Для создания нового пользователя базы данных используют команду представленую на рисунке 48.

CREATE USER ‘имя\_пользователя’@'localhost'

IDENTIFIED BY ‘пароль’;

Рисунок 48 – Общая команда создания пользователя

Создадим трех пользователей для наглядности: boss, admin и manager.

Boss – это самый босс нашего автосалона, admin – это главный администратор, а manager – главный менеджер автосалона. В дальнейшем мы подробно рассмотрим их полномочия и назначим соответствующие права доступа.

Код создания всех трех пользователей представлен на рисунке 49.

CREATE USER 'boss'@'localhost' IDENTIFIED BY 'boss';

CREATE USER 'admin'@'localhost' IDENTIFIED BY 'admin';

CREATE USER 'manager'@'localhost' IDENTIFIED BY 'manager';

Рисунок 49 – Создание пользователей базы данных

Права доступа пользователей предоставляются с помощью команды GRANT:

GRANT права ON база\_данных.таблица TO 'логин'@'localhost';

Права доступа могут быть следующие:

* ALL – дает полный доступ к базе данных. Кстати, если база данных не определена в команде, то данный запрос даст полный доступ ко всему в MySQL, что делать не рекомендуется;
* CREATE – позволяет пользователю создавать базы данных и таблицы;
* DELETE – дает право пользователю удалять строки из таблиц;
* DROP – дает право удалять базы данных и таблица целиком;
* EXECUTE – дает право пользователю выполнять хранимые процедуры;
* GRANT OPTION – с этой опцией пользователь сможет давать права (или удалять) другим пользователям;
* INSERT – это право на добавление новых строк в таблицу;
* SELECT – самое распространенное право – парсить (извлекать) данные из SQL для чтения;
* SHOW DATABASES - этому пользователю можно будет смотреть на список баз данных;
* UPDATE – дает право пользователю изменять текущие строки в таблице;

Итак, назначим теперь права доступа трем нашим пользователям. У пользователя boss будут все полномочия, поскольку он на правах главного администратора может просматривать информацию об автомобилях, сотрудниках, клиентах, заказах, а также контакты автосалона.

Назначение прав доступа для босса автосалон показано на рисунке 50.

GRANT ALL

ON bmw\_dealership.\*

TO 'boss'@'localhost';

Рисунок 50 – Назначение прав доступа для босса автосалона

Далее рассмотрим пользователя admin: человек с такой должностью ведет оперативный о контактной информации, а также ведет учет всех сотрудниках и клиентах автосалона. Поэтому у пользователя admin должен быть доступ ко всем таблицам, но без возможности удаления таблиц и базы данных, а также создания новых сотрудников.

Назначение прав доступа для главного администратора представлено на рисунке 51.

GRANT SELECT, INSERT, DELETE, UPDATE

ON bmw\_dealership\*

to 'admin'@'localhost';

Рисунок 51 – Назначение прав доступа для главного администратора

И, наконец, пришла очередь менеджера. В его обязанности входит продажа автомобилей. Это значит, что человек с такой должностью должен иметь право просматривать, менять, заполнять и удалять записи из таблиц заказы, модели и автомобили.

Назначение прав доступа для менеджера представлено на рисунке 52.

GRANT SELECT

ON autosview

TO 'manager'@'localhost';

GRANT SELECT

ON ordersview

TO 'manager'@'localhost';

GRANT INSERT,SELECT,UPDATE, DELETE

ON auto

TO 'manager'@'localhost';

GRANT INSERT,SELECT,UPDATE, DELETE

ON models

TO 'manager'@'localhost';

GRANT INSERT,SELECT,UPDATE, DELETE

ON orders

TO 'manager'@'localhost';

Рисунок 52 – Назначение прав доступа для менеджера

Итоговая таблица пользователей представлена на рисунке 53.

Изображение выглядит как снимок экрана, текст, линия, число

Автоматически созданное описание

Рисунок 53 – Итоговая таблица пользователей

### 3.2.4 Создание индексов

Индекс – объект базы данных, создаваемый с целью повышения производительности поиска данных. Таблицы в базе данных могут иметь большое количество строк, которые хранятся в произвольном порядке, и их поиск по заданному критерию путём последовательного просмотра таблицы строка за строкой может занимать много времени. Для повышения производительности поиска данных могут быть проиндексированы некоторые столбцы таблиц.

Для создания индекса в таблице используется следующая команда (не допускаются повторяющиеся значения) представленая на рисунке 54.

CREATE UNIQUE INDEX index\_name

ON table\_name (column1, column2, ...);

Рисунок 54 – Общая команда создания индексов

Типы MySql индексов:

PRIMARY KEY - Первичный ключ: основной ключ, который позволяет хранить свои значения как уникальные записи таблицы. Данный тип ключа должен присутствовать в таблице в единственном экземпляре. Обычно данный тип ключа определяют колонке с наименованием id.

UNIQUE - Уникальный ключ: Частично похож на первичный ключ, за счет того, что значения колонки таблицы должны быть уникальными (не должны повторяться) и не должны быть равны NULL.

В работе были проиндексированы следующие поля:

* + - 1. Фамилия клиента в таблице «Clients». Код индексирования представления представлен на рисунке 55:

CREATE INDEX last\_name

ON clients(last\_name)

Рисунок 55 – Код индексирования в таблице «Clients»

* + - 1. Фамилия сотрудника в таблице «Employees». Код индексирования представлен на рисунке 56:

CREATE INDEX last\_name

ON employees(last\_name)

Рисунок 56 – Код индексирования в таблице «Employees»

* + - 1. Цена за авто в таблице «Auto». Код индексирования представлен на рисунке 57:

CREATE INDEX price

ON auto(price)

Рисунок 57 – Код индексирования в таблице «Auto»

* + - 1. Количество автомобилей данной модели в таблице «Models». Код индексирования представлен на рисунке 58:

CREATE INDEX model

ON models(model)

Рисунок 58 – Код индексирования в таблице «Models»

## 3.2.5 Создание триггеров

Для полноценного функционирования базы данных были созданы несколько триггеров.

Создание триггера «insert\_auto\_update\_counts». При добавлении нового автомобиля в таблицу «auto», исходя из его модели в таблице «models» в поле соответствующей модели, в столбце «counts» прибавляется +1. Код создания триггера «insert\_auto\_update\_counts» представлен на рисунке 59.

CREATE TRIGGER insert\_auto\_update\_count

AFTER INSERT ON auto

FOR EACH ROW

UPDATE models

SET models.counts = models.counts + 1

WHERE models.id = NEW.id\_model

Рисунок 59 – Код создания триггера «insert\_auto\_update\_counts»

Создание триггера «delete\_auto\_update\_counts. При удалении автомобиля из таблицы «auto», исходя из его модели в таблице «models» в поле соответствующей модели, в столбце «counts» отнимается -1. Код создания триггера «delete\_auto\_update\_counts» представлен на рисунке 60.

CREATE TRIGGER delete\_auto\_update\_count

AFTER DELETE ON auto

FOR EACH ROW UPDATE models

SET models.counts = models.counts - 1

WHERE models.id = OLD.id\_model

Рисунок 60 – Код создания триггера «delete\_auto\_update\_counts»

Создание триггера «delete\_auto\_update\_counts». При удалении автомобиля из таблицы «auto», исходя из его модели в таблице «models» в поле соответствующей модели, в столбце «counts» отнимается -1. Код создания триггера «delete\_auto\_update\_counts» представлен на рисунке 61.

CREATE TRIGGER delete\_auto\_update\_count

AFTER DELETE ON auto

FOR EACH ROW

UPDATE models

SET models.counts = models.counts - 1

WHERE models.id = OLD.id\_model

Рисунок 61 – Код создания триггера «delete\_auto\_update\_counts»

Создание триггера «insert\_update\_price». При добавлении нового автомобиля с фиксированной ценой, к цене автоматически добавляется 3000 как налог. Код создания триггера «insert\_update\_counts» представлен на рисунке 62.

CREATE TRIGGER insert\_update\_price

BEFORE INSERT ON auto

FOR EACH ROW

SET NEW.price = NEW.price + 3000

Рисунок 62 – Код создания триггера «insert\_update\_counts»

Создание триггера «update\_update\_price». При добавлении нового автомобиля с фиксированной ценой, к цене автоматически добавляется 3000 как налог. Код создания триггера «update\_update\_counts» представлен на рисунке 63.

CREATE TRIGGER update\_update\_price

BEFORE UPDATE ON auto

FOR EACH ROW

SET NEW.price = NEW.price + 3000

Рисунок 63 – Код создания триггера «update\_update\_counts»

Cоздадим триггеры «delete\_client\_address», «delete\_client\_passport», «delete\_client\_auth» для удаления всех данных о клиенте, чтобы при удалении клиента, также удалялись его адресные данные, паспортные данные и данные для авторизации. Код ссоздания триггеров представлен на рисунках 64–66.

CREATE TRIGGER delete\_client\_address

BEFORE DELETE

ON clients

FOR EACH ROW

DELETE FROM clients\_address

WHERE id = OLD.id\_address

Рисунок 64 – Код создания триггера «delete\_client\_address»

CREATE TRIGGER delete\_client\_passport

BEFORE DELETE

ON clients

FOR EACH ROW

DELETE FROM clients\_passport

WHERE id = OLD.id\_passport

Рисунок 65 – Код создания триггера «delete\_client\_passport»

CREATE TRIGGER delete\_client\_auth

BEFORE DELETE

ON clients

FOR EACH ROW

DELETE FROM authorization

WHERE id = OLD.id\_auth

Рисунок 66 – Код создания триггера «delete\_client\_auth»

Cоздадим триггеры «delete\_employee\_address», «delete\_employee\_passport», «delete\_employee\_auth» для удаления всех данных о клиенте, чтобы при удалении клиента, также удалялись его адресные данные, паспортные данные и данные для авторизации. Код ссоздания триггеров представлен на рисунках 67–69.

CREATE TRIGGER delete\_employee\_address

BEFORE DELETE ON emloyees

FOR EACH ROW

DELETE FROM employees\_address

WHERE id = OLD.id\_address

Рисунок 67 – Код создания триггера «delete\_employee\_address»

CREATE TRIGGER delete\_employee\_passport

BEFORE DELETE ON employees

FOR EACH ROW

DELETE FROM employees\_passport

WHERE id = OLD.id\_passport

Рисунок 68 – Код создания триггера «delete\_employee\_passport»

CREATE TRIGGER delete\_employee\_auth

BEFORE DELETE ON employees

FOR EACH ROW

DELETE FROM authorization

WHERE id = OLD.id\_auth

Рисунок 69 – Код создания триггера «delete\_employee\_auth»

Cоздадим триггер «delete\_contact\_address» для удаления контактных данных, чтобы при удалении контакты, также удалялись его адресные данные. Код создания триггера представлен на рисунках 70.

CREATE TRIGGER delete\_contact\_address

BEFORE DELETE ON contacts

FOR EACH ROW

DELETE FROM contacts\_address

WHERE id = OLD.id\_address

Рисунок 70 – Код создания триггера «delete\_contact\_address»

## 3.3 Структура приложения

У многих организаций есть свой веб-сайт, с помощью которого сотрудники получают доступ к базам данных своей компании, а администрация может легко и удобно работать с информацией о сотрудниках. В нашем случае веб-сайт будет создан на языке программирования PHP, а также c использованием HTML, SCSS, GULP, JavaScript с помощью средства разработки Visual Code.

Для начала создадим стуктуру проектных папок, для удобства работы с большим количеством файлов. Затем настроим GULP-сборку, для оптимизации файлов и улучшения загрузки веб-сайта в будущем.

Напишем для начала frontend часть веб-сайта, для этого создадим необходимое количество файлов для каждой страницы, отрисуем разметку, пропишем стили и настроим визуальные эффекты с помощью JS.

Далее перейдем в backend часть.Для связи базы данных с веб-сайтом, запустим OpenServer и произведем подключение к базе данных. Код подключения к базе данных представлен на рисунке 71.

$driver = "mysql";

$host = "localhost";

$db\_name = "bmv\_dealership";

$db\_user = "root";

$db\_pass = "";

$charset = "utf8";

$options = [PDO::ATTR\_ERRMODE => PDO::ERRMODE\_EXCEPTION,

PDO::ATTR\_DEFAULT\_FETCH\_MODE => PDO::FETCH\_ASSOC];

try {

$pdo = new PDO(

"$driver:host=$host; dbname=$db\_name; charset=$charset",

$db\_user, $db\_pass, $options

);

} catch(PDOException $i) die("Ошибка подключения к базе данных");

Рисунок 71 – Подключение к базе данных

В классе Order описываются функции, которые будут срабатывать по определенным ссылкам при запуске приложения. Это такие функции как добавление заказа сотрудником, добавление заказа клиентом, удаление заказа. На рисунке 72 показан класс Order.

class Order {

public $NEW\_AUTO = 19;

public $OLD\_AUTO = 20;

public $CLIENT = 0;

public $NEW = 'Новое';

//Добавление заказа клиентом

public function addOrderClient(){…}

//Добавление заказа сотрудником

public function addOrderEmploee(){…}

//Удаление заказа

public function deleteOrder($id) {…}

}

Рисунок 72 – Класс Order

В классе User описываются функции, которые будут срабатывать по определенным ссылкам при запуске приложения. Это такие функции как создание пользовательской сессии для авторизации, регистрация, авторизация, изменение пароля, редактирование персональных данных, удаление аккаунта, добавление заказа, удаление заказа. На рисунке 73 показан класс User.

class User {

  public $ACCESS = 1;

  public $NO\_ACCESS = 0;

  public $CLIENT = 0;

  public $EMPLOYEE = 1;

//Создаем сессию для авторизации

  public function userAuth($arr) {…}

//Регистрация

  public function registration() {…}

//Авторизация

  public function authorization() {…}

//Изменение пароля

  public function editPassword($id) {…}

//Редактирование персональных данных

  public function updateUser() {…}

//Удаление аккаунта

  public function deleteUser($id) {…}

//Добавление заказа

  public function addOrder() {…}}

Рисунок 73 – Класс User

В классе Models описываются функции, которые будут срабатывать по определенным ссылкам при запуске приложения. Это такие функции как добавление, удаление и редактиование модели. На рисунке 74 показан класс Model.

class Model {

  //Добавление модели

  public function addModel() {…}

//Редактирование модели

  public function updateModel() {…}

//Удаление модели

  public function deleteModel($id) {…}

}

Рисунок 74 – Класс Model

В классе Contact описываются функции, которые будут срабатывать по определенным ссылкам при запуске приложения. Это такие функции как добавление, удаление и редактиование модели. На рисунке 75 показан класс Contact.

class Contact {

  //Добавление контакта

  public function addContact() {…}

//Редактирование контакта

  public function updateContact() {…}

//Удаление контакта

  public function deleteContact($id) {…}

}

Рисунок 75 – Класс Contact

В классе Auto описываются функции, которые будут срабатывать по определенным ссылкам при запуске приложения. Это такие функции как добавление, удаление, редактиование модели и редактирование статуса модели. На рисунке 76 показан класс Auto.

class Auto {

  public $AVAILABLE = 1;

  public $NO\_AVAILABLE = 0;

  //Добавление авто

  public function addAuto() {…}

  //Редактирование авто

  public function updateAuto() {…}

//Редактирование статуса авто

  public function updateStatusAuto($id) {…}

//Удаление авто

  public function deleteAuto($id) {…}

}

Рисунок 76 – Класс Auto

В классе Client описываются функции, которые будут срабатывать по определенным ссылкам при запуске приложения. Это такие функции как добавление, удаление, редактиование клиента и редактирование доступа клиента. На рисунке 77 показан класс Client.

class Client {

  public $ACCESS = 1;

  public $NO\_ACCESS = 0;

  public $CLIENT = 0;

  public $errMsg = [];

  //Добавление клиента

  public function addClient() {…}

//Редактирование клиента

  public function updateClient() {…}

  //Редактирование доступа клиента

  public function updateStatusClient($id) {…}

  //Удаление клиента

  public function deleteClient($id) {…}

}

Рисунок 77 – Класс Client

В классе Employee описываются функции, которые будут срабатывать по определенным ссылкам при запуске приложения. Это такие функции как добавление, удаление, редактиование сотрудника и редактирование доступа сотрудника. На рисунке 78 показан класс Employee.

class Employee {

  public $ACCESS = 1;

  public $NO\_ACCESS = 0;

  public $ADMIN = 1;

  public $errMsg = [];

  //Добавление сотрудника

  public function addEmployee() {…}

//Редактирование сотрудника

  public function updateEmployee() {…}

//Редактирование доступа сотрудника

  public function updateStatusEmployee($id) {…}

//Удаление сотрудника

  public function deleteEmployee($id) {…}

}

Рисунок 78 – Класс Employee

После создания классов и файлов .php для остальных отношений БД, приложение готово к запуску. Всякий запуск приложения начинается с авторизации.

Функция авторизации пользователя заключается в проверке существования в БД пользователей введённого email и соответствующего ему пароля. Также авторизация пользователя подразумевает получение его роли из отношения User.

После успешной авторизации пользователя создаётся пользовательская сессия с соответствующими привилегиями, согласно роли пользователя.

# Тестирование

Рассмотрим несколько исключительных ситуаций, показанных на рисунках 79–86, которые могут быть вызваны при некорректном вводе данных.

Рассмотрим случаи, когда пользователь которого нет в базе данных, пытается войти. Результат обработки данной ситуации показан на рисунках 87-96.

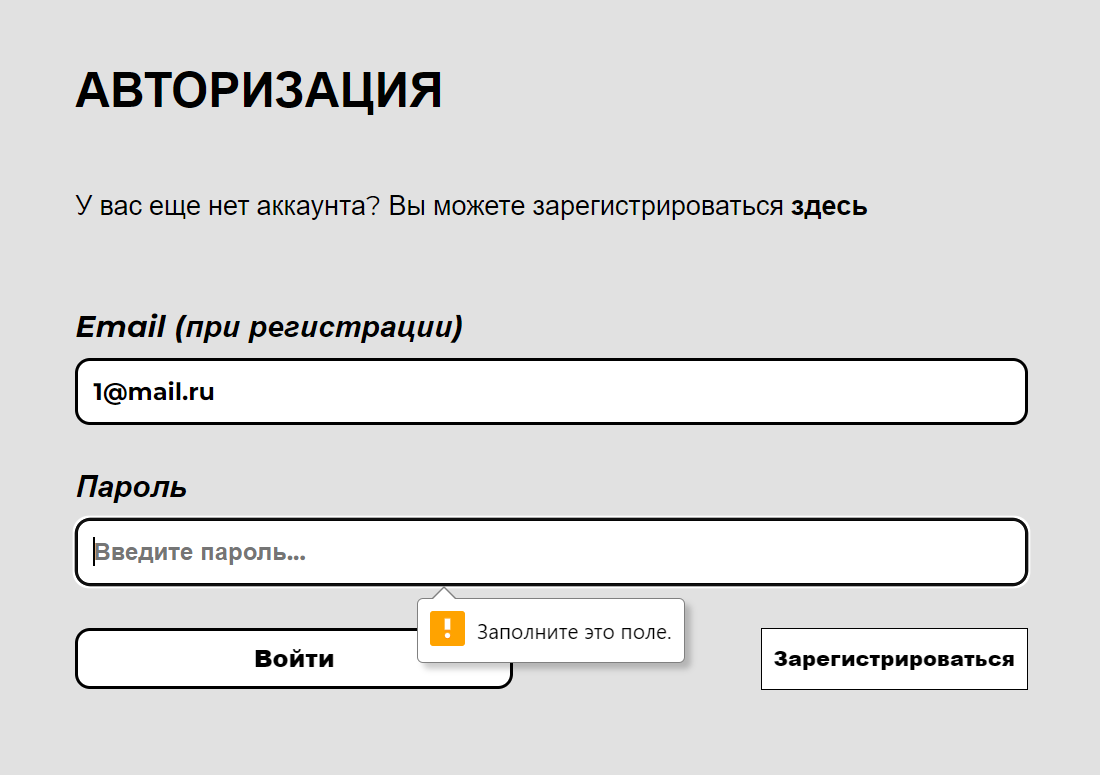


Рисунок 79 – Пример входа, когда не все поля заполнены



Рисунок 80 – Пример входа, когда неправильно введен логин или пароль

Рассмотрим случаи, когда пользователь пытается зарегистрироваться. Результат обработки данной ситуации показан на рисунках 81–82.

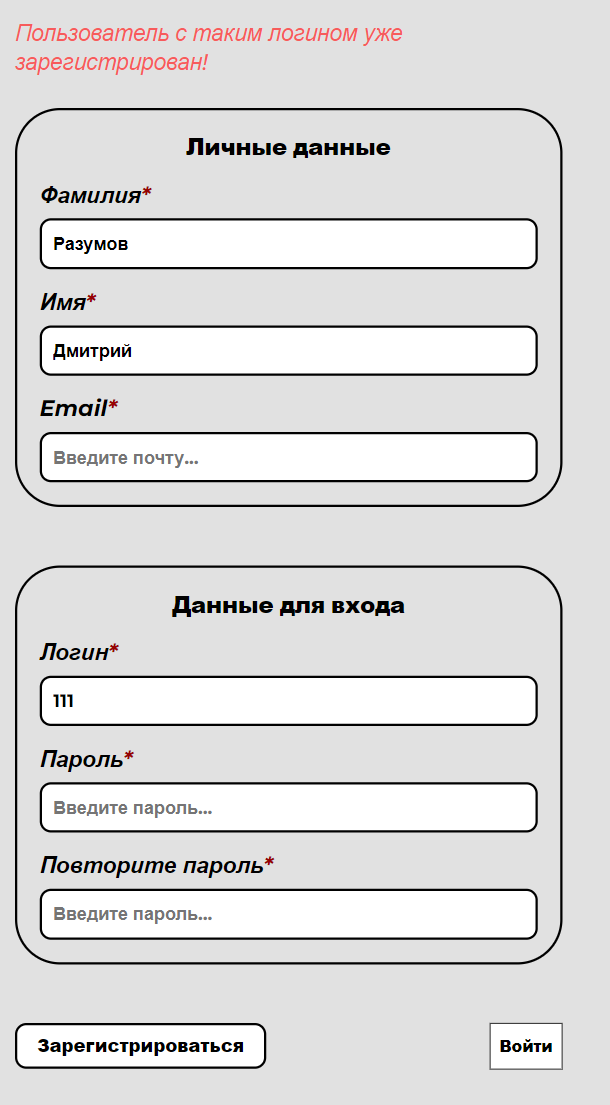


Рисунок 81 – Пример регистрации, когда пользователь с таким логином уже существует в базе данных

Рассмотрим случай, когда пользователь меняет пароль, но пароли не совпадают. Результат обработки данной ситуации представлен на рисунке 82.

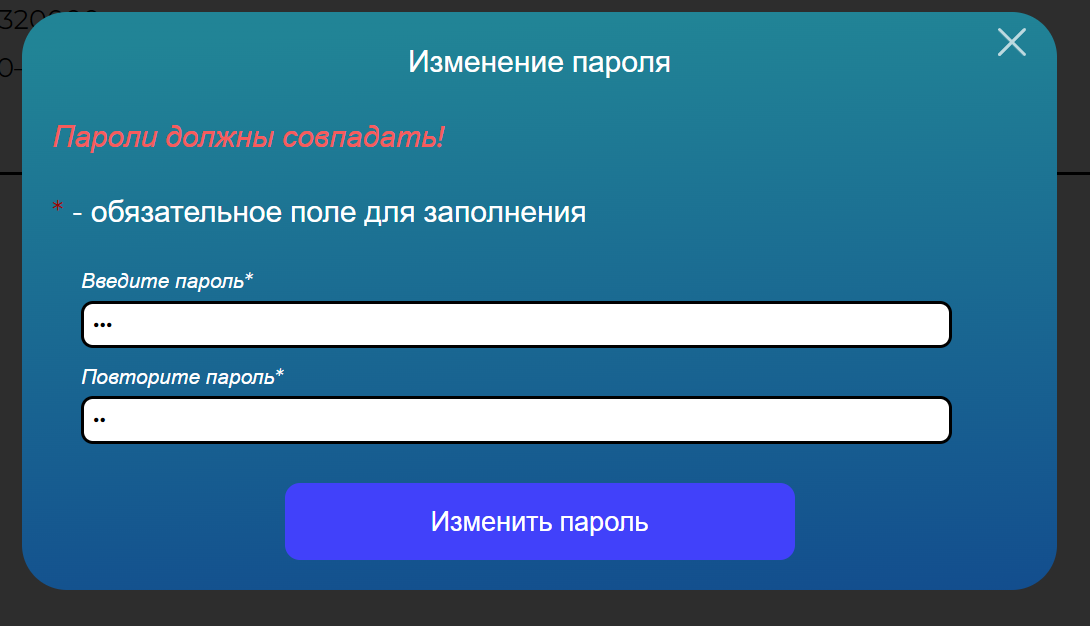


Рисунок 82 – Пример, когда пользователь меняет пароль

Пример поиска автомобиля, когда по данному запросу ничего не найдено. Результат обработки данной ситуации представлен на рисунке 83.

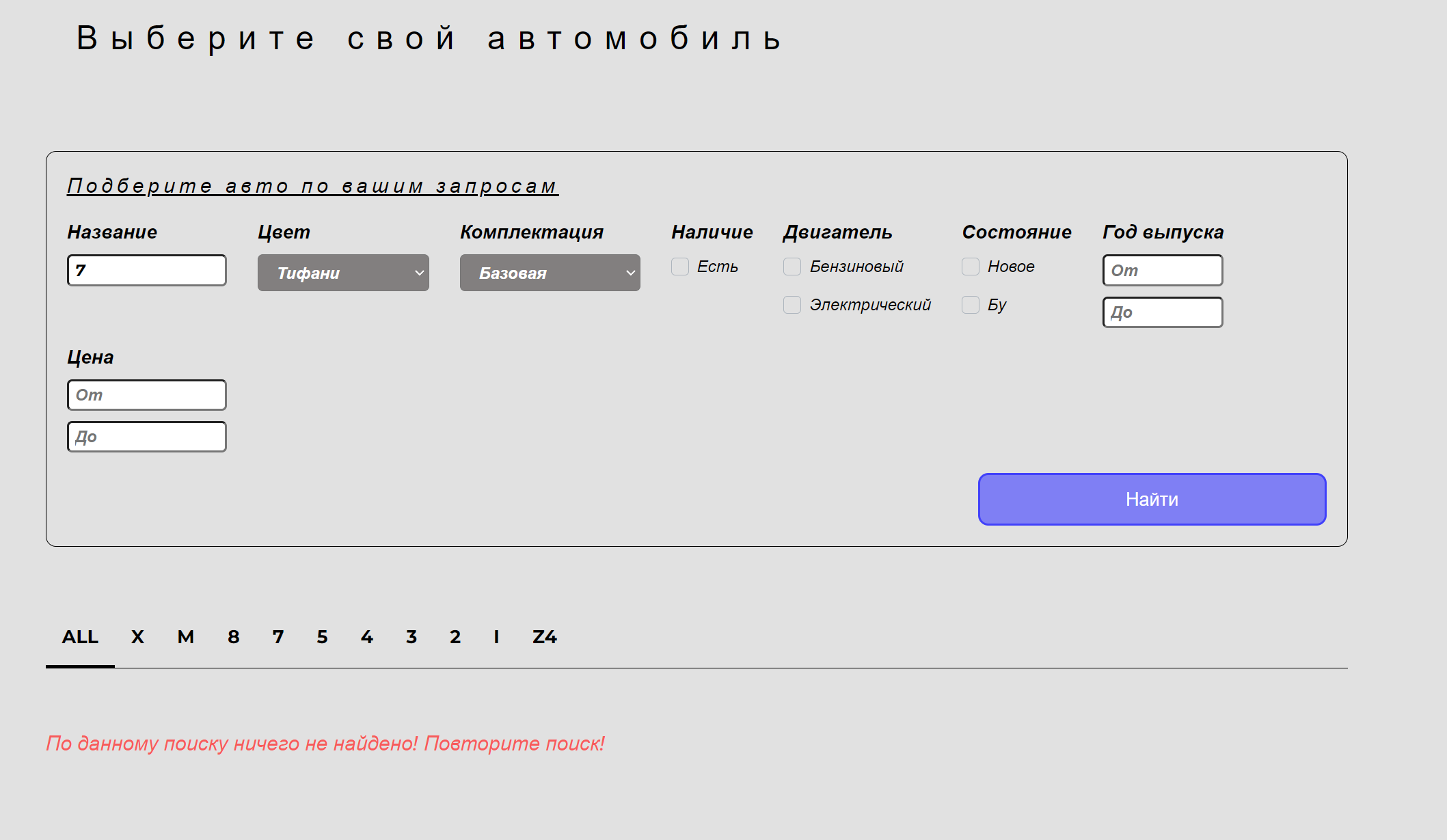


Рисунок 83 – Пример неудачного поиска

Пример неудачного поиска в админ панеле. Результат обработки данной ситуации представлен на рисунке 84.

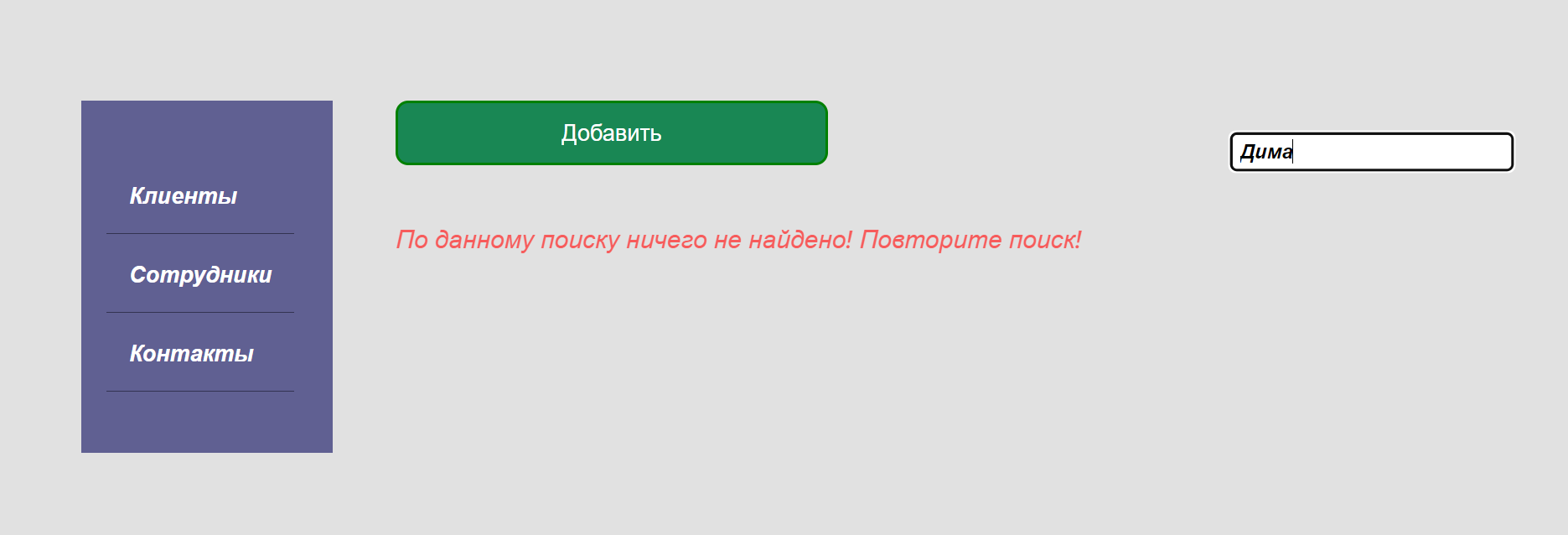


Рисунок 84 – Пример неудачного поиска в админ панеле

Пример добавленя клиента или сотрудника при вводе логина или email, которые уже имеются в базе данных. Результат обработки данной ситуации представлен на рисунке 85.

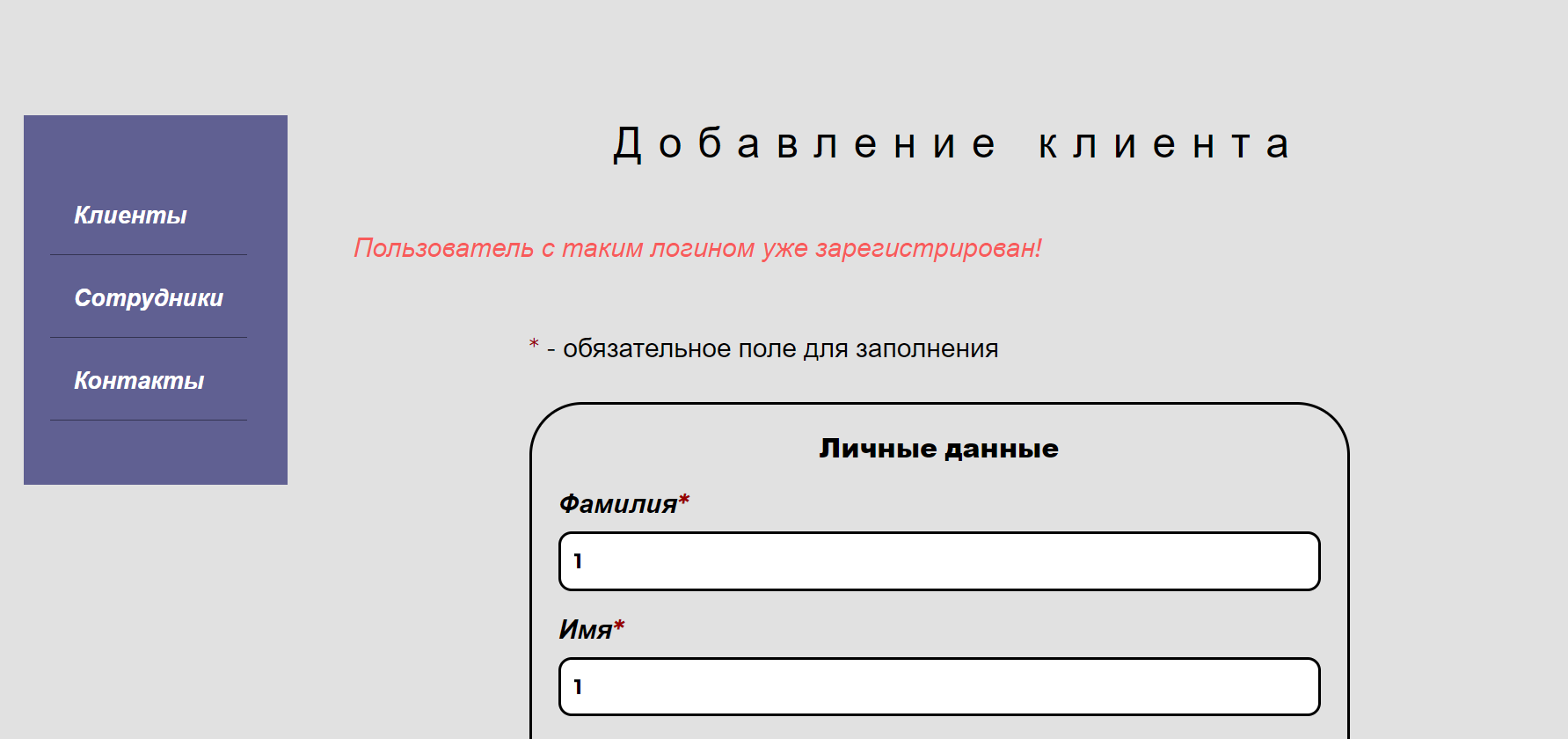


Рисунок 85 – Пример неудачного добавления пользователя

Пример заполнения каких-либо данных, когда поле является обязательным, но оно не заполнено. Результат обработки данной ситуации представлен на рисунке 86.

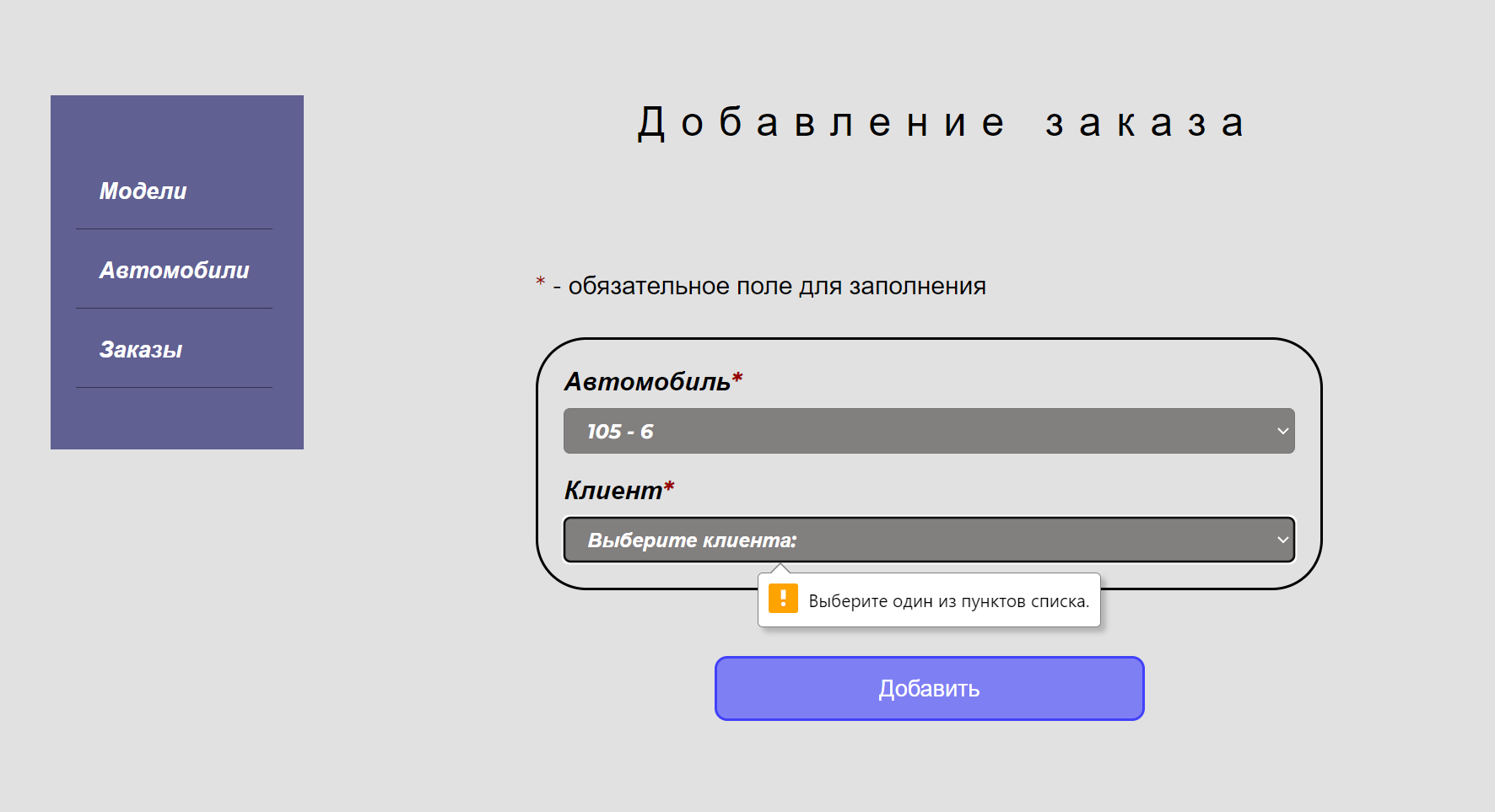


Рисунок 86 – Пример незаполненого обязательного поля

Если у пользователя не заполнены все персональные данные, то он не может совершить заказ, до тех пор, пока не заполнет всю личные данные. Результат обработки данной ситуации представлен на рисунке 87.

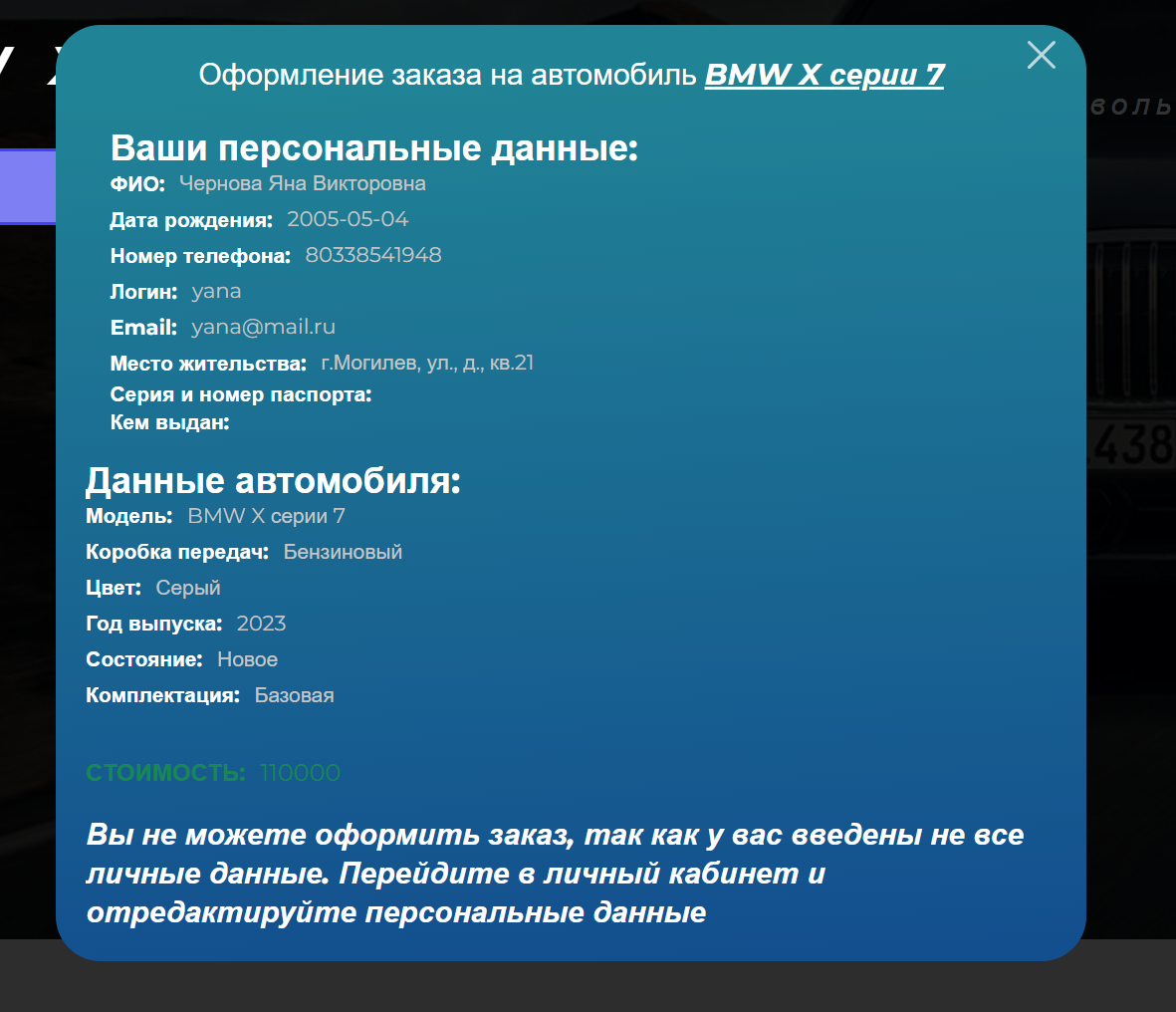


Рисунок 87 – Пример оформления заказа, когда заполнены не все персональные данные

Далее рассмотрим несколько ситуаций работы веб-приложения в обычном режиме, представленных на рисунках 88–96.

Расмотрим, как реализовано редактирование данных клиента. Данные таблицы клиенты показано на рисунке 88.

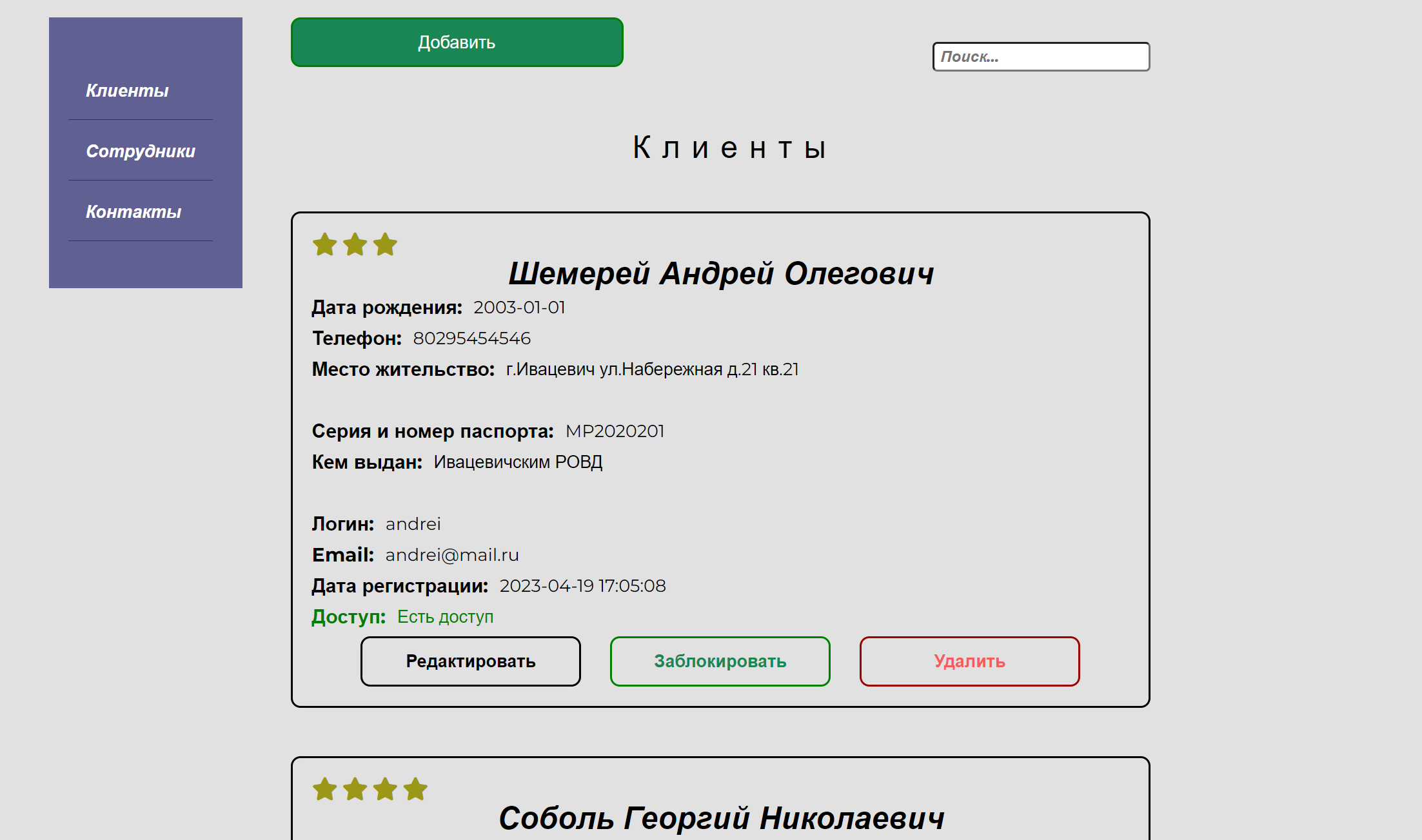


Рисунок 88 – Пример вывода клиентов

После перехода на данную страницу пользователь выбирает клиента и нажимает кнопку «Редактировать». После нажатия переходим на страницу редактирования данного клиента, показанную на рисунке 89.

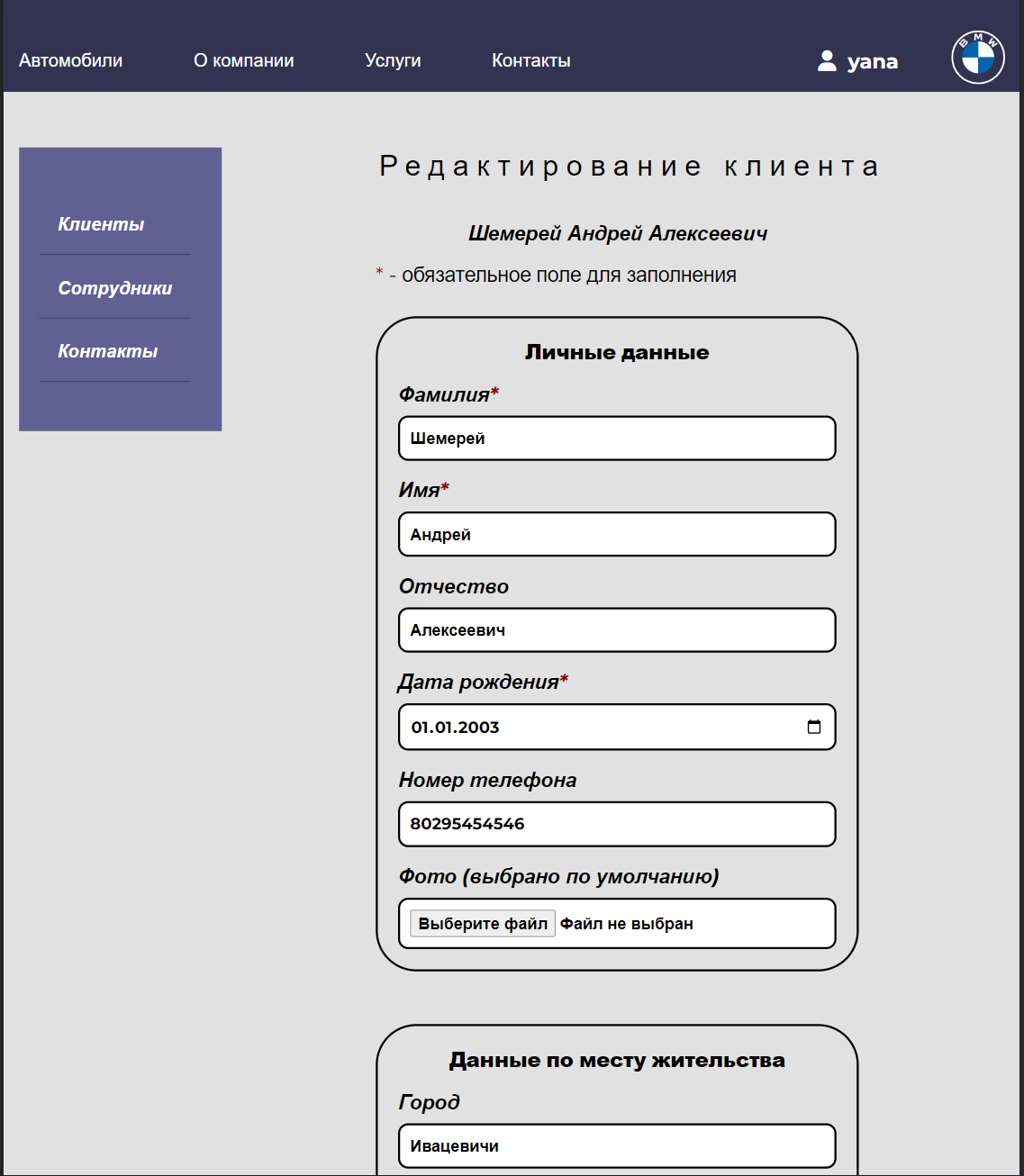


Рисунок 89 – Редактирование данных о клиенте

После перехода на данную страницу в полях ввода по умолчанию стоят старые данные, пользователь выбирает, что хочет изменить и нажимает кнопку «Сохранить». Изменим, например, отчество. Результат показан на рисунке 90.

Изображение выглядит как текст, снимок экрана, программное обеспечение, веб-страница

Автоматически созданное описание

Рисунок 90 – Результат редактирования данных о клиенте

Аналочиный вывод подробной информации о работниках, контактах, моделях, авто, заказах, но в заказах не предусмотрено редактирование.

Расмотрим, как реализовано оформление заказа клиентов. Данные таблицы автомобили показано на рисунке 91.

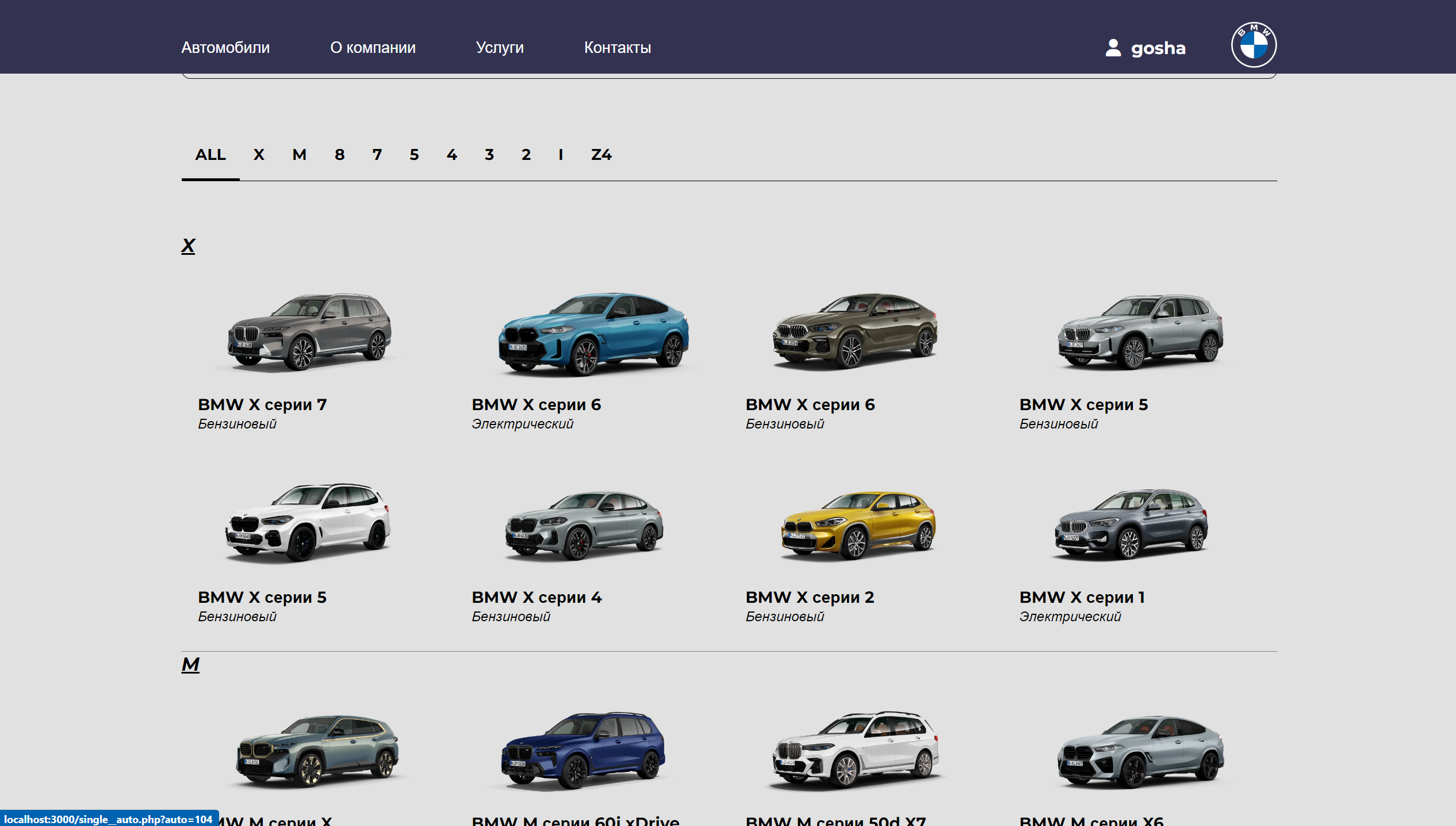


Рисунок 91 – Пример вывода таблицы автомобили

При выборе какого-либо автомобиля мы переходим на персональную страницу этого автомобиля, где находится главное фото данной марки авто, данные о данном автомобиле, маленькое фото, и кнопки для оформления заказ. Пример персональной страницы автомобиля показан на рисунке 92.



Рисунок 92 – Пример вывода персональной страницы каждого автомобиля

При нажатии на кнопку «Оформить заказ», всплывает окно для оформления заказа, показанное на рисунке 93.

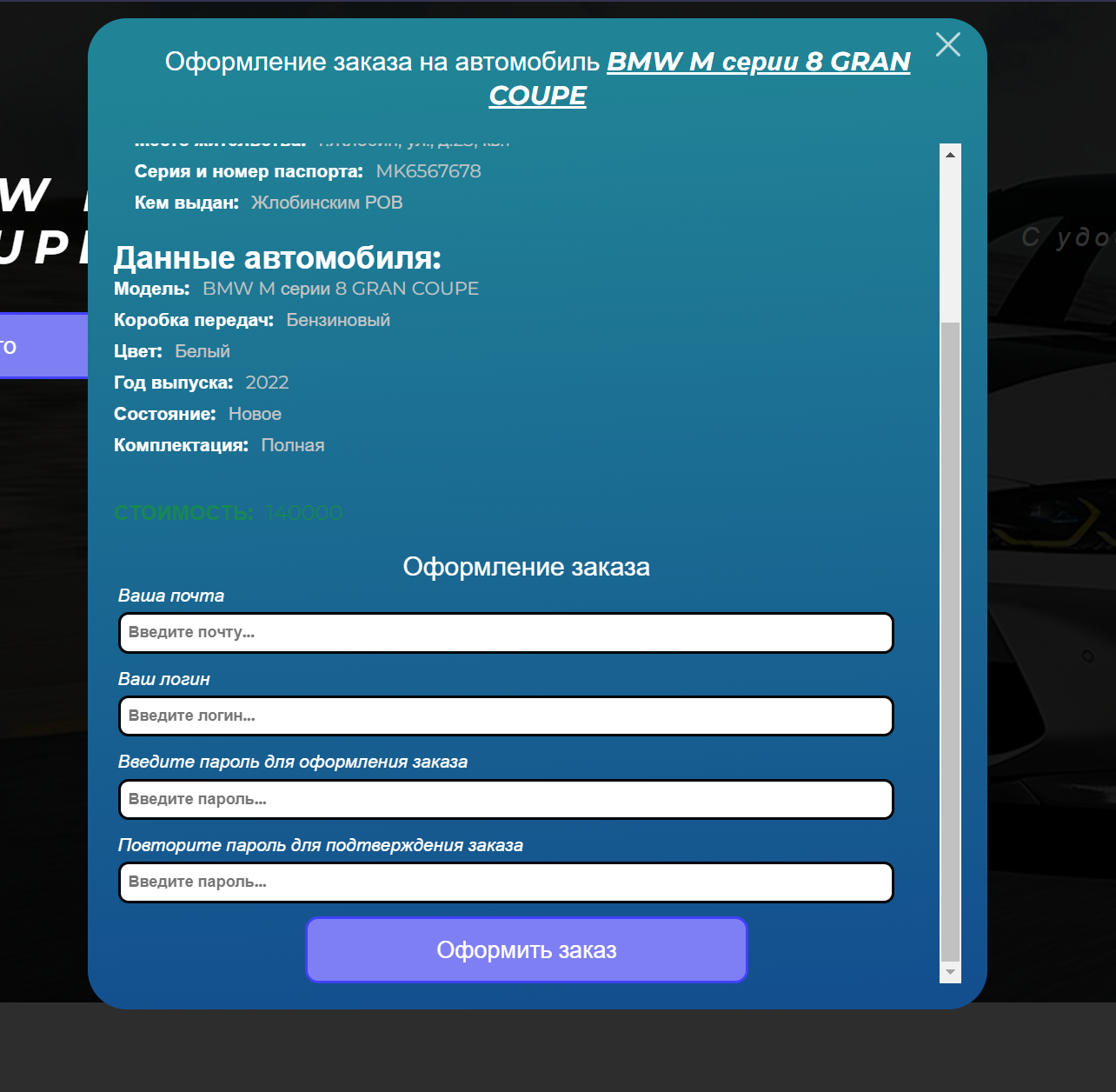


Рисунок 93 – Пример оформления заказа

Пример заполения поля email, когда email не соответствует email. Результат обработки данной ситуации представлен на рисунке 94.

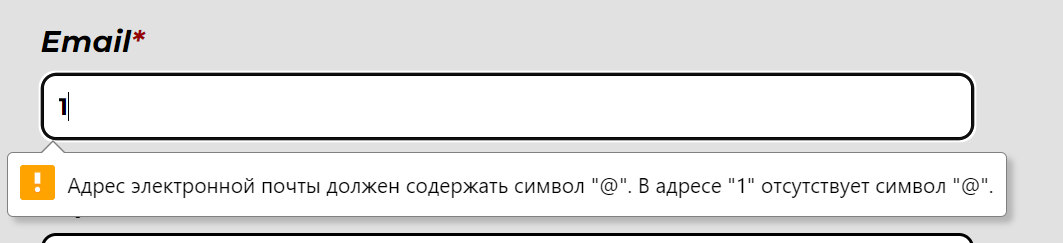


Рисунок 94 – Пример неправильного ввода email

После успешного оформления заказа, в личном кабинете пользователя, а также в «Панели менеджера» добавиться данный заказ. Пример заказов каждого пользователя представлен на рисунке 95.

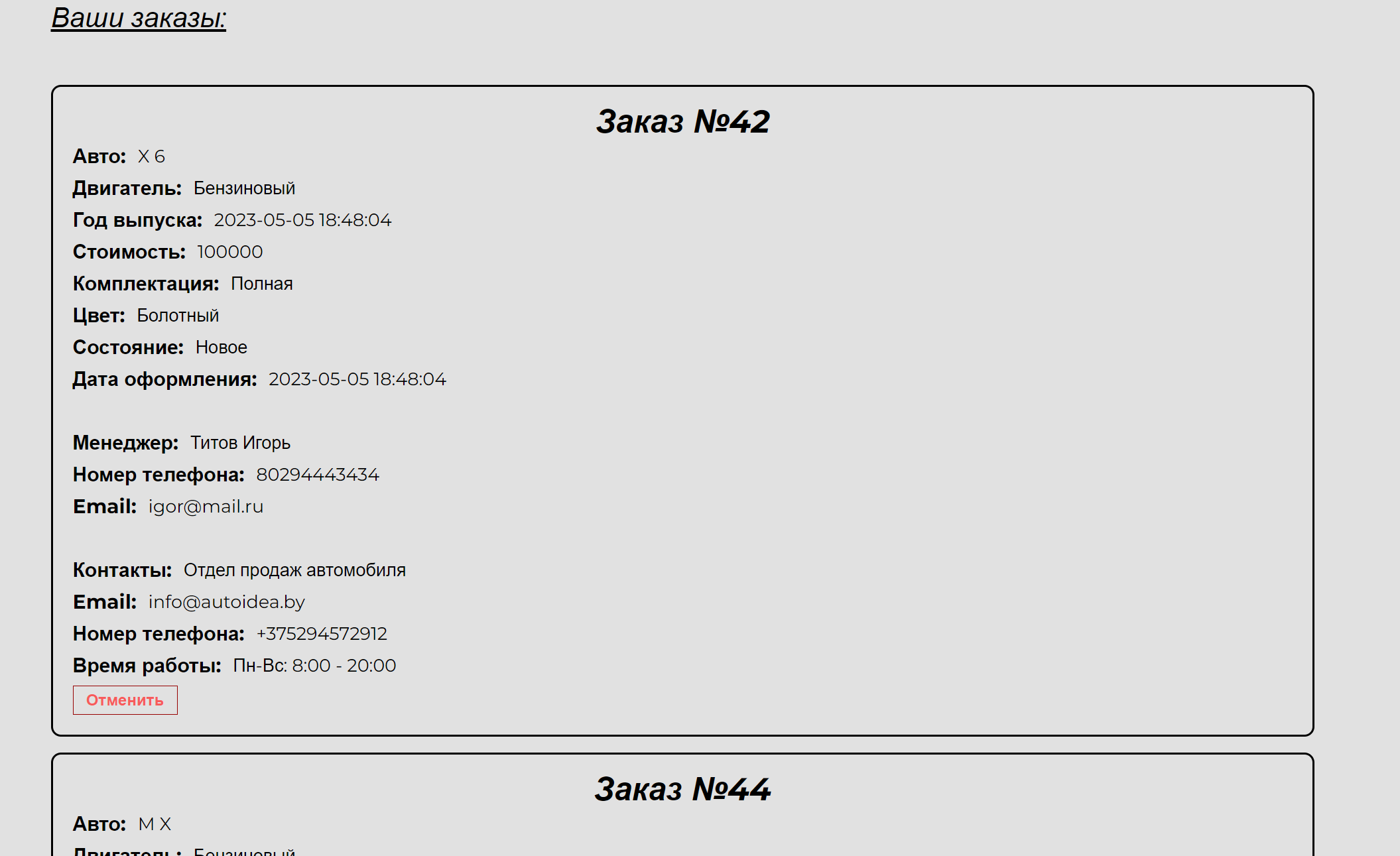


Рисунок 95 – Вывод заказов каждого клиента в личном кабинете

На рисунке 96 показан пример поиска авто, по выбранным критериям.

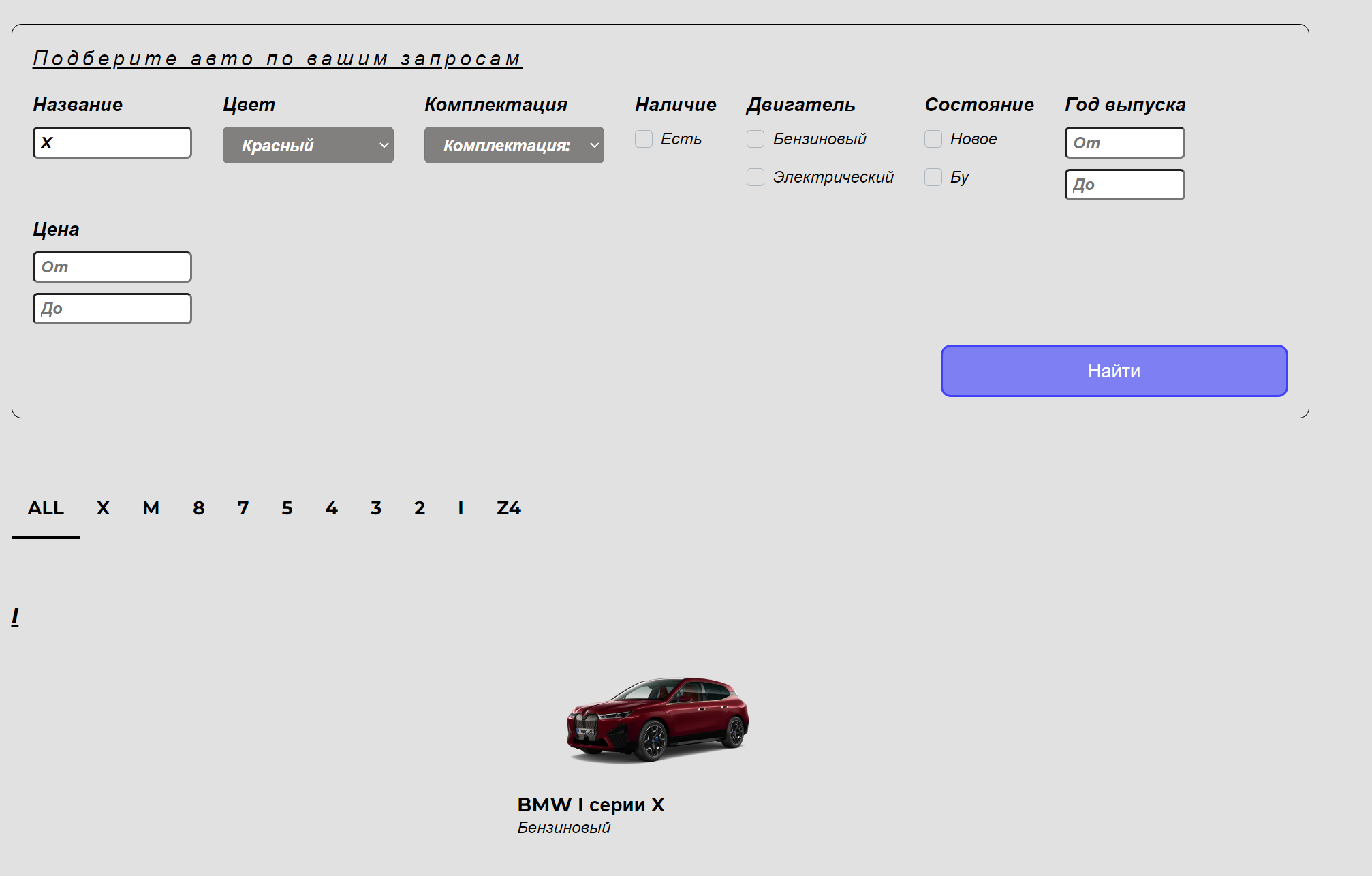


Рисунок 96 – Пример поиска автомобиля

# описание применения

Данный сайт (интернет-магазин) был создан на языке программирования PHP, JavaScript, а также с использованием SCSS, HTML, а также сборщика проектов GULP, поэтому для его запуска на компьютере необходимо установить и запустить сервер Open Server Panel. Определенный тип операционной системы не требуется.

Сайт прост в использовании: в нем имеется хедер-меню со ссылками, при нажатии на которые будет осуществляться переход на нужную страницу или кнопка в меню которого можно зайти в свой аккаунт или зарегистрироваться.

Данное приложение может быть полезно работникам, а также клиентам или просто посетителям сайта данного автосалона.

Администраторы могут со всеми удобствами и комфортом работать с данными о сотрудниках, клиентах и контактными данными автосалона. Он может при необходимости отредактировать их личные и персональные данные, добавить, удалить, а также просмотреть всю необходимую личную и персональную информацию, получить данные об их заказах с помощью поиска и сортировки.

Менеджер, работающий в автосалоне, сможет с помощью данного сайта добавлять, удалять, искать, редактировать автомобили и их модели. А также оформлять или удалять заказы с особой скоростью.

Клиенты могут получить полный список имеющихся автомобилей в данном автосалоне, произвести сортировку или поиск по своим предпочтениям. А также оформить заказ в один клик.

Также каждый посетитель, вне зависимости от его статуса и регистрации, может просмотреть всю необходимую контакную информацию о нашем автосалоне, ознакомиться с сервисными услугами, просмотреть всю информацию о каждом автомобиле, и зарегистрироваться для дальнейших действий.

Каждый пользователь нашего автосалона, имеет возможность изменить свои личные и персональные данные, удалить аккаунт или выйти из него в случае необходимости, изменить пароль для входа.

Таким образом, данный сайт в перспективе сможет принести много пользы множеству работников автосалона, клиентам и посетителям, посколько оно создавалось специально для упрощения взаимодействия с данными в этой области.

**ЗАКЛЮЧЕНИЕ**

В результате выполнения работы была достигнута цель – разработана база данных и веб-сайт автосалона. В ходе разработки были решены следующие задачи:

Был проведен анализ информационных потребностей пользователей и сформулирована задача, реализация которой позволила удовлетворить их потребности, описана предметная область, инфологический, логический и физические этапы проектирования, описана среда и язык реализации.

Были разработаны инфологическая и физическая модели БД, определены типы данных и ограничения. Были освоены некоторые приемы написания запросов и представлений на языке SQL.

В данной курсовой работе была спроектирована (до 3Н) и реализована реляционная база данных (СУБД MySQL SERVER), создали программное средство, позволяющее осуществлять вывод данных и их изменение. В базе данных имеются ограничения: на некоторых полях установлено ограничение not null, во всех таблицах выбрано ключевое поле, на него наложено ограничение auto\_increment.

В дальнейшем разработанную систему можно усовершенствовать до полноценного интернет-магазина автосалона какого-либо бренда и использовать каркас и структуру для любого автомобильного бренда.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Кузнецов Максим Валерьевич, Симдянов Игорь Вячеславович. MySQL на примерах.
2. А. Т. Доманов, Н. И. Сорока, БГУИР. Стандарт предприятия. Дипломные проекты (работы). Общие требования. СТП 01–2017.
3. Wiki-учебник по веб-технологиям: MySQL [Электронный ресурс] / Режим доступа: <https://www.webmasterwiki.ru/MySQL>. - Дата доступа: 21.04.2023.
4. Обучение основам MySQL [Электронный ресурс] / Режим доступа:  <https://itproger.com/course/php-mysql/21>. - Дата доступа: 29.04.2023.
5. Основные команды sql [Электронный ресурс]. – Режим доступа: https://tproger.ru/translations/sql-recap/. – Дата доступа: 20.02.2023.
6. Sql теория и практика [Электронный ресурс]. – Режим доступа: https://proglib.io/p/sql-practice-sites– Дата доступа: 29.04.2023.
7. Основы sql. [Электронный ресурс]. – Режим доступа: https://htmlacademy.ru/tutorial/php/sql. – Дата доступа: 16.03.2023.
8. Gulp - сборка. [Электронный ресурс]. – Режим доступа: https://www.youtube.com/watch?v=jU88mLuLWlk. – Дата доступа: 12.02.2023.
9. Динамический веб-сайт на HTML/CSS/JS/PHP. [Электронный ресурс]. – Режим доступа: https://www.youtube.com/@andrievskii/playlists. – Дата доступа: 17.04.2023.
10. Официальный сайт BMW. [Электронный ресурс]. – Режим доступа: https://www.bmw.by/. – Дата доступа: 12.03.2023.

**ПРИЛОЖЕНИЕ A**

**(обязательное)**

**Листинг кода с комментариями**

**Файл database.php**

<?php

session\_start();

require('connect.php');

//Проверяем выполнение запроса к БД

function dbCheckErr($query) {

  $errInfo = $query->errorInfo(); //Получаем ошибки в массив

  if($errInfo[0] !== PDO::ERR\_NONE){

    echo $errInfo[2]; //Вывод ошибки

    exit(); //Завершаем функцию

  }

  return true;

}

//Запрос на получение всех данных из одной таблицы

function selectAll($table, $params = []) {

  global $pdo; //Глобальная переменная

  //Формируем sql запрос (Пример: SELECT \* FROM models)

  $sql = "SELECT \* FROM $table";

  if(!empty($params)) { //Если массив параметров не пустой, то разбираем данные

    $i = 0; //Если i=0, то where иначе and

    foreach($params as $key => $value){ //Разбираем данные

      if(!is\_numeric($value)) { //Если параметр строка, то оборачиваем ее в кавычки

        $value = "'" . $value . "'";

      }

      if($i === 0) { //Если первый параметр добавляем впереди "WHERE

        $sql = $sql . " WHERE  $key = $value";

      } else { //Если параметр не первый добавляем впередм "AND

        $sql = $sql . " AND  $key = $value";

      }

      $i++;

    }

  }

  //Подготовка sql запроса для отправки на сервер

  $query = $pdo->prepare($sql);

  $query->execute();

  //Проверка запроса на ошибки

  dbCheckErr($query);

  //Возвращаем полученный результат

  return $query->fetchAll();

}

//Запрос на получение одной строки из одной выбранной таблицы

function selectOne($table, $params = []) {

  global $pdo; //Глобальная переменная

  //Формируем sql запрос (Пример: SELECT \* FROM clients)

  $sql = "SELECT \* FROM $table";

  if(!empty($params)) { //Если массив параметров не пустой, то разбираем данные

    $i = 0;  //Если i=0, то where иначе and

    foreach($params as $key => $value){  //Разбираем данные

      if(!is\_numeric($value)) { //Если параметр строка, то оборачиваем ее в кавычки

        $value = "'" . $value . "'";

      }

      if($i === 0) { //Если первый параметр добавляем впереди "WHERE

        $sql = $sql . " WHERE  $key = $value";

      } else {  //Если параметр не первый добавляем впередм "AND

        $sql = $sql . " AND  $key = $value";

      }

      $i++;

    }

  }

  //Подготовка sql запроса для отправки на сервер

  $query = $pdo->prepare($sql);

  $query->execute();

  //Проверка запроса на ошибки

  dbCheckErr($query);

  //Возвращаем полученный результат

  return $query->fetch();

}

//Добавление записи в таблицу

function insert($table, $params) {

  global $pdo; //Глобальная переменная

  $i = 0; //Если 0, то запятую в sql запросе не ставим

  $coll = ''; //Ключи

  $mask = ''; //Значения

  //Разбираем параметры на данные для запроса (названия столбцов и значение)

  foreach($params as $key => $value) {

    if($i === 0) {

      $coll = $coll . $key;

      $mask = $mask . "'" . $value . "'";

    } else {

      $coll = $coll . ", $key";

      $mask = $mask . ", '$value'";

    }

    $i++;

  }

  //Формируем sql запрос (Пример: INSERT INTO contacts (name, phone, work\_time, email) VALUES ('Call-center', '80447104585', '8-21', 'call@autoidea.by'))

  $sql = "INSERT INTO $table ($coll) VALUES ($mask)";

  //Подготовка sql запроса для отправки на сервер

  $query = $pdo->prepare($sql);

  $query->execute($params);

  //Проверка запроса на ошибки

  dbCheckErr($query);

  //Возвращаем полученный результат

  return $pdo->lastInsertId();

}

//Обноваление данных в таблице БД

function update($table, $id, $params) {

  global $pdo; //Глобальная переменная

  $i = 0; //Если 0, то запятую в sql запросе не ставим

  $str = ''; //Ключи и параметры для обнолвения

  //Разбираем параметры на данные для запроса (названия столбцов и значение)

  foreach($params as $key => $value) {

    if($i === 0) {

      $str = $str . $key . " = '" . $value . "'";

    } else {

      $str = $str . ", " . $key . " = '" . $value . "'";

    }

    $i++;

  }

  //Формируем sql запрос (Пример: UPDATE contacts SET name = 'Отдел продаж автомобиля', phone = '+375447104585', work\_time = 'Пн-Вс: 8:00 - 20:00', email = 'info@autoidea.by' WHERE id = 4)

  $sql = "UPDATE $table SET $str WHERE id = $id";

  //Подготовка sql запроса для отправки на сервер

  $query = $pdo->prepare($sql);

  $query->execute($params);

  //Проверка запроса на ошибки

  dbCheckErr($query);

}

//Удаление данных из таблицы БД

function delete($table, $id) {

  global $pdo; //Глобальная переменная

  //Формируем sql запрос (Пример: DELETE FROM contacts WHERE id =4)

  $sql = "DELETE FROM $table WHERE id =" . $id;

  //Подготовка sql запроса для отправки на сервер

  $query = $pdo->prepare($sql);

  $query->execute();

  //Проверка запроса на ошибки

  dbCheckErr($query);

}

function selectAutoFromAutosWithModelsOnSingle($table1, $table2, $id) {

  global $pdo;

  $sql = "SELECT t1.\*, t2.model, t2.main\_foto FROM $table1 AS t1 JOIN $table2 AS t2 ON t1.id\_model = t2.id WHERE t1.id = $id";

  $query = $pdo->prepare($sql);

  $query->execute();

  dbCheckErr($query); //Проверка запроса на ошибки

  return $query->fetch();

}

//Получить количество авто каждой модели

function getCountModel($idModel) {

  global $pdo; //Глобальная переменная

  //Формируем sql запрос (Пример: SELECT COUNT(id\_model) AS count FROM auto JOIN models ON auto.id\_model = models.id WHERE id\_model = 56)

  $sql = "SELECT COUNT(id\_model) AS count FROM auto JOIN models ON auto.id\_model = models.id WHERE id\_model = $idModel";

  //Подготовка sql запроса для отправки на сервер

  $query = $pdo->prepare($sql);

  $query->execute();

  //Проверка запроса на ошибки

  dbCheckErr($query);

  //Возвращаем полученный результат

  return $query->fetchAll();

}

//Выборка личных данных из сессии (того кто сейчас авторизирован)

function getPersonalData($table1, $table2, $table3, $table4, $id) {

  global $pdo; //Глобальная переменная

  //Формируем sql запрос

  $sql = "SELECT

    t1.id,

    t1.last\_name,

    t1.first\_name,

    t1.surname,

    t1.date\_birth,

    t1.phone,

    t2.city,

    t2.street,

    t2.house,

    t2.apartment,

    t3.series,

    t3.number,

    t3.issued\_by,

    t4.login,

    t4.email,

    t4.date\_regist

  FROM $table1 AS t1 JOIN $table2 AS t2 ON t1.id\_address = t2.id

              JOIN $table3 AS t3 ON t1.id\_passport = t3.id

              JOIN $table4 AS t4 ON t1.id\_auth = t4.id

  WHERE t4.id = $id";

  //Подготовка sql запроса для отправки на сервер

  $query = $pdo->prepare($sql);

  $query->execute();

  //Проверка запроса на ошибки

  dbCheckErr($query);

  //Возвращаем полученный результат

  return $query->fetch();

}

//Выборка заказов определенного клиента

function getOrders($id) {

  global $pdo; //Глобальная переменная

  //Формируем sql запрос

  $sql = "SELECT

    t1.id,

    t1.date,

    t2.last\_name,

    t2.first\_name,

    t2.phone,

    t5.email AS emailEmployee,

    t4.name AS nameContact,

    t4.email AS emailContact,

    t4.phone AS phoneContact,

    t4.work\_time,

    t3.name,

    t6.model,

    t3.engine,

    t3.year,

    t3.price,

    t3.color,

    t3.complexion,

    t3.state

  FROM orders AS t1 JOIN employees AS t2 ON t1.id\_employee = t2.id

              JOIN `auto` AS t3 ON t1.id\_auto = t3.id

              JOIN contacts AS t4 ON t1.id\_contact = t4.id

              JOIN authorization AS t5 ON t2.id\_auth = t5.id

              JOIN models AS t6 ON t3.id\_model = t6.id

  WHERE t1.id\_client = $id";

  //Подготовка sql запроса для отправки на сервер

  $query = $pdo->prepare($sql);

  $query->execute();

  //Проверка запроса на ошибки

  dbCheckErr($query);

  //Возвращаем полученный результат

  return $query->fetchAll();

}

//Получаем уникальные значения цветов автомобиля

function getColorsAutos() {

  global $pdo; //Глобальная переменная

  //Формируем sql запрос

  $sql = "SELECT DISTINCT color FROM `auto`";

  //Подготовка sql запроса для отправки на сервер

  $query = $pdo->prepare($sql);

  $query->execute();

  //Проверка запроса на ошибки

  dbCheckErr($query);

  //Возвращаем полученный результат

  return $query->fetchAll();

}

//Поиск авто по выбранным критериям

function searchAutos($params, $paramsPrice, $paramsYear) {

  global $pdo; //Глобальная переменная

  //Получаем строку для поиска цены

  $i = 0;

  $price = '';

  if($paramsPrice['price\_\_from'] == '' && $paramsPrice['price\_\_to'] != '') { // Если введена только сумма "до" => price < 70000

    $price = ' price' . ' < ' . $paramsPrice['price\_\_to'];

  } elseif($paramsPrice['price\_\_from'] != '' && $paramsPrice['price\_\_to'] == '') { // Если введена только сумма "от" => price > 50000

    $price = ' price' . ' > ' . $paramsPrice['price\_\_from'];

  } elseif($paramsPrice['price\_\_from'] != '' && $paramsPrice['price\_\_to'] != '') { // Если введена сумма и "от" и "до" => price BETWEEN 50000 AND 70000

    $price = 'price ' . 'BETWEEN ' . $paramsPrice['price\_\_from'] . ' AND ' . $paramsPrice['price\_\_to'];

  }

  //Полуаем строку для поиска года выпуска

  $year = '';

  if($paramsYear['year\_\_from'] == '' && $paramsYear['year\_\_to'] != '') { // Если введен год только "до" => year < 2022

    $year = ' year' . ' < ' . $paramsYear['year\_\_to'];

  } elseif($paramsYear['year\_\_from'] != '' && $paramsYear['year\_\_to'] == '') { // Если введен год только "от" => year > 2019

    $year = ' year' . ' > ' . $paramsYear['year\_\_from'];

  } elseif($paramsYear['year\_\_from'] != '' && $paramsYear['year\_\_to'] != '') { // Если введен год и "от" и "до" => year BETWEEN 2019 AND 2022

    $year = 'year ' . 'BETWEEN ' . $paramsYear['year\_\_from'] . ' AND ' . $paramsYear['year\_\_to'];

  }

  //Получаем строку для поиска остальных данных

  $str = '';

  $i = 0;

  foreach($params as $key => $value) {

    if(!empty($value) && $value != '' && $i == 0) { //Если значение первое в строке то не ставим впереди "AND"

      $str = $str . '`' . $key .  '` = ' . "'" . $value . "'" . ' ';  // `name` = '7'  AND `color` = 'Серый'

      $i++;

    } else if(!empty($value) && $value != '' && $i != 0) {  //Если значение первое в строке то ставим впереди "AND"

      $str = $str . ' AND '  . '`' . $key .  '` = ' . "'" . $value . "'" . ' ';

    }

  }

  //Формируем итоговую строку запроса (проверка для правильности написания "WHERE" и "AND")

  if($price != '' && $year != '' && $str != '') { //Если заполнены все поля

    $result = 'WHERE ' . $price . ' AND ' . $year . ' AND ' . $str; //WHERE price BETWEEN 50000 AND 70000 AND year BETWEEN 2019 AND 2023 AND `name` = '7'  AND `complexion` = 'Средняя'  AND `color` = 'Серый'  AND `status` = '1'  AND `engine` = 'Бензиновый'

  } else if($price != '' && $year != '' && $str == '') { //Если азполнены поля цены и гола, но не заполнены другие данные

    $result = 'WHERE ' . $price . ' AND ' . $year; // WHERE price BETWEEN 50000 AND 70000 AND year BETWEEN 2020 AND 2023

  } else if($price != '' && $year == '' && $str != '') { //Если заполнены поля цены и других данных, но не заполнен год

    $result = 'WHERE ' . $price . ' AND ' . $str; //WHERE price BETWEEN 55000 AND 65000 AND `name` = '7'  AND `color` = 'Синий'

  } else if($price == '' && $year != '' && $str != '') { //Если заполнены год и другие данные, но не заполнена цена

    $result = 'WHERE ' . $year . ' AND ' . $str;   //WHERE year BETWEEN 2019 AND 2023 AND `name` = '7'  AND `complexion` = 'Полная'  AND `color` = 'Голубой'  AND `status` = '1'  AND `engine` = 'Бензиновый'

  } else if($price != '' && $year == '' && $str == '') { //Если заполнена только цена, а год и другие данные не заполнены

    $result = 'WHERE ' . $price; //WHERE price BETWEEN 50000 AND 65000

  } else if($price == '' && $year != '' && $str == '') {  //Если заполнены только год, а цена и другие данные не заполнены

    $result = 'WHERE ' . $year; //WHERE year BETWEEN 2020 AND 2022

  } else if($price == '' && $year == '' && $str != '') { //Если заполнены только другие данные, а цена и год не заполнены

    $result = 'WHERE ' . $str; //WHERE `complexion` = 'Полная'  AND `color` = 'Черный'

  }

  //Формируем sql запрос

  $sql = "SELECT

    auto.id,

    auto.name,

    auto.engine,

    auto.year,

    auto.price,

    auto.color,

    auto.complexion,

    auto.img,

    auto.status,

    models.model

    FROM `auto` JOIN `models` ON auto.id\_model = models.id $result";

  //Подготовка sql запроса для отправки на сервер

  $query = $pdo->prepare($sql);

  $query->execute();

  //Проверка запроса на ошибки

  dbCheckErr($query);

  //Возвращаем полученный результат

  return $query->fetchAll();

}

//Поиск в панели админа по почти всем данным

function searchAdmin($search, $table) {

  $search = trim(strip\_tags(stripcslashes(htmlspecialchars($search)))); //Проверка вводимой строки

  global $pdo; //Глобальная переменная

  //Формируем sql запрос

  $sql = "SELECT \* FROM $table WHERE

      last\_name LIKE '%$search%' OR

      first\_name LIKE '%$search%'OR

      surname LIKE '%$search%' OR

      date\_birth LIKE '%$search%' OR

      phone LIKE '%$search%' OR

      city LIKE '%$search%' OR

      `number` LIKE '%$search%' OR

      `login` LIKE '%$search%'  OR

      email LIKE '%$search%' OR

      series LIKE '%$search%' OR

      issued\_by LIKE '%$search%'";

  //Подготовка sql запроса для отправки на сервер

  $query = $pdo->prepare($sql);

  $query->execute();

  //Проверка запроса на ошибки

  dbCheckErr($query);

  //Возвращаем полученный результат

  return $query->fetchAll();

}

?>