**Министерство науки и образования Республики Молдовы**

**Технический университет Молдовы**

**Факультет компьютеров, информатики и микроэлектроники**

**Департамент «Информатика и Системная инженерия»**

**Курсовая работа**

по предмету: **Базы Данных**

***Разработка дизайна*** БД для проекта "**Услуги аренды автомобилей в Республике Молдове**"

***используя:*** СУБД PostgreSQL, инструмент pgAdmin 4

***реализация:*** Веб-Приложение с использованием Python Django REST Framework и JavaScript Vue.js Framework

для предоставления пользователям доступа к просмотру автомобилей и оформлению заказа.

Принял:

Унив. конф., док.

**Михаил Перебинос**

Выполнил:

Студент гр. IA-192,

**Кирьяк Александр**

**Кишинев 2020**

1. **Глава I (Введение в СУБД)**
   1. Концепция Базы Данных
   2. Управление Базами Данных
2. **Глава II (Концептуальная модель, Логическая модель)**
   1. Цели и задачи:
   2. Описание области исследования:
   3. Схема связей между подобластями области исследования:
   4. Информационные характеристики Веб-Приложения ***„Услуги аренды автомобилей в Республике Молдове***”.
   5. Таблица покрытия:
   6. Связи между объектами Области исследования:
   7. Связи между атрибутами объектов Области исследования ER – схема БД:
   8. Логическая модель
   9. Макет Базы Данных
3. **Глава III (Введение Программирование WEB)**
   1. Шаблон Веб-Приложения
   2. Функциональность Веб-Приложения
   3. Back-End Веб-Приложения
   4. Соединение Front- и Back-End
   5. Управление Базами Данных при помощи PostgreSQL
4. **Глава IV (Структура Базы Данных)**
   1. Дизайн Базы Данных:
   2. Таблица «Пользователь»
   3. Таблица «Машина»
   4. Таблица «Автопарк»
   5. Таблица «Заказ»
   6. Таблица «»

**Глава I (Введение в СУБД)**

## Концепция Базы Данных

База данных – организованная структура, предназначенная для хранения, изменения, обработки взаимосвязанной информации.

Данные внутри базы могут быть структурированы по 3 уровням:

* Концептуальный (глобальный) – отражает общее представление о данных. Этому уровню соответствует концептуальная схема базы данных, где данные описаны независимо от приложения.
* Логический – отражает представление разработчика приложения о данных. На этом уровне данные описываются в зависимости от конкретной программы.
* Физический – отражает представление системного инженера о данных.

Существуют и другие классификации баз данных:

* Примитивные модели – данные организованы в файле на логическом уровне.
* Иерархические базы данных – строго отсортированные данные, где доступ к ним осуществляется только с вершины иерархии, у записи-потомка может быть только одна запись-родитель.
* Сетевые базы данных – расширение иерархического подхода, у одного потомка может быть любое количество родителей.
* Реляционные базы данных – структурированные данные в форме отношение – таблица, расположенные на компьютере. Такие базы позволяют доступ к данным за минимальное время. Для управления данными используется язык SQL (Structured Query Language).
* Распределенные базы данных – базы данных, логически совмещенные с компьютерными сетями. Реализуются за счет:
  + Глобальная схема – определяет и описывает данные в базе
  + Схема фрагментации – описывает связи между глобальной коллекцией с каждым ее фрагментом в виде иерархии.
  + Распределительная схема – описывает форму распределения сегмента данных между компьютерами в сети. Каждый сегмент может физически располагаться на одном или более компьютерах.

## Управление Базами Данных

Система управлением базами данных (СУБД) – информационная система, обеспечивающая хранение и обработку данных.

Под управлением данными предусматривается обращение к данным, вставка и удаление данных.

СУБД должна обеспечить:

* Определение – создание базы данных
* Ввод данных в базу
* Изменение уже введенных данных

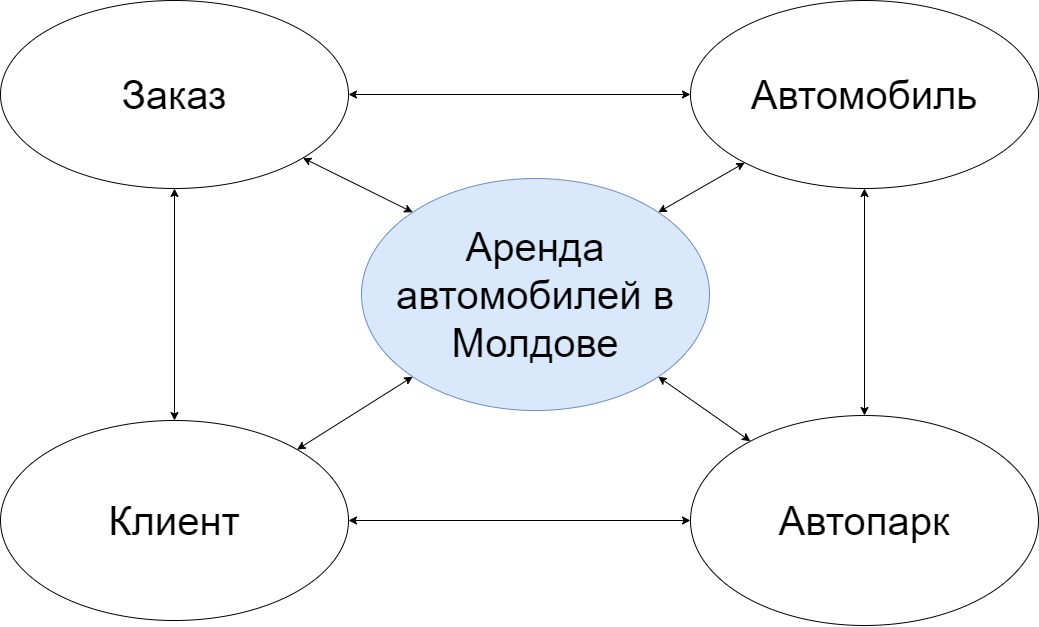
СУБД должна поддерживать языки БД (язык манипулирования данными)

**Глава II (Концептуальная модель, логическая модель)**

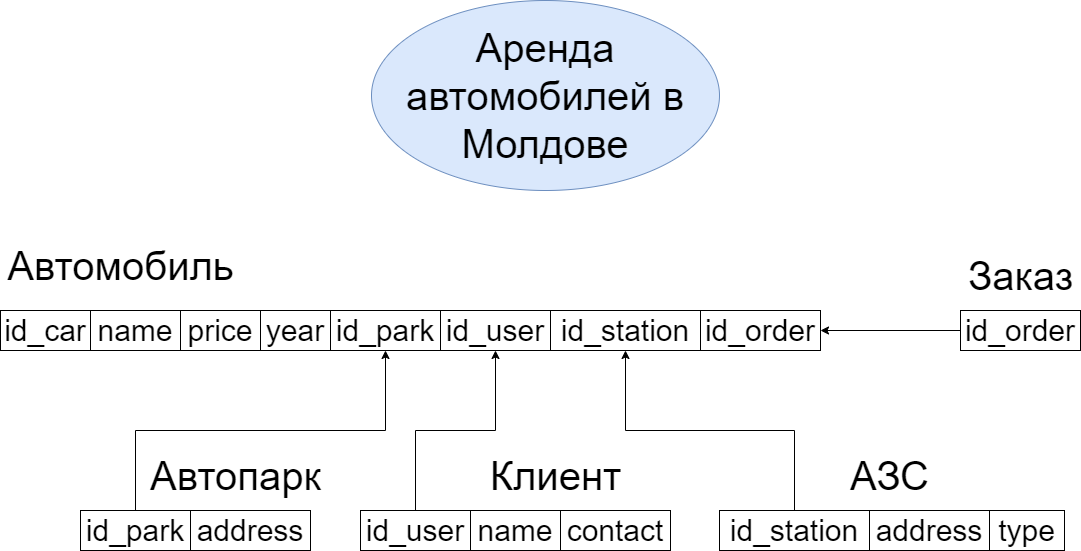
***Задание: Услуги аренды автомобилей в Республике Молдове***

1. **Моделирование Input-Output:**

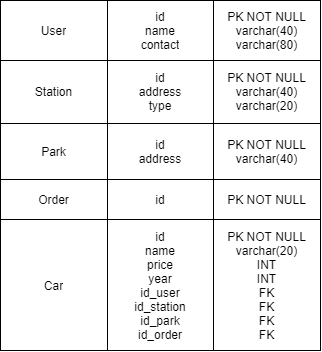
***1.1. Концептуальная модель Области Исследования:***



* 1. ***Логическая модель Области Исследования:***



***1.3. Макет Базы Данных:***



1. **Моделирование Output-Input:**

***2.1. Цели работы:***

*Цель:*

Разработка базы данных для веб-приложения аренды авто для хранения и обработки информации об автомобилях, автопарках и заказов.

*Задача 1:*

Удобный вывод информации пользователям.

*Задача 2:*

Возможность оформить заказ на автомобиль через веб-приложение.

***2.2. Описание Области Исследования:***

Сервис аренды автомобилей предоставляет временный доступ к пользованию транспорта разных классов (в т. ч. спортивного и представительского).

Создание веб-приложения намного упростит заполнение заявок пользователей на аренду автомобилей. Онлайн форма заполнения облегчит работу с клиентами и рассмотрение заявок. Также, приятный дизайн веб-приложения повысит популярность и престиж предоставляющей услуги компании.

Исходя из целей веб-приложения, выделяются следующие сущности:

**"Пользователь"**

Клиенты сервиса. Содержит основную информацию о пользователях.

**"АЗС"**

Заправочные станции, на которых компания предоставляет бесплатную заправку автомобилям, в зависимости от типа топлива.

**"Автопарк"**

Места, в которых можно забрать автомобиль после оплаты аренды.

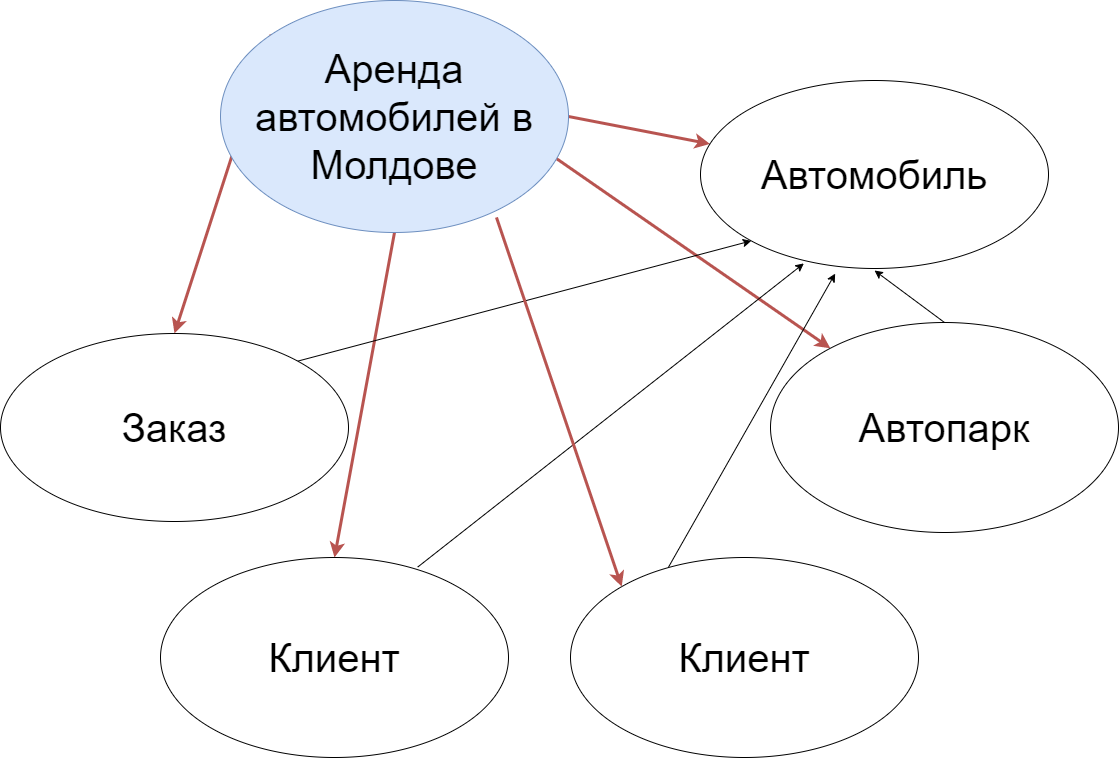
**"Заказ"**

Зарегистрированная и обработанная заявка пользователя на аренду автомобиля.

**"Автомобиль"**

Главная подобласть базы данных. Содержит информацию об автомобиле, номере заказа, текущем пользователе, автопарк, к которому он привязан.

***2.3. Схема Области Исследования:***



***2.4. Информационные характеристики:***

а) Информация

**Характеристика пользователя**

|  |  |  |
| --- | --- | --- |
| user\_id | name | contact |

**Характеристика АЗС**

|  |  |  |
| --- | --- | --- |
| station\_id | address | type |

**Характеристика Автопарка**

|  |  |
| --- | --- |
| park\_id | address |

**Характеристика Заказа**

|  |
| --- |
| order\_id |

**Характеристика Автомобиля**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| car\_id | name | price | year | id\_user | id\_station | id\_park | id\_ |

б) Запросы

Известен номер автомобиля, определить его пользователей на данный момент

|  |  |
| --- | --- |
| car\_id | user\_name |

Известен номер автомобиля, определить парк, к которому он привязан

|  |  |
| --- | --- |
| car\_id | park\_address |

Известен номер автомобиля, определить его номер заказа на данный момент

|  |  |
| --- | --- |
| car\_id | order\_id |

в) Сервисы

Известна марка автомобиля, определить машины этой марки

|  |  |
| --- | --- |
| car\_id | name |

Известен тип трансмиссии автомобиля, определить машины этого типа

|  |  |
| --- | --- |
| car\_id | trans |

Известен район расположения автопарка, определить точный адрес парка

|  |  |
| --- | --- |
| park\_id | address |

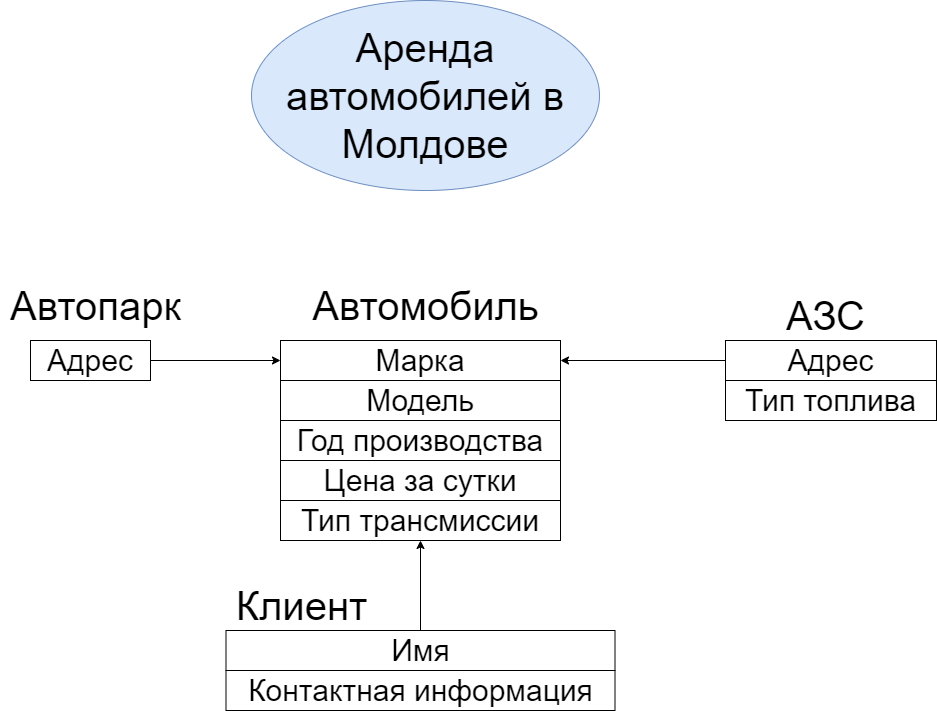
Ограничения:

1. На каждой АЗС находится лишь один вид топлива (например, бензин, электро)

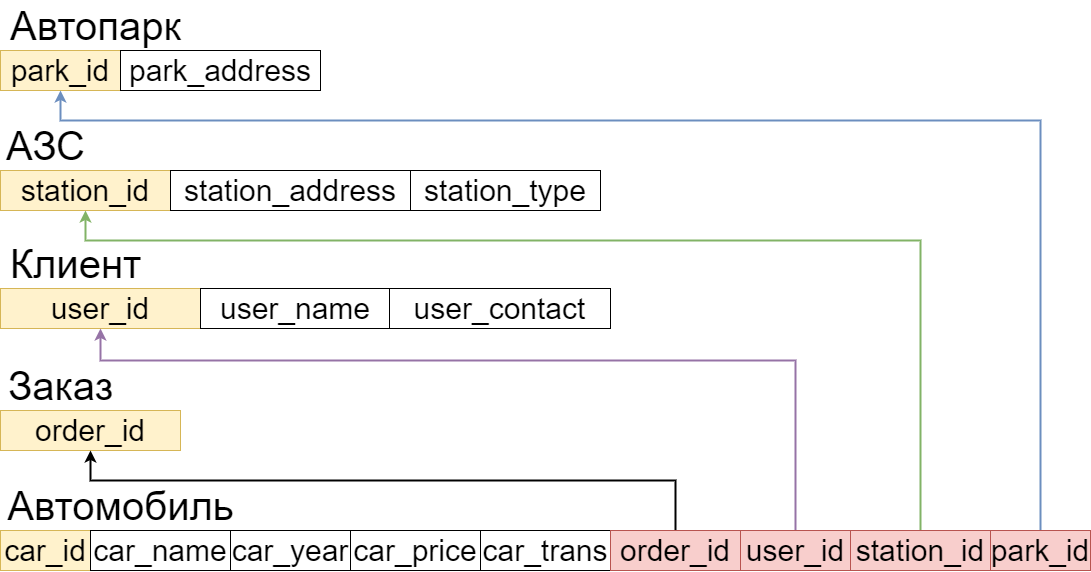
***2.5. Таблица покрытия:***

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Атрибуты \ Описание | Информация | | | | | Запросы | | | Сервисы | | |
|  | 1 | 2 | 3 | 4 | 5 | 1 | 2 | 3 | 1 | 2 | 3 |
| User\_id | + |  |  |  | + |  |  |  |  |  |  |
| Station\_id |  | + |  |  | + |  |  |  |  |  |  |
| Park\_id |  |  | + |  | + |  |  |  |  |  | + |
| Order\_id |  |  |  | + | + |  |  | + |  |  |  |
| Car\_id |  |  |  |  | + | + | + | + | + | + |  |
| User\_name | + |  |  |  |  | + |  |  | + |  |  |
| User\_contact | + |  |  |  |  |  |  |  |  |  |  |
| Station\_address |  | + |  |  |  |  |  |  |  |  |  |
| Station\_type |  | + |  |  |  |  |  |  |  |  |  |
| Park\_address |  |  | + |  |  |  | + |  |  |  | + |
| Car\_name |  |  |  |  | + |  |  |  |  |  |  |
| Car\_price |  |  |  |  | + |  |  |  |  | + |  |
| Car\_year |  |  |  |  | + |  |  |  |  |  |  |

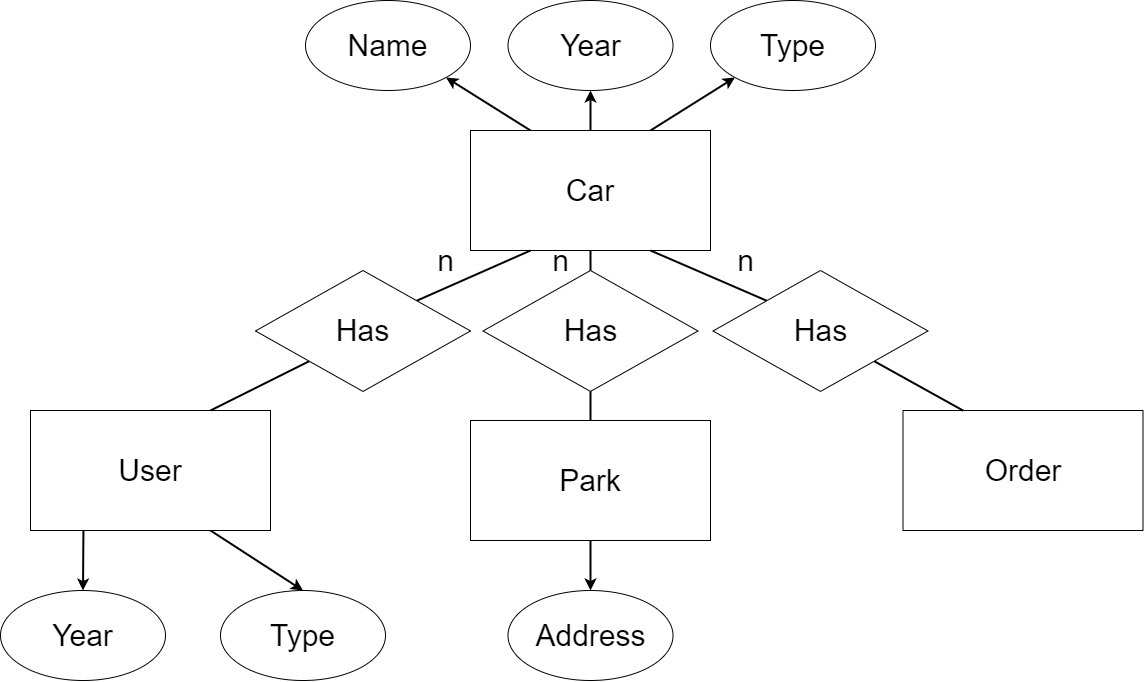
***2.6. Связи между объектами Области Исследования:***



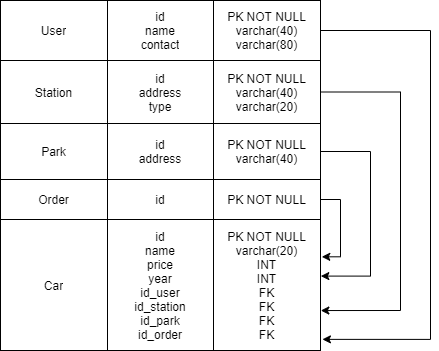
***2.7. Связи между атрибутами Области Исследования:***



***2.8. Схема ER:***



***2.9. Макет Базы Данных:***



**Глава III (Веб-разработка)**

***3.1. Шаблон веб-приложения:***

Для каркаса веб-приложения используется язык разметки HTML и фреймворк Vuetify.

**HTML** (HyperText Markup Language) - язык разметки веб-страниц в Интернете. Код HTML интерпретируется [браузерами](https://ru.wikipedia.org/wiki/%D0%91%D1%80%D0%B0%D1%83%D0%B7%D0%B5%D1%80); полученная в результате интерпретации страница отображается на экране монитора компьютера или мобильного устройства.

HTML-страницы открываются браузерами, обмениваясь с сервером информацией по протоколу [HTTP](https://ru.wikipedia.org/wiki/HTTP) или [HTTPS](https://ru.wikipedia.org/wiki/HTTPS), в виде простого текста или с использованием [шифрования](https://ru.wikipedia.org/wiki/%D0%A8%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5).

Пример кода на HTML:

<!DOCTYPE html>

<html>

<head>

</head>

<body>

<p>Text.</p>

<p>Another text.</p>

</body>

</html>

**Vuetify**  - фреймворк, обеспечивающий более высокий уровень веб-разработки. Vuetify поддерживает **SSR** (рендеринг на стороне сервера), **SPA** (одностраничное приложение), **PWA** (прогрессивное веб-приложение) и стандартные HTML страницы.

Пример HTML кода с использованием Vuetify:

<!DOCTYPE html>

<html>

<head>

<link href="https://fonts.googleapis.com/css?family=Roboto:100,300,400,500,700,900|Material+Icons" rel="stylesheet">

<link href="https://cdn.jsdelivr.net/npm/vuetify@1.x/dist/vuetify.min.css" rel="stylesheet">

<meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1, user-scalable=no, minimal-ui">

</head>

<body>

<div id="app">

<v-app>

<v-content>

<v-container>Hello world</v-container>

</v-content>

</v-app>

</div>

<script src="https://cdn.jsdelivr.net/npm/vue@2.x/dist/vue.js"></script>

<script src="https://cdn.jsdelivr.net/npm/vuetify@1.x/dist/vuetify.js"></script>

<script>

new Vue({ el: '#app' })

</script>

</body>

* 1. ***Функциональность веб-приложения:***

Функциональность реализована за счет JavaScript фреймворка Vue.js.

Vue - фреймворк для создания пользовательских интерфейсов. В отличие от фреймворков-монолитов, Vue создан пригодным для постепенного внедрения. Его ядро в первую очередь решает задачи уровня представления (view), что упрощает интеграцию с другими библиотеками и существующими проектами.

С другой стороны, Vue полностью подходит и для создания сложных одностраничных приложений (SPA, Single-Page Applications), если использовать его совместно с [современными инструментами](https://ru.vuejs.org/v2/guide/single-file-components.html) и [дополнительными библиотеками](https://github.com/vuejs/awesome-vue#components--libraries).

Пример кода на Vue.js:

<div id="app-7">

<ol>

<todo-item

v-for="item in groceryList"

v-bind:todo="item"

v-bind:key="item.id"

></todo-item>

</ol>

</div>

Vue.component('todo-item', {

props: ['todo'],

template: '<li>{{ todo.text }}</li>'

})

var app7 = new Vue({

el: '#app-7',

data: {

groceryList: [

{ id: 0, text: 'Vegetables' },

{ id: 1, text: 'Chees' },

{ id: 2, text: 'Food' }

]

}

})

Компоненты Vue довольно похожи на пользовательские элементы, являющиеся частью [спецификации W3C Web Components](https://www.w3.org/wiki/WebComponents/). Дело в том, что синтаксис компонентов Vue и правда намеренно следует этой спецификации. В частности, компоненты Vue реализуют [API слотов](https://github.com/w3c/webcomponents/blob/gh-pages/proposals/Slots-Proposal.md) и специальный атрибут is. Но есть и несколько ключевых различий:

Спецификация Web Components была завершена, но она реализована ещё не во всех браузерах. Safari 10.1+, Chrome 54+ и Firefox 63+ уже поддерживают веб-компоненты. Компоненты Vue, напротив, не требуют никаких полифилов и работают во всех поддерживаемых браузерах (IE9 и выше). При необходимости компоненты Vue могут быть «обёрнуты» в нативные пользовательские элементы.

Компоненты Vue предоставляют возможности, недоступные в простых пользовательских элементах. Самые значимые из них: кросс-компонентная передача данных, коммуникация с использованием пользовательских событий и интеграция с инструментами сборок.

* 1. ***Реализация Back-End:***

Серверная часть веб-приложения реализована на Python фреймворке Django.

Python - [высокоуровневый язык программирования](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%81%D0%BE%D0%BA%D0%BE%D1%83%D1%80%D0%BE%D0%B2%D0%BD%D0%B5%D0%B2%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) общего назначения, ориентированный на повышение производительности разработчика и читаемости кода. В то же время [стандартная библиотека](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82%D0%BD%D0%B0%D1%8F_%D0%B1%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B0_Python) включает большой набор полезных функций.

Python поддерживает [структурное](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5), [обобщенное](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D0%BE%D0%B1%D1%89%D1%91%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5), [объектно-ориентированное](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5), [функциональное](https://ru.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) и [аспектно-ориентированное программирование](https://ru.wikipedia.org/wiki/%D0%90%D1%81%D0%BF%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5). Основные архитектурные черты — [динамическая типизация](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%BD%D0%B0%D0%BC%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%82%D0%B8%D0%BF%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F), [автоматическое управление памятью](https://ru.wikipedia.org/wiki/%D0%A1%D0%B1%D0%BE%D1%80%D0%BA%D0%B0_%D0%BC%D1%83%D1%81%D0%BE%D1%80%D0%B0_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), полная [интроспекция](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D1%80%D0%BE%D1%81%D0%BF%D0%B5%D0%BA%D1%86%D0%B8%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), механизм [обработки исключений](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%D0%B8%D1%81%D0%BA%D0%BB%D1%8E%D1%87%D0%B5%D0%BD%D0%B8%D0%B9), поддержка [многопоточных вычислений](https://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B3%D0%BE%D0%BF%D0%BE%D1%82%D0%BE%D1%87%D0%BD%D0%BE%D1%81%D1%82%D1%8C), высокоуровневые [структуры данных](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85). Поддерживается разбиение программ на [модули](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D1%83%D0%BB%D1%8C_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), которые, в свою очередь, могут объединяться в пакеты.

Django - [свободный](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%BE%D0%B1%D0%BE%D0%B4%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) [фреймворк](https://ru.wikipedia.org/wiki/%D0%A4%D1%80%D0%B5%D0%B9%D0%BC%D0%B2%D0%BE%D1%80%D0%BA" \o "Фреймворк) для [веб-приложений](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5) на языке [Python](https://ru.wikipedia.org/wiki/Python" \o "Python), использующий шаблон проектирования [MVC](https://ru.wikipedia.org/wiki/Model-View-Controller).

Сайт на Django строится из одного или нескольких приложений, которые рекомендуется делать отчуждаемыми и подключаемыми. Это одно из существенных архитектурных отличий этого фреймворка от некоторых других (например, [Ruby on Rails](https://ru.wikipedia.org/wiki/Ruby_on_Rails" \o "Ruby on Rails)). Один из основных принципов фреймворка — [DRY](https://ru.wikipedia.org/wiki/DRY) ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Don't repeat yourself)

Также, в отличие от других фреймворков, обработчики [URL](https://ru.wikipedia.org/wiki/URL) в Django конфигурируются явно при помощи [регулярных выражений](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B3%D1%83%D0%BB%D1%8F%D1%80%D0%BD%D1%8B%D0%B5_%D0%B2%D1%8B%D1%80%D0%B0%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F).

Для работы с [базой данных](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) Django использует собственный [ORM](https://ru.wikipedia.org/wiki/ORM), в котором модель данных описывается классами Python, и по ней генерируется схема [базы данных](https://ru.wikipedia.org/wiki/%D0%91%D0%94).

Пример кода на Django:

from django.http import HttpResponse

def hello(request):

return HttpResponse("Hello world")

Архитектура Django похожа на «[Модель-Представление-Контроллер](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C-%D0%9F%D1%80%D0%B5%D0%B4%D1%81%D1%82%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5-%D0%9A%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D0%BB%D0%B5%D1%80)» (MVC). Контроллер классической модели MVC примерно соответствует уровню, который в Django называется Представление ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) View), а презентационная логика Представления реализуется в Django уровнем Шаблонов ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Template). Из-за этого уровневую архитектуру Django часто называют «Модель-Шаблон-Представление» (MTV).

Предоставляет ряд средств, которые помогают в быстрой разработке веб-сайтов информационного характера. Так, например, разработчику не требуется создавать контроллеры и страницы для административной части сайта, в Django есть встроенное приложение для управления содержимым, которое можно включить в любой сайт, сделанный на Django, и которое может управлять сразу несколькими сайтами на одном сервере. Административное приложение позволяет создавать, изменять и удалять любые объекты наполнения сайта, протоколируя все совершённые действия, и предоставляет интерфейс для управления пользователями и группами (с пообъектным назначением прав).

Некоторые возможности Django:

* [ORM](https://ru.wikipedia.org/wiki/ORM), [API](https://ru.wikipedia.org/wiki/API) доступа к [БД](https://ru.wikipedia.org/wiki/%D0%91%D0%94) с поддержкой [транзакций](https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D0%BD%D0%B7%D0%B0%D0%BA%D1%86%D0%B8%D1%8F_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0))
* встроенный интерфейс администратора, с уже имеющимися переводами на многие языки
* диспетчер URL на основе [регулярных выражений](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B3%D1%83%D0%BB%D1%8F%D1%80%D0%BD%D1%8B%D0%B5_%D0%B2%D1%8B%D1%80%D0%B0%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F)
* расширяемая система шаблонов с тегами и наследованием
* система кеширования
* [интернационализация](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F)
* подключаемая архитектура приложений, которые можно устанавливать на любые Django-сайты
* «generic views» — шаблоны функций контроллеров
* авторизация и аутентификация, подключение внешних модулей аутентификации: [LDAP](https://ru.wikipedia.org/wiki/LDAP), [OpenID](https://ru.wikipedia.org/wiki/OpenID" \o "OpenID) и проч.
* система фильтров («middleware») для построения дополнительных обработчиков запросов, как например включённые в дистрибутив фильтры для кеширования, сжатия, нормализации URL и поддержки анонимных сессий
* библиотека для работы с формами (наследование, построение форм по существующей модели БД)
* встроенная автоматическая документация по тегам шаблонов и моделям данных, доступная через административное приложение
  1. ***Обмен данными между клиентской и серверной частями:***

В основе взаимодействия систем лежит технология AJAX, в рамках применения которой используется экземпляр объекта XMLHttpRequest. Для того чтобы облегчить программистам работу с AJAX и XMLHttpRequest, созданы специализированные библиотеки, которые предоставляют разработчикам удобные интерфейсы.

Axios - HTTP-клиент, основанный на промисах и предназначенный для браузеров и для Node.js.

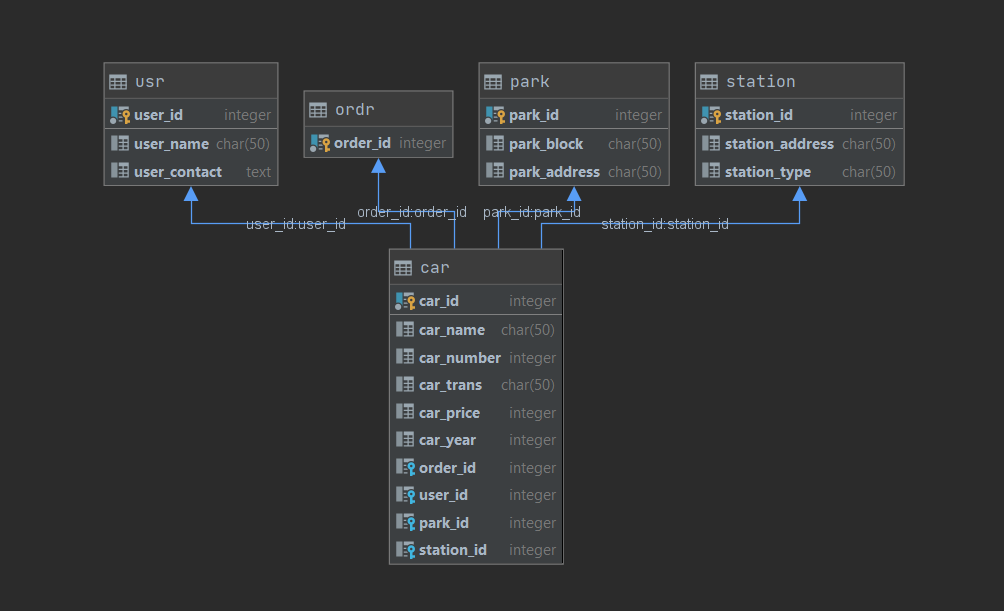
Promise - это специальный объект, который связывает «создающий» и «потребляющий» коды вместе. «Создающий» код может выполняться сколько потребуется, чтобы получить результат, а промис делает результат доступным для кода, когда результат готов.

Преимущества Axios:

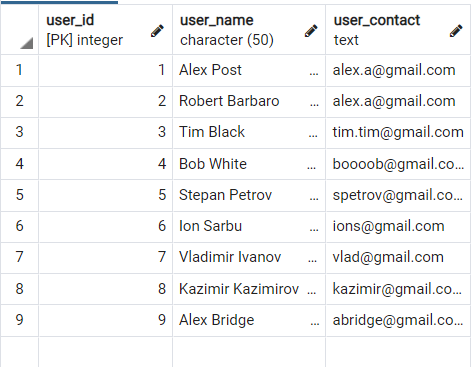
* Работает в среде Node.js и в браузерах.
* Поддерживает промисы.
* Позволяет выполнять и отменять запросы.
* Позволяет задавать тайм-аут ответа.
* Поддерживает защиту от XSRF-атак.
* Позволяет перехватывать запросы и ответы.
* Поддерживает индикацию прогресса выгрузки данных.
* Широко используется в проектах, основанных на React и Vue.

**Глава IV (Структура Базы данных)**

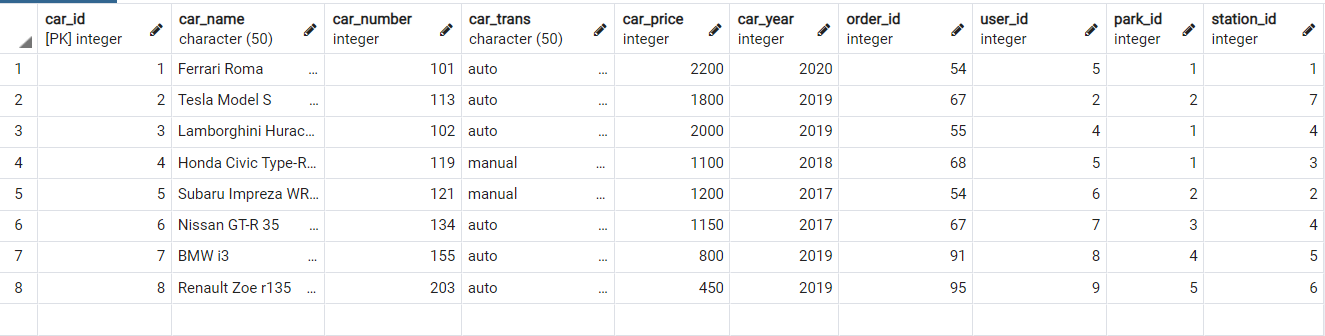
* 1. ***Схема Базы данных:***



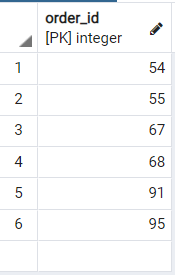
* 1. ***Таблица Клиент:***



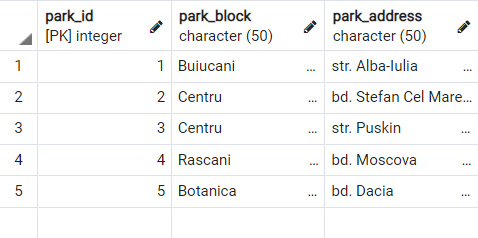
* 1. ***Таблица Машина:***

****

* 1. ***Таблица Заказ:***



* 1. ***Таблица Автопарк:***



* 1. ***Таблица АЗС:***

