**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ **«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»**

**(БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Расчетно-графическое задание

дисциплина: Технологии web-программирования

тема: «Сайт службы такси»

Выполнил: ст. гр. ПВ-42

Курмачев Сергей Васильевич

Проверил: Картамышев С. В.

Белгород 2020

**Оглавление**

[Цель работы 3](#_Toc59814242)

[Разработка макетов и верстка шаблонов web-приложения с помощью языков HTML и CSS 4](#_Toc59814243)

[Клиентское программирование. Vue.js 15](#_Toc59814244)

[Серверное программирование. Yii2 23](#_Toc59814245)

[Разработка и проектирование базы данных 27](#_Toc59814246)

[REST API 34](#_Toc59814247)

[Работа с HTTP запросами 37](#_Toc59814248)

[Вывод 41](#_Toc59814249)

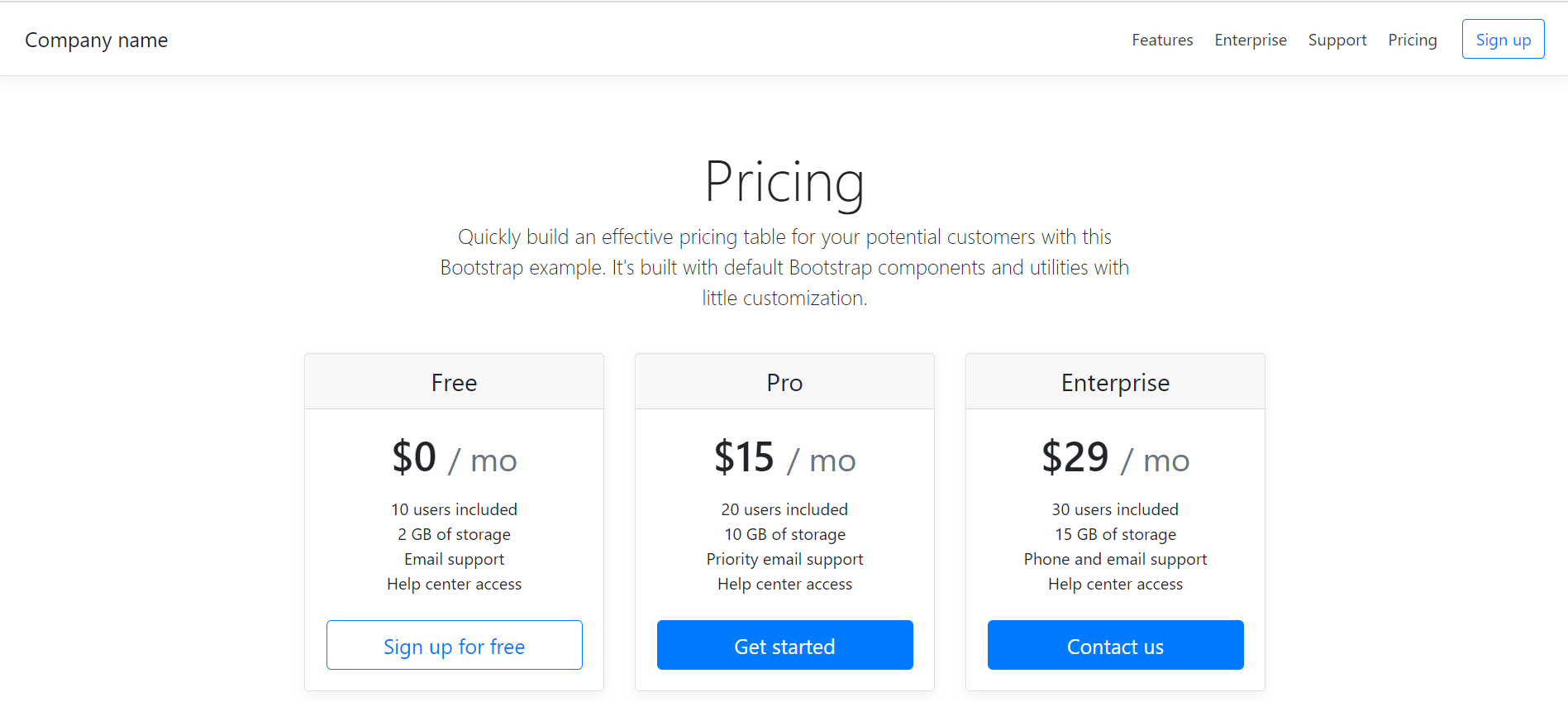
# **Цель работы**

Цель проекта: разработать веб-приложение по выбранной теме – сайт службы такси. Для реализации поставленной задачи необходимо разработать макеты и сверстать страницы, перенести верстку в компоненты фреймворка Vue.js, настроить бэкенд с docker и yii2, спроектировать и создать базу данных, реализовать REST API и связать бэкенд с фронтендом.

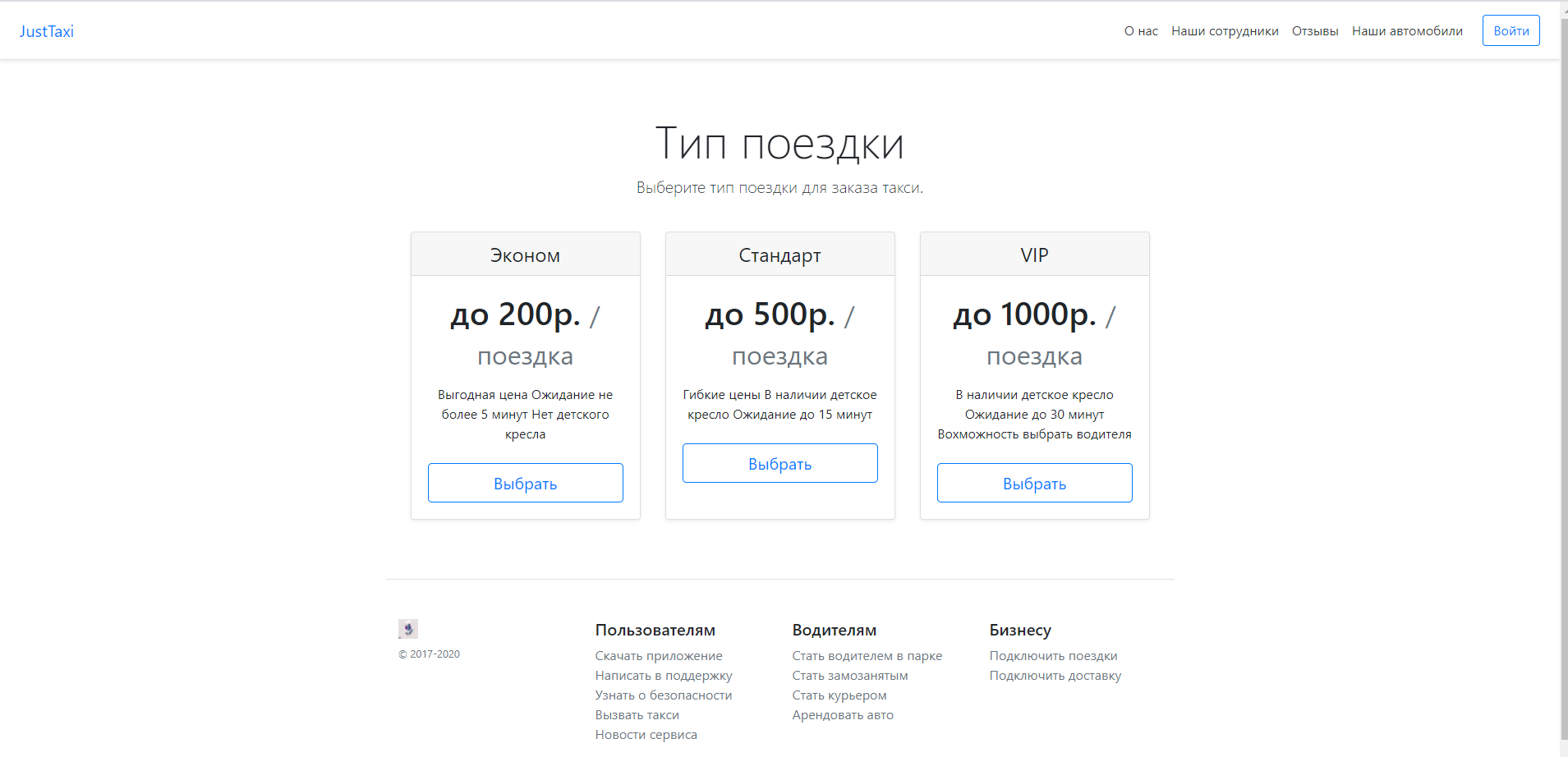
Функционал приложения прост. Пользователь может авторизоваться на сайте и ознакомиться с информацией, предоставляемой службой такси (для этого ознакомления авторизация необязательна). На главной странице, помимо кнопок навигации, выводятся тарифы для поездок, предлагаемые на данный момент компанией. Кликая на навигационную панель, пользователь будет перемещаться по разделам сайта.

# **Разработка макетов и верстка шаблонов web-приложения с помощью языков HTML и CSS**

В качестве темы проекта было решено разработать сайт, посвященный творчеству В. В. Набокова. Для реализации идеи был выбран шаблон Bootstrap:



В результате работы с этим шаблоном были создана страница авторизации:



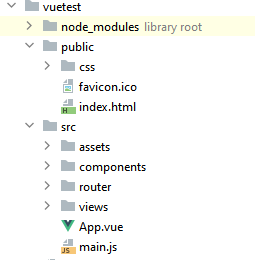
*Index.html:*

<!doctype html>  
<html lang="en">  
 <head>  
 <meta charset="utf-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">  
 <meta name="description" content="">  
 <meta name="author" content="Mark Otto, Jacob Thornton, and Bootstrap contributors">  
 <meta name="generator" content="Jekyll v4.1.1">  
 <title>JustTaxi</title>  
  
 <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" integrity="sha384-JcKb8q3iqJ61gNV9KGb8thSsNjpSL0n8PARn9HuZOnIxN0hoP+VmmDGMN5t9UJ0Z" crossorigin="anonymous">  
 <link rel="canonical" href="https://getbootstrap.com/docs/4.5/examples/pricing/">  
  
  
  
 <style>  
 .bd-placeholder-img {  
 font-size: 1.125rem;  
 text-anchor: middle;  
 -webkit-user-select: none;  
 -moz-user-select: none;  
 -ms-user-select: none;  
 user-select: none;  
 }  
  
 @media (min-width: 768px) {  
 .bd-placeholder-img-lg {  
 font-size: 3.5rem;  
 }  
 }  
 </style>  
 <!-- Custom styles for this template -->  
 <link href="css/style.css" rel="stylesheet">  
 </head>  
 <body>  
 <div class="d-flex flex-column flex-md-row align-items-center p-3 px-md-4 mb-3 bg-white border-bottom shadow-sm">  
 <h5 class="my-0 mr-md-auto font-weight-normal">JustTaxi</h5>  
 <nav class="my-2 my-md-0 mr-md-3">  
 <a class="p-2 text-dark" href="#">О нас</a>  
 <a class="p-2 text-dark" href="#">Наши сотрудники</a>  
 <a class="p-2 text-dark" href="#">Отзывы</a>  
 <a class="p-2 text-dark" href="#">Наши автомобили</a>  
 </nav>  
 <a class="btn btn-outline-primary" href="#">Войти</a>  
 </div>  
  
<div class="pricing-header px-3 py-3 pt-md-5 pb-md-4 mx-auto text-center">  
 <h1 class="display-4">Тип поездки</h1>  
 <p class="lead"> Выберите тип поездки для заказа такси.</p>  
</div>  
  
<div class="container">  
 <div class="card-deck mb-3 text-center">  
 <div class="card mb-4 shadow-sm">  
 <div class="card-header">  
 <h4 class="my-0 font-weight-normal">Эконом</h4>  
 </div>  
 <div class="card-body">  
 <h1 class="card-title pricing-card-title"> до 200р. <small class="text-muted">/ поездка</small></h1>  
 <ul class="list-unstyled mt-3 mb-4">  
 <li>Выгодная цена</li>  
 <li>Ожидание не более 5 минут</li>  
 <li>Нет детского кресла</li>  
 </ul>  
 <button type="button" class="btn btn-lg btn-block btn-outline-primary">Выбрать</button>  
 </div>  
 </div>  
 <div class="card mb-4 shadow-sm">  
 <div class="card-header">  
 <h4 class="my-0 font-weight-normal">Стандарт</h4>  
 </div>  
 <div class="card-body">  
 <h1 class="card-title pricing-card-title">до 500р. <small class="text-muted">/ поездка</small></h1>  
 <ul class="list-unstyled mt-3 mb-4">  
 <li>Гибкие цены</li>  
 <li>В наличии детское кресло</li>  
 <li>Ожидание до 15 мин.</li>  
 </ul>  
 <button type="button" class="btn btn-lg btn-block btn-primary">Выбрать</button>  
 </div>  
 </div>  
 <div class="card mb-4 shadow-sm">  
 <div class="card-header">  
 <h4 class="my-0 font-weight-normal">VIP</h4>  
 </div>  
 <div class="card-body">  
 <h1 class="card-title pricing-card-title">до 1000руб. <small class="text-muted">/ поездка</small></h1>  
 <ul class="list-unstyled mt-3 mb-4">  
 <li>В наличии детское кресло</li>  
 <li>Ожидание до 30 мин.</li>  
 <li>Возможность выбрать водителя</li>  
 </ul>  
 <button type="button" class="btn btn-lg btn-block btn-primary">Выбрать</button>  
 </div>  
 </div>  
 </div>  
  
 <footer class="pt-4 my-md-5 pt-md-5 border-top">  
 <div class="row">  
 <div class="col-12 col-md">  
 <img class="mb-2" src="img/logo.jpg" alt="" width="24" height="24">  
 <small class="d-block mb-3 text-muted">&copy; 2017-2020</small>  
 </div>  
 <div class="col-6 col-md">  
 <h5>Пользователям</h5>  
 <ul class="list-unstyled text-small">  
 <li><a class="text-muted" href="#">Скачать приложение</a></li>  
 <li><a class="text-muted" href="#">Написать в поддержку</a></li>  
 <li><a class="text-muted" href="#">Узнать о безопасности</a></li>  
 <li><a class="text-muted" href="#">Вызвать такси</a></li>  
 <li><a class="text-muted" href="#">Новости сервиса</a></li>  
 </ul>  
 </div>  
 <div class="col-6 col-md">  
 <h5>Водителям</h5>  
 <ul class="list-unstyled text-small">  
 <li><a class="text-muted" href="#">Стать водителем в парке</a></li>  
 <li><a class="text-muted" href="#">Стать замозанятым</a></li>  
 <li><a class="text-muted" href="#">Стать курьером</a></li>  
 <li><a class="text-muted" href="#">Арендовать авто</a></li>  
 </ul>  
 </div>  
 <div class="col-6 col-md">  
 <h5>Бизнесу</h5>  
 <ul class="list-unstyled text-small">  
 <li><a class="text-muted" href="#">Подключить поездки</a></li>  
 <li><a class="text-muted" href="#">Подключить доставку</a></li>  
 </ul>  
 </div>  
 </div>  
 </footer>  
</div>  
</body>  
</html>

# **Клиентское программирование. Vue.js**

На данном этапе мы познакомились с основами разработки на языке JavaScript, с основами разработки frontend приложения, в частности с использованием клиентских фреймворков (Vue.js). Научились переносить вёрстку в отдельные компоненты.

Перед началом работы необходимо было развернуть в проекте базовое приложение на фреймворке Vue.js – для этого использовался консольная команда **npm install vue@cli**. В результате последующих требуемых действий для развертки приложения была получена следующая структура приложения:



После этого можно приступать к переносу верстки, выполненной в предыдущей лабораторной работе. В отдельные представления был внесен код html-верстки страниц проекта, а они в свою очередь были разбиты на отдельные компоненты.

Результат выделения компонентов:

*CarCard.vue:*

<template>  
 <div class="card">  
 <img class="card-img-top" v-bind:src="car\_props.image" alt="Card image cap">  
 <div class="card-body">  
 <h5>  
 <router-link v-bind:to="{name: 'Car', params: {url: car\_props.url}}"> {{car\_props.model}} </router-link>  
 </h5>  
 <p class="card-text">  
 <hr>  
 Год выпуска: {{car\_props.release\_year}}  
 <hr>  
 Пробег: {{car\_props.mileage}}  
 </p>  
 </div>  
 </div>  
</template>  
  
<script>  
 export default {  
 name: "CarCard",  
 props: ['car\_props']  
}  
</script>

*CarsList.vue:*

<template>  
 <div class="cars-list">  
  
 <div class="row">  
 <div class="col-md-3">  
 <div>  
 <h5> Фирма </h5>  
 <ul class="list-unstyled">  
 <li v-for="manufacturer in filter.manufacturers">  
 <label>  
 <input type="checkbox"> {{ manufacturer }}  
 </label>  
 </li>  
 </ul>  
 </div>  
 <div>  
 <h5> Год выпуска </h5>  
 <ul class="list-unstyled">  
 <li v-for="release\_year in filter.release\_year">  
 <label>  
 <input type="checkbox"> {{ release\_year }}  
 </label>  
 </li>  
 </ul>  
 </div>  
 <div>  
 <h5> Пробег </h5>  
 <div class="d-flex">  
 <input type="number" class="form-control" placeholder="от">  
 <input type="number" class="form-control" placeholder="до">  
 </div>  
 </div>  
 </div>  
  
  
 <div class="col-md-9">  
 <div class="row">  
 <div class="col-lg-4" v-for="car\_properties in cars\_properties">  
 <CarCard v-bind:car\_props="car\_properties"></CarCard>  
 </div>  
 </div>  
 </div>  
 </div>  
 </div>  
</template>  
  
<script>  
 import CarCard from "@/components/CarCard";  
 import ***Axios*** from "axios";  
 export default {  
 name: "CarsList",  
 components: {CarCard},  
 data() {  
 return {  
 filter: {  
 manufacturers: ['BMW', 'Lada', 'Mersedes', 'VAZ'],  
 release\_year: [2010, 2015, 2016, 2019]  
 },  
 cars\_properties: []  
 }  
 }  
</script>

*Review.vue:*

<template>  
 <div class="container review">  
 <img v-bind:src="review.picture" alt="avatar">  
 <p><span > {{review.author}} </span> {{review.author\_status}}</p>  
 <p> {{review.text}} </p>  
 </div>  
</template>  
  
<script>  
 export default {  
 name: "Review",  
 props: ['review']  
}  
</script>  
  
<style scoped>  
.container { width: 75%;  
 border: 2px solid #ccc;  
 background-color: #eee;  
 border-radius: 5px; /\* скругление углов блока \*/  
 padding: 16px; /\* внутренние поля \*/  
 margin: 16px auto; /\* внешние отступы \*/  
}  
  
.container img {  
 float: left; /\* обтекание слева \*/  
 margin-right: 20px; /\* пространство между аватаркой и абзацем \*/  
 border-radius: 50%; /\* скругляет аватарку \*/  
 width: 90px;  
}  
  
.container span {  
 font-size: 18px;  
 margin-right: 15px;  
}  
  
@media (max-width: 450px) {  
 .container {  
 text-align: center;  
 }  
 .container img {  
 margin: auto;  
 float: none;  
 display: block;  
 }  
}  
</style>

*TripTypeCard.vue:*

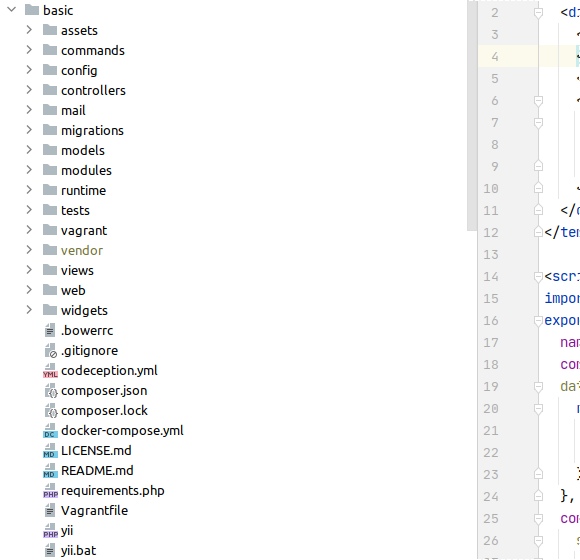
<template>  
 <div class="product">  
 <div class="card-header">  
 <h4 class="my-0 font-weight-normal">{{trip\_props.name}}</h4>  
 </div>  
 <div class="card-body">  
 <h1 class="card-title pricing-card-title"> до {{trip\_props.max\_price}}р. <small class="text-muted">/ поездка</small></h1>  
 <ul class="list-unstyled mt-3 mb-4">  
 <li>{{trip\_props.description}}</li>  
 </ul>  
 <button type="button" class="btn btn-lg btn-block btn-outline-primary">Выбрать</button>  
 </div>  
 </div>  
</template>  
  
<script>  
 export default {  
 name: "TripTypeCard",  
 props: ['trip\_props']  
  
}  
</script>

# **Серверное программирование. Yii2**

На данном этапе мы ознакомились с основами backend разработки web-приложений, научились писать скрипты на языке PHP. Познакомились с основами работы docker. Познакомились с фреймворком Yii2 и научились разворачивать проект, производить его настройку. Научились работать с API в приложении Postman.

Базовое приложение Yii2 разворачивается с помощью php-композера командой **composer create-project --prefer-dist yiisoft/yii2-app-basic basic.**

В результате выполнения команды были получены следующие директории:



Перед запуском сервера необходимо настроить файл конфигурации **docker-compose.yml**:

version: '3.1'  
services:  
 php:  
 image: yiisoftware/yii2-php:7.3-apache  
 volumes:  
 - ~/.composer-docker/cache:/root/.composer/cache:delegated  
 - ./:/app:delegated  
 ports:  
 - '1199:80'  
  
 mysql:  
 image: mysql  
 command: --default-authentication-plugin=mysql\_native\_password --character-set-server=utf8 --collation-server=utf8\_general\_ci  
 environment:  
 - MYSQL\_ROOT\_PASSWORD=verysecret  
 - MYSQL\_DATABASE=bstu  
 - MYSQL\_USER=bstu  
 - MYSQL\_PASSWORD=bstu321!

Для запуска сервера используется команда **docker-compose up -d**.

После проделанных операций можно перейти к добавлению контроллеров со статическими данными (перед этим создав контроллер для разрешения cors-запросов).

*ApiController.php:*

<?php  
  
  
namespace app\modules\v1\controllers;  
  
  
use yii\filters\auth\CompositeAuth;  
use yii\filters\ContentNegotiator;  
use yii\filters\Cors;  
use yii\filters\RateLimiter;  
use yii\filters\VerbFilter;  
use yii\rest\Controller;  
use yii\web\Response;  
  
class ApiController extends Controller  
{  
 public function behaviors()  
 {  
 return [  
 'contentNegotiator' => [  
 'class' => ContentNegotiator::className(),  
 'formats' => [  
 'application/json' => Response::FORMAT\_JSON,  
 ],  
 ],  
 'verbFilter' => [  
 'class' => VerbFilter::className(),  
 'actions' => $this->verbs(),  
 ],  
 'authenticator' => [  
 'class' => CompositeAuth::className(),  
 ],  
 'rateLimiter' => [  
 'class' => RateLimiter::className(),  
 ],  
 'corsFilter' => [  
 'class' => Cors::class,  
 ]  
 ];  
 }  
}

*CarsListController.php:*

<?php  
  
  
namespace app\modules\v1\controllers;  
  
use app\modules\v1\models\Car;  
use Yii;  
use yii\web\NotFoundHttpException;  
  
class CarsListController extends ApiController  
{  
 public function actionCars()  
 {  
 $cars\_properties = /\*\*/  
  
 return $cars\_properties;  
 }  
  
  
 public function actionInfo($url)  
 {  
 $car = /\*\*/  
 return $car;  
 }  
  
}

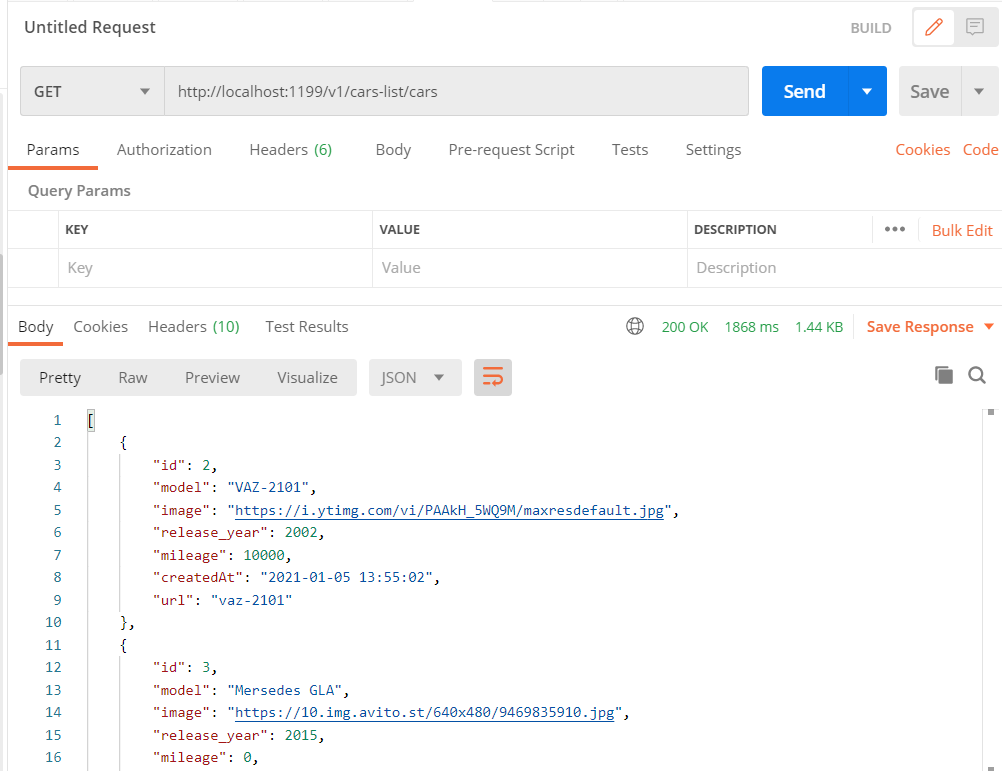
*ReviewController:*

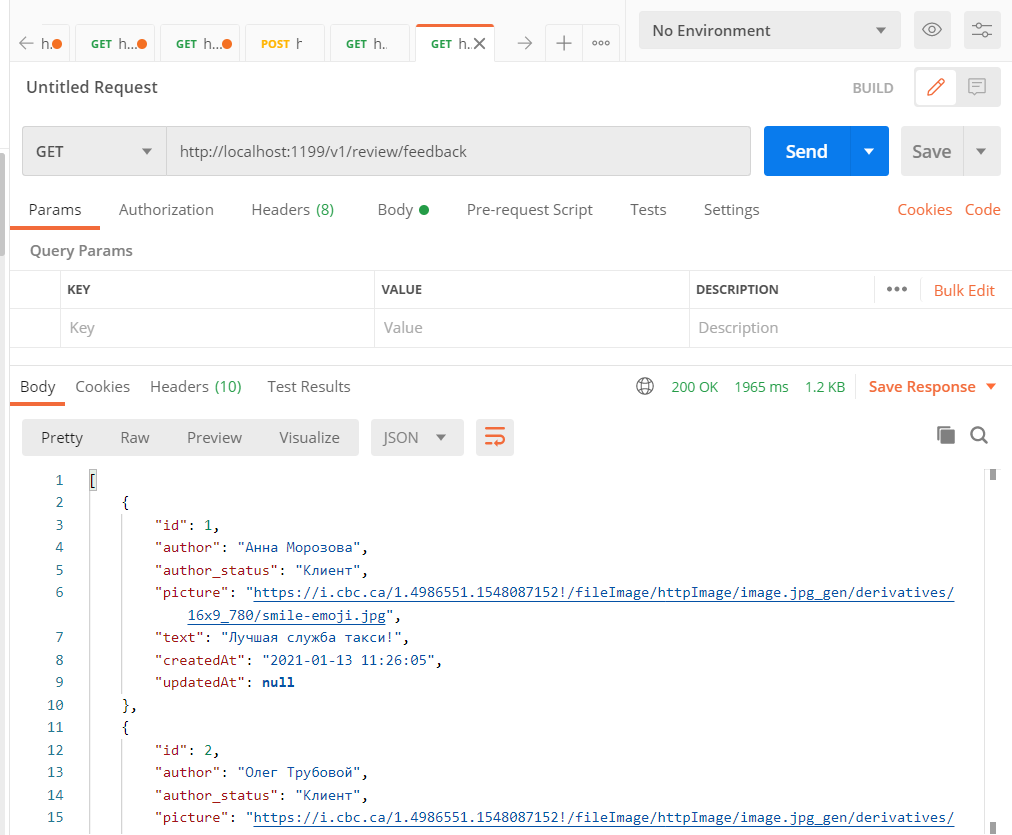
<?php  
  
  
namespace app\modules\v1\controllers;  
  
  
use app\modules\v1\models\Review;  
  
class ReviewController extends ApiController  
{  
 public function actionFeedback()  
 {  
 $reviews = /\*\*/  
  
 return $reviews;  
 }  
}

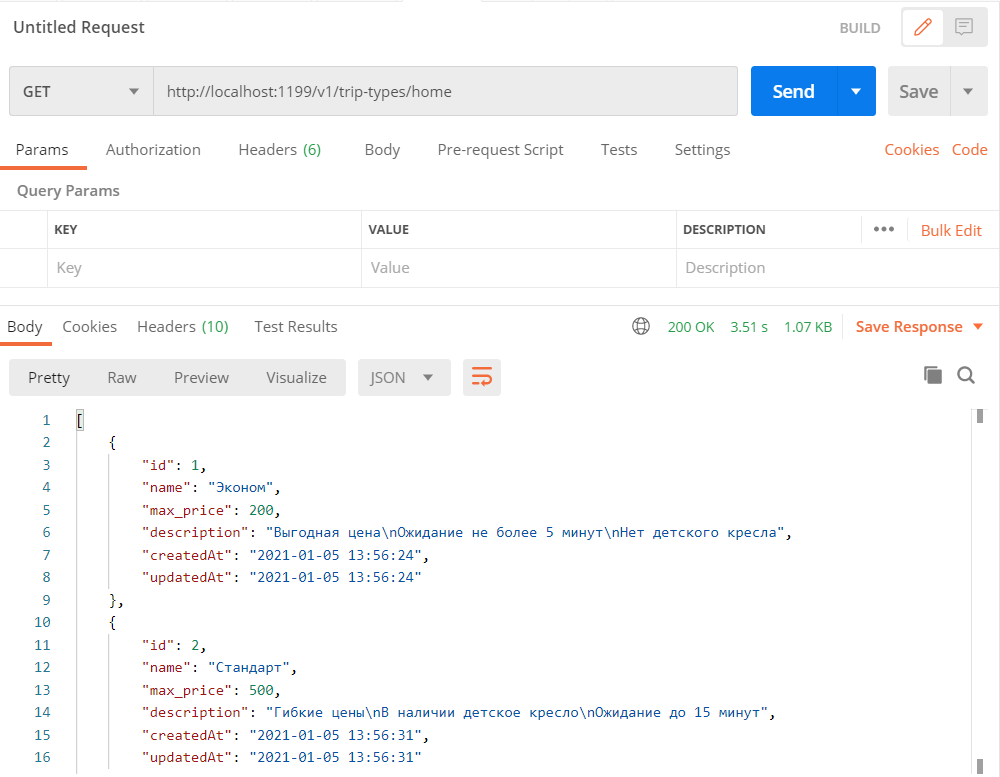
*TripTypesController:*

<?php  
  
  
namespace app\modules\v1\controllers;  
  
  
use app\modules\v1\models\TripType;  
use Yii;  
  
class TripTypesController extends ApiController  
{  
 public function actionHome()  
 {  
 $trip\_types = /\*\*/  
  
 return $trip\_types;  
 }  
  
  
  
  
}

Скриншоты работы контроллеров в Postman:







# **Разработка и проектирование базы данных**

На данном этапе были изучены основы взаимодействия web-приложения с базой данных, спроектирована база данных для хранения информации приложения (страницы, пользователи и т.п.).

Для создания таблиц в базе данных использовались миграции.

*Create\_cars\_table.php:*

<?php  
  
use yii\db\Migration;  
  
*/\*\*  
 \* Handles the creation of table `{{%cars}}`.  
 \*/*class m201222\_143521\_create\_cars\_table extends Migration  
{  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public function safeUp()  
 {  
 $this->createTable('{{%cars}}', [  
 'id' => $this->primaryKey(),  
 'model' => $this->string(128) -> notNull() ->comment('Марка'),  
 'image' => $this->string(128) -> notNull() ->comment('Фотография'),  
 'release\_year' => $this->integer() -> notNull() -> comment('Год выпуска'),  
 'mileage' => $this->integer() ->notNull()->comment('Пробег'),  
 'createdAt' => $this->dateTime()->notNull()->comment('Дата создания'),  
 'updatedAt' => $this->dateTime()->comment('Дата изменения')  
 ]);  
 }  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public function safeDown()  
 {  
 $this->dropTable('{{%cars}}');  
 }  
}

*Create\_trips\_types\_table.php:*

<?php  
  
use yii\db\Migration;  
  
*/\*\*  
 \* Handles the creation of table `{{%trips\_types}}`.  
 \*/*class m201224\_115543\_create\_trips\_types\_table extends Migration  
{  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public function safeUp()  
 {  
 $this->createTable('{{%trips\_types}}', [  
 'id' => $this->primaryKey(),  
 'name' => $this->string(128) -> notNull() ->comment('Название'),  
 'max\_price' => $this->integer() -> notNull() -> comment('Максимальная цена'),  
 'description' => $this->text()->notNull()->comment('Описание'),  
 'createdAt' => $this->dateTime()->notNull()->comment('Дата создания'),  
 'updatedAt' => $this->dateTime()->comment('Дата изменения')  
 ]);  
 }  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public function safeDown()  
 {  
 $this->dropTable('{{%trips\_types}}');  
 }  
}

*Create\_users\_table.php:*

<?php  
  
use yii\db\Migration;  
  
*/\*\*  
 \* Handles the creation of table `{{%users}}`.  
 \*/*class m210106\_103208\_create\_users\_table extends Migration  
{  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public function safeUp()  
 {  
 $this->createTable('{{%users}}', [  
 'id' => $this->primaryKey(),  
 'login' => $this->string(512)->notNull()->comment('Логин'),  
 'password' => $this->string(512)->notNull()->comment('Пароль'),  
 'accessToken' => $this->string(512)->comment('Токен пользователя')  
 ]);  
 }  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public function safeDown()  
 {  
 $this->dropTable('{{%users}}');  
 }  
}

*Create\_reviews\_table.php:*

<?php  
  
use yii\db\Migration;  
  
*/\*\*  
 \* Handles the creation of table `{{%reviews}}`.  
 \*/*class m210113\_105327\_create\_reviews\_table extends Migration  
{  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public function safeUp()  
 {  
 $this->createTable('{{%reviews}}', [  
 'id' => $this->primaryKey(),  
 'author' => $this->string(128)->notNull()->comment('Название'),  
 'author\_status' => $this->string(128)->notNull()->comment('Статус относительно службы такси'),  
 'picture' => $this->string(128) -> notNull() ->comment('Картинка'),  
 'text' => $this->string(256) -> notNull() ->comment('Текст отзыва'),  
 'createdAt' => $this->dateTime()->notNull()->comment('Дата создания'),  
 'updatedAt' => $this->dateTime()->comment('Дата изменения')  
 ]);  
 }  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public function safeDown()  
 {  
 $this->dropTable('{{%reviews}}');  
 }  
}

В результате база данных стала выглядеть следующим образом:









С помощью генератора Yii/gii были сгенерированы модели для таблиц базы данных. Например, модель Car:

<?php  
  
namespace app\modules\v1\models;  
  
use Yii;  
use yii\behaviors\SluggableBehavior;  
  
*/\*\*  
 \* This is the model class for table "cars".  
 \*  
 \** ***@property*** *int $id  
 \** ***@property*** *string $model Марка  
 \** ***@property*** *string $image Фотография  
 \** ***@property*** *int $release\_year Год выпуска  
 \** ***@property*** *int $mileage Пробег  
 \** ***@property*** *string $createdAt Дата создания  
 \** ***@property*** *string|null $updatedAt Дата изменения  
 \** ***@property*** *string $url URL  
 \*/*class Car extends BaseModel  
{  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public static function tableName()  
 {  
 return 'cars';  
 }  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public function rules()  
 {  
 return [  
 [['model', 'image', 'release\_year', 'mileage'], 'required'],  
 [['release\_year', 'mileage'], 'integer'],  
 [['createdAt', 'updatedAt'], 'safe'],  
 [['model', 'image', 'url'], 'string', 'max' => 128],  
 ];  
 }  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public function attributeLabels()  
 {  
 return [  
 'id' => 'ID',  
 'model' => 'Марка',  
 'image' => 'Фотография',  
 'release\_year' => 'Год выпуска',  
 'mileage' => 'Пробег',  
 'createdAt' => 'Дата создания',  
 'updatedAt' => 'Дата изменения',  
 'url' => 'URL',  
 ];  
 }  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public function behaviors()  
 {  
 $behaviors = parent::*behaviors*();  
  
 $behaviors[] = [  
 'class' => SluggableBehavior::class,  
 'attribute' => 'model',  
 'slugAttribute' => 'url'  
 ];  
  
 return $behaviors;  
 }  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public function toArray(array $fields = [], array $expand = [], $recursive = true)  
 {  
 return [  
 'id' => $this->id,  
 'model' => $this->model,  
 'image' => $this->image,  
 'release\_year' => $this->release\_year,  
 'mileage' => $this->mileage,  
 'createdAt' => $this->createdAt,  
 'url' => $this->url  
 ];  
 }  
}

# **REST API**

На данном этапе были изучены основы разработки API для web-приложений, а также разработан REST API для своего проекта.

Базовый URL: localhost:1199/api/v1

1. Получение cписка всех имеющихся автомобилей

URL: /cars-list/cars

Тип запроса: GET

Параметры: нет

1. Получение информации о конкретном автомобиле

URL: /cars-list/info

Тип запроса: GET

Параметры:

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Описание | Тип | Обязательный? |
| url | URL нужного автомобиля | String | Да |

1. Получение всех отзывов

URL: /review/feedback

Тип запроса: GET

Параметры: нет

1. Получение всех доступных тарифов поездки

URL: /trip-types/home

Тип запроса: GET

Параметры: нет

1. Авторизация пользователя

URL: /user/login

Тип запроса: POST

Параметры:

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Описание | Тип | Обязательный? |
| username | Логин пользователя | String | Да |
| password | Пароль пользователя | String | Да |

1. Регистрация пользователя

URL: /user/signup

Тип запроса: POST

Параметры:

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Описание | Тип | Обязательный? |
| username | Логин пользователя | String | Да |
| password | Пароль пользователя | String | Да |

# **Работа с HTTP запросами**

На данном этапе были изучены принципы работы отправки Ajax запросов. Изучена работу отправки синхронных и асинхронных запросов в Vue JS. Получены навыки работы с промисами.

Вход и выход пользователя:

signIn: function (e) {  
 e.preventDefault()  
 const submissionData = {  
 username: this.username,  
 password: this.password  
 }  
 this.$http.post('/user/login', submissionData)  
 .then((response) => {  
 user.login(response.data)  
 ***location***.href = '/'  
 })  
 .catch((error) => alert(error.response.data.statusText))  
}

logout() {  
 user.logout()  
 ***location***.href = '/'  
}

Получение списка всех автомобилей:

created() {  
 const instance = ***Axios***.create({  
 baseURL: 'http://localhost:1199/v1'  
 });  
  
 instance.get( '/cars-list/cars')  
 .then((response) => this.cars\_properties = response.data)  
}

Получение всех отзывов:

created() {  
 const instance = ***Axios***.create({  
 baseURL: 'http://localhost:1199/v1'  
 });  
  
 instance.get( '/review/feedback')  
 .then((response) => this.reviews = response.data)  
}

Получение конкретного автомобиля:

created() {  
 const url = this.$route.params.url  
  
 this.$http.get('/cars-list/info', {params: {url: url}})  
 .then((response) => this.car = response.data)  
  
}

Получение всех тарифов для поездки:

created() {  
 const instance = ***Axios***.create({  
 baseURL: 'http://localhost:1199/v1'  
 });  
  
 instance.get( '/trip-types/home')  
 .then((response) => this.trip\_types = response.data)  
}

# **Вывод**

В результате выполнения работы нам удалось разработать полноценное веб-приложение. Сначала мы сверстали страницы с помощью HTML, CSS и шаблона Bootstrap. Затем использовали Vue.js для написания клиентской части будущего приложения. Развернув серверную часть приложения с помощью docker и yii2 и создав базу данных MySQL, мы написали REST API для запросов. После заполнения базы данных мы соединили клиентскую и серверную часть, использовав для запросов к API библиотеку Axios.