МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ 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

Технологии Web-программирования

Расчетно – графическое задание

Разработка веб-приложения «Автовокзал»

Выполнил: студент группы ВТ-41

Шевченко К.К.

Проверил:

Картамышев С.В.

Белгород, 2020

**Оглавление**

[Цель работы 3](#_Toc60196554)

[Постановка задачи и выбор средств разработки 3](#_Toc60196555)

[1. Разработка макетов страниц 3](#_Toc60196556)

[2. Реализация frontend составляющей приложения 4](#_Toc60196557)

[3. Реализация backend составляющей приложения 5](#_Toc60196558)

[4. Разработка БД 5](#_Toc60196559)

[5. Разработка Rest API 7](#_Toc60196560)

[6. Взаимодействие frontend'а и backend'а 9](#_Toc60196561)

[Заключение 10](#_Toc60196562)

[Приложение 11](#_Toc60196563)

# **Цель работы**

Целью данной работы является ознакомление с процессом Web-разработки, изучение языка разметки HTML и языка стилей CSS, ознакомление с основами разработки frontend и backend приложений. В процессе выполнения придется также повторить основы JavaScript и БД. Ну и конечно же, все это делается ради того, чтобы получить практические навыки в создании Web-приложений.

# **Постановка задачи и выбор средств разработки**

В качестве темы работы был выбран веб-сервис “Автовокзал”. В веб-сервисе реализованы такие базовые функции, как: авторизация/регистрация, получение списка поездок из пункта отправления в пункт назначения по нужной дате, просмотр свободных билетов. В дальнейшем планируется расширить функционал до покупки билетов и выдачи информации по купленным билетам пользователю.

Для разработки будет использоваться фреймворк Vue JS для реализации frontend составляющей, Yii2 для реализации backend составляющей, MySQL для управления базами данных. Также будут использованы вспомогательные инструменты для облегчения процесса создания веб-приложения, такие как Adminer – инструмент для работы с базами данных (с помощью него будет выполняться заполнение БД, проверка работоспособности миграций и т.д.), Postman – приложение для проверки работоспособности API, а точнее говоря проверки запросов с клиента на сервер и получения ответа от backend, а также Composer - это пакетный менеджер уровня приложений для языка программирования PHP, который предоставляет средства по управлению зависимостями в PHP-приложении.

1. **Разработка макетов страниц**

Начало данной работы было положено с разработки базовых макетов страниц. В данном случае были реализованы две HTML страницы с использованием CSS стилей: главная и страница авторизации. В дальнейшем эти страницы могут изменяться. Они были созданы на основе Bootstrap шаблона.

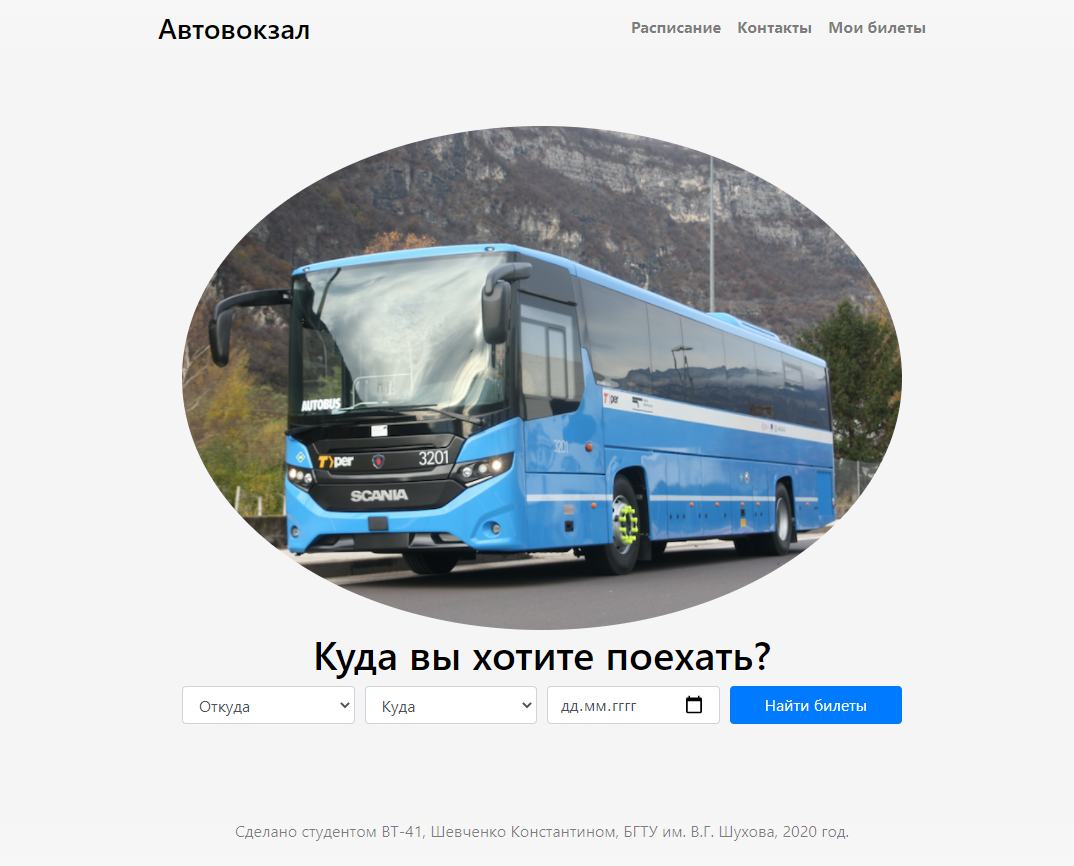


Рис. 1.1. “Главная страница”

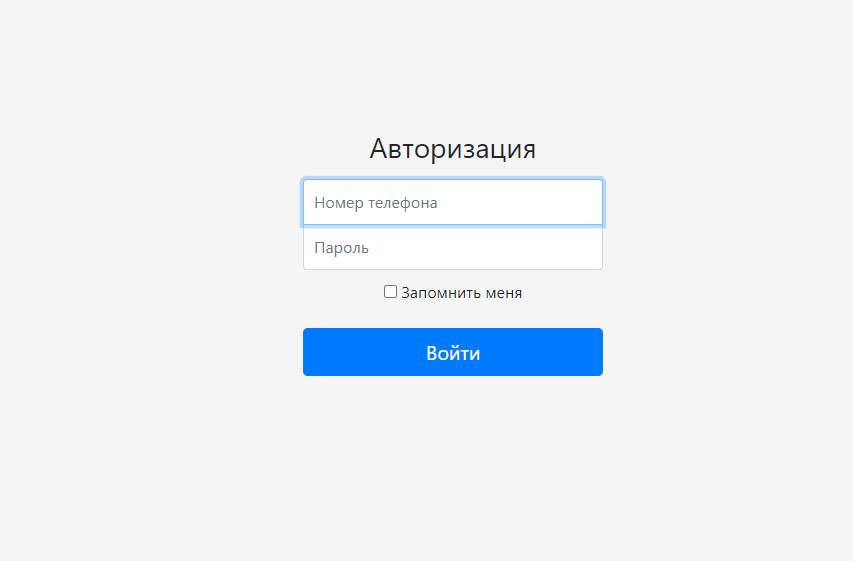


Рис. 1.2. “Страница авторизации”

1. **Реализация frontend составляющей приложения**

На данном этапе разработки все макеты страниц из предыдущего пункта были перенесены в компоненты фреймворка Vue JS. Опять же, как сказано ранее, страница будет дополняться, а также будут исправляться мелкие недочеты.

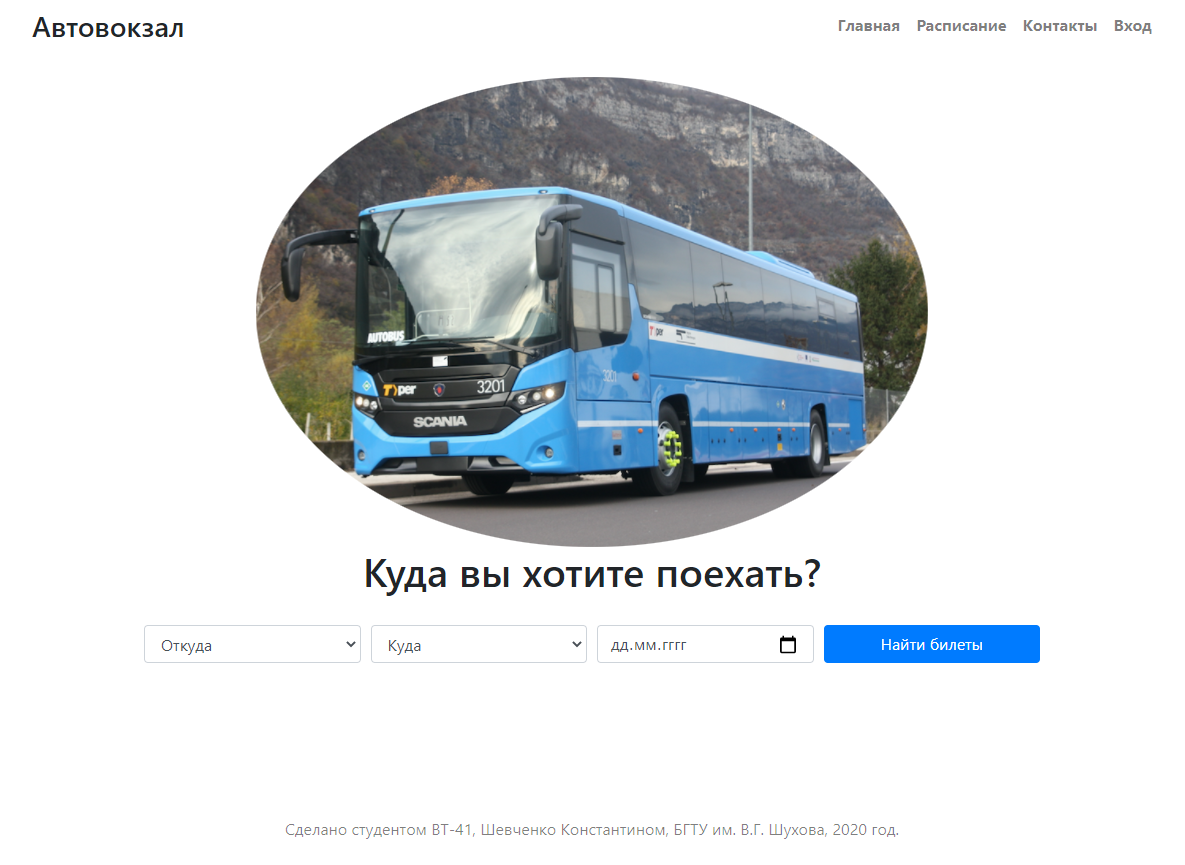


Рис. 2.1. “Главная страница, реализованная с помощью Vue JS”

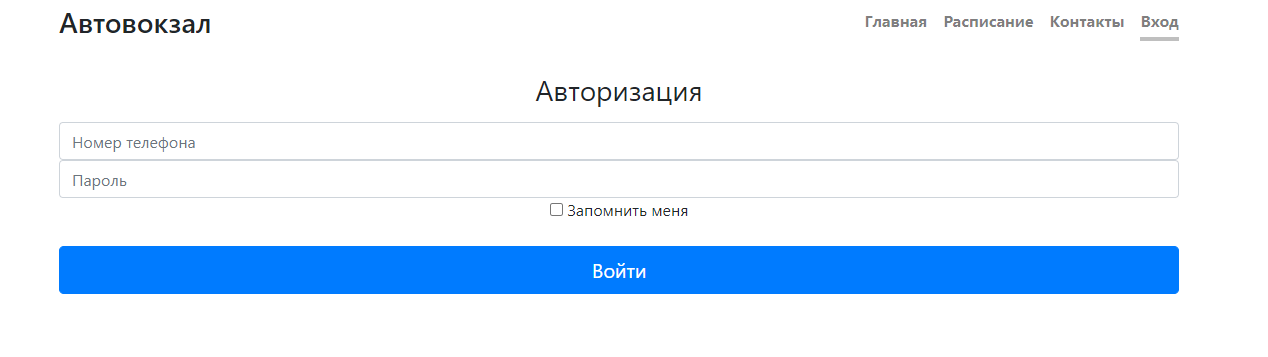


Рис. 2.2. “Страница авторизации, реализованная с помощью Vue JS”

Таким образом, мы реализовали нашу базовую frontend часть, и теперь можно приступать к разработке backend составляющей.

1. **Реализация backend составляющей приложения**

На данном этапе разработки был установлен Composer для того, чтобы развернуть базовый шаблон Yii2. Этот шаблон был доработан для возможности взаимодействия с frontend'ом из пункта 2. Таким образом, для проверки работоспособности мы настроили запросы для получения населенных пунктов, т.е. пунктов отправления и назначения. Далее будут предоставлены снимки экрана.

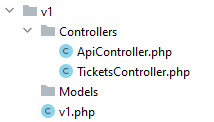


Рис. 3.1. “Созданные компоненты серверной части”

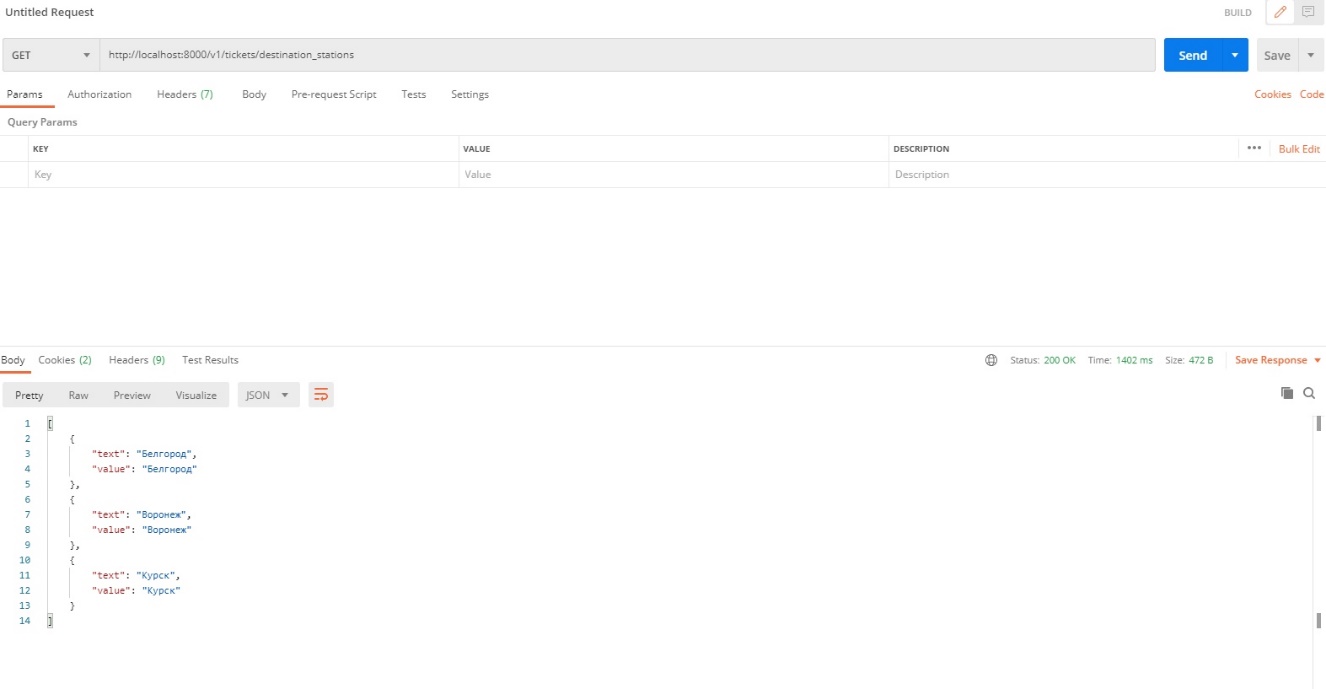


Рис 3.2. “Тестирование API (с помощью Postman)”

1. **Разработка БД**

В дальнейшем нам потребуется использовать БД, чтобы хранить всю информацию о маршрутах, поездках, билетах и пользователях. Для начала, базу данных нужно спроектировать. Результат находится на рисунке 4.1.

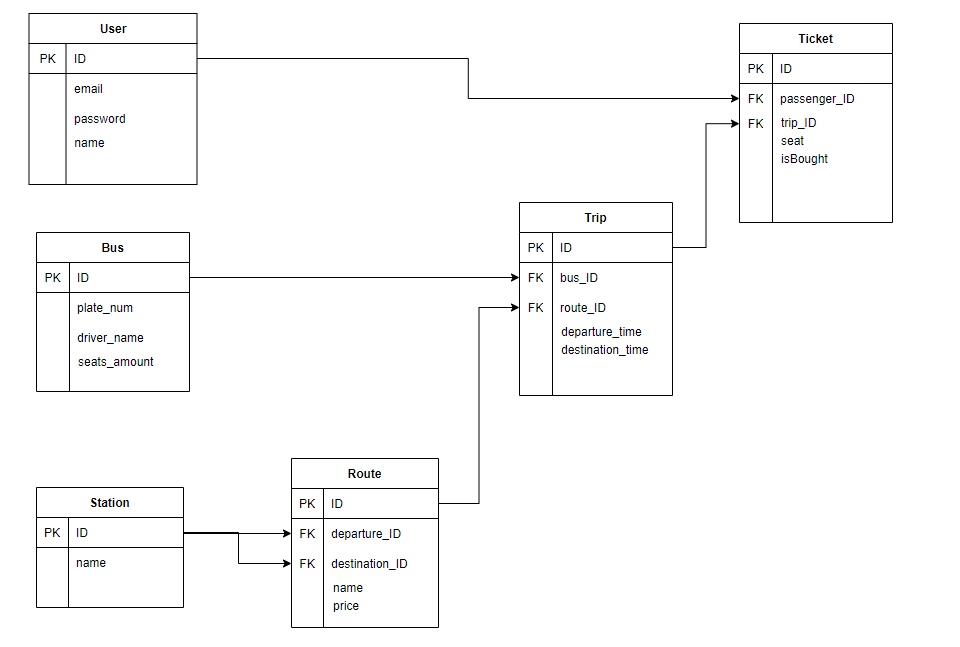


Рис. 4.1. “БД веб-приложения Автовокзал”

Вероятно, в дальнейшем она немного видоизменится, что-то будет добавлено, что-то будет выброшено. Но на данный момент так.

В качестве СУБД была выбрана MySQL, далее был сконфигурирован Docker для работы с БД и созданы миграции для генерирования таблиц БД:

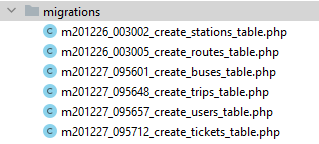


Рис. 4.2. “Миграции веб-приложения”

Убедимся, что миграции были выполнены успешно. Чтобы это сделать, воспользуемся Adminer.



Рис. 4.3. “Структура tickets, созданная с помощью миграции”

Теперь попробуем сделать запрос к БД:

Выведем все непроданные билеты.

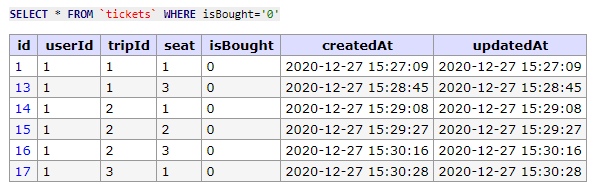


Рис. 4.4. “Результат выполнения запроса”

После этого, с помощью Gii сгенерируем модели.

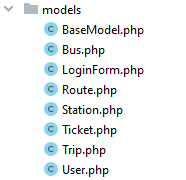


Рис. 4.5. “Модели, созданные с помощью Gii”

1. **Разработка Rest API**

В ходе выполнения данной работы было разработано Rest API к проекту. Далее внизу будет предоставлена документация.

Контроллеры:

1. Контроллер станций StationsController

URL: /stations

Тип запроса: GET

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название/url | Описание | Параметры | | |
| параметр | тип | обязательный |
| actionStation | Возвращает id населенного пункта | id | Integer | Да |
| actionAll\_departures | Возвращает пункт отправления | Нет |  |  |
| actionAll\_destinations | Возвращает пункт назначения | Нет |  |  |

1. Контроллер билетов TicketsController

URL: /tickets

Тип запроса: GET

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название/url | Описание | Параметры | | |
| параметр | тип | обязательный |
| actionUser\_tickets | Возвращает билеты пользователя по его id | userId | Integer | Да |
| actionTrip\_tickets | Возвращает билеты рейса по его id | tripId | Integer | Да |
| actionTrip\_empty\_seats | Возвращает все пустые билеты | tripId | Integer | Да |
| actionTickets | Возвращает все билеты | Нет |  |  |

1. Контроллер пользователей UsersController

URL: /users

Тип запроса: GET

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название/url | Описание | Параметры | | |
| параметр | тип | обязательный |
| actionLogin | Выполняет авторизацию пользователя по его email и password. Возвращает соответствующего пользователя или Exception. | email  password | Integer  String(32) | Да  Да |

Тип запроса: POST

|  |  |  |
| --- | --- | --- |
| Название/url | Описание | Данные |
| actionRegister | Создаёт пользователя с указанным email и password, если такого пользователя не существует, иначе throw Exception | {email, password} |

1. Контроллер поездок TripsController

URL: /trips

Тип запроса: GET

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название/url | Описание | Параметры | | |
| параметр | тип | обязательный |
| actionTrips | Возвращает поездки на дату departureTime, со станции departure и на станцию destination | id | Integer, Integer, data | Да |
| actionTrip\_from\_id | Возвращает поездку по его id | id | Integer | Да |

1. **Взаимодействие frontend'а и backend'а**

С помощью библиотеки Axios было реализовано взаимодействие с запросами. Для взаимодействия frontend составляющей с Rest API использовались кроссдоменные HTTP запросы.

Взаимодействия из выполненной работы:

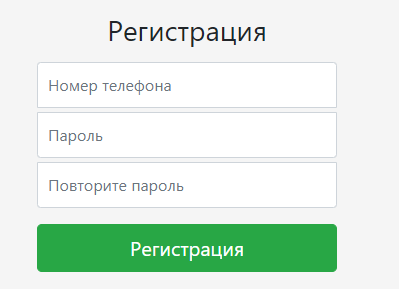
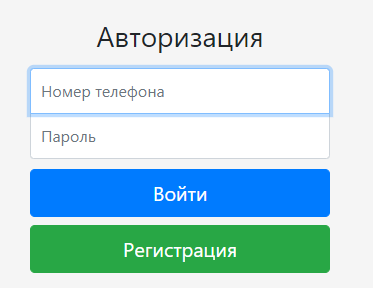


Рис. 6.1. “Авторизация и регистрация”

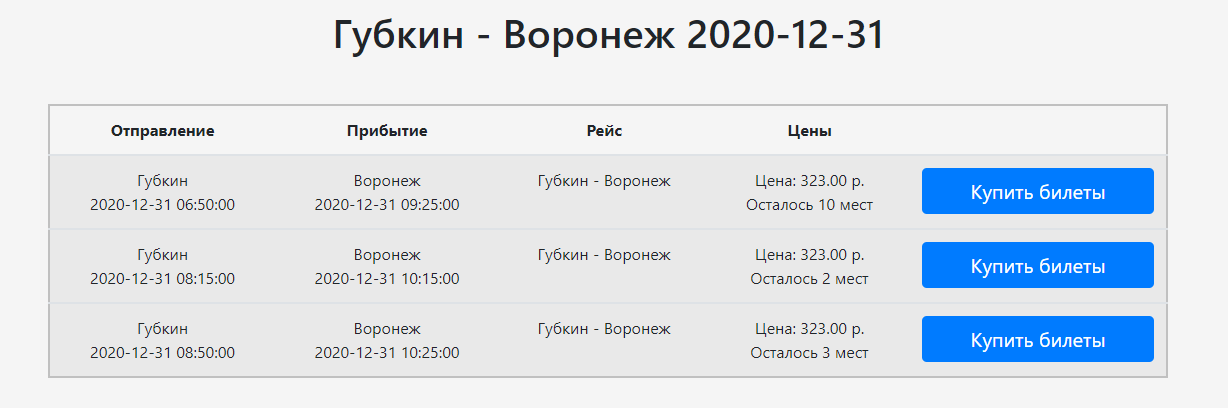


Рис. 6.2. “Получение требуемых поездок”

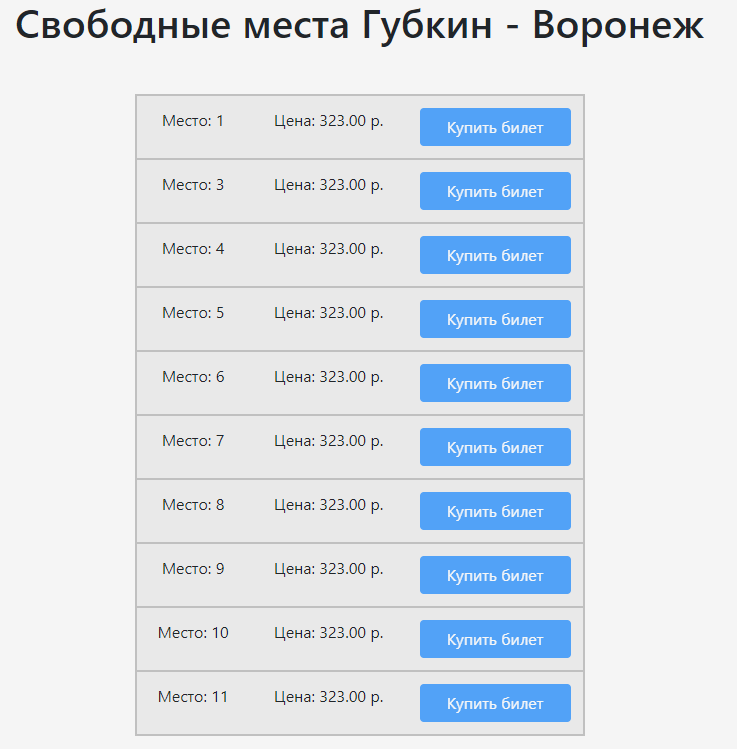


Рис. 6.3. “Получение билетов для требуемой поездки”

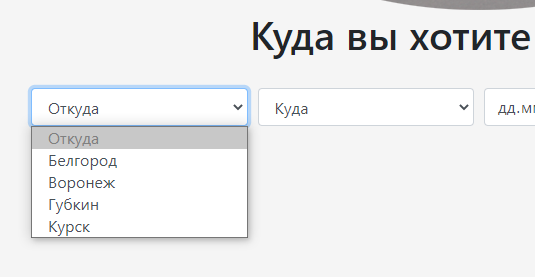


Рис. 6.4. “Выбор пунктов отправления и прибытия”

# **Заключение**

В результате выполнения данной работы были изучены основы принципы Web-разработки, а также получены практические навыки в реализации приложений на языках программирования JavaScript и PHP. В настоящее время популярность информационных технологий растет, а вместе с ними растет и популярность веб-сервисов: люди все чаще ищут различную информации, пользуются интернет-услугами и т.п. И конечно же, не стоят и технологии веб-разработки. С использованием клиент-серверной структуры можно в кратчайшие сроки создать удобное, красивое и эффективное веб-приложение для различных потребностей.

# **Приложение**

Пример кода компонента Vue:

<**template**>  
 <**main role="main"**>  
 <**img src="../../public/img/theme.png" style="width: 35vw; min-width: 330px;" class="img-fluid rounded-circle" alt="Responsive image"**>  
 <**h1 class="cover-heading"**>Куда вы хотите поехать?</**h1**>  
 <**form class="form-travel"**><**div class="form-row row-travel"**>  
 <**div class="col-md-3 mb-3"**>  
 <**select class="form-control" v-bind:class="departure\_status" id="select-from" v-model="departureId" v-on:change="**fieldChanged(1)**"**>  
 <**option disabled value="0"**>Откуда</**option**>  
 <**option v-for="**option **in departure\_options" v-bind:value="**option.**id"**>  
 {{option.**name**}}  
 </**option**>  
 </**select**>  
 </**div**>  
 <**div class="col-md-3 mb-3"**>  
 <**select class="form-control" v-bind:class="destination\_status" id="select-to" v-model="destinationId" v-on:change="**fieldChanged(2)**"**>  
 <**option disabled value="0"**>Куда</**option**>  
 <**option v-for="**option **in destination\_options" v-bind:value="**option.**id"** >  
 {{option.**name**}}  
 </**option**>  
 </**select**>  
 </**div**>  
 <**div class="col-md-3 mb-3"**>  
 <**input class="form-control" v-bind:class="date\_status" type="date" id="travelDate" v-model="date" v-bind:min="**minDate**" v-on:change="**fieldChanged(3)**"**>  
 </**div**>  
 <**div class="col-md-3 mb-3"**>  
 <**input type="button" v-on:click="**btnClick()**" class="btn btn-primary btn-block" value="Найти билеты"** >  
 </**div**>  
 </**div**>  
 </**form**>  
 </**main**>  
</**template**>  
  
<**script**>  
**export default** {  
 **name**: **'Home'**,  
 **components**: {},  
 data() {  
 **return** {  
 **departureId**: **"0"**,  
 **destinationId**: **"0"**,  
 **date**: **""**,  
 **departure\_options**: [],  
 **destination\_options**: [],  
 **departure\_status**: **""**,  
 **destination\_status**: **""**,  
 **date\_status**: **""** }  
 },  
 created() {  
 **this**.**minDate** = **new *Date***().toISOString().split(**"T"**)[0];  
 **this**.**$http**.get(**'/stations/all\_departures'**).then((response) => **this**.**departure\_options** = response.**data**)  
 **this**.**$http**.get(**'/stations/all\_destinations'**).then((response) => **this**.**destination\_options** = response.**data**)  
 },  
 **methods**: {  
 btnClick(){  
 **if** (**this**.**departureId** == **"0"**)  
 **this**.**departure\_status** = **"border-danger text-danger"**;  
 **else  
 this**.**departure\_status** = **""**;  
 **if** (**this**.**destinationId** == **"0"**)  
 **this**.**destination\_status** = **"border-danger text-danger"**;  
 **else  
 this**.**destination\_status** = **""**;  
 **if** (**this**.**date** == **""**)  
 **this**.**date\_status** = **"border-danger text-danger"**;  
 **else  
 this**.**date\_status** = **""**;  
 **if**(**this**.**departure\_status** == **""** && **this**.**destination\_status** == **""** && **this**.**date\_status** ==**""**)  
 **this**.**$router**.push({**name**: **'trips'**, **query**: {**departureId**: **this**.**departureId**, **destinationId** : **this**.**destinationId**, **date**: **this**.**date**}});  
 },  
 fieldChanged(field){  
 **switch** (field){  
 **case** 1: **this**.**departure\_status** = **""**;  
 **break**;  
 **case** 2: **this**.**destination\_status** = **""**;  
 **break**;  
 **case** 3: **this**.**date\_status** = **""**;  
 }  
 }  
 }  
}  
</**script**>

Пример кода миграции:

**<?php  
  
use** yii\db\Migration;  
  
*/\*\*  
 \* Handles the creation of table `{{%tickets}}`.  
 \*/***class** m201227\_095712\_create\_tickets\_table **extends** Migration  
{  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* **public function** safeUp()  
 {  
 $this->createTable(**'{{%tickets}}'**, [  
 **'id'** => $this->primaryKey(),  
 **'userId'** => $this->integer()->notNull()->comment(**'ID пассажира? Если ID = 1 - значит пассажира нет.'**),  
 **'tripId'** => $this->integer()->notNull()->comment(**'ID маршрута'**),  
 **'seat'** => $this->integer()->notNull()->comment(**'номер места в автобусе'**),  
  
 **'createdAt'** => $this->dateTime()->notNull()->comment(**'Дата создания'**),  
 **'updatedAt'** => $this->dateTime()->comment(**'Дата изменения'**)  
 ]);  
 $this->addForeignKey(**'fk\_userId'**, **'{{%tickets}}'**, **'userId'**, **'{{%users}}'**, **'id'**);  
 $this->addForeignKey(**'fk\_tripId'**, **'{{%tickets}}'**, **'userId'**, **'{{%trips}}'**, **'id'**);  
 }  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* **public function** safeDown()  
 {  
 $this->dropForeignKey(**'fk\_userId'**, **'{{%tickets}}'**);  
 $this->dropForeignKey(**'fk\_tripId'**, **'{{%tickets}}'**);  
 $this->dropTable(**'{{%tickets}}'**);  
 }  
}

Пример кода модели:

**<?php  
  
namespace** app\modules\v1\models;  
  
**use** app\modules\v1\models\BaseModel;  
**use** Yii;  
  
*/\*\*  
 \* This is the model class for table "tickets".  
 \*  
 \** ***@property*** *int* **$id** *\** ***@property*** *int* **$userId** *ID пассажира? Если ID = 1 - значит пассажира нет.  
 \** ***@property*** *int* **$tripId** *ID маршрута  
 \** ***@property*** *int* **$seat** *номер места в автобусе  
 \** ***@property*** *string* **$createdAt** *Дата создания  
 \** ***@property*** *string|null* **$updatedAt** *Дата изменения  
 \*  
 \** ***@property*** *User* **$user** *\** ***@property*** *Trip* **$trip** *\** ***@property*** *User* **$user0** *\*/***class** Ticket **extends** BaseModel  
{  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* **public static function** tableName()  
 {  
 **return 'tickets'**;  
 }  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* **public function** rules()  
 {  
 **return** [  
 [[**'userId'**, **'tripId'**, **'seat'**, **'createdAt'**], **'required'**],  
 [[**'userId'**, **'tripId'**, **'seat'**], **'integer'**],  
 [[**'createdAt'**, **'updatedAt'**], **'safe'**],  
 [[**'userId'**], **'exist'**, **'skipOnError'** => **true**, **'targetClass'** => Trip::*className*(), **'targetAttribute'** => [**'userId'** => **'id'**]],  
 [[**'userId'**], **'exist'**, **'skipOnError'** => **true**, **'targetClass'** => User::*className*(), **'targetAttribute'** => [**'userId'** => **'id'**]],  
 ];  
 }  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* **public function** attributeLabels()  
 {  
 **return** [  
 **'id'** => **'ID'**,  
 **'userId'** => **'User ID'**,  
 **'tripId'** => **'Trip ID'**,  
 **'seat'** => **'Seat'**,  
 **'createdAt'** => **'Created At'**,  
 **'updatedAt'** => **'Updated At'**,  
 ];  
 }  
  
 **public function** toArray(**array** $fields = [], **array** $expand = [], $recursive = **true**)  
 {  
 $res = [  
 **'id'** => $this->**id**,  
 **'user'** => $this->**user**,  
 **'seat'** => $this->**seat**,  
 **'trip'** => $this->**trip**,  
 ];  
 **return** [  
 **'trip'** => $res[**'trip'**],  
 **'tripname'** => $res[**'trip'**][**'name'**],  
 **'seat'** => $res[**'seat'**],  
 ];  
 }  
  
 **public function** behaviors()  
 {  
 **return parent**::*behaviors*();  
 }  
  
 */\*\*  
 \* Gets query for [[User]].  
 \*  
 \** ***@return*** *\yii\db\ActiveQuery  
 \*/* **public function** getUser()  
 {  
 **return** $this->**hasOne**(Trip::*className*(), [**'id'** => **'userId'**]);  
 }  
  
 */\*\*  
 \* Gets query for [[User0]].  
 \*  
 \** ***@return*** *\yii\db\ActiveQuery  
 \*/* **public function** getUser0()  
 {  
 **return** $this->**hasOne**(User::*className*(), [**'id'** => **'userId'**]);  
 }  
  
 **public function** getTrip()  
 {  
 **return** $this->**hasOne**(Trip::*className*(), [**'id'** => **'tripId'**]);  
 }  
  
  
}

Пример кода контроллера:

**<?php  
  
namespace** app\modules\v1\controllers;  
**use** app\modules\v1\models\Ticket;  
**use** yii\db\Query;  
  
**class** TicketsController **extends** ApiController {  
  
 **public function** actionUser\_tickets($userId) {  
 **return** Ticket::*find*()  
 ->where([**'userId'** => $userId])  
 ->all();  
 }  
  
 **public function** actionTrip\_empty\_seats($tripId){  
 **return** Ticket::*find*()  
 ->with(**'trip'**)  
 ->where([**'tripId'** => $tripId, **'userId'** => **'1'**])  
 ->all();  
 }  
  
 **public function** actionTrip\_tickets($tripId) {  
 **return** Ticket::*find*()  
 ->where([**'tripId'** => $tripId])  
 ->all();  
 }  
  
 **public function** actionTickets() {  
 **return** Ticket::*find*()  
 ->all();  
 }

}