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

Лабораторная работа №4

**Разработка и проектирование базы данных**

Выполнил: студент группы ВТ-41

Шевченко К.К.

Проверил:

Картамышев С.В.

Белгород, 2020

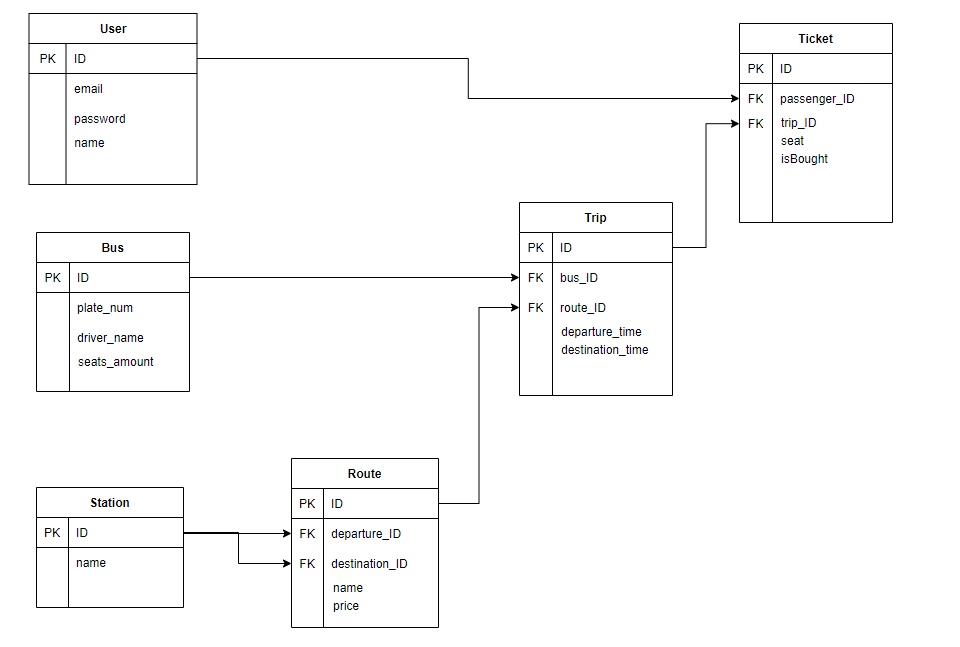
**Цель работы:** изучить основы взаимодействия web-приложения с базой данных. Спроектировать базу данных для хранения информации приложения (страницы, пользователи и т.п.).

**Задание к лабораторной работе:**

1. Развернуть базовое приложение Yii2 App Basic.
2. Настроить конфигурацию работы приложения с docker.
3. Добавить модуль для работы с API.
4. Добавить несколько контроллеров со статическими данными.
5. Продемонстрировать работу API в Postman.

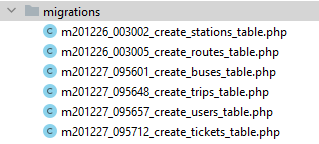
**Выполнение**

В ходе работы была спроектирована схема базы данных:



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

В качестве СУБД был выбран MySQL. Были созданы миграции для создания структуры базы данных:



Пример исходного кода миграции:

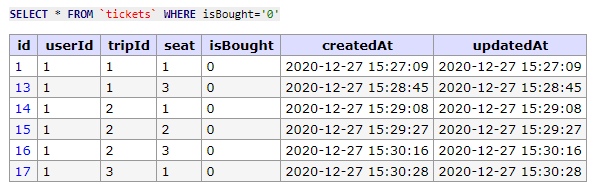
<?php  
  
use yii\db\Migration;  
  
*/\*\*  
 \* Handles the creation of table `{{%tickets}}`.  
 \*/*class m201227\_095712\_create\_tickets\_table extends Migration  
{  
 */\*\*  
 \* {@inheritdoc}  
 \*/* public function safeUp()  
 {  
 $this->createTable('{{%tickets}}', [  
 'id' => $this->primaryKey(),  
 'userId' => $this->integer()->notNull()->comment('ID пассажира'),  
 'tripId' => $this->integer()->notNull()->comment('ID маршрута'),  
 'seat' => $this->integer()->notNull()->comment('номер места в автобусе'),  
 *//'isBought' => $this->boolean()->notNull()->comment('Куплен ли билет? true - да, false - нет'),* '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}}');  
 }  
}

Посмотрим результат выполнения миграций с помощью Adminer на примере tickets:



Попробуем сделать запрос к БД:

Выведем все непроданные билеты.



После этого с помощью Gii сгенерируем модели, далее пример модели Ticket:

<?php  
  
namespace app\models;  
  
use Yii;  
  
*/\*\*  
 \* This is the model class for table "tickets".  
 \*  
 \* @property int $id  
 \* @property int $userId ID пассажира  
 \* @property int $tripId ID маршрута  
 \* @property int $seat номер места в автобусе  
 \* @property int $isBought Куплен ли билет? true - да, false - нет  
 \* @property string $createdAt Дата создания  
 \* @property string|null $updatedAt Дата изменения  
 \*  
 \* @property Trips $user  
 \* @property Users $user0  
 \*/*class Ticket extends \yii\db\ActiveRecord  
{  
 */\*\*  
 \* {@inheritdoc}  
 \*/* public static function tableName()  
 {  
 return 'tickets';  
 }  
  
 */\*\*  
 \* {@inheritdoc}  
 \*/* public function rules()  
 {  
 return [  
 [['userId', 'tripId', 'seat', 'isBought', 'createdAt'], 'required'],  
 [['userId', 'tripId', 'seat', 'isBought'], 'integer'],  
 [['createdAt', 'updatedAt'], 'safe'],  
 [['userId'], 'exist', 'skipOnError' => true, 'targetClass' => Trips::*className*(), 'targetAttribute' => ['userId' => 'id']],  
 [['userId'], 'exist', 'skipOnError' => true, 'targetClass' => Users::*className*(), 'targetAttribute' => ['userId' => 'id']],  
 ];  
 }  
  
 */\*\*  
 \* {@inheritdoc}  
 \*/* public function attributeLabels()  
 {  
 return [  
 'id' => 'ID',  
 'userId' => 'User ID',  
 'tripId' => 'Trip ID',  
 'seat' => 'Seat',  
 'isBought' => 'Is Bought',  
 'createdAt' => 'Created At',  
 'updatedAt' => 'Updated At',  
 ];  
 }  
  
 */\*\*  
 \* Gets query for [[User]].  
 \*  
 \* @return \yii\db\ActiveQuery  
 \*/* public function getUser()  
 {  
 return $this->hasOne(Trips::*className*(), ['id' => 'userId']);  
 }  
  
 */\*\*  
 \* Gets query for [[User0]].  
 \*  
 \* @return \yii\db\ActiveQuery  
 \*/* public function getUser0()  
 {  
 return $this->hasOne(Users::*className*(), ['id' => 'userId']);  
 }  
}