**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ**

**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

Лабораторная работа №4  
дисциплина: Технологии Web-программирования  
тема: «Разработка и проектирование базы данных»

Выполнила: ст. группы ПВ-41  
Немшилова А.Г.  
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. Выбрать подходящую СУБД.

2. Изучить методы взаимодействия web-приложения с базой данных (ORM, Active Record).

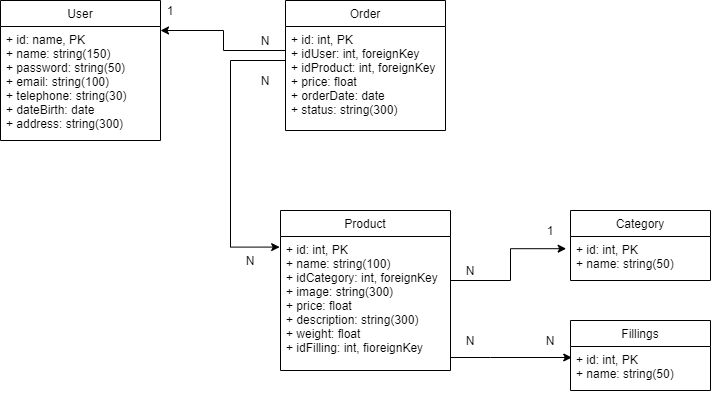
3. Разработать структуру базы данных.

4. Разработать соответствующие модели в приложении.

5. В отчёт приложить схему базы данных, а так же код одной из моделей (на своё усмотрение).

**Выполнение работы**

Для сайта интернет-магазина выпечки и десертов была разработана следующая структура базы данных:



В файле конфигурации db.php и файле docker-compose были прописаны следующие настройки базы данных:

<?*php*return [  
 'class' => 'yii\db\Connection',  
 'dsn' => 'mysql:host=mysql;dbname=db\_web',  
 'username' => 'user',  
 'password' => 'password',  
 'charset' => 'utf8',  
];

mysql:  
 image: mysql  
 command: --default-authentication-plugin=mysql\_native\_password --character-set-server=utf8 --collation-server=utf8\_general\_ci  
 environment:  
 - MYSQL\_ROOT\_PASSWORD=password  
 - MYSQL\_DATABASE=db\_web  
 - MYSQL\_USER=user  
 - MYSQL\_PASSWORD=password

Для создания таблиц с помощью команды docker-compose run php yii migrate/create create\_product\_table были созданы миграции:

<?*php*use yii\db\Migration;  
  
/\*\*  
 \* Handles the creation of table `{{%category}}`.  
 \*/  
class m201210\_111847\_create\_category\_table extends Migration  
{  
 /\*\*  
 \* {@inheritdoc}  
 \*/  
 public function safeUp()  
 {  
 $this->createTable('{{%category}}', [  
 'id' => $this->primaryKey(),  
 'name' => $this->string(*100*)->notNull()->comment('Название'),  
 ]);  
 }  
  
 /\*\*  
 \* {@inheritdoc}  
 \*/  
 public function safeDown()  
 {  
 $this->dropTable('{{%category}}');  
 }  
}

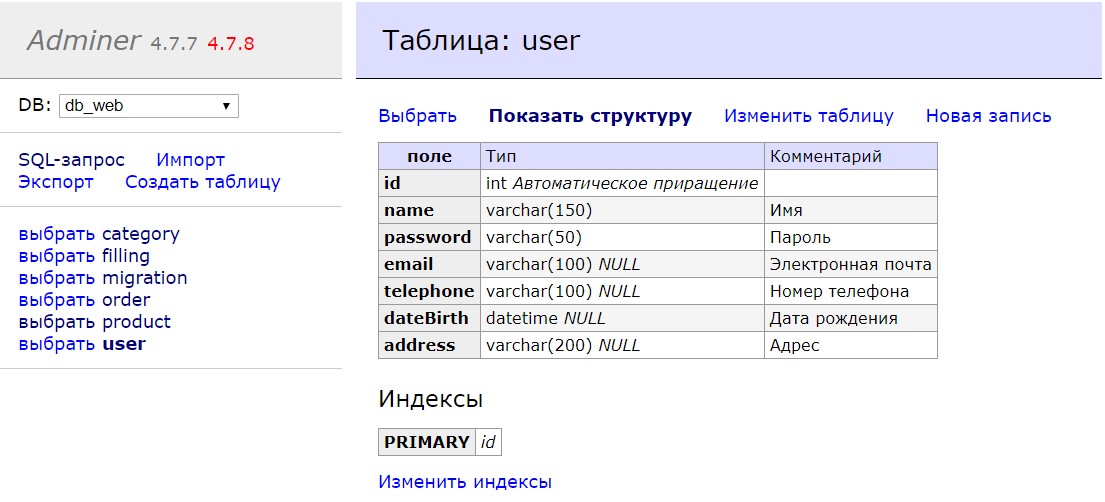
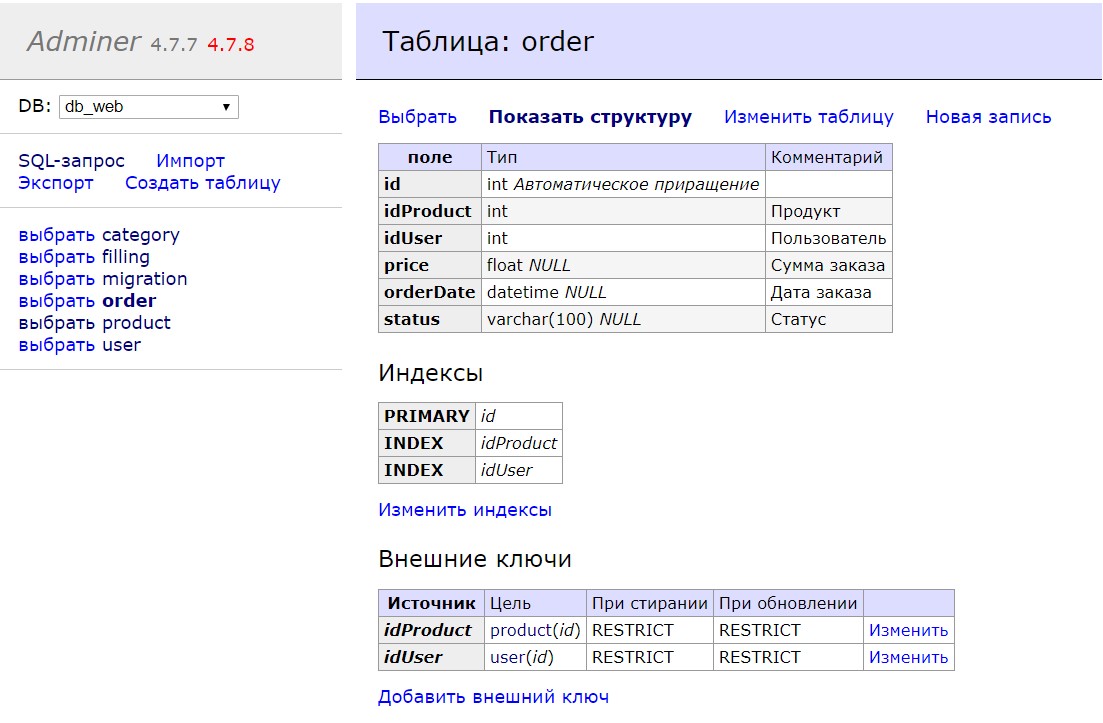
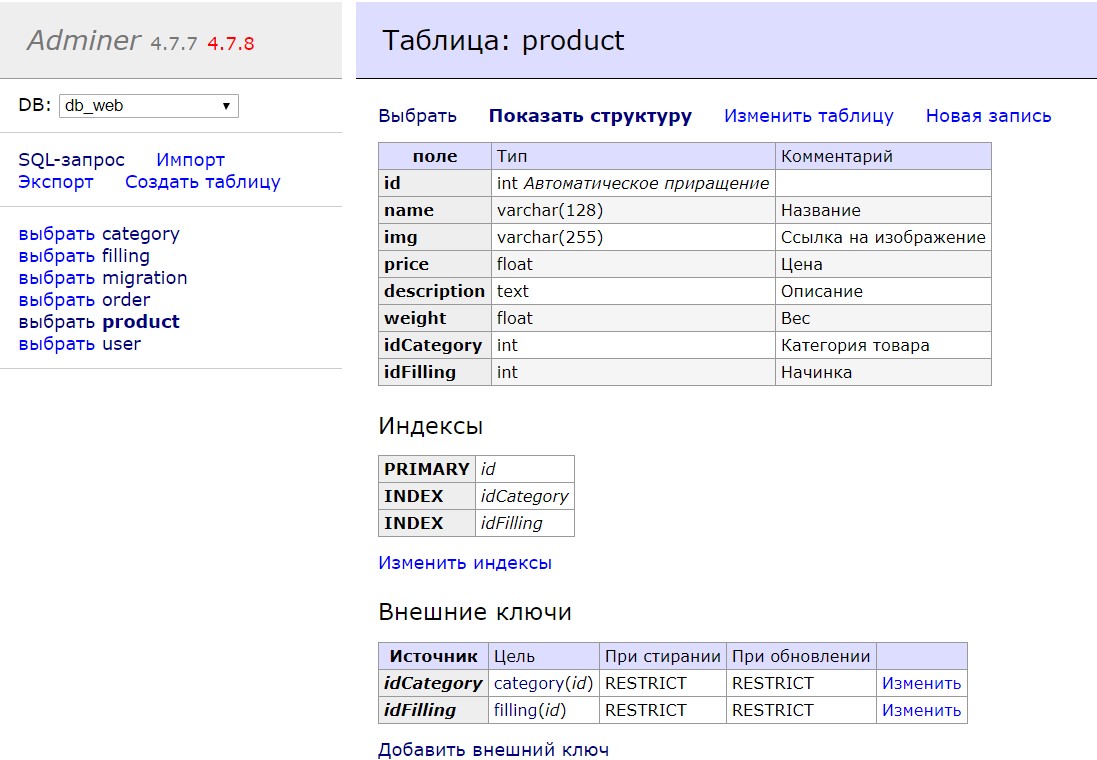
*php*use yii\db\Migration;  
  
/\*\*  
 \* Handles the creation of table `{{%filling}}`.  
 \*/  
class m201210\_112121\_create\_filling\_table extends Migration  
{  
 /\*\*  
 \* {@inheritdoc}  
 \*/  
 public function safeUp()  
 {  
 $this->createTable('{{%filling}}', [  
 'id' => $this->primaryKey(),  
 'name' => $this->string(*100*)->notNull()->comment('Название'),  
 ]);  
 }  
  
 /\*\*  
 \* {@inheritdoc}  
 \*/  
 public function safeDown()  
 {  
 $this->dropTable('{{%filling}}');  
 }  
}

<?*php*use yii\db\Migration;  
  
/\*\*  
 \* Handles the creation of table `{{%product}}`.  
 \*/  
class m201210\_114515\_create\_product\_table extends Migration  
{  
 /\*\*  
 \* {@inheritdoc}  
 \*/  
 public function safeUp()  
 {  
 $this->createTable('{{%product}}', [  
 'id' => $this->primaryKey()->notNull(),  
 'name' => $this->string(*128*)->notNull()->comment('Название'),  
 'img' => $this->string()->notNull()->comment('Ссылка на изображение'),  
 'price' => $this->float()->notNull()->comment('Цена'),  
 'description' => $this->text()->notNull()->comment('Описание'),  
 'weight' => $this->float()->notNull()->comment('Вес'),  
 'idCategory' => $this->integer()->notNull()->comment('Категория товара'),  
 'idFilling' => $this->integer()->notNull()->comment('Начинка'),  
  
 ]);  
  
 $this->addForeignKey('fk-product-idCategory', '{{%product}}', 'idCategory', '{{%category}}', 'id', 'NO ACTION', 'NO ACTION');  
 $this->addForeignKey('fk-product-idFilling', '{{%product}}', 'idFilling', '{{%filling}}', 'id', 'NO ACTION', 'NO ACTION');  
 }  
  
 /\*\*  
 \* {@inheritdoc}  
 \*/  
 public function safeDown()  
 {  
 $this->dropForeignKey('fk-product-idCategory', '{{%product}}');  
 $this->dropForeignKey('fk-product-idFilling', '{{%product}}');  
 $this->dropTable('{{%product}}');  
 }  
}

<?*php*use yii\db\Migration;  
  
/\*\*  
 \* Handles the creation of table `{{%user}}`.  
 \*/  
class m201210\_121311\_create\_user\_table extends Migration  
{  
 /\*\*  
 \* {@inheritdoc}  
 \*/  
 public function safeUp()  
 {  
 $this->createTable('{{%user}}', [  
 'id' => $this->primaryKey(),  
 'name' => $this->string(*150*)->notNull()->comment('Имя'),  
 'password' => $this->string(*50*)->notNull()->comment('Пароль'),  
 'email' => $this->string(*100*)->comment('Электронная почта'),  
 'telephone' => $this->string(*100*)->comment('Номер телефона'),  
 'dateBirth' => $this->dateTime()->comment('Дата рождения'),  
 'address' => $this->string(*200*)->comment('Адрес'),  
 ]);  
 }  
  
 /\*\*  
 \* {@inheritdoc}  
 \*/  
 public function safeDown()  
 {  
 $this->dropTable('{{%user}}');  
 }  
}

<?*php*use yii\db\Migration;  
  
/\*\*  
 \* Handles the creation of table `{{%order}}`.  
 \*/  
class m201210\_122133\_create\_order\_table extends Migration  
{  
 /\*\*  
 \* {@inheritdoc}  
 \*/  
 public function safeUp()  
 {  
 $this->createTable('{{%order}}', [  
 'id' => $this->primaryKey(),  
 'idProduct' => $this->integer()->notNull()->comment('Продукт'),  
 'idUser' => $this->integer()->notNull()->comment('Пользователь'),  
 'price' => $this->float()->comment('Сумма заказа'),  
 'orderDate' => $this->dateTime()->comment('Дата заказа'),  
 'status' => $this->string(*100*)->comment('Статус'),  
 ]);  
  
 $this->addForeignKey('fk-order-idProduct', '{{%order}}', 'idProduct', '{{%product}}', 'id', 'NO ACTION', 'NO ACTION');  
 $this->addForeignKey('fk-order-idUser', '{{%order}}', 'idUser', '{{%user}}', 'id', 'NO ACTION', 'NO ACTION');  
 }  
  
 /\*\*  
 \* {@inheritdoc}  
 \*/  
 public function safeDown()  
 {  
 $this->dropForeignKey('fk-order-idProduct', '{{%order}}');  
 $this->dropForeignKey('fk-order-idUser', '{{%order}}');  
 $this->dropTable('{{%order}}');  
 }  
}

После применении команды docker-compose run php yii migrate/up проверили в adminer созданные таблицы:



Далее с помощью генератора моделей yii code generator были сгенерированы соответствующие модели. Для примера приведем код модели Product.php:

<?*php*namespace app\modules\v1\models;  
  
use Yii;  
  
/\*\*  
 \* This is the model class for table "product".  
 \*  
 \* @property int $id  
 \* @property string $name Название  
 \* @property string $img Ссылка на изображение  
 \* @property float $price Цена  
 \* @property string $description Описание  
 \* @property float $weight Вес  
 \* @property int $idCategory Категория товара  
 \* @property int $idFilling Начинка  
 \*  
 \* @property Order[] $orders  
 \* @property Category $idCategory0  
 \* @property Filling $idFilling0  
 \*/  
class Product extends \yii\db\ActiveRecord  
{  
 /\*\*  
 \* {@inheritdoc}  
 \*/  
 public static function tableName()  
 {  
 return 'product';  
 }  
  
 /\*\*  
 \* {@inheritdoc}  
 \*/  
 public function rules()  
 {  
 return [  
 [['name', 'img', 'price', 'description', 'weight', 'idCategory', 'idFilling'], 'required'],  
 [['price', 'weight'], 'number'],  
 [['description'], 'string'],  
 [['idCategory', 'idFilling'], 'integer'],  
 [['name'], 'string', 'max' => *128*],  
 [['img'], 'string', 'max' => *255*],  
 [['idCategory'], 'exist', 'skipOnError' => *true*, 'targetClass' => Category::className(), 'targetAttribute' => ['idCategory' => 'id']],  
 [['idFilling'], 'exist', 'skipOnError' => *true*, 'targetClass' => Filling::className(), 'targetAttribute' => ['idFilling' => 'id']],  
 ];  
 }  
  
 /\*\*  
 \* {@inheritdoc}  
 \*/  
 public function attributeLabels()  
 {  
 return [  
 'id' => 'ID',  
 'name' => 'Название',  
 'img' => 'Ссылка на изображение',  
 'price' => 'Цена',  
 'description' => 'Описание',  
 'weight' => 'Вес',  
 'idCategory' => 'Категория товара',  
 'idFilling' => 'Начинка',  
 ];  
 }  
  
 /\*\*  
 \* Gets query for [[Orders]].  
 \*  
 \* @return \yii\db\ActiveQuery  
 \*/  
 public function getOrders()  
 {  
 return $this->hasMany(Order::className(), ['idProduct' => 'id']);  
 }  
  
 /\*\*  
 \* Gets query for [[IdCategory0]].  
 \*  
 \* @return \yii\db\ActiveQuery  
 \*/  
 public function getIdCategory0()  
 {  
 return $this->hasOne(Category::className(), ['id' => 'idCategory']);  
 }  
  
 /\*\*  
 \* Gets query for [[IdFilling0]].  
 \*  
 \* @return \yii\db\ActiveQuery  
 \*/  
 public function getIdFilling0()  
 {  
 return $this->hasOne(Filling::className(), ['id' => 'idFilling']);  
 }  
}

**Вывод:** в результате выполнения лабораторной работы изучили основы взаимодействия web-приложения с базой данных, спроектировали базу данных для хранения информации приложения (продукты, категории, начинки для продуктов, заказы, пользователи) и сгенерировали соответствующие модели.