**ПРИЛОЖЕНИЕ А**

*(обязательное)*

Код программного средства (frontend)

OrderControllep.php:

<?php

namespace frontend\controllers;

use frontend\models\Order;

use frontend\models\Card;

use frontend\models\Client;

use frontend\models\OrderModel;

use frontend\models\Product;

use Yii;

use yii\data\ActiveDataProvider;

use yii\web\Controller;

use yii\web\ForbiddenHttpException;

use yii\web\NotFoundHttpException;

/\*\*

\* Class OrderController

\* @package frontend\controllers

\*/

class OrderController extends Controller

{

/\*\*

\* @return string

\*/

public function actionIndex()

{

$ids = Yii::$app->order->get();

$query = Product::find()

->select([

'id',

'type',

'amount',

'title',

'image',

])

->where([

'id' => $ids,

'status' => Product::STATUS\_ACTIVE,

]);

$dataProvider = new ActiveDataProvider([

'query' => $query,

'sort' => false,

'pagination' => false,

]);

return $this->render('index', [

'dataProvider' => $dataProvider,

]);

}

public function actionUserInfo()

{

$user = Yii::$app->getUser();

if ($user->getIsGuest()) {

throw new ForbiddenHttpException('Извините!');

}

$model = $user->getIdentity();

if ($model->load(Yii::$app->request->post()) && $model->save()) {

return $this->redirect(['pay']);

}

return $this->render('user\_info', [

'model' => $model,

]);

}

public function actionPay()

{

$model = new Card();

if ($model->load(Yii::$app->request->post()) && $model->pay()) {

$order\_id = $this->saveOrder(

Yii::$app->getUser()->getId(),

Yii::$app->order->get()

);

Yii::$app->order->clear();

return $this->redirect(['success', 'id' => $order\_id]);

}

return $this->render('pay', ['model' => $model]);

}

/\*\*

\* @param int $client\_id

\* @param array $products\_id

\* @return int

\*/

protected function saveOrder($client\_id, array $products\_id)

{

return (new OrderModel($client\_id, $products\_id))->save();

}

/\*\*

\* @param int $id

\* @return \yii\web\Response

\*/

public function actionAdd($id)

{

$exist = Product::find()

->where([

'id' => $id,

'status' => Product::STATUS\_ACTIVE,

])

->exists();

if ($exist === false) {

throw new NotFoundHttpException('Запрошенная страница не существует.');

}

$this->userLogin();

Yii::$app->order->add($id);

return $this->redirect(['product/view', 'id' => $id]);

}

public function actionSuccess($id)

{

return $this->render('success', ['id' => $id]);

}

protected function userLogin()

{

$user = Yii::$app->getUser();

if ($user->getIsGuest() === false) {

return;

}

$model = new Client();

$model->generateAuthKey();

$model->save(false);

$user->login($model);

}

/\*\*

\* @param int $id

\* @return \yii\web\Response

\*/

public function actionDelete($id)

{

Yii::$app->order->delete($id);

return $this->redirect(['index']);

}

public function actionView($id)

{

$model = Order::find()

->where(['id' => $id])

->one();

$dataProvider = new ActiveDataProvider([

'query' => $model->getProducts(),

'pagination' => false,

'sort' => false,

]);

return $this->render('view', [

'model' => $model,

'dataProvider' => $dataProvider

]);

}

}

ProductController.php:

<?php

namespace frontend\controllers;

use frontend\models\Product;

use Yii;

use yii\data\ActiveDataProvider;

use yii\web\Controller;

use yii\web\NotFoundHttpException;

/\*\*

\* Class ProductController

\* @package frontend\controllers

\*/

class ProductController extends Controller

{

/\*\*

\* @param int $type

\* @return string

\*/

public function actionIndex($type)

{

$query = Product::find()

->select([

'id',

'type',

'amount',

'title',

'image',

])

->where([

'type' => $type,

'status' => Product::STATUS\_ACTIVE,

])

->orderBy(['id' => SORT\_DESC]);

$dataProvider = new ActiveDataProvider([

'query' => $query,

'sort' => false,

'pagination' => [

'pageSize' => 24,

],

]);

return $this->render('index', [

'dataProvider' => $dataProvider,

]);

}

/\*\*

\* @param int $id

\* @return string

\* @throws NotFoundHttpException

\*/

public function actionView($id)

{

$model = $this->findModel($id);

return $this->render('view', [

'model' => $model,

'inOrder' => Yii::$app->order->inOrder($id)

]);

}

/\*\*

\* @param int $id

\* @return Product|null

\* @throws NotFoundHttpException

\*/

protected function findModel($id)

{

if (($model = Product::findOne($id)) !== null) {

return $model;

}

throw new NotFoundHttpException('The requested page does not exist.');

}

}

SiteController.php:

<?php

namespace frontend\controllers;

use frontend\models\Product;

use yii\web\Controller;

/\*\*

\* Site controller

\*/

class SiteController extends Controller

{

/\*\*

\* {@inheritdoc}

\*/

public function actions()

{

return [

'error' => [

'class' => 'yii\web\ErrorAction',

],

];

}

/\*\*

\* Displays homepage.

\*

\* @return mixed

\*/

public function actionIndex()

{

$models = Product::find()

->where(['status' => Product::STATUS\_ACTIVE])

->orderBy(['id' => SORT\_DESC])

->limit(8)

->all();

return $this->render('index', [

'models' => $models

]);

}

}

ChairMassager.php:

<?php

namespace frontend\models\products;

class ChairMassager extends \common\models\products\ChairMassager

{

public function getParams()

{

return [

'voltage' => $this->voltage[0] . ' - ' . $this->voltage[1] . ' В',

'frequency' => $this->frequency[0] . ' - ' . $this->frequency[1] . ' Гц',

'power' => $this->power . ' Вт',

'weight' => $this->weight . ' Кг',

'is\_remote\_control'=> $this->is\_remote\_control ,

'hand\_programs' => implode(', ', $this->hand\_programs),

'auto\_program' => implode(', ', $this->auto\_program),

'colors' => implode(', ', $this->colors),

'person\_max\_weight' => $this->person\_max\_weight . ' См',

'person\_max\_height' => $this->person\_max\_height . ' См',

'size\_folded' => $this->size\_folded . ' См',

'size\_decomposed' => $this->size\_decomposed . ' См',

'guarantee' => $this->guarantee,

];

}

}

Handmassager.php:

<?php

namespace frontend\models\products;

class HandMassager extends \common\models\products\HandMassager

{

public function getParams()

{

return [

'voltage' => $this->voltage[0] . ' - ' . $this->voltage[1] . ' В',

'frequency' => $this->frequency[0] . ' - ' . $this->frequency[1] . ' Гц',

'power' => $this->power . ' Вт',

'weight' => $this->weight . ' Грамм',

'size' => $this->size . ' См',

'number\_of\_programs' => $this->number\_of\_programs,

'is\_timer\_exist' => $this->is\_timer\_exist,

];

}

}

Card.php:

<?php

namespace frontend\models;

use yii\base\Model;

/\*\*

\* Class Card

\* @package frontend\models

\*/

class Card extends Model

{

/\*\* @var string \*/

public $code1;

/\*\* @var string \*/

public $code2;

/\*\* @var string \*/

public $code3;

/\*\* @var string \*/

public $code4;

/\*\* @var int \*/

public $month;

/\*\* @var int \*/

public $year;

/\*\* @var int \*/

public $cvc;

/\*\*

\* @return array

\*/

public function rules()

{

return [

[['code1', 'code2', 'code3', 'code4'], 'required'],

[['code1', 'code2', 'code3', 'code4'], 'string', 'min' => 4, 'max' => 4],

[['code1', 'code2', 'code3', 'code4'], 'match', 'pattern' => '~[0-9]+~'],

[['month'], 'required'],

[['month'], 'integer', 'min' => 1, 'max' => 12],

[['year'], 'required'],

[['year'], 'integer', 'min' => date('Y'), 'max' => date('Y') + 3],

[['cvc'], 'required'],

[['cvc'], 'string', 'min' => 3, 'max' => 3],

[['cvc'], 'match', 'pattern' => '~[0-9]+~'],

];

}

/\*\*

\* @return bool

\*/

public function pay()

{

return $this->validate();

}

}

Client.php:

<?php

namespace frontend\models;

use yii\base\NotSupportedException;

use yii\web\IdentityInterface;

/\*\*

\* Class Client

\* @package frontend\models

\*/

class Client extends \common\models\Client implements IdentityInterface

{

/\*\*

\* {@inheritdoc}

\*/

public function rules()

{

return [

[['address', 'phone', 'email', 'full\_name'], 'required'],

[['address', 'phone', 'full\_name'], 'string', 'max' => 255],

[['email'], 'email'],

];

}

/\*\*

\* {@inheritdoc}

\*/

public static function findIdentity($id)

{

return static::findOne(['id' => $id]);

}

/\*\*

\* {@inheritdoc}

\*/

public static function findIdentityByAccessToken($token, $type = null)

{

Throw new NotSupportedException

('"findIdentityByAccessToken" is not implemented.');

}

/\*\*

\* {@inheritdoc}

\*/

public function getId()

{

return $this->getPrimaryKey();

}

/\*\*

\* {@inheritdoc}

\*/

public function getAuthKey()

{

return $this->auth\_key;

}

/\*\*

\* {@inheritdoc}

\*/

public function validateAuthKey($authKey)

{

return $this->getAuthKey() === $authKey;

}

}

Order.php:

<?php

namespace frontend\models;

class Order extends \common\models\Order

{

public function getProducts()

{

$query = parent::getProducts();

$query->modelClass = Product::class;

return $query;

}

}

OrderModel.php:

<?php

namespace frontend\models;

use common\models\Order;

use common\models\OrderProduct;

/\*\*

\* Class OrderModel

\* @package frontend\models

\*/

class OrderModel

{

private $products\_id;

private $client\_id;

public function \_\_construct($client\_id, $products\_id)

{

$this->client\_id = $client\_id;

$this->products\_id = $products\_id;

}

public function save()

{

$order = new Order();

$order->client\_id = $this->client\_id;

$order->status = Order::STATUS\_NEW;

$order->is\_paid = true;

$order->save(false);

$product\_amount = Product::find()

->select(['amount', 'id'])

->where(['id' => $this->products\_id])

->indexBy('id')

->column();

foreach ($this->products\_id as $id) {

$product = new OrderProduct();

$product->product\_id = $id;

$product->order\_id = $order->id;

$product->amount = $product\_amount[$id];

$product->save(false);

}

return $order->id;

}

}

Product.php:

<?php

namespace frontend\models;

use common\models\Order;

use common\models\OrderProduct;

/\*\*

\* Class OrderModel

\* @package frontend\models

\*/

class OrderModel

{

private $products\_id;

private $client\_id;

public function \_\_construct($client\_id, $products\_id)

{

$this->client\_id = $client\_id;

$this->products\_id = $products\_id;

}

public function save()

{

$order = new Order();

$order->client\_id = $this->client\_id;

$order->status = Order::STATUS\_NEW;

$order->is\_paid = true;

$order->save(false);

$product\_amount = Product::find()

->select(['amount', 'id'])

->where(['id' => $this->products\_id])

->indexBy('id')

->column();

foreach ($this->products\_id as $id) {

$product = new OrderProduct();

$product->product\_id = $id;

$product->order\_id = $order->id;

$product->amount = $product\_amount[$id];

$product->save(false);

}

return $order->id;

}

}