Zeal	Файл проекта	Код
Настройка проекта		
	Config/db.php	'class' => 'yii\db\Connection', 'dsn' => 'mysql:host=localhost;dbname=yiiapp2', 'username' => 'root', 'password' => '', 'charset' => 'utf8',
	Config/web.php	8 строка 'language'=>'ru-RU', 18 строка 'cookieValidationKey' => 'randomstring', 71 строка 'allowedIPs' => ['*'], 19 строка 'baseUrl'=>'', 48-54 раскомментировать 'urlManager' => ['enablePrettyUrl' => true, 'showScriptName' => false, 'rules' => [ ],],
	Из папки web скопировать .htaccess в корень проекта В скопированном файле заменить	RewriteEngine On RewriteRule ^(.*)$ web/$1
Создание миграций ВАЖНО!!! Создавать миграции в логическом порядке – сначала таблицы с первичным ключевым полем, а только ПОТОМ таблицы с внешним ключом и связи!!! Если миграция не получается, так как нарушен порядок, то сначала нужно переименовать файлы миграции, чтобы они шли по порядку, а затем внутри каждой миграции ИМЯ ФАЙЛА==ИМЯ МИГРАЦИИ		
	php yii migrate/create create_role_table	public function safeUp()

		{ $this->createTable('{{%role}}', ['id' => $this->primaryKey(), 'code' => $this->string()->unique()->notNull(), 'name' => $this->string()->notNull(),]); $this->insert('{{%role}}', ['code' => 'user', 'name' => 'Зарегистрированный пользователь',]); $this->insert('{{%role}}', ['code' => 'admin', 'name' => 'Администратор']); }
Guides/database migration	php yii migrate/create create_user_table отредактировать файл миграции в папке migration	public function safeUp() { $this->createTable('{{%user}}', ['id' => $this->primaryKey(), 'name' => $this->string()->notNull(), 'surname' => $this->string()->notNull(), 'patronymic' => $this->string()->null(), 'username' => $this->string()->unique()->notNull(), 'email' => $this->string()->unique()->notNull(), 'password' => $this->string()->notNull(), 'role_id' => $this->integer()->defaultValue(1),]); $this->createIndex('idx-user-role_id', 'user', 'role_id'); $this->addForeignKey('fk-user-role_id', 'user', 'role_id',

		'role', 'id', 'CASCADE'); $this->insert('{{%user}}', ['name' => 'Иванов', 'surname' => 'Иван', 'patronymic' => 'Иванович', 'username' => 'admin', 'email' => 'admin@admin.ru', 'password' => md5('admin00'), 'role_id' => 2,]); }
	php yii migrate/create create_category_table	public function safeUp() { $this->createTable('{{%category}}', ['id' => $this->primaryKey(), 'name' => $this->string()->notNull(),]); $this->insert('{{%category}}', ['name' => 'Лазерные принтеры',]); $this->insert('{{%category}}', ['name' => 'Струйные принтеры',]); $this->insert('{{%category}}', ['name' => 'Термопринтеры',]); }
	php yii migrate/create create_product_table	public function safeUp() { $this->createTable('{{%product}}', [

```
            'id' => $this->primaryKey(),
            'date' => $this->timestamp(),
            'name' => $this->string()->notNull(),
            'file' => $this->string()->null(),
            'count' => $this->integer()->defaultValue(0),
            'price' => $this->decimal()->defaultValue(0),
            'year' => $this->integer()->defaultValue(2023),
            'model' => $this->string()->null(),
            'country' => $this->string()->null(),
            'category_id' => $this->integer()->defaultValue(1),
        ]);
        $this->createIndex(
            'idx-product-category_id',
            'product',
            'category_id'
        );
        $this->addForeignKey(
            'fk-product-category_id',
            'product',
            'category_id',
            'category',
            'id',
            'CASCADE'
        );
        $this->insert('{{%product}}', [
            'name' => 'Принтер 1',
            'count' => 10,
        ]);
        $this->insert('{{%product}}', [
            'name' => 'Принтер 2',
            'count' => 5,
        ]);
        $this->insert('{{%product}}', [
            'name' => 'Принтер 3',
            'count' => 50,
        ]);
    }
```

	php yii migrate/create create_status_table	public function safeUp() { $this->createTable('{{%status}}', ['id' => $this->primaryKey(), 'code' => $this->string()->unique()->notNull(), 'name' => $this->string()->notNull(),]); $this->insert('{{%status}}', ['name' => 'Новый', 'code' => 'new',]); $this->insert('{{%status}}', ['name' => 'Подтвержденный', 'code' => 'confirm',]); $this->insert('{{%status}}', ['name' => 'Отклоненный', 'code' => 'rejected',]); }
	php yii migrate/create create_order_table	public function safeUp() { $this->createTable('{{%order}}', ['id' => $this->primaryKey(), 'date' => $this->timestamp(), 'user_id' => $this->integer()->notNull(), 'status_id' => $this->integer()->notNull()->defaultValue(1), 'rejection_reason' => $this->text()->null(),]); $this->createIndex('idx-order-user_id', 'order', 'user_id'

		); $this->addForeignKey('fk-order-user_id', 'order', 'user_id', 'user', 'id', 'CASCADE'); $this->createIndex('idx-order-status_id', 'order', 'status_id'); $this->addForeignKey('fk-order-status_id', 'order', 'status_id', 'status', 'id', 'CASCADE'); }
	php yii migrate/create create_product_order_table	public function safeUp() { $this->createTable('{{%product_order}}', ['id' => $this->primaryKey(), 'order_id' => $this->integer()->notNull(), 'product_id' => $this->integer()->notNull(), 'count' => $this->integer()->notNull()->defaultValue(1), 'price' => $this->integer()->notNull()->defaultValue(0),]); $this->createIndex(

		'idx-product_order-order_id', 'product_order', 'order_id'); $this->addForeignKey('fk-product_order-order_id', 'product_order', 'order_id', 'order', 'id', 'CASCADE'); $this->createIndex('idx-product_order-product_id', 'product_order', 'product_id'); $this->addForeignKey('fk-product_order-product_id', 'product_order', 'product_id', 'product', 'id', 'CASCADE'); }
	php yii migrate/create create_cart_table	public function safeUp() { $this->createTable('{{%cart}}', ['id' => $this->primaryKey(), 'user_id' => $this->integer()->notNull(), 'product_id' => $this->integer()->notNull(), 'count' => $this->integer()->notNull()->defaultValue(1),]);

		$this->createIndex('idx-cart-user_id', 'cart', 'user_id'); $this->addForeignKey('fk-cart-user_id', 'cart', 'user_id', 'user', 'id', 'CASCADE'); $this->createIndex('idx-cart-product_id', 'cart', 'product_id'); $this->addForeignKey('fk-cart-product_id', 'cart', 'product_id', 'product', 'id', 'CASCADE'); }
	Php yii migrate	
Создание моделей		
	http://yiiapp2/gii Model Creation	

	Database Connection: db Table Name: *	
Авторизация		
Guides/Authentification Импортировать интерфейс IdentityInterface к классу ActiveRecord use yii\web\IdentityInterface; class User extends ActiveRecord implements IdentityInterface	Models/User.php	use yii\web\IdentityInterface; class User extends \yii\db\ActiveRecord implements IdentityInterface
Guides/Authentification (2 статья) Скопировать функции, поставить заглучшки public static function findIdentityByAccessToken($token, $type = null) public function getAuthKey() public function validateAuthKey($authKey)	Models/User.php Models/UserOld.php Скопировать функции	public static function findIdentity($id) { return static::findOne($id); } public static function findIdentityByAccessToken($token, $type = null) { return null; } public function getId() { return $this->id; } public function getAuthKey() { return null; } public function validateAuthKey($authKey) { return false; }

		public function validatePassword($password) { return $this->password === md5($password); } public function isAdmin() { return $this->role->code ==='admin'; }
Guides/Authentification (2 статья) **public function beforeSave**($insert) { **if** (**parent**::beforeSave($insert)) { **if** (**$this**->isNewRecord) { **$this**->auth_key = \Yii::$app->security->generateRandomString(); } **return true**; } **return false**; }	Models/UserOld.php Скопировать функции	public function beforeSave($insert) { $this->password= md5($this->password); return parent::beforeSave($insert); }
	Русифицировать ссылку на страницу входа (файл views/layouts/main.php):	Yii::$app->user->isGuest ? ['label' => 'Вход', 'url' => ['/site/login']] : '<li class="nav-item">' . Html::beginForm(['/site/logout']) . Html::submitButton('Выход (' . Yii::$app->user->identity->username . ')', ['class' => 'nav-link btn btn-link logout']) . Html::endForm() . ' ']]);

	Привести подписи к форме авторизации в файле models/LoginForm.php в соответствии с заданием	public function validatePassword($attribute, $params) { if (!$this->hasErrors()) { $user = $this->getUser(); if (!$user || !$user->validatePassword($this->password)) { $this->addError($attribute, 'Неверный логин или пароль'); } } } public function attributeLabels() { return ['username' => 'login', 'password' => 'password', 'rememberMe' => 'Запомнить в системе',]; }
	Русифицировать подписи на странице авторизации в файле views/site/login.php:	$this->title = 'Вход'; $this->params['breadcrumbs'][] = $this->title; ?> <?= Html::encode($this->title) ?>
Регистрация		
	Перейти в кодогенератор **Gii(адрес вида login-m1.wsr.ru/gii)** и сгенерировать **CRUD** для модели **User**: Grud Generator Model Class : app\model\User Controller Class: App\controller\UserController	Проверяем hаботу функции сохранения нового пользователя beforeSave Для этого в файле **models/User.php** нужно переопределить метод **beforeSave:** public function beforeSave($insert) { $this -> password = md5($this-> password); return parent::beforeSave($insert); }

	Зарегистрировать нового пользователя user/createю Проверить новой записи в базе данных. Отредактировать форму в соответствии с заданием 1.	файл **views/user/_form.php** убрав лишние поля (нельзя дать пользователю менять роль, добавить поле повторения пароля и знакомства с правилами- их нет в базе данных): <?php $form = ActiveForm::begin(); ?> <?= $form->field($model, 'name')->textInput(['maxlength' => true]) ?> <?= $form->field($model, 'surname')->textInput(['maxlength' => true]) ?> <?= $form->field($model, 'patronymic')->textInput(['maxlength' => true]) ?> <?= $form->field($model, 'username', ['enableAjaxValidation' => true])->textInput(['maxlength' => true]) ?> <?= $form->field($model, 'email')->textInput(['maxlength' => true]) ?> <?= $form->field($model, 'password')->passwordInput(['maxlength' => true]) ?> <?= $form->field($model, 'password_repeat')->passwordInput(['maxlength' => true]) ?> <?= $form->field($model, 'rules')->checkbox() ?> <?= Html::submitButton('Save', ['class' => 'btn btn-success']) ?> <?php ActiveForm::end(); ?>
	Настроить ссылки и перенаправления	Сделать переход по правильной ссылке **в views/layouts/main.php** ['label' => 'Регистрация', 'url' => ['/user/create'], 'visible'=>Yii::$app->user->isGuest], Настроить, чтобы после регистрации пользователь переходил на страницу авторизации при успешной регистрации в файле controllers/UserController.php, правил метод actionCreate if ($this->request->isPost) { if ($model->load($this->request->post()) && $model->save()) { return $this->redirect(['site/login']); } По заданию при выходе пользователь переходит на страницу О нас в SiteController в метод actionLogout вместо return Home() на return $this->redirect(['site/about']);
	Валидация формы регистрации	
	Нужно в модель User Добавть новые параметры	Model/User.php и там создаем два свойства public $password_repeat; public $rules;

Zeal=Guides-Core Validator Статья match [// checks if "username" starts with a letter and contains only word characters ['username', 'match', 'pattern' => '/^[a-z]\w*$/i']] compare [// validates if the value of "password" attribute equals to that of "password_repeat" ['password', 'compare'], // same as above but with explicitly specifying the attribute to compare with ['password', 'compare', 'compareAttribute' => 'password_repeat'], // validates if age is greater than or equal to 30 ['age', 'compare', 'compareValue' => 30, 'operator' => '>=', 'type' => 'number'],]	name – обязательное поле, разрешенные символы (кириллица, пробел и тире); surname – обязательное поле, разрешенные символы (кириллица, пробел и тире); patronymic – не обязательное поле, разрешенные символы (кириллица, пробел и тире); login – обязательное и уникальное поле, разрешенные символы (латиница, цифры и тире); password – обязательное поле, не менее 6-ти символов; password_repeat – обязательное поле, должно совпадать с полем password; rules - согласие с правилами регистрации.	[['name', 'surname', 'username'], 'match', 'pattern' => '/^[а-яА-Я0-9-]*$/u','message'=>'Только буквы кирилицы'], ['username', 'match', 'pattern' => '/^[a-zA-Z0-9-]*$/i','message'=>'Только латиница'], [['password'],'string', 'min' => 6], ['password_repeat', 'compare', 'compareAttribute' => 'password'], ['rules','compare','compareValue' => 1,'message'=>'Примите условия регистрации'],
	Привести форму в соответствии с требованиями к заданию: Проверка пароля Согласие с обработкой персональных данных	 <?php $form = ActiveForm::begin(); ?> <?= $form->field($model, 'name')->textInput(['maxlength' => true]) ?> <?= $form->field($model, 'surname')->textInput(['maxlength' => true]) ?> <?= $form->field($model, 'patronymic')->textInput(['maxlength' => true]) ?>

	Добавить недостающие поля в представление формы регистрации в файле **views/user/_form.php**	<?= $form->field($model, 'username', ['enableAjaxValidation' => true])->textInput(['maxlength' => true]) ?> <?= $form->field($model, 'email')->textInput(['maxlength' => true]) ?> <?= $form->field($model, 'password')->passwordInput(['maxlength' => true]) ?> <?= $form->field($model, 'password_repeat')->passwordInput(['maxlength' => true]) ?> <?= $form->field($model, 'rules')->checkbox() ?> <?= Html::submitButton('Save', ['class' => 'btn btn-success']) ?> <?php ActiveForm::end(); ?>
	Организация доступа для авторизированного доступа Заходим в UserController и добавим правила	return array_merge(parent::behaviors(), ['access' => ['class' => AccessControl::class, 'only' => ['create'], 'rules' => [['actions' => ['create'], 'allow' => true, 'roles' => ['?'],],],], 'verbs' => ['class' => VerbFilter::className(), 'actions' => ['delete' => ['POST'],],],]);
		Импортировать use yii\filters\AccessControl;

	Для гостей сделать видимой ссылку на регистрацию Зайдем в views/layouts/main.php и настроим свойство visible	['label' => 'Регистрация', 'url' => ['/user/create'], 'visible'=>Yii::$app->user->isGuest],
Создание корзины		
	Перейти в кодогенератор **Gii(адрес вида login-m1.wsr.ru/gii)** и сгенерировать **CRUD** для модели **Cart**: Grud Generator Model Class : app\models\Cart Controller Class: App\controllers\CartsController	Сделать так, чтобы доступ в корзину имел только зарегистрированный пользователь. 1. Из файла **controllers/SiteController.php** скопировать правила доступа к методу logout (только для авторизированного пользователя) в файл **controllers/CartController.php:** public function behaviors() { return ['access' => ['class' => AccessControl::class, 'only' => ['logout'], 'rules' => [['actions' => ['logout'], 'allow' => true, 'roles' => ['@'],],],], 'verbs' => ['class' => VerbFilter::class, 'actions' => ['logout' => ['post'],],],]; }
Zeal – статья Working with Data Keys из **Data Providers**:	Сделать так, чтобы для каждого пользователя отображалась корзина только с его товарами	В методе index в файле controllers/CartController.php добавить условие поиска товаров в корзине только для текущего пользователя public function actionIndex() {

		$dataProvider = new ActiveDataProvider(['query' => Cart::find()->where(['user_id' => \Yii::$app->user->identity->id]), /* 'pagination' => ['pageSize' => 50], 'sort' => ['defaultOrder' => ['id' => SORT_DESC,]], */]); return $this->render('index', ['dataProvider' => $dataProvider,]); } НЕ ЗАБЫТЬ ИМПОРТИРОВАТЬ классы use yii\filters\AccessControl; use yii\data\ActiveDataProvider; use Yii;
Используя код из Zeal статьи **Data Widgets** раздел **Data column** отредактировать вид корзины пользователя: 	Структуру корзины сделать согласно заданию: Сделать столбики наименование товара, количество и кнопки управления количеством товара (добавление, удаление)	Views/cart/index.php <?= GridView::widget(['dataProvider' => $dataProvider, 'columns' => [['class' => 'yii\grid\SerialColumn'], 'product.name', 'count', ['format' => 'raw', 'value' => function ($data) { return "<button onclick='addCart($data->product_id)' class='btn btn-success'>+</button> <button onclick='removeCart($data->product_id)' class='btn btn-success'>-</button>";

```
                },
            ],

        ],
    ]); ?>
 <?php

use app\models\Cart;
use yii\helpers\Html;
use yii\helpers\Url;
use yii\grid\ActionColumn;
use yii\grid\GridView;
use yii\widgets\Pjax;

/** @var yii\web\View $this */
/** @var yii\data\ActiveDataProvider $dataProvider */

$this->title = 'Корзина';
$this->params['breadcrumbs'][] = $this->title;
$this->registerJsFile(
    '@web/js/main.js',
    ['depends' => [\yii\web\JqueryAsset::class]]
);

?>
<div class="cart-index">

    <h1><?= Html::encode($this->title) ?></h1>
    <div class="info alert alert-primary"></div>
    <?php Pjax::begin(['id'=>'cart']) ?>
    <?= GridView::widget([
        'dataProvider' => $dataProvider,
        'columns' => [
            ['class' => 'yii\grid\SerialColumn'],
            'product.name',
            'count',
            [
                'format' => 'raw',
```

		'value' => function ($data) { return "<button onclick='addCart($data->product_id)' class='btn btn-success'>+</button> <button onclick='removeCart($data->product_id)' class='btn btn-success'>-</button>"; },],],]); ?> <?php Pjax::end() ?> <label for="inputPassword5" class="form-label">Password</label> <input type="password" id="inputPassword5" class="form-control" aria-describedby="passwordHelpBlock"> Введите пароль для оформления заказа <button onclick='byOrder()' class='btn btn-success'>Оформить заказ</button>
	Русифицировать подписи столбцов можно в файлах **models/Cart.php** и **models/Product.php:**	public function attributeLabels() { return ['id' => 'ID', 'user_id' => 'User ID', 'product_id' => 'Product ID', 'count' => 'Количество',]; } public function attributeLabels() { return ['id' => 'ID', 'date' => 'Дата', 'name' => 'Наименование',

<table>
<tr><td></td><td></td><td>'file' => 'Фото',
'count' => 'Количество',
'price' => 'Цена',
'year' => 'Год выпуска',
'model' => 'Модель',
'country' => 'Производитель',
'category_id' => 'Category ID',
];
}</td></tr>
<tr><td></td><td>1. Добавить в меню ссылку на корзину пользователя (файл views/layouts/main.php):</td><td>['label' => 'Корзина', 'url' => ['/cart/index'], 'visible'=>!Yii::$app->user->isGuest],</td></tr>
<tr><td colspan="3">Управление количенством товаров в корзине</td></tr>
<tr><td>Примеры задания условий выборки находятся в Zeal - ActiveRecord</td><td>Отредактировать метод actionCreate в файле controllers/CartController.php согласно требованиям задания.
//Работа с корзиной на стороне сервера</td><td>

```
public function actionCreate($id_product)
  {
    $product = Product::find()->where(['id' => $id_product])->andWhere(['>', 'count', 0])->one();

    if (!$product) {
      return "Такого продукта нет";
    }

    $itemInCart = Cart::find()
      ->where(['product_id' => $id_product])
      ->andWhere(['user_id' => \Yii::$app->user->identity->id])
      ->one();

    if (!$itemInCart) {
      $itemInCart = new Cart([
        'product_id' => $id_product,
        'user_id' => \Yii::$app->user->identity->id,
        'count' => 1
      ]);
      $itemInCart->save();
```

</td></tr>
</table>

<table>
<tr>
<td></td>
<td></td>
<td>

```
        return "Продукт добавлен. Количество товаров в корзине = $itemInCart->count";
    }

    if ($itemInCart->count + 1 > $product->count) {
        return "Нельзя больше добавить";
    }

    $itemInCart->count++;
    $itemInCart->save();
    return "Продукт добавлен. Количество товаров в корзине = $itemInCart->count";
}
```

Импортировать классы:
Use Yii;

```
use app\models\Cart;
```

</td>
</tr>
<tr>
<td>Zeal – Worcing with Clirnt Script – статья Registering script files

```
$this->registerJsFile(
    '@web/js/main.js',
    ['depends' => [\yii\web\JqueryAsset::class]]
);
```

</td>
<td>Работа с корзиной на стороне клиента
Views/cart/index.php</td>
<td>.

```
13  $this->title = 'Корзина';
14  $this->params['breadcrumbs'][] = $this->title;
15  $this->registerJsFile(
16      '@web/js/main.js',
17      ['depends' => [\yii\web\JqueryAsset::class]]
18  );
19
```

</td>
</tr>
<tr>
<td>Можно использовать обычную верстку div из бутстрап</td>
<td>Создать поле, в которое будем выводить сообщения об изменения в корзине (все наши return)</td>
<td>

```
21
22      <h1><?= Html::encode($this->title) ?></h1>
23      <div class="info alert alert-primary"></div>
24
25
```

</td>
</tr>
<tr>
<td>Пример функции мы берем из Zeal – jQuery-Ajax-jQuery.ajax - context</td>
<td>Создадим папку и файл web/js/main.js и прописать код для проверки функции</td>
<td>

```
function addCart(id_product) {
  $.ajax({
    method: 'GET',
    url: `/cart/create?id_product=${id_product}`,
```

</td>
</tr>
</table>

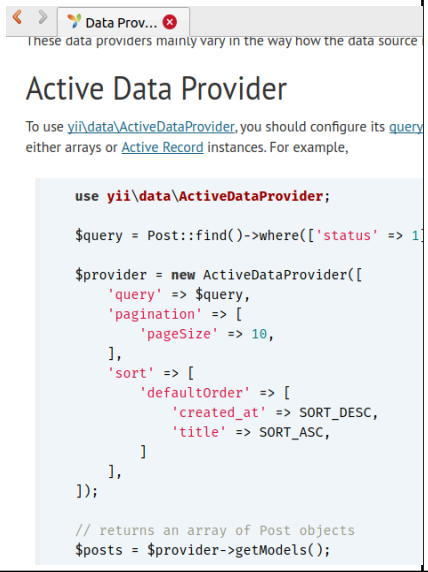
```
    }).done(function (message) {
        $.pjax.reload({
            container: '#cart'
        });
        $('.info').html(message);
        setTimeout(() => {
            $('.info').text('');
        }, 1000);
    });
}
```

```
div class="cart-index">

    <h1><?= Html::encode($this->title) ?></h1>
    <div class="info alert alert-primary"></div>

    <?php Pjax::begin(['id' => 'cart']) ?>
    <?= GridView::widget([
        'dataProvider' => $dataProvider,
        'columns' => [
            ['class' => 'yii\grid\SerialColumn'],
            'product.name',
            'count',
            [
                'format' => 'raw',
                'value' => function ($data) {
                    return "<button onclick='addCart($data->product_id)' class='btn
btn-success'>+</button>
                        <button onclick='removeCart($data->product_id)' class='btn btn-success'>-</button>";
                },
            ],
        ],
    ]); ?>
    <?php Pjax::end() ?>

Импортировать Pjax
```

		use yii\widgets\Pjax;
	Выполняем действия на стороне сервера 1. Для уменьшения товара в корзине нужно изменить метод **delete** (файл **controllers/CartController.php**) в соответствии с заданием: 1.Проверяем есть ли продукт в корзине 2.Если продукт найден проверяем, если количество при удалении станет ноль, то товар удаляем из корзины, если не ноль 3. Уменьшаем количество товара	public function actionDelete($id_product) { $itemInCart = Cart::find() ->where(['product_id' => $id_product]) ->andWhere(['user_id' => \Yii::$app->user->identity->id]) ->one(); if (!$itemInCart) { return "Такого продукта не найдено"; } if ($itemInCart->count-1==0) { $itemInCart->delete(); return "Продукт удален"; } $itemInCart->count--; $itemInCart->save(); return "Количество продуктов в корзине уменьшено"; }
	Выполняем действия на стороне клиента Добавить обработку сообщения в функции **removeCart** (файл **web/js/main.js**):	function removeCart(id_product) { $.ajax({ method: 'POST', url: `/cart/delete?id_product=${id_product}`, }).done(function (message) { $.pjax.reload({ container: '#cart' }); $('.info').html(message); setTimeout(() => { $('.info').text(''); }, 1000); }); }

Boostrap-forms	Создать форму для ввода пароля для оформления заказа gjckt	<?php Pjax::end() ?> <label for="inputPassword5" class="form-label">Password</label> <input type="password" id="inputPassword5" class="form-control" aria-describedby="passwordHelpBlock"> Введите пароль для оформления заказа <button onclick='byOrder()' class='btn btn-success'>Оформить заказ</button>
	Написали функцию для обработки пароля на стороне клиента в файле **web/js/main.js**:	function byOrder() { let password=$('#inputPassword5').val(); console.log(password); if (!password) { $('.info').html("Введите пароль"); return } $.ajax({ method: "GET", url: `/cart/by-order?password=${password}`, }) .done(function (message) { $.pjax.reload({ container: '#cart' }); $('.info').html(message); }) }
	Добавить метод для оформления заказа в файл **controllers/CartController.php**: ОБРАТИТЬ ВНИМАНИЕ НА НАПИСАНИЕ АДРЕСА by-order и методов byOrder. В самом конце	public function actionByOrder($password) { if (!\Yii::$app->user->identity->validatePassword($password)) { return "Пароль не верный"; } $itemInCart = \Yii::$app->user->identity->carts;

	перед последней закрывающейся фигурной скобкой: 1. Проверяет, правильно ли пользователь указал пароль 2. Все заказы из корзины мы перемещаем в переменную $itemInCart 3. Проверяем есть ли товары в корзине. 4. Создаем объект заказа 5. Организуем цикл для переноса товаров из корзины в заказ посредством таблицы ProductOrder	``` if (!$itemInCart) { return "Корзина пуста"; } $order = new Order(['user_id' => \Yii::$app->user->identity->id, 'status_id' => Status::find()->where(['code' => 'new'])->one()->id //'status_id' => 1]); $order->save(); foreach ($itemInCart as $item) { $itemInOrder = new ProductOrder(['order_id' => $order->id, 'product_id' => $item->product_id, 'count' => $item->count, 'price' => $item->count * $item->product->price]); $itemInOrder->save(); $item->delete(); } return "Заказ сформирован"; } ```
		Выполнить импортирование методов ``` use app\models\Product; use app\models\Order; use app\models\ProductOrder; ```
Отображение заказов у пользователя		

<table>
<tr>
<td>Zeal – DataProvider
 
</td>
<td>Перейти в кодогенератор Gii(адрес вида login-m1.wsr.ru/gii) и сгенерировать CRUD для модели Order:

Grud Generator
Model Class : app\models\Order
Controller Class:
App\controllers\OrderController</td>
<td>Изменить метод index для выборки данных только авторизованного пользователя и сортировки по дате добавления в файле controllers/OrderController.php:
<pre>public function actionIndex()
{
$dataProvider = new ActiveDataProvider([
'query' => Order::find()->where(['user_id' => \Yii::$app->user->id]),</pre></td>
</tr>
<tr>
<td></td>
<td>1. Ограничить доступ к index только для авторизованного пользователя в файле controllers/OrderController.php:
2. Скопировать права доступа из CartController behaviors</td>
<td><pre>'access' => [
'class' => AccessControl::class,
'only' => ['index'],
'rules' => [
[
'actions' => ['index'],
'allow' => true,
'roles' => ['@'],
],
],
],</pre></td>
</tr>
<tr>
<td></td>
<td>1. Изменить отображение списка ордеров в представлении (файл views/order/index.php):
2. Редактируем по примеру корзины.
3. Нужно оставить количество, наименование и статус</td>
<td><pre>$this->title = 'Мои заказы';
$this->params['breadcrumbs'][] = $this->title;
?>

<h1><?= Html::encode($this->title) ?></h1></pre></td>
</tr>
</table>

	заказа, притом упорядочены должны быть от новых к старым	<?= GridView::widget(['dataProvider' => $dataProvider, 'columns' => [['class' => 'yii\grid\SerialColumn'], 'date', 'status.name', ['label'=>"Количество товаров", 'format' => 'raw', 'value' => function ($data) { return count($data->productOrders); },], ['format' => 'html', 'label' => 'Список товаров', 'value' => function ($data) { $orders = []; foreach ($data->productOrders as $item) { $orders[] = $item->product->name; } return join(' ', $orders); },], ['class' => ActionColumn::className(), 'template' => '{delete}', 'visibleButtons'=> ['delete'=>function ($model, $key, $index) { return $model->status->code === 'new'; }]],],]); ?>

	Добавить ссылку на страницу с ордерами (файл **views/layouts/main.php**):	['label' => 'Заказ', 'url' => ['/order/index'], 'visible'=>!Yii::$app->user->isGuest],
Организация панели управления сайтом администратора		
Zeal- третья ссылка Autorization Access Control Filter 'access' => ['class' => AccessControl::class, 'only' => ['login', 'logout', 'signup'], 'rules' => [['allow' => **true**, 'actions' => ['login', 'signup'], 'roles' => ['?'],], ['allow' => **true**, 'actions' => ['logout'], 'roles' => ['@'],],	Используя Gii создать в режиме Controller Generator AdminController для панели администрирования сайта/ ControllerClass: app\controller\AdminController	Проверить получение конроллера в папке controller/ AdminController и отображения по адресу admin/index Из CartControllers полностью скопировать метод behaviors в AdminController и дописать правило parent::behaviors(), ['access' => ['class' => AccessControl::class, 'only' => ['index'], 'rules' => [['actions' => ['index'], 'allow' => true, 'roles' => ['@'], 'matchCallback' => function ($rule, $action) { return \Yii::$app->user->identity->isAdmin(); }],
	Изменим наполнение страницы admin/index Используя Gii создать модель **SearchOrder** для поиска и отображение для формы поиска (файл **_search.php**): Мы **файлы создадим, но НЕ БУДЕМ ИХ ПЕРЕЗАПИСЫВАТЬ**	Добавилась новая модель и дополнительное представление в views/order/ Изменить код метода **actionIndex** в файле **controllers/AdminController.php** (код за основу можно взять из файла **controllers/OrderController.php**) public function actionIndex() { $searchModel = new OrderSearch(); $dataProvider = $searchModel->search($this->request->queryParams); return $this->render('index', ['searchModel' => $searchModel,

	Используя Gii создать в режиме GRUD Generator ModelClass: app\models\Order Search Model Class: app\models\SearchOrder ControllerClass: app\controller\OrderController **Генерировать только Models/SearchOrder И views/order/_search.php**	'dataProvider' => $dataProvider,]); } Изменим представление admin/index. Из файла генератора order/index скопировать весь файл и вставить его в admin/index, который практически пустой. Изменить название на Админ панель, убрать кнопку добавления заказа. Используя код из статьи **Data Widgets** раздел **Data column** изменить представление **index** в файле **views/admin/index.php**: use app\models\Order; use yii\helpers\Html; use yii\helpers\Url; use yii\grid\ActionColumn; use yii\grid\GridView; /** @var yii\web\View $this */ /** @var app\models\OrderSearch $searchModel */ /** @var yii\data\ActiveDataProvider $dataProvider */ $this->title = 'Административная панель'; $this->params['breadcrumbs'][] = $this->title; ?> <?= Html::encode($this->title) ?> <?= Html::a('Управление категориями', ['category/index'], ['class' => 'profile-link']) ?> <?= Html::a('Управление продуктами', ['product/index'], ['class' => 'profile-link']) ?> <?php // echo $this->render('_search', ['model' => $searchModel]); ?> <?= GridView::widget(['dataProvider' => $dataProvider, 'filterModel' => $searchModel, 'columns' => [['class' => 'yii\grid\SerialColumn'], 'date', ['label' => 'Статус',

```
            'attribute' => 'status_id',
            'filter' => ['1' => 'Новый', '2' => 'Подтвержденный', '3' => 'Отклоненный'],
            'value'=>'status.name',
            'filterInputOptions' => ['prompt' => 'Все статусы', 'class' => 'form-control', 'id' => null]
        ],
        [
            'format' => 'raw',
            'label'=> 'Количество товаров в заказе',
            'value' => function ($data) {
              return count($data->productOrders);
            },
        ],

        [
            'format' => 'raw',
            'label'=> 'ФИО заказчика',
            'value' => function ($data) {
              return $data->user->name.' '.$data->user->surname.' '.$data->user->patronymic;
            },
        ],
        [
            'class' => ActionColumn::className(),
            'template' => '{update}',
            'visibleButtons' => [
              'update' => function ($model, $key, $index) {
                    return $model->status->code == 'new';
                }
            ]
        ],
    ],
]); ?>

</div>
```

Изменить редирект при авторизации (файл **controllers/SiteController.php**)

```
public function actionLogin()
  {
    if (!Yii::$app->user->isGuest) {
      return $this->goHome();
```

<table>
<tr>
<td></td>
<td></td>
<td>

```
    }

    $model = new LoginForm();
    if ($model->load(Yii::$app->request->post()) && $model->login()) {
        if (Yii::$app->user->identity->isAdmin()) {
            return $this->redirect(['/admin']);
        }
        return $this->goBack();
    }

    $model->password = '';
    return $this->render('login', [
        'model' => $model,
    ]);
}
```

Добавить ссылку на админ-панель (файл **views/layouts/main.php**)

```
['label' => 'Админка', 'url' => ['/admin'], 'visible' => !Yii::$app->user->isGuest &&
Yii::$app->user->identity->isAdmin()],
```

</td>
</tr>
<tr>
<td colspan="3">Управление категориями</td>
</tr>
<tr>
<td>

Zeal- третья ссылка Autorization

Access Control Filter

```
'access' => [
        'class' =>
AccessControl::class,
        'only' => ['login', 'logout',
'signup'],
        'rules' => [
            [
                'allow' => true,
                'actions' => ['login',
'signup'],
                'roles' => ['?'],
            ],
            [
                'allow' => true,
                'actions' => ['logout'],
```

</td>
<td>

Используя Gii создать в режиме GRUD Generator

ModelClass: app\models\Category

ControllerClass: app\controller\CategoryController

Проверить переход category/index

</td>
<td>

Ограничить доступ к методам контроллера только для администратора (файл **controllers/CategoryController.php**): так как все действия должны быть доступны только администраторы

```
'access' => [
        'class' => AccessControl::class,
        'only' => ['index'],
        'rules' => [
            [
                'actions' => ['index','view', 'create', 'delete', 'update'],
                'allow' => true,
                'roles' => ['@'],
                'matchCallback' => function ($rule, $action) {
                    return \Yii::$app->user->identity->isAdmin();
                }
            ],
        ],
    ],
```

</td>
</tr>
</table>

<table>
<tr>
<td>

```
                'roles' => ['@'],
            ],
```

Zeal – HTML Helper ссылки

```
<?= Html::a('Profile', ['user/view', 'id' => $id], ['class' => 'profile-link']) ?>
```

</td>
<td></td>
<td>
В админпанель добавить ссылку на управление категориями (файл views/admin/index.php):

```
<?= Html::a('Управление категориями', ['category/index'], ['class' => 'profile-link']) ?>
```

</td>
</tr>
<tr>
<td colspan="3">Управление товаром</td>
</tr>
<tr>
<td></td>
<td>
Используя Gii создать в режиме GRUD Generator
ModelClass: app\models\Product
ControllerClass: app\controller\ProductController

Проверить переход product/index
</td>
<td>
Проверить отображение всех продуктов. Добавить ссылку управления продуктами по аналогии Управления категориями.

```
<?= Html::a('Управление продуктами', ['product/index'], ['class' => 'profile-link']) ?>
```

Для всех сгенерированных методов контроллера указать доступ только для администратора (файл controllers/ProductController.php)

```
            'access' => [
                'class' => AccessControl::class,
                'only' => ['index'],
                'rules' => [
                    [
                        'actions' => ['index', 'view', 'create', 'delete', 'update'],
                        'allow' => true,
                        'roles' => ['@'],
                        'matchCallback' => function ($rule, $action) {
                            return \Yii::$app->user->identity->isAdmin();
                        }
                    ],
                ],
            ],
```

Копируем из HTML Helper DropDownList для создания списка категорий.Изменить отображение формы для добавления продукта (добавить список категорий, указать возможность загрузки изображения и т.д) в файле views/product/_form.php:

```
<div class="product-form">

    <?php $form = ActiveForm::begin(['options' => ['enctype' => 'multipart/form-data']]); ?>

    <?= $form->field($model, 'name')->textInput(['maxlength' => true]) ?>
```

</td>
</tr>
</table>

		<?= $form->field($model, 'imageFile')->fileInput() ?> <?= $form->field($model, 'count')->textInput() ?> <?= $form->field($model, 'price')->textInput(['maxlength' => true]) ?> <?= $form->field($model, 'year')->textInput() ?> <?= $form->field($model, 'model')->textInput(['maxlength' => true]) ?> <?= $form->field($model, 'country')->textInput(['maxlength' => true]) ?> <?= $form->field($model, 'category_id')->dropDownList(ArrayHelper::map(Category::find()->asArray()->all(), 'id', 'name')) ?> <?= Html::submitButton('Save', ['class' => 'btn btn-success']) ?> <?php ActiveForm::end(); ?> Импортировать в начало файла use app\models\Category; use yii\helpers\ArrayHelper;
Zeal-UploaderFiles - Creating Models use **yii\web\UploadedFile**; **return** [[['imageFile'], 'file', 'skipOnEmpty' => **false**, 'extensions' => 'png, jpg'],];	Реализация загрузки фотографии. Для реализации загрузки фото товара можно использовать код из статьи Uploading Files. Добавить в модель **Product** поле **imageFile** и логику загрузки в файл модели (файл **models/Product.php**):	Добавить новое свойство в модель Product class Product extends \yii\db\ActiveRecord { public $imageFile; Настроить валидацию расширений изображений [['imageFile'], 'file', 'skipOnEmpty' => false, 'extensions' => 'png, jpg'], Добавить папку Uploads в папку web Создать директорию **web/uploads** Добавить функцию загрузки изображения $fileName будет содержать путь к файлу в самый конец

		public function upload() { if ($this->validate()) { $fileName='uploads/'.$this->imageFile->baseName.'.'.$this->imageFile->extension; $this->imageFile->saveAs('uploads/' . $this->imageFile->baseName . '.' . $this->imageFile->extension); $this->imageFile->saveAs($fileName); $this->file='/'.$fileName; return true; } else { return false; } }
	Изменить действие контроллера для обработки загрузки (**controllers/ProductController.php**) В методе ActionCreate нужно дописать файл загрузки: Если пришли данные, то мы их загрухаем в модель - если модель загружена, сохраним imageFile и сохраняем модель - если не получилось сохранить модель, тогда редирект на саму страницу	public function actionCreate() { $model = new Product(); if ($this->request->isPost) { if ($model->load($this->request->post())) { $model->imageFile = UploadedFile::getInstance($model, 'imageFile'); if ($model->upload() && $model->save(false)) { return $this->redirect(['index']); } } } else { $model->loadDefaultValues(); } return $this->render('create', ['model' => $model,]); } Импортировать use app\models\Product; use yii\data\ActiveDataProvider; use yii\web\Controller;

		use yii\web\NotFoundHttpException; use yii\filters\VerbFilter; use yii\filters\AccessControl; use yii\web\UploadedFile;
		Аналогично созданию необходимо поправить и редактирование товара (**controllers/ProductController.php**): public function actionUpdate($id) { $model = $this->findModel($id); if ($this->request->isPost) { if ($model->load($this->request->post())) { $model->imageFile = UploadedFile::getInstance($model, 'imageFile'); if ($model->upload() && $model->save(false)) { return $this->redirect(['index']); } } } return $this->render('update', ['model' => $model,]); }
Управление заказом		
	Добавить правила доступа администратора к редактированию заказов (можно изменять статус только нового заказа) в файле **controllers/AdminController.php**	['actions' => ['update'], 'allow' => true, 'roles' => ['@'], 'matchCallback' => function ($rule, $action) { $id_order=\Yii::$app->request->get('id'); $order=Order::findOne($id_order); return \Yii::$app->user->identity->isAdmin() && $order->status-> code ==='new'; }], Проверить из-под администратора, верно ли, что администратор может редактировать только новые заказы.

		Добавить файлы представления **update.php** и **_form.php** для редактирования заказа в директорию **views/admin, скопировав их из директории views/order**/ Отредактировать файлы. Update.php `<?php` `use yii\helpers\Html;` `/** @var yii\web\View $this */` `/** @var app\models\Order $model */` `$this->title = 'Изменение заказа: ' . $model->id;` `?>` `<div class="order-update">` `<h1><?= Html::encode($this->title) ?></h1>` `<?= $this->render('_form', [` `'model' => $model,` `]) ?>` `</div>` _form.php `<?php` `use yii\helpers\ArrayHelper;` `use yii\helpers\Html;` `use yii\widgets\ActiveForm;` `/** @var yii\web\View $this */` `/** @var app\models\Order $model */` `/** @var yii\widgets\ActiveForm $form */` `?>` `<div class="order-form">`

		<?php $form = ActiveForm::begin(); ?> <?= $form->field($model, 'status_id') ->dropDownList(ArrayHelper::map(\app\models\Status::find()->asArray()->all(),'id','name')) ?> <?= $form->field($model, 'rejection_reason')->textarea(['rows' => 6]) ?> <?= Html::submitButton('Сохранить', ['class' => 'btn btn-success']) ?> <?php ActiveForm::end(); ?>
	Добавить метод **update** для редактирования заказа в класс **controllers/AdminController.php: Эти функции можно взять из метода OrderControler** **Но нужно изменить редирект на основную страницу**	public function actionUpdate($id) { $model = Order::findOne(['id' => $id]); if ($this->request->isPost && $model->load($this->request->post()) && $model->save()) { return $this->redirect(['index']); } return $this->render('update', ['model' => $model,]); }
Zeal – Validation – **Conditional Validation** ['state', 'required', 'when' => **function** ($model) { **return** $model->country == 'USA'; }, 'whenClient' => "function (attribute, value) {	Добавить валидацию (обязательность указания комментария при отмене заказа) в файле **models/Order.php**:	['rejection_reason', 'required', 'when' => function ($model) { return $model->status->code == 'rejected'; }, 'whenClient' => "function (attribute, value) { return $('#order-status_id').val() == 'Отклоненный'; }"],

return $('#country').val() == 'USA'; }"]		
Отображение товаров в каталоге		
	Изменить index метод в контроллере **controllers/SiteController.php** Импортировать Active DataProvider	public function actionIndex($id_category = null) { $dataProvider = new ActiveDataProvider(['query' => $id_category? Product::find()->where(['>', 'count', 0])->andWhere(['category_id' => $id_category]) : Product::find()->where(['>', 'count', 0]), 'pagination' => ['pageSize' => 3,], 'sort' => ['defaultOrder' => ['date' => SORT_DESC,]],]); return $this->render('index',['dataProvider'=> $dataProvider]); }
	Изменить отображение в представлении (файл **views/site/index.php**): Представление скопировать cart/index все скопировать и вставить, изменив название Каталог, изменив product.name на name. Надо добавить цену., страну, год производства Для вставки рисунка можно воспользоваться HtmlHelper статья Images	<?php use app\models\Cart; use app\models\Category; use yii\helpers\ArrayHelper; use yii\helpers\Html; use yii\helpers\Url; use yii\grid\ActionColumn; use yii\grid\GridView; use yii\widgets\Pjax; /** @var yii\web\View $this */ /** @var yii\data\ActiveDataProvider $dataProvider */ $this->title = 'Каталог';

```
$this->params['breadcrumbs'][] = $this->title;
$this->registerJsFile(
   '@web/js/main.js',
   ['depends' => [\yii\web\JqueryAsset::class]]
);

?>
<div class="cart-index">

   <h1><?= Html::encode($this->title) ?></h1>
   <div class="info alert alert-primary"></div>
   <?php Pjax::begin(['id' => 'cart']) ?>
   <?= GridView::widget([
      'dataProvider' => $dataProvider,
      'columns' => [
         ['class' => 'yii\grid\SerialColumn'],
         'name',
         'price',
         'country',
         'year',
         'count',
         [
            'format' => 'html',
            'value' => function ($data){
               return Html::img('@web'.$data->file, ['alt' => 'Товар не имеет изображения', 'width'=>
200]);
            },
         ],
         [
            'format' => 'raw',
            'value' => function ($data) {
                  return "<button onclick='addCart($data->id)' class='btn btn-success'>+</button>";

               },
         ],
      ],
   ]); ?>
   <?php Pjax::end() ?>
```

	Можно сразу русифицировать модель продукта	public function attributeLabels() { return ['id' => 'ID', 'date' => 'Дата', 'name' => 'Наименоване продукта', 'file' => 'Изображение товара', 'count' => 'Количество товаров', 'price' => 'Цена', 'year' => 'Год изготовления', 'model' => 'Модел', 'country' => 'Страна', 'category_id' => 'Category ID',]; }
Фильтрация списка товаров		
	Из product/_form можно взять список категорий В site/index до pjax добавляем В представление **views/site/index.php** добавить выпадающий список категорий товаров (пример кода можно взять в Zeal в статьях **Html helpers** -> **Input Fields** и **Creating Forms -> Creating Lists**	<?php $category = ArrayHelper::map(Category::find()->asArray()->all(), 'id', 'name'); echo Html::dropDownList('list', null, $category, ['prompt' => 'Выберите категорию', 'onchange' => 'getProduct(this.options[this.selectedIndex].value)', 'class'=>'form-select']) ?>
	Добавить js функцию **getProduct** в файл **web/js/main.js:**	function getProduct(id_category) { $.pjax.reload({ url: `/?id_category=${id_category}`, container: '#cart' });

		} }
	Доработать метод index для получения товаров определенной категории (файл **controllers/SiteController.php**)	public function actionIndex($id_category = null) { $dataProvider = new ActiveDataProvider(['query' => $id_category? Product::find()->where(['>', 'count', 0])->andWhere(['category_id' => $id_category]) : Product::find()->where(['>', 'count', 0]), 'pagination' => ['pageSize' => 3,], 'sort' => ['defaultOrder' => ['date' => SORT_DESC,]],]);
Отображение карточки товара		
	Добавить метод **view из ProductController** в файле **controllers/SiteController.php и скопировать метод из ProductController** для получения данных товара:	public function actionView($id) { return $this->render('view', ['model' => Product::findOne($id),]); }
	Добавить файл **views/site/view.php из product/view**:и и вставить туда отображение карточки товара (пример кода можно взять из шаблона который генерирует gii)	$this->registerJsFile('@web/js/main.js', ['depends' => [\yii\web\JqueryAsset::class]]); ?> <?= Html::encode($this->title) ?> <?php Pjax::begin(['id' => 'cart']) ?> <?= DetailView::widget([

		'model' => $model, 'attributes' => ['name', ['label'=> 'Изображение товара', 'format' => 'html', 'value' => Html::img('@web' . $model->file, ['alt' => 'Товар не имеет изображения', 'width' => 200]),], 'count', 'price', 'year', 'model', 'country', 'category.name', ['attribute' => 'Добавить в корзину', 'format' => 'raw', 'value' => function ($model) { return "<button onclick='addCart($model->id)' class='btn btn-success'> + </button>"; }, 'visible' => !\Yii::$app->user->isGuest,],],]) ?> <?php Pjax::end() ?>
Реализация дополнительных страниц		
	Переименовать ссылки в главном меню (файл **views/layouts/main.php**)	echo Nav::widget(['options' => ['class' => 'navbar-nav'], 'items' => [['label' => 'Каталог', 'url' => ['/site/index']], ['label' => 'О нас', 'url' => ['/site/about']], ['label' => 'Где нас найти', 'url' => ['/site/contact']], ['label' => 'Регистрация', 'url' => ['/user/create'],'visible'=> Yii::$app->user->isGuest], ['label' => 'Заказ', 'url' => ['/order/index'], 'visible'=>!Yii::$app->user->isGuest],

		['label' => 'Корзина', 'url' => ['/cart/index'], 'visible' => !Yii::$app->user->isGuest], ['label' => 'Админка', 'url' => ['/admin'], 'visible' => !Yii::$app->user->isGuest&&Yii::$app->user->identity->isAdmin()], Yii::$app->user->isGuest ? ['label' => 'Вход', 'url' => ['/site/login']] : '<li class="nav-item">' . Html::beginForm(['/site/logout']) . Html::submitButton('Выход (' . Yii::$app->user->identity->username . ')', ['class' => 'nav-link btn btn-link logout']) . Html::endForm() . ' ']]);
	Проверить медиафайлы, если их нет, то их можно придумать, взяв из доступных ресурсов. Создавать структуру можжно как из библиотеки boostrap, так из Zeal-HTML Helper. Создать каталог img и поместить туда файл с картой:	
	На странице “Где нас найти” (**views/site/contact.php**) сверстать данные согласно заданию (карта и контактные данные (адрес, номер телефона, email)):	$this->title = 'Contact'; $this->params['breadcrumbs'][] = $this->title; ?> <h1>Где нас найти?</h1> <img src="../web/img/map.png" class="card-img-top" alt="Карта"> <h5 class="card-title">Наши координаты:</h5>

		Some quick example text to build on the card title and make up the bulk of the card's content. <ul class="list-group list-group-flush"> <li class="list-group-item">Email:ty@mail.ru <li class="list-group-item">Телефон:8 (919) 555-44-33 <li class="list-group-item">Адрес: смотрните по карте
	На странице “О нас” (**views/site/about.php**) сверстать данные согласно заданию (логотип компании и написан девиз компании): 1. Используя верстку из статьи **Bootstrap 5** -> **Carousel** раздел **Dark variant** сверстать карусель последних 5 продуктов (файл **views/site/about.php**): 2. Скопируем выборку продуктов из CartController $product = Product::find()->where(['id' => $id_product])->andWhere(['>', 'count', 0])->one(); Изменим запрос на пять отсортированных по дате (можно найти в DataProvider Active record)	this->title = 'О нас'; $this->params['breadcrumbs'][] = $this->title; $product= Product::find()->where(['>', 'count', 0])->orderBy(['date' => SORT_DESC,])->limit(5)->all(); ?> <h1><?= Html::encode($this->title) ?></h1> <img src="../web/img/dog_logo.jpg" class="rounded" alt="Логотип компании" width="200px"> Все лучшее - у нас! <?php foreach($product as $item) { ?> <img src="<?= $item->file ?>" class="d-block w-100" alt="<?= $item->name ?>"> <h1 align='center'><?= $item->name ?></h1>

		<?php } ?> <button class="carousel-control-prev" type="button" data-bs-target="#carouselExampleControls" data-bs-slide="prev"> Previous </button> <button class="carousel-control-next" type="button" data-bs-target="#carouselExampleControls" data-bs-slide="next"> Next </button> <code><?= __FILE__ ?></code>
	В папке web создать папку , а в ней папку css В этой папке создать файл main.css	@font-face { font-family: "new-font"; src: url("../web/fonts/helvetica_bold.otf") ; } .container { font-family: new-font, serif; }
	В файле views/layout/main подключить стиль	$this->registerCssFile('@web/css/main.css');
	Закомментировать функции в файле main.js	//Добавление товара в корзину function addCart(id_product) { $.ajax({ method: 'GET', url: `/cart/create?id_product=${id_product}`, }).done(function (message) {

```
        $.pjax.reload({
            container: '#cart'
        });
        $('.info').html(message);
        setTimeout(() => {
            $('.info').text('');
        }, 1000);
    });
}
//Удаление товара из корзины
function removeCart(id_product) {
    $.ajax({
        method: 'POST',
        url: `/cart/delete?id_product=${id_product}`,

    }).done(function (message) {
        $.pjax.reload({
            container: '#cart'
        });
        $('.info').html(message);
        setTimeout(() => {
            $('.info').text('');
        }, 1000);
    });
}
//Подтверждение заказа
function byOrder() {
    let password = $('#inputPassword5').val();
    console.log(password);
    if (!password) {
        $('.info').html("Введите пароль");
        return
    }
    $.ajax({
        method: "GET",
        url: `/cart/by-order?password=${password}`,
    })
        .done(function (message) {
```

```
            $.pjax.reload({
                container: '#cart'
            });
            $('.info').html(message);
        })
}
//Фитрация каталога по категориям
function getProduct(id_category) {
    $.pjax.reload({
        url: `/?id_category=${id_category}`,
        container: '#cart'
    });
}
```