**Введение**

Специальность 09.02.07 «Информационные системы и программирование» квалификации «Разработчик веб и мультимедиа приложений» является востребованной в реалиях современного мира. Это связано с реализацией новых технологий и необходимостью грамотных специалистов, способных решать нестандартные профессиональные задачи.

Целью данного отчёта будет практико-теоретическое воспроизведение содержания учебной практики. Вся оценочная документация, отраженная в дневнике по практике и характеристике будет предложена руководителю практики от ОУ в виде отдельных документов.

В качестве средств документирования практико-теоретических результатов УП будут использоваться скриншоты и фотографии с пояснениями различных этапов выполнения работ. Поскольку для выполнения заданий необходимо уделять внимание теоретической составляющей – формированию основных понятий и представлений, связанных с предметной областью, в отчёт будет включена информация из дополнительных источников. Результаты работы будут ежедневно фиксироваться в дневнике по практике и сопровождаться оценкой руководителя практики, в отчёте эти данные представлены не будут.

**I часть содержания учебной практики**

Передо мной была поставлена задача: спроектировать и создать интернет-ресурс молодой компании «Copy Star», которая занимается продажей копировального оборудования.

Мне предстояло продемонстрировать свои знания в разработке веб-приложения с использованием технологий *HTML, CSS, Bootstrap* и *PHP (Yii)*. Вдобавок мне было необходимо позаботиться о защите проекта от взлома и несанкционированного доступа к административным функциям приложения.

**Практическая работа №1. Проектирование физической модели базы данных.**

На основе выданного задания для дэмо-экзамена составить EER диаграмму и физическую модель базы данных. Для этого я воспользовался встроенным в локальный веб-сервер *Open Server Panel* инструментом – *PhpMyAdmin.*



Рис. 1 – Таблица «Товары» (Product).

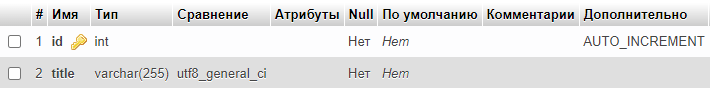


Рис. 2 – Таблица «Категории» (category).



Рис. 3 – Таблица «Заказы» (order).



Рис. 4 – Таблица «Состав заказа» (order\_structure).

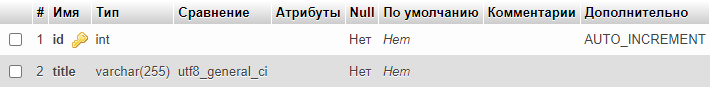


Рис. 5 – Таблица «Роль пользователя» (role).



Рис. 6 – Таблица «Пользователи» (user).

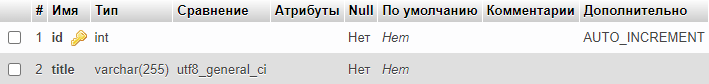


Рис. 7 – Таблица «Статус заказа» (status).

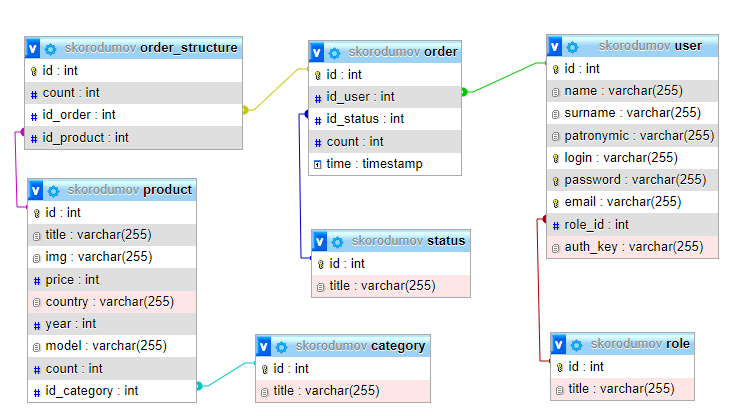


Рис. 8 – Таблица EER-диаграмма базы данных.

Следующим шагом заполнил таблицы со статусами заказа и ролью пользователя:

|  |  |
| --- | --- |
|  |  |
| Рис. 9 – user | Рис. 10 – status |

**Практическая работа №2. Установка и первичная настройка Yii2.**

После успешного создания базы данных, я приступил к созданию basic-приложения фреймворка Yii2. Для этого в директории с доменами *OpenServer’a* создал отдельную папку для своего проекта и поместил в неё файлы basic-приложения, после чего произвёл настройку конфигурации в файле config/web.php задав произвольное значение cookie-токену, пустое *baseUrl’у*, перенеся приложение на русский язык ('*language*' => *'ru-RU'*), включив *UrlManager* и произведя настройку файлов *.htaccess*, для того чтобы пользователь получал доступ к порталу напрямую, без папки *web*.



Рис. 11 – Содержание .htaccess в корневой директории.

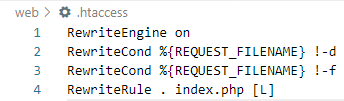


Рис. 12 – Содержание .htaccess в папке web.

|  |  |
| --- | --- |
|  |  |
| Рис. 13-14 - Настройка параметров в config/web.php | |

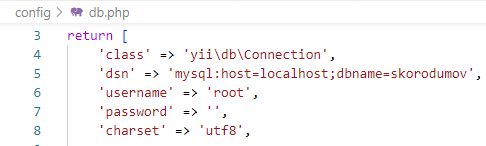


Рис. 15 - Настройка файла конфигурации для подключения к базе

данных (*config/db.php*).

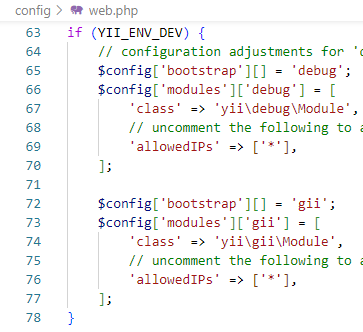


Рис. 15 - Включение панели отладки и доступ к генератору кода gii, указав в

параметре «*allowedIPs*» это [‘\*’].

**Практическая работа №3. Формирование модели User и создание статического метода возвращающего id для роли.**

**Задача 1.** Доработать в таблицу User для последующей работы с интерфейсом IdentityInterface. Реализация IdentityInterface.

**Задача 2.** Сформировать модели используя генератор моделей *gii* из состава фреймворка Yii2 на основании базы данных, разработанной в практической работе №1.

Для этого переименовываю *user* в *users*, захожу в *gii* по адресу *localhost/gii/model*, в поле *table name* ставим \*.

**Задача 3.** Разработать и добавить в модель user следующие методы: *validatePassword* (метод проверки пароля пользователя с тем, что хранится в базе данных), *validateAuthKey* (метод проверки ключа аутентификации пришедшего от пользователя с тем, что есть у пользователя в базе данных), *findByUsername* (поиск в базе данных статическим методом *findOne* и возврат экземпляра модели User по переданному в метод ключевому полю определяющего пользователя).

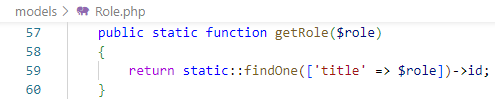


Рис. 16 – Упомянутые методы.

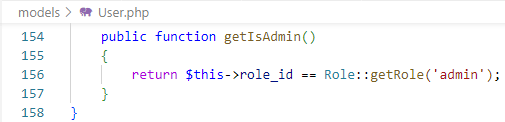


Рис. 17 – Заполнение полей, содержащих пароль (используя *Yii::$app->security->generatePasswordHash();* ) и ключ аутентификации (используя *Yii::$app->security->security->generateRandomString();* )

**Задача 4.** Создание статического метода в модели *Role*, возвращающего значение ключевого поля для искомой роли. (Рис. 17)



**Задача 5.** Создание метода *getIsAdmin* в модели *User*, возвращающий булевое значение результата проверки роли текущего пользователя с ролью «admin» (используя метод getRole созданный ранее). (Рис. 18)



**Практическая работа №4. Создание процессов регистрации, аутентификации и «разлогирования» пользователей.**

**Задача 1.** Доработать меню сайта путём переименования меню «*Contact*» в «*Регистрация*» (данный пункт меню должен быть доступен только для незарегистрированного пользователя).



Рис. 19 – Изменения в 42-43 строках кода.

**Задача 2-3.** На основе исходных данных необходимо создать модель регистрации пользователя с учетом всех правил валидации. Создать на основе страницы *contact.php* форму регистрации пользователя.

В директории *models* создаю *RegisterForm.php* (копия *ContactForm.php*)

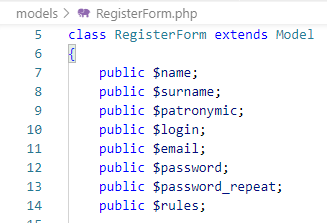
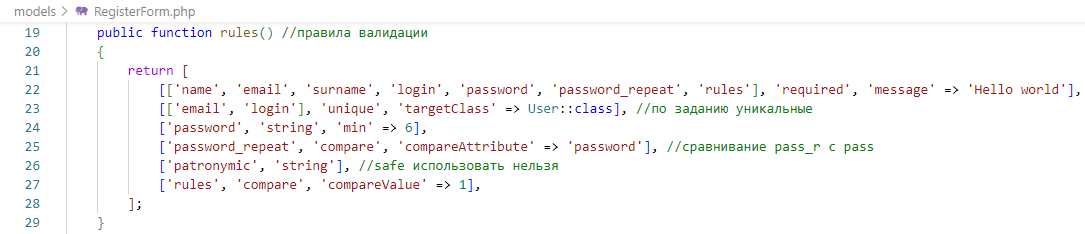
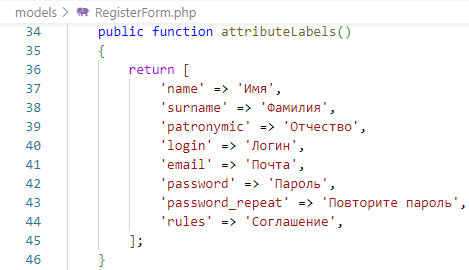
 

Рис. 20-22 – Модель регистрации пользователя.

Делаю копию *contact.php*, удаляю лишние строки кода в файле, переименовываю его в *register.php*, меняю на новое название файла путь в файле *main.php*.(Рис.23)





Рис. 24 – Форма регистрации пользователя.

**Задача 4:** В контроллере SiteController, создать метод регистрации пользователя на полученные данные от формы регистрации и провести их валидацию.



Рис. 25 – Форма регистрации пользователя.



В модели регистрации создаю метод регистрации пользователя. Метод создания пользователя реализовываю через загрузку в модель пользователя данных от модели регистрации.



Рис. 26 – Метод создания пользователя в модели регистрации.

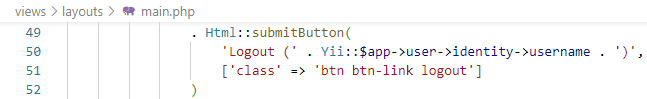
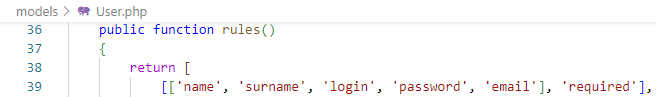
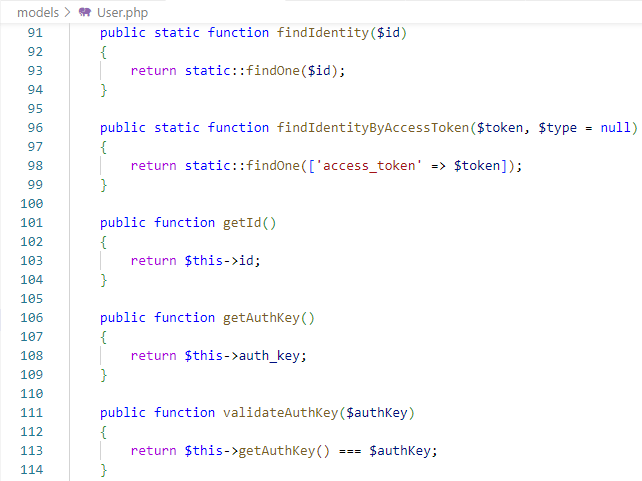


Рис. 27 – Кнопка выхода у авторизованного пользователя.

**Задача 5.** Переработать модель входа (аутентификации) пользователя на сайт. В модели *user* подключим следующее:



Ещё добавил 4 метода:



Так же, во избежание ошибки, я удалил в файле index.php представлений admin и category-admin следующий фрагмент кода:

'urlCreator' => function ($action, Category $model, $key, $index, $column) {

                    return Url::toRoute([$action, 'id' => $model->id]);

                 }

**Практическая работа №5. Создание валидации формы без перезагрузки страницы.**

Необходимо организовать валидацию формы регистрации пользователя без перезагрузки страницы с использованием Ajax.

Первым действием я включаю AJAX-валидацию для тех полей у формы, где это необходимо, задав значение у формы *enableAjaxValidation – true;* (Рис. 28)





Рис. 28 – Подключение использования класса ActiveForm в контроллере.

Далее я создаю объект класса формы регистрации и произвожу приём данных от формы. Указываю формат принимаемых данных: *FORMAT\_JSON*. Возвращаю результат валидации формы вызвав метод *ActiveForm::validate*;



Рис. 29 – Изменениям в экшене Register.

**Практическая работа №6. Cоздание личного кабинета администратора.**

В начале я создаю кнопку перехода в панель администратора в главном меню сайта:



**Задача 1.** С помощью генератора кода gii произвести генерацию CRUD контроллера: CRUD построить на основе модели с заказами, с использованием модели для поиска (*AdminSearch*). Название контроллера установить согласно заданию (если в задании определено по какому адресу должен быть личный кабинет пользователя), либо *AdminController*.

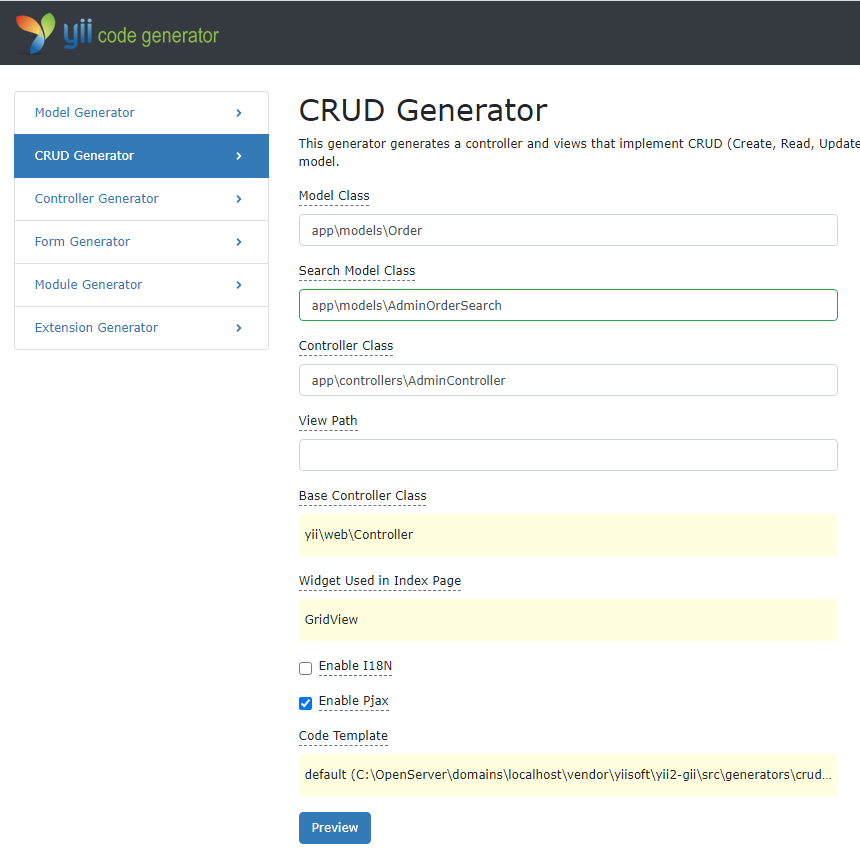


Рис. 30 – Генерация кода.

**Задача 2.** С помощью генератора кода gii произвести генерацию CRUD контроллера (*CategorAdminyController*) для управления категориями, с использованием модели для поиска (*CategoryAdminSearch*).

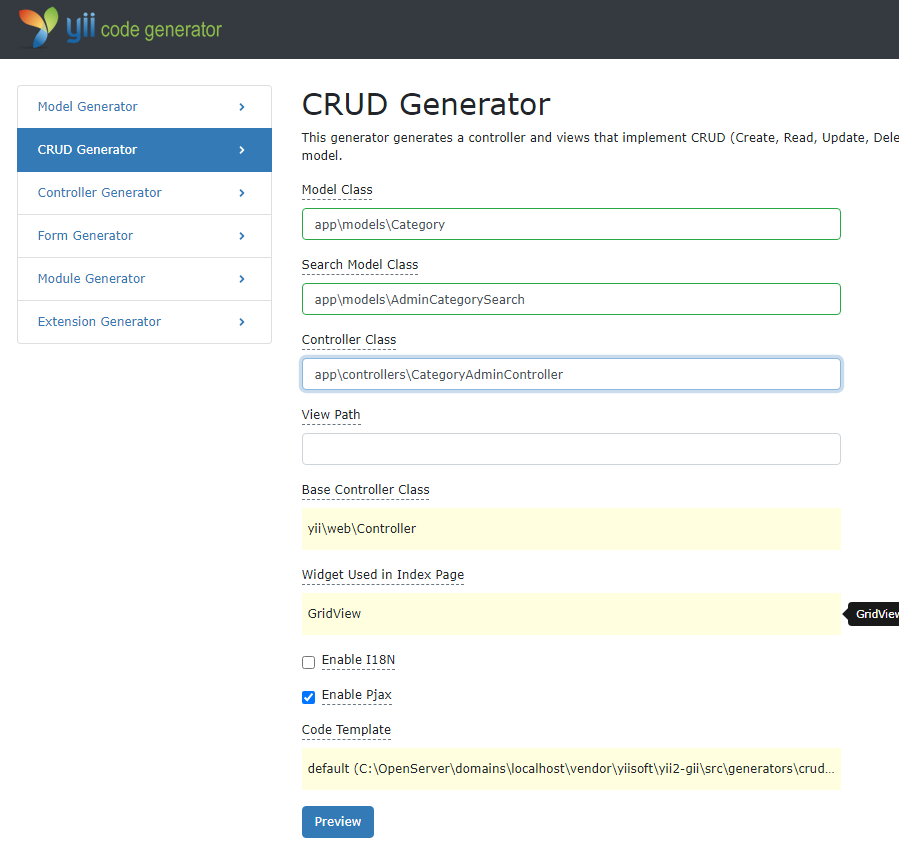


Рис. 31 – Генерация кода.

**Задача 3.** Реализация ограничения доступа к контроллеру администратора через фильтр доступа *AccessControl*;

Используя поведение *behaviors* и фильтр доступа AccessControl, я настроил доступ к контроллеру панели администратора только зарегистрированного пользователя - администратора. Доступ определил через вызов *matchCallback* для правил и *denyCallback.*

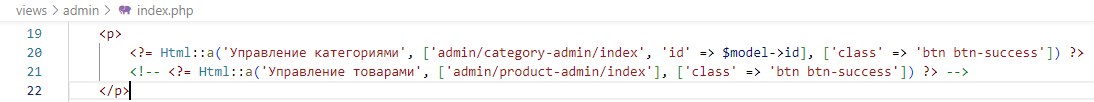
**



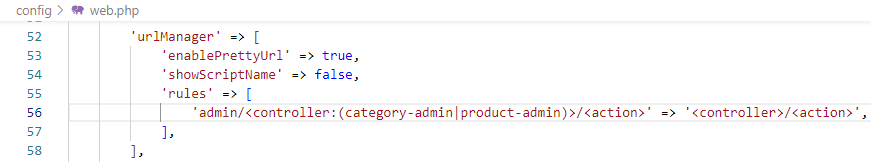
Рис. 32 – Доработка метода behaviors().

Аналогичные действия я произвел в файле CategoryAdminController.php

**Задача 5.** На главной странице панели управления администратора создать две кнопки: «Управление категориями» с url - *'admin/category-admin/index*'; и «Управление товарами» с url - ' *admin/product-admin/index'*;



**Задача 6.** В файле конфигурации приложения (*config/web.php*), в секции «*urlManager*», параметра «*rules*» создать правило для доступа в контроллеры «*админки*»:



**Задача 7.** Со всех страниц панели администратора удалить *breadcrumbs* («хлебные крошки»). Выполнил.

**Практическая работа №9. Настройка информационных страниц.**

**Задача 1.** Разработать страницу «Где нас найти?». В папке *views/site* создать файл-представление, в котором разместить разделы и контент (карта и контактные данные).

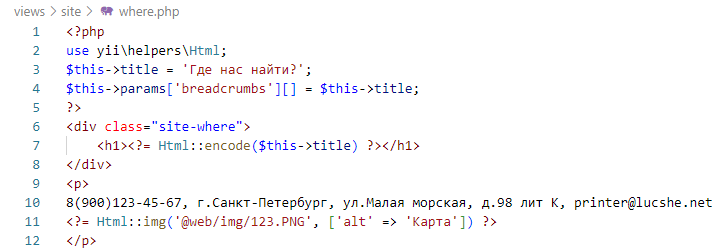


Рис. 33 – Представление where (страница «Где нас найти?»).

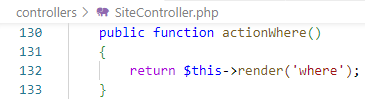
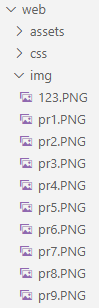
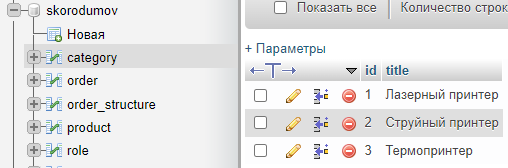
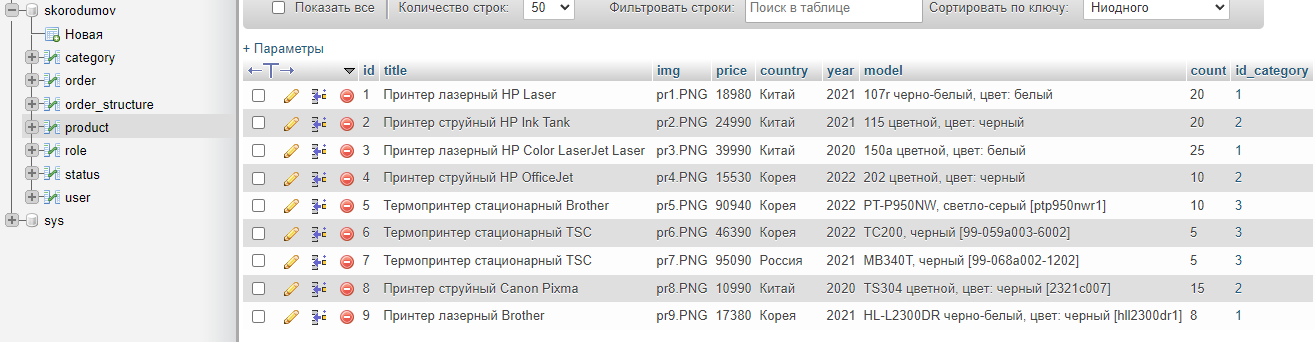


Рис. 34 – Action страницы «Где нас найти?».



Рис. 35 – Добавленная строчка кода в main.php

**Задача 2**. Разработать страницу «О нас». Для реализации контента согласно заданию мне необходимо создать папку *web/img* и загрузить в неё фотокарточки. Далее я заполнил таблицу товаров и категорий в ранее созданной базе данных:

Следующим шагом я перехожу к разработке кода для слайдера с товарами магазина.



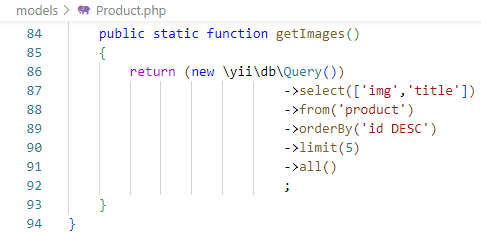


Рис. 36 – Создание SQL-запроса *SELECT img FROM `product` ORDER BY id DESC LIMIT 5*

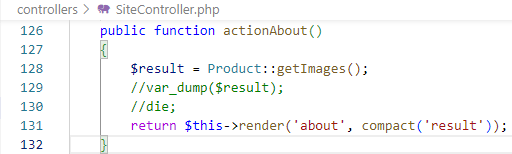


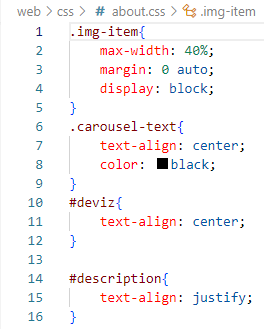
Рис. 37 – Создание переменной, содержащей список последних 5 товаров, вызвав ранее созданный метод *getImages*.



Рис. 38 – Добавленный на страницу виджет bootstrap «Карусель»

Для получения эстетического удовольствия при посещении данной страницы я добавил файл с css-стилями:





**Практическая работа №10. Создание каталога товаров.**

Для создания страниц каталога с товарами необходимо выполнить создание и настройку контроллера.

**Задача 1.** С помощью генератора кода gii создать CRUD контроллер «*каталога товаров*», на основе модели «*товар*», используя поисковую модель. При генерации контроллера, я использовал *widget listView*, а также отметить пункт для включения *Pjax*.

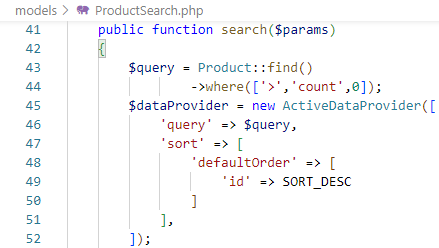
В поле Model Class: app\models\Product

В поле Search Model Class: app\models\ProductSearch

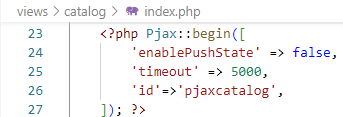
В поле Controller Class: app\controllers\CatalogController

После я внёс изменение в файл конфигурации *config/web.php*, создав параметр '*defaultRoute*' => *'catalog/index'*. Удалил ненужные методы в контроллере каталога (create, update, delete).

В модели поиска доработал SQL-запрос (вывод только существующих товаров):



Установил параметры и задал *id* для контейнера *Pjax*:



Так же я удалил кнопку поиска и временно закомментировал форму поиска. Настроил ListView в catalog/index.php добавив следующие параметры:

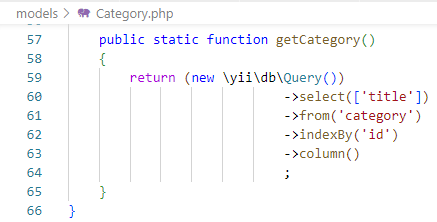




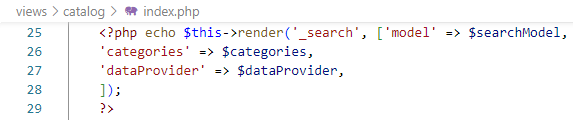
Рис. 39 – Установка id для блока, оборачивающего Pjax.

**Задача 2.** Вызвать статический метод возвращающий индексированный массив с категориями:

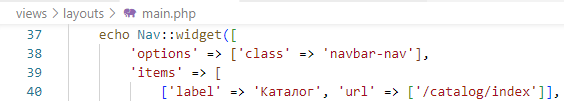








**Задача 3.** В файле виде - шаблоне приложения, добавить пункт «каталог»:



**Задача 4.** Используя библиотеку Bootstrap 4 – настроить внешний вид карточки товара, добавить для кнопку возвращения в каталог:





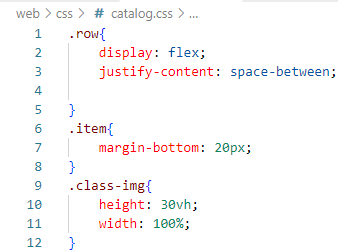
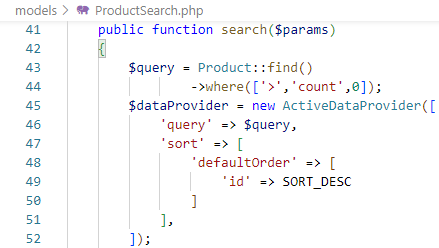


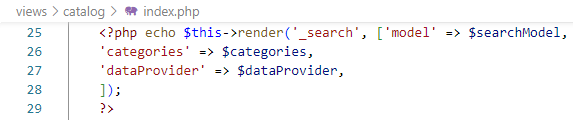
Рис. 40 – Подключение CSS-стиля для эстетического вида карточки товара.

**Практическая работа №11. Создание сортировки и фильтрации в каталоге товаров.**

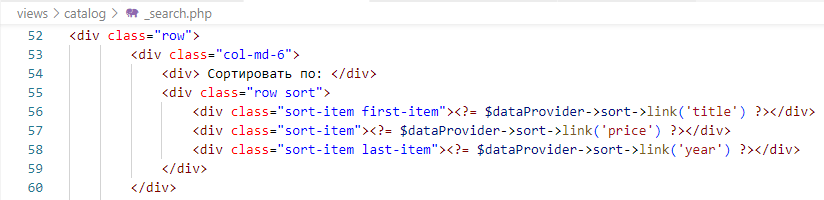
**Задача 1.** В файле поисковой модели каталога доработал запрос к базе данных, настроил сортировку товаров по умолчанию, для этого, у *ActiveDataProvider* настроил параметр *sort*, указав в нем параметр *defaultOrder* с названием поля и типом сортировки по умолчанию:



**Задача 2.** В файле вида каталога *(\_search.php*) раскомментировал показ формы поиска и дополнительно передал в форму поиска переменную, содержащую *DataProvider*, используемый в *ListView*:



**Задача 3.** Создание сортировки. Сначала в файле с формой я создаю два блока, расположенные в одной строке. В левом блоке будет находится сортировка, в правом фильтрация.





**Задача 4.** В контроллере каталога, в методе *index*, сформировать переменную, содержащую список категорий, для это необходимо вызвать метод из модели «категории», возвращающий проиндексированный массив со списком категорий. Переменную с массивом передать в представление, а далее передать ее в форму поиска.

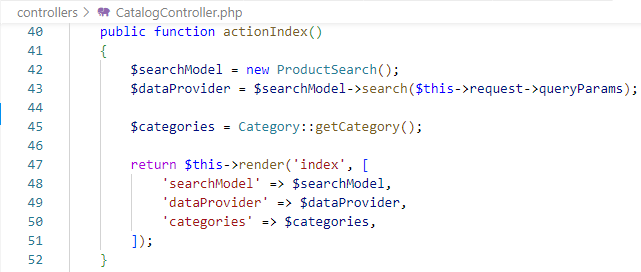


Рис. 41 – Вызов метода getCategory.

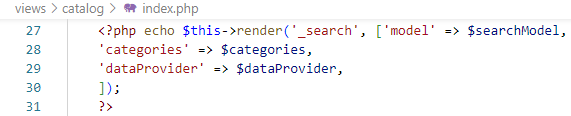


Рис. 42 – Вызов переменной, содержащей уже сформированный список категорий.