ОБЛАСТНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

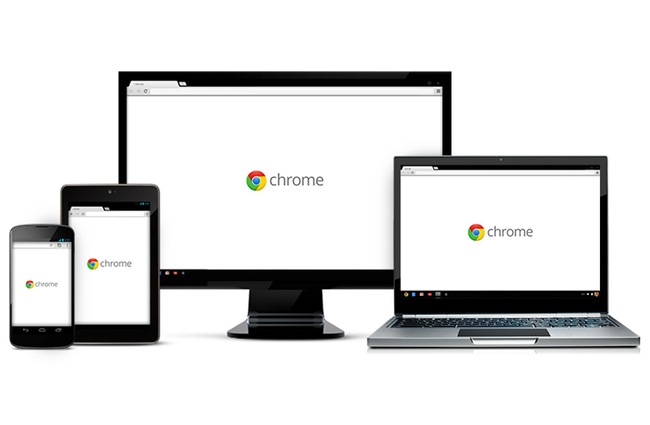
**«ТОМСКИЙ ЭКОНОМИКО-ПРОМЫШЛЕННЫЙ КОЛЛЕДЖ»**

*Методическое объединение информационных технологий*

**УЛЯХИН ВАСИЛИЙ АЛЕКСЕЕВИЧ**

**ПОДГОТОВКА К ДЭ ПО КОМПЕТЕНЦИИ «ВЕБ-ТЕХНОЛОГИИ»**

*Практикум*



ТОМСК 2022

ОГЛАВЛЕНИЕ

[ЗАДАНИЕ ДЛЯ ДЕМОНСТРАЦИОННОГО ЭКЗАМЕНА ПО КОМПЛЕКТУ ОЦЕНОЧНОЙ ДОКУМЕНТАЦИИ 3](#_Toc130953682)

[КРИТЕРИИ ОЦЕНИВАНИЯ 7](#_Toc130953683)

[ОБЗОР ФРЕЙМВОРКА YII2 15](#_Toc130953684)

[Контроллеры 16](#_Toc130953685)

[Модели 18](#_Toc130953686)

[Представления 21](#_Toc130953687)

[Виджеты 22](#_Toc130953688)

[Аутентификация 23](#_Toc130953689)

[ПРИМЕР РЕШЕНИЯ ЗАДАНИЯ ДЭМОНСТРАЦИОННОГО ЭКЗАМЕНА 24](#_Toc130953690)

[Установка и настройка фреймворка YII2 24](#_Toc130953691)

[Аутентификация 26](#_Toc130953692)

[Регистрация 29](#_Toc130953693)

[Меню сайта 32](#_Toc130953694)

[Админ панель 33](#_Toc130953695)

[Категории товаров 34](#_Toc130953696)

[Товары 35](#_Toc130953697)

# ЗАДАНИЕ ДЛЯ ДЕМОНСТРАЦИОННОГО ЭКЗАМЕНА ПО КОМПЛЕКТУ ОЦЕНОЧНОЙ ДОКУМЕНТАЦИИ

|  |  |
| --- | --- |
| Номер компетенции | 17 |
| Название компетенции | Веб-технологии |
| Номер КОД | 1.4 |

**ВВЕДЕНИЕ**

**Технологии этого модуля:** Разработка интернет-магазина.

**Время на выполнение:** 5 часов.

К вам обратилась молодая компания «Copy Star», которая занимается продажей копировального оборудования. Заказчик хочет получить небольшой сайт для продажи своего оборудования.

Так как заказчик ограничен в деньгах, но в будущем планируется расширять ассортимент, то он просит сделать небольшую админ-панель.

Заказчик планирует дальнейшее развитие проекта, поэтому использование фреймворков будет плюсом.

**ВНИМАНИЕ! Проверяться будут только работы, загруженные на сервер!**

**ОПИСАНИЕ ПРОЕКТА И ЗАДАЧ.**

**ОБЩЕЕ ОПИСАНИЕ ПРОЕКТА**

Все пользователи системы подразделяются на три группы:

1. Администраторы;
2. Посетитель сайта;
3. Клиент.

Функционал администратора:

1. Добавление, удаление, редактирование данных о товаре;
2. Добавление, удаление категорий товаров;
3. Просмотр всех заказов и смена их статусов.

Функционал посетителя сайта:

1. Просмотр товаров и рубрик сайта.

Функционал клиента:

1. Добавление товара(ов) в корзину;
2. Редактирование корзины;
3. Создание заказа;
4. Просмотр своих заказов.

Ваша задача – реализовать интернет-магазин.

**ОБЩИЕ ТРЕБОВАНИЯ**

Админ панель должная быть доступна по адресу http://xxxxxx/admin под логином admin и паролем admin11. Она не должна быть доступна для других пользователей.

**СПЕЦИФИЧЕСКИЕ ТРЕБОВАНИЯ К ИНТЕРНЕТ-МАГАЗИНУ**

**Рубрики сайта**

Всем пользователям доступны рубрики: "О нас", "Каталог", "Где нас найти?". Также в меню нужно сделать кнопку вход и регистрация.

**Страница «О нас»**

На странице представлен логотип компании и написан девиз компании. Также представлен слайдер ("Новинки компании") с пятью последними добавленными товарами (выводится фото товара и его название).

**Страница «Каталог»**

На странице по умолчанию представлены все товары, упорядоченные по новизне (от новых добавленных к более старым). Упорядочить товары можно: по году производства, по наименованию, по цене. Также товары можно отфильтровать по категориям (лазерные принтеры, струйные принтеры, термопринтеры). Каждая карточка товара представлена изображением, названием и ценой. При клике на товар открывается отдельная страница товара. Пользователю всегда показывает товары, которые только есть в наличии.

**Страница «Товар»**

Каждая страница содержит фото товара, наименование, цену и характеристики (страна-производитель, год выпуска, модель).

**Страница «Где нас найти?»**

На странице располагается карта и контактные данные (адрес, номер телефона, email).

**Регистрация нового клиента**

Форма регистрации нового пользователя в системе должна содержать следующие поля:

* 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);
* email – обязательное и уникальное поле, проверка на email;
* password – обязательное поле, не менее 6-ти символов;
* password\_repeat – обязательное поле, должно совпадать с полем password;
* rules - согласие с правилами регистрации.

При ошибках валидации пользователю должны выводиться сообщения без перезагрузки страницы.

**Аутентификация**

Форма авторизации пользователя в системе должна содержать следующие поля:

* login;
* password;

При ошибках валидации пользователю должны выводиться сообщения.

**Выход**

Позволяет пользователю сделать логаут, после перенаправляет на страницу "О нас".

**ФУНКЦИОНАЛ КЛИЕНТА**

**Формирование корзины**

После авторизации у пользователя появляется кнопка "В корзину" на карточках товаров в каталоге, а также на странице самого товара. Каждое нажатие добавляет 1 товар в корзину. Нельзя добавить в корзину больше товаров, чем есть в наличии (выдается сообщение об ошибке без перезагрузки страницы).

**Управление корзиной**

В корзине можно добавить или убрать единицу каждого товара (учитывая ограничения по количеству в наличии). После чего пользователь может сформировать заказ.

**Оформление заказа**

Пользователь вводит свой пароль для подтверждения заказа и нажимает кнопку "Сформировать заказ" (выдается сообщение об ошибке без перезагрузки страницы в случае неверного ввода пароля).

**Просмотр заказов**

Пользователь может просмотреть список своих заказов (в каждом есть количество товаров, его наименование и их статус), упорядоченных от новых к старым. Пользователь также имеет возможность удалить новые заказы.

**ФУНКЦИОНАЛ АДМИНИСТРАТОРА**

**Просмотр списка всех заказов**

Администратор может просматривать все заказы, фильтруя их по: новые, подтвержденные, отмененные. В списке видны таймстамп заказа, ФИО заказчика и количество заказанных товаров. По умолчанию у всех заказов статус "Новый".

**Управление заказом**

Администратор может отменить заказ, указав причину отказа или подтвердить заказ.

**Управление товарами**

Администратор опубликовывает, удаляет и редактирует товары.

**Управление категориями**

В будущем планируется расширение спектра товаров для этого нужно сделать функцию добавления и удаления категории.

**ИНСТРУКЦИЯ ДЛЯ КОНКУРСАНТА**

Разработанный интернет магазин должен быть доступен по адресу http://xxxxxx-m5.wsr.ru/, где xxxxxx - логин участника (указан на индивидуальной карточке).

**СИСТЕМА ОЦЕНКИ**

|  |  |  |
| --- | --- | --- |
| **Секция** | **Критерий** | **Сумма** |
| 1 | Организация работы и управление | 2,00 |
| 2 | Коммуникация и навыки межличностного общения | 2,00 |
| 3 | Графический дизайн | 10,00 |
| 4 | Верстка | 10,00 |
| 5 | Программирование на стороне клиента | 10,00 |
| 6 | Программирование на стороне сервера | 10,00 |
| 7 | CMS | 0,00 |
| **Всего** |  | 44,00 |

# КРИТЕРИИ ОЦЕНИВАНИЯ

| Sub Criteria ID | Sub Criteria  Name or Description | Aspect Type  O = Obj  J = Judg | Aspect - Description | Judg Score | Extra Aspect Description (Obj or Subj)  OR  Judgement Score Description (Judg only) | WSSS Section | Max Mark |
| --- | --- | --- | --- | --- | --- | --- | --- |
| A1 | Организация работы и управления |  |  |  |  |  |  |
|  |  | O | Результаты работы представлены по пути согласно заданию |  |  | 1 | 0.25 |
|  |  | O | Данные для авторизации администратора соответствуют заданию |  | Логин: admin  Пароль: admin11 | 1 | 0.25 |
|  |  | O | Админ панель доступна по адресу из задания |  |  | 1 | 0.25 |
|  |  | J | Представленный результат соответствует требованиям задания |  |  | 1 | 1.25 |
|  |  |  |  | 0 | Работа не выполнена |  |  |
|  |  |  |  | 1 | Реализована минимально необходимая часть функционала |  |  |
|  |  |  |  | 2 | Работа соответствует заданию, но имеются некоторые несоответствия |  |  |
|  |  |  |  | 3 | Работа полностью соответствует заданию |  |  |
| A2 | Коммуникативные и межличностные навыки |  |  |  |  |  |  |
|  |  | O | Осмысленные имена функций и переменных |  | За использование транслита штраф – 0.25 | 2 | 0.5 |
|  |  | J | Качество PHP кода |  |  |  | 0.75 |
|  |  |  |  | 0 | В коде нельзя быстро разобраться |  |  |
|  |  |  |  | 1 | Осмысленное именование переменных, функций |  |  |
|  |  |  |  | 2 | Уровень 1 + Используется PHP framework или чистый PHP (с использованием классов, ООП) |  |  |
|  |  |  |  | 3 | Используется PHP framework, код организован и легко читаем |  |  |
|  |  | J | Использование комментариев в коде |  |  | 2 | 0.25 |
|  |  |  |  | 0 | Отсутствуют комментарии, созданные пользователем |  |  |
|  |  |  |  | 1 | Используются комментарии, но их недостаточно и/или комментарии некорректны |  |  |
|  |  |  |  | 2 | Осмысленные комментарии, но возможны улучшения |  |  |
|  |  |  |  | 3 | Используются осмысленные вводные и пояснительные комментарии |  |  |
|  |  | J | Файлы веб-проекта хорошо организованы. |  |  | 2 | 0.50 |
|  |  |  |  | 0 | Файлы не структурированы |  |  |
|  |  |  |  | 1 | Названия файлов указывают на их содержимое |  |  |
|  |  |  |  | 2 | Файлы сгруппированы в каталоги, имена каталогов указывают на их содержимое |  |  |
|  |  |  |  | 3 | Структура файлов позволяет без проблем найти необходимые данные |  |  |
| A3 | Графический дизайн веб-страниц |  |  |  |  |  |  |
|  |  | O | На сайте используется логотип компании |  |  | 3 | 0.25 |
|  |  | O | Используется хотя бы один из предоставленных шрифтов |  |  | 3 | 0.50 |
|  |  | J | Разработанный логотип подчеркивает тему сайта |  |  | 3 | 0.75 |
|  |  |  |  | 0 | Логотип не разработан либо крайне низкого качества |  |  |
|  |  |  |  | 1 | Логотип не подходит под тему сайта |  |  |
|  |  |  |  | 2 | Логотип в целом подходит под тему сайта |  |  |
|  |  |  |  | 3 | Идеальное попадание в тему сайта |  |  |
|  |  | J | Качество реализации логотипа |  |  | 3 | 0.75 |
|  |  |  |  | 0 | Низкий уровень реализации или отсутствие логотипа |  |  |
|  |  |  |  | 1 | Некоторые элементы реализованы хорошо, используются некоторые цвета сайта |  |  |
|  |  |  |  | 2 | Все элементы реализованы хорошо, используются основные цвета сайта |  |  |
|  |  |  |  | 3 | Профессиональная реализация работы |  |  |
|  |  | J | Интерфейс понятен и удобен для использования |  |  | 3 | 1.00 |
|  |  |  |  | 0 | Сайтом крайне сложно пользоваться или работа не представлена |  |  |
|  |  |  |  | 1 | Неудобный интерфейс |  |  |
|  |  |  |  | 2 | Удобный интерфейс |  |  |
|  |  |  |  | 3 | Крайне хорошо проработанный интерфейс, использовать удобно |  |  |
|  |  | J | Учитываются тенденции и требования отрасли к мобильной (смартфон или планшет по заданию) версии страницы |  |  | 3 | 1.00 |
|  |  |  |  | 0 | Мобильная версия не представлена или не учитываются особенности мобильных платформ |  |  |
|  |  |  |  | 1 | Слабо учитываются особенности мобильных платформ |  |  |
|  |  |  |  | 2 | Особенности мобильных платформ отражены, но возможны улучшения |  |  |
|  |  |  |  | 3 | Полностью учитываются особенности мобильных платформ (увеличенный шрифт, измененная структура, крупные элементы для touch-взаимодействия, скрыты некоторые медиа-объекты и т.д.) |  |  |
|  |  | J | Качество реализации страницы "Каталог" |  |  | 3 | 1.50 |
|  |  |  |  | 0 | Страница не создана |  |  |
|  |  |  |  | 1 | Дизайн страницы простой |  |  |
|  |  |  |  | 2 | Дизайн страницы реализован, но стиль отличается от других страниц |  |  |
|  |  |  |  | 3 | Дизайн страницы полностью советует заданию |  |  |
|  |  | J | Визуальная согласованность элементов управления |  |  | 3 | 0.25 |
|  |  |  |  | 0 | Большинство элементов необоснованно отличаются по размеру, оформлению |  |  |
|  |  |  |  | 1 | Некоторые элементы согласованы между собой |  |  |
|  |  |  |  | 2 | Большинство элементов согласованы между собой, но возможны улучшения |  |  |
|  |  |  |  | 3 | Полная согласованность |  |  |
|  |  | J | Идентичность используемых изображений |  |  | 3 | 0.50 |
|  |  |  |  | 0 | Изображения разные, какие-то обработаны, какие-то нет |  |  |
|  |  |  |  | 1 | Некоторые изображения имеют одинаковый стиль |  |  |
|  |  |  |  | 2 | Почти все изображения обработаны в одном стиле |  |  |
|  |  |  |  | 3 | Все изображения отлично обработаны в одинаковом стиле |  |  |
|  |  | J | Единая типографика для всех экранов |  |  | 3 | 0.75 |
|  |  |  |  | 0 | Нормы типографики не соблюдаются |  |  |
|  |  |  |  | 1 | Минимальное соблюдение норм типографики |  |  |
|  |  |  |  | 2 | Нормы типографики соблюдаются, но возможно улучшение |  |  |
|  |  |  |  | 3 | Полное соблюдение норм типографики |  |  |
|  |  | J | Качество информативных сообщений о выполнении каких-либо действий |  |  | 3 | 0.50 |
|  |  |  |  | 0 | Нет или alert |  |  |
|  |  |  |  | 1 | Простые сообщения |  |  |
|  |  |  |  | 2 | Сообщения красивые и вписываются в дизайн |  |  |
|  |  |  |  | 3 | Необычные сообщения, улучшающие впечатление |  |  |
|  |  | J | Визуальная согласованность страниц |  |  | 3 | 0.75 |
|  |  |  |  | 0 | Страницы существенно отличаются по оформлению, размерам элементов и т.д. |  |  |
|  |  |  |  | 1 | Некоторые страницы согласованы между собой |  |  |
|  |  |  |  | 2 | Большинство страниц согласованы между собой, но возможны улучшения |  |  |
|  |  |  |  | 3 | Полная согласованность страниц |  |  |
|  |  | J | Качество реализации админ панели |  |  | 3 | 1.50 |
|  |  |  |  | 0 | Админ панель не создана |  |  |
|  |  |  |  | 1 | Дизайн простой |  |  |
|  |  |  |  | 2 | Дизайн реализован, но стиль отличается от других страниц |  |  |
|  |  |  |  | 3 | Дизайн админ панели полностью соответствует заданию |  |  |
| A4 | Верстка страниц |  |  |  |  |  |  |
|  |  | O | Валидность HTML кода |  | Штраф – 0.25 за каждый тип ошибки | 4 | 0.75 |
|  |  | O | Валидность CSS кода |  | Штраф – 0.5 за каждый тип ошибки | 4 | 1.50 |
|  |  | O | Свёрстан слайдер и девиз компании на главной странице |  | Штраф – 0.4 за отсутствующий элемент | 4 | 0.80 |
|  |  | O | В мобильной версии слайдер отображается корректно |  |  | 4 | 0.50 |
|  |  | O | Свёрстан экран с карточками товара со всеми данными |  | Штраф – 0.3 за отсутствующий элемент | 4 | 0.70 |
|  |  | O | Свёрстан экран с товаром |  | Штраф – 0.25 за каждый пропущенный элемент | 4 | 0.75 |
|  |  | O | Сверстана страница "Где нас найти?" |  | Штраф – 0.3 за отсутствующий элемент | 4 | 0.60 |
|  |  | O | Сверстана кнопка (ссылка) Выход |  |  | 4 | 0.20 |
|  |  | O | Сверстана форма регистрации |  | Штраф за каждый отсутствующий элемент 0.2 | 4 | 0.80 |
|  |  | O | В форме регистрации поля ввода e-mail имеет корректный тип |  | Или проводится валидация адреса e-mail средствами js | 4 | 0.20 |
|  |  | O | Сверстана форма авторизации |  |  | 4 | 0.50 |
|  |  | O | Сверстана админ панель |  | Штраф за каждый отсутствующий элемент 0.25 (просмотр и изменение статуса заказов, удаление, добавление категории) | 4 | 0.75 |
|  |  | O | Сверстана корзина |  | Штраф за каждый отсутствующий элемент 0.3 | 4 | 0.60 |
|  |  | O | Сверстан экран подтверждения заказа |  |  | 4 | 0.25 |
|  |  | O | Сверстана страница просмотра своих заявок |  | Штраф за каждый отсутствующий элемент 0.5 | 4 | 0.25 |
|  |  | O | Сверстаны элементы для фильтрации товаров по категориям |  |  | 4 | 0.25 |
|  |  | O | Сверстаны элементы для упорядочивания товаров |  | Штраф – 0.3 за отсутствующий элемент | 4 | 0.60 |
| A5 | Программирование на стороне клиента |  |  |  |  |  |  |
|  |  | O | Реализованы анимированные сообщения об ошибках при валидации вводимых значений |  | штраф 0,3 без анимации | 5 | 0.50 |
|  |  | O | При ошибке заполнения поле визуально выделяется |  | штраф 0,3, если выделение поля происходит только стандартными возможностями браузера | 5 | 0.50 |
|  |  | O | Проверка валидации полей ФИО выполняется |  |  | 5 | 0.50 |
|  |  | O | Форма регистрации не отправляется, если данные не заполнены |  |  | 5 | 0.25 |
|  |  | O | Проверка на уникальность логина проходит без перезагрузки страницы |  |  | 5 | 0.75 |
|  |  | O | Проверка повтора пароля |  |  | 5 | 0.25 |
|  |  | O | Проверка согласия с правилами регистрации |  |  | 5 | 0.25 |
|  |  | O | Реализована анимация слайдера |  |  | 5 | 0.50 |
|  |  | O | Требования к полям формы регистрации проходят без перезагрузки страницы |  |  | 5 | 0.50 |
|  |  | O | Проверка при добавлении товара в корзину учитывает наличие товаров |  |  | 5 | 0.75 |
|  |  | O | Появляется сообщение об ошибке добавления товара в корзину/корзине |  |  | 5 | 0.50 |
|  |  | O | Подтверждение заказа происходит без перезагрузки страницы |  |  | 5 | 0.20 |
|  |  | O | В слайдере выводиться пять последних товаров |  |  | 5 | 0.50 |
|  |  | O | JS код работает без отображения ошибок |  |  | 5 | 0.30 |
|  |  | O | Реализована сортировка товаров в каталоге |  |  | 5 | 0.75 |
|  |  | O | Кнопка "В корзину" есть только у авторизированного пользователя |  |  | 5 | 0.75 |
|  |  | J | Качество JS кода |  |  | 5 | 1.00 |
|  |  |  |  | 0 | В коде нельзя быстро разобраться |  |  |
|  |  |  |  | 1 | Осмысленное именование переменных, функций |  |  |
|  |  |  |  | 2 | Уровень 1 + Используется js framework или чистый JS (с использованием классов, ООП) |  |  |
|  |  |  |  | 3 | Используется js framework, код организован и легко читаем |  |  |
|  |  | J | Общее впечатление от реализации анимации слайдера |  |  | 5 | 1.25 |
|  |  |  |  | 0 | Отсутствует |  |  |
|  |  |  |  | 1 | Анимация реализована стандартно |  |  |
|  |  |  |  | 2 | Анимация имеет интересные фичи |  |  |
|  |  |  |  | 3 | Уникальная анимация |  |  |
| A6 | Программирование на стороне сервера |  |  |  |  |  |  |
|  |  | O | Данные сохраняются в БД |  |  | 6 | 0.50 |
|  |  | O | Реализована авторизация пользователей |  |  | 6 | 0.50 |
|  |  | O | Админ панель доступна только администратору |  |  | 6 | 0.25 |
|  |  | O | Просмотр своих заявок доступен только зарегистрированному пользователю |  |  | 6 | 0.25 |
|  |  | O | В каталоге не отображаются товары, которых нет в наличии |  |  | 6 | 0.25 |
|  |  | O | При не правильной паре логин/пароль выводится сообщение |  |  | 6 | 0.25 |
|  |  | O | Реализована регистрация |  |  | 6 | 0.75 |
|  |  | O | Проверка валидности логина |  |  | 6 | 0.25 |
|  |  | O | Проверка уникальности логина |  |  | 6 | 0.50 |
|  |  | O | Реализован выход |  |  | 6 | 0.25 |
|  |  | O | Клиент может сделать заказ |  |  | 6 | 0.50 |
|  |  | O | При добавлении товара в корзину учитывается его количество |  |  | 6 | 0.25 |
|  |  | O | При создании заказа в БД создается временная метка |  |  | 6 | 0.25 |
|  |  | O | При создании заказа ему присваивается статус "Новый" |  |  | 6 | 0.25 |
|  |  | O | Пользователь может просмотреть свои заказы |  |  | 6 | 0.50 |
|  |  | O | При отображении заказов сортируются в соответствии с заданием |  |  | 6 | 0.25 |
|  |  | O | Для администратора реализована фильтрация заказов по статусу |  |  | 6 | 0.25 |
|  |  | O | Пользователь может удалить заказы со статусом "Новый" |  |  | 6 | 0.50 |
|  |  | O | Администратор может менять статус заказа на "Отменен" |  | Если комментарий не добавляется в БД штраф 0,25 | 6 | 0.50 |
|  |  | O | Администратор может менять статус заказа на "Подтверждён" |  |  | 6 | 0.50 |
|  |  | O | Администратор может добавить, редактировать и удалять товар |  | Штраф 0,25 за отсутствие функции | 6 | 0.50 |
|  |  | O | Администратор может добавлять новые категории |  |  | 6 | 0.25 |
|  |  | O | Администратор может удалять категории |  |  | 6 | 0.25 |
|  |  | O | При удалении категории администратором удаляются все товары данной категории |  |  | 6 | 0.50 |
|  |  | O | Отсутствуют ошибки, связанные с безопасностью |  | mysql инъекции, XSS-атаки | 6 | 1.00 |
| **Итого** | | | | | | | 44 |

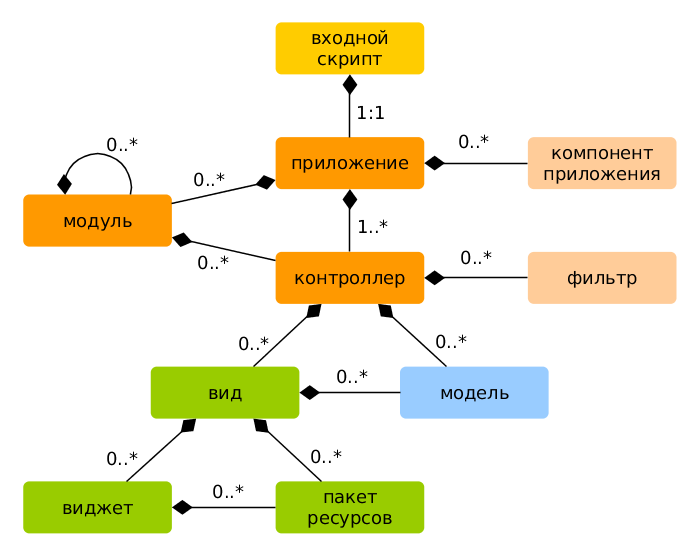
# ОБЗОР ФРЕЙМВОРКА YII2

Yii приложения организованы согласно шаблону проектирования модель-представление-контроллер (MVC). Модели представляют собой данные, бизнес логику и бизнес правила; представления отвечают за отображение информации, в том числе и на основе данных, полученных из моделей; контроллеры принимают входные данные от пользователя и преобразовывают их в понятный для моделей формат и команды, а также отвечают за отображение нужного представления.

Кроме MVC, Yii приложения также имеют следующие сущности:

* входные скрипты: это PHP скрипты, которые доступны напрямую конечному пользователю приложения. Они ответственны за запуск и обработку входящего запроса;
* приложения: это глобально доступные объекты, которые осуществляют корректную работу различных компонентов приложения и их координацию для обработки запроса;
* компоненты приложения: это объекты, зарегистрированные в приложении и предоставляющие различные возможности для обработки текущего запроса;
* модули: это самодостаточные пакеты, которые включают в себя полностью все средства для MVC. Приложение может быть организованно с помощью нескольких модулей;
* фильтры: это код, который должен быть выполнен до и после обработки запроса контроллерами;
* виджеты: это объекты, которые могут включать в себя представления. Они могут содержать различную логику и быть использованы в различных представлениях.

Ниже на диаграмме представлена структурная схема приложения:



Приложения — это объекты, которые управляют всей структурой и жизненным циклом прикладной системы Yii. Каждая прикладная система Yii включает в себя один объект приложения, который создается во входном скрипте и глобально доступен через \Yii::$app.

## Контроллеры

Контроллеры являются частью MVC архитектуры. Это объекты классов, унаследованных от yii\base\Controller, отвечающие за обработку запроса и генерирование ответа. В сущности, после обработки запроса приложениями, контроллеры проанализируют входные данные, передадут их в модели, вставят результаты модели в представления, и в конечном итоге сгенерируют исходящие ответы.

**Действия**

Контроллеры состоят из действий, которые являются основными блоками, к которым может обращаться конечный пользователь и запрашивать исполнение того или иного функционала. В контроллере может быть одно или несколько действий.

Следующий пример показывает post контроллер с одним действием view:

namespace app\controllers;  
  
use Yii;  
use app\models\Post;  
use yii\web\Controller;  
use yii\web\NotFoundHttpException;

class PostController extends Controller

{  
 public function actionView($id)  
 {  
 $model = Post::*findOne*($id);  
 if ($model === null) {  
 throw new NotFoundHttpException;  
 }  
  
 return $this->render('view', [  
 'model' => $model,  
 ]);  
 }  
}

В действии view (определенном методом actionView()), код сначала загружает модель согласно запрошенному ID модели; Если модель успешно загружена, то код отобразит ее с помощью представления под названием view. В противном случае будет брошено исключение.

**Маршруты**

Конечные пользователи обращаются к действиям через так называемые маршруты. Маршрут — это строка, состоящая из следующих частей:

* ID контроллера: строка, которая уникально идентифицирует контроллер среди всех других контроллеров одного и того же приложения (или одного и того же модуля, если контроллер принадлежит модулю);
* ID действия: строка, которая уникально идентифицирует действие среди всех других действия одного и того же контроллера.

Маршруты имеют следующий формат:

ControllerID/ActionID

**Создание действий**

Создание действий не представляет сложностей так же, как и объявление так называемых методов действий в классе контроллера. Метод действия это public метод, имя которого начинается со слова action. Возвращаемое значение метода действия представляет собой ответные данные, которые будут высланы конечному пользователю. Приведенный ниже код определяет два действия index и hello-world:

<?php  
  
namespace app\controllers;  
  
use yii\web\Controller;  
  
class SiteController extends Controller  
{  
 public function actionIndex()  
 {  
 return $this->render('index');  
 }  
  
 public function actionHelloWorld()  
 {  
 return 'Hello World';  
 }  
}

**Правила наименования классов контроллеров**

Названия классов контроллеров могут быть получены из ID контроллеров следующими способами:

* Привести в верхний регистр первый символ в каждом слове, разделенном дефисами. Обратите внимание что, если ID контроллера содержит слэш, то данное правило распространяется только на часть после последнего слэша в ID контроллера;
* Убрать дефисы и заменить любой прямой слэш на обратный;
* Добавить суффикс Controller;
* Добавить в начало пространство имен контроллеров.

Ниже приведены несколько примеров, с учетом того, что пространство имен контроллеров имеет значение по умолчанию равное app\controllers:

* article соответствует app\controllers\ArticleController;
* post-comment соответствует app\controllers\PostCommentController;
* admin/post-comment соответствует app\controllers\admin\PostCommentController;

## Модели

Модели являются частью архитектуры MVC (Модель-Вид-Контроллер). Они представляют собой объекты бизнес-данных, правил и логики.

Вы можете создавать классы моделей путём расширения класса yii\base\Model или его дочерних классов. Базовый класс yii\base\Model поддерживает много полезных функций:

* Атрибуты: представляют собой рабочие данные и могут быть доступны как обычные свойства объекта или элементы массива;
* Метки атрибутов: задают отображение атрибута;
* Массовое присвоение: поддержка заполнения нескольких атрибутов в один шаг;
* Правила проверки: обеспечивают ввод данных на основе заявленных правил проверки;
* Экспорт Данных: разрешает данным модели быть экспортированными в массивы с настройкой форматов.

Класс Model также является базовым классом для многих расширенных моделей, таких как Active Record.

Модели предоставляют рабочие данные в терминах атрибутах. Каждый атрибут представляет собой публично доступное свойство модели. Метод yii\base\Model::attributes() определяет какие атрибуты имеет класс модели.

Вы можете получить доступ к атрибуту как к обычному свойству объекта:

$model->name = 'example';

echo $model->name;

**Определение Атрибутов**

По умолчанию, если ваш класс модели расширяется напрямую от yii\base\Model, то все не статичные публичные переменные являются атрибутами. Например, у класса модели ContactForm , который находится ниже, четыре атрибута: name, email, subject и body. Модель ContactForm используется для представления входных данных, полученных из HTML формы.

namespace app\models;  
  
use yii\base\Model;  
  
class ContactForm extends Model  
{  
 public $name;  
 public $email;  
 public $subject;  
 public $body;  
}

**Метки атрибутов**

При отображении значений или при получении ввода значений атрибутов, часто требуется отобразить некоторые надписи, связанные с атрибутами. Например, если атрибут назван firstName, Вы можете отобразить его как First Name, что является более удобным для пользователя, в тех случаях, когда атрибут отображается конечным пользователям в таких местах, как форма входа и сообщения об ошибках.

По умолчанию, метки атрибутов автоматически генерируются из названия атрибута. Генерация выполняется методом yii\base\Model::generateAttributeLabel(). Если Вы не хотите использовать автоматически сгенерированные метки, Вы можете переопределить метод yii\base\Model::attributeLabels(), чтобы явно объявить метку атрибута.

namespace app\models;  
  
use yii\base\Model;  
  
class ContactForm extends Model  
{  
 public $name;  
 public $email;  
 public $subject;  
 public $body;  
  
 public function attributeLabels()  
 {  
 return [  
 'name' => 'Your name',  
 'email' => 'Your email address',  
 'subject' => 'Subject',  
 'body' => 'Content',  
 ];  
 }  
}

**Правила валидации**

Когда данные модели, получены от конечных пользователей, они должны быть проверены, для того чтобы убедиться, что данные удовлетворяют определенным правилам (так называемым правилам валидации также известными как бизнес-правила). Например, дана модель ContactForm, возможно, Вы захотите убедиться, что все атрибуты являются не пустыми значениями, а атрибут email содержит допустимый адрес электронной почты. Если значения нескольких атрибутов не удовлетворяют соответствующим бизнес-правилам, то должны быть показаны соответствующие сообщения об ошибках, чтобы помочь конечному пользователю исправить допущенные ошибки.

Вы можете вызвать yii\base\Model::validate() для проверки полученных данных. Данный метод будет использовать правила валидации определённые в yii\base\Model::rules() для проверки каждого соответствующего атрибута. Если ошибок не найдено, то возвращается true, в противном случае возвращается false, а ошибки содержит свойство yii\base\Model::$errors. Например,

$model = new \app\models\ContactForm;  
  
*// модель заполнения атрибутов данными, вводимыми пользователем*$model->attributes = \Yii::*$app*->request->post('ContactForm');  
  
if ($model->validate()) {  
 *// все данные верны*} else {  
 *// иначе: $errors - это массив содержащий сообщения об ошибках* $errors = $model->errors;  
}

Объявляем правила валидации связанные с моделью, переопределяем метод [yii\base\Model::rules()](https://www.yiiframework.com/doc/api/2.0/yii-base-model#rules()-detail) возврата правил атрибутов модели которые следует удовлетворить. В следующем примере показаны правила проверки, объявленные в модели ContactForm:

public function rules()  
{  
 return [  
 *// name, email, subject и body атрибуты обязательны* [['name', 'email', 'subject', 'body'], 'required'],  
  
 *// атрибут email должен быть правильным email адресом* ['email', 'email'],  
 ];  
}

## Представления

Виды — это часть MVC архитектуры, это код, который отвечает за представление данных конечным пользователям. В веб приложениях виды создаются обычно в виде видов - шаблонов, которые суть PHP скрипты, в основном содержащие HTML код и код PHP, отвечающий за представление и внешний вид. Виды управляются компонентом приложения view, который содержит часто используемые методы для упорядочивания видов и их рендеринга. Для упрощения, мы будем называть виды - шаблоны просто видами.

Создание видов

Как мы упоминали ранее, вид - это просто PHP скрипт, состоящий из PHP и HTML кодa. В примере ниже - вид, который представляет форму авторизации. Как видите, PHP код здесь генерирует динамический контент, как, например, заголовок страницы и саму форму, тогда как HTML организует полученные данные в готовую html страницу.

<?php  
use yii\helpers\Html;  
use yii\widgets\ActiveForm;  
  
*/\* @var $this yii\web\View \*/  
/\* @var $form yii\widgets\ActiveForm \*/  
/\* @var $model app\models\LoginForm \*/*$this->title = 'Вход';  
?>  
 <h1><?= Html::*encode*($this->title) ?></h1>  
  
 <p>Пожалуйста, заполните следующие поля для входа на сайт:</p>  
  
<?php $form = ActiveForm::*begin*(); ?>  
<?= $form->field($model, 'username') ?>  
<?= $form->field($model, 'password')->passwordInput() ?>  
<?= Html::*submitButton*('Login') ?>  
<?php ActiveForm::*end*(); ?>

Внутри вида, вы можете использовать $this, которое представляет собой компонент вид, управляющий этим шаблоном и обеспечивающий его рендеринг.

Кроме $this, в виде могут быть доступны другие переменные, такие как $form и $model из примера выше. Эти переменные представляют собой данные, которые передаются в вид контроллерами или другими объектами, которые вызывают рендеринг вида.

## Виджеты

Виджеты представляют собой многоразовые строительные блоки, используемые в представлениях для создания сложных и настраиваемых элементов пользовательского интерфейса в рамках объектно-ориентированного подхода. Например, виджет выбора даты (date picker) позволяет генерировать интерактивный интерфейс для выбора дат, предоставляя пользователям приложения удобный способ для ввода данных такого типа. Все, что нужно для подключения виджета — это добавить следующий код в представление:

<?php use yii\bootstrap\DatePicker; ?>  
<?= DatePicker::*widget*(['name' => 'date']) ?>

В комплект Yii входит большое количество виджетов, например: active form, menu, виджеты jQuery UI, виджеты Twitter Bootstrap. Далее будут представлены базовые сведения о виджетах. Для получения сведений относительно использования конкретного виджета, следует обратиться к документации соответствующего класса.

**Использование Виджетов**

Главным образом, виджеты применяют в представлениях. Для того, чтобы использовать виджет в представлении, достаточно вызвать метод yii\base\Widget::widget(). Метод принимает массив настроек для инициализации виджета и возвращает результат его рендеринга. Например, следующий код добавляет виджет для выбора даты, сконфигурированный для использования русского в качестве языка интерфейса виджета и хранения вводимых данных в атрибуте from\_date модели $model.

<?php  
use yii\bootstrap\DatePicker;  
?>  
<?= DatePicker::*widget*([  
 'model' => $model,  
 'attribute' => 'from\_date',  
 'language' => 'ru',  
 'clientOptions' => [  
 'dateFormat' => 'yy-mm-dd',  
 ],  
]) ?>

Некоторые виджеты могут иметь внутреннее содержимое, которое следует располагать между вызовами методов yii\base\Widget::begin() и yii\base\Widget::end(). Например, для генерации формы входа, в следующем фрагменте кода используется виджет yii\widgets\ActiveForm. Этот виджет сгенерирует открывающий и закрывающий тэги <form> в местах вызова методов begin() и end() соответственно. При этом, содержимое, расположенное между вызовами указанных методов, будет выведено без каких-либо изменений.

<?php  
use yii\widgets\ActiveForm;  
use yii\helpers\Html;  
?>  
<?php $form = ActiveForm::*begin*(['id' => 'login-form']); ?>  
<?= $form->field($model, 'username') ?>  
<?= $form->field($model, 'password')->passwordInput() ?>  
<div class="form-group">  
 <?= Html::*submitButton*('Login') ?>  
</div>  
<?php ActiveForm::*end*(); ?>

## Аутентификация

Аутентификация — это процесс проверки подлинности пользователя. Обычно используется идентификатор (например, username или адрес электронной почты) и секретный токен (например, пароль или ключ доступа), чтобы судить о том, что пользователь именно тот, за кого себя выдаёт. Аутентификация является основной функцией формы входа.

Yii предоставляет фреймворк авторизации с различными компонентами, обеспечивающими процесс входа. Для использования этого фреймворка вам нужно проделать следующее:

* Настроить компонент приложения user;
* Создать класс, реализующий интерфейс yii\web\IdentityInterface.

Компонент user управляет статусом аутентификации пользователя. Он требует, чтобы вы указали identity class, который будет содержать текущую логику аутентификации. В конфигурации приложения по умолчанию, identity class для user задан как app\models\User.

Identity class должен реализовывать yii\web\IdentityInterface, который содержит следующие методы:

* findIdentity() – этот метод находит экземпляр identity class, используя ID пользователя. Этот метод используется, когда необходимо поддерживать состояние аутентификации через сессии.
* findIdentityByAccessToken() – этот метод находит экземпляр identity class, используя токен доступа. Метод используется, когда требуется аутентифицировать пользователя только по секретному токену (например, в RESTful приложениях, не сохраняющих состояние между запросами).
* getId()– этот метод возвращает ID пользователя, представленного данным экземпляром identity.
* getAuthKey() – этот метод возвращает ключ, используемый для основанной на cookie аутентификации. Ключ сохраняется в аутентификационной cookie и позже сравнивается с версией, находящейся на сервере, чтобы удостоверится, что аутентификационная cookie верная.
* validateAuthKey() – этот метод реализует логику проверки ключа для основанной на cookie аутентификации.

Если какой-то из методов не требуется, то можно реализовать его с пустым телом. Для примера, если у вас RESTful приложение, не сохраняющее состояние между запросами, вы можете реализовать только findIdentityByAccessToken() и getId(), тогда как остальные методы оставить пустыми.

В основном класс yii\web\User используют как компонент приложения user.

Можно получить identity текущего пользователя, используя выражение Yii::$app->user->identity. Оно вернёт экземпляр identity class, представляющий текущего аутентифицированного пользователя, или null, если текущий пользователь не аутентифицирован (например, гость).

# ПРИМЕР РЕШЕНИЯ ЗАДАНИЯ ДЭМОНСТРАЦИОННОГО ЭКЗАМЕНА

## Установка и настройка фреймворка YII2

Для установки Yii2 выполните следующие действия (локальная установка):

1. Запустите локальный веб-сервер OSPanel (C:\OSPanel\ Open Server Panel.exe).
2. Запустите консоль веб-сервера (Выполните [Меню → Дополнительно → Консоль]).
3. Обновите менеджер пакетов Composer (composer self-update).
4. Перейдите в каталог domains (cd domains).
5. Запустите установку фреймворка Yii (composer create-project yiisoft/yii2-app-basic xxxxx-m4.wsr.ru, где xxxxx – ваш логин).

Для установки Yii2 выполните следующие действия (удаленная установка):

1. Зайдите в личный кабинет по адресу <http://lk.wsr.ru/> используя данные с карточек.
2. Подключите сетевой диск W (запустите командную строку CMD и выполните представленную в личном кабинете команду net use).
3. Скачайте из личного кабинета архив с Yii2.
4. Перейдите в каталог W:\xxxxx-m4.wsr.ru\, где xxxxx – ваш логин.
5. Распакуйте содержимое архива (все файлы и папки внутри basic) в текущий каталог.

Для первоначальной настройки фреймворка Yii2 выполните следующие действия:

1. В корневой директории вашего проекта создайте файл .htaccess со следующим содержимым:

RewriteEngine on  
RewriteRule (.\*)? /web/$1

Данная настройка перенаправит все запросы, идущие на фреймворк, в папку web – точку входа в приложение.

1. Откройте файл assets\AppAssett.php и подключите к вашему проекту свою каскадную таблицу стилей и свой JS-скрипт (создайте данные файлы и папки в директории web).

class AppAsset extends AssetBundle  
{  
 public $basePath = '@webroot';  
 public $baseUrl = '@web';  
 public $css = [  
 'css/site.css',

*/\* Подключаем свой CSS \*/*

'css/my.css',  
 ];  
 public $js = [

*/\* Подключаем свой JS \*/*  
 'js/my.js',  
 ];  
 public $depends = [  
 'yii\web\YiiAsset',  
 'yii\bootstrap5\BootstrapAsset'  
 ];  
}

1. Откройте файл config\db.php и укажите настройки для подключения к базе данных.

<?php  
  
return [  
 'class' => 'yii\db\Connection',  
 'dsn' => 'mysql:host=localhost;dbname=ИмяБазыДанных',  
 'username' => 'ЛогинПользователяБД',  
 'password' => 'ПарольПользователяБД',  
 'charset' => 'utf8',];

1. Откройте файл config/web.php и установите значение текущего языка локализации сообщений на русский язык, задайте случайный набор символов параметру валидации cookieValidationKey и объявите параметр базового URL-адрес проекта baseUrl.

$config = [  
 'id' => 'basic',  
 'basePath' => *dirname*(*\_\_DIR\_\_*),  
 'bootstrap' => ['log'],  
 'aliases' => [  
 '@bower' => '@vendor/bower-asset',  
 '@npm' => '@vendor/npm-asset',  
 ],  
 'language' => 'ru-RU',  
 'components' => [  
 'request' => [  
 *// Вставьте секретный ключ* 'cookieValidationKey' => 'randomString',  
 'baseUrl' => '',  
 ],

1. Далее раскомментируйте параметр urlManager, который позволяет собирать URL динамически на основе правил маршрутизации и параметр allowedIPs раздела конфигурации gii модуля, отвечающий за открытые доступа к gii с конкретных IP адресов.

'urlManager' => [  
 'enablePrettyUrl' => true,  
 'showScriptName' => false,  
 'rules' => [  
 ],  
],

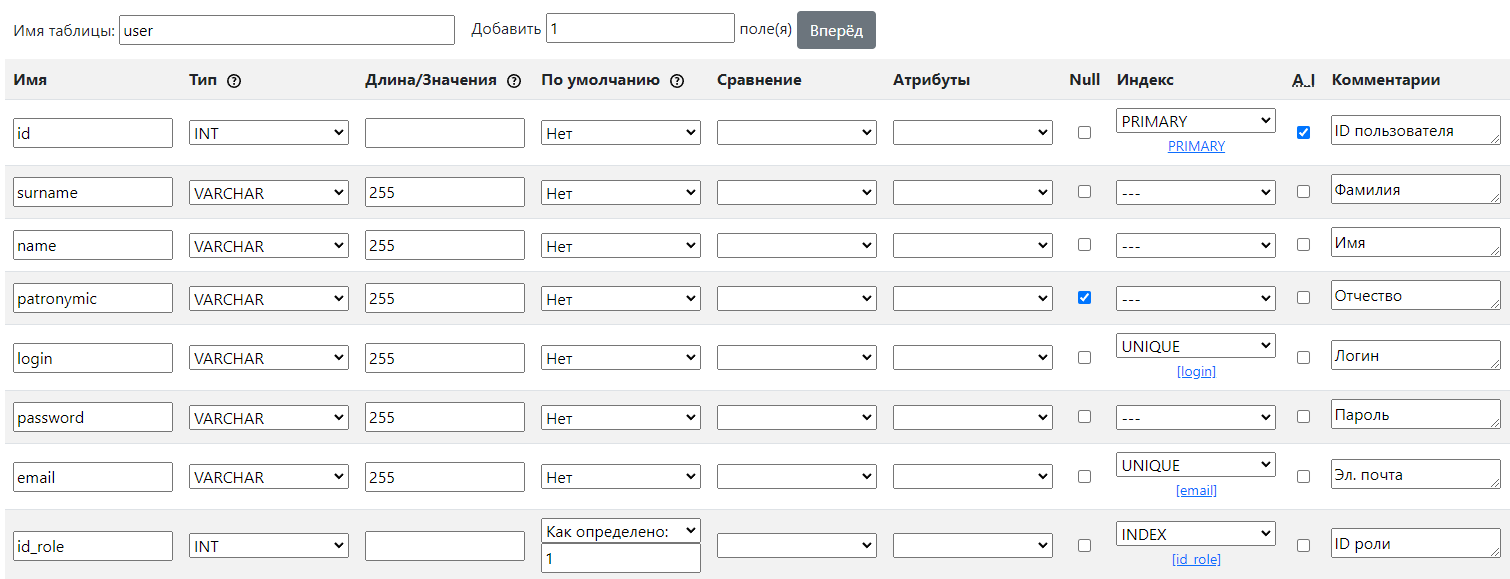
$config['modules']['gii'] = [  
 'class' => 'yii\gii\Module',  
 *//Раскомментируйте следующее правило, чтобы добавить свой IP-адрес.* 'allowedIPs' => ['\*', '::1'],  
];

1. Скопируйте выбранный вами файл шрифта из представленных медиафайлов в папку со шрифтами (***web\css\fonts\***).
2. Разработайте логотип компании размером 200x50 пикселей c прозрачным фоном и экспортируйте его в формате png (***web\images\logo.png***).
3. В PhpStorm переименуйте стандартную модель models\User.php в models\UserOld.php (операция Refactor → Rename). ***Обязательно снимите все чек-боксы!!!*** В файле models\UserOld.php также переименуйте название класса с User на UserOld.

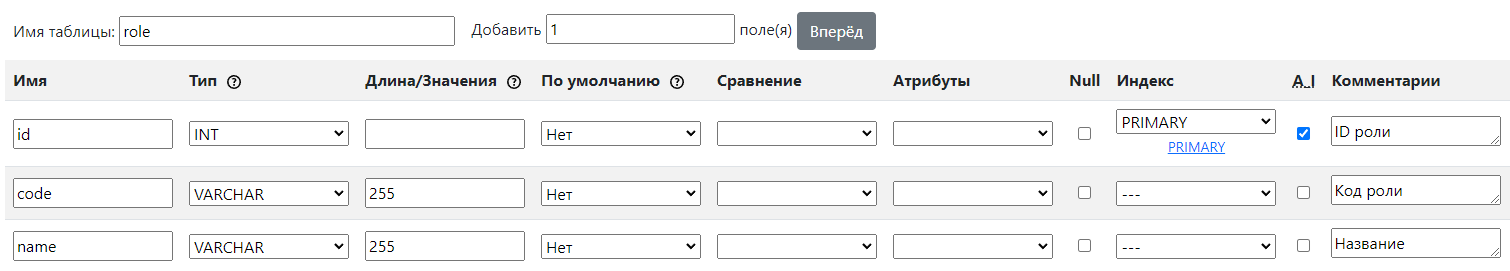
## Аутентификация

Зайдите в СУБД phpMyAdmin и создайте 2 таблицы.

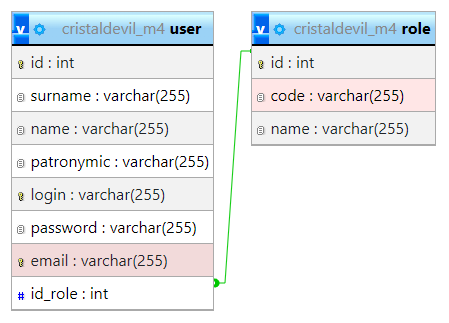
1. Таблица user хранит информацию о пользователях интернет-магазина.



1. Таблица role хранит информацию о ролях пользователей интернет-магазина.



В режиме дизайнера установите связи между таблицами, тип связи – CASCADE.

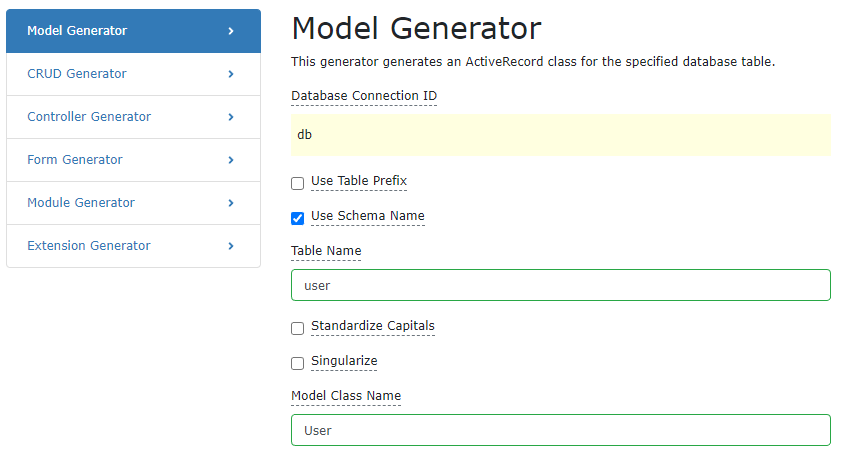


В таблицу role добавьте две записи.

|  |  |  |
| --- | --- | --- |
| **id** | **code** | **name** |
| 1 | user | Пользователь |
| 2 | admin | Администратор |

В таблицу user добавьте администратора интернет-магазина с логином и паролем из задания (логин – admin, пароль – admin11). Не забудьте указать для поля id\_role значение 2.

Зайдите в генератор Gii по адресу: http://xxxxx-m4.wsr.ru/gii, где xxxxx – ваш логин. Перейдите в Model Generator и сгенерируйте модели для 2х таблиц user и role.



В модели models\User.php укажите что класс User реализуется с помощью интерфейса IdentityInterface, напишите реализацию наследуемых методов данного интерфейса (все методы копируем с модели UserOld).

class User extends \yii\db\ActiveRecord implements IdentityInterface  
{  
 */\* Реализуем методы наследуемого интерфейса \*/* public function getId()  
 {  
 return $this->id;  
 }  
 public function validateAuthKey($authKey)  
 {  
 return false;  
 }  
 public function getAuthKey()  
 {  
 return null;  
 }  
 public static function findIdentity($id)  
 {  
 return static::*findOne*($id);  
 }  
 public static function findIdentityByAccessToken($token, $type = null)  
 {  
 return null;  
 }  
  
 */\* Поиск пользователя по логину и проверка введенного пароля \*/* public static function findByUsername($login)  
 {  
 return User::*findOne*(['login' => $login]);  
 }  
 public function validatePassword($password)  
 {  
 return $this->password === $password;  
 }  
   
 */\* Проверка на вход администратора \*/* public function isAdmin(){  
 return $this->role->code === "admin";  
 }

В файле шаблона views\layouts\main.php на 50 строке найдите строку Yii::$app->user->identity->username и измените на Yii::$app->user->identity->email (либо на любой другой существующий атрибут), чтобы избавиться от ошибки, так как атрибута username нет в модели User.

Русифицируйте поля меток в форме авторизации, для этого допишите метод attributeLabels() в модели LoginForm.

class LoginForm extends Model  
{  
 */\* Описание названий меток \*/* public function attributeLabels()  
 {  
 return [  
 'username' => 'Логин',  
 'password' => 'Пароль',  
 'rememberMe' => 'Запомнить меня',  
 ];  
 }

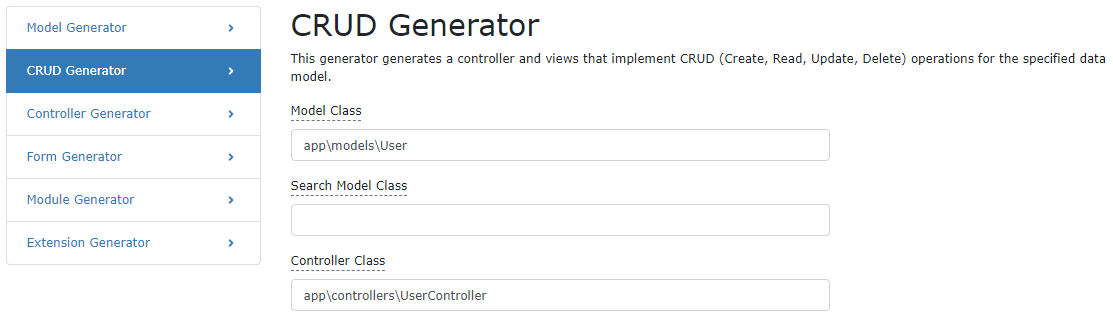
При авторизации администратора перенаправьте его в админ панель, смените редиректор в контроллере controllers\SiteController.php в методе actionLogon.

public function actionLogin()  
{  
 if (!Yii::*$app*->user->isGuest) {  
 return $this->goHome();  
 }  
  
 $model = new LoginForm();  
 if ($model->load(Yii::*$app*->request->post()) && $model->login()) {  
 */\* Если администратор, перенаправляем в админ-панель \*/* if (Yii::*$app*->user->identity->isAdmin()){  
 return $this->redirect(['/admin']);  
 }  
 return $this->goBack();  
 }  
  
 $model->password = '';  
 return $this->render('login', [  
 'model' => $model,  
 ]);  
}

Русифицируйте представление формы входа views\site\login.php .

## Регистрация

Зайдите в генератор Gii и перейдите в CRUD Generator, сгенерируйте контроллер и представления для модели User.



В модели models\User.php добавьте переменные для полей подтверждения пароля и согласия с правилами регистрации на сайте. Также добавьте подписи к новым полям и русифицируйте используемые подписи в форме.

*/\* Добавляем поля для подтверждения пароля и согласия с правилами сайта \*/*public $password\_repeat;  
public $rule;

*/\* Описание названий меток \*/*public function attributeLabels()  
{  
 return [  
 'id' => 'ID',  
 'surname' => 'Фамилия',  
 'name' => 'Имя',  
 'patronymic' => 'Отчество',  
 'login' => 'Логин',  
 'password' => 'Пароль',  
 'email' => 'Почта',  
 'id\_role' => 'Id Role',  
 *//Добавляем метки для полей password\_repeat и rule* 'password\_repeat' => 'Повтор пароля',  
 'rule' => 'Я согласен с правила регистрации на сайте',  
 ];  
}

Далее добавьте правила валидации полей из задания.

*/\* Правила валидации данных \*/*

public function rules()  
{  
 return [  
 [['surname', 'name', 'login', 'password', 'email'], 'required'],  
 [['id\_role'], 'integer'],  
 [['surname','name','patronymic','login','password','email'], 'string', 'max'=>255],  
 [['login'], 'unique'],  
 [['email'], 'unique'],  
 [['id\_role'], 'exist', 'skipOnError' => true,   
 'targetClass' => Role::class, 'targetAttribute' => ['id\_role' => 'id']],  
 */\* Дописываем правила валидации по заданию \*/* [['surname', 'name', 'patronymic'], 'match', 'pattern' => '/^[а-яА-ЯёЁ -]+$/u',  
 'message' => 'Разрешены русские буквы, пробел и тире'],  
 [['login'], 'match', 'pattern' => '/^[a-zA-Z0-9-]+$/i',  
 'message' => 'Разрешены латинские буквы, цифры и тире'],  
 [['email'], 'email'],  
 [['password\_repeat'], 'compare', 'compareAttribute' => 'password',  
 'message' => 'Пароли не совпадают'],  
 [['rule'], 'compare', 'compareValue' => 1,  
 'message' => 'Согласитесь с правилами регистрации на сайте'],  
 ];  
}

После регистрации перенаправьте пользователя на страницу авторизации, для этого смените редиректор в контроллере controllers\UserController.php в методе actionCreate.

public function actionCreate()  
{  
 $model = new User();  
  
 if ($this->request->isPost) {  
 if ($model->load($this->request->post()) && $model->save()) {  
 return $this->redirect(['/site/login']);  
 }  
 } else {  
 $model->loadDefaultValues();  
 }  
  
 return $this->render('create', [  
 'model' => $model,  
 ]);  
}

В файле представления views\user\\_form.php отредактируйте форму, удалите поле id\_role и добавьте два поля для подтверждения пароля и согласия с правила регистрации на сайте. Добавьте возможность проверки на уникальность логина и пароля.

<?php $form = ActiveForm::*begin*(); ?>  
  
<?= $form->field($model, 'surname')->textInput(['maxlength' => true]) ?>  
  
<?= $form->field($model, 'name')->textInput(['maxlength' => true]) ?>  
  
<?= $form->field($model, 'patronymic')->textInput(['maxlength' => true]) ?>  
  
<?= $form->field($model, 'login', ['enableAjaxValidation' => true])->textInput(['maxlength' => true]) ?>  
  
<?= $form->field($model, 'password')->passwordInput(['maxlength' => true]) ?>  
<?= $form->field($model, 'password\_repeat')->passwordInput(['maxlength' => true]) ?>  
  
<?= $form->field($model, 'email', ['enableAjaxValidation' => true])->textInput(['maxlength' => true]) ?>  
  
<?= $form->field($model, 'rule')->checkbox() ?>  
  
<div class="form-group">  
 <?= Html::*submitButton*('Создать', ['class' => 'btn btn-success']) ?>  
</div>  
  
<?php ActiveForm::*end*(); ?>

В контроллере controllers\UserController.php в методе actionCreate добавьте условие получение AJAX и POST запроса для выполнения валидации формы.

use yii\web\Response;

use yii\bootstrap5\ActiveForm;

...

public function actionCreate()  
{  
 $model = new User();  
  
 if (\Yii::*$app*->request->isAjax && $model->load(\Yii::*$app*->request->post())){  
 \Yii::*$app*->response->format = Response::*FORMAT\_JSON*;  
 return ActiveForm::*validate*($model);  
 }

Русифицируйте все представления для user.

Разрешите просмотр, удаление и обновление пользователей только администраторам, для этого измените метод behaviors в контроллере controllers\UserController.php (метод можно скопировать из SiteController, изменив действия и дописав параметр matchCallback).

class UserController extends Controller  
{  
 */\* Поведение. Доступ только для админа \*/* public function behaviors()  
 {  
 return [  
 'access' => [  
 'class' => AccessControl::class,  
 'only' => ['index', 'view', 'update', 'delete'],  
 'rules' => [  
 [  
 'actions' => ['index', 'view', 'update', 'delete'],  
 'allow' => true,  
 'roles' => ['@'],  
 'matchCallback' => function($rule, $action){  
 return \Yii::*$app*->user->identity->isAdmin();  
 }  
  
 ],  
 ],  
 ],  
 ];  
 }

В каскадных таблицах стилей web\css\my.css подключите шрифт и напишите правила для выделения ошибок в форме регистрации.

*/\* Подключаем внешний шрифт \*/*@font-face {  
 font-family: Aegean;  
 src: url("fonts/Aegean-8jJ2.ttf");  
}  
*/\* Применяем внешний шрифт для всех элементов \*/*\* {  
 font-family: Aegean;  
}  
*/\* Выделяем поля в форме не прошедшие валидацию \*/*input[aria-invalid="true"] {  
 border-color: red;  
}  
*/\* Выделяем сообщения с ошибками валидации полей в форме \*/*.help-block {  
 color: red;  
}

## Меню сайта

Откройте файл шаблона views\layouts\main.php , добавьте в меню навигации логотип компании и измените меню сайта согласно заданию, с учетом роли пользователя.

<?php  
NavBar::*begin*([  
 'brandLabel' => '<img src="http://'. $\_SERVER["SERVER\_NAME"] .'/images/logo.png" alt="Логотип" height="30" />',  
 'brandUrl' => Yii::*$app*->homeUrl,  
 'options' => ['class' => 'navbar-expand-md navbar-dark bg-dark fixed-top']  
]);  
echo Nav::*widget*([  
 'options' => ['class' => 'navbar-nav'],  
 'items' => [  
 */\* Меню для всех \*/* ['label' => 'О нас', 'url' => ['/site/index']],  
 ['label' => 'Каталог', 'url' => ['/product/index']],  
 ['label' => 'Где нас найти?', 'url' => ['/site/contact']],  
 */\* Меню для админа \*/* ['label' => 'Админ панель', 'url' => ['/admin/index'], 'visible' => !Yii::*$app*->user->isGuest && Yii::*$app*->user->identity->isAdmin()],  
['label' => 'Пользователи', 'url' => ['/user/index'], 'visible' => !Yii::*$app*->user->isGuest && Yii::*$app*->user->identity->isAdmin()],  
 ['label' => 'Категории', 'url' => ['/category/index'], 'visible' => !Yii::*$app*->user->isGuest && Yii::*$app*->user->identity->isAdmin()],  
 ['label' => 'Товары', 'url' => ['/product/index'], 'visible' => !Yii::*$app*->user->isGuest && Yii::*$app*->user->identity->isAdmin()],  
 ['label' => 'Заказы', 'url' => ['/order/index'], 'visible' => !Yii::*$app*->user->isGuest && Yii::*$app*->user->identity->isAdmin()],  
 */\* Меню для пользователя \*/* ['label' => 'Корзина', 'url' => ['/cart/index'], 'visible' => !Yii::*$app*->user->isGuest && !Yii::*$app*->user->identity->isAdmin()],  
 ['label' => 'Мои заказы', 'url' => ['/order/index'], 'visible' => !Yii::*$app*->user->isGuest && !Yii::*$app*->user->identity->isAdmin()],  
 */\* Регистрация и вход \*/* ['label' => 'Регистрация', 'url' => ['/user/create'], 'visible' => Yii::*$app*->user->isGuest],  
 Yii::*$app*->user->isGuest  
 ? ['label' => 'Вход', 'url' => ['/site/login']]  
 : '<li class="nav-item">'  
 . Html::*beginForm*(['/site/logout'])  
 . Html::*submitButton*(  
 'Выход (' . Yii::*$app*->user->identity->email . ')',  
 ['class' => 'nav-link btn btn-link logout']  
 )  
 . Html::*endForm*()  
 . '</li>'  
 ]  
]);

Добавьте блок для информации (необходим для оповещений в разделе в корзина и т.д.).

<main id="main" class="flex-shrink-0" role="main">  
 <div class="container">  
 <?php if (!empty($this->params['breadcrumbs'])): ?>  
 <?= Breadcrumbs::*widget*(['links' => $this->params['breadcrumbs']]) ?>  
 <?php endif ?>  
 <?= Alert::*widget*() ?>  
 <?= $content ?>  
 <div id="info-message"></div>  
 </div>  
</main>

## Админ панель

Создайте контроллер controllers\AdminController.php .

<?php  
  
namespace app\controllers;  
  
use yii\filters\AccessControl;  
use yii\web\Controller;

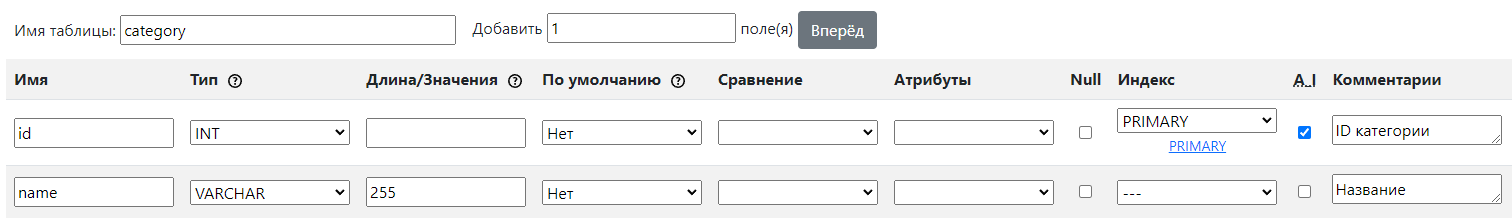
class AdminController extends Controller  
{  
 */\* Поведение. Доступ только для админа \*/* public function behaviors()  
 {  
 return [  
 'access' => [  
 'class' => AccessControl::class,  
 'only' => ['index'],  
 'rules' => [  
 [  
 'actions' => ['index'],  
 'allow' => true,  
 'roles' => ['@'],  
 'matchCallback' => function($rule, $action){  
 return \Yii::*$app*->user->identity->isAdmin();  
 }  
  
 ],  
 ],  
 ],  
 ];  
 }  
 */\* Маршрут admin/index отрисовывает представление view/admin/index.php \*/* public function actionIndex(){  
 return $this->render('index');  
 }  
}

Создайте представление view\admin\index.php .

<?php  
use yii\helpers\Html;  
?>  
<section>  
 <h2>Админ панель</h2>  
 <div>  
 <?= Html::*a*('Работа с пользователя', ['/user'], ['class' => 'btn btn-success']) ?>  
 <?= Html::*a*('Работа с категориями', ['/category'], ['class' => 'btn btn-success']) ?>  
 <?= Html::*a*('Работа с товарами', ['/product'], ['class' => 'btn btn-success']) ?>  
 <?= Html::*a*('Работа с заказами', ['/order'], ['class' => 'btn btn-success']) ?>  
 </div>  
</section>

## Категории товаров

В phpMyAdmin создайте таблицу category.



В таблицу category добавьте три записи.

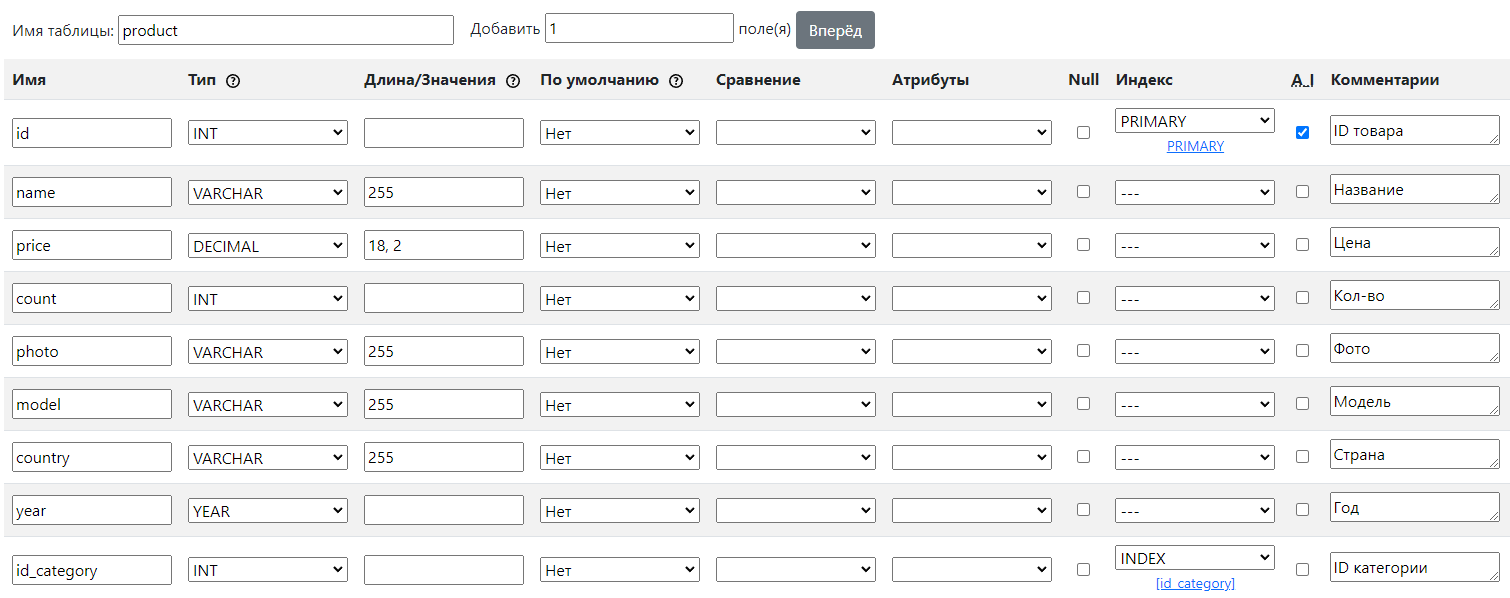
|  |  |
| --- | --- |
| **id** | **name** |
| 1 | Лазерные принтеры |
| 2 | Струйные принтеры |
| 3 | Термопринтеры |

Используя генератор Gii создайте модель, контроллер и представления. Русифицируйте представления и метки в модели (метод attributeLabels()). Разрешите создание, просмотр, удаление и обновление категорий только администраторам, для этого в контроллере controllers\CategoryController.php измените метод behaviors().

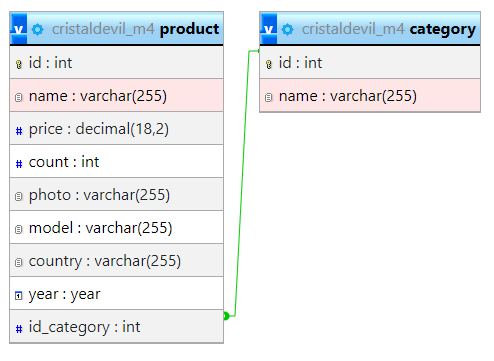
class CategoryController extends Controller  
{  
 */\* Поведение. Доступ только для админа \*/* public function behaviors()  
 {  
 return [  
 'access' => [  
 'class' => AccessControl::class,  
 'only' => ['create', 'index', 'view', 'update', 'delete'],  
 'rules' => [  
 [  
 'actions' => ['create', 'index', 'view', 'update', 'delete'],  
 'allow' => true,  
 'roles' => ['@'],  
 'matchCallback' => function($rule, $action){  
 return \Yii::*$app*->user->identity->isAdmin();  
 }  
 ],  
 ],  
 ],  
 ];  
 }

## Товары

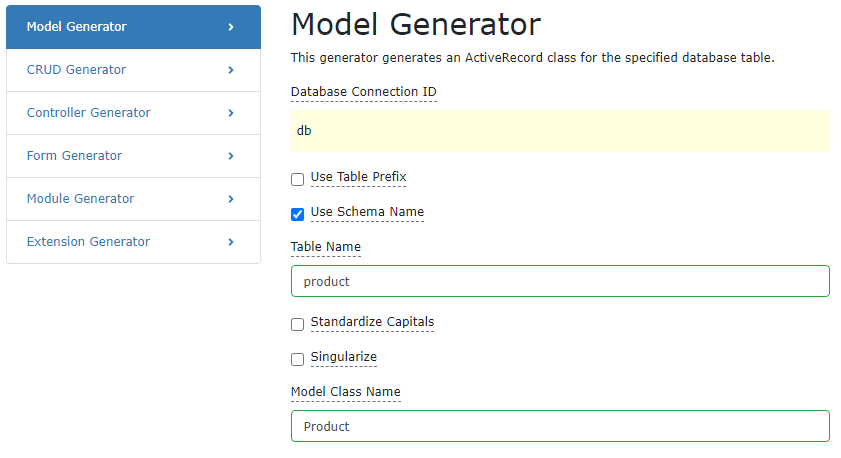
В phpMyAdmin создайте таблицу product.

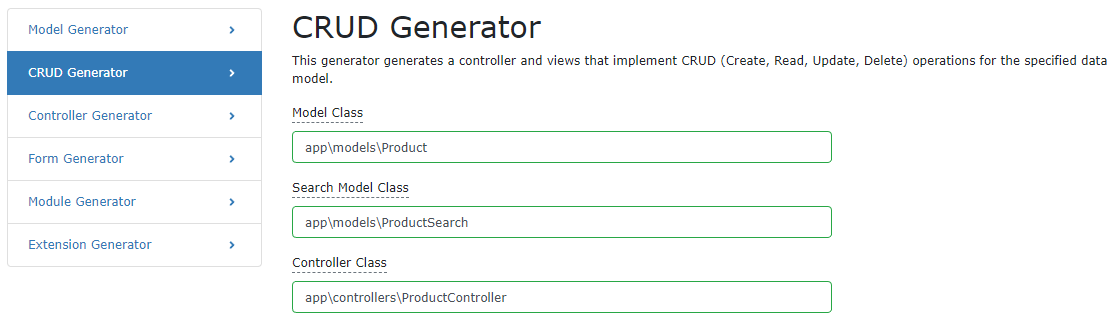


В режиме дизайнера установите связи между таблицами, тип связи – CASCADE.



Используя генератор Gii создайте модель, модель поиска, контроллер и представления.





Русифицируйте представления и метки в модели (метод attributeLabels()).

В модели models\Product.php добавьте правило валидации для загрузки изображений.

*/\* Правила валидации данных \*/*public function rules()  
{  
 return [  
 [['name', 'price', 'count', 'photo', 'model', 'country', 'year', 'id\_category'], 'required'],  
 [['price'], 'number'],  
 [['count', 'id\_category'], 'integer'],  
 [['year'], 'safe'],  
 [['name', 'model', 'country'], 'string', 'max' => 255],  
 [['id\_category'], 'exist', 'skipOnError' => true, 'targetClass' => Category::class, 'targetAttribute' => ['id\_category' => 'id']],  
 *//Добавляем правило валидации для загрузки изображений* [['photo'], 'file', 'skipOnEmpty' => false, 'extensions' => 'png, jpg', 'message' => 'Разрешены форматы PNG, JPG'],  
 ];  
}

Далее в модели добавьте метод для загрузки изображений.

*/\* Метод загрузки изображений в директорию uploads \*/*public function upload()  
{  
 if ($this->validate()) {  
 $fileName = 'uploads/' . $this->photo->baseName . '.' . $this->photo->extension;  
 $this->photo->saveAs($fileName);  
 return true;  
 } else {  
 return false;  
 }  
}

Создайте папку uploads в директории web.

Разрешите создание (create), обновление (update) и удаление (delete) товаров только администраторам, для этого в контроллере controllers\ProductController.php измените метод behaviors().

class ProductController extends Controller  
{  
 */\* Поведение. Доступ только для админа \*/* public function behaviors()  
 {  
 return [  
 'access' => [  
 'class' => AccessControl::class,  
 'only' => ['create', 'update', 'delete'],  
 'rules' => [  
 [  
 'actions' => ['create', 'update', 'delete'],  
 'allow' => true,  
 'roles' => ['@'],  
 'matchCallback' => function($rule, $action){  
 return \Yii::*$app*->user->identity->isAdmin();  
 }  
 ],  
 ],  
 ],  
 ];  
 }

Далее в контроллере измените метод actionCreate(), который перед созданием товара будет загружать изображение на сервер.

public function actionCreate()  
{  
 $model = new Product();  
  
 if ($this->request->isPost) {  
 if ($model->load($this->request->post())) {  
 $model->photo = UploadedFile::*getInstance*($model, 'photo');  
 if ($model->upload() && $model->save(false)) {  
 return $this->redirect(['index']);  
 }  
 }  
 } else {  
 $model->loadDefaultValues();  
 }  
  
 return $this->render('create', [  
 'model' => $model,  
 ]);  
}

Аналогично измените метод actionUpdate(), который позволяет редактировать данные.

public function actionUpdate($id)  
{  
 $model = $this->findModel($id);  
  
 *//Загрузка файла на сервер перед обновлением товара* if ($this->request->isPost) {  
 if ($model->load($this->request->post())) {  
 $model->photo = UploadedFile::*getInstance*($model, 'photo');  
 if ($model->upload() && $model->save(false)) {  
 return $this->redirect(['index']);  
 }  
 }  
 } else {  
 $model->loadDefaultValues();  
 }  
  
 return $this->render('update', [  
 'model' => $model,  
 ]);  
}

Измените представление view\product\\_form.php , добавив возможность выбора категории из списка, изменив поле для ввода id\_category, а также изменив тип поля для изображения товара.

<?php $form = ActiveForm::*begin*(); ?>  
  
<?= $form->field($model, 'name')->textInput(['maxlength' => true]) ?>  
  
<?= $form->field($model, 'price')->textInput(['maxlength' => true]) ?>  
  
<?= $form->field($model, 'count')->textInput() ?>  
  
<?= $form->field($model, 'photo')->fileInput(['maxlength' => true]) ?>  
  
<?= $form->field($model, 'model')->textInput(['maxlength' => true]) ?>  
  
<?= $form->field($model, 'country')->textInput(['maxlength' => true]) ?>  
  
<?= $form->field($model, 'year')->textInput() ?>  
  
<?= $form->field($model, 'id\_category')  
 ->dropDownList(\yii\helpers\ArrayHelper::*map*(\app\models\Category::*find*()  
 ->all(), 'id', 'name')) ?>  
  
<div class="form-group">  
 <?= Html::*submitButton*('Сохранить', ['class' => 'btn btn-success']) ?>  
</div>  
  
<?php ActiveForm::*end*(); ?>

Используйте Админ-панель и добавьте 6 товаров с их описанием.

Измените виджет GridView в представление view\product\index.php .

<div class="product-index">  
  
 <h1><?= Html::*encode*($this->title) ?></h1>  
 *<!-- Кнопка Добавить товар только для админа-->* <p>  
 <?php  
 if (!Yii::*$app*->user->isGuest && Yii::*$app*->user->identity->isAdmin()){  
 echo Html::*a*('Добавить товар', ['create'], ['class' => 'btn btn-success']);  
 } ?>  
 </p>  
  
 <?php *// echo $this->render('\_search', ['model' => $searchModel]);* ?>  
 <?= GridView::*widget*([  
 'dataProvider' => $dataProvider,  
 'filterModel' => $searchModel,  
 'columns' => [  
 ['class' => 'yii\grid\SerialColumn'],  
 'name',  
 'price',  
 *// Поле для вывода фото* [  
 'label' => 'Фото товара',  
 'format' => 'html',  
 'value' => function($model) {  
 return Html::*img*('http://' . $\_SERVER["SERVER\_NAME"] . '/uploads/' . $model->photo, ['width' => 250, 'alt' => $model->name, 'title' => $model->name]);  
 }  
 ],  
 *// Поле для вывода кнопок управления для админа* [  
 'class' => ActionColumn::*className*(),  
 'urlCreator' => function ($action, Product $model, $key, $index, $column) {  
 return Url::*toRoute*([$action, 'id' => $model->id]);  
 },  
 'visible' => !Yii::*$app*->user->isGuest && Yii::*$app*->user->identity->isAdmin()  
 ],  
 *// Поле для вывода кнопки Добавить в корзину для пользователя* [  
 'label' => 'Добавить в корзину',  
 'format' => 'raw',  
 'value' => function($model) {  
 return Html::*button*('Добавить в корзину', ['class' => 'btn btn-success', 'onclick' => 'addCart('.$model->id.')']);  
 },  
 'visible' => !Yii::*$app*->user->isGuest && !Yii::*$app*->user->identity->isAdmin()  
 ],  
 *// Поле для вывода кнопки Пoдробнее для всех* [  
 'label' => 'Подробнее',  
 'format' => 'raw',  
 'value' => function($model) {  
 return Html::*a*('Пoдробнее', ['view', 'id' => $model->id], ['class' => 'btn btn-success']);  
 }  
 ],  
 *//'id',  
 //'count',  
 //'model',  
 //'country',  
 //'year',  
 //'id\_category',* ],  
 ]); ?>  
  
</div>

Измените виджет GridView в представление view\product\view.php .

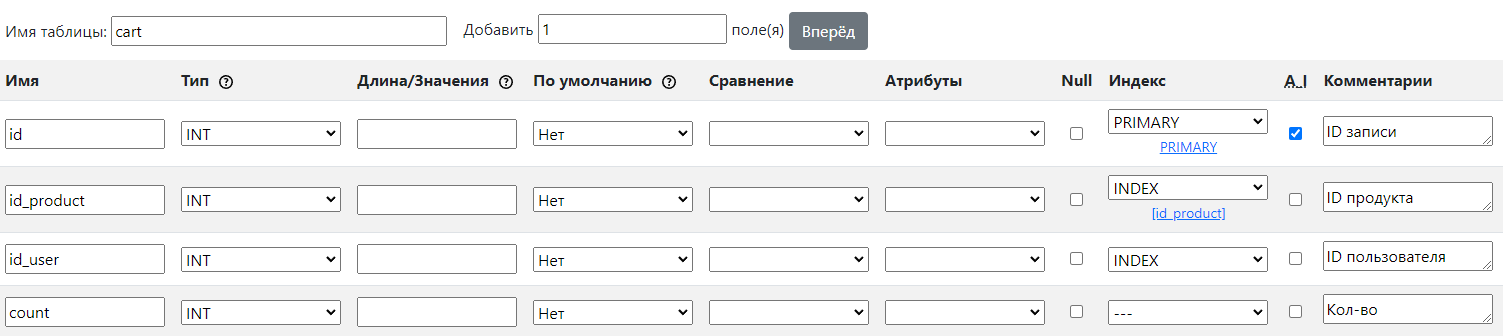
<?= DetailView::*widget*([  
 'model' => $model,  
 'attributes' => [  
 'id',  
 'name',  
 'price',  
 'count',  
 'photo',  
 'model',  
 'country',  
 'year',  
 *//'id\_category',* [  
 'label' => 'Категория товаров',  
 'format' => 'raw',  
 'value' => function($model) {  
 return $model->category->name;  
 },  
 ],  
 *// Поле для вывода фото* [  
 'label' => 'Фото товара',  
 'format' => 'html',  
 'value' => function($model) {  
 return Html::*img*('http://' . $\_SERVER["SERVER\_NAME"] . '/uploads/' . $model->photo, ['width' => 250, 'alt' => $model->name, 'title' => $model->name]);  
 }  
 ],  
 *// Поле для вывода кнопки Добавить в корзину для пользователя* [  
 'label' => 'Добавить в корзину',  
 'format' => 'raw',  
 'value' => function($model) {  
 return Html::*button*('Добавить в корзину', ['class' => 'btn btn-success', 'onclick' => 'addCart('.$model->id.')']);  
 },  
 'visible' => !Yii::*$app*->user->isGuest && !Yii::*$app*->user->identity->isAdmin()  
 ],  
 ],  
]) ?>

В модели models\ProductSearch.php измените запрос на поиск товаров, чтобы он выводил только товары имеющиеся в наличие и сортировал их от новых к старым.

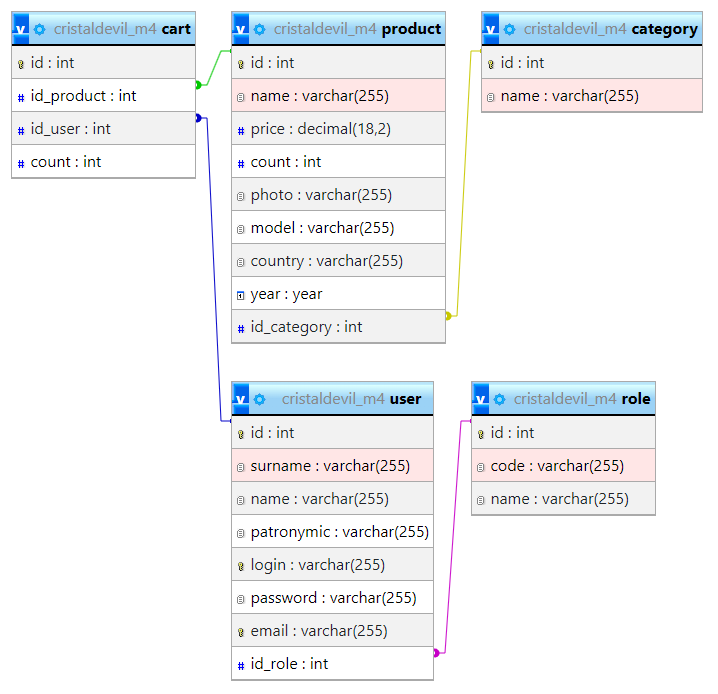
public function search($params)  
{  
 $query = Product::*find*()->where(['>', 'count', 0])->orderBy(['id' => *SORT\_DESC*]);

## Корзина

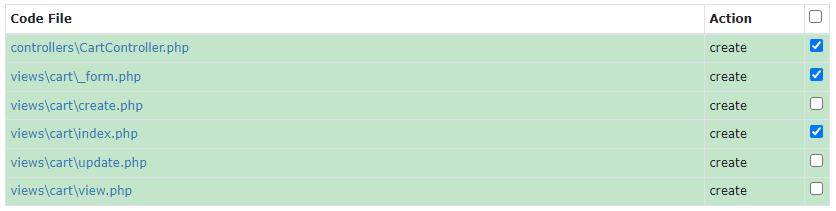
В phpMyAdmin создайте таблицу cart.



В режиме дизайнера установите связи между таблицами, тип связи – CASCADE.



Используя генератор Gii создайте модель, контроллер и представления (только форму и index).



Разрешите просмотр корзины только авторизированным пользователям, для этого в контроллере controllers\CartController.php измените метод behaviors().

class CartController extends Controller  
{  
 */\* Поведение. Доступ только для админа \*/* public function behaviors()  
 {  
 return [  
 'access' => [  
 'class' => AccessControl::class,  
 'only' => ['index'],  
 'rules' => [  
 [  
 'actions' => ['index'],  
 'allow' => true,  
 'roles' => ['@'],  
 'matchCallback' => function($rule, $action){  
 return \!Yii::*$app*->user->isGuest && \!Yii::*$app*->user->identity->isAdmin();

}  
 ],  
 ],  
 ],  
 ];  
 }

В методе actionIndex() измените параметр поиска данных в таблице cart, чтобы выводились данные только текущего пользователя.

public function actionIndex()  
{  
 $dataProvider = new ActiveDataProvider([  
 'query' => Cart::*find*()->where(['id\_user' => \Yii::*$app*->user->getId()]),

Измените представление view\product\\_form.php . Виджет Pjax необходим для обновления блоков без перезагрузки страницы, после выполнения ajax запросов (изменение количества заказанного товара, удаление товара из корзины). Добавьте кнопки + и – для добавления и уменьшения (удаления) товара в корзине. Ниже добавьте кнопку оформить заказ и поле для ввода пароля пользователя, который необходимо ввести прежде чем оформить заказ.

*<!-- Pjax для обновления данных без перезагрузки из JS скрипта -->*<?php \yii\widgets\Pjax::*begin*(['id' => 'cart']);?>  
  
<div class="cart-index">  
  
 <h1><?= Html::*encode*($this->title) ?></h1>  
  
 <?= GridView::*widget*([  
 'dataProvider' => $dataProvider,  
 'columns' => [  
 ['class' => 'yii\grid\SerialColumn'],  
 [  
 'label' => 'Наименование товара',  
 'format' => 'html',  
 'value' => function($model) {  
 return $model->product->name;  
 }  
 ],  
 [  
 'label' => 'Фото товара',  
 'format' => 'html',  
 'value' => function($model) {  
 return Html::*img*('http://' . $\_SERVER["SERVER\_NAME"] . '/uploads/' . $model->product->photo,  
 ['width' => 250, 'alt' => $model->product->name, 'title' => $model->product->name]);  
 }  
 ],  
 'count',  
 [  
 'label' => 'Добавить',  
 'format' => 'raw',  
 'value' => function($model){  
 return '<button onclick="addCart('.$model->product->id.');" class="btn btn-success"> + </button>';  
 },  
 ],  
 [  
 'label' => 'Удалить',  
 'format' => 'raw',  
 'value' => function($model){  
 return '<button onclick="removeCart('.$model->product->id.');" class="btn btn-success"> - </button>';  
 },  
 ]  
 ],  
 ]); ?>  
  
 <?= Html::*input*('password','password', '',  
 ['class' => 'form-control password mt-2', 'placeholder' => 'Введите ваш пароль']); ?>  
 <?= Html::*button*('Оформить заказ',  
 ['class' => 'form-control btn btn-success mt-2', 'onclick' => 'byOrder()']); ?>  
  
</div>  
  
<?php \yii\widgets\Pjax::*end*()?>

Добавьте в файл web\js\my.js описание функций addCart() и removeCart(), которые вызываются при нажатии на кнопки 'Добавить' и 'Удалить'.

*//Функция добавления/увеличения товара в корзину*function *addCart*(id) {  
 $.ajax({  
 method: "GET",  
 url: `/cart/create?id\_product=${id}`  
 }).done(function(message){  
 $.pjax.reload(  
 {  
 container:"#cart",  
 });  
 $('#info-message').html(message).fadeIn().animate({opacity: 1.0}, 3000).fadeOut("slow");  
 });  
}  
  
*//Функция удаления/уменьшения товара в корзине*function *removeCart*(id) {  
 $.ajax({  
 method: "GET",  
 url: `/cart/delete?id\_product=${id}`  
 }).done(function(message){  
 $.pjax.reload(  
 {  
 container:"#cart",  
 });  
 $('#info-message').html(message).fadeIn().animate({opacity: 1.0}, 3000).fadeOut("slow");  
 });  
}

В контроллере controllers\CartController.php измените методы actionCreate() отвечающий за добавление/увеличение товара в корзине и actionDelete() отвечающий за удаление/уменьшение товара из корзины. В методе actionCreate() предусмотрите возможность проверки количества заказываемого товара (нельзя заказать больше, чем есть в наличии).

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(['id\_product' => $id\_product])->andWhere(['id\_user' => \Yii::*$app*->user->getId()])->one();  
  
 if (!$itemInCart){  
 $itemInCart = new Cart([  
 'id\_product' => $id\_product,  
 'id\_user' => \Yii::*$app*->user->getId(),  
 'count' => 1  
 ]);  
 $itemInCart->save();  
 return "Продукт добавлен. Количество товара в корзине = $itemInCart->count";  
 } else {  
 if ($itemInCart->count + 1 > $product->count){  
 return "Нельзя больше добавить";  
 } else {  
 $itemInCart->count++;  
 $itemInCart->save();  
 return "Продукт добавлен. Количество товара в корзине = $itemInCart->count";  
 }  
 }  
}

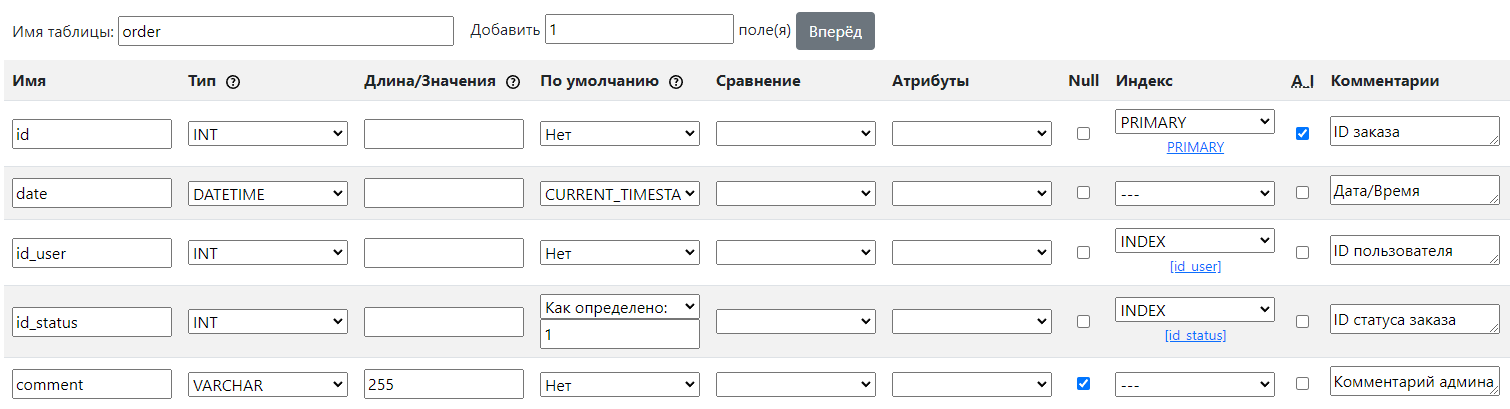
public function actionDelete($id\_product)  
{  
 $itemInCart = Cart::*find*()->where(['id\_product' => $id\_product])->andWhere(['id\_user' => \Yii::*$app*->user->getId()])->one();  
  
 if (!$itemInCart) {  
 return "Товар не найден";  
 }  
  
 if ($itemInCart->count - 1 == 0){  
 $itemInCart->delete();  
 return "Товар удален";  
 }  
  
 $itemInCart->count--;  
 $itemInCart->save();  
 return "Товар уменьшено. Количество товара в корзине = $itemInCart->count";  
}

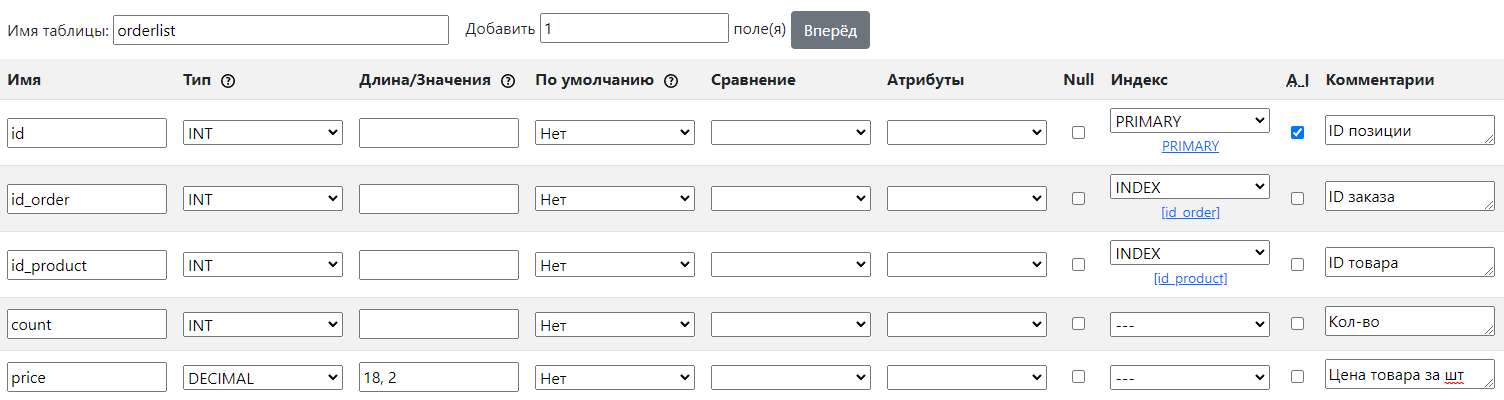
В каскадных таблицах стилей web\css\my.css добавьте правило описания блока с информацией info-message .

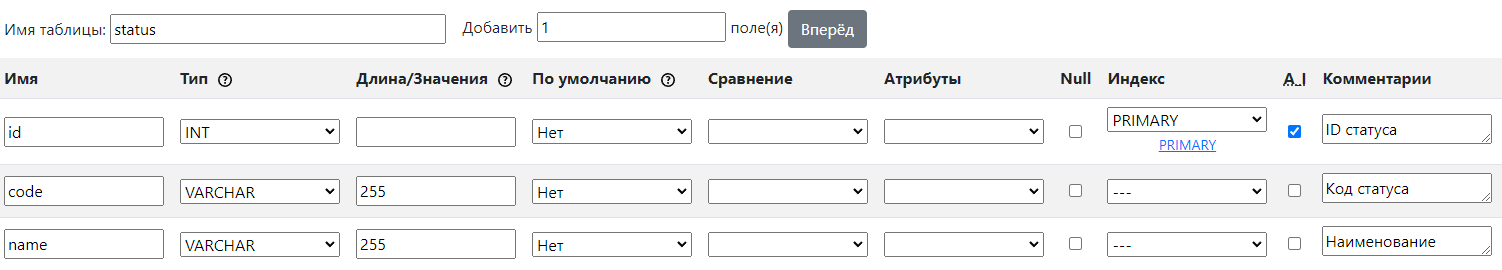
#info-message {  
 position: fixed;  
 top: 75px;  
 right: 50px;  
 background-color: #198754;  
 padding: 20px;  
 color: white;  
 opacity: 0;  
}

## Оформление заказа

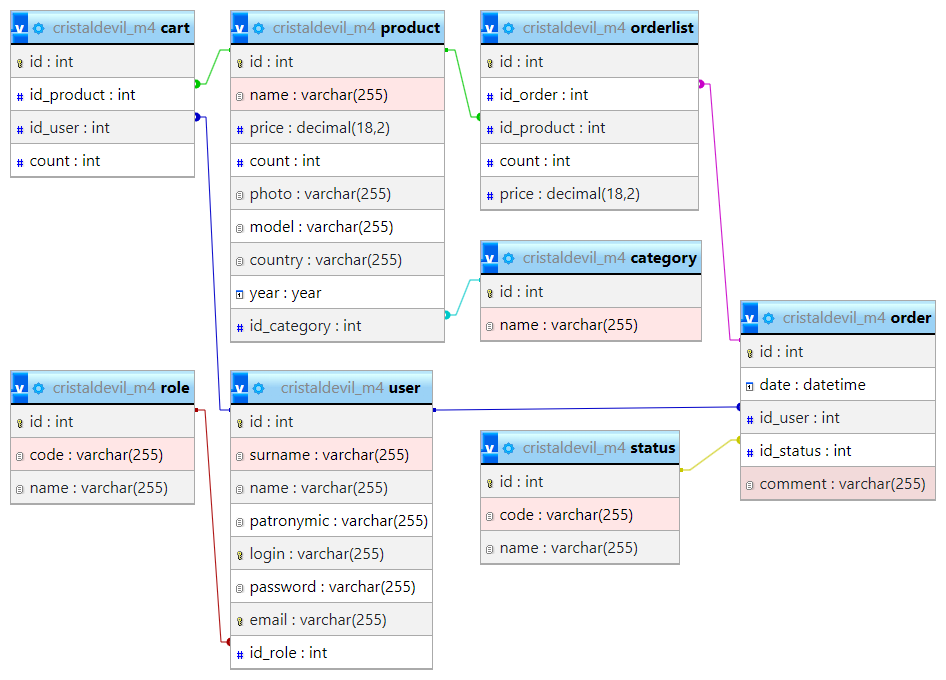
В phpMyAdmin создайте таблицу status, order и orderlist.







В режиме дизайнера установите связи между таблицами, тип связи – CASCADE.



В таблицу role добавьте две записи.

|  |  |  |
| --- | --- | --- |
| **id** | **code** | **name** |
| 1 | new | Новый |
| 2 | confirm | Подтвержденный |
| 3 | reject | Отклоненный |