Титульный лист к отчету о расчетно-графической работе

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Пензенский государственный университет архитектуры и строительства»

Кафедра «Информационно-вычислительные системы»

ОТЧЕТ

о расчетно-графической работе по дисциплине

«Web-дизайн, проектирование и разработка сайтов»

Автор работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Куприянко Г.М. гр. ИСТ-22

(подпись)

Руководитель работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ к.т.н., доцент В.В. Кузина

(подпись)

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Пенза 2018

**Содержание**

[Задание 3](#_Toc515855482)

[Теоретическая часть 4](#_Toc515855483)

[Основы 4](#_Toc515855484)

[Emmet – плагин для быстрой и качественной вёрстки 6](#_Toc515855485)

[Bootstrap 7](#_Toc515855486)

[Less – надстройка над CSS 8](#_Toc515855487)

[Java Script 8](#_Toc515855488)

[jQuery 9](#_Toc515855489)

[MySql 9](#_Toc515855490)

[PHP 10](#_Toc515855491)

[Доменное имя 10](#_Toc515855492)

[Хостинг 11](#_Toc515855493)

[Система управления контентом - CMS 11](#_Toc515855494)

[FTP – удобная работа с файлами прямо на сервере 12](#_Toc515855495)

[Подготовка среды разработки 13](#_Toc515855496)

[Доменное имя 13](#_Toc515855497)

[Хостинг 13](#_Toc515855498)

[CMS – WordPress 13](#_Toc515855499)

[IDE – Php Storm 13](#_Toc515855500)

[FTP 14](#_Toc515855501)

[MySql 15](#_Toc515855502)

[Практическая часть 16](#_Toc515855503)

[Плагины WordPress 16](#_Toc515855504)

[Создание собственной темы оформления. Часть 1. Подключение основных файлов. 16](#_Toc515855505)

[Создание собственной темы оформления. Часть 2. Базовая структура сайта. 20](#_Toc515855506)

[Создание собственной темы оформления. Часть 3. Создание прелоадера. 21](#_Toc515855507)

[Создание собственной темы оформления. Часть 4. Обновлённые футер и хедер. 24](#_Toc515855508)

[Создание собственной темы оформления. Часть 5. Вывод контента страницы. Управление меню. 27](#_Toc515855509)

[Создание собственной темы оформления. Часть 6. Новый тип записи – слайды. 28](#_Toc515855510)

[Создание собственной темы оформления. Часть 7. Шаблоны страниц. Обо мне. 31](#_Toc515855511)

[Создание собственной темы оформления. Часть 8. Шаблоны страниц. Навыки. 35](#_Toc515855512)

[Создание собственной темы оформления. Часть 9. Шаблоны страниц. Игры. 36](#_Toc515855513)

[Создание собственной темы оформления. Часть 10. Шаблоны страниц. Контакты. 38](#_Toc515855514)

[Создание собственной темы оформления. Часть 11. Новости. 41](#_Toc515855515)

[Вывод 47](#_Toc515855516)

[Список использованной литературы 48](#_Toc515855517)

# Задание

Разработать Web-сайт на тему «портфолио» с использованием технологий CSS, JS и PHP. Связать Web-страницы с базой данных.

# Теоретическая часть

## Основы

Веб-разработка — процесс создания веб-сайта или веб-приложения. Основными этапами процесса являются веб-дизайн, вёрстка страниц, программирование для веб на стороне клиента и сервера, а также конфигурирование веб-сервера.

Веб-дизайн — отрасль веб-разработки и разновидность дизайна, в задачи которой входит проектирование пользовательских веб-интерфейсов для сайтов или веб-приложений.

Веб-дизайнеры:

• проектируют логическую структуру веб-страниц;

• продумывают наиболее удобные решения подачи информации;

• занимаются художественным оформлением веб-проекта.

В результате пересечения двух отраслей человеческой деятельности грамотный веб-дизайнер должен быть знаком с последними веб-технологиями и обладать соответствующими художественными качествами. Большая часть специалистов, работающих в области дизайна, обычно концентрирует в себе такое творческое образование, как студия дизайна.

Конечным продуктом работы веб-дизайнера является дизайн-макет: картинка, представляющая предполагаемый будущий внешний вид страниц сайта. Картинка эта является многослойной, где, на усмотрение дизайнера, почти каждая деталь — отдельный слой, приложенный к другим слоям-картинкам, за счёт чего может легко выполняться доработка, замена, перекомпоновка и другие задачи.

Вёрстка веб-страниц — создание структуры html-кода, размещающего элементы веб-страницы (изображения, текст и т. д.) в окне браузера, согласно разработанному макету, таким образом, чтобы элементы дизайна выглядели аналогично макету.

Вёрстка веб-страниц отличается от полиграфической тем, что необходимо учитывать разницу отображения элементов в различных браузерах и разницу в размерах рабочего пространства устройств.

Процесс сложен и имеет творческую основу, ни один из способов не является каноничным и принятым как основа. Все подходы к вёрстке имеют как преимущества, так и недостатки. Модульность и виды подходов к вёрстке

**Табличный подход**

Таблицы являются удобным инструментом для выравнивания блоков. В html таблица делится на строки, которые в свою очередь делятся на ячейки, таким образом можно создавать неограниченное количество строк с неограниченным числом ячеек в каждой. Ячейки таблицы имеют особые настройки выравнивания, которые не встречаются более в других элементах. Это значит, что некоторые элементы дизайна крайне сложно сверстать без использования таблиц.

В деловом мире таблицы используется для сравнения данных, а также для наглядности.

Недостатком табличного подхода является то, что страница не будет отображена до тех пор, пока не будет загружен закрывающий тег таблицы, что является критичным при обрыве связи и медленном соединении. Также к недостаткам можно отнести избыток кода, усложняющий его понимание.

**Вёрстка слоями**

Слой (от имени тега <layer>) — разработка компании Netscape, использованная в браузере Netscape Navigator. Тег позволял показывать или скрывать его содержимое, устанавливать положение относительно окна браузера, накладывать слои друг на друга и включать содержимое блока из файла. В спецификацию HTML не был включён.

**Блочная вёрстка**

Блочная вёрстка — верстка с помощью тега <div> и описывающего их множество таблицы стилей (CSS). Тег появился вследствие сознательного отказа консорциума «CSS Positioning (CSS-P)» от тега <layer>, чтобы создать аналогичный тег, но с поддержкой CSS в духе остальных тегов. С помощью блочной верстки реализуют концепцию семантической вёрстки.

**Вёрстка фреймами**

Вёрстка фреймами — верстка с помощью одноимённого тега <frame>, который имеет иной способ позиционирования, а внутри себя включает отдельные html-страницы.

Фреймы просты в применении, поскольку необходимо заботиться только об отображении и перерисовке части окна браузера — одного фрейма, — остальные части находятся в другом фрейме и не перерисовываются без взаимодействия с ними. Также возможно открытие ссылки в другом фрейме, что удобно при использовании главного меню в отдельном фрейме. Страницы, поделённые на фреймы, имеют меньше кода, вследствие отсутствия повторяемых частей, которые не перезагружаются. Они размещаются точно в нужном месте окна браузера.

CSS (англ. Cascading Style Sheets — каскадные таблицы стилей) — формальный язык описания внешнего вида документа, написанного с использованием языка разметки.

Преимущественно используется как средство описания, оформления внешнего вида веб-страниц, написанных с помощью языков разметки HTML и XHTML, но может также применяться к любым XML-документам, например, к SVG или XUL.

CSS используется создателями веб-страниц для задания цветов, шрифтов, расположения отдельных блоков и других аспектов представления внешнего вида этих веб-страниц.

Основной целью разработки CSS являлось разделение описания логической структуры веб-страницы (которое производится с помощью HTML или других языков разметки) от описания внешнего вида этой веб-страницы (которое теперь производится с помощью формального языка CSS). Такое разделение может увеличить доступность документа, предоставить большую гибкость и возможность управления его представлением, а также уменьшить сложность и повторяемость в структурном содержимом. Кроме того, CSS позволяет представлять один и тот же документ в различных стилях или методах вывода, таких как экранное представление, печатное представление, чтение голосом (специальным голосовым браузером или программой чтения с экрана), или при выводе устройствами, использующими шрифт Брайля.

## Emmet – плагин для быстрой и качественной вёрстки

Emmet — это не самостоятельный редактор, а инструмент для ускорения работы с HTML и CSS, который реализован в виде плагинов для соответсвующих редакторов. На сегодняшний день его можно подключить (использовать) со следующими приложениями: Sublime Text, PHPStorm и WebStorm, NetBeans,Eclipse, Aptana, Coda, TextMate, Komodo Edit, Notepad++, CodeMirror, Brackets, Adobe Dreamviewer и Ace.

Существует плагин достаточно давно, правда, ранее он носил название Zen Coding. В основе проекта лежит механизм динамических аббревиатур, которые разбираются «на лету» и из которых генерируется готовый фрагмент кода.

Этот способ ускоренного написания HTML и CSS кода разрабатывается Сергеем Чикуёнком, хотя первоначальная концепция принадлежала Вадиму Макееву, который в одной из своих статей изложил базовые принципы.

Для написания аббревиатур используется синтаксис, во многом похожий на CSS-селекторы.

Например:

1. знак «>» (признак дочернего или вложенного селектора);
2. знак «+» (признак соседнего селектора)
3. точка «.» (название класса)
4. октоторп или хэш-знак «#» (название идентификатора)
5. и целый ряд других.

Помимо перечисленных символов в Emmet используется еще целый ряд специфических знаков, которые позволяют упростить и ускорить написание динамических аббревиатур. К их числу можно отнести хотя бы знак знак доллара («$») (в Emmet он обозначает счетчик) или круглые скобки, позволяющие делать группировки.

## Bootstrap

Bootstrap — фреймворк, набор HTML+CSS инструментов и шаблонов для верстки и более эффективного и быстрого создания сайтов и веб-приложений более эффективно и быстро.

Бутстрап — современный помощник, разработчиков интерфейсов, дизайнеров и вебмастеров, доступный для использования по открытой лицензии. Обратите внимание что этот фреймворк очень динамичный и регулярно обновляемый, поэтому не все его функции могут корректно поддерживаться старыми браузерами. Бутстрап сейчас на пике популярности и можно уверенно сказать, что на его основе можно сделать любой веб-интерфейс.

В нем есть следующие шаблоны:

1. Шрифты
2. Кнопки
3. Формы
4. Метки
5. Навигация
6. Сетка
7. JavaScript-расширения
8. Прочее

**Основные преимущества Бутстрапа**:

1. Экономия времени - вы экономите усилия потому что используете уже готовые классы и дизайн. Направьте сэкономленную энергию и деньги на разработку дополнительной функциональности и вы только выиграете.
2. Адаптивность, высокая скорость и оптимизация, стандартизация интерфейсов — динамичные макеты Бутстрапа качественно отображаются на самых разных устройствах без необходимости внесения изменений в разметку.
3. Дизайн — единые шаблоны и стилевое оформление элементов макета и всех страниц на сайте в целом. И при этом Bootstrap кроссбраузерный и хорошо отображается во всех браузерах Safari, Firefox, IE, EDGE и тех, что на основе Chromium (движок Blink на основе Webkit: Яндекс.Браузер, Опера, Гугл Хром). Регулярное обновление и дополнение фреймворка самыми современными возможностями HTML и CSS вносит некоторые ограничения в использовании с IE7 и IE8.
4. Простота и открытость — использовать Бутстрап настолько просто, что с ним справляются даже школьники и начинающие веб-разработчики, а открытый исходный код позволяет самому участвовать в разработке, модифицировать под свои нужды или просто пользоваться хорошим бесплатным решением.
5. При этом код HTML, JavaScript и CSS в Бутстрапе продуман и рассмотрен под микроскопом сотнями разработчиков со всего мира — все для того, чтобы рядовые вебмастера и верстальщики могли легко и просто настроить сетку сайта или встроить необходимые элементы в интерфейс.
6. Также, в Bootstrap используется динамический язык стилей **LESS**, которые расширяет возможности CSS: разработчики могут управлять цветами, создавать вложенные колонки и переменные.

## Less – надстройка над CSS

LESS – это надстройка над CSS. Это значит, что любой CSS код – это валидный LESS, но дополнительные элементы LESS не будут работать в простом CSS. Это замечательно, потому что существующий CSS уже является работоспособным LESS кодом, что уменьшает порог вхождения в новую технологию.

LESS добавляет много нужных динамических свойств в CSS. Он вводит переменные, операции, function-like элементы и примеси. Возможность писать таблицы стилей модульно избавит вас от многих хлопот.

**Как использовать LESS**  
Есть два способа использования LESS. Вы можете создать LESS файл и конвертировать его при помощи Javascript на лету или скомпилировать его заранее и использовать получившийся CSS файл. Не бойтесь слова «компилировать» (я постоянно его использую), это очень просто.

## Java Script

JavaScript - это язык программирования, используемый при разработке сайтов.

Программы на JavaScript называют сценариями. Сценарии JavaScript встраиваются в HTML-код страницы и выполняются браузером. При помощи сценариев JavaScript странице можно придать динамических эффектов, таких как меняющиеся рисунки, бегающий текст и многое другое.

1. Использование JavaScript делает документы HTML динамическими. Прокручивающиеся слайды из нескольких картинок или динамический текст - это результат работы JavaScript.
2. Проверка данных, введённых пользователей в форму, до её отправки на сервер - это работа JavaScript.
3. Открытие новых окон без участия пользователя - это делается используя JavaScript.
4. Изменение содержимого окна браузера, в зависимости от действий пользователя - это тоже работа JavaScript.
5. Многие другие эффекты.

## jQuery

jQuery – это javascript библиотека, т.е. библиотека, написанная на языке JavaScript. Проще говоря, это некий набор готовых функций, которые упрощают жизнь разработчика и написание кода на JavaScript.

 jQuery хорош не только тем, что обеспечивает выборку любого элемента на странице, при этом выборку гибкую. Используя jQuery вы получаете еще ряд плюсов по сравнению с нативным JS. Вот некоторые из них:

1. простая работа с событиями;
2. кроссбраузерность;
3. удобная работа с AJAX (асинхронные запросы к серверу);
4. удобные методы для работы с эффектами (скрытие / появление элементов с добавлением визуальных эффектов);
5. тысячи готовых плагинов на любой вкус (галереи, слайдшоу, работа с формами и другими элементами страницы).

На сегодняшний день знание и работа с библиотекой jQuery – это фактически стандарт для любого веб-разработчика. Без нее не обходится практически ни один проект в сети, поскольку jQuery реально упрощает написание кода на JavaScript.

## MySql

Вопрос хранения информации с появлением баз данных преобразился, с появлением облачных идей видоизменился, но в конечном итоге разработчик имеет в своем распоряжении полнофункциональную базу и хранит информацию так, как ему кажется правильным, или так, как того требует проект.

MySQL имеет заслуженную репутацию. Это простая, эффективная и надежная система управления информацией, использующая традиционный язык SQL. В умелых руках MySQL - это реляционные идеи хранения и обработки информации в максимально широком спектре услуг, которые исторически сложились. Они отражают практическую целесообразность и представляют собой функциональность современной базы данных.

В большинстве случаев объективная закономерность выводит разработчика именно на PHP/MySQL. Эта пара стала де-факто лучшей в программировании сайтов и иных ресурсов для локальных и распределенных сетей обработки информации.

## PHP

PHP — это язык программирования, специально разработанный для написания web-приложений (сценариев). Существует два вида языков программирования — клиентский и серверный. PHP — это серверный язык.

Клиентскими называются языки, которые обрабатываются непосредственно в браузере вашего компьютера. С php несколько иначе. Браузер на вашем компьютере его не читает. PHP отсылается на сервер, обрабатывается на нём и браузеру возвращается готовый результат на одном из клиентских языков.

И увидеть код php на мониторе невозможно. Допустим, если Вы щёлкните правой клавишей мыши и откроете Просмотр кода страницы, то код первых трёх языков Вы увидите, а вот php нет. Вроде бы получается что для php нужно лишнее звено — лишняя работа. В чём смысл?

А смысл в том, что клиентские языки относительно статичны. Если программа, допустим, сайт, имеет множество страниц, то придётся прописывать полностью каждую, хотя на них всё кроме текста одинаковое или почти одинаковое. PHP же позволяет создавать динамические сайты.

Все одинаковые блоки сайта прописываются в отдельные файлы, т.е. код для них пишется только один раз, а уж при загрузке страницы, эти файлы собираются вместе и подгружаются к тексту, или наоборот, не важно, в результате чего на мониторе отображается целая страница. Код сайта сокращается в сотни раз.

Пропорционально числу страниц. Доля смысла есть и в конфедециальности и безопасности. PHP невозможно прочитать в браузере кликнув по «Просмотреть код страницы». А сервер обязан защищать информацию своих клиентов. То есть Ваш сайт, помимо защиты которую Вы сделаете сами, будет оберегать защита сервера.

Из всего вышесказанного следует, что для работы с php необходим сервер и его можно установить на компьютер, либо использовать сервер хостинга.

## Доменное имя

Для размещения сайта в интернете необходимы хостинг и домен. Существует две разные трактовки понятия домен: официальная и неофициальная. Чаще используют неофициальную: «домен или доменное имя – это фактический адрес сайта во всемирной сети интернет». Доменное имя включает в себя уникальное имя и доменную зону, которая пишется после имени и точки (в моем случае com). Домен отображается в адресной строке браузера при посещении web-сайта.

Почти все вебмастера и различные интернет-сервисы используют именно эту трактовку, хотя с технической точки зрения это не совсем верно, так как официально доменная зона (ru, com, net и т.д.) считается доменом первого уровня, а уникальное имя – второго уровня. Проще говоря, официально доменом считается часть адреса сайта, отделенная от другой части точкой. Полноценные доменные имена покупаются у так называемых регистраторов, а если быть точнее, то берутся в аренду, так как оплата за домен осуществляется за каждый год использования.

## Хостинг

Хостинг – это пакет услуг по предоставлению ресурсов для размещения web-сайта на специальном сервере и возможности постоянного отображения сайта в глобальной сети интернет. Организация, предоставляющая такой пакет услуг, называется хостинг-провайдер.

Web-сайт состоит из множества файлов, сгруппированных в различных папках. Для их размещения в интернете хостинг-провайдеры предоставляют web-серверы, на которых выделяется место для хранения сайта и установлено необходимое программное обеспечение для корректной работы и круглосуточного отображения сайта в сети.

Хостинг бывает нескольких видов, выбор того или иного пакета услуг зависит от задач, которые хостинг должен решать. Естественно, чем выше задачи, тем дороже услуги хостинг-провайдера.

Говоря простыми словами, хостинг – это жилище для сайта в интернете, которое может быть бюджетным или элитным, а домен – его адрес, по которому пользователи находят сайт в сети.

## Система управления контентом - CMS

CMS расшифровывается как “Content Management System”, что на русский переводится как “Система Управления Контентом”. Но гораздо понятнее и точнее звучит “Система Управления Сайтом”. То есть CMS — это движок, который позволяет управлять сайтом даже без навыков веб-программирования.

По сути, любая система управления сайтом похожа на Windows, Linux, MacOS и другие операционные системы. То есть она делает сайт понятным как для администратора сайта, так и для его посетителей.

Допустим, мы купили себе [домен](https://copywriting-rules.ru/chto-takoe-domen/). Если мы введем его в адресной строке, то браузер вернет ошибку, так как не будет знать, что и откуда загружать.

Если установить CMS на [хостинг](https://copywriting-rules.ru/chto-takoe-hosting/) и привязать ее к нашему домену, то при подключении к сайту загрузится тема “по умолчанию”, которая была создана при установке.

Если мы не будем устанавливать CMS, а просто создадим на хостинге файл index.html, то при подключении загрузится именно он и покажет то содержимое, что в файле. Если файл пустой, то браузер отобразит пустой экран.

## FTP – удобная работа с файлами прямо на сервере

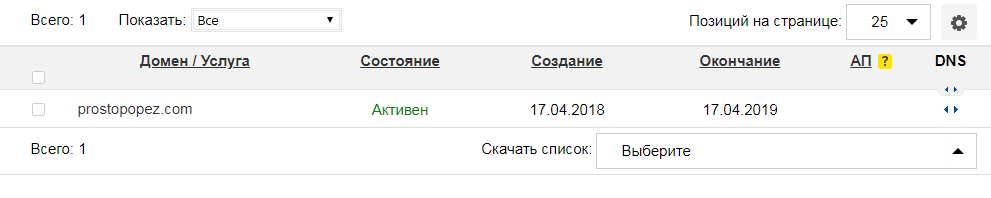
Можно работать с сайтом и не пользуясь FTP доступом, но с помощью него можно лего и просто сделать много вещей.

FTP – это протокол передачи файлов. Через него файлы можно скопировать с компъютера на сервер и обратно, отредактировать.

Для использования FTP необходима специальная программа или плагин. Их существует большое количество, некоторые из них бесплатные, некоторые могут быть коммерческими. Практически все программы FTP работают таким образом: появляются 2 окна, где 1-е – ваши файлы на сервере в интернет, а 2-е - файлы на компъютере, тем самым можно легко передавать файлы на компъютер/сервер.

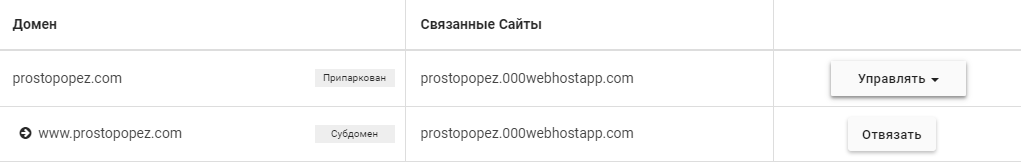
# Подготовка среды разработки

## Доменное имя

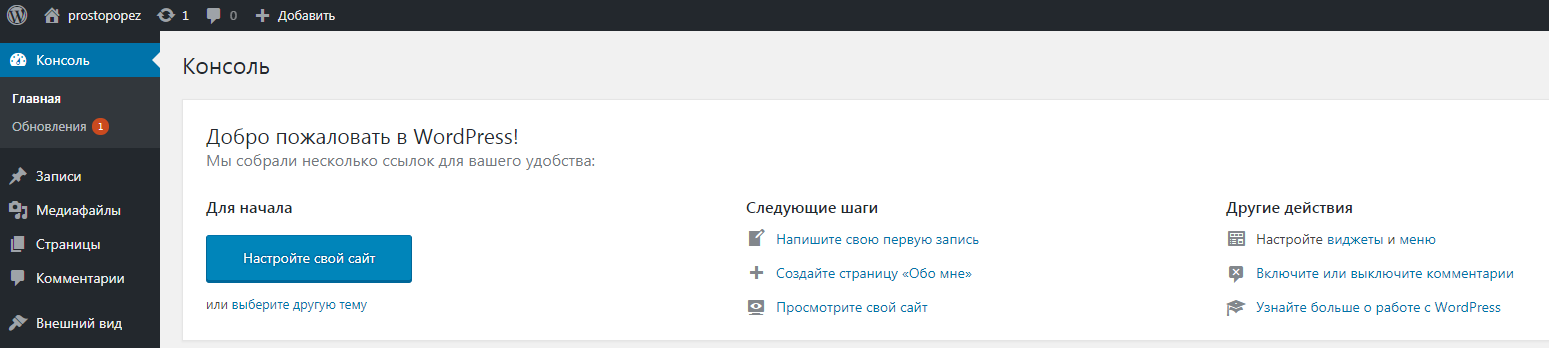
Доменное имя я решил купить на сайте <https://www.reg.ru>, потому что он показался мне наиболее подходящим для моих целей. 

## Хостинг

После покупки доменного имени мне предстояло выбрать хостинг. Так как ни расширенный функционал, ни большой трафик мне не понадобится, я решил обратиться к бесплатному хостингу - <https://www.000webhost.com/>

Зарегистрировавшись, я припарковал к своему сайту купленный ранее домен. 

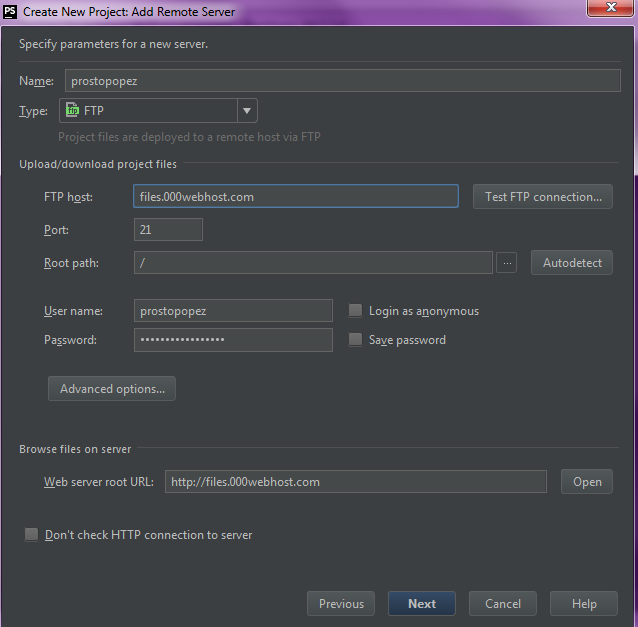
## CMS – WordPress

Далее мне предстояло установить на хостинг CMS WordPress, одновременно с этим подключая к сайту выделенную хостингом базу данных.

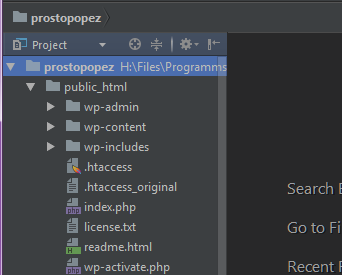
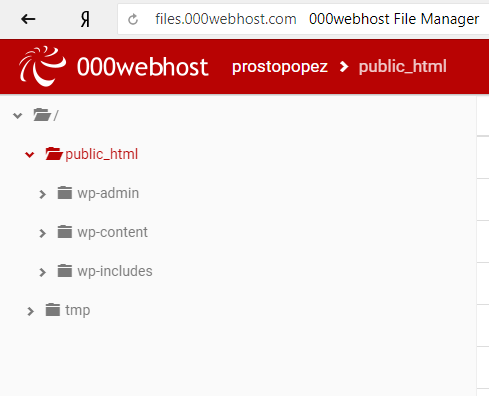
## IDE – Php Storm

Среди IDE для веб-разработки я предпочитаю [**Php Storm**](http://www.jetbrains.com/phpstorm/), так как она удобна, приятна на вид и ощупь, а также имеет большой функционал. В том числе в её функционал входит **плагин Emmet**, несколько плагинов для упрощённого комментирования и подключения FTP соединения.

## FTP

Так как в Php Storm имеется встроенный FTP менеджер

, мне оставалось только подключиться к файловой системе моего сайта, создав на компьютере аналогичный проект.

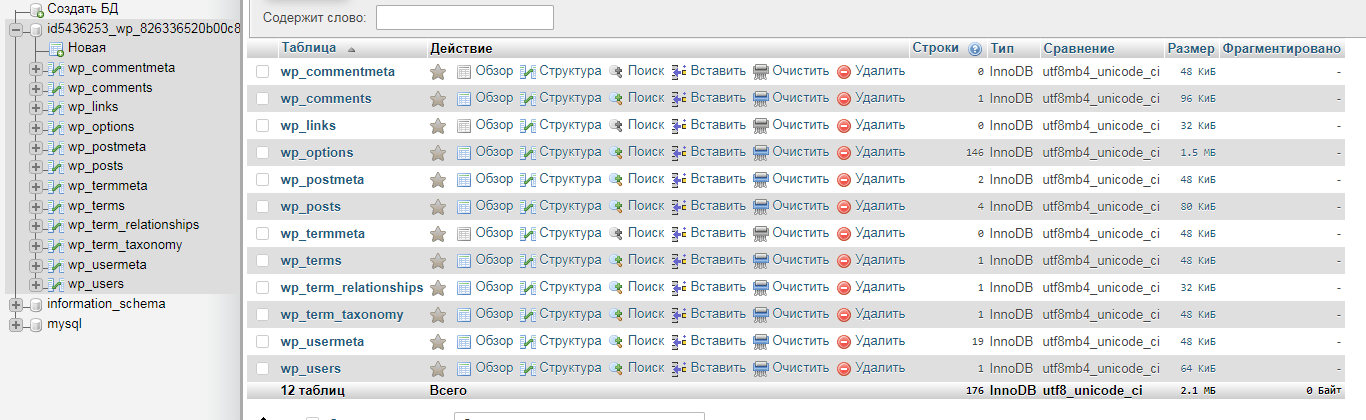


## MySql

WordPress по умолчанию создаёт себе базу данных, которая подключена ко всем необходимым элементам на сайте.

Каждая таблица содержит определённую хорошо структурированную информацию, манипулируя которой с помощью php, можно использовать базу данных максимально эффективно.

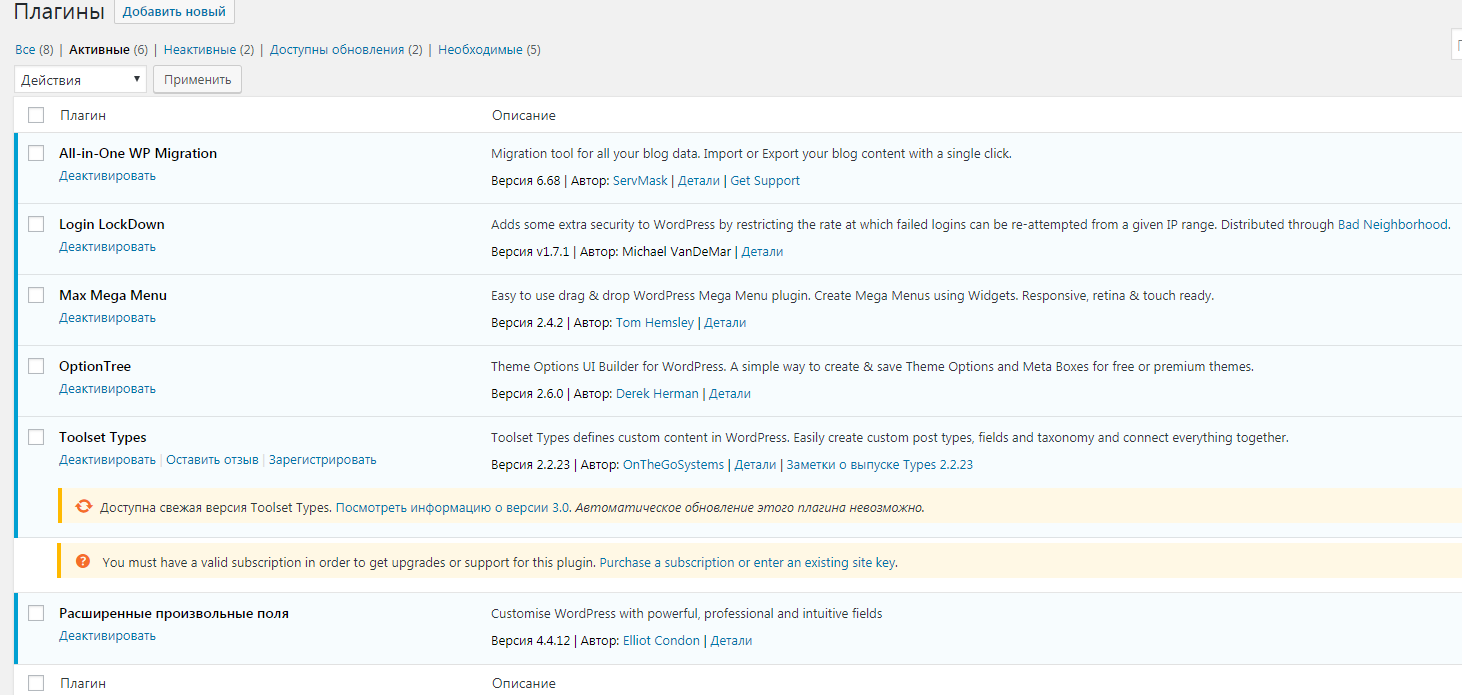
Редактировать её можно с помощью phpmyadmin, прямо как на локальном сервере.



# Практическая часть

## Плагины WordPress

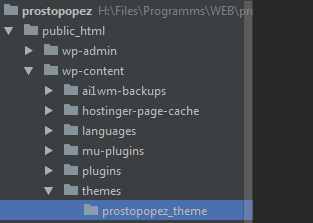
В самом начале желательно поставить все необходимые плагины для более комфортной работы с CMS. На данный момент для меня это такие плагины:



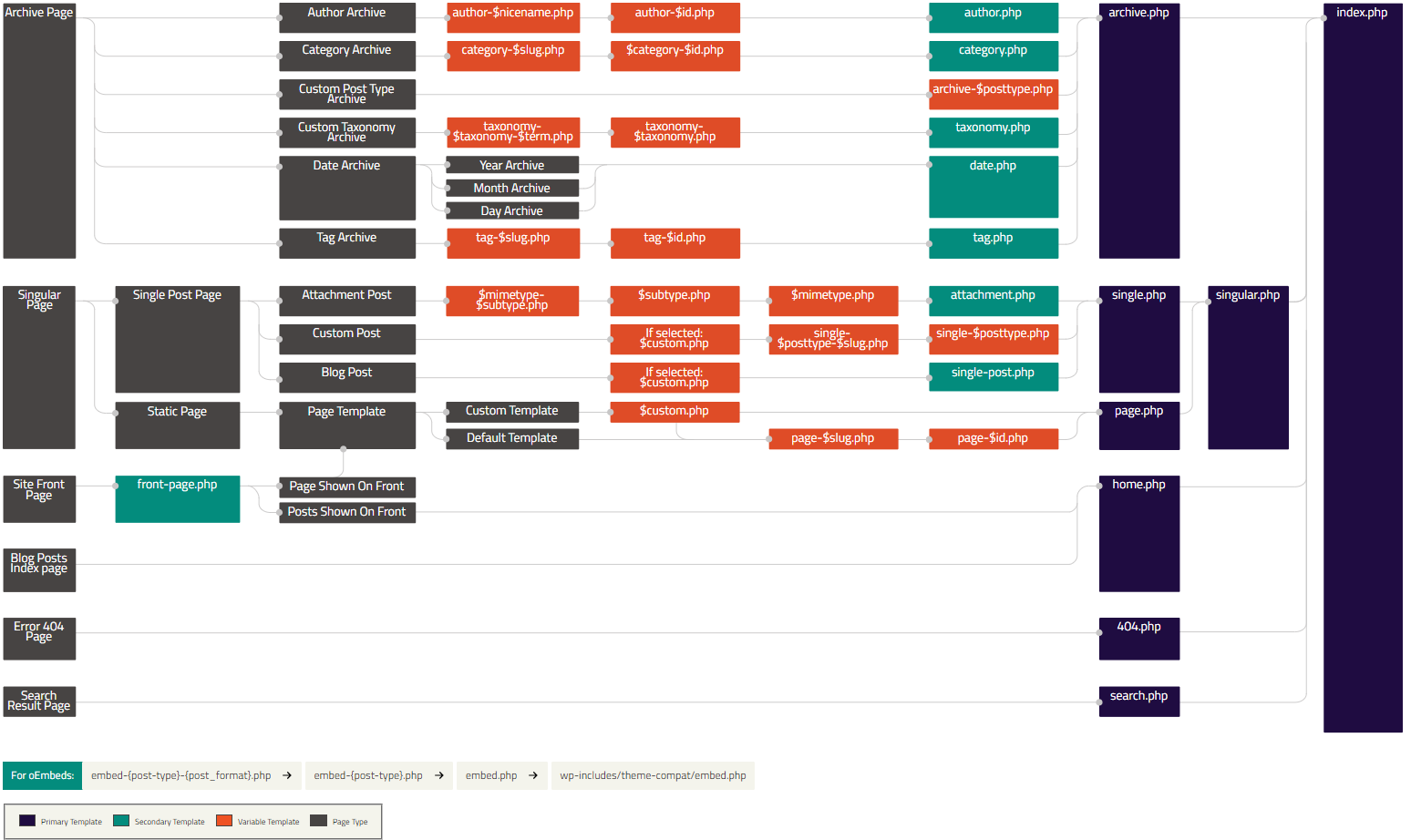
1. **All-in-One WP Migration** – по необходимости позволяет осуществлять импорт/экспорт сайта.
2. **Login LockDown** – защищает административную панель WordPress сайта от перебора пароля с помощью ограничения на кол-во ошибочных вводов пароля за определённое время.
3. **Max Mega Menu** – помогает создать хорошо структурированное меню для сайта.
4. **OptionTree –** помогает редактировать настройки отдельных разделов сайта.
5. **Toolset Types** – помогает создать пользовательские типы публикаций, помимо базовых в WordPress.
6. **Advanced Custom Fields** – позволяет добавлять произвольные типы полей при редактировании.

## Создание собственной темы оформления. Часть 1. Подключение основных файлов.

В разделе, предназначенном для хранения тем оформления, создаю новую папку.

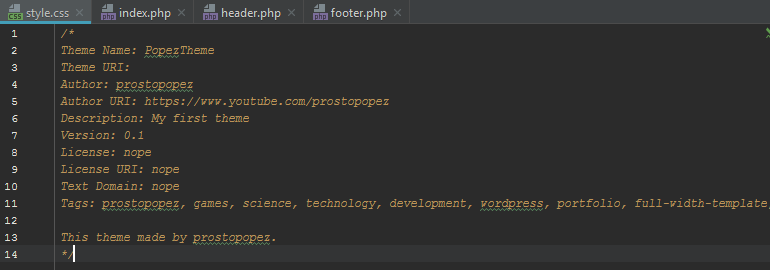


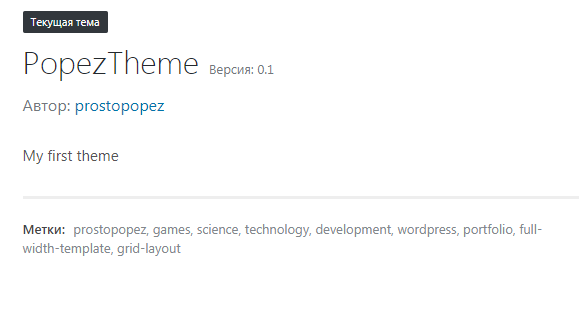
У WordPress есть базовая иерархия вызова файлов темы, которая может очень пригодиться:

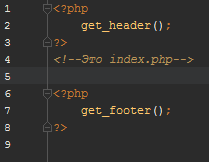


При обычной вёрстке вся страница находится в едином файле, но в WordPress есть возможность сократить вёрстку, вызывая повторяющиеся фрагменты с помощью функций. Так, сразу же создаём “header.php”, “footer.php” и “index.php”.

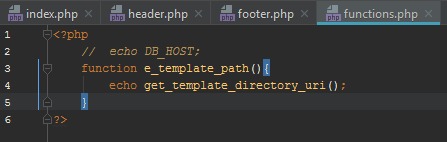
Прежде чем начинать работать с темой, необходимо сделать её видимой для административной панели WordPress. Создаётся файл style.css, в котором необходимо прописать основные строки, считываемые WordPress при поиске темы.





К файлу index.php с помощью функций подключаются header.php и footer.php. Таким образом, вёрстка нарезается на отдельные кусочки. 

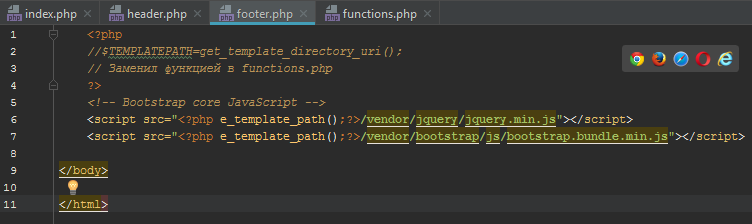
Так же создаю файл functions.php, в котором буду хранить функции, для которых необязательно создание плагина. К примеру, эта функция напишет путь к корневой папке моей темы. Далее это поможет упростить различные подключения.



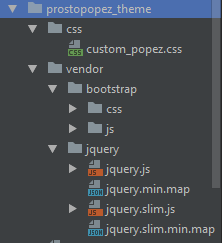
В header.php находится подключение стилей, мета-тегов, общего меню.



В footer.php – подключение скриптов и закрывание тегов body и html.



Таким образом, я создал основу страницы сайта и подключил стили и скрипты Bootstrap, а так же собственный файл стилей:



## Создание собственной темы оформления. Часть 2. Базовая структура сайта.

Самое время позаботиться о базовой структуре сайта.

Для начала продумаем основные компоненты:

1. Главная страница
2. Обо мне
3. Навыки
4. Игры
5. Контакты
6. Новости
7. Поиск

Здесь стоит упомянуть многоязычность. На данном этапе сайт будет исключителоьно русскоязычный, однако в дальнейшем я собираюсь добавить версию для англоязычных пользователей.

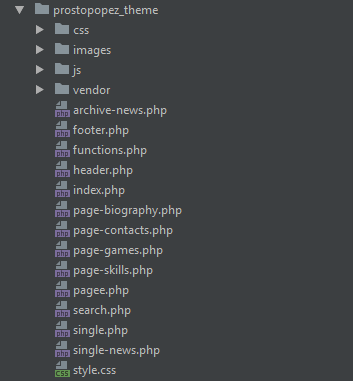
Зная будущий контент сайта, можно попытаться «набросать» структуру в виде php-файлов:

1. Главная страница – index.php
2. Обо мне – page-biography.php
3. Навыки – page-skills.php
4. Игры – page-games.php
5. Контакты – page-contacts.php
6. Новости
   1. Архив новостей – archive-news.php
   2. Страница новостей – single-news.php
7. Поиск – search.php

Отлично. Так же необходимы шаблоны страницы и записи, чтобы при расширении сайта они изначально не пустовали:

1. page.php
2. single.php

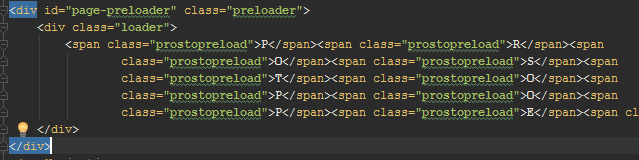
Хорошо, базовая структура сайта готова.



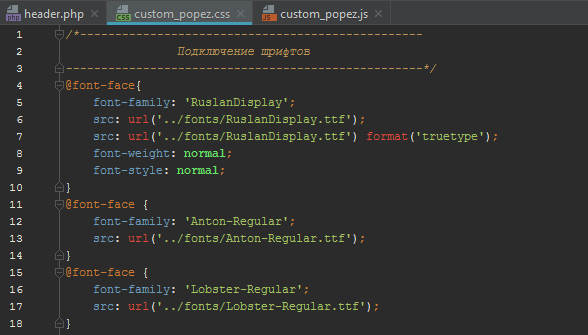
## Создание собственной темы оформления. Часть 3. Создание прелоадера.

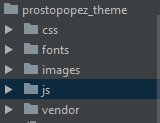
Думаю, прелоадер сайту не помешает. Он поможет избавить пользователя от наблюдения за загрузкой страницы по частям.

В файле header.php создаю контейнер для моего прелоадера:

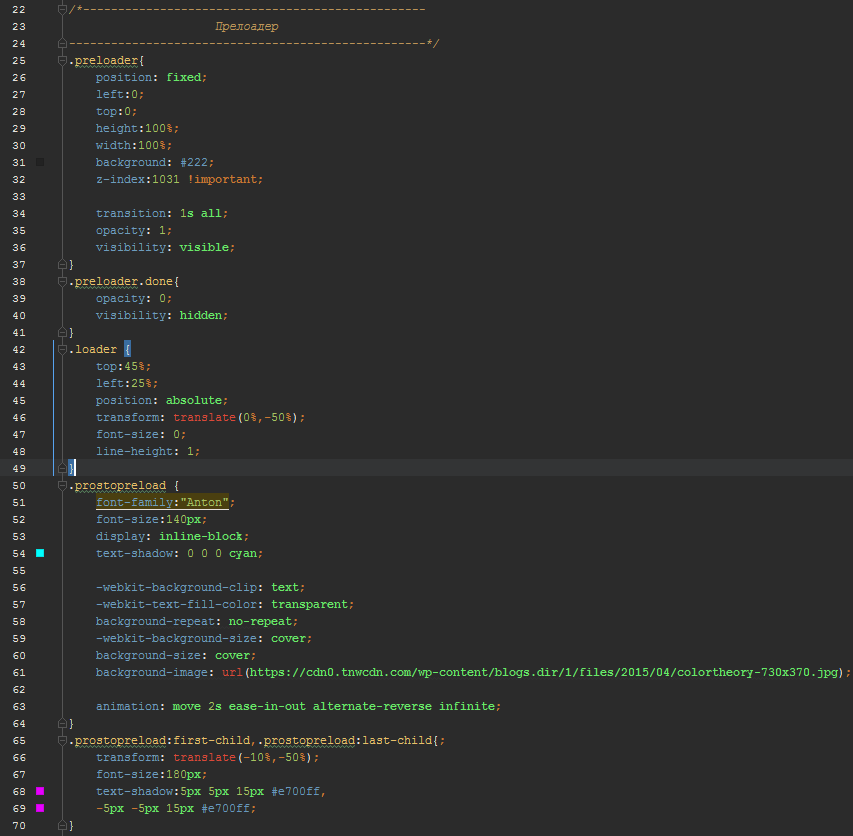


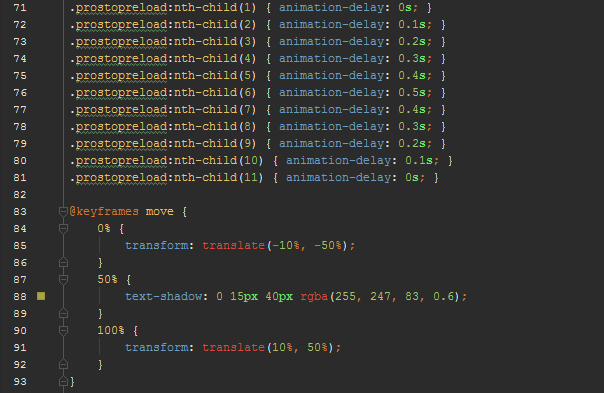
На этом же этапе создам раздел в стилях специально для подключения кастомных шрифтов, а так же папку в теме под эти шрифты.

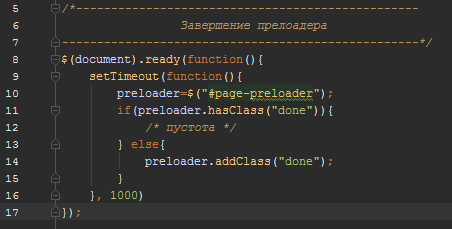




Прелоадером я решил сделать свой анимированный никнейм:





Для корректной работы прелоадера в кастомном js-файлер создаю функцию, которая в момент полной подгрузки страницы добавляет к блоку прелоадера класс со свойством-невидимкой:

Результат ( скрин момента начала анимации ):



## Создание собственной темы оформления. Часть 4. Обновлённые футер и хедер.

Шапку сайта я стилизовал с помощью bootstrap классов и стилей css, подключив меню с помощью функции php:





К примеру, размер пунктов меню я задал с помощью boostrap класса “h2”, добавив его с помощью js:

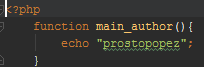




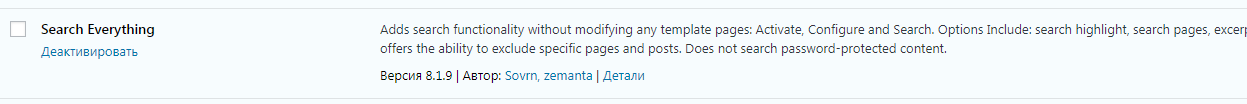
Далее стоит решить, за что будет отвечать футер сайта.

Пожалуй, стоит добавить туда поиск и значок авторства.

Задам написание ника автора как функцию:

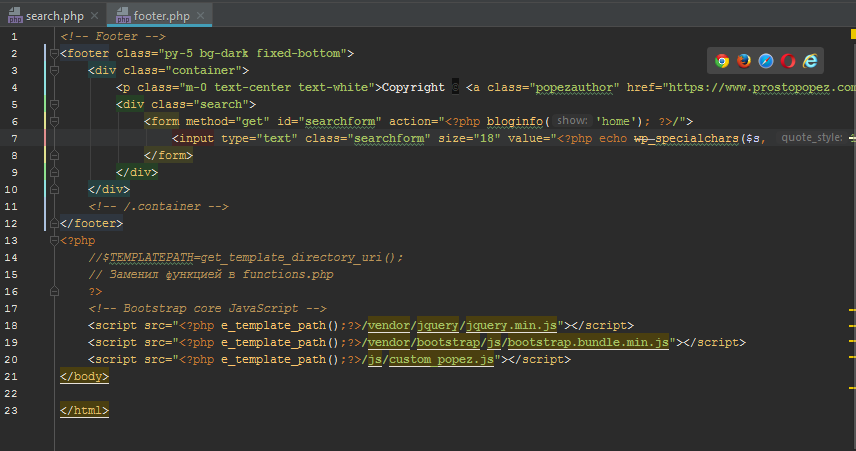


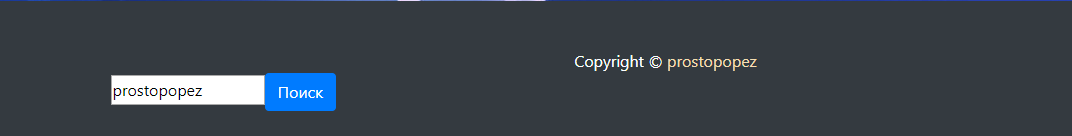
Тут же добавляю новый плагин wordpress для более удобной настройки поиска по сайту:



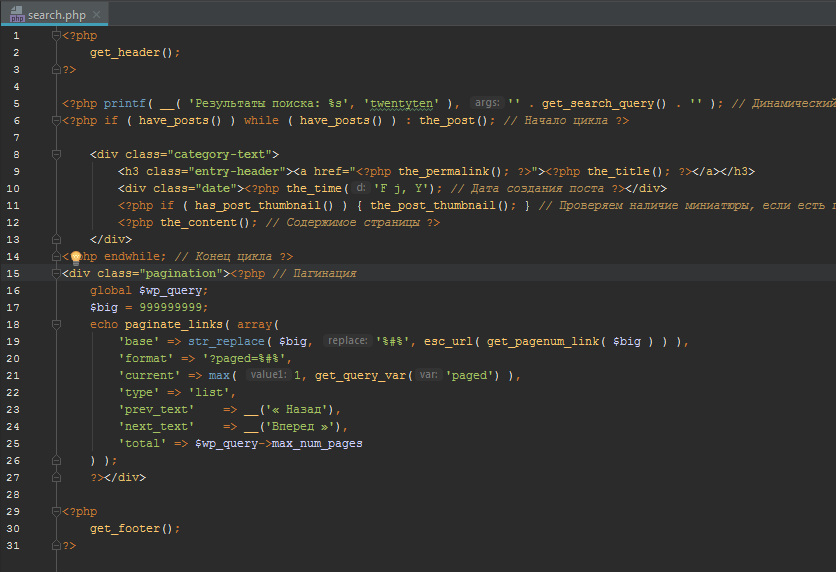
Редактирую файл footer.php, перенося футер сайта в него и добавляя форму поиска.

Вот так теперь выглядит файл “footer.php“.





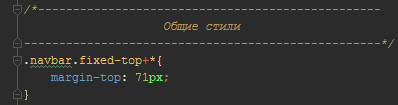
После поиска должны выводиться результаты. Для этого и служит файл seacrh.php. Для вывода результатов буду использовать типичный цикл WordPress с wp\_query, have\_posts() и the\_post(). Для начала подойдёт такое наполнение:



## Создание собственной темы оформления. Часть 5. Вывод контента страницы. Управление меню.

Пора починить вывод контента страницы.

Для начала исправлю недоразумение с «проглатыванием» контента страницы основным меню:

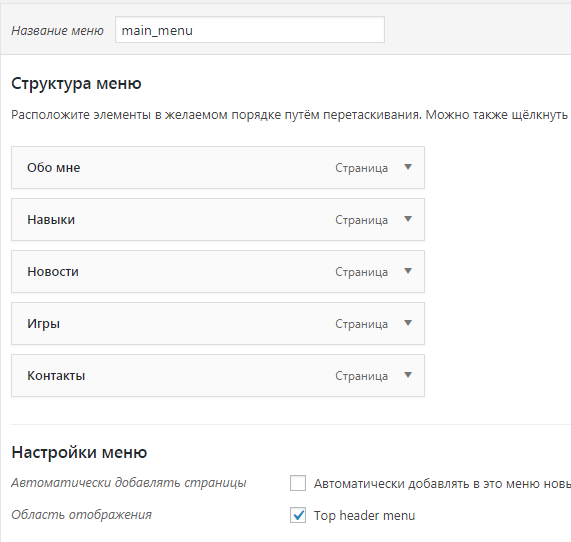


Футер же переносится в самый низ страницы, чтобы никому не мешаться.

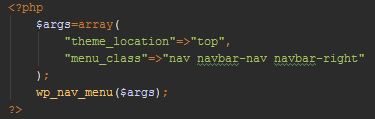
С помощью functions.php «регистрирую» меню:

Далее предстоит подключить его через административную панель WordPress, по необходимости воспользоваться помощью плагина Max Mega Menu.

Вот так выглядит структура меню в административной панели:

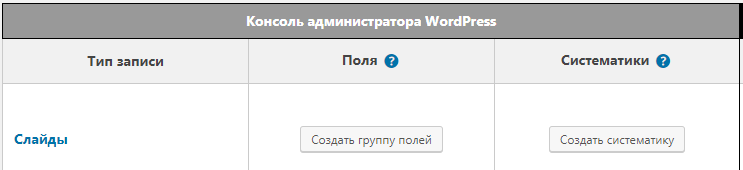


А вот уже знакомое подключение с помощью php в header.php с помощью значения свойства “theme location”, которое было задано при регистрации меню.



## Создание собственной темы оформления. Часть 6. Новый тип записи – слайды.

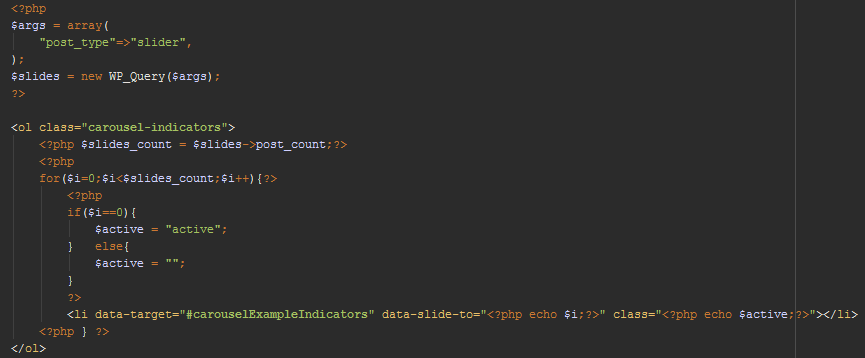
Чтобы заполнить главную страницу, начну с чего-то красочного. С помощью плагина Toolsep создаю пользовательский тип записей – Слайды.



Далее объясню работу созданного мной цикла в index.php. В дальнейшнем на схожих циклах останавливаться не буду.

Создаю переменную slides, которая берёт как аргумент wp\_query тип записи “slider“, созданный мной ранее. По сути, переопределяю тип записи, с которым работает на данный момент WordPress.

Далее создаю классы Bootstrap для корректного отображения будущего слайдера.

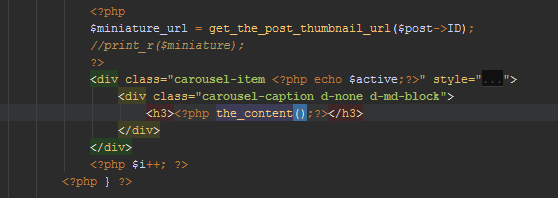
С помощью несложных манипуляций с php создаю механизм, отвечающий за правильную работу мини-переключателей внизу слайдов.



Теперь перехожу к выводу самих слайдов.

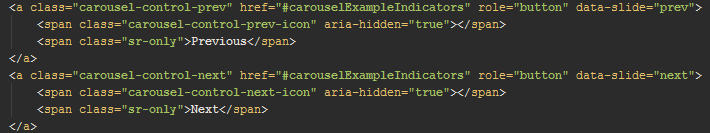
С помощью заданной ранее переменной проверяю наличие записей нужного типа и запускаю цикл, выводящий все запрошенные данные о нынешней «the\_post()» записи с помощью WordPress-функций, таких как:

1. get\_the\_post\_thumbnail\_url();
2. the\_content();
3. the\_title();
4. the\_excerpt();
5. и так далее.

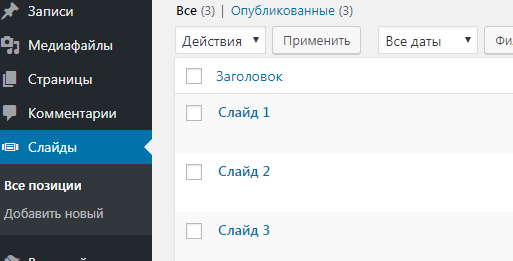


После завершения цикла нужно «замести следы», убрав из wp\_query нынешний тип записи.

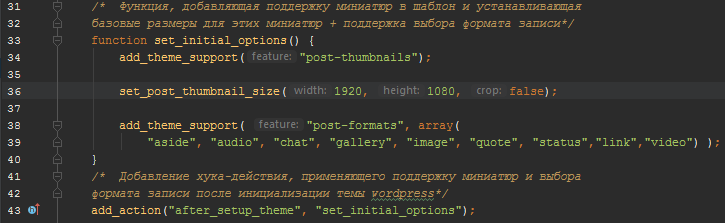


И в завершение добавлю боковые переключатели:

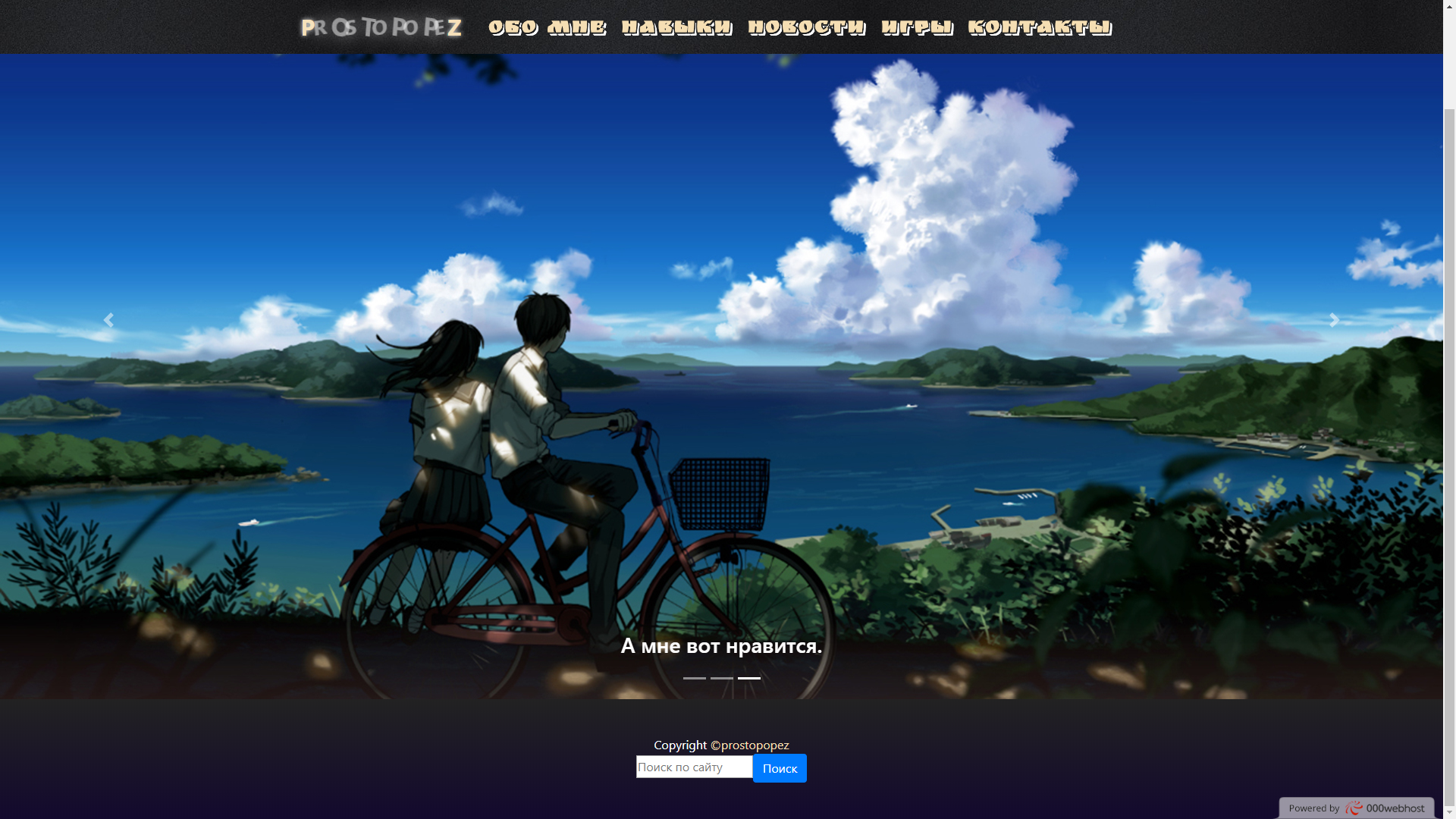
После добавления нового типа записи появится возможность добавлять записи этого типа:



Так как в index.php я использую именно миниатюры изображений, стоит добавить их поддержку в functions.php:



Вот что получилось:

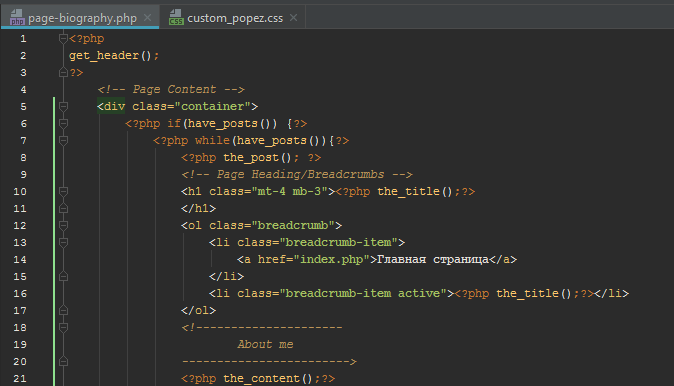


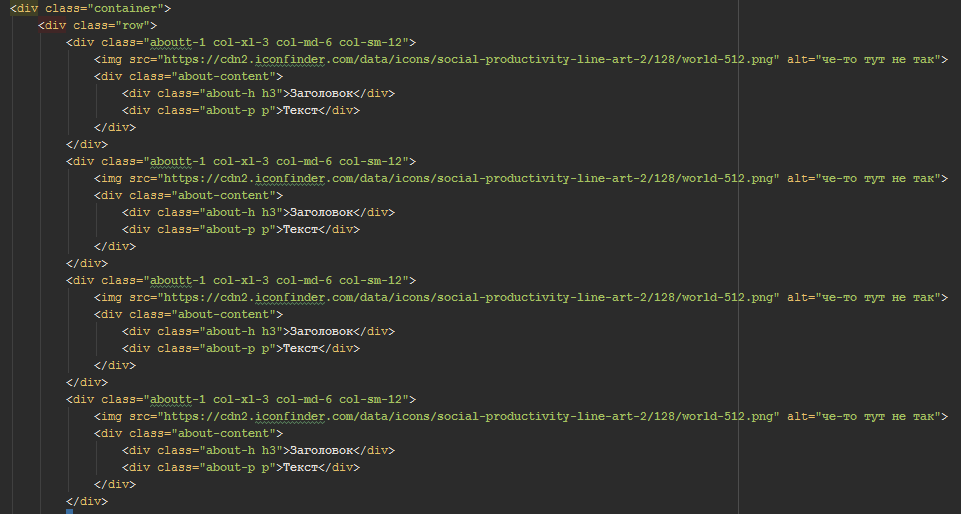
## Создание собственной темы оформления. Часть 7. Шаблоны страниц. Обо мне.

Конечно, на создание законченной и проработанной темы оформления у меня уйдёт довольно много времени, но прямо здесь и сейчас никто не мешает сделать базовые шаблоны.

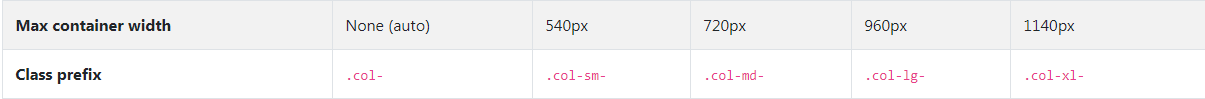
Начну с раздела «Обо мне».

Подключаю цикл WordPress, внутри него вывожу заголовок, «хлебные крошки» ( разделительная черта, вложенность) и контент страницы, редактируемый в административной панели.



Чтобы создать внутри раздела «обо мне» его собственные разделы, создам блок, внутри которого помещу, к примеру, 12 равноценных блоков, используя сетку boostrap. Сделаю её такой, чтобы при изменении расширения браузера менялось количество блоков в строке. За это отвечают классы, начинающиеся с “col-“.

Вот как эта сетка устроена в самом бутстрапе:



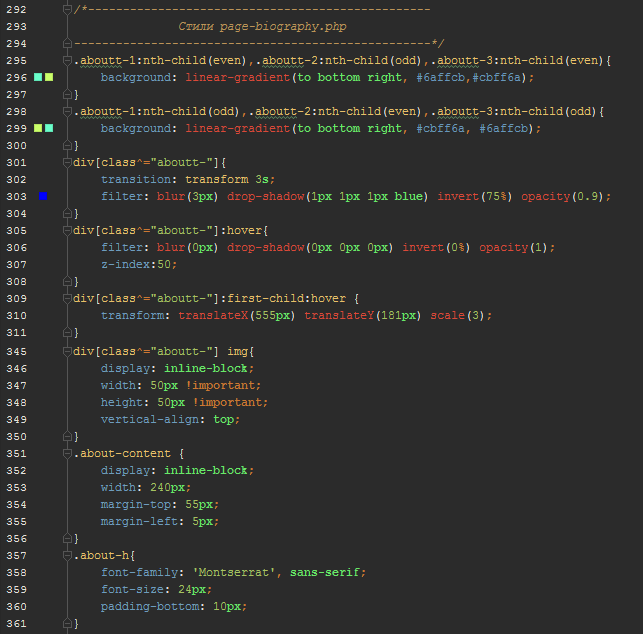
Второй «аргумент» класса отвечает за ширину браузерного окна, а третий – за пространство, занимаемое данным элементом в строке ( кол-во столбцов, всего их по умолчанию 12 ).

Мои блоки будут состоять из 3 элементов:

1. Картинка
2. Заголовок
3. Текст

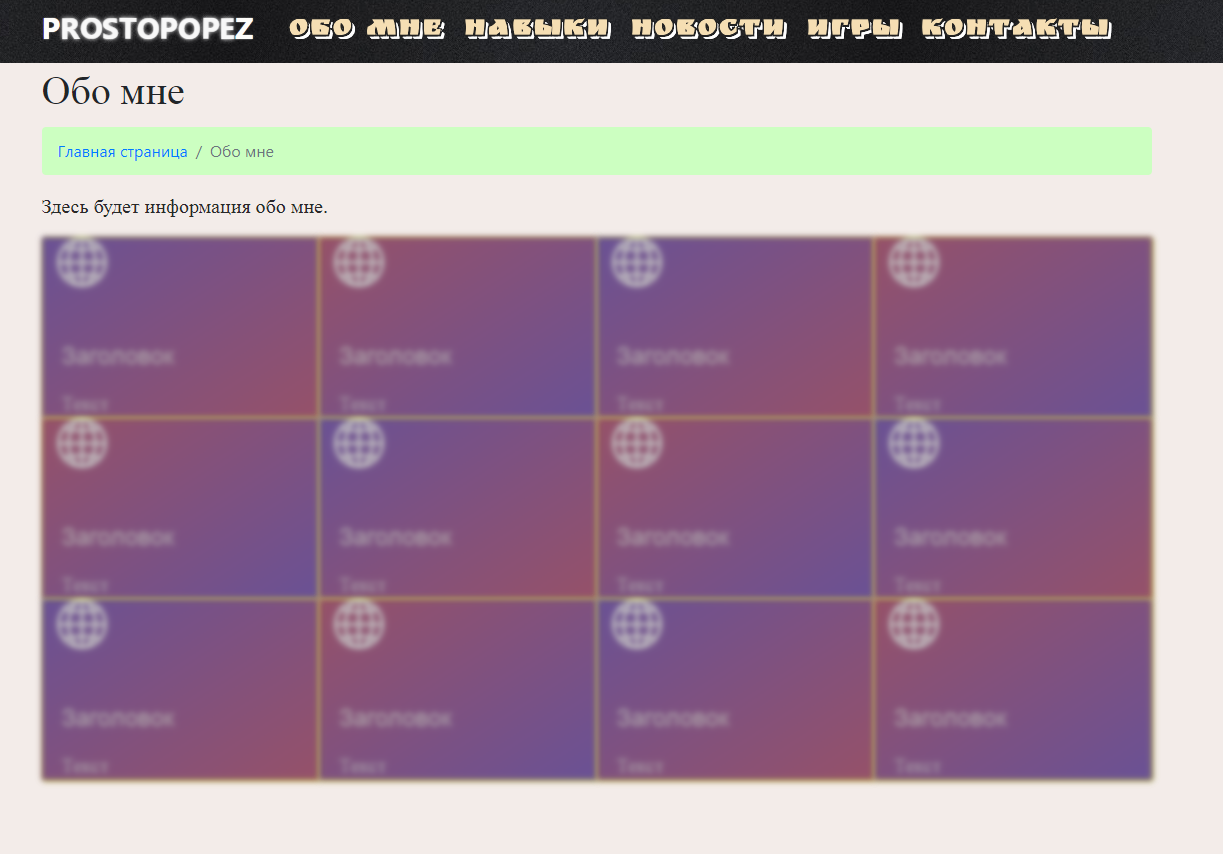
Каждый раздел будет кратко описывать автора в выбранном аспекте.

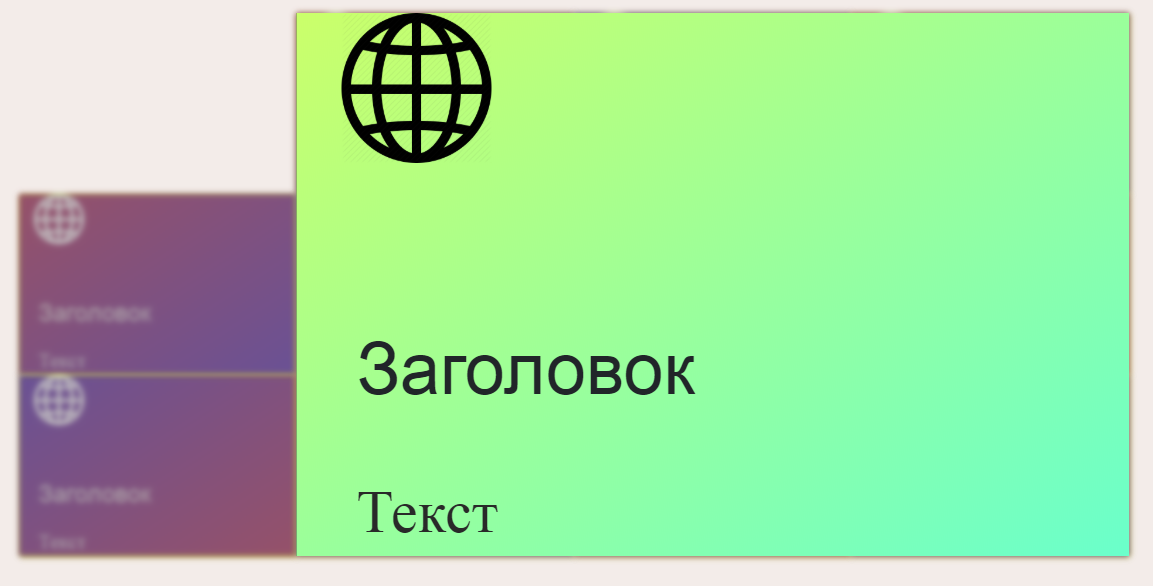
Теперь стоит позаботиться о некоторой стилизации:



Изначально каждый блок заблюрен, имеет тень, а так же инверсию цвета.

После наведения все эти свойства сводятся к нулю, сам же блок ощутимо увеличивается в размерах, хорошо вставая в свободное пространство 9 из 12 блоков. Перекрытие осуществляется за счёт изменения z-index-а элемента.





Скорее всего, в дальнейшем я сокращу код, переделав эти операции с помощью jQuery, а отдельно выбранные параметры можно будет вынести как новый тип постов, но пока что подобное я проделаю только в разделе «Новости» ( далее ).

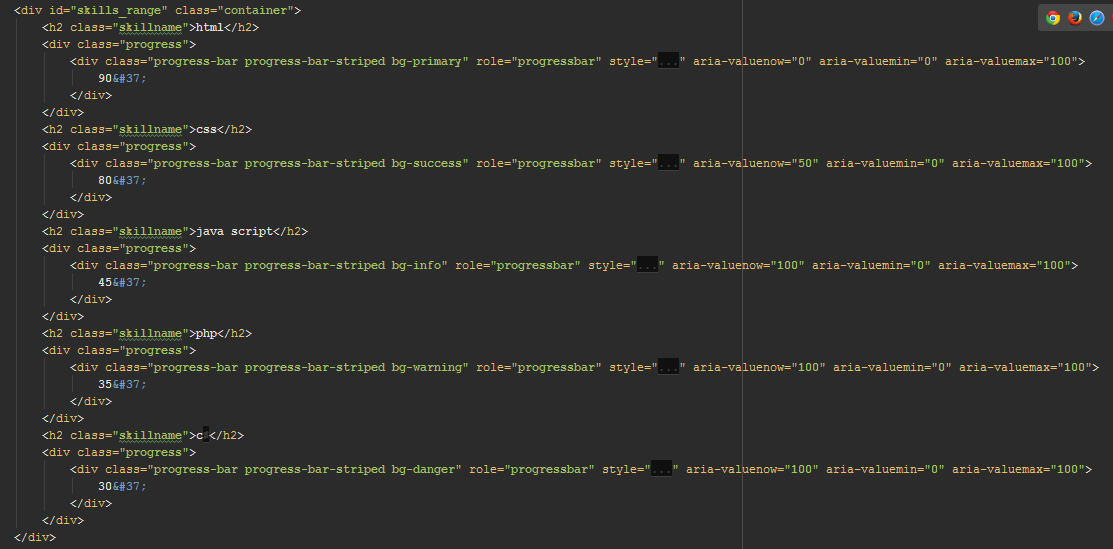
## Создание собственной темы оформления. Часть 8. Шаблоны страниц. Навыки.

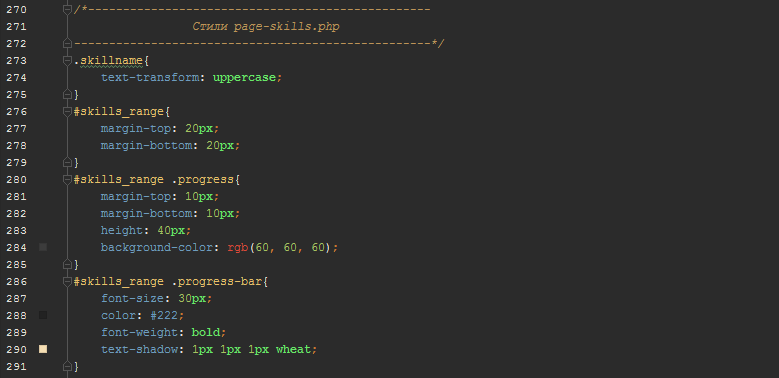
Цикл WordPress здесь работает так же, как и в разделе «Обо мне», поэтому заострять на нём внимание незачем.

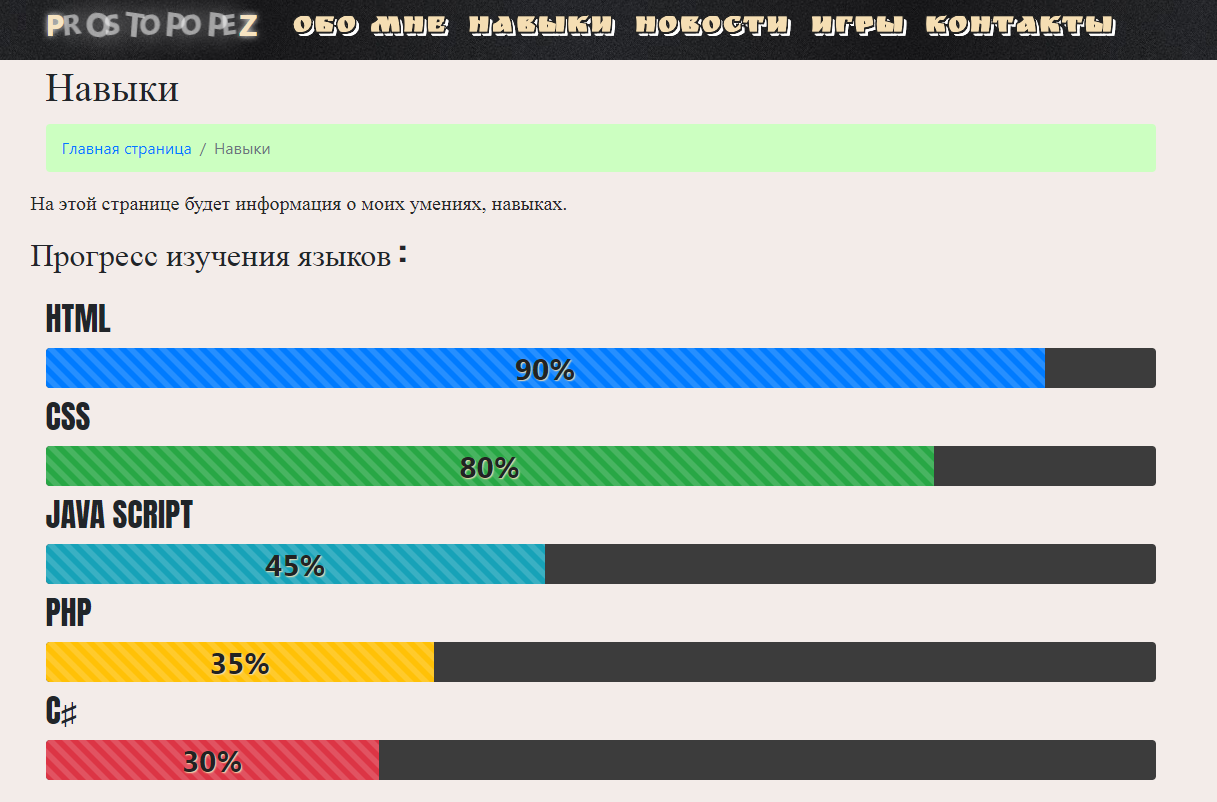
Для отображения навыков я создаю прогресс-бары, по пути стилизуя их с помощью css. Допустим, выберу 5 языков:

1. HTML
2. CSS
3. Java Script
4. PHP
5. C#

Отображу свой ориентировочный уровень знаний каждого языка:



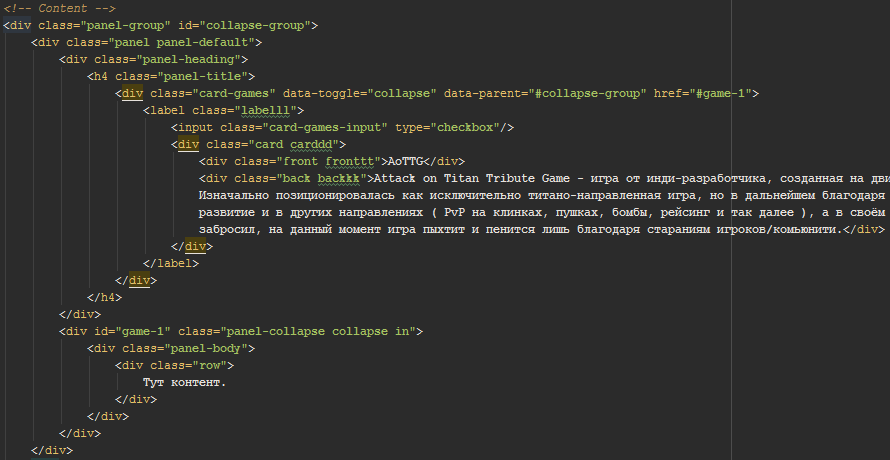




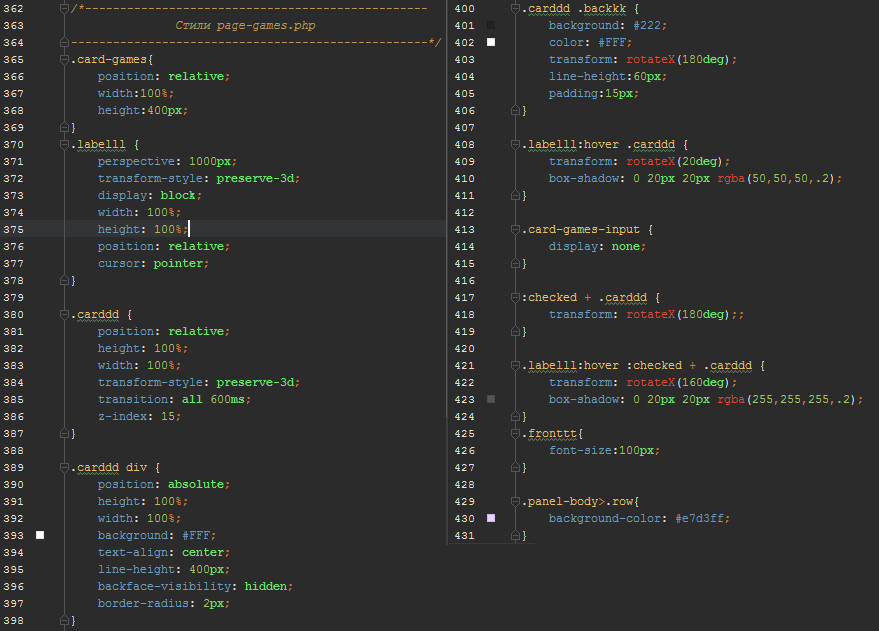
## Создание собственной темы оформления. Часть 9. Шаблоны страниц. Игры.

Для выбора игры сделаю группу из спойлеров, элементами on-click-а при этом станут карточки, которые с помощью трансформации «rotate» меняют сторону, которой повёрнуты к пользователю.

На задней стороны карточки будет минимально необходимая информация о конкретной игре, а чуть ниже полноценный блок контента.

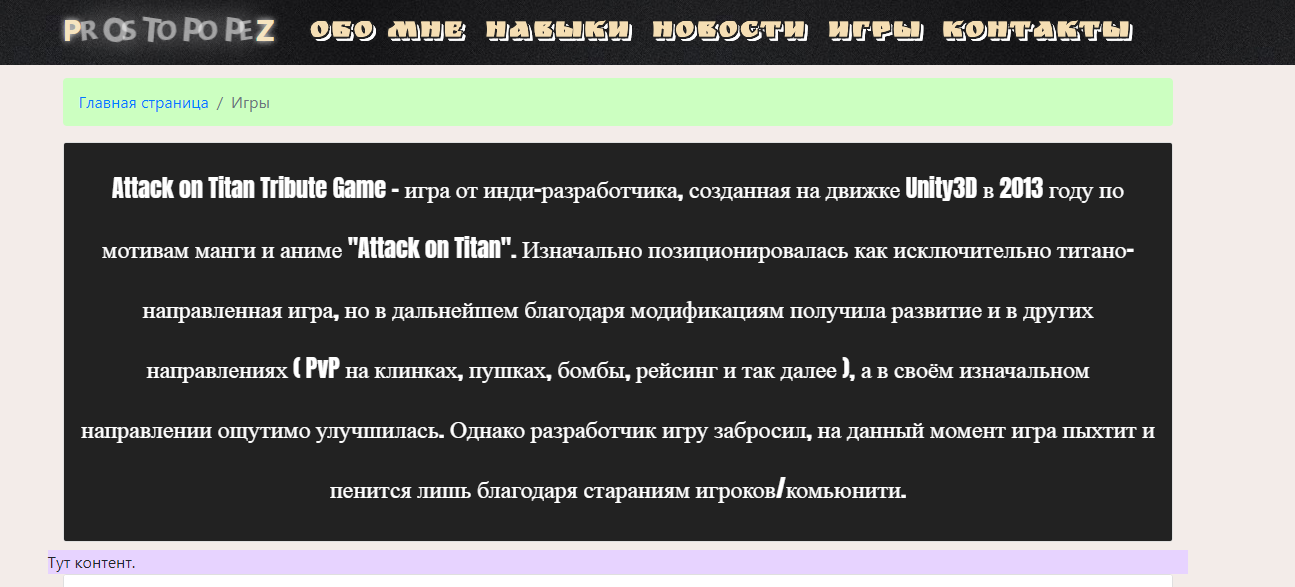


Css-код для спойлеров:



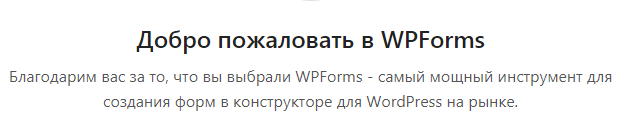
Вот что получается на выходе:





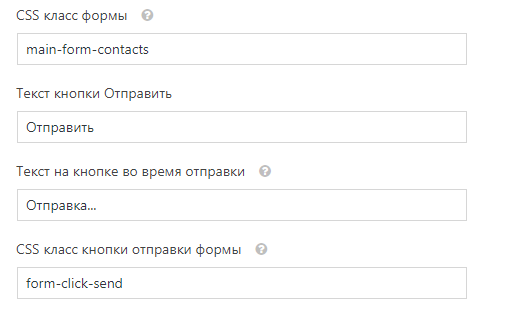
## Создание собственной темы оформления. Часть 10. Шаблоны страниц. Контакты.

Для создания контактной формы воспользуюсь плагином WordPress:

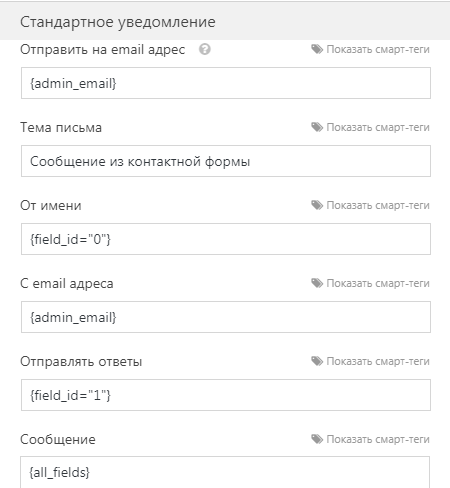


Настраивать форму полностью самостоятельно – занятие очень муторное, а при изменении формы добавление изменений вручную занимает большое количество времени.

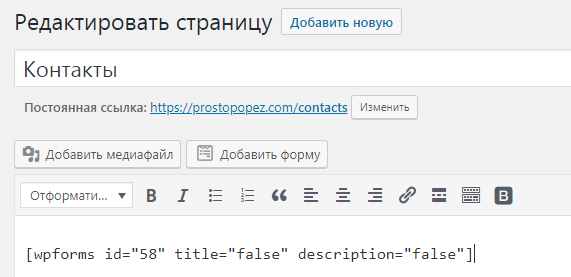
Так, во время создания формы для всех элементов задаются классы, используя которые можно будет стилизовать полученную форму.



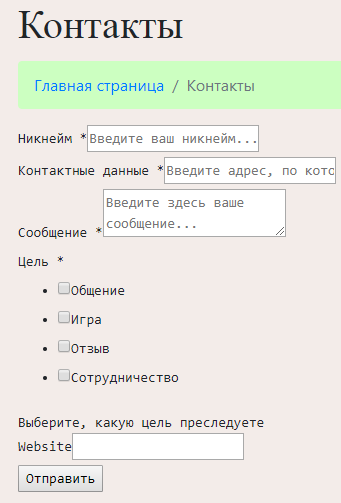
По умолчанию сообщение приходит на адрес, указанный администратором консоли WordPress, что очень удобно:



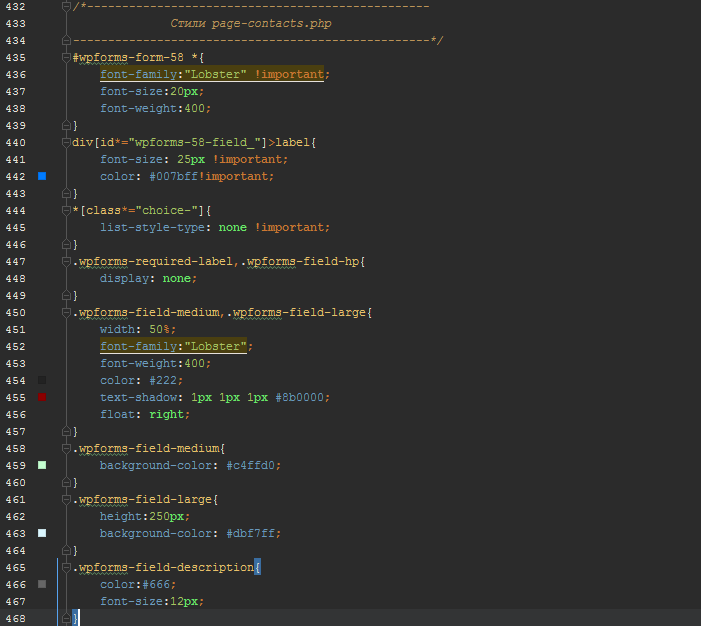
Далее в меню редактирования страницы вставляется шорткод, создающий форму:

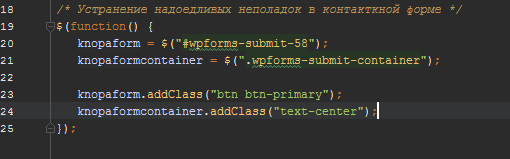


С помощью php функции “the\_content()“ выводится вот такая замечательная формочка:

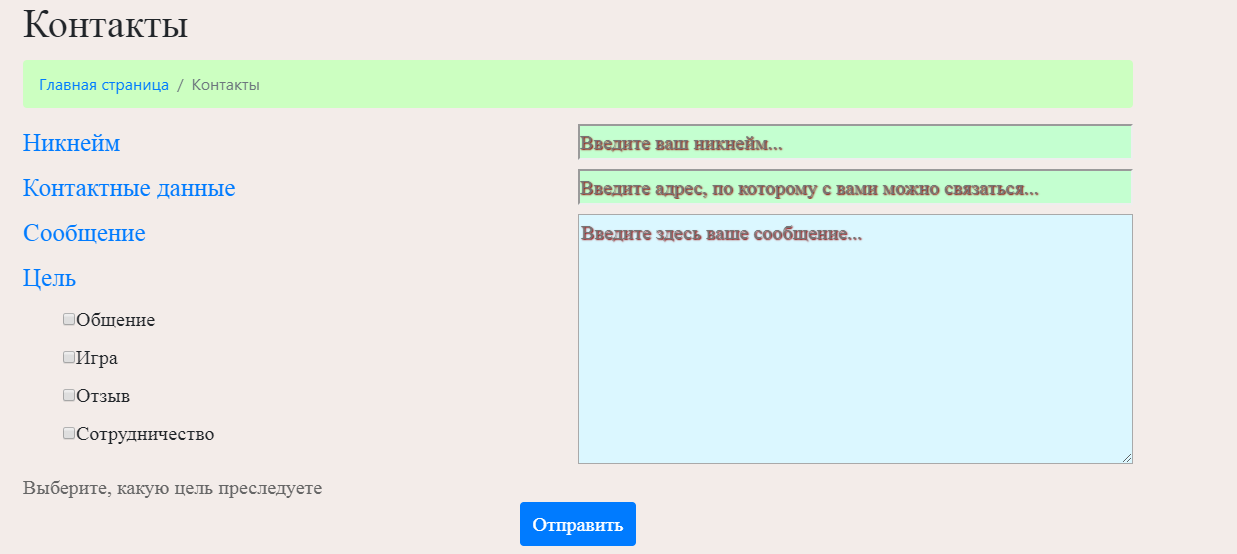


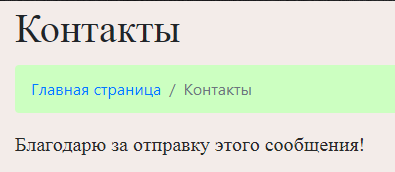
Пора привести её в приличный вид.





Вот так для начала будет получше:





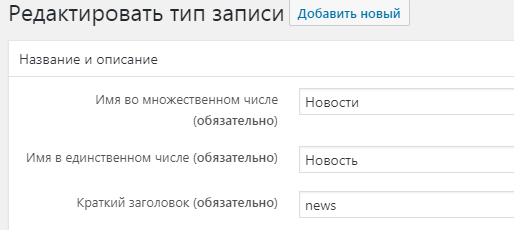
## Создание собственной темы оформления. Часть 11. Новости.

К разделу новостей я решил подойти с особым трепетом, поэтому воспользуюсь множеством функций WordPress.



Эти два файлика – шаблоны для вывода страницы-архива новостей и страницы – единичной новости.

Создам новый произвольный тип публикации WordPress – news ( Новости ):



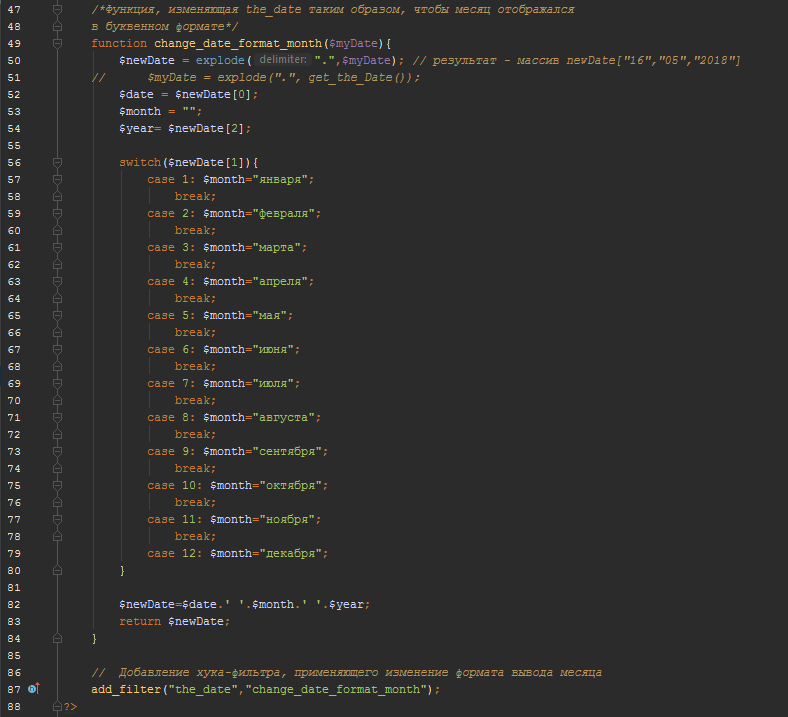
В файле archive-news.php создаю цикл WordPress, перебирающий записи того типа, который указан после “archive-“. Вывожу базовый шаблон новости:



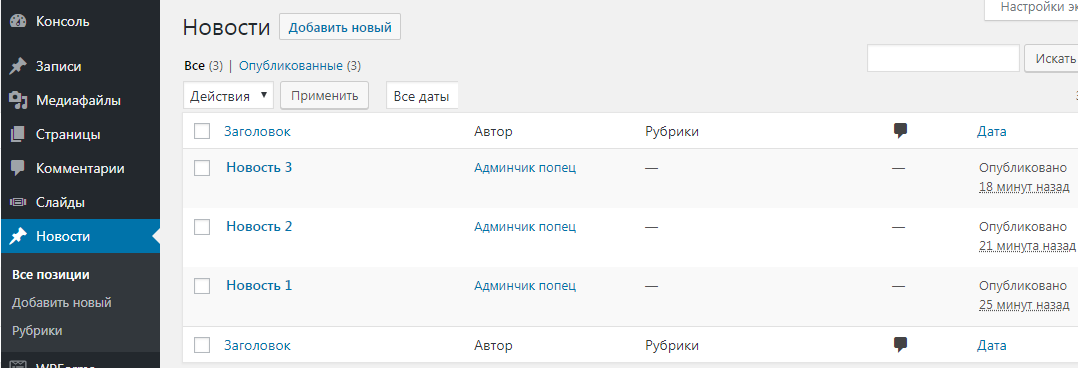
Большую часть работы над стилем я осуществил с помощью классов Bootstrap. Оставшиеся необходимые изменения можно добавить в css-файле:



Изменяю формат отображения даты новости с помощью написания хука в functions.php:

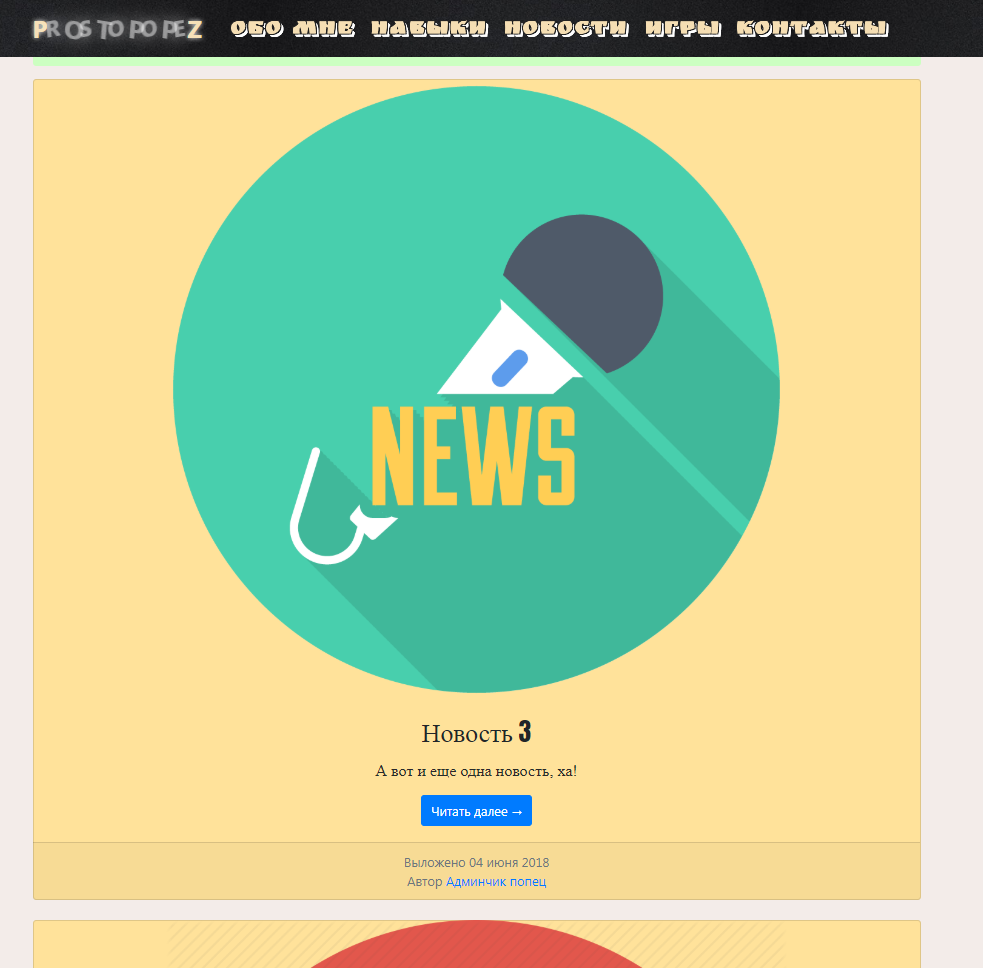


Через административную панель добавляю 3 новых публикации типа «Новости»:



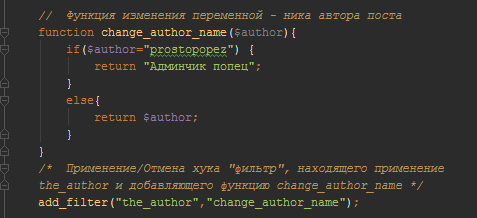
По пути пришлось немного перекроить менюшку, ведь статичную страницу page-news.php я уже удалил.

А вот и готовые новости:



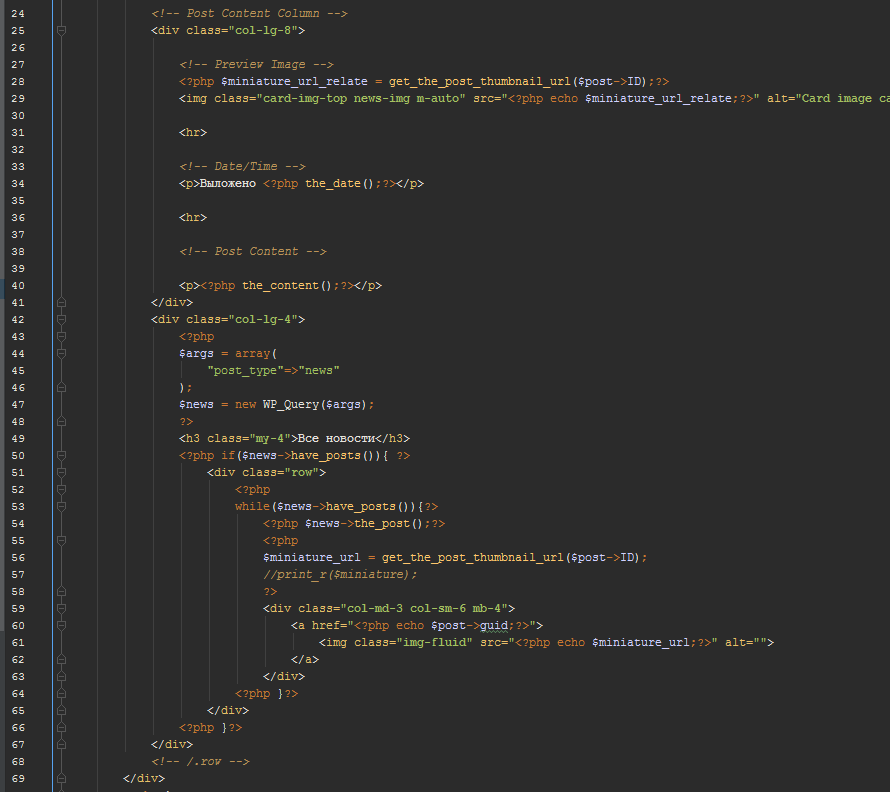
Внизу виднеется вторая новость, все функции работают отлично.

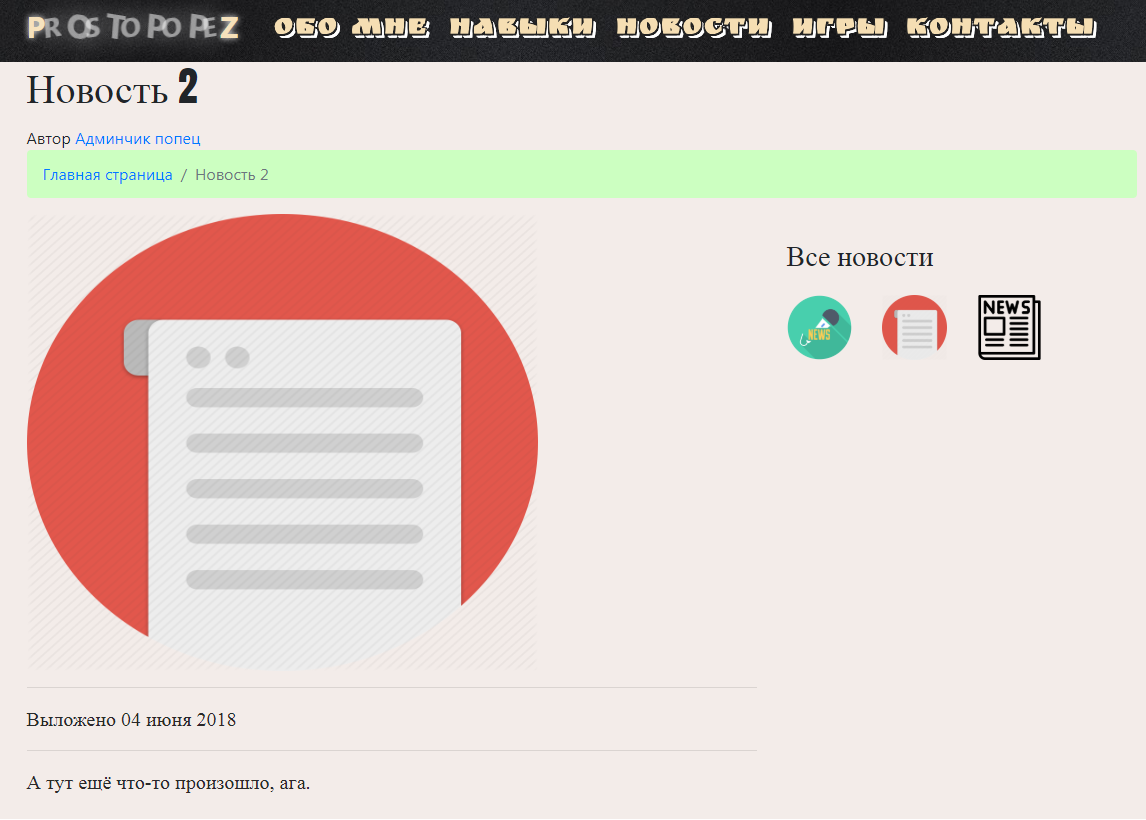
Никнейм автора я отредактировал с помощью хука в functions.php:



Самое время создать шаблон отображения одиночных новостей – single-news.php.

При нажатии «Читать далее» в архиве Новостей пользователь будет попадать именно туда.



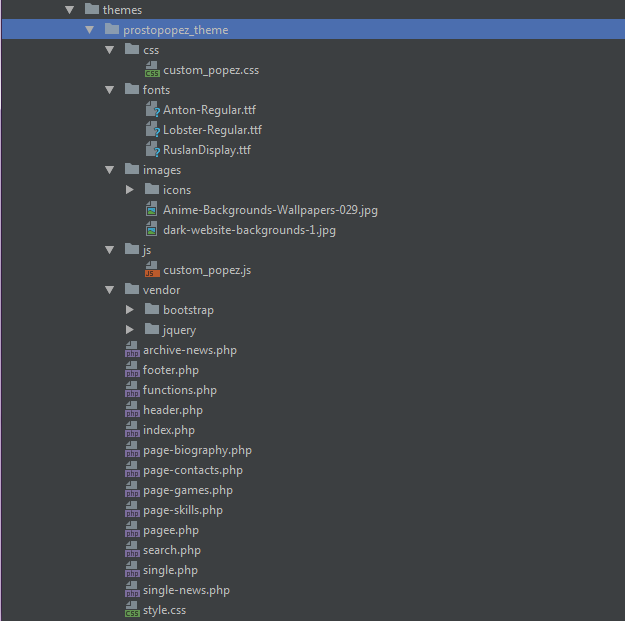
Помимо вывода контента я добавил что-то наподобие сайдбара, который выводит все посты с гиперссылками справа от контента нынешнего.

Вот и всё. Все базовые страницы готовы.

# Вывод

Я разобрался в CMS WordPress, закрепил свои знания в вёрстке и веб-программировании, создал первую версию собственной темы оформления.

Мне ещё предстоит много практиковаться, научиться настраивать безопасное соединение, познакомиться с SEO и т.д. и т.п.



# Список использованной литературы

1. Собственные знания
2. Кузина В.В. Web-дизайн, проектирование и разработка сайтов: учебное пособие по направлению подготовки 09.03.02 «Информационные системы и технологии» / В.В. Кузина, И.Г. Гвоздева. - Пенза: ПГУАС, 2018. - 156 с.
3. <https://copywriting-rules.ru/chto-takoe-cms/>
4. <https://zyubin.ru/saitostroenie/web-info/>
5. <https://ru.wikipedia.org/wiki/>
6. <http://komotoz.ru/uroki/javascript/>
7. <https://starper55plys.ru/php/>
8. <http://blogwork.ru/chto-takoe-bootstrap/>
9. <https://habr.com/post/136525/>
10. <http://designcenter.org.ua/articles/web-verstka/>
11. <https://webformyself.com/chto-takoe-jquery-2/>
12. <http://fb.ru/article/332183/mysql---eto-chto-takoe-i-gde-primenyaetsya>
13. <http://joomla.ru/docs/administrator/dokumentatsiya-po-joomla-2-5/1675-chto-takoe-ftp-i-dlja-chego-on-nuzhen>