**ТИТУЛЬНЫЙ ЛИСТ**

**Cодержание**

**Введение**

Мой условный путь в программирование начался задолго до обучения в GeekBrains. Будучи студентом машиностроительного техникума, я изучал веб-разработку с использованием довольно простых инструментов по типу Adobe Muse или Tilda, что по своей сути является конструктором сайтов. Если заглянуть непосредственно в код проектов, сделанных таким образом, можно ужаснуться. Немного позже я углубился в изучение HTML и CSS по огромной книге «Head First HTML and CSS» (Робсон Элизабет и Фримен Эрик) после этого JavaScript по книге «Изучаем программирование на JavaScript» за авторством тех же писателей. Но после недолгих практических шагов в JavaScript, жизненные обстоятельства заставили на какое-то время отстраниться от этой ниши.

Спустя шесть лет мне придется вернуться к одному из своих проектов, чтобы продемонстрировать результат своего обучения.

Размышляя над темой дипломного проекта, мне очень пригодился свой предыдущий опыт. Чтобы проект не выглядел выдуманным из головы и не высосанным из пальца, я решил модернизировать существующую франшизу. Раньше приходилось брать много задач и делать их в одиночку, не считая настройки рекламных кампаний. А это написание текста для сайта, создание оффера, создание дизайн макета сайта, логотипа, и, конечно же все это необходимо было сверстать и заставить работать.

В плане на эту дипломную работу – взять созданный мной ранее проект, который реализован на HTML/CSS и JavaScript, без использования каких-либо фреймворков и сделать из него полноценное веб-приложение. Для этого я воспользуюсь фреймворком Laravel.

Мы рассмотрим его преимущества в дальнейшем, но если говорить вкратце, то я остановил свой выбор на конкретном фреймворком по простым причинам:

* открытый исходный код
* безопасность
* возможность масштабирования
* простота работы с базой данных

И в целом, мне как новичку Laravel показался очень удобным и интересным фреймворком, который я хочу в дальнейшем изучить глубже.

В задуманном мной проекте, конечно, можно обойтись без таких функциональных фреймворков, но в качестве презентации своих навыков и умений, как fullstack разработчика – отлично подойдет, плюсом послужит и то, что приложение можно будет масштабировать по своему желанию.

Сам заказчик, на момент обращения ко мне, занимался изготовлением loft мебели в Санкт-Петербурге под заказ. Насколько я понимаю на текущий момент продолжает заниматься, но сам сайт уже изменил. Во избежание недопониманий и просто на всякий случай я заменю всю контактную информацию и некоторые фотографии заглушками. Но все созданное в проекте является моей интеллектуальной собственностью (помимо фотографий), и я считаю допустимым непубличное использование веб-сайта в качестве основы моего дипломного проекта.

План проекта:

1. Разработать структуру и дизайн веб-страницы.
2. Сверстать страницу, используя HTML.
3. Использовать CSS, чтобы оформить страницу.
4. Использовать JavaScript, чтобы написать логику взаимодействия со стороны пользователя.
5. Подключить базу данных для работы с пользователями и товарами.
6. Разработать систему нотификации.
7. Создать административную панель для управления контентом, не прибегая к помощи специалистов.

Инструменты, которые будут использованы в проекте:

- Adobe Photoshop ()

- Visual Studio Code (для работы с языком программирования PHP лучше использовать PHP Storm, но для нашего проекта можно обойтись и VSCode) \*можно описать какие внутри инструменты

* Git
* Laravel 11.0
* Postman
* PhpMyAdmin
* Composer

Состав предполагаемой команды: маркетолог, копирайтер, дизайнер, frontend разработчик,

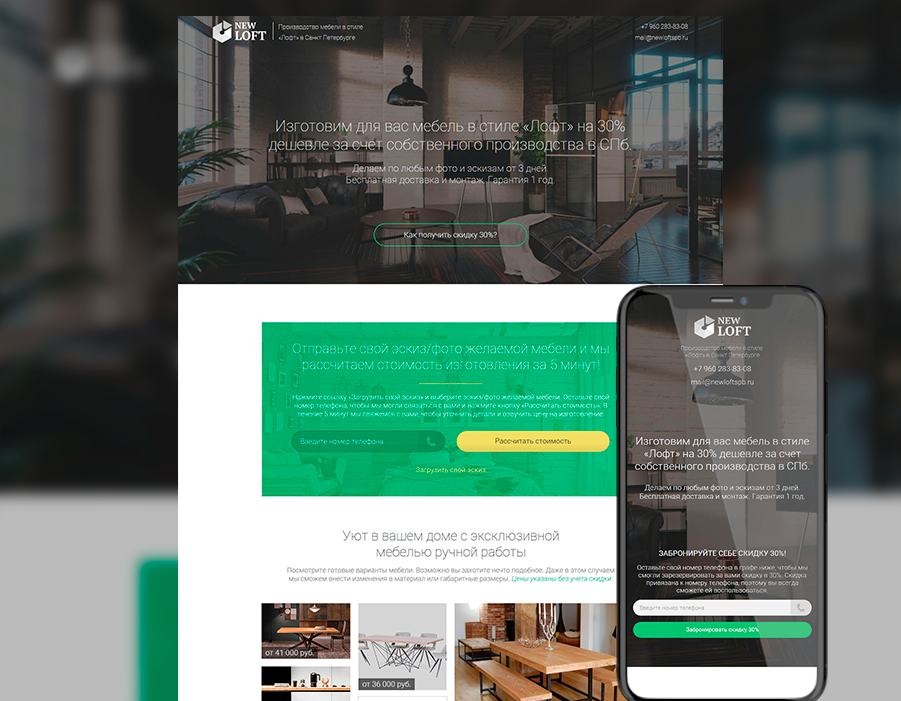
backend разработчик (или fullstack вместо frontend и backend). В качестве вышеперечисленных работников буду выступать непосредственно я сам.

**Верстание страницы по дизайн-макету.**

На текущий момент разработчики в большинстве своем используют Figma или аналогичные онлайн-сервисы для работы с прототипами интерфейсов в реальном времени. В нашем случае дизайн-проект разрабатывался в Adobe Photoshop. Ничего плохого в этом нет, но сам Photoshop довольно ресурсоемкий для операционной системы и не очень подходит для таких задач, как посмотреть расстояние между элементами. Естественно, что лучше использовать что-то проще.

Итак, допустим, что дизайнер разработал будущий вид сайта и нам, как fullstack разработчику, необходимо все это собрать в уже работающее приложение.

Так как «скелет» проекта у меня уже есть, я попросту его доработаю, используя новые технологии и знания. При этом не будем углубляться во внешний вид. Оставим все так, как и было.



Начнем с верстки, используя HTML.

**HTML5**

HTML (HyperText Markup Language или язык гипертекстовой разметки) – это язык разметки, который используют для создания веб-страниц. Если говорить простым языком, весь контент, который вам необходимо разместить помещается в определенные теги. Для каждого типа контента предназначен свой собственный тег.

На текущий момент используют последнюю пятую версию HTML. Она позволяет располагать на странице мультимедиа, не прибегая к помощи других программ, как это было ранее, также улучшилась сама семантика: появились новые теги, которые помимо всего прочего помогают браузеру лучше ранжировать страницы, а это плюс к SEO продвижению, ну и конечно, с ростом технологий HTML5 лучше поддерживается смартфонами.

**CSS3**

Если говорить о верстке страницы, нам не обойтись без визуальной составляющей. В этом нам поможет CSS (Cascading Style Sheets / каскадная таблица стилей). Это своего рода правила визуализации HTML элементов. С помощью CSS мы «сообщаем» браузеру как в конечном итоге должна выглядеть страница.

В наше время CSS перетянул очень много функций с JavaScript. Поэтому очень многое можно сделать, не прибегая к прямому программированию. А как мы знаем, что HTML, что CSS не являются языками программирования.

**JavaScript**

Хотя я и сказал, что очень многое можно сделать, не прибегая к JS, это вовсе не значит, что можно обойтись без него. Вся логика внесена именно сюда. А такие фреймворки как ReactJS упрощают конечное производство за счет огромных библиотек готовых решений. Вам уже не придется писать какие-либо банальные функции, тратя свое драгоценное время.

**Подытожим…**

Итак, три кита построения веб страницы: HTML, CSS, JavaScript.

Сделаю некоторое отступление от темы и объясню некоторые моменты на будущее.

В сложном проекте скорее всего придется работать с фреймворками по типу Vue или библиотекой ReactJS, которые упрощают жизнь фронтенд-разработчикам. Они используют декларативный подход, который состоит в описании конечного результата, вместо императивного, где описываются конкретные шаги для получения результата.

В данном проекте не имеет смысла использовать ничего такого. Потому что, по сути, у нас одна страница. В конечном итоге ее можно разбить на компоненты, при масштабировании проекта или при его переосмыслении. В текущем проекте мы используем простой шаблон blade. Но это я забегаю вперед.

**Перед началом работы.**

Чтобы как-то контролировать процесс разработки: иметь возможность быстро откатывать все назад при необходимости, мы будем пользоваться системой контроля версий GitHub.

В целом вся суть таких систем заключается в том, чтобы синхронизировать работу нескольких разработчиков. То есть где-то на сервере хранится исходный код веб-приложения (репозиторий) и каждый из команды имеет к нему доступ (а может и не только из команды), имеет возможность загрузить этот код себе, работать с кодом и отправить изменения в репозиторий. Однако только владелец может принять изменения и сделать слияние с исходным кодом.

Инициализировать такую систему лучше при старте разработки и не забывать вовремя коммитить свою работу (то есть зафиксировать определенный результат на текущее время).

**Начинаем верстку**

Заходим в наш редактор кода и начинаем с базовой разметки HTML. Для облегчения и скорости работы нам понадобятся некоторые расширение по типу HTML5 Snippets, JS-CSS-HTML Formatter, Emmet и Sass.

**<head>** - записываем сюда все метаданные, описание страницы, ключи для ранжирования в поиске, заголовок страницы, подключаемые шрифты, стили и прочее. Иногда сюда, при определенных случаях, можно вставить и ссылку на какой-либо скрипт. В моем случае была подключена метрика.

**<body> -** основное тело нашей страницы. Размещаем сюда весь контент, а это такие теги, как заголовки разного уровня (<h1>, <h2> и т.д.), параграфы (<p>), ссылки (<a>), картинки (<img>), видео (<video>), аудио <audio>, списки (<ul>, <li>) и прочие.

У большинства тегов есть свои атрибуты – это дополнительный значения, которые настраивают элемент соответствующим образом. Есть обязательные атрибуты, такие как href у ссылки, а есть и дополнительные, такие как id или class. Один из основных атрибутов, которые нам нужно использовать практически для каждого элемента это class:

**<a href=”#” class=”some-class”/>**

Атрибут class присваивает идентификатор элементу, по которому мы сможем обращаться в CSS. Причем атрибут class с одним и тем же именем может быть присвоен группе элементов в отличие от атрибута id, который, во избежание ошибок, должен быть уникальным и единственным.

Разрабатывая разметку нам сразу предстоит думать и о названиях классов для будущей работы со стилями. Мы будем использовать препроцессор Sass, который ускоряет написание стилей. Sass позволяет создавать переменные, делать вложенные стили, миксины, использовать фрагментирование и множество других функций, упрощая разработку и делая ее очень удобной.

Вот наш начальный блок. Обратите внимание на название классов.

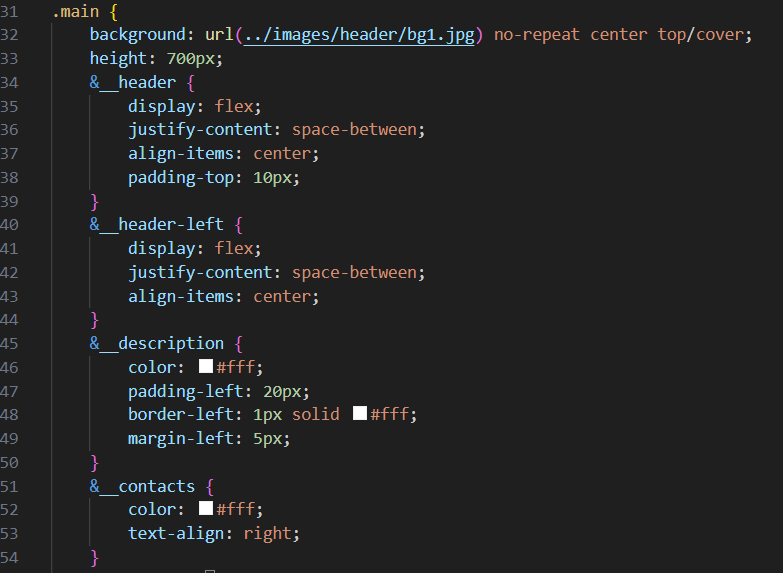
Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Получается некая древесная структура, где main – корень, а приписки вида \_\_header – ветви. Отчетливо это будет видно в файле css.

В CSS мы начинаем прописывать некие правила по схеме «свойство: значение». Описывать их все не имеет смысла, потому что их очень много, не говоря уже о системе позиционирования элементов. Но можно сказать, что если вам что-то придет в голову в плане визуализации элементов, то вероятнее всего это можно сделать с помощью стилей.

Обратите внимание как препроцессоры облегчают читаемость кода. У нас есть основной класс main и все свойства, что описаны внутри этого блока, касаются конкретного блока в html.



Таким образом описываем каждый блок с дизайн макета и получаем практически готовую веб-страницу. Что касается форм на странице, то мы пропишем их логику и взаимодействие непосредственно в Laravel. Сейчас нам остается прописать некоторый функционал по всплывающим окнам на JavaScript.

Список источников:

Документация по Laravel Git и прочим инструментам.

«Head First HTML and CSS» Робсон Элизабет и Фримен Эрик

Изучаем программирование на JavaScript Робсон Элизабет и Фримен Эрик

<https://ru.hexlet.io/blog/posts/pochemu-laravel-odin-iz-luchshih-php-freymvorkov-dlya-startapov-i-enterprayza>

<https://education.yandex.ru/journal/chto-takoe-react-i-kak-ego-osvoit>

https://sass-scss.ru/guide/

<https://code.tutsplus.com/ru/build-a-react-app-with-laravel-restful-backend-part-1-laravel-5-api--cms-29442t>

https://delphinpro.ru/a/6