Geekbrains

**Погружение в деятельность fullstack-разработчика.**

**Создание сайта-приложения по заданному макету и техническому заданию при помощи фреймворка Laravel.**

Программа: Программист

Специализация: Fullstack-разработчик

Никулин Дмитрий Сергеевич

Санкт-Петербург

2025 год

**Содержание**

**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 прочтения мне захотелось углубиться в программирование на языке JavaScript. И следующей книгой стала «Изучаем программирование на JavaScript» за авторством тех же писателей. Но после недолгих практических шагов в JS, жизненные обстоятельства заставили на какое-то время отстраниться от этого.

До этого я успел поработать со многими людьми и разработать несколько простых сайтов, которые назывались посадочными страницами или landing page. По сути, одностраничный сайт-визитка без особого функционала.

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

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. Сайт, который я делал в 2018 году для одного из своих клиентов.

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 можно обойтись без таких функциональных фреймворков, но в качестве презентации своих навыков и умений, как fullstack-разработчика – отлично подойдет, тем более придется много всего затронуть. 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. На текущий момент он продолжает этим, но сам сайт уже изменил. Во избежание недопониманий и просто на всякий случай я заменю всю контактную информацию и некоторые фотографии заглушками. Все созданное в проекте является моей интеллектуальной собственностью (помимо фотографий), и я считаю допустимым непубличное использование веб-сайта в качестве основы моего дипломного проекта.

**Цель проекта**

Создать веб-сайт, через который посетитель сможет ознакомиться с продукцией и оставить заявку. Занести информацию о клиенте в базу данных для дальнейшей передачи менеджеру. Разработать систему уведомлений для оповещения менеджеров.

**План проекта**

1. Разработать структуру и дизайн веб-страницы.
2. Сверстать страницу на HTML, в дальнейшем это будет blade-шаблон
3. Оформить страницу с помощью CSS, использовать SASS для удобства.
4. С помощью JavaScript создать логику взаимодействия со стороны посетителя.
5. Подключить базу данных для работы с пользователями, а в будущем с товарами.
6. Разработать систему нотификации.

**Инструменты**

* Adobe Photoshop – для работы с макетом
* Visual Studio Code – удобный редактор кода для всех языков
* Git – система управления версиями
* Laravel 11.0 – бесплатный веб-фреймворк
* PhpMyAdmin – интерфейс для работы с базой данных
* Composer – менеджер пакетов

**Состав команды**

* Маркетолог
* Копирайтер
* Дизайнер
* Frontend-разработчик
* Backend-разработчик
* Fullstack-разработчик (вместо двух предыдущих)

В качестве вышеперечисленных работников буду выступать непосредственно я сам.

**Глава 1. Задачи fullstack-разработчика.**

**1.1 Кто такой fullstack-разработчик и какие у него задачи**

Fullstack-разработчик – это специалист, который воплощает в себе функции двух других специалистов frontend-разработчика и backend-разработчика.

Frontend занимается разработкой клиентской части приложения. Грубо говоря, это то, что вы видите через свои мониторы и тот функционал, с которым непосредственно можете взаимодействовать: картинки, видео, кнопки, формы и т. д. Фактически frontend-разработчик в среде веб-сайтов занимается версткой страницы на HTML, задает этой странице внешний вид (желательно по дизайн-макету, который создает уже другой специалист – веб-дизайнер). За внешний вид отвечает каскадная таблица стилей CSS. Следующий задачей разработчика является разработка функционала с помощью JavaScript. В нем можно реализовать любую логику, а это переключение картинок, всплывающие окна, анимации, подключение сторонних плагинов. Что не менее важно, с помощью JS клиентская часть приложения генерирует условный контент на основе приходящей информации из сервера.

Backend-разработчик занимается тем, что конечный пользователь не увидит. Это серверная часть веб-приложения. Обычно бэкенд-разработчики заняты написанием серверного кода, разработкой логики работы приложения, и поддержанием его инфраструктуры. В их обязанности может входить множество вещей, в зависимости от конкретного приложения: оптимизация кода, работа с базой данных, настройка API, отладка и написание тестов.

Fullstack-разработчик совмещает в себе все эти элементы и отвечает за обе части веб приложения: клиентскую и серверную. Обычно на них лежит огромный объем работы, но при этом в молодых компаниях они пользуются спросом, потому что можно завербовать специалиста 2 в 1.

**1.2. Необходимые инструменты для разработчика.**

Чтобы эффективно работать и не тратить свое время на выполнение простых и рутинных вещей, разработчику требуется установить некоторый стек инструментов.

Для начала нужен редактор кода, в котором будет происходить большая часть работы. Вряд ли кто-то в здравом уме будет писать код в текстовом редакторе. Какой выбрать зависит только от предпочтений разработчика и языка программирования, на котором будет происходить работа. Самый универсальный – это Visual Studio Code. Он не тяжеловесный, подходит для большинства языков программирования и имеет достаточно хорошую библиотеку расширений, которые облегчат работу программиста.

Для управления версиями приложения понадобится такая система, как Git. О ней подробнее в следующей главе.

В моем случае понадобится менеджер пакетов Composer. Он выполняет функцию загрузки необходимых пакетов для работы приложения. Представим, что для работы приложения необходимо двадцать файлов с кодом и все они находятся в разных местах интернета. Конечно, можно потратить время и качать каждый файл по отдельности. А можно использовать composer-файл, в котором прописаны все зависимости что и откуда брать и как установить. Поэтому одной командой можно освободить себе кучу времени.

Также необходимо будет установить графический интерфейс для работы с базами данных PhpMyAdmin. Можно управлять базой данных и через консоль, но так гораздо удобнее. Есть и альтернативные приложения по типу MySQL Workbrench, но тут уже на вкус и цвет.

Еще понадобится Open Server — это портативный локальный WAMP/WNMP сервер, имеющий многофункциональную управляющую программу и большой выбор подключаемых компонентов. Так как при разработке внешнего сервера скорее всего нет, необходимо его чем-то заменить, в конкретном случае локальным сервером.

**1.2 Перед началом любой работы. GIT.**

Прежде, чем говорить о какой-либо разработке стоит задуматься о какой-нибудь системе бэкапов и задаться правильными вопросами.

Как контролировать процесс разработки? Иметь возможность быстро откатить изменения при необходимости? А если над проектом трудится не один человек, как сравнивать и анализировать или сливать изменения между разработчиками? Решением этих задач является система контроля версий Git.

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, 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).

Один коммит — это пакет изменений, хранящий информацию с добавленными, отредактированными или удалёнными файлами кода. В один коммит принято добавлять не более десяти изменений — так получается длинная история версий, которая позволяет в случае ошибки откатиться с минимальной потерей работоспособного кода (см рис. 1 в приложении).

Таким образом при любой непредвиденной ситуации можно избавить себя от лишний мороки, начав все с последней работающей версии.

**1.3 Что такое HTML и зачем он нужен?**

Вся работа так или иначе начинается с идеи, которая перерастает в прототип, которая перерастает в дизайн-макет. Так как фронтенд-разработчику приходится совмещать все виды работ, он должен уметь верстать страницы. Для этого существует HTML.

HTML (Hyper Text 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 в интернете 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.

HTML сам по себе не является языком программирования, как может быть указано на различных ресурсах, однако это важнейший инструмент разметки с большим разнообразием тегов (см. рис. 2 в приложении). Все страницы, что можно увидеть в интернете – это HTML-страницы с подключенными к нему стилями CSS.

**1.4 CSS и Sass. Как раскрасить веб-сайт.**

В базовом виде HTML разметка не представляет из себя веб-сайты, которые мы привыкли видеть, потому что это не задача HTML. Чтобы заставить страницу выглядеть приемлемо нам нужен другой инструмент.

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 итоге должна выглядеть страница по принципу свойство: значение. Этих самых свойств или параметров в CSS огромное количество и можно управлять практически каждой визуальной составляющей: цвет, форма, тени, расположение, размер и даже несложная анимация. Можно сказать, что CSS3 перетянул на себя некоторый функционал JavaScript в плане анимации. Теперь, даже не зная языка программирования можно делать очень красивые проекты. Но прогресс не стоит на месте и появились такие вещи как препроцессоры, такие как Sass.

Sass — это метаязык (язык для описания другого языка), который упрощает и ускоряет написание CSS-кода. Его называют препроцессором. То есть у Sass свой удобный синтаксис, который генерируется в обычный CSS. Благодаря препроцессорам код стилей стал выглядеть еще более удобно, потому что даже в самом простом файле CSS может быть уйма правил.

Sass позволяет создавать переменные и использовать их в любом участке кода, делать вложенные стили, миксины (примеси - по своей сути тоже является переменной, но внутри нее целый участок кода CSS), использовать фрагментирование и множество других функций, упрощая разработку и делая ее очень удобной, а главное ёмкой и легкой для чтения.

**1.5 Javascript. Оживи свой сайт.**

JavaScript - это легкий, интерпретируемый, объектно-ориентированный язык программирования данных, основанный на прототипах и первоклассных функциях.

Изначально JavaScript был создан, чтобы описывать функционал для веб-страниц, тем самым делая его более «живым». Программы на этом языке называются скриптами. Они могут встраиваться в HTML и выполняться автоматически при загрузке веб-страницы. Скрипты распространяются и выполняются, как простой текст. Им не нужна специальная подготовка или компиляция для запуска, они обрабатываются непосредственно браузером внутренним «движком». С помощью сценариев JavaScript можно включать в статическую страницу динамические элементы, такие как карты, формы, числовые операции, анимацию, интерактивную инфографику и другое.

Преимущества JavaScript:

* скорость чтения и исполнения;
* понятный синтаксис;
* его не нужно компилировать - браузеры его интерпретируют;
* совместимость с широким спектром браузеров и платформ;

Сейчас JS не только браузерный язык, который изначально так задумывался. С появлением различных фреймворков, API, улучшений и созданием сотен функций, можно использовать JavaScript в мобильных приложениях, десктопном софте и даже в бэкенде.

Также с языком JS связаны популярные фреймворки по типу Vue или библиотека 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.

Еще существует библиотека jQuery, ныне устаревшая из-за более популярных фреймворков, но в свое время использовалась довольно часто.

Все это говорит о том, что большинство решений можно найти, даже хорошо не зная JavaScript, а изучать его по лекалам других разработчиков.

Вот мы и рассмотрели трех китов создания веб-сайта. Однако при написании кода нужны и другие инструменты, такие как отладчик в браузере.

**1.6 Отладка написанного кода в браузере**

Отладка – это процесс поиска ошибок и, говоря о наших «трех китах» веб-сайта, я имею в виду поиск ошибок непосредственно в HTML, CSS и JavaScript. Все браузеры имеют графический интерфейс отладчика (см рис. 3 в приложении), с помощью которого можно смотреть что и каким образом обрабатывается браузером, что сильно упрощает процесс разработки. Чем же отладочная панель так полезна?

* Можно посмотреть, как элементы выглядят в HTML файле;
* Какие стили работают для конкретного файла или всего сайта;
* Какие ресурсы загружает браузер;
* Есть встроенная консоль, помогающая с работой в JS и не только;
* Можно пошагово отследить работу исполняемого скрипта;

При разработке к ней обращаются очень часто. В некоторых ситуациях можно написать html или css прямо в отладочной панели, чтобы отследить как задумка будет работать и будет ли работать вообще.

**1.7 Фреймворк Laravel**

Многофункциональный фреймворк с открытым исходным кодом, который использует за основу язык программирования PHP. Во фреймворке построение архитектуры разрабатывается по принципу Model View Controller (MVC). Такую архитектуру или паттерн, если угодно, использует, например, Angular JS (см. рис. 4 в приложении)

Как же работает архитектура MVC? По сути, это три сущности, каждая из которых отвечает за определенный аспект работы и при этом они тесно связаны между собой.

* Model (модель) – данные и методы работы с этими данными (проверки и запросы в базу данных)
* View (представление) – внешний вид. То, с чем взаимодействует конечный пользователь.
* Controller (контроллер) – устанавливает взаимосвязь между представлением и моделью. Направляет данные от пользователя к системе.

Пользователь взаимодействует с представлением или интерфейсом и у него есть некие варианты управления: отправка форм, нажатие кнопок, переходы по ссылкам и прочее. Это взаимодействие как раз и обрабатывается контроллером. Контроллер в свою очередь взаимодействует с моделью, сообщая ему о поступивших изменениях или запросах. А результаты этих изменений снова попадают в представление, сообщая пользователю итог.

Преимущества Laravel

* Открытый исходный код. Это значит, что можно досконально узнать и разобрать, как и что работает.
* Безопасность. Собственная технология программирования связей базы данных и языка Eloquent, в которую встроена защита от SQL инъекций. Также есть возможность экранировать теги, что позволяет организовать дополнительную защиту от XSS-атак. Есть встроенная функция генерации CSRF токена для каждой сессии и многое другое.
* Мультиязычность. Возможность использовать несколько языков для проекта.
* Масштабирование. Огромное количество пакетов, которые пригодятся для решения различных задач, которые могут возникнуть в будущем.

Недостатки:

* Обновления и совместимость. При обновлении Laravel могут возникнуть различные проблемы с пакетами и зависимостями, на решение которых может уйти приличное количество времени.
* Зависимость от сторонних пакетов. Как и в предыдущем пункте, если какой-то из пакетов перестанет поддерживаться или будут проблемы с зависимостями, нужно будет потратить какое-то время на решение этих проблем.
* Объем кода. Имея такой огромный потенциал к масштабированию, можно предположить какой объем кода может содержаться в этом проекте, что в конечном итоге может повлиять на производительность.

Но недостатки есть в каждом фреймворке, поэтому несмотря ни на что laravel остается одним из самых популярных.

**1.8 Структура каталогов Laravel.**

При установке Laravel приложения можно столкнуться с довольно внушительным каталогом файлов (см. рис 5 в приложении). Всю структуру можно настроить как нравится конкретному разработчику, так как Laravel не накладывает на это ограничений, однако Composer должен иметь возможность автоматически загружать классы.

Каталог app содержит основной код все классы архитектуры MVC будут находиться в нем.

Каталог bootstrap содержит файл app.php, который загружает фреймворк. В этом каталоге также находится каталог cache, содержащий файлы, сгенерированные фреймворком для оптимизации производительности, например файлы кеша маршрутов и служб.

Каталог config содержит все файлы конфигурации приложения.

Каталог database содержит миграции баз данных, фабрики моделей и наполнители.

Каталог public содержит файл index.php, который является точкой входа, также сюда размещаются стили, картинки и скрипты.

Каталог resources содержит шаблоны, о них в следующей главе.

Каталог routes содержит все определения маршрутов. Простыми словами ссылки сайта и их обработка (тело функции).

В файле console.php можно определить анонимные консольные команды и задачи.

Каталог storage содержит журналы (логи), скомпилированные шаблоны Blade, файлы сессий, кеша и другие файлы, созданные фреймворком.

Каталог tests содержит автоматизированные тесты. Примеры модульного Pest или PHPUnit и функционального тестов предоставляются из коробки.

Каталог vendor содержит ваши Composer-зависимости.

**1.9 Blade шаблоны**

Blade – это движок шаблонов, входящий в состав Laravel. Можно сказать, что это html структура с вкраплением PHP кода, который исполняется при работе приложения. На самом деле, все шаблоны Blade компилируются в обычный PHP-код и кешируются до тех пор, пока не будут изменены, что означает, что Blade добавляет фактически нулевую нагрузку приложению. Файлы шаблонов Blade используют расширение файла .blade.php и хранятся в каталоге resources/views.

**Приложения**.

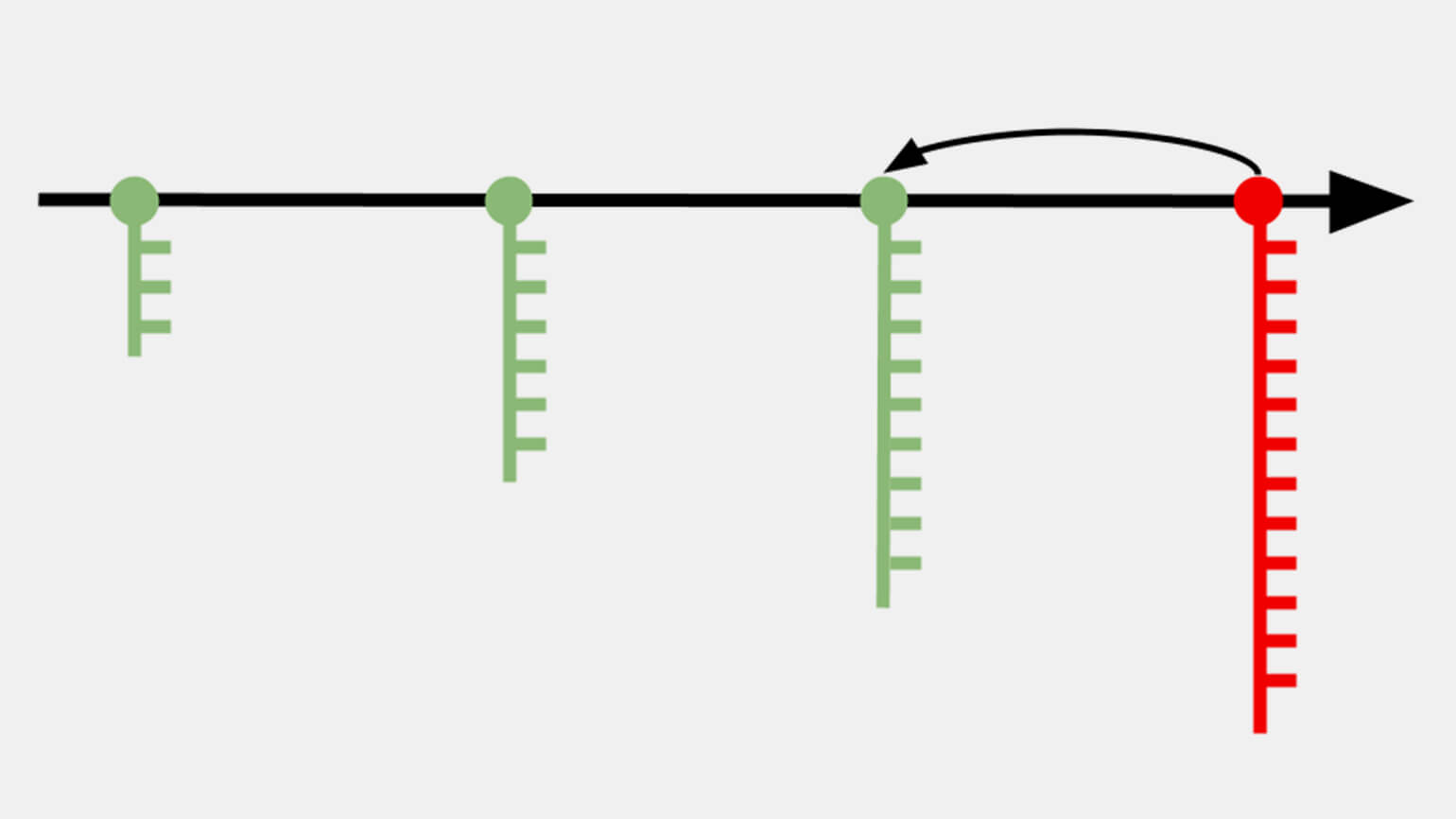


Рисунок 1 – Схема контроля версий

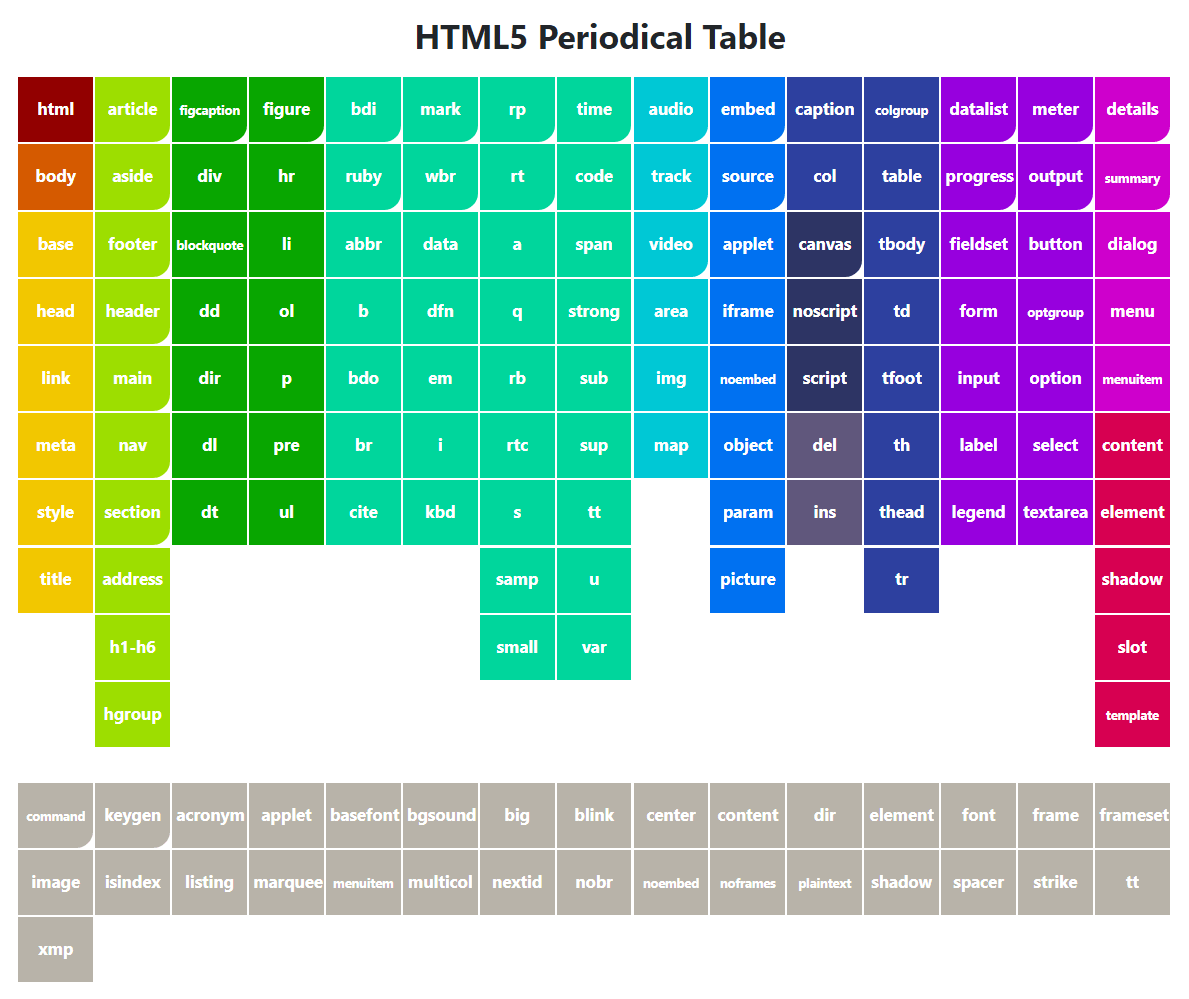


Рисунок 2 - Разнообразие HTML тегов

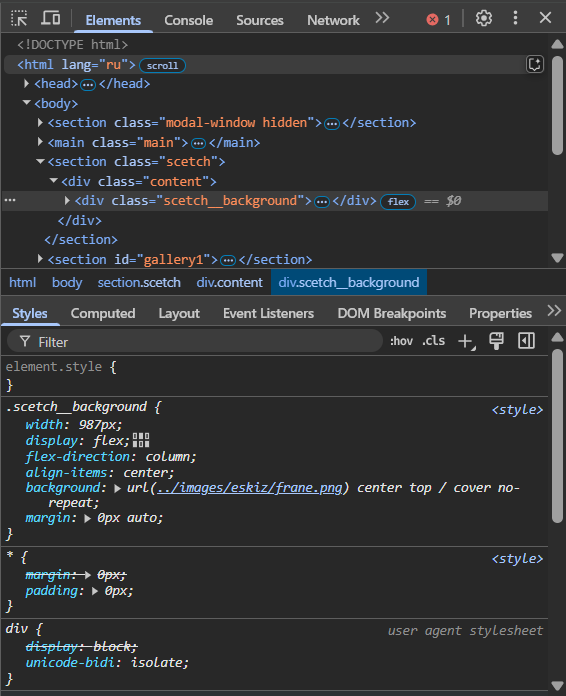


Рисунок 3 – Отладочная панель в браузере Chrome

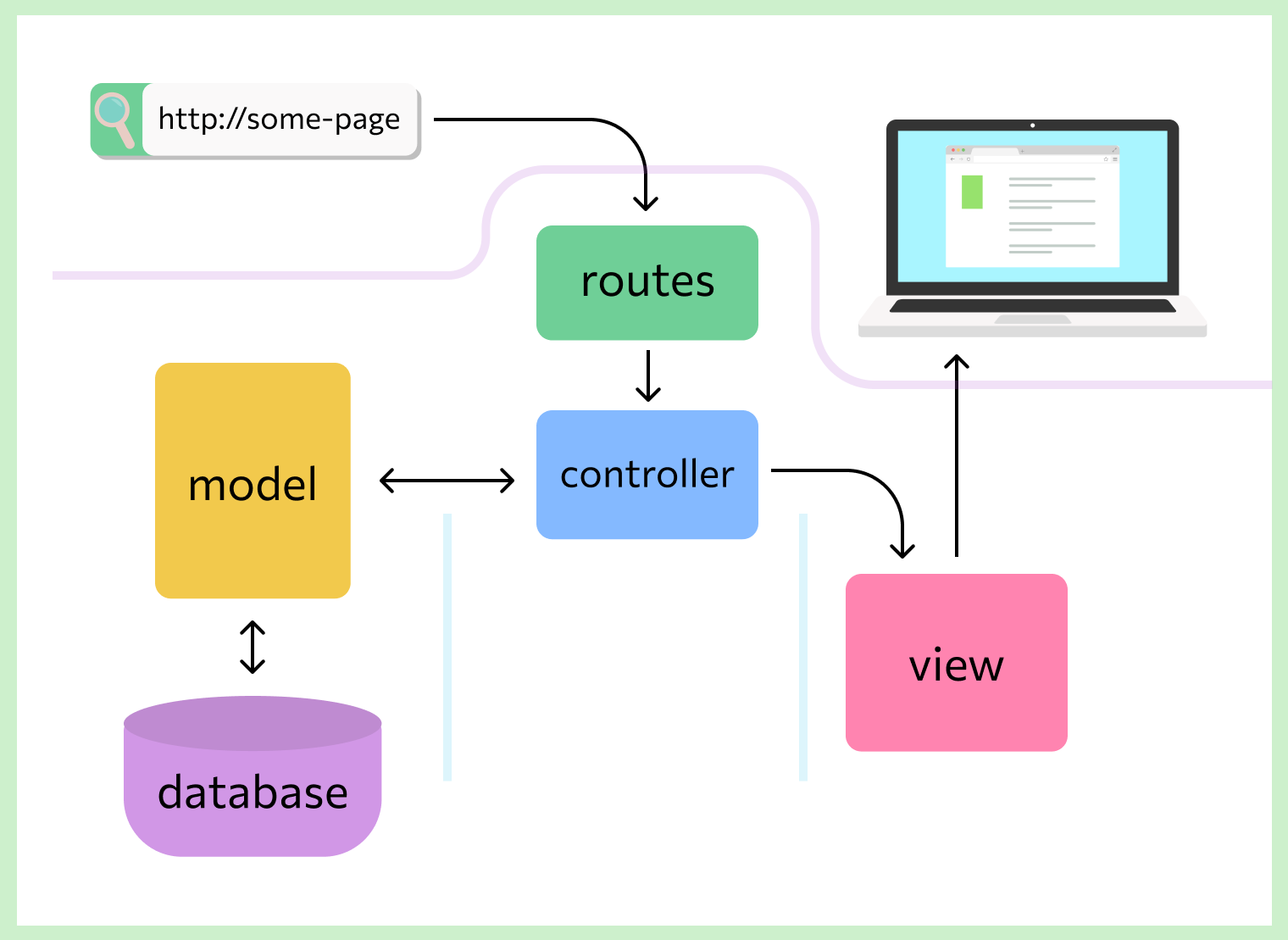


Рисунок 4 – Архитектура MVC

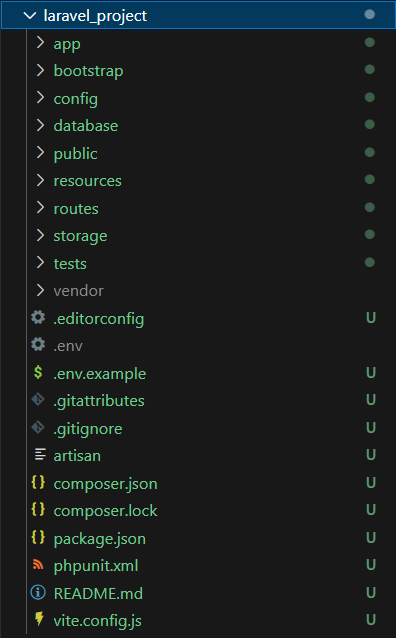


Рисунок 5 – Стандартный каталог файлов laravel