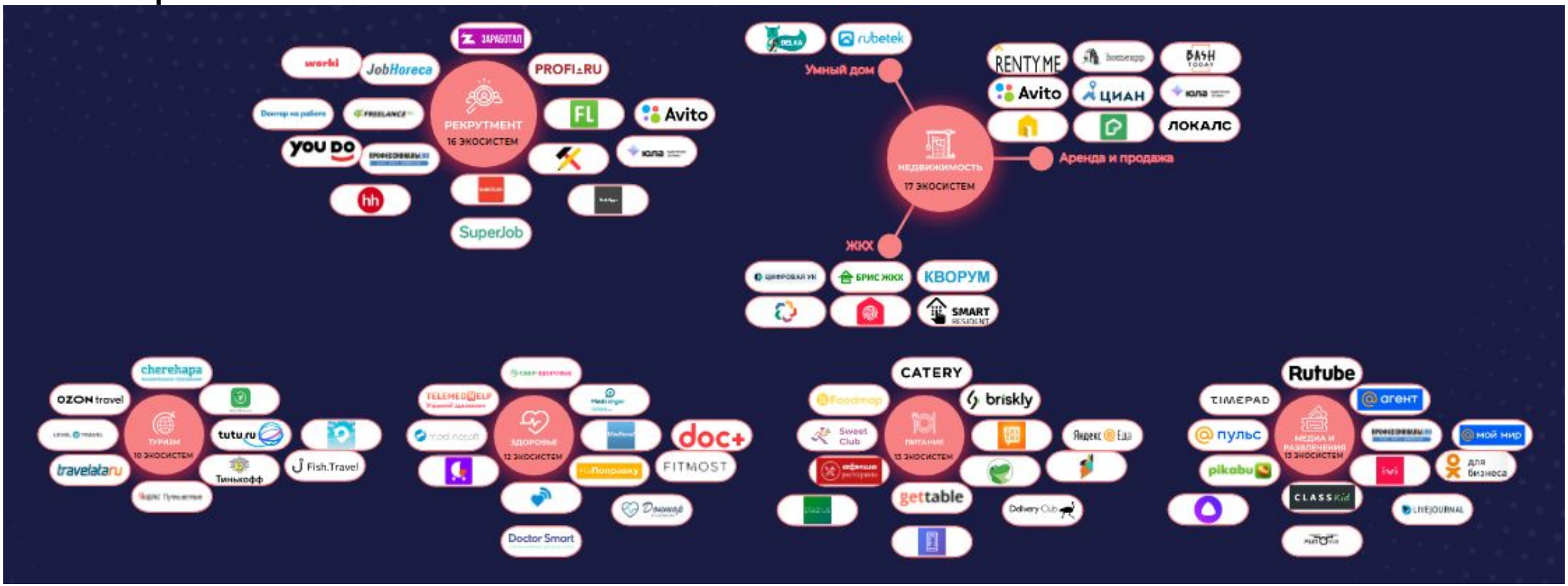
Экосистема — это комплексный проект, объединяющий множество участников, информационных сервисов и бизнес-процессов по принципу win-win. Она может развиваться как вокруг одной компании и услуги, так и в виде маркетплейса со множеством игроков. В обоих сценариях важно, чтобы все процессы были взаимосвязаны, а все участники экосистемы — в выигрыше.



Фреймворк (framework) - готовый каркас с набором модулей, компонентов, расширения для быстрой, простой и качественной разработки программ.

PHP фреймворки позволяют:

- ускорить процесс разработки веб-приложений
- помогают писать простой и качественный код
- повторно использовать код в проектах
- легко масштабировать проекты
- использовать современные практики программирования
- легче тестировать программный код
- обеспечить безопасность проекта

Laravel
Yii
Symfony
Zend Framework
Codeigniter

Laravel - это бесплатный PHP фреймворк с открытым исходным кодом, созданный Тейлором Отвеллом для разработки веб-приложений по архитектурному шаблону MVC.

Фреймворк Laravel очень популярен среди западных разработчиков веб-приложений.

С помощью менеджера пакетов Composer, фреймворк Laravel позволяет легко устанавливать и подключать различные компоненты для использования в веб-приложении.

Реализация шаблона ActiveRecord - Eloquent ORM, позволяет установить отношения между объектами базы данных веб-приложения и выстраивать удобные запросы для манипуляции данными.

Механизм автозагрузки классов позволяет не подключать вручную файлы через include и предотвращает загрузку не используемых компонентов.

Удобная система миграций помогает упростить развертывание и обновление веб-приложения.

При создании приложения можно использовать Artisan - интерфейс командной строки для ввода встроенных команд, а также создания своих собственных.

В Laravel есть много полезных функций, позволяющих сделать процесс разработки веб-приложений быстрым, простым и качественным.

Из подобных функций можно отметить dd() - удобный аналог стандартной функции PHP var_dump(). Функция выводит информацию переменной в более понятной форме, разделяя данные на дерево атрибутов и значений, в возможностью поиска и перехода по ним.

Плюсы

Достаточно неплохая и понятная документация.

Этот фреймворк имеет мощную экосистему. Различные курсы, конференции, обучающие материалы позволяют собрать вокруг фреймворка большое количество разработчиков и спонсоров, которые заинтересованы в развитии инструмента и принимают в этом участие. Да, здесь чувствуется запах маркетинга, и неплохой.

Одним из самых очевидных плюсов Laravel, является гибкая система маршрутизации, позволяющая составить самые разные проверки маршрута веб-приложения. Вы можете выделить маршруты в специальные группы, использовать пространство имен, указать параметры маршрута, использовать регулярные выражения, настроить поддоменную маршрутизацию и многое другое.

В Laravel много синтаксического сахара. Синтаксис API фреймворка достаточно простой и понятный. Здесь нет длинных и сложных конструкций, а только краткие и продуманные названия функций.

Laravel содержит удобный механизм обработки ошибок и исключений.

Фреймворк включает в себя встроенные механизмы аутентификации и авторизации пользователей, которую можно перенастроить под свои потребности.

Laravel предоставляет чистый и простой API поверх популярной библиотеки SwiftMailer с драйверами для SMTP, Mailgun, SparkPost, Amazon SES и sendmail, чтобы сделать отправку почты через локальную или облачную службу по выбору. В том числе есть механизм для построения очередей отправки почты.

Laravel Cashier обеспечивает выразительный, свободный интерфейс к сервисам биллинга по

Минусы

Для русскоязычных разработчиков, без знания английского языка или его слабым знанием, к минусам фреймворка можно отнести довольно небольшое количество статей, примеров кода, переводов официальной документации. Для тех, кто знает английский на уровне чтения технической документации, данный минус можно опустить.

Синтаксический сахар в Laravel как плюс, так может быть и минусом. Очень легко привыкнуть к нему и позабыть, как пишутся чистые запросы и функции.

Нарушение обратной совместимости между версиями фреймворка.

Не логичное расположение каталогов и файлов. Например, по умолчанию в прямо в каталоге /app расположена модель User.php, которую логичней было бы расположить в каталоге /app/Models. Каталог resources с файлами представления размещен в корне приложения, хотя логичней будет его разместить в /app/resources.

Yii - это бесплатный объектно-ориентированный компонентный full-stack PHP фреймворк. В основе Yii лежит другой фреймворк - PRADO, написанный на ASP.NET и впоследствии перенесенный на PHP. Вскоре после построения новой архитектуры, фреймворк PRADO был переименован на Yii. Название фреймворка является аббревиатурой слова «Yes It Is!». Прародителем фреймворка является китайский разработчик Qiang Xue.

Yii можно использовать для разработки любого вида веб-приложений. Благодаря своей основе компонентов, архитектуре и сложной поддержки кэширования, фреймворк подходит для разработки крупномасштабных проектов, таких как порталы, форумы, системы управления контентом (CMS), систем электронной коммерции, RESTful веб-сервисов и т.д.

Yii реализует для использования MVC (Model-View-Controller) архитектурный шаблон и способствует организации кода на основе этого шаблона.

Yii является full-stack фреймворком, предоставляя множество проверенных и готовых к использованию функций: построитель запросов и ActiveRecord для реляционных и NoSQL баз данных, RESTful API, поддержку многоуровневого кэширования и т.п.

Yii расширяемый фреймворк в котором можно заменить почти каждый кусочек кода и разрабатывать нужные расширения.

Yii предоставляет Gii - визуальный интерфейс для автоматической генерации контроллеров, моделей и отображений.

Плюсы

Для русскоязычных разработчиков большим плюсом фреймворка, является хорошая документация, множество статей с примерами кода и сообщество.

Yii не показатель одного человека, фреймворк подкреплен сильной командой разработчиков ядра, а также большим сообществом профессионалов постоянно способствующих развитию Yii.

Yii способствует быстрому прототипированию веб-приложения. Он относится к инструментам RAID разработки.

Встроенный механизм создания виджетов представления, например, для размещения на сайте различных блоков: последние посты, категории, навигация, блоки рекламы и т.п.

Компонент приложения i18n позволяет производить автоматический перевод сообщений веб-приложения.

Встроенная поддержка автоматической валидации форм и вывода сообщений об ошибках на основе данных из моделей веб-приложения.

Механизм Active Record для построения реляционной обработки запросов базы данных.

Множество готовых расширений на Github и их установка через Composer

Встроенные виджеты для отображения данных: DetailView (строки в таблице), ListView (Список), GridView (таблицы)

Встроенные механизмы для аутентификации, авторизации, регистрации пользователей

Содержит встроенную и очень удобную debug панель.

Минусы

Слабая экосистема вокруг фреймворка среди англоязычного сегмента разработчиков. Сообщество слишком сильно размазано по разным местам: несколько форумов с небольшой активностью, stackoverflow и т.п.

Хоть фреймворк и позволяет делать код простым, но далеко не элегантным. Если его синтаксис сравнивать с фреймворком Laravel, то он уступает.

Yii отстает от языка, стандартов и других фреймворков. Новые обновления с действительно полезными функциями выходят не так часто.

Слишком большая связанность backend и frontend частей Yii2. Фреймворк предлагает использовать библиотеку jQuery и Bootstrap, которые встроены по умолчанию в ядро фреймворка. Этот минус планируется исправить в новой версии фреймворка Yii 3, сделав его компоненты менее связанными.

Не очень гибкая система маршрутизации: нет возможности сгруппировать роуты.

Symfony - свободный PHP фреймворк для быстрой разработки веб-приложений и решения рутинных задач веб-программистов. Разработка и поддержка фреймворка спонсируется французской компанией Sensio.

Symfony состоит из набора не связанных между собой компонентов, которые можно использовать повторно в проектах.

С помощью Symfony было разработано множество крупных проектов:

- систем управления контентом: Magento, Drupal, Opencart
- сервис социальных закладок Delicious
- французский видеохостинг Dailymotion
- движок форума phpbb

В том числе, Symfony повлиял на разработку фреймворка Laravel, где были задействованы его компоненты.

Symfony позволяет устанавливать сторонние пакеты, библиотеки, компоненты и настраивать их с помощью конфигурации в форматах YAML, XML, PHP, а также .env файлах.

Symfony не обеспечивает компонент для работы с базой данных, но обеспечивает тесную интеграцию с библиотекой Doctrine.

Symfony предоставляет функцию почтовой программы на основе популярной библиотеки Swift Mailer. Эта почтовая программа поддерживает отправку сообщений с ваших собственных почтовых серверов, а также с использованием популярных почтовых провайдеров, таких как Mandrill, SendGrid и Amazon SES.

Механизм интернационализации позволяет установить и произвести перевод сообщений веб-приложения на основе выбранного языка или страны.

Плюсы
Мощная экосистема вокруг фреймворка, с хорошим сообществом и множеством разработчиков.

Хорошая и постоянно обновляемая документация для всех версий фреймворка.

Множество различных не связанных компонентов для повторного использования.

Предлагает механизм функциональных и модульных тестов для нахождения ошибок в веб-приложении.

Подходит для сложных и нагруженных веб-проектов. Электронной коммерции.

Минусы

Несмотря на хорошую документацию, фреймворк является сложным для изучения.

Zend Framework - это свободный объектно-ориентированный PHP фреймворк для разработки веб-приложений, разработанный и поддерживаемый компанией Zend. Данный фреймворк как правило больше всего используют при разработке крупных коммерческих проектов.

Основным спонсором Zend Framework является Zend и компания Rogue Wave , но многие другие внесли компоненты или важные функции в платформу. Такие компании, как Google, Microsoft и StrikeIron, сотрудничают с Zend для предоставления интерфейсов для веб-сервисов и других технологий, которые они хотят сделать доступными для разработчиков ZF.

Компания Zend участвует в разработке ядра языка программирования PHP.

В качестве менеджера зависимостей пакетов Zend Framework использует Composer. Для тестирования веб-приложения применяется PHPUnit, а для непрерывной интеграции служба Travis CI.

Zend Framework следует стандартам PHP-FIG и включает реализацию PSR-7 для интерфейсов HTTP-сообщений.

Поддержка множества баз данных: MariaDB, MySQL, Oracle Database, IBM DB2, Microsoft SQL Server, PostgreSQL, SQLite и Informix.

Гибкий механизм кэширования по памяти или файловой системы.

Плюсы
Отлично подходит для разработки коммерческих веб-приложений.
Объектно-ориентированный подход к разработке.
Несвязанные компоненты для повторного использования в проектах.

Минусы
Не подходит для быстрой разработки проектов.
Для русскоязычного сегмента разработчиков мало полезных материалов по разработке.

CodeIgniter - это популярный PHP микро-фреймворк с открытым исходным кодом, для разработки веб-систем и приложений. Разработан компанией EllisLab, а также Риком Эллисом и Полом Бурдиком.

В CodeIgniter компоненты загружаются и процедуры выполняются только по запросу, а не глобально. Система не делает никаких предположений относительно того, что может потребоваться помимо минимальных основных ресурсов, поэтому система по умолчанию очень легкая.

Компоненты фреймворка слабо связаны между собой и не зависят друг от друга. Чем меньше компонентов зависит друг от друга, тем более гибкой и многоразовой становится система.

Хотя CodeIgniter работает довольно быстро, объем динамической информации, отображаемой на страницах, будет напрямую зависеть от используемых ресурсов сервера, памяти и циклов обработки, которые влияют на скорость загрузки страниц.

Поэтому CodeIgniter позволяет кэшировать страницы для достижения максимальной производительности с помощью встроенного компонента кэширования.

CodeIgniter послужил основой в разработке новых фреймворков: Kohana и Rain Framework. Многие идеи CodeIgniter были применены во фреймворках Fuel PHP и CodeLighter.

Плюсы

Отличная документация и англоязычное сообщество.
Высокая производительность фреймворка.
Небольшой размер фреймворка.
Предоставляет легкие и простые решения для разработки.
Подходит для быстрой разработки небольших сайтов и веб-приложений.
Структура фреймворка не требует строгих правил кодирования.
Не требует сложной настройки, почти нулевая конфигурация.
MVC-архитектура веб-приложения.
Слабая связанность компонентов.
Множество подключаемых библиотек и помощников.

Минусы

Долгий застой в развитии Codeigniter 3. Сейчас перешел к новым владельцам Технологическому институту Британской Колумбии (British Columbia Institute of Technology — BCIT) и находится в стадии разработки новой версии - CodeIgniter 4.

Системные требования

Фреймворк Laravel 5.3 требует:

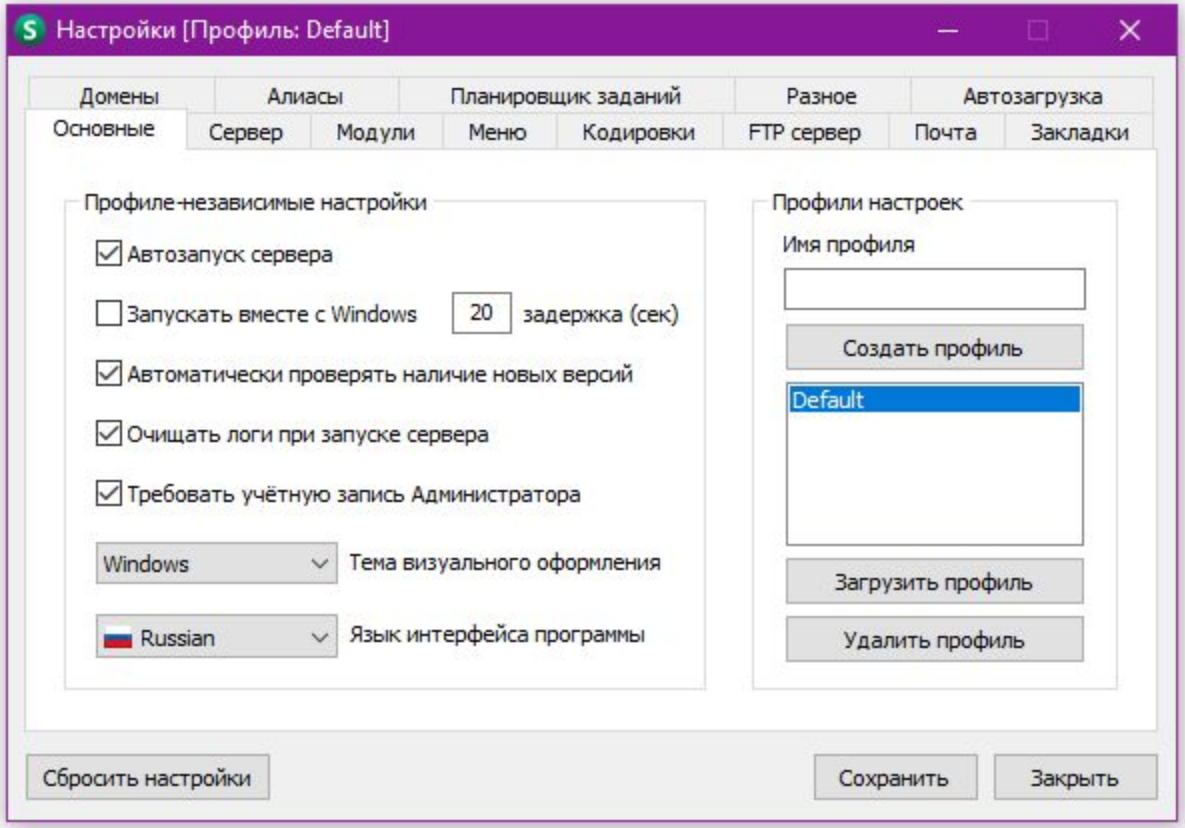
PHP >= 5.6.4
OpenSSL PHP Extension
PDO PHP Extension
Mbstring PHP Extension
Tokenizer PHP Extension

Весь этот набор компонентов присутствует в OpenServer под Windows.

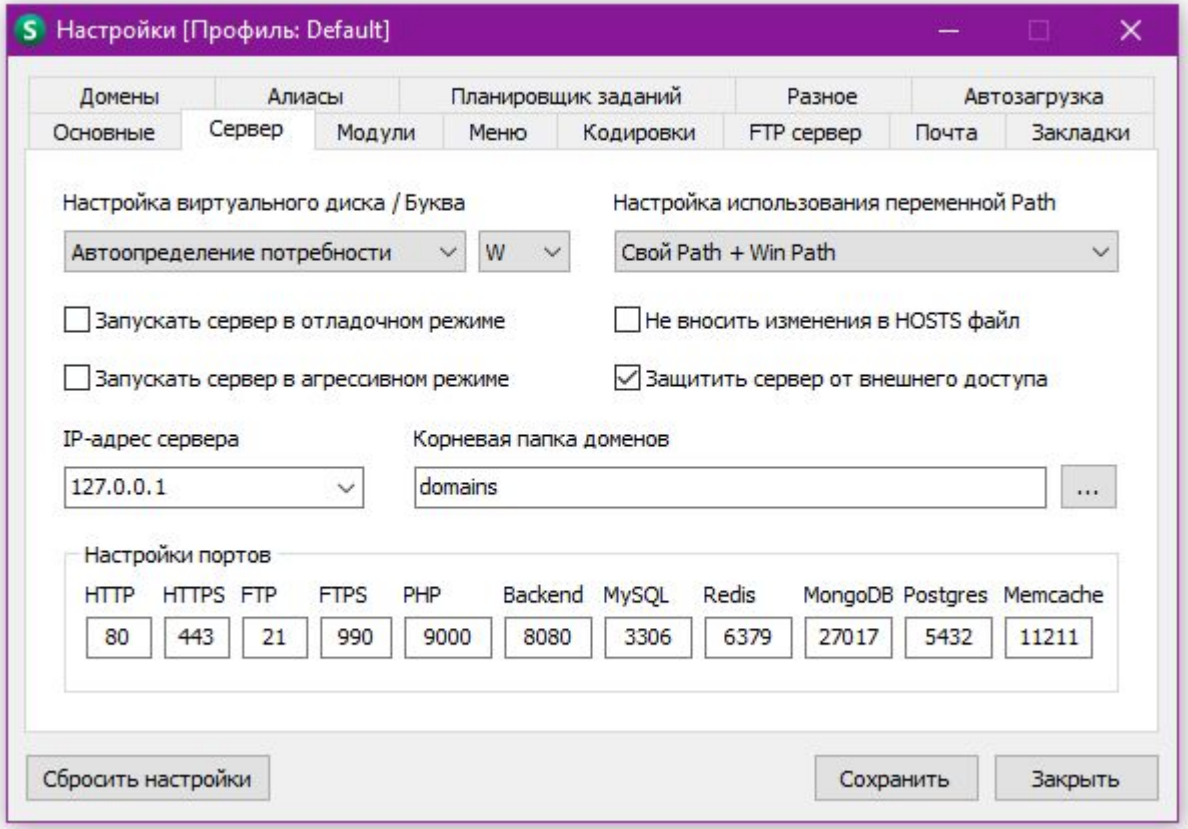
Также необходим Composer и NodeJS.

Настройка
OpenServer

Основные
На вкладке «Основные» включаем автозапуск сервера и требование учетной записи Администратора, она нужна при работе с файлом хостов.



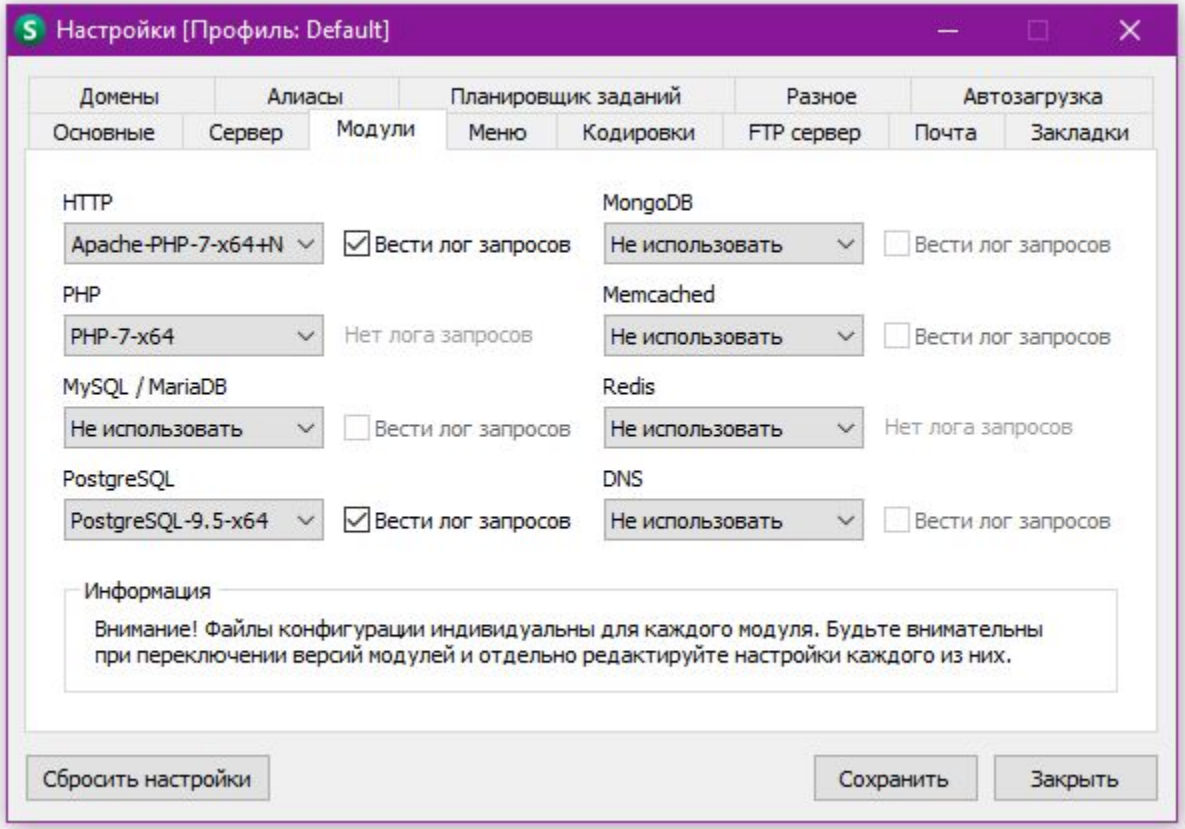
Сервер
На этой вкладке устанавливаем настройки виртуального диска в «Автоопределение потребности», далее настройку использования переменной Path в «Свой Path + Win Path».
Включаем свойство «Защитить сервер от внешнего дос



Настройка
OpenServer

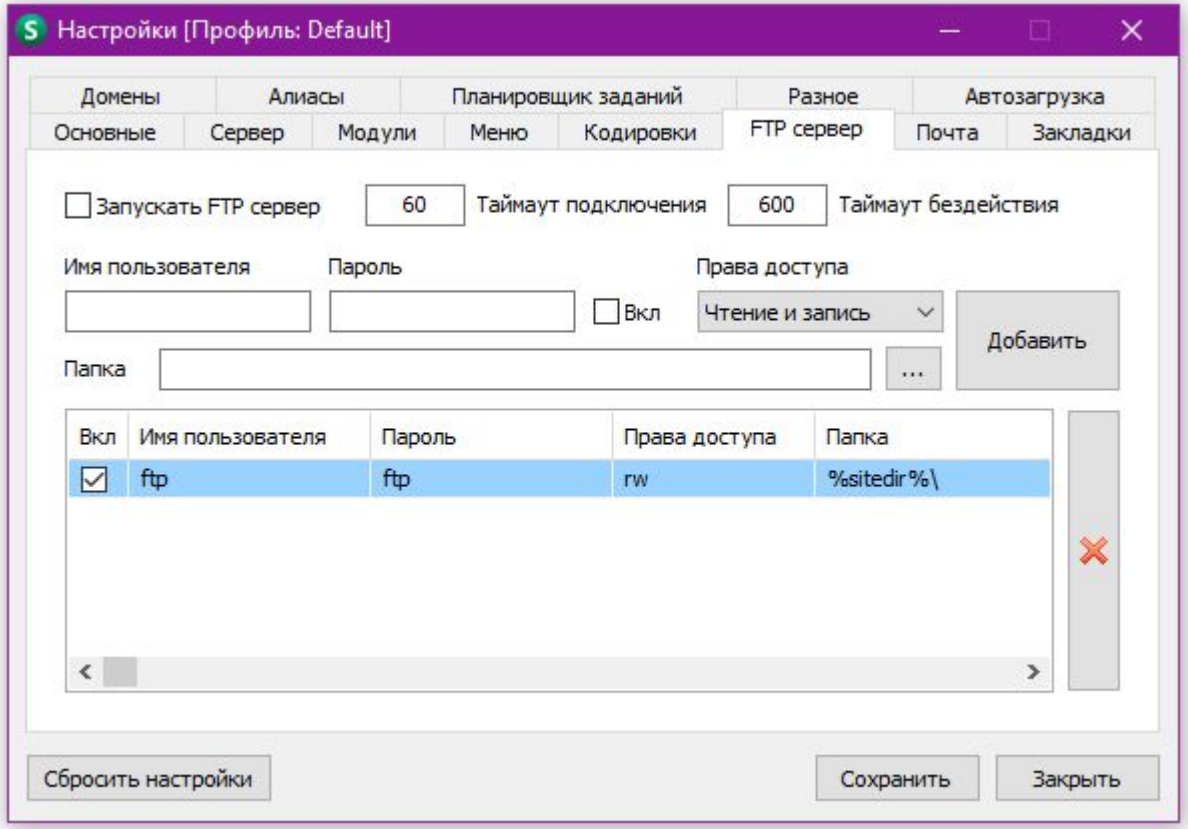
Модули

Используем следующую связку модулей:
HTTP: Apache — PHP 7 x64 — NGINX 1.10
PHP: PHP 7 x64
PostgreSQL 9.5 x64
Redis 3.0
Остальные пункты надо указать «не использовать».



FTP-сервер

На этой вкладке, выключаем свойство запуска FTP-сервера. Т.к. используем локальный сервер.



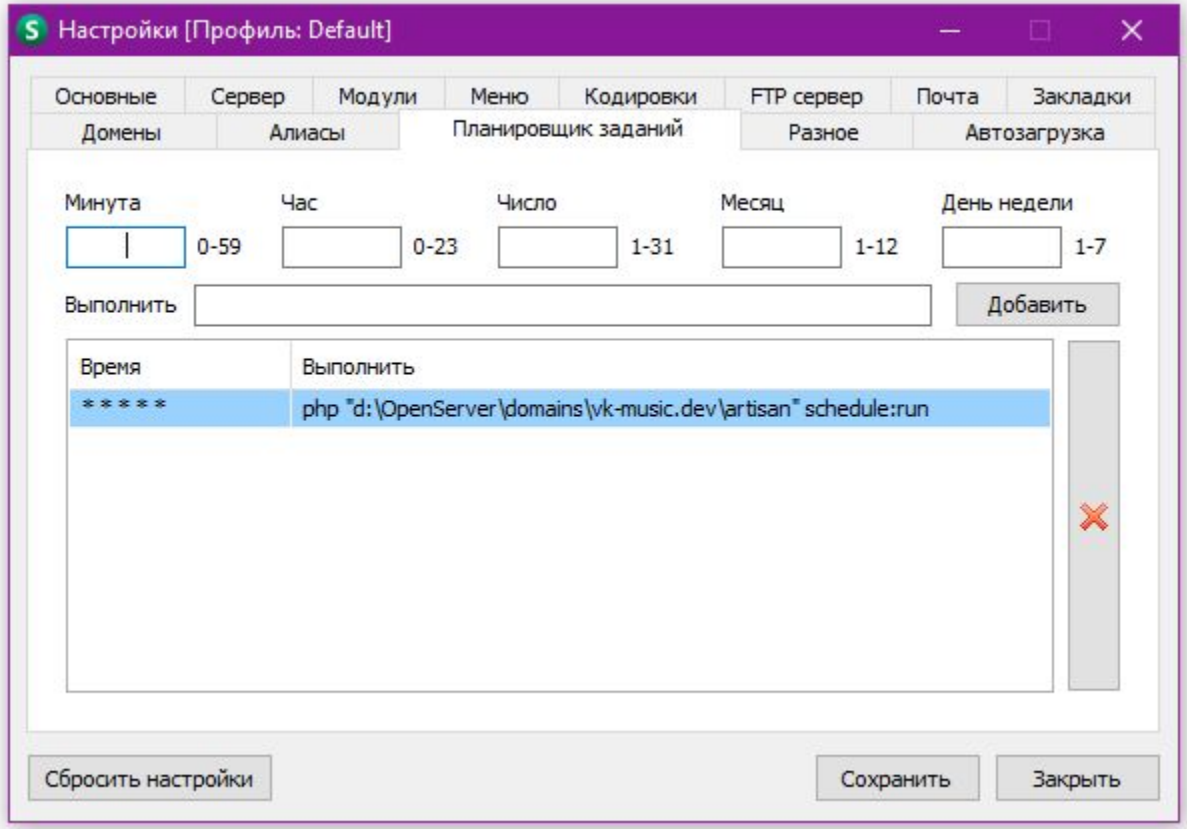
Настройка OpenServer

Планировщик задания (Cron)

Во все значения времени ставим символ «звездочки» (*), а в графу «выполнить»:

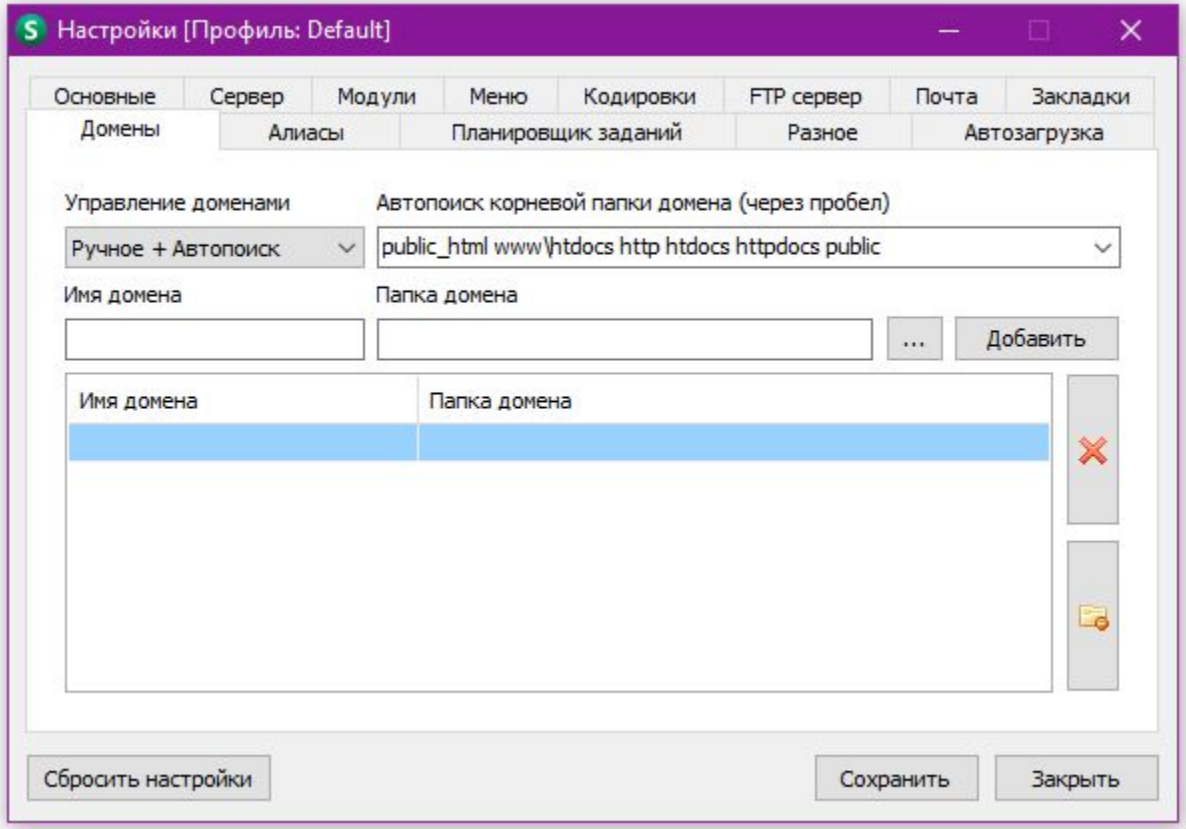
php "D:\OpenServer\domains\mysite.com\artisan" schedule:run

Где указываем полный путь к файлу «artisan» в Вашем проекте.



Домены

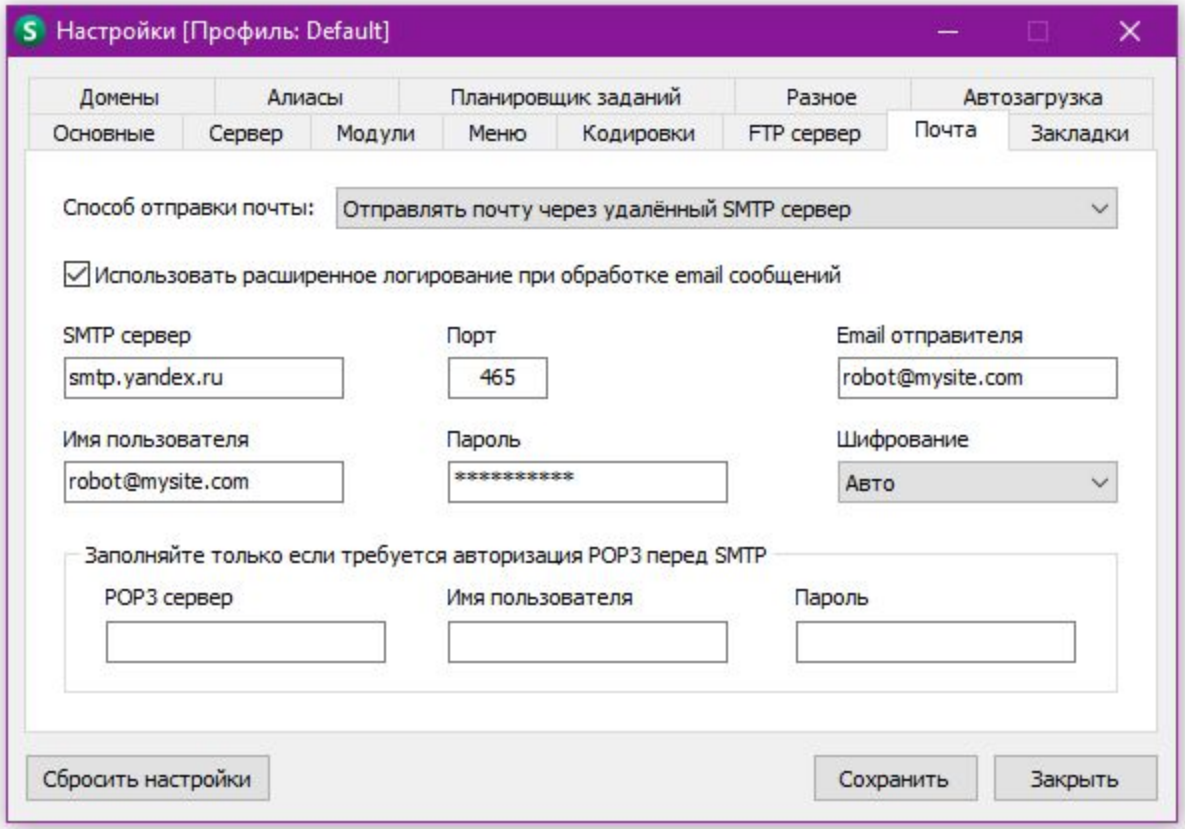
Для того, чтобы каждый раз руками не добавлять новый сайт, в списке «Управление доменами» выбираем «Ручное + Автопоиск», а в правой части указываем «public», т.к. в ней и находится исполняемый файл.

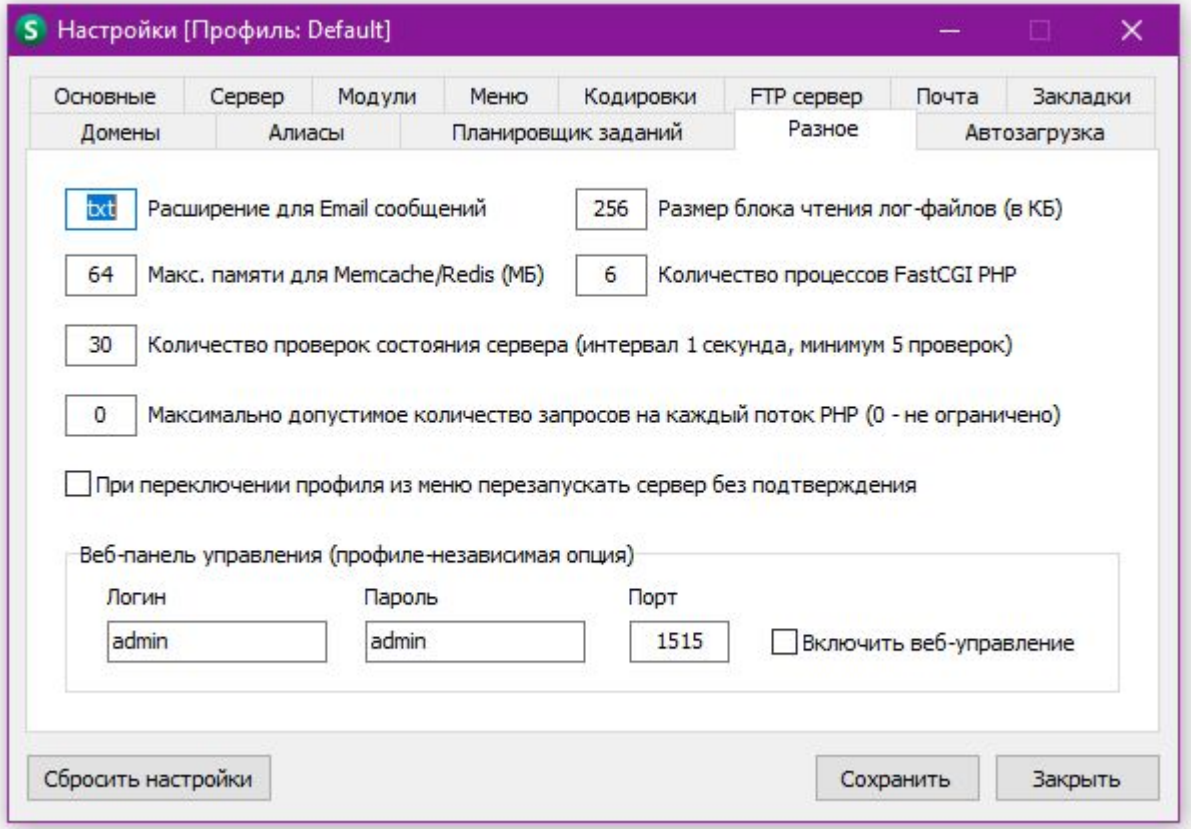


Настройка OpenServer

Почта

По умолчанию, все сообщения сохраняются локально в папку, но для удобства проверки устанавливаем: способ отправки через SMTP, далее сервер, порт, email-отправителя, имя пользователя (логин), пароль от учетки, шифрование в «авто».





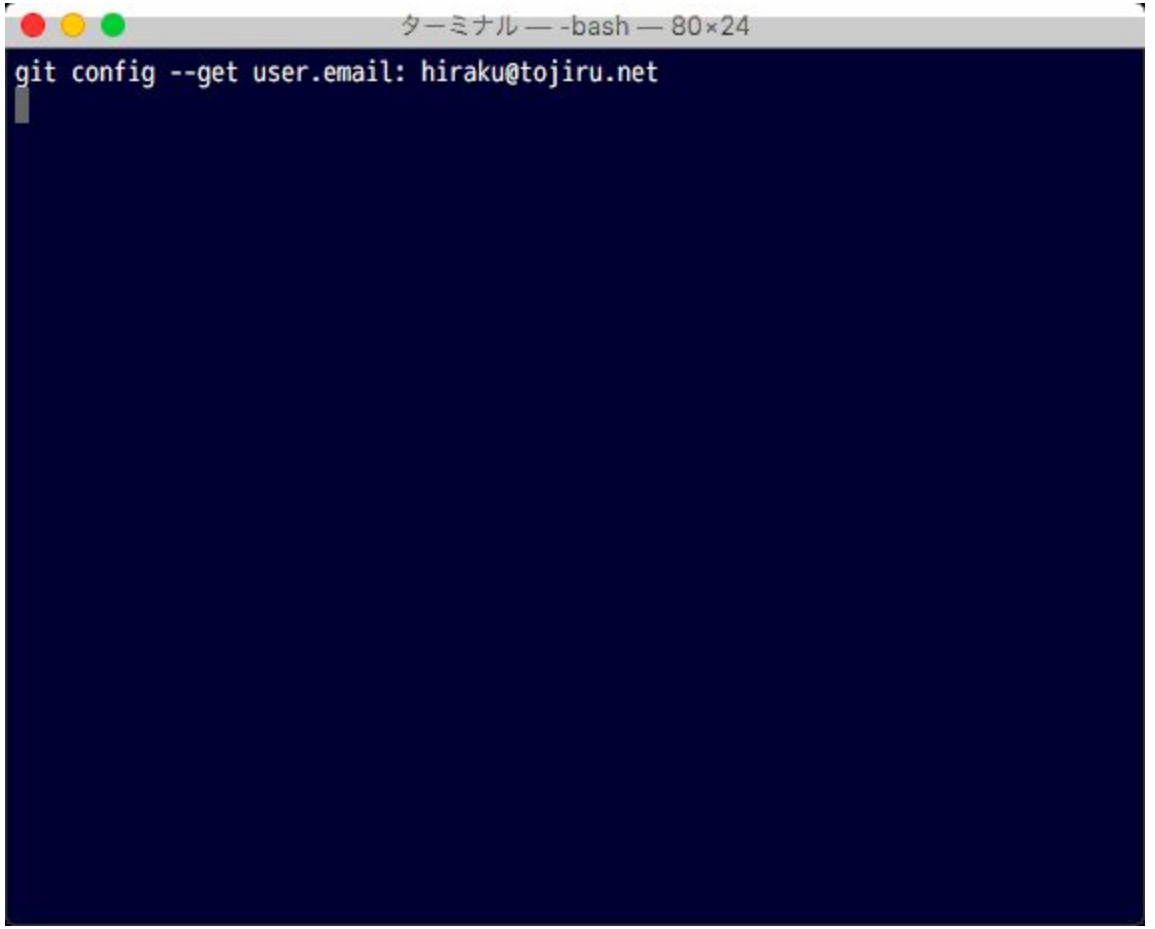
Установка Laravel 5.3

Перед началом развертывания фреймворка убедитесь, что на компьютере установлены утилиты Composer и NodeJS.

Перед установкой фреймворка, рекомендую воспользоваться пакетом hirak/prestissimo, позволяющим загружать несколько пакетов при установке/обновлении фреймворка одновременно. Без него процесс установки/обновления линейный, то есть запрашивает один пакет, ждет ответ, после скачивает (если ответ от сервера успешный), далее переходит к следующему. Пакет `hirak/prestissimo` же позволяет одновременно скачивать все необходимые пакеты и уже после этого начинает их ставить.

Для установки пакета выполните в консоли:

composer global require "hirak/prestissimo:^0.3"



Итак, открываем командную консоль и вводим:

composer global require "laravel/installer"

Это нужно делать всего один раз при самой первой установке. При разворачивании последующих приложений, данную команду пропускаем.

После того, как скачали установщик фреймворка в глобальное хранилище *composer*, нужно скачать сам фреймворк Laravel 5.3. Для этого у нас есть две команды :

laravel new blog

и

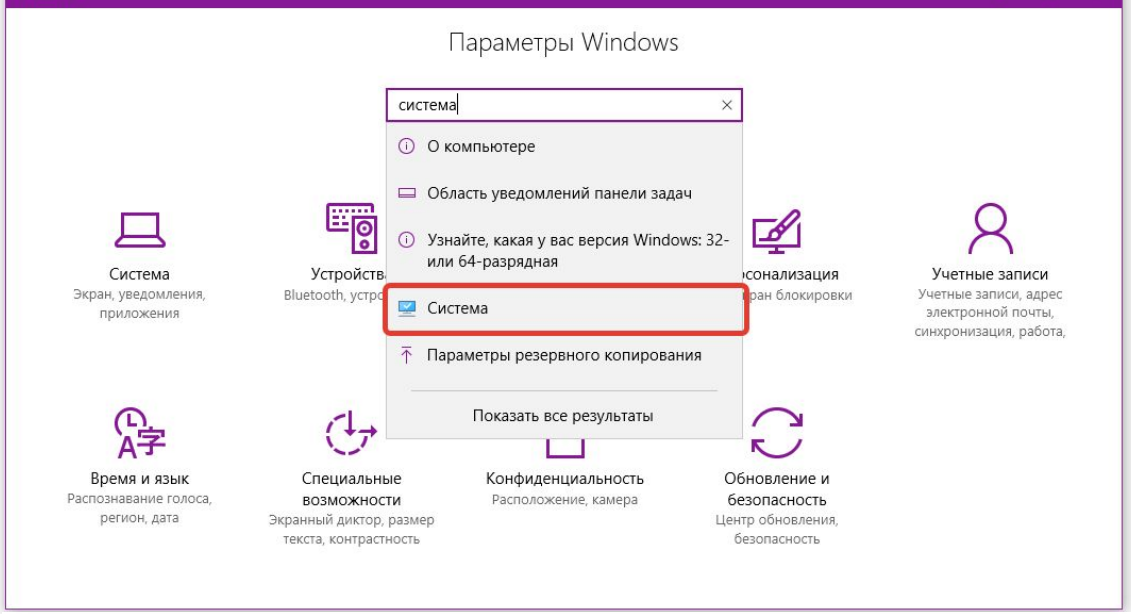
composer create-project --prefer-dist laravel/laravel blog

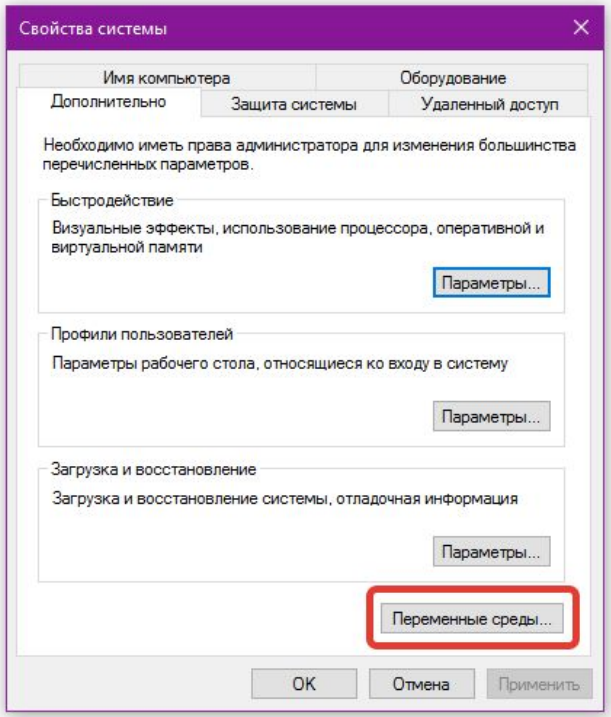
Они обе выполняют одно и то же действие

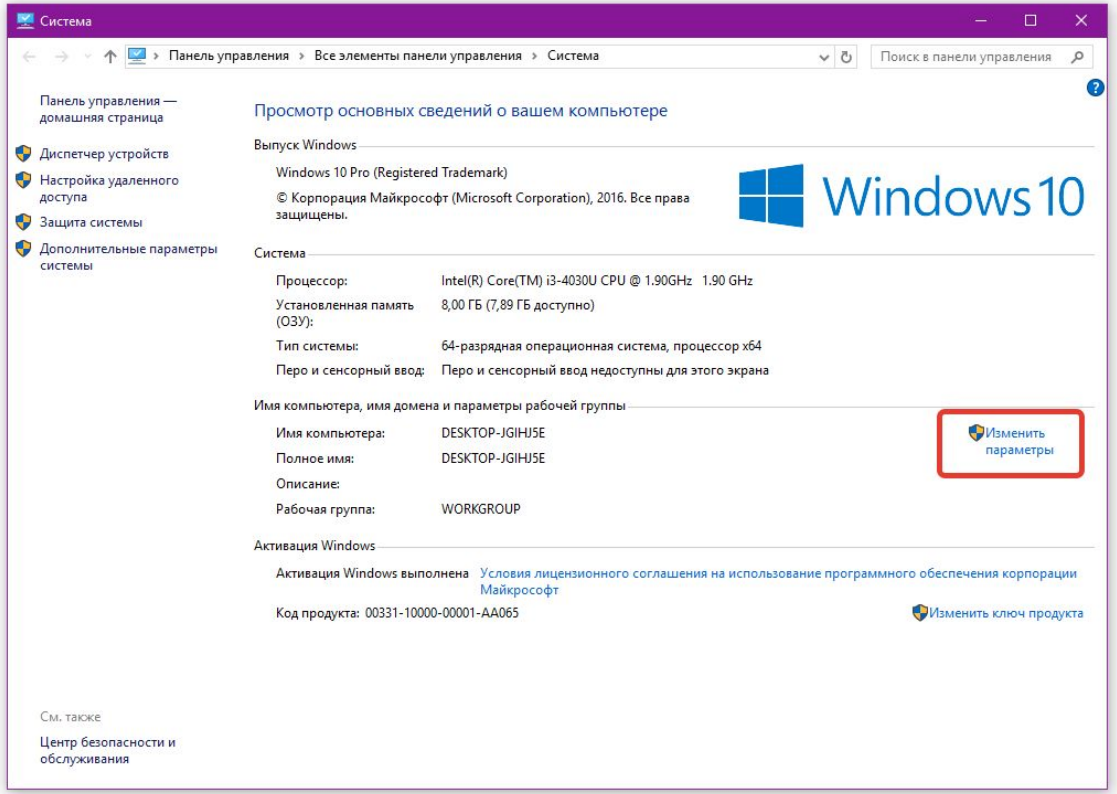
Установка Laravel 5.3

При первой установке путь к исполняемому файлу *composer* прописывается автоматически при его установке, а путь к команде «*laravel*» — нет. Поэтому для использования команды «*laravel*», необходимо откорректировать параметры операционной системы.

Открываем «Параметры» и в поисковой строке вводим «Система», далее «Изменить параметры». В окне «Свойства системы» переходим во вкладку «Дополнительно» и жмем кнопку «Переменные среды».

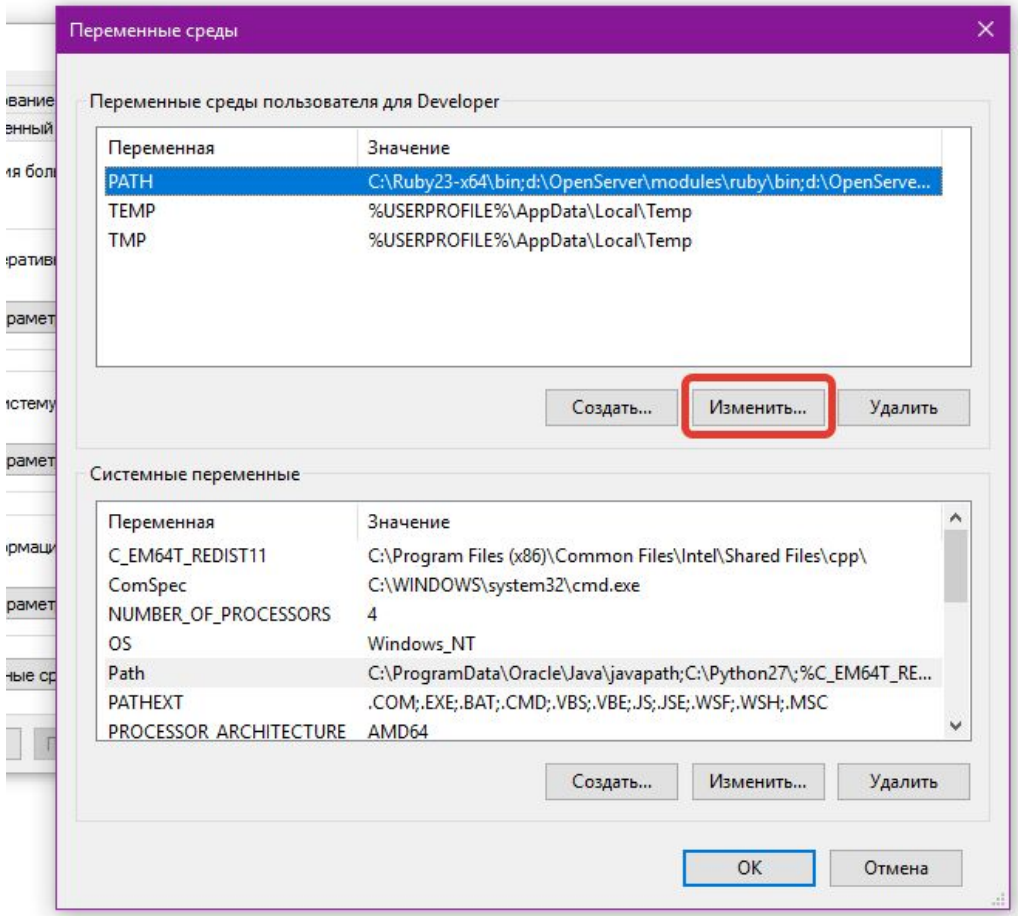


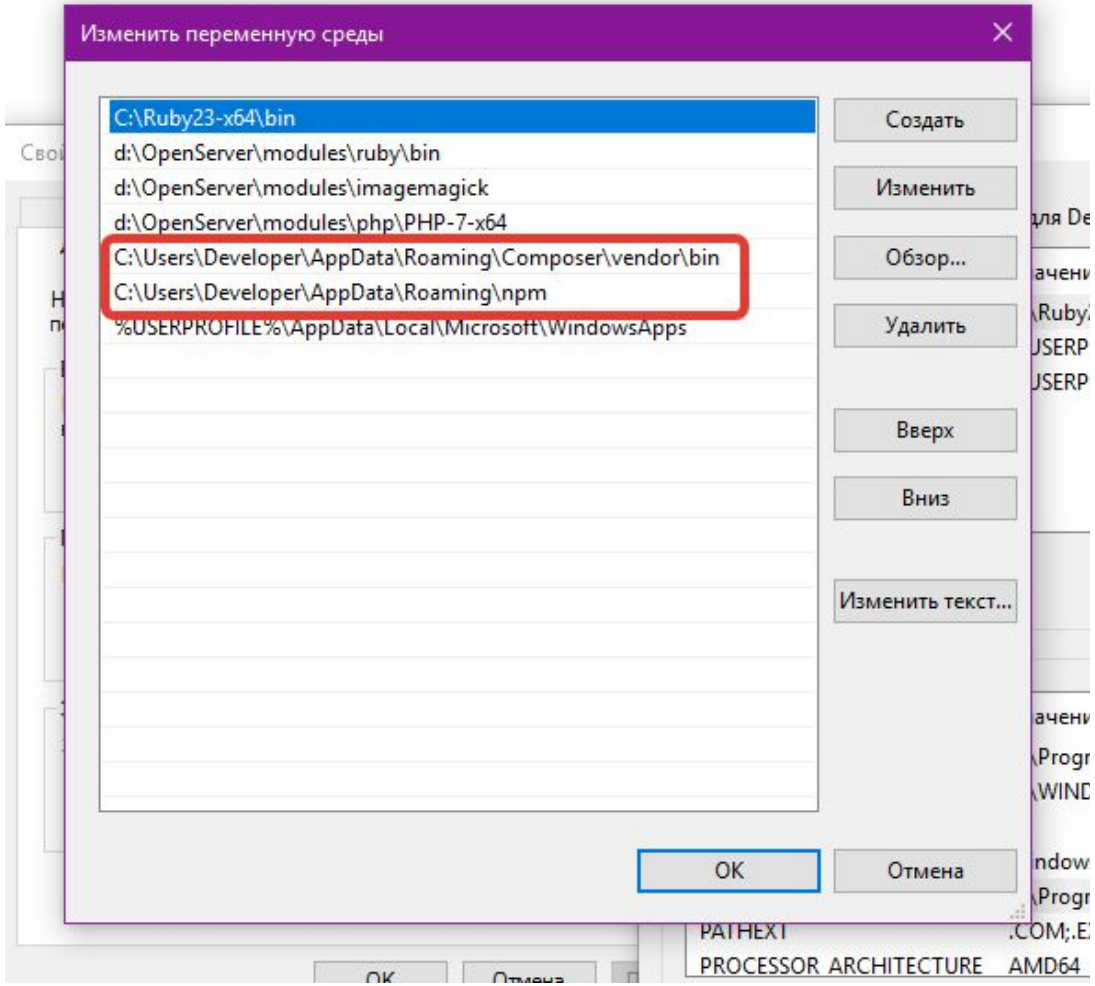




Установка Laravel 5.3

В открывшемся окне, в переменных средах для пользователя, открываем изменение `PATH`, где добавляем путь к файлу исполняемому `laravel`. У меня он находится в папке `c:\Users\Developer\AppData\Roaming\Composer\vendor\bin\`, где «Developer» — имя учетной записи (пользователя)





Установка Laravel 5.3

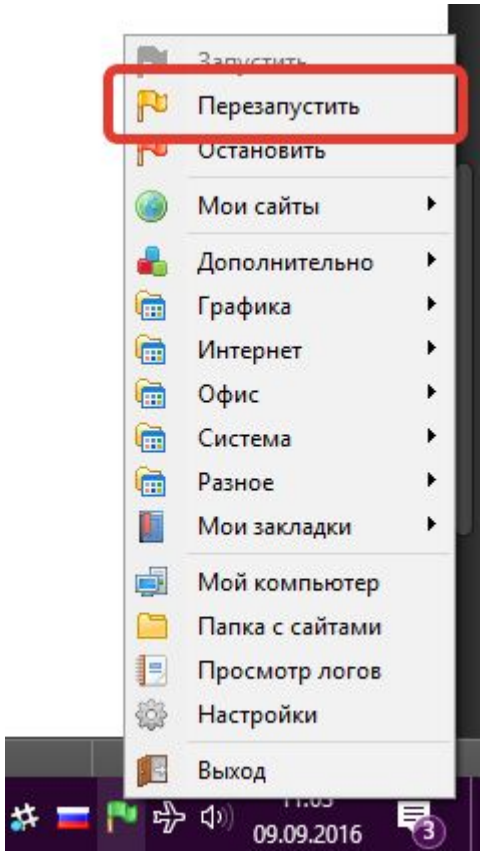
Выполнив одну из команд:

laravel new blog
или
composer create-project --prefer-dist laravel/laravel blog

Фреймворк скачается в папку «blog». Можно сразу прописать адрес сайта. Для этого необходимо видоизменить команду:

laravel new mysite.dev
или
composer create-project --prefer-dist laravel/laravel mysite.dev

После этого необходимо перезапустить `OpenServer`, чтобы он «увидел» новый сайт.



Настройка Laravel 5.3

.gitignore

Так как многие пользуются репозиториями, сразу идем в файл `.gitignore`, приводя его к вид

```
.env
/vendor
/node_modules
/public/storage
/.idea
Homestead.yaml
Homestead.json
*npm-debug.log*
```

Таким образом включая в запрет выгрузки информации из node_modules и автогенерируемых файлов из `public/build` и другие «системные» файлы.
Содержимое папок `vendor` и `node_modules` подгружаются автоматически

Environment (.env)

```
APP_ENV=local
APP_DEBUG=true

DB_CONNECTION=pgsql
DB_HOST=127.0.0.1
DB_PORT=5432
DB_DATABASE=my_database
DB_USERNAME=root
DB_PASSWORD=
```

`DB_CONNECTION` — имя подключения соответствующей базы данных из конфигурации `config/databases.php >> conntections`.
В случае с PostgreSQL указываем `pgsql`, а в случае с MySQL/MariaDB — `mysql`.
Также не забываем указывать порт, используемый для базы данных. Номер порта можно посмотреть в `OpenServer` в на вкладке `Сервер`.
Перед этим, воспользовавшись приложением `PgAdmin III` из состава `OpenServer`, необходимо добавить базу данных и прописать параметры в файле `.env`.

Настройка Laravel 5.3

Для дальнейшей работы понадобятся следующие пакеты:

- graham-campbell/exceptions
- barryvdh/laravel-debugbar
- barryvdh/laravel-ide-helper
- laracasts/generators

Laravel Exceptions

Пакет нужен для улучшенного вывода информации о возникающих ошибках при разработке в режиме дебага.
Для установки поочередно выполняем в консоли команды:

```
composer require graham-campbell/exceptions
composer require filp/whoops --dev
```

Далее, в файле `config/app.php` добавляем сервис-провайдер `GrahamCampbell\Exceptions\ExceptionsServiceProvider::class` в блок `providers`.

После этого, в файле `App\Exceptions\Handler.php` в блоке `use` меняем `use Illuminate\Foundation\Exceptions\Handler as ExceptionHandler;` на `use GrahamCampbell\Exceptions\NewExceptionHandler as ExceptionHandler;`

Настройка Laravel 5.3

Laravel Debugbar

Это пакет для интеграции панели отладки PHP с Laravel. Он включает в себя ServiceProvider для регистрации панели отладки и подключения ее к выходным данным

Для установки выполняем команду:

```
composer require barryvdh/laravel-debugbar
```

Далее, в файле `config/app.php` добавляем сервис-провайдер `Barryvdh\Debugbar\ServiceProvider::class,` в блок `providers`.

Laravel 5 IDE Helper Generator

Этот пакет генерирует вспомогательные файлы, которые позволяют IDE обеспечить точное автозаполнение. Генерация производится на основе файлов в текущем проекте, поэтому они всегда актуальны.

Для установки поочередно выполняем ком

```
composer require barryvdh/laravel-ide-helper
composer require doctrine/dbal
```

Далее, в файле `config/app.php` добавляем сервис-провайдер `Barryvdh\LaravelIdeHelper\IdeHelperServiceProvider::class,` в блок `providers`.

После этого блок `scripts/post-update-cmd` в файле `composer.json` из корневого каталога, приводим к виду:

```
{
  "scripts": {
    "post-update-cmd": [
      "Illuminate\\Foundation\\ComposerScripts::postUpdate",
      "php artisan ide-helper:generate",
      "php artisan ide-helper:meta",
      "php artisan optimize"
    ]
  }
}
```

Настройка Laravel 5.3

Laravel 5 Extended Generators

Пакет для создания миграций и моделей. Например, для создания таблицы `news` с полями `slug`, `title`, `content`
можно использовать всего одну команду

php artisan make:migration:schema create_news_table --schema="slug:string:unique, title:string:unique, content:text"

Для установки пакета под Laravel 5.3, нужно – выполнить команду:

```
composer require laracasts/generators --dev
```

Далее, в файле `config/app.php` добавляем сервис-провайдер `Laracasts\Generators\GeneratorsServiceProvider::class,` в блок `providers`.

После установки всех пакетов, выполняем в консоли:

```
php artisan vendor:publish
composer update
```

Установка Elixir

В консоли, находясь в папке проекта, поочередно выполняем команды:

```
npm install --global gulp-cli
npm install --no-bin-links
```

Для использования VueJs, который в 5.3 идет в комплектации, подключаем и его со всеми необходимыми пакетами:

```
npm rebuild node-sass
npm install hammerjs vue-async-data materialize-css
```

Обратите внимание: такие компоненты, как `vue` и `vue-resource` не вписаны, так как они уже в комплектации.
Далее, в файле `resources/assets/js/bootstrap.js` после строки `window.$ = window.jQuery = require('jquery');` найдете подключение CSS-фреймворка Twitter Bootstrap (`require('bootstrap-sass');`). Так как используем не его, а MaterializeCSS — заменяем на `require('../vendor/materialize-css/js/bin/materialize.min');`.
Далее, находим:

```
window.Vue = require('vue');
require('vue-resource');
```

И приводим к виду:

```
window.Vue = require('vue');
var VueResource = require('vue-resource');
var VueAsyncData = require('vue-async-data');

Vue.use(VueResource);
Vue.use(VueAsyncData);
```

Установка Elixir

Elixir-это язык программирования для виртуальной машины Erlang. Elixir обеспечивает макромеханизм, поддерживает полиморфизм с помощью протоколов (подобных Clojure) и многие другие функции, сохраняя при этом функциональные аспекты Erlang, используемые для построения распределенных отказоустойчивых приложений.

MaterializeCSS

Для его установки необходимо выполнить команду

```
npm install materialize-css
```

Далее, необходимо подключить используемый MaterializeCSS шрифт `Material Icon` в шаблоне:

```
<!DOCTYPE html>
<html>
<head>
  <!--Import Google Icon Font-->
  <link href="http://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
</head>

<body>
  //
</body>
</html>
```

Установка Elixir

Настройка gulpfile.js

При разработке очень удобный инструмент браузера — консоль. Чтобы организовать «быстрый поиск» проблемной строки, в файл `gulpfile.js` добавляем строку `elixir.config.sourcemaps = true;`, отвечающую за генерацию.

При работе с MaterializeCSS и SASS он имеет вид:

```
const elixir = require('laravel-elixir');
require('laravel-elixir-vue');
elixir.config.sourcemaps = true;
elixir(mix =>
    {
        var
            assets       = 'resources/assets/',
            node_modules = '../../../node_modules/';
        mix
            .sass('app.scss', 'public/css/app.css')
            .copy(assets + 'images', 'public/images')
            .copy(node_modules + 'materialize-css/fonts', 'public/build/fonts')
            .webpack('app.js')
            /*
             * Version
             */
            .version(
                [
                    'css/app.css',
                    'js/app.js'
                ]
            );
    }
);
```

Установка

Elixir

Resources: SASS

Так как не все компоненты из состава фреймворка MaterializeCSS нужны для разработки того или иного проекта, добавляем ссылки на них в наш файл `resources/assets/sass/app.scss`:

```
@charset "UTF-8";
// Mixins
// @import "../../../node_modules/materialize-css/sass/components/prefixer";
@import "../../../node_modules/materialize-css/sass/components/mixins";
@import "../../../node_modules/materialize-css/sass/components/color";
// Variables;
@import "../../../node_modules/materialize-css/sass/components/variables";
// Reset
@import "../../../node_modules/materialize-css/sass/components/normalize";
// components
@import "../../../node_modules/materialize-css/sass/components/global";
@import "../../../node_modules/materialize-css/sass/components/icons-material-design";
@import "../../../node_modules/materialize-css/sass/components/grid";
@import "../../../node_modules/materialize-css/sass/components/navbar";
@import "../../../node_modules/materialize-css/sass/components/roboto";
@import "../../../node_modules/materialize-css/sass/components/typography";
@import "../../../node_modules/materialize-css/sass/components/cards";
@import "../../../node_modules/materialize-css/sass/components/toast";
@import "../../../node_modules/materialize-css/sass/components/tabs";
@import "../../../node_modules/materialize-css/sass/components/tooltip";
```

Установка
Elixir
Resources: SASS

```
@import "../../../node_modules/materialize-css/sass/components/buttons";
@import "../../../node_modules/materialize-css/sass/components/dropdown";
@import "../../../node_modules/materialize-css/sass/components/waves";
@import "../../../node_modules/materialize-css/sass/components/modal";
@import "../../../node_modules/materialize-css/sass/components/collapsible";
@import "../../../node_modules/materialize-css/sass/components/chips";
@import "../../../node_modules/materialize-css/sass/components/materialbox";
@import "../../../node_modules/materialize-css/sass/components/forms/forms";
@import "../../../node_modules/materialize-css/sass/components/table_of_contents";
@import "../../../node_modules/materialize-css/sass/components/sideNav";
@import "../../../node_modules/materialize-css/sass/components/preloader";
@import "../../../node_modules/materialize-css/sass/components/slider";
@import "../../../node_modules/materialize-css/sass/components/carousel";
@import "../../../node_modules/materialize-css/sass/components/date_picker/default";
@import "../../../node_modules/materialize-css/sass/components/date_picker/default.date";
@import "../../../node_modules/materialize-css/sass/components/date_picker/default.time";
```

Установка

Elixir

Resources: SASS

Так как не все компоненты из состава фреймворка MaterializeCSS нужны для разработки того или иного проекта, добавляем ссылки на них в наш файл `resources/assets/sass/app.scss`:

```
@charset "UTF-8";

// Mixins
// @import "../../../node_modules/materialize-css/sass/components/prefixer";
@import "../../../node_modules/materialize-css/sass/components/mixins";
@import "../../../node_modules/materialize-css/sass/components/color";
// Variables;
@import "../../../node_modules/materialize-css/sass/components/variables";
// Reset
@import "../../../node_modules/materialize-css/sass/components/normalize";
// components
@import "../../../node_modules/materialize-css/sass/components/global";
@import "../../../node_modules/materialize-css/sass/components/icons-material-design";
@import "../../../node_modules/materialize-css/sass/components/grid";
@import "../../../node_modules/materialize-css/sass/components/navbar";
@import "../../../node_modules/materialize-css/sass/components/roboto";
@import "../../../node_modules/materialize-css/sass/components/typography";
```

```
@import "../../../node_modules/materialize-css/sass/components/cards";
@import "../../../node_modules/materialize-css/sass/components/toast";
@import "../../../node_modules/materialize-css/sass/components/tabs";
@import "../../../node_modules/materialize-css/sass/components/tooltip";
@import "../../../node_modules/materialize-css/sass/components/buttons";
@import "../../../node_modules/materialize-css/sass/components/dropdown";
@import "../../../node_modules/materialize-css/sass/components/waves";
@import "../../../node_modules/materialize-css/sass/components/modal";
@import "../../../node_modules/materialize-css/sass/components/collapsible";
@import "../../../node_modules/materialize-css/sass/components/chips";
@import "../../../node_modules/materialize-css/sass/components/materialbox";
@import "../../../node_modules/materialize-css/sass/components/forms/forms";
@import "../../../node_modules/materialize-css/sass/components/table_of_contents";
@import "../../../node_modules/materialize-css/sass/components/sideNav";
@import "../../../node_modules/materialize-css/sass/components/preloader";
@import "../../../node_modules/materialize-css/sass/components/slider";
@import "../../../node_modules/materialize-css/sass/components/carousel";
@import "../../../node_modules/materialize-css/sass/components/date_picker/default";
@import "../../../node_modules/materialize-css/sass/components/date_picker/default.date";
@import "../../../node_modules/materialize-css/sass/components/date_picker/default.time";
```

Установка Elixir

Resources: View

Итак, в ходе предыдущих действий, выполнив команду `gulp` в консоли, на выходе мы получим 2 файла: `app.js` и `app.css`. Так как была использована система контроля версий в шаблоне `resources/views/layouts/app.blade.php` необходимо прописать:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1">

  {{--<!-- CSRF Token -->--}}
  <meta name="csrf-token" content="{{ csrf_token() }}">
  {{--<!--Import Google Icon Font-->--}}
  <link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
  {{--<!-- Styles -->--}}
  <link href="{{ elixir('css/app.css') }}" rel="stylesheet">
</head>
<body>
  // Content
  {{--<!-- JavaScripts -->--}}
  <script src="{{ elixir('js/app.js') }}"></script>
</body>
</html>
```