ÍNDICE

INTRODUCCIÓN XV

PARTE I. ASOMÁNDONOS A LARAVEL 5.8 1

CAPÍTULO 1 - EL PRIMER PROYECTO 3

PHP ARTISAN 4
¿Y POR QUÉ USAR ARTISAN? 5
USANDO LARAGON 10

CAPÍTULO 2 - ESTRUCTURA DE LARAVEL 11

LA ESTRUCTURA DE DIRECTORIOS Y ARCHIVOS 11
EL ARCHIVO composer.json 11
EL DIRECTORIO app/ 14
EL DIRECTORIO bootstrap/ 15
EL DIRECTORIO config/ 15
EL DIRECTORIO database/ 15
EL DIRECTORIO public/ 16
EL DIRECTORIO resources/ 16
EL DIRECTORIO routes/ 17
EL DIRECTORIO vendor/ 17
LOS ARCHIVOS DEL DIRECTORIO RAÍZ 17
LA GENERACIÓN DE LA CLAVE DE LA APLICACIÓN 18
ESTRUCTURA DE UNA APLICACIÓN LARAVEL 19
UN EJEMPLO DE LA ESTRUCTURA LÓGICA 20
UNA VISTA CREADA CON EL MOTOR BLADE 22

CAPÍTULO 3 - AGREGAR PAQUETES 25

CAPÍTULO 4 - INTRODUCCIÓN A LAS BASES DE DATOS 29

LA CONFIGURACIÓN BÁSICA 29

Instalar nuevos motores 30
El ORM Eloquent 31
LAS MIGRATIONS *32*
LAS FACTORIES *32*
EL SEEDER *32*
LOS MODELOS *32*

CAPÍTULO 5 - HOLA, MUNDO 35

La forma chapucera 35
Primera mejora 37
Las cosas bien hechas 41
Excepciones 43

CAPÍTULO 6 - MÁS SOBRE PETICIONES 45

Parámetros en las rutas 45
La petición como argumento 48
Los helpers de volcado de datos 50

CAPÍTULO 7 - ENTENDER LAS RUTAS 53

La lista de las rutas 53
Crear un nuevo ámbito de rutas 55
Creando el controlador 57
Registrando el ámbito de rutas 57
Los métodos de controlador y vistas 60

CAPÍTULO 8 - OTRAS LLAMADAS A RUTAS 63

Configurando el enrutamiento 64
Envíos con cuerpo 65
Combinando GET y POST 69
Inyección de dependencias 70
El helper request() 71
Más sobre envío de parámetros 71
Colisión de parámetros 72

CAPÍTULO 9 - NOMBRES DE RUTAS 73

Asignar alias a las rutas 73
Enlaces a rutas y alias 74

CAPÍTULO 10 - EL SERVIDOR VIRTUAL 77

Creando un servidor virtual 78
Probando el invento 79

CAPÍTULO 11 - EL MOTOR DE PLANTILLAS DE LARAVEL (BLADE) 81

Crear una vista completa 82
Evaluar o no evaluar 83

CREAR Y USAR LAYOUTS 85
REUTILIZANDO EL LAYOUT 89
CONDICIONALES 90
BUCLES 92
BUCLES SOBRE ARREGLOS VACÍOS 96
CUANDO LOS ELEMENTOS SON ARREGLOS 97
EL OBJETO $LOOP 99

CAPÍTULO 12 - PROFUNDIZANDO EN BLADE 101

COMENTARIOS 101
INTEGRANDO BOOTSTRAP Y JQUERY 102
INTEGRANDO MATERIALIZE 102
CREANDO DIRECTIVAS PROPIAS DE BLADE 103
INCLUYENDO VISTAS PARCIALES 107
SEPARANDO LOS ESTILOS 108

PARTE II. INICIACIÓN A BD'S Y FORMULARIOS 111

CAPÍTULO 13 - MÁS SOBRE BASES DE DATOS 113

PREPARANDO EL ESCENARIO 113
CREANDO UNA BASE DE DATOS 116
LA TABLA DE ARTÍCULOS 117
RECREAR LA MIGRACIÓN 123
EL MODELO 124
LA PROPIEDAD $table 125
LAS PROPIEDADES $fillable Y $hidden 126
LA FACTORY 128
EL SEEDER 130

CAPÍTULO 14 - MODIFICAR LA TABLA 133

BASE DE DATOS CON FAKER 133
BASE DE DATOS REAL 134
REVIRTIENDO LA MIGRATION 136
EL MODELO 137
CAMBIOS EN LAS COLUMNAS 137
PROBLEMAS CON DOCTRINE 139

CAPÍTULO 15 - EL CRUD DE LA TABLA DE ARTÍCULOS (I) 141

CREAR UN CONTROLADOR 141
LA CLASE DEL MODELO 145
EL ENRUTAMIENTO PARA EL CONTROLADOR 145
UNA PEQUEÑA PRUEBA 146
LEYENDO TODOS LOS ARTÍCULOS 148
USANDO UNA DIRECTIVA PERSONALIZADA 150
COLECCIONES 151

CAPÍTULO 16 - EL CRUD DE LA TABLA DE ARTÍCULOS (II) 153

MOSTRAR UN ARTÍCULO 153

CAPÍTULO 17 - EL CRUD DE LA TABLA DE ARTÍCULOS (III) 161

EL FORMULARIO PARA NUEVOS ARTÍCULOS 162
EL TOKEN CSRF 165
RECIBIENDO EL FORMULARIO 166
VALIDANDO EL FORMULARIO 167
CREANDO LAS REGLAS DE VALIDACIÓN 167
VALIDANDO CON LAS REGLAS 169
LA VALIDACIÓN HA PASADO 169
LA GRABACIÓN "LARGA" 170
LA GRABACIÓN "CORTA" 170
LA GRABACIÓN "MIXTA" 171
EL FINAL DE LA GRABACIÓN 172
LA VALIDACIÓN HA FALLADO 172
LA VARIABLE $errors 173
LOS MENSAJES EN ESPAÑOL 174
LOS CAMPOS APARECEN VACÍOS 175

CAPÍTULO 18 - EL CRUD DE LA TABLA DE ARTÍCULOS (IV) 177

PREPARANDO LA LISTA DE ARTÍCULOS 177
EL MÉTODO EDIT() Y LA VISTA DE EDICIÓN 178
VALIDACIÓN Y ACTUALIZACIÓN 185
LA GRABACIÓN 186

CAPÍTULO 19 - EL CRUD DE LA TABLA DE ARTÍCULOS (V) 187

PREPARANDO LA LISTA DE ARTÍCULOS 187
EL MODAL DE CONFIRMACIÓN 189
EL FORMULARIO 190
EL CÓDIGO jQuery 191
BORRANDO EN EL CONTROLADOR 193

CAPÍTULO 20 - CONSULTAS A LA BASE DE DATOS 195

A TRAVÉS DEL MODELO, DE FORMA IMPLÍCITA 196
ERRORES FRECUENTES 197
A TRAVÉS DEL MODELO, DE FORMA EXPLÍCITA 198
COLECCIONES 199
OTROS MÉTODOS DEL MODELO 199
EL QUERY BUILDER 203

CAPÍTULO 21 - DEPURACIÓN 205

DUMP-SERVER 205
TINKER 206
TINKER Y LA BASE DE DATOS 207

CAPÍTULO 22 - CONSIDERACIONES FINALES 209

EL ÚLTIMO REGISTRO INSERTADO 209
CAMBIOS EN EL ENRUTADO 210

PARTE III. BASES DE DATOS CON RELACIONES 213

CAPÍTULO 23 - ESTRUCTURA DE DATOS 215

CREANDO EL PROYECTO 216
DISEÑANDO LAS TABLAS 217
CREANDO LAS MIGRATIONS 221
LA MIGRATION DE operators 223
LA MIGRATION DE tours 225
LA MIGRATION DE customers 228
LA MIGRATION DE LA TABLA PIVOTE (customer_tour) 229
CREANDO LOS MODELOS 230
EL MODELO Operator 231
EL MODELO Tour 234
EL MODELO Customer 236
RELACIONES ENTRE MODELOS 237
CREANDO LAS FACTORIES 237
LA FACTORY OperatorFactory 238
LA FACTORY TourFactory 239
LA FACTORY CustomersFactory 240
CREANDO EL SEEDER 241
LA EJECUCIÓN 244
CIRUGÍA CORRECTORA 244
EN DESARROLLO 245
EN PRODUCCIÓN 248

CAPÍTULO 24 - RUTAS, VISTAS Y CONTROLADORES 251

EL ESCENARIO BÁSICO 251
LOS CONTROLADORES 254
LAS RUTAS 255

CAPÍTULO 25 - EL LISTADO DE LOS OPERADORES 259

EL CONTROLADOR 262
LA PRIMERA FASE DE LA VISTA 266
PREPARANDO LA INTERACTIVIDAD 268
EL ESTADO DE LOS OPERADORES 269
BUSCAR POR NOMBRE O CIUDAD 271
ACOTAR POR RANGO DE OPERADOR 272
ORDENACIÓN POR NOMBRE O CIUDAD 273
LA PAGINACIÓN 275
EN EL CONTROLADOR 275
EN LA VISTA 276

EL PROBLEMA *276*
EL MÉTODO paginate() *278*
EL NÚMERO DE RESULTADOS *279*
LOS ENLACES COMPLEMENTARIOS 282
EL BOTÓN DE NUEVO OPERADOR *282*
EL ENLACE DE EDICIÓN *283*
EL ENLACE DE VER OPERADOR *283*
EL ENLACE DE CAMBIAR ESTADO *284*
EL ENLACE DE VER LA LISTA DE VIAJES *286*

CAPÍTULO 26 - MEJORAR EL LISTADO DE OPERADORES 289

FRAGMENTANDO EL CÓDIGO 289
EL HTML *290*
EL JAVASCRIPT *292*

CAPÍTULO 27 - FUNCIONALIDADES CON LOS OPERADORES 297

EL CAMBIO DE ESTADO 297
LOS VIAJES DEL OPERADOR 300
LOS ENLACES AL LISTADO DE VIAJES *301*
EN EL CONTROLADOR *301*
LA VISTA DE LA LISTA DE VIAJES *303*
VER LA FICHA DE UN OPERADOR 306
EN EL CONTROLADOR *307*
LA VISTA *307*
CREAR UN NUEVO OPERADOR 309
LAS REGLAS DE VALIDACIÓN *312*
EL MENSAJE DE ERROR *314*
EDITAR UN OPERADOR 315
EL MÉTODO edit() *316*
LA VISTA DE EDICIÓN *316*

CAPÍTULO 28 - LOS VIAJES 321

PREPARANDO LO NECESARIO 321
CREAR UN NUEVO VIAJE 323
EL FORMULARIO DE NUEVO VIAJE *323*
LA VISTA DE NUEVO VIAJE *324*
DE VUELTA AL CONTROLADOR *327*
EDITAR UN VIAJE 329
CONCLUYENDO *330*

CAPÍTULO 29 - LOS CLIENTES 331

LA LISTA DE VIAJES 331
LA LISTA DE CLIENTES DE CADA VIAJE 332
EL CONTROLADOR *332*
LA VISTA *334*

DE VUELTA AL CONTROLADOR *335*
CONCLUYENDO 336

PARTE IV. DE COLECCIONES Y CONSULTAS 337

CAPÍTULO 30 - MÁS SOBRE BASES DE DATOS (SQLITE) 339

PREPARANDO EL ESCENARIO 339
EL PROYECTO *340*
PREPARANDO LA APLICACIÓN *340*
EL DISEÑO DE LA BASE DE DATOS *341*
LAS MIGRATIONS *343*
LOS MODELOS *344*
LAS FACTORIES *344*
EL SEEDER *344*
CREAR LA BASE DE DATOS 345
LA BASE DE DATOS *346*
LA CREACIÓN *347*

CAPÍTULO 31 - SOBRE LAS CONSULTAS 349

LA RAZÓN DE SER DE $FILLABLE 349
SI LOS NOMBRES DE CAMPOS NO COINCIDEN *352*
¿Y SI NO TENEMOS $fillable? *352*
LAS CONSULTAS 353
PASAR UNA CONSULTA DIRECTAMENTE *354*
CONSULTAS DE SELECCIÓN *355*
CONSULTAS DE NO SELECCIÓN *369*
REGISTRO DE CONSULTAS EN TINKER 371

CAPÍTULO 32 - COLECCIONES 373

CREAR UNA COLECCIÓN 374
LOS MÉTODOS DE LAS COLECCIONES 375
EL PUNTO DE PARTIDA *376*
MAPEAR UNA COLECCIÓN *378*
PARTIR UNA COLECCIÓN EN SUBCOLECCIONES *380*
DETERMINAR SI UNA COLECCIÓN CONTIENE UN ELEMENTO *380*
CONVERTIR UNA COLECCIÓN A JSON *380*
AÑADIR DATOS DE UNA COLECCIÓN A OTRA *381*
CREAR NUEVOS MÉTODOS 382
CREAR EL SERVICIO *383*
FILTRAR UNA COLECCIÓN *384*

PARTE V. RELACIONES MÁS COMPLEJAS 387

CAPÍTULO 33 - RELACIONES M-N ENTRE TRES TABLAS 389

PREPARANDO EL ESCENARIO 389
EL PROYECTO *390*

LAS MIGRATIONS *390*
LOS MODELOS *391*
LAS FACTORIES *397*
EL SEEDER *398*

CAPÍTULO 34 - LISTAR TAREAS CON TRES TABLAS 403

LA PREPARACIÓN VISUAL 404
EL CONTROLADOR 405
LA VISTA DEL LISTADO DE TAREAS 409

CAPÍTULO 35 - EDITAR, CREAR Y BORRAR TAREAS 411

EDICIÓN DE TAREAS 411
CREACIÓN DE TAREAS 414
ELIMINAR TAREAS 417
DE MAYÚSCULAS Y MINÚSCULAS 419

CAPÍTULO 36 - EDITAR, BORRAR Y CREAR RELACIONES 421

BORRAR RELACIONES 422
EDITAR RELACIONES 423
EL CONTROLADOR *424*
LA VISTA *427*
LA ACTUALIZACIÓN *429*
CREAR NUEVAS RELACIONES 430
EL CONTROLADOR 431
LA VISTA *432*
GRABAR LA NUEVA RELACIÓN *433*

CAPÍTULO 37 - RELACIONES 1-N SOBRE LA MISMA TABLA 435

EMPEZANDO POR LO MÁS FÁCIL 436
LA MIGRATION *436*
EL MODELO *437*
LA BASE DE DATOS *439*
PROBANDO EL FUNCIONAMIENTO 439

CAPÍTULO 38 - RELACIONES MULTINIVEL 1-N (I) 445

PREPARANDO EL ESCENARIO 445
LOS LISTADOS DE CATEGORÍAS 447
EL CONTROLADOR *447*
LA VISTA *448*
CREAR NUEVAS CATEGORÍAS 449
EL SERVICIO DE LECTURA DE CATEGORÍAS 451
DENTRO DEL SERVICIO *452*

CAPÍTULO 39 - RELACIONES MULTINIVEL 1-N (II) 455

EDICIÓN DE CATEGORÍAS 455

EL CONTROLADOR *456*
LA VISTA *458*
DE VUELTA AL CONTROLADOR *459*
BORRADO DE CATEGORÍAS 459
EL CONTROLADOR *460*
REFLEXIONES SOBRE EL BORRADO 461

CAPÍTULO 40 - RELACIONES M-N SOBRE LA MISMA TABLA 463

LAS TABLAS 464
PREPARANDO EL PROYECTO 465
LAS MIGRATIONS 466
EL MODELO 468
EL PROYECTO 470
EL LISTADO DE CATEGORÍAS 470
LA PAGINACIÓN *471*
CATEGORÍAS Y SUBCATEGORÍAS *474*
LA VISTA *476*
LA ASOCIACIÓN DE CATEGORÍAS 478
LA DESASOCIACIÓN DE CATEGORÍAS 481

APÉNDICES LARAVEL 5.8 Y 6 483

A. CORREOS SIMULADOS 485

CONFIGURANDO NUESTRA APLICACIÓN 486
PROBANDO EL SISTEMA 487

B. INSTALANDO LARAGON 489

INSTALANDO LARAGON 490
CONFIGURANDO phpMyAdmin *491*
CONFIGURANDO PHP *491*
MIGRAR DESDE OTRA APLICACIÓN 492
LAS BASES DE DATOS *492*
LOS PROYECTOS *494*
LOS SERVIDORES VIRTUALES *494*
Y YA ESTÁ *496*

C. USAR SPARKPOST 497

ÍNDICE ALFABÉTICO 499

INTRODUCCIÓN

Este es un documento que resume cómo empezar a usar y sacar partido del framework Laravel, en su versión 5.8 (la más empleada en el momento de escribir estas líneas).

Para ello se le suponen al lector algunos conocimientos previos. Un framework PHP es una herramienta que nos permite crear aplicaciones basadas en patrones de programación adecuados y buenas prácticas de programación. Aunque nos facilita muchas de las tareas más rutinarias del desarrollo de una aplicación, no lo hace todo por nosotros. Debemos saber lo que estamos haciendo, y cómo decirle al framework lo que necesitamos.

Laravel es fácil de entender y aprender, con una curva de aprendizaje más pronunciada al principio, pero mucho más suave cuando uno se va familiarizando con el framework. En este manual va a aprender a realizar aplicaciones con Laravel. Verá que resulta más fácil y potente de lo que imagina. Sin embargo, hay una herramienta de la que no debe prescindir y que usará con mucha frecuencia, dada la ayuda que supone. Es la documentación oficial de Laravel (**https://laravel.com/docs/5.8**).

Un framework PHP nos permite crear aplicaciones completas empleando un patrón basado en MVC (Model - View - Controller o, en español, Modelo - Vista - Controlador), facilitándonos su correcta implementación.

Para poder aprender a usar cualquier framework PHP y crear aplicaciones web, es imprescindible tener unos conocimientos previos de desarrollo web.

PHP

Debemos conocer este lenguaje a fondo. Evidentemente, no se trata de saber de memoria todo el manual de PHP (**http://php.net/manual/es/index.php**), ni absolutamente todas las instrucciones y sus sintaxis. De hecho, PHP es un lenguaje tan potente, extenso y versátil que dudo mucho que haya una sola persona en el mundo (incluidos los desarrolladores) que se sepa esto de memoria. Y tampoco tiene ningún sentido. Debemos saber lo que PHP puede hacer, y ser capaces de buscar, en el manual y otras fuentes, la información específica sobre instrucciones o sintaxis que necesitemos. Sin embargo, sí debemos tener el conocimiento y la experiencia necesarios para desarrollar una aplicación en "código plano", es decir, sin ayuda de un framework. Debemos entender lo que es una aplicación web, en arquitectura cliente-servidor, y entender lo que supone ejecutar un proceso en el servidor y enviar el resultado al cliente. Debemos entender lo que es una petición y una respuesta, y ser capaces de conectar con una base de datos u otros recursos externos. Si no está familiarizado con estos asuntos, le recomiendo los artículos de **https://eldesvandejose.com/category/php/**.

HTML 5, CSS y JavaScript

Estas tres tecnologías se emplean para el desarrollo de frontend, es decir, de las vistas que se renderizan en el navegador de un cliente. Debe estar familiarizado con ellas para poder crear las vistas. Los protocolos de las distintas versiones que van saliendo de estas tecnologías son definidos por el W3C, que es el organismo internacional encargado de esto. Así mismo, debe ser consciente de que no todos los últimos estándares aprobados por dicha entidad pueden ser usados "libremente", en el sentido de que pasa un tiempo hasta que los navegadores los adoptan, con lo cual es posible que algún recurso de estas tecnologías, de reciente aprobación, aún no funcione adecuadamente en todos los navegadores. Si tiene dudas sobre si tal o cual característica es accesible a la mayoría de los usuarios, puede verificarlo en la página **https://caniuse.com/**. Evidentemente, deberá centrarse en los resultados de navegadores mayoritarios. Al igual que no tiene sentido emplear en su página recursos que no estén disponibles aún en la mayoría de los navegadores actuales, tampoco lo tiene renunciar a un recurso porque no funcione, por ejemplo, en Internet Explorer 8, porque es un navegador que, a día de hoy, ya no usa casi nadie (y el que lo use, que se actualice a un navegador de verdad; es su problema, no el del desarrollador).

SASS, BootStrap y Materialize

Son conocimientos que, aunque son exclusivamente relativos al desarrollo frontend (recuerda que Laravel, al ser un framework PHP, está más orientado al desarrollo backend), deberá tener para crear vistas que se integren con su aplicación de un modo eficiente y elegante. Puede documentarse sobre estas tecnologías en las correspondientes URL's (**https://sass-lang.com/**, en **https://getbootstrap.com/** y en **https://materializecss.com/**).

Otros conocimientos previos

Además, deberá tener conocimiento, si bien sea "por encima", de herramientas de desarrollo como Composer (**https://getcomposer.org/**), NodeJS (**https://nodejs.org/es/**) y git / github (**https://git-scm.com/**). Deberá conocer, lo mejor posible, jQuery (**https://jquery.com/**), así como, por supuesto, TypeScript (**https://www.typescriptlang.org/**). También deberá estar familiarizado con algún editor de texto plano. Dado que de estos hay gran variedad, de los cuales algunos son gratuitos y otros de pago, nosotros vamos a usar editores gratuitos, que tienen todas las prestaciones que vayamos a necesitar. Yo le recomendaría emplear Visual Studio Code (**https://code.visualstudio.com/**) pero, si lo prefiere, puede usar Sublime Text (**https://www.sublimetext.com/**), Notepad ++ (**https://notepad-plus-plus.org/**), Atom (**https://atom.io/**), o cualquier otro de su elección.

El entorno de trabajo

Para desarrollar con Laravel (para el desarrollo web, en general) deberá tener instalado, en su ordenador, un entorno básico de trabajo, que incluya los elementos que mencionamos en este apartado.

Lo primero es un entorno de desarrollo en servidor local, incluyendo Apache, PHP y MySQL, así como algún sistema visual de gestión de las bases de datos de MySQL. Si, como en mi caso, emplea Windows, le recomiendo descargar y usar la última versión disponible de Laragon (es el que empleo yo actualmente), en **https://laragon.org/download/index.html**. Si lo prefiere, puede usar XAMPP (**https://www.apachefriends.org/es/index.html**). Si no ha trabajado antes con esta herramienta, puede leer en **https://eldesvandejose.com/2016/04/22/montando-los-servidores/**. Si emplea Mac, su mejor opción será Mamp (**https://www.mamp.info/en/**). Los usuarios de Linux, seguramente, ya tienen instaladas estas herramientas pero, en caso de no tenerlas actualizadas, pueden leer un interesante artículo introduciendo en el navegador la siguiente URL:

https://www.digitalocean.com/community/tutorials/como-instalar-en-ubuntu-18-04-la-pila-lamp-linux-apache-mysql-y-php-es.

Si prefiere, como visualizador de bases de datos, emplear MySQL Workbench, en lugar del clásico PHPMyAdmin, puede encontrarlo en **https://www.mysql.com/products/workbench/**.

Dado que Laravel le permite trabajar con múltiples motores de base de datos, aprenderemos, también, a crear miniaplicaciones con SQLite. En ese sentido le vendrá bien contar con SQLite Studio (**https://sqlitestudio.pl/index.rvt**).

También necesitará Postman (aquí aprenderemos qué es y cómo usarlo). Se lo puede descargar en **https://www.getpostman.com/apps**. Le aconsejo que, aunque su equipo sea de 64 bits, descargue la versión de 32 bits. Le irá más fluida.

Además, deberá tener instalado Composer, NodeJS, y otras herramientas que iremos conociendo sobre la marcha.

Cómo leer este libro

El libro que tiene en sus manos es el trabajo de varios meses desarrollando un método de aprendizaje de Laravel que a mí me ha funcionado, y que me consta que le funcionará a usted también. Puede leerlo de dos formas: si es nuevo en Laravel, y nunca antes ha desarrollado un proyecto, debería empezar por el principio, y seguir la línea de capítulos tal como están. Poco a poco irá aprendiendo cada vez más. Por otro lado, si ya ha experimentado con este framework, los primeros capítulos podrá leerlos "en diagonal", o incluso saltárselos directamente, e ir a aquellos que más le interesen. Quizá quiera solo una determinada habilidad. En ese caso encontrará en este volumen, y en otro avanzado, que será publicado posteriormente, la información que necesita.

Este libro tiene, como todos (o la mayoría) de los textos sobre programación y/o desarrollo, una doble vertiente: por un lado está el texto en sí, destinado a presentarle conceptos y técnicas de desarrollo web de última generación, facilitando su aprendizaje. Por otra parte, y como complemento, tiene los códigos completos de todos los ejercicios prácticos que se muestran en el libro, en la web del editor. Además, en el texto están solo las partes relevantes de cada código. En el material complementario están todos los archivos completos para que pueda probar el funcionamiento, y analizar los listados, reforzando su aprendizaje. Por tanto, si es usted comprador del mismo podrá acceder gratuitamente a ellos, visitando: **http://rclibros.es**, y después la página individual del libro.

Al copiar el material en su ordenador, póngalo en carpetas, dentro de su localhost, con los nombres con los que aparecen. Además, en cada proyecto, tendrá que instalar el core de Laravel. Yo no lo he incluido porque es muy pesado para replicarlo en cada ejemplo, y se puede descargar desde Internet. Para ello, una vez en su ordenador la carpeta de un proyecto, entre en la misma y, en la consola, teclee:

```
composer update
```

y

```
npm install
```

Con esto se descargará el core de Laravel, y los paquetes complementarios, ya que los archivos json, de los que hablaremos en su momento, tienen la información exacta de lo que es necesario para cada proyecto.

Por lo tanto, para seguir este libro, es necesario disponer de un equipo y de una conexión a Internet, y aparte de eso, solo ganas de aprender. Espero que disfrute leyéndolo tanto como yo he hecho escribiéndolo.

Recuerde que debe descargar el material complementario ANTES de empezar a leer. Si bien en el libro se han replicado los fragmentos de código que requieren mayor atención, necesitará tener los listados originales para seguir las explicaciones con aprovechamiento.

Cómo está escrito este libro

Este libro ha sido escrito, sobre todo, con mucha ilusión y ganas de transmitir. He escrito el libro que a mí me hubiera gustado tener cuando empecé a adentrarme en el fascinante mundo de Laravel. El libro está redactado en un lenguaje cercano, sencillo, huyendo de formalismos incómodos. Los capítulos han sido estructurados de una forma concisa, procurando no alargarlos más de lo necesario. Es cierto que algunos puntos de Laravel requieren más explicación que otros, pero siempre me he ceñido a lo que hace falta conocer, sin ambages.

La estructura y desarrollo de este trabajo se ha realizado en dos fases: básico y avanzado. El primero, este libro que tiene en sus manos, introduce los conceptos fundamentales necesarios para empezar a desarrollar en Laravel. Un segundo libro, más avanzado, actualmente en preparación, incluye temáticas como autenticación de usuarios, recreación de comandos de Artisan, colas y trabajos, uso de paquetes para Excel y PDF o Datatables en Laravel, y muchos otros que llevarán al lector “al siguiente nivel”, permitiéndole desarrollar con facilidad aplicaciones de primera

línea. Si este primer tomo le gusta, el segundo le enamorará (conmigo lo hizo). Sin embargo, no empiece la casa por el tejado. Lea primero este volumen, y aprenda de él, antes de sumergirse en el siguiente.

Y allá vamos

Tras esta introducción, breve pero eminentemente práctica, vamos a empezar a trabajar con Laravel. Para ello, el primer paso es montar, en nuestro equipo, el propio instalador de Laravel, que será lo que nos permita crear luego proyectos usando este framework. Para ello, abrimos la consola de mandatos de nuestro ordenador. Puede usar la propia consola del sistema, la que le proporciona Git (el llamado Git Bash), el Cygwin (**https://www.cygwin.com/**), o la que prefiera.

Una vez vea el prompt del sistema, teclee lo siguiente:

```
composer global require "laravel/installer"
```

Esto crea, en su ordenador, un instalador de Laravel disponible desde cualquier punto del sistema. Solo tendrá que hacer esto una vez, y ya lo tendrá disponible. Lo que le permitirá este instalador es crear cualquier proyecto Laravel en su ordenador, de una forma más fácil y rápida.

En este libro se le sugerirá iniciar la creación de cada proyecto tecleando en la consola, en su localhost, lo siguiente:

```
laravel new proyecto
```

donde proyecto es el nombre que le daremos al proyecto. Sin embargo, esto creará un proyecto de Laravel 6, que es la última versión liberada. En este libro nos hemos centrado en Laravel 5.8, ya que es la versión más estable y empleada en la actualidad. Para crear un proyecto con esta versión teclee, en la consola:

```
composer create-project laravel/laravel proyecto '5.8.*'
```

poniendo, en lugar de proyecto, el nombre de su proyecto.

En el volumen avanzado se comentarán las novedades de Laravel 6 pero, por ahora, hágame caso y céntrese en la versión 5.8. Todo lo que aprenda en este volumen le servirá, también, en la versión 6, pero la 5.8 es más estable. Siga mi consejo y, al terminar de leer este volumen y el siguiente, me lo agradecerá.

Reconocimientos

Quiero agradecer a Laravel por la magnífica herramienta que nos proporciona, y el esfuerzo realizado para que la versión 5.8 sea, sin duda, uno de los mejores frameworks PHP del mercado. Además, es de agradecer el esfuerzo que han hecho para presentar una documentación tan detallada, amplia y bien estructurada.

Así mismo, quiero valorar la inestimable ayuda que siempre ha representado el foro Stack Overflow (**https://es.stackoverflow.com/** y **https://stackoverflow.com/**), tanto para mí, como para mis compañeros de trabajo.

También quiero reconocer a Google, por sus magníficas aportaciones al mundo del desarrollo web, en especial (aunque no únicamente), por su buscador, por Angular 6 (**https://angular.io/**) y por su magnífica aportación en lo que a CSS se refiere, con Material Design (**https://material.io/design/**).

El autor

José López Quijado es un desarrollador vocacional que lleva coqueteando con el mundo de la informática desde mediados de los años 80 del pasado siglo, y que se enfrentó con la falta de documentación, ya que la poca que entonces circulaba era en lengua inglesa, rudimentaria y escasa. Sus primeros ordenadores personales se los confeccionó él mismo comprando las piezas, y ensamblándolos en su propia casa; las largas horas transcurridas conectando dispositivos a una placa base, atornillando todo en una caja de las que llamaban XT y AT, y buscando en bibliotecas y grupos de estudio toda la información necesaria para que sus pequeños modelos cobraran vida, le han permitido crecer profesionalmente.

Todavía faltaban varios años para el advenimiento de Internet, los ordenadores eran enormes aparatos, toscos, con grandes consumos de energía, y unos precios prohibitivos para la mayoría de los mortales, sin embargo, él no encuentra barreras y descubre su Santo Grial, entre placas base de más de medio metro de lado, y procesadores de 4 y, posteriormente, 8 bits; en aquella época el Saturno H48, la serie Z de Zilog y la serie 6800 de Motorola.

Con la llegada de los 90 y el comienzo de la popularidad de la World Wide Web, continúa su proceso de aprendizaje con Internet tanto con el desarrollo de los toscos documentos HTML, las primeras bases de datos como dBase II, III y IV, explorando al máximo cada tecnología, cliente servidor, o cualquiera de las que considera que mantienen posibilidades de las nuevas tecnologías y que le permitan dar rienda suelta a su creatividad.

Actualmente es el responsable de desarrollo en el Grupo Quirón, donde se ocupa de desplegar la operativa de grandes cuentas en los sectores financiero, industrial, alimentario, inmobiliario y de otros clientes; también mantiene dos blogs sobre temas técnicos (**https.eldesvandejose.com** y **https://httpmasters.es**) donde escribe sobre aquellas tecnologías que más le gustan, procurando mantenerse actualizado y evolucionar con todo lo que significa el desarrollo de la Red de Redes.

PARTE I. ASOMÁNDONOS A LARAVEL 5.8

1 EL PRIMER PROYECTO

Vamos a empezar creando un proyecto Laravel con el que, por ahora, no haremos gran cosa. Es solo para crearlo y conocer la forma general de trabajar de las aplicaciones basadas en esta herramienta. Además, nos permitirá sentar algunos conceptos básicos. Para ello vamos a ir al directorio donde tengamos nuestros proyectos web. En mi caso, dado que trabajo con un entorno Windows y empleo Laragon (ver el Apéndice **B**), el servidor local (**localhost**) está apuntando a **c:/laragon/www**. En su caso, si emplea otro entorno u otro modo de trabajo, podría ser diferente. El caso es que, una vez en el directorio que corresponde a su **localhost** abra la consola de mandatos. Deberá asegurarse, mirando el prompt de la consola, de que en la línea de mandatos también esté en **localhost**. En mi caso, como empleo el Git Bash de Git, el prompt que me aparece es:

```
JoseQuijado@Jose MINGW64 /c/laragon/www
$
```

En la terminal de mandatos nativa de Windows sería:

```
C:\laragon\www>
```

Da lo mismo. Lo que importa es que tengo la terminal que voy a usar abierta en el directorio adecuado. Para crear un proyecto inicial, al que llamaremos **`PruebaLaravel`**, tecleamos la siguiente línea:

```
laravel new PruebaLaravel
```

Esto lanza el instalador de Laravel y crea un directorio con el nombre que le hemos dado al proyecto (**`PruebaLaravel`**). El instalador descarga de Internet los paquetes necesarios y crea, dentro del directorio, el proyecto "en blanco" con la estructura básica completa.

Si observa la lista de paquetes que se descargan, le llamará la atención ver que algunos de ellos se refieren a Symfony, otro popular framework de PHP. Esto es porque Laravel (al igual que otros frameworks existentes) emplea algunas de las herramientas de Symfony. No se trata de que tenga que aprender también este framework, cuya curva de aprendizaje es más pronunciada, sino que, simplemente, Laravel los emplea. No necesita preocuparse por ello.

Tras un breve periodo, que puede ser cosa de unos segundos o dos o tres minutos, dependiendo de la potencia de su equipo y la velocidad de su conexión, la estructura completa del proyecto se habrá creado.

PHP Artisan

En Laravel vamos a contar con una herramienta llamada ***Artisan***. Se trata de un comando de PHP (implementado por Laravel, no nativo del lenguaje) que nos servirá como interfaz para ejecutar los comandos propios de Laravel que debamos usar en la consola de mandatos. Así, por ejemplo, si tecleamos **`php artisan comando`**, donde **`comando`** es un comando implementado por Laravel, se ejecutará el comando indicado. Lo veremos en seguida. Para ello tenemos que situarnos, en la terminal de mandatos, en el directorio correspondiente al proyecto en el que vamos a trabajar. En mi caso, el prompt de Git Bash me indica:

```
JoseQuijado@Jose MINGW64 /c/laragon/www/PruebaLaravel
$
```

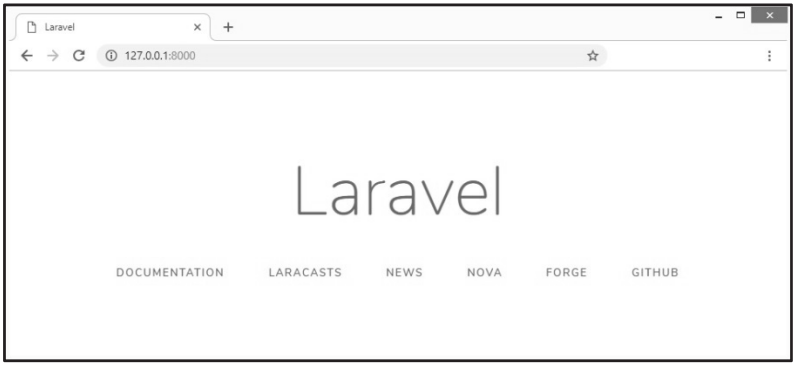
Ahora vamos a emplear Artisan para poner en marcha un servidor en el puerto **`8000`**, a fin de ejecutar la aplicación Laravel que acabamos de crear. Para ello tecleamos:

```
php artisan serve
```

La consola nos responde lo siguiente:

```
Laravel development server started: <http://127.0.0.1:8000>
```

Ahora abra su navegador e introduzca, en la barra de direcciones, la URL `127.0.0.1:8000`. El resultado que se carga es el siguiente:



Vale. No es muy atractivo ni espectacular, pero nos permite ver que Laravel está funcionando. Como ve, lo hemos cargado en el puerto `8000`, que es el que nos abrió el mandato `php artisan serve`.

Si quiere comprobarlo, vaya a la terminal de mandatos (que, ahora, parece estar bloqueada por Artisan, ya que no le permite teclear nada) y pulse `Ctrl – C`. Verá que Artisan termina su ejecución y la consola se desbloquea. Si ahora vamos al navegador y tratamos de recargar la página veremos que no funciona.

ATENCIÓN. Si tiene un antivirus instalado (algo que todos deberíamos tener), es posible que, cuando el comando `php artisan serve` intente ejecutar un proceso de su proyecto Laravel que se llama `server.php` le salte la alarma y el antivirus le diga que no es seguro. Se trata de un falso positivo. Deberá crear una excepción en el antivirus para que permita la ejecución de este proceso y no lo mande al baúl de virus. En este sentido, por mi experiencia, Avast resulta especialmente irritante. Por supuesto, poniéndonos en lo peor, siempre es preferible la molestia de un falso positivo que un exceso de permisividad por parte del antivirus. En todo caso, creamos la excepción, y el antivirus ya no volverá a saltar… al menos en ese proyecto en concreto.

¿Y por qué usar Artisan?

¿Por qué usar Artisan y el puerto `8000`? De hecho, yo puedo tener desactivado Artisan (lo que ocurrió cuando, en la terminal de mandatos, pulsé `CTRL-C`) y teclear,

en la barra de direcciones de mi navegador, una llamada a localhost, indicando la ruta del script **server.php** en el directorio de mi proyecto, así:

```
http://127.0.0.1/PruebaLaravel/server.php
```

Con esto obtengo el mismo resultado que en el caso anterior. Sin embargo, hay varias razones para emplear Artisan. En primer lugar, una vez arrancado el servidor del proyecto Laravel con Artisan, la ruta que hay que poner en el navegador es más corta, y más clara. Además, al usar el puerto **8000**, el puerto habitual de mi Apache local (el **80**) me queda disponible por si necesito cargar algún proceso local mientras se está ejecutando mi aplicación Laravel. Es un modo de evitar colisiones de puertos. Siguiendo con este razonamiento, Artisan NO me obliga a emplear el puerto **8000**. Si yo deseo tener este puerto disponible, y ejecutar mi aplicación en otro puerto puedo, por ejemplo, teclear:

```
php artisan serve --port=8300
```

Ahora podré iniciar mi aplicación Laravel tecleando en la barra de direcciones:

```
127.0.0.1:8300
```

De este modo, Artisan me da más flexibilidad a la hora de iniciar mi aplicación.

Pero no se trata solo de eso, evidentemente. Lo cierto es que Artisan nos permite ejecutar un gran número de comandos de Laravel que nos van a facilitar enormemente las tareas de desarrollo, como iremos viendo a lo largo de este libro. Pruebe a teclear en la consola, simplemente:

```
php artisan
```

Verá una respuesta como la siguiente:

```
Laravel Framework 5.8.11

Usage:
  command [options] [arguments]

Options:
  -h, --help            Display this help message
  -q, --quiet           Do not output any message
  -V, --version         Display this application version
      --ansi            Force ANSI output
      --no-ansi         Disable ANSI output
  -n, --no-interaction  Do not ask any interactive question
```

```
      --env[=ENV]       The environment the command should run
under
  -v|vv|vvv, --verbose  Increase the verbosity of messages: 1
for normal output,
 2 for more verbose output and 3 for debug

Available commands:
  clear-compiled       Remove the compiled class file
  down                 Put the application into maintenance
mode
  dump-server          Start the dump server to collect dump
information.
  env                  Display the current framework
environment
  help                 Displays help for a command
  inspire              Display an inspiring quote
  list                 Lists commands
  migrate              Run the database migrations
  optimize             Cache the framework bootstrap files
  preset                       Swap the front-end scaffolding
for the application
  serve                Serve the application on the PHP
development server
  tinker               Interact with your application
  up                   Bring the application out of maintenance
mode
 app
  app:name             Set the application namespace
 auth
  auth:clear-resets    Flush expired password reset tokens
 cache
  cache:clear          Flush the application cache
  cache:forget         Remove an item from the cache
  cache:table          Create a migration for the cache
database table
 config
  config:cache         Create a cache file for faster
configuration loading
  config:clear         Remove the configuration cache file
 db
  db:seed              Seed the database with records
 event
  event:generate       Generate the missing events and
listeners based on registration
 key
```

```
  key:generate          Set the application key
 make
  make:auth             Scaffold basic login and registration
views and routes
  make:channel          Create a new channel class
  make:command          Create a new Artisan command
  make:controller       Create a new controller class
  make:event            Create a new event class
  make:exception        Create a new custom exception class
  make:factory          Create a new model factory
  make:job              Create a new job class
  make:listener         Create a new event listener class
  make:mail             Create a new email class
  make:middleware       Create a new middleware class
  make:migration        Create a new migration file
  make:model            Create a new Eloquent model class
  make:notification     Create a new notification class
  make:observer         Create a new observer class
  make:policy           Create a new policy class
  make:provider         Create a new service provider class
  make:request          Create a new form request class
  make:resource         Create a new resource
  make:rule             Create a new validation rule
  make:seeder           Create a new seeder class
  make:test             Create a new test class
 migrate
  migrate:fresh         Drop all tables and re-run all
migrations
  migrate:install       Create the migration repository
  migrate:refresh       Reset and re-run all migrations
  migrate:reset         Rollback all database migrations
  migrate:rollback      Rollback the last database migration
  migrate:status        Show the status of each migration
 notifications
  notifications:table   Create a migration for the notifications
table
 optimize
  optimize:clear        Remove the cached bootstrap files
 package
  package:discover      Rebuild the cached package manifest
 queue
  queue:failed          List all of the failed queue jobs
  queue:failed-table    Create a migration for the failed queue
jobs database table
  queue:flush           Flush all of the failed queue jobs
```

```
  queue:forget          Delete a failed queue job
  queue:listen          Listen to a given queue
  queue:restart         Restart queue worker daemons after their
current job
  queue:retry           Retry a failed queue job
  queue:table           Create a migration for the queue jobs
database table
  queue:work            Start processing jobs on the queue as a
daemon
 route
  route:cache           Create a route cache file for faster
route registration
  route:clear           Remove the route cache file
  route:list            List all registered routes
 schedule
  schedule:finish       Handle the completion of a scheduled
command
  schedule:run          Run the scheduled commands
 session
  session:table         Create a migration for the session
database table
 storage
  storage:link          Create a symbolic link from
"public/storage" to "storage/
app/public"
 vendor
  vendor:publish        Publish any publishable assets from
vendor packages
 view
  view:cache            Compile all of the application's Blade
templates
  view:clear            Clear all compiled view files
```

Es una lista completa de todos los comandos que puede ejecutar con Artisan. No importa que ahora no entienda qué hacen o para qué están. Eso ya lo iremos viendo. Lo que realmente importa es que Artisan nos abre un sinfín de posibilidades para el desarrollo de un proyecto Laravel. Lo emplearemos mucho en este libro.

Lógicamente, cuando utiliza Artisan para arrancar el servidor de desarrollo para su aplicación Laravel, y dado que la consola queda bloqueada durante la ejecución, si necesita ejecutar de forma simultánea otro comando con Artisan deberá abrir otra instancia de la consola de mandatos. Esa es una de las razones por las que a mí me gusta trabajar con Visual Studio Code (VSC, en adelante, para abreviar). Me permite, entre otras cosas, usar una consola propia del editor, de la que puedo abrir tantas

instancias como necesite. No obstante, también puede abrir distintas instancias de cualquier consola de mandatos (la del sistema, la de GIT, o la que prefiera). Le va a funcionar lo mismo. A veces es bueno, incluso, tener un segundo monitor para las consolas y el navegador, y el principal para el editor.

Usando Laragon

Lo que acabamos de ver es, por así decirlo, la forma "clásica" de arrancar un proyecto Laravel en el navegador. Si está usando Laragon (y le recomiendo que lo haga) esta herramienta se ocupará de crear un servidor virtual por cada proyecto que hagamos, con solo reiniciar el servidor Apache después de crear el nuevo proyecto. En el capítulo **10** se habla de cómo crear servidores virtuales si tiene otra herramienta, como Xampp. En el Apéndice **B** se relata cómo instalar y poner en marcha Laragon. A partir de ahí, su proyecto puede iniciarse si teclea, en la barra de direcciones del navegador, lo siguiente:

```
http://PruebaLaravel.test
```

Además, para los proyectos Laravel, yo tengo configurado Laragon para que use la extensión **.lar**, en lugar de **.test**, con lo que mi proyecto arranca tecleando:

```
http://PruebaLaravel.lar
```

Lo del servidor virtual es muy cómodo, y Artisan lo reservaremos para otras cosas más importantes, como iremos viendo a lo largo del libro.