

Laravel



Tecnologías Avanzadas de Desarrollo

EB3.1: MODELADO HOMOGÉNEO DE PROYECTOS
SOFTWARE MEDIANTE FRAMEWORKS AVANZADOS.

Licencia




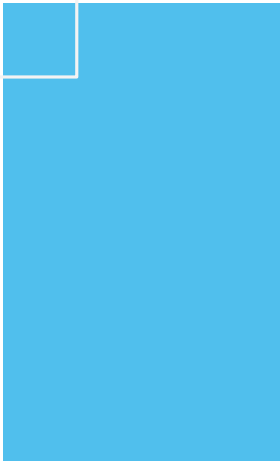
Toda la documentación de esta asignatura queda recogida bajo la licencia de Creative Commons

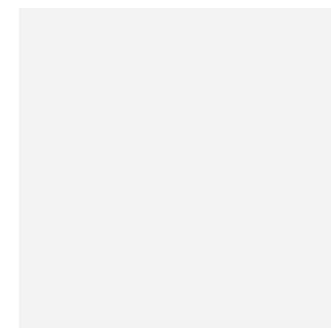
<https://creativecommons.org/licenses/by-nc-nd/4.0/>

En el caso de incumplimiento o infracción de una licencia Creative Commons, el autor, como con cualquier otra obra y licencia, habrá de recurrir a los tribunales. Cuando se trate de una infracción directa (por un usuario de la licencia Creative Commons), el autor le podrá demandar tanto por infracción de la propiedad intelectual como por incumplimiento contractual (ya que la licencia crea un vínculo directo entre autor y usuario/licenciatarario). El derecho moral de integridad recogido por la legislación española queda protegido aunque no aparezca en las licencias Creative Commons. Estas licencias no sustituyen ni reducen los derechos que la ley confiere al autor; por tanto, el autor podría demandar a un usuario que, con cualquier licencia Creative Commons, hubiera modificado o mutilado su obra causando un perjuicio a su reputación o sus intereses. Por descontado, la decisión de cuándo ha habido mutilación y de cuándo la mutilación perjudica la reputación o los intereses del autor quedaría en manos de cCutacia Juez o Tribunal.



ÍNDICE

- 
- 
1. Composer
 2. Laravel 9.x
 3. Node.js



Introducción



Para poder empezar a interactuar con frameworks de desarrollo web (backend) de alto nivel, primero debemos elegir uno.

Nosotros vamos a hacer uso de Laravel, en su versión 9.x. Ya que a nivel mundial es el framework más usado según “los datos de distintas plataformas”. Y en la asignatura de Programación Avanzada, trabajamos con PHP, pero no con frameworks.

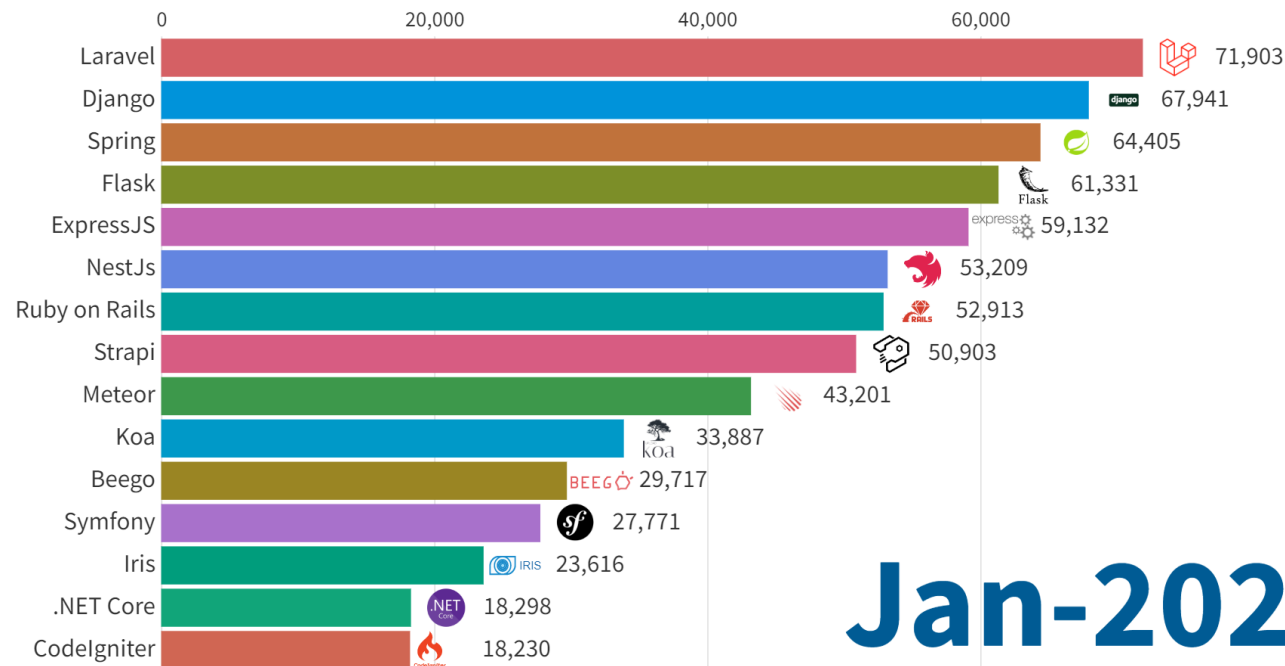
Para ello necesitamos conocer un poco del stack técnico ligado a él.

Laravel
9.0



Introducción

Most Popular Backend Frameworks



Jan-2023

Composer

Herramienta por excelencia en PHP para la gestión de librerías y dependencias, de manera que instala y las actualiza asegurando que todo el equipo de desarrollo tiene el mismo entorno y versiones. Además, ofrece autoloading de nuestro código, de manera que no tengamos que hacerlo nosotros "a mano".

Está escrito en PHP, y podéis consultar toda su documentación en <https://getcomposer.org/>.

Utiliza [Packagist](#) como repositorio de librerías.

Funcionalmente, es similar a Maven (Java) / npm (JS).



Primeros pasos

Cuando creamos un proyecto por primera vez, hemos de inicializar el repositorio. Para ello, ejecutaremos el comando `composer init` donde:

- Configuramos el nombre del paquete, descripción, autor (nombre), tipo de paquete (project), etc...
- Definimos las dependencias del proyecto (require) y las de desarrollo (require-dev) de manera interactiva.

En las de desarrollo se indica aquellas que no se instalarán en el entorno de producción, por ejemplo, las librerías de pruebas.

Tras su configuración, se creará automáticamente el archivo `composer.json` con los datos introducidos y descarga las librerías en la carpeta `vendor`. La instalación de las librerías siempre se realiza de manera local para cada proyecto.

A la hora de indicar cada librería introduciremos:

El nombre de la librería, compuesta tanto por el creador o "vendor", como por el nombre del proyecto.

Ejemplos: `monolog/monolog` o `laravel/installer`.

UD4 – ejemplo.json

```
{
  "name": "dwes/log",
  "description": "Pruebas con Monolog",
  "type": "project",
  "require": {
    "monolog/monolog": "^2.1"
  },
  "license": "LGPL-3.0-only",
  "authors": [
    {
      "name": "Olga M. Moreno",
      "email": "olga3emes@gmail.com"
    }
  ]
}
```

La versión de cada librería. Tenemos diversas opciones para indicarla:
Directamente: 1.4.2, Con comodines: 1.* , A partir de: >= 2.0.3, Sin rotura de cambios: ^1.3.2 // >=1.3.2 <2.0.0

Actualizar librerías

Podemos definir las dependencias vía el archivo `composer.json` o mediante comandos con el formato `composer require vendor/package:version`. Por ejemplo, si queremos añadir `phpUnit` como librería de desarrollo, haremos: `composer require phpunit/phpunit --dev`

Tras añadir nuevas librerías, hemos de actualizar nuestro proyecto: `composer update`

Si creamos el archivo `composer.json` nosotros directamente sin inicializar el repositorio, hemos de instalar las dependencias: `composer install`

Al hacer este paso (tanto instalar como actualizar), como ya hemos comentado, se descargan las librerías en dentro de la carpeta `vendor`. Es muy importante añadir esta carpeta al archivo `.gitignore` para no subirlas a GitHub.

Además, se crea el archivo `composer.lock`, que almacena la versión exacta que se ha instalado de cada librería (este archivo no se toca).

Autoload



Composer crea de forma automática en `vendor/autoload.php` el código para incluir de forma automática todas las librerías que tengamos configuradas en `composer.json`.

Para utilizarlo, en la cabecera de nuestro archivos pondremos:

```
<?php require 'vendor/autoload.php';
```

En nuestro caso, de momento sólo lo podremos en los archivos donde probamos las clases

Si queremos que Composer también se encargue de cargar de forma automática nuestras clases de dominio, dentro del archivo `composer.json`, definiremos la propiedad `autoload`.

```
"autoload": { "psr-4": {"MiNamespace\\": "mi_carpeta/"} }
```

Esto indica que las clases del namespace "MiNamespace" las tiene que ir a buscar al directorio "mi_carpeta" (suponiendo que "mi_carpeta" se encuentra colgando la raíz, por lo que sería un hermano del directorio "vendor").

Posteriormente, hemos de volver a generar el autoload de Composer mediante la opción `dump-autoload` : `composer dump-autoload`

Laravel

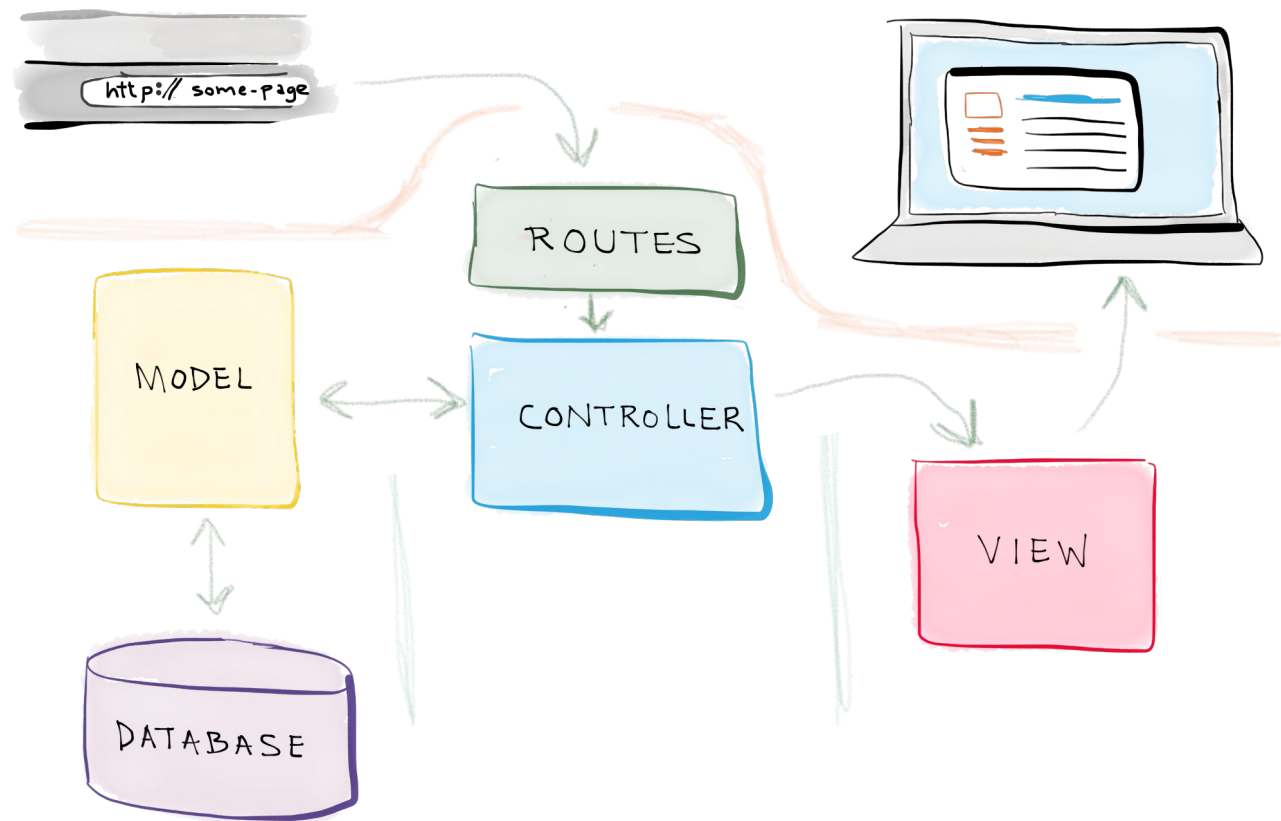


Laravel es un framework de PHP para ayudarnos en un tipo de desarrollo sobre aplicaciones escritas en este lenguaje de programación. Este framework o más bien podríamos llamarlo compañero de ahora en adelante, nos ayuda en muchas cosas al desarrollar una aplicación, por medio de sus sistema de paquetes y de ser un framework del tipo MVC (Modelo-Vista-Controlador) da como resultado que podamos “despreocuparnos” en ciertos aspecto del desarrollo, cómo instanciar clases y métodos para usarlos en muchas partes de nuestra aplicación sin la necesidad de escribirlo y repetirlos muchas veces con lo que eso conlleva a la hora de modificar algo en el código.

Funciona como muchos otros, pero desde la línea de comandos contamos con el famoso Artisan que es el nombre que le dan a esta interfaz por comandos para ejecutar muchas funcionalidades, como ver todas las rutas de la aplicación disponible, o poner a correr la aplicación o pararla. Es tan potente y sencillo de usar que una vez que lo has probado en alguna aplicación te puedes “malacostumbrar” y echarlo de menos cuando haces otra aplicación con otro framework que no cuenta con él.

Documentación Oficial MUY BUENA: <https://laravel.com/docs/9.x/>

Laravel



Laravel - características



Su [motor de plantilla](#), llamado [Blade](#), da numerosas posibilidades para hacer unas páginas visualmente muy potentes y eficaces, capaz de utilizar sus propias variables y reutilizarlas.

Su arquitectura es conocida como [MVC \(Modelo-Vista-Controlador\)](#) que da muchas facilidades para relacionar de manera clara y sencilla todas las partes de una aplicación. Esta arquitectura es muy usada en el mundo del software, otros framework pueden distintos de Laravel pueden resultar muy similares gracias a compartir la misma arquitectura MVC.

[Eloquent ORM](#), es muy intuitivo para [escribir consultas en PHP sobre objetos](#). Otros framework cuenta con Doctrine por ejemplo, otro tipo de ORM que quizás te podría sonar más que el que usa Laravel.

En [seguridad](#), ofrece un nivel bastante fuerte con mecanismos de hash y salt para encriptar por medio de librerías como BCrypt, que también lo usa por ejemplo Zend Framework.

Laravel - características

























Artisan, su **sistema de comandos** otorga al framework gran poder y a los programadores grandes facilidades y posibilidades, para crear controladores, entidades o actualizar la base de datos por ejemplo entre muchísimas cosas.

Librerías y modularidad. Laravel aparte de sus propias librerías cuenta con ayuda de Symfony en otras muchas, otro MVC de los más usados en los últimos tiempos y con una gran comunidad detrás que hace que su avance y evolución sea muy significativo. También condiciona que la evolución de Laravel en parte dependa de Symfony en estos aspectos.

Base de datos y migraciones. Permite actualizar y migrar la base de datos una vez que el desarrollo ya está comenzamos y hay cambios en el código conforme se requiera sin necesidad de borrarla y volverla a crear, gracias a esto el riesgo de perder datos sean del valor que sean es mínimo. Además, gracias a su Schema Builder hace que no requiera usar el SQL, cuenta con un sistema intuitivo en PHP para hacerlo más fácil.

Laravel - Estructura

- >  app
- >  bootstrap
- >  config
- >  database
- >  lang
- >  public
- >  resources / views
- >  routes
- >  storage
- >  tests
- >  vendor
-  .editorconfig
-  .env
-  .env.example
-  .gitattributes
-  .gitignore
-  .phpunit.result.cache
-  artisan
-  composer.json
-  composer.lock
-  phpunit.xml
-  README.md

Node.js

Node.js® is a platform built on Chrome's JavaScript runtime for easily building fast, scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.

- Basada en la **máquina virtual de Google Chrome**, lo que permitirá construir **aplicaciones de red escalables y muy rápidas**.
- Usa un modelo de **eventos no bloqueantes** y de entradas y salidas, haciéndolo eficiente y potente, perfecto para desarrollar aplicaciones en **tiempo real** y consumiéndose en **dispositivos distribuidos**.



Node.js



¿ES NODE.JS UN FRAMEWORK JS?

¡¡NO ES UN FRAMEWORK!!

ES UN ENTORNO DE PROGRAMACIÓN.

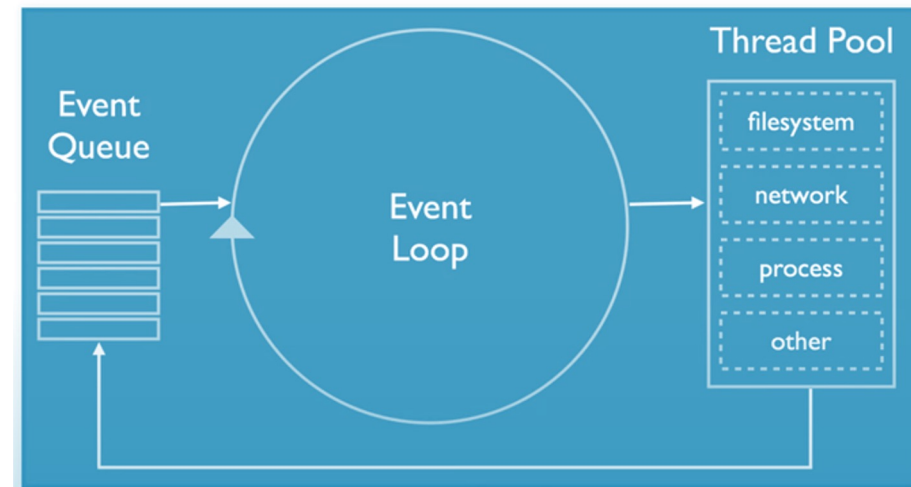
¿Cómo podemos saber que es un entorno?

¿Qué características tiene un entorno de programación?

Node.js

CARACTERÍSTICAS DE NODE.JS

- Es concurrente sin paralelismo.
- Es asíncrono y no bloqueante.
- Es orientado a eventos.
- Single thread basado en callbacks.



Node.js



<https://nodejs.org/es/> → Nos aconseja la Long Term Support y la Latest del SO de nuestro equipo.

A screenshot of the Node.js website in Spanish. The header includes the Node.js logo and navigation links: INICIO, ACERCA, DESCARGAS, DOCUMENTACIÓN, PARTICIPE, SEGURIDAD, NOTICIAS, and CERTIFICATION. The main content area states that Node.js is a JavaScript execution environment built with V8 and Chrome's JavaScript engine. Below this, there is a green banner for a security advisory. The 'Descargar para macOS' section features two green buttons: '18.12.1 LTS' (Recommended for most) and '19.3.0 Actual' (Latest features). Each button has links for 'Otras Descargas', 'Cambios', and 'Documentación de la API'. At the bottom, there is a link to the LTS support program.

nodejs

INICIO | ACERCA | DESCARGAS | DOCUMENTACIÓN | PARTICIPE | SEGURIDAD | NOTICIAS | CERTIFICATION

Node.js® es un entorno de ejecución para JavaScript construido con V8, motor de JavaScript de Chrome.

Node.js assessment of OpenSSL 3.0.7 security advisory

Descargar para macOS

18.12.1 LTS
Recomendado para la mayoría

19.3.0 Actual
Últimas características

Otras Descargas | Cambios | Documentación de la API

O eche un vistazo al Programa de soporte a largo plazo (LTS).

Node.js

- Una vez finalice la instalación, abriremos PowerShell de Windows y lanzaremos el comando `node -v` o la consola de git/linux:

```
Last login: Wed Dec  7 10:14:44 on ttys000
olga@MacBook-Pro-de-olga ~ % node -v
v18.12.1
olga@MacBook-Pro-de-olga ~ %
```

- También comprobaremos el gestor de paquetes que incorpora node, llamado NPM, mediante el comando `npm -v`

```
olga@MacBook-Pro-de-olga ~ % npm -v
8.19.2
olga@MacBook-Pro-de-olga ~ %
```

¿Qué nos van a permitir Node.js y npm?



Cuando trabajamos con Laravel, podemos y debemos hacer uso de librerías que complementen el desarrollo front-end de nuestra aplicación.

Ahora mismo el único framework que conocemos de desarrollo para cliente, es Bootstrap 5 y solo interviene en el diseño de interfaz de usuario.

Para poder complementar nuestra UI, vamos a hacer uso de Node, en especial de Vite, que nos ayudará a gestionar Bootstrap con Laravel.

No debemos olvidar que Node es un entorno de programación, mientras que Laravel es un framework de desarrollo web. Si queremos alcanzar el llamado full-stack: Tanto Laravel como Node proporcionan un desarrollo full-stack. En el front-end, Laravel emplea JavaScript y PHP en el back-end, por tanto, vamos a complementarlos.

¿Preguntas?

Olga M. Moreno Martín

