

Manual de instalación

Documentación del proyecto integrado

2º DESARROLLO DE APLICACIONES WEB

RAÚL MORA

Introducción.....	3
Arquitectura del sistema.....	3
Requisitos de hardware.....	3
Requisitos de software.....	3
Instalación.....	4
Dependencias de aplicación.....	4
Explicación general.....	4
Detalles.....	5
Composer.....	5
Node.js.....	6
Pasos a seguir.....	7
Paso 1.....	7
Paso 2.....	7
Paso 3.....	8
Paso 4.....	8
Paso 5.....	9
Verificación de instalación.....	9

Introducción

Bienvenido al **manual de instalación** del proyecto. En este documento se recoge una guía detallada **paso a paso** sobre cómo instalar la aplicación en un nuevo equipo con detalles sobre las dependencias de aplicación, así como la arquitectura que debe tener el sistema en referencia a los requisitos de hardware y software. Por último, también se proporciona un apartado para verificar que la instalación funciona correctamente.

Arquitectura del sistema

En este apartado se recogen los requisitos necesarios para realizar la instalación de la aplicación web, clasificado en dos áreas, dependiendo de si se trata de requisito de hardware o de software.

Requisitos de hardware

- **Procesador:** Al menos un procesador multicore.
- **Memoria RAM:** Al menos 4 GB de RAM.
- **Almacenamiento:** Suficiente espacio en disco para el código y la base de datos. Al menos, 30 GB sin contar el ocupado por el sistema operativo.

Requisitos de software

- Sistema operativo (Linux, Windows o macOS).
- Una **pila LAMP** con las siguientes especificaciones:
 - o Servidor web (preferentemente Apache).
 - o Base de datos MySQL.
 - o PHP **8.x**.
- Composer.
- Node.js y npm.

Instalación

Dependencias de aplicación

Explicación general

- **Dependencias de Composer:** Estas son las dependencias PHP específicas de Laravel y otros paquetes utilizados para la lógica de la aplicación, autenticación, manejo de roles, internacionalización, y herramientas para desarrollo y pruebas.
- **Dependencias de Node.js:** Estas dependencias son principalmente para el desarrollo frontend de la aplicación, incluyendo frameworks para CSS (Tailwind CSS), JavaScript (Alpine.js, axios), compilación de activos (Vite), gráficos (Chart.js), y tablas interactivas (DataTables).

Detalles

Composer

- Dependencias principales.
 - o **php**: Define la versión mínima de PHP requerida para ejecutar Laravel.
 - o **laravel/framework**: El framework Laravel en sí mismo, que proporciona la estructura y funcionalidad principal de la aplicación web.
 - o **laravel/tinker**: Herramienta de REPL (Read-Eval-Print Loop) para interactuar con la aplicación Laravel.
 - o **spatie/laravel-permission**: Librería para gestionar roles y permisos en Laravel.
 - o **spatie/laravel-translatable**: Facilita la traducción de modelos Eloquent en Laravel.
- Dependencias de desarrollo.
 - o **fakerphp/faker**: Generación de datos falsos para pruebas y desarrollo.
 - o **laravel/breeze**: Kit de inicio para la autenticación de usuarios en aplicaciones Laravel.
 - o **laravel/sail**: Conjunto de herramientas para desarrollar aplicaciones Laravel con Docker.
 - o **mockery/mockery**: Biblioteca de pruebas para PHP.
 - o **nunomaduro/collision**: Mejora la salida de errores de la consola para aplicaciones Laravel.
 - o **pestphp/pest**: Framework de pruebas para PHP.
 - o **pestphp/pest-plugin-laravel**: Plugin para Pest que agrega soporte específico para Laravel.
 - o **spatie/laravel-ignition**: Mejora la gestión de errores y depuración en Laravel.

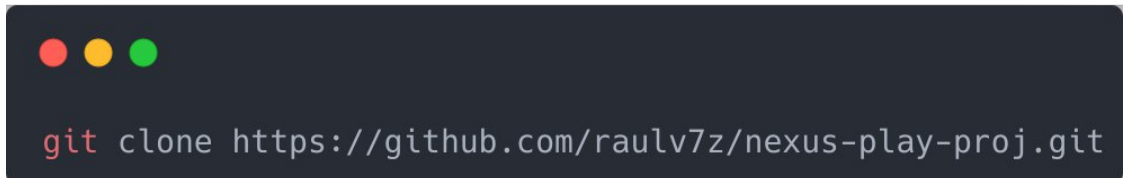
Node.js

- Dependencias de desarrollo.
 - o **@tailwindcss/forms**: Componentes de formularios para Tailwind CSS.
 - o **alpinejs**: Framework liviano de JavaScript para agregar interactividad a la interfaz de usuario.
 - o **autoprefixer**: Plugin de PostCSS que analiza el CSS y añade prefijos de proveedores según sea necesario.
 - o **axios**: Cliente HTTP basado en Promesas para el navegador y Node.js.
 - o **flowbite**: Componentes y estilos UI para aplicaciones web modernas.
 - o **laravel-vite-plugin**: Plugin para Laravel que integra Vite para la compilación de activos.
 - o **postcss**: Herramienta de transformación de CSS con soporte para plugins como autoprefixer.
 - o **tailwindcss**: Framework de CSS utility-first para construir diseños personalizados.
- Dependencias de producción.
 - o **chart.js**: Biblioteca JavaScript para generar gráficos.
 - o **datatables.net** y **datatables.net-dt**: Componentes para tablas interactivas.
 - o **datatables.net-responsive** y **datatables.net-responsive-dt**: Extensiones para tablas Datatables responsivas.
 - o **jquery**: Biblioteca JavaScript para manipulación del DOM y eventos.

Pasos a seguir

Paso 1

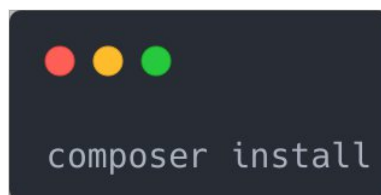
El primer paso es **solicitar el código** fuente del proyecto o, en caso de tener acceso al repositorio de GitHub, **clonarlo** de allí. Esto último se puede hacer con el siguiente comando:

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. The command `git clone https://github.com/raulv7z/nexus-play-proj.git` is entered in a light-colored monospace font.

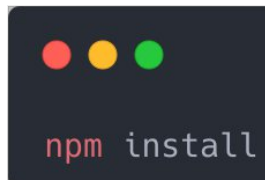
```
git clone https://github.com/raulv7z/nexus-play-proj.git
```

Paso 2

Una vez se tiene el código fuente, hay que **instalar las dependencias** del proyecto. En este caso, las dependencias de Composer y de Node (estas últimas con npm). Para ello, es necesario ejecutar los siguientes comandos:

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. The command `composer install` is entered in a light-colored monospace font.

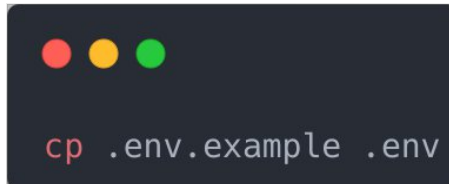
```
composer install
```

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. The command `npm install` is entered in a light-colored monospace font.

```
npm install
```

Paso 3

El siguiente paso consiste en copiar el ejemplo `.env` de ejemplo como `.env` para definir las **variables de entorno** y **configurarlas**. Para copiar el fichero de ejemplo, se puede hacer con este comando en la raíz del proyecto:

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top left corner. The command `cp .env.example .env` is displayed in a light-colored monospace font.

```
cp .env.example .env
```

Esto creará una copia del `.env.example` en la raíz del proyecto renombrado como `.env`. El `.env` ya viene preparado para desarrollo, pero en caso de necesitar alguna modificación como las credenciales para la base de datos u otros aspectos, debes ajustarlo aquí.

Paso 4

El paso 4 será **correr las migraciones**. Opcionalmente, se pueden correr los **seeders** (ficheros con instrucciones para llenar la base de datos de registros predefinidos), aunque esto último sólo debe hacerse mientras se esté **en desarrollo**, para no desvirtuar los datos reales de producción.

Para correr las migraciones sólo hay que ejecutar el siguiente comando:

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top left corner. The command `php artisan migrate` is displayed in a light-colored monospace font.

```
php artisan migrate
```

En caso de querer correr las migraciones y sus seeders, puedes ejecutar este comando directamente:

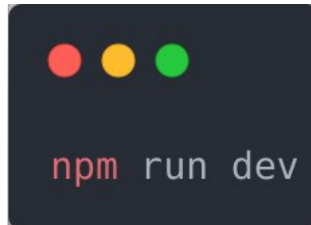
A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top left corner. The command `php artisan migrate:fresh --seed` is displayed in a light-colored monospace font.

```
php artisan migrate:fresh --seed
```

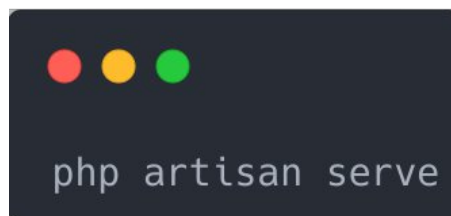

Paso 5

El proceso ya está terminado prácticamente. El único paso que queda es la **puesta en marcha** para comprobar que funciona correctamente. Lo único que hay que hacer es levantar el servidor de artisan y observar los cambios con npm.

Se puede hacer con los siguientes comandos:



```
npm run dev
```



```
php artisan serve
```

Verificación de instalación

Si has terminado los pasos para la instalación, el siguiente paso es **comprobar que todo funciona** correctamente. Por defecto, el servidor de artisan se levanta en local en el puerto 8000. Sólo hay que apuntar a esa dirección en la url para verificar que todo funciona:

