

# WordPress + Laravel?



**WORDCAMP**  
MANAGUA 2017

# GRACIAS A

**RAIN**

**tigo** business  
Una solución para cada negocio

 **BOLDGRID**

 **movistar**

 **KRONOSCODE**

 **JETPACK**

 **Woo** **COMMERCE**

 **bluehost**

 **monchi.to**  
DISEÑO & DESARROLLO WEB

 **MODERN TRIBE**

 **SiteGround**

 **dotcreek**

 **accedo**  
prestados - resultados - servicios

 **TOPFLOOR**  
MARKETING

 **DreamHost**

 **WPML.ORG**



 **SiteLock**



**UCA**  
UNIVERSIDAD  
CENTROAMERICANA

 **dinterweb**  
Inbound Marketing Agency

**SUCURI**

 **Hotel Zansión**  **Teodolinda**

 **Bacanalnica**  
.com



# Tony Dinarte

- Amo PHP
- @dinartux
- ceo@dreamdevelopment.net

# WordPress + Laravel????



# Requerimientos del cliente

- Rápido.
- Escalable.
- Social share (Open Graph friendly).
- Vistas con restricción de acceso.
- Integraciones de almacenamiento.

# Area de administracion

[Iniciar sesion en Facebook](#)

Insertar nueva publicacion en blog

[Crear nuevo](#)

Editar publicacion de blog

Eliminar publicacion

Insertar nueva foto en galeria

[Crear nuevo](#)

Editar publicacion de galeria

Eliminar publicacion de galeria

Y yo...



# Cuidado con tu Stack.

Una mala elección la pagas tú o tu cliente.





# WordPress al rescate! yey!

The image shows the WordPress dashboard for a user named 'dinartux'. The top navigation bar includes the WordPress logo, the site name 'Blog de DreamDevelopment', a comment count of '0', a '+ Nuevo' button, and a user profile dropdown. The left sidebar contains a menu with 'Escritorio' (highlighted), 'Inicio', 'Actualizaciones', 'Entradas', 'Medios', 'Páginas', 'Comentarios', 'Apariencia', 'Plugins', 'Usuarios', 'Herramientas', 'Ajustes', and 'Cerrar menú'. The main content area is titled 'Escritorio' and features a 'Bienvenido a WordPress' message with a 'Descartar' button. Below this, there are three sections: 'Comienza' with a 'Personaliza tu sitio' button and a link to change the theme; 'Sigüientes pasos' with links to write the first post, add a page, and view the site; and 'Más acciones' with links to manage widgets, activate/deactivate comments, and learn more. At the bottom, there are two sidebars: 'De un vistazo' showing 2 posts and 1 page, and 'Borrador rápido' with a text area for a quick draft.

WordPress dashboard interface showing the 'Escritorio' (Dashboard) view.

**Top Bar:** Blog de DreamDevelopment, 0 comments, + Nuevo, Hola, dinartux.

**Left Sidebar:** Inicio, Actualizaciones, Entradas, Medios, Páginas, Comentarios, Apariencia, Plugins, Usuarios, Herramientas, Ajustes, Cerrar menú.

**Main Content Area:**

- Bienvenido a WordPress** (Descartar)
- Estamos preparando algunos enlaces para que puedas comenzar:
- Comienza**
  - Personaliza tu sitio
  - o, cambia tu tema por completo
- Sigüientes pasos**
  - Escribe tu primera entrada en el blog
  - Añade una página Sobre mí
  - Ver tu sitio
- Más acciones**
  - Gestiona widgets o menús
  - Activa o desactiva los comentarios
  - Aprende más de cómo comenzar

**Bottom Section:**

- De un vistazo**
  - 2 entradas
  - 1 página
  - 1 comentario
  - WordPress 4.7.5 funciona con el tema Twenty Seventeen.
  - Motores de búsqueda disuadidos
- Borrador rápido**
  - Título
  - ¿Qué te está pasando por la cabeza?

# Razones

- Curva de Aprendizaje.
- Mindsets.
- Tiempo (Deadline).
- Costos/Recursos.



# Mindsets

## WordPress:

- Facilidad de manejo para usuario final.
- Orientado a Contenido.

## Laravel:

- Experiencia para el desarrollador.
- Entorno de herramientas de desarrollo.

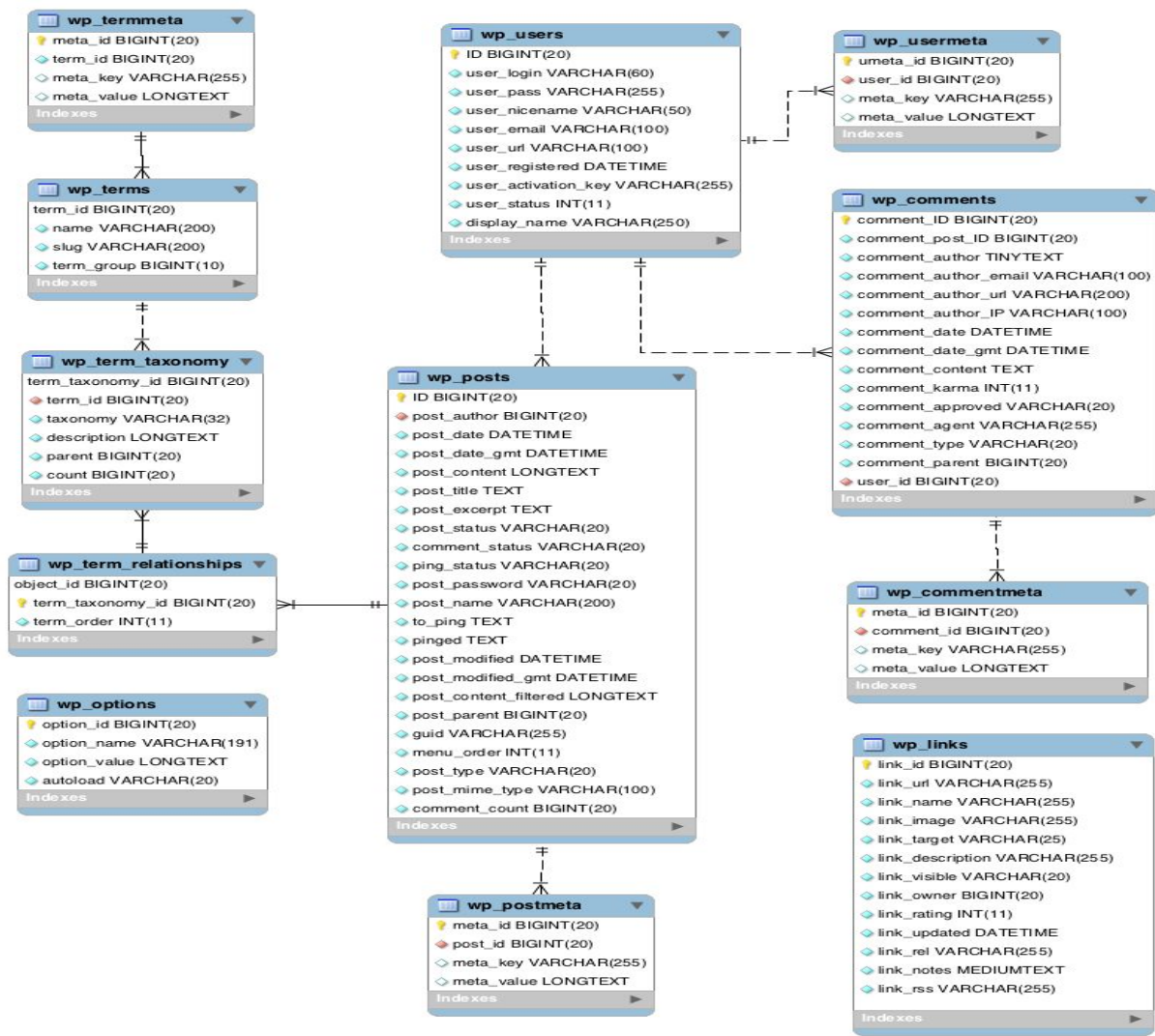
# WordPress

es

# Consistente

y eso es

# COOL!



Routing  
+  
WP REST API  
(core)





# Routing:



# Opciones para lograrlo:

- Corcel ( “jgrossi/corsel”), tratar con la DB.
- Aprovechar funcionalidad REST API.

Bonus:

- WordPressToLaravel ("jasonherndon/wptolaravel")
- laravel-wp-api (“cyberduck/laravel-wp-api”)
- Wp-Eloquent ("tareq1988/wp-eloquent")

```
/*
```

```
Agregar datos de la base de datos en archivo config/database.php
```

```
*/
```

```
'wordpress' => [ //Este es el nombre de su conexion
```

```
    'driver'    => 'mysql',
```

```
    'host'      => 'localhost',
```

```
    //Datos y credenciales de la base manejada por wordpress
```

```
    'database'  => '',
```

```
    'username'  => '',
```

```
    'password'  => '',
```

```
    'charset'   => 'utf8',
```

```
    'collation' => 'utf8_unicode_ci',
```

```
    'prefix'    => 'wp_',
```

```
    'strict'    => false,
```

```
    'engine'    => null,
```

```
],
```

Corcel



```
namespace App;  
use Corcel\Post as Corcel;
```

```
class Publicacion extends Corcel  
{  
    protected $connection = 'wordpress';  
  
    /*  
    Esto permitiria obtener las Publicaciones a manera Eloquent  
  
    App\Publicacion::all();  
  
    */  
}
```

Creando el modelo con `php artisan make:model Publicacion` donde extenderemos la clase Corcel

# WordPress REST API

(Nuestra elección)



# Primero, limpiemos.

```
[{"id":4,"date":"2017-05-22T08:30:29","date_gmt":"2017-05-22T08:30:29","guid":{"rendered":"http://blog.dream.dev/?p=4"},"modified":"2017-05-22T08:30:29","modified_gmt":"2017-05-22T08:30:29","slug":"entrada-de-prueba-1","status":"publish","type":"post","link":"http://blog.dream.dev/entrada-de-prueba-1/","title":{"rendered":"Entrada de Prueba 1"},"content":{"rendered":"<p>Lorem ipsum\u00a0Lorem ipsumLorem ipsumLorem ipsumLorem ipsumLorem ipsumLorem ipsumLorem ipsumLorem ipsumLorem ipsumLorem ipsumLorem ipsumLorem ipsumLorem ipsumLorem ipsumLorem ipsumLorem ipsumLorem ipsumLorem ipsum</p>\n","protected":false},"excerpt":{"rendered":"<p>Lorem ipsum\u00a0Lorem ipsumLorem ipsumLorem ipsumLorem ipsumLorem ipsumLorem ipsumLorem ipsumLorem ipsumLorem ipsumLorem ipsumLorem ipsumLorem ipsumLorem ipsumLorem ipsumLorem ipsumLorem ipsumLorem ipsum</p>\n","protected":false},"author":1,"featured_media":0,"comment_status":"open","ping_status":"open","sticky":false,"template":"","format":"standard","meta":[],"categories":[1],"tags":[],"_links":{"self":[{"href":"http://blog.dream.dev/wp-json/wp/v2/posts/4"}],"collection":[{"href":"http://blog.dream.dev/wp-json/wp/v2/posts"}],"about":[{"href":"http://blog.dream.dev/wp-json/wp/v2/types/post"}],"author":[{"embeddable":true,"href":"http://blog.dream.dev/wp-json/wp/v2/users/1"}],"replies":[{"embeddable":true,"href":"http://blog.dream.dev/wp-json/wp/v2/comments?post=4"}],"version-history":[{"href":"http://blog.dream.dev/wp-json/wp/v2/posts/4/revisions"}],"wp:attachment":[{"href":"http://blog.dream.dev/wp-json/wp/v2/media?parent=4"}],"wp:term":[{"taxonomy":"category","embeddable":true,"href":"http://blog.dream.dev/wp-json/wp/v2/categories?post=4"}, {"taxonomy":"post_tag","embeddable":true,"href":"http://blog.dream.dev/wp-json/wp/v2/tags?post=4"}],"curies":[{"name":"wp","href":"https://api.w.org/{rel}","templated":true}]}],{"id":1,"date":"2017-05-22T08:28:15","date_gmt":"2017-05-22T08:28:15","guid":{"rendered":"http://blog.dream.dev/?p=1"},"modified":"2017-05-22T08:28:15","modified_gmt":"2017-05-22T08:28:15","slug":"hola-mundo","status":"publish","type":"post","link":"http://blog.dream.dev/hola-mundo/","title":{"rendered":"\u00a1Hola mundo!"},"content":{"rendered":"<p>Bienvenido a WordPress. Esta es tu primera entrada. Ed\u00edtala o b\u00fcrala, \u00a1y comienza a escribir!</p>\n","protected":false},"excerpt":{"rendered":"<p>Bienvenido a WordPress. Esta es tu primera entrada. Ed\u00edtala o b\u00fcrala, \u00a1y comienza a escribir!</p>\n","protected":false},"author":1,"featured_media":0,"comment_status":"open","ping_status":"open","sticky":false,"template":"","format":"standard","meta":[],"categories":[1],"tags":[],"_links":{"self":[{"href":"http://blog.dream.dev/wp-json/wp/v2/posts/1"}],"collection":[{"href":"http://blog.dream.dev/wp-json/wp/v2/posts"}],"about":[{"href":"http://blog.dream.dev/wp-json/wp/v2/types/post"}],"author":[{"embeddable":true,"href":"http://blog.dream.dev/wp-json/wp/v2/users/1"}],"replies":[{"embeddable":true,"href":"http://blog.dream.dev/wp-json/wp/v2/comments?post=1"}],"version-history":[{"href":"http://blog.dream.dev/wp-json/wp/v2/posts/1/revisions"}],"wp:attachment":[{"href":"http://blog.dream.dev/wp-json/wp/v2/media?parent=1"}],"wp:term":[{"taxonomy":"category","embeddable":true,"href":"http://blog.dream.dev/wp-json/wp/v2/categories?post=1"}, {"taxonomy":"post_tag","embeddable":true,"href":"http://blog.dream.dev/wp-json/wp/v2/tags?post=1"}],"curies":[{"name":"wp","href":"https://api.w.org/{rel}","templated":true}]}
```

```
// http://blog.dream.dev/wp-json/wp/v2/posts?fields=id,title,link
```

```
[  
  {  
    "id": 4,  
    "link": "http://blog.dream.dev/entrada-de-prueba-1/",  
    "title": {  
      "rendered": "Entrada de Prueba 1"  
    }  
  },  
  {  
    "id": 1,  
    "link": "http://blog.dream.dev/hola-mundo/",  
    "title": {  
      "rendered": "¡Hola mundo!"  
    }  
  }  
]
```

WP REST API- filter fields



# Bonus: Better REST API Featured images.

```
"better_featured_image": {
  "id": 13,
  "alt_text": "Hot Air Balloons",
  "caption": "The event featured hot air balloon rides",
  "description": "The hot air balloons from the big event",
  "media_type": "image",
  "media_details": {
    "width": 5760,
    "height": 3840,
    "file": "2015/09/balloons.jpg",
    "sizes": {
      "thumbnail": {
        "file": "balloons-150x150.jpg",
        "width": 150,
        "height": 150,
        "mime-type": "image/jpeg",
        "source_url": "http://api.example.com/wp-content/uploads/2015/09/balloons-150x150.jpg"
      },
      "medium": {
        "file": "balloons-300x200.jpg",
        "width": 300,
        "height": 200,
        "mime-type": "image/jpeg",
        "source_url": "http://api.example.com/wp-content/uploads/2015/09/balloons-300x200.jpg"
      },
      "large": {
        "file": "balloons-1024x683.jpg",
        "width": 1024,
        "height": 683,
        "mime-type": "image/jpeg",
        "source_url": "http://api.example.com/wp-content/uploads/2015/09/balloons-1024x683.jpg"
      }
    }
  }
},
```

# Empecemos con Laravel:

```
/*  
URL de nuestro blog y su path al JSON API  
*/  
protected $url = 'http://blog.dream.dev.com/wp-json/wp/v2/';  
  
/*  
Primero pedimos el contenido, lo traemos y manejamos como JSON  
*/  
protected function getContent($url)  
{  
    $response = file_get_contents($url, false);  
    return json_decode($response);  
}
```

Creamos una clase WpApi en la ruta app/WpApi.php para aprovechar el namespace global de laravel

```

/*
Aca mapeamos los datos a base, iterando sobre una coleccion
creada del archivo JSON que nos da el API
*/

public function importPosts($page = 1)
{
    if ($page !=1) {
        $posts = collect($this->getContent($this->url .
            'posts?fields=id,title,better_featured_image,date,modified,content&page=' . $page));
        foreach ($posts as $post) {
            $this->savePost($post);
        }
    }

    $posts = collect($this->getContent($this->url .
        'posts?fields=id,title,better_featured_image,date,modified,content'));
    foreach ($posts as $post) {
        $this->savePost($post);
    }
}

```

Método para guardar los publicaciones, creando una coleccion en \$posts usando el helper collect( )

```
protected function savePost($data)
{
    $found = Post::where('id', $data->id)->first();

    if (! $found) {
        return $this->createPost($data);
    }

    if ($found && $found->updated_at->format("Y-m-d H:i:s") <
        $this->carbonDate($data->modified)->format("Y-m-d H:i:s")) {
        return $this->updatePost($found, $data);
    }
}
```

Método para determinar si se debe guardar o actualizar publicaciones, comparando fechas (forma para no usar revisions en el API de WordPress)



```
protected function carbonDate($date)
{
    return Carbon::parse($date);
}

protected function createPost($data)
{
    $post = new Post();
    $post->id = $data->id;
    $post->featured = ($data->better_featured_image) ? 1 : null;
    $post->content = $data->content->rendered;
    $post->published_at = $this->carbonDate($data->date);
    $post->created_at = $this->carbonDate($data->date);
    $post->updated_at = $this->carbonDate($data->modified);
    $post->save();
    return $post;
}

protected function updatePost($old, $new)
{
    $old->featured = ($new->better_featured_image) ? 1 : null;
    $old->content = $new->content->rendered;
    $old->updated_at = $this->carbonDate($new->modified);
    $old->save();
    return $old;
}
```

Métodos para actualizar o guardar publicaciones ya habiendo comparando fechas en la función anterior, usando instancias de un modelo Post que crearemos.

Usando el comando ``php artisan make:model Post -m`` creamos los archivos de migración (database/migrations/) y modelo (app/Post.php). Nuestra migración para este ejemplo se debería mirar así.

```
public function up()
{
    Schema::create('posts', function (Blueprint $table) {
        $table->increments('id');
        $table->date('published_at');
        $table->bool('featured');
        $table->text('content');
        $table->timestamps();
    });
}

/**
 * Reverse the migrations.
 *
 * @return void
 */
public function down()
{
    Schema::dropIfExists('posts');
}
```

# Finalmente...

usando el comando de  
artisan `php artisan  
make:command  
Importer` agregamos la  
lógica del comando en  
app/Console/Commands  
/Importer.php

Comando que  
agregaremos al scheduler  
de laravel en  
app/Console/Kernel.php

(<https://laravel.com/docs/5.4/scheduling> para mas  
información del  
scheduling en laravel)

```
class Importer extends Command
{
    protected $wpApi;
    protected $signature = 'import:wordpress {page? : Argumento para especifica
        numero de pagina de publicaciones}
    {--checkUpdates : Bandera para revisar actualizaciones en las publicaciones}

    protected $description = 'Comando para extraer publicaciones';

    public function __construct()
    {
        parent::__construct();
        $this->wpApi= new WpApi();
    }

    public function handle($page = 1)
    {
        $page = $this->argument('page');
        $this->wpApi->importPosts();
    }
}
```

Listo!



Gracias por su atención.

