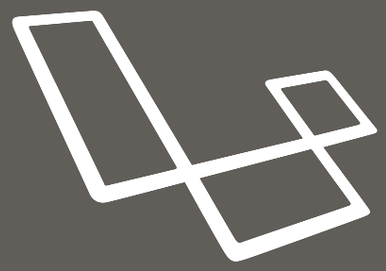
LARAVEL 5

Server-Side - framework PHP

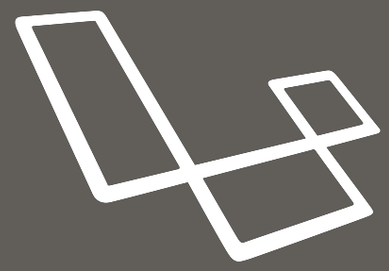
Oscar Fernando Aristizábal Cardona

INTRODUCCIÓN



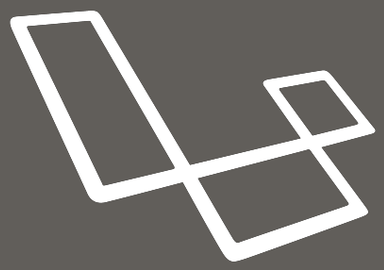
- **Laravel** es un framework de código abierto para desarrollar aplicaciones y servicios web con PHP 5.
- Su filosofía es desarrollar código PHP de forma elegante y simple, evitando "código espagueti".

CARACTERÍSTICAS



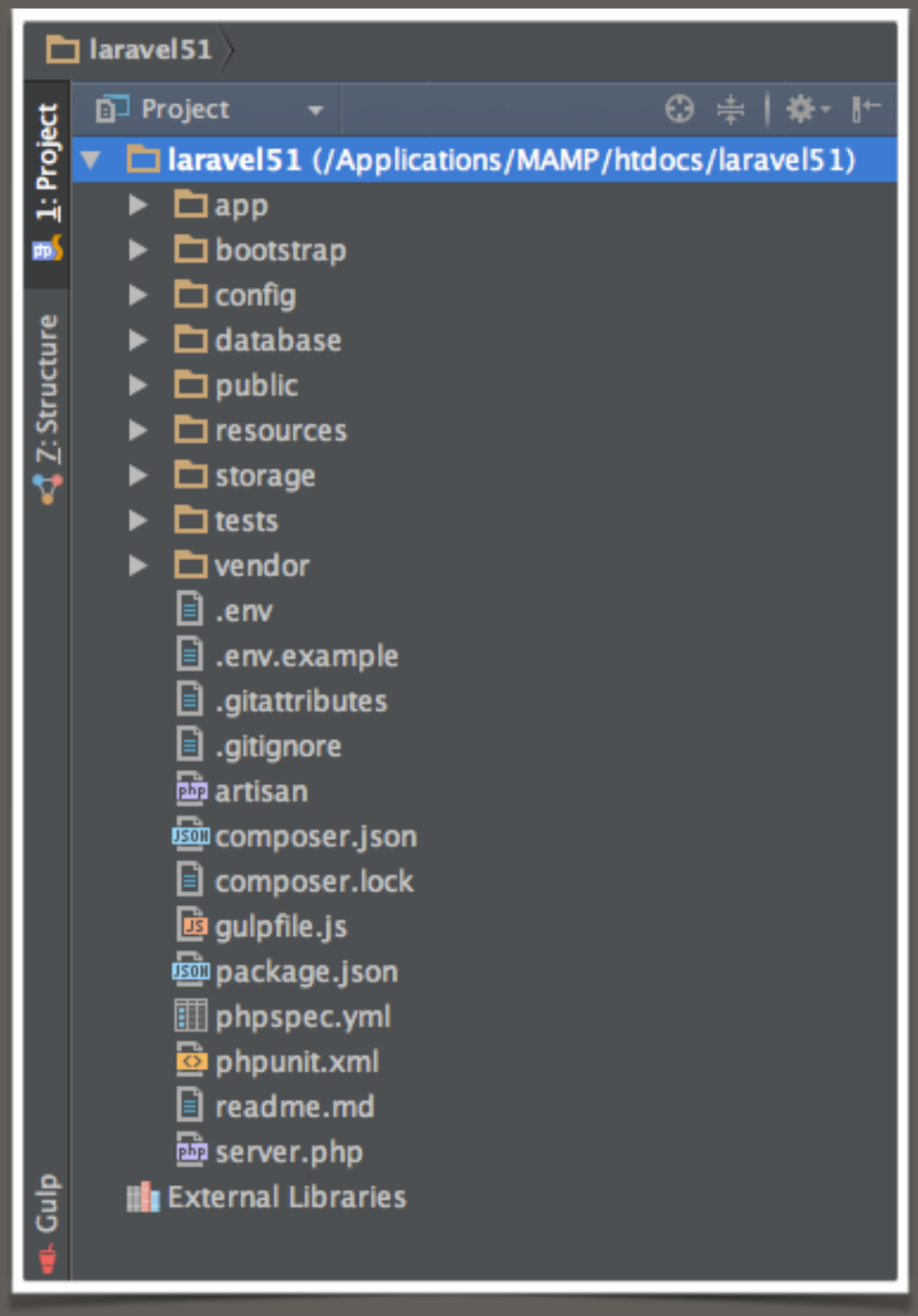
- Sistema de ruteo, también RESTful
- Motor de plantillas (Blade)
- Peticiones (Fluent)
- Eloquent ORM (*Object Relational Mapper*)
- Basado en Composer
- Soporte para patrón MVC

REQUERIMIENTOS

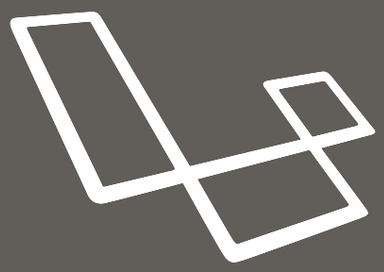


- PHP >= 5.5.9
- Extensión PHP: OpenSSL
- Extensión PHP: PDO
- Extensión PHP: Mbstring
- Extensión PHP: Tokenizer

ESTRUCTURA DIRECTORIOS



INSTALAR COMPOSER



- **Descargar Instalador:**

  https://getcomposer.org/Composer-Setup.exe

- **Comandos:**

```
// Ver la versión de composer
>> composer --version

// Actualizar composer
>> composer self-update

// Descargar instalador de laravel usando composer
>> composer global require “laravel/installer"

// Crea un proyecto de laravel limpio
>> laravel new nombreapp

// Crea un proyecto de laravel limpio desde composer
>> composer create-project laravel/laravel nombreapp
```

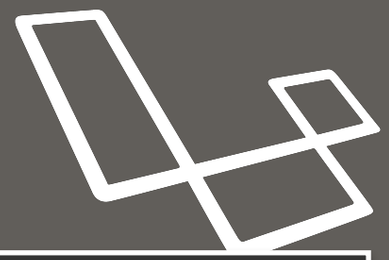
COMANDOS ARTISAN

```
// General ------------------------------------ //

// Lista de comandos
>> php artisan list

// Ayuda de un commando determinado
>> php artisan help migrate

// Proyecto ----------------------------------- //

// Iniciar Servidor Local
>> php artisan serve

// Generador de la llave de la aplicación
>> php artisan key:generate

// Cambiar de nombre a la aplicación
>> php artisan app:name NombreApp

// Modo Mantenimiento (sitio offline)
>> php artisan down

// Modo (online)
>> php artisan up
```

COMANDOS ARTISAN

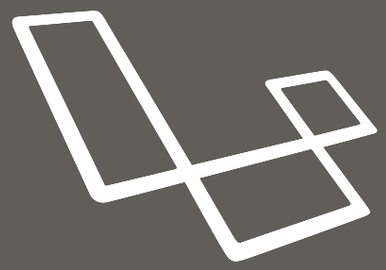
```
// Controlador ------------------------------------ //

// Crear un controlador
>> php artisan make:controller ArticleController --resource

// Crear un controlador plano
>> php artisan make:controller TestController


// Modelo ----------------------------------------- //

// Crear un modelo
>> php artisan make:model Article


// Rutas ------------------------------------------ //

// Listar rutas
>> php artisan route:list

// Guardar rutas en cache
>> php artisan route:cache

// Limpiar rutas
>> php artisan route:clear
```

COMANDOS ARTISAN



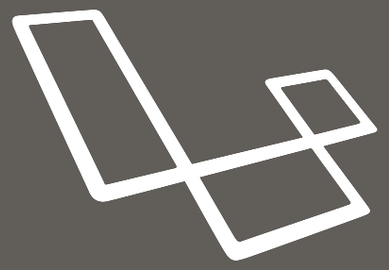
```
// Migraciones ------------------------------------ //

// Crear una migración
>> php artisan make:migration create_articles_table
>> php artisan make:migration create_articles_table --create=articles

// Ejecutar una migración
>> php artisan migrate


// Deshacer la última migración
>> php artisan migrate:rollback

// Deshacer todas las migraciones
>> php artisan migrate:reset

// Deshacer todas las migraciones y luego ejecuta el comando migrate
>> php artisan migrate:refresh


// Importante: Si al ejecutar migrate muestra error "class not found"
>> composer dump-autoload
```

COMANDOS ARTISAN



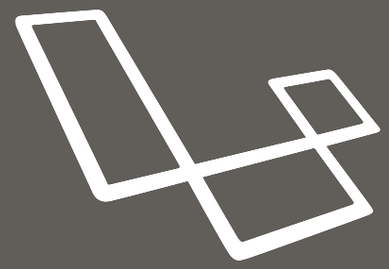
```
// Semillas ------------------------------------ //

// Generar semilla
>> php artisan make:seeder ArticleTableSeeder

// Sembrar semilla
>> php artisan db:seed

// Sembrar semilla al refrescar una migración
>> php artisan migrate:refresh --seed


// Peticiones ---------------------------------- //

// Generar petición
>> php artisan make:request ArticleRequest
```

CRUD
1. LISTAR REGISTROS



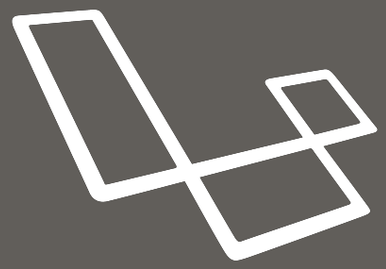
```php
// ArticleController.php      ---------------------------------- //
// Listar todos los registros ---------------------------------- //

// Función Inicial / defecto
public function index() {

    // Sin paginación
    // $query = Article::all();

// Forma larga con paginación
/* $query = DB::table('articles')->paginate(6);
   $query->setPath('article');
   return view('article.index', compact(‘query'));
*/

// Forma corta con paginación
    return view("article.index")
           ->with('query', Article::paginate(6)
           ->setPath('article'));
}
```

CRUD
2. LISTAR REGISTROS



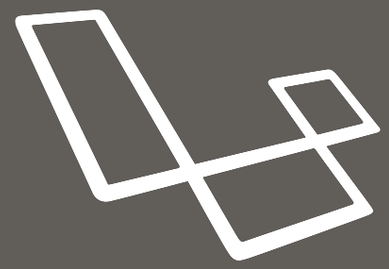
```
// Vista: articles/index.blade.php ---------------------------------- //


@if (!$query->isEmpty())
  @foreach ($query as $row)
    {{ $row->id }}
    {{ $row->title }}
    <a href="{{ url('article/'.$row->id) }}">Show</a>
    <a href="{{ url('article/'.$row->id.'/edit') }}">Edit</a>
    <form action="{{ url('article/'.$row->id) }}" method="post">
       <input type="hidden" name="_token" value="{{ csrf_token() }}">
       <input type="hidden" name="_method" value="delete">
       <input type="button" class="btn-destroy" value="Delete">
    </form>
  @endforeach
 // Con paginación
 // {!! $query->render() !!}
@else
  <h3>No existen Registros!</h3>
@endif
```

CRUD
1. CREAR REGISTRO



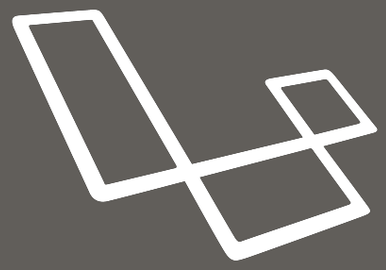
```php
// ArticleController.php       ----------------------------------- //
// Crear registros             ----------------------------------- //

// Función cargar vista “create”
public function create() {
     return view('article.create');
}

// Función almacena datos
public function store(ArticleRequest $request) {
     Article::create($request->all());
     return redirect('article')->with('message', 'Articulo guardado!’);
     // return redirect(‘article'); // Redireccionar sin mensajes
}
```

CRUD
2. CREAR REGISTRO



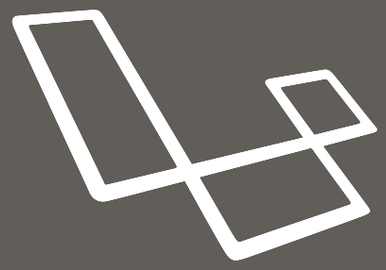
```
// Vista: articles/create.blade.php ----------------------------------- //


<form action="{{ url('article') }}" method="post">
    <input type="hidden" name="_token" value="{{ csrf_token() }}">

    <input type="text" name="title" value="{{ old('title') }}">
    <textarea name="body">{{ old('body') }}</textarea>
    <input type="submit" value="Save">
</form>

<a href="{{ url('article') }}"> Back</a>
```

CRUD
3. CREAR REGISTRO



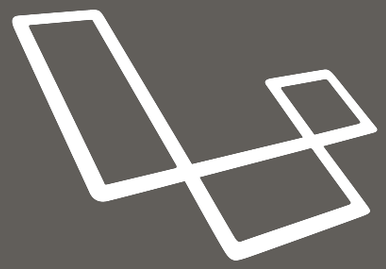
```php
// Validar Campos
// Requests/ArticlesRequest.php ---------------------------------- //

public function rules() {
    return [
        'title' => 'required|min:5|max:10',
        'body'  => 'required|min:10|max:128'
    ];
}

public function messages() {
    return [
        'title.required' => 'El campo title es requerido!',
        'title.min'      => 'El campo title no puede tener menos de 5 carácteres',
        'title.max'      => 'El campo title no puede tener más de 10 carácteres',
        'body.required'  => 'El campo body es requerido!',
        'body.min'       => 'El campo body no puede tener menos de 10 carácteres',
        'body.max'       => 'El campo body no puede tener más de 128 carácteres'
    ];
}
```

CRUD
1. MOSTRAR REGISTRO



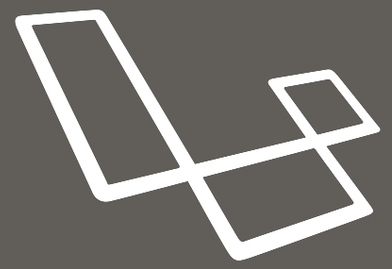
```php
// ArticleController.php       ------------------------------------ //
// Mostrar registro            ------------------------------------ //

// Función mostrar “show”
public function show($id)
{
     $query = Article::find($id);
     return view('article.show', compact('query'));
}
```

CRUD
2. MOSTRAR REGISTRO



```
// Vista: articles/show.blade.php ---------------------------------- //

{{ $query->id }}
{{ $query->title }}
{{ $query->body }}
{{ $query->created_at }}
{{ $query->updated_at }}

<a href="{{ url('article') }}"> Back</a>
```

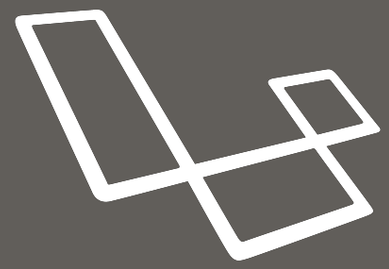
CRUD
1. MODIFICAR REGISTRO



```php
// ArticleController.php       ------------------------------------ //
// Modificar registro          ------------------------------------ //

public function edit($id)
{
    // $query = Article::find($id);
    // return view('article.edit', compact('query'));

    return view('article.edit')->with('query', Article::find($id));
}
```

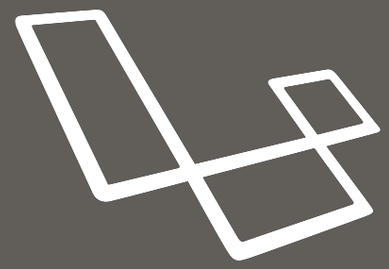
```php
public function update(ArticleRequest $request, $id)
{
    /* Article::where('id', $id)
       ->update(['title' => $request->get('title'),'body' => $request->get('body')]);
    */

    $art = Article::find($id);
    $art->title = $request->get('title');
    $art->body  = $request->get('body');
    $art->save();

    return redirect('article')->with('message', 'Articulo modificado!');
}
```

CRUD
2. MODIFICAR REGISTRO



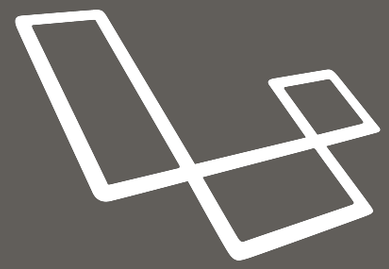
```
// Vista: articles/edit.blade.php ---------------------------------- //


<form action="{{ url(‘article/‘.$query->id) }}" method="post">
    <input type="hidden" name="_token" value="{{ csrf_token() }}”>
    <input type="hidden" name="_method" value="PUT">

    <input type="text" name="title" value="{{ $query->title }}">
    <textarea name="body">{{ $query->body }}</textarea>
    <input type="submit" value="Update">
</form>

<a href="{{ url('article') }}"> Back</a>
```

CRUD
ELIMINAR REGISTRO



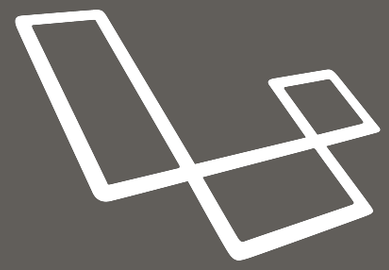
```php
// ArticleController.php     ____________________________________ //
// Eliminar registro         ____________________________________ //


// Función eliminar registro
public function destroy($id)
{
     // $art = Article::find($id);
     // $art->delete();
     // return redirect('article');

     Article::destroy($id);
     return redirect('article')->with('message', 'Articulo eliminado');
}
```

AUTH
AUTENTICACIÓN



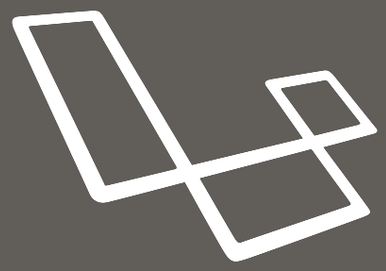
```php
use Auth;

// Si es usuario visitante
Auth::guest()

// Si el usuario ha iniciado sesión
Auth::check()

// Retorna el valor de la credencial “id” del usuario que ha iniciado sesión
Auth::user()->id

Auth::logout(); // Cerrar sesión

// Autenticación de Laravel por defecto, utilizando Middleware
// Desde método constructor
public function __construct()
{
  $this->middleware(‘auth’); // Todo necesita autenticación
  // ó
  $this->middleware(‘auth’, [‘except’ => [‘nombremetodo’]]); // Excepto ‘nombremetodo’
}

// Desde una ruta
Route::get('/', ['middleware' => 'auth']);
```

AUTH
AUTENTICACIÓN



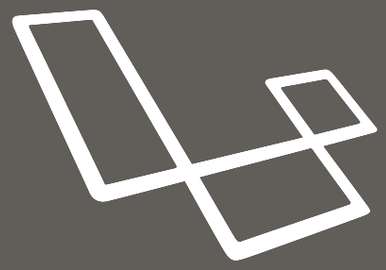
```
// Autenticación ------------------------------------ //
// Genera todas las vistas y rutas para la autenticación.

// Generar Auth
>> php artisan make:auth
```

SESSION
SESIONES



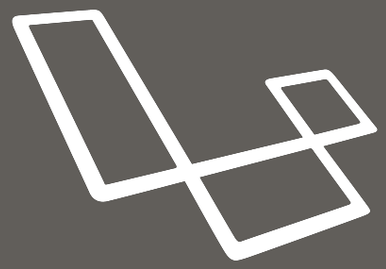
```php
use Session;

// Definir variable de sesión
Session::put('nombrevar', 'varlor');


// Obtener valor de la variable de sesión
Session::get('nombrevar');
```

RELATIONSHIP
RELACIONES



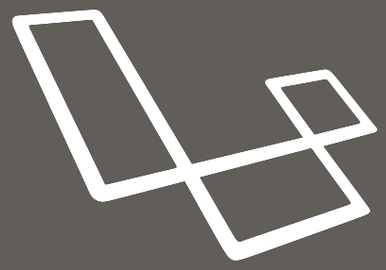
```php
// Modelo: Article.php
// Un artículo pertenece a un usuario
public function user() {
    return $this->belongsTo('App\User');
}

// Modelo: User
// Un usuario tiene un teléfono
public function phone() {
    return $this->hasOne('App\Phone');
}

// Un usuario tiene muchos artículos
public function articles() {
    return $this->hasMany('App\Article');
}

{{ $query->user->name  }} // Mostrar nombre del usuario que creo un artículo
$phone = User::find(1)->phone; // Mostrar datos del teléfono del usuario
$articles = App\User::find(1)->articles;
foreach ($articles as $article) {
    // Mostrar los artículos del usuario
}
```

UPLOAD FILES
SUBIR ARCHIVOS



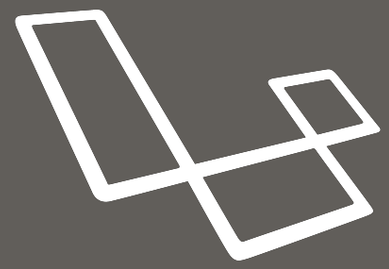
```
// Vista: create.blade.php

<form action="{{ url('articles') }}" method="post" enctype="multipart/form-data">
  <input type="file" name="image" accept="image/png">
</form>

// Solicitud: ArticleRequest.php -> rules()

'image' => 'required|mimes:png|max:1000',


// Controlador: ArticleController.php

$file    = $request->file('image');
$urlfile = $file->getClientOriginalName();

$art = new Article();
$art->image = '/images/'.$urlfile;
$art->save();

$request->file('image')->move(base_path().'/public/images/', $urlfile);
```

PDF
1. EXPORTAR PDF



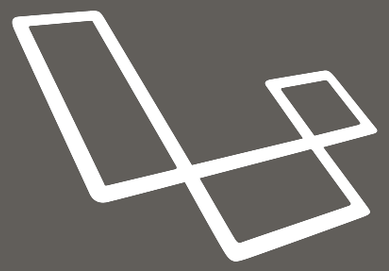
```
// composer.json
"require": {
    "barryvdh/laravel-dompdf": "0.6.*",
},

// Linea de comandos
>> composer update

// config/app.php

// Providers
Barryvdh\DomPDF\ServiceProvider::class,

// Aliases
'PDF' => ‘Barryvdh\DomPDF\Facade',

// Ruta: routes.php
Route::get('/showpdf', 'ArticleController@showpdf');
```

PDF
2. EXPORTAR PDF



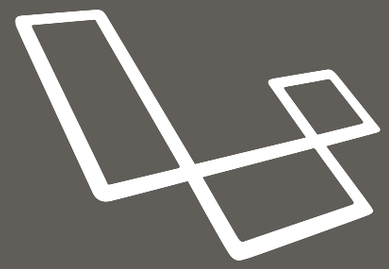
```php
// Controlador: ArticleController.php
public function showpdf()
{
    $articles = Article::all();

    $view = \View::make('articles.pdf', compact('articles'))->render();
    $pdf  = \App::make('dompdf.wrapper');
    $pdf->loadHTML($view);
    return $pdf->download('articlesfile');
}

// Vista: articles/pdf.blade.php

@foreach($articles as $article)
<tr>
    <td>{{ $article->id }}</td>
    <td>{{ $article->title }}</td>
    <td>{{ $article->body }}</td>
</tr>
@endforeach
```

EXCEL
1. EXPORTAR EXCEL



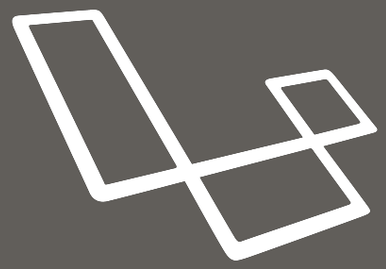
```
// composer.json
"require": {
    "maatwebsite/excel": "~2.0",
},

// Linea de comandos
>> composer update

// config/app.php

// Providers
Maatwebsite\Excel\ExcelServiceProvider::class,


// Aliases
'Excel' => 'Maatwebsite\Excel\Facades\Excel',


// Ruta: routes.php
Route::get('/showexcel', 'ArticleController@showexcel');
```

EXCEL
2. EXPORTAR EXCEL



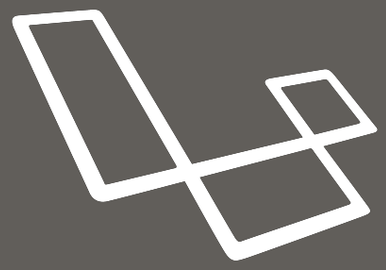
```php
// Controlador: ArticleController.php
public function showexcel()
{
    \Excel::create('articlesfile', function($excel) {
      $excel->sheet('List', function($sheet) {
          $articles = Article::all();
          $sheet->loadView('articles.excel', array('articles' => $articles));
      });
    })->download('xls');

}

// Vista: articles/excel.blade.php

@foreach($articles as $article)
<tr>
    <td>{{ $article->id }}</td>
    <td>{{ $article->title }}</td>
    <td>{{ $article->body }}</td>
</tr>
@endforeach
```

SEARCH
BUSCAR



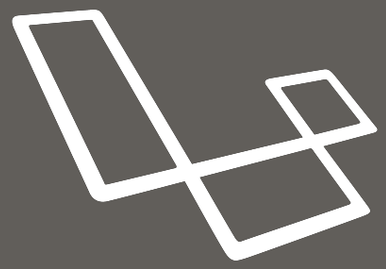
```php
// Ruta: routes.php
Route::get('/search', 'ArticleController@search');

// Controlador: ArticleController.php
public function search(Request $request)
{
    $query = Article::title($request->get('title'))->orderBy('id', 'ASC')
        ->paginate(10)
        ->setPath('articles');
    return view('articles.index', compact('query'));
}

// Modelo: Article.php
public function scopeTitle($query, $title)
{
    if(trim($title) != '')
    {
        $query->where('title', "LIKE", "%$title%");
    }
}

// Vista: articles/index.blade.php
<form action="{{ url('/search') }}">
        <input type="text" name="title" autocomplete="off">
        <button type="submit">Buscar</button>
</form>
```

AJAX SEARCH
1. BUSCAR



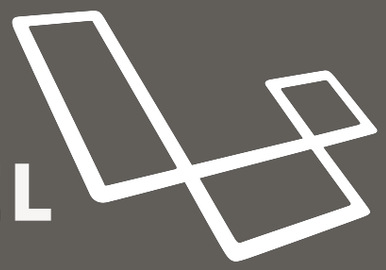
```php
// Ruta: routes.php
Route::get('/ajaxsearch', 'ArticleController@ajaxsearch');

// Controlador: ArticleController.php
public function ajaxsearch(Request $request)
{
    $query = Article::title($request->get('title'))->orderBy('id', 'ASC')
        ->paginate(10)
        ->setPath('articles');
    return view('articles.ajax', compact('query'));
}

// Modelo: Article.php
public function scopeTitle($query, $title)
{
    if(trim($title) != '')
    {
        $query->where('title', "LIKE", "%$title%");
    }
}

// Vista: articles/index.blade.php
<form action="{{ url('/search') }}">
        <input type="text" name="title" id=“stitle" autocomplete="off">
        <button type="submit">Buscar</button>
</form>
```

AJAX SEARCH
2. BUSCAR



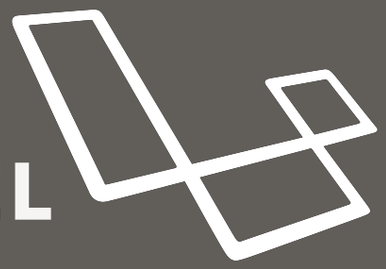
```
// Vista: app.blade.php
$('#stitle').keyup(function() {
    $title = $(this).val();
    $.get('/ajaxsearch', {title: $title}, function(data) {
        $('.table').html(data);
    });
});


// Vista: articles/ajax.blade.php
<tr>
    <th>Id</th>
    <th>Título</th>
</tr>
@foreach ($query as $var)
    <tr>
        <td>{{ $var->id }}</td>
        <td>{{ $var->title }}</td>
    </tr>
@endforeach
```

TINKER
1. LINEA DE COMANDOS LARAVEL



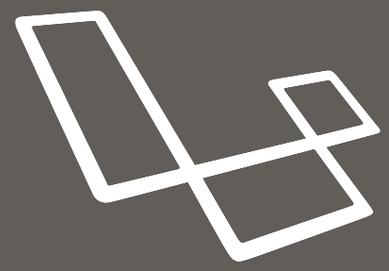
```
// CRUD Articles ---------------------------------- //

// Linea de comandos Laravel
>> php artisan tinker

// Crear Objeto
>> $user = new App\User;

// Insertar Usuario
>> $user->name = 'Jeremias Springfield';
>> $user->username = 'jeremias';
>> $user->email = 'jeremias@gmail.com';
>> $user->password = bcrypt('secreto');
>> $user->save();

// Consultar Usuario
>> $user->find(1);
>> $user->findOrFail(1);
>> $user->where('username', 'jeremias')->get();
>> $user->all();
>> $user->count();
```

TINKER
2. LINEA DE COMANDOS LARAVEL

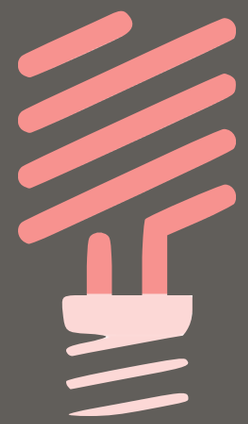


```
// CRUD Articles ---------------------------------- //


// Modificar Usuario
>> $userm = $user->find(1);
>> $userm->username = 'homero';
>> $userm->email = 'homero@gmail.com';
>> $userm->save();

// Eliminar Usuario
>> $userm->destroy(1);
```

LUMEN



- El micro-framework increíblemente rápido por laravel.



```
// Descargar instalador de lumen usando composer
composer global require "laravel/lumen-installer=~1.0"

// Crea un proyecto de lumen limpio
>> lumen new nombreapp

// Crea un proyecto de lumen limpio desde composer
>> composer create-project laravel/lumen --prefer-dist
```