**CLASE FS**

**EXPORTAR A EXCEL**

<http://www.maatwebsite.nl/laravel-excel/docs/getting-started>

* En composer.json en require colocamos: "maatwebsite/excel": "~2.1.0"
* Ejecutamos en la consola: composer update
* Incluimos los providers y los aliasses
  + Provider: Maatwebsite\Excel\ExcelServiceProvider::class,
  + Aliases: 'Excel' => Maatwebsite\Excel\Facades\Excel::class,
* Ejecutamos en la consola: *php artisan vendor:publish --provider="Maatwebsite\Excel\ExcelServiceProvider"*

**Controlador:**

*php artisan make:controller ExcelController*

*<?php*

*use Maatwebsite\Excel\Facades\Excel;*

*use App\Cliente;*

*class ExcelController extends Controller{*

*public function index(){*

*Excel::create('Clientes', function($excel) {*

*$excel->sheet('Clientes', function($sheet) {*

*$clientes = Cliente::all();*

*$sheet->fromArray($clientes);*

*});*

*})->export('xlsx');*

*}*

*}*

**Ruta**

*Route::resource('excel','ExcelController');*

**Vista (en clientes.index)**

*<div class="text-center">*

*<a class="btn btn-info" href="{{route('excel.index')}}" target="blank">Exporta a Excel</a>*

*</div>*

**IMPORTAR EXCEL**

**Ruta:** *Route::get('importar',['as'=>'importar','uses'=>'ImportarController@importar']);*

**Controlador**

*<?php*

*use Maatwebsite\Excel\Facades\Excel;*

*use App\Cliente;*

*class ImportarController extends Controller{*

*public function importar(){*

*Excel::load('clientes8.xlsx', function($reader) {*

*foreach ($reader->get() as $cliente) {*

*Cliente::create([*

*'nombre' => $cliente->nombre,*

*'nit' => $cliente->nit,*

*'direccion' => $cliente->direccion,*

*'telefono' => $cliente->telefono,*

*'email' => $cliente->email,*

*'ciudad' => $cliente->ciudad,*

*]);*

*}*

*});*

*return Cliente::all();*

*}*

*}*

*TRANSACCIONES*

**MIDDLEWARES**

Los middleware proporcionan un mecanismo conveniente para filtrar peticiones HTTP que ingresan a la aplicación. La finalidad de este componente es disminuir la carga de trabajo en los controladores y proporcionar una solución mucho más simple y estándar a la hora de aplicar las restricciones necesarias en el proyecto. Los middleware te permiten agregar restricciones por zona horaria, geolocalización, login, edad, genero, rol o tipo de usuario, y básicamente todo lo que quieras. Esto te permite bloquear acceso a ciertas áreas de tu aplicación para un usuario o un grupo de usuarios en específico

Por ejemplo, Laravel incluye un middleware que verifica que el usuario de tu aplicación está autenticado. Si el usuario no está autenticado, el middleware redirigirá al usuario a la pantalla de inicio de sesión. Sin embargo, si el usuario es autenticado, el middleware permitirá que la solicitud continúe en la aplicación.

**Validar autenticación**

Middleware En la ruta:

*Route::get('/', function () { return view('index'); })->middleware('auth');*

Middleware en el Controller

*public function \_\_construct(){*

*$this->middleware('auth');*

*}*

Variación con una excepción en una función del Controlador

*public function \_\_construct(){*

*$this->middleware('auth',['except'=>'index']);*

*}*

**Validar roles de usuario:**

\* Loguerme – desloguearme y explicar que los ítems de menús no se están mostrando si no estoy logueado, pero puedo entrar a las rutas sin problema, entonces la idea es proteger la ruta y validar que a algunas de esas rutas solo puedan ingresar los usuarios tipo admin

Comando para crear Middleware: *php artisan make:middleware Admin*

Se ha creado **Admin.php** en el directorio app/Http/Middleware

*public function handle($request, Closure $next) {*

*if ($request->user()->idperfil != 1) {*

*return redirect()->back()->with('mensaje','No tiene permisos');*

*}*

*return $next($request);*

*}*

En protected $routeMiddleware de app/Http **Kernel.php** ingresamos:

'admin' => \App\Http\Middleware\Admin::class,

En Facturas Controller ponemos:

*public function \_\_construct(){*

*$this->middleware('admin',['except'=>'index']);*

*}*

*Variación*

*$this->middleware(['auth','admin'],['except'=>'index']);*

*Variación*

*$this->middleware('auth');*

*$this->middleware('admin',['except'=>'index']);*

**ELOQUENT**

Para trabajar con bases de datos en Laravel no necesitamos escribir SQL de forma manual, ni siquiera utilizar el constructor de consultas. Laravel nos permite interactuar con la base de datos a un nivel mucho más alto a través del ORM Eloquent. Utilizando Eloquent podemos trabajar con modelos, que son clases que representan nuestras tablas en la base de datos y nos proveen de métodos con los que podemos interactuar en una interfaz más “elocuente” y orientada a objetos.

**Realizar consultas**

Podemos utilizar los modelos para hacer consultas a la base de datos. Utilizando el método all() obtenemos todo el contenido de la tabla:

$professions = Profession::all();

También podemos encadenar métodos del constructor de consultas para obtener resultados más específicos:

$professionId = Profession::where('title', 'desarrollador back-end')->value('id');

Podemos retonar un resultado dependiendo de su id mediante el método find():

$profession = Profession::find(1);

Unas de las instrucciones basicas de este son **get()** que nos regresa todos los registros de la BD y **first()** que nos regresa el primer registro de una seleccion.