* Comandos Artisan
  + /\*\*
  + \* Run the migrations.
  + \* php artisan make:migration create\_cursos\_table => Crea una nueva tabla.
  + \* php artisan migrate => Ejecuta las migraciones pendientes.
  + \* php artisan migrate:fresh => Elimina todas las tablas (SIN EJECUTAR método DOWN) y las vuelve a crear (Ejecuta Metodo UP). (Perdida de registros).
  + \* php artisan migrate:refresh =>  Elimina todas las tablas (EJECUTA método DOWN) y las vuelve a crear (Ejecuta Metodo UP).
  + \* php artisan make:migration add\_tipodoc\_to\_users\_table => Convención para agregar una columna sin borrar datos
  + \* php artisan make:migration cambiar\_propiedades\_to\_users\_table  => Convención para modificar propiedades de una columna.
  + \* php artisan migrate:reset: Elimina todas las tablas que se han creado actualmente en la base de datos, sin correr nuevamente las migraciones (Es decir sin ejecutar el método UP) y borra la tabla de migraciones.
  + php artisan db:seed => Ejecuta lo definido en el archivo /Database/seeders/DatabaseSeeder.php
  + php artisan migrate:fresh –seed => Combina Fresh con db:Seed
  + php artisan make:seeder CursoSeeder => Crea un archivo de Seeder independiente en el cual crear el código personalizado de Seeder para cada modelo
  + php artisan make:factory –model=Curso CursoFactory => Crea un archivo para crear Datos tipo Seeder a través de Factory. El achivo se crea en Database/Factories
  + php artisan make:request StoreCurso => Crea un FormRequest
* Base de Datos (Migraciones)
  + Schema::Create
    - * UP:

Schema::create('cursos', function (Blueprint $table) {

$table->id();

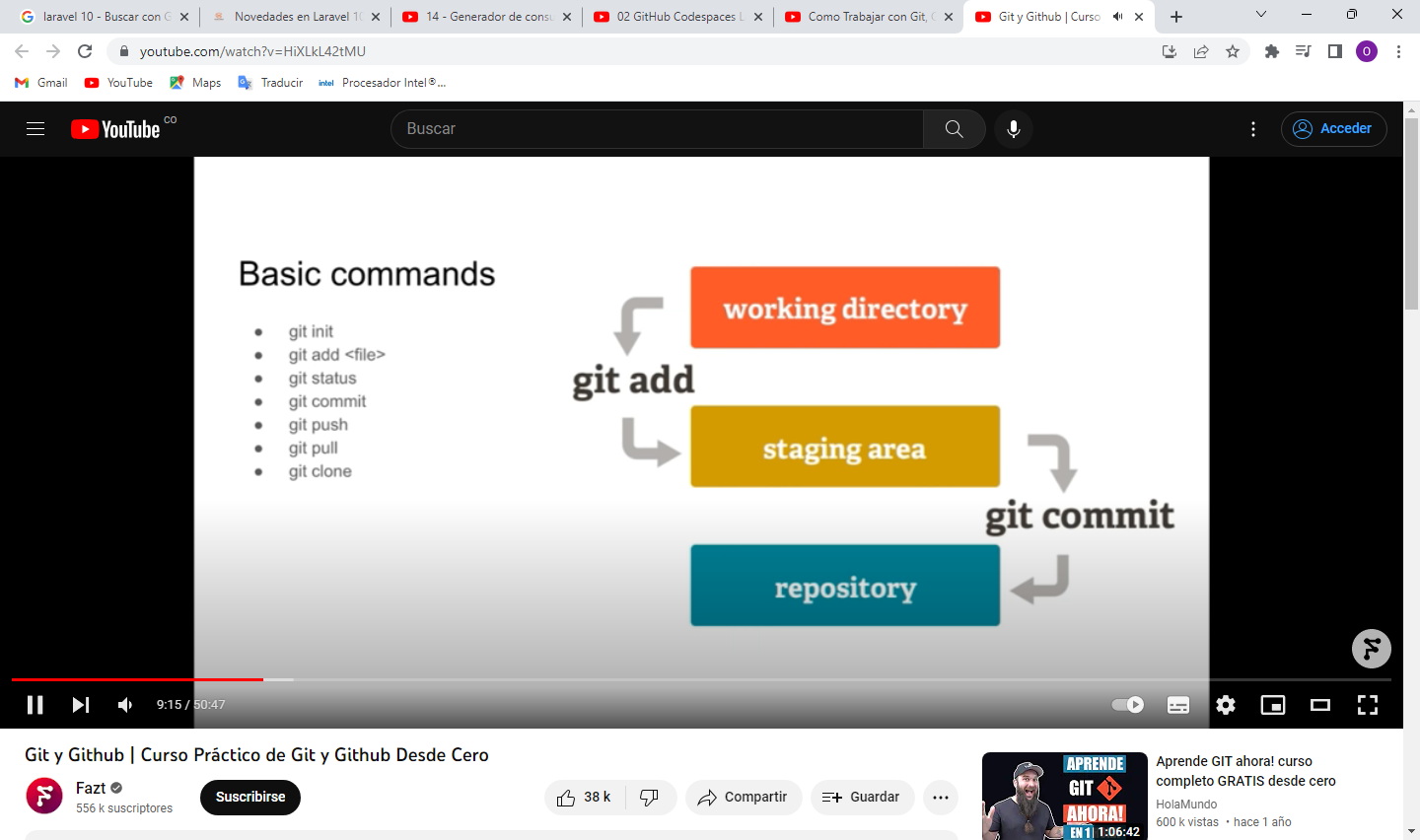
$table->string('name');

$table->text('descripcion');

$table->timestamps();

});

* + - * Down
        + Schema::dropIfExists('cursos');
  + Schema::Table
    - Add
      * UP: $table->unsignedTinyInteger('tipodoc')->nullable()->after('avatar');
      * Down: $table->dropColumn('tipodoc');
    - Modify
      * UP: $table->string('name', 150)->nullable()->change();
      * Down: $table->string('name', 255)->nullable(false)->change();
* Uso de Tinker
  + Permite ejecutar código php en linea de comando, para por ejemplo crear registros en bd
* Seeders
  + Factories para llenar datos de prueba.
    - En el archivo Seeder
      * User::factory(10)->create();
      * Curso::factory(50)->create();
    - En el factory se definen las reglas para la generación de los datos:
* return [
* //
* 'name' => $this->faker->sentence(),
* 'descripcion' => $this->faker->paragraph(),
* 'categoria' => $this->faker->randomElement(['Desarrollo Web', 'Diseño Web', 'Idiomas'])
* ];
* Eloquent
  + Todos los registros: $cursos = Curso::all()
  + Con Filtro y ordenación: $cursos = Curso::where('categoria','Desarrollo Web')->orderby('id','desc')->get()
  + Filtro diferente a igual: $cursos = Curso::where('id','>',45)->get()
  + Búsqueda con like: $curso = Curso::where('descripcion','like','% error %')->get()
  + El primer registro de la colección: Curso::where('categoria','Desarrollo Web')->orderby('name','desc')->first()
  + $cursos = Curso::select('name','descripcion')->get()
  + Select con campos con alias $cursos = Curso::select('id as numeroId','name as title')->where('categoria','Desarrollo Web')->orderby('name','desc')->get(Idiomas)
  + Búsqueda por ID (por la llave primaria de la tabla): $cursos = Curso::find(5)
  + Paginar los resultados: $cursos = Curso::Paginate()
* Mutadores y Accesores:
  + Mutador: Cambiar lo que trae de la vista antes de guardar.
* //Mutador
* protected function name(): Attribute
* {
* return new Attribute(
* set: function($value){
* return strtolower($value);
* }
* );
* }
  + Accesor: Cambiar lo que trae la base de datos de mostrar en la vista (app). No modificar valor en base de datos, solo transforma su representación (forma de mostrarlo)
* Agregar Lenguaje / Traducción al español de los mensajes:
  + composer require laravel-lang/common --dev
  + php artisan lang:add es
  + php artisan lang:update
* Crear Form Request: Sirven para hacer las validaciones de formulario
  + php artisan make:request StoreCurso
  + Métodos
    - public function authorize(): bool => Verifica permisos de usuario para usar el componente
    - public function rules(): array => Reglas de validación
    - public function attributes(): array => Permite personalizar el nombre con el que se quiere mostrar el campo.
    - public function messages() => Personalizar mensajes de validación
* GIT



* CONCEPTOS BASICOS DE GIT
  + https://www.youtube.com/watch?v=HiXLkL42tMU
  + <https://www.youtube.com/watch?v=jGehuhFhtnE>
* PUll request:
  + <https://platzi.com/clases/1557-git-github/19957-utilizando-pull-requests-en-github/>
* Git Fetch vs Git Pull:
  + <https://www.freecodecamp.org/espanol/news/git-fetch-vs-pull-cual-es-la-diferencia-entre-los-comandos-git-fetch-y-git-pull/>
* Curso de Git y Github: 7. Git como servidor y cliente
  + https://www.youtube.com/watch?v=iWDBh4zYNcU