

Universidad Nacional de San Agustín

Facultad de Producción y Servicio

Escuela Profesional de Ciencia de la Computación



CURSO ING. DE SOFTWARE

Codificación Legible (Clean Code)

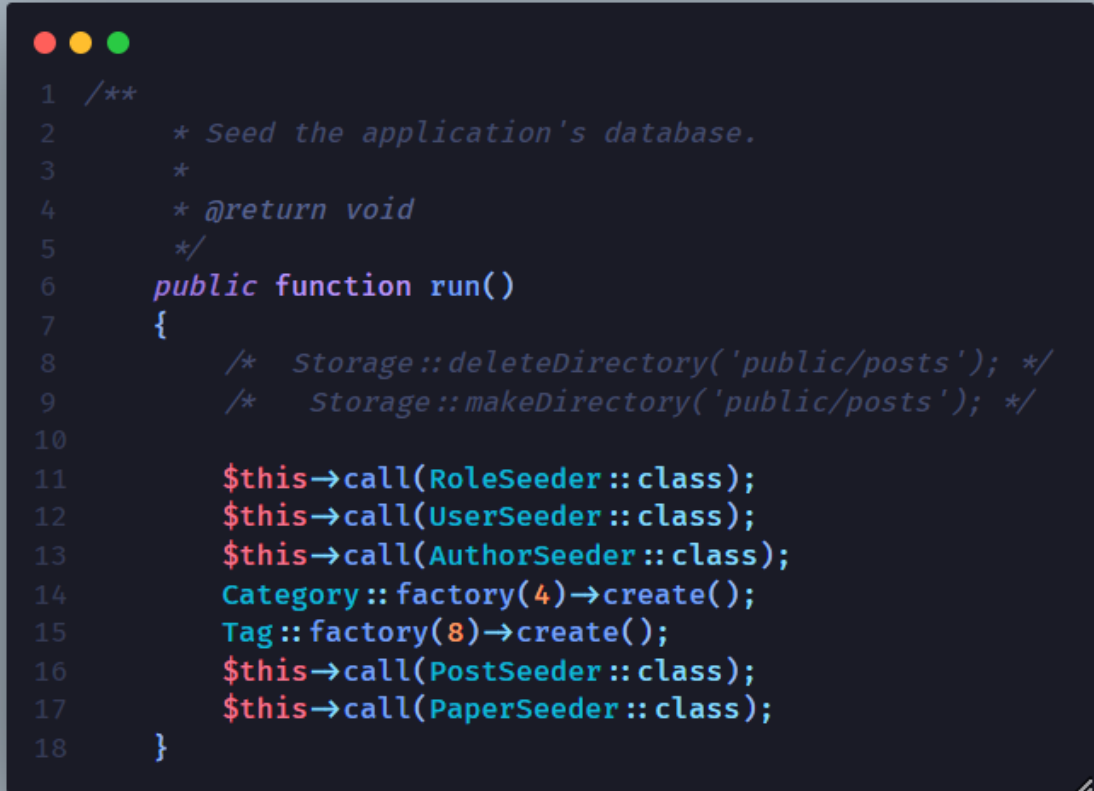
Alumno:

- Diego Alonso Zanabria Sacsi

Arequipa – Perú

2023

1. Keep all scopes (file/class/function) small and sorted. Haciendo uso de clases logramos mantener un código más pequeño y ordenado:

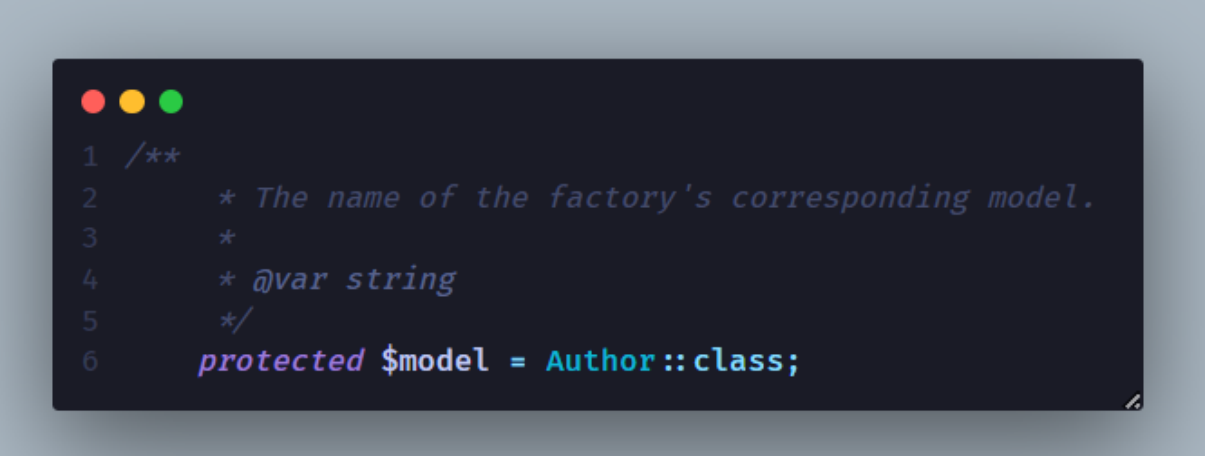


```
1  /**
2      * Seed the application's database.
3      *
4      * @return void
5      */
6  public function run()
7  {
8      /* Storage::deleteDirectory('public/posts'); */
9      /* Storage::makeDirectory('public/posts'); */
10
11      $this->call(RoleSeeder::class);
12      $this->call(UserSeeder::class);
13      $this->call(AuthorSeeder::class);
14      Category::factory(4)->create();
15      Tag::factory(8)->create();
16      $this->call(PostSeeder::class);
17      $this->call(PaperSeeder::class);
18  }
```

2. Sufijos de nombre de clase En el nombre de las clases tenemos un sufijo que en este caso indican polimorfismo

```
1
2 class AuthorFactory extends Factory
3 {
4
5     /**
6      * The name of the factory's corresponding model.
7      *
8      * @var string
9      */
10    protected $model = Author::class;
11
12    /**
13     * Define the model's default state.
14     *
15     * @return array
16     */
17    public function definition()
18    {
19        return [
20            'name' => $this->faker->name(),
21            'linkedin_url' => $this->faker->url(),
22            'email' => $this->faker->email()
23        ];
24    }
25 }
26
```

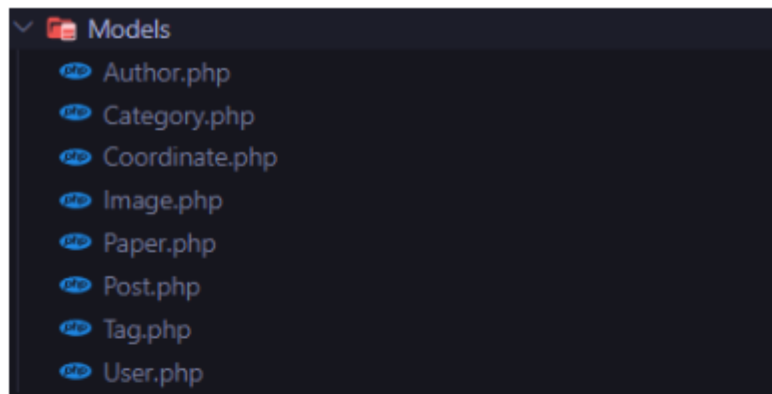
3. Docstrings: Documentación del código



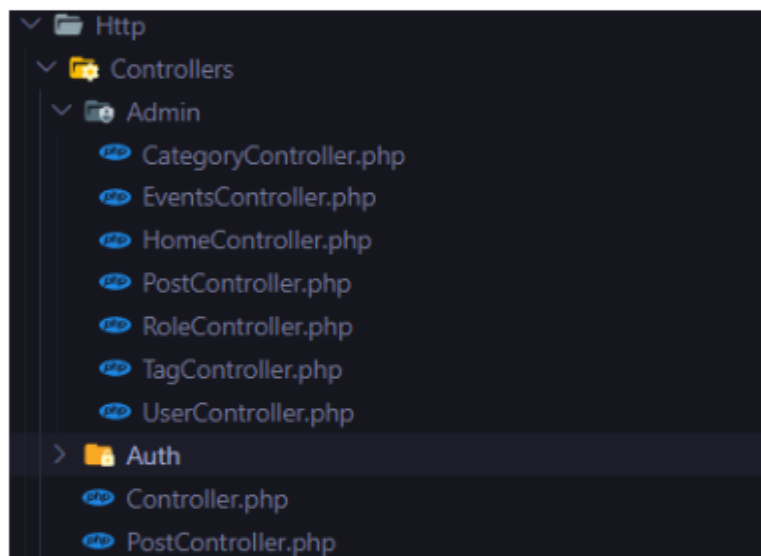
```
1 /**
2     * The name of the factory's corresponding model.
3     *
4     * @var string
5     */
6     protected $model = Author::class;
```

4. Separar los modelos de las vistas y el controlador(MVC) El framework usado para el proyecto aplica lo que es MVC dado que todo esta separado segun esta arquitectura.

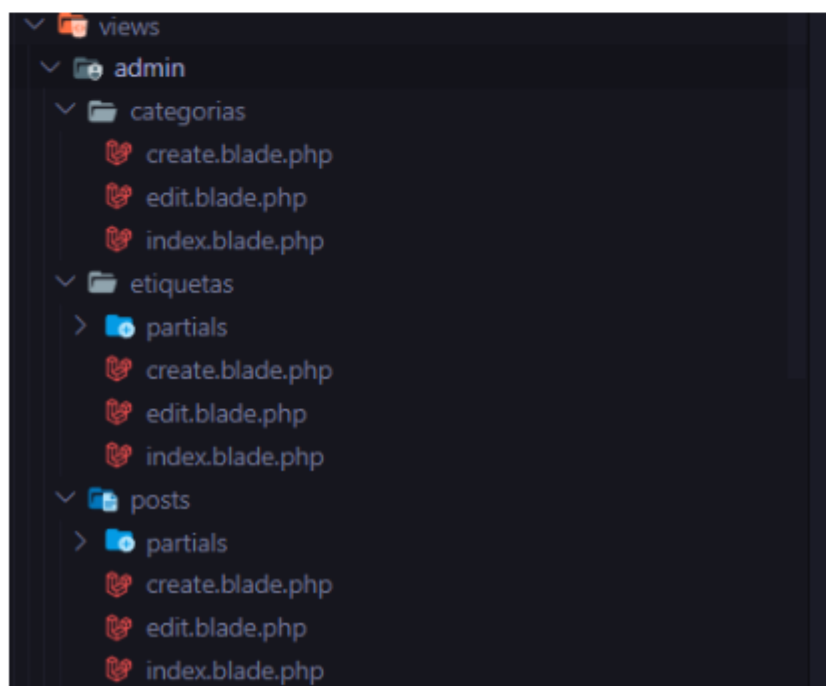
1. Modelos



2. Controladores



3. Vistas



5. Coherent Abstraction No mezclar diferentes niveles de abstracción en la misma función

```
1  <?php
2
3  namespace Database\Seeders;
4
5  use App\Models\Category;
6  use App\Models\Tag;
7  use Illuminate\Database\Seeder;
8  use Illuminate\Support\Facades\Storage;
9
10 class DatabaseSeeder extends Seeder
11 {
12     /**
13      * Seed the application's database.
14      *
15      * @return void
16      */
17     public function run()
18     {
19         Storage::deleteDirectory('public/posts');
20         Storage::makeDirectory('public/posts');
21
22         $this->call(RoleSeeder::class);
23         $this->call(UserSeeder::class);
24         Category::factory(4)->create();
25         Tag::factory(8)->create();
26         $this->call(PostSeeder::class);
27     }
28 }
```