



Facultad de Ingeniería de Sistemas y Electrónica
Carrera profesional de Ingeniería de Sistemas

Informe del proyecto final:

Detección de Bad Smells y Refactoring
Aplicación Web de Matrículas

Autor:

Canaza Tito, Eddy Wilmer

Asesor(a):

Ing. Paola Ana Zevallos Oporto

Arequipa, febrero de 2017

Índice general

1. Introducción	2
1.1. Planteamiento del Problema	2
1.2. Objetivos	2
1.2.1. General	2
1.2.2. Específicos	2
1.3. Justificación	2
2. Revisión de la literatura	3
3. Marco teórico	4
4. Metodología	5
4.1. Revisión de los Shared Layout de las vistas	5
4.1.1. Librerías de terceros innecesarias	5
4.1.2. Error en la carga de las vistas	6
4.1.3. Error en la carga de las vistas	6
4.1.4. Validación de los nested objects	7
4.1.5. Duplicidad en el mapeo	7
4.1.6. Ruta de los controlador API	9
Appendices	11
A. Requisitos funcionales	12

1. Introducción

1.1. Planteamiento del Problema

1.2. Objetivos

1.2.1. General

Incrementar el nivel de mantenibilidad de la Aplicación Web de Matrículas a través de Refactoring

1.2.2. Específicos

- Investigar técnicas de Refactoring
- Estudiar la Aplicación Web de Matrículas
- Aplicar técnicas de Refactoring
- Validar la solución

1.3. Justificación

2. Revisión de la literatura

3. Marco teórico

4. Metodología

Para esta etapa del proyecto se detectarán los bad smells del código existente del pro proyecto y se aplicarán técnicas de refactoring e ingeniería para resolverlos.

4.1. Revisión de los Shared Layout de las vistas

4.1.1. Librerías de terceros innecesarias

- **Síntoma:** Existen librerías de terceros procedentes de la plantilla que no se utilizan en ningún momento.
- **Solución:** Revisar que librerías se están usando dentro del proyecto.
- **Procedimiento:** Se eliminó la carpeta que contenía los elementos de la plantilla, se migraron los paquetes a bower y se reciclo las hojas de estilos. Además, se escribió comentarios indicando la función que cumple cada uno.

```
1  @* CSS *@
2  <!-- Fuentes -->
3  <!-- Plugins -->
4  <!-- Bootstrap -->
5  <link href="~/lib/bootstrap/dist/css/bootstrap.css" rel="stylesheet" />
6  <!-- Mensajes emergentes -->
7  <link href="~/lib/toastr/toastr.css" rel="stylesheet" />
8  <!-- Iconos -->
9  <!-- Iconos principales -->
10 <link href="~/lib/fontawesome/css/font-awesome.css" rel="stylesheet" />
11 <!-- Estilos de la aplicacion -->
12 <!-- Normalizacion de elementos -->
13 <link href="~/css/components.css" rel="stylesheet" />
14
15 @* JavaScript *@
16 <!-- jQuery -->
17 <script src="~/lib/jquery/dist/jquery.min.js"></script>
18 <!-- Bootstrap -->
19 <script src="~/lib/bootstrap/dist/js/bootstrap.min.js"></script>
20 <!-- Waypoints: Trigger para iniciar funcion -->
21 <script src="~/lib/waypoints/lib/jquery.waypoints.min.js"></script>
```

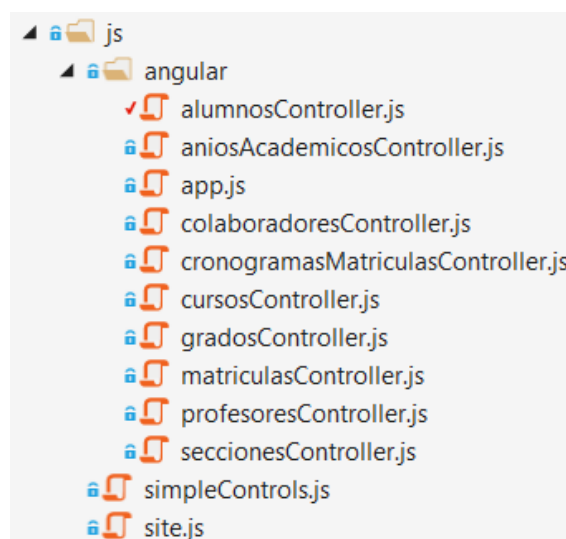
4.1.2. Error en la carga de las vistas

- **Síntoma:** Las vistas de la aplicación presentan un ligero retardo al cargar la data.
- **Solución:** Revisar las directivas de Angular.
- **Procedimiento:** Se agregó la directiva ng-cloak

```
1 <div class="row" ng-app="app" ng-controller="alumnosController as vm">
2   <wait-cursor display-when="vm.isBusy"></wait-cursor>
3   <!-- Tabla de visualizacion -->
4   <div ... ng-cloak>
5     ...
6   </div>
7 </div>
```

4.1.3. Error en la carga de las vistas

- **Síntoma:** Los archivos javascript de la aplicación están mezclados con los controladores angular.
- **Solución:** Colocar todos los controladores dentro de una carpeta.
- **Procedimiento:** Se creó la carpeta “angular” para trasladar los controladores y se cambió la ruta de los paquetes en las vistas.



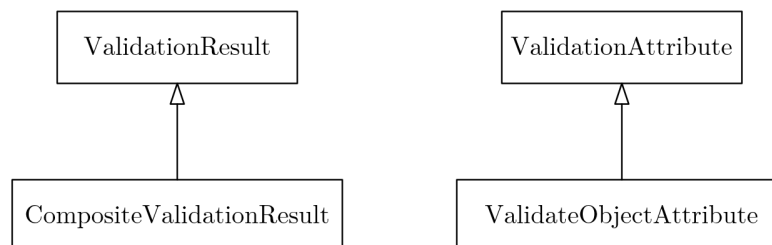
```

1 @section scripts {
2     ...
3     <script src="~/js/angular/app.js"></script>
4     <script src="~/js/angular/alumnosController.js"></script>
5     ...
6 }

```

4.1.4. Validación de los nested objects

- **Síntoma:** El controlador no valida los nested objects y se valida los objetos manualmente.
- **Bad Smell:** Duplicated code.
- **Solución:** Crear una clase que realice este proceso iterativamente.
- **Procedimiento:** Se creó dos clases, una llamada “CompositeValidationResult” la otra “ValidateObjectAttribute” que agregan un atributo a los decoradores de validación del modelo para que puedan usarse en todos los DTO.



```

1 public class AlumnoViewModel
2 {
3     ...
4     [Required, ValidateObject]
5     public virtual ApoderadoViewModel Apoderado { get; set; }
6     ...
7 }

```

4.1.5. Duplicidad en el mapeo

- **Síntoma:** Se está mapeando el objeto dos veces, una para validar algún atributo y otro para transaccionar.

- **Bad Smell:** Duplicated code.

```
1 public class AlumnoViewModel
2 [HttpPost()]
3 public async Task<IActionResult> PostAlumno([FromBody] AlumnoViewModel
    alumnoDetails)
4 {
5     ...
6     if (!_repository.IsDniValido(Mapper.Map<Alumno>(alumnoDetails)))
7         ModelState.AddModelError("otros", "Este DNI ya fue registrado.");
8
9     if (ModelState.IsValid)
10    {
11        var _newAlumno = Mapper.Map<Alumno>(alumnoDetails);
12        ...
13    }
14    ...
15 }
```

- **Solución:** Extraer las sentencias duplicadas y colocarlo en una variable.

- **Técnica:** Extract variable.

```
1 [HttpPost()]
2 public async Task<IActionResult> PostAlumno([FromBody] AlumnoViewModel
    alumnoDetails)
3 {
4     ...
5     var alumno = Mapper.Map<Alumno>(alumnoDetails);
6
7     if (!_repository.IsDniValido(alumno))
8         ModelState.AddModelError("otros", "Este DNI ya fue registrado.");
9
10    if (ModelState.IsValid)
11    {
12        _repository.AddAlumno(alumno);
13        ...
14    }
15    ...
16 }
```

4.1.6. Ruta de los controlador API

- **Síntoma:** Se está repitiendo en cada acción la ruta del controlador.
- **Bad Smell:** Duplicated code.

```
1 public class AlumnosController : Controller
2 {
3     ...
4
5     [HttpGet("api/alumnos")]
6     public IActionResult GetAllAlumnos()
7     { ... }
8
9     [HttpGet("api/alumnos/{id}")]
10    public IActionResult GetAlumno(int id)
11    { ... }
12
13    ...
14 }
```

- **Solución:** Utilizar el decorador Route para indicar la ruta del controlador. También indicar la versión del API y cambiar el nombre de las acciones según convenciones.

```
1 [Route("api/v2/[controller]")]
2 public class AlumnosController : Controller
3 {
4     ...
5
6     [HttpGet()]
7     public IActionResult GetAllAlumnos()
8     { ... }
9
10    [HttpGet("{id}")]
11    public IActionResult GetAlumno(int id)
12    { ... }
13
14    ...
15 }
```

Referencias bibliográficas

Anexos

A. Requisitos funcionales