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

4.1.7. Sobrecarga de Entity Framework 10

4.1.8. Actualización de campos 10

4.1.9. Método SearchCursos en controlador de cursos 11

4.1.10. Lógica de negocio en el controlador 13

4.1.11. Uso inadecuado de procedimientos almacenados 14

4.1.12. Lógica de negocio en el controlador 15

4.1.13. Atributos que no se usan 16

4.1.14. Cambiar la definición de la clase CronogramaMatricula 16

Appendices **19**

A. Requisitos funcionales **20**

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.

```
@* CSS *@
<!-- Fuentes -->
<!-- Plugins -->
<!-- Bootstrap -->
<link href="~/lib/bootstrap/dist/css/bootstrap.css" rel="stylesheet" />
 <!-- Mensajes emergentes -->
<link href="~/lib/toastr/toastr.css" rel="stylesheet" />
<!-- Iconos -->
<!-- Iconos principales -->
<link href="~/lib/fontawesome/css/font-awesome.css" rel="stylesheet" />
<!-- Estilos de la aplicacion -->
<!-- Normalizacion de elementos -->
<link href="~/css/components.css" rel="stylesheet" />

@* JavaScript *@
<!-- jQuery -->
<script src="~/lib/jquery/dist/jquery.min.js"></script>
<!-- Bootstrap -->
<script src="~/lib/bootstrap/dist/js/bootstrap.min.js"></script>
<!-- Waypoints: Trigger para iniciar funcion -->
<script src="~/lib/waypoints/lib/jquery.waypoints.min.js"></script>
```

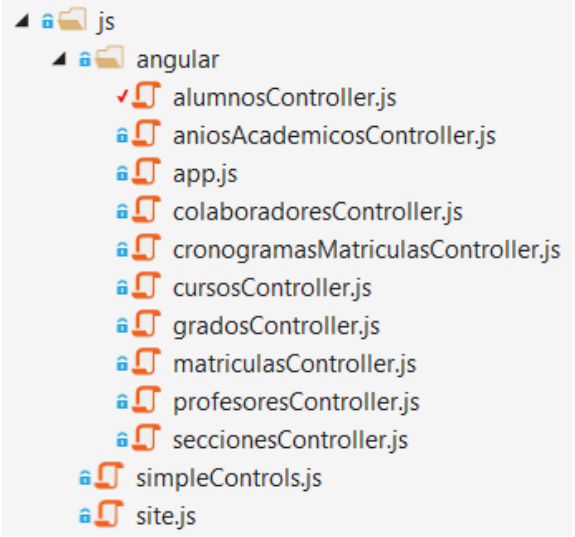
4.1.2. Error en la carga de las vistas

- **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

```
<div class="row" ng-app="app" ng-controller="alumnosController as vm">
  <wait-cursor display-when="vm.isBusy"></wait-cursor>
  <!-- Tabla de visualizacion -->
  <div ... ng-cloak>
    ...
  </div>
</div>
```

4.1.3. Error en la carga de las vistas

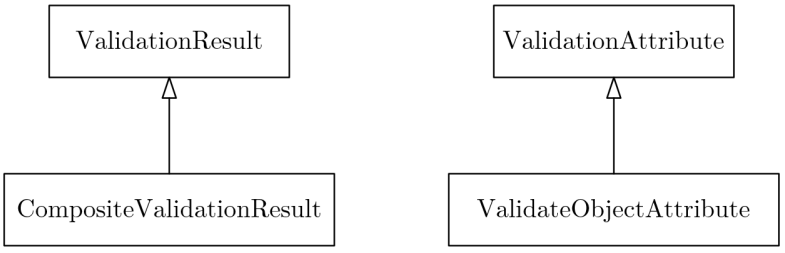
- **Síntoma:** Los archivos javascript de la aplicación están mezclados con los controladores angular.
- **Solución:** Colocar todos los controladores dentro de una carpeta.
- **Procedimiento:** Se creó la carpeta "angular"para trasladar los controladores y se cambió la ruta de los paquetes en las vistas.



```
@section scripts {
  ...
  <script src="~/js/angular/app.js"></script>
  <script src="~/js/angular/alumnosController.js"></script>
  ...
}
```

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 "CompositeValidationResultz la otra "ValidateObjectAttribute"que agregan un atributo a los decoradores de validación del modelo para que puedan usarse en todos los DTO.




```
public class AlumnoViewModel
{
  ...
  [Required, ValidateObject]
  public virtual ApoderadoViewModel Apoderado { get; set; }
  ...
}
```

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.

```
public class AlumnoViewModel
[HttpPost()]
public async Task<IActionResult> PostAlumno([FromBody] AlumnoViewModel
    alumnoDetails)
{
  ...
  if (!_repository.IsDniValido(Mapper.Map<Alumno>(alumnoDetails)))
    ModelState.AddModelError("otros", "Este DNI ya fue registrado.");

  if (ModelState.IsValid)
  {
    var _newAlumno = Mapper.Map<Alumno>(alumnoDetails);
    ...
  }
  ...
}
```

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

```
[HttpPost()]
public async Task<IActionResult> PostAlumno([FromBody] AlumnoViewModel
    alumnoDetails)
{
  ...
  var alumno = Mapper.Map<Alumno>(alumnoDetails);

  if (!_repository.IsDniValido(alumno))
    ModelState.AddModelError("otros", "Este DNI ya fue registrado.");

  if (ModelState.IsValid)
  {
    _repository.AddAlumno(alumno);
    ...
  }
  ...
}
```

4.1.6. Ruta de los controlador API

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

```
public class AlumnosController : Controller
{
  ...

  [HttpGet("api/alumnos"))]
  public IActionResult GetAllAlumnos()
  { ... }

  [HttpGet("api/alumnos/{id}")]
  public IActionResult GetAlumno(int id)
  { ... }

  ...
}
```

- **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.

```
[Route("api/v2/[controller]")]
public class AlumnosController : Controller
{
  ...

  [HttpGet()]
  public IActionResult GetAllAlumnos()
  { ... }

  [HttpGet("{id}")]
  public IActionResult GetAlumno(int id)
  { ... }

  ...
}
```

4.1.7. Sobrecarga de Entity Framework

- **Síntoma:** Se está accediendo muchas veces a un atributo del objeto

```
public void ToggleEstado(int id)
{
  var colaborador = Get(id);
  _context.Colaboradores.Attach(colaborador).State = EntityState.
    Modified;

  if (colaborador.Estado == "1")
    colaborador.Estado = "3";
  else if (colaborador.Estado == "3")
    colaborador.Estado = "1";

  _context.Update(colaborador);
}
```

- **Técnica:** Extract variable.

- **Solución:** Extraer la operación en una variable temporal.

```
public void ToggleEstado(int id)
{
  var colaborador = Get(id);
  _context.Colaboradores.Attach(colaborador).State = EntityState.
    Modified;

  var estado = colaborador.Estado;

  colaborador.Estado = estado == "1" ?  "3" : estado == "3" ? "1" :
    estado;
}
```

4.1.8. Actualización de campos

- **Síntoma:** Se está pasando cada uno de los atributos para la actualización del objeto.

```
public void UpdateAlumno(Alumno entity)
{
  var thisAlumno = GetAlumnoById(entity.Id);
  thisAlumno.ApellidoPaterno = alumnoToUpdate.ApellidoPaterno;
  thisAlumno.ApellidoMaterno = alumnoToUpdate.ApellidoMaterno;
  thisAlumno.Nombres = alumnoToUpdate.Nombres;
  thisAlumno.Dni = alumnoToUpdate.Dni;
  thisAlumno.Sexo = alumnoToUpdate.Sexo;
  thisAlumno.Direccion = alumnoToUpdate.Direccion;
  thisAlumno.FechaNacimiento = alumnoToUpdate.FechaNacimiento;

  _context.Update(thisAlumno);
}
```

- **Solución:** Utilizar Entries de Entity Framework para determinar los comportamientos de las entidades y no alterar los Nested Objects.

```
public Alumno UpdateAlumno(Alumno entity)
{
  _context.Entry(entity).State = EntityState.Modified;
}
```

4.1.9. Método SearchCursos en controlador de cursos

- **Síntoma:** El método SearchCursos se encarga de listar los cursos que dicta un profesor está en un controlador que no le corresponde.
- **Bad Smell:** Intimidad inadecuada.

```
public class CursosController : Controller
{
  ...
  [HttpGet("{id}")]
  public IActionResult GetCursosDisponibles(int id)
  {
    try
    {
      _logger.LogInformation("Recuperando la lista de cursos que dicta
   el profesor.");
```

```
10        var results = _repository.GetCursosByIdProfesor(id);
11        return Ok(Mapper.Map<IEnumerable<CursoViewModel>>(results));
12      }
13      catch (Exception ex)
14      {
15        _logger.LogError($"No se pudo recuperar los cursos: {ex}");
16        return BadRequest("No se pudo recuperar la información.");
17      }
18    }
19    ...
20  }
```

- **Técnica:** Move Method.
- **Solución:** Trasladar este método al controlador de profesores para implementar el API Rest correctamente.

```
1  public class ProfesoresController : Controller
2  {
3    ...
4    [HttpGet("{id}")]
5    public IActionResult GetCursosDisponibles(int id)
6    {
7      try
8      {
9        _logger.LogInformation("Recuperando la lista de cursos que dicta
       el profesor.");
10       var results = _repository.GetCursosByIdProfesor(id);
11       return Ok(Mapper.Map<IEnumerable<CursoViewModel>>(results));
12     }
13     catch (Exception ex)
14     {
15       _logger.LogError($"No se pudo recuperar los cursos: {ex}");
16       return BadRequest("No se pudo recuperar la información.");
17     }
18   }
19   ...
20 }
```

4.1.10. Lógica de negocio en el controlador

- **Síntoma:** El controlador tiene código de la lógica del negocio; las tareas que deberían ejecutarse en el repositorio.
- **Bad Smell:** Intimidad inadecuada.

```
public async Task<IActionResult> PostUpdateProfesorCurso([FromBody]
    CursoProfesorViewModel profesorDetails)
{
  ...
  if (ModelState.IsValid)
  {
    var profesor = Mapper.Map<Profesor>(profesorDetails.Profesor);
    var curso = Mapper.Map<Curso>(profesorDetails.Curso);
    ...
  }
  ...
}
```

- **Técnica:** Move Method y Extract Method.
- **Solución:** Trasladar el código al repositorio creando un método que realice las tareas de mapeo.

```
public void AssignProfesor(int id, int idProfesor)
{
  var anioAcademico = new AniosAcademicosRepository(_context).
    GetAnioAcademico(DateTime.Now.Year);

  if (anioAcademico != null)
  {
    var cursoAnioAcademico = _context.CursosAnioAcademico
      .Where(t => t.AnioAcademicoId == anioAcademico.Id)
      .Where(t => t.CursoId == id)
      .FirstOrDefault();
    ...
  }
}
```

4.1.11. Uso inadecuado de procedimientos almacenados

- **Síntoma:** El método llama a un procedimiento en la base de datos.

```
public void Update(Profesor entity)
{
  ...
  _context.Database.ExecuteSqlCommand(String.Format("EXEC
   SP_DeleteCursos @idProfesor='{0}'", profesor.Id));
  foreach (Curso curso in cursos)
  {
      _context.Database.ExecuteSqlCommand(
        String.Format("EXEC SP_AddProfesorCurso @idProfesor='{0}',
   @idCurso='{1}'",
          profesor.Id, curso.Id));
  }
  ...
}
```

- **Técnica:** Extract Method.

- **Solución:** Remplazar el procedimiento almacenado con sentencias LINQ, y extraer el segmento de código que elimina los cursos actuales del profesor en un nuevo método.

```
public void EliminarCursos(int id)
{
  var cursos = _context.ProfesorCursos
    .Where(t => t.ProfesorId == id)
    .ToList();

  _context.ProfesorCursos.RemoveRange(cursos);
  _context.SaveChanges();
}
```

4.1.12. Lógica de negocio en el controlador

- **Síntoma:** El controlador tiene código de la lógica del negocio; las tareas que deberían ejecutarse en el repositorio.
- **Bad Smell:** Intimidad inadecuada.

```
public async Task<IActionResult> PostUpdateProfesorCurso([FromBody]
    CursoProfesorViewModel profesorDetails)
{
  ...
  if (ModelState.IsValid)
  {
    var profesor = Mapper.Map<Profesor>(profesorDetails.Profesor);
    var curso = Mapper.Map<Curso>(profesorDetails.Curso);
    ...
  }
  ...
}
```

- **Técnica:** Move Method y Extract Method.
- **Solución:** Trasladar el código al repositorio creando un método que realice las tareas de mapeo.

```
public void AssignProfesor(int id, int idProfesor)
{
  var anioAcademico = new AniosAcademicosRepository(_context).
    GetAnioAcademico(DateTime.Now.Year);

  if (anioAcademico != null)
  {
    var cursoAnioAcademico = _context.CursosAnioAcademico
    .Where(t => t.AnioAcademicoId == anioAcademico.Id)
    .Where(t => t.CursoId == id)
    .FirstOrDefault();
    ...
  }
}
```

4.1.13. Atributos que no se usan

- **Síntoma:** La clase Cursos tiene atributos que no usa.

```
public class Curso
{
  ...
  public virtual IEnumerable<Profesor> Profesores { get; set; }
  public virtual ICollection<ProfesorCurso> ProfesorCurso { get; set; }
  ...
}
```

- **Solución:** Eliminar los atributos que no usa en el modelo.

4.1.14. Cambiar la definición de la clase CronogramaMatricula

- **Síntoma:** La clase CronogramaMatricula que era el modelo de los cronogramas de matrículas; ahora, se utilizará para cualquier tipo de cronograma dentro de un año académico. Además se usando un atributo "Nombreçomo Primary Key.

```
public class CronogramaMatricula
{
  [Key]
  [ForeignKey("AnioAcademico")]
  public int AnioAcademicoId { get; set; }

  [Key]
  [StringLength(30)]
  public string Nombre { get; set; }

  ...
}
```

- **Solución:** Renombrar la clase CronogramaMatricula a Cronograma. Agregar un Atributo Id para establecerlo como Primary Key.

```
public class Cronograma
{
  [Key]
  public int Id { get; set; }

  [ForeignKey("AnioAcademico")]
  public int AnioAcademicoId { get; set; }

  [StringLength(30)]
  public string Nombre { get; set; }

  ...
}
```

Referencias bibliográficas

Anexos

A. Requisitos funcionales