| | |
|---|---|
| **Ingeniería en Sistemas de Computación**<br>Examen de ampliación | **Curso:** Programación Avanzada Web |
| | **Prof:** Diego Rojas Valverde |
| | **Código curso:** SC-701 |

# 1 Índice

## Contents

# 2 Instrucciones generales

El presente documento define las pautas para la elaboración **Examen de ampliación** del curso **Programación Avanzada Web (SC-701)**

Esta evaluación está incluida dentro de las evaluaciones de la **Directriz sobre Honestidad Académica**, presentada y aceptada en el **Programa del Curso**, el incumplimiento con la directriz mencionada generará la aplicación correspondiente del artículo 31 del reglamento estudiantil vigente.

La entrega del proyecto se debe realizar de la siguiente manera:

1. Todo el código del proyecto debe estar en GitHub en la asignación correspondiente en el classroom.
2. Es indispensable que el código compile sin errores para poder evaluarlo.

# 3 Examen

Se requiere construir un sistema web que posea un API Rest que a su vez sea consumido por una aplicación Web realizada con Razor Pages, el sistema debe incluir todas las operaciones CRUD del modelo especificado más adelante.

## 3.1 Componentes tecnológicos requeridos

La solución debe utilizar los siguientes componentes tecnológicos:

- .NET 8 o superior
- Lenguaje de Programación C#
- Dapper la gestión del modelo de datos
- Web API basada en Controladores
- Swagger para la documentación y uso de la aplicación
- Razor Pages (.NET 8 o superior)

## 3.2 Arquitectura de la aplicación

La aplicación de debe cumplir con la siguiente arquitectura de componentes:

- Application Programming Interfaz (API).
- Flujo.
- Data Access (DA).
- Base de datos.
- Abstracciones.
- Interface web

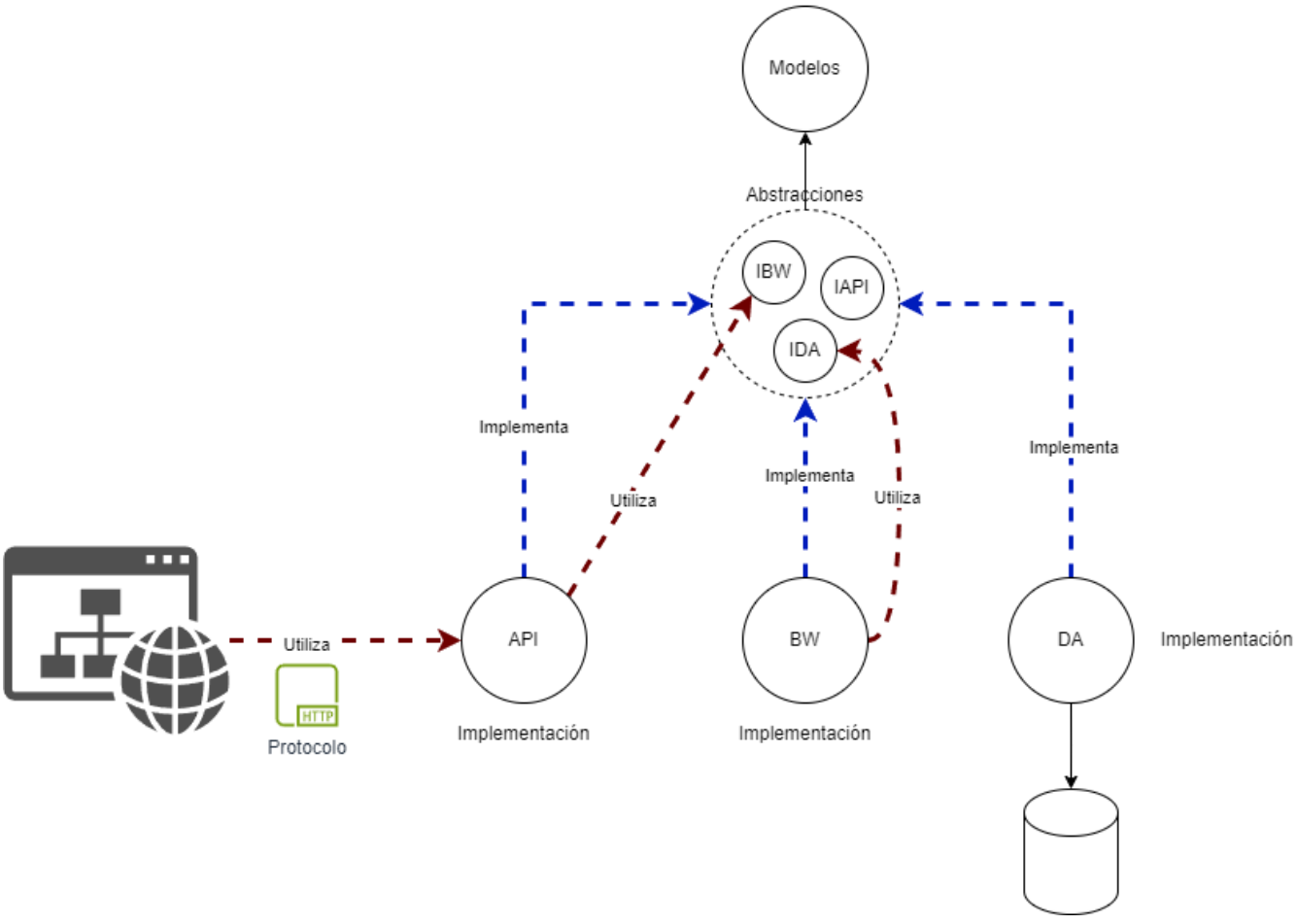



*Ilustración 1: Arquitectura de componentes*

## 3.3 Prácticas de desarrollo

Se debe realizar la aplicación siguiendo prácticas de desarrollo de programación orientada a objetos que permiten que el código sea fácil de leer, mantener y probar (Se puede tomar de referencia los principios SOLID), en todo momento se debe hacer uso de interfaz e inyección de dependencias.

## 3.4 Conocimientos por aplicar

| Semana | Temas |
|---|---|
| 2 | Response |
| 2 | Status |
| 2 | Headers |
| 2 | Request |
| 2 | Postman |
| 2 | Get |
| 2 | Post |
| 3 | parámetros |
| 3 | Restricciones de enrutamiento |
| 3 | Web root y archivos estáticos |
| 4 | creación de controles |
| 4 | “Action Methods” |
| 4 | ContentResult |
| 4 | JsonResult |
| 4 | IActionResult |
| 4 | Redireccionamiento |
| 5 | Introducción |
| 5 | Model Class |
| 5 | Model Validations |
| 5 | Validaciones Personalizadas |
| 5 | Bind vs BindNever |
| 6 | Estudio Práctico |
| 7 | Layout Views, para uso singular o múltiple. |
| 7 | ViewData |
| 7 | Vistas Dinámicas |
| 7 | Vistas Anidadas |
| 7 | Vistas Parciales |
| 7 | Vistas Parciales con ViewData |
| 8 | FromService |
| 8 | Transient, Scoped, Singleton |
| 8 | View Injection |
| 8 | Excepciones y excepciones personalizdas |
| 9 | Configuración de inicio |
| 9 | “Tag Helper” |
| 9 | Iconfiguration |
| 9 | Configuraciones como servicio. |
| 9 | Configuración con JSON |
| 9 | Controladores asincronos |
| 10 | Interfaz de usuario. |
| 10 | Vista de listas. (buscar, ordenar) |
| 10 | Vistas de creación |
| 11 | Db Context |
| 11 | Db Set |
| 11 | Cadenas de Conexión |
| 11 | Migraciones |
| 11 | CRUD |
| 11 | Procedimientos Almacenados |

*Tabla 1: Conocimientos por aplicar*

# 4 Parte 1: API RESTful

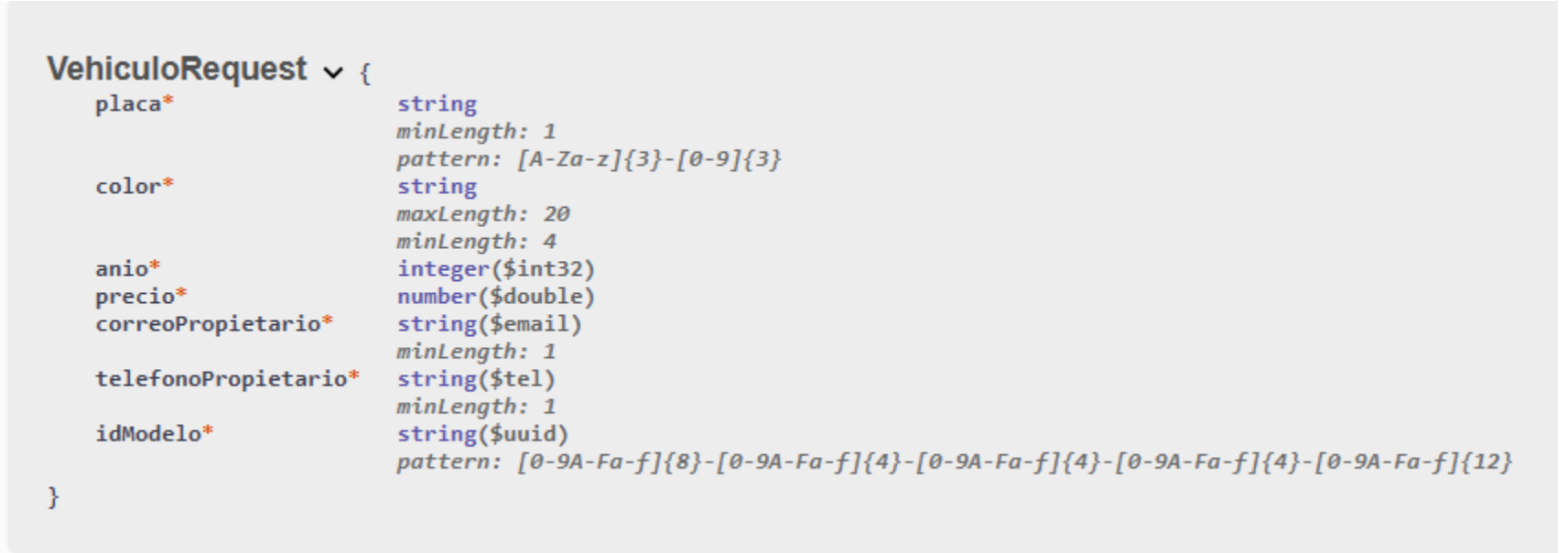
Se debe implementar un RestAPI que implemente el siguiente CRUD:

- Obtener todos los registros
- Obtener un registro por Id
- Agregar un registro
- Modificar un registro
- Eliminar un registro

El modelo para utilizar para implementar el CRUD es el siguiente:

| Propiedad | Tipo de datos | Formato | Observaciones | Expresión regular |
|---|---|---|---|---|
| Id | GUID | XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX | | [0-9A-Fa-f]{8}-[0-9A-Fa-f]{4}-[0-9A-Fa-f]{4}-[0-9A-Fa-f]{4}-[0-9A-Fa-f]{12} |
| Marca | GUID-Texto | | El tipo de datos debe ser texto para mostrar el nombre de la marca en los CRUD de consulta, pero GUID en los otros CRUD. | |
| Modelo | GUID-Texto | | | |
| Placa | Texto | XXX-### | | [A-Za-z]{3}-[0-9]{3} |
| Color | Texto | | | |
| Año | Número entero | | | |
| Precio | Número decimal | | | |
| Correo del propietario | Texto | | | |
| Teléfono del propietario | Texto | | | |

*Tabla 2: Modelo de datos*



**VehiculoRequest** {

| | |
|---|---|
| placa* | string<br>*minLength: 1*<br>*pattern: [A-Za-z]{3}-[0-9]{3}* |
| color* | string<br>*maxLength: 20*<br>*minLength: 4* |
| anio* | integer($int32) |
| precio* | number($double) |
| correoPropietario* | string($email)<br>*minLength: 1* |
| telefonoPropietario* | string($tel)<br>*minLength: 1* |
| idModelo* | string($uuid)<br>*pattern: [0-9A-Fa-f]{8}-[0-9A-Fa-f]{4}-[0-9A-Fa-f]{4}-[0-9A-Fa-f]{4}-[0-9A-Fa-f]{12}* |

}

*Ilustración 2: Modelo con validaciones*

### 4.1.1 Base de datos

La base de datos proporcionada posee las siguientes entidades:

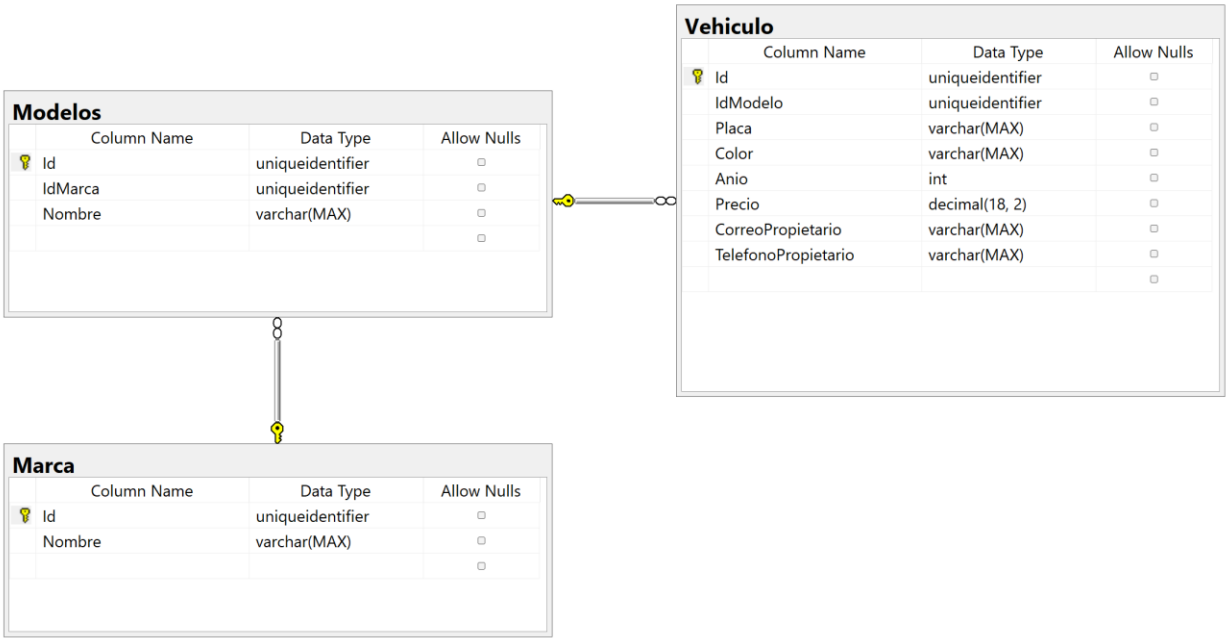



*Ilustración 3: Diagrama de base de datos*

- Se deben generar los procedimientos almacenados para cada CRUD uso de Procedimientos almacenados.
- El Procedimiento almacenado InicializarBD llena las tablas de Marca y modelo, se debe ejecutar antes.

### 4.1.2 RestAPI

Implementada mediante el uso de arquitectura limpia, debe poseer al menos los proyectos los siguientes proyectos:

| Proyecto | Tipo | Descripción | Observaciones |
|---|---|---|---|
| Abstracciones | Biblioteca de clases | Encargado de definir los modelos e interfaces. | Requerido |
| DA | Biblioteca de clases | Encargada del acceso a la base de datos. | Requerido |
| Flujo | Biblioteca de clases | Encargado de dirigir el flujo de información entre las capas. | Requerido |
| API | ASP.NET Core Web API | Encargado de gestionar las solicitudes y respuestas HTTP. | Requerido |

*Tabla 3: Proyectos del API*

Toda clase no estática debe poseer una interfaz.

Para la implementación de los EndPoints del API RestFul se debe considerar lo siguiente:

- Uso correcto parámetros, indicando el atributo correcto para la obtención de la información (FromHeader, FromBody, FromRoute, etc.)
- Uso correo del método a utilizar (GET, POST, PUT, etc.)
- Todas las validaciones se deben realizar mediante el uso de anotaciones en los modelos. Deben de realizar para todas las propiedades las siguientes validaciones:
    - Propiedad requerida
    - Longitud mínima y máxima
    - Expresión regular para las propiedades que posean una (tomar de referencia la Tabla 2)
    - Validación de formato de correo y teléfono.
- Establecer Routes en cada EdPoint que faciliten el uso del API.
- Todas las respuestas HTTP que lo requieran deben de tener la validación de si el recurso existe, si no se encontró deberán devolver la respuesta http correspondiente.
- Las repuestas de los modelos deben ser por medio de actions methods y el contenido debe estar en formato JSON.
- Uso correcto de códigos HTTP en las respuestas de los métodos (200,201, 204, 404, 500).
- Uso de métodos asíncronos.

No realizar instancias de implementaciones concretas de objetos con excepción de las clases estáticas, en su lugar hacer referencia a las abstracciones en forma de Interfaces y por medio de la inyección de dependencias.

Todos los parámetros de configuración como cadenas de conexión, rutas o urls deben ser gestionados utilizando el appsettings.

A continuación, se detallan las solicitudes y respuestas HTTP esperados por cada CRUD:

#### *4.1.2.1 Agregar*

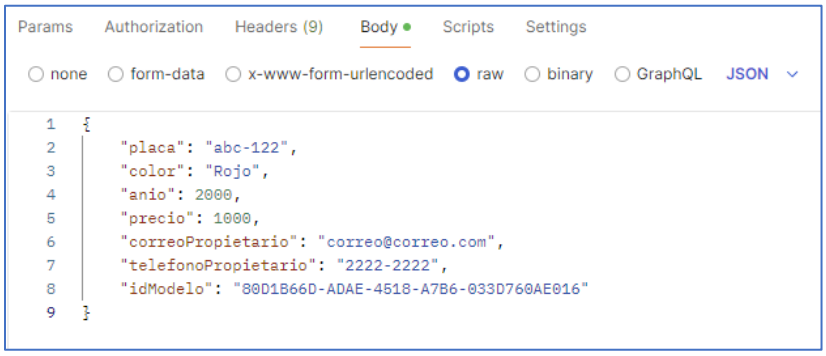



*Ilustración 4: Request*

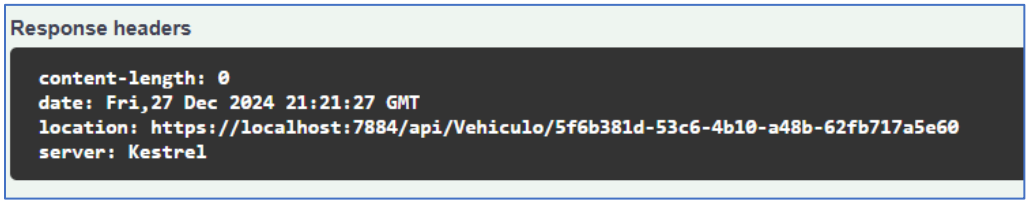



*Ilustración 5:Response*

#### *4.1.2.2 Editar*

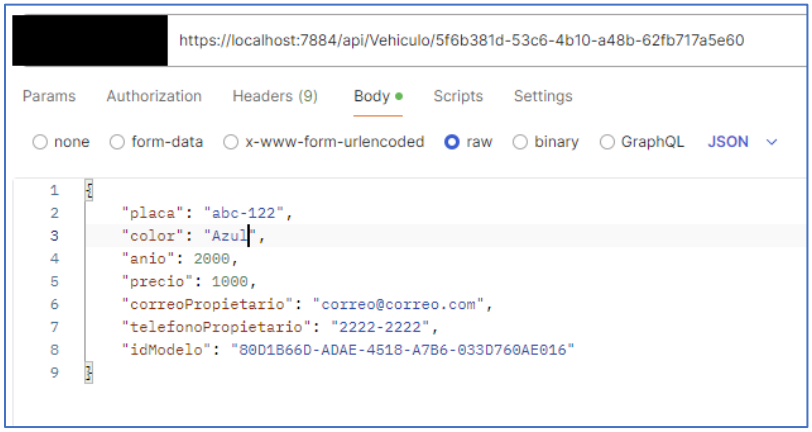



*Ilustración 6: Request*

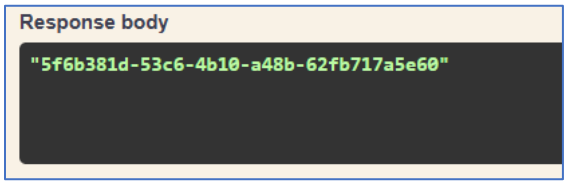



*Ilustración 7:Response*

### 4.1.2.3 Eliminar

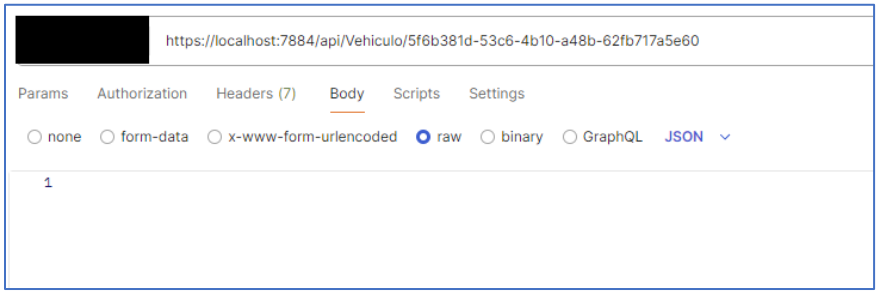



*Ilustración 8: Request*

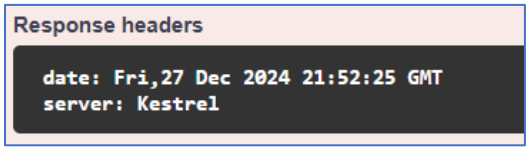



*Ilustración 9:Response*

### 4.1.2.4 Obtener Todos

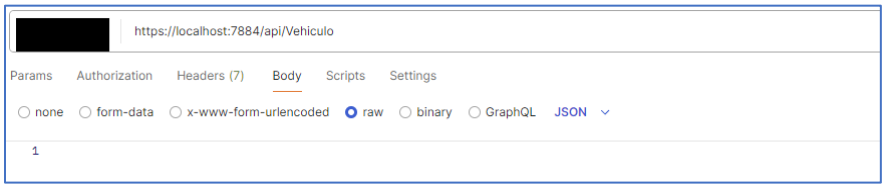



*Ilustración 10: Request*


*Ilustración 11:Response*

### 4.1.2.5 Obtener un registro

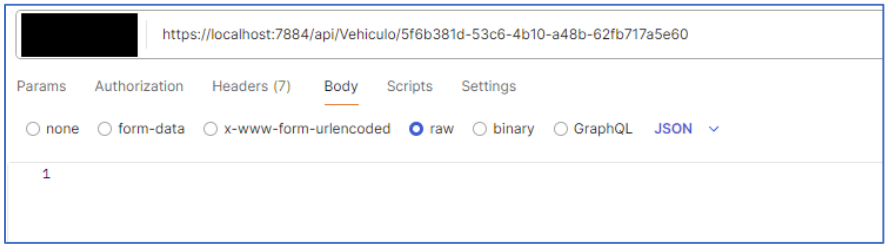



*Ilustración 12: Request*


*Ilustración 13:Response*

# 5 Parte 2: Interfaz Web

Implementada mediante el uso de arquitectura limpia, debe poseer al menos los proyectos los siguientes proyectos:

| Proyecto | Tipo | Descripción | Observaciones |
|---|---|---|---|
| Abstracciones | Biblioteca de clases | Encargado de definir los modelos e interfaces. | Requerido |
| WebApp | ASP.NET Core Web APP (Razor Pages) o React | Encargado de implementar la interfaz web mediante los llamados al API RESTful. | Requerido |

*Tabla 4: Proyectos de la Web:*

Para la interfaz web se deben seguir las siguientes consideraciones:

- Solución aparte al Web API.
- Responsable únicamente de presentar la información y solicitar su modificación al API Rest, no debe transformar la información o implementar lógica de negocio.
- Comunicación con el Rest API por medio del protocolo HTTP.
- Todas las validaciones se deben realizar mediante el uso de anotaciones en los modelos. Deben de realizar para todas las propiedades las siguientes validaciones:
    - Propiedad requerida
    - Longitud mínima y máxima
    - Expresión regular para las propiedades que posean una (tomar de referencia la Tabla 2)
    - Validación de formato de correo y teléfono.
- La repuesta del HTTP request debe ser deserializada en un modelo.
- El diseño visual debe ser responsivo y acorde a lo presentado en el prototipo.
- Manejo de la información en las vistas por medio de Tag Helper.
- Vistas separadas por acción.
- Uso de métodos asíncronos.

# 6 Detalle del puntaje

En la siguiente tabla se muestran la rúbrica de evaluación, para evaluar cada rubro las acciones deben funcionar, de lo contrario se omite la evaluación de los rubros:

| | | | Acciones | | | | |
|---|---|---|---|---|---|---|---|
| **Proyecto** | **Capa** | **Rubro** | **Agregar** | **Editar** | **Eliminar** | **Obtener todos** | **Obtener registro** |
| **API RESTful** | BD | Procedimiento almacenado para la acción del CRUD | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 |
| **API RESTful** | BD | Procedimientos almacenados con respuesta correcta | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 |
| **API RESTful** | Abstracciones | Validación de expresiones regulares | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 |
| **API RESTful** | Abstracciones | Validación de campos requeridos | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 |
| **API RESTful** | Abstracciones | Validación de tamaños | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 |
| **API RESTful** | Abstracciones | Validación de teléfono | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 |
| **API RESTful** | Abstracciones | Validación de correo | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 |
| **API RESTful** | Abstracciones | Correcto uso de la herencia | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 |
| **API RESTful** | DA | Uso correcto de dapper | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 |
| **API RESTful** | DA | Acciones hacia la base de datos por medio de procedimientos almacenados | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 |
| **API RESTful** | DA | Respuesta correcta de los procedimientos almacenados | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 |
| **API RESTful** | Flujo | Correcta implementación del flujo de los controles hacia el DA | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 |
| **API RESTful** | API | Correcta implementación de interfaces | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 |
| **API RESTful** | API | Validación de retorno de la información | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 |
| **API RESTful** | API | Correcto uso de los códigos de respuesta | 2 | 2 | 2 | 0.5 | 0.5 |
| **API RESTful** | API | Correcto uso de las acciones HTTP | 1 | 1 | 1 | 0.5 | 1 |
| **API RESTful** | API | Correcto uso de los parámetros de enlace | 1 | 2 | 1 | 1 | 1 |
| **API RESTful** | API | Correcto uso de los Route | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 |
| **API RESTful** | API | Correcta inyección de interfaces | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 |
| **API RESTful** | API | No uso de implementaciones concretas | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 |
| **API RESTful** | API | Uso de métodos asíncronos | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 |
| **API RESTful** | API | Uso del appsettings para obtener datos de configuración cómo el string de conexión | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 |

| | | | Acciones | | | | |
|---|---|---|---|---|---|---|---|
| **Proyecto** | **Capa** | **Rubro** | **Agregar** | **Editar** | **Eliminar** | **Obtener todos** | **Obtener registro** |
| **Web** | Abstracciones | Validación de expresiones regulares | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 |
| **Web** | Abstracciones | Validación de campos requeridos | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 |
| **Web** | Abstracciones | Validación de tamaños | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 |
| **Web** | Abstracciones | Validación de teléfono | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 |
| **Web** | Abstracciones | Validación de correo | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 |
| **Web** | Abstracciones | Correcto uso de la herencia | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 |
| **Web** | Web Razor | Correcta retroalimentación de las validaciones en los HTML | 1 | 1 | 1 | 1 | 1 |
| **Web** | Web Razor | Uso de TagHelper para presentar los datos | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 |
| **Web** | Web Razor | Deserialización de las respuestas HTML en modelos | 1 | 1 | 1 | 1 | 1 |
| **Web** | Web Razor | Uso de métodos asíncronos | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 |
| **Web** | Web Razor | Uso del appsettings para obtener datos de configuración cómo el string de conexión o endpoints | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 |
| **Web** | Web Razor | Páginas separadas por acción del CRUD | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 |
| **Subtotal** | | | 20.5 | 21.5 | 20.5 | 18.5 | 19 |
| **Total** | | | | | | | **100** |

*Tabla 5: Rubrica de evaluación.*